Efectos de las actividades conectadas y desconectadas en el desarrollo del pensamiento computacional durante la solución de problemas de programación siguiendo el modelo de progresión de tres estados

Carlos Andrés Pérez Oñate

Luz Mila Urrea Rodríguez

Universidad Pedagógica Nacional

Facultad de Ciencia y Tecnología

Maestría en Tecnologías de la Información Aplicadas a la Educación

BOGOTÁ D.C.

2022

Efectos de las actividades conectadas y desconectadas en el desarrollo del pensamiento computacional durante la solución de problemas de programación siguiendo el modelo de progresión de tres estados

Carlos Andrés Pérez Oñate

Luz Mila Urrea Rodríguez

Directora

Dra. Linda Alejandra Leal Urueña

Trabajo de grado para optar al título Magíster en Tecnologías de la Información Aplicadas a la Educación

Universidad Pedagógica Nacional

Facultad de Ciencia y Tecnología

Maestría en Tecnologías de la Información Aplicadas a la Educación

BOGOTÁ D.C.

2022

Derechos de autor

"Para todos los efectos, declaro que el presente trabajo es original y de mi total autoría; en aquellos casos en los cuales he requerido del trabajo de otros autores o investigadores, he dado los respectivos créditos". (Artículo 42, parágrafo 2, del Acuerdo 031 del 4 de diciembre de 2007 del Consejo Superior de la Universidad Pedagógica Nacional).

Agradecimientos

En primer lugar, queremos agradecer a Dios por la oportunidad de afianzar nuestro desarrollo profesional y nuestros conocimientos a través del proceso llevado a cabo durante la maestría.

Segundo, a nuestras familias por su comprensión y todo el apoyo brindado durante el tiempo dedicado a este documento, lo que nos permitió aumentar nuestra resiliencia a pesar de las dificultades presentadas.

Tercero, a los estudiantes de grado décimo de la Institución Educativa SaludCoop Sur que amablemente participaron de este trabajo de investigación.

Finalmente, a nuestra asesora, profesora Linda Leal y demás profesores de la Universidad Pedagógica Nacional que con sus valiosos aportes orientaron nuestro trabajo.

Dedicatoria

A nuestras familias, amigos y demás personas que de alguna forma pusieron su granito de arena para que fuera posible este proyecto.

Tabla de Contenido

Introducción 1

1. Aspectos preliminares 6

1.1. Justificación 6

1.2. Planteamiento del problema 14

1.3 Objetivos 18

2. Antecedentes y marco teórico 19

2.1. Antecedentes 19

2.2. Marco teórico 41

3. Descripción de actividades usadas para el desarrollo del pensamiento computacional 60

3.1 Actividades conectadas 60

3.2 Actividades desconectadas 63

3.3 Actividades de programación usando el modelo usa-modifica-crea 64

3.4 Actividades usa-modifica-crea para la resolución de problemas. 66

4. Metodología 69

4.1. Tipo de investigación. 69

4.2. Diseño de la investigación. 69

4.3. Instrumentos de recolección de datos 72

4.4. Procedimiento 78

4.5. Técnicas de análisis de datos. 81

5. Análisis e interpretación de resultados 83

5.1 Logro previo de aprendizaje en el área de matemáticas 83

5.2 Logro previo de aprendizaje en el área de tecnología e informática 84

5.3 Pre-test de habilidades básicas de programación. 84

5.4. Pre-test de pensamiento computacional 85

5.5 Pre-test de pensamiento computacional por concepto 86

5.6 Pos-test de habilidades básicas de programación. 88

5.8 Pos-test de pensamiento computacional por concepto 90

5.9 Análisis MANCOVA. 94

5.10 Análisis cualitativo del modelo de progresión de tres estados: usa-modifica-crea. 105

5.11 Contraste del test de pensamiento computacional, test de habilidades básicas de programación y modelo de progresión de tres estados. 117

5.12 Aportes de las actividades conectadas y desconectadas en la solución de problemas mediante el modelo de progresión de tres estados: usa-modifica-crea. 122

6. Discusión de resultados 124

7. Conclusiones 133

8. Contribuciones, limitaciones y proyecciones 135

Referencias 137

Anexos

Lista de Figuras

Figura 1. Panorama del pensamiento computacional en educación primaria y secundaria en Latinoamérica----------30

Figura 2. Plan de lecciones usando el modelo usa-modifica-crea ----------37

Figura 3. Clasificación e ítems expuestos en las definiciones del pensamiento computacional analizadas por Lodi (2020) ----------45

Figura 4. Marco conceptual propuesto por Brennan y Resnick (2012) ----------47

Figura 5. Entornos de programación por bloques más populares ----------51

Figura 6. Árbol de taxonomía de actividades desconectadas ----------55

Figura 7. Secuencia de progresión de tres estados propuesta por Lee, et al (2011) ----------58

Figura 8. Grupo de Classroom para las sesiones de actividades conectadas----------61

Figura 9. Temáticas sesión 1 para el grupo conectadas----------62

Figura 10. Ejercicios de Code.org propuestos en la sesión 2 – semana 1---------- 62

Figura 11. Progreso de lecciones en la plataforma Code.org por estudiante ----------63

Figura 12. Classroom usado para las actividades usa-modifica-crea del grupo control----------65

Figura 13. Grupo de Classroom para las unidades bajo el modelo usa-modifica-crea ----------66

Figura 14. Momentos para la sesión 1 de las unidades usa-modifica-crea----------67

Figura 15. Momentos para la sesión 2 de las unidades usa-modifica-crea----------68

Figura 16. Diseño de la investigación----------70

Figura 17. Ejemplo I concepto computacional direcciones básicas ----------74

Figura 18. Ejemplo II concepto computacional direcciones básicas----------75

Figura 19. Ejemplo III concepto computacional bucles ‘repetir veces’ ----------75

Figura 20. Ejemplo de pregunta test de habilidades básicas de programación ----------77

Figura 21. Comparación pre-test y post-test habilidades básicas de programación----------89

Figura 22. Comparación pre-test y post-test pensamiento computacional ----------90

Figura 23. Comparación pre-test y pos-test por concepto computacional ----------102

Lista de Tablas

Tabla 1. Conceptos computacionales identificados por Brenann y Resnick (2012) ----------48
Tabla 2. Prácticas computacionales observadas por Brenann y Resnick (2012) ----------48
Tabla 3. Perspectivas computacionales observadas por Brenann y Resnick (2012) ----------49
Tabla 4. Clasificación de las actividades desconectadas----------55
Tabla 5. Relación de actividades conectadas con los conceptos computacionales----------61
Tabla 6. Relación de actividades desconectadas con los conceptos computacionales----------64
Tabla 7. Datos descriptivos población participante ----------72
Tabla 8. Actividades semanales por cada sesión de la intervención ----------80
Tabla 9. Momentos y tiempo para cada sesión ----------80
Tabla 10. Momentos y tiempo para cada semana de la cuarta fase----------81
Tabla 11. Estadísticos descriptivos del logro previo en el área de matemáticas ----------83
Tabla 12. Estadísticos descriptivos del logro previo en el área de tecnología e informática -----84
Tabla 13. Estadísticos descriptivos del pre-test habilidades básicas de programación ----------85
Tabla 14. Estadísticos descriptivos del pre-test de pensamiento computacional ----------85
Tabla 15. Estadísticos descriptivos pre-test pensamiento computacional por concepto ---------86
Tabla 16. Estadísticos descriptivos del logro posterior en el área de matemáticas ----------88
Tabla 17. Estadísticos descriptivos globales del pos-test de pensamiento computacional -------89
Tabla 18. Estadísticos descriptivos pos-test de pensamiento computacional por concepto ----- 91
Tabla 19. Estadísticos descriptivos prueba MANCOVA ----------94
Tabla 20. Valores de simetría y curtosis de las covariables ----------95
Tabla 21. Lambda de Wilks interacción entre la variable independiente y las covariables ----96
Tabla 22. Prueba de Homocedasticidad de Levene ----------97
Tabla 23. Resúmenes resultados MANCOVA de forma multivariada ----------98
Tabla 24. Resumen MANCOVA de forma univariada ----------99
Tabla 25. Efectos entre factores de la variable independiente ----------101

Tabla 26. Hallazgos unidad 1 ----------106
Tabla 27. Hallazgos unidad 2 ----------109
Tabla 28. Hallazgos unidad 3----------111
Tabla 29. Hallazgos unidad 4 ----------113
Tabla 30. Hallazgos unidad 5 ----------115
Tabla 31. Apropiación de conceptos computacionales según los test y el usa-modifica-crea--118

Introducción

Las tecnologías de la información y la comunicación se encuentran en una posición destacada en diversos campos, reflejo de ello es un mundo cada vez más digitalizado que demanda a los individuos la habilidad y la actitud de pensar computacionalmente para enfrentar los retos del siglo XXI. En este sentido, Adell et al. (2019) afirman que: *"cualquier persona necesitará esta capacidad para vivir, trabajar, aprender, comunicarse o participar como ciudadano o ciudadana de pleno derecho en la sociedad de la información"* (p.175). Sin duda, la formación en estos temas ya no es exclusiva de la educación superior en carreras de ciencias de la computación, de modo que, la enseñanza de los conceptos básicos de programación y el desarrollo del pensamiento computacional son imprescindibles en todos los niveles escolares.

El pensamiento computacional, definido por Wing (2006) como la habilidad para: "*resolver problemas, diseñar sistemas y comprender el comportamiento humano haciendo uso de los conceptos fundamentales de la informática"* (p.33), es un área que se ha retomado con interés en la investigación educativa en los últimos años, ya que contribuye a la formación de profesionales con habilidades para afrontar los cambios tecnológicos, además de contribuir a su propio bienestar y el de la comunidad. (García-Peñalvo y Mendes, 2018). Desde luego, son varios los países que están desarrollando el pensamiento computacional e incorporándolo en el currículo escolar de sus instituciones de educación básica y secundaria, usando como estrategia la programación y creando asignaturas para tal fin (González, 2018; Posso y Murcia, 2022).

En Colombia, el Ministerio de Tecnologías de la Información y las Comunicaciones, expone la importancia de empezar a articular el desarrollo del pensamiento computacional en la formación de niños, niñas y adolescentes, a través de: 1) impulsar competencias como resolución de problemas, creatividad y colaboración indispensables para enfrentar los retos del siglo XXI, 2)

fomentar las habilidades y competencias en disciplinas como robótica, programación, automatización, entre otras, permitiendo mejorar la calidad de la educación en tecnologías a nivel territorial, 3) brindar formación a docentes en los conocimientos, habilidades y estrategias que permite hacer una transferencia eficaz a sus estudiantes, 4) continuar con el desarrollo de aplicaciones y juegos incluyendo dinámicas que fomentan el desarrollo de habilidades del pensamiento computacional (MinTic, 2021).

A pesar de que las programaciones curriculares para el área de tecnología e informática se plantean de acuerdo a los lineamientos sugeridos por el Ministerio de Educación de Colombia y las Instituciones de Educación Superior (IES) que apoyan el proceso de articulación con la educación media, se evidencian dificultades en los estudiantes tanto en la conceptualización como en la práctica en temas relacionados con la programación, resolución de problemas, algoritmos, electrónica, robótica, entre otros. Como resultado se encuentran grandes vacíos no sólo en conceptos básicos, sino también en aspectos relacionados con las matemáticas (operadores lógicos, operaciones aritméticas básicas, variables, funciones, entre otros), incluso, se proyecta un amplio vació de profesionales informáticos en Colombia a corto plazo, según el déficit de más de 112.000 programadores para el año 2025 reportado por el Ministerio de Educación (MinTic, 2021).

Es por ello que esta investigación se enfoca en el tema del pensamiento computacional como parte fundamental para el aprendizaje de la programación y la resolución de problemas en el contexto educativo, principalmente en estudiantes de educación media. Partiendo de la problemática que se presenta en este nivel educativo, donde se evidencia el bajo rendimiento académico y la perdida de interés por la asignaturas relacionadas con las ciencias de la computación (Montes et al., 2020; Lee et al., 2011; Huang & Looi, 2020), se busca evaluar el impacto de actividades apoyadas en recursos digitales conectados y desconectados que

contribuyen a la apropiación de conceptos computacionales y al desarrollo de competencias propias del pensamiento computacional, las cuales sirven como herramienta para que los jóvenes logren apropiarse de la tecnología y que podrían ayudar a incentivar que más estudiantes se interesen por estudiar carreras afines a la computación e informática y de paso contribuir a disminuir el alto déficit en estas áreas. (González, 2018).

Para ello, una serie de actividades conectadas y desconectadas fueron seleccionadas de la plataforma *Code.org*, como herramienta computacional con el fin de apoyar el aprendizaje de la programación, al igual que relacionar y comprender conceptos básicos inmersos en los entornos de programación por bloques, comúnmente usados para resolver problemas. Algunas de ellas fueron adaptadas y modificadas de acuerdo con el nivel de escolaridad y usaron recursos tangibles en el aula de clase (lápiz, hojas, vasos y otros elementos) con el propósito de que los estudiantes se apropiaran de los conceptos computacionales de forma más fácil y práctica.

Por otro lado, y con el fin de determinar la apropiación de estos conceptos computacionales orientados a la programación, se desarrollaron actividades aplicando el modelo de progresión de tres estados: usa-modifica-crea, lo que permitió a los estudiantes, no sólo afianzar sus conocimientos, sino que también realizaron simulaciones en la plataforma *MakeCode* a través de la tarjeta *Micro:bit*, resolviendo problemas de su propio contexto.

En el capítulo uno, denominado aspectos preliminares, se explica el planteamiento del problema y por qué el tema del pensamiento computacional hoy en día ha suscitado tanto interés para los investigadores. Adicionalmente, se indican los motivos que justifican esta investigación desde las necesidades evidenciadas en el contexto académico de la educación; el panorama general del desarrollo del pensamiento computacional en aspectos como: las reformas curriculares a nivel internacional y nacional; las iniciativas y retos para integrar el pensamiento computacional en la

educación secundaria y media; la formación en pensamiento computacional en Colombia y los resultados de las pruebas saber en el área de matemáticas; finalizando con el propósito, la formulación de la pregunta de investigación y los objetivos del presente estudio.

En el segundo capítulo, se abordan principalmente los antecedentes y el marco teórico en torno al pensamiento computacional. Para ello, se realizó una búsqueda en bases de consulta especializadas de las investigaciones y proyectos más relevantes publicados en los últimos cinco años de acuerdo con las variables de estudio y el grupo de edad específico, incluyendo aquellos desarrollados mediante actividades conectadas, desconectadas y el modelo de progresión de tres estados: usa-modifica-crea. Acto seguido, se exponen algunos marcos conceptuales y la falta de consenso en los autores para establecer una definición general del pensamiento computacional, junto con la descripción de las actividades conectadas, desconectadas y el modelo de progresión que promueven el desarrollo del pensamiento computacional.

Para el capítulo tres, se realiza una descripción del proceso de selección de las plataformas y el diseño de los tres tipos de actividades empleadas como andamiaje en este estudio para el desarrollo del pensamiento computacional, a saber: conectadas, desconectadas y usa-modifica-crea.

El capítulo cuatro, explica la metodología empleada en la investigación, el diseño de las sesiones semanales de trabajo, las variables e indicadores presentes en los dos pruebas aplicadas para mediar la apropiación de conceptos computacionales, el desarrollo de habilidades de programación sobre estudiantes de grado 10° de educación media y el procedimiento empleado para la recolección de datos, así como la descripción de las técnicas de análisis cuantitativo de datos MANCOVA y el análisis cualitativo de la transferencia de habilidades a la solución de problemas a partir de las evidencias de aprendizaje de los estudiantes.

El capítulo cinco enuncia la interpretación de resultados partiendo de las pruebas iniciales, los efectos del entrenamiento basado en las actividades propuestas sobre las habilidades de los estudiantes en cada uno de los grupos, para finalmente, develar los hallazgos encontrados.

El capítulo seis inicia la discusión a la luz de los estudios previos y antecedentes sobre actividades conectadas, actividades desconectadas y la progresión usa-modifica-crea y su influencia sobre el desarrollo de las habilidades del pensamiento computacional y las habilidades básicas de programación. Posteriormente, está el capítulo siete que expone las conclusiones de la investigación.

El documento finaliza con el capítulo de contribuciones donde se reconocen las limitaciones del estudio y otros aspectos técnicos en la participación de los estudiantes, así como los aportes y recomendaciones para futuros estudios en el campo del pensamiento computacional y las ciencias de la computación.

1. Aspectos preliminares

1.1. Justificación

Hoy en día, los cambios tecnológicos y la demanda de profesiones emergentes han promovido la formación de personas para adquirir las habilidades y capacidades que les permita afrontar los desafíos sociales del siglo XXI (Zapata-Roz, 2015). Los sucesos mundiales, como, por ejemplo, la emergencia sanitaria COVID-19, requirió que la educación interviniera en los diferentes niveles educativos mejorando la alfabetización digital en toda la población y "*fortaleciendo diferentes habilidades para expresarse, explorar, cuestionar, comunicar y comprender la circulación de ideas en contextos tecnológicos en rápida transformación".* (Carmona et al., 2021, p.4).

En este sentido, es necesario habituarse a los nuevos entornos que involucran habilidades del siglo XXI, entre ellas, las relacionadas con el pensamiento computacional. Es por esto que, para desarrollar estas habilidades, las instituciones planean y proponen cuales deberían ser los conceptos y las habilidades que se deben fomentar en los estudiantes, para lograr un mejor desempeño en el campo laboral, en el sistema educativo y contexto social (Coronel y Lima,2020).

Al mismo tiempo, el desarrollo e implementación de proyectos creativos desde los niveles iniciales de educación en niñas, niños, adolescentes y jóvenes y en la adquisición de herramientas para la solución de problemas se ha constituido en una competencia básica en el proceso de formación y con ello en el desarrollo de habilidades que les permita enfrentar los retos del siglo XXI y suplir los requerimientos que demanda la sociedad en el campo laboral, investigativo y productivo (García, 2022).

En este mismo contexto, Brennan y Resnick (2012) mencionan que: "l*a gente está rodeada de medios interactivos, pero la mayoría de nuestras experiencias con ellos son como*

consumidores. Pasamos el tiempo señalando, haciendo clic, navegando y chateando, actividades que son importantes para aprender a utilizar la tecnología, pero no suficientes para desarrollarnos como pensadores computacionales" (p.10). Por ello, uno de los grandes retos del sistema educativo actual es lograr que las futuras generaciones sean productoras de tecnología, y es allí donde el desarrollo del pensamiento computacional fortalece competencias y habilidades en los estudiantes, para la resolución de problemas en su contexto real haciendo uso de conceptos informáticos, y contribuyendo al propósito de formar profesionales capaces de enfrentar los desafíos tecnológicos de la sociedad actual, el cambio tecnológico y las demandas laborales emergentes (Zapata-Ros, 2015; Weintrop et al., 2015).

En este orden de ideas, el siglo XXI traerá consigo una nueva revolución, donde el desarrollo de habilidades y competencias para enfrentar la era digital será una exigencia para las generaciones futuras, "*El pensamiento computacional incluye una gama de herramientas mentales que reflejan la amplitud del campo de la informática*" (Wing, 2006, pág. 33). Una de estas competencias es el desarrollo del pensamiento computacional, la cual debe estar presente en el ámbito escolar, laboral, social y en general en la vida diaria de las personas, las cuales deben aprender a lidiar y controlar la tecnología, con el fin de dar solución a los problemas del mundo real. (Tellez,2019).

Por otro lado, Zapata-Ros (2019) considera que el pensamiento computacional es una nueva alfabetización, que debe estar constituido por una competencia o una serie de competencias claves como lo son las competencias básicas de alfabetización: la lectura, la escritura, el cálculo elemental y la geometría. (p.2). De igual manera, Zapata-Ros (2020), propone una estrecha relación del pensamiento computacional con la alfabetización digital, en el mismo sentido: *"constituido por competencias claves que sirven para aprender y comprender ideas, procesos y*

fenómenos no sólo en el ámbito de la programación de ordenadores o incluso del mundo de la computación, de Internet o de la nueva sociedad del conocimiento, sino que es sobre todo útil para emprender operaciones cognitivas y elaboración complejas que de otra forma sería más complejo, o imposible, realizar." (p.13).

En el panorama internacional se ha evidenciado que el desarrollo del pensamiento computacional está enfocado hacia una propuesta curricular de competencias transversales en los diferentes grados escolares (Zhang y Nouri, 2019) en virtud de las oportunidades que ofrece para resolver problemas sociales, de educación y aplicación a varios niveles de formación, consolidándose como una de las áreas de investigación de mayor interés (Lodi y Martini, 2021; Grover y Pea, 2013).

Casali et al. (2020), plantean que el pensamiento computacional es benéfico no solo para los jóvenes informáticos sino también para la sociedad y que debe estar presente en todos los niveles educativos, integrando conceptos de programación y de tecnología e implementando estrategias, principalmente en aquellos que inician su proceso de desarrollo en la lógica, con el fin de incentivar y desarrollar destrezas en los estudiantes mediante actividades complementarias como clubes académicos y semilleros de investigación.

Frente a las reformas curriculares a nivel internacional, en España, De las Heras et al. (2018), concluyeron que la enseñanza de la programación, la robótica y el pensamiento computacional tanto en primaria como bachillerato debe ser transversal a otros proyectos y/o asignaturas, por lo que los estudiantes deben cursar una asignatura sobre educación informática obligatoria. Estados Unidos, uno de los países de mayor tradición en la enseñanza de la computación en las escuelas secundarias, no cuenta con currículo unificado a nivel nacional, sin embargo, incorpora cursos de informática y contenidos relacionados con el enfoque pedagógico

STEAM, a través de la *Computer Science Teachers Association* (CSTA), encargada de otorgar a los docentes de primaria y secundaria beneficios importantes que contribuyan a la alfabetización en el campo informático (Sykora, 2021). Por esta razón, desde al año 2016 se creó el proyecto *Computer Science for All* (CS for All) con el fin de capacitar a estudiantes en conceptos computacionales y desarrollo de habilidades del pensamiento computacional.

En Latinoamérica se observa un crecimiento de estudios realizados sobre el pensamiento computacional para educación primaria y secundaria, en diferentes aspectos como: su desarrollo, consolidación y desafíos a nivel regional e internacional, su inmersión en los niveles de educación, la relevancia en la formación docente y el uso de diferentes herramientas para su implementación. (Muñoz et al., 2020; Roig y Moreno, 2020; Villa y Castrillón, 2020). Costa Rica desarrolló una propuesta pedagógica centrada en el aprendizaje de la programación para fortalecer capacidades en niños, jóvenes y docentes, con la incorporación del pensamiento computacional en el currículo a través del aprendizaje basado en proyectos (ABP), así como Ecuador con la estrategia digital 2.0 (Ministerio de Educación de Ecuador, 2017), enmarcada en varios ejes, entre ellos el "aprendizaje digital" que busca la implementación de las ciencias de la computación como asignatura para promover la integración del pensamiento computacional en el currículo nacional. Por su parte, Argentina apoya el desarrollo de la educación digital, programación y robótica para los niveles: inicial, primario, secundario e institutos de formación docente de acuerdo con el objetivo de la Ley de Educación Nacional de garantizar que todas las escuelas del país tengan acceso a las TIC (Coronel y Lima, 2020).

En Colombia, el Ministerio de Educación Nacional y el Ministerio de las Tecnologías de la Información y las Comunicaciones en alianza con otras entidades, han abierto diferentes convocatorias para capacitar a docentes de colegios públicos y privados en programación y

pensamiento computacional, con el propósito de impulsar el desarrollo de proyectos creativos en el aula de clase y de este modo fortalecer habilidades como el pensamiento crítico y pensamiento lateral en niños, niñas y jóvenes.

Desde el año 2019, se ha desarrollado el programa *Coding For Kids*: "programación para niños y niñas", en alianza con el *British Council,* introduciendo el pensamiento computacional en los ambientes de aprendizaje a través de actividades que integran tanto el enfoque desconectado como de resolución de problemas, haciendo uso de herramientas computacionales como la tarjeta *Micro:bit* programable con *MakeCode* y el uso de aplicaciones como *GreenTic*, motivando el desarrollo del pensamiento computacional a través de la programación (MinTIC, 2020).

Por su parte, Velásquez (2021) inició el proyecto para fomentar el desarrollo del pensamiento computacional en la primera infancia, llamado "jugando y kreando" mediante un diplomado de 90 horas para docentes del sector oficial enfocado a desarrollar el pensamiento computacional a través de herramientas no digitales y metodologías basadas en el juego, lúdica y la didáctica (Ministerio de Tecnologías de la Información y Comunicaciones, MinTIC, 2021).

De manera similar, existe el programa de formación docente en pensamiento computacional para la primera infancia, que se realiza a través de la Red Interamericana de Educación Docente (RIED) en alianza con la Academia Colombiana de Ciencias STEM, la Universidad del Norte y un grupo de expertos provenientes de Colombia, Chile, Costa Rica y Perú. Incluso, el Ministerio de Educación Nacional junto con ASCOFADE (Asociación Colombiana de Facultades de Educación) participan en la renovación y actualización de la guía 30, la cual fue publicada desde el 2008 e incluye los lineamientos y, orientaciones generales para la educación, actuando como único referente de Colombia para la formación en el área de tecnología.

Por lo anterior, algunos expertos coinciden que la formación de los docentes en pensamiento computacional es una prioridad para lograr una integración efectiva en Latinoamérica, no solo enfocados en su concepción, su campo de acción y el uso consciente de las tecnologías, sino también el considerarla frente a su evaluación en los sistemas escolares y cómo realizar cambios curriculares reconociendo aquellos aspectos que permita generar impacto en sus instituciones (Carmona-Mesa et al., 2021).

En el campo laboral, el déficit de programadores en el mercado colombiano es alto, para el año 2021 ya reportaba más de 70 mil. Los estudios realizados por la consultora McKinsey reportan que para el año 2025 se presentará un déficit de 112.000 programadores. (MinTIC, 2021). Por lo anterior, la iniciativa por parte del Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia a través del programa "Misión TIC 2022", consiste en capacitar a jóvenes y ciudadanos en programación y formar miles de programadores que les permita abrirse paso en el mercado laboral.

Esta iniciativa es apoyada por la Federación Colombiana de la Industria del Software y Tecnologías Informáticas Relacionadas (FEDESOFT), ofrece capacitación en programación y desarrollo de software, enfocada a la formación de tecnologías y competencias transversales, que brinde a los jóvenes mejores oportunidades laborales. Así mismo, en alianza con entidades como Redis, *Easy Think* y Recursos Educativos y Tecnológicos del Ministerio de Educación Nacional y el Ministerio de las TIC, son los principales promotores del pensamiento computacional a través de la participación en la competencia internacional *Bebras*, que fomenta la resolución de problemas, así como el uso de conceptos informáticos y demás habilidades propias del pensamiento computacional a niños y jóvenes entre 5 y 19 años de colegios públicos y privados. De acuerdo con los resultados de la prueba piloto realizada en el año 2021, donde se inscribieron

49 colegios a nivel nacional, la participación del sector público (63.4%) fue mayor frente a la del sector privado (36.6%) (FEDESOFT, 2022).

En este orden de ideas, para satisfacer las necesidades del mercado en general, se requieren de personas cualificadas en el dominio y destreza de cualquier lenguaje de programación que puedan dar solución a los problemas y retos que la sociedad les demande, lo cual es posible gracias a diferentes profesiones que han surgido entorno a las tecnologías y al desarrollo de la programación, entre ellas los programadores de software, desarrolladores de aplicaciones y los científicos de datos: "*personas especializadas en matemáticas y estadísticas que dominan la programación y sus diferentes lenguajes, ciencias de la computación y analítica aprovechando los grandes volúmenes de información como el BigData".*(Romero et al., 2019, p.3).

En la transferencia de habilidades cognitivas (por ejemplo, la capacidad de resolver problemas) es clave en las reivindicaciones a favor de introducir el pensamiento computacional en los planes de estudios obligatorios (Bocconi et al., 2016). Si bien algunos estudios indican una fuerte correlación entre el desempeño en matemáticas, la informática y la resolución de problemas por parte de los alumnos (Baldwin et al., 2013; Mohaghegh & McCauley, 2016; Bocconi et al., 2016; Tabesh, 2018; Terroba et al. 2021), la afirmación de que la cultura informática podría moldear el pensamiento de los niños, convirtiéndose en un objeto con el que pensar, ha estimulado una corriente de investigación que busca pruebas de la transferencia de habilidades cognitivas de las actividades de programación a otras áreas, entre ellas, las matemáticas (Buitrago-Flórez, 2018).

Stephens y Kadijevich (2020) indican una fuerte relación entre el pensamiento computacional y las matemáticas, encontrando que, en las diferentes definiciones del pensamiento computacional presentes en la literatura, existen componentes en común, entre ellos: la lógica y el pensamiento lógico, los algoritmos y el pensamiento algorítmico, la abstracción y generalización

y la descomposición del problema (Grover y Pea, 2017). Por ejemplo, los algoritmos, pueden definirse como soluciones a un problema matemático expresadas en una secuencia de instrucciones claramente definidas que procesan datos numéricos y recurren a diversos símbolos, es por ello que para implementarlos con éxito, se requieren distintas habilidades cognitivas, incluyendo la descomposición (descomponer un problema en subproblemas) y la abstracción (hacer afirmaciones generales resumiendo ejemplos particulares sobre conceptos, procedimientos, relaciones y modelos subyacentes) Stephens y Kadijevich (2020).

Aunque en el contexto actual varios ejemplos ilustran y avalan la relación bidireccional entre el pensamiento algorítmico y las matemáticas, aún no existe una relación clara en cómo las matemáticas contribuyen a la alfabetización digital, el pensamiento computacional y el pensamiento algorítmico, los estudios que relacionan el aprendizaje de las matemáticas con áreas tradicionalmente relacionadas con la programación son escasos.

En Colombia no hay pruebas estandarizadas específicas para medir el desempeño en el pensamiento computacional. Sin embargo, hay una clara asociación entre el pensamiento computacional y el ámbito de la resolución de problemas en las pruebas de matemáticas. El Ministerio de Educación Nacional es la entidad que a través de las pruebas Saber 11, evalúa las competencias básicas que debe alcanzar un estudiante de educación básica y media en el área de matemáticas: razonamiento y argumentación, comunicación y representación, modelación, planteamiento y resolución de problemas (ICFES, 2019). Los resultados correspondientes al año 2020, muestran que: el 4% de los estudiantes tuvo un desempeño insuficiente, en el nivel mínimo el 15%, para el nivel satisfactorio se obtuvo un 66% y en el nivel avanzado el 14%, respecto a la entidad territorial certificada. (Ministerio de Educación de Colombia).

1.2. Planteamiento del problema

Recientemente, los esfuerzos para apoyar el desarrollo del pensamiento computacional e involucrar a los estudiantes en entornos computacionales, tecnologías digitales y programación se han centrado tanto en enfoques informáticos digitales como en recursos no digitales aplicados a la resolución de problemas (Caeli & Yadav, 2019).

En este contexto resulta esencial determinar la pertinencia de dichas actividades en el aprendizaje de las ciencias de la computación durante la educación secundaria y media, donde es más frecuente la baja participación, falta de interés y desmotivación de los estudiantes hacia las ciencias de la computación (Montes et al., 2020; Lee et al., 2011; Huang & Looi, 2020). Un método frecuentemente referenciado en la investigación en este campo consiste en llevar a cabo actividades previas a la enseñanza de la programación en las que se promueva el desarrollo de las habilidades de pensamiento computacional de manera articulada con el currículo abordando estrategias de enseñanza conectadas y desconectadas (Caeli & Yadav, 2019), con la misma relevancia de asignaturas como las matemáticas y el lenguaje (Montes et al., 2020; Proyecto KDE y la comunidad, 2018; Ortega, 2020).

Algunos estudios reportan como actividades conectadas el uso de ambientes de programación basados en bloques para introducir conceptos y habilidades del pensamiento computacional sin involucran propiamente la sintaxis de un lenguaje de programación (Bocconi et al., 2016). Sin embargo, existe una tendencia hacia el uso de métodos desconectados, una aproximación popular adoptada en muchos países, que permite involucrar conceptos de computación mediante la integración de diversas actividades con el propósito de resolver problemas sin el uso de tecnología digital (Bocconi et al., 2016).

Algunos ejemplos incluyen: los talleres de pensamiento computacional para todos, de la Asociación de Profesores de Ciencias de la Computación (CSTA) y la Sociedad Internacional de Tecnología en la Educación (ISTE) del proyecto Líderes de pensamiento computacional (Lee et al., 2011), y el proyecto denominado *Computer Science Unplugged (CS Unplugged*) de la universidad de Canterbury, en Nueva Zelanda. Asimismo, se han estudiado modelos mixtos, que combinan recursos conectados y desconectados, como el programa *Coding for Kids* del *British Council* y el Ministerio de Tecnologías de la Información y Comunicaciones de Colombia (MinTIC, 2020) para el desarrollo de habilidades del siglo XXI en torno al pensamiento crítico, la creatividad y la resolución de problemas, el cual recurre al uso de fichas metodológicas basadas en un modelo de progresión de tres estados: usa-modifica-crea, con el fin de ayudar al estudiante en la transición progresiva de familiarizarse con las herramientas y representaciones computacionales que va a usar, modificarlas, comprender cómo se construyeron y cómo se acoplan a otros problemas, hasta lograr la creación de nuevos productos que brinden soluciones propias desde su contexto real (Vieira et al., 2019).

Por su parte, Peel et al. (2022), proponen utilizar actividades desconectadas como una forma de alfabetizar a los estudiantes e integrar el pensamiento computacional en los cursos iniciales de ciencias, sin embargo, no se muestran cambios significativos con respecto a la educación científica y su implementación aún tiene dificultades. De allí, que los esfuerzos de algunos países destacados en educación en tecnología como Estados Unidos, España, Francia, Finlandia, entre otros, se ha orientado a incluir el pensamiento computacional en sus currículos (Motoa, 2019).

Por otra parte, como estrategia para el aprendizaje del pensamiento computacional se recomienda el modelo usa-modifica-crea. Este modelo sugiere que para evitar sobrecargas

cognitivas en los estudiantes, se desarrolle por etapas, en primer lugar, es pertinente que los estudiantes usen herramientas y representaciones computacionales para que se familiaricen con ellas y establezcan su valor; en segunda instancia, los estudiantes modifican estas representaciones para comprender como fueron construidas y como pueden aplicarlas a otros problemas o contextos; por último, una vez que los estudiantes adquieren el conocimiento suficiente para hacerlo, es factible que creen artefactos y soluciones desde cero (Vieira, 2019).

Autores como Lee et al. (2011) y Martin et al. (2020), coinciden en que es una estrategia cuyos resultados son a largo plazo, mejorando en los estudiantes las habilidades adquiridas en las primeras etapas y finalmente realizar nuevos proyectos con sus propios diseños; también se fortalecen aspectos como el compromiso, el tiempo y la adaptación creativa, donde los estudiantes hacen la transición de usuarios a creadores de artefactos computacionales. Este modelo es considerado como parte de la metodología de la última fase para esta investigación, después de realizar la intervención con actividades conectadas y desconectadas con los dos grupos experimentales.

En este escenario, esta investigación busca contrastar el efecto que puedan tener las actividades conectadas y desconectadas en el desarrollo del pensamiento computacional y en apropiación de conceptos computacionales para la solución de problemas de programación usando el modelo usa-modifica-crea. Específicamente se hará uso de las actividades conectadas y desconectadas que ofrece la plataforma *Code.org* y la interfaz gráfica para programación con bloques, herramienta de apoyo para la resolución de problemas que ofrece la plataforma *MakeCode,* para desarrollar este modelo.

En este sentido, nuestra investigación pretende hacer aportes al estudio de habilidades del pensamiento computacional en estudiantes de educación media, ya que la literatura muestra

resultados y establece comparaciones en los niveles primaria y algunos grados de bachillerato en el uso de actividades conectadas, desconectadas y la combinación de las dos (Mantilla y Negre, 2021, Duarte, et al., 2019, Rodríguez et al., 2020).

A la luz de lo anterior, la relevancia de hacer esta investigación radica en abordar la incorporación del entrenamiento mediante actividades conectadas y desconectadas previo al trabajo con el modelo de progresión de tres estados: usa-modifica-crea enfocado a la resolución de problemas para entender cómo se desarrolla el pensamiento computacional en este nivel educativo.

La pregunta de investigación que servirá de guía para el desarrollo de la tesis es: ¿Existen diferencias significativas en la apropiación de los conceptos fundamentales de programación y en el desarrollo de habilidades de pensamiento computacional durante la solución de problemas de programación en el entorno *MakeCode*, siguiendo el modelo de progresión de tres estados: usa-modifica-crea, cuando se entrenan previamente habilidades de pensamiento computacional a través de actividades desconectadas y conectadas?

Adicionalmente, se seleccionó como marco de referencia el propuesto por Brennan & Resnick (2012), el cual se basa en tres dimensiones: conceptos, prácticas y perspectivas computacionales, dando mayor relevancia a los conceptos claves que definen el pensamiento computacional y que son parte fundamental en las actividades propuestas para que los estudiantes realicen y logren desarrollar habilidades de pensamiento computacional.

1.3 Objetivos

1.3.1 Objetivo General

Comparar el efecto diferencial de las actividades conectadas y desconectadas en la apropiación de conceptos computacionales y en el desarrollo de habilidades de pensamiento computacional cuando se integran como actividades previas al trabajo en el entorno de aprendizaje *MakeCode,* siguiendo el modelo usa-modifica-crea.

1.3.2. Objetivos Específicos

- Evaluar la incidencia del trabajo con actividades conectadas y desconectadas en desarrollo de habilidades de pensamiento computacional.
- Evaluar la incidencia del trabajo previo con actividades conectadas y desconectadas en la apropiación de conceptos computacionales durante la solución de problemas en *MakeCode* con *Micro:bit.*
- Contrastar los aportes de las actividades conectadas y desconectadas a la solución de problemas de programación en *MakeCode* cuando se sigue el modelo usa-modifica-crea.

2. Antecedentes y marco teórico

2.1. Antecedentes

Este capítulo ilustra los antecedentes más recientes entorno a las reformas educativas, avances y proyectos más relevantes del pensamiento computacional, las actividades conectadas y desconectadas y el modelo usa-modifica-crea. Las consultas fueron realizadas en bases de datos especializadas como *Science.gov*, *Scopus*, *ERIC*, *EBSCO host*, *SciELO*, *IEEE Xplore Digital Library*, *Computers & Applied Sciences Complete*, y *Google Scholar*, usando como descriptores diferentes combinaciones de palabras clave como *"computational thinking", "plugged activities", "unplugged activities"* y *"use-modify-create"* o su correspondiente traducción al idioma español. Se usaron como criterios de selección: 1) publicaciones de acceso libre; 2) escrito en idioma inglés o español; 3) antigüedad no mayor a cinco años; 4) estudios que incluyeron como participantes estudiantes con edad entre 6 a 18 años; y 5) estudios orientados a grados de educación primaria, secundaria y media o su equivalente al sistema educativo colombiano.

2.1.1 La educación en pensamiento computacional

Las reformas educativas en Estados Unidos y algunos países de la Unión Europea han introducido reorganizaciones curriculares que incluyen la programación con el objetivo de mejorar las habilidades digitales de los estudiantes y desarrollar el pensamiento computacional (Bocconi et al. 2016) e incorporando la programación basada en bloques en los materiales educativos junto con la programación convencional basada en textos (Weintrop et al., 2018). Tal es el caso de la Asociación de Educadores en Ciencias de la Computación y la Sociedad internacional para la Educación en Tecnología quienes promueven el pensamiento computacional con el objetivo de mejorar las habilidades en la solución de problemas y el pensamiento crítico aprovechando todo el potencial que ofrece la computación (Sykora, 2021) así como Grover et al. (2019) quienes

diseñaron el plan de estudios denominado 'VELA' para ayudar a estudiantes de secundaria a explorar y utilizar conceptos como: variables, expresiones, bucles, abstracción y condicionales, en contextos no conectados antes de abordar herramientas de programación. Otros esfuerzos se han enfocado en aprovechar el potencial del pensamiento computacional de manera transversal en disciplinas diferentes a las ciencias de la computación y encontrar formas de facilitar el aprendizaje (Quiroz-Vallejo et al., 2021) en especial las clases de matemáticas y ciencias (Weintrop et al. 2015) o través de disciplinas como las artes y la literatura, idiomas, sociedad, cultura y deportes (Díaz y Molina, 2020).

En cuanto a los niveles de educación, en la educación primaria los principales trabajos se enfocan desde el área de lenguaje en la comunicación y representación; en la educación secundaria se concentran en el aprendizaje basado en proyectos (ABP), la robótica y la programación en lenguajes con bloques; y en el bachillerato se enfoca en lenguajes de programación (Mantilla y Negre, 2021; Coronel y Lima, 2020). Sin embargo, aún persisten diferencias entre las conceptualizaciones y las estrategias usadas para la integración del pensamiento computacional a las ciencias de la computación en el ámbito educativo como un recurso metodológico (Quiroz-Vallejo et al., 2021).

Frente a la resolución de problemas y el pensamiento computacional, Ortega (2017) indagó cómo se mejora la resolución de problemas complejos a partir de su representación y proceso, con una muestra de 38 estudiantes de grado 4º de educación secundaria. La validación de datos a partir de las pruebas PISA 2012, permitió determinar que el grupo experimental (N=21) que realizó actividades de programación y pensamiento computacional, obtuvo mejores resultados al comparar el pre-test (M=6,95) frente al post-test (M=9,90), en comparación con el grupo control (17 participantes) pre-test (M=7,00) y pos-test (M=7,23), demostrando que el desarrollo del

pensamiento computacional facilita y mejora la resolución de problemas complejos y aporta un mayor valor a la aplicación del conocimiento.

En el ámbito nacional, algunas investigaciones se han enfocado en: 1) la creación de clubes con niños entre 5 y 17 años enfocados al aprendizaje mediante recursos en línea como *Scratch* (García, 2022) y *Code.org*, incentivando el fortalecimiento de competencias del pensamiento computacional y la adquisición de conocimientos en temas de electricidad, electrónica, mecánica y programación (García et al., 2021); 2) el uso de diferentes actividades desconectadas y conectadas (Mantilla y Negre, 2021); y 3) el uso de entornos de programación para la creación y muestra de textos literarios (Sarmiento, 2018). De acuerdo con los resultados, estas metodologías apuntan a que los estudiantes reconozcan y hagan uso de estrategias para solucionar problemas académicos y de su contexto, así como despertar el interés de docentes de diferentes áreas para involucrarse y desarrollar experiencias interdisciplinarias incentivando el fortalecimiento del pensamiento computacional.

Un resultado relevante en el uso de recursos en línea es la experiencia "Enseñanza de la programación a través de *Scratch*" desarrollada por García (2022) con 19 estudiantes del grado 7° distribuidos en dos grupos: grupo experimental cuyo proceso de formación se desarrolló y fortaleció a través de la herramienta tecnológica de *S*cratch; y grupo control, el cual desarrolló la temática de una manera tradicional sin recursos tecnológicos. De acuerdo con los resultados obtenidos en un test de 25 preguntas cerradas con opción múltiple basadas en pruebas *Bebras* y el test de pensamiento computacional de Román-González et al. (2015), se evidenció una mejora en el pos-test del grupo experimental (M=13.63, SD=2.96) en comparación con el grupo control (M=9.0, SD=1.45) frente a los resultados del pre-test para el grupo experimental (M=7.63, SD=2.40) y el grupo control (M=7.57, SD=1.67) luego de incorporar conceptos básicos de

programación en el currículo del área de tecnología e informática. Más aún, el grupo experimental, demostró que el implementar metodologías que involucran herramientas tecnológicas en el proceso de enseñanza-aprendizaje, posibilita el desarrollo e incremento de las habilidades del pensamiento computacional (García, 2022).

2.1.2 Desarrollo del pensamiento computacional en la educación media

Los estudios relacionados en educación media de autores como Mantilla y Negre (2021), Rúales (2022), Sinisterra (2018), Buitrago-Flórez et al. (2021) y Rojas (2021) han mostrado resultados positivos cuando se recurre al pensamiento computacional como una estrategia efectiva para potenciar las habilidades y competencias en la enseñanza y el desarrollo pedagógico en tres ejes centrales: programación, resolución de problemas, TIC y gamificación. No obstante, otros autores como Guggemos (2021) han mostrado un bajo impacto en las habilidades del pensamiento computacional en estudiantes de educación media debido a aspectos motivacionales.

Mantilla y Negre (2021) realizaron un diagnóstico del nivel de formación de competencias del pensamiento computacional para resolver problemas basado en el test de pensamiento computacional (Román-González et al.,2015) con 133 estudiantes de grado 11°. De acuerdo con los resultados, la correlación de Pearson entre la dimensión conceptual, práctica y perspectivas del pensamiento computacional permitió concluir que la mayoría de los estudiantes lograron un dominio medio-alto en la dimensión de apropiación de conceptos computacionales ($r = 0.463$, $\rho < 0.001$), dominio medio en prácticas ($r = 0.462$, $\rho < 0.001$) y dominio medio-alto en perspectivas del pensamiento computacional ($r = 0.602$, $\rho < 0.001$), mejorando así sus habilidades para la resolución de problemas (Mantilla y Negre, 2021).

De igual forma, la experiencia educativa para estudiantes de 10° y 11°denominada "Inteligencia en un bolsillo: el pensamiento computacional" orientada a fortalecer la programación y la resolución de problemas, involucrando como estrategia el uso de actividades desconectadas y conectadas a partir de la manipulación de la tarjeta de programación *Micro:bit* y el software *MakeCode*, contribuyó a enriquecer los conocimientos previos de los estudiantes participantes sobre prácticas de resolución de problemas a partir del pensamiento computacional (Rúales, 2022).

En cuanto al aprendizaje por proyectos, Sinisterra (2018) expone un estudio con 40 estudiantes de grados 10° y 11°, con el proyecto titulado "enREDAdos" (Creación de Materiales para Recursos Educativos Digitales Abiertos), el cual favoreció la apropiación de habilidades asociadas al pensamiento lógico – algorítmico con una diferencia de 30.7 % al inicio del proyecto a 87.1% al final de la prueba piloto. Por su parte Rojas (2021), realiza una investigación centrada en evaluar la implementación del programa *Coding for Kids* del *British Council*, para el desarrollo del pensamiento computacional a través del aprendizaje de la programación, en 75 estudiantes del grado 10° (M=16.05, SD=4.54) con los que se aplicó el test de pensamiento computacional (Román-González et al., 2015). Los resultados muestran una relación significativa entre el resultado de la prueba y la asistencia al curso CFK al aplicar el test de Yuen (t=2.4897, df=41.41, t-value=0.01689) y en la correlación de las variables del estudio (r=0.253, t=2.23, ρ=0.030), comprobando una correlación positiva entre asistir al programa CFK y los resultados obtenidos en la prueba, soportando que las habilidades cognitivas desarrolladas por la experiencia están directamente relacionadas con la ejecución de los procesos del pensamiento computacional.

De igual forma, Buitrago-Flórez et al. (2021) aplicaron un método mixto integrado para evaluar las mejoras en el desarrollo de las competencias en cinco habilidades relacionadas con el pensamiento computacional en una muestra de 42 estudiantes de grado 11° entre 16 y 18 años

mediante actividades con enfoque en aprendizaje basado en problemas y recursos didácticos como *LEGO*. La media de respuestas correctas en el pre-test (M=4,19, SD= 1,49) en comparación con el post-test (M=14,5, SD=3,0) junto con la prueba-T mostraron una diferencia significativa (ρ = 1,7 E-25) confirmando los beneficios de incluir estrategias pedagógicas centradas en el alumno para la mejora de las habilidades de pensamiento computacional.

En contraste con los anteriores estudios, Guggemos (2021) observo un crecimiento negativo no significativo en las habilidades de pensamiento computacional debido a una disminución en la motivación para diferentes periodos del año escolar, t1 (M = 3,36, SD = 1,66) t2 (M = 4,18, SD = 1,47) y t3 (M = 4,75, SD = 1,42) al validar y utilizar el test de pensamiento computacional (Román-González et al., 2018) en una muestra de 202 estudiantes del 11º grado que recibieron 90 minutos de instrucción en ciencias de la computación. Aunque se esperaba una asociación positiva entre las habilidades matemáticas y el pensamiento computacional, gracias a una varianza explicada del 70,4% y un aumento del 61,2%, sorprendentemente no ocurrió. Otro punto que podría contribuir a explicar el hallazgo es el auto reporte de los estudiantes sobre la poca experiencia de oportunidades de aprendizaje durante el año escolar (M = 2,63, SD = 0,93 en una escala de siete puntos) gracias a que la informática se introdujo recientemente como asignatura curricular. Sin embargo, encontró como las habilidades matemáticas pueden predecir el nivel del pensamiento computacional ya que guardan una relación más estrecha con éste que con las habilidades lingüísticas.

2.1.3 Desarrollo del pensamiento computacional mediante actividades conectadas, desconectadas y modelos mixtos.

2.1.3.1 Actividades conectadas

El uso de entornos de programación basados en bloques es una forma cada vez más popular para que los jóvenes se introduzcan en el campo de la informática en general, proporcionando así caminos atractivos y accesibles en el mundo de la programación (Weintrop, 2021). Por ello, las apuestas de incorporar la programación basada en bloques en los materiales educativos mediante el uso de plataformas como *Code.org* y *Scratch* han sido evaluados por diferentes investigadores desde la pertinencia de su uso en diferentes niveles de educación, los componentes del pensamiento computacional vinculados con la resolución de problemas, así como los métodos de enseñanza utilizados para abordar las habilidades del pensamiento computacional (Kale & Yuan, 2020).

En un análisis de más de 5.000 estudiantes en Estados Unidos a través de un examen que evalúa el rendimiento de los estudiantes en contenidos de ciencias de la computación, Weintrop et al. (2018) y la asociación *Code.org* descubrieron que los estudiantes obtienen puntuaciones significativamente más altas en la formulación de preguntas basada en bloques que en las preguntas basadas en texto. Adicionalmente, Weintrop (2021) encontró que, la transición a un lenguaje de programación profesional basado en texto como Java es más fácil si, previamente, se aprende a programar en una herramienta basada en bloques en comparación con un lenguaje basado en texto. Este hallazgo es una prueba importante que demuestra que la programación basada en bloques es una forma eficaz de introducir a los novatos en la programación y proporciona evidencia adicional para la importancia de incluir la programación basada en bloques en la educación formal (Weintrop, 2021). Otros estudios que respaldan el uso de plataformas como *Code.org* y *Scratch*, basadas en programación por bloques, muestran resultados contradictorios en educación primaria y secundaria tanto en la apropiación de conceptos básicos de programación para la resolución de problemas (Barradas et al., 2020, Kalelioğlu, 2015) como en las actitudes hacia la programación y

la percepción de la dificultad de estas plataformas, las cuales puede influir significativamente en los resultados de los estudiantes de edades inferiores (Lambić, Đorić & Ivakić, 2020).

Barradas et al. (2020) realizó un estudio con 130 alumnos de 4º grado con edades entre 9 y 10 años con el objetivo de enseñar conceptos básicos de programación mediante 16 lecciones de Fundamentos de la Informática de la plataforma *Code.org* y desarrollar el pensamiento computacional. En los resultados de los ejercicios desarrollados, encontraron que el 85,8% de problemas fueron resueltos y sólo el 3,7% de ellos no obtuvieron la solución óptima. En cuanto a los conceptos computacionales abordados, se registraron los siguientes porcentajes de aprobación: secuencialidad 89.0%; bucles 86.5%; eventos 69.7%; paralelismo 69.7%; condicionales 72.8%; operadores 71.6%; datos 69.7. También pudieron observar que en el uso de *Code.org* los alumnos se sintieron más implicados en los ejercicios gracias al reconocimiento de algunos de los personajes utilizados en los retos y el hecho de que estaban aprendiendo conceptos de informática mientras se divertían (Barradas et al., 2020).

De igual forma, Lambić, Đorić & Ivakić (2020) investigaron el efecto del uso de *Code.org* en 293 estudiantes de primaria de 7 a 10 años, divididos en dos grupos, el primero de 7 a 8 años y el segundo de 9 a 10. Allí los alumnos de mayor edad expresaron una actitud significativamente más positiva hacia la programación que los más jóvenes, encontrando una diferencia estadísticamente significativa entre el número de tareas resueltas, mientras que los estudiantes de mayor edad realizaron con éxito un número significativamente mayor de tareas (M=31,40) frente a los estudiantes de menor edad (M= 25,72), que aún no son capaces de resolver un número significativo de tareas de programación, lo que podría haber influido en esta diferencia. Adicional a esto, utilizado una escala Likert de 1 a 5 para evaluar las actitudes de los estudiantes, el grupo de estudiantes de mayor edad evaluó más positivamente su deseo de aprender sobre programación

(M=4.16) que los de menor edad (M= 3.97), sin embargo, los grupos de estudiantes de mayor edad como los de menor edad expresaron la opinión de que la programación es difícil (M= 3.81 y M= 3.45 respectivamente). Finalmente, debido a la complejidad del contenido, el grupo de estudiantes más jóvenes, que son menos hábiles en la resolución de problemas y no poseen los suficientes conocimientos necesarios, mostró más ansiedad al resolver las tareas (Lambić, Đorić & Ivakić, 2020).

Este hecho también se respalda al comparar *Code.org* frente a otros entornos de programación basados en bloques como *Scratch*. Krugel & Ruf (2020) realizaron un estudio con 122 estudiantes de grado 7° entre 12 y 14 años para determinar los efectos de ambos entornos de aprendizaje midiendo la autorregulación percibida por los alumnos y la motivación intrínseca. El análisis cualitativo de este estudio encontró que los alumnos que interactuaron con *Code.org* mostraron una mayor motivación intrínseca en comparación con el grupo de *Scratch*, revelando aspectos positivos de ambos entornos de aprendizaje como la diversión y los retos, en el caso de *Code.org*, mientras que, en *Scratch*, la sensación de programar en un entorno más libre y con mayores posibilidades. El estudio también mostró aspectos negativos, como la dificultad para programar, especialmente al principio de la intervención debido a algunos problemas técnicos y al hecho de que *Scratch* ofrece un rico conjunto de funcionalidades que podría ser abrumador para algunos principiantes; lo cual también es una explicación relacionada con la baja motivación intrínseca hacia *Scratch*. Basándose en estos resultados, se puede inferir que el uso de plataformas de programación basadas en bloques como *Code.org* o *Scratch* por parte de los estudiantes más jóvenes puede afectar negativamente su actitud hacia la programación, y a pesar de que algunas actividades podrían ser utilizadas por ellos, deben ser cuidadosamente seleccionadas por los

profesores de acuerdo con su nivel de conocimientos previos en programación (Lambić, Đorić & Ivakić, 2020).

Otra tendencia actual relacionada con los ambientes de programación basados en bloques es el uso de dispositivos programables como *Micro:bit*, el cual puede complementar el trabajo orientado a la programación para la resolución de problemas. Básicamente, *Micro:bit* es un dispositivo informático codificable de bolsillo, que dispone de sensores, botones y LED integrados, diseñado para que los niños se involucren y sean creativos con la tecnología, el cual se puede programar a través de *MakeCode*, un sitio web dedicado que permite a los usuarios elegir entre una programación basada en bloques o serie de editores de código en línea disponibles en la mayoría de los navegadores web modernos (Ball et al., 2016). En esa línea de trabajo, Tyrén et al. (2018) y Cederqvist (2020) exponen la creación de talleres para estudiantes de 4º a 6º y 8° grado en con el objetivo de entrenar las habilidades de pensamiento computacional en la resolución de problemas utilizando como plataforma educativa el dispositivo *Micro:bit*. Los resultados señalan la utilidad del conjunto de las herramientas básicas disponibles en la *Micro:bit* para introducir los conceptos de programación antes de trabajar con ejercicios de programación y resaltan la práctica de estos conceptos fundamentales en paralelo, en lugar de trabajar con cada uno de ellos por separado. Por su parte, Bocanegra (2020) realizó una investigación con 77 estudiantes de grado 9° enfocada en el desarrollo de habilidades del pensamiento computacional con el manejo del editor *MakeCode* y programación de las tarjetas *Micro:bit* empleando recursos TIC y metodología STEAM. Mediante una encuesta para identificar la disponibilidad de recursos computacionales en la casa y una evaluación diagnóstica pre-test y pos-test sobre conceptos de programación, evidenció el desarrollo de las capacidades de los estudiantes para resolver problemas mediante conceptos en programación, gracias al conocimiento teórico y talleres empleados para obtener un

conocimiento práctico y apropiar el pensamiento computacional (Bocanegra, 2020). Es así como estos hallazgos apuntan a que la combinación de la programación basada en bloques y los dispositivos de computación podrían convertirse en un enfoque prometedor para promover las habilidades de pensamiento computacional tanto en grados escolares inferiores (Kastner-Hauler et al, 2022) como superiores (Vostinar & Kneznik, 2020) generando una menor barrera de entrada para empezar a introducir el pensamiento computacional en las aulas.

2.1.3.2 Actividades desconectadas

El uso de actividades desconectadas como estrategia para enseñar a programar y ayudar a los estudiantes a entender el diseño de un programa informático o lenguaje de programación utilizando una explicación o recorrido desconectado antes de recurrir al computador, ha sido el interés de varios autores en la última década (Saxena, et al. 2019, Silva et al., 2020, Correa, 2020, Looi, et al., 2018).

Quiroz-Vallejo et al. (2021) afirman que el uso de dispositivos electrónicos es una de las estrategias con mejores resultados e interés en las clases de tecnología e informática para fomentar el pensamiento computacional, sin embargo, reconocen que es necesario tener en cuenta que existen áreas sin acceso a internet o tecnologías digitales, por lo que es preciso ampliar y enfocarse en actividades desconectadas para lograr su desarrollo. En la Figura 1 se muestra algunas de las estrategias y desafíos que los autores encontraron para la integración del pensamiento computacional en las aulas escolares (Carmona-Mesa et al., 2021).

Figura 1.

Panorama del pensamiento computacional en educación primaria y secundaria en Latinoamérica

Nota: Tomado de *integración del pensamiento computacional en educación primaria y secundaria: documento de posición* (p.15) por Carmona-Mesa et al. (2021).

En niveles de educación inferior se ha descubierto que "*tener experiencias más concretas en actividades no conectadas puede ayudar a los estudiantes de preescolar a tener una mejor base para cultivar el pensamiento computacional y aplicar las habilidades desarrolladas en una actividad conectada*" (Saxena et al 2019, p.57), mientras que en educación primaria Silva et al., (2020) y Correa (2020) demostraron que el uso de actividades desconectadas genera un impacto positivo en la motivación y el desarrollo de la creatividad, por cuanto se ponen a prueba sus habilidades en la construcción de artefactos y la resolución de problemas matemáticos, confirmando que las actividades desconectadas a temprana edad pueden proporcionar a los estudiantes experiencias concretas para cultivar el pensamiento computacional.

Análogamente, el aprendizaje de conceptos del pensamiento computacional a través de una actividad desconectada podría contribuir al aprendizaje significativo en alumnos de 6° y 9° grado, tal como lo indica Delal & Oner (2020) y Looi, et al. (2018) quienes sugieren que los alumnos en

estos grados pueden beneficiarse más de las actividades desconectadas al desarrollar sus habilidades de descomposición, diseño algorítmico, abstracción, evaluación y generalización, mejorando su capacidad de expresar más asertivamente un "algoritmo de ordenación" mediante pseudocódigo o un diagrama de flujo. Asimismo, Brackmann et al (2017) llevaron a cabo un cuasiexperimento con 73 alumnos de 5º y 6º grado, encontrando diferencias estadísticamente significativas entre el grupo experimental que trabajó actividades desconectadas y el grupo de control quienes no participaron de estas clases. Después de una intervención de una vez a la semana durante cinco semanas, el grupo de control no obtuvo una mejora estadísticamente significativa entre el pre-test y el pos-test (t= 1,128; p(t)= .267> .05) y el tamaño del efecto de la mejora para este grupo fue bajo (d=.17). Por el contrario, se encontró una mejora estadísticamente significativa pre-post en la puntuación del test en el grupo experimental (t=4,431; p(t)= .000 <.001) y un tamaño del efecto 'grande' (d=.80). De igual forma se encontraron diferencias estadísticamente significativas entre el grupo de control y el grupo experimental una vez finalizada la intervención (t=3,730; p(t)= .000 < .001) reafirmando las mejoras en las habilidades de los estudiantes después de recibir actividades desconectadas.

Por su parte, Tonbuloğlu y Tonbuloğlu (2019) examinaron el efecto de las actividades desconectadas sobre las habilidades de pensamiento computacional en 114 estudiantes de 5° grado sin formación previa en codificación, los cuales recibieron clases de informática 2 horas a la semana durante un periodo académico de 10 semanas. Luego de medir los niveles de habilidades de pensamiento computacional mediante la escala desarrollada por Korkmaz et al. (2017) a través de 22 ítems con una escala tipo Likert de 5 puntos, encontraron una diferencia significativa (p < 0,05) entre las puntuaciones del pre-test y del post-test y un efecto positivo en la mejora de subdimensiones como la creatividad, el pensamiento algorítmico, la colaboración y el pensamiento

crítico, excepto en la habilidad para resolver problemas. Adicionalmente, durante las actividades de codificación no conectadas, se observó que los alumnos estaban entretenidos, tenían un alto nivel de motivación y de asistencia a clase, pero también tenían dificultades para el uso de operadores lógicos, diagramas de flujo y variables, así como en los temas que requerían conocimientos matemáticos. Estos resultados muestran que los alumnos de los grupos experimentales, que participaron en actividades desconectadas, mejoraron significativamente sus habilidades más que aquellos que no participan de dichas actividades durante las clases, lo que demuestra que el enfoque desconectado puede ser eficaz para el desarrollo de las habilidades del pensamiento computacional en educación secundaria.

Al igual que con los métodos conectados, los enfoques de enseñanza desconectados en educación media son un apoyo fundamental en la educación de las ciencias de la computación y la programación (Huang & Looi, 2021; Bell & Vahenrenhold, 2018; Looi, et al. 2018). No obstante, el estudio cuasiexperimental de Rondón (2020) con 59 estudiantes de grado décimo, utilizando actividades desconectadas como alternativa a la enseñanza tradicional en programación, no reportó diferencias significativas ($\sigma1 = 1.43$, $\sigma2 = 1.81$) entre el grupo experimental ($X1 = 4.05$) y el grupo de control ($X2 = 3.70$), por lo que se puede afirmar que los resultados respecto al uso de las actividades desconectadas en el desarrollo y fortalecimiento de las habilidades del pensamiento computacional en educación media aún no son concluyentes. Es por esto que autores como Bell & Vahenrenhold (2018), Iglesias & Bordignon (2021) y Huang & Looi (2021) exponen la necesidad de contar con nuevas herramientas para organizar, analizar y medir la efectividad de las actividades desconectadas, tal que las investigaciones futuras estén fundamentadas en el análisis y valoración del impacto que tienen en el aprendizaje respecto a otras estrategias de enseñanza, permitiendo

comprender mejor la naturaleza y el desarrollo del pensamiento computacional, principalmente en grupos de estudiantes sin acceso a la computación.

2.1.3.3 Enfoques mixtos

Algunos estudios recientes han abordado diferentes métodos con el propósito de evaluar la combinación de actividades en el desarrollo del pensamiento computacional, entre ellos Del Olmo-Muñoz et al (2020) quienes combinaron actividades conectadas y desconectadas en educación Primaria, o Ramírez et al. (2022) y López y Pineda (2022) en estudiantes de secundaria.

Del Olmo-Muñoz et al (2020) realizó un estudio de ocho semanas en el que participaron 84 alumnos de 2° de primaria sobre una selección de actividades extraídas de los cursos de *Code.org*, conformado por un grupo control con actividades desconectadas y un grupo experimental que trabajó con actividades conectadas complementarias. El impacto de estas actividades se evaluó mediante una selección de tareas-problemas en un orden de menor a mayor dificultad extraídas del "Concurso Internacional *Bebras*". Los datos descriptivos y una prueba de Mann-Whitney indicaron que a pesar de las diferencias a favor del grupo conectado (Pre-testCT = 3,87) en comparación con el grupo desconectado (Pre-testCT = 3,26) antes de la intervención (U=506,00, p=.009), evidencian una mejora experimentada por el grupo desconectado a lo largo del estudio debido a la diferencia media entre el Pre-testCT y el Mid-testCT (U = 369,00, p = .033), y entre el Pre-testCT y el Post-testCT (U = 373,00, p = .003). Por su parte, Leifheit et al (2018) realizaron un estudio impartido a 33 alumnos de 3º y 4º de primaria con lecciones basadas en *Code.org* que abordaron como eje central los algoritmos mediante el aprendizaje basado en juegos desconectados para aumentar la comprensión de los estudiantes en conceptos como secuencias, bucles, ramas condicionales y eventos. El interés y la motivación de los alumnos por la enseñanza de la programación se midieron con cuestionarios de autoevaluación antes y después del curso. En

particular, observaron que (a) para todos los conceptos, excepto en ramas condicionales, los estudiantes alcanzaron una media del 82% de los objetivos de aprendizaje, y (b) los estudiantes valoraron positivamente su experiencia de aprendizaje en el curso (M=4.36, SD=0.89) e informaron altos niveles de interés en aprender más sobre temas relacionados con la computación (M=3.36, SD=1.34). Además, los resultados indicaron los beneficios del enfoque basado en juegos desconectados para la enseñanza de conceptos, sin embargo, los análisis cualitativos indicaron que parte del plan de estudios era muy complejo para el grupo objetivo, por ejemplo, los condicionales anidados.

En secundaria, Ramírez et al. (2022) trabajaron con 22 alumnos del grado 6° en talleres de programación en bloques basados en *Scratch* y recursos didácticos aplicados a la programación conectada y desconectada con el fin de incorporar las habilidades básicas del pensamiento computacional. Para medir el grado de aceptación de las actividades conectadas y desconectadas, se realizó un test de 15 peguntas, que fue validado con un alfa de Cronbach de 0.717. De acuerdo con el trabajo realizado, la estrategia pedagógica combinada entre conectada y desconectada, si es aspecto diferenciador en la enseñanza tradicional, así como también incentiva a los estudiantes a participar de manera activa en la implementación de proyectos de pensamiento computacional. En el mismo sentido, López y Pineda (2022) quienes, en su investigación desarrollada con una muestra poblacional de 107 estudiantes de los grados 6° y 7°, determinaron la incidencia de las actividades conectadas y desconectadas en el desarrollo de las habilidades de pensamiento computacional mediante el test de pensamiento computacional propuesto por Román-González et al. (2015) y el logro previo en el área de matemáticas como elemento importante a considerar en la resolución de problemas. Los datos de pretest y pos-test del test fueron analizados mediante un análisis MANCOVA y arrojo que no se encontraron diferencias significativas entre el pretest (conectadas,

M=1.95, SD=0.54 y desconectadas, M=2.04, SD=0.57), y el pos-test (conectadas: M=2.38, SD=0.88) y desconectadas (M=2.46, SD=0.81). Sin embargo, esta investigación concluye que independientemente del medio que se utilice para la apropiación de conceptos computacionales si hay una mejoría en el desarrollo de habilidades del pensamiento computacional en los estudiantes.

Cabe resaltar el trabajo realizado por Sun et al. (2021) con un modelo mixto más amplio en un total de 158 estudiantes de 7° de secundaria con cuatro grupos experimentales y cuatro enfoques de programación: conectadas, desconectadas, desconectadas y conectadas, conectadas y desconectadas, y un grupo de control sin programación. Los resultados del análisis ANOVA de una vía para el pretest y pos-test mostró que los grupos conectadas (M=25.76, SD=12.99, t=11.384, p<.001), desconectadas (M=19.70, SD=14.68, t=7.707, p<.001), desconectadas-conectadas (M=31.72, SD=15.37, t=11.116, p<. 001) y el grupo conectadas-desconectadas (M=21.08, SD=17.51, t=15.133, p<.001) alcanzaron una mejora significativa en las habilidades de pensamiento computacional en comparación con el grupo de control que no participó en ninguna actividad de programación (M=6.77, SD=17.96, t=2.100, p=.054). Sin embargo, el modelo combinado desconectadas-conectadas mostró una mayor diferencia indicando que fue particularmente eficaz en mejorar las habilidades de pensamiento computacional. De igual forma, Zhang & Cui (2021) desarrollaron un estudio con 70 estudiantes de 7° grado con edades entre los 11 y los 13 años quienes participaron en el curso con sesiones conectadas y desconectadas que abarcaba conceptos computacionales, resolución de problemas y prácticas de programación. Los resultados indicaron que los estudiantes con bajo rendimiento fueron los que más se beneficiaron del curso, experimentando una mejora en los exámenes presentados, lo que indica que el enfoque de aprendizaje mixto del curso puede tener un mayor impacto en los estudiantes con menor rendimiento en programación. Curiosamente, los estudiantes con experiencia en programación

tendían a no gustarles las actividades desconectadas, lo cual fue inesperado dado que algunos estudios mostraron resultados positivos sobre estas actividades.

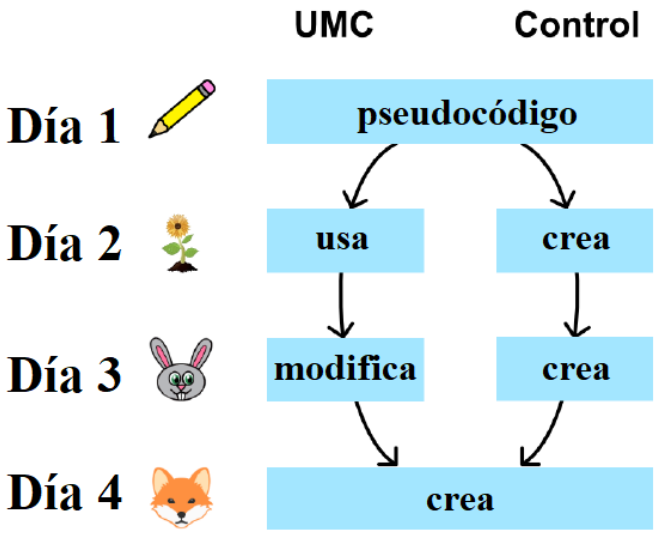
En ese orden de ideas, incluir actividades conjuntas en la instrucción de conceptos de pensamiento computacional parece beneficiosa para el alumnado, más aún, se requieren más estudios para determinar sí un enfoque mixto que combine actividades desconectadas y conectadas en lugar de hacerlo por separado resulta más adecuado para trabajar el pensamiento computacional (Del Olmo-Muñoz et al, 2020).

2.1.4 Desarrollo del pensamiento computacional mediante el modelo usa-modifica-crea

Una estrategia curricular para abordar el desarrollo del pensamiento computacional, denominada usa-modifica-crea, ha sido recientemente estudiada por diferentes autores (Lytle, Catete, Boulden et al, 2019; Lytle, Catete, Isvik et al., 2019; Estévez et al, 2019; Houchins et al., 2021; Boulden et al., 2021 y Rachmatullah, et al. 2021). Esta estrategia, presenta un enfoque orientado a llevar progresivamente al estudiante a la resolución de problemas mediante el uso, modificación y creación de artefactos computacionales, proporcionando un andamiaje suficiente de apoyo y profundización para la adquisición de conceptos de pensamiento computacional. Ante esto, Lytle, Catete, Boulden et al. (2019) realizaron una comparación entre dos progresiones de lecciones usa-modifica-crea para el pensamiento computacional en clases de ciencias de secundarias con un total de 160 estudiantes de 6º entre los 11 y 12 años. Un primer grupo de estudiantes participó en una actividad de cuatro días centrada en el desarrollo de una simulación basada en agentes que usa un entorno de programación basado en bloques, mientras que el segundo grupo utilizó un plan de lección con el andamiaje usa-modifica-crea propuesto por Lee et al (2014) (ver Figura 2).

Figura 2

Plan de lecciones usando el modelo usa-modifica-crea



Nota. Adaptado de "Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes". (p. 397) por Lytle, Catete, Boulden et al., 2019, In Innovation and Technology in Computer Science Education, ITiCSE'19, 1.

Según los resultados, el grupo que usó el andamiaje usa-modifica-crea, no reportó diferencias significativas entre los cuatro días de trabajo ($\chi 2$= 1,879, p = 0,598), sin embargo, el grupo de control si mostró diferencias significativas ($\chi 2$ = 9.984, p = 0,019). Al comparar la introducción a la codificación del día 2 frente a los otros, se encontró una diferencia significativa (W = 2238, p = .002), siendo más fácil el trabajo para los estudiantes del grupo experimental que para el grupo control quienes percibieron mayor dificultad. Posteriormente, este diseño fue evaluado por Lytle, Catete, Isvik et al. (2019) con una segunda adaptación del modelo usa-modifica-crea para tareas de pensamiento computacional, la cual sirvió como andamiaje a 38 estudiantes y 2 profesores de 6º grado en entornos de programación basados en bloques, mediante lecciones de cuatro días en las que participaron de una actividad desconectada y el desarrollo de

una simulación en un entorno basado en bloques llamado *Snap*. La triangulación entre las evidencias de programación de los estudiantes, las observaciones y encuestas a profesores en una escala Likert de 1 a 5, demostró que, aunque los estudiantes tuvieron una tasa más alta de completar los ejercicios del modelo original (15/28) que en el nuevo diseño (15/38) ésta no fue estadísticamente significativa ($p = .38$). De igual forma, una prueba de Friedman no mostró una diferencia significativa en la dificultad percibida por los estudiantes a través de las lecciones en los cuatro días ($p =0.10$), ni tampoco para el modelo adaptado ($p = 0.92$). De igual forma, las pruebas U de Mann-Whitney no arrojaron diferencias significativas en las respuestas de dificultad entre condiciones en los días uno ($p = .22$), dos ($p= .34$), tres ($p = .52$) o cuatro ($p= .91$). Al no encontrar diferencias significativas en las percepciones de dificultad de los alumnos, haber completado la mayoría de la actividad base (18/28) y no registrar dificultades para enseñar el nuevo modelo, descubrieron que el diseño adaptado a elección del alumno no era más difícil ni para los alumnos ni para los profesores. Por lo tanto, creen que el modelo original mantiene todas las ventajas de la estrategia curricular original, usa-modifica-crea, para facilitar el desarrollo de habilidades de pensamiento computacional en la programación.

Estévez et al (2019) siguieron un estudio con enfoque en el pensamiento computacional para introducir a 37 estudiantes de educación media secundaria (16-17 años) en los fundamentos y el funcionamiento de dos algoritmos de inteligencia artificial: *K-means* y una red neuronal artificial (RNA). Mediante la progresión usa-modifica-crea, los estudiantes pudieron explorar el comportamiento de ambos algoritmos mediante la herramienta de *Scratch* y realizar algunos experimentos numéricos, obteniendo una mayor comprensión del pensamiento computacional subyacente a los procesos de inteligencia artificial. Luego de medir el impacto diferencial de la experiencia a través de cuestionarios antes y después del taller, el análisis cualitativo de las

respuestas a preguntas abiertas muestra un buen grado de impacto del taller, mostrando una diferencia significativa en el nivel de conocimientos técnicos necesarios para entender la inteligencia artificial ($p<1e-5$, OR=20,7), la forma en que un algoritmo puede predecir un evento ($p<1e-5$, OR=0,17) y la posibilidad de que a futuro se desarrolle una inteligencia artificial avanzada ($p<1e-8$, OR=7,0). Gracias al taller, los alumnos adquirieron las condiciones necesarias para comprender el funcionamiento de los algoritmos de inteligencia artificial, mejorando su capacidad de contemplar la similitud entre dos situaciones aparentemente muy diferentes formuladas como problemas de predicción y adquiriendo una visión más realista alejada de la cultura cinematográfica.

Boulden et al. (2021) diseño y desarrollo un juego para enseñar ciencias de la computación a estudiantes de 6° a 8° de educación media (N = 77) con edades de 11 a 13 años mediante retos de programación basada en bloques. El juego integra un marco pedagógico usa-modifica-crea para promover resultados exitosos en una amplia variedad de conceptos básicos de ciencias de la computación (variables, condicionales, bucles y algoritmos) los cuales componen los niveles temáticos del juego. Los resultados cuantitativos obtenidos antes y después en una evaluación de conceptos de programación asistida por computador mostraron una varianza significativa ($p < 0,001$), encontrando que el 27% de la variabilidad en la comprensión conceptual fue propio del estudiante ($\sigma2= 46.19$, $p <.001$) y el 73% fue entre estudiantes ($\tau= 127.90$, $p < .001$), indicando un cambio positivo significativo en la comprensión conceptual desde el pre-test hasta el pos-test ($t = 2,06$, $p = 0,045$). Esto representa que las puntuaciones obtenidas por los alumnos en el pre-test ($M= 48,57$) fueron significativamente inferiores a las obtenidas en el pos-test ($M= 51,78$). Además, la comprensión conceptual de los estudiantes no estuvo significativamente asociada con su experiencia previa en programación informática ($t = -1,71$, $p = 0,090$), ni de su experiencia previa

en ciencias de la computación (t = -0,87, p = 0,386) indicando que mejoraron significativamente su aprendizaje antes y después de interactuar con el juego. Por su parte los resultados cualitativos revelaron que la progresión del andamiaje usa-modifica-crea proporcionó a los estudiantes los conocimientos básicos necesarios para progresar a través de los niveles y retos del juego, reduciendo la carga cognitiva y facilitando la comprensión conceptual necesaria para resolver las tareas de programación abiertas de las fases de modificación y creación del juego, especialmente a aquellos con poca o ninguna experiencia previa en programación.

Rachmatullah, et al. (2021) demostró que la progresión usa-modifica-crea fue capaz de reducir la brecha de género en el rendimiento de ciencias de la computación a partir de un estudio sobre 255 alumnos de 6° a 8° grado de educación media con un diseño pretest-postest sin grupo control. Esta intervención de cuatro días con una sesión de 35-40 minutos permitió a los participantes realizar actividades de programación bajo la progresión usa-modifica-crea y simular un fenómeno científico, utilizando código de programación basado en bloques a partir de una actividad desconectada la inspección y ejecución de modelos existentes (durante la fase usa), un conjunto completo de códigos disponibles para modificar (en la fase modifica) y la creación de un modelo completamente nuevo para representar el fenómeno científico objetivo (en la fase final, crea). Al medir el rendimiento y comprensión de los estudiantes en cuatro conceptos básicos (variables, condicionales, bucles y algoritmos) utilizando una evaluación de 24 preguntas de opción múltiple con resultado final entre 0 y 1, validada por Rachmatullah, Akram, et al. (2020), se encontró que el 76% de la variabilidad en el rendimiento fue entre estudiantes ($\tau 00=0.04$, $z = 9.62$, $p < .001$) y el 24% fue a causa del estudiante ($\sigma 2= 0.01$, $z = 11,29$, $p < .001$), que la variabilidad en la certeza a las respuestas correctas entre alumnos fue del 78% ($t00 = 0.07$, $z = 9.83$, $p < .001$) mientras que dentro del estudiante fue del 22% ($\sigma 2=0.02$, $z = 11.29$, $p < .001$). Por

último, la variabilidad del sentimiento de incertidumbre entre estudiantes y dentro de un mismo estudiante fueron del 72% y el 28%, respectivamente ($t00=0.05$, $z = 9.61$, $p < 0.001$; $\sigma2 =0.02$, $z = 11.29$, $p < 0.001$). Además, se encontró que no había un efecto significativo de la interacción entre el género y la experiencia previa de los estudiantes sobre su aprendizaje al trabajar los conceptos en la progresión usa-modifica-crea (, $p < 0.05$), concluyendo que la estrategia usa-modifica-crea no solo ayudó a reducir los efectos del género y la experiencia previa en programación que pudieran existir, si no que mejoró significativamente tanto la sensibilidad a las respuestas correctas como el sentimiento de incertidumbre de los alumnos hacia las respuestas incorrectas.

Finalmente, las observaciones de una investigación cualitativa sobre enfermedades epidémicas impartida en cinco aulas de ciencias de tres colegios con estudiantes de secundaria, realizada por Houchins et al. (2021) demostró que mediante la adopción de la progresión usa-modifica-crea más estudiantes fueron capaces de construir de forma independiente modelos computacionales, haciendo que se sintieran más activos y dueños de los artefactos computacionales que produjeron. Las anécdotas recogidas de los profesores también sugieren que la secuencia de andamiaje del usa-modifica-crea tiene beneficios para aquellos que son nuevos en la implementación del modelado computacional en ciencias (Houchins et al, 2021).

2.2. Marco teórico

De acuerdo con los antecedentes se evidencia que la teoría entorno al concepto del pensamiento computacional es diversa y suele ser abordada desde diferentes disciplinas, es por ello que este capítulo aborda las diferentes definiciones, la falta de consenso en una definición general y los retos para su desarrollo y evaluación. Finalmente se exponen, las actividades conectadas y desconectadas en el desarrollo del pensamiento computacional, en qué consisten, su

taxonomía, diseños, artefactos utilizados y principales escenarios, así como el modelo usa-modifica-crea y su relación con el pensamiento computacional.

2.2.1 Pensamiento computacional

El pensamiento computacional es una expresión que ha recibido mucha atención en los últimos años, sin embargo, hay poco acuerdo sobre lo que abarca y aún menos sobre las estrategias para evaluar su desarrollo en los jóvenes (Brennan & Resnick, 2012). Las bases del pensamiento computacional se forjaron a partir de las ideas planteadas por Seymour Papert (1980) en su libro "*Mindstorms*: niños, computadoras e ideas poderosas" pero fue Jeannette Wing, quien lo definió describiendo la forma de cómo piensa un científico de computadoras, partiendo no sólo de sus habilidades específicas sino del conjunto de herramientas mentales que usa para resolver problemas en diferentes áreas. Ante esto, Wing (2008) afirma que todo ser humano tiene la capacidad de desarrollar el pensamiento computacional y que por lo tanto debería ser parte de las habilidades de análisis de todo estudiante, argumentando que además de implicar la resolución de problemas, el diseño de sistemas y la comprensión del comportamiento humano también está relacionado con la habilidad para entender cómo funcionan las herramientas computacionales y como se resuelven problemas tal como "un agente de procesamiento de información" lo haría (Wing, 2008). Años más tarde Wing complementa el concepto, definiéndolo como una actividad mental para formular un problema que permita una solución informática, producto de la combinación de personas y máquinas (Wing, 2011). Por su parte, Grover y Pea (2017) coincide con la definición planteada por Wing al afirmar que es "El proceso del pensamiento involucrado en formular un problema y expresar su solución en la forma en la que un computador, humano o máquina pueda efectivamente darle solución". (Grover y Pea, 2017, p.21). Estas dos visiones reconocidas internacionalmente, sirvieron de apoyo para algunas investigaciones en el marco de

las discusiones del pensamiento computacional, especialmente en aspectos como la: integración en el sistema educativo y las acciones en política pública que plantean nuevos retos para la educación (Quiroz-Vallejo et al., 2021).

Otros autores como Brennan y Resnick (2012), Selby & Woollard (2013), Grover y Pea (2017), e instituciones como *Computer Science Teachers Association* (CSTA) e *International Society for Technology in Education* (ISTE) realizaron diferentes aportes desde lo conceptual y lo operacional a una posible construcción y definición formal del pensamiento computacional. Según el estudio desarrollado por Brennan y Resnick (2012), los estudiantes que interactúan con *Scratch* han adoptado una serie de estrategias y prácticas para desarrollar sus medios interactivos, entre ellas ser gradual e iterativo, probar y depurar, reutilizar y remezclar, y abstraer y modular. Esta creación de medios interactivos resulta en un contexto favorable para el desarrollo de estas prácticas, las cuales son bastante útiles en una variedad de actividades de diseño, no solamente en programación sino también en espacios que fomentan la capacidad de expresión y conexión con otros, lo que favorece el trabajo colaborativo. Para poder evaluar el desarrollo del pensamiento computacional y la presencia de los conceptos computacionales los autores proponen tres aproximaciones, primero, usar un software que examine el portafolio de proyectos creados por los usuarios de *Scratch* tal que arroje cuales son los bloques más usados en cada categoría y aquellos que aún no han sido usados; segundo, una entrevista con los usuarios; y tercero una valoración directa sobre un escenario con tres proyectos susceptibles a ser modificados por el usuario.

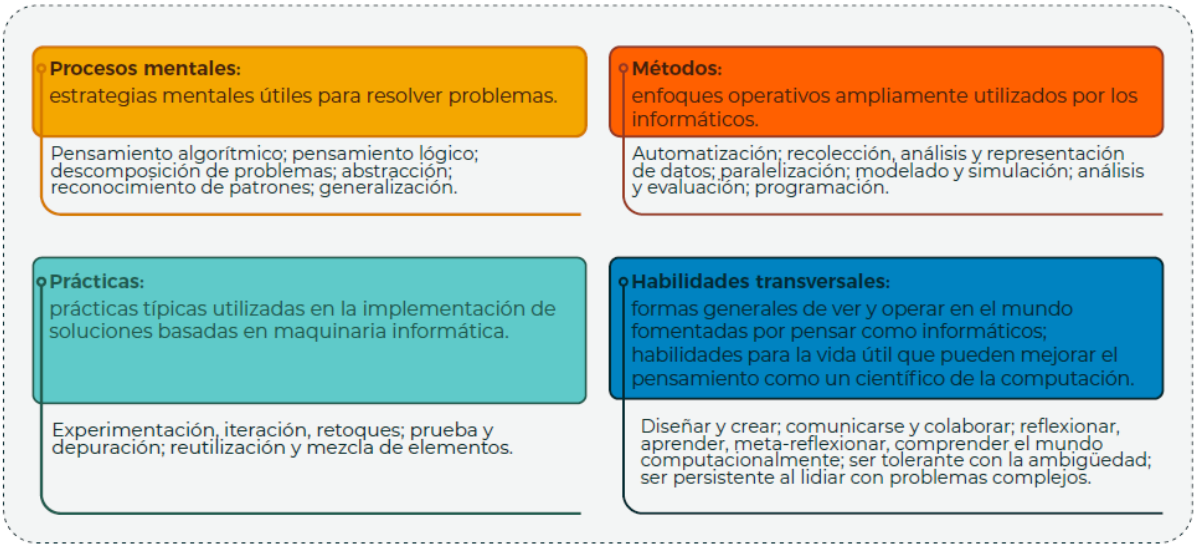
De acuerdo con Selby & Woollard (2013), se puede afirmar que el pensamiento computacional es una actividad, a menudo orientada al producto, asociada a la resolución de problemas, pero no limitada a ella, cuyo enfoque se centra en: la resolución de problemas, en procesos de abstracción, descomposición, diseño algorítmico, evaluación y generalizaciones. Por

su parte, Grover y Pea (2017) afirman que el pensamiento computacional es: El proceso del pensamiento involucrado en formular un problema y expresar su solución en la forma en la que un computador, humano o máquina pueda efectivamente darle solución", en otras palabras, "resolver un problema usando conceptos y estrategias que se relacionan a las ciencias de la computación. Para ello, reúnen los conceptos de lógica y pensamiento lógico, algoritmos y pensamiento algorítmico, patrones y reconocimiento de patrones, abstracción y generalización, evaluación y automatización; en cuanto a las prácticas, consideran la descomposición del problema, creación de artefactos computacionales, prueba y depuración, desarrollo incremental y colaboración y creatividad, las cuales hacen parte de competencias más amplias del siglo XXI. Mientras que la CSTA y la ISTE elaboran una definición operacional del pensamiento computacional, cuyo enfoque es resolver un determinado problema integrando las tecnologías digitales con ideas humanas y reforzando las habilidades de creatividad, razonamiento y pensamiento crítico.

Por su parte, Lodi (2020) comparó los elementos presentes en diferentes denominaciones del pensamiento computacional, encontrando que, aunque coinciden en que es una forma de pensar o un proceso de pensamiento, se enumeran algunos elementos constitutivos de diversa índole, que van desde hábitos de pensamiento hasta conceptos específicos de programación, agrupándolos en categorías, pero sin un acuerdo universal sobre su clasificación. De igual forma, Lodi (2020) argumenta que, aunque los puntos de vista de Papert (1980) y Wing (2006) se relacionan y coinciden en la discusión del potencial del pensamiento computacional, existen diferencias en sus fundamentos e interpretaciones por lo que propone una clasificación común bajo cuatro categorías que resumen aquellos ítems expuestos en las definiciones analizadas: procesos mentales, métodos, prácticas y habilidades transversales (ver Figura 3).

Figura 3

Clasificación e ítems expuestos en las definiciones del pensamiento computacional analizadas por Lodi (2020)



Fuente: Reproducido de Integración del pensamiento computacional en educación primaria y secundaria documento de posición. (p.14) por Carmona-Mesa et al., 2021, Siemens Stiftung, Siemens Caring Hands y Universidad de Antioquia.

Por lo anterior, las diversas definiciones del pensamiento computacional no logran plantear o llegar a un acuerdo en su clasificación, aunque sí identifican dos visiones desde el objeto de estudio y el medio, lo cual resulta en una alternativa para encaminar los procesos educativos de las clases en ciencias de la computación (Quiroz-Vallejo et al., 2021), además el hecho de que muchas de las prácticas asociadas al pensamiento computacional aún no han sido claramente definidas y relacionadas con el pensamiento matemático, el pensamiento algorítmico o la resolución de problemas, plantea un reto no solo para los docentes, sino también para las ciencias de la computación en general (Weintrop, 2016).

2.2.2 Integración del pensamiento computacional en el currículo

Mantilla & Negre (2021) mencionan que para incorporar el desarrollo del pensamiento computacional desde los grados iniciales hasta bachillerato, la educación tiene como reto preparar

individuos que no sólo logren dominar escenarios digitales y resolver problemas con ayuda de la tecnología, sino que, enfrenten situaciones que surgen en una sociedad, como es el caso, de las fronteras físicas, la pandemia por COVID-19, y otros factores que obligan al aislamiento físico social, por lo que es imperativo que la formación desde la escuela este orientada a aumentar su creatividad y toma de decisiones que involucren actividades para diseñar algoritmos y programar. En este mismo contexto, García (2022), coincide en fomentar el pensamiento computacional en las nuevas generaciones desde los niveles iniciales, así como también las competencias como: la innovación, la creatividad, el pensamiento analítico y el trabajo colaborativo, entre otras. Por su parte, Rojas (2021), considera que el pensamiento computacional es un tema que está al orden del día de las instituciones educativas colombianas en básica y media, pero que el desafío es encontrar la forma para evaluar e interpretar los resultados de su aplicación en el currículo. Esto lleva a Quiroz-Vallejo et al. (2021) a plantear cuatro retos fundamentales para la integración del pensamiento computacional en educación primaria y secundaria: 1) La consolidación de orientaciones teóricas y metodológicas para llevar a la práctica el pensamiento computacional; 2) La necesidad de generar disposiciones curriculares y políticas públicas que garanticen continuidad y durabilidad de las iniciativas; 3) Formación inicial y continuada de profesores en pensamiento computacional, tal que contribuya a los sistemas educativos latinoamericanos y a su integración en la educación primaria y secundaria4) Promover la cooperación entre instituciones educativas, entes gubernamentales y empresas, de tal manera que se generen espacios para la sistematización y divulgación de experiencias.

Aunque aún no se plantea una descripción precisa de cómo se debe aprender a pensar computacionalmente entre los jóvenes, estos autores concuerdan que integrarlo en los entornos escolares, mejora su participación en diferentes programas orientados a las ciencias de la

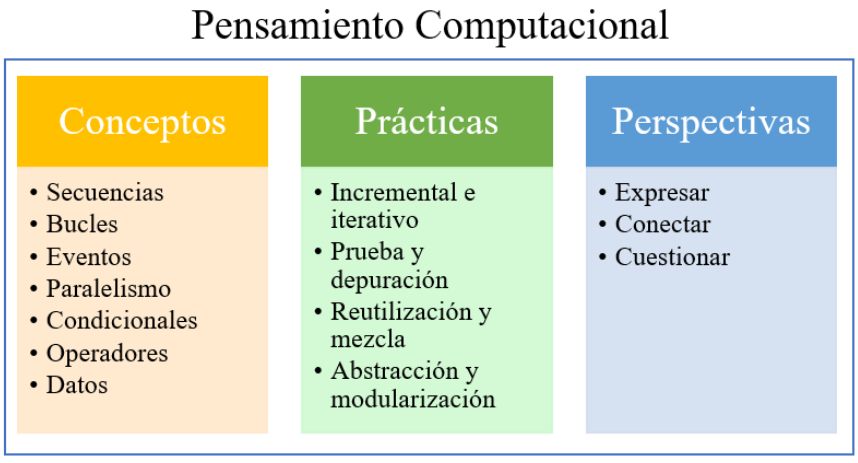
computación y un mayor acceso a entornos de aprendizaje enriquecidos. (Lee et al., 2011, Huang & Looi, 2021). Por su parte, Polanco et al. (2020) encontraron que el pensamiento computacional es visto como opción para la representación, adquisición de conocimientos y atención a problemas, que hay diversidad de opiniones con respecto a la definición, pero que la mejor opción es la incorporación al currículo escolar obligatorio desde los grados iniciales, con el fin de aplicar los conceptos computacionales para fortalecer y potenciar la capacidad de análisis en los niños. En esta misma dirección, Quiroz-Vallejo et al. (2021), identifica al menos tres posibles perspectivas frente a la discusión de la integración del pensamiento computacional en los currículos: 1) incluirlo en informática, tecnología o una asignatura diferente, 2) una integración transversal, y 3) no integrarlo, sino que se promueva a través de actividades extracurriculares.

2.2.3 Lo que se entiende por pensamiento computacional en este trabajo

Para los efectos de esta investigación, se considera el marco conceptual propuesto por Brennan y Resnick (2012), que surgió a partir de los estudios sobre las actividades desarrolladas en el contexto de *Scratch* y que se basa en tres dimensiones: conceptos, prácticas y perspectivas computacionales como lo muestra la Figura 4.

Figura 4

Marco conceptual propuesto por Brennan y Resnick (2012)



Para Brennan y Resnick (2012), los conceptos se adquieren en la medida en que los diseñadores van programando e identificando un conjunto de términos que son comunes a los lenguajes de programación y son útiles en una amplia gama de proyectos, entre ellos: secuencias, bucles, paralelismo, eventos, condicionales, operadores y datos (ver tabla 1)

Tabla 1

Conceptos computacionales identificados por Brenann y Resnick (2012).

Concepto	Descripción
Secuencias	Serie de pasos individuales o instrucciones que pueden ser ejecutadas por el computador.
Bucles	Mecanismo para ejecutar la misma secuencia varias veces.
Eventos	Componente esencial de los medios interactivos que produce una acción.
Paralelismo	Secuencia de instrucciones que suceden al mismo tiempo que otras.
Condicionales	Instrucción con capacidad de tomar decisiones en función de determinadas condiciones.
Operadores	Elementos que sirven de soporte para expresiones matemáticas, lógicas y de cadena, permitiendo realizar manipulaciones numéricas y de cadena.
Datos	Elementos que permiten almacenar, recuperar y actualizar valores

Las prácticas informáticas hacen referencia a las estrategias que desarrollan los diseñadores para crear sus proyectos, las cuales se centran en el proceso de pensamiento y aprendizaje, yendo más allá de lo que se aprende y centrándose más en cómo se aprende. Ante esto, Brennan y Resnick (2012) observaron cuatro conjuntos principales de prácticas: ser incremental e iterativo, probar y depurar, reutilizar y remezclar, y abstraer y modularizar, de allí que la creación de medios interactivos es un contexto poderoso para desarrollar prácticas útiles en diversas actividades de diseño, no sólo en la programación (ver Tabla 2).

Tabla 2

Prácticas computacionales observadas por Brenann y Resnick (2012).

Práctica	Descripción
Incremental e iterativo	Proceso adaptativo en el que el diseño puede cambiar en respuesta a la búsqueda de una solución mediante pequeños pasos.
Pruebas y depuración	Desarrollo de estrategias para afrontar y anticiparse a los problemas que conlleva la implementación de una solución.

Reutilización y remezcla	Práctica arraigada en la programación que permite basarse en el trabajo de otras personas y crear cosas mucho más complejas de las que se pueden crear por sí mismo.
Abstracción y modularización	Construir algo grande juntando colecciones de partes más pequeñas, desde el trabajo inicial de conceptualizar el problema hasta traducir el concepto en partes individuales.

Por último, están las perspectivas del pensamiento computacional que describen el proceso de construcción y representan otros elementos inmersos en el aprendizaje, las cuales se basan en la representación que tienen los estudiantes sobre sobre sí mismos, sus relaciones con los demás y con el mundo tecnológico que les rodea (ver tabla 3).

Tabla 3

Perspectivas computacionales observadas por Brenann y Resnick (2012).

Perspectiva	**Descripción**
Expresar	Expresión de actividades importantes para aprender a utilizar la tecnología, el diseño y la autoexpresión.
Conectar	Cómo el diseño de medios computacionales se enriquece de las interacciones con los demás.
Cuestionar	La capacidad de hacer preguntas sobre y con la tecnología

2.2.4 Actividades conectadas

Las actividades conectadas plantean interacciones con dispositivos digitales con el fin de lograr una aproximación a la programación (Rondón, 2020; Sanabria, 2022). En este sentido, la exploración de ambientes conectados permite indagar entornos informáticos, lenguajes de programación, simulaciones y plataformas digitales con el propósito de desarrollar habilidades propias del pensamiento computacional (Sanabria, 2022). De allí que Brennan & Resnick (2012) afirmen que la programación mediada por actividades de aprendizaje basadas en el diseño, en particular, la programación de medios interactivos proporciona un contexto y un conjunto de oportunidades que contribuyen al pensamiento computacional, y que son útiles en una variedad de actividades de diseño, no solo de programación.

Según González (2018), se distinguen dos categorías de actividades conectadas para desarrollar el pensamiento computacional: 1) Actividades basadas en dispositivos digitales, tales como juegos lógicos, narración de historias, entre otros y 2) Actividades basadas en computador que involucran distintas tareas de programación, ya sea utilizando herramientas visuales por bloques (como Scratch o Alice) o lenguajes de programación de bajo nivel (como Java o C++). Gracias al desarrollo de recursos y actividades orientadas a la programación por bloques se han desarrollado una variedad de aplicaciones y entornos de programación que favorecen el aprendizaje en términos de codificación sin tener que profundizar en la sintaxis de los lenguajes (Ortega, 2020; Kalelioğlu, 2015). La característica principal de los entornos de programación basados en bloques es que las instrucciones se presentan como piezas de un rompecabezas, donde la forma e incluso el color del bloque define cómo y dónde puede utilizarse, de ahí la facilidad de arrastrar los bloques y armar un programa ya que sólo pueden encajarse combinaciones válidas de bloques, evitando errores de sintaxis que no sean válidos (Weintrop, 2021). Estas plataformas comparten un conjunto de características, entre las que se encuentran: un entorno gráfico en el que los alumnos construyen sus algoritmos; bloques organizados por submenús o categorías; comandos para crear código utilizando animaciones, sonidos o personajes según la edad del público objetivo; y otras entre las que se proporciona retroalimentación instantáneo o recompensa con mensajes y sonidos que indican alegría y éxito, tal como un sistema de gamificación (Barradas et al., 2020).

En la actualidad, los entornos de programación más populares que favorecen el aprendizaje en términos de codificación sin tener que profundizar en la sintaxis de los lenguajes de programación utilizan *Blocky*, una serie de librerías de lenguaje de programación visual que permite arrastrar y soltar bloques, utilizando la metáfora de unir piezas de un rompecabezas para crear programas (ver Figura 5). Estas herramientas de desarrollo son lo suficientemente potentes

para ampliar los conceptos computacionales de los estudiantes y dependiendo del nivel de complejidad de los artefactos computacionales creados, brindan la oportunidad de aplicar la mayoría de los conceptos identificados por Brennan y Resnick (2012) fomentando sus habilidades en la resolución de problemas (Ching et al, 2018).

Figura 5

Entornos de programación por bloques más populares

Scratch, un entorno de programación con una interfaz sencilla que permite a los jóvenes crear historias digitales, juegos y animaciones, promueve el pensamiento computacional y las habilidades en resolución de problemas, la enseñanza y aprendizaje creativo, la auto expresión y colaboración, y la igualdad de oportunidades en informática (Scratch, 2022). La interfaz de Scratch permite crear programas encajando bloques de instrucciones de programación, igual como se encajan piezas de un rompecabezas, de tal forma que "cuando los jóvenes diseñan medios interactivos con *Scratch,* se comprometen con un conjunto de conceptos computacionales (correspondientes con los bloques de programación de *Scratch*) que son comunes en muchos lenguajes de programación" (Brennan & Resnick, 2012, p. 3).

Alice es un innovador entorno de programación basado en bloques que facilita la creación de animaciones, la construcción de narraciones interactivas o la programación de juegos sencillos en 3D, que a diferencia de muchas de las aplicaciones de programación basadas en bloques motiva

el aprendizaje a través de la exploración creativa. *Alice* está diseñado para enseñar habilidades de pensamiento lógico y computacional, y los principios fundamentales de programación como una primera exposición a la programación orientada a objetos, proporcionando herramientas y materiales complementarios para la enseñanza de las ciencias de la computación en un espectro amplio de edades (Cooper, 2010).

Kodu, permite crear juegos en un computador con Windows mediante un sencillo lenguaje de programación visual, el cual puede utilizarse para el desarrollo de la creatividad, resolución de problemas y la narración de historias, sin conocimientos previos de diseño o programación. *Kodu* incluye un completo entorno de desarrollo de juegos en 3D, que incluye un editor de terrenos, herramientas de diseño, menús de personajes y una serie de complementos tecnológicos como: controles de cámara, detección de colisiones, física, etc. que permiten a los usuarios finales crear los escenarios del mundo con los que funcionará su código (MacLaurin, 2011).

Code.org, una plataforma de codificación que permite a los estudiantes aprender informática a través de la programación con bloques, incluye actividades desconectadas que complementan el aprendizaje de conceptos en torno a las ciencias de la computación (Kalelioğlu, 2015). Frente a este recurso, Barradas et al. (2020) señala que *Code.org* es una opción válida para desarrollar el pensamiento computacional en edades tempranas y una buena forma de que los alumnos comiencen a resolver problemas de la vida real estimulando la capacidad de abstracción mediante la práctica simulada y experimentada al mismo tiempo que entrenan sus habilidades en resolución de problemas, construyendo y reteniendo mejor los conocimientos abordados. Es por esto que la combinación de programación basada en bloques y la programación basada en texto, facilitan la comprensión de conceptos de programación, la creación de programas, y el desarrollo

de habilidades relacionadas con el pensamiento computacional (Laura-Ochoa y Bedregal-Alpaca, 2021; y Weintrop, 2015).

MakeCode es una plataforma de código abierto, diseñada por *Microsoft,* que permite el desarrollo de aplicaciones integradas por programadores no expertos y está orientada a dispositivos embebidos. *MakeCode* recurre a la programación por bloques sobre la tarjeta electrónica *Micro:bit* (Ball et al., 2016), la cual ha reportado un impacto positivo en la implementación de programas en más de 17 países y ha sido recientemente usada por Tyrén et al. (2018) para relacionar los conceptos básicos de programación en el proceso de diseño de las actividades de enseñanza que promuevan el pensamiento computacional y la resolución de problemas. Por su parte, *Micro:bit* es un dispositivo electrónico integrado diseñado específicamente por la BBC (*British Broadcasting Corporation)* para él ámbito educativo, que contiene un procesador ARM Cortex-M0 de 32 bits, sensores de nivel de luz, temperatura, aceleración, magnéticos, entradas táctiles, comunicaciones por USB y radio a 2,4 GHz.

2.2.5 Actividades desconectadas

Una estrategia utilizada para ejemplificar los principios del pensamiento computacional y proporcionar una experiencia práctica son las actividades desconectadas. Estas se presentan como una analogía de fácil comprensión que transmite conceptos de la informática sin necesidad de que los estudiantes tengan acceso a un computador (Bell et al., 2009). El término desconectado ha llegado a utilizarse para actividades de enseñanza de la informática que no implican programación con éxito, para desarrollar el pensamiento computacional desde la básica primaria hasta la educación superior, con el objetivo de que el estudiante pueda apropiarse de uno o varios conceptos específicos y usarlos en la solución correcta de un problema (Bell & Vahrenhold, 2018). Estas actividades tienen ventajas sobre la forma tradicional de resolución de problemas (Bell &

Vahrenhold, 2018; Vieira et al. 2019), permitiendo a los estudiantes explorar actividades físicas y utilizar recursos tangibles para representar y comprender los conceptos requeridos en el diseño de algoritmos, antes de hacer uso de un dispositivo.

En este mismo contexto, Delal & Oner (2020), encontraron que la enseñanza de la computación basada en actividades desconectadas es una estrategia pedagógica eficaz que ayuda a conseguir un alto nivel de compromiso y participación de los estudiantes, fomentando el trabajo en equipo y la colaboración, gracias a la constante vinculación de los escenarios informáticos con ejemplos de la vida real y al uso de objetos tangibles no relacionados con la informática. Estas actividades también ayudan a mejorar la retención de los conocimientos académicos adquiridos en el tiempo por parte de los participantes y resultaron ser más eficaces cuando se usan como actividades de introducción a nuevos temas. Un aspecto que se resalta de las actividades desconectadas es el factor de diversión, ya que juega un papel importante para que los estudiantes se involucren en las actividades y se diviertan en comparación son las lecciones regulares de computación, motivándolos a pensar creativamente y alentándolos a trabajar en equipos para alcanzar su objetivo.

Respecto a la naturaleza de las actividades desconectadas, Iglesias y Bordignon (2021) presentan, a manera de síntesis, una clasificación relacionada con el pensamiento computacional de acuerdo con: el componente lúdico que abarcan, las habilidades transversales que desarrollan y aquellas relacionadas con el pensamiento algorítmico, como se puede apreciar en las categorías identificadas en la Figura 6.

Figura 6

Árbol de taxonomía de actividades desconectadas

Fuente: Taxonomía de actividades desconectadas para el desarrollo de pensamiento computacional. Universidad Pedagógica Nacional por Iglesias y Bordignon, 2021 (p.123).

Siguiendo con la taxonomía propuesta de la Figura 6, Iglesias y Bordignon (2021) describen estos tipos de actividades desconectadas, usando un criterio las acciones que debe realizar el estudiante para su resolución la cual se muestra en la Tabla 4.

Tabla 4

Clasificación de las actividades desconectadas

Componente	***Actividades***	***Nivel Educativo***	***Ejemplos***
LÚDICO **Introducen juegos e incorporan desafíos para motivar a los estudiantes a exploran nuevos elementos, reglas y materiales, en los diferentes retos.**	Kinestésicas (Involucran Movimientos físicos o corporales)	Inicial Primaria	- Diseñar o crear rutinas de un baile - Construir con tarjetas una actividad que indiquen los pasos de una coreografía: aplaudir, saltar, girar, de acuerdo con como escogieron las tarjetas. - Búsqueda de caminos - Navegación en grafos, utilizando mapas dibujados en el suelo.
	Con recursos tangibles (Involucra el uso de objetos tangibles para representar un problema y su resolución)	Inicial	- Tangram, Rompecabezas, Caja de recursos didácticos de Froebel, Juegos de ingenio - Torre de Hanói) - Plataformas digitales interactivas que responden a ordenes mediante botones o tarjetas (sin computadora)
	Juegos de Mesa (Involucra diferentes elementos para plantear	Primaria	- Code & Roby o Code Master.

	problemas propios de la ciencia de la computación)		- Uso de tarjetas con instrucciones para guiar a los robots sobre una cuadrícula para alcanzar diferentes objetivos. - Guía a un personaje a través de un grafo utilizando fichas de colores que indican las aristas a recorrer por el avatar.
	Razonamiento lógico (Involucran estructuras de decisión)		- Uso de operadores lógicos aplicados a situaciones problemáticas -Enunciados que expresan restricciones que se deben cumplir para lograr un objetivo -Arboles de decisión.
HABILIDADES TRANSVERSALES Involucran el desarrollo de habilidades como la capacidad de abstracción, descomposición, reconocimiento de patrones, generalización y razonamiento lógico	Reconocimiento de patrones (Buscan trabajar las habilidades relacionadas con el descubrimiento de patrones y la capacidad de crear generalizaciones y abstracciones)		- Actividades basadas en elementos gráficos o en las características que poseen los ítems de una serie de elementos a los que hay que aplicarles una etiqueta o separarlos en alguna categoría. - Proyecto *Bebras*
	Cambio de representación (Desarrollan habilidades relacionadas con la capacidad de abstracción y el pensamiento lateral o divergente)		- El aprendiz debe realizar un cambio en la representación de datos para llegar más fácilmente a la solución de un problema - Tarjetas *Bebras* - Representar los datos de un problema mediante grafos.
	Optimización (Se basan en probar soluciones para evaluar sus resultados)		- Solucionar problemas optimizando el uso de un determinado recurso. -Maximizar o minimizar una variable.
PENSAMIENTO ALGORITMICO **Desarrollar algoritmos con diferentes niveles de dificultad.**	Ejecución de Algoritmos (Constituye un paso previo a la creación de programas)	Todos	- Construir imágenes en grillas a partir de reglas - Aplicar algoritmos de ordenamiento a un conjunto de cartas numeradas.
	Creación de Algoritmos (Involucran problemas que tienen soluciones más amplias donde sobresale la creatividad)	Diferentes niveles	- Enseñanza de la programación creando programas en “pseudocódigo” - Secuencias creadas con las tarjetas y se ejecuten en el juego Cody & Ruby
	Descubrimiento de Algoritmos (Busca que los estudiantes descubran cual es el algoritmo que hay detrás de alguna tarea para llegar a una solución)		- Búsqueda Binaria (buscar un elemento en una lista ordenada a partir de sucesivas comparaciones por mayor y menor) - Actividades que involucran ciclos

Fuente: Adaptado de la Taxonomía de actividades desconectadas, por Iglesias & Bordignon, 2021, p.5-15.

De acuerdo con lo anterior, dentro de la inmensa variedad de actividades desconectadas se encuentran ejercicios, juegos y problemas que se desarrollan sin tener un conocimiento técnico específico. Estos recursos fortalecen aspectos claves del pensamiento computacional como la capacidad de abstracción, el reconocimiento de patrones y la creación de un algoritmo (Ozcinar et al. 2017). Un referente son las actividades desconectadas “*CS Unplugged*”, las cuales han sido traducidas a más de 20 idiomas y usadas en más de 50 países para involucrar al público, sin tener que aprender a programar o utilizar un dispositivo digital (Iglesias & Bordignon 2021; Bell &

Vahrenhold, 2018) permitiendo eliminar la barrera del aprendizaje de la programación, especialmente en situaciones en las que no se dispone de ordenadores o existen otros problemas técnicos. Finalmente, cabe destacar las actividades desconectadas propuestas por *Code.org*, una organización sin fines de lucro dedicada a promover el acceso a la ciencia de la computación y aumentar la participación de mujeres, jóvenes y estudiantes en centros educativos, quienes ofrecen un plan de estudios de uso gratuito bajo licencias abiertas de *Creative Commons* y que reúne una lista de actividades desconectadas que no requieren conexión a internet ni dispositivos digitales.

En lo que se refiere a su naturaleza, estas actividades otorgan varias ventajas sobre los métodos tradicionales de la enseñanza de la resolución de problemas. Por un lado, nos permite centrarnos en los problemas y los conceptos a trabajar y no en las tecnologías que los resuelven; por otro lado, permiten trabajar los conceptos del pensamiento computacional con un nivel bajo de abstracción, ya que suelen ser representados por metáforas y objetos tangibles; y finalmente, su implementación no requiere una infraestructura tecnológica especial, en términos económicos son de bajo costo y pueden funcionar en cualquier institución (Iglesias & Bordignon, 2021).

2.2.6 Modelo de progresión de tres estados usa-modifica-crea

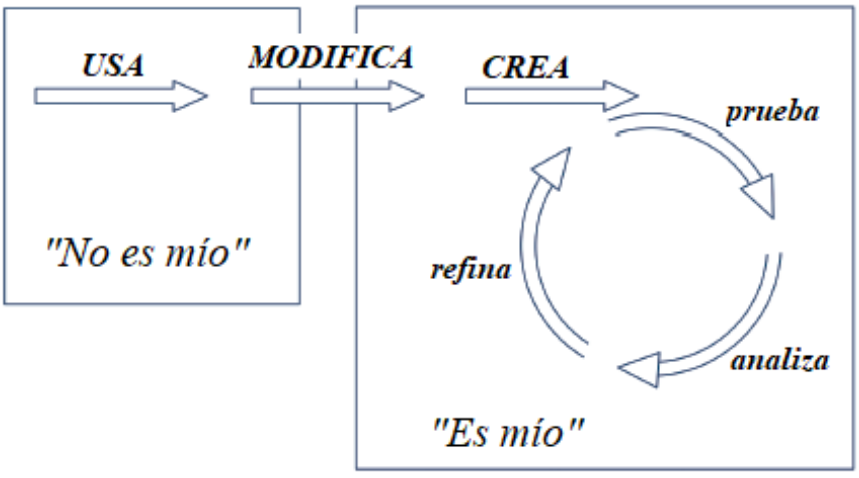
A partir de las recomendaciones realizadas por Lee et al. (2011) orientadas al aprendizaje y la enseñanza del pensamiento computacional en los grados K-12 surgió la propuesta de utilizar una progresión de tres etapas llamada usa-modifica-crea. Esta progresión involucra a los jóvenes en el pensamiento computacional dentro de entornos computacionales enriquecidos, describiendo un patrón de compromiso que apoya y profundiza el proceso de adquisición de conceptos computacionales en la creación de artefactos computacionales, basándose en la premisa de que “un andamiaje con interacciones cada vez más profundas promueve la adquisición y el desarrollo del pensamiento computacional” (Lee et al., 2011, p. 35). Estos autores afirman que esta estrategia

resulta útil a lo largo del tiempo, pues mejora la confianza y habilidades obtenidas en las primeras etapas por los estudiantes, permitiéndoles crear, desarrollar nuevas ideas y nuevos proyectos desde cero, o abordando problemas de su preferencia con su propio diseño.

Los pasos que conforman un modelo progresivo de tres estados se muestran en la Figura 7. En la primera etapa de este modelo, denominada "usa", los estudiantes son usuarios de la creación de otra persona, luego en la segunda etapa "modifica", el estudiante modifica el modelo, juego o programa con niveles crecientes de sofisticación para comprender aspectos claves del pensamiento computacional, como la abstracción y automatización, contenidas en el programa inicial. En la medida que los participantes realizan modificaciones también adquieren nuevas habilidades, lo que les permite desarrollar ideas para nuevos proyectos computacionales de su propio diseño y contextos de su preferencia. Finalmente, la etapa "crea", involucra los aspectos de abstracción, automatización y análisis, lo que permite poner a prueba la capacidad del estudiante para la creación de artefactos computacionales (Lee et al. 2011, p.35).

Figura 7

Secuencia de progresión de tres estados propuesta por Lee, et al (2011)



Nota. Adaptado de "Computational thinking for youth in practice" (p. 35) por Lee et al, 2011, ACM Inroads, 2(1).

Es así que, a través de una serie de modificaciones y mejoras iterativas, se desarrollan nuevas habilidades y conocimientos a medida que lo que antes era de otros, ahora, se convierte en propio. En consecuencia, los jóvenes adquieren habilidades y confianza, que los anima a desarrollar ideas para nuevos proyectos computacionales de su propio diseño en temas de su elección. Por estas razones, el modelo usa-modifica-crea ha sido ampliamente promovido como un medio para fomentar el aprendizaje del pensamiento computacional y el compromiso por parte de los estudiantes, al tiempo que permite la personalización y la adaptación creativa transformando a los alumnos de usuarios a creadores de artefactos computacionales (Martin et al., 2020).

3. Descripción de actividades usadas para el desarrollo del pensamiento computacional

En este capítulo se describen las actividades usadas para el desarrollo del pensamiento computacional en estudiantes de grado décimo basadas en actividades conectadas, desconectadas y en el modelo usa-modifica-crea. Para el diseño de los tres tipos de actividades se tuvo en cuenta los conceptos computacionales abordados en el test de Román-González et al. (2015) y el marco conceptual propuesto por Brennan y Resnick (2012), así como el desarrollo de habilidades enfocadas a la resolución de problemas.

Para las actividades conectadas y desconectadas se seleccionó la plataforma *Code.org* gracias a que es una buena opción para apoyar a los alumnos en los fundamentos de programación (Kalelioğlu, 2015), promueve la enseñanza del pensamiento computacional de jóvenes entre edades de 9 a 18 años que no tengan experiencia previa en programación (Díaz y Molina, 2020) e incluye en sus actividades conceptos computacionales como "direcciones", "secuencias", "condicionales" y otros más complejos como "bucles" y "funciones". Por su parte, las actividades del modelo usa-modifica-crea fueron creación propia de los autores de la presente investigación.

3.1 Actividades conectadas

Previo al desarrollo de las actividades conectadas se evaluaron 28 lecciones del curso introductorio a las ciencias de la computación *Express* 2021 que ofrece *Code.org*. con el propósito de que los estudiantes se familiaricen con actividades orientadas al desarrollo de la programación a través de bloques sin tener que preocuparse por la sintaxis de un programa. Allí se seleccionaron 13 lecciones relacionadas con el conjunto de términos computacionales asociados a los lenguajes de programación y la resolución de problemas, como lo muestra la Tabla 5, posteriormente, se creó el curso llamado “Pensamiento-Computacional” en la plataforma *Code.org* para que los estudiantes tuvieran acceso a las actividades señaladas.

Tabla 5

Relación de actividades conectadas con los conceptos computacionales

				Bucles			Condicionales		
Actividad	**Lección y Tema (Curso Express, 2021)**	**Direcciones básicas**	**Repetir veces**	**Repetir hasta**	**Mientras que**	**Para**	**Simple – *if***	**Doble – *if/else***	**Funciones**
1	Lección 1: Algoritmos secuenciales	X							
2	Lección 2: Depuración	X							
3	Lección 3: Algoritmos secuenciales + depuración	X	X						
4	Lección 4: Crear arte con código: imágenes de mayor complejidad		X						
5	Lección 24: Variables: Diseños repetitivos	X	X						
6	Lección 14: Introducción a las Condicionales	X							
7	Lección 15: Condicionales (*if, if-else*)						X	X	
8	Lección 17: Bucles y condicionales (*while*)				X		X	X	
9	Lección 19: Bucles: *while*, hasta y condicionales *if / else*			X	X			X	
10	Lección 18: Bucles: hasta (*do-while*)			X					
11	Lección 26: Bucles: Para (*for*) y variables					X			
12	Lección 20: Funciones: Reconocer patrones para volver a usar	X	X						X
13	Lección 22: Funciones: Bucles y condicionales				X			X	X

Durante la intervención los estudiantes accedieron a la plataforma *Classroom* donde se organizaron cada una de las cuatro sesiones semanales del curso (ver figura 8).

Figura 8.

Grupo de Classroom para las sesiones de actividades conectadas

En la sesión 1 y 3 se introdujeron los conceptos teóricos previos y se mostraron ejemplos prácticos de los conceptos abordados, ambas sesiones se trabajaron mediante una presentación de *Genially* como lo muestra la Figura 9.

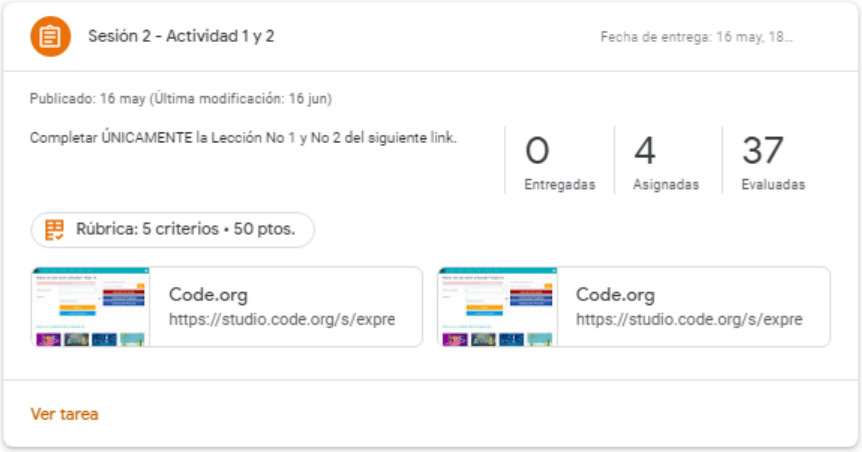
Figura 9.

Temáticas sesión 1 para el grupo conectadas.

En la sesión 2 y 4 se desarrollaron las lecciones seleccionadas del curso *Express* 2021 de la plataforma *Code.org*, publicando el link de acceso correspondiente en el *Classroom* (ver Figura 10).

Figura 10.

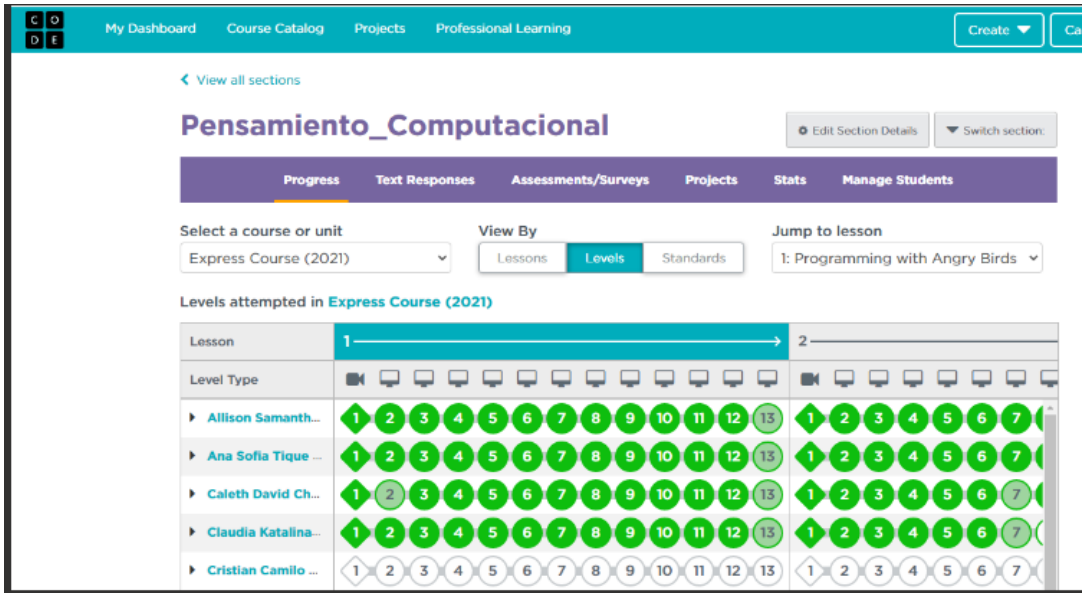
Ejercicios de Code.org propuestos en la sesión 2 – semana 1.



A través de *Code.org* se realizó el seguimiento al progreso de cada uno de los ejercicios desarrollados por los estudiantes (ver figura 11) los cuales fueron usados como insumo para el análisis cualitativo respecto a la apropiación de conceptos y prácticas del pensamiento computacional.

Figura 11.

Progreso de lecciones en la plataforma Code.org por estudiante.



Para facilitar el análisis cualitativo de las actividades conectadas se diseñaron rúbricas que permitieran evaluar la apropiación de conceptos y prácticas computacionales a las que se refiere Brennan y Resnick en su marco conceptual sobre las evidencias de aprendizaje de los estudiantes. El diseño para cada una de las sesiones empleadas en las actividades conectadas basadas en el curso *Express* 2021 de *Code.org* y las rúbricas se pueden consultar en el Anexo 1.

3.2 Actividades desconectadas

Las actividades planteadas para el grupo que trabajó actividades desconectadas se obtuvieron del curso, lecciones sin conexión, de la plataforma *Code.Org*. Allí se identificó que estas actividades también tienen en común el conjunto de términos y prácticas del pensamiento

computacional trabajado en las actividades conectadas. Se seleccionaron en total 10 actividades para ser desarrolladas por los estudiantes, las cuales fueron clasificadas, adaptadas de acuerdo con la edad y nivel de escolaridad, y posteriormente traducidas del idioma inglés al español (ver Tabla 6).

Tabla 6

Relación de actividades desconectadas con los conceptos computacionales

			Condicional		Bucle					
Actividad	**Código y nombre actividad desconectada (*Code.org*)**	**Direcciones básicas**	**simple – *if***	**Compuesto – if/else**	**Repetir veces**	**Para**	**Funciones simples**	**Depuración**	**Algoritmos**	**Variables**
1	A.3 - Mapas felices	X								
2	C.2 - Mis amigos robóticos Jr. (PARTE 1)	X							X	
3	D.4 - Programación por relevos	X						X		
4	D.2 - Programación en papel cuadriculado	X								
5	F.6 -Historias del espacio en blanco									X
6	D.12 - Condicionales con naipes		X	X						
7	B.6 - Girar y girar				X					
8	A.7 - Bucles felices				X					
9	F.13 - Diversión con bucles para					X				
10	E.11 - Composición de canciones (Parte 1)						X			

El diseño para cada una de las sesiones empleadas en las actividades desconectadas basadas en el curso *Express* 2021 de *Code.org* se pueden consultar en el Anexo 2.

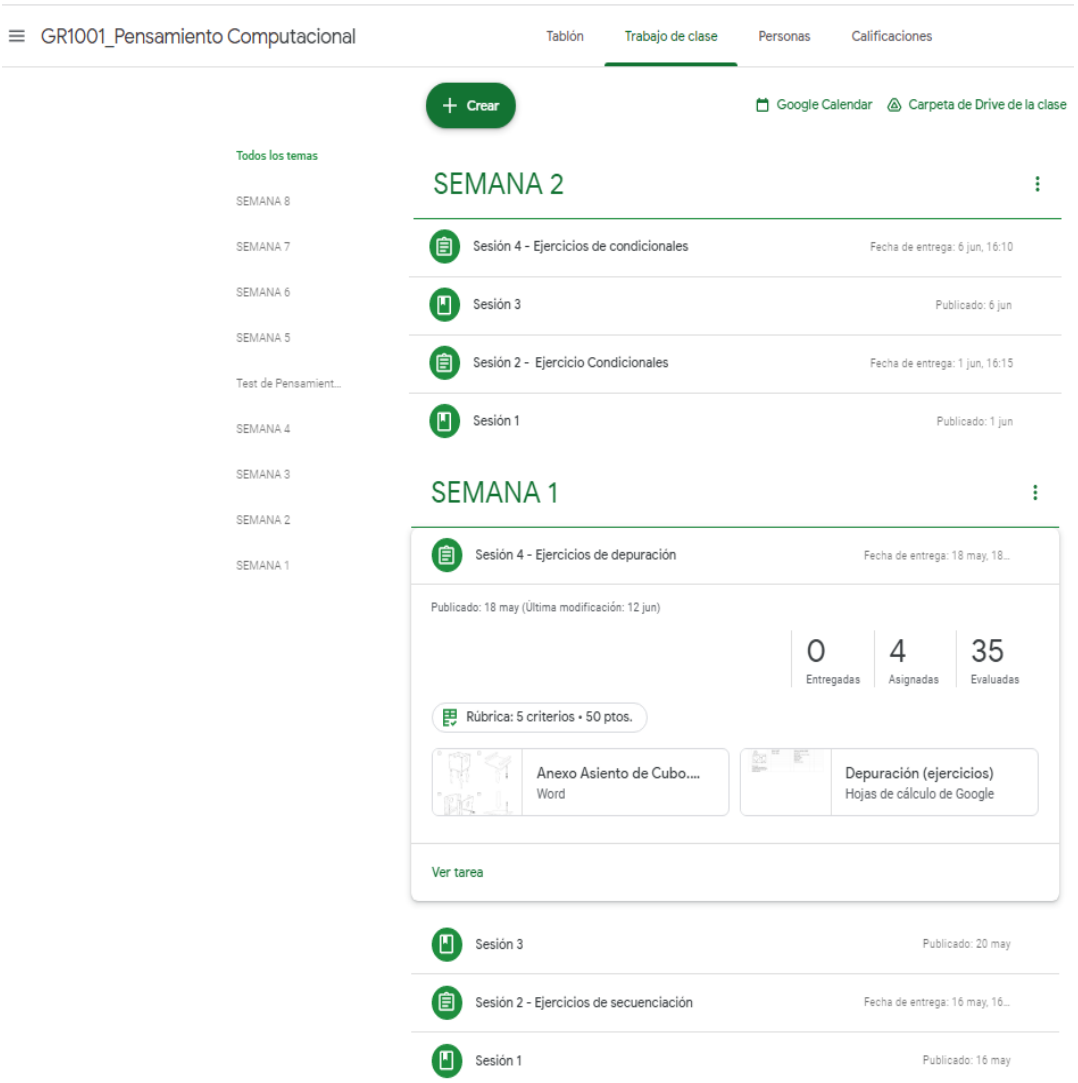
3.3 Actividades de programación usando el modelo usa-modifica-crea

Las actividades del grupo control se llevaron a cabo mediante ejercicios de creación propia diseñados para abordar los conceptos computacionales de: secuenciación, depuración, condicionales, bucles, ciclos y funciones bajo la estrategia usa-modifica-crea. La introducción teórica de los conceptos en los pasos usa-modifica se trabajó en la sesión 1 y 3 mediante una presentación de *Genially* con ejemplos y ejercicios prácticos, mientras que el paso crea de la sesión

2 y 4 se trabajó mediante archivos en hojas de cálculo y documentos de *Google* compartidos en la plataforma *Classroom* como lo muestra la Figura 12.

Figura 12.

Classroom usado para las actividades usa-modifica-crea del grupo control.



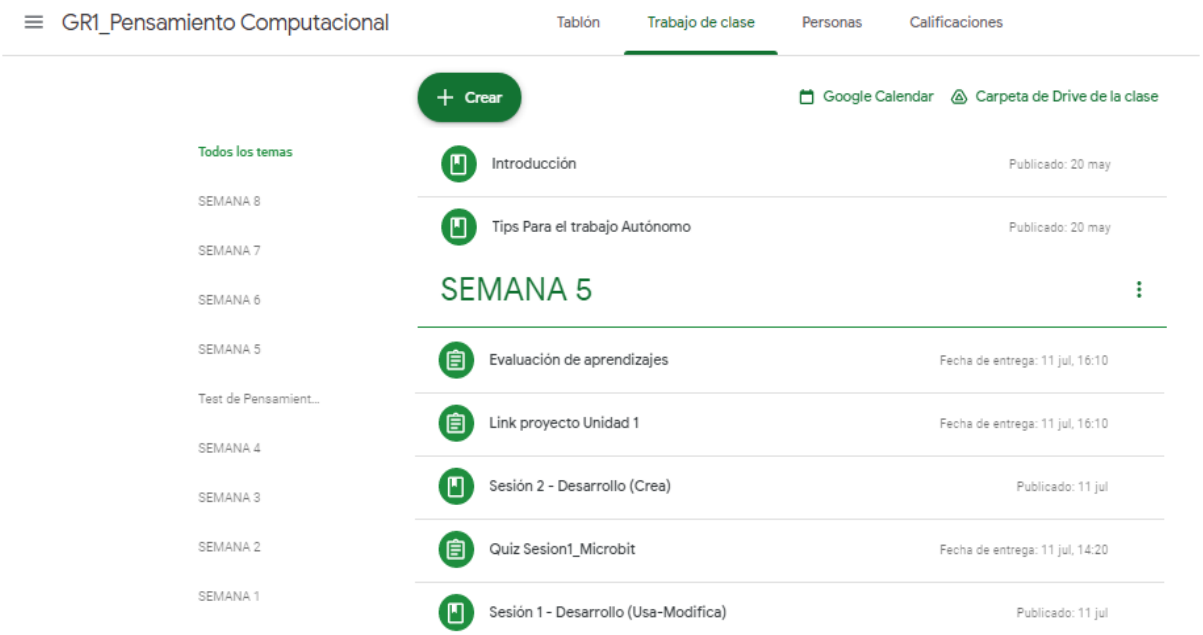
El diseño propio para cada una de las sesiones empleadas en las actividades usa-modifica-crea para el grupo control se pueden consultar en el Anexo 3.

3.4 Actividades usa-modifica-crea para la resolución de problemas.

Como estrategia para afianzar el pensamiento computacional en los tres grupos de estudiantes se usó el modelo usa-modifica-crea mediante 5 unidades desarrolladas por el *British Council* Colombia y el Ministerio de las TIC (MinTic, 2020). Estas unidades que involucran actividades de programación en la plataforma *MakeCode* se adecuaron y publicaron a través del *Classroom* como aparece en la Figura 13.

Figura 13

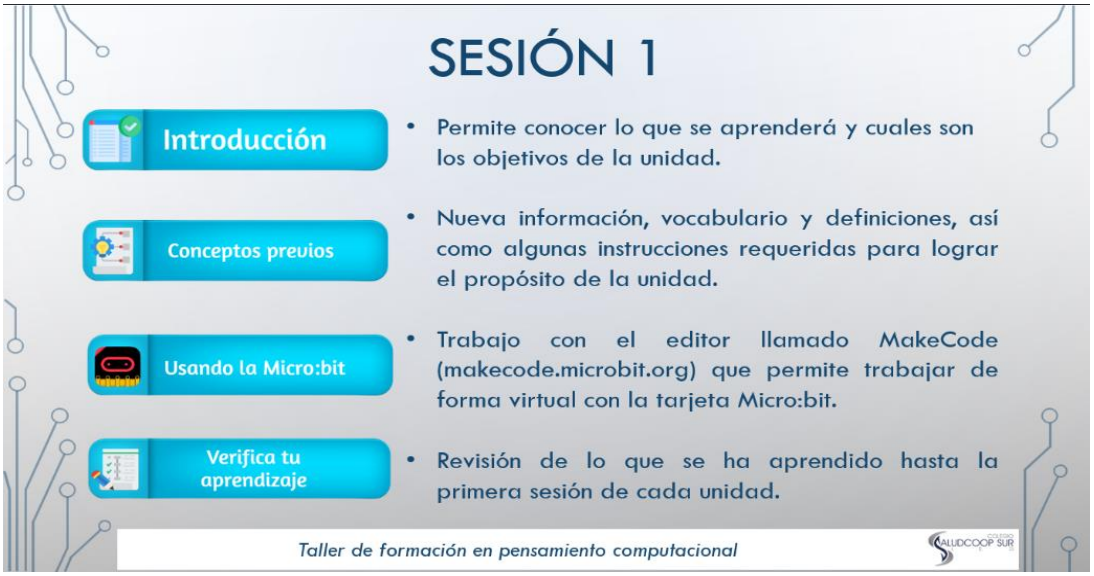
Grupo de Classroom para las unidades bajo el modelo usa-modifica-crea.



Cada unidad inició con la introducción, indicando los objetivos y conceptos computacionales previos propuestos para la unidad. Posterior a ello, se siguió con el paso usa abordado en la sección "usando la *Micro:bit*" mediante un ejemplo práctico de programación a través de la plataforma *MakeCode* que permitió realizar la simulación del ejercicio planteado. Luego se dio paso a la etapa modifica, incorporando nuevos elementos para que el estudiante modifique el modelo de la etapa anterior y verifique el aprendizaje adquirido (ver Figura 14)

Figura 14

Momentos para la sesión 1 de las unidades usa-modifica-crea.



La segunda sesión, arrancó con "creando con *Micro:bit*", donde se planteó un ejercicio en el cual cada estudiante crea y programa su propio artefacto computacional en *MakeCode*, posterior a ello debía compartir el link de la simulación como evidencia del aprendizaje. Un aspecto adicional en cada unidad es una serie de retos que afianzan los conceptos previos mediante mejoras o modificaciones adicionales al ejercicio realizado en el paso crea, este reto también se envía como evidencia a través del link de la simulación (ver Figura 15).

Figura 15

Momentos para la sesión 2 de las unidades usa-modifica-crea.

El diseño para cada una de las semanas bajo las actividades de programación usando el modelo usa-modifica-crea se pueden consultar en el Anexo 4.

4. Metodología

Este capítulo expone el tipo y diseño de la presente investigación, las variables de estudio, la descripción de la población, los instrumentos utilizados para la recolección de datos y la organización del trabajo que se realizó en las cuatro fases de la presente investigación: pre-test, intervención, post-test y resolución de problemas mediante el modelo usa-modifica-crea.

4.1. Tipo de investigación.

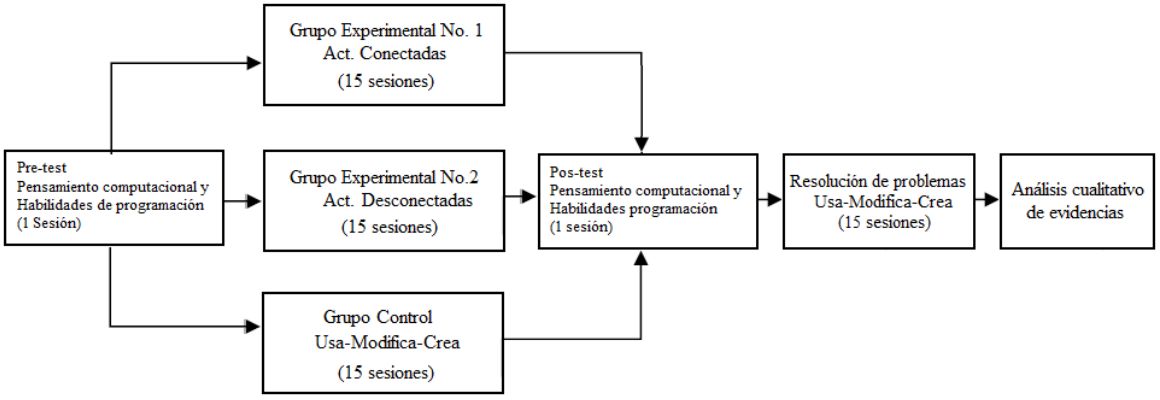
Se realizó una investigación cuasi experimental con un diseño pre-test y pos-test orientado a tres cursos previamente conformados por la institución, de allí se seleccionaron dos grupos para los tratamientos mediante actividades conectadas y desconectadas y un tercero como grupo control para garantizar la validez interna, confirmar los efectos de los grupos experimentales y comprobar las hipótesis de la investigación una vez finalizada la intervención.

4.2. Diseño de la investigación.

Para el desarrollo de habilidades de pensamiento computacional previo al trabajo de solución de problemas siguiendo el modelo usa-modifica-crea**,** se organizaron 32 sesiones en 8 semanas para cada uno de los grupos incluyendo el pre-test y pos-test de los instrumentos escogidos para el registro de datos, con el fin de diagnosticar el estado inicial de los conceptos propios del pensamiento computacional. El primer grupo experimental trabajo con actividades conectadas, el segundo grupo experimental con actividades desconectadas y el grupo control con actividades de diseño propio basadas en el modelo usa-modifica-crea (ver Figura 16).

Figura 16

Diseño de la investigación



4.2.1. Definición de variables e indicadores

Para lograr el objetivo de la investigación, se definieron las siguientes variables para dar respuesta a la pregunta de investigación y definir la validez de las hipótesis planteadas.

4.2.1.1 Variables dependientes

Para establecer la apropiación de conceptos computacionales, se consideran como variables dependientes: 1) el dominio de conocimiento de las habilidades de pensamiento computacional mediante los resultados del test de pensamiento computacional (Román-González et al., 2015), en una escala de 0 a 28 puntos y 2) las capacidades de aplicación de estructuras de control en programación a través del test de habilidades de programación (Mühling et al., 2015), en una escala de 0 a 8 puntos.

4.2.1.2 Variables independientes

La variable independiente corresponde al entrenamiento en habilidades de pensamiento computacional a través de actividades conectadas y desconectadas, esta se operacionalizó nominalmente como conectadas, desconectadas y control.

4.2.1.3 Covariables

Como covariables se empleó el logro de aprendizaje previo en las áreas de tecnología e informática y matemáticas de los dos primeros periodos del año, los cuales, de acuerdo con el sistema institucional de evaluación, se miden mediante una escala de 1.0 a 5.0. Así mismo, se consideran los resultados del pre-test de pensamiento computacional y del test de habilidades de programación.

4.2.2. Hipótesis

De acuerdo con las variables seleccionadas y los objetivos de investigación, las hipótesis planteadas fueron:

H_1 – Existen diferencias significativas en el desarrollo de habilidades del pensamiento computacional y en la apropiación de conceptos computacionales para la resolución de problemas mediante la estrategia usa-modifica-crea cuando se realiza entrenamiento previo mediante actividades conectadas y desconectadas.

H_0 - No existen diferencias significativas en el desarrollo de habilidades del pensamiento computacional ni en la apropiación de conceptos computacionales para la resolución de problemas mediante la estrategia usa-modifica-crea cuando se realiza entrenamiento previo mediante actividades conectadas y desconectadas.

4.2.3. Población y muestra

En la investigación participaron un total de 111 estudiantes de tres cursos del grado décimo del Colegio SaludCoop Sur, institución educativa pública ubicada en la localidad de Kennedy en la ciudad de Bogotá, Distrito Capital que asisten en contra jornada como parte del proyecto de educación media integral, con una intensidad de 8 horas semanales. El 60.4% de los estudiantes fueron mujeres y el 39,6% hombres, ambos con un rango de edad entre 15 y 18 años (M=15,71 SD=,939). Teniendo en cuenta la población (n=111), un nivel de confianza esperado del 95%, un

margen de error de 5% y un nivel de heterogeneidad de la muestra del 50%, calculamos que el tamaño mínimo de la muestra para la investigación debía ser de 87 estudiantes. Finalmente, la muestra de estudio estuvo integrada por un total de 96 estudiantes, 59 mujeres y 37 hombres quienes fueron autorizados por sus acudientes, completaron el entrenamiento y presentaron los test propuestos, cumpliendo con el mínimo esperado de participantes. De igual forma los grupos quedaron distribuidos así: 30 estudiantes en el grupo conectadas, 33 estudiantes en el grupo desconectadas y 33 en el grupo de control (ver Tabla 7).

Tabla 7.

Datos descriptivos población participante.

Grupo	N	Genero		Edad Promedio
		Femenino	**Masculino**	
Conectadas	30	19 (63%)	11 (37%)	15.73
Desconectadas	33	17 (52%)	16 (48%)	15.71
Control	33	23 (70%)	10 (30%)	15.73
Total	96	59	37	

4.3. Instrumentos de recolección de datos

Los dos instrumentos usados en la investigación fueron: el test de pensamiento computacional propuesto por Román-González et al. (2015a) y el test psicométrico para medir las habilidades básicas de programación de Mühling et al. (2015), los cuales fueron aplicados antes y después del entrenamiento mediante actividades conectadas y desconectadas. Las preguntas de ambos test fueron evaluadas por dos docentes con posgrado en el área de ciencias de la computación y experiencia en la implementación de planes curriculares de tecnología, los cuales realizaron aportes a la claridad en el lenguaje usado y pertinencia para el nivel de educación media. Ambos instrumentos fueron digitalizados en *Google Forms* de manera tal que las respuestas quedaron registradas en hojas de cálculo de *Google*. Las versiones finales de los test fueron

validadas mediante una prueba piloto con estudiantes del mismo grado de la jornada contraria, obteniendo una confiabilidad del instrumento buena para el test de pensamiento computacional ($\alpha = 0.707$) y alta para el test de habilidades básicas de programación ($\alpha = 0.823$) medidas con el alfa de *Cronbach*.

4.3.1. Test de Pensamiento Computacional

El test de pensamiento computacional propuesto por Román-González et al. (2015) versión 2.0, de noviembre de 2014, está conformado por 28 ítems, los cuales se correlacionan con la habilidad espacial, de razonamiento y de resolución de problemas. El objetivo del test es medir el nivel de desarrollo del pensamiento computacional en estudiantes del primer ciclo de Educación Secundaria Obligatoria de España, lo que es equivalente a los grados séptimo y octavo en el sistema educativo colombiano. Esto determina en cierto grado, la capacidad de formular y solucionar problemas apoyándose en los conceptos fundamentales de la computación, y usando la lógica-sintaxis de los lenguajes informáticos de programación, compuesta por: secuencias básicas, bucles, iteraciones, condicionales, funciones y variables.

La prueba consta de 28 ítems de elección múltiple con 4 opciones de respuesta y sólo una correcta, cada uno está diseñado y caracterizado en cinco dimensiones:

- Conceptos computacionales: direcciones básicas (ítems 1-4), bucles–'repetir veces' (ítems 5-8), bucles–'repetir hasta' (ítems 9-12), condicional simple–'*if*' (ítems 13-16), condicional compuesto–'*if/else*' (ítems 17-20), mientras que–'*while*' (ítems 21-24) y funciones simples (ítems 25-28);
- Entorno-interfaz: los ítems del se presentan en alguno de los siguientes dos entornos gráficos o interfaces: 'El laberinto' (23 ítems); 'El lienzo' (5 ítems).

- Estilo de las alternativas de respuesta: en cada ítem, las alternativas de respuesta se pueden presentar en alguno de estos dos estilos: ‘visual por flechas’ (8 ítems); ‘visual por bloques’ (20 ítems).
- Existencia o inexistencia de anidamiento: dependiendo de si la solución del ítem involucra una secuencia de comandos-órdenes con (19 ítems) o sin (9 ítems) conceptos computacionales anidados (un concepto embebido en otro concepto en un orden de jerarquía superior).
- Tarea requerida: dependiendo de cuál de las siguientes tareas cognitivas es necesaria para la resolución del ítem: ‘secuenciación’, enunciar de manera ordenada una serie de comandos-órdenes (14 ítems); ‘completamiento’, completar un conjunto incompleto de comandos previamente dado (9 ítems); ‘depuración’, depurar (‘*debug*’) un conjunto incorrecto de comandos previamente dado (5 ítems). (Román-González 2015, pp 6-7).

4.3.1.1. Tipos de preguntas

Por ejemplo, la pregunta que evalúa el concepto de “direcciones básicas” lo hace a través de las ordenes que debe realizar el ‘*Pac-Man*’ hasta llegar al fantasma donde la opción de respuesta son las flechas (ver figura 17), también plantea otro tipo de pregunta donde la opción de respuesta es a través de bloques (ver figura 18).

Figura 17

Ejemplo I concepto computacional ‘Direcciones básicas’

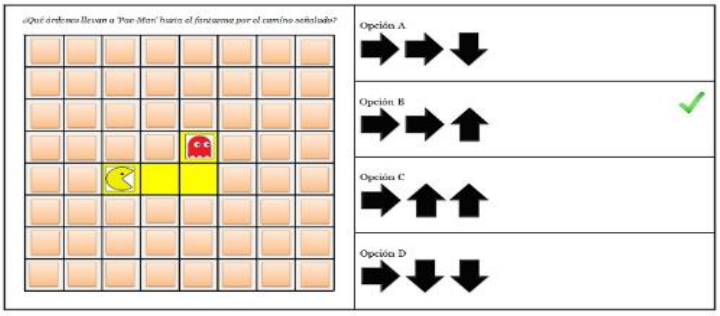
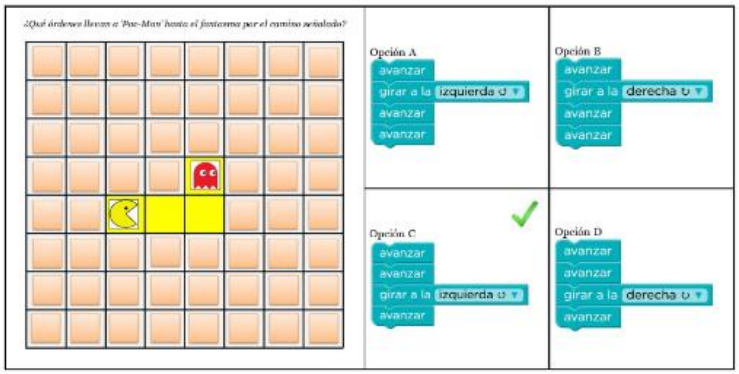


Figura 18

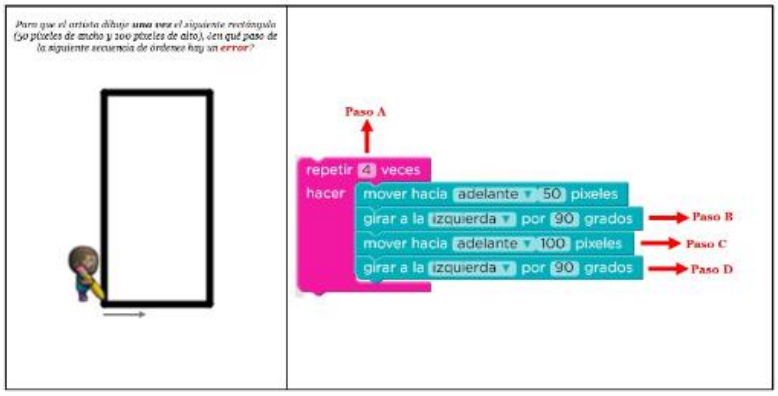
Ejemplo II concepto computacional 'Direcciones básicas'



Para el concepto de bucles 'repetir veces", lo hace a través de preguntas que implica realizar ordenes como: mover o saltar dibujando una figura, repitiendo una cantidad de veces el proceso (ver figura 19).

Figura 19

Ejemplo III concepto computacional bucles 'repetir veces'



Cada ítem del test recibe una única puntuación compuesta por tres valores, un puntaje de 2 si los alumnos únicamente seleccionan la respuesta correcta, un 1 si marcan cualquier otra respuesta incorrecta y un 0 si no responden a la pregunta. El valor final del test se obtiene con la suma del valor obtenido en cada uno de los ítems.

4.3.1.2. Confiabilidad del test

Román-González et al. (2015) indica que la fiabilidad de este instrumento, en su conjunto arroja un valor de $\alpha = 0.743$, lo cual se considera como un índice bueno. De igual forma, el análisis de consistencia mediante el alfa de Cronbach para de esta investigación, obtuvo una confiabilidad buena tanto en el pre-test ($\alpha = 0.704$) como en el pos-test ($\alpha = 0.728$), los cuales son equiparables con el resultado obtenido previamente en la prueba piloto ($\alpha = 0.707$) y otros autores como Mantilla y Negre (2021) quienes reportaron $\alpha = 0.80$.

4.3.2. Test psicométrico para medir las Habilidades Básicas de Programación

El test psicométrico de habilidades básicas de programación propuesto por Mühling et al. (2015) cubre las estructuras de control típicas que se encuentran en la programación procedimental y se basa en la teoría de la respuesta a los elementos, en particular el modelo de Rasch (1993). La prueba comprende un conjunto de elementos que mide las capacidades de los estudiantes en relación con la aplicación de estructuras de control, que se mencionan como parte destacada de los resultados de aprendizaje previstos. Entre los conceptos computacionales que evalúa este test se encuentran: secuencia de operaciones o secuencialidad, declaración condicional con y sin alternativa, bucle con un número determinado de iteraciones, bucle con condición de salida (bucle condicional) y anidación de estas estructuras para crear programas más complejos.

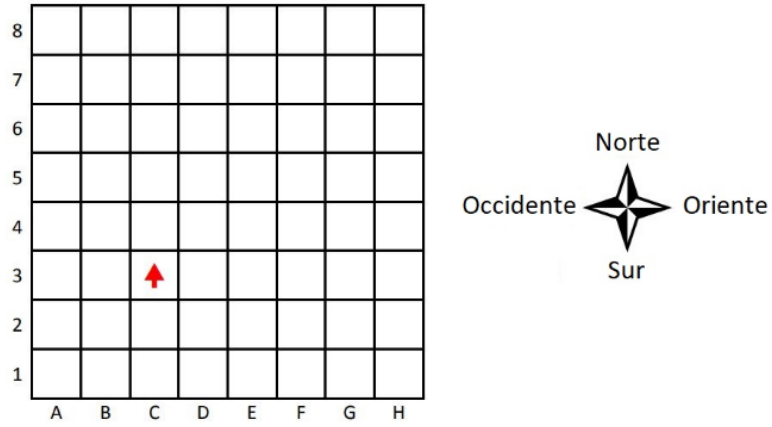
4.3.2.1. Tipos de preguntas

El formato que sigue cada una de las 8 preguntas del test de habilidades básicas de programación (Mühling et al., 2015) se basa en una cuadrícula en la que una flecha puede ser movida por un breve código de programa formado por conjunto de instrucciones sencillas. La respuesta a cada pregunta será la ubicación final de la figura después de la ejecución del programa

dado y para ello se escriben las coordenadas de la celda y la orientación de la figura. La figura 20 muestra un ejemplo de pregunta.

Figura 20

Ejemplo de pregunta test de habilidades básicas de programación.



Una figura, llamada "Alex", se encuentra actualmente en las coordenadas C3 (véase la flecha roja). La dirección de la flecha indica la dirección en la que la figura está mirando actualmente (en este caso el Norte). La figura puede ejecutar las siguientes instrucciones:

- **Paso:** La figura realiza un paso en dirección a la flecha hacia la casilla vecina.
- **Giro a la izquierda:** La figura gira 90° a la izquierda desde su dirección actual.
- **Girar a la derecha:** La figura gira 90° a la derecha desde su dirección actual.

Alex ahora ejecuta las siguientes instrucciones de forma secuencial:

```
Paso
Paso
Girar a la derecha
Paso
Paso
Girar a la izquierda
Girar a la izquierda
Paso
```

Por favor, indique en qué coordenadas se encuentra ahora Alex y hacia qué dirección mira:

Para efectos de esta investigación se realizó la traducción del idioma inglés a español de las preguntas de este test y posteriormente se evaluaron en términos de comprensión por dos docentes expertos en las áreas de inglés-español.

4.3.2.2 Puntaje del test

Cada ítem del test recibe una única puntuación dicotómica, donde se asigna un puntaje de 1 si los alumnos dan la posición y dirección correcta del objeto tras la ejecución del código en un escenario determinado. El ítem se considera correcto si y sólo si tanto la posición como la dirección son correctas, de lo contrario obtiene un puntaje de 0. La única excepción es el ítem 4 que consta de dos escenarios diferentes y dos respuestas diferentes y se puntúa correctamente si y sólo si ambas partes se responden correctamente. El valor final del test será el acumulado obtenido en cada uno de los ítems.

4.3.2.3 Confiabilidad del test

Aunque Mühling et al. (2015) no indican la fiabilidad de este instrumento, en el análisis de la consistencia del test para efectos de esta investigación se obtuvo una fiabilidad en el pre-test de $\alpha = 0.498$ y en el pos-test de $\alpha = 0.561$, lo cual se considera como moderada.

4.4. Procedimiento

La investigación se desarrolló en 4 fases mediante un ambiente de aprendizaje en la plataforma *Google Classroom* donde se organizó los contenidos y especificaciones para cada una de las sesiones. En la primera fase se realizó la presentación del proyecto, recopilación de los consentimientos, aplicación del pre-test de pensamiento computacional y habilidades básicas de programación, así como una encuesta de caracterización. En la segunda fase se aplicó el entrenamiento con actividades conectadas y desconectadas para los dos grupos experimentales, mientras que para el grupo control se organizaron actividades basadas en el modelo usa-modifica-crea (Lee et al., 2011). En la tercera fase se aplicó el post-test usando las mismas pruebas de la fase inicial. En la fase final, se realizó el trabajo de solución de problemas siguiendo el modelo

usa-modifica-crea a través de ejercicios de programación por bloques en la plataforma *MakeCode* simulando la implementación en la tarjeta *Micro:bit*.

4.4.1 Primera fase - preparación

Para la fase inicial se recopilaron y verificaron los consentimientos para el tratamiento de datos y participación en la investigación, y se asignaron los pre-test como prueba diagnóstica en el *Classroom* para realizarlos en un tiempo máximo de 50 minutos. A continuación, se realizó el test de caracterización en *Google Forms*, con el fin de determinar si los grupos eran equiparables en términos de edad, estrato, acceso a equipos de cómputo o dispositivos electrónicos, verificando de esta forma la homogeneidad entre individuos del mismo grupo previo al estudio. Esta fase finalizó con una exposición introductoria sobre el pensamiento computacional, el por qué es importante programar y aspectos relacionados con el campo laboral y los empleos del futuro donde la programación juega un papel muy importante.

4.4.2 Segunda fase - intervención

Durante cuatro semanas los dos grupos experimentales tuvieron un entrenamiento de 4 semanas utilizando como principal estrategia las actividades conectadas y desconectadas que se encuentran en la plataforma *Code.org*. El plan de trabajo que se siguió durante la intervención se dividió en dos partes, fundamentos teóricos y tipo de actividad de acuerdo con el entrenamiento (ver Tabla 8.)

Tabla 8.

Actividades semanales por cada sesión de la intervención

Sesión	Grupo Experimental 1	Grupo Experimental 2	Grupo Control
1 y 3	Fundamentos teóricos	Fundamentos teóricos	Fundamentos teóricos
2 y 4	Actividad conectada (*Code.org*)	Actividad desconectada (*Code.org unplugged*)	Actividades usa-modifica-crea

Para los fundamentos teóricos se diseñó una presentación en *Genially* que sirvió como guía didáctica en cada sesión de trabajo, esto con el fin de facilitar la apropiación de los conceptos computacionales y el desarrollo de las actividades previas mediante los pasos propuestos e instrucciones dadas por el docente. En el desarrollo de las actividades de la segunda fase para los tres grupos se tuvo en cuenta los momentos y tiempos mostrados en la Tabla 9.

Tabla 9

Momentos y tiempo para cada sesión.

Momento	Tiempo (min)
Introducción	10
Preguntas orientadoras	10
Descripción de la actividad	10
Desarrollo actividad	50
Cierre actividad	15
Entrega evidencias de aprendizaje	5
Total	100

En cuanto a los recursos usados en las actividades desconectadas, se utilizaron diferentes elementos tangibles como hojas, vasos de plástico, dados, cartas, entre otros, mientras que, para las actividades conectadas y usa-modifica-crea se asignó a cada estudiante un computador, lo que permitió trabajar de forma individual la mayor parte del tiempo.

4.4.3 Tercera fase - aplicación del pos-test

En la tercera fase se aplicó el pos-test como prueba final para los tres grupos, utilizando los mismos instrumentos de la fase inicial, los cuales permitieron realizar el análisis cuantitativo del estudio.

4.4.4. Cuarta fase- solución de problemas

Para la cuarta fase se siguió el modelo usa-modifica-crea planteado por Lee et al. (2011) mediante la plataforma *MakeCode* para desarrollar actividades de programación por bloques. La plataforma *MakeCode* y el simulador de la tarjeta electrónica *Micro:bit*, permitió evaluar la capacidad de los estudiantes para transferir sus habilidades de pensamiento computacional a nivel conceptual a diferentes tipos de problemas y situaciones particulares de su contexto, similar al enfoque propuesto por Brenann y Resnick (2012). Aquí cada estudiante interactuó con un ejemplo previo para probar y modificar su funcionamiento, y luego realizar un artefacto computacional propio que dé solución a un problema específico. En el desarrollo de las actividades de la cuarta fase se tuvo en cuenta los momentos y tiempos mostrados en la Tabla 10. En cada sesión de esta fase se realizó un seguimiento sobre el proceso de solución de los ejercicios, lo que permitió realizar una evaluación cualitativa de los resultados a través de una rúbrica.

Tabla 10

Momentos y tiempo para cada semana de la cuarta fase.

Sesión	Momento	Tiempo (min)
1 y 3	Introducción	5
	Conceptos previos	15
	Usando la *Micro:bit*	70
	Total	**100**
	Verificación de aprendizajes	10
2 y 4	Creando con *Micro:bit*	70
	Evaluación de aprendizajes	30
	Total	**100**

4.5. Técnicas de análisis de datos.

Una vez recolectados los datos se realizaron dos análisis, un análisis cuantitativo de covarianza (MANCOVA) utilizando el programa estadístico IBM SPSS (*Statistical Package for the Social Science*) versión 25 para poder determinar el efecto de las actividades conectadas y desconectadas sobre el desarrollo de las habilidades del pensamiento computacional y la

apropiación de conceptos computacionales, y un análisis cualitativo para determinar el efecto de las actividades en la resolución de problemas mediante el modelo usa-modifica-crea.

Previo a la aplicación del análisis estadístico MANCOVA se realizó la validación del test de pensamiento computacional y el test de habilidades de programación a través del cálculo del alfa de Cronbach. Posteriormente se verificó que: 1) la base de datos no contuviera datos atípicos con la distancia de Mahalanobis; 2) la distribución de datos era normal; y 3) se cumplieran los supuestos de normalidad, homogeneidad, homocedasticidad o igualdad de varianzas entre las variables de manera univariada a través de la prueba de Levene y de forma multivariada con la prueba *M de Box*. Una vez se cumplieron los supuestos, se revisó que el nivel de significancia fuera mínimo del 5% en la prueba de efectos inter-sujetos para las variables de estudio.

5. Análisis e interpretación de resultados

En este capítulo se presentan los resultados del análisis estadístico descriptivo de los resultados previos y posteriores a la intervención. En primer lugar, se exponen los resultados del análisis MANCOVA, incluyendo la verificación de supuestos de las variables dependientes y covariables de manera multivariada, así como la homocedasticidad en las variables independientes e indicando las pruebas inter-sujetos y post hoc sobre las variables objeto de estudio, sobre datos validados y verificados sin datos perdidos o atípicos. Finalmente se expone el análisis detallado de los pasos del modelo usa-modifica-crea desarrollado en cada uno de los grupos, para la aplicación de conceptos computacionales en la resolución de problemas de su contexto.

5.1 Logro previo de aprendizaje en el área de matemáticas

El logro de aprendizaje previo que muestra la Tabla 11 está sustentado bajo el promedio de los periodos académicos en el área de matemáticas antes de la intervención, permitiendo evidenciar que el grupo desconectadas cuenta con un promedio más alto (M=3.42, SD=0.69); seguido por el grupo experimental de actividades conectadas (M= 3.40, SD=0.66); finalmente el grupo de control con el promedio más bajo (M=3.06, SD=0.84). Ante esto, se puede afirmar que los estudiantes del grupo de actividades conectadas y desconectadas se pueden consideran equiparables en las habilidades lógico-matemáticas.

Tabla 11

Estadísticos descriptivos del logro previo en el área de matemáticas

Grupo	N	Media (SD)	Mínimo	Máximo	Rango
Conectadas	30	3.40 (0.66)	1.9	4.4	2.5
Desconectadas	33	3.42 (0.69)	1.8	4.9	3.1
Control	33	3.06 (0.84)	1.6	5.0	3.4
Total	96				

5.2 Logro previo de aprendizaje en el área de tecnología e informática

El logro de aprendizaje previo en el área de tecnología e informática de la Tabla 12 se calculó a partir del promedio de los periodos académicos antes de la intervención. Como se puede apreciar, el valor medio en los tres grupos conectadas (M=3.34, SD=0.39), desconectadas (M=3.45, SD=0.34) y de control (M=3.34, SD=0.50) muestra un valor muy cercano, permitiendo evidenciar que los grupos son equiparables entre sí ya que los estudiantes poseen un desempeño similar en esta área, sin embargo, la distribución de datos que muestra la desviación estándar es mayor en el grupo control, frente a los grupos experimentales.

Tabla 12

Estadísticos descriptivos del logro previo en el área de tecnología e informática

Grupo	N	Media (SD)	Mínimo	Máximo	Rango
Conectadas	30	3.34(0.39)	2.4	3.9	1.5
Desconectadas	33	3.44(0.34)	2.7	4.3	1.6
Control	33	3.44(0.50)	2.6	4.8	2.2
Total	96				

5.3 Pre-test de habilidades básicas de programación.

De acuerdo con los promedios obtenidos en el pre-test psicométrico para medir las habilidades básicas de programación (Mühling et al., 2015) de la Tabla 13 se puede observar que el grupo con actividades desconectadas cuenta con un promedio más alto y su distribución de datos es mayor respecto a los demás grupos (M=0.97, SD=1.21), por su parte el grupo de actividades conectadas tiene un promedio dos puntos por debajo del grupo con actividades desconectadas (M= 0.70. SD=1.055) y una distribución de datos agrupados en su mayoría hacia valores inferiores; finalmente se encuentra el grupo de control con el promedio más bajo de los tres grupos, el cual es más homogéneo mostrando una distribución cercana al valor medio (M=0.67, SD=0.85).

Tabla 13

Estadísticos descriptivos del pre-test habilidades básicas de programación

Grupo	*N*	*Media (SD)*	*Mínimo*	*Máximo*	*Rango*
Conectadas	30	0.70 (1.06)	0	3	3
Desconectadas	33	0.97 (1.21)	0	4	4
Control	33	0.67 (0.85)	0	3	3
Total	96				

El bajo índice de asertividad en este test se puede argumentar debido a que es la primera vez que los estudiantes se enfrentan a una prueba enfocada a medir sus habilidades de programación.

5.4. Pre-test de pensamiento computacional

En la Tabla 14 se puede observar los resultados correspondientes al puntaje obtenido por los estudiantes que participaron en el pre-test de pensamiento computacional en cada uno de los grupos. Allí el grupo control obtuvo mejor resultado, con un promedio (M=15,58, SD=4,17); le sigue el grupo de desconectadas (M=15,30, SD=3,85) y por último el grupo conectadas (M=15,13, SD=4,06), teniendo en cuenta la escala de puntuación del test que va de 0 a 28.

Tabla 14

Estadísticos descriptivos del pre-test de pensamiento computacional

Grupo	*N*	*Media (SD)*	*Mínimo*	*Máximo*	*Rango*
Conectadas	30	15.13 (4,06)	7	23	16
Desconectadas	33	15.30 (3.85)	6	21	15
Control	33	15.58 (4.17)	7	23	16
Total	96				

De acuerdo con estos resultados generales, los grupos control y desconectadas tuvieron más aciertos en las 28 preguntas frente al grupo conectadas, específicamente sobre algunos conceptos computacionales; explicación que se detalla más adelante en el análisis por cada

concepto computacional. En los puntajes obtenidos en el pre-test, se observa una mínima diferencia entre los grupos, el nivel de aptitudes en el desarrollo del pensamiento computacional alcanzó en promedio un 33% sobre los conceptos computacionales abordados. El puntaje con mayor frecuencia fue 15, que corresponde a los aciertos de 12 estudiantes, mientras que puntajes como 6 y 23 fueron los menos frecuentes con 1 y 3 estudiantes respectivamente. Por otra parte, la mayoría de los estudiantes obtuvieron un puntaje entre 9 y 19 puntos, ningún estudiante logró obtener un puntaje superior a 23.

5.5 Pre-test de pensamiento computacional por concepto

A continuación, la Tabla 15 muestra el consolidado de los estadísticos descriptivos del pre-test, para cada uno de los grupos frente a los siete conceptos computacionales de acuerdo con el test de pensamiento computacional (Román-González et al., 2015).

Tabla 15

Estadísticos descriptivos pre-test pensamiento computacional por concepto

Grupo	Concepto Computacional	Media (SD)	Mínimo	Máximo	Rango
Actividades conectadas	Direcciones básicas	0.75 (0.23)	0.00	1.00	1.00
	Bucles-'repetir veces'	0.68 (0.18)	0.50	1.00	0.50
	Bucles-'repetir hasta'	0.63 (0.24)	0.00	1.00	1.00
	Condicional Simple-'if'	0.43 (0.20)	0.00	0.75	0.75
	Condicional compuesto-'if-else'	0.51 (0.32)	0.00	1.00	1.00
	Mientras que-'While'	0.33 (0.30)	0.00	1.00	1.00
	Funciones simples	0.40 (0.22)	0.00	0.75	0.75
Actividades desconectadas	Direcciones básicas	0.73 (0.24)	0.00	1.00	1.00
	Bucles-'repetir veces'	0.65 (0.25)	0.00	1.00	1.00
	Bucles-'repetir hasta'	0.65 (0.18)	0.00	0.75	0.75
	Condicional simple-'if'	0.50 (0.23)	0.00	1.00	1.00
	Condicional compuesto-'if-else'	0.52 (0.29)	0.00	1.00	1.00
	Mientras que-'While'	0.30 (0.22)	0.00	0.75	0.75
	Funciones simples	0.47 (0.30)	0.00	1.00	1.00
	Direcciones básicas	0.78 (0.24)	0.25	1.00	0.75
	Bucles-'repetir veces'	0.71 (0.23)	0.25	1.00	0.75

Control	Bucles-'repetir hasta'	0.60 (0.28)	0.00	1.00	1.00
	Condicional simple-'if'	0.40 (0.24)	0.00	1.00	1.00
	Condicional compuesto-'if-else'	0.58 (0.29)	0.00	1.00	1.00
	Mientras que-'While'	0.33 (0.25)	0.00	1.00	1.00
	Funciones simples	0.43 (0.25)	0.00	0.75	0.75

El análisis de resultados del pre-test de pensamiento computacional se describe a continuación de acuerdo con el consolidado de medias y desviación estándar de la Tabla 15, diferenciando los ítems según el concepto computacional en forma descendente de acuerdo con el puntaje obtenido en cada grupo de preguntas.

Frente al concepto "direcciones básicas (ítems 1-4)", Los datos muestran que la distribución de los datos en los tres grupos está en un rango superior (más del 70%); el grupo de control respondió acertadamente por lo menos 3 de las cuatro preguntas del concepto evaluado con un porcentaje de aciertos del 78%, siendo el concepto computacional evaluado más desarrollado en los estudiantes mejorando el nivel de comprensión en lo que respecta al pre-test.

Para los conceptos evaluados "bucles repetir-'veces' (ítems 5-8)" y "bucles repetir-'hasta' (ítems 9-12)", alcanzaron un porcentaje de aciertos de 60% o más; sobresale el grupo control con un porcentaje de aciertos del 71% en el concepto repetir-'veces' frente al concepto repetir-'hasta' con un 65% de aciertos en el grupo desconectadas. Sin embargo, dentro del concepto de bucles, se encontró una gran diferencia en el concepto "mientras que-'*while'* (ítem 21-24)", con el menor porcentaje de aciertos en un rango de 33% a 30% en el pre-test, siendo el de menor puntaje el grupo desconectadas. Lo anterior significa que en los tres grupos se evidencia dificultad para comprender el concepto del bucle "mientras que-" y cuantas veces se repite un proceso hasta que se cumpla una condición.

Respecto al concepto computacional "condicional compuesto *'if/else'* (ítems 17-20)", los puntajes obtenidos en los tres grupos están por encima del 50% de aciertos; sobresale el grupo

control con un 58%. Por otro lado, los resultados obtenidos en los tres grupos para los conceptos "funciones simples (ítems 25-28)" y "condicional simple *'if'* (ítems 13-16)", estuvieron por debajo del 50%: "funciones simples" con un 40% para el grupo conectadas y en el concepto evaluado "condicional simple", que obtuvo un porcentaje del (39%) de aciertos por parte del grupo control.

5.6 Pos-test de habilidades básicas de programación.

Según los promedios obtenidos en el pos-test psicométrico de habilidades básicas de programación (Mühling et al., 2015) que muestran la Tabla 16 se puede observar que los tres grupos registraron un puntaje promedio más alto en comparación con el pre-test, allí sobresale el grupo de control con un puntaje promedio mayor y una distribución de datos moderada (M=1.39, SD=1.30) frente a los grupos experimentales de actividades desconectadas (M= 1.18, SD=1.33) y conectadas (M=1.07, SD=1.11). Estos resultados evidencian mayor asertividad en el test por parte de los estudiantes del grupo de control pese a las ventajas en habilidades de programación que reportaron los grupos experimentales previo a la intervención (ver Figura 21).

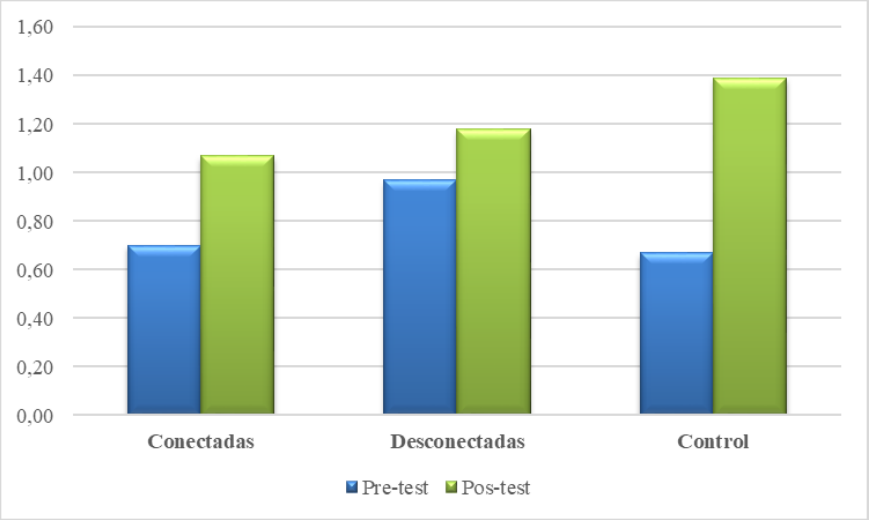
Tabla 16

Estadísticos descriptivos del pos-test de habilidades de programación

Grupo	N	Media (SD)	Mínimo	Máximo	Rango
Conectadas	30	1.07 (1.11)	0	4	4
Desconectadas	33	1.18 (1.33)	0	5	5
Control	33	1.39 (1.30)	0	5	5
Total	96				

Figura 21

Comparación pre-test y post-test habilidades básicas de programación



5.7. Pos-test de pensamiento computacional

Una vez culminada la intervención en cada uno de los grupos se realizó el pos-test, con el objetivo de evaluar los avances de los estudiantes con respecto al desarrollo del pensamiento computacional. En la Tabla 17 se puede observar los resultados correspondientes al puntaje obtenido por los estudiantes en cada uno de los grupos manteniendo la misma escala de puntuación del test que va de 1 a 28.

Tabla 17

Estadísticos descriptivos globales del pos-test de pensamiento computacional

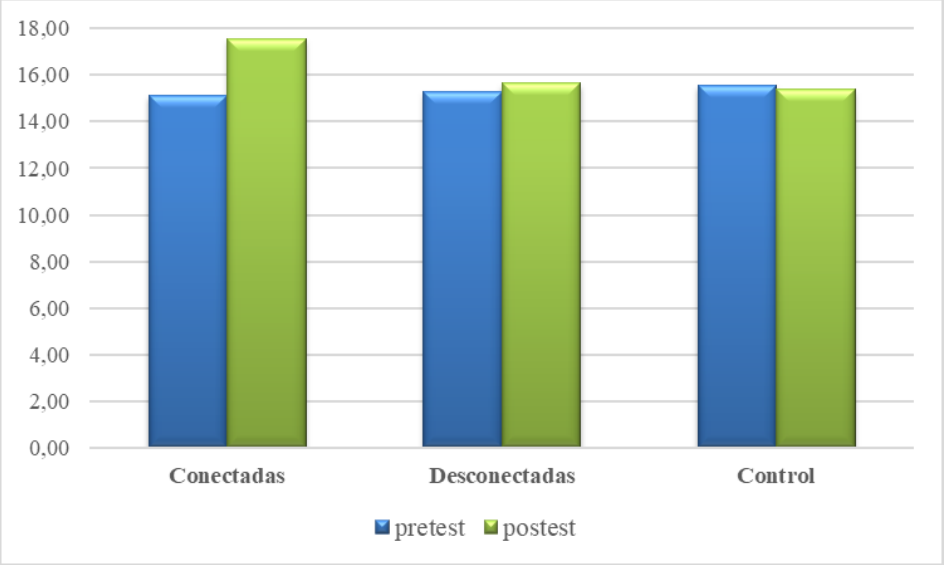
Grupo	***N***	***Media (SD)***	***Mínimo***	***Máximo***	***Rango***
Conectadas	30	17.57 (3.64)	8	23	15
Desconectadas	33	15.64 (4.00)	9	25	16
Control	33	15.39 (4.54)	6	23	17
Total	96				

El grupo conectadas presenta mejor resultado, con un promedio (M=17.57, SD=3.64); le sigue el grupo de desconectadas (M=15.64, SD=4.00) y por último el grupo control (M=15.39,

SD=4.54). De acuerdo con estos resultados, el grupo que recibió entrenamiento mediante actividades conectadas muestra una importante diferencia en la cantidad de preguntas acertadas; en el grupo desconectadas no hubo mayor variación, sin embargo, en el grupo control se evidencia una disminución en la cantidad de aciertos frente a la prueba inicial. Se encontró una similitud entre los grupos desconectadas y control con puntajes de 16 a 17 aciertos. Por otro lado, los puntajes obtenidos en el pos-test muestran una diferencia del grupo conectadas frente a los demás grupos, el nivel de aptitudes en el desarrollo del pensamiento computacional alcanzó un puntaje 17,57 puntos en las preguntas acertadas frente a un promedio de 15,51 de los grupos control y desconectadas (ver Figura 22).

Figura 22

Comparación pre-test y post-test pensamiento computacional



5.8 Pos-test de pensamiento computacional por concepto

En la Tabla 18 se muestran los estadísticos descriptivos para cada uno de los grupos frente a los siete conceptos computacionales del pos-test de pensamiento computacional.

Tabla 18

Estadísticos descriptivos pos-test de pensamiento computacional por concepto

Grupo	Concepto	Media (D)	Mínimo	Máximo	Rango
Actividades conectadas	Direcciones básicas	0.92 (0.14)	0.50	1.00	0.50
	Bucles-'repetir veces'	0.73 (0.18)	0.25	1.00	0.75
	Bucles-'repetir hasta'	0.73 (0.18)	0.25	1.00	0.75
	Condicional simple-'if'	0.58 (0.30)	0.00	1.00	1.00
	Condicional compuesto-'if-else'	0.57 (0.30)	0.00	1.00	1.00
	Mientras que-'While'	0.44 (0.22)	0.00	1.00	1.00
	Funciones simples	0.37 (0.20)	0.00	0.75	0.75
Actividades desconectadas	Direcciones básicas	0.74 (0.21)	0.25	1.00	0.75
	Bucles-'repetir veces'	0.70 (0.20)	0.50	1.00	0.50
	Bucles-'repetir hasta'	0.64 (0.25)	0.25	1.00	0.75
	Condicional simple-'if'	0.43 (0.24)	0.00	1.00	1.00
	Condicional compuesto-'if-else'	0.50 (0.32)	0.00	1.00	1.00
	Mientras que-'While'	0.40 (0.25)	0.00	1.00	1.00
	Funciones simples	0.50 (0.23)	0.00	0.75	0.75
Control	Direcciones básicas	0.83 (0.22)	0.25	1.00	0.75
	Bucles-'repetir veces'	0.72 (0.23)	0.25	1.00	0.75
	Bucles-'repetir hasta'	0.64 (0.23)	0.00	1.00	1.00
	Condicional simple-'if'	0.40 (0.26)	0.00	1.00	1.00
	Condicional compuesto-'if-else'	0.47 (0.30)	0.00	1.00	1.00
	Mientras que-'While'	0.38 (0.26)	0.00	0.75	0.75
	Funciones simples	0.36 (0.22)	0.00	0.75	0.75

El análisis de resultados del pos-test de pensamiento computacional se describe a continuación de acuerdo con el consolidado de medias y desviación estándar de la Tabla 18, diferenciando los ítems según el concepto computacional en forma descendente de acuerdo con el puntaje obtenido en cada grupo de preguntas.

En el concepto "direcciones básicas (ítems 1-4)", los resultados obtenidos mostraron que la distribución de los datos en los tres grupos está en un rango por encima de un 73% siendo el concepto con mayor puntaje en el pos-test de pensamiento computacional. Aunque los puntajes de aciertos en los tres grupos aumentaron en comparación con el pre-test, siendo el grupo de

conectadas con el porcentaje más alto (92%), se encontró diferencias relevantes entre los 3 grupos, siendo el grupo desconectadas con el puntaje más bajo (74%), como se muestra en la Figura 23. Por otra parte, el grupo desconectadas sólo obtuvo un punto por encima de la prueba inicial, manteniendo el mismo nivel con respecto a los aciertos en las preguntas del pre-test.

Para los conceptos evaluados *"*bucles repetir-'veces' (ítems 5-8*)"* y "bucles repetir-'Hasta' (ítems 9-12)", el grupo conectadas obtuvo el puntaje de aciertos más alto (73%) en los dos conceptos respecto a la prueba inicial. Por su parte, el resultado del grupo control y desconectadas para el concepto "bucles repetir-'veces'", estuvo por encima del 70% con respecto al pre-test, sin embargo, en los resultados del concepto "bucles repetir-hasta', se encontró una diferencia por debajo del puntaje obtenido en la prueba inicial por parte del grupo desconectadas. Por otro lado, el concepto de bucles, frente al pre-test del concepto "mientras que -*'while'* (ítem 21-24)*"*, el porcentaje de aciertos alcanzó un 44% para el grupo conectadas, mientras que el menor puntaje fue para el grupo control con un porcentaje de aciertos del 38%. Lo anterior indica que los grupos conectadas y control mejoraron en comparación con el pre-test, caso contrario, el grupo desconectadas, que disminuyo en la cantidad de aciertos en las preguntas del concepto computacional evaluado.

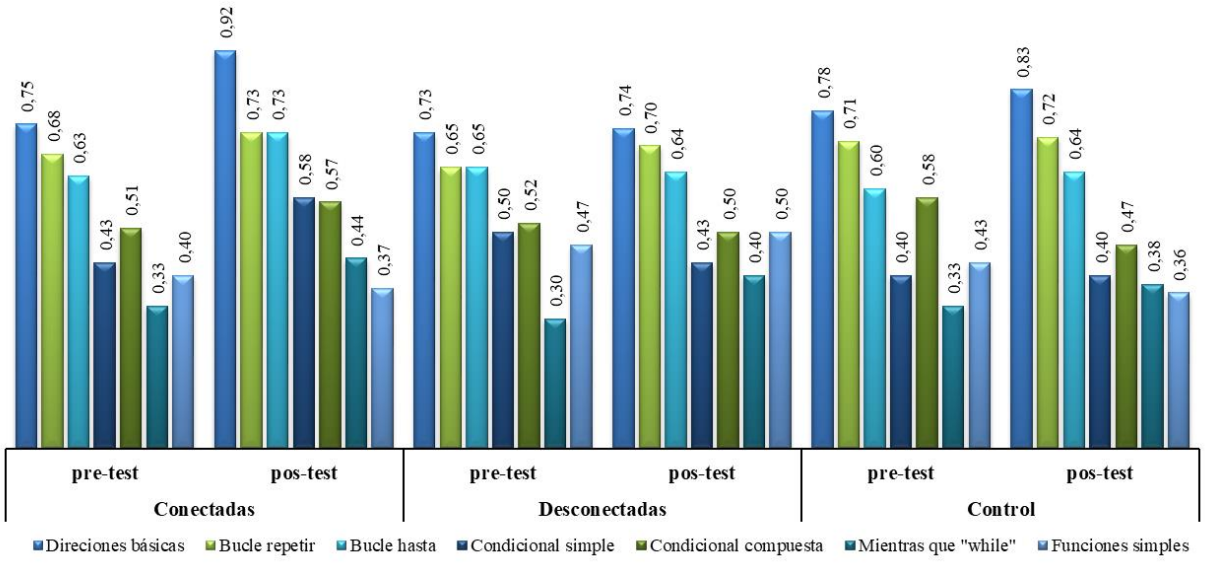
Dentro del concepto de condicionales, se encontró un porcentaje de aciertos en un rango entre 40% y 58%, siendo el grupo conectadas con el valor más alto en los dos conceptos condicionales: simple y compuesto, con respecto al pre-test. Los resultados en el concepto evaluado "condicional simple *'if'* (ítems 13-16)" el grupo control obtiene el puntaje de aciertos más bajo con 40%. Aunque en el concepto "condicional compuesto-'*if/else*'(ítems 17-20)", el grupo conectadas obtuvo el 57% de aciertos frente al pre-test, en los grupos control y desconectadas el porcentaje de aciertos estuvo por debajo de los resultados de la prueba inicial con

47 y 49% respectivamente. Por lo anterior, estos grupos muestran una disminución importante, en lo que respecta a la cantidad de aciertos en las preguntas sobre el concepto computacional evaluado. Estos resultados se muestran en la Figura 23.

Por último, el resultado obtenido en el concepto computacional “funciones simples (ítems 25-28)”, obtuvo el porcentaje de aciertos más bajo del post-test con un 36%. Aquí el grupo desconectadas mostro una mejoría obteniendo 50% frente el pre-test, sin embargo, los grupos control y conectadas obtuvieron un porcentaje por debajo de los resultados de la prueba inicial, siendo el grupo control con el puntaje más bajo.

Figura 23

Comparación pre-test y pos-test por concepto computacional



5.9 Análisis MANCOVA.

En el análisis MANCOVA se incluye la variable independiente grupo con 3 valores: conectadas, desconectadas y control; las variables dependientes: pos-test psicométrico de habilidades básicas de programación (Mühling et al., 2015) y los siete (7) conceptos del pos-test de pensamiento computacional (Román-González et al., 2015); de igual forma las covariables: logro previo y logro posterior en las áreas de matemáticas y tecnología e informática. Los descriptivos de estas variables se muestran en la Tabla 19.

Tabla 19

Estadísticos descriptivos prueba MANCOVA

Estadísticos descriptivos				
	Grupo	Media	Std Desv. Desviación	N
Post pensamiento computacional	Conectadas	17.567	3.645	30
	Desconectadas	15.636	3.999	33
	Control	15.394	4.541	33
	Total	16.156	4.161	96
Post programación	Conectadas	1.07	1.112	30
	Desconectadas	1.18	1.334	33
	Control	1.39	1.298	33
	Total	1.22	1.250	96
Post direcciones básicas	Conectadas	0.9167	0.13667	30
	Desconectadas	0.7424	0.21181	33
	Control	0.8258	0.22084	33
	Total	0.8255	0.20558	96
Post bucle repetir	Conectadas	0.7333	0.18492	30
	Desconectadas	0.6970	0.20499	33
	Control	0.7197	0.23182	33
	Total	0.7161	0.20717	96
Post bucle hasta	Conectadas	0.7333	0.18492	30
	Desconectadas	0.6364	0.25071	33
	Control	0.6439	0.22561	33
	Total	0.6693	0.22506	96
Post condicional Simple	Conectadas	0.5833	0.29605	30
	Desconectadas	0.4318	0.23612	33
	Control	0.4015	0.25723	33

	Total	0.4688	0.27205	96
Post condicional Compuesto	Conectadas	0.5667	0.30038	30
	Desconectadas	0.4924	0.31552	33
	Control	0.4697	0.29153	33
	Total	0.5078	0.30230	96
Post mientras que	Conectadas	0.4417	0.22441	30
	Desconectadas	0.3939	0.25024	33
	Control	0.3788	0.25861	33
	Total	0.4036	0.24433	96
Post función simple	Conectadas	0.3667	0.19402	30
	Desconectadas	0.5000	0.22535	33
	Control	0.3636	0.21732	33
	Total	0.4115	0.22058	96

Luego se verifica la existencia de datos atípicos de manera multivariada mediante la distancia de Mahalanobis, teniendo en cuenta que $\rho>0.01$. de tal forma que la base de datos no presente datos atípicos multivariados. En cuanto al supuesto de normalidad sobre la variable dependiente se usó como criterio el valor de simetría y curtosis mostrado en la Tabla 20, debido a que este valor se encuentra entre el rango de ±1.5, las variables dependientes siguen una distribución normal, según Tabachnick & Fidell (2013).

Tabla 20

Valores de simetría y curtosis de las covariables

Descriptivos			
		Asimetría	Curtosis
Post Pensamiento computacional	Conectadas	-0.546	-0.138
	Desconectadas	0.422	-0.575
	Control	0.30	-1.066
Post programación	Conectadas	0.828	0.069
	Desconectadas	1.077	0.736
	Control	0.659	0.045
Post direcciones básicas	Conectadas	-1.407	1.201
	Desconectadas	-0.925	0.855
	Control	-1.236	0.970
Post bucle repetir	Conectadas	-0.440	0.388
	Desconectadas	0.423	-1.379
	Control	-0.248	-0.922

Post bucle hasta	Conectadas	-0.440	0.388
	Desconectadas	0.065	-1.018
	Control	-0.919	1.035
Post condicional simple	Conectadas	-0.307	-0.773
	Desconectadas	0.119	-0.016
	Control	0.338	-0.475
Post condicional compuesto	Conectadas	-0.172	-0.924
	Desconectadas	0.358	-0.906
	Control	0.123	-0.714
Post mientras que	Conectadas	0.191	0.372
	Desconectadas	0.177	-0.173
	Control	-0.133	-1.088
Post funciones simples	Conectadas	-0.119	-0.232
	Desconectadas	-0.545	-0.448
	Control	0.148	-0.511

En cuanto al supuesto de homocedasticidad o igualdad de varianzas de forma multivariada el valor de la prueba M de Box registro un valor de ρ=0.769 (>0.05) indicando que las varianzas entre todos los grupos del estudio son similares. Como el estudio sigue un diseño cuasi experimental se revisó la homogeneidad de los hiperplanos de regresión mediante el valor de lambda de Wilks. En las interacciones de la variable independiente grupo con las covariables que muestra la Tabla 21 se observa que ρ>0.05, lo que efectivamente indica que se cumple el supuesto de homogeneidad de las pendientes de regresión y que las covariables seleccionadas son relevantes para el estudio, especialmente en este caso donde los grupos no son homogéneos.

Tabla 21

Lambda de Wilks interacción entre la variable independiente y las covariables.

Pruebas multivariante					
		Lambda de Wilks			
Efecto	Valor	F	gl de hipótesis	gl de error	Sig.
Grupo * Pre-test pensamiento computacional	0.817	1,464[b]	20	108	0.183
Grupo * Pre-test programación	0.785	,695[b]	20	108	0.823
Grupo * Direcciones básicas	0.781	,711[b]	20	108	0.807
Grupo * Bucle repetir	0.791	,672[b]	20	108	0.846

Grupo * Bucle hasta	0.853	,445[b]	20	108	0.980
Grupo * Condicional simple	0.778	,723[b]	20	108	0.795
Grupo * Condicional compuesto	0.689	1.105[b]	20	108	0.356
Grupo * Mientras que	0.842	,484[b]	20	108	0.968
Grupo * Funciones simples	0.735	,897[b]	20	108	0.591

b. Estadístico exacto

Finalmente, se verificó que se cumple el principio de homocedasticidad o igualdad de varianza de manera univariada ($\rho>0.05$) según la prueba de Levene que muestra la Tabla 22.

Tabla 22

Prueba de Homocedasticidad de Levene

Prueba de igualdad de Levene de varianzas de error [a]				
	F	gl1	gl2	Sig.
Post pensamiento computacional	0.681	2	93	0.509
Post programación	0.326	2	93	0.723
Post direcciones básicas	0.851	2	93	0.430
Post bucle repetir	0.378	2	93	0.686
Post bucle hasta	1.733	2	93	0.182
Post condicional Simple	0.563	2	93	0.572
Post condicional compuesto	1.905	2	93	0.155
Post mientras que	0.682	2	93	0.508
Post funciones simples	0.132	2	93	0.877

Prueba la hipótesis nula de que la varianza de error de la variable dependiente es igual entre grupos.
a. Diseño: Intersección + Pretest programación + Pretest pensamiento computacional + Direcciones básicas + Bucle repetir + Bucle hasta + Condicional simple + Condicional compuesto + Mientras que + Funciones simples + GRUPO

Una vez se confirma que se cumplen los supuestos descritos anteriormente, se procede a realizar el análisis MANCOVA. Teniendo en cuenta que la variable independiente posee tres valores, se realizó el análisis Post-Hoc usando el ajuste de intervalo de confianza de Bonferroni, de tal forma que se pueda establecer sobre cual grupo se presentó un mayor impacto gracias al entrenamiento. Al revisar los resultados de la prueba multivariante mostrados en la Tabla 23 para

la variable independiente (Wilks ‘Λ=0.464, F (20.148) =3.463, ρ<0.01. parcial η²=0.319) se confirma que existen diferencias significativas entre las variables independientes.

Tabla 23

Resúmenes resultados MANCOVA de forma multivariada.

Pruebas multivariante[a]

Efecto	Valor	F	gl de hipótesis	gl de error	Sig.	Eta parcial al cuadrado
Intersección	0.641	4.148[b]	10	74	0.000	0.359
Pre-test pensamiento computacional	0.817	1.464[b]	10	74	0.165	0.183
Pre-test programación	0.737	2.644[b]	10	74	0.008	0.263
Direcciones básicas	0.790	1.963[b]	10	74	0.050	0.210
Bucle repetir	0.814	1.686[b]	10	74	0.100	0.186
Bucle hasta	0.927	0.579[b]	10	74	0.826	0.073
Condicional simple	0.758	2.366[b]	10	74	0.017	0.242
Condicional compuesto	0.925	0.603[b]	10	74	0.807	0.075
Mientras que	0.852	1.289[b]	10	74	0.253	0.148
Funciones simples	0.709	3.038[b]	10	74	0.003	0.291
GRUPO	**0.464**	**3.463[b]**	**20**	**148**	**0.000**	**0.319**

a. Diseño: Intersección + Logro Previo MAT + Logro Previo TI + Pre-test pensamiento computacional + Pre-test programación + Direcciones básicas + Bucle repetir + Bucle hasta + Condicional simple + Condicional compuesto + Mientras que + Funciones simples + GRUPO, b. Estadístico exacto.

Los resultados de la Tabla 24 indican que se presentaron diferencias significativas en los conceptos computacionales de: direcciones básicas [F (2.83) =6.624, ρ=0.002. parcial η² =0.138] y condicional simple [F (2.83) =3.808, ρ=0.026, parcial η² =0.084] del test de pensamiento computacional (Román-González, 2015).

Por otro lado, no fue posible establecer diferencias significativas en el test psicométrico de habilidades básicas de programación [F(2.83) =1.280. ρ=0.283, parcial η² =0.030] ni en los conceptos computacionales de: bucle repetir [F (2.83) =1.020. ρ=0.365, parcial η² =0.024], bucle hasta [F (2.83) =2.299, ρ=0.107, parcial η² =0.052], condicional compuesto [F (2.83) =0.963, ρ=0.386, parcial η² =0.023], bucle mientras [F (2.83) =1.104, ρ=0.336, parcial η² =0.026] y funciones simples [F (2.83)=2.289, ρ=0.108, parcial η² =0.052].

Tabla 24

Resumen MANCOVA de forma univariada.

Pruebas de efectos inter-sujetos							
Origen		Tipo III de suma de cuadrados	gl	Media cuadrática	F	Sig.	Eta parcial al cuadrado
Modelo corregido	Post Pensamiento computacional	732.045	13	56.311	5.060	0.000	0.445
	Post Habilidades programación	77.370[c]	12	6.447	7.533	0.000	0.521
	Post Direcciones Básicas	1.534[d]	12	0.128	4.277	0.000	0.382
	Post Bucle repetir	1.379[e]	12	0.115	3.534	0.000	0.338
	Post Bucle hasta	1.239[f]	12	0.103	2.399	0.010	0.257
	Post Condicional simple	2.092[g]	12	0.174	2.929	0.002	0.297
	Post Condicional compuesto	1.648[h]	12	0.137	1.621	0.101	0.190
	Post Mientras que	1.700[i]	12	0.142	2.962	0.002	0.300
	Post Funciones simples	1.420[j]	12	0.118	3.066	0.001	0.307
Intersección	Post Pensamiento computacional	18.781	1	18.781	1.687	0.198	0.020
	Post Programación	3.654	1	3.654	4.269	0.042	0.049
	Post Direcciones Básicas	0.275	1	0.275	9.216	0.003	0.100
	Post Bucle repetir	0.008	1	0.008	0.261	0.611	0.003
	Post Bucle hasta	0.016	1	0.016	0.382	0.538	0.005
	Post Condicional simple	0.021	1	0.021	0.353	0.554	0.004
	Post Condicional compuesto	0.000	1	0.000	0.000	0.993	0.000
	Post Mientras que	0.008	1	0.008	0.173	0.678	0.002
	Post Funciones simples	0.004	1	0.004	0.113	0.737	0.001
GRUPO	Post Pensamiento computacional	97.091	2	48.546	**4.362**	**0.016**	**0.096**
	Post Programación	2.191	2	1.095	1.280	0.283	0.030
	Post Direcciones Básicas	0.396	2	0.198	**6.624**	**0.002**	**0.138**
	Post Bucle repetir	0.066	2	0.033	1.020	0.365	0.024
	Post Bucle hasta	0.198	2	0.099	2.299	0.107	0.052
	Post Condicional Simple	0.453	2	0.227	**3.808**	**0.026**	**0.084**
	Post Condicional compuesto	0.163	2	0.082	0.963	0.386	0.023
	Post Mientras que	0.106	2	0.053	1.104	0.336	0.026
	Post Funciones simples	0.177	2	0.088	2.289	0.108	0.052
Error	Post Pensamiento computacional	912.612	83	11.129			
	Post Programación	71.036	83	0.856			
	Post Direcciones básicas	2.481	83	0.030			
	Post Bucle repetir	2.699	83	0.033			
	Post Bucle hasta	3.573	83	0.043			
	Post Condicional Simple	4.940	83	0.060			

	Post Condicional compuesto	7.033	83	0.085
	Post Mientras que	3.971	83	0.048
	Post Funciones simples	3.203	83	0.039
Total	Post Programación	291.000	96	
	Post Pensamiento computacional	26703.000	96	
	Post Direcciones básicas	69.438	96	
	Post Bucle repetir	53.313	96	
	Post Bucle hasta	47.813	96	
	Post Condicional Simple	28.125	96	
	Post Condicional compuesto	33.438	96	
	Post Mientras que	21.313	96	
	Post Funciones simples	20.875	96	
Total corregido	Post Pensamiento computacional	1644.656	95	
	Post Programación	148.406	95	
	Post Direcciones básicas	4.015	95	
	Post Bucle repetir	4.077	95	
	Post Bucle hasta	4.812	95	
	Post Condicional Simple	7.031	95	
	Post Condicional compuesto	8.682	95	
	Post Mientras que	5.671	95	
	Post Funciones simples	4.622	95	

a. R al cuadrado = ,632 (R al cuadrado ajustada = ,579), b. R al cuadrado = ,613 (R al cuadrado ajustada = ,557)

c. R al cuadrado = ,521 (R al cuadrado ajustada = ,452), d. R al cuadrado = ,382 (R al cuadrado ajustada = ,293)

e. R al cuadrado = ,338 (R al cuadrado ajustada = ,242), f. R al cuadrado = ,257 (R al cuadrado ajustada = ,150)

Una vez se determinó la existencia de diferencias entre los grupos de estudio, se realizó la comparación por parejas para determinar la interacción entre los factores de la variable Grupo. Los resultados de la Tabla 25 son consistentes con los estadísticos descriptivos y las diferencias solo se dan entre los grupos desconectadas y conectadas para las variables dependientes: logro posterior en tecnología e informática, direcciones básicas y condicionales simples, debido al entrenamiento recibido durante la intervención.

Tabla 25

Efectos entre factores de la variable independiente.

Comparaciones por parejas							
						95% de intervalo de confianza para diferencia [b]	
Variable dependiente			Diferencia de medias (I-J)	Desv. Error	Sig.[b]	Límite inferior	Límite superior
Post Pensamiento computacional	**Conectadas**	Control	2.050	0.921	0.086	-0.202	4.302
		Desconectadas	**2.450***	0.887	0.021	0.283	4.617
	Desconectadas	Control	-0.400	0.929	1.000	-2.670	1.870
		Conectadas	**-2.450***	0.887	0.021	-4.617	-0.283
	Control	Desconectadas	0.400	0.929	1.000	-1.870	2.670
		Conectadas	-2.050	0.921	0.086	-4.302	0.202
Post Programación	Conectadas	Control	-0.338	0.255	0.568	-0.962	0.286
		Desconectadas	0.037	0.244	1.000	-0.559	0.633
	Desconectadas	Control	-0.375	0.255	0.433	-0.997	0.247
		Conectadas	-0.037	0.244	1.000	-0.633	0.559
	Control	Desconectadas	0.375	0.255	0.433	-0.247	0.997
		Conectadas	0.338	0.255	0.568	-0.286	0.962
Post Direcciones básicas	**Conectadas**	Control	0.103	0.048	0.100	-0.013	0.220
		Desconectadas	**,164***	0.046	0.002	0.053	0.276
	Desconectadas	Control	-0.061	0.048	0.608	-0.177	0.055
		Conectadas	**-,164***	0.046	0.002	-0.276	-0.053
	Control	Desconectadas	0.061	0.048	0.608	-0.055	0.177
		Conectadas	-0.103	0.048	0.100	-0.220	0.013
Post Bucle repetir	Conectadas	Control	0.008	0.050	1.000	-0.114	0.129
		Desconectadas	0.063	0.048	0.574	-0.054	0.179
	Desconectadas	Control	-0.055	0.050	0.809	-0.176	0.066
		Conectadas	-0.063	0.048	0.574	-0.179	0.054
	Control	Desconectadas	0.055	0.050	0.809	-0.066	0.176
		Conectadas	-0.008	0.050	1.000	-0.129	0.114
Post Bucle hasta	Conectadas	Control	0.077	0.057	0.553	-0.063	0.217
		Desconectadas	0.116	0.055	0.113	-0.018	0.249
	Desconectadas	Control	-0.039	0.057	1.000	-0.178	0.101
		Conectadas	-0.116	0.055	0.113	-0.249	0.018
	Control	Desconectadas	0.039	0.057	1.000	-0.101	0.178
		Conectadas	-0.077	0.057	0.553	-0.217	0.063
Post Condicional Simple	**Conectadas**	Control	0.154	0.067	0.074	-0.010	0.319
		Desconectadas	**,158***	0.064	0.049	0.001	0.315
	Desconectadas	Control	-0.004	0.067	1.000	-0.168	0.160
		Conectadas	**-,158***	0.064	0.049	-0.315	-0.001

	Control	Desconectadas	0.004	0.067	1.000	-0.160	0.168
		Conectadas	-0.154	0.067	0.074	-0.319	0.010
Post Condicional compuesto	Conectadas	Control	0.097	0.080	0.694	-0.099	0.293
		Desconectadas	0.091	0.077	0.721	-0.097	0.278
	Desconectadas	Control	0.006	0.080	1.000	-0.190	0.202
		Conectadas	-0.091	0.077	0.721	-0.278	0.097
	Control	Desconectadas	-0.006	0.080	1.000	-0.202	0.190
		Conectadas	-0.097	0.080	0.694	-0.293	0.099
Post Mientras que	Conectadas	Control	0.089	0.060	0.434	-0.059	0.236
		Desconectadas	0.051	0.058	1.000	-0.090	0.192
	Desconectadas	Control	0.038	0.060	1.000	-0.109	0.185
		Conectadas	-0.051	0.058	1.000	-0.192	0.090
	Control	Desconectadas	-0.038	0.060	1.000	-0.185	0.109
		Conectadas	-0.089	0.060	0.434	-0.236	0.059
Post Funciones simples	Conectadas	Control	0.000	0.054	1.000	-0.132	0.133
		Desconectadas	-0.097	0.052	0.194	-0.224	0.030
	Desconectadas	Control	0.097	0.054	0.226	-0.035	0.229
		Conectadas	0.097	0.052	0.194	-0.030	0.224
	Control	Desconectadas	-0.097	0.054	0.226	-0.229	0.035
		Conectadas	0.000	0.054	1.000	-0.133	0.132

Se basa en medias marginales estimadas

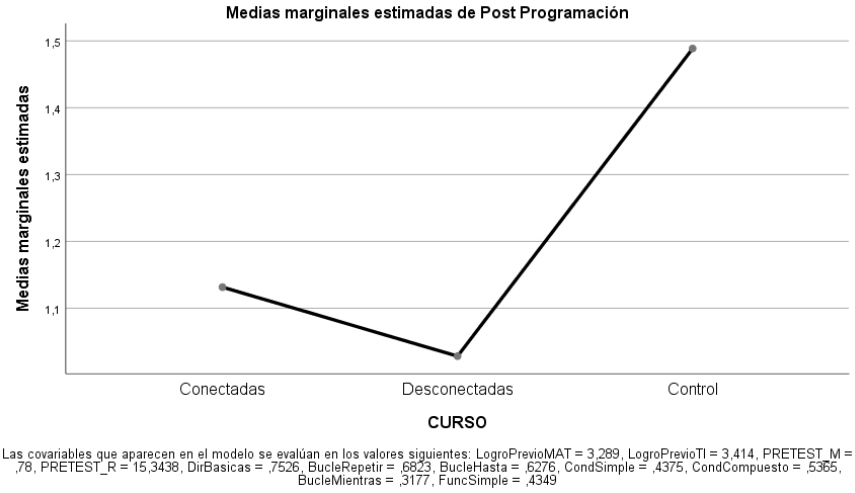
*. La diferencia de medias es significativa en el nivel ,05.

b. Ajuste para varias comparaciones: Bonferroni.

Respecto al test de habilidades de programación, las medias marginales de la Figura 26 confirman los resultados previos ya que no se evidencia una diferencia significativa entre los grupos experimentales y se observa que el grupo control tuvo mejor desempeño en ese test.

Figura 26

Medias marginales test de habilidades de programación.



En cuanto a cada uno de los ítems del test de pensamiento computacional, las medias marginales de la Figura 27 y 28 permiten corroborar que hay diferencias entre los grupos experimentales, de igual forma se destaca que los grupos experimentales alcanzaron mejores logros de aprendizaje en los conceptos de condicional compuesto y bucle “mientras” frente al grupo control debido al entrenamiento previo mediante actividades conectadas y desconectadas.

Figura 27

Medias marginales test de pensamiento computacional.

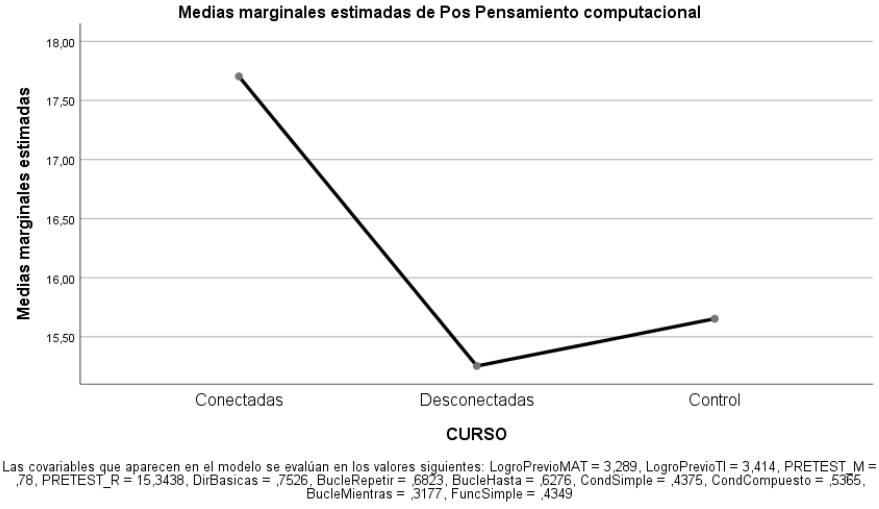
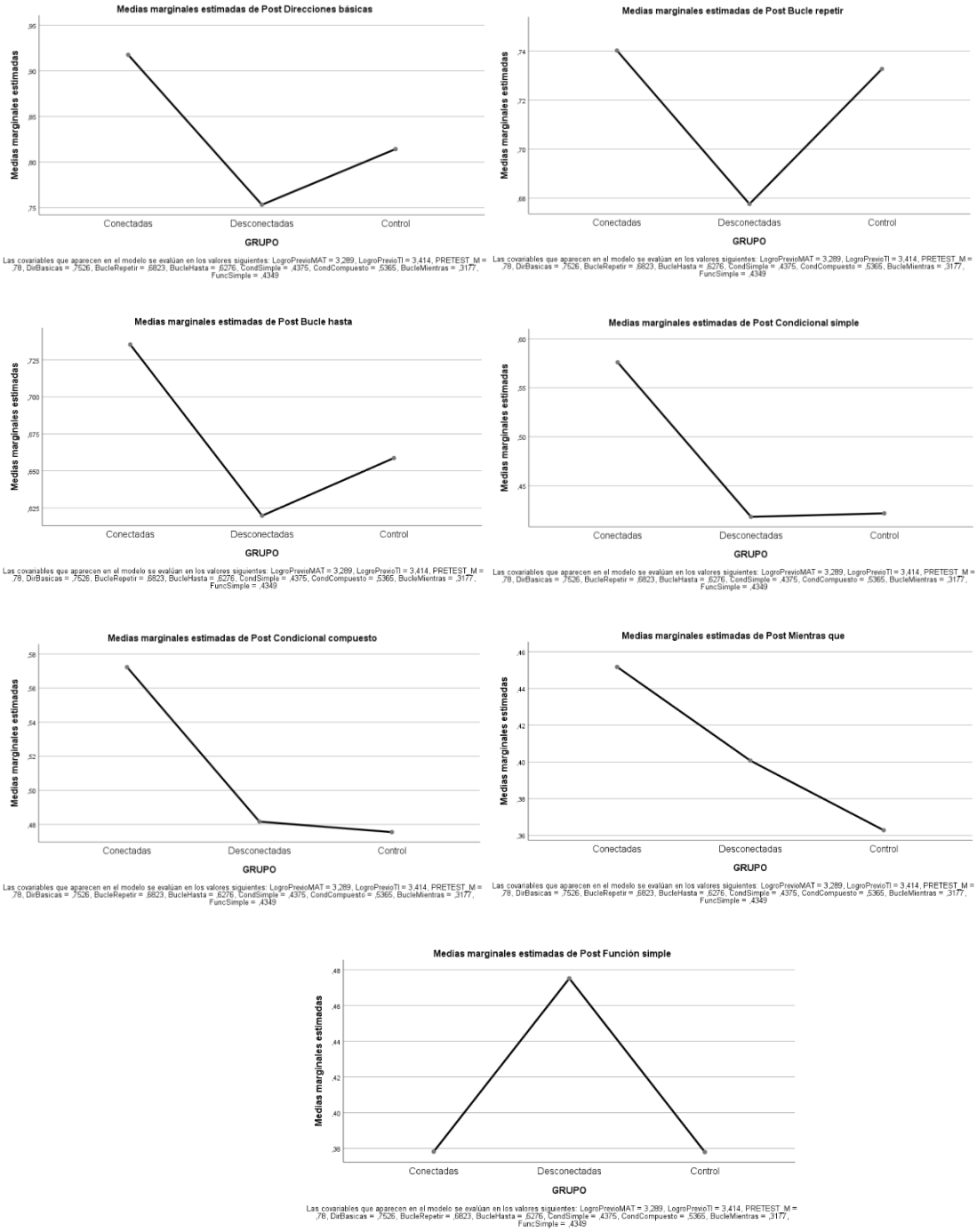


Figura 28

Medias marginales conceptos computacionales test de pensamiento computacional.



5.10 Análisis cualitativo del modelo de progresión de tres estados: usa-modifica-crea.

Con el fin de evaluar la incidencia del trabajo previo con actividades conectadas y desconectadas en la apropiación de conceptos computacionales durante la solución de problemas en *MakeCode* con *Micro:bit*, así como en la solución de problemas cuando se sigue el modelo usa-modifica-crea, se revisó cualitativamente las evidencias de aprendizaje de los estudiantes que fueron publicadas en la plataforma *Classroom* y se compararon con el resultado esperado para cada unidad. Los hallazgos más relevantes en cada una de las actividades propuestas y en cada uno de los grupos se sintetizan en este apartado, mientras que las figuras a las que hace referencia esta sección se encuentran en el Anexo 5.

5.10.1 Hallazgos unidad 1 - conceptos iniciales *MakeCode* y *Micro:bit*.

Los hallazgos en el manejo de los conceptos computacionales de secuencialidad, entrada y salida de datos y uso de variables booleanas durante la implementación del ejercicio de introducción a *MakeCode* y *Micro:bit* se muestran en la Tabla 26. Allí se observa que el grupo desconectadas y control presentaron menor dificultad en la implementación del ejercicio de la unidad 1 ya que la mayoría de los estudiantes lograron resolver el problema planteado siguiendo el modelo usa-modifica-crea y haciendo uso de la plataforma *MakeCode* (ver Figura 1, Anexo 5). También se evidenció el seguimiento de las instrucciones iniciales planteadas en el problema para la construcción del programa a través de bloques como: la distinción de los símbolos y tiempos como entradas requeridas, las reglas propuestas para el envío de mensajes y la realización de la simulación que representa la salida de información.

Tabla 26

Hallazgos unidad 1

Grupo / Concepto computacional	Conectadas	Desconectadas	Control
Secuencialidad	- No se incluye el tiempo de pausa requerido entre cada símbolo. - Incluye bloques que muestran una consonante o vocal sin codificar o que muestran números. - No se tiene en cuenta el símbolo adicional cuando se repite una letra	+ Se muestra una secuencia lógica de bloques que corresponde al uso de símbolos para cada vocal y consonante. + La programación de los tiempos van de acuerdo con las reglas propuestas en el ejercicio. - Se programan bloques de pausa después del símbolo que corresponde al inicio de la transmisión	+ Se aplican las reglas propias del ejercicio para codificar los mensajes y programar los tiempos de pausa asignados a las vocales y consonantes - Uso de otros símbolos que no incluía la tabla propuesta.
Entradas y salidas de datos	- Uso de leds adicionales a los establecidos en los símbolos de la tabla. - Asigna un tiempo de pausa menor al símbolo en cada vocal. - No se incluyen los tiempos de pausa requeridos entre cada símbolo	- Se programan tiempos diferentes a los requeridos. - No se incluyeron los tiempos requeridos después de cada bloque.	- Se usan leds adicionales que no corresponden al símbolo. - Se programa el tiempo antes del símbolo
Variables booleanas	- Representa dos símbolos al mismo tiempo en la matriz de leds.	+ Uso de bloques individuales para cada símbolo que representa vocales o consonantes.	- Se programa sobre la matriz de leds el símbolo en vez de codificarlo según la tabla

Nota: + indica un hallazgo positivo en la apropiación del concepto computacional, mientras que - indica una deficiencia o dificultad encontrada.

Los resultados también sugieren que la mayor parte de los estudiantes lograron identificar las entradas y salidas para la *Micro:bit*, abstraer los elementos requeridos para la solución, hacer buen uso de variables booleanas y comprender el seguimiento de instrucciones del ejercicio. En la solución particular que muestra el ejemplo 1 de la Figura 2, Anexo 5, se evidencia el uso del bloque

“al presionar el botón B” donde se retoma el ejercicio del paso usa-modifica como estrategia valida de solución.

Las dificultades presentadas en los grupos desconectadas y control se relacionan, principalmente, con la identificación de símbolos para representar vocales y consonantes (ver ejemplo 3, Figura 3, Anexo 5), el seguimiento de las reglas para codificar el mensaje (ver ejemplo 1, Figura 3, Anexo 5) y los tiempos asignados a cada uno de los símbolos (ver Figura 2 y ejemplo 2 Figura 3, Anexo 5). En contraste, algunos estudiantes del grupo conectadas presentaron dificultades para comprender los conceptos computacionales abordados para la unidad 1, principalmente en secuencialidad (ver ejemplo 2 Figura 4, Anexo 5), el uso de entradas booleanas (ver ejemplo 1 Figura 4 y ejemplo 3 Figura 5, Anexo 5), y el manejo de las unidades para el tiempo de pausa en milisegundos (ver ejemplo 1 Figura 5, Anexo 5). Esto apunta a una baja comprensión en la codificación y seguimiento de las reglas establecidas previamente para cada uno de los símbolos (ver Figura 5, Anexo 5).

De igual forma se descubrió que una estrategia de solución comúnmente usada por los estudiantes en esta unidad fue seguir los ejemplos previos de los dos primeros pasos del modelo usa-modifica-crea. Las variaciones respecto al resultado esperado se presentaron en el uso de los bloques “para siempre”, “al iniciar” y “al presionar el botón A”. Aunque el uso de cualquiera de esto es válido, se encontraron diferencias en la visualización de la solución, ya que en el primero se repiten continuamente los símbolos, mientras que en el segundo y tercero solo ocurre una vez. Esto demuestra divergencia en el planteamiento de soluciones particulares del estudiante y diversas formas de apropiar el uso de los bloques de programación. Por lo anterior, se puede corroborar que los tres grupos cumplieron con los objetivos propuestos para la unidad en el sentido que siguen un conjunto de pasos e instrucciones para simular la solución a un problema, identifican

las entradas y salidas para la *Micro:bit* y hacen uso adecuado de las variables booleanas. No obstante, algunos estudiantes requieren un refuerzo para lograr identificar los símbolos usados en la codificación y las unidades de tiempo (en milisegundos) tal que mejoren su asertividad en el planteamiento del ejercicio.

5.10.1.1 Hallazgos reto unidad 1

En cuanto a la implementación realizada por los estudiantes de los tres grupos a la solución del reto, sobresale la creatividad y originalidad en el diseño de símbolos que representan de forma abstracta las acciones que puede realizar un artefacto de su contexto como la lavadora de ropa. Sin embargo, al revisar las soluciones planteadas para el reto se encontró que no todas cumplen con las indicaciones dadas para resolverlo y no representan cada uno de los procesos de la lavadora. En el ejemplo 1 de la Figura 6, Anexo 5, se observa que no hay una progresión entre los símbolos y los tiempos que representan los procesos de la lavadora, en el ejemplo 2 algunos de los símbolos se asemejan más a caracteres alfanuméricos mientras que en el ejemplo 3 los tiempos usados son muy cortos para representar los minutos de cada proceso.

5.10.2 Hallazgos unidad 2 - Entrada y salida de datos.

Los hallazgos en el manejo de los conceptos computacionales de condicionales, algoritmos y variables de entrada y salida durante la implementación del ejercicio del uso de variables para la medición de temperatura usando *MakeCode* y *Micro:bit* se muestran en la Tabla 27. Los resultados señalan que el grupo conectadas y desconectadas presentaron más dificultades en la implementación del programa que el grupo control. Las dificultades principalmente están relacionadas con seleccionar el operador relacional para el rango en el cual la variable temperatura

es mayor o menor que un valor establecido y entender cómo realizar la comparación en cada condicional (ver Figura 7, Anexo 5).

Tabla 27

Hallazgos unidad 2

Grupo / Concepto computacional	Conectadas	Desconectadas	Control
Condicionales	- Carece de algunas condiciones establecidas en el ejercicio. - Usó del símbolo igual en la condicional para un valor específico. - Se dejaron algunos bloques de condicionales sin usar.	- Dificultades en la definición de las condiciones requeridas en los rangos de temperatura - Errores de compilación del programa debido al orden de ubicación de las condiciones de comparación	- Uso del símbolo "menor que" en vez de "mayor o igual que" para el rango de temperatura de 26 a 34°C. - Uso repetido del bloque "para siempre". + Uso de condicionales y bucles anidados.
Variable numérica	- Faltó incluir algunos valores del rango de temperatura considerada como normal. - No se incluyó el mensaje "t. normal" para el rango de temperatura de 26°C a 34°C.	- Los rangos de temperatura no corresponden con los valores propuestos en el ejercicio. - No corresponden los mensajes de alerta con el rango de temperatura.	- No corresponden los mensajes de "T. Normal" y "T Baja con el valor que indica la simulación. - No se tiene en cuenta que se trata de un rango y no de un valor puntual.

Nota: + indica un hallazgo positivo en la apropiación del concepto computacional, mientras que - indica una deficiencia o dificultad encontrada.

En cuanto al grupo desconectadas, se presentaron casos en los que no es posible ejecutar la simulación, esto debido a errores de compilación del programa que se presentan por la ubicación de los bloques de comparación de las condicionales (ver ejemplos 1 y 2, Figura 8, Anexo 5). Adicionalmente, los ejercicios desarrollados por los estudiantes del grupo control en la unidad 2 muestran un buen manejo y comprensión de las condicionales, sin embargo, comparten los mismos errores que los otros dos grupos respecto al uso de los operadores relacionales para los rangos en los que la variable temperatura es válida (Ver Figura 9, Anexo 5).

En este caso frente a los resultados se puede afirmar que los estudiantes de los tres grupos pudieron identificar el principal uso de las condicionales en el programa, a pesar de faltarles

precisión en la selección del símbolo adecuado para cada caso. Complementando el análisis, durante el desarrollo de esta unidad se evidencio un mayor uso del ensayo y error, demostrando como los estudiantes apropian técnicas de programación para cumplir los objetivos propuestos de la unidad y plantear una solución del problema.

5.10.2.1 Hallazgos reto unidad 2

En cuanto a la solución del reto, si bien, muchos estudiantes mantuvieron los errores del anterior ejercicio (ver ejemplo 1, Figura 10, Anexo 5) y solo añadieron el bloque de "reproducir tono", otros realizaron bien el ejercicio e incluyeron el bloque "reproducir melodía" (ver ejemplo 3, Figura 10, Anexo 5) para indicar mediante un sonido cuando la temperatura excedía el máximo permitido o cuando algún valor caía por debajo del mínimo.

5.10.3 Hallazgos unidad 3 - Condicionales y bucles.

Los hallazgos en el manejo de los conceptos computacionales de bucles, diagrama de flujo y operaciones lógicas durante la implementación del ejercicio para comunicar instrucciones utilizando la pantalla de LEDs de la *Micro:bit* mediante *MakeCode* se muestran en la Tabla 28. Una vez consolidados los resultados, se observó que los tres grupos no tuvieron mayor dificultad en la implementación del ejercicio de la unidad 3. La mayoría de los estudiantes lograron resolver el problema planteado, siguiendo el modelo usa-modifica-crea y haciendo uso de la plataforma *MakeCode*. En la construcción del programa siguiendo las instrucciones iniciales se evidenció como los estudiantes: adicionaron animaciones en la matriz de LEDs que simulan el movimiento de la bicicleta a través de un código de flechas como "ir a la izquierda", "ir a la derecha" y otras señales como "Parar"; utilizaron bloques condicionales que se encuentra en la categoría de "Lógica*"* programando operaciones lógicas que permitan decidir qué acción tomar al oprimir el botón A, B o A+B (ejemplo 1, Figura 11, Anexo 5); e identificaron los bucles con bloques como

"para siempre" o "si agitado" que permite realizar procesos o acciones repetitivas de forma indefinida (ver Figura 11 y 12, Anexo 5)

Tabla 28

Hallazgos unidad 3

Grupo / Concepto computacional	Conectadas	Desconectadas	Control
Bucles	- No se utilizó el bucle "para siempre", se hace una programación como entradas independientes para cada uno de los botones A, B y A+B.	- Se reemplaza el bucle "para siempre" por los bloques "al presionar A", "al presionar B" y "al presionar A+B",	- Se omitió el bloque "para siempre" - Se usa el bloque "al iniciar"
Diagrama de flujo	+ Se muestra una secuencia lógica de bloques para simular el movimiento hacia adelante y la programación para los botones A, B y A+B.	+ Se muestra una secuencia lógica al agregar otras animaciones a la programación del ejercicio	+ Se observa una secuencia lógica de instrucciones para el bloque "si agitado" y cada uno de los botones A, B y A+B
Operaciones lógicas	- Se usa un bloque "Condicional" con un operador lógico 'o', programando la misma acción para el botón A y B. - Se encontró una condicional en blanco para tomar una decisión, el bucle "para siempre" quedó incompleto. - Dificultad para entender la estructura "Condicional" y, por consiguiente, programar correctamente sus acciones	+ El bloque "al presionar A" se reemplaza por "si inclinación hacia la izquierda". - No se programan las instrucciones para los botones B y A+B. - Se programa una flecha hacia abajo, que no corresponde a una acción de "pare". - Uso de un bloque con el icono de corazón en reemplazo.	+ Las animaciones simulan un movimiento más adecuado para el uso de la bicicleta. - Se duplicaron instrucciones - Se programó la misma instrucción en el bloque "para siempre" y también para el botón A+B

Nota: + indica un hallazgo positivo en la apropiación del concepto computacional, mientras que - indica una deficiencia o dificultad encontrada.

Por lo anterior, se afirma que la mayoría de los estudiantes lograron comprender los conceptos computacionales abordados para esta unidad, en lo que se refiere a estructuras condicionales simples y dobles, así como como identificar y usar los tipos de bucles en la solución del problema. Las dificultades presentadas en estos grupos se relacionan, principalmente, con el seguimiento de las reglas e instrucciones iniciales y en algunos casos, el uso incorrecto del bloque

“al iniciar”, que no corresponde a un bucle y condicionales que no fueron usadas o están sin bloques (ejemplo 1 y 2, Figura 13 y ejemplo 2, Figura 15, Anexo 5). En otros casos las reglas e indicaciones dadas para el ejercicio no se tuvieron en cuenta, por lo que las imágenes usadas para las animaciones no corresponden a la acción que buscaban representar (ejemplo 3, Figura 13 y ejemplo 3, Figura 14, Anexo 5). En consecuencia, es necesario plantear ejercicios que involucren diferentes tipos de bucles frente a los usados comúnmente en el *MakeCode* como el bloque “para siempre” y “al iniciar”, con el objetivo de entender las diferencias, como estos funcionan y que los estudiantes aprendan a usarlos en la solución de un problema.

5.10.3.1 Hallazgos reto unidad 3

Con respecto al reto, se encontró que la mayoría de las soluciones no cumplieron con los requerimientos iniciales; en algunos casos se entregó el mismo ejercicio del paso “crea”, en otros, la programación de bloques no corresponde a las acciones planteadas en el ejercicio. También se realizaron parcialmente los ejercicios sin programar las instrucciones con el bloque *“*es un gesto agitado*”* que ofrece el sensor del acelerómetro codificar los cambios, movimientos, giros y dirección de la bicicleta, sin recurrir a ninguno de los botones de la *Micro:bit* (Ver Figura 16, Anexo 5).

5.10.4 Hallazgos unidad 4 - variables

Los hallazgos en el manejo de los conceptos computacionales de variables, diagrama de flujo y operaciones aritméticas básicas en la implementación del ejercicio de simulación de datos en un pluviómetro usando *MakeCode* y *Micro:bit* se indican en la Tabla 29. De acuerdo con los resultados obtenidos en los tres grupos, se puede observar que la principal dificultad a nivel general fue la programación de los comandos “establecer” y “cambiar”, propios de las variables creadas en *MakeCode* (ver Figura 17, Anexo 5). Aquí la mayoría de los estudiantes lograron resolver el

problema planteado, sin embargo, se evidenció que el resultado fue más producto de los ejercicios propuestos en el paso usa-modifica, que del paso crea, donde debían seguir las instrucciones implementadas en el diagrama de flujo y crear nuevas variables (ver ejemplo 1, Figura 19, Anexo 5).

Tabla 29

Hallazgos unidad 4

Grupo / Concepto computacional	Conectadas	Desconectadas	Control
Variables	- La variable Valor_dado no fue creada, tampoco se inicializo, ni está en los procesos de cálculo. - Se crea una variable nueva como "Cantidad_dias" que no es necesaria y tampoco se usó en los procesos. - El comando "Cambiar" se usa incorrectamente para las variables definidas en el ejercicio.	- Dificultad para programar los comandos "Establecer" y "Cambiar" para las variables definidas - Dificultad a la asignación de un número al azar a la variable "Valor_dado" - El comando "Establecer" para la variable "Valor_dado", no corresponde a los valores establecidos	- La programación del comando "Cambiar" para la variable "Valor_dado", no es correcta, se esperaba el comando "Establecer".
Diagrama de flujo	- Los bloques programados no corresponden a la secuencia lógica del Diagrama de Flujo propuesto.	- Las instrucciones no corresponden a la secuencia lógica de bloques que se plantearon en el Diagrama de Flujo.	- La secuencia lógica de bloques no corresponde a las instrucciones planteadas en el Diagrama de Flujo.
Operaciones aritméticas	+Se realiza primero la suma y luego se calcula el promedio. -No se usa la variable promedio en el cálculo.	- Dificultad para calcular la cantidad de agua lluvia acumulada. -El promedio se calcula en el ciclo.	+El promedio se calcula al final del ciclo. +Se almacena el valor del total de días y promedio en las variables correspondientes.

Nota: + indica un hallazgo positivo en la apropiación del concepto computacional, mientras que - indica una deficiencia o dificultad encontrada.

En el grupo desconectadas los resultados muestran una mejor comprensión y desarrollo de las actividades respecto a los conceptos computacionales evaluados para la unidad 4, específicamente en seguir las instrucciones propuestas en el diagrama de flujo. Esto se puede

evidenciar en la Figura 18, Anexo 5, allí se muestra una secuencia lógica de bloques que cumplen con lo que planteaba el diagrama de flujo para resolver el problema.

En el grupo conectadas se observa mayor dificultad para comprender la definición y creación de variables, uso de los comandos “establecer” y “cambiar” para una variable y seguimiento a las instrucciones planteadas en el diagrama de flujo del ejercicio (ver ejemplo 1, Figura 19, Anexo 5). Allí también se presentó dificultad en programar bloques que siguieran la secuencia lógica para calcular el promedio o cantidad acumulada de agua en un periodo de tiempo de 1825 días y que el resultado o salida fuera coherente con las condiciones planteadas, tal como lo muestra el ejemplo 2 de la Figura 19, Anexo 5.

Por su parte, la mayoría de los estudiantes del grupo control no comprendieron y desarrollaron correctamente las actividades respecto a los conceptos computacionales evaluados para la unidad 4. La Figura 20, Anexo 5, muestra un ejemplo típico de solución donde se evidencian los hallazgos mencionados en la Tabla 4. En la Figura 21, Anexo 5, se muestra evidencia que sugiere que gran parte de los estudiantes del grupo control no siguieron las indicaciones del ejercicio propuestas en el diagrama de flujo, mostrando programación con bloques diferentes cuya función y uso no se explica en ninguna actividad.

5.10.4.1 Hallazgos reto unidad 4

Con respecto al reto en esta unidad, fueron pocos estudiantes los que realizaron y enviaron la evidencia, más ninguno acertó. Un ejemplo de ello se señala en la Figura 22, Anexo 5 donde ningún estudiante se logró acercar a la solución.

5.10.5 Hallazgos unidad 5 - Funciones

Los hallazgos en el manejo de los conceptos computacionales de funciones, variables y coordenadas aplicados en la implementación del ejercicio de vuelo del *Mars helicopter* mediante

MakeCode y *Micro:bit* se muestran en la Tabla 30. Estos resultados apuntan a que, aunque los estudiantes de los tres grupos entendieron el concepto de función y crearon la función requerida para validar las coordenadas del vuelo del *Mars helicopter*, no lograron un dominio tal que les permitiera comprender en que parte del programa hacer el llamado a la función y como combinarla con el uso de condicionales, lo que afectó claramente la visualización de la posición del helicóptero durante el vuelo. En este aspecto, los errores de definición de los valores iniciales de las variables, la validación del valor máximo para cada variable que permite la matriz de leds y el incremento que debían tener a lo largo de la ejecución del programa dificultaron la implementación del ejercicio de la unidad 5.

Tabla 30

Hallazgos Unidad 5

Grupo / Concepto computacional	Conectadas	Desconectadas	Control
Funciones	+ Funciones completas de acuerdo con el paso *Usa-Modifica.* - No se realiza el llamado de la función el oprimir los botones.	- No seguir las instrucciones correctamente usando otros símbolos que no establecía el ejercicio.	-Funciones incompletas. - Uso incorrecto de los tiempos de pausa para la visualización.
Variables	- Se usa el símbolo “menor que” en vez de “menor o igual” - Solo se valida la posición en X y hasta la coordenada 3, en vez de la 4.	- Uso incorrecto de los operadores relacionales “menor o igual” y “mayor o igual” - Uso contrario de los bloques “establecer para” y “cambiar por” en el manejo de variables.	- Uso incorrecto de los operadores relacionales “menor o igual” - Uso incorrecto de los comandos “establecer” y “cambiar”.
Coordenadas	- El incremento del valor de las coordenadas X y cada vez que se presionaban los botones era superior a la unidad.	+ Existe un buen manejo del valor inicial de las coordenadas. +El incremento del valor de la coordenada es acorde con el ejercicio	+ Se incrementan y validan las coordenadas de forma correcta.

Nota: + indica un hallazgo positivo en la apropiación del concepto computacional, mientras que - indica una deficiencia o dificultad encontrada.

Tanto en los grupos conectadas se observa un buen dominio de los pasos para lograr definir una función en el programa, sin embargo, se observan errores recurrentes presentados en la unidad 1 y 2 respecto a la definición de los tiempos de pausa y el uso de símbolos “menor” e “igual que” los cuales no corresponden al ejercicio planteado (ver Figura 23, Anexo 5). Otro factor que influyó en el desarrollo de la solución de esta unidad tuvo que ver con que algunos estudiantes de este grupo no realizaron las modificaciones necesarias para validar las coordenadas cada vez que se presionan los botones, manteniendo el ejemplo original de los pasos previos usa-modifica (ver Figura 24, Anexo 5).

En el grupo desconectadas la mayoría de los estudiantes completaron los ejercicios de la unidad, mostrando una buena comprensión del ejercicio, aunque el desarrollo de la solución respecto a los otros dos grupos fue similar. A pesar de ello, en este grupo algunos estudiantes no completaron en su totalidad el ejercicio, tal como se puede evidenciar en los dos ejemplos de la Figura 24, Anexo 5, donde hay confusión en los símbolos de comparación que se deben usar, o se usan íconos básicos trabajados previamente en la unidad 1. Finalmente, en el grupo control se presentaron más casos donde los estudiantes mostraron una baja comprensión del uso de condicionales respecto a los otros dos grupos, no completaron los ejercicios de la unidad y dejaron extractos de código sin usar (ver Figura 25 y Figura 26, Anexo 5).

Las estrategias usadas en busca de la solución fueron mínimas en los tres grupos, cumpliendo parcialmente con los objetivos propuestos para la unidad respecto al uso de funciones y la verificación del programa usando casos de prueba. No obstante, algunos estudiantes mostraron esforzarse por construir una solución válida de acuerdo con el planteamiento del ejercicio dándole continuidad al ejercicio previo de los pasos usa-modifica.

5.10.5.1 Hallazgos reto unidad 5

En el caso del reto muy pocos estudiantes se atrevieron a plantear una posible solución, por lo que en esta unidad la mayoría de los estudiantes no lograron resolverlo siguiendo las instrucciones planteadas. La figura 27, Anexo 5 muestra algunos ejemplos para la solución del reto.

5.11 Contraste del test de pensamiento computacional, test de habilidades básicas de programación y modelo de progresión de tres estados.

En este apartado se expone la triangulación de los resultados cuantitativos del test de pensamiento computacional frente a los hallazgos cualitativos del modelo usa-modifica-crea en el manejo de los conceptos computacionales abordados para cada una de las unidades. Allí encontramos que existen conceptos en común como: bucles–'repetir veces', condicional simple – '*if*', condicional compuesto –'*if/else*' y funciones simples; mientras que otros conceptos se trabajaron únicamente en el entrenamiento: direcciones básicas, bucle 'repetir-hasta' y mientras 'que-*while*'; o fueron abordados exclusivamente en la fase usa-modifica-crea: entradas y salidas de datos, variables booleanas, algoritmos, variables, diagrama de flujo, operaciones lógicas, operaciones aritméticas y coordenadas. Por lo tanto, el análisis cualitativo no solo se orientó a los conceptos que convergen en el estudio, si no que se amplió a los que se potencian mediante la progresión usa-modifica-crea. Para ello indicamos en la Tabla 33 de forma consolidada si se evidencia o no una aplicación correcta del concepto para llegar a la solución propuesta o si se realizó de forma parcial, es decir, que la aplicación del concepto no fue completamente satisfactoria (ver Tabla 33).

Tabla 31

Apropiación de conceptos computacionales según los test y el usa-modifica-crea

	Test de pensamiento computacional			Usa-modifica-crea		
Concepto computacional	**Conectadas**	**Desconectadas**	**Control**	**Conectadas**	**Desconectadas**	**Control**
Direcciones básicas	0.9167	0.7424	0.8258	*No se trabajó*	*No se trabajó*	*No se trabajó*
Bucles–'repetir veces'	**0.7333**	**0.697**	0.7197	**Parcialmente**	**SI**	SI
Bucles–'repetir hasta'	0.7333	0.6364	0.6439	*No se trabajó*	*No se trabajó*	*No se trabajó*
Condicional simple –'if'*	**0.5833**	**0.4318**	**0.4015**	**SI**	**SI**	**SI**
Condicional compuesto – 'if/else'*	0.5667	0.4924	0.4697	SI	SI	SI
Mientras que– 'while'	0.4417	0.3939	0.3788	*No se trabajó*	*No se trabajó*	*No se trabajó*
Funciones simples	0.3667	**0.5**	**0.3636**	Parcialmente	**SI**	**SI**
Entradas y salidas de datos (unidad 1)	-	-	-	SI	SI	SI
Variables Booleanas (unidad 1)	-	-	-	SI	SI	SI
Algoritmos (unidad 2)	-	-	-	SI	SI	SI
Variables (unidad 2, 4 y 5)	-	-	-	**Parcialmente**	**SI**	SI
Diagrama de flujo (unidad 3 y 4)	-	-	-	Parcialmente	Parcialmente	Parcialmente
Operaciones lógicas (unidad 3)	-	-	-	Parcialmente	Parcialmente	SI
Operaciones aritméticas (unidad 4)	-	-	-	**NO**	**Parcialmente**	SI
Coordenadas (unidad 5)	-	-	-	SI	SI	SI

*Nota: * indica un concepto que también se evalúo en el test de habilidades de programación.*

Aunque los resultados del análisis MANCOVA para los conceptos bucle 'repetir veces', condicional compuesto '*if/else*' y funciones simples, no mostraron una mejora significativa en los resultados del test, en la fase usa-modifica-crea si fue posible evidenciar que los estudiantes aplicaron correctamente estos conceptos para llegar a la solución propuesta. La diferencia más evidente se dio en el concepto bucle 'repetir veces' donde el grupo desconectadas logro una mejor apropiación igualando en resultados al grupo control, mientras que el grupo conectadas solo logró parcialmente dicha apropiación, pese a que era el grupo con mejor resultado del pos-test

(M=0.733). En cuanto al concepto de funciones simples, el grupo control fue el más beneficiado al demostrar una mayor apropiación del concepto e igualando al grupo desconectadas en sus resultados, caso contrario al del grupo conectadas donde solo se logró parcialmente.

Respecto a los conceptos de direcciones básicas, bucle 'repetir-hasta' y mientras 'que-*while*' no se encontró evidencia que demuestre claramente su apropiación en las actividades planteadas, debido a que estos no se trabajaron en los ejercicios usa-modifica-crea. Ante esto no es posible asegurar que se dio la transferencia de estos conceptos a la solución de problemas usando como referente las diferencias significativas que mostró el análisis MANCOVA entre el grupo conectadas y desconectadas para el concepto de direcciones básicas. En cuanto al concepto condicional simple-*'if'* en el modelo usa-modifica-crea los resultados fueron más evidentes. Allí, la mayoría de los estudiantes en los tres grupos pudieron aplicar este concepto a los ejercicios desarrollados, comprendiendo las estructuras condicionales al programar secuencias lógicas de bloques que correspondían a las indicaciones propuestas, por ejemplo, para la unidad 2 y 3 que implicaba definir decisiones para resolver correctamente el problema. Esto es acorde con los resultados del pos-test donde se presentó una mejora en el puntaje posterior a la intervención. Finalmente, para el concepto de funciones simples, se evidenció una apropiación básica en cuanto a la creación y llamado de una función a lo largo del programa, no obstante, debido a la complejidad de este concepto varios estudiantes erraron en la implementación de la solución lo cual es coherente con los resultados obtenidos del test de pensamiento computacional.

Respecto a los ejercicios abordados en la fase usa-modifica-crea, se pudo encontrar evidencias de la apropiación de los conceptos de entradas y salidas de datos, variables booleanas, algoritmos y coordenadas, generando un impacto positivo en los tres grupos. En contraste, para los conceptos de variables, diagrama de flujo, operaciones lógicas y operaciones aritméticas se

presentaron diferencias en la apropiación de estos conceptos que bien pueden obedecer al entrenamiento previo mediante actividades conectadas y desconectadas. Ante esto podemos inferir que las actividades conectadas no influyeron en el manejo de operaciones aritméticas a pesar del entrenamiento previo y el trabajo del usa-modifica-crea, mientras que las actividades desconectadas si lograron ser efectivas para comprender el uso de variables, más no para lo otros conceptos. Cabe mencionar que no se evidencio una apropiación lo suficientemente fuerte en el concepto de diagrama de flujo para ninguno de los grupos, lo cual es sorprendente ya que fue el concepto que más se recalcó a lo largo de las unidades 3 y 4.

Otro aspecto importante que brinda el análisis del usa-modifica-crea es poder visualizar cada una de las estrategias de aprendizaje empleadas por los estudiantes en la construcción de su artefacto computacional que a su vez reflejan prácticas del pensamiento computacional, entre ellas: 1) reutilización de bloques usados previamente en los pasos usa-modifica; 2) experimentación en las variaciones que hicieron del programa con otros bloques y funciones adicionales a las propuestas en el ejercicio; 3) iteración al incluir bucles y ciclos en la simulación de eventos aleatorios y el cálculo de probabilidades; 4) prueba y depuración en la verificación del programa usando casos de prueba antes de mostrar la solución final al ejercicio propuesto. El análisis de los resultados del modelo usa-modifica-crea frente a los conceptos evaluados en el test de habilidades de programación muestran una clara diferencia con los resultados del pos-test, ya que permitió evidenciar de una forma más clara la apropiación de conceptos como: secuencialidad ya que tienen un orden en los pasos que proponen para llegar a la solución del programa; y el uso de condicionales y bucles con determinadas iteraciones en los bloques de entrada cuando se interactúa con los botones de la *Micro:bit* bajo diferentes situaciones. Sin embargo, desde la perspectiva de evaluar la aplicación del concepto para desarrollar un artefacto computacional, persistió la

dificultad en el concepto de Bucles con condición de salida ya que no tuvieron en cuenta la cantidad de veces que se buscaba en el ciclo en comparación con el anterior bucle. En cuanto al concepto de anidación, bien sea para estructuras condicionales o bucles, solo algunos estudiantes del grupo desconectadas y del grupo control mostraron hallazgos relevantes en la solución de las actividades de la unidad 2.

De acuerdo con los resultados, se observa que el modelo usa-modifica-crea enriquece el trabajo previo realizado mediante las actividades conectadas y desconectadas debido a que no se restringe únicamente a las dimensiones que se evalúan en el test de pensamiento computacional y el test de habilidades de programación, si no que permite evaluar otros aspectos relacionados con el desarrollo del pensamiento computacional y profundizar en el análisis de cómo los estudiantes están aplicando esos conocimientos en la resolución de problemas. En este sentido, encontramos aspectos positivos evaluados en ambos test que convergen en este modelo como lo son: bucles, condicional simple – *if* y *c*ondicional compuesto – *if/else,* pero también hay otros presentes en cada una de las etapas del usa-modifica-crea que permiten ver el impacto del entrenamiento previo sobre la segunda fase de la intervención desde el punto de vista de: secuencialidad, anidación y funciones.

5.12 Aportes de las actividades conectadas y desconectadas en la solución de problemas mediante el modelo de progresión de tres estados: usa-modifica-crea.

En cuanto a la estructura general del modelo usa-modifica-crea, evidenciamos que cada una de las etapas realiza un aporte en la consolidación tanto del concepto como de la práctica computacional que principalmente está orientada a la resolución de problemas, pero que dependiendo del trabajo previo con actividades conectadas o desconectadas permite potenciar cierto tipo de habilidades requeridas por el estudiante y facilita el proceso desde el planteamiento a la implementación de un artefacto computacional.

Para el análisis puntual de los tres estados del modelo usa-modifica-crea, se observó que el paso "usa" permitió enseñar programación mediante plantillas de código con las que los estudiantes pudieron interactuar sin necesidad de escribir todo el código por sí mismos, concentrándose en los componentes básicos del algoritmo y mejorando su capacidad para identificar las partes relevantes del problema. Frente a esto, pudimos evidenciar que en su mayoría los estudiantes del grupo desconectadas fueron quienes siguieron mejor las instrucciones para el desarrollo final del ejercicio e interactuaron de forma asertiva con la plataforma *MakeCode*. Ahora, en el segundo paso "modifica", los estudiantes de los tres grupos, como usuarios de la creación de otra persona, se enfrentaron al hecho de abstraer los elementos relevantes del modelo que se presentó ante ellos, interpretarlo y reconocer que parámetros eran susceptibles a cambios, sin embargo, los grupos conectadas y control mostraron una mayor disposición a interactuar con los diferentes bloques de la plataforma gracias a la posibilidad de adaptar el modelo inicial, indicando que los ejercicios del entrenamiento previo a esta etapa influyeron positivamente en los resultados, especialmente en el grupo conectadas quienes interactuaron con ejercicios de programación por bloques en la plataforma *Code.org*. Finalmente, las evidencias de aprendizaje analizadas en el paso

“crea” permitieron ver que la mayoría de los estudiantes en los tres grupos pusieron a prueba su capacidad en la creación de artefactos computacionales al enfrentarse al reto de tomar medidas lógicas que los llevaran a construir una posible solución y finalmente, supervisar y evaluar la implementación propuesta al ejercicio planteado. De esta forma lograron plasmar sus ideas en un diseño propio de acuerdo con sus preferencias, demostrando originalidad y creatividad en la solución del problema, especialmente en las unidades donde se trabajaron situaciones cercanas a su entorno, como lo fue el ejercicio de simular el tablero digital de una lavadora y el juego de luces traseras para la bicicleta.

6. Discusión de resultados

El propósito de esta investigación fue comparar el efecto diferencial de las actividades conectadas y desconectadas en la apropiación de conceptos computacionales y en el desarrollo de habilidades de pensamiento computacional cuando se integran como actividades previas al trabajo en el entorno de aprendizaje *MakeCode*, siguiendo el modelo usa-modifica-crea. Para ello, se seleccionó un conjunto de actividades conectadas y desconectadas que permitieron la apropiación de los conceptos computacionales de secuencias, bucles, condicionales, entre otros, según el marco propuesto por Brennan y Resnick (2012).

Los resultados obtenidos en el pos-test psicométrico de habilidades básicas de programación (Mühling et al., 2015) si bien se pudo observar que los tres grupos mejoraron el puntaje respecto al pre-test, el grupo control se destacó por registrar un puntaje superior, mientras que el grupo que realizó actividades conectadas obtuvo el menor puntaje frente a los otros dos grupos. Los ejercicios del entrenamiento mediante actividades conectadas desarrolladas en *Code.org* fueron los que más se acercaban a las preguntas que evaluaba el test de Mühling (2015). Una posible explicación del resultado radica en el diseño mismo de las actividades conectadas, ya que no siempre existe correspondencia entre la programación por bloques del entrenamiento y el pseudocódigo que expone el test en sus preguntas, caso contrario a las actividades desconectadas que en algunos ejercicios incluyo pseudocódigo para hacer referencia a secuencias de pasos, condicionales y ciclos. De manera que la estructura de las actividades propuestas y el formato en el que se presentaban los ejercicios previos resulto ser más relevante para la apropiación del concepto de lo que se esperaba.

En cuanto al pos-test de pensamiento computacional (Román-González, 2015) indican un avance en los grupos experimentales, en especial para el grupo de actividades conectadas, que

mejoró significativamente el puntaje global del test, mientras que el grupo desconectadas no mostró una variación significativa y el grupo control disminuyo el puntaje obtenido frente a la prueba inicial.

Al evaluar la incidencia del trabajo previo con actividades conectadas y desconectadas en la apropiación de conceptos computacionales durante la solución de problemas en *MakeCode* con *Micro:bit* se encontró que para el concepto de direcciones básicas, el grupo conectadas fue el que más sobresalió al obtener el más alto puntaje del test, lo que indica que las actividades conectadas desarrolladas en *Code.org* favorecieron el desarrollo de habilidades del pensamiento computacional en los estudiantes, mejorando su nivel de comprensión para establecer las rutas que debían seguir los personajes y a la interacción visual en cada uno de los ejercicios propuestos. Por el contrario, las actividades desconectadas no indican aportes relevantes al desarrollo de este concepto ni diferencias significativas en los resultados del MANCOVA. Estos resultados son consistentes con otras investigaciones que también involucraron recursos en línea basados en bloques como *Scratch* o *Code.org* (García, 2022; Weintrop et al., 2018; Krugel & Ruf, 2020) demostrando que el implementar metodologías que involucran herramientas tecnológicas en el proceso de enseñanza-aprendizaje, posibilita el desarrollo e incremento de las habilidades del pensamiento computacional (Brennan y Resnick, 2012).

De igual forma, en el concepto computacional condicional simple el grupo de actividades conectadas mostró un mejor puntaje frente a los otros dos grupos y registró diferencia significativa con respecto al pre-test, mientras que el grupo que trabajo actividades desconectadas presentó una disminución en su puntaje frente al pre-test. Ante esto, se puede inferir que la programación por bloques que ofrece *Code.org*, el diseño de las actividades, el tiempo dedicado a ejercicios

relacionados con condicionales y el uso frecuente de este concepto a través de las lecciones pudo haber influido considerablemente en la apropiación de este concepto básico de programación.

En los grupos desconectadas y control la principal dificultad radicó en evaluar la acción dada por la condición, así como determinar si esta era verdadera o falsa, lo cual no siempre es posible al llevarse a la práctica y realizar la programación en el modelo propuesto. Esto coincide con el trabajo realizado por López y Pineda (2022) quienes evidenciaron que el condicional simple fue uno de los conceptos que presentó mayores dificultades en su apropiación y el más complejo de interpretar tanto para el grupo que trabajo actividades conectadas como para el grupo de actividades desconectadas. Un mayor tiempo de dedicación a ejercicios relacionados, por ejemplo, con la toma de decisiones, situaciones cotidianas con única elección o dicotómicas, podría mejorar notablemente la comprensión y aplicación de este concepto computacional y ayudar a los estudiantes en su apropiación.

A pesar del leve aumento del puntaje en el pos-test de los tres grupos para los conceptos computacionales bucle ‘repetir-veces” y bucle “hasta”, no se registraron diferencias significativas en los resultados del MANCOVA. Frente a ello, la comprensión previa que se tuvo con los condicionales fue un factor determinante para la comprensión y aplicación de este concepto en el grupo de actividades conectadas.

Otro aspecto relevante es la retroalimentación durante las actividades, lo pudo contribuir a que los estudiantes entendieran el tipo de estructura repetitiva y su funcionamiento. En el grupo de actividades desconectadas la retroalimentación se recibió al cierre de la actividad por parte del docente tras la revisión de las evidencias de aprendizaje, mientras que en el grupo conectadas se realizaba de forma instantánea a través de los mensajes en pantalla que arrojaba la interfaz gráfica.

Esta diferencia en los tiempos de retroalimentación es un aspecto que vale la pena profundizar en futuros estudios.

De esta manera, las actividades realizadas en *Code.org* parecen haber facilitado aún más la comprensión de este concepto al ofrecer una retroalimentación pertinente y brindarle al estudiante la oportunidad de depurar y corregir su código de forma inmediata, lo cual es posible gracias a las características propias de los entornos gráficos de programación por bloques. Estos resultados son contrarios al trabajo realizado por López y Pineda (2022), donde se evidenció dificultad en este mismo concepto cuando los estudiantes debían determinar el número de repeticiones en los ejercicios propuestos, atribuyéndole a la interactividad que ofrece la plataforma de *Scratch*, los inconvenientes presentados para que los estudiantes pudieran determinar la cantidad de veces que se debía repetir una secuencia de instrucciones. No obstante, se requiere dedicar más tiempo al entrenamiento para lograr minimizar los efectos inherentes al diseño de las actividades y ayudar al estudiante a comprender como trabajar el concepto de bucles.

Respecto a los resultados del condicional compuesto los resultados fueron consistentes con los obtenidos en el condicional simple. Los grupos que trabajaron actividades desconectadas y grupo de control mostraron una disminución significativa frente al pre-test, observando que al igual que con el condicional simple, existe una falencia en la selección de los operadores relacionales (mayor, menor, igual, etc.) y la identificación de las acciones a ejecutar en las dos posibles soluciones que presenta cada parte de este condicional. Contrario a ello, el grupo conectadas se vio beneficiado por el entrenamiento y obtuvo un mejor resultado, lo cual es atribuible al hecho de que las actividades conectadas de *Code.org* tienden a combinar en un misma lección, condicionales simples y dobles, o en su defecto condicionales con bucles, lo que implica que los estudiantes usen de forma reiterativa el concepto de condicional y sea más fácil para ellos

identificar cuándo se ejecutan ciertas acciones para el caso en que la condición sea verdadera o falsa.

Aun cuando los resultados del MANCOVA no mostraron diferencias significativas entre los dos tipos de entrenamientos para el condicional compuesto, se puedo observar que las actividades conectadas permitieron reforzaran el concepto de condicional simple y programar otras instrucciones que funcionan de forma diferente de acuerdo con los criterios específicos del problema. Análogamente, en el trabajo realizado por López y Pineda (2022), ninguno de los dos grupos experimentales logró comprender que acciones o instrucciones programar en el caso de que se cumpla o no la condición, encontrándose las mismas dificultades para la comprensión de los conceptos de condicional simple y compuesto.

Aunque el concepto de funciones simples fue el segundo concepto con puntaje más bajo en el pos-test, el grupo de actividades desconectadas mostró una mejor apropiación al lograr identificar cuáles son las instrucciones que se repetían y que les permitía simplificar el código. Los grupos conectadas y control obtuvieron un puntaje inferior al pre-test. El análisis de las evidencias de aprendizaje de los estudiantes da cuenta de que este concepto resultó más complejo para los estudiantes por lo cual emplearon ciclos y bucles en las soluciones, en vez de crear una función que incluyera las instrucciones que se repetían en el programa.

Una explicación a este hecho puede plantearse desde el diseño propio de las actividades que realizó el grupo desconectadas para el concepto funciones, que planteaban hacer una semejanza entre las frases que se repiten en una canción y las instrucciones repetitivas de un código, por lo que hacer una similitud entre el coro de una canción y una función fue una estrategia acertada para que los estudiantes pudieran identificar un concepto que implica un mayor nivel de abstracción. En consecuencia, los estudiantes tuvieron dificultad para establecer cuáles eran las

instrucciones que se repetían en un código y que se podían agrupar para crear una función que simplificar el código. Estos hallazgos son consistentes con los de Sun et al (2021) y Zhang & Cui (2021) quienes indicaron que las actividades desconectadas pueden no ser eficaces a largo plazo para cultivar habilidades de pensamiento computacional más complejas.

En cuanto al concepto de secuencialidad, el grupo desconectadas mostró un pensamiento más estructurado al plantear secuencias lógicas de bloques de programación en sus algoritmos y el seguimiento de reglas para representar vocales y consonantes en el ejercicio de mensajes codificados. En este sentido, pudo evidenciarse que las actividades propuestas para este grupo, que involucraron seguir una serie de pasos e instrucciones e identificar símbolos para ser procesados hasta completar la tarea, y que estuvieron orientadas a mejorar la precisión para crear, identificar y solucionar fallos en las secuencias de instrucciones, fueron efectivas para comprender la secuencialidad lo que les permitió adquirir experiencia en la lectura y escritura de código. Lo anterior es consecuente con el trabajo realizado por Grover, Lundh & Jackiw (2019), Delal & Oner (2020) y Looi, et al. (2018), quienes indican que las actividades desconectadas contribuyen significativamente a la capacidad de expresar más asertivamente un algoritmo mediante un pseudocódigo o un diagrama de flujo.

Por otra parte, el grupo de actividades conectadas fue el que más dificultad presentó en el manejo de la secuencialidad y orden de los bloques de programación requeridos para algunos ejercicios. Esto da indicios de que el tiempo de entrenamiento previo mediante las actividades de *Code.org* pudo ser insuficiente para afianzar la creación de algoritmos secuenciales y la creación de diseños repetitivos mediante el entorno de programación. Esto también influyo en el concepto de variables booleanas, donde presentaron mayor dificultad para establecer que leds en la *Micro:bit*

eran los indicados para representar los símbolos asociados a las letras y vocales de los ejercicios iniciales.

El análisis cualitativo de las evidencias de aprendizaje de los estudiante muestra que los estudiantes lograron comprender el concepto de entradas y salidas por medio de la simulación en *MakeCode*. Esto se puede atribuir a las características propias de la *Micro:bit*, como la incorporación de leds y botones, los cuales representan los dos posibles estados booleanos mediante el encendido y apagado de un led, o al presionar o no el botón, lo cual se asemeja a la condición *true* y *false* en programación.

De igual forma, se observó dificultad en la creación y uso de variables en el entorno de programación *MakeCode*, debido a la definición que usa la interfaz para los comandos "establecer" y "cambiar", los cuales generaron confusión entre las acciones de asignar, guardar o acumular un valor a la variable, limitando la comprensión del estudiante frente al concepto. Esta dificultad también se trasladó al manejo de funciones, donde los errores de definición de los valores iniciales de las variables, la validación del valor máximo para cada variable que permite la matriz de leds y el incremento que debían tener a lo largo de la ejecución del programa dificultaron la implementación de los ejercicios propuestos.

Al contrastar los aportes de las actividades desconectadas a la solución de problemas de programación *en MakeCode,* cuando se sigue el modelo de progresión de tres estados (usa-modifica-crea), puede afirmarse que los objetivos de las actividades desconectadas estuvieron más alineados con los propósitos del usa-modifica-crea en el sentido que la mayoría de las actividades estaban orientadas a la decodificación y ejecución de un programa, así como a la identificación y solución de fallos o errores en las instrucciones secuenciadas, esta coincidencia facilitó a los estudiantes convertir un algoritmo en un programa y que las evidencias de aprendizaje se acercarán

al resultado esperado en la mayoría de las unidades. Al tiempo que trabajaron reiteradamente ejercicios con condicionales simples, dobles y anidados; su narrativa partió de una situación cercana al contexto, permitiendo desarrollar la capacidad de abstracción de los estudiantes. Dichas actividades se desarrollaron de forma grupal y la interacción con otros estudiantes pudo contribuir a encontrar maneras diferentes de solucionar el problema y plantear estrategias. Y estuvieron enfocadas a realizar ejercicios de codificación sin limitarse al uso exclusivo de bloques de programación establecidos en una interfaz.

Por otra parte, en el grupo que recibió entrenamiento en actividades conectadas, la cantidad de actividades que involucran movimientos y acciones repetitivas de los personajes ayudaron a identificar más fácilmente el uso de bucles y ciclos para dar solución al problema. Y el refuerzo de los conceptos computacionales se realizó de forma reiterada a lo largo de las lecciones, lo que permitió desarrollar y solucionar eficazmente los problemas planteados en cada actividad.

Por estas razones, puede afirmarse que el entrenamiento previo en entornos desconectados bajo las condiciones expuestas facilitó el trabajo posterior en la interfaz de *MakeCode* y *Micro:bit,* reforzando la apropiación de los conceptos computacionales para transferir ese conocimiento directamente a la resolución de problemas. Esto reafirma resultados previos obtenidos por Rúales (2022), Tyrén et al. (2018), Cederqvist (2020) y Bocanegra (2020), quienes encontraron que involucrar el uso de actividades desconectadas a partir del uso de *MakeCode* y la tarjeta *Micro:bit,* contribuye significativamente al fortalecimiento de las habilidades de pensamiento computacional orientadas a la programación.

Otros aspectos que mostraron diferencias en la apropiación de estos conceptos a la solución de problemas fue la narrativa, en las conectadas los personajes de videojuegos no fue relevante, mientras que en las actividades desconectadas fue más cercana al contexto del estudiante,

permitiendo desarrollar su capacidad de abstracción; la interacción en el desarrollo de las actividades, que en el grupo conectadas fue de forma individual, mientras que en las desconectadas fue grupal, lo que contribuyó a encontrar maneras diferentes de solucionar el problema y plantear estrategias; y el enfoque de las actividades a realizar ejercicios de codificación, que en el caso de las actividades desconectadas no se limitó al uso exclusivo de bloques de programación establecidos en una interfaz

7. Conclusiones

Respecto al efecto diferencial de las actividades conectadas y desconectadas en la apropiación de conceptos computacionales y en el desarrollo de habilidades de pensamiento computacional cuando se integran como actividades previas al trabajo en el entorno de aprendizaje *MakeCode*, siguiendo el modelo de progresión de tres estados: usa-modifica-crea, puede concluirse que:

El desarrollo de habilidades de pensamiento computacional y la apropiación de conceptos computacionales es más efectivo cuando se entrena a través de actividades conectadas. Estos resultados son atribuibles a que las actividades conectadas mejoraron el resultado del pos-test luego del entrenamiento previo en los conceptos computacionales: direcciones básicas y condicionales simples, reforzándolos de forma reiterativa a lo largo de las lecciones. Asimismo, la estructura de las actividades de la plataforma *Code.org*, la interfaz gráfica de bloques de programación, el grado de dificultad incremental en la medida que se avanza de nivel, la retroalimentación instantánea y la repetitividad de los ejercicios a través de las lecciones, son componentes que permiten a los estudiantes comprender cómo funciona una estructura en programación.

En cuanto a la incidencia del trabajo con actividades conectadas y desconectadas en la apropiación de conceptos computacionales durante la solución de problemas en *MakeCode* con *Micro:bit*, pudo establecerse que el trabajo en el entorno de programación MakeCode siguiendo el modelo usa-modifica-crea promueve la apropiación y transferencia a la solución de problemas de otros conceptos computacionales (condicionales compuestos, bucles, funciones simples, variables booleanas). Así mismo, el entrenamiento previo específico sobre los conceptos computacionales y su uso reiterativo en las actividades incide en su apropiación. No obstante, es conveniente una mayor dedicación a ejercicios relacionados con estos conceptos para afianzar el desarrollo de

habilidades del pensamiento computacional orientadas a la solución de problemas en diferentes contextos.

Con relación a los aportes del entrenamiento previo mediante actividades conectadas y desconectadas a la solución de problemas de programación en *MakeCode* cuando se sigue el modelo de progresión de tres estados (usa- modifica-crea), puede concluirse que un diseño de actividades de entrenamiento más cercano a la narrativa del contexto del estudiante, que promueva el trabajo grupal y que no se limite a la programación por bloques, desarrolla mejor la apropiación de conceptos computacionales, su transferencia a la solución de problemas y la aplicación de estrategias tales como ensayo y error, depuración, decodificación y remezcla. De igual forma, el entrenamiento con actividades previas que siguen la progresión usa-modifica-crea generan un mejor desempeño en la solución de problemas cuando se trabajan en ambientes como *MakeCode* con *Micro:bit*.

Adicionalmente, la progresión usa-modifica-crea aplicada al entorno *MakeCode* es un andamiaje adecuado para la apropiación y transferencia de conceptos computacionales a la solución de problemas que en este estudio resultó ser más efectivo que el entrenamiento previo con actividades conectadas y desconectadas. Esto permite proyectar futuros estudios que analicen la aplicación de este modelo en actividades de entrenamiento previo, favoreciendo la elección de estrategias adecuadas para la resolución de problemas y el desarrollo del pensamiento computacional a largo plazo.

8. Contribuciones, limitaciones y proyecciones

Este estudio tuvo las siguientes limitaciones. Primera, el tiempo de ocho semanas dedicado al entrenamiento, el cual no resultó ser suficiente para un desarrollo más amplio de las habilidades del pensamiento computacional y para la apropiación de los conceptos computacionales propuestos. Segunda, el tamaño de la muestra, que, aunque fue aceptable para este estudio, podría haber afectado al análisis estadístico y limitar la posibilidad de generalizar nuestros resultados. Tercera, las diferencias en los procesos de colaboración y trabajo individual en el desarrollo de las actividades conectadas y desconectadas, lo que pudo afectar la motivación y el desempeño de los estudiantes.

En todo caso, esta investigación genera aportes en el campo del pensamiento computacional, sino que tiene repercusión para el estudio sobre su relación con la resolución de problemas gracias a la integración del componente usa-modifica-crea, el cual permite esbozar un acercamiento más profundo al uso de estrategias computacionales a las que recurren los estudiantes cuando se enfrentan a un problema.

En consecuencia, la inclusión de diferentes estrategias que promueven el desarrollo del pensamiento computacional en el currículo es, sin duda, fundamental para que la educación en ciencias de la computación resulte beneficiosa a corto plazo para el futuro profesional de los estudiantes. En este mismo sentido, la aplicación de un modelo mixto más amplio que combine las actividades desconectadas, conectadas y el modelo usa-modifica-crea, permitirían dar indicios hacía que estrategia que sea particularmente eficaz en mejorar las habilidades de pensamiento computacional y generar una mayor apropiación de los conceptos computacionales básicos que se requieren para la programación.

Como proyección para futuras investigaciones, se puede abordar de manera más integral el desarrollo del pensamiento computacional incluyendo otras variables que se relacionan con las actividades de programación, como la actitud de los estudiantes hacia la programación, la motivación, el trabajo colaborativo, entre otros factores. Ya que este estudio estuvo enmarcado en el primer componente del marco conceptual propuesto por Brennan y Resnick (2012), es decir, la apropiación de los conceptos computacionales, futuros estudios podrían ampliar su alcance para determinar la influencia de las actividades conectadas y desconectadas en las prácticas y perspectivas del pensamiento computacional. Finalmente, sería importante ampliar la investigación sobre un grupo de edad más amplio, ya sea incluyendo como participantes otros grados de educación media o instituciones.

Referencias

Adell, J., Llopis, M., Esteve, F., & Valdeolivas, M. (2019). El debate sobre el pensamiento computacional en educación. RIED-Revista Iberoamericana De Educación a Distancia, 22(1), 171–186. https://doi.org/10.5944/ried.22.1.22303

Baldwin, D., Walker, H., & Henderson, P. (2013). The roles of mathematics in computer science. ACM Inroads, 4(4), 74-80. https://doi.org/10.1145/2537753.2537777

Ball T., Protzenko J., Bishop J., Moskal M., de Halleux J., Braun M., Hodges S. y Riley C. (2016) - Microsoft touch develop and the BBC *Micro:bit*: 637-640, doi: https://dl.acm.org/doi/10.1145/2889160.2889179

Barradas, R., Lencastre, J., Soares, S. y Valente, A. (2020). Developing Computational Thinking in Early Ages: A Review of the code.org Platform. 157-168. DOI: 10.5220/0009576801570168. https://hdl.handle.net/1822/65432.

Bell, T. Alexander, J., Freeman, I. & Grimley, M. (2009). Computer Science Unplugged: school students doing real computing without computers. The New Zealand Journal of Applied Computing and Information Technology. 13(1), 20-29.

Bell, T., & Vahrenhold, J. (2018). CS Unplugged—How Is It Used, and Does It Work? Lecture Notes in Computer Science, 497–521. doi:10.1007/978-3-319-98355-4_29

Bocanegra, N. (2020). El pensamiento computacional con las tarjetas *Micro:bit*, como estrategia pedagógica para el desarrollo de habilidades en estudiantes de noveno. https://repositorio.udes.edu.co/server/api/core/bitstreams/6606a3af-84e3-4700-babb-b0237edb805e/content

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing computational thinking in compulsory education implications for policy and practice. Sevilla: Joint Research Centre. doi: https://doi.org/10.2791/792158

Boulden, D., Rachmatullah, A., Hinckle, M., Bounajim, D., Mott, B., Boyer, K., Lester J. y Wiebe, E. (2021). Supporting Students' Computer Science Learning with a Game-based Learning Environment that Integrates a Use-Modify-Create Scaffolding Framework. Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V.1. https://dl.acm.org/doi/10.1145/3430665.3456349

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., y Barone, D. (2017). Development of Computational Thinking Skills through Unplugged Activities in Primary School. Proceedings of the 12th Workshop on Primary and Secondary Computing Education – WiPSCE '17.doi: https://doi.org/10.1145/3137065.3137069

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking [Conference presentation]. AERA. Annual American Educational Research Association Meeting, Vancouver, BC, Canada. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Buitrago-Flórez, F., Danies, G., Restrepo, S., y Hernández, C. (2021). Fostering 21st century competences through computational thinking and active learning: a mixed method study. International Journal of Instruction, 14(3), 737-754. https://files.eric.ed.gov/fulltext/EJ1304694.pdf

Buitrago-Flórez, F. (2018). Designing, implementing and evaluating, a first step approach for computational thinking development in novice students. Uniandes. https://repositorio.uniandes.edu.co/handle/1992/38738

Carmona-Mesa, J., Castrillón-Yepes, A., Quiroz-Vallejo, D. y Villa-Ochoa, J. (2021). Integración del Pensamiento Computacional en educación primaria y secundaria: documento de posición. Medellín: Siemens Stiftung, Siemens Caring Hands y Universidad de Antioquia. https://crea-portaldemedios.siemens-stiftung.org/integracion-del-pensamiento-computacional-en-educacion-primaria-y-secundaria-documento-de-posicion-102868

Casali, A., Deco, C., Viale, P., Bender, C., Zanarini, D. & Monjelat, N. (2020). Enseñanza y Aprendizaje del Pensamiento Computacional y la Programación en los distintos Niveles Educativos. http://sedici.unlp.edu.ar/handle/10915/104106

Cederqvist (2020) An exploratory study of technological knowledge when pupils are designing a programmed technological solution using BBC *Micro:bit*. Int J Technol Des Educ 32, 355–381. https://doi.org/10.1007/s10798-020-09618-6

Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. TechTrends. doi: https://doi.org/10.1007/s11528-018-0292-7

Cooper, S. (2010). The Design of Alice. ACM Transactions on Computing Education, 10(4), 1–16. doi:10.1145/1868358.1868362

Coronel, E. y Lima, G. (2020). El pensamiento computacional. Nuevos retos para la educación del siglo XXI Virtualidad, Educación y Ciencia, 20 (11), pp. 115-137. https://revistas.unc.edu.ar/index.php/vesc/article/view/27451

Correa, F. (2020). El uso Tecnologías educativas para el desarrollo del pensamiento computacional de los estudiantes de quinto de primaria del establecimiento educativo rural el Pital Caldono Cauca. https://repositorio.udes.edu.co/server/api/core/bitstreams/28062a68-a43c-4d6e-8747-7216980252c9/content

Delal, H., & Oner, D. (2020). Developing middle school students' computational thinking skills using unplugged computing activities. Informatics in Education, 19(1), 1–13. https://doi.org/10.15388/INFEDU.2020.01

De las Heras, P., Araguz, A., González, J., Moreno, J. (2018). Pensamiento computacional en la educación [mesa redonda], Congreso Nacional 'Transformación Digital Educativa', Madrid, España. http://media.educalab.es/JCswLCdgMywoeX97d3Z8fGwXeH57CHZ9dXN0e39sdw~~.mp4

Díaz, B. y Molina, A. (2020). Las Habilidades del pensamiento computacional en estudiantes de educación básica, alineados con objetivos de aprendizaje de la asignatura de Matemáticas. http://repositorio.udec.cl/xmlui/handle/11594/6496

Del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of Primary Education. Computers & Education, 150, 103832. doi: https://doi.org/10.1016/j.compedu.2020.103832

Duarte V., Carillo G., Carrera A. y Falcones C. (2019) Experiencias sobre la inducción de tecnologías programables para el desarrollo del pensamiento computacional en escuelas de zonas rurales y urbano marginales. Vínculos-Espe (2020) VOL.5, No.3: 67 - 81 doi: 10.24133/vinculosespe.v5i3.1720

Estévez J., Garate G. & Graña M. (2019), *Gentle Introduction to Artificial Intelligence for High-School Students Using Scratch*, IEEE Access, *7*, 179027-179036, doi: https://doi.org/10.1109/ACCESS.2019.2956136

Federación Colombiana de la Industria de Software y Tecnologías Informáticas Relacionadas, FEDESOFT (2022) *Bebras* - Desafío Internacional de Informática y Pensamiento

Computacional. https://old.fedesoft.org/bebras-desafio-internacional-de-informatica-y-pensamiento-computacional-2022/

Caeli, E. N., & Yadav, A. (2019). Unplugged Approaches to Computational Thinking: A Historical Perspective. TechTrends. doi:10.1007/s11528-019-00410-5

García, M., Deco C., Bender C. & Collazos C. (2021). Una Propuesta para el Desarrollo de pensamiento computacional en Niños y Jóvenes. Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología, no. 30, pp. 16-27, 2021. doi: https://doi.org/10.24215/18509959.30.e2

García, A. (2022). Enseñanza de la programación a través de Scratch para el desarrollo del pensamiento computacional en educación básica secundaria. Revista Academia y Virtualidad, ISSN 2011-0731, 15 (1), 2022, pp. 161-182. Doi: https://doi.org/10.18359/ravi.5883

García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in pre-university education. Computers in Human Behavior, 80, 407–411. https://doi.org/10.1016/j.chb.2017.12.005

González, F., López, C., & Castro, C. (2018). Development of Computational Thinking in High School Students: A Case Study in Chile. 1-8. 10.1109/SCCC.2018.8705239. Doi: 10.1109/SCCC.2018.8705239

González, P. (2018). Estudio de evaluación preliminar del aporte del pensamiento computacional en la educación media. Universidad Técnica Federico Santa María, Chile. https://repositorio.usm.cl/handle/11673/41243

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43. DOI:10.3102/0013189X12463051. https://www.researchgate.net/publication/258134754_Computational_Thinking_in_K-12_A_Review_of_the_State_of_the_Field

Grover S. y Pea R. (2017). Computational Thinking: A competency whose time has come. Computer science education: Perspectives on teaching and learning in school, 19. doi.10.5040/9781350057142.ch-003

Grover, S., Lundh, P., & Jackiw, N. (2019). Non-Programming Activities for Engagement with Foundational Concepts in Introductory Programming. Proceedings of the 50th ACM Technical Symposium on Computer Science Education. doi:10.1145/3287324.3287468

Guggemos, J. (2021). On the predictors of computational thinking and its growth at the high-school level. Computers & Education, 161, 104060. doi: https://doi.org/10.1016/j.compedu.2020.104060

Houchins, J. K., Boulden, D. C., Lester, J., Mott, B., Boyer, K. E., y Wiebe, E. N. (2021). How Use-Modify-Create Brings Middle Grades Students to Computational Thinking. International Journal of Designs for Learning, 12(3), 1–20. https://scholarworks.iu.edu/journals/index.php/ijdl/article/view/30733

Huang, W., & Looi, C.-K. (2020). A critical review of literature on "unplugged" pedagogies in K-12 computer science and computational thinking education. Computer Science Education, 1–29. doi:10.1080/08993408.2020.1789411

Iglesias, A. y Bordignon, F. (2021). Taxonomía de actividades desconectadas para el desarrollo de pensamiento computacional. Universidad Pedagógica Nacional, Argentina 22. 119-135. https://www.academia.edu/45111806/Taxonom%C3%ADa_de_actividades_desconectadas_para_el_desarrollo_de_pensamiento_computacional_A_taxonomy_of_unplugged_activities_for_computational_thinking_development

Instituto Colombiano para la Evaluación de la Educación, ICFES (2019). Marco de referencia de la prueba de matemáticas Saber 11.°. Bogotá: Dirección de Evaluación, ICFES. https://www.icfes.gov.co/documents/39286/2215178/Marco+de+referencia+-+Prueba+de+matem%C3%A1ticas+saber+11.pdf/2a46a871-4706-b56f-4da7-0c607f448c69?t=1647954909577.

Kale, U., & Yuan, J. (2020). Still a New Kid on the Block? Computational Thinking as Problem Solving in Code.org. Journal of Educational Computing Research, 59(4), 620–644. doi:10.1177/0735633120972050

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. Computers in Human Behavior, 52, 200–210. doi: https://doi.org/10.1016/j.chb.2015.05.047

Kastner-Hauler O, Tengler K, Sabitzer B and Lavicza Z (2022) Combined Effects of Block-Based Programming and Physical Computing on Primary Students' Computational Thinking Skills. Front. Psychol. 13:875382. doi: 10.3389/fpsyg.2022.875382

Krugel J. & Ruf A. (2020). Learners' perspectives on block-based programming environments: code.org vs. scratch. In Proceedings of the 15th Workshop on Primary and Secondary Computing Education (WiPSCE '20). Association for Computing Machinery, New York, NY, USA, Article 27, 1–2. https://dl.acm.org/doi/abs/10.1145/3421590.3421615

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). Computers in Human Behavior, 72, 558–569. doi:10.1016/j.chb.2017.01.005

Laura-Ochoa, L., & Bedregal-Alpaca, N. (2021). Análisis de entornos de programación para el desarrollo de habilidades del pensamiento computacional y enseñanza de programación a principiantes. RISTI - Revista Ibérica de Sistemas e Tecnologías de Informacao, (E43), 533-548. https://www.proquest.com/openview/7504eba5c6fe8f2ba820160b1fda9a0e/1?pq-origsite=gscholar&cbl=1006393

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith J., & Werner, L. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32. doi: https://doi.org/10.1145/1929887.1929902

Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K--8 curriculum. ACM Inroads, 5(4), 64–71. doi:10.1145/2684721.2684736

Leifheit, L., Jabs, J., Ninaus, M., Moeller, K., & Ostermann, K. (2018). Programming Unplugged: An Evaluation of Game-Based Methods for Teaching Computational Thinking in Primary School.

Lytle N., Catete V., Boulden D., Dong Y, Houchins J., Milliken A., Isvik A., Bounajim D., Wiebe E., & Barnes T. (2019). Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes. In Innovation and Technology in Computer Science Education (ITiCSE '19), July 15–17, 2019, Aberdeen, Scotland Uk. ACM, New York, NY, USA, Article, 7 pages. https://dl.acm.org/doi/10.1145/3304221.3319786

Lytle, N., Catete, V., Isvik, A., Boulden, D., Dong, Y., Wiebe, E., & Barnes, T. (2019). From "Use" to "Choose." Proceedings of the 14th Workshop in Primary and Secondary Computing Education on - WiPSCE'19. doi:10.1145/3361721.3362110

Lodi, M. (2020) Introducing Computational Thinking in K-12 Education: Historical, Epistemological, Pedagogical, Cognitive, and Affective Aspects, [Dissertation thesis],

Alma Mater Studiorum Università di Bologna. Dottorato di ricerca in Computer science and engineering, 32 Ciclo. DOI: https://doi.org/10.6092/unibo/amsdottorato/9188

Lodi, M. & Martini S. (2021). Computational Thinking, Between Papert and Wing. Science & Education (2021) 30:883–908. Doi: https://doi.org/10.1007/s11191-021-00202-5

Looi, C.-K., How, M.-L., Longkai, W., Seow, P., & Liu, L. (2018). Analysis of linkages between an unplugged activity and the development of computational thinking. Computer Science Education, 1–25. doi:10.1080/08993408.2018.1533297

López, L. K. y Pineda, J. O. (2022). Desarrollo de habilidades de pensamiento computacional por medio de actividades conectadas y desconectadas en estudiantes de grados sexto y séptimo. http://hdl.handle.net/20.500.12209/17730.

MacLaurin, M. (2011). The design of kodu. ACM SIGPLAN Notices, 46(1), 241. doi: https://dl.acm.org/doi/10.1145/1925844.1926413

Mantilla, R. & Negre, F. (2021). pensamiento computacional, una estrategia educativa en épocas de pandemia. Innoeduca. International Journal of Technology and Educational Innovation, 7(1), 89-106. https://doi.org/10.24310/innoeduca.2021.v7i1.10593

Martin F., Lee I., Lytle N., Sentence S., & Lao N. 2020. Extending and Evaluating the Use-Modify-Create Progression for Engaging Youth in Computational Thinking. In The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3328778.3366971

Ministerio de Educación de Ecuador (2017). Agenda Educativa Digital 2017-2021. https://educacion.gob.ec/wp-content/uploads/downloads/2017/11/Agenda-Educativa-Digital.pdf#page=3&zoom=auto,-15,674

Ministerio de Tecnologías de la Información y las Comunicaciones, (2021). Gobierno de Colombia, British Council. Pensamiento Computacional | Coding for kids. https://codingforkids.mintic.gov.co/pensamiento-computacional

Ministerio de Educación de Colombia, MEN (2021). Ministerio de Educación Nacional. Pruebas Saber. https://diae.mineducacion.gov.co/dia_e/documentos/111001107778.pdf

Ministerio de Educación de Colombia, MEN (2022). Ministerio de Educación Nacional. El Ministerio TIC abre inscripciones gratuitas para formar en pensamiento computacional a docentes de colegios públicos y privados. https://mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/210956:El-Ministerio-TIC-abre-inscripciones-gratuitas-para-formar-en-pensamiento-computacional-a-docentes-de-colegios-publicos-y-privados

Ministerio de las Tecnologías de la Información y Comunicación, MinTIC (2020). En 2020, más de 8.500 docentes colombianos fueron formados en programación por MinTIC y British Council. Página oficial MinTIC. https://mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/161280:En-2020-mas-de-8-500-docentes-colombianos-fueron-formados-en-programacion-por-MinTIC-y-British-Council

Ministerio de las Tecnologías de la Información y Comunicación, MinTIC (2021). Hasta el 12 de febrero están abiertas las inscripciones para que 50 mil colombianos se formen gratis en programación. Página oficial MinTIC. https://mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/161628:Hasta-el-12-de-febrero-estan-abiertas-las-inscripciones-para-que-50-mil-colombianos-se-formen-gratis-en-programacion

Mohaghegh, M. y McCauley, M. (2016). Computational thinking: The skill set of the 21st century. International Journal of Computer Science and Information Technologies, Vol 7(3), 1524-1530. https://ijcsit.com/docs/Volume%207/vol7issue3/ijcsit20160703104.pdf

Montes-León, H., Hijón- Neira, R., Pérez-Marín, D., & Montes-León, S. R. (2020). Mejora del Pensamiento Computacional en Estudiantes de Secundaria con Tareas Unplugged. *Education in the Knowledge Society (EKS)*, *21*, 12. https://doi.org/10.14201/eks.23002

Motoa S. (2019). Revista educación y Pensamiento. Colombia, ISSN-e 2590-8340, Versión N° 26, 2019, 107-111. http://educacionypensamiento.colegiohispano.edu.co/index.php/revistaeyp/article/view/104

Muñoz, M., Cruz, L., Herrera, E., Jiménez, J., Muñoz, A. y Ramos, D. (2020). Pensamiento Computacional para la formación de maestros: Una revisión sistemática de literatura. Proceedings of the 18th LACCEI International Multi-Conference for Engineering, Education, and Technology: Engineering, Integration, And Alliances for A Sustainable Development" "Hemispheric Cooperation for Competitiveness and Prosperity on A Knowledge-Bas. Latin American and Caribbean Consortium of Engineering. Doi:10.18687/LACCEI2020.1.1.135

Mühling, A., Ruf, A., & Hubwieser, P. (2015). Design and First Results of a Psychometric Test for Measuring Basic Programming Abilities. Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ – WiPSCE' 15. New York. 2-10. doi:10.1145/2818314.2818320

Ortega, B. (2017). pensamiento computacional y Resolución de Problemas. https://repositorio.uam.es/handle/10486/683810

Ortega, W. (2020). Recursos didácticos para el desarrollo del pensamiento computacional en estudiantes de básica secundaria y media. [Monografía]. Repositorio Institucional UNAD. https://repository.unad.edu.co/handle/10596/37455.

Ozcinar, H.; Wong, G. & Ozturk, H. T. (2017). Teaching Computational Thinking in Primary Education. IGI Global. DOI: 10.4018/978-1-5225-3200-2.ch010

Papert, S. (1980). Mindstorms—Children, Computers and Powerful Ideas. New York: Basic Books, Inc.

Peel A., Sadler T., & Friedrichsen P. (2022) Algorithmic Explanations: An Unplugged Instructional Approach to Integrate Science and Computational Thinking. J Sci Educ Technol. 2022;31(4):428-441. doi: https://doi.org/10.1007/s10956-022-09965-0

Polanco N., Planchart S. & Fernández M. (2020). Aproximación a una definición de pensamiento computacional. 10.5944/ried.24.1.27419. Revista Iberoamericana de Educación a Distancia. https://www.researchgate.net/publication/347110679_Aproximacion_a_una_definicion_de_pensamiento_computacional

Posso D., y Murcia L. (2022). Las "Actividades desconectadas" y el desarrollo del pensamiento algorítmico. Monografía; DDMPDH196. http://hdl.handle.net/10785/9635.

Quiroz-Vallejo, D., Carmona-Mesa, J., Castrillón-Yepes, A. & Villa-Ochoa, J. (2021). Integración del pensamiento computacional en la educación primaria y secundaria en Latinoamérica: una revisión sistemática de literatura. Revista de Educación a Distancia (RED), 21(68). https://doi.org/10.6018/red.485321

Rachmatullah, A., Akram, B., Boulden, D., Mott, B., Boyer, K., Lester, J., & Wiebe, E. (2020). Development and validation of the middle grades computer science concept inventory (MG-CSCI) assessment. EURASIA Journal of Mathematics, Science and Technology Education, 16(5), 1–11. https://doi.org/10.29333/ejmste/116600.

Rachmatullah, A., Mayhorn, C. B., & Wiebe, E. N. (2021). The effects of prior experience and gender on middle school students' computer science learning and monitoring accuracy in the Use-Modify-Create progression. Learning and Individual Differences, 86, 101983. doi:10.1016/j.lindif.2021.101983

Ramírez C., Raluy M. & Vega L. (2022). Desarrollo del pensamiento computacional en niñas y niños usando actividades desconectadas y conectadas de computadora. RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo. Vol.12 no.23 Guadalajara jul/dic 2021 Epub14-Feb-2022. https://doi.org/10.23913/ride.v12i23.1079.

Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020). Computational thinking and mathematics using Scratch: an experiment with sixth grade students. Interactive Learning Environments, 28(3), 316–327. https://doi.org/10.1080/10494820.2019.1612448

Roig-Vila, R., & Moreno-Isac, V. (2020). El pensamiento computacional en Educación. Análisis bibliométrico y temático. Revista de Educación a Distancia (RED), 20(63). https://doi.org/10.6018/red.402621

Rojas, L. (2021). Factores asociados a los resultados de la evaluación de habilidades en pensamiento computacional https://repositorio.uniandes.edu.co/handle/1992/53272.

Román-González. M, Pérez G. & Jiménez F. (2015). test de pensamiento computacional: diseño y psicometría general [Computational Thinking test: design & general psychometry]. Congreso Internacional Sobre Aprendizaje, Innovación y Competitividad (CINAIC). Madrid, España. DOI:10.13140/RG.2.1.3056.5521

Román-González, M., Pérez-González, J.-C., Moreno-León, J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the Computational Thinking Test. International Journal of Child-Computer Interaction. doi: https://doi.org/10.1016/j.ijcci.2018.06.004

Romero, H., Cano, L., Charry, C. y Pardo, J. (2019). Deficiencia de adquisición de competencias mínimas en estudiantes de desarrollo de software: Inclusión & Desarrollo. 6. 85-97. https://doi.org/10.26620/uniminuto.inclusion.6.2.2019.85-97

Rondón, G. A. (2020). Propuesta para desarrollar habilidades de pensamiento computacional en estudiantes de décimo grado del Colegio Facundo Navas Mantilla. http://hdl.handle.net/20.500.12749/11689.

Rúales C. (2022). Inteligencia en un bolsillo: el pensamiento computacional. Innovación, TIC y gamificación Aportes desde el saber pedagógico para la educación del siglo XXI, *1*, 47-56. DOI:10.36737/9786287535084

Sanabria, J. (2022). Efecto en el desarrollo del pensamiento computacional a través de una actividad tecnológica escolar que promueve un sistema Mocap. Colombia: [Tesis para optar por el título de magister, Universidad distrital]. Recuperado de Repositorio institucional universidad distrital. https://repository.udistrital.edu.co/handle/11349/28500

Sarmiento B. M. (2018). Diseño de una propuesta metodológica para el desarrollo de competencias relacionadas con el pensamiento computacional. http://sedici.unlp.edu.ar/handle/10915/71938.

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2019). Designing Unplugged and Plugged Activities to Cultivate Computational Thinking: An Exploratory Study in Early Childhood Education. The Asia-Pacific Education Researcher. https://link.springer.com/article/10.1007/s40299-019-00478-w

Selby, C. & Woollard, J. (2013). Computational thinking: the developing definition. [Conference] Special Interest Group on Computer Science Education (SIGCSE). Atlanta.

Scratch (2022) Scratch - About. https://scratch.mit.edu/about

Silva C., Tonguino Q. & Mantilla G. (2020). El pensamiento computacional en la Resolución de Problemas Matemáticos en Básica Primaria a través de Computación Desconectada. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, 26., 2020, Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2020. p. 151-160. DOI: https://doi.org/10.5753/cbie.wie.2020.151.

Sinisterra, B. (2018). Creación de Materiales para Recursos Educativos Digitales Abiertos (REDA): Una Estrategia de Aprendizaje por Proyectos que aporta al Desarrollo de pensamiento computacional en el ciclo de Educación Media en la Escuela Normal Superior de Leticia. https://intellectum.unisabana.edu.co/handle/10818/33818.

Stephens M., Kadijevich D.M. (2019) Computational/Algorithmic Thinking. In: Lerman S. (eds) Encyclopedia of Mathematics Education. Springer, Cham. https://doi.org/10.1007/978-3-319-77487-9_100044-1

Sun, L., Hu, L., & Zhou, D. (2021). Single or combined? A Study on Programming to Promote Junior High School Students' Computational Thinking Skills. Journal of Educational Computing Research, 073563312110351. doi:10.1177/07356331211035182.

Sykora, C. (2021). Computational Thinking for all. ISTE. https://www.iste.org/es/explore/computational-thinking/computational-thinking-all

Tabesh, Y. (2018). Digital Pedagogy in Mathematical Learning. In: Kaiser, G., Forgasz, H., Graven, M., Kuzniak, A., Simmt, E., Xu, B. (eds) Invited Lectures from the 13th International Congress on Mathematical Education. ICME-13 Monographs. Springer, Cham. https://doi.org/10.1007/978-3-319-72170-5_37

Tellez, R., (2019). Pensamiento computacional: una competencia del siglo XXI. https://ojs.cepies.umsa.bo/index.php/RCV/article/view/23

Terroba, A., Ribera, P., & Lapresa, A. (2021). Cultivando el talento matemático en Educación Infantil mediante la resolución de problemas para favorecer el desarrollo del pensamiento computacional. Contextos Educativos. Revista De Educación, (28), 65–85. https://doi.org/10.18172/con.5008

Tonbuloğlu, B. & Tonbuloğlu, I. (2019). The Effect of Unplugged Coding Activities on Computational Thinking Skills of Middle School Students. Informatics in Education. 18. 403-426. 10.15388/infedu.2019.19.

Tyrén, M., Carlborg, N., Heath, C., and Eriksson, E. (2018). "Considerations and technical pitfalls for teaching computational thinking with BBC micro:bit," in Proceedings of the Conference on Creativity and Making in Education – FabLearn Europe'18 (Trondheim:ACM Press), 81–86.

Velásquez, M. (2021). Desarrollo del Pensamiento Computacional en la primera infancia: https://manglar.uninorte.edu.co/handle/10584/10267#page=1

Vieira, C., Gómez, M., Canu, M., Duque M. (2019). Pensamiento computacional- STEM-Academia. https://www.pequenoscientificos.org/uploads/7/6/6/4/76644211/pensamiento_computacional.pdf

Villa-Ochoa, J. y Castrillón Y. (2020). Temas y tendencias de investigación en América Latina a la luz del Pensamiento Computacional en Educación Superior. En Políticas, Universidad e

innovación: retos y perspectivas (pp. 235-248). Librería Bosch. https://dialnet.unirioja.es/servlet/articulo?codigo=7754299

Vostinar, P., & Kneznik, J. (2020). Experience with teaching with BBC *Micro:bit*. 2020 IEEE Global Engineering Education Conference (EDUCON). doi: 10.1109/EDUCON45650.2020.9125278

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2015). Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology, 25(1), 127–147. doi: https://doi.org/10.1007/s10956-015-9581-5

Weintrop, D, Killen, H., & Franke, B. (2018). Blocks or Text? How programming language modality makes a difference in assessing underrepresented populations. Proc. of the International Conference on the Learning Sciences 2018, 328–335. https://www.terpconnect.umd.edu/~weintrop/papers/WeintropKillenFranke_ICLS2018.pdf

Weintrop, D (2021). The role of block-based programming in computer science education. In Understanding computing education (Vol 1). Proceedings of the Raspberry Pi Foundation Research Seminar series. https://www.terpconnect.umd.edu/~weintrop/papers/Weintrop_RaspPi_2021.pdf

Wing, J. M. (2006). Computational Thinking. View Point. Comunication of ACM. 49(3). 35. Disponible en http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf

Wing, J. M. (2008). Computational thinking and thinking about computing. Physical, and Engineering Sciences, 366(1881), 3717–3725. http://doi.org/10.1098/rsta.2008.01.18

Wing, J. M. (2011). Computational thinking. 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2011) (pp. 3-3). IEEE. doi: https://doi.org/10.1109/VLHCC.2011.6070404

Zapata-Ros, M. (2020). El pensamiento computacional, una cuarta competencia clave planteada por la nueva alfabetización. DOI:10.13140/RG.2.2.15575.91049

Zapata-Ros, M. (2019). Pensamiento computacional desenchufado. Universidad de Murcia, Murcia (España). http://orcid.org/0000-0003-4185-5024. DOI:10.13140/RG.2.2.12945.48481

Zapata-Ros, M. (2015). Pensamiento Computacional: Una nueva alfabetización digital. Revista de Educación a Distancia (RED). (46). https://revistas.um.es/red/article/view/240321

Zhang, L. C. & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. Computers and Education, 141(June), 103607. https://doi.org/10.1016/j.compedu.2019.103607

Zhang, S. & Cui, C. (2021). Implementing Blended Learning in K-12 Programming Course: Lesson Design and Student Feedback. 10.1109/ISEC52395.2021.9764091. https://ieeexplore.ieee.org/document/9764091

Anexo 1. Sesiones desarrolladas durante las actividades conectadas.

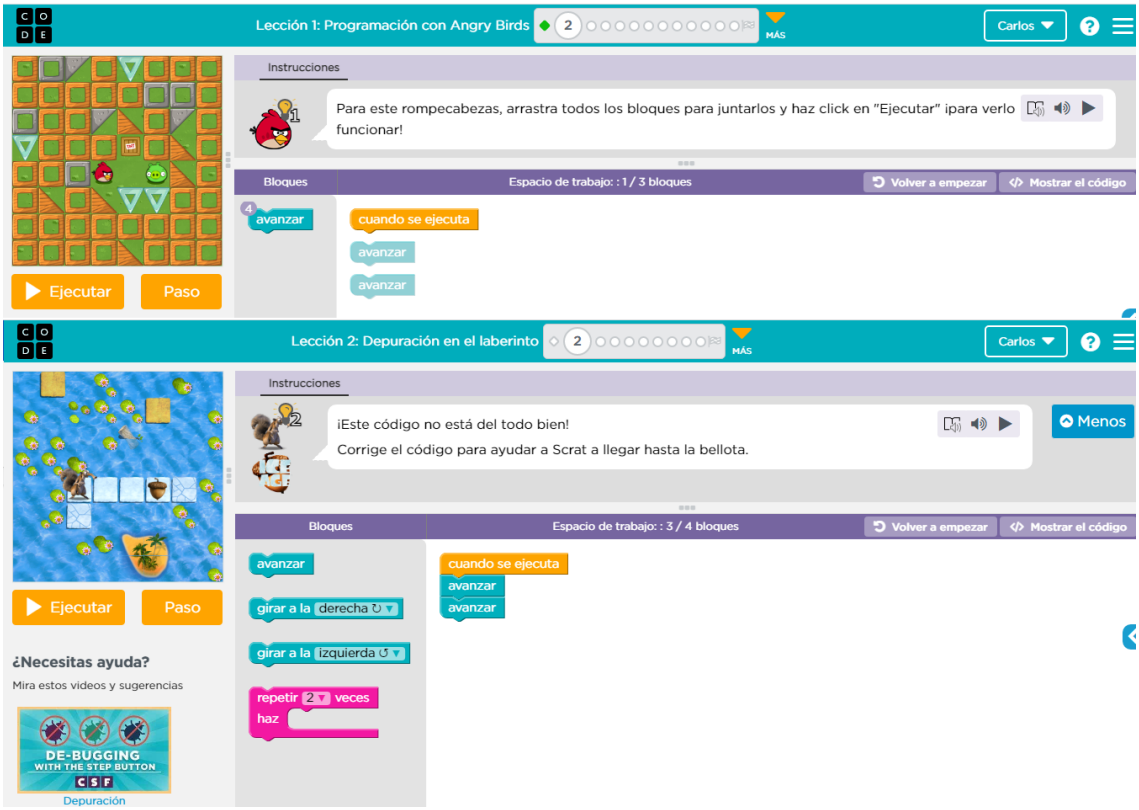
Sesión conectada 1 - secuencialidad y depuración

En esta sesión los estudiantes realizaron las actividades propuestas en las lecciones 1 y 2, las cuales introducen los conceptos computacionales de secuencialidad y depuración. En la lección 1, programación con *Angry Birds*, se buscó que los estudiantes desarrollaran habilidades de programación basada en bloques, mientras que en la lección 2, depuración en el laberinto, trabajaron el concepto de depuración el cual es importante para encontrar los errores en sus propios programas. Esto lleva a los estudiantes a reconocer los problemas y a superarlos mientras desarrollan habilidades de pensamiento crítico y resolución de problemas. Otros aspectos relevantes de estas unidades se resumen en la Tabla 1.

Tabla 1

Sesión 1 – lecciones 1 y 2 grupo conectadas

	Lección 1 - Programación con *Angry Birds*.	**Lección 2 - Depuración en el laberinto.**
Objetivos	- Traducir movimientos en una serie de comandos - Identificar y localizar errores en un programa.	- Modificar un programa existente para solucionar errores. - Predecir dónde fallará un programa. - Reflexionar sobre el proceso de depuración de manera apropiada para cada edad.
Conceptos computacionales	Secuencialidad	Depuración
Descripción	A partir de los personajes del juego Angry Birds, los estudiantes desarrollarán algoritmos secuenciales que les permitirán mover al pájaro desde un lado del laberinto hacia el cerdo que se encuentra al otro lado. Para ello, apilarán bloques de código en una secuencia lineal, lo que hará que el pájaro se mueva derecho, y luego gire a la izquierda o a la derecha.	Deberán revisar el código existente para identificar los errores, tales como bucles incorrectos, bloques que faltan, bloques de más y bloques que no funcionan.
Tiempo	38 min	5 min
Recursos		



Nota: Tomado del plan de estudios de la sesión secuencias. Curso Express 2021 (Code.org) https://studio.code.org/s/express-2021/lessons/1/levels/2, https://studio.code.org/s/express-2021/lessons/2/levels/2

Sesión conectada 2 – Secuencialidad, depuración y variables

En la segunda sesión los estudiantes realizaron las actividades propuestas en las lecciones 3, 4 y 24, las cuales introducen los conceptos de secuencialidad, depuración y variables. En la lección 3, recoger tesoros con Laurel, los estudiantes continúan practicando sus habilidades de programación agrupando las instrucciones en un orden específico para comprender cómo un ordenador navega por las instrucciones y en qué orden. Aquí el uso de nuevos personajes con un objetivo de rompecabezas diferente ayudará a los estudiantes a ampliar su experiencia con la secuenciación y los algoritmos en la programación.

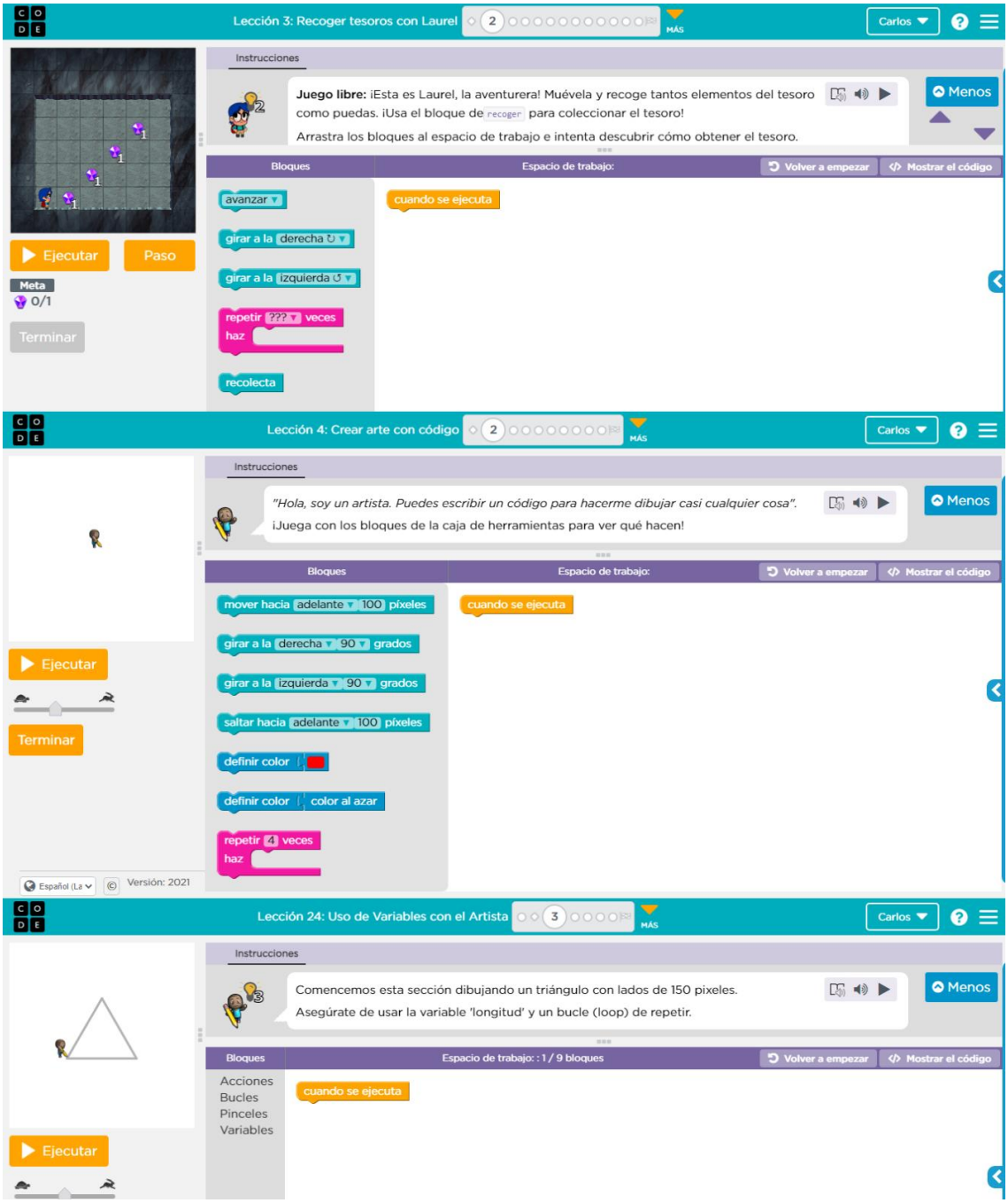
La lección 4, crear arte con código**,** parte de la experiencia previa de los estudiantes en secuenciación, consolidando su conocimiento mediante la introducción de nuevos bloques y objetivos. En este caso, los estudiantes aprenden más sobre píxeles y ángulos, mientras practican sus habilidades de secuenciación, al programar al Artista para dibujar un cuadrado.

En la lección 24, uso de variables con el artista, se introducen el concepto de variable, el cual es vital para crear un código dinámico que permita a un programa crecer en función de cualquier número de modificaciones potenciales. Esta etapa refuerza el uso de *variables*, utilizando las capacidades más básicas de configuración y uso de las mismas. Otros aspectos relevantes de estas unidades se resumen en la Tabla 2.

Tabla 2

Sesión 2 – Lecciones 3, 4 y 24 grupo conectadas

	Lección 3 - Recoger tesoros con Laurel	**Lección 4 - Crear arte con código**	**Lección 24 - Uso de variables con el artista**
Objetivos	- Desarrollar habilidades de resolución de problemas y pensamiento crítico mediante la revisión de las prácticas de depuración. - Ordenar comandos de movimiento como pasos secuenciales en un programa. - Representar un algoritmo como un programa de computadora.	- Divide las formas complejas en partes simples. - Crear un programa para completar una imagen mediante pasos secuenciales.	- Asignar valores a las variables existentes. - Usar variables para cambiar valores dentro de un bucle. - Utilizar variables en lugar de valores repetitivos dentro de un programa.
Conceptos computacionales	Secuencialidad	Depuración	Variables
Descripción	En esta serie de desafíos, los estudiantes continuarán desarrollando su comprensión de los algoritmos y la depuración. Crearán algoritmos secuenciales para que el nuevo personaje Laurel la Aventurera pueda recoger el tesoro mientras recorre el camino.	En esta lección, los estudiantes tomarán control del Artista para completar los dibujos en pantalla. En esta etapa del Artista, los estudiantes podrán crear imágenes de mayor complejidad mediante nuevos bloques, tales como como: avanzar 100 pixeles y girar 90 grados a la derecha.	En esta lección, los alumnos explorarán la creación de diseños repetitivos mediante variables en el entorno de Artista. Los alumnos aprenderán cómo usar las variables para que el código sea más fácil de escribir y leer. Luego de los desafíos guiados, los alumnos terminarán en un nivel de juego libre para demostrar lo que han aprendido y crear sus propios diseños.
Tiempo	50 min	55 min	40 min
Recursos			



Nota: Tomado del plan de estudios de la sesión secuencias. Curso Express 2021 (Code.org). https://studio.code.org/s/express-2021/lessons/3/levels/2, https://studio.code.org/s/express-2021/lessons/4/levels/2, https://studio.code.org/s/express-2021/lessons/24/levels/3

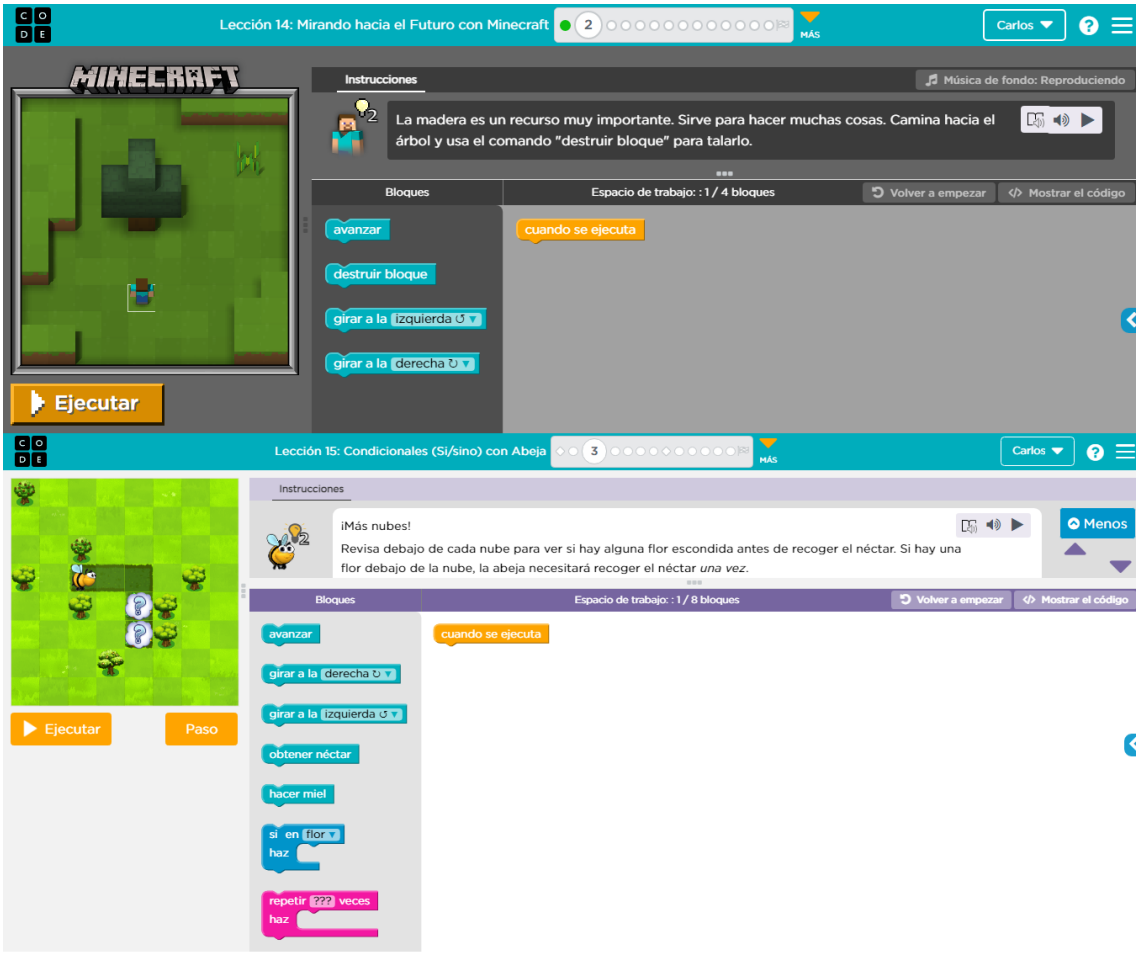
Sesión conectada 3 – Condicionales

En la tercera sesión los estudiantes realizaron las actividades propuestas en las lecciones 14 y 15 que introducen los conceptos de condicionales. En la lección 14, mirando hacia el futuro con *Minecraft* y la lección 15, condicionales con abeja, los estudiantes comenzarán a codificar con condicionales, permitiéndoles escribir un código que funcione de forma diferente dependiendo de las condiciones específicas que el programa encuentre. Otros aspectos de estas unidades se resumen en la Tabla 3.

Tabla 3

Sesión 3 – Lecciones 14 y 15 grupo conectadas

	Lección 14 - Mirando hacia el futuro con *Minecraft.*	**Lección 15 - Condicionales con abeja**
Objetivos	- Definir cuándo deben ejecutarse ciertas partes de un programa y cuándo no. - Determinar si se cumple una condicional basándose en unos criterios.	- Resolver desafíos mediante una combinación de secuencias en bucle y condicionales. - Convertir sentencias condicionales de lenguaje hablado en un programa.
Conceptos computacionales	Condicionales	Condicionales
Descripción	Este conjunto de rompecabezas trabajará para solidificar y desarrollar el conocimiento de los bucles e introducir condicionales. Al combinar estos dos conceptos, los estudiantes podrán explorar el potencial de crear programas divertidos e innovadores en un entorno nuevo y emocionante.	En esta lección practicarán el uso de condicionales en sus programas. Los bloques if/else permitirán un programa más flexible. La abeja solo recogerá néctar si hay una flor o hacer miel si hay panal de abeja.
Tiempo	45 min	55 min
Recursos		



Nota: Tomado del plan de estudios de la sesión condicionales. Curso Express 2021 (Code.org). https://studio.code.org/s/express-2021/lessons/14/levels/2, https://studio.code.org/s/express-2021/lessons/15/levels/2.

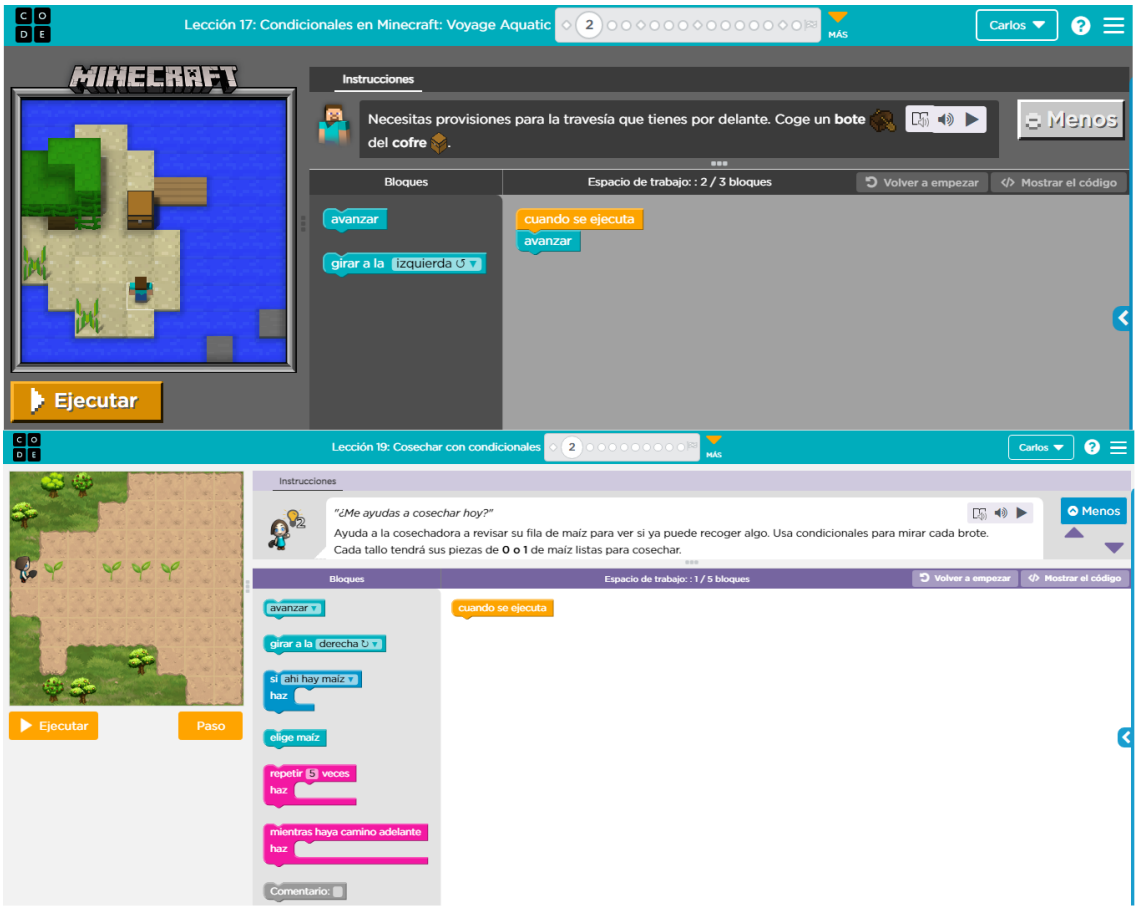
Sesión conectada 4 – Bucles y condicionales

En la cuarta sesión se realizaron las actividades propuestas en las lecciones 17 y 19, introduciendo los conceptos de bucles y condicionales. En la lección 17, condicionales en *Minecraft: Voyage Aquatic*, los estudiantes tendrán la oportunidad de practicar las ideas que han aprendido hasta este punto e iniciarán con el manejo de condicionales y bucles, mientras que en la lección 19, cosechar con condicionales, practicarán con los bucles mientras que-*while* y bucle hasta, así como las declaraciones *if/else*. Gracias a estos nuevos bloques aprenderán a escribir código más complejo y flexible. Otros aspectos relevantes de estas unidades se resumen en la Tabla 4.

Tabla 4

Sesión 4 – Lecciones 17 y 19 grupo conectadas

	Lección 17 - Condicionales en *Minecraft: Voyage Aquatic*	**Lección 19 - Cosechar con condicionales**
Objetivos	-Definir cuándo deben ejecutarse ciertas partes de un programa y cuándo no. - Determinar si se cumple una condicional basándose en unos criterios.	-Anidar condicionales para analizar condiciones de valor múltiple mediante la lógica, si, si no si, si no. - Integrar un bucle y una sentencia condicional.
Conceptos computacionales	Bucles Condicionales	Bucles Condicionales
Descripción	Este conjunto de rompecabezas trabajará para solidificar y desarrollar el conocimiento de los bucles y los condicionales. Al combinar estos dos conceptos, los estudiantes podrán explorar el potencial de crear programas divertidos e innovadores en un entorno nuevo y emocionante.	En la lección anterior, los/as estudiantes sólo usaron condicionales para moverse alrededor de un laberinto. En esta lección, los/as estudiantes usarán condicionales para ayudar al agricultor a saber cuándo cosechar cultivos. Los nuevos patrones surgirán y los estudiantes usarán creatividad y pensamiento lógico para determinar las condiciones en las que el código debe ejecutarse y repetirse.
Tiempo	60 min	50 min
Recursos		



Nota: Tomado del plan de estudios de la sesión condicionales. Curso Express 2021 (Code.org). https://studio.code.org/s/express-2021/lessons/17/levels/2, https://studio.code.org/s/express-2021/lessons/19/levels/2

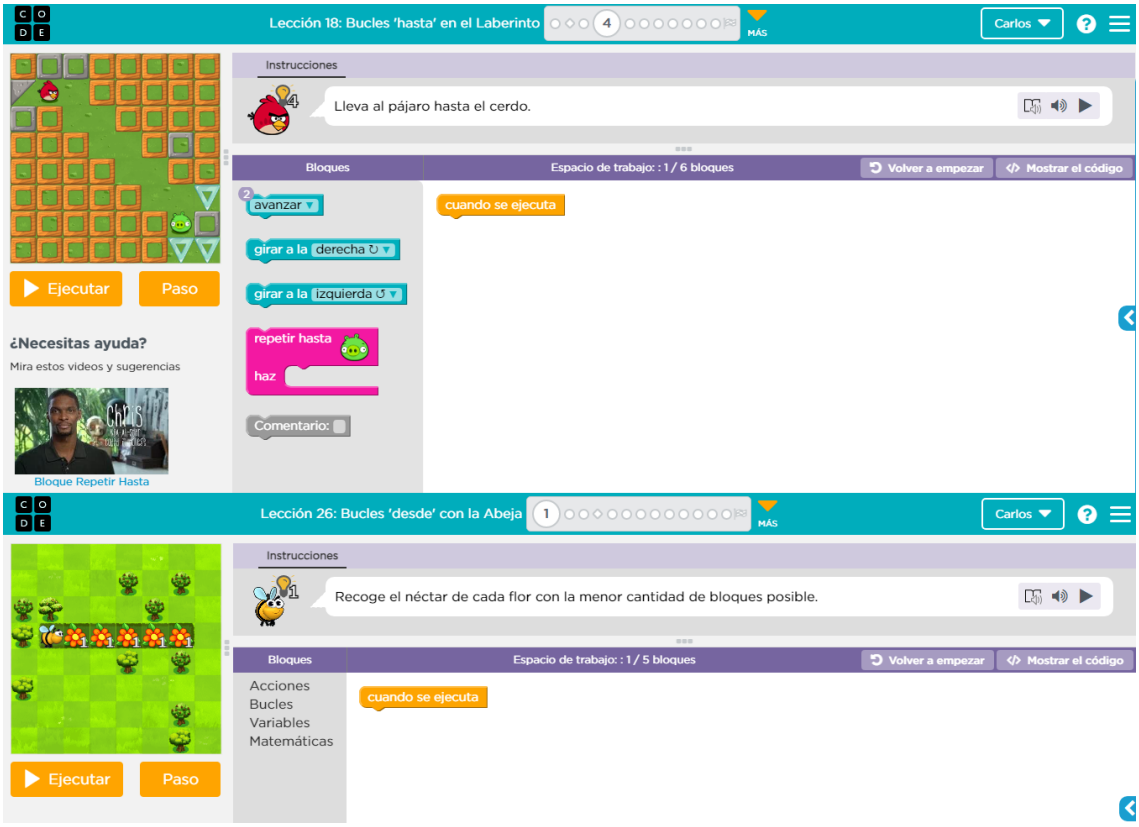
Sesión conectada 5 – Bucles

En la quinta sesión, los estudiantes realizaron las actividades propuestas para las lecciones 18 y 26, orientadas principalmente a introducir el concepto de Bucles. En la lección 18, bucles 'mientras' en el laberinto**,** los estudiantes aprenderán acerca de los bucles hasta, creando programas en los cuales el personaje repetirá acciones hasta que llegue al punto de detención deseado. Como complemento, la lección 26, bucles ´desde´ con la abeja, usa una variable gradual para solucionar desafíos más complejos. Allí los estudiantes repasarán bucles de lecciones anteriores y luego abordarán su uso para resolver problemas difíciles de manera más efectiva. Los demás aspectos relevantes de estas unidades se resumen en la Tabla 5.

Tabla 5

Sesión 5 – Lecciones 18 y 26 grupo conectadas

	Lección 18 - Bucles 'mientras' en el laberinto	**Lección 26 - Bucles ´desde´ con la abeja**
Objetivos	- Desarrollar programas con la comprensión de varias estrategias para implementar condicionales. - Convertir sentencias condicionales de lenguaje hablado y bucles en un programa.	- Determine el valor inicial, el valor de detención y el valor de escalón para un bucle *for*. - Reconoce cuándo usar un bucle *for* y cuándo usar otros bucles, como los bucles *repeat* y while.
Conceptos computacionales	Bucles Condicionales	Bucles
Descripción	Este conjunto de rompecabezas trabajará para solidificar y desarrollar el conocimiento de los bucles y los condicionales. Al combinar estos dos conceptos, los estudiantes podrán explorar el potencial de crear programas divertidos e innovadores en un entorno nuevo y emocionante.	El concepto de los bucles *for*, mejorará el aprendizaje de otros conceptos importantes como variables y parámetros. Los estudiantes tendrán mucha práctica para pensar críticamente en los problemas determinando los valores iniciales, finales y escalonados para cada ciclo *for*.
Tiempo	60 min	60 min
Recursos		



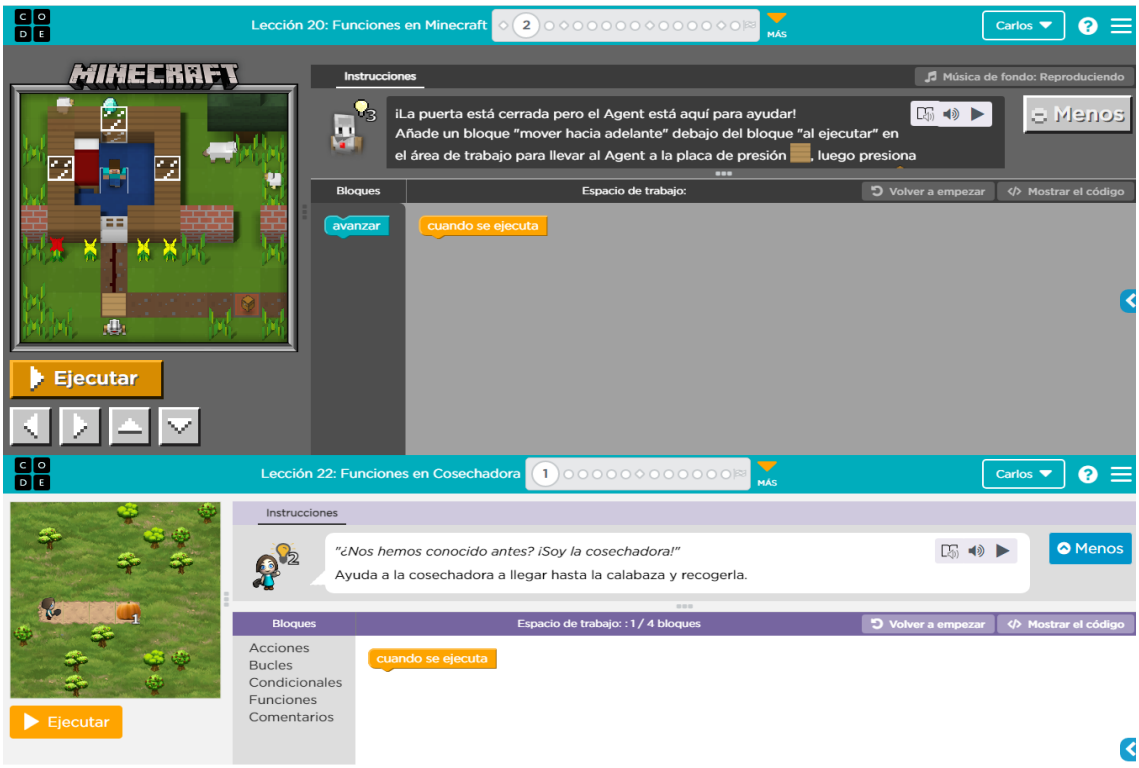
Nota: Tomado del plan de estudios de la sesión condicionales y bucles para. Curso Express 2021 (Code.org). https://studio.code.org/s/express-2021/lessons/18/levels/3, https://studio.code.org/s/express-2021/lessons/26/levels/3

Sesión conectada 6 – Funciones

En la sexta sesión, los estudiantes realizaron las actividades propuestas para las lecciones 20 y 22, las cuales introducen los conceptos de *funciones*. Mientras que en la lección 20, funciones en *Minecraft*, los estudiantes comprendieron de qué manera las funciones pueden ser útiles en una divertida e interactiva aventura de *Minecraft*, la lección 22, funciones en cosechadora, impulsó aún más a los alumnos a utilizar las funciones de forma más creativa. En ambas lecciones aprendieron que existen diversas maneras de abordar un problema, algunas más eficientes que otras, aumentando sus habilidades en la resolución de problemas y el pensamiento crítico. Otros aspectos relevantes de estas unidades se resumen en la Tabla 6.

Tabla 6

Sesión 6 – Lecciones 20 y 22 grupo conectadas

	Lección 20 - Funciones en *Minecraft*	**Lección 22 - Funciones en cosechadora**
Objetivos	- Usar funciones para simplificar programas complejos. -Utilizar funciones predeterminadas para completar tareas que se repiten mucho.	- Reconocer cuándo una función podría ayudar a simplificar un programa. - Utilizar funciones predeterminadas para completar tareas que se repiten mucho.
Conceptos computacionales	Funciones	Funciones
Descripción	Los alumnos descubrirán la versatilidad de la programación practicando funciones en diferentes entornos. Aquí, los alumnos reconocerán patrones reutilizables y podrán incorporar bloques con nombre para llamar a funciones predefinidas.	Los estudiantes practicaron cómo crear diseños maravillosos en el Artista y recorrer laberintos en la Abeja, pero hoy usarán las funciones para cosechar cultivos en la Granjera. Esta lección les permitirá usar las funciones de nuevas maneras y combinarlas con los bucles mientras y las sentencias si / si no.
Tiempo	60 min	55 min
Recursos	

Nota: Tomado del plan de estudios de la sesión funciones curso Express 2021 (Code.org). https://studio.code.org/s/express-2021/lessons/20/levels/2, https://studio.code.org/s/express-2021/lessons/22/levels/2

Tabla 7.

Rúbrica de evaluación para las actividades conectadas semana 1.

Lección 1 y 2	SUPERIOR (10)	ALTO (8)	MEDIO (6)	BÁSICO (4)	BAJO (2)
REPRESENTACIÓN DE DATOS	Modela la forma en que los programas almacenan y manipulan datos utilizando números u otros símbolos para representar información	Identifica y manipula los datos en un programa para representar información	Identifica y manipula algunos datos en un programa para representar información	Presenta dificultades para identificar datos en un programa para representar información.	No Identifica, ni manipula ningún dato en un programa para representar información
PENSAMIENTO ALGORÍTMICO	Crea una secuencia optima de instrucciones recurriendo a una cantidad de bloques menor a la sugerida en todas las lecciones	Crea una secuencia precisa de instrucciones recurriendo únicamente a la cantidad de bloques sugerida en cada lección	Crea una secuencia de instrucciones usando una cantidad mayor de bloques a la sugerida en algunas lecciones	Genera una secuencia incompleta de instrucciones de bloques en algunas lecciones	No genera una secuencia de instrucciones usando los bloques sugeridos en las lecciones
DESCOMPOSICIÓN DEL PROBLEMA	Desglosa los pasos necesarios para descomponer un problema en una secuencia coherente de instrucciones y los articula en una solución adecuada.	Plantea los pasos necesarios para descomponer un problema en una secuencia coherente de instrucciones e implementar una única solución.	Identifica algunos pasos requeridos para descomponer un problema y generar una secuencia de instrucciones que se acerca a una posible solución.	Propone algunos pasos requeridos para descomponer un problema y generar una secuencia de instrucciones que componen una solución.	No logra identificar ni plantear los pasos requeridos para descomponer un problema mediante una secuencia de instrucciones que conlleven a la solución.
PRUEBA Y DEPURACIÓN	Identifica y corrige todos los errores en algoritmos o programas que incluyen instrucciones secuenciales.	Identifica y corrige algunos errores en algoritmos o programas que incluyen instrucciones secuenciales.	Solo identifica errores en algoritmos o programas que incluyen instrucciones secuenciales, no logra realizar la corrección.	Identifica al menos un error en los algoritmos o programas que incluyen instrucciones secuenciales.	No logra identificar ningún error en los algoritmos o programas
CREACIÓN DE ARTEFACTOS COMPUTACIONALES	Completa todos los ejercicios de las lecciones incluyendo los desafíos	Completa la mayoría de los ejercicios de las lecciones y algunos desafíos	Completa únicamente los ejercicios de las lecciones más no los desafíos	Trabaja en algunos ejercicios de la lección, más no los completa en su totalidad.	No completa los ejercicios ni los desafíos propuestos en las lecciones

Anexo 2. Sesiones desarrolladas durante las actividades desconectadas.

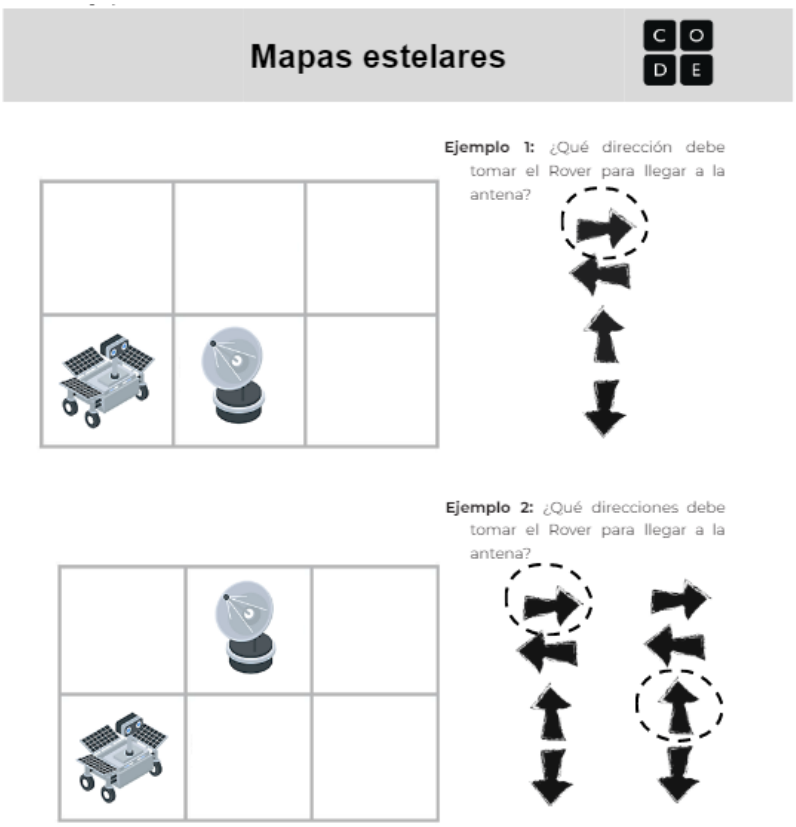
Sesión desconectada 1: Mapas estelares y mis amigos robóticos Jr.

En la actividad mapas estelares los estudiantes realizaron las actividades propuestas para la lección 3, permitiendo que la transición desde los algoritmos hacia la programación fuera más corta para entender la diferencia entre planificar una secuencia y codificarla en un lenguaje apropiado. Otros aspectos relevantes de esta actividad se resumen en la Tabla 8.

Tabla 8

Actividad 1 - Lección 3 Actividades desconectadas Code.org

Actividad 1	**Descripción**
Objetivos	- Decodificar y ejecutar un programa creado por otra persona. - Identificar y solucionar fallos o errores en las instrucciones secuenciadas. - Convertir un algoritmo en un programa.
Conceptos computacionales	- Secuencialidad
Descripción	Esta actividad ayudo a los estudiantes a adquirir experiencia en la lectura y escritura de código abreviado.
Tiempo	53 min
Recursos	



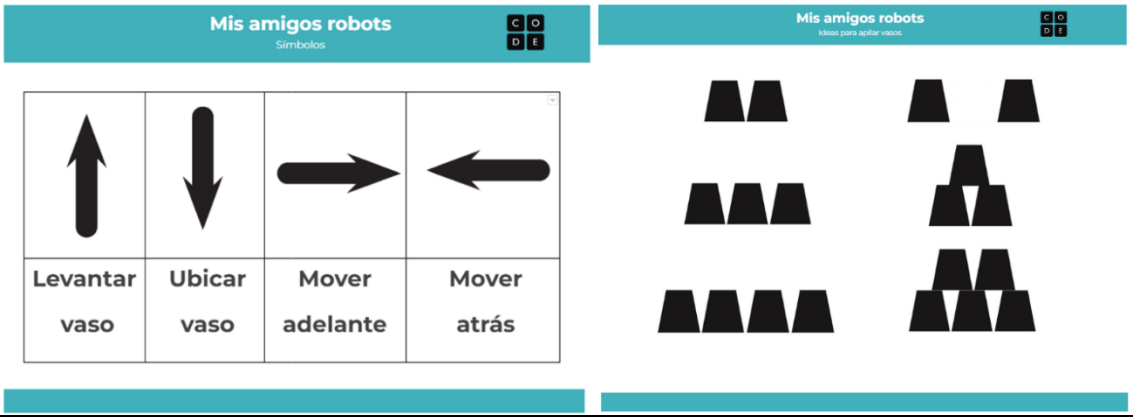
Nota: Adaptado de Code.org [*https://studio.code.org/s/coursea-2021/lessons/3*](https://studio.code.org/s/coursea-2021/lessons/3)

Para la actividad mis amigos robóticos Jr se organizó a la clase por equipos con la tarea de hacer que un compañero actúe como un “robot” y apile vasos plásticos de una forma específica. Esta actividad sentó las bases para entender como programar algo a partir de ciertas instrucciones, mientras que aprendieron la importancia de definirlas en un algoritmo. Otros aspectos relevantes de esta actividad se resumen en la Tabla 9.

Tabla 9

Actividad 2 – Lección 2 Actividades desconectadas Code.org

Actividad 2	**Descripción**
Objetivos	- Prestar atención a la precisión a la hora de crear instrucciones - Identificar y solucionar fallos o errores en las instrucciones secuenciadas
Conceptos computacionales	- Secuencialidad - Depuración
Descripción	Mediante un conjunto de símbolos en lugar de código, los estudiantes diseñaron algoritmos para indicar a un "robot" que apile tazas en diferentes patrones. Los estudiantes se turnaron para cumplir el rol del robot y respondieron únicamente al algoritmo definido por sus compañeros. Este segmento enseño a los estudiantes la conexión entre los símbolos y las

	acciones, la diferencia entre un algoritmo y un programa, y lo valiosa que es la depuración.
Tiempo	45 min
Recursos	

Nota: Tomado de Code.org https://studio.code.org/s/coursec-2021/lessons/2

Sesión desconectada 2: Programación por relevos.

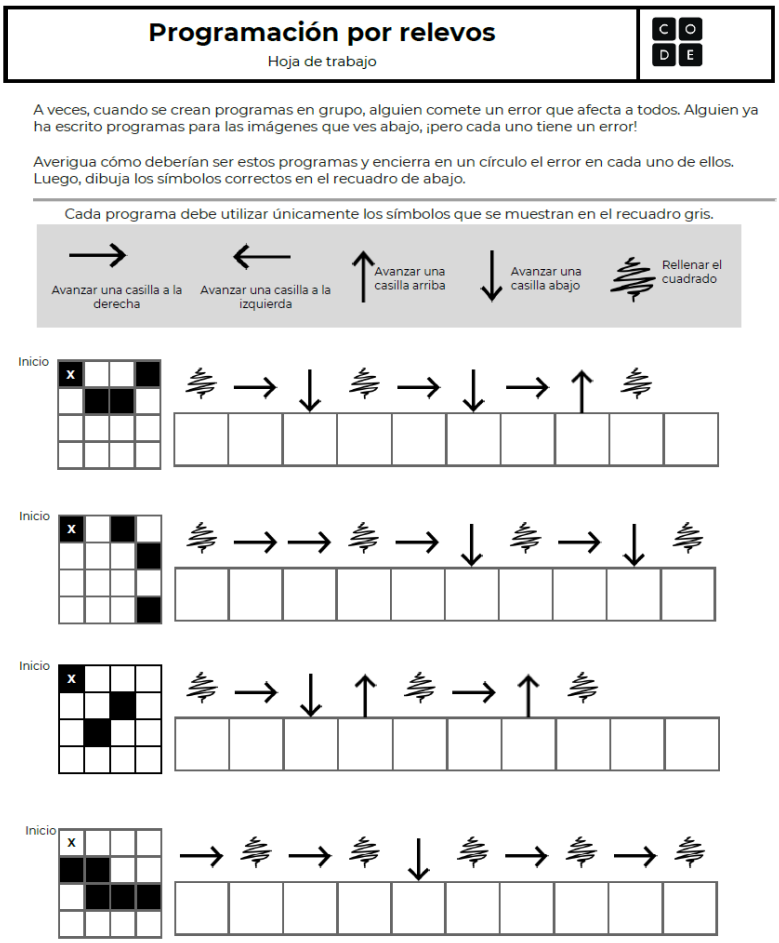
Para esta actividad se seleccionó la lección 4, con el propósito de enseñar a los estudiantes que deben administrar su tiempo cuidadosamente, evitando cometer errores al programar y evitando comprometer mucho tiempo al hacerlo. Los demás aspectos relacionados con esta actividad se resumen en la Tabla 10.

Tabla 10

Actividad 3 - Lección 4 - Actividades desconectadas Code.org

Actividad 3	**Descripción**
Objetivos	- Definir ideas usando código y símbolos. - Identificar las señales de frustración. - Verificar el trabajo hecho por compañeros.
Conceptos computacionales	- Direcciones Básicas - Depuración
Descripción	Esta lección de contexto comenzó con una breve lección sobre la depuración y la constancia, y luego paso rápidamente a una carrera contrarreloj en la que los alumnos se dividieron en equipos y trabajaron juntos para escribir la instrucción de un programa a la vez.
Tiempo	50 min

Recursos



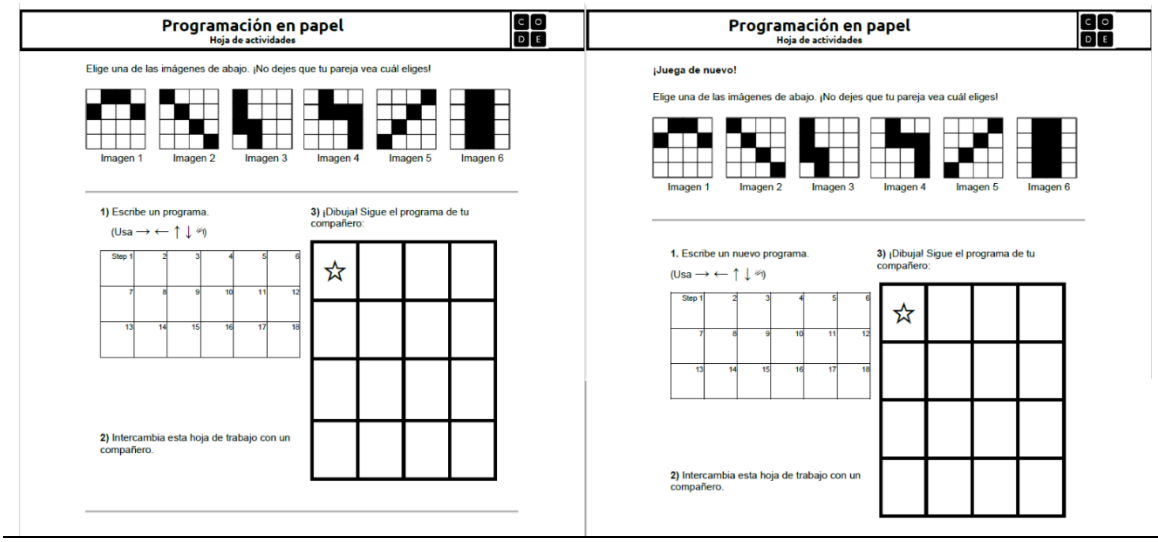
Nota: tomado de Code.org https://studio.code.org/s/coursed-2021/lessons/4

Sesión desconectada 3: Programación en papel e historias en blanco.

En la actividad programación en papel los estudiantes realizaron las actividades propuestas para la lección 2 permitiéndoles desarrollar su habilidad de pensamiento crítico e introduciendo los conceptos de secuenciación y algoritmos, que fueron usados a lo largo del curso, ayudando a familiarizarlos con el concepto de programación y sentar las bases para desarrollar un algoritmo y codificarlo en un programa. Los demás aspectos relacionados con esta actividad se resumen en la Tabla 11.

Tabla 11

Actividad 4 - Lección 2 - Actividades desconectadas Code.org

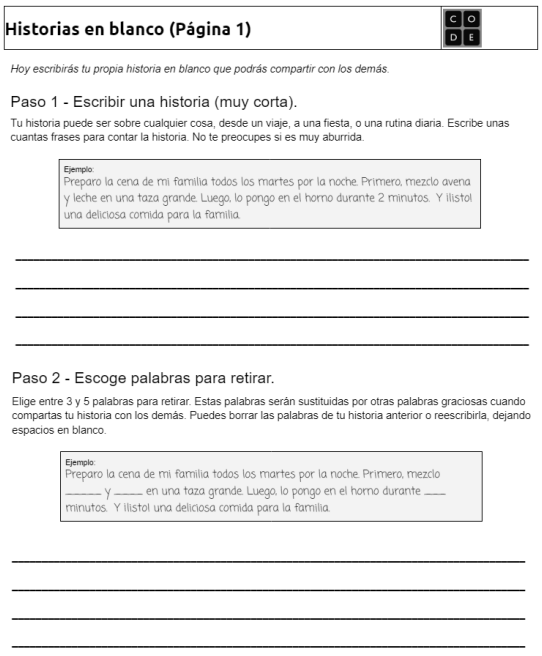
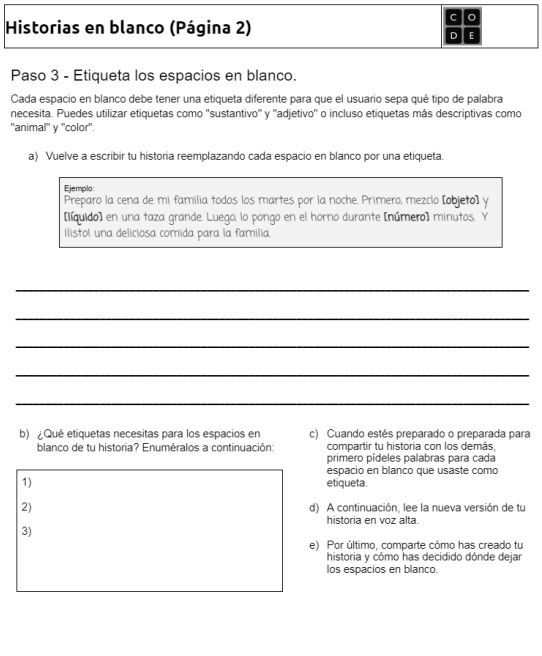
Actividad 4	**Descripción**
Objetivos	- Explicar las limitaciones de traducir problemas desde el lenguaje humano al lenguaje de las máquinas. - Restructurar una secuencia de pasos en un programa codificado.
Conceptos computacionales	- Direcciones Básicas
Descripción	Al "programarse" entre sí para dibujar imágenes, los estudiantes obtienen una oportunidad de experimentar algunos de los conceptos centrales de programar de una manera divertida y accesible. La clase comenzó haciendo que los estudiantes usen símbolos para instruirse entre ellos para colorear cuadrados en papel cuadriculado en un intento por producir una imagen existente.
Tiempo	65 min
Recursos	

Nota: Tomado de Code.org https://studio.code.org/s/coursed-2021/lessons/2

En la actividad historias en blanco actividad los estudiantes realizaron las actividades de la lección 6, proporcionándoles un contexto más cercano a sus vivencias para comprender cómo se escogen y funcionan las variables en un programa y como se recurren a ellas cada vez que vuelve a programar. Los aspectos relacionados con esta actividad se resumen en la Tabla 12.

Tabla 12

Actividad 5 - Lección 6 - Actividades desconectadas Code.org

Actividad 5	**Descripción**
Objetivos	- Asignar un valor a una variable - Llama a una variable varias veces en un programa. - Declarar una variable - Determinar la relación entre cómo se define, almacena y recupera una variable cuando se ejecuta en un programa.
Conceptos computacionales	- Variables
Descripción	En esta actividad, los alumnos utilizaron historias para rellenar los espacios en blanco como contexto para comprender cómo los ordenadores reciben y almacenan las entradas de un usuario, para luego utilizarlas cuando se ejecuta un programa.
Tiempo	55 min
Recursos	 

Nota: tomado de Code.org https://studio.code.org/s/coursef-2021/lessons/6

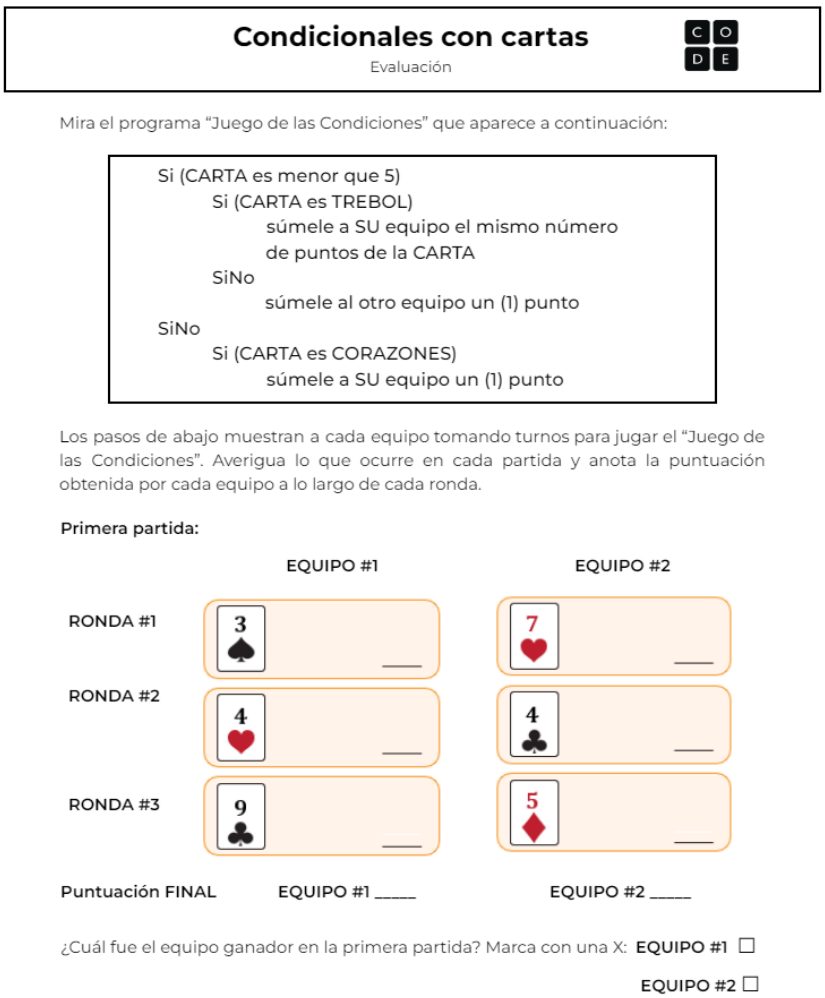
Sesión desconectada 4: Condicionales con cartas.

Para trabajar el concepto de condicionales se seleccionó la lección 12 gracias a que combina la informática con el mundo real, basándose en la capacidad para entender si una condición es verdadera o falsa. Aquí los estudiantes aprendieron a utilizar las sentencias "si" para declarar cuándo debe ejecutarse un determinado comando, así como las sentencias "si /

sino" para declarar qué debe ejecutarse en caso contrario. Los demás aspectos relacionados con esta actividad se resumen en la Tabla 13.

Tabla 13

Actividad 6 - Lección 12 Actividades desconectadas Code.org

Actividad 6	**Descripción**
Objetivos	- Definir cuándo deben ejecutarse ciertas partes de un programa y cuándo no. - Determinar si se cumple una condicional basándose en unos criterios. - Recorrer un programa y predecir el resultado, teniendo en cuenta un conjunto de entradas.
Conceptos computacionales	- Condicionales
Descripción	En esta lección de contexto, los alumnos escribieron sentencias condicionales (si/sino) para establecer las reglas de juegos de cartas sencillos.
Tiempo	55 min
Recursos	

Nota: Tomado de Code.org https://studio.code.org/s/coursed-2021/lessons/12

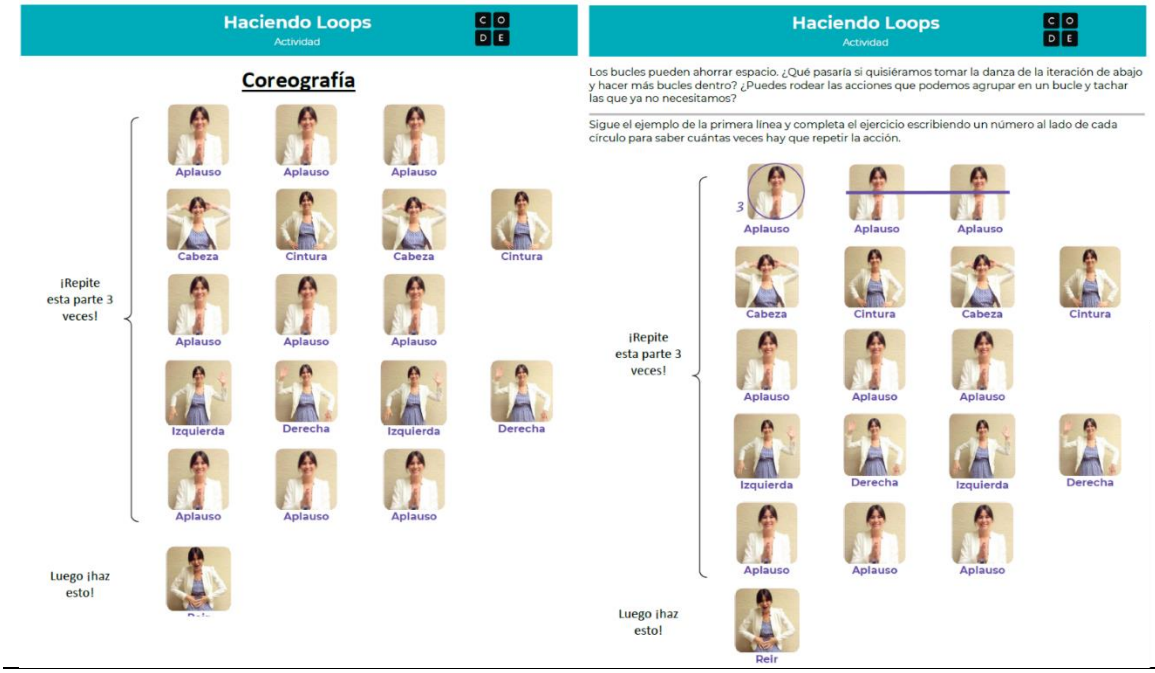
Sesión desconectada 5: Haciendo *Loops* y *Loops* con mapas estelares.

En esta actividad se señaló como un conjunto lineal de instrucciones, con frecuencia, incluye patrones que se repiten múltiples veces y a medida que se escriben programas más complejos e interesantes, es común duplicar manualmente parte del código, lo cual se vuelve engorroso e ineficiente. Para que los estudiantes puedan escribir programas más potentes, se recurrió a los bucles como estructuras que se salen de ese patrón lineal y permiten un código de forma que se repita. Es por esto que mediante la lección 6, los estudiantes identificaron patrones en el movimiento físico para buscar patrones que se repiten en un código. Otros aspectos relacionados con esta actividad se resumen en la Tabla 14.

Tabla 14

Actividad 7 - Lección 6 - Actividades desconectadas Code.org

Actividad 7	**Descripción**
Objetivos	- Convertir una serie de acciones en un solo bucle. - Repetir las acciones iniciadas por el instructor. - Convertir un programa de imágenes en un baile del mundo real.
Conceptos computacionales	- Bucles
Descripción	A medida que empezamos a escribir programas más largos y más interesantes, nuestro código tendrá más repeticiones. En esta lección, los estudiantes aprendieron a usar los bucles para mirar los patrones repetidos de movimiento en un baile y comunicar más fácilmente aquellas instrucciones que tienen mucha repetición.
Tiempo	45 min
Recursos	



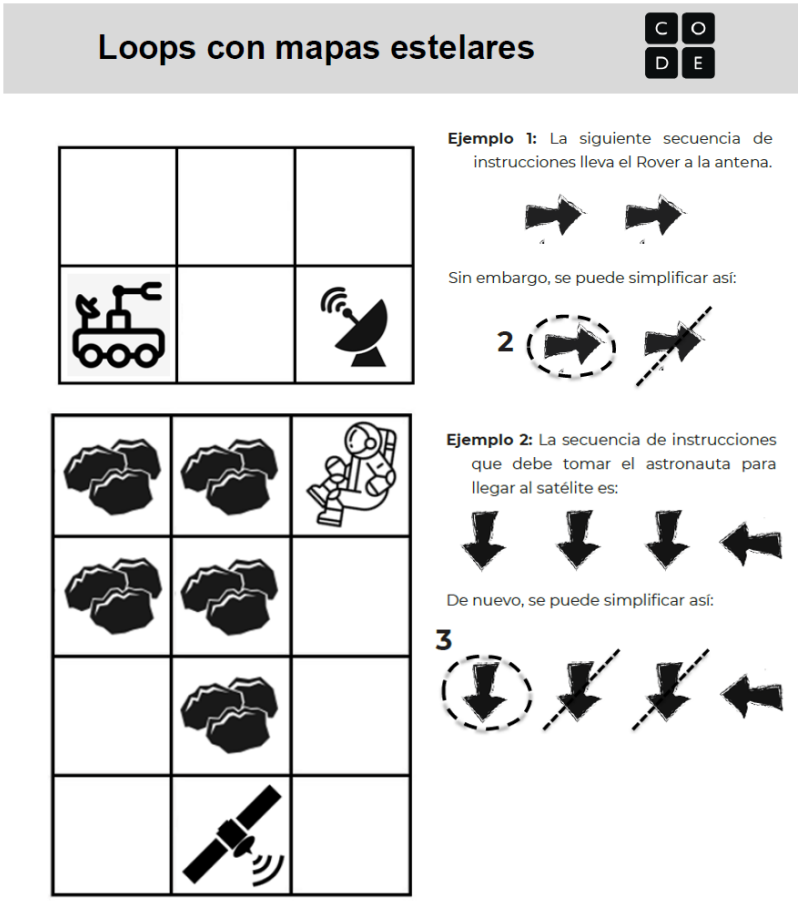
Nota: Tomado de Code.org https://studio.code.org/s/courseb-2021/lessons/6

La lección 7 sirvió de introducción a los bucles, permitiendo a los alumnos simplificar su código, agrupar los comandos que se repiten al notar la reproducción en los movimientos de sus compañeros y determinar cuántas veces deben repetirse dichos comandos. Al retomar los mapas estelares de la actividad 1, los alumnos tuvieron la oportunidad de relacionar viejos conceptos como la secuenciación con el nuevo concepto de bucles de repetición. Los demás aspectos relacionados con esta actividad se resumen en la Tabla 15.

Tabla 15

Actividad 8 - Lección 7 - Actividades desconectadas Code.org

Actividad 8	**Descripción**
Objetivos	- Identificar código repetitivo y sintetizar varias acciones en un solo bucle. - Interpretar un programa con bucles como una serie de acciones.
Conceptos computacionales	- Bucles
Descripción	Los bucles son una herramienta muy útil e importante en la programación. Para entender cuán útiles pueden ser, los estudiantes deben ser motivados a querer resolver problemas cotidianos de manera más fácil.
Tiempo	33 min
Recursos	



Nota: Tomado de Code.org https://studio.code.org/s/coursea-2021/lessons/7

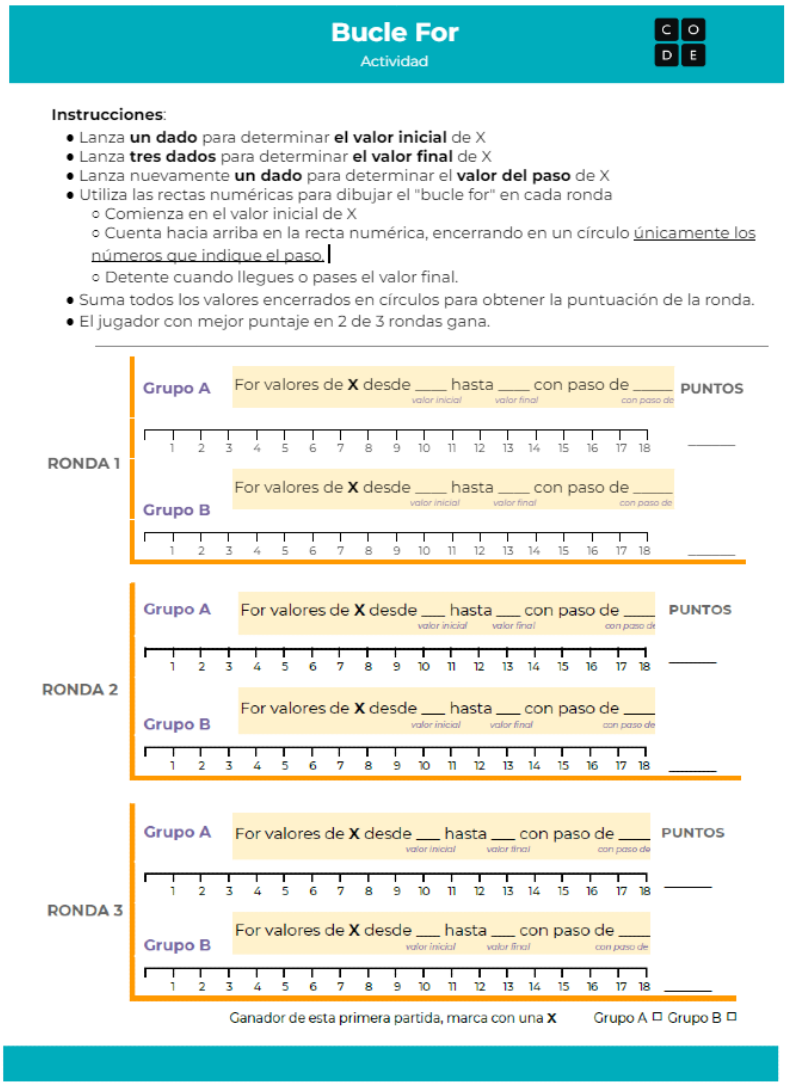
Sesión desconectada 6: Bucles *For*.

El objetivo de esta actividad fue aprender mediante la lección 13 sobre El bucle *for* que se usa comúnmente en programación. Este ciclo repite comandos un cierto número de veces, pero también realiza un seguimiento de los valores sobre los que está iterando, lo cual permitió que los estudiantes hicieran el paso a paso de la ejecución de este ciclo junto a una estructura con variables (ver Tabla 16)

Tabla 16

Actividad 9 - Lección 13 - Actividades desconectadas Code.org

Actividad 9	**Descripción**
Objetivos	- Determine el valor inicial, el valor de detención y el valor de escalón para un «bucle *for*». - Ilustra los valores de contador que se alcanzan cada vez a través de un «bucle *for*» durante el tiempo de ejecución.

Conceptos computacionales	- Bucles
Descripción	Sabemos que los bucles nos permiten hacer cosas una y otra vez, pero ahora vamos a aprender cómo usarlos con otras estructuras integradas. A través de estas nuevas estructuras, los estudiantes pudieron crear un código más potente y dinámico.
Tiempo	60 min
Recursos	

Nota: Tomado de Code.org https://studio.code.org/s/coursef-2021/lessons/13

Sesión desconectada 7: Composición Musical.

En la séptima sesión se trabajó el uso de funciones de la lección 11, que ayudan a simplificar el código y desarrollar la capacidad del estudiante para organizar su programa. Allí los estudiantes reconocieron rápidamente que la escritura de funciones puede hacer que sus programas largos sean más fáciles de leer y depurar si algo va mal. (ver tabla 17).

Tabla 17

Actividad 10: Lección 11 - Actividades desconectadas Code.org

Actividad 10	**Descripción**
Objetivos	- Describir cómo las funciones pueden hacer que los programas sean más fáciles de escribir. - Identificar secciones de una canción para incorporar a una función. - Localizar frases repetidas dentro de las letras de las canciones.
Conceptos computacionales	- Funciones
Descripción	Una de las estructuras más magníficas en el mundo de las ciencias de la computación es la función. Las funciones (a veces llamadas procedimientos) son pequeños programas que puedes utilizar una y otra vez dentro de tu programa más grande. Esta clase ayudo a los estudiantes a entender intuitivamente por qué la combinación de partes de código en funciones es una práctica tan útil.
Tiempo	50 min
Recursos	

Nota: Tomado de Code.org https://studio.code.org/s/coursee-2021/lessons/11.

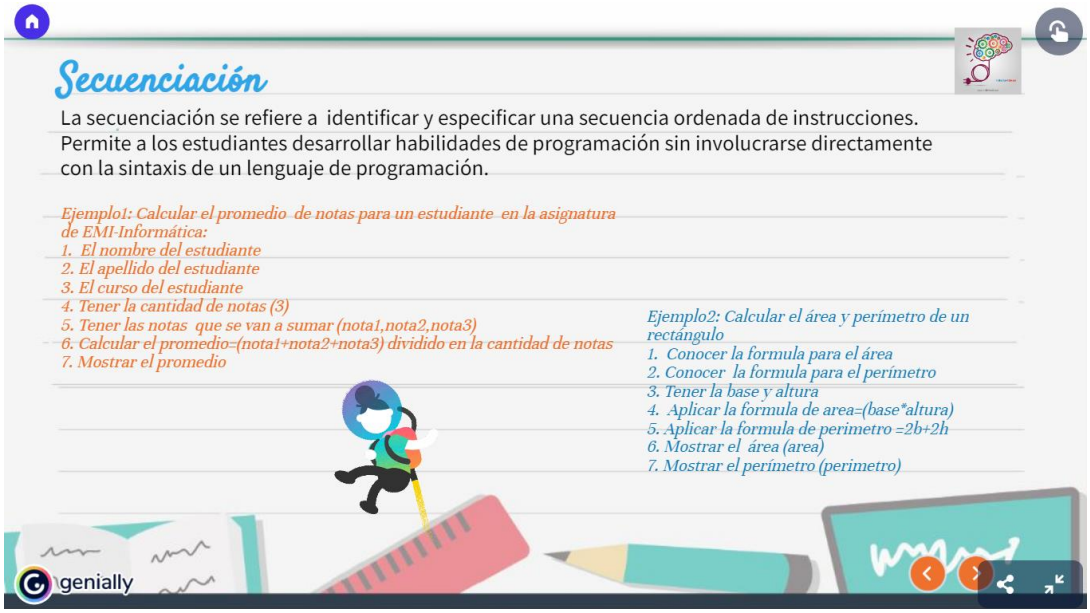
Anexo 3. Sesiones desarrolladas durante las actividades usa-modifica-crea.

Semana 1 usa-modifica-crea - Secuenciación y depuración

En las actividades de la semana 1 los estudiantes trabajaron los conceptos de secuenciación y depuración, ambos importantes para crear un programa mediante una serie de pasos sucesivos y encontrar los errores en sus propios programas, desarrollando sus habilidades en pensamiento crítico y resolución de problemas. En la sesión 1 se trabajó con una presentación de *Genially* como la que muestra la Figura 1, donde se describen ejemplos relacionados con su contexto escolar para ilustrar el concepto de secuenciación.

Figura 1.

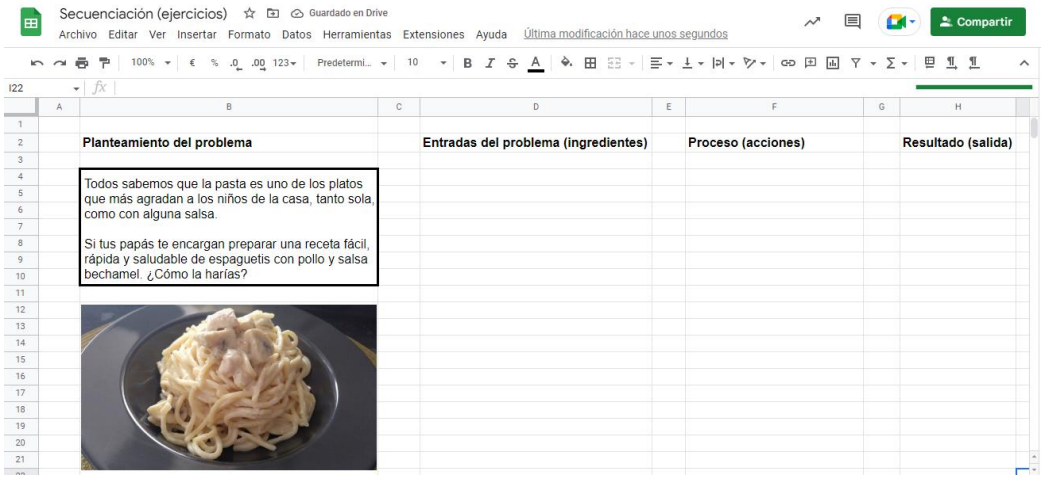
Ejemplos propuestos para el paso usa-modifica de la sesión 1.



En el paso crea de la sesión 2 se asignó a cada estudiante un archivo de hojas de cálculo de *Google* con cuatro situaciones problema: planear un viaje, calcular una nota final, escribir una receta y crear un manual de usuario. En cada ejercicio se le solicitó identificar los elementos básicos para el planteamiento de un problema (entrada, proceso y resultado) tal como lo muestra la Figura 2, continuando con los ejemplos desarrollados en la sesión 1.

Figura 2.

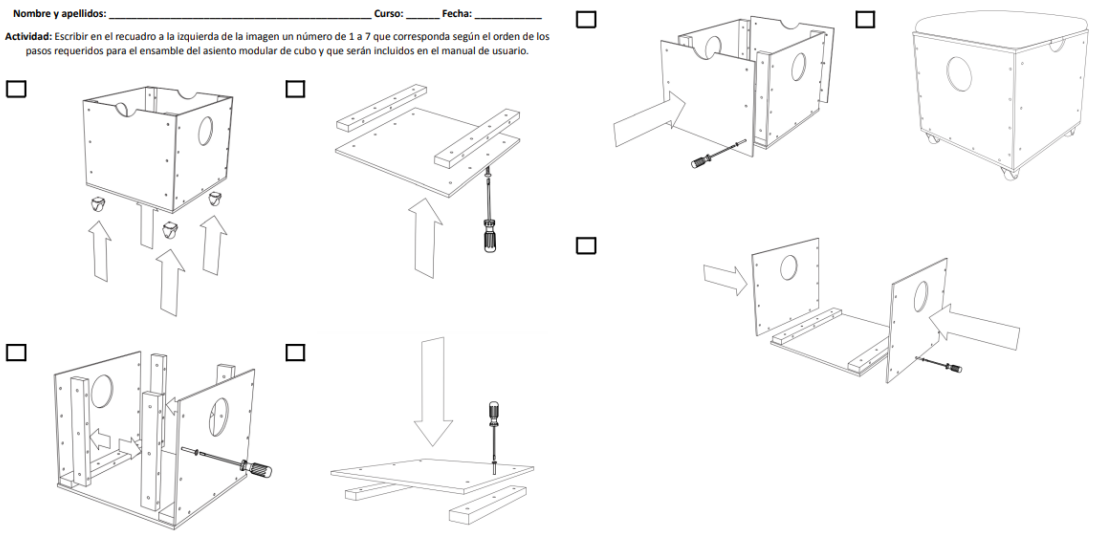
Ejercicio propuesto para el concepto de secuenciación de la sesión 1.



Para reforzar el concepto de secuenciación, se trabajó un archivo anexo en documentos de *Google* donde el estudiante debía identificar el orden de pasos requeridos en la solución del problema, por ejemplo, para crear el manual de usuario de un asiento modular con forma de cubo como el que muestra la Figura 3 de tal forma que pudiesen ser llevadas a cabo por cualquier persona que compre ese producto.

Figura 3.

Anexo usado en el paso crea de la sesión 2.



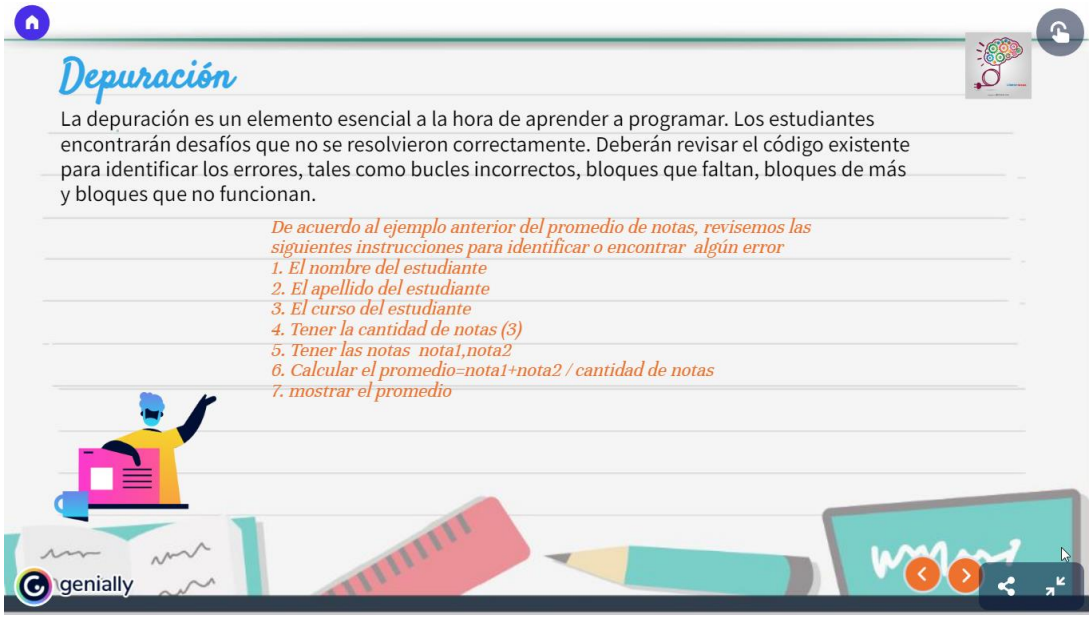
Nota:

Adaptado de TOCOMADERA, Sistema de mobiliario social, https://tocomadera.org/cubo-asiento/

En la sesión 3 se hizo énfasis en el concepto de depuración siguiendo los ejemplos relacionados con su contexto escolar como se muestra en la Figura 4.

Figura 4.

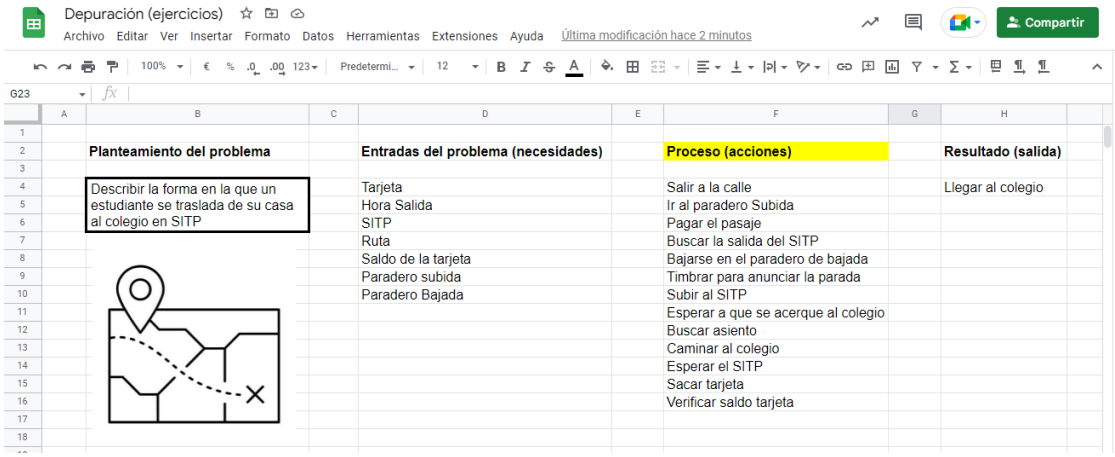
Ejemplos propuestos para el paso usa-modifica de la sesión 3.



De igual forma, en la sesión 4, se trabajaron las mismas situaciones problema mediante un segundo archivo, pero bajo el concepto de depuración, solicitándole al estudiante que encontrara los errores en el orden de la lista de acciones (resaltadas en amarillo) y las reorganizara para obtener el resultado. Un ejemplo de ello se muestra en la Figura 5 donde el estudiante debía pensar la secuencia lógica de pasos que llevan a que un estudiante pueda trasladarse desde su casa al colegio usando el transporte público.

Figura 5.

Ejercicio propuesto para el concepto de depuración de la sesión 4.



Depuración (ejercicios)

Archivo Editar Ver Insertar Formato Datos Herramientas Extensiones Ayuda Última modificación hace 2 minutos

Compartir

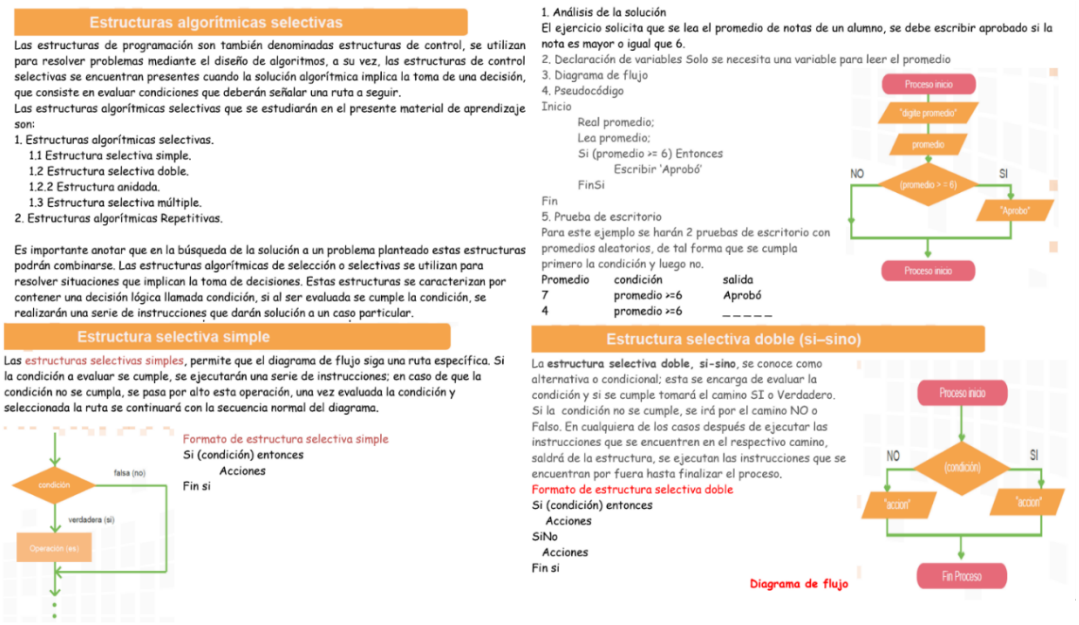
	Planteamiento del problema	Entradas del problema (necesidades)	Proceso (acciones)	Resultado (salida)
	Describir la forma en la que un estudiante se traslada de su casa al colegio en SITP	Tarjeta	Salir a la calle	Llegar al colegio
		Hora Salida	Ir al paradero Subida	
		SITP	Pagar el pasaje	
		Ruta	Buscar la salida del SITP	
		Saldo de la tarjeta	Bajarse en el paradero de bajada	
		Paradero subida	Timbrar para anunciar la parada	
		Paradero Bajada	Subir al SITP	
			Esperar a que se acerque al colegio	
			Buscar asiento	
			Caminar al colegio	
			Esperar el SITP	
			Sacar tarjeta	
			Verificar saldo tarjeta	

Semana 2 usa-modifica-crea - Condicionales

En las actividades de la semana 2 los estudiantes trabajaron el concepto de condicionales, con el propósito de que darle mayor versatilidad a un programa mediante estructuras de control, desarrollando sus habilidades de abstracción y modularización para la resolución de problemas. En la sesión 5 se trabajó con un documento como el que muestra la Figura 6, donde se describen los tipos de estructuras y ejemplos relacionados con cada una de ellas. Como refuerzo del concepto de condicionales, se trabajaron los tres ejercicios vistos en la sesión 5: horas de servicio social, promedio de tres notas y cálculo de una nota definitiva en el software *PseInt*, de tal forma que pudieran plasmar la solución mediante el uso de pseudocódigo y un diagrama de flujo.

Figura 6.

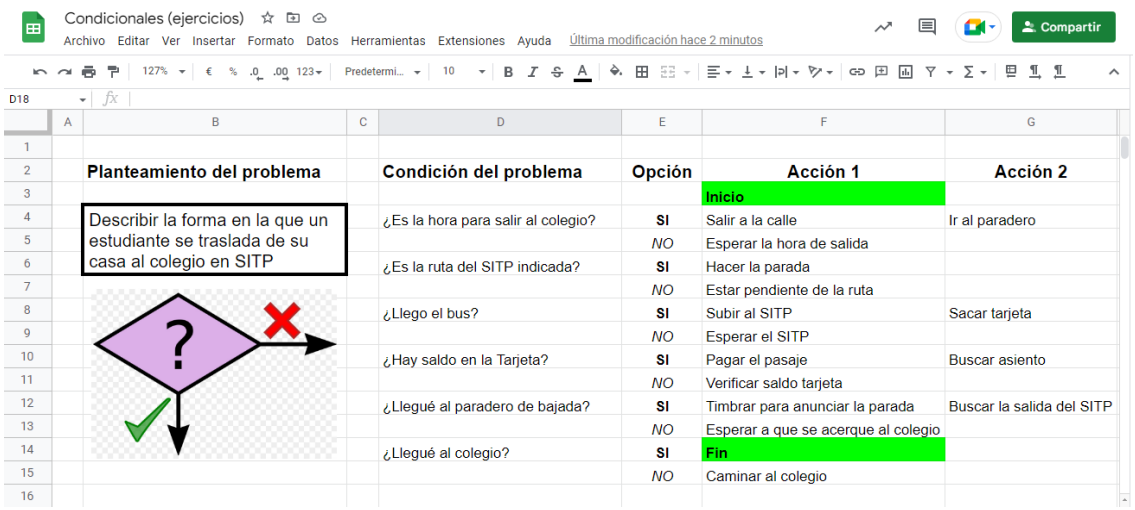
Ejemplos propuestos para el paso usa-modifica de la sesión 5.



En el paso crea de la sesión 6 cada estudiante trabajó en un archivo de hojas de cálculo de *Google* con tres situaciones problema: planear un viaje, calcular una nota final y escribir una receta. En cada ejercicio se le solicitó identificar los elementos básicos para el uso de condicionales (condición, opción y acciones) tal como lo muestra la Figura 7, de acuerdo con los ejemplos previos desarrollados en la sesión 5.

Figura 7.

Ejercicio propuesto para el concepto de condicionales de la unidad 6.



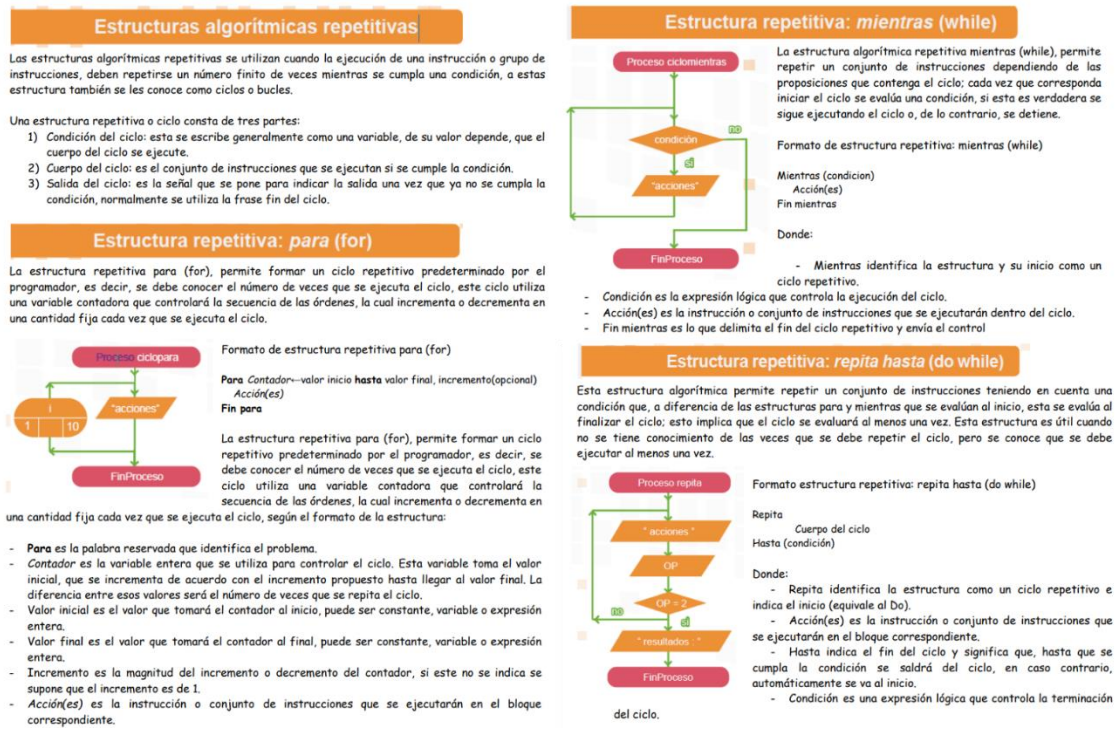
Planteamiento del problema	Condición del problema	Opción	Acción 1	Acción 2
			Inicio	
Describir la forma en la que un estudiante se traslada de su casa al colegio en SITP	¿Es la hora para salir al colegio?	SI	Salir a la calle	Ir al paradero
		NO	Esperar la hora de salida	
	¿Es la ruta del SITP indicada?	SI	Hacer la parada	
		NO	Estar pendiente de la ruta	
	¿Llego el bus?	SI	Subir al SITP	Sacar tarjeta
		NO	Esperar el SITP	
	¿Hay saldo en la Tarjeta?	SI	Pagar el pasaje	Buscar asiento
		NO	Verificar saldo tarjeta	
	¿Llegué al paradero de bajada?	SI	Timbrar para anunciar la parada	Buscar la salida del SITP
		NO	Esperar a que se acerque al colegio	
	¿Llegué al colegio?	SI	Fin	
		NO	Caminar al colegio	

Semana 3 usa-modifica-crea – Ciclos y bucles

En las actividades de la semana 3 los estudiantes trabajaron los conceptos de ciclos y bucles, los cuales son útiles para repetir acciones en un programa sin necesidad de escribir varias veces una misma parte del código. En la sesión 7 se trabajó con un documento como el que muestra la Figura 8, donde se describen los tipos de estructuras repetitivas y ejemplos relacionados con cada una de ellas.

Figura 8.

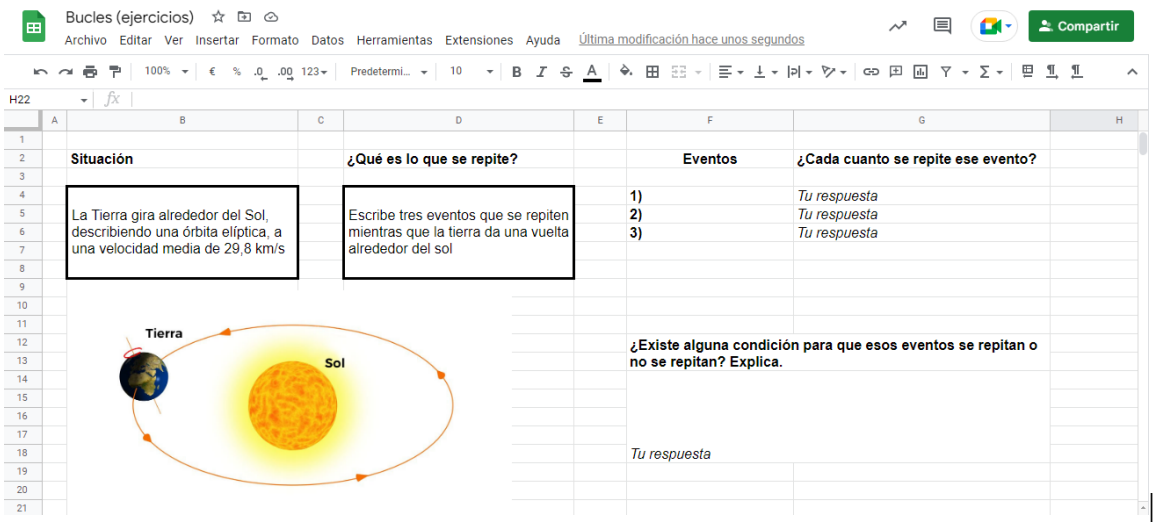
Ejemplos propuestos para el paso usa-modifica de la sesión 7.



En el paso crea de la sesión 8 cada estudiante trabajó en un archivo de hojas de cálculo de *Google* con tres planteamientos: sistema solar, eventos mundiales, Letras de canciones y pasos de baile. En cada ejercicio se le solicitó identificar eventos o elementos que se repitan, así como algunas condiciones esenciales para que ocurran, tal como lo muestra la Figura 9, haciendo referencia a las estructuras previas trabajadas en la sesión 11.

Figura 9.

Ejercicio propuesto para los conceptos de ciclos y bucles de la sesión 8.



Para reforzar el concepto de ciclos, se trabajó un archivo anexo en documentos de *Google* como el que muestra la Figura 10 donde el estudiante debía observar los pasos de una coreografía, darles un nombre, identificar aquellos que se repiten y simplificarlos, de tal forma que pudiesen ser enseñados a cualquier grupo de compañeros de su clase.

Figura 10.

Archivo anexo usado en el paso crea de la sesión 8.

Nombre: _____ **Curso:** _____ **Fecha:** _____

En las canciones es común encontrar (x3) si se quiere repetir una misma parte de la canción o como parte del coro. En el siguiente ejemplo, si quisiéramos simplificar la canción "Baby Shark" del ejercicio de bucles, es posible hacerlo de la siguiente manera:

Baby shark, doo doo doo doo doo doo	[Baby shark, doo(x6)](x3)
Baby shark, doo doo doo doo doo doo	Baby shark!
Baby shark, doo doo doo doo doo doo	
Baby shark!	
Mommy shark, doo doo doo doo doo doo	[Mommy shark,doo(x6)](x3)
Mommy shark, doo doo doo doo doo doo	Mommy shark!
Mommy shark, doo doo doo doo doo doo	
Mommy shark!	

Ese mismo método puede ser aplicado a cada una de las estrofas, lo cual simplifica aún más escribir la canción.

Ejercicio:
Observa los pasos de la coreografía del siguiente link, dales un nombre, identifica aquellos que se repiten y plantea un método para simplificarlos mediante el uso de ciclos.

https://www.youtube.com/watch?v=9HtRyc3ixrc

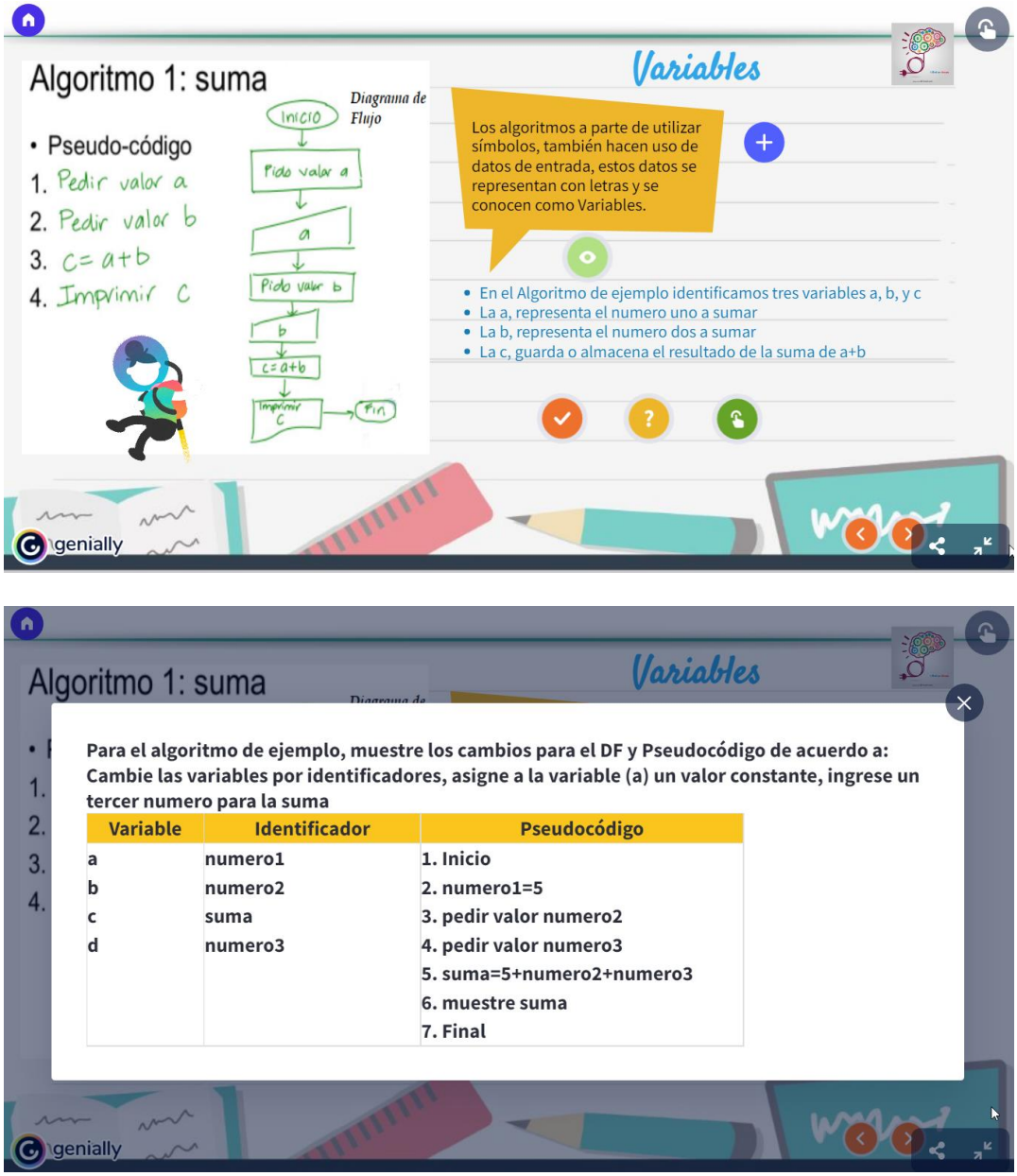
Escribe tu solución:

Semana 4 usa-modifica-crea – Variables y funciones

En las actividades de la semana 3 los estudiantes trabajaron los conceptos de variables y funciones, los cuales son necesarios para guardar información y hacer el llamado recurrente a otras partes del código. En la sesión 9 se trabajó con una presentación de *Genially* como la que muestra la Figura 11, donde se describen ejemplos relacionados con su contexto escolar y otros conceptos relacionados con la programación y solución de problemas.

Figura 11.

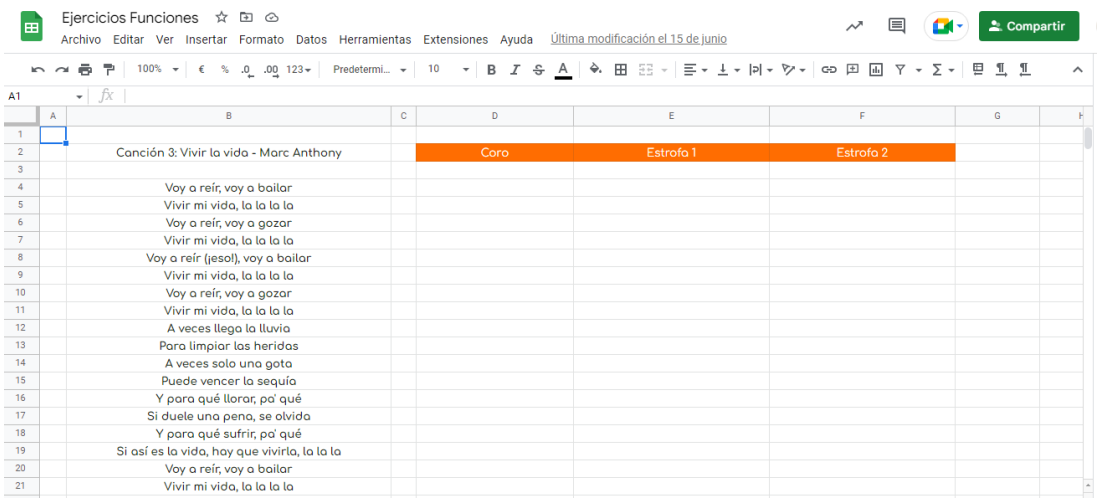
Ejercicios propuestos para el Paso Usa-Modifica de la sesión 9.



En el paso crea de la sesión 10 se compartió con cada estudiante un archivo de hoja de cálculo de *Google* con la letra de cinco canciones. Para cada canción se le solicitó identificar el coro y las estrofas de acuerdo con los párrafos que más se repetían en cada una de ellas, tal como lo muestra la Figura 12.

Figura 12.

Ejercicio propuesto para el concepto de funciones de la sesión 10.



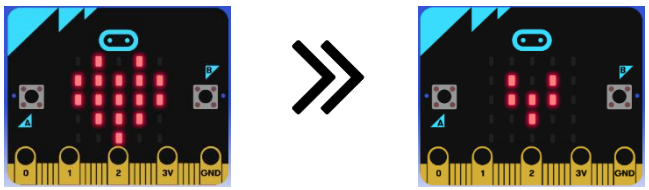
Anexo 4. Unidades desarrolladas durante las actividades usa-modifica-crea para la resolución de problemas.

Unidad 1 - Conceptos Iniciales *MakeCode* y *Micro:bit*.

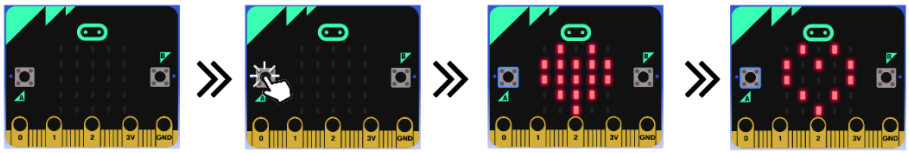
Para la unidad 1 se abordaron los conceptos computacionales a través del modelo usa-modifica-crea, entre ellos: secuenciación, tal que el estudiante pueda identificar un conjunto de pasos, reglas e instrucciones que debe seguir para la solución de un problema; uso de variables booleanas; y el uso de entradas y salidas para el manejo de información y la visualización de datos. El diseño final implementado en cada uno de estos pasos se describe en detalle en la tabla 17

Tabla 17

Diseño modelo de tres estados usa-modifica-crea, unidad 1

Unidad 1	**Descripción**
Objetivos	- Identificar un conjunto de pasos e instrucciones para realizar una tarea. - Simular la ejecución de un conjunto de instrucciones y pasos para saber si funcionan bien. - Manejar el editor *MakeCode* de la *Micro:bit* para escribir un programa y simular su funcionamiento. - Utilizar entradas y salidas de la *Micro:bit*, así como variables Booleanas.
Conceptos computacionales	- Secuencialidad - Entradas y salidas de datos - Variables Booleanas
Usa	Hacer uso de algunos bloques de la categoría Básico que incluye los bloques “mostrar icono”, “mostrar LEDs” y “pausa (100) ms” dentro del bucle “para siempre” para crear la animación de un corazón palpitando.
Resultado esperado	
Modifica	Hacer el corazón latir más lento incorporando los bloques “borrar pantalla” que limpia el panel de LEDs y “al presionar el botón A” de la categoría de “Entrada”, para visualizar un corazón palpitando cada vez que se accione el botón A de la micro:bit.

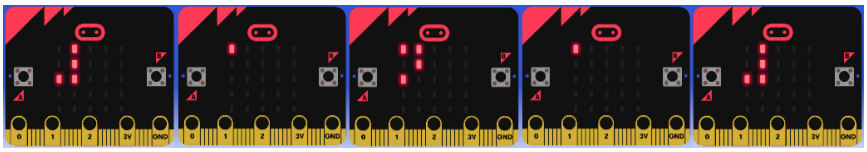
Resultado esperado



Crea

Envío de mensajes codificados usando la matriz de LEDs de la *Micro:bit* recurriendo a diversos símbolos que representan las letras del alfabeto y tiempos de pausa y teniendo en cuenta los tiempos de visualización para vocales y consonantes

Resultado esperado

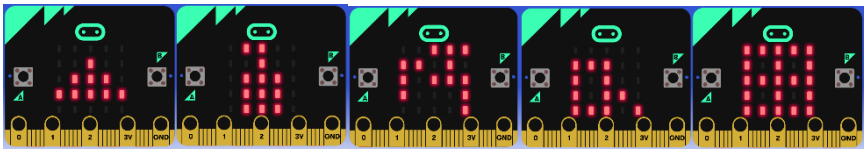


\# A N A #

Reto

- Usar el arreglo de LEDs para simular el funcionamiento de una lavadora
- Crear símbolos que representen cada una de las siguientes etapas: agregar agua, agregar jabón, enjuagar, sacar el agua, centrifugar.
- Representar los minutos de cada etapa mediante segundos de pausa.

Resultado esperado



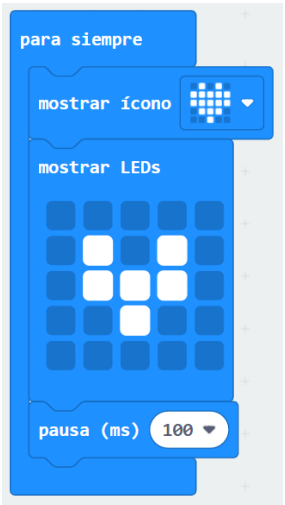
"Agregar agua" "Agregar jabón" "Enjuagar" "Sacar el agua" "Centrifugar"

En la Figura 13, se muestran los programas propuestos que debían seguir los estudiantes para los pasos usa-modifica de la unidad 1.

Figura 13

Ejercicios propuestos en los pasos usa-modifica para la unidad 1.

Usa

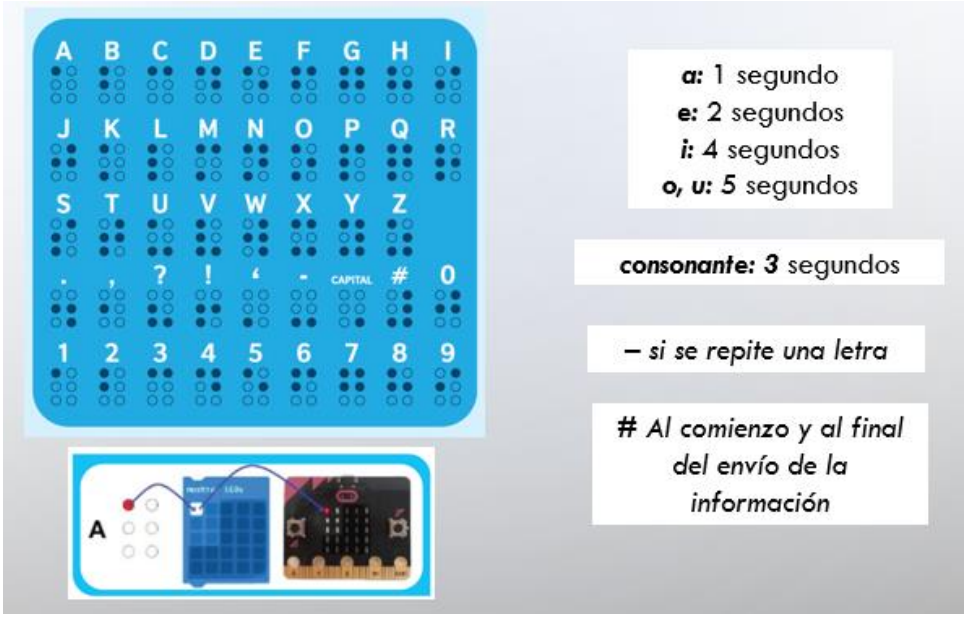


Modifica

Un aspecto importante del paso crea en la unidad 1 es que los estudiantes tengan en cuenta los símbolos acordados y cumplan con las reglas para el envío de mensajes que se muestran en la Figura 14.

Figura 14

Reglas situación problema paso crea unidad 1

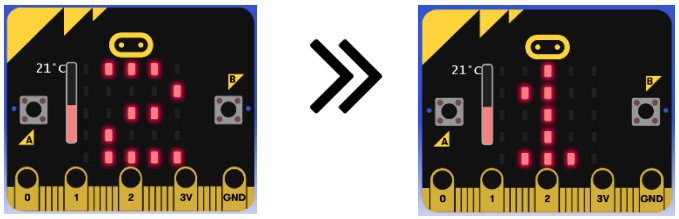
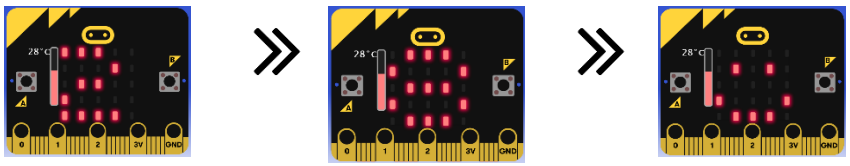
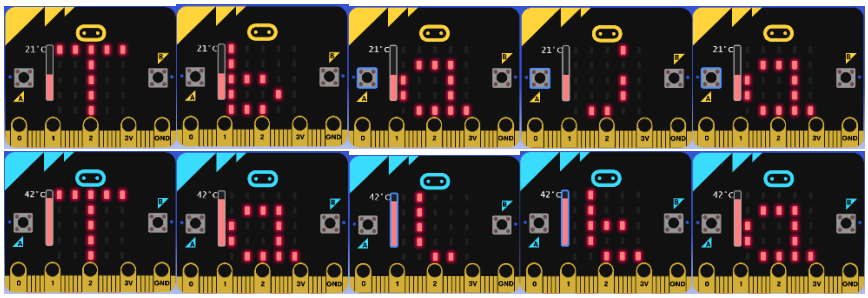


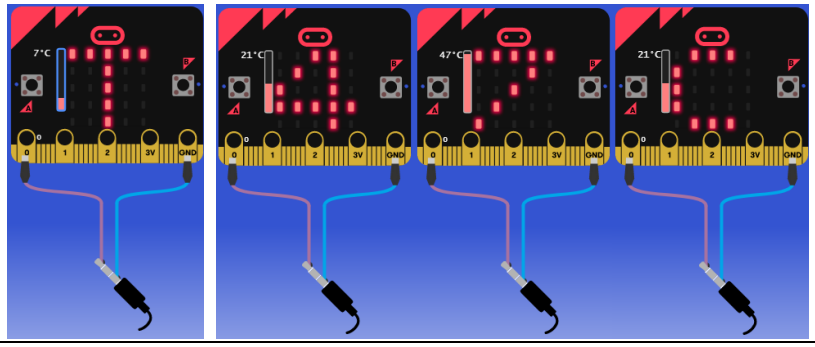
Unidad 2 - Entrada y salida de datos

En la unidad 2 se trabajaron los conceptos computacionales de condicionales y algoritmos, los cuales permiten representar las decisiones y pasos en la ejecución del programa de acuerdo con la información disponible, en ese sentido el modelo usa-modifica-crea, ayuda al estudiante a identificar la instrucción o grupo de instrucciones a seguir en función del valor de una condición, considerando los posibles casos que se puedan presentar a lo largo del programa. La tabla 18 muestra detalles del diseño final implementado.

Tabla 18

Diseño modelo de tres estados usa-modifica-crea, unidad 2

Unidad 2	**Descripción**
Objetivos	- Utilizar condicionales para decidir realizar o no una acción. - Utilizar condicionales para controlar la repetición de un conjunto de acciones. - Utilizar variables de entrada de magnitudes físicas como la temperatura. - Mostrar una variable numérica, como la temperatura, en el arreglo de LEDs.
Conceptos computacionales	- Condicionales. - Algoritmos - Variables de entrada y salida.
Usa	Hacer uso del sensor de temperatura que posee la Micro:bit junto con bloques “mostrar número” y “temperatura” de las categorías básico, entrada, bucles y lógica con el ánimo de visualizar en el arreglo de leds el valor de temperatura.
Resultado esperado	
Modifica	Modificar el valor de temperatura como condición para ejecutar o no las instrucciones dentro del bucle y complementar el programa incluyendo un icono de carita feliz mientras que la temperatura sea mayor a 25 grados.
Resultado esperado	
Crea	Programar la *Micro:bit* para que avise cuando la temperatura sea menor a 26°C con un mensaje “T. baja”, cuando la temperatura este entre 26°C y 34°C con un mensaje “T. normal” y cuando la temperatura sea mayor a 34°C con un mensaje “T. alta”.
Resultado esperado	
Reto	- Complementar el programa anterior anunciando mediante algún sonido cuando la temperatura sea superior al máximo y mediante otro cuando baje del mínimo admisible. - Explorar otros bloques de salida de la categoría Música

Resultado esperado	

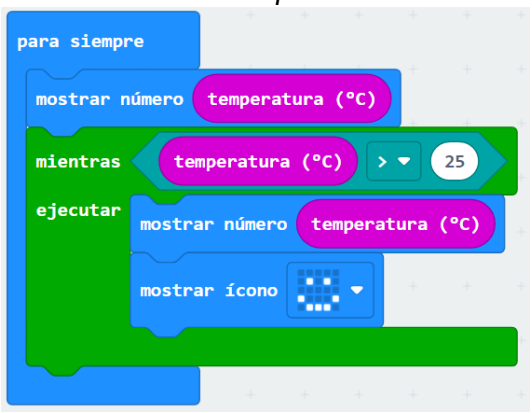
En el paso crea únicamente se incluye el bloque “mostrar número” junto con “temperatura”, mientras que el paso modifica se añade el bucle “mientras” como se aprecia en la Figura 15.

Figura 15

Ejercicios propuestos en los pasos usa-modifica para la unidad 2.

Usa

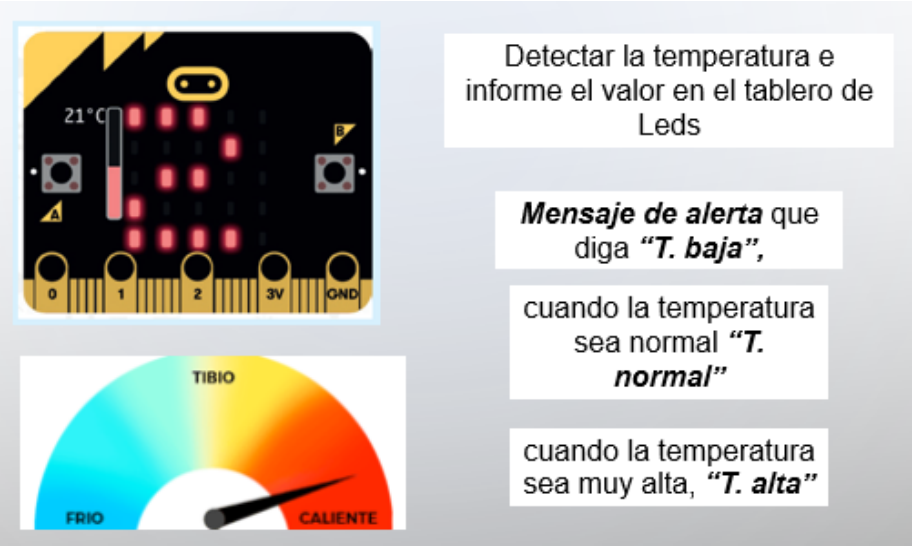
Modifica



El aspecto por revisar del paso crea en la unidad 2 es que los estudiantes tengan en cuenta los rangos de temperatura para cada condición y asocien correctamente los mensajes que se muestran en la Figura 16.

Figura 16

Planteamiento situación problema para el paso crea de la unidad 2



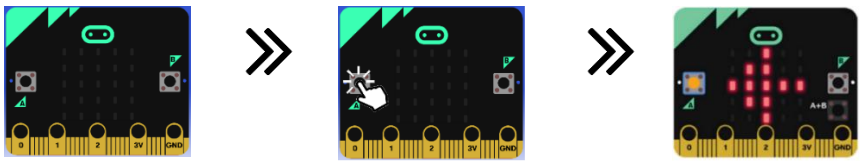
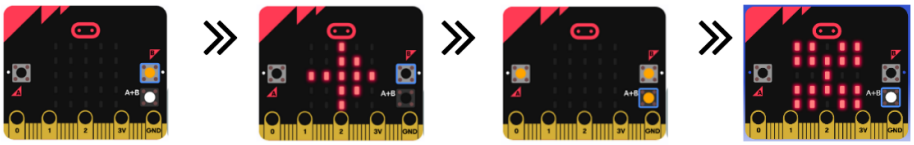
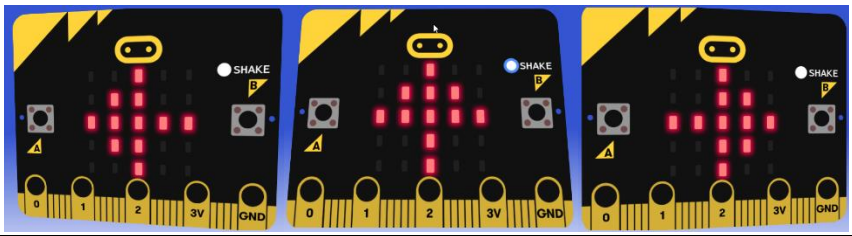
Unidad 3 - Condicionales y Bucles

La unidad 3 aborda los conceptos computacionales de diagramas de flujo, como una de las técnicas para resolver algoritmos, tal que el estudiante pueda identificar un conjunto de pasos, reglas e instrucciones que debe seguir para la solución de un problema; Condicionales, utilizando operaciones lógicas para decidir qué acción se ejecuta; y bucles, identificando las acciones que se repiten y el tipo de bucle a utilizar en la solución del problema. La tabla 19 muestra detalles de la secuencia implementada en esta unidad.

Tabla 19

Diseño modelo de tres estados usa-modifica-crea, unidad 3

Unidad 3	**Descripción**
Objetivos	- Utilizar variables booleanas de entrada. - Comunicar instrucciones utilizando la pantalla de LED y un código de flechas. - Interpretar un diagrama de flujo para resolver problemas como el de un laberinto. - Utilizar operaciones lógicas para decidir qué acción se ejecuta. - Utilizar lazos que se repiten hasta terminar la tarea
Conceptos computacionales	- Bucles. - Diagrama de flujo. - Operaciones lógicas.

Usa	Crear un dispositivo que permita ayudar a las personas que se desplazan en bicicleta a transitar de forma más segura por las ciclovías, incorporando luces y señales informativas mientras está en movimiento la bicicleta, tal que al presionar el botón A, se muestre en la micro:bit una luz direccional a la izquierda parpadeando, al presionar el botón B, una luz direccional a la derecha parpadeando y cuando se presionen los dos botones A+B, mostrar un indicativo de frenar para alertar a quien va detrás.
Resultado esperado	
Modifica	Completar la programación para los botones B y A+B dentro del mismo bloque “para siempre”, agregando las condiciones restantes, complete el sistema de luces agregando animaciones para los otros tres casos, izquierda, derecha y freno.
Resultado esperado	
Crea	Agregar otras animaciones a la programación dando la sensación de movimiento hacia adelante de la bicicleta, incluyendo el sensor acelerómetro para que el simulador muestre un botón que dice “*SHAKE*” (agitar) para representar que la tarjeta está siendo agitada y las flechas indican el movimiento del desplazamiento hacia adelante.
Resultado esperado	
Reto	- Mejorar la programación de modo que incorpore el sensor “acelerómetro” para medir si hay cambios en el movimiento y determinar el giro y la dirección sin necesidad de presionar los botones A o B. - Explorar el bloque que se encuentra en la categoría “Entrada” identificado como “es un gesto agitado”
Resultado esperado	

En la Figura 17, ejercicio 1, se ilustra los bloques para simular el desplazamiento hacia adelante usando el bloque “si agitado” de la categoría “entrada” y que funciona como el bucle general “para siempre”. Dentro de este bucle hay dos bloques que se repetirán uno tras otro hasta que se termine la tarea. Por su parte la Figura 17, ejercicio 2, muestra la programación

para que al presionar el botón A la matriz de led muestre una flecha que simula la luz de la direccional a la izquierda. Allí se incluye el bloque “para siempre” como bucle de repetición indefinido y un condicional para el bloque “botón A presionado” de la categoría “Entrada”. Para efectos de visualizar el resultado como un parpadeo de la flecha se incluye el bloque “borrar la pantalla”.

Figura 17

Ejercicios propuestos en los pasos usa-modifica para la unidad 3.

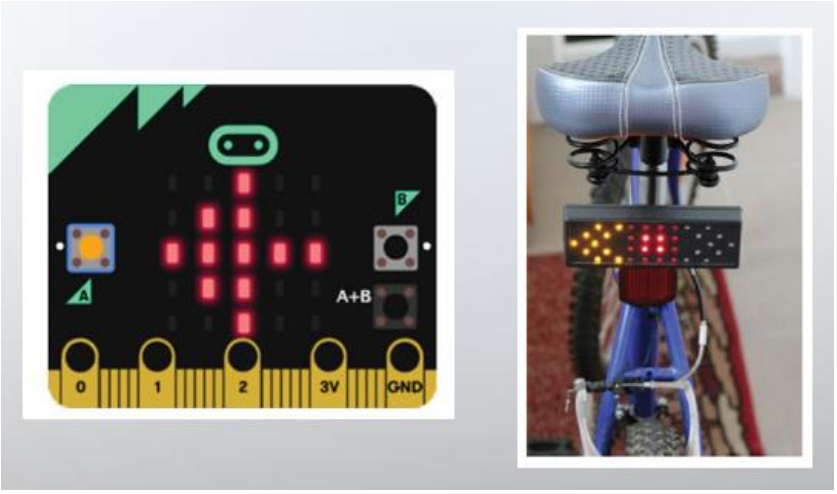
Ejercicio 1

Ejercicio 2

En el paso usa-modifica se busca que el estudiante asocie correctamente el presionar el botón con una acción que muestre correctamente la dirección hacia la cual apunta la flecha que indica el giro de la bicicleta, tal como se muestra en la Figura 18.

Figura 18

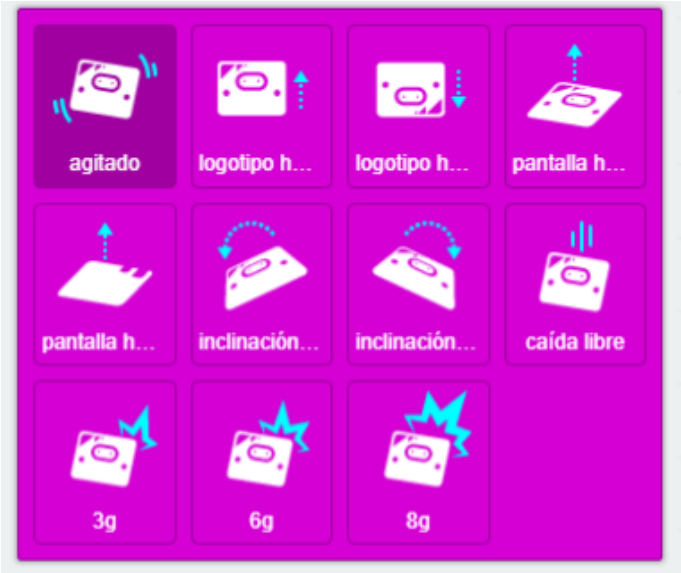
Planteamiento del ejercicio usa-modifica para la unidad 3



Posteriormente el ejercicio planteado en el reto amplia otras funcionalidades de la tarjeta *Micro:bit* al usar el acelerómetro como variable de entrada de datos de acuerdo con las funciones que muestra la Figura 19.

Figura 19

Funcionalidades del acelerómetro en la Micro:bit

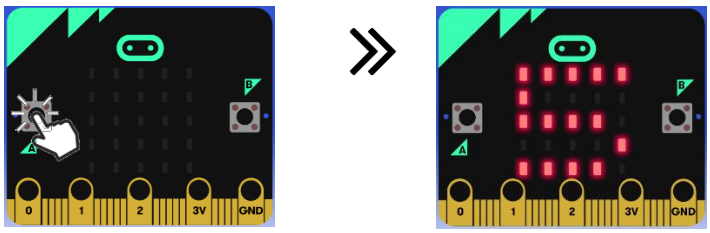
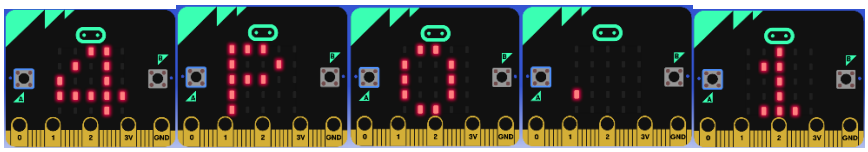
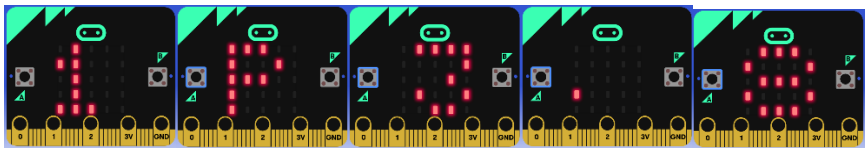


Unidad 4 – Variables

Para la unidad 4 se plantea abordar los conceptos computacionales de: variables, tal que se pueda definir una variable que guarde un valor numérico y realizar operaciones matemáticas con los valores en variables internas, así como obtener un número aleatorio con la *Micro:bit* para simular eventos que suceden al azar como el lanzamiento de dados**.** La secuencia implementada en esta unidad se muestra en la Tabla 20.

Tabla 20

Diseño modelo de tres estados Usa-modifica-crea, unidad 4

Unidad 4	**Descripción**
Objetivos	- Definir una variable interna que guarde un valor numérico. - Realizar operaciones con los valores en variables internas. - Obtener un número aleatorio con la *Micro:bit.*
Conceptos computacionales	- Variables. - Diagrama de flujo. - Operaciones matemáticas.
Usa	Simular la cantidad de agua que cae a diario en su barrio usando el lanzamiento de un dado donde cualquier valor entre 1 y 6 que aparezca en el dado corresponda a la cantidad de lluvia que cae en un día en m.m.
Resultado esperado	
Modifica	Seguir un diagrama de flujo tal que simule y ejecute el algoritmo para calcular el promedio de lluvia que cae en una semana (7 días)
Resultado esperado	
Crea	Continuar resolviendo el problema teniendo en cuenta crear un programa que calcule el promedio de lluvia que cae por cinco años (1825 días), siguiendo un diagrama de flujo.
Resultado esperado	
Reto	Mejorar el programa de modo que al presionar el botón B pueda mostrar cuantos días al año caen menos de 2 mm de agua.

Resultado esperado

En el paso usa se propone como ejercicio hacer simulaciones a partir de datos, teniendo en cuenta la información de la Figura 20, donde se muestran los datos para calcular la cantidad de agua que cae en un periodo de tiempo, por ejemplo, para 7 días de una semana. La cantidad de agua está determinada por el número de milímetros de agua que se acumulan en un recipiente estándar llamado pluviómetro.

Figura 20

Representación de datos: calcular la cantidad de agua en un periodo de tiempo

DIA	Cantidad de agua (mm)
Lunes	10
Martes	5
Miércoles	8
Jueves	6
Viernes	4
Sábado	5
Domingo	8
PROMEDIO	6.6

Para el paso modifica, el estudiante debe seguir el diagrama de flujo que se muestra en la Figura 21 y posteriormente ejecutar el algoritmo para una semana (7 días) de la Figura 22 para simular la cantidad de agua que cae a diario en su barrio.

Figura 21

Representación diagrama de flujo.

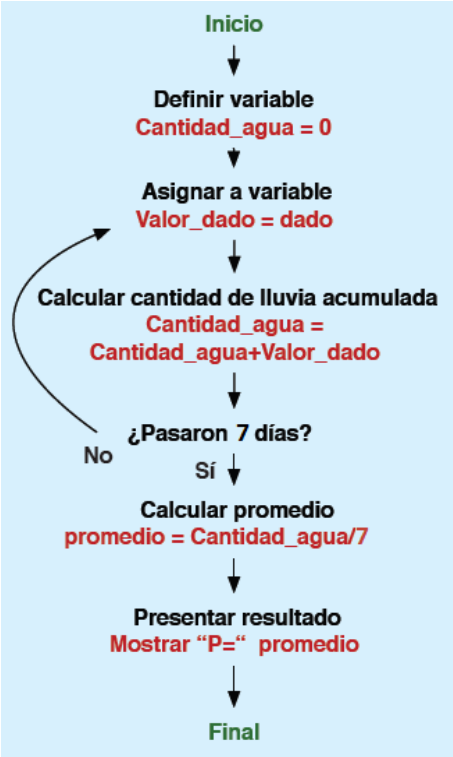


Figura 22

Algoritmo del ejercicio propuesto en palabras

Ejemplo de algoritmo en palabras:

Paso 0: Colocar la variable ***Cantidad_agua*** en cero.

Paso 1: Lanzar el dado.

Paso 2: Guardar el valor del dado en la variable ***Valor_dado.***

Paso 3: Acumular en ***Cantidad_agua*** el ***Valor_dado;*** es decir, en la celda ***Cantidad_agua*** colocar el acumulado de la iteración anterior más la cantidad actual.

Paso 4: Si aún no se han hecho 7 lanzamientos, ir de nuevo al paso (1).

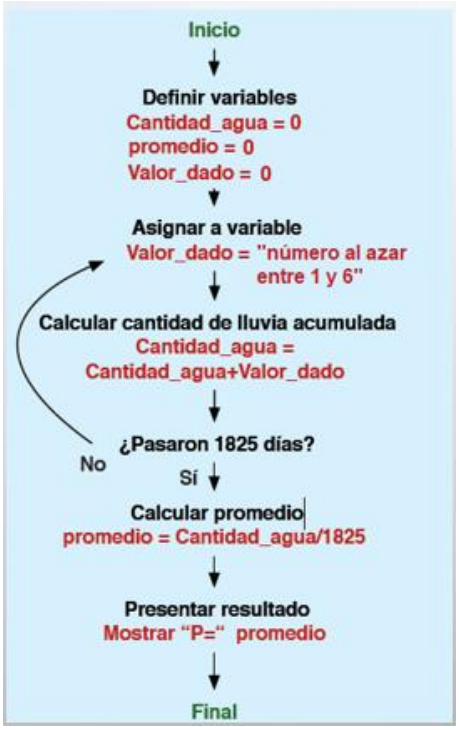
Paso 5: Encontrar el valor promedio dividiendo ***Cantidad_agua*** entre 7.

Paso 6: Reportar resultado.

Con base en lo realizado en el paso usa-modifica, el estudiante debe ajustar el algoritmo para calcular la cantidad promedio de lluvia que cae en una región en 5 años y crear un programa teniendo en cuenta las variables ”Cantidad_agua“, ”Valor_dado” y “Promedio” tal que cumpla con lo planteado en el diagrama de flujo de la Figura 23.

Figura 23

Diagrama de flujo para calcular el promedio de agua que cae en 5 años.



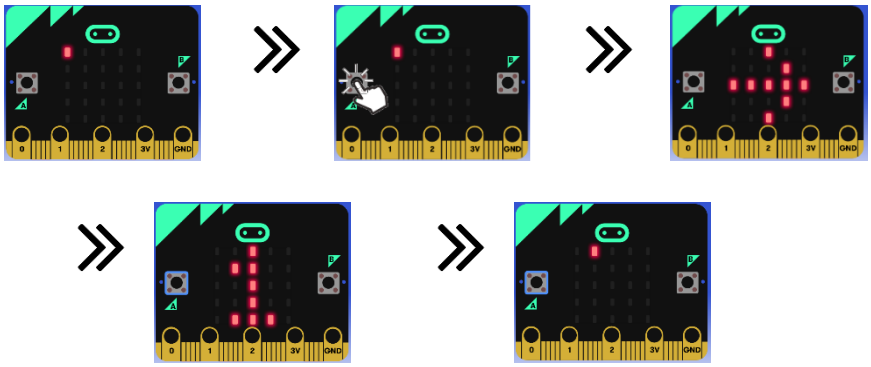
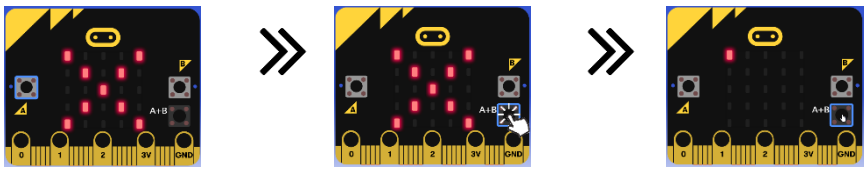
En la revisión del paso crea, se busca incluir instrucciones que usen los comandos establecer para asignar un valor a la variable y cambiar que permite adicionar el valor especificado a la variable. En este ejercicio no fue necesario incluir los bloques “al iniciar” y “para siempre**”.**

3.4.5 Unidad 5 – Funciones

En la unidad 5 se trabajó principalmente el concepto de funciones con el objetivo de usar diferentes variables, así como entradas y salidas de la micro:bit, además de usar funciones para estructurar y simplificar el código, de tal forma que los estudiantes hicieran pruebas del código y mejoraran sus programas usando casos de prueba. La secuencia implementada en esta unidad se muestra en la tabla 21.

Tabla 21

Diseño Modelo de tres estados Usa-modifica-crea, unidad 5

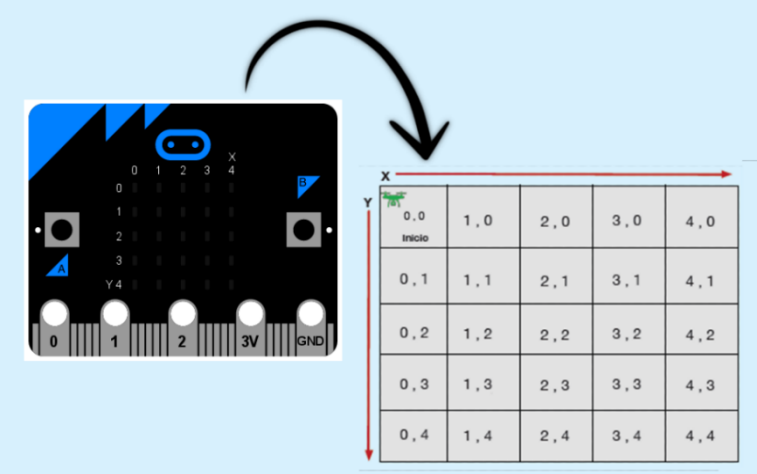
Unidad 5	**Descripción**
Objetivos	- Crear una simulación que incluya conceptos de programación como variables y funciones. - Usar diferentes entradas, salidas y variables. - Hacer pruebas y mejorar los programas usando casos de prueba. - Usar funciones para estructurar y simplificar un código
Conceptos computacionales	- Funciones -Variables -Coordenadas
Usa	Hacer un programa que permita simular algunas de las instrucciones necesarias para lograr que el helicóptero vuele dentro de un rango de coordenadas limitado.
Resultado esperado	
Modifica	Construir la función Validar_coordenadas y modificar los bloques para que se alcancen a ver la flecha y la coordenada después de presionar el botón A o B.
Resultado esperado	
Crea	Incluir una acción que inicie las coordenadas en cero cuando se opriman los botones A y B simultáneamente.
Resultado esperado	
Reto	Modificar el programa para que al menos la condición del número de vuelos sea tenida en cuenta y avise cuando ya se hayan probado los 5 vuelos posibles

En el paso usa-modifica se propuso como ejercicio representar el proceso para hacer volar el helicóptero robótico de la NASA llamado *Mars Helicopter*, el cual debido a su tamaño

solo puede realizar máximo 5 vuelos en el planeta Marte con una duración máxima de 90 segundos cada uno. Los vuelos de exploración se simulan tal que el helicóptero vuele dentro de un rango de coordenadas limitado definido por la matriz de leds que posee la *Micro:bit* (ver Figura 24), de esta forma, dos variables denominadas Coordenada_X y Coordenada_Y servirán para almacenar la trayectoria de vuelo.

Figura 24

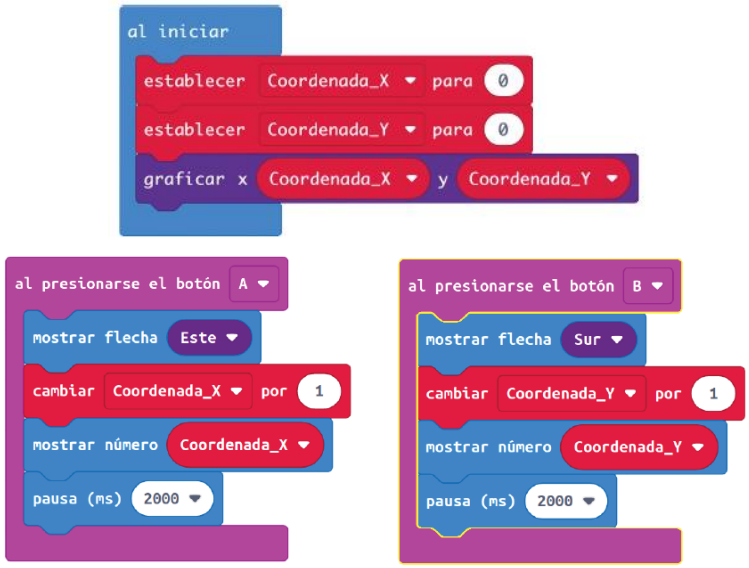
Matriz de coordenadas usadas en la unidad 5.



En la Figura 25, se ve el programa de ejemplo que establece el valor inicial de las variables en cero y grafica la posición inicial del helicóptero.

Figura 25

Programa para el primer ejercicio del usa-modifica.



Posterior a la secuencia usa-modifica se plantearon dos momentos para la implementación del programa propuesto. En primer lugar, crear la función de la Figura 26 la cual permite realizar un movimiento solicitado mostrándolo en la matriz de LEDs de la *Micro:bit* y verificar que esté dentro del campo de acción del helicóptero. Si las coordenadas (x, y) llegan a estar fuera del área de trabajo, la función no debe realizar el movimiento y muestra una X, regresando un valor de “verdadero” si esto es válido y “falso” si no lo es.

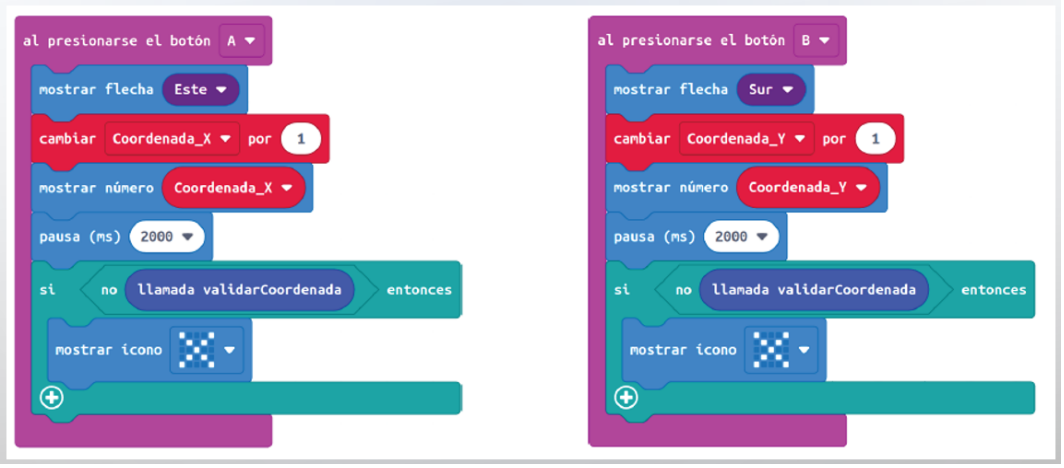
Figura 26

Creación de la función para la unidad 5

En segundo lugar, se propuso ajustar los bloques de “al presionar el botón A” “y al presionar el botón B” para validar las coordenadas en vez de hacerlo en el bloque “para siempre” tal como se muestra en la Figura 27.

Figura 27

Ajuste del programa inicial de la unidad 5



Como reto para esta unidad se propuso implementar el programa para que el programa avise cuando ya se hayan probado el máximo de 5 vuelos posibles. Para ello, se requiere definir un plan de vuelo conformado por varios movimientos, sin que necesariamente se apruebe cada uno de estos de forma independientemente, y luego se apruebe el plan de vuelo completo.

Anexo 5. Hallazgos de las evidencias de aprendizaje para las actividades usa-modifica-crea

Figura 1

Ejemplo solución general unidad 1 estudiantes grupo desconectadas y control.

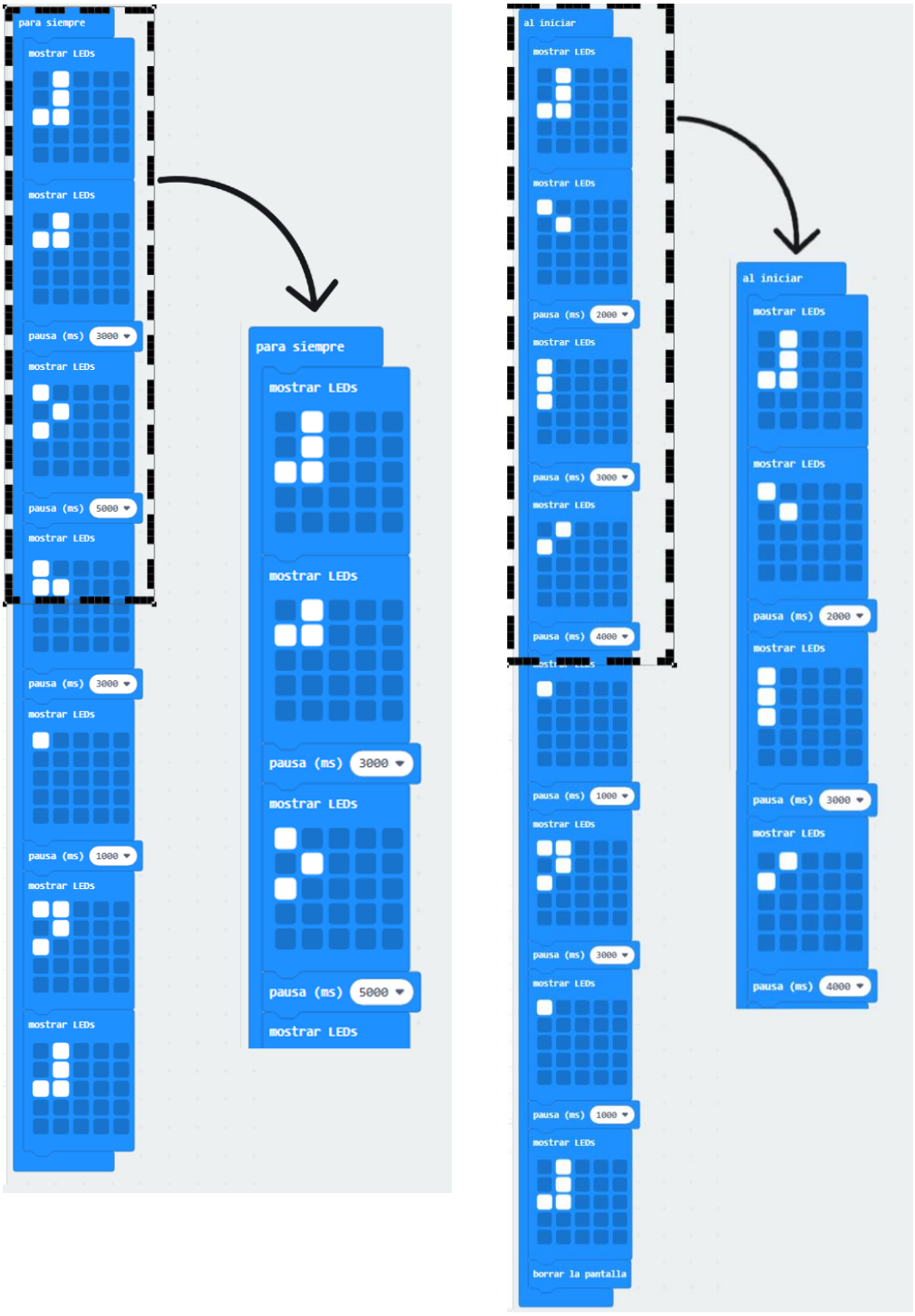


Figura 2

Hallazgos en las soluciones del grupo desconectadas para la unidad 1

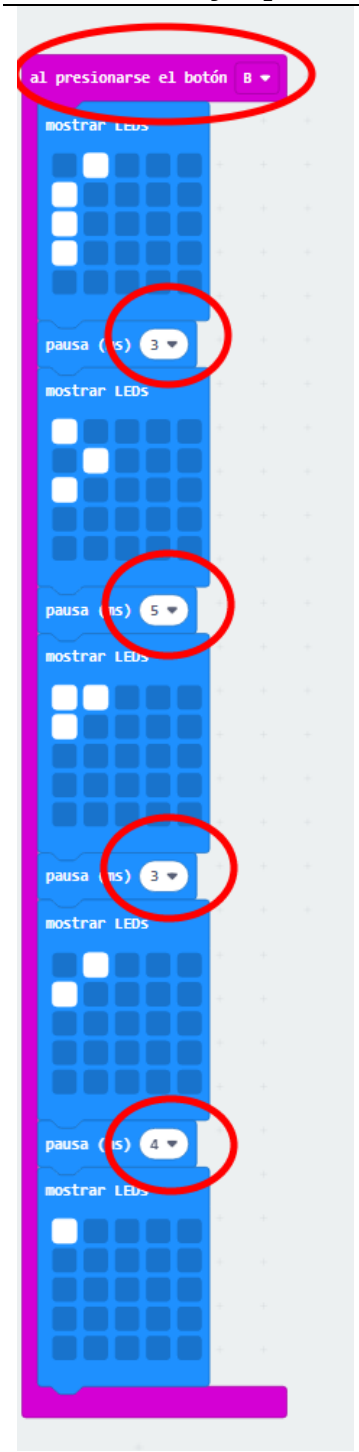
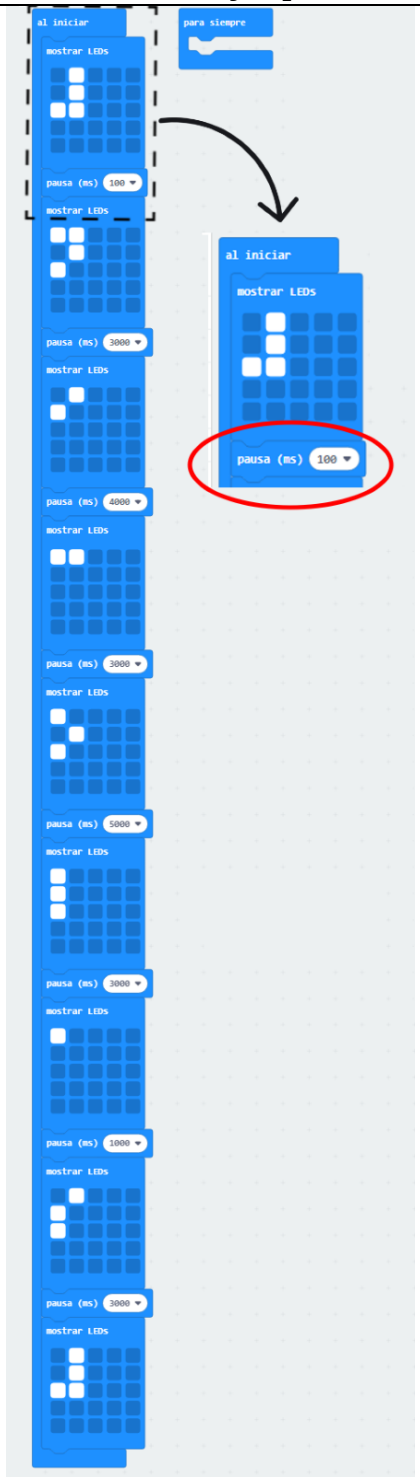
Ejemplo 1	Ejemplo 2
	

Figura 3

Errores en las soluciones del grupo control para la unidad 1.

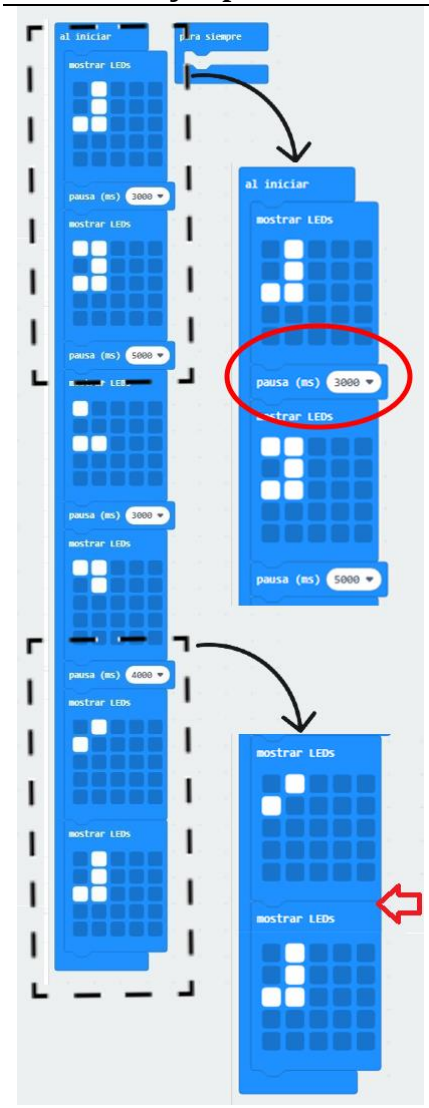
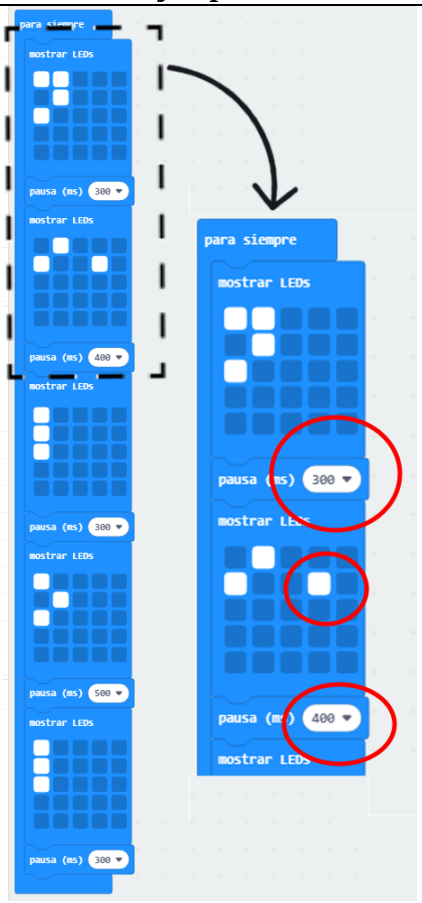
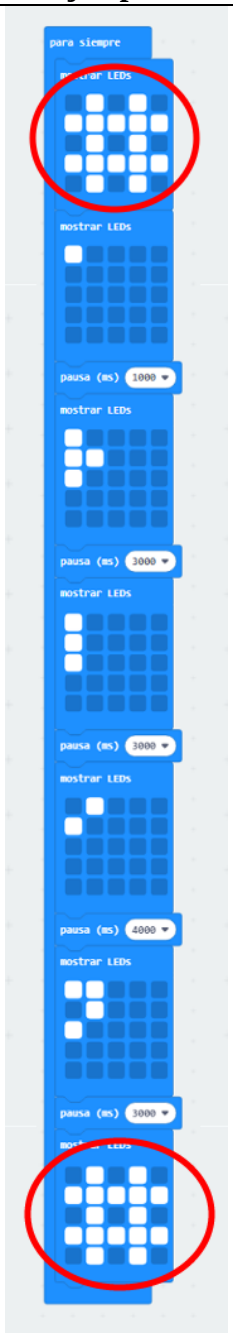
Ejemplo 1	Ejemplo 2	Ejemplo 3
		

Figura 4

Ejemplos de diagrama de bloques unidad 1 estudiantes grupo conectadas.

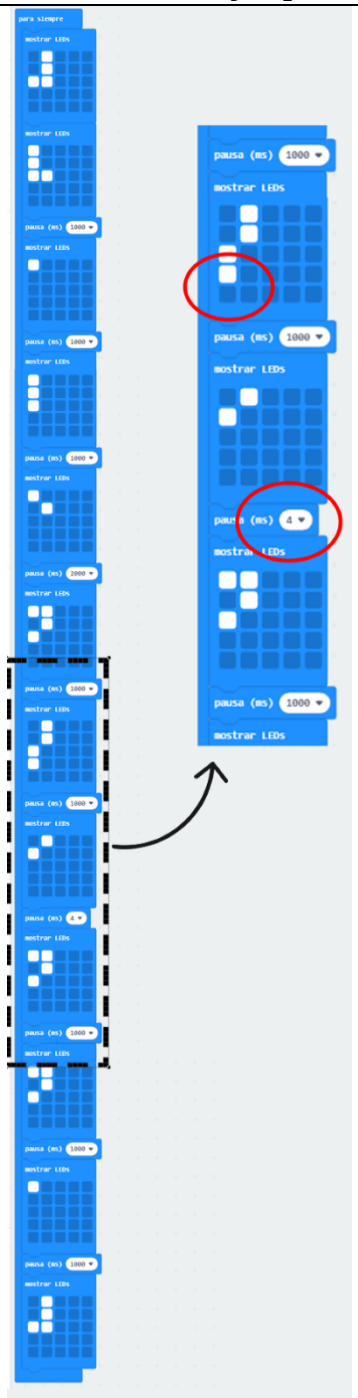
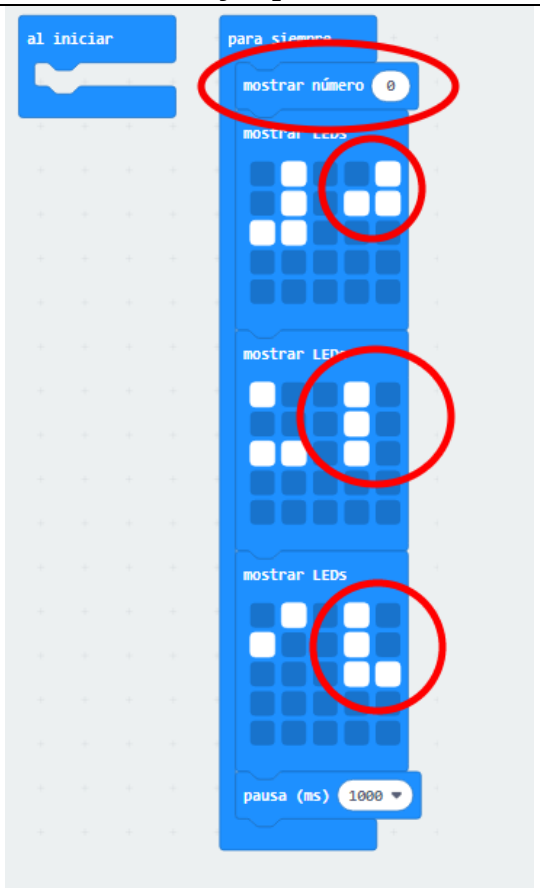
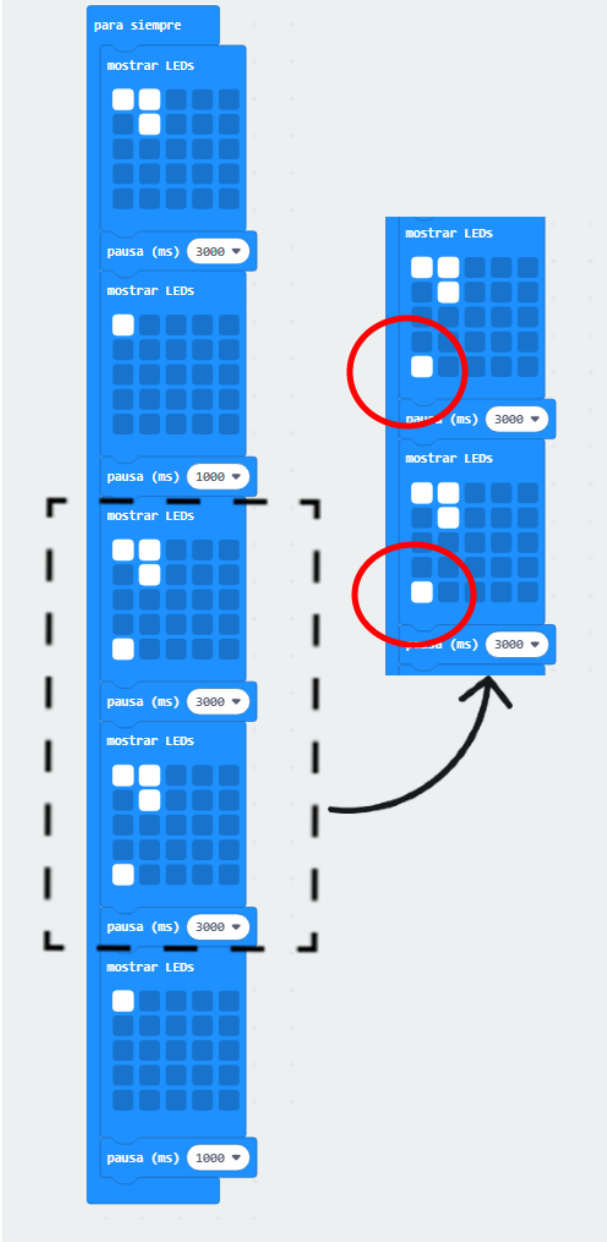
Ejemplo 1	Ejemplo 2
	

Figura 5

Ejemplos de diagrama de bloques unidad 1 estudiantes grupo conectadas.

Ejemplo 3



Ejemplo 4

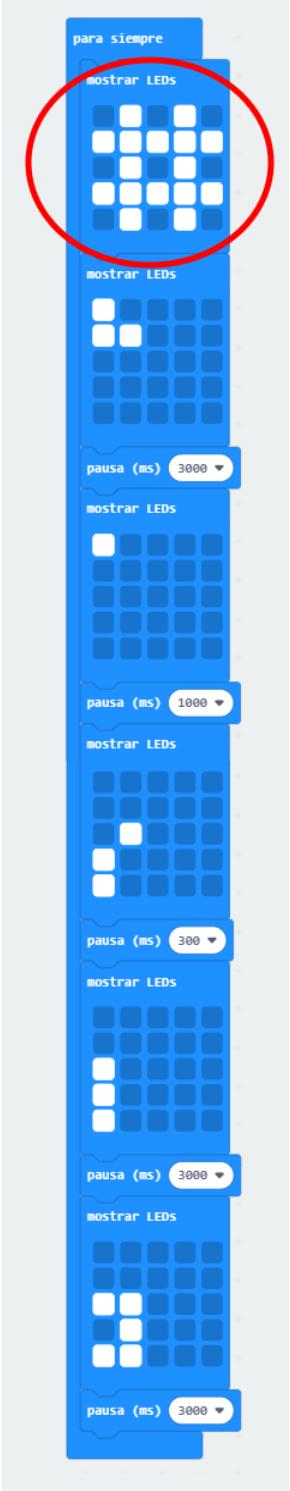
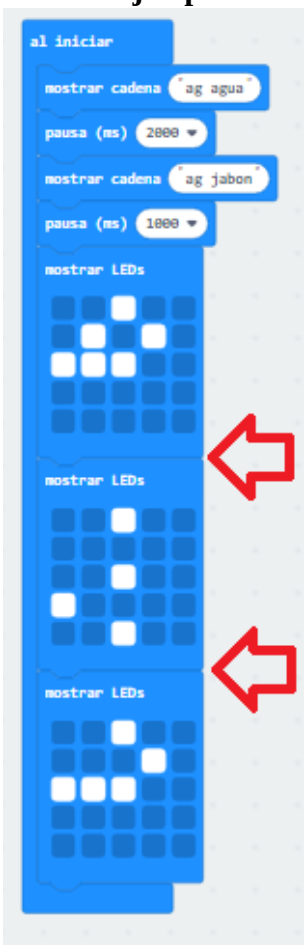


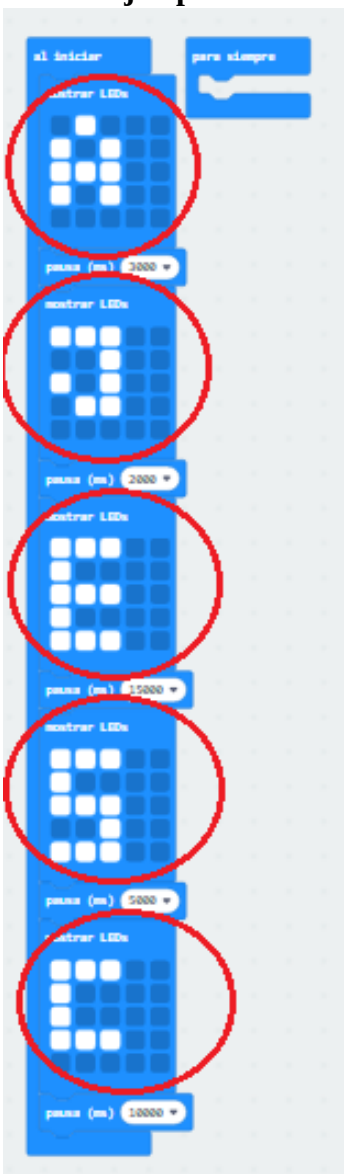
Figura 6

Ejemplos solución reto unidad 1

Ejemplo 1



Ejemplo 2

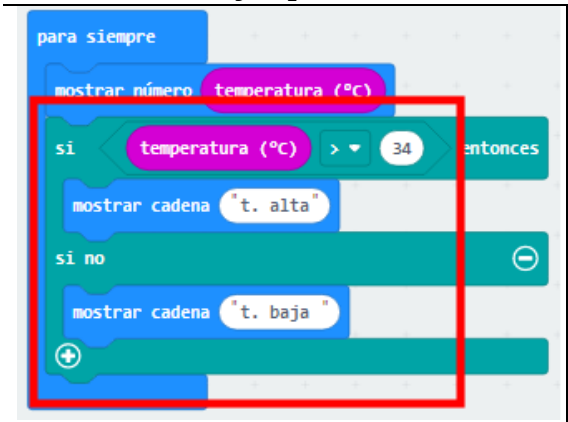


Ejemplo 3

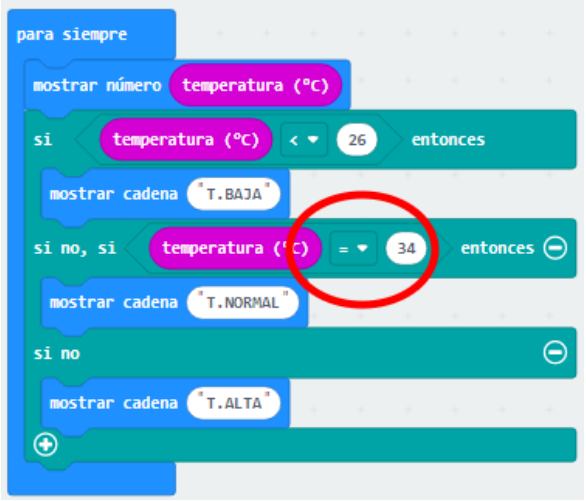
Figura 7

Hallazgos unidad 2 grupo conectadas.

Ejemplo 1



Ejemplo 2



Ejemplo 3

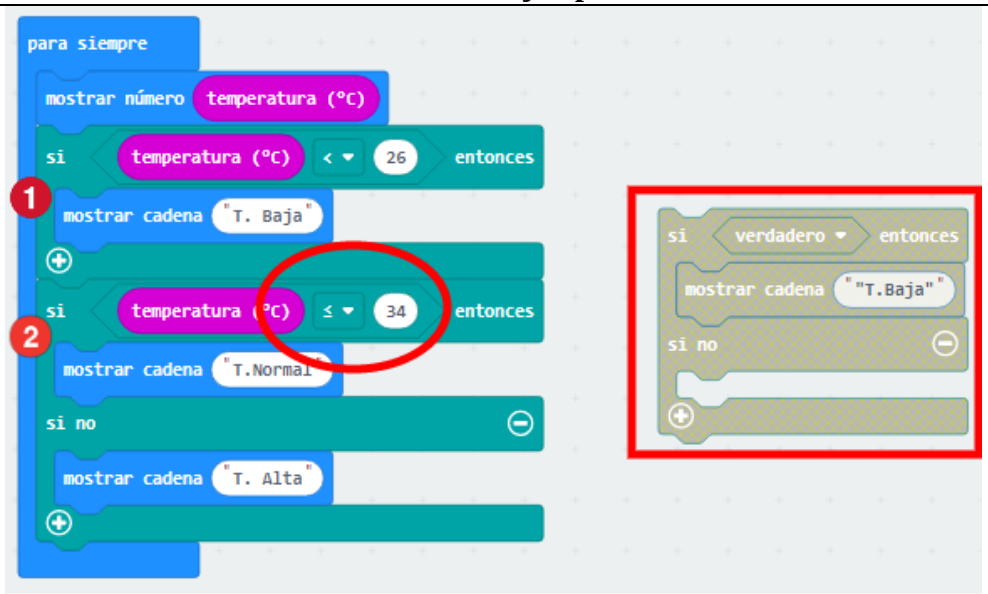
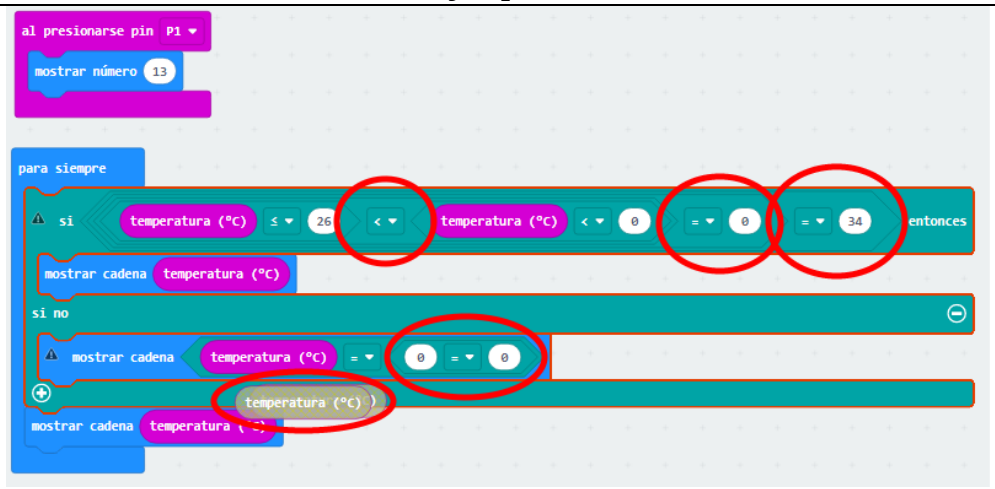


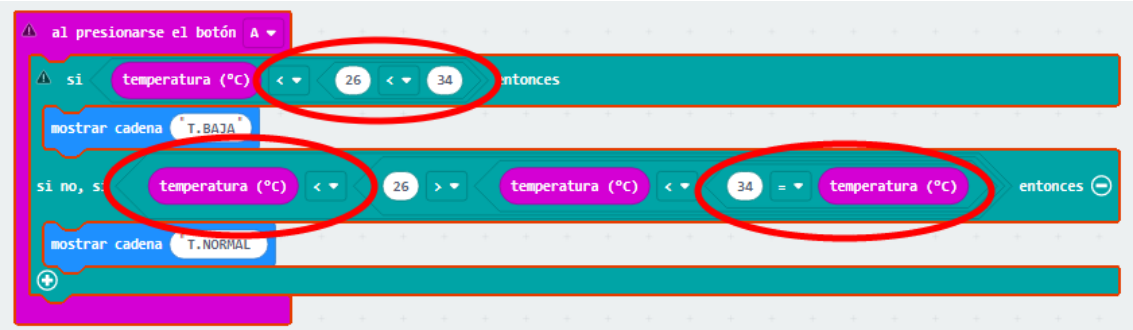
Figura 8

Ejemplos solución unidad 2 estudiantes grupo desconectadas.

Ejemplo 1



Ejemplo 2

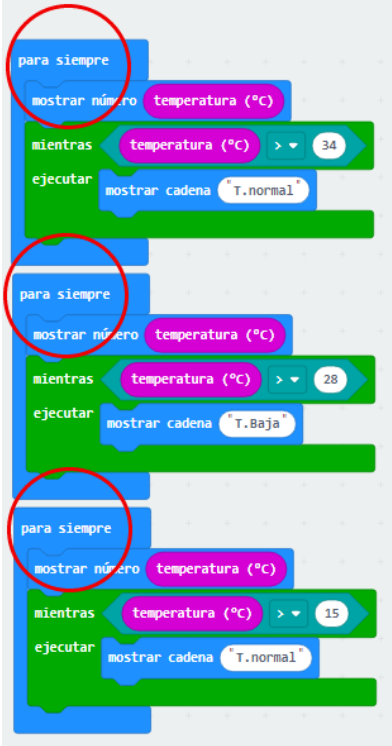


Ejemplo 3

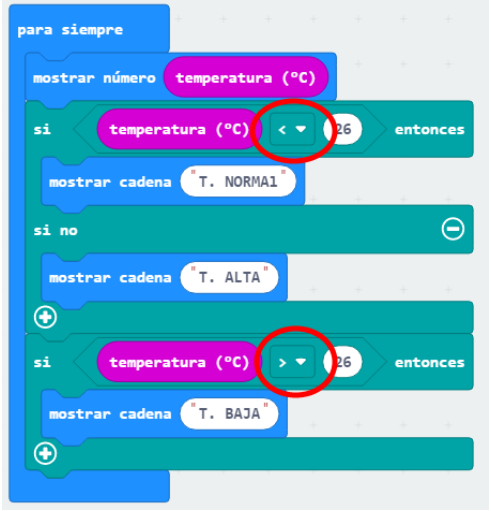
Figura 9

Ejemplos de diagrama de bloques unidad 2 estudiantes grupo control.

Ejemplo 1



Ejemplo 2



Ejemplo 3

Ejemplo 4

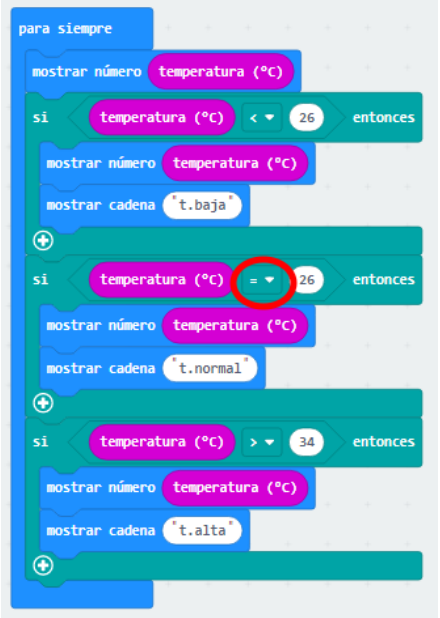
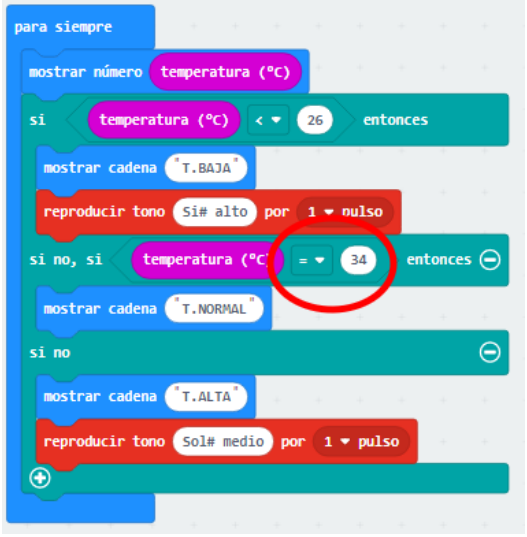


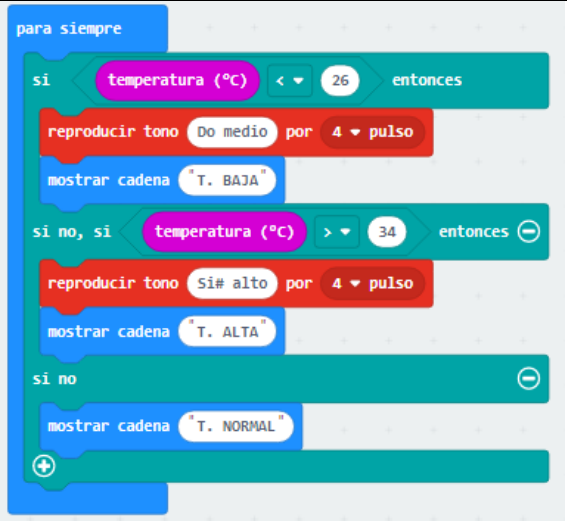
Figura 10

Ejemplos solución reto unidad 2

Ejemplo 1



Ejemplo 2



Ejemplo 3

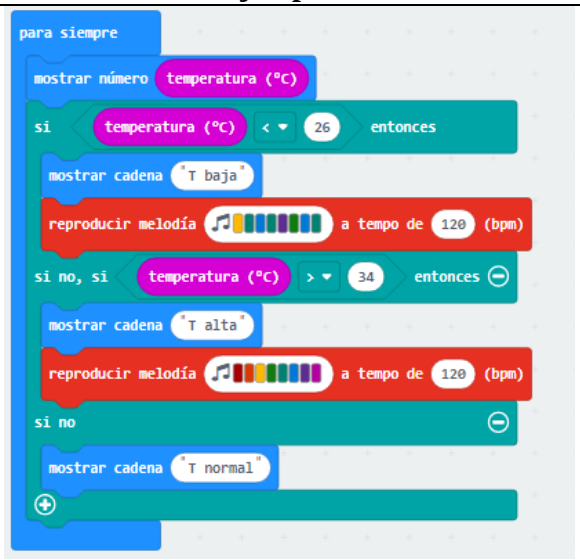
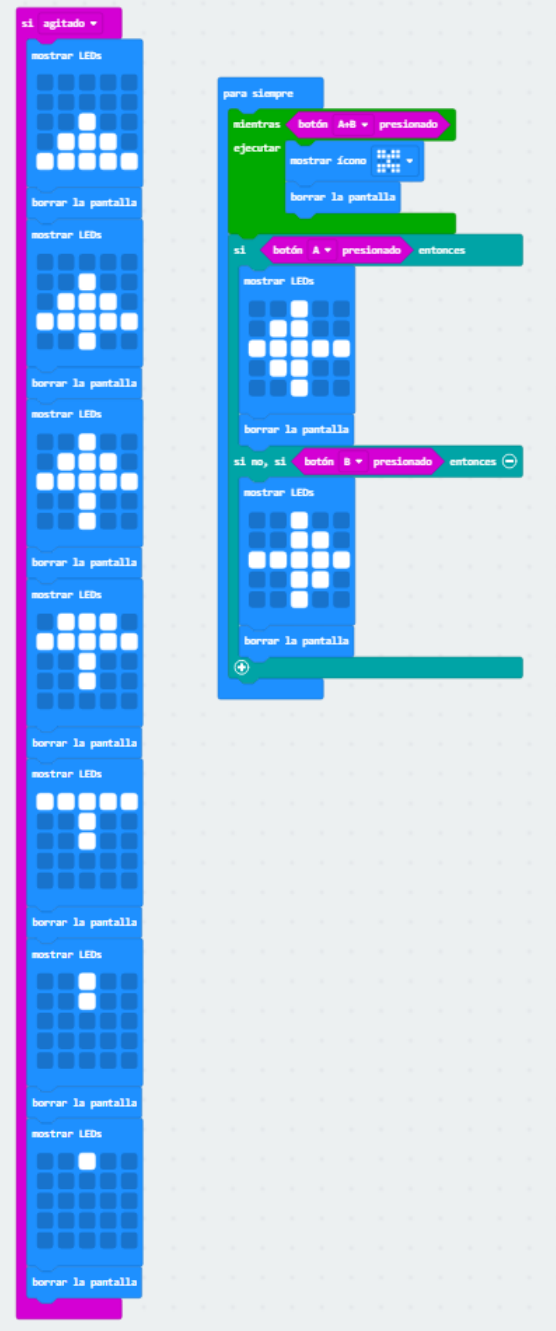


Figura 11

Ejemplos solución general unidad 3 grupo conectadas y desconectadas.

Ejemplo 1



Ejemplo 2

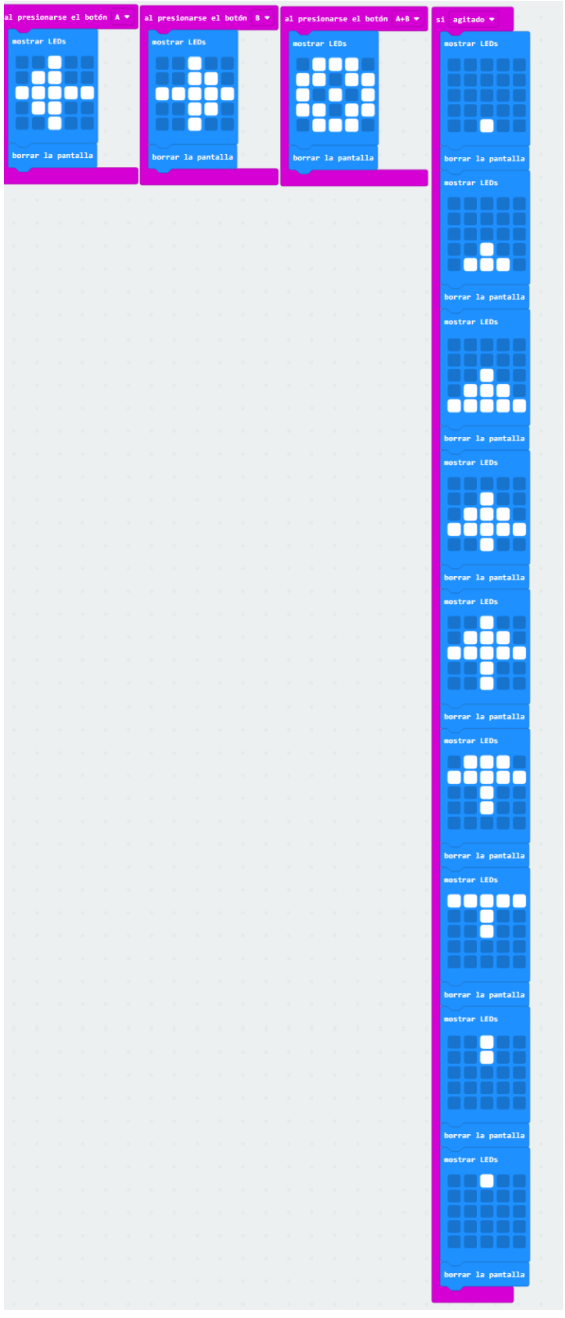


Figura 12

Ejemplo solución general unidad 3 estudiantes grupo de control.

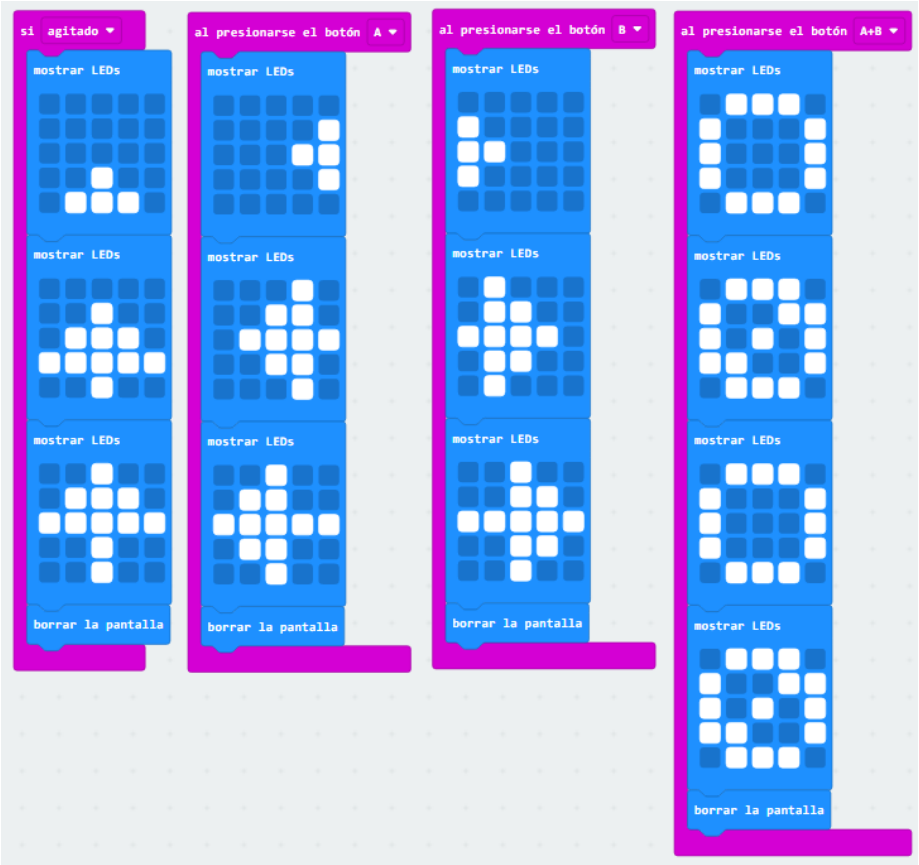
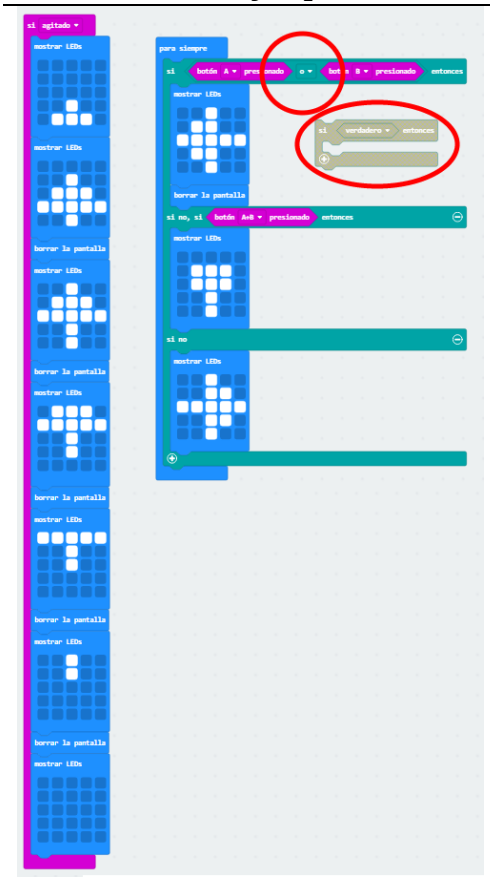


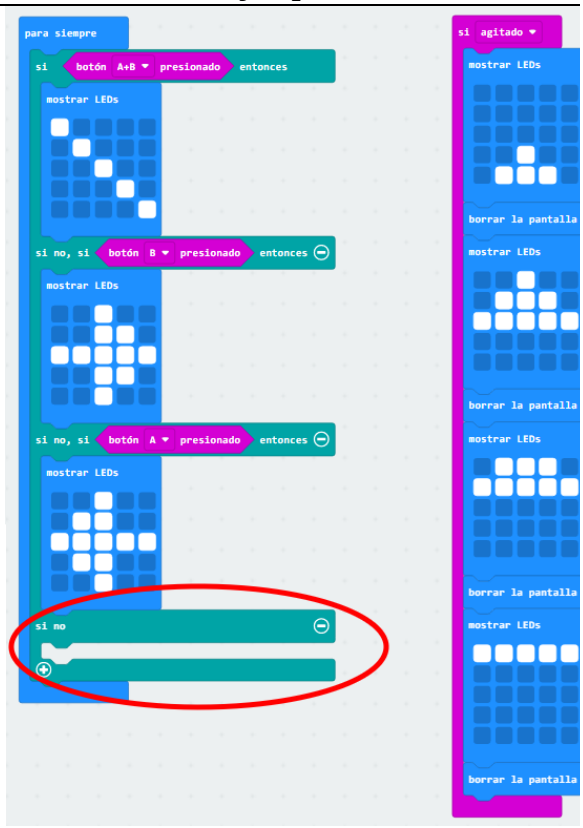
Figura 13

Hallazgos en las soluciones del grupo conectadas para la unidad 3

Ejemplo1



Ejemplo2



Ejemplo3

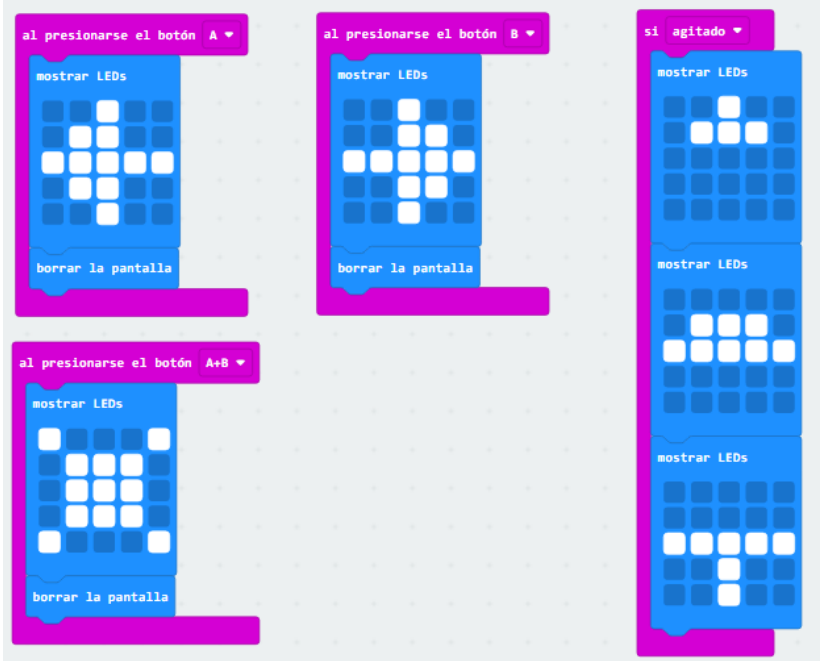
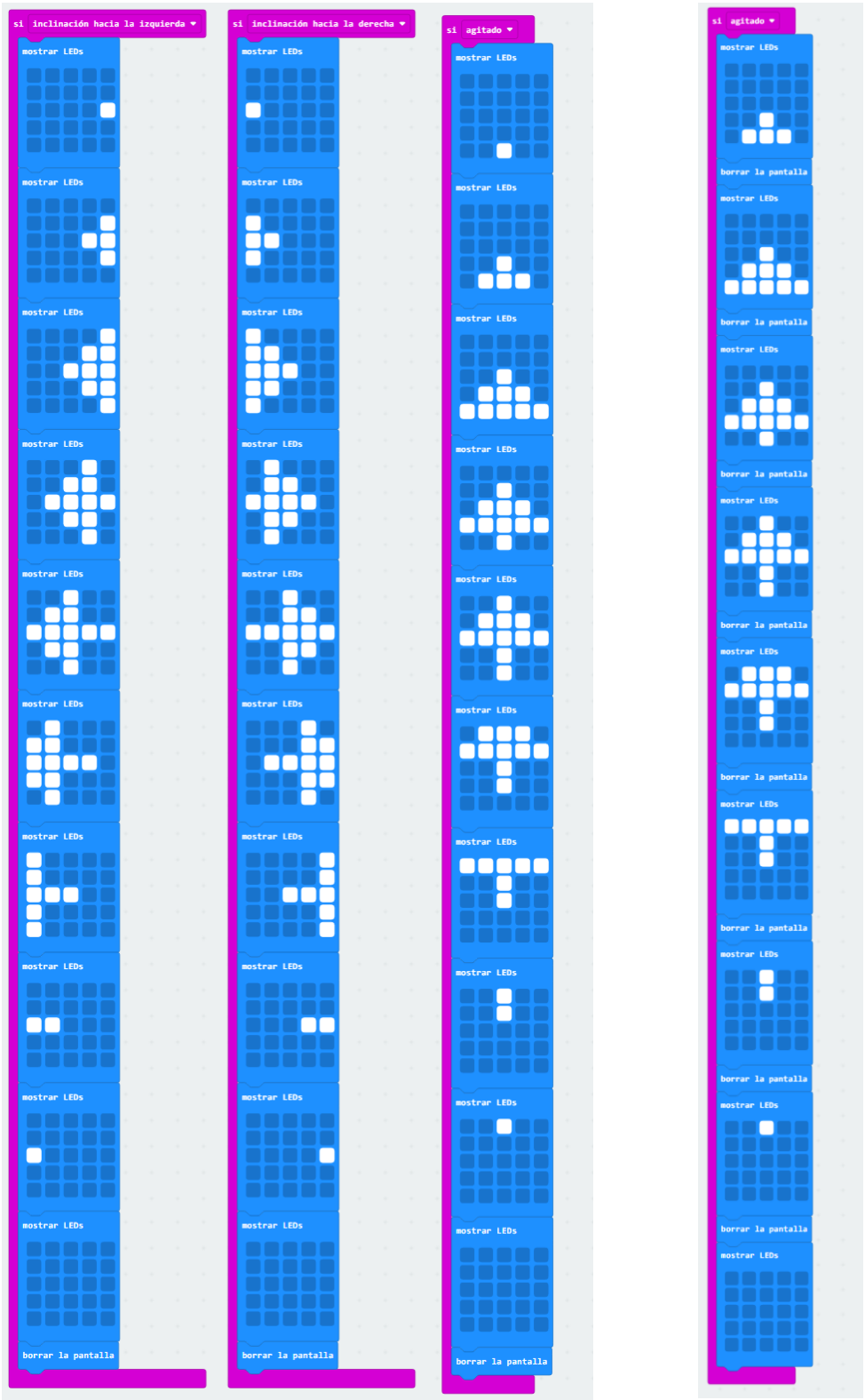


Figura 14

Hallazgos en las soluciones del grupo desconectadas para la unidad 3

Ejemplo 1 **Ejemplo 2**



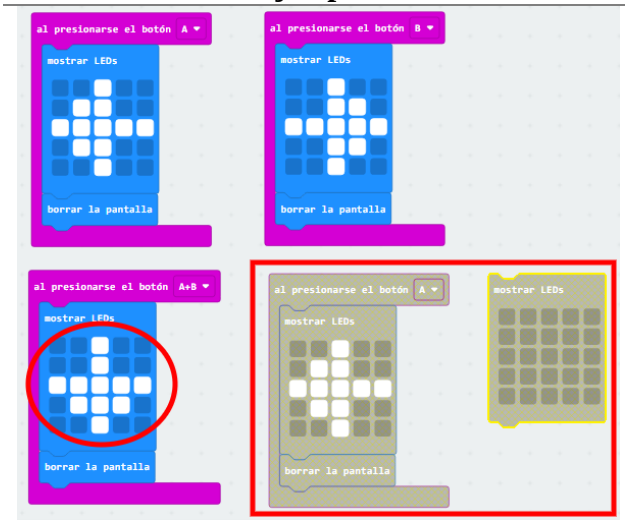
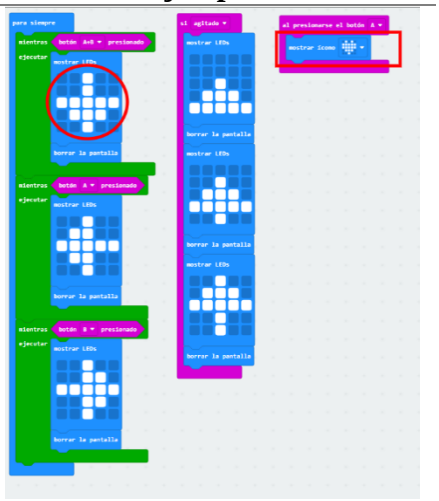
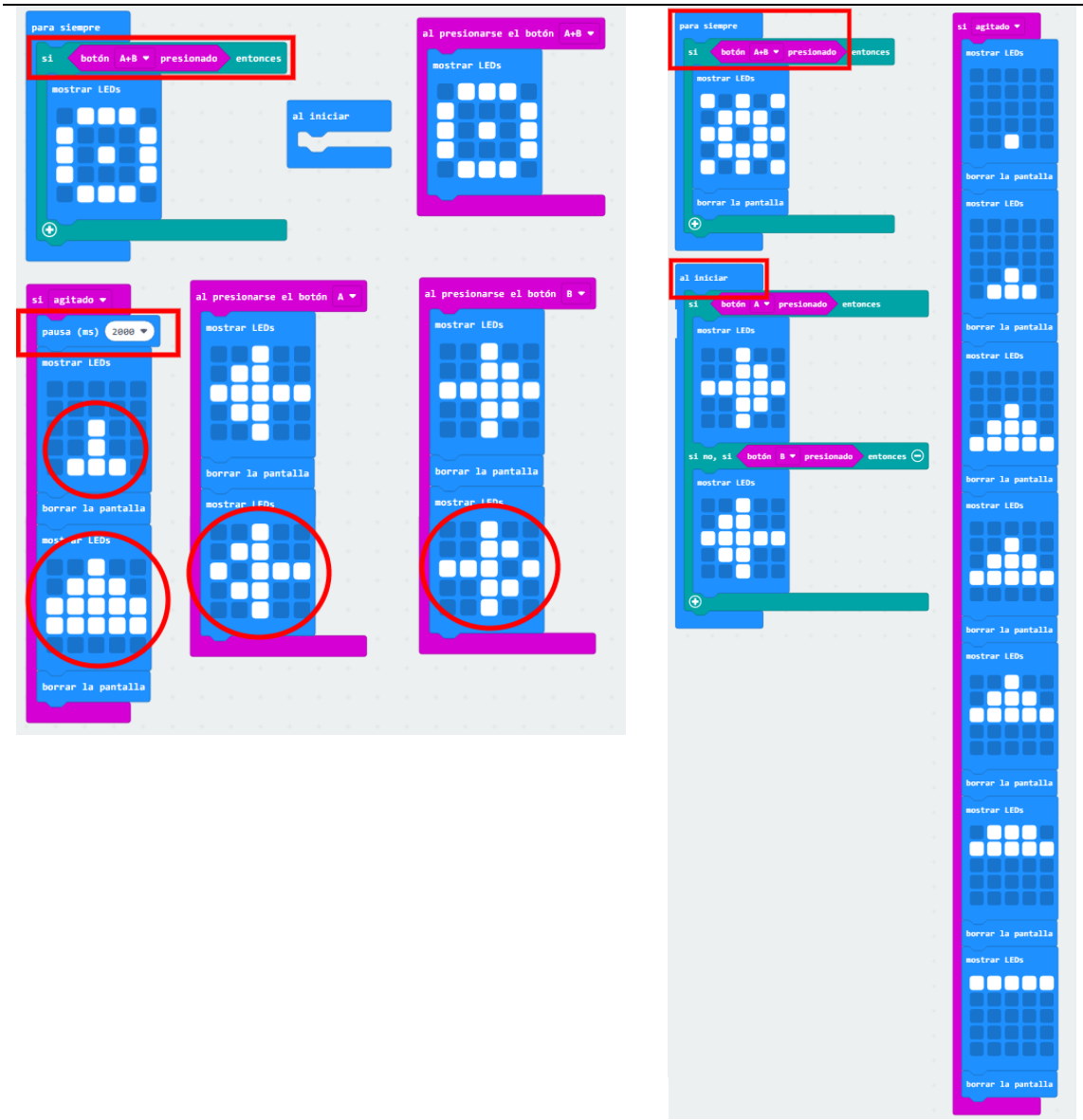
Ejemplo 3	Ejemplo 4
	

Figura 15

Hallazgos en las soluciones del grupo control para la unidad 3

Ejemplo1	*Ejemplo2*


para siempre
si botón A+B presionado entonces
mostrar LEDs
al iniciar
al presionarse el botón A+B
mostrar LEDs
si agitado
pausa (ms) 2000
mostrar LEDs
borrar la pantalla
mostrar LEDs
borrar la pantalla
al presionarse el botón A
mostrar LEDs
borrar la pantalla
mostrar LEDs
al presionarse el botón B
mostrar LEDs
borrar la pantalla
mostrar LEDs
para siempre
si botón A+B presionado entonces
mostrar LEDs
borrar la pantalla
al iniciar
si botón A presionado entonces
mostrar LEDs
si no, si botón B presionado entonces
mostrar LEDs
si agitado
mostrar LEDs
borrar la pantalla
mostrar LEDs
borrar la pantalla
mostrar LEDs
borrar la pantalla
mostrar LEDs
borrar la pantalla
mostrar LEDs
borrar la pantalla
mostrar LEDs
borrar la pantalla
mostrar LEDs
borrar la pantalla

Ejemplo3


si agitado
mostrar LEDs
mostrar LEDs
borrar la pantalla
al presionarse el botón A
mostrar LEDs
mostrar LEDs
mostrar LEDs
borrar la pantalla
al presionarse el botón B
mostrar LEDs
mostrar LEDs
mostrar LEDs
borrar la pantalla
al presionarse el botón A+B
mostrar LEDs
mostrar LEDs
borrar la pantalla
mostrar LEDs
mostrar LEDs

Figura 16

Ejemplo solución reto unidad 3

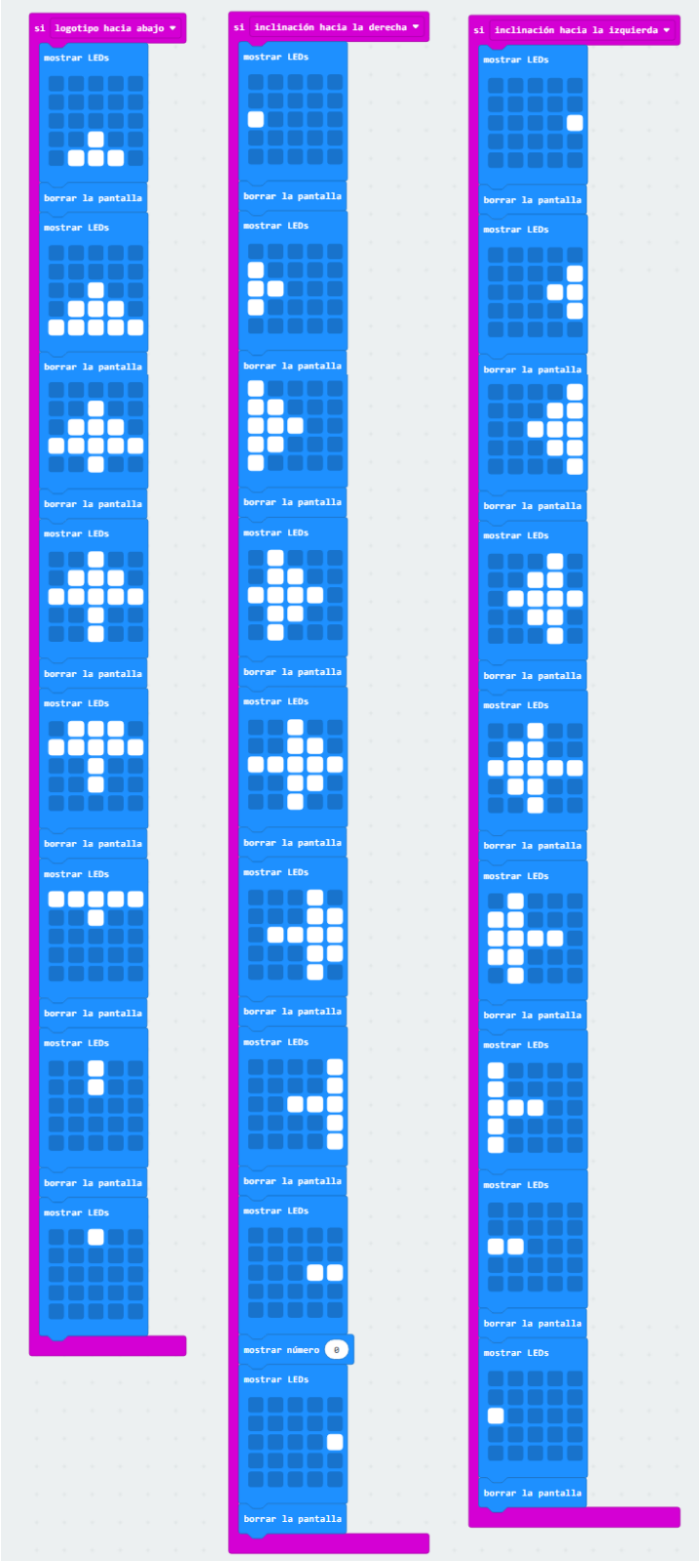


Figura 17

Hallazgos en las soluciones del grupo desconectadas para la unidad 4

Ejemplo 1

Ejemplo 2

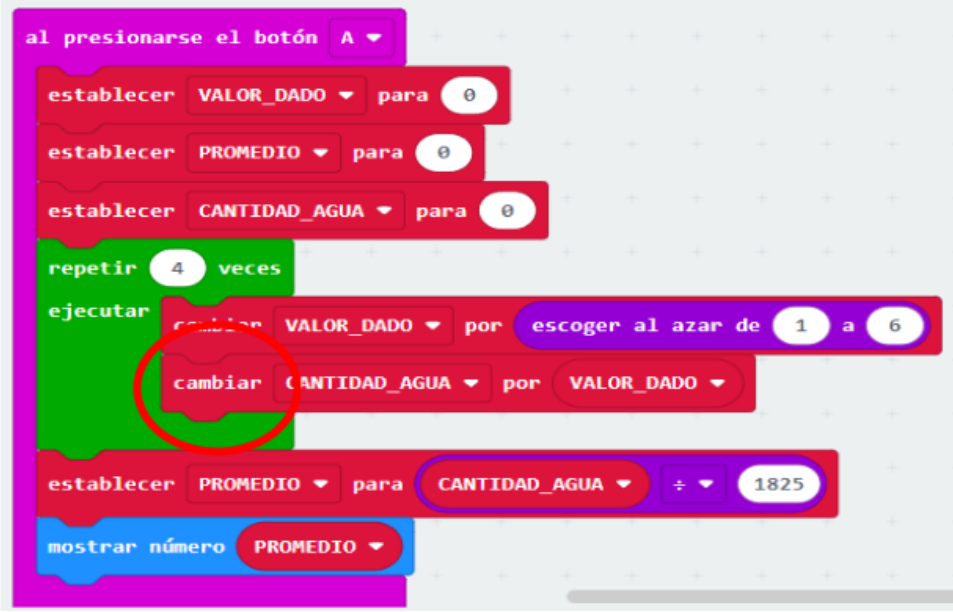


Figura 18

Ejemplo solución general unidad 4 estudiantes grupo desconectadas.

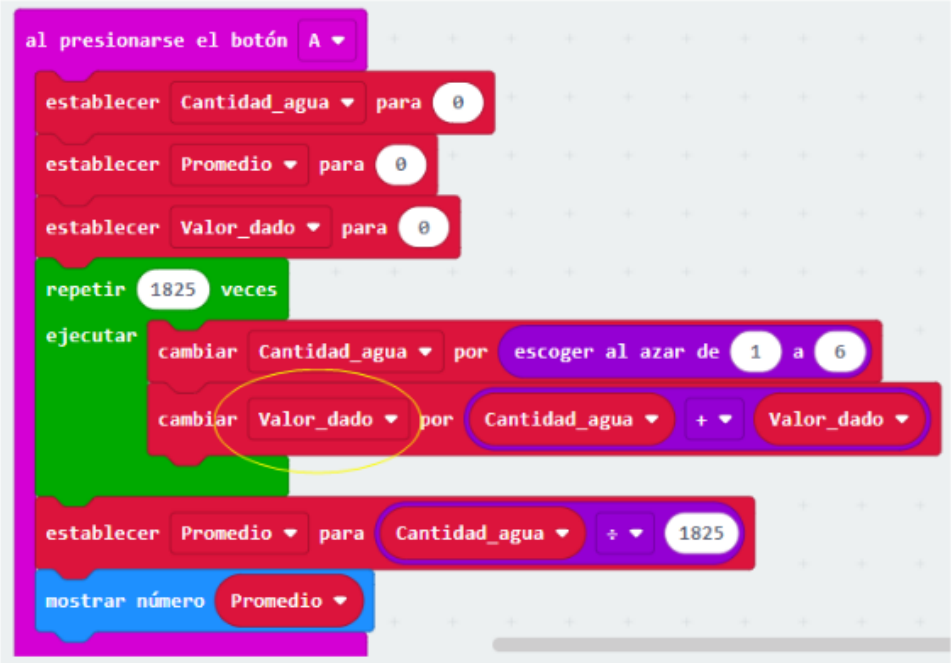


Figura 19

Ejemplos de diagrama de bloques unidad 4 estudiantes grupo conectadas.

Ejemplo 1

Ejemplo 2

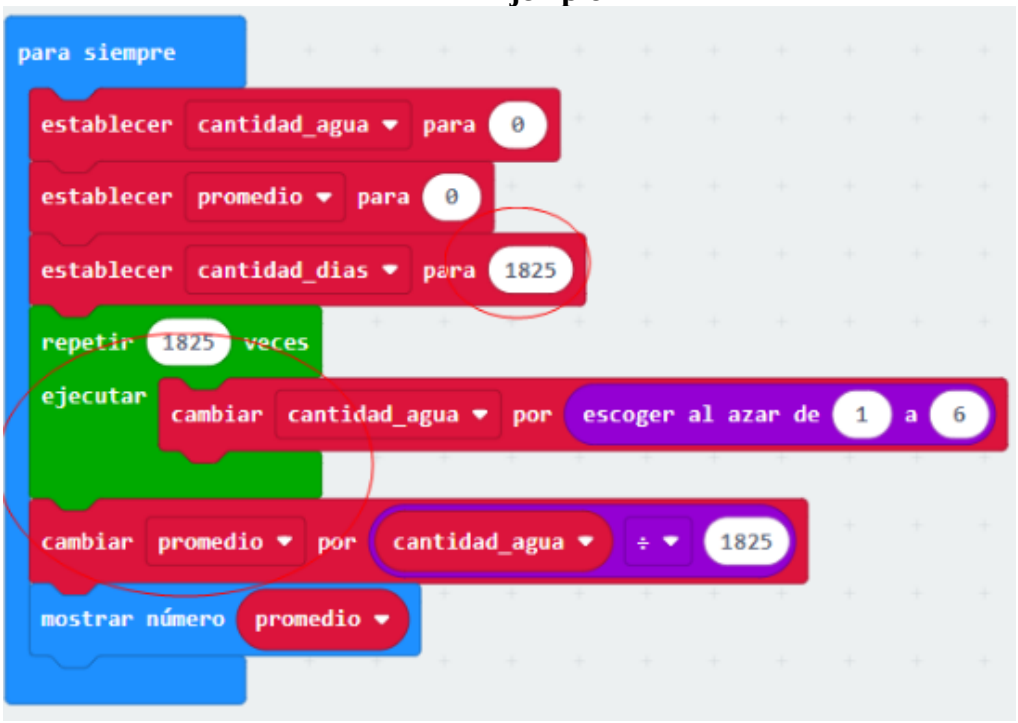


Figura 20

Ejemplo solución general unidad 4 estudiantes grupo de control.

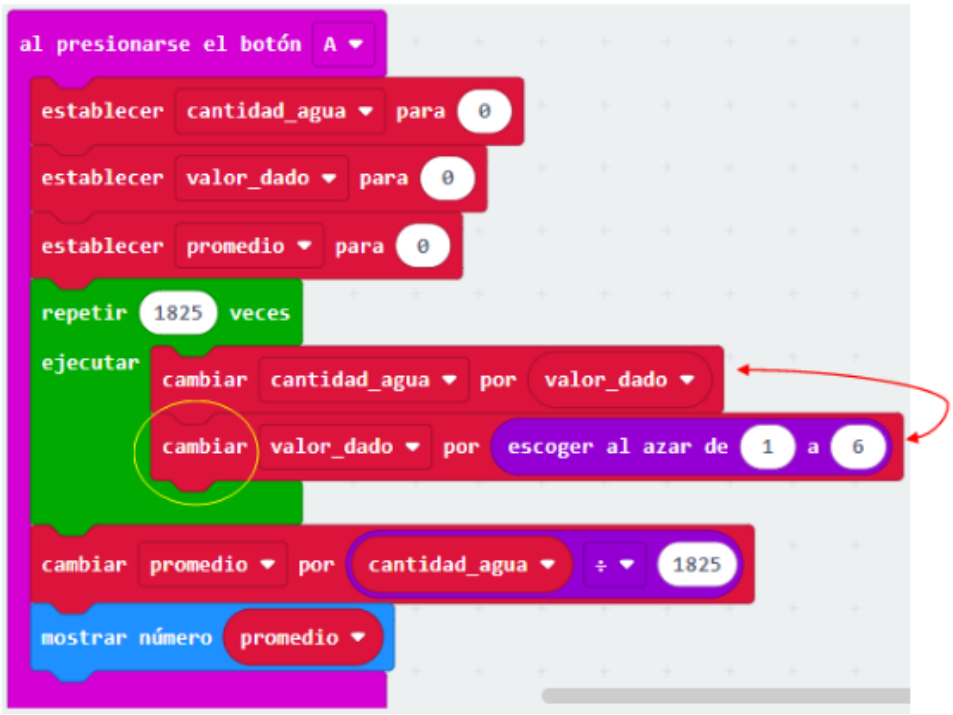
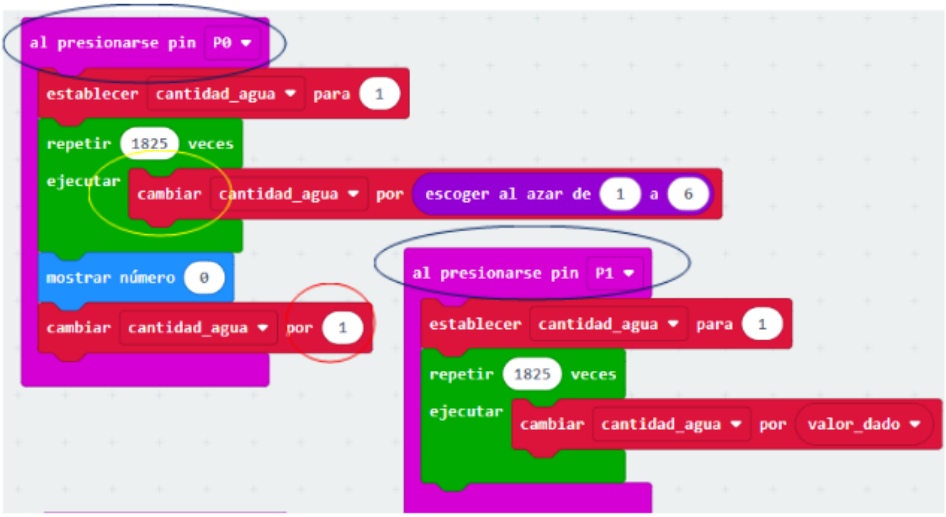


Figura 21

Errores en las soluciones del grupo control para la unidad 4

Ejemplo 1



Ejemplo2

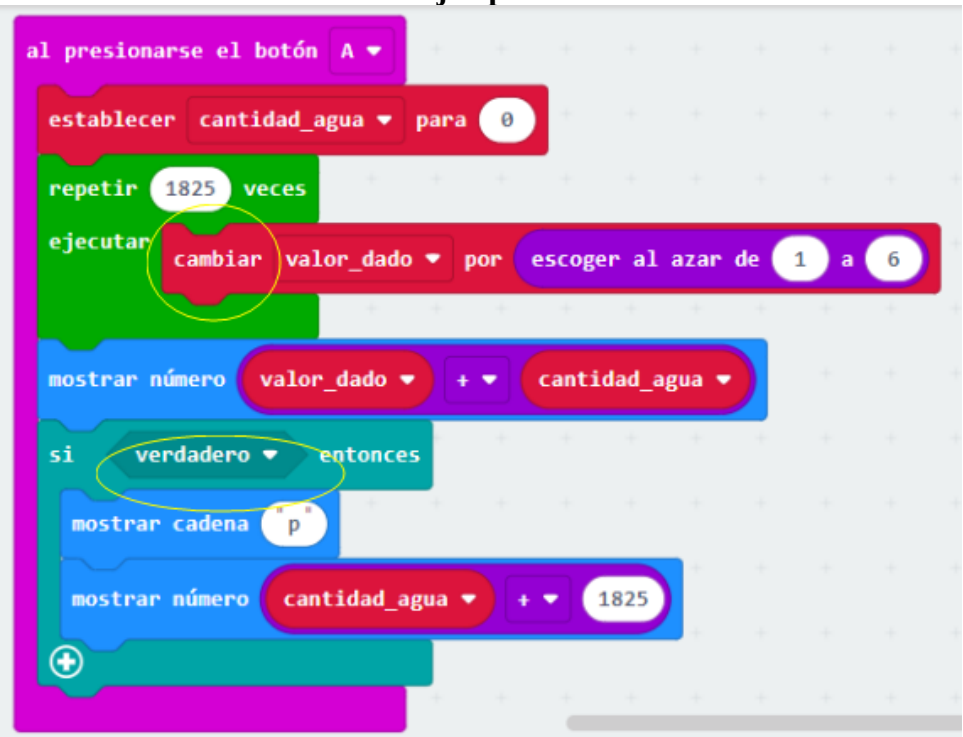


Figura 22

Ejemplo de solución del reto

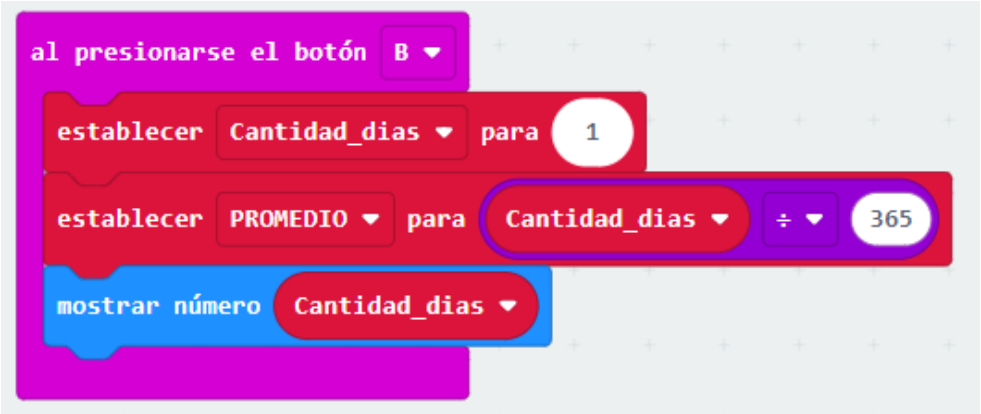
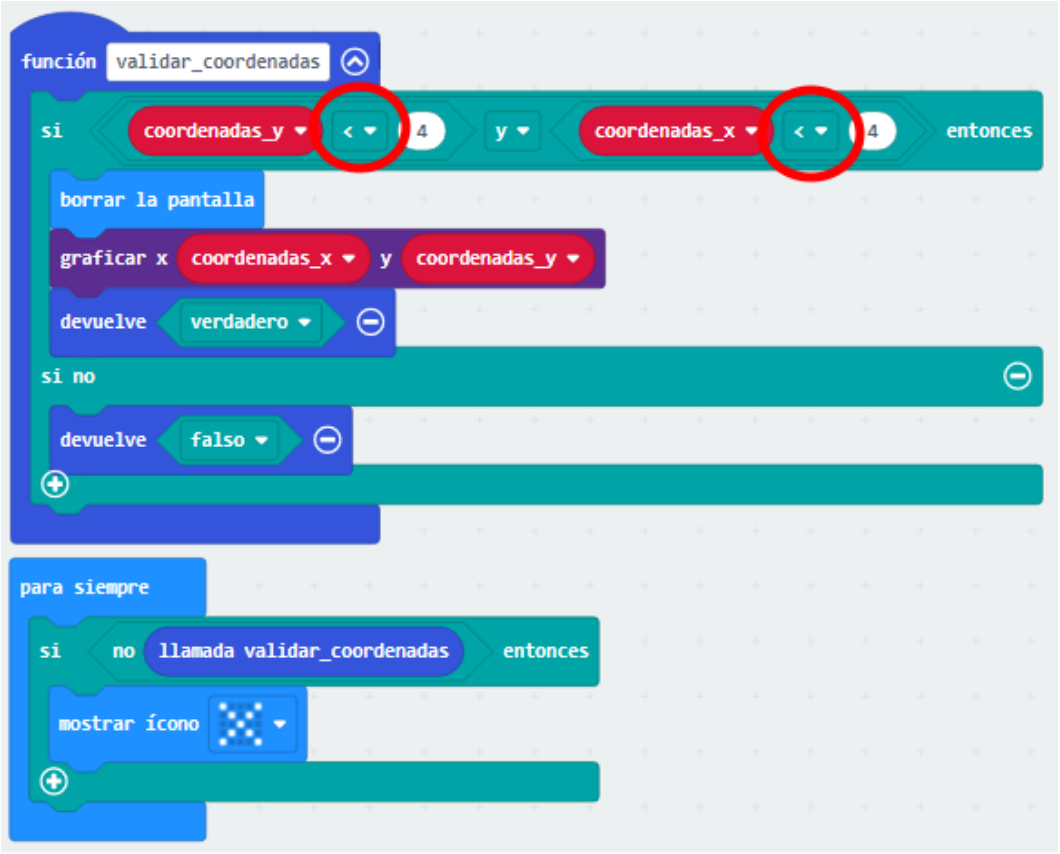


Figura 23

Ejemplos de diagrama de bloques unidad 5 estudiantes grupo conectadas.



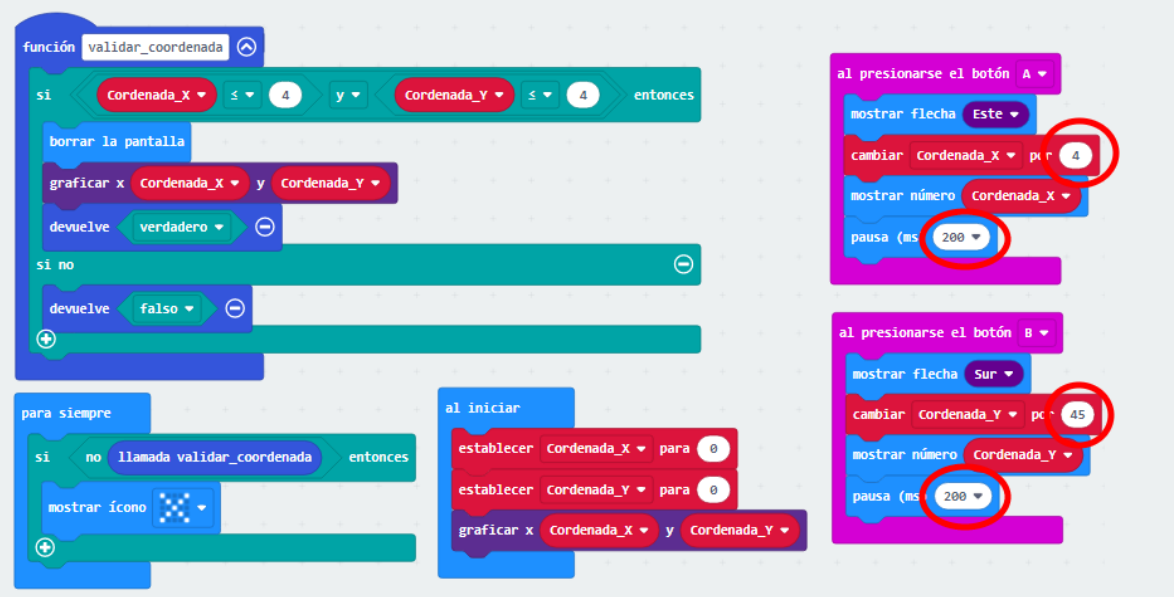
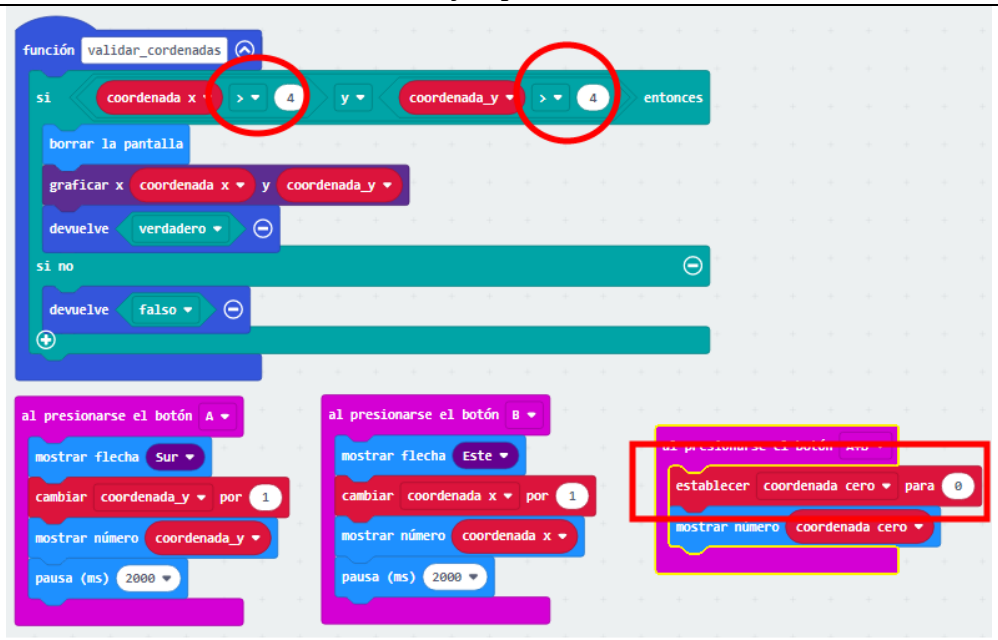


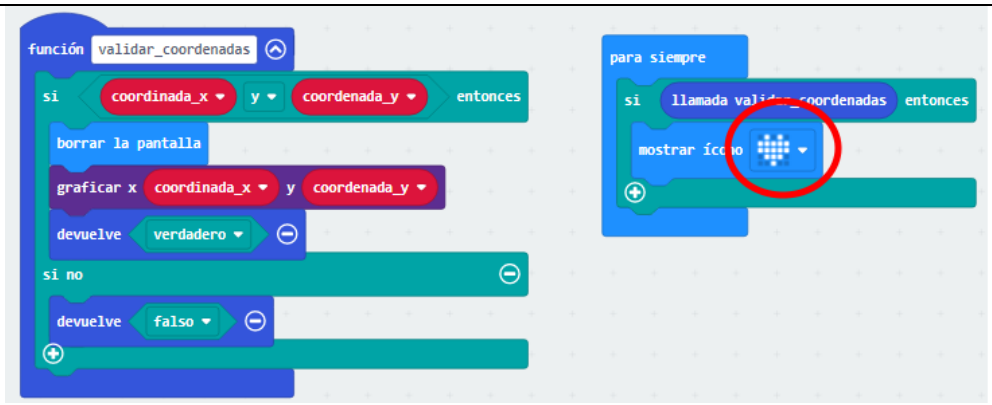
Figura 24

Ejemplo diagrama de bloques unidad 5 estudiantes grupo desconectadas.

Ejemplo 1



Ejemplo 2


función validar_coordenadas
si coordinada_x y coordenada_y entonces
borrar la pantalla
graficar x coordinada_x y coordenada_y
devuelve verdadero
si no
devuelve falso
para siempre
si entonces
mostrar ícono

Ejemplo 3

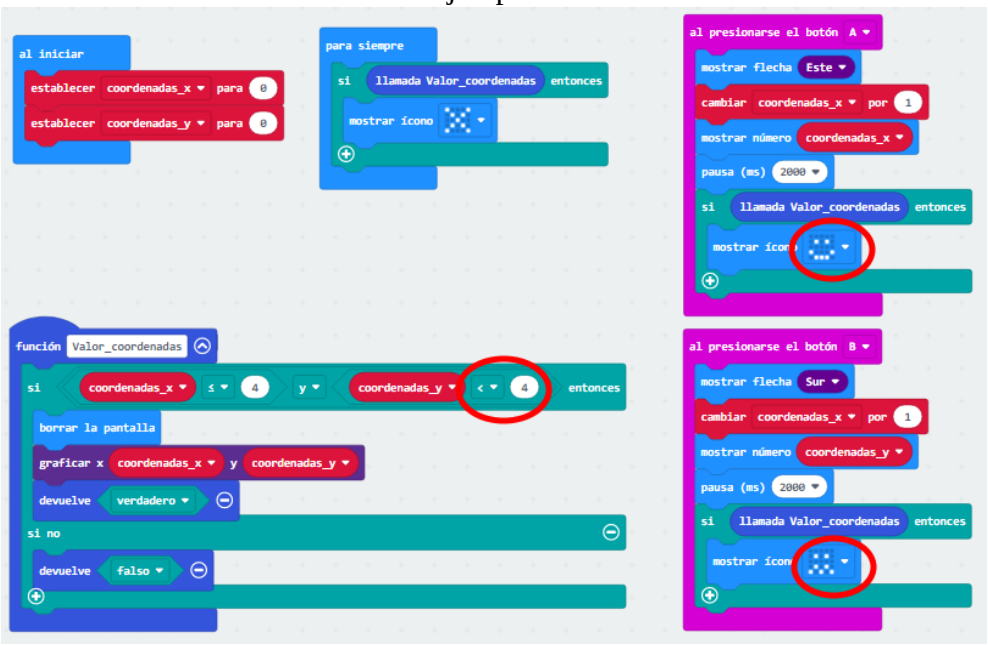

al iniciar
establecer coordenadas_x para 0
establecer coordenadas_y para 0
para siempre
si llamada Valor_coordenadas entonces
mostrar ícono
al presionarse el botón A
mostrar flecha Este
cambiar coordenadas_x por 1
mostrar número coordenadas_x
pausa (ms) 2000
si llamada Valor_coordenadas entonces
función Valor_coordenadas
si coordenadas_x ≤ 4 y coordenadas_y < 4 entonces
borrar la pantalla
graficar x coordenadas_x y coordenadas_y
devuelve verdadero
si no
devuelve falso
al presionarse el botón B
mostrar flecha Sur
cambiar coordenadas_x por 1
mostrar número coordenadas_y
pausa (ms) 2000
si llamada Valor_coordenadas entonces

Figura 25

Ejemplo solución unidad 5 estudiantes grupo de control.

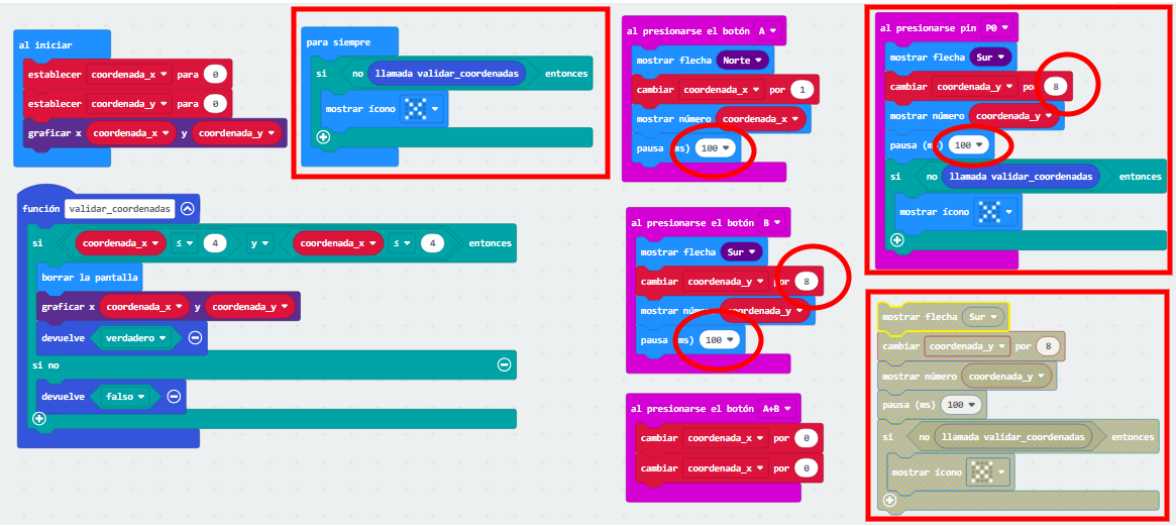


Figura 26

Ejemplo solución unidad 5 estudiantes grupo de control.

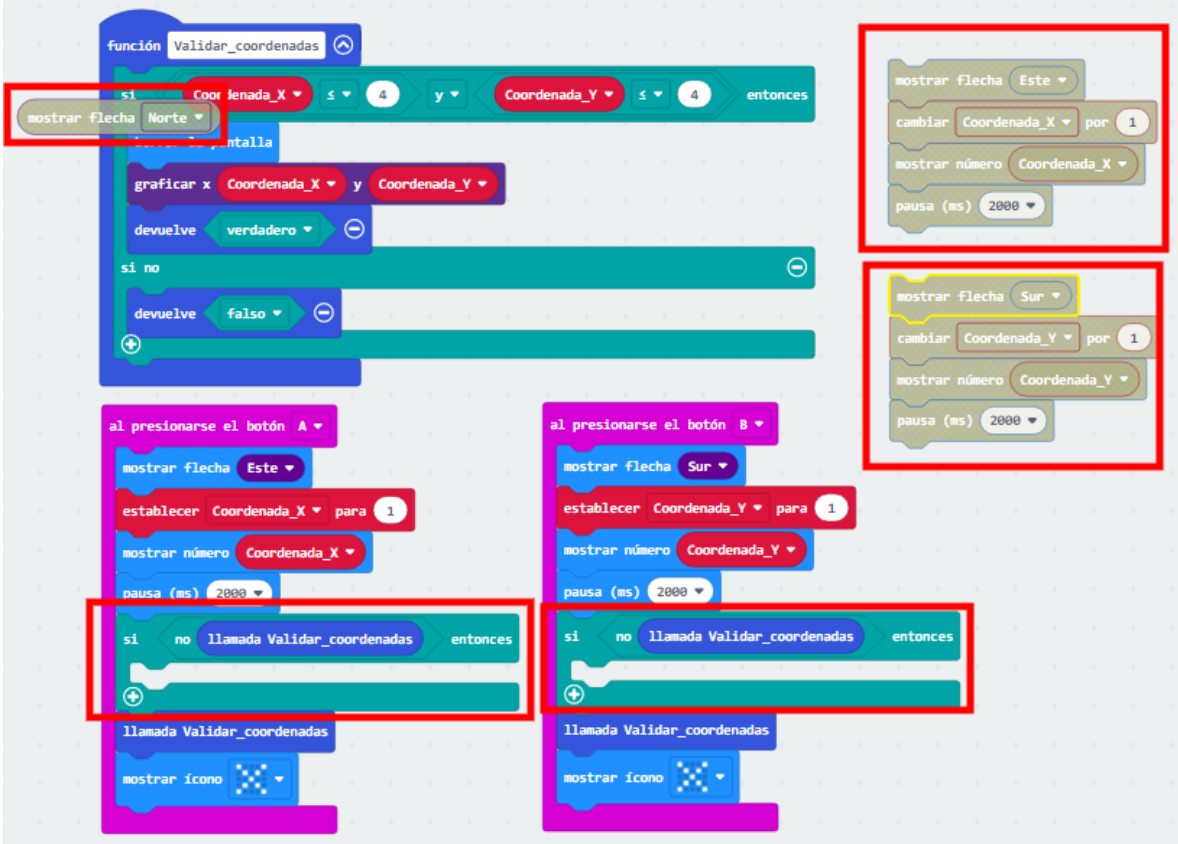
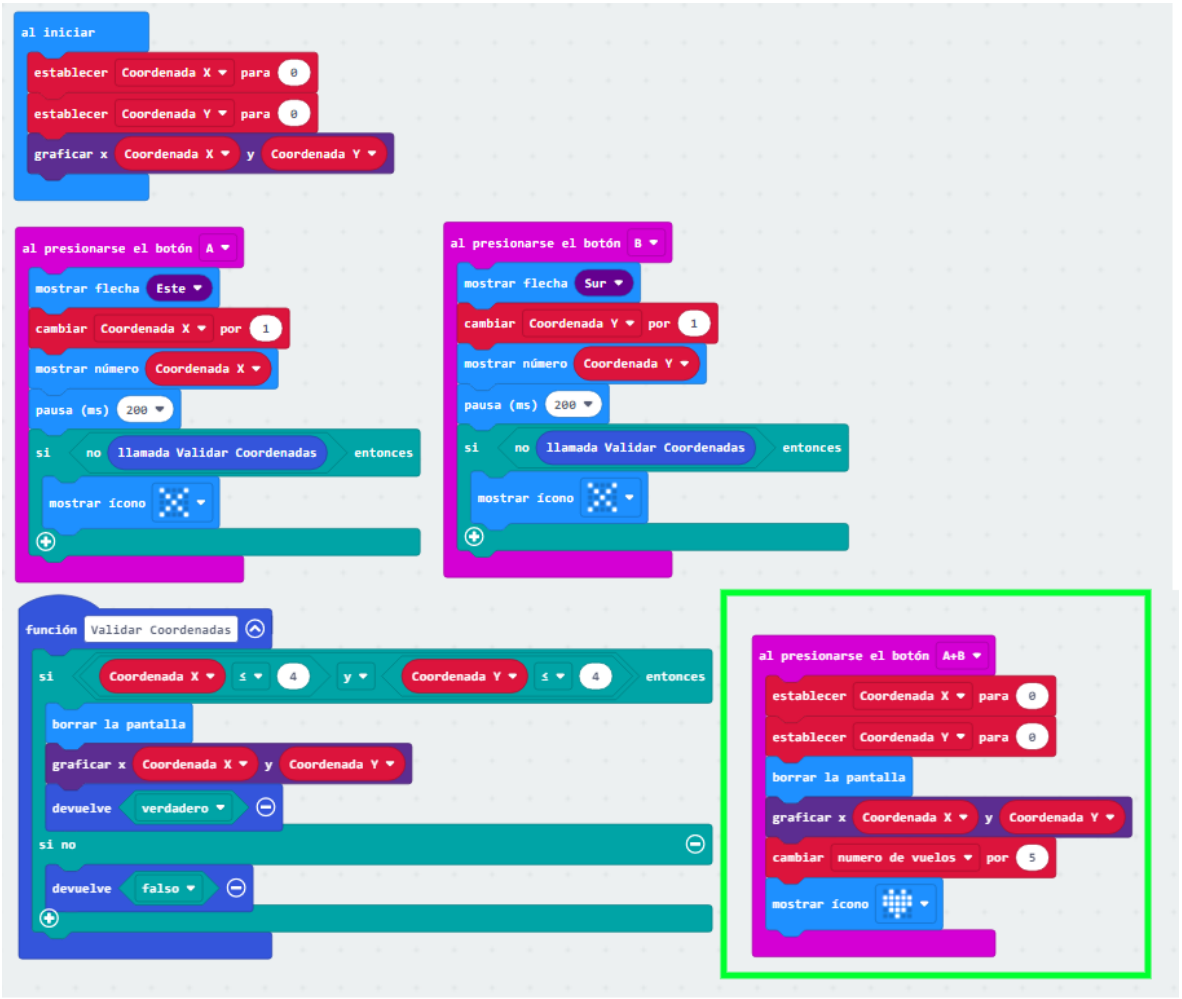


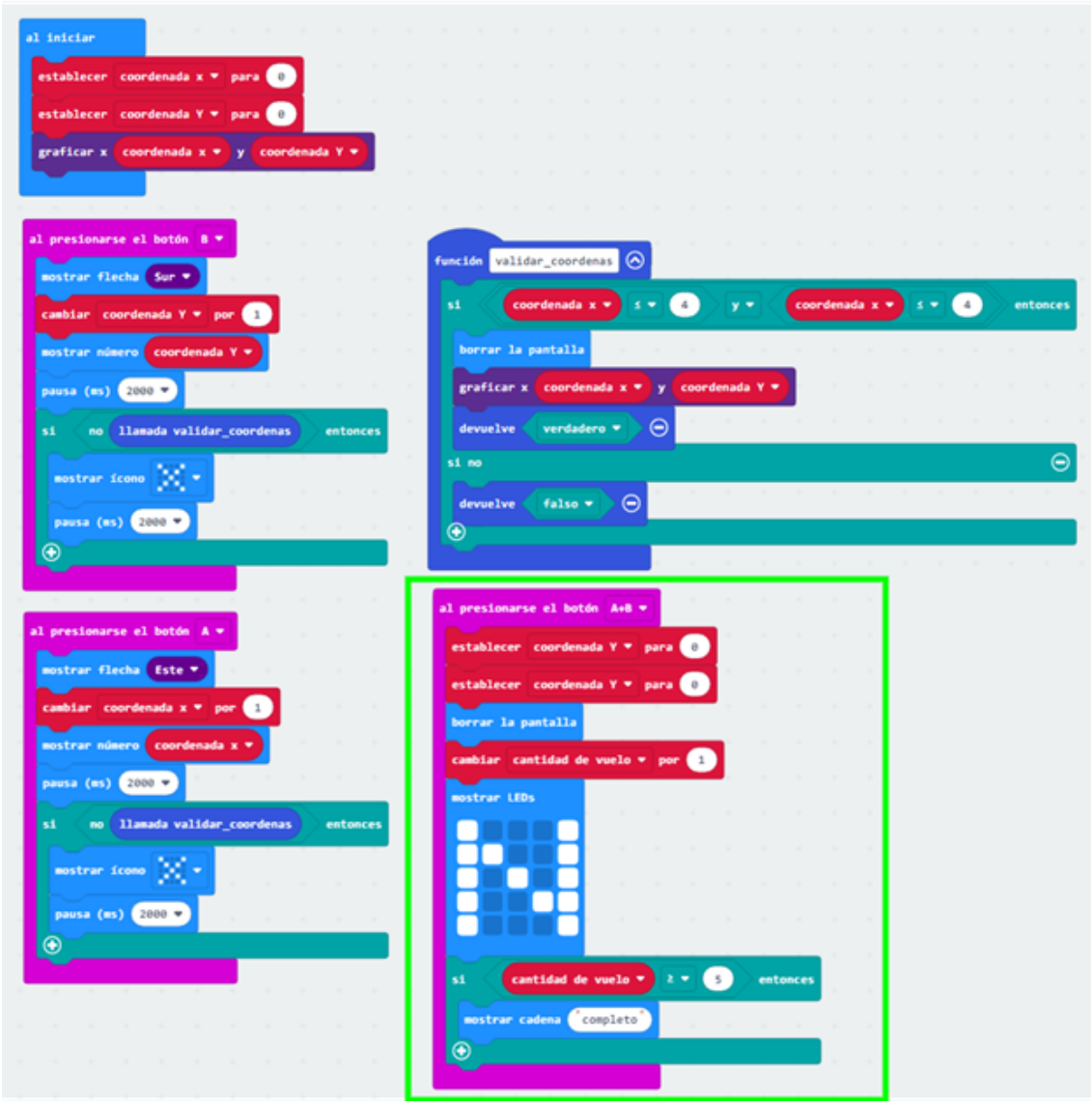
Figura 27

Solución reto unidad 5

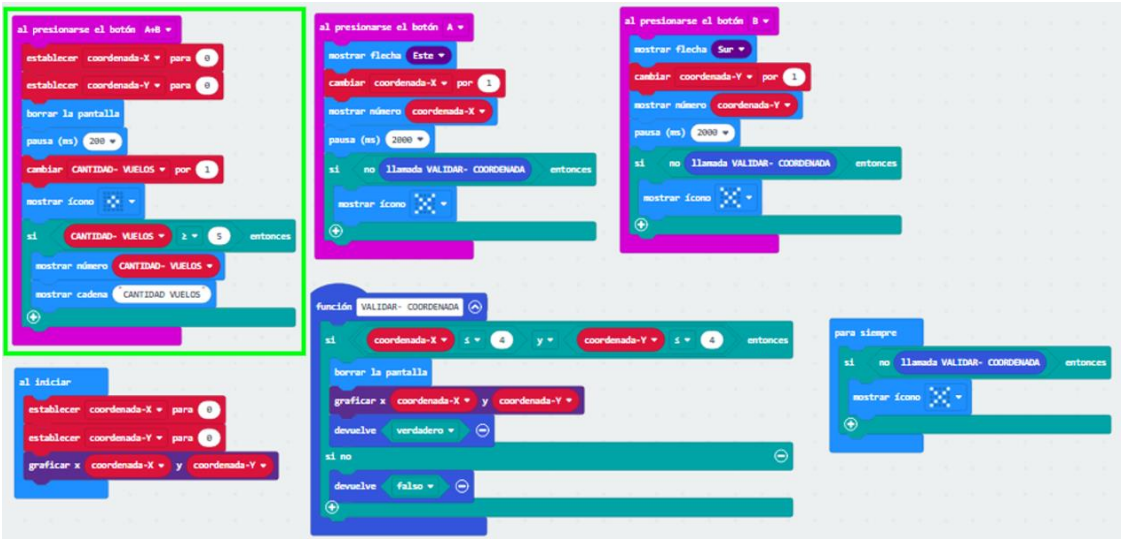
Ejemplo 1



Ejemplo2


al iniciar
establecer coordenada x para 0
establecer coordenada Y para 0
graficar x coordenada x y coordenada Y
al presionarse el botón B
mostrar flecha Sur
cambiar coordenada Y por 1
mostrar número coordenada Y
pausa (ms) 2000
si no llamada validar_coordenas entonces
mostrar ícono
pausa (ms) 2000
función validar_coordenas
si coordenada x ≤ 4 y coordenada x ≤ 4 entonces
borrar la pantalla
graficar x coordenada x y coordenada Y
devuelve verdadero
si no
devuelve falso
al presionarse el botón A
mostrar flecha Este
cambiar coordenada x por 1
mostrar número coordenada x
pausa (ms) 2000
si no llamada validar_coordenas entonces
mostrar ícono
pausa (ms) 2000
al presionarse el botón A+B
establecer coordenada Y para 0
establecer coordenada Y para 0
borrar la pantalla
cambiar cantidad de vuelo por 1
mostrar LEDs
si cantidad de vuelo ≥ 5 entonces
mostrar cadena completo

Ejemplo 3


al presionarse el botón A+B
establecer coordenada-X para 0
establecer coordenada-Y para 0
borrar la pantalla
pausa (ms) 200
cambiar CANTIDAD- VUELOS por 1
mostrar ícono
si CANTIDAD- VUELOS ≥ 5 entonces
mostrar número CANTIDAD- VUELOS
mostrar cadena CANTIDAD VUELOS
al presionarse el botón A
mostrar flecha Este
cambiar coordenada-X por 1
mostrar número coordenada-X
pausa (ms) 2000
si no llamada VALIDAR- COORDENADA entonces
mostrar ícono
al presionarse el botón B
mostrar flecha Sur
cambiar coordenada-Y por 1
mostrar número coordenada-Y
pausa (ms) 2000
si no llamada VALIDAR- COORDENADA entonces
mostrar ícono
al iniciar
establecer coordenada-X para 0
establecer coordenada-Y para 0
graficar x coordenada-X y coordenada-Y
función VALIDAR- COORDENADA
si coordenada-X ≤ 4 y coordenada-Y ≤ 4 entonces
borrar la pantalla
graficar x coordenada-X y coordenada-Y
devuelve verdadero
si no
devuelve falso
para siempre
si no llamada VALIDAR- COORDENADA entonces
mostrar ícono