Facultad de Educación

MÁSTER EN FORMACIÓN DEL PROFESORADO DE EDUCACIÓN SECUNDARIA

Propuesta de uso de la plataforma micro:bit para le enseñanza de la programación en Bachillerato.

A Proposal for the use of the micro:bit platform for teaching programming in high school.

Carlos Alberto Meneses Agudo

Física, Química y Tecnología

Ángel Cuesta García

19-20

Septiembre de 2020

Índice

I. Resumen 3
II. Abstract 4
III. Introducción 5
1. Situación actual y futuro 5
2. Ciencias de la computación y pensamiento computacional 6
3. Pensamiento computacional y programación 6
4. Iniciativas STEM 7
5. Objetivo 8
6. Estructura del documento 8
IV. Estado de la cuestión 9
1. Contexto 9
2. Situación en España 12
3. Situación en Cantabria 14
V. Herramientas tecnológicas 24
1. Placas programables 24
2. Lenguaje de programación 28
VI. Propuesta didáctica: Introducción a la programación textual en 1º de Bachillerato con micro:bit y JavaScript. 32
1. Micro:bit 33
2. Guía de inicio rápido. 35
3. Entorno de desarrollo MakeCode. 36
4. Propuesta didáctica. 38
5. Competencias involucradas 45
6. Evaluación 47
VII. Conclusiones 49
VIII. Bibliografía 51
IX. Anexo: Prácticas 56
1. P0: Primeros pasos con micro:bit y JavaScript. 56
2. P1: Variables y constantes. 59
3. P2: Funciones y métodos. 64
4. P3 Estructuras de control I 68
5. P4 Estructuras de control II. 71
6. P5 Proyecto. 73

I. Resumen

Este trabajo de Fin de Máster consiste en una propuesta didáctica con el objetivo de introducir la programación textual en 1º de Bachillerato. Debido al amplio crecimiento de plataformas basadas en placas programables en proyectos de robótica las introduciremos en la propuesta como un soporte físico donde poder experimentar los programas. En esta línea se realiza la primera criba en el presente trabajo: la selección de la placa. Finalmente se optó por la placa micro:bit por ofrecer un sistema sencillo e intuitivo, soportar programación visual y textual y tener un coste muy reducido. Una vez elegida la plataforma se eligió el lenguaje de programación (esto se realizó de esta forma ya que la placa limitaba las opciones de lenguajes disponibles). Elegimos JavaScript como lenguaje vehicular de nuestra propuesta por su alta popularidad.

La propuesta didáctica se debe tomar como una propuesta parcial y no una unidad didáctica completa. Ésta consiste en 6 sesiones teórico-prácticas en las que se parte desde lo básico hasta una sesión final en la que la tarea primordial es comenzar a diseñar de forma creativa una solución programática a un problema concreto. Además, la propuesta didáctica conlleva una fuerte contribución a las competencias clave de la etapa educativa y, además, se ha permitido el uso de distintas metodologías didácticas en las distintas sesiones: aprendizaje basado en proyectos, aprender haciendo, aprendizaje colaborativo, gamificación o aprendizaje basado en competencias.

Palabras clave: Programación, robótica, micro:bit, JavaScript, pensamiento computacional, competencias.

II. Abstract

The objective of this work is to elaborate a didactic proposal in charge of introducing the textual programming in the subject of Information and Communication Technologies in 2nd year of High School. Due to the growth of the use of programmable boards and other robotic platforms in educational projects we have decided to use them as a physical complement to experience the results of the programs. Firstly, we had to select the programmable board. In the end we chose micro:bit because it is a very simple and intuitive system, supports visual and textual programming and is very economical. Then the programming language had to be chosen. JavaScript was chosen because of its high popularity in academic and professional environments.

The didactic proposal presented in the document must be treated as a partial proposal and not as a complete didactic unit. Our proposal consists of 6 theoretical-practical sessions starting from the basics to a higher level of complexity. This final session will have as its main task the creation of a software solution to a problem without explicit guidance. Furthermore, the proposal makes a strong contribution to the core competences of the educational stage and the use of different teaching methodologies has been allowed in each session: project based learning, learning by doing, collaborative learning, gamification or competence based learning.

Keywords: Programming, robotics, micro:bit, JavaScript, computer thinking, competencies.

III. Introducción

1. Situación actual y futuro

Las Tecnologías de la Información y Comunicación es uno de los sectores más relevantes y condicionantes en el progreso de nuestras sociedades. Este motivo es radical a la hora de decidir formar ciudadanos tecnológicamente adaptables a la sociedad actual y a las venideras y que, a su vez, sean capaces de participar en estas futuras transformaciones. Estudios procedentes de la OCDE pronostican que numerosos puestos de trabajo corren peligro debido a esta extrema digitalización y automatización (Cristina Delgado, 2019). Sobre todo, en sectores de manufacturación, servicios, etc. Por otro lado, los empleos que requieren cierta creatividad, un alto nivel de complejidad se verá en auge.

Debido al espectacular aumento de la importancia que tienen las Ciencias de la Computación en la sociedad actual, tanto en la vida cotidiana como en ámbitos profesionales, muchos países han adoptado estrategias de incorporación en las etapas obligatorias de su currículo (García-Peñalvo, 2016; Hubwieser et al., 2015). Dichas estrategias varían de forma muy dispar a lo largo de los países y sistemas educativos. Así nos encontramos la troncalidad de la asignatura de Computación, tanto en Primaria como Secundaria, en el Reino Unido (U.K, s. f.) o estrategias de más largo recorrido como en Israel (Gal-Ezer y Stephenson, 2014; Hazzan et al., 2008) hasta el escaso tratamiento presente en España (Mercedes Martín López, 2017).

A pesar de la proliferación de diversas iniciativas en nuestro entorno ello no se ha traducido en un avance curricular claro a nivel nacional y todo ha quedado en decisiones por parte de las comunidades autónomas. El auge de estas y las nuevas iniciativas, tanto en nuestro entorno nacional como el de la Unión Europea empujará a la conclusión de introducir a nivel nacional una nueva asignatura que recoja esta necesidad. Queda por determinar la dedicación (troncal u optativa), el enfoque y el modelo (basado en casos de éxito de otros países como Israel o Estonia (Aru-Chabilan, 2020).

2. Ciencias de la computación y pensamiento computacional

Tal como se indica en *The great principles of computing* (Denning, 2003), la Computación puede ser considerada el cuarto gran dominio científico, junto con la Física, la Biología o las Ciencias Sociales. En este artículo las ciencias de la Computación tendrían 7 categorías: "*computation, communication, coordination, recollection, automation, evaluation and design*". Así pues, las ciencias de la computación tienen entidad suficiente como para reclamar su propia asignatura troncal en, al menos, toda la enseñanza secundaria.

De forma reciente el pensamiento computacional se ha convertido en un tema central dentro del marco de iniciativas a nivel global que pretenden llevar las Ciencias de la Computación a la enseñanza preuniversitaria (Riesco Albizu et al., 2014). Debemos tener claro que los términos Computación y Pensamiento Computacional no son conceptos equivalentes. El campo de la computación es mucho más amplio, así como el pensamiento computacional es un conocimiento más abstracto que la programación que debatiremos a continuación.

3. Pensamiento computacional y programación

Aunque no haya una definición académica aceptada sobre pensamiento computacional, en la extensa revisión de la literatura (Adell Segura María Ángeles Llopis Nebot Francesc Esteve Mon María Gracia Valdeolivas Novella, 2019; García-Peñalvo, 2016) podemos encontrar el origen del término en una columna de opinión de Jeannette Wing publicada en 2006 en la que se sostiene lo siguiente:

> El pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, basándose en los conceptos fundamentales de la ciencia de la computación. El pensamiento computacional incluye una amplia variedad de herramientas mentales que reflejan la amplitud del campo de la computación… [además] representa una actitud y unas habilidades universales que todos los individuos, no sólo los científicos computacionales, deberían aprender y usar" (Wing, 2006)

Es decir, Wing definió el pensamiento computacional como un conjunto de habilidades y destrezas, habituales en los profesionales de las ciencias de la computación, pero que todos los seres humanos deberían poseer y utilizar para resolver problemas, diseñar sistemas y, sorprendentemente, comprender el comportamiento humano. Por todo ello afirma que el pensamiento computacional debería formar parte troncal de la educación de todo ser humano.

El pensamiento computacional debe entenderse como una herramienta más, similar a la lectura o las matemáticas. A través de esta herramienta el alumno puede enfrentarse a distintos problemas y, además, resolverlos.

Por otro lado, algunas iniciativas pueden caer en el error de equiparar pensamiento computacional con programación. Se debe entender que la programación es la tarea que nos va a ayudar a desarrollar el pensamiento computacional en los alumnos. Así, el pensamiento computacional nos es algo más cercano y disponible para todo ser humano, algo que se presume alejar cuando se habla de dotar de la habilidad de programar a todo ser humano.

Otra disciplina relacionada con el pensamiento computacional y que nos va a ayudar en nuestra propuesta es la robótica. Esta disciplina va ganando popularidad y nos va a permitir traer al mundo cotidiano nuestras programaciones en una tarea completa. Este carácter práctico resulta imprescindible en nuestra propuesta y, además, refuerza la motivación del alumno al poder observar los resultados de su programación.

4. Iniciativas STEM

El término STEM es el acrónimo en ingles de los términos *Science, Technology, Engineering and Mathematics.* Este término fue acuñado por la *National Science Foundation* en los años 90 (Sanders, 2008). Este término viene ligado con numerosas iniciativas de integración educativa de estos bloques de forma conjunta en la que encontramos dos características bien diferenciadas.

- Proceso de enseñanza-aprendizaje de los cuatro bloques STEM de una manera integrada en lugar de la enseñanza tradicional como áreas de conocimiento compartimentadas.

- Con cierto enfoque de Ingeniería en cuanto al desarrollo de conocimientos teóricos para la satisfacción práctica ante distintos problemas o retos tecnológicos.

En ocasiones a este acrónimo se le añade de forma explícita la parte de creatividad del arte (STEAM). De esta forma se le otorga aún más importancia a los rasgos creativos que se desarrollan en la búsqueda de diseños a las soluciones de los problemas propuestos.

En general estas iniciativas presentan problemas o situaciones cotidianas, lo que nos permite una mayor contextualización de los contenidos y visión práctica de los mismos. Así, estas iniciativas están íntimamente relacionadas con la metodología de aprendizaje basada en proyectos (Domènech-Casal et al., 2019) y metodologías de aprender haciendo (Anzai y Simon, 1979).

5. Objetivo

El principal objetivo perseguido por este trabajo es plantear y proponer el uso de una plataforma basada en robótica para el aprendizaje de un lenguaje de programación. A través de distintas propuestas didácticas basadas en problemas y proyectos. Este objetivo se engloba dentro de la asignatura de Tecnologías de la Información y Comunicación de 1º Bachillerato en la Comunidad Autónoma de Cantabria.

6. Estructura del documento.

En este documento se analizan los diferentes elementos que componen el marco de trabajo y a continuación se realiza nuestra propuesta didáctica. En el capítulo 2 se presenta el estado actual de la cuestión teórica a tratar. Es decir, se presenta una contextualización, se elaboran los marcos legales y se presentan las posibilidades que se nos abren para el tratamiento del problema. El capítulo 3 se centrará en la elección de las tecnologías a usar durante el proyecto y la justificación de éstas. A lo largo del capítulo 4 se presenta nuestra propuesta didáctica y englobará la parte central del trabajo. Finalmente se presentarán las conclusiones relevantes extraídas en el capítulo 5 y se presentará un anexo con actividades elaboradas para poder impartir plenamente dicha propuesta.

IV. Estado de la cuestión

Como se ha comentado anteriormente este capítulo será el encargado de presentarnos un contexto de la situación de la enseñanza de la programación a distintos niveles. Además, nos centraremos en presentar los marcos legales en los que basaremos nuestra actuación didáctica y se presentarán las principales plataformas y lenguajes de programación candidatos.

1. Contexto

Desde principios de la década pasada y coincidiendo con la "salida" de la crisis económica se desarrolló una Agenda Digital para Europa con el fin de impulsar la economía europea aprovechando las ventajas económicas del mercado único digital. En la misma presentación de la Agenda Digital Europea se estimaban que para 2020 (horizonte fijado para la primera Agenda Digital Europea) habría 16 millones de empleos nuevos relacionados con las competencias en tecnologías de la información y comunicación.

Con el objetivo de monitorizar estos resultados cada año se publica el marcador de la Agenda Digital DESI (*digital economy and society index)*. A continuación, se presenta el índice DESI para 2020 para todos los países de la UE en la Figura 1.

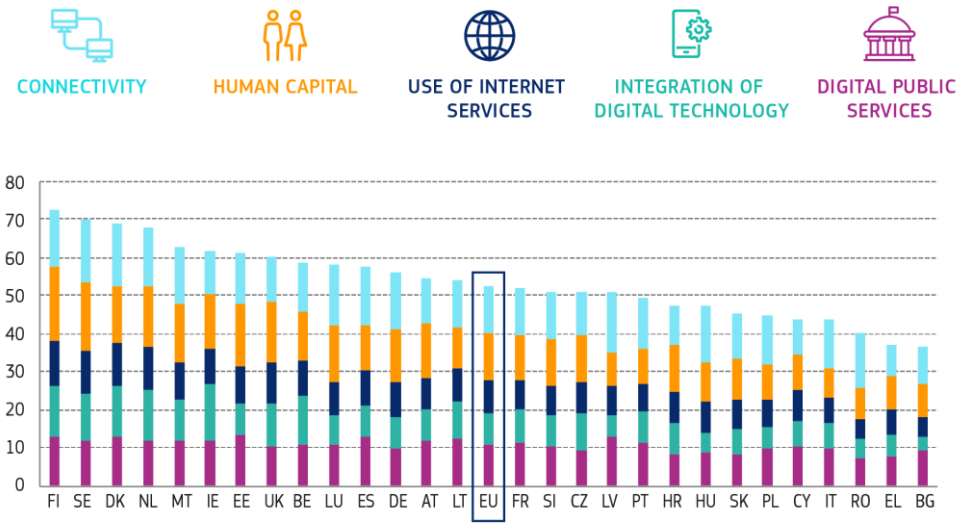


Figura 1: DESI 2020

De entre todas las iniciativas propuestas se explicita la elaboración de un programa de investigación TIC en el que están muy presentes los dos temas que nos atañen en este trabajo. La robótica y la programación.

De acuerdo con esto en 2016 se presenta un informe europeo (Bocconi et al., 2016) en el que se trata la cuestión referente al pensamiento computacional y como construir procesos de enseñanza-aprendizaje adecuados para lograr este fin. Además, se analiza la cuestión en Europa encontrándose unos resultados interesantes. En la Figura 2: Justificaciones de inclusión del pensamiento computacional para cada país. (Bocconi et al., 2016) se presentan las justificaciones de la inclusión del pensamiento computacional en los distintos currículos de cada país.

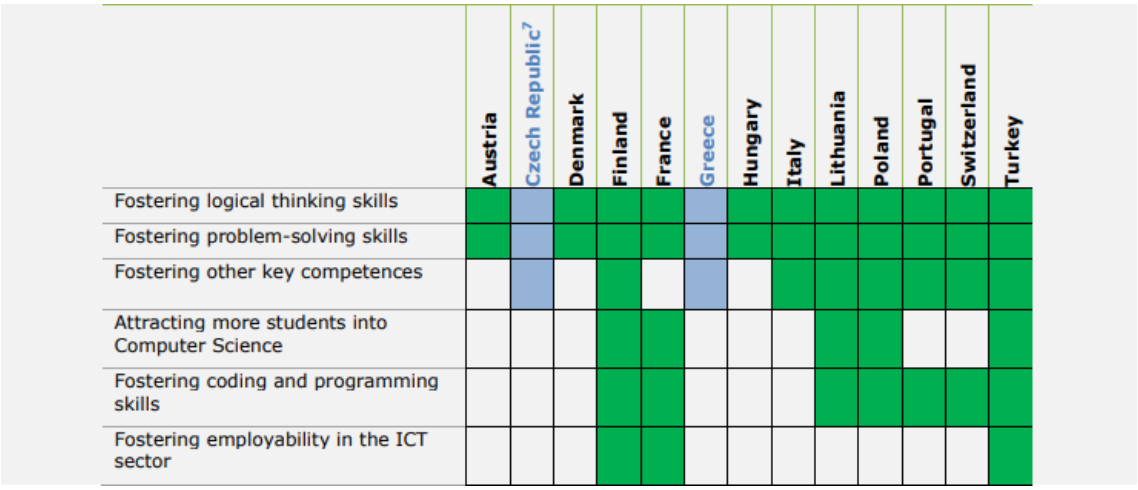


Figura 2: Justificaciones de inclusión del pensamiento computacional para cada país. (Bocconi et al., 2016)

Como se puede comprobar en esta tabla no podemos ver la situación española y esto está relacionado con que las tareas referidas a la programación y pensamiento computacional han quedado relegadas al nivel regional como se puede observar en la Figura 3.

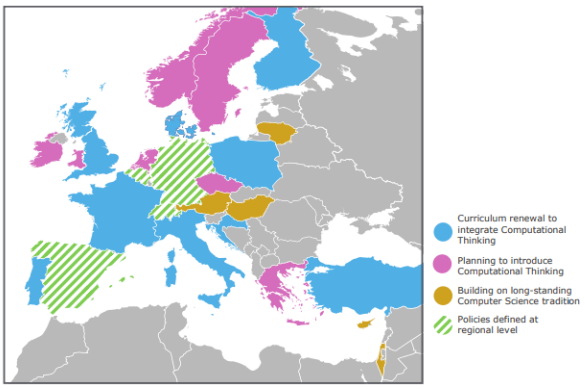


Figura 3: Integración del pensamiento computacional en Europa (Bocconi et al., 2016)

Encontramos 4 grandes grupos de los que podemos extraer alguna conclusión interesante como inspiración para la adopción de estos planes para nuestro trabajo.

En un primer y destacado lugar encontramos una serie de países con una larga tradición en ciencias de la computación. Tomaremos los más interesantes y se comentarán brevemente.

Israel tiene una larga tradición en la enseñanza de la informática. Aunque el pensamiento computacional se ofrece como asignatura optativa en la mayoría de las escuelas secundarias, los estudiantes de la educación pública convencional son instruido en la alfabetización digital e informática como un medio importante y una metodología que contribuye al aprendizaje en todas las materias (Webb et al., 2017). El Ministerio de Educación de Israel tiene una estrategia bien organizada y detallada para el estudio de la informática en las escuelas secundarias. Esto considera la ciencia de computación como una asignatura independiente y la promoción de la alfabetización digital e informática como una prioridad en todas las áreas temáticas (Webb et al., 2017). El plan de estudios consta de módulos obligatorios y optativos. La Introducción a la ciencia de computación, por ejemplo, hace hincapié en los fundamentos del pensamiento algorítmico. La intención del curso no es entrenar a los estudiantes para que se conviertan en programadores, sino más bien introducir a los estudiantes a la lógica y pensamiento algorítmico y exponerlos a diferentes entornos de desarrollo en una etapa temprana (Bargury et al., 2012).

Inglaterra (Reino Unido) ha sido uno de los primeros países europeos en crear una asignatura de ciencias de computación en escuelas primarias y secundarias (a partir de septiembre de 2014) de forma obligatoria. Esta decisión tan temprana impulsó numerosas iniciativas de todos los niveles. En un nivel político allanó el camino para el resto de los países a la hora de realizar una integración similar. Por otro lado se crearon plataformas como micro:bit con el objetivo de motivar la tarea de la creación en los alumnos.

Finlandia es uno de los primeros países de la UE en introducir el "pensamiento algorítmico" y la programación como una actividad obligatoria e interdisciplinaria desde el primer año de la escuela primaria. Este reciente plan de estudios básico nacional para primaria y secundaria inferior se publicó en 2014 y se espera ser ampliado en las futuras reformas.

Cabe destacar la experiencia de otros países alrededor del mundo y su forma de abordar esta cuestión. Así nos encontramos el programa de educación de Corea del Sur basado en desarrollo de software. Este programa se centra en el desarrollo del pensamiento computacional, las habilidades de programación y la expresión creativa a través del software. El programa se despliega en todos los niveles de educación, desde primaria hasta universitaria. Los niveles más bajos son los que más fuertemente han cambiado ya que este programa es obligatorio desde 2018 y la formación de los maestros resultó muy crítica ya que no existía ese perfil tecnólogo en los maestros del país. En este sentido se elaboraron numerosos programas de capacitación y formación para estos profesores en lo que supone una tarea bastante extrapolable a cualquier país del mundo.

En resumen, podemos ver el consenso más o menos logrado por todos los países del mundo en su consideración hacia el pensamiento computacional y su importancia en el currículo no solamente por sus repercusiones laborales sino por un gran abanico de competencias que son desarrolladas en ellas.

2. Situación en España

Como se podía ver en la figura 3 España se encuentra en el grupo de países donde las autonomías son las encargadas de la definición de los currículos y,

con ello, la adopción de asignaturas sobre pensamiento computacional. Este trabajo se enmarca en las propuestas de integración de la programación en Bachillerato por ese motivo nuestro encaje legal es el que nos define dentro del periodo y del inmediatamente anterior que define los conocimientos previos. Esto queda englobado en las dos leyes siguientes:

- Ley Orgánica 2/2006, de 3 de mayo, de Educación (LOE).
- Ley Orgánica 8/2013, de 9 de diciembre, para la mejora de la calidad educativa (LOMCE).

A su vez se debe acudir a los decretos y reales decretos que rigen los currículos, en este caso los siguientes:

- Real Decreto 1105/2014, de 26 de diciembre, por el que se establece el currículo básico de la Educación Secundaria Obligatoria y del Bachillerato. (BOE del 3 de enero de 2015).
- Decreto 38/2015, de 22 de mayo, que establece el currículo de la Educación Secundaria Obligatoria y del Bachillerato en la Comunidad Autónoma de Cantabria. (BOC del 5 de junio de 2015).

En estos documentos se determina el encaje de la asignatura de Tecnología en el periodo de la ESO y Bachillerato. Siendo ésta una asignatura específica en el primer y segundo ciclo de la ESO, siendo las comunidades las que toman la decisión de los cursos en los que serían cursada. Además, aparece una asignatura específica denominada Tecnologías de la Información y la Comunicación en 4 de ESO. En el periodo de Bachillerato la asignatura de tecnología se renombra a Tecnología Industrial (I y II), presente en ambos cursos en las modalidades correspondientes y se vuelve a tener la posibilidad de cursar Tecnologías de la Información y Comunicación (I y II) en ambos cursos.

Antes de entrar de lleno en nuestra región y sus detalles legales vamos a comentar un informe que trata esta cuestión sobre las diferencias regionales sobre esta temática. El Instituto Nacional de Tecnologías Educativas y de Formación del profesorado (INTEF) publicó en 2018 un informe (Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado, 2018)

donde analiza la situación y el contexto de distintos conceptos en España. De este estudio destaca el caso de Madrid en el que se detalla que existe una asignatura de Tecnología y Recursos Digitales de libre configuración para toda la etapa de Primaria. Desde esta asignatura se introducen las primeras nociones de programación visual o por bloques. Además, en los periodos de secundaria se ha renombrado la asignatura de Tecnología por Tecnología, Programación y Robótica, obligatoria en los tres cursos del primer ciclo de secundaria. Esta asignatura se basa en 5 ejes en los que la programación y la robótica tienen uno propio. Estas medidas vinieron de la mano de una inversión en material y formaciones al profesorado.

3. Situación en Cantabria

Debido a la baja calidad de las aportaciones del informe respecto a la situación en Cantabria (lo limitan a la presentación de CantabRobots) lo dejamos como tarea propia. A continuación, se van a presentar las principales concreciones en el currículo de Cantabria respecto a la temática.

Debido al enfoque de nuestro trabajo orientado a las actuaciones en el primer curso de Bachillerato en la asignatura de Tecnología de la Información y la Comunicación I presentamos los currículos de contenido (con sus respectivos estándares de aprendizaje) referentes a esa asignatura, a las anteriores relacionadas y la inmediatamente posterior.

Asignatura	Bloque	Contenido	Estándares de aprendizaje
Tecnología 2º ESO	Bloque 5: **Tecnologías de la Información y la Comunicación**	• Lenguajes de programación con interfaz gráfica.	Crea pequeños programas informáticos para realizar cálculos matemáticos utilizando lenguajes de programación de entorno gráfico.
			Diseña y elabora la programación de un juego sencillo, animación o historia interactiva mediante un entorno de programación gráfico.
Tecnología 3º ESO	Bloque 4: **Estructuras y mecanismos: máquinas y sistemas**	• Programación mediante diagramas de flujo. • Programación por ordenador de un sistema electromecánico mediante plataforma de software y hardware abierto	Utiliza correctamente los elementos eléctricos y electrónicos como sensores y actuadores en circuitos de control programado describiendo su funcionamiento.
			Diseña y monta circuitos de control automático que realicen las tareas propuestas para un prototipo de forma autónoma.
			Elabora un programa informático que controle el funcionamiento de un sistema técnico.
Tecnología 4º ESO	Bloque 4: **Control y Robótica**	• Sistemas automáticos, componentes característicos de dispositivos de control. • El ordenador como elemento de programación y control. • Lenguajes básicos de programación.	Desarrolla un programa para controlar un sistema automático o un robot que funcione de forma autónoma en función de la realimentación que recibe del entorno.

Tecnología de la Información y la Comunicación I 1º Bach	Bloque 5: **Programación**	• Elementos de programación. • Conceptos básicos. • Lenguajes de Programación. Tipos • Historia de la Evolución de la Programación • Técnicas de análisis para resolver problemas: Elaboración de diagramas de flujo y pseudocódigos. • Elementos de un programa: • Valores y Tipos. • Expresiones Aritméticas. Operaciones de Escritura Simple. • Estructura de un Programa. • Constantes y variables. • Metodología de desarrollo de programas. • Resolución de problemas mediante programación. • Descomposición de problemas mayores en otros más pequeños. • Estructuras básicas de la programación.	Desarrolla algoritmos que permitan resolver problemas aritméticos sencillos elaborando sus diagramas de flujo correspondientes. Escribe programas que incluyan bucles de programación para solucionar problemas que implique la división del conjunto en parte más pequeñas. Obtiene el resultado de seguir un pequeño programa escrito en un código determinado, partiendo de determinadas condiciones. Define qué se entiende por sintaxis de un lenguaje de programación proponiendo ejemplos concretos de un lenguaje determinado. Realiza programas de aplicación sencillos en un lenguaje determinado que solucionen problemas de la vida real.

		• Programación estructurada. • Expresiones Condicionales. • Selección y bucles de programación • Seguimiento y verificación de programas. • Estructuras de datos estáticas	
Tecnología Industrial I 1º Bach.	Bloque 3: **Máquinas y sistemas**	• Análisis de máquinas. Sistemas de generación, transformación y transmisión del movimiento. Sistemas auxiliares. • Programación de máquinas. Automatización de procesos empleando dispositivos programables. • Circuitos eléctricos. Componentes. Asociación serie, paralelo y mixta de componentes. Ley de Ohm. Potencia. Energía. Resolución de circuitos eléctricos con una o varias fuentes de alimentación. Diseño, simulación, montaje y verificación de circuitos.	• Describe la función de los bloques que constituyen una máquina dada, explicando de forma clara y con el vocabulario adecuado su contribución al conjunto. • Describe mediante diagramas de bloques el funcionamiento de máquinas herramientas, explicando la contribución de cada bloque al conjunto de la máquina. • Diseña y realiza el montaje de una máquina automatizada con lógica cableada o programada. • Verifica la evolución de las señales en circuitos eléctrico-electrónicos, neumáticos o hidráulicos dibujando sus formas y valores en los puntos característicos. • Interpreta y valora los resultados obtenidos de circuitos eléctrico-electrónicos, neumáticos o hidráulicos.

		• Circuitos electrónicos. Componentes. Circuitos de aplicación práctica. Cálculo de magnitudes en los circuitos. Diseño, simulación, montaje y verificación de circuitos. • Neumática. Componentes de tratamiento del fluido, control y actuación. Circuitos básicos. Análisis de circuitos de aplicación práctica. Diseño, simulación, montaje y verificación de circuitos.	
Tecnología de la Información y la Comunicación II 2º Bach.	Bloque 1: **Programación**	<u>Programación orientada a objetos</u> • Clases y objetos: definición y conceptos básicos de la Programación Orientada a Objetos. • Elementos de programación: Variables, operadores, métodos, estructuras de control de flujo. • Escritura/lectura de datos en archivos y consola.	Comprende y maneja las técnicas de Programación Orientada a Objetos implementación de clases y objetos. Explica las estructuras de almacenamiento para diferentes aplicaciones teniendo en cuenta sus características. • Elabora diagramas de flujo de mediana complejidad usando elementos gráficos e interrelacionándolos entre sí para dar respuesta a problemas concretos • Utiliza pseudocódigo para transformar los diagramas de flujo

		• Estructuras de almacenamiento estáticas y dinámicas: definición, creación y operaciones. • Algoritmia. Definición de algoritmo. Complejidad de algoritmos y notación O(n). Recursividad, ordenación y búsqueda. • Programación avanzada: control de excepciones. Programación multihilo. <u>Ingeniería de Software</u> • Metodología y ciclo de vida de una aplicación • Análisis y diseño de software. Diagramas de flujo y pseudocódigo. Unified Modeling Language. • Características y criterios de elección de un IDE. Uso básico. • Depuración, optimización y pruebas de software. <u>Desarrollo de software para resolución de tareas en diferentes ámbitos</u>	• Desarrolla código empleando los elementos léxicos, sintácticos y semánticos apropiados.
			• Elabora programas de mediana complejidad definiendo el flujograma correspondiente y escribiendo el código correspondiente. • Descompone problemas de cierta complejidad en problemas más pequeños susceptibles de ser programados como partes separadas.
			• Diseña proyectos de acuerdo con las diferentes metodologías disponibles • Describe las fases de ejecución de un proyecto empleando protocolos de gestión.
			• Identifica los diferentes tipos de diagramas integrados en UML para comprender la documentación asociada a un producto software. • Utiliza la metodología UML para documentar el programa.
			• Elabora programas de mediana complejidad utilizando entornos de programación. • Lleva a cabo las operaciones básicas de gestión de un proyecto empleando el entorno de desarrollo integrado.
			• Obtiene el resultado de seguir un programa escrito en un código determinado, partiendo de determinadas condiciones.

		• Programación de interfaces gráficas para aplicaciones de usuario. • Implementación de aplicaciones en red para acceso a bases de datos remotas. • Programación aplicada a robótica y control de procesos a través de sistemas embebidos hardware-software. • Programación de dispositivos móviles: características e implementación de los elementos básicos de una aplicación. Comunicación con otras plataformas. • Programación en entornos de cálculo numérico y simulación.	• Optimiza el código de un programa dado aplicando procedimientos de depuración. • Emplea herramientas específicas para realizar pruebas de software, interpreta y contrasta los resultados.
			• Programa interfaces gráficos con los que interactuar con el programa que se implementa por debajo. • Implementa aplicaciones sencillas para tareas de comunicación de datos a través de la red. • Usa las técnicas de programación estudiadas aplicándolas sobre dispositivos de hardware/software embebido integrados en sistemas robóticos y/o de control de procesos. • Crea aplicaciones sencillas para dispositivos móviles que luego instalará para su propio uso. • Emplea la programación para realizar tareas de simulación numérica sobre aplicaciones de tipo científico-matemático.

Tecnología Industrial II 2º Bach.	Bloque 3: **Sistemas automáticos**	• Automatismos eléctricos y neumáticos. ○ Elementos y funcionamiento: ▪ Transductores, captadores y actuadores. • Estructuras de sistema automáticos: ○ De lazo abierto y cerrado. ○ Función de transferencia ○ Operación y simplificación de bloques. ○ Estabilidad. • Sistemas neumáticos. ○ Producción, conducción y depuración de fluidos. ○ Elementos de accionamiento, regulación y control. ○ Circuitos característicos de aplicación. ○ Diseño y montaje de circuitos neumáticos.	• Define las características y función de los elementos de un sistema automático interpretando planos/esquemas de estos. • Diferencia entre sistemas de control de lazo abierto y cerrado proponiendo ejemplos razonados de los mismos. • Explica la función de los elementos basados en diferentes tecnologías que pueden formar parte de un sistema automático de control realizando esquemas de estos.
			• Diseña mediante bloques genéricos sistemas de control para aplicaciones concretas describiendo la función de cada bloque en el conjunto y justificando la tecnología empleada. • Realiza operaciones de simplificación de la función de transferencia de un sistema automático para posteriormente realizar un análisis de su estabilidad.
			• Monta físicamente o de forma simulada circuitos simples eléctricos o neumáticos interpretando esquemas y realizando gráficos de las señales en los puntos significativos.
			• Diseña y comprueba utilizando software o equipos de simulación circuitos eléctricos o neumáticos

			que respondan a unas especificaciones dadas.
			• Visualiza señales en circuitos automáticos mediante equipos reales o simulados verificando la forma de estas.
	Bloque 4: **Circuitos y sistemas lógicos**	• Circuitos digitales. ○ Señales digitales y lenguaje binario. ○ Circuitos lógicos combinacionales ▪ Puertas lógicas y algebra de Boole. ▪ Métodos de simplificación de funciones lógicas. ▪ Circuitos característicos. • Circuitos lógicos secuenciales. ○ Biestables. ○ Circuitos característicos.	• Realiza tablas de verdad de sistemas combinacionales identificando las condiciones de entrada y su relación con las salidas solicitadas.
			• Diseña circuitos lógicos combinacionales con puertas lógicas a partir de especificaciones concretas, aplicando técnicas de simplificación de funciones y proponiendo el posible esquema del circuito. • Diseña circuitos lógicos combinacionales con bloques integrados partiendo de especificaciones concretas y proponiendo el posible esquema del circuito.
			• Explica el funcionamiento de los biestables indicando los diferentes tipos y sus tablas de verdad asociadas. • Dibuja el cronograma de un contador explicando los cambios que se producen en las señales. • Analiza el funcionamiento de circuitos secuenciales típicos realizando gráficas de las

			señales que proporcionan a partir de simuladores.
	Bloque 5: **Control y programación de sistemas automáticos**	• Circuitos de control programado. ○ Programación rígida y flexible. ○ Circuitos lógicos secuenciales ▪ Cronogramas ▪ Técnicas de análisis y diseño • Microprocesadores, microcontroladores y autómatas programables ○ Estructura y funcionamiento. ○ Aplicación al control programado	• Obtiene señales de circuitos secuenciales típicos utilizando software de simulación. • Dibuja cronogramas de circuitos secuenciales partiendo de los esquemas de estos y de las características de los elementos que lo componen.
			• Diseña circuitos lógicos secuenciales sencillos con biestables a partir de especificaciones concretas y elaborando el esquema del circuito. • Utiliza programas de simulación para comprobar el funcionamiento de circuitos secuenciales que resuelvan problemas de automatización.
			• Identifica los principales elementos que componen un microprocesador tipo y lo compara con algún microprocesador comercial. • Utiliza el ordenador como elemento de control programado para su aplicación en sistemas automáticos sencillos. • Realiza búsquedas de información relacionadas con las características de los autómatas programables y su utilización industrial

Centrándonos en la parte que nos atañe podemos comprobar cuales son las contribuciones a las competencias adquiridas en el bloque de Programación que serán resumidas a continuación:

- Competencia lingüística: Trabajo sobre vocabulario específico del campo de conocimiento. Especialmente en las fases de análisis y diseño de código.
- Competencia digital: Núcleo de la asignatura. Enfoque de creador de TIC a nivel de software con el aprendizaje de la programación.
- Competencia matemática y competencias básicas en Ciencia y Tecnología: Desarrollo de pensamiento lógico y abstracto. Desarrollo de algoritmos para resolver problemas.
- Competencia para Aprender a aprender: Se permiten herramientas de autoaprendizaje y autoevaluación gracias al contenido aportado en clase y las facilidades para obtener nuevo contenido en Internet
- Sentido de iniciativa y espíritu emprendedor: Desarrollo de soluciones ante problemas de forma libre y creativa ayuda a la adquisición de esta competencia.

V. Herramientas tecnológicas.

En este bloque trataremos de justificar nuestra elección sobre unas tecnologías para el desarrollo de la propuesta didáctica. Estas elecciones serán tomadas de forma iterativa. En un primer momento escogeremos nuestro soporte físico y una vez elegido determinaremos el lenguaje de programación más adecuado para los fines propuestos.

1. Placas programables.

En este subapartado vamos a presentar las principales opciones que se nos presentan en el mercado para la impartición de la temática de robótica y programación. Debemos tener en cuenta el destinatario de la placa, así como el enfoque propio que queremos otorgarle a nuestra propuesta. Por un lado, conocemos bien el alumno de 1º de Bachillerato y por otro lado tenemos claro que el enfoque que queremos conseguir es el del aprendizaje de programación

con la ayuda de un soporte físico. Es decir, debemos priorizar la parte destinada a la programación en nuestra elección.

Arduino

Comenzamos por la placa más extendida de todas: Arduino (Arduino, s. f.). Es interesante conocer los orígenes de esta placa que se remonta al proyecto Wiring (Hernando Barragán., 2003) en el cual se propone una placa programable como herramienta para estudiantes en el Interaction Design Institute en Ivrea, Italia. El principal objetivo de este proyecto era proporcionar una forma fácil y económica de que estudiantes y principiantes pudieran crear dispositivos que pudieran interactuar con su entorno mediante sensores y actuadores. La primera placa Arduino comercial fue introducida en el año 2005. Y en el año 2012 se introdujeron procesadores ARM de 32 bits (similar al procesador de un smartphone) que sustituyeron a los originales AVR de 8 bits. A pesar del cambio en el procesador y, por consiguiente, en el set de instrucciones a bajo nivel el entorno de desarrollo nos permite trabajar con un código a alto nivel interoperable entre las distintas placas.

Su gran ventaja frente a otras opciones es la gran comunidad que existe detrás y, esto es así, en gran medida, debido al carácter *open source* que tiene esta placa.

Su principal desventaja (aunque podría considerarse superficial) es la solución nativa para la programación de la placa. Esta está basada en la plataforma Wiring que, a su vez, usa como lenguaje de programación principal C++. Este podríamos decir que es un lenguaje que requiere de cierta experiencia y no suele ser recomendado como la primera opción a aprender. Al inicio he recordado que esta desventaja podría considerarse superficial y es debido a que podríamos ejecutar cualquier código de programación en nuestra placa Arduino siempre y cuando aseguremos el protocolo de transmisión de datos. Una opción que puede ser elegida es la combinación Python + Protocolo Firmata.

Micro:bit

Una más reciente pero no menos importante es la placa Micro:bit (BBC, 2015). El nombre de esta placa proviene de uno de los primeros ordenadores

domésticos diseñado por Acorn Computers (impulsores y desarrolladores de la arquitectura ARM que pretende dominar el mercado más allá del mercado móvil) para la BBC, el BBC Micro. Este fue un equipo de un gran éxito en el mundo académico en las décadas de los 80 y 90 y queriendo reeditar ese viejo éxito surge esta placa programable.

La placa surge a raíz de los impulsos surgidos a principios de década en el Reino Unido con el fin de lograr una mayor alfabetización digital y, este proyecto en concreto surge para paliar la tarea en el ámbito educativo.

La placa es una pequeña tarjeta en la que el foco principal de diseño fue el de lograr que la tarea del aprendizaje a la programación sea fácil y accesible para todos. A diferencia de Arduino esta placa presenta ya una cantidad de sensores integrados y un entorno de programación integrado online. Esta última parte quizá es la más importante para nosotros ya que nos permite comenzar a programar de forma muy veloz. Además, aporta tres entornos en uno: Uno visual muy similar a scratch y dos lenguajes de programación (JavaScript y Python). De esta forma, la placa cubre muy bien la necesidad de tener un gran enfoque hacia la programación, siendo la placa un mero soporte físico con el que ver nuestros resultados. Además, la presencia de los dos tipos de programación nos ayuda a que el progreso sea más suave y se pueda evolucionar de forma satisfactoria.

Lego Mindstorm

Esta plataforma se diferencia de las del resto de las comparativas en que el núcleo de la tecnología no es la placa programable sino el conjunto. De hecho, Lego Mindstorm (Lego, s. f.) es una línea de robótica para niños comercializado por primera vez en 1998. Comenzó siendo comercializado como una herramienta educacional en una colaboración con el MIT. En la actualidad esta marca tiene una amplia gama de productos enfocados a diferentes edades.

Un aspecto positivo es la variedad de opciones en cuanto a entornos de programación disponibles. Encontramos un entorno propio y distintos plugins para el uso de entornos de desarrollo profesionales. Entre estas integraciones

se puede llegar a usar el portal Makecode (Microsoft, s. f.-a) lo que permite tener disponible programación visual.

En este sentido la parte de software disponible es similar a Micro:bit pero presenta unas claras desventajas a nivel de precio.

Raspberry Pi

La Raspberry Pi (Raspberry Pi Foundation, s. f.) es también una placa programable pero con una mayor potencia. En general se asemeja más a un ordenador de bajo coste que sus alternativas. Su origen también está relacionado con el mundo educativo siendo diseñada para estimular el aprendizaje de la informática en la escuela. Esta placa fue tan popular que ya ha excedido su nicho original vendiéndose como core para numerosas aplicaciones de robótica.

Tanto se asemeja a un ordenador que se ejecuta una versión reducida de Debian, denominada Raspbian. Incluso se ha podido ejecutar sistemas operativos más pesados como Windows 10. Debido a su increíble éxito la comunidad es enorme y está en continuo crecimiento.

La principal ventaja derivada de su potencia es que podemos ejecutar casi cualquier lenguaje de programación en ella. A pesar de ello no hay una solución nativa en la que esta tarea se convierta en algo amigable y fácil de iniciar. En general, es una placa con muchísimas posibilidades, pero con una curva de aprendizaje bastante grande.

Elección final

A modo de resumen vamos a comentar cual es la necesidad principal de nuestro proyecto. En este caso, la necesidad que hemos encontrado es poder usar la plataforma MakeCode en la programación de nuestra placa. Hemos decidido esto debido a diversos factores: Curva de aprendizaje suave, alta productividad, soporte a programación visual y textual, documentación y comunidad online con ideas y proyectos a realizar. Esta condición nos descarta a la Raspberry Pi y Arduino (aunque hay alguna implementación en fase temprana del MakeCode). Nos quedan dos opciones pero hay un gran hándicap para los Lego y es su

elevado precio en el que un solo kit puede irse a los 400€ mientras que la placa micro:bit ronda los 20€.

Sumando estas condiciones la placa que mejor cumple nuestras condiciones es micro:bit y es la elegida para el desarrollo del trabajo. Una vez elegida la placa se va a discutir la elección de lenguaje textual para el desarrollo del trabajo.

2. Lenguaje de programación

Un lenguaje de programación es un lenguaje formal (Chomsky, 1956) que proporciona una serie de símbolos y reglas gramaticales y semánticas con las que construir instrucciones o algoritmos para controlar una máquina u ordenador.

Desde una perspectiva histórica nos encontramos que la primera noción de lenguaje de programación la encontramos en el lenguaje máquina. Este lenguaje es el que puede interpretar la máquina ya que es, literalmente, una secuencia de binaria. Rápidamente se vio el problema para trabajar con este lenguaje desde una óptica humana.

Figura 4: Lenguaje máquina

En ocasiones la representación del lenguaje máquina no se genera en codificación binaria sino en hexadecimal, pero es indistinto a la hora de lograr legibilidad.

Para facilitar el trabajo del programador, los primeros pasos fueron crear un traductor para reemplazar estas cadenas binarias por palabras o abstracciones

provenientes del inglés. Así nace el lenguaje ensamblador. Por ejemplo, para simplificar la operación suma *(add*) se representa con la letra A. El lenguaje ensamblador sigue la misma estructura que el lenguaje máquina, pero es algo más fácil de recordar.

A finales de 1953, dentro de IBM se propuso la elaboración de un nuevo lenguaje más práctico que los lenguajes ensamblador de la época. Así nació, en 1956, Fortran el primer lenguaje de alto nivel. Es decir, este lenguaje tenía un nivel de abstracción más elevado y disponía de un "traductor" a código máquina, en este caso el compilador (aunque no es necesaria la pieza del compilador, es un aspecto más técnico que se escapa del alcance de este trabajo).

Desde los años 60 se han creado numerosos nuevos lenguajes de programación de alto nivel enfocados desde distintos paradigmas, con características diversas y de diferentes propósitos. Por ello, se ha generado distintas clasificaciones atendiendo a ciertas características básicas de los lenguajes. Se van a ver a continuación la clasificación básica.

Clasificación por paradigma.

Los distintos paradigmas de programación distinguen diferentes modelos de computación y estilos de estructurar y organizar las tareas que debe realizar un programa. Un mismo lenguaje de programación puede ofrecer soporte a uno o varios paradigmas de programación.

- Programación imperativa: Es el paradigma más usado y se basa en dar instrucciones de cómo realizar las tareas para lograr la solución.
- Programación declarativa: Este paradigma se basa en describir el problema declarando propiedades y reglas que deben cumplirse, en lugar de instrucciones. En este tipo de lenguaje no se específica como obtener la solución como una lista de instrucciones, sino que existen mecanismos internos de control/inferencia que se encargan de obtener la solución.

Dentro de estos dos grandes paradigmas encontramos subparadigmas de cierta relevancia pero que, al fin y al cabo, se encuentran incluidos en estos dos grupos.

De esta primera clasificación podemos concluir que un lenguaje de programación imperativa sería lo más adecuado como primer tipo de lenguaje, ya que los lenguajes declarativos requieren un mayor conocimiento lógico-matemático.

Tras esta breve descripción y teniendo en cuenta la limitación que tenemos a la hora de elegir un lenguaje (micro:bit solo soporta dos) pasamos a describir cada una de nuestras opciones y finalmente elegiremos una.

Python

Python (Python Foundation, s. f.-b) es un lenguaje de alto nivel, interpretado, multiparadigma (imperativa y funcional). Además, es un lenguaje interpretado, dinámico, multiplataforma y *open source*. Ya hemos comentado lo que es un lenguaje de alto nivel y el paradigma, pero el resto de las características no han sido comentadas. Por un lado, se especifica que es un lenguaje interpretado y esto no es más de cómo se realiza la "traducción" a lenguaje máquina. La tendencia en el futuro es que un mismo lenguaje pueda ofrecer las dos opciones (interprete o compilador) ya que no es algo limitante dentro del lenguaje. También se ha detallado que es un lenguaje dinámico, esto significa que una misma variable puede tomar valores de distintos tipos.

A parte de todas estas consideraciones de corte más técnico una de las características más importantes de Python es su orientación hacia la legibilidad del código. Es decir, tiene una sintaxis sencilla e intuitiva. Normalmente estas ideas sobre el código residen en el Zen de Python (Python Foundation, s. f.-a).

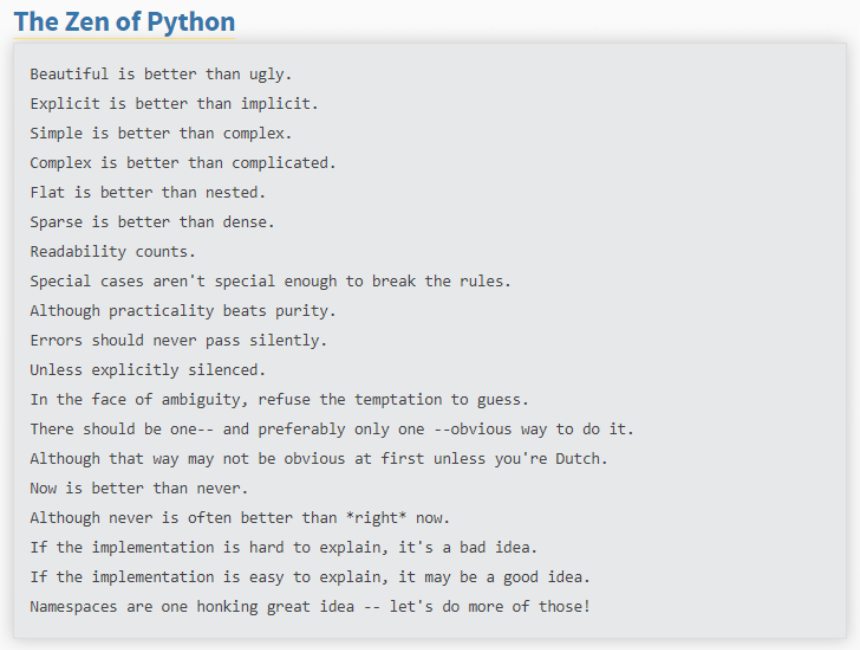


Figura 5 Zen de Python (Python Foundation, s. f.-a)

Recientemente se ha puesto el foco encima de Python como una muy buena opción como primer lenguaje de programación para los alumnos. Podemos ver estudios que van en esta línea en (Peña, 2015). En este artículo se discute la idoneidad del lenguaje para alumnos de Ingeniería Informática concluyéndose que es bastante buena opción para salvar las principales barreras que se han encontrado los alumnos a la hora del aprendizaje de la programación.

JavaScript

JavaScript (Mozilla Foundation, s. f.) es un lenguaje de programación liviano, interpretado (estrictamente es compilado justo a tiempo, pero es un detalle técnico que no repercute en la comparativa con Python). JavaScript es un lenguaje conocido por ser un lenguaje de scripting para desarrollo web, pero también es un lenguaje de propósito general, multiparadigma (imperativa y funcional), dinámico, etc. A un primer vistazo cumple con los mismos requisitos que Python en cuanto a sus características básicas, pero que tiene ciertas ventajas en ciertos aspectos que nos atañen en nuestro ejemplo.

JavaScript se ejecuta en un navegador web (además es un estándar de los navegadores, lo que asegura la interoperabilidad) lo que hace que su rendimiento sea muy bueno. Por este motivo el proyecto MakeCode comenzó

con un editor único de JavaScript (En realidad se trata de la extensión Typescript, pero esto es algo inapreciable para el usuario).

Además, es un lenguaje con una sintaxis relativamente sencilla y bastante indulgente con los errores del programador (podríamos decir que en ocasiones resulta difícil que el código genere un error). Si acompañamos estos motivos con su alta demanda laboral (Figura 6), basada en que las tecnologías web están dominando el mercado, escogemos este lenguaje como nuestro lenguaje base del curso.

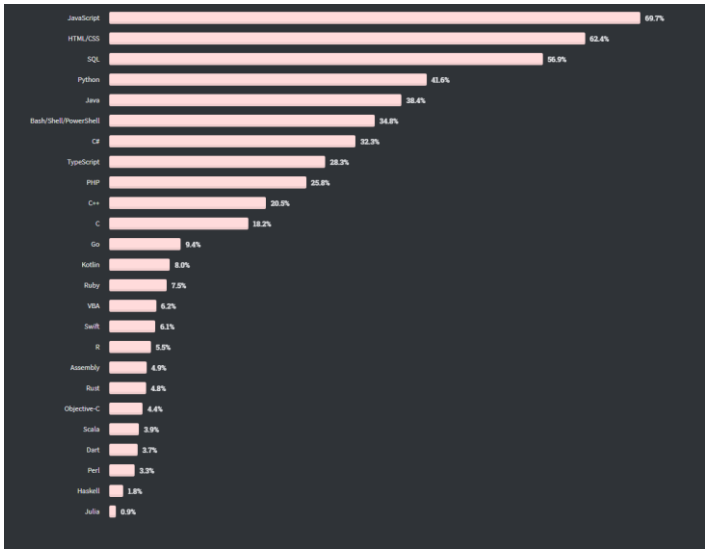


Figura 6: Lenguajes más populares según la encuesta anual de Stack Overflow (Stack Overflow, s. f.)

Por todo ello queda elegido las plataformas y tecnologías a usar en el desarrollo de la propuesta didáctica.

VI. Propuesta didáctica: Introducción a la programación textual en 1º de Bachillerato con micro:bit y JavaScript.

Tras la elección de plataforma y lenguaje textual se dedicará este apartado para expresar las distintas propuestas didácticas para la introducción a la programación textual a través de la plataforma de micro:bit. Antes de comenzar con las propuestas se va a repasar en un nivel mayor de profundidad qué es micro:bit y cómo podemos trabajar con ello como docentes.

1. Micro:bit

Tras la breve revisión realizada en el apartado anterior aquí nos centraremos más en conocer la placa y sus especificaciones técnicas y como aprovecharla.

En la Figura 7se puede ver una representación de la placa y sus especificaciones técnicas básicas.

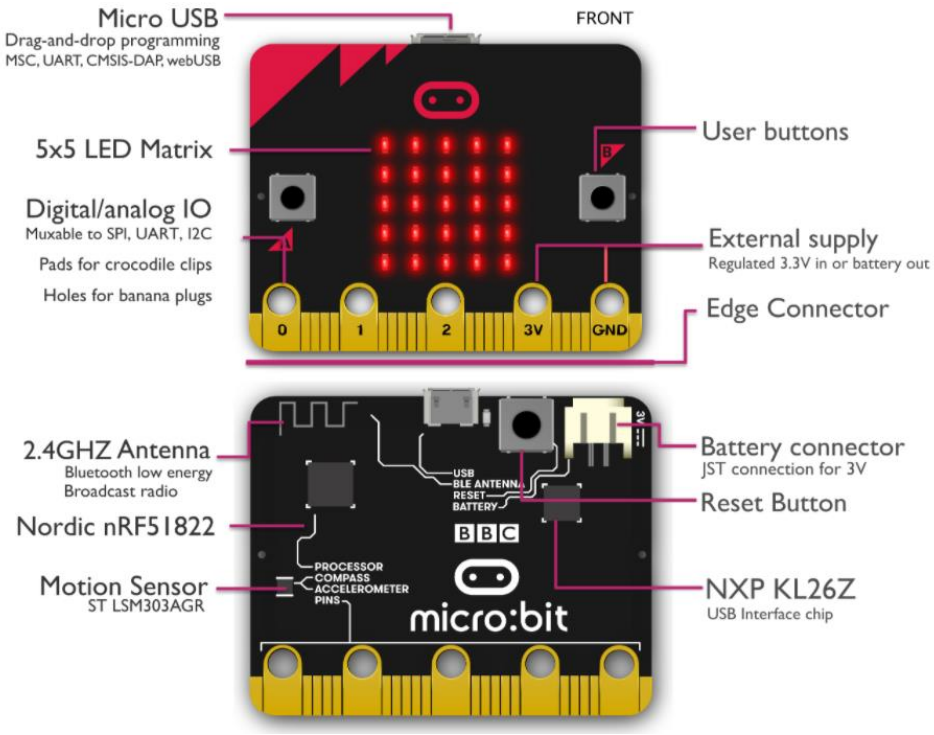


Figura 7: Representación placa micro:bit y sus componentes (Micro:bit Foundation, s. f.)

A continuación, se resumen las características básicas de la placa.

Procesador.

Un procesador ARM de 32 bits de bajo consumo (basado en un Cortex-M0) a una frecuencia de 16MHz. Además, dispone de 256 KB de memoria ROM y 16KB de memoria RAM.

Bluetooth Low Energy (BLE).

Usa bluetooth 4.1 bajo el modo de bajo consumo (BLE) lo que permite que la placa se pueda comunicar con otros dispositivos.

Se puede consultar la documentación de MakeCode sobre esta característica (MakeCode, s. f.-a).

Radio.

Dispone de radio operando bajo protocolo propietario Nordic Gazell a 2.4GHz. Este protocolo permite la difusión de pequeños paquetes de información entre equipos que soportan esta tecnología. En nuestro caso servirá para conectar dos placas micro:bit de forma sencilla.

Se puede consultar la documentación de MakeCode sobre esta característica (MakeCode, s. f.-h).

Botones.

La placa dispone de tres botones. Uno de ellos destinado al reinicio de la placa y tiene poco interés. En cambio, los dos botones frontales son programables.

Se puede consultar la documentación de MakeCode sobre esta característica (MakeCode, s. f.-b)

Matriz LED.

La pantalla lo conforma una mátriz 5x5 de LEDs. Internamente se ejecuta como una matriz 3x9. El software de micro:bit actualiza esta matriz a alta velocidad, de tal manera que está dentro del rango de persistencia de visión del usuario, y no se detecta ningún parpadeo. Esta matriz de LED también se usa para detectar la luz ambiental, cambiando repetidamente algunas de las clavijas de la unidad de LED en las entradas y muestreando el tiempo de decaimiento del voltaje, que es aproximadamente proporcional al nivel de luz ambiental.

Se puede consultar la documentación de MakeCode sobre estas características (MakeCode, s. f.-d, s. f.-e)

Sensor de movimiento.

La placa tiene un chip acelerómetro y magnetómetro combinado que proporciona una detección tridimensional y una detección de la intensidad del campo magnético. También incluye algo de detección de gestos a bordo (como la detección de caídas) en el hardware, y detección de gestos adicionales (por ejemplo, logo-arriba, logo-abajo, sacudida) a través de algoritmos de software. Un algoritmo de software en tiempo de ejecución estándar utiliza el acelerómetro de a bordo para convertir las lecturas en una lectura de brújula independiente de

la orientación de la placa. La brújula debe calibrarse antes de su uso, y el proceso de calibración se inicia automáticamente por el software.

Se puede consultar la documentación de MakeCode sobre estas características (MakeCode, s. f.-f, s. f.-c).

Sensor de temperatura.

El sensor de temperatura está integrado en el procesador y proporciona un rango que opera desde los -25ºC hasta los 75ºC. La resolución es de 0.25ºC y la precisión es de 4ºC.

Se puede consultar la documentación de MakeCode sobre estas características (MakeCode, s. f.-i).

Pines.

La placa dispone de 20 pines. Algunos de ellos son de uso compartido con características de la propia placa, pero también pueden ser usados para otros propósitos.

Se puede consultar la documentación de MakeCode sobre estas características (MakeCode, s. f.-g).

Fuente de energía.

La alimentación de la placa puede ser obtenida a través de la conexión USB o a través de una batería conectada al conector de corriente. También es posible alimentar la placa desde el *pad* de 3V de la parte inferior.

USB.

La conexión e interfaz USB permite conectar la placa al ordenador y transferir el código. Además, puede alimentar a la placa.

2. Guía de inicio rápido.

Desde la documentación oficial nos detallan que comenzar a trabajar con micro:bit es muy sencillo y rápido y se puede resumir en 3 pasos (Micro:bit, s. f.). Es recomendable seguir los videos presentes en la presentación de micro:bit pero lo resumimos a continuación.

- Programar: Crea un programa a través de los editores disponibles. En nuestro caso usaremos MakeCode en sus modos visual y textual (a través de JavaScript).
- Conectar: Conecta la placa al ordenador a través de un cable micro-USB. La placa es fácilmente reconocida por cualquier SO.
- Transferir: Una vez el programa ha sido creado y validado con el simulador podemos descargar un archivo .hex (archivo que podrá leer la placa) y transferirlo por usb a la placa. (Es posible transferirlo por usb pero es más complejo). Mientras se está realizando la transferencia un led amarillo en la parte posterior parpadeará indicando que la transferencia está en curso.
- Ejecutar: Una vez la transferencia haya finalizado el programa ha sido cargado y puede observarse el programa ejecutándose.

3. Entorno de desarrollo MakeCode.

Como se ha comentado anteriormente nuestro entorno de desarrollo será el MakeCode de Microsoft. Este entorno tiene una vista muy intuitiva que puede ser observada en la Figura 8. Teniendo en cuenta los objetivos de este trabajo el principal punto a favor de este entorno es la posibilidad de programación por bloques y bajo un editor de JavaScript (así como una traducción automática entre ambas representaciones).

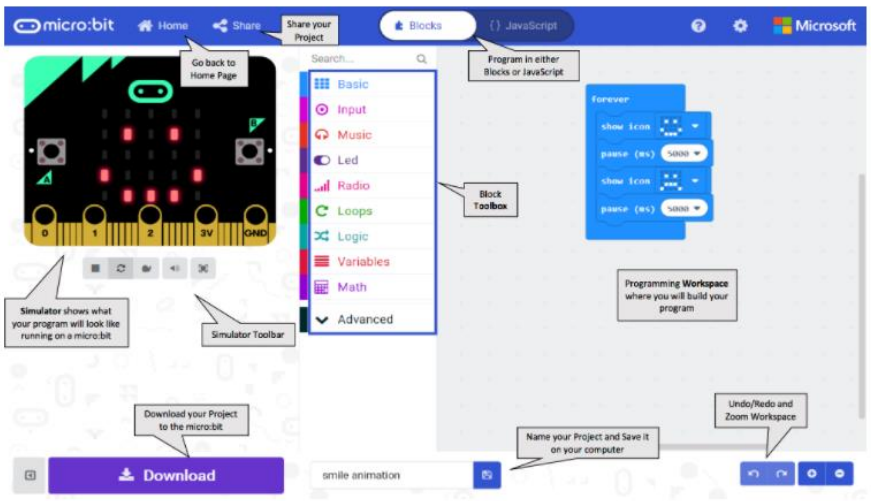


Figura 8: Componentes del editor de código MakeCode (Kidscode, s. f.)

En la Figura 9 se puede ver esta característica de la “traducción” entre representaciones del programa. Esta funcionalidad nos ayudará mucho en las primeras etapas en las que deseamos evolucionar desde la enseñanza de la programación visual a la textual.

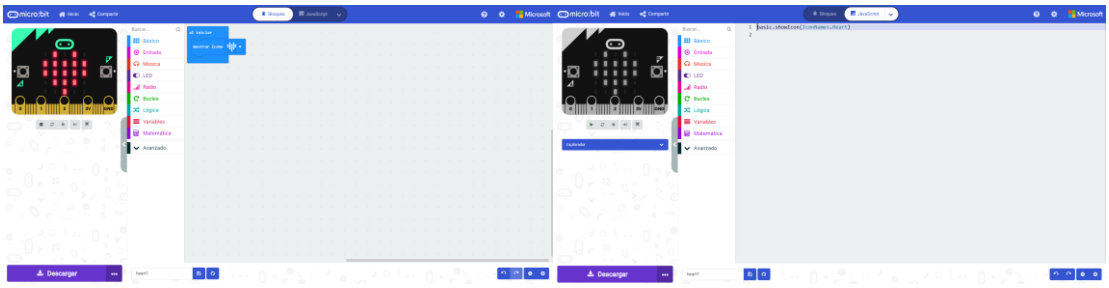


Figura 9: Comparación del mismo programa a través de las dos representaciones. El código de la derecha es generado automáticamente tras seleccionar los bloques deseados.

Debido al interés que tenemos sobre el paso a la programación textual vamos a desgranar las características que nos presenta el editor JavaScript.

- Basado en el editor Monaco (Microsoft, s. f.-b) que es el editor en el que se basa VScode (un entorno de desarrollo ligero muy popular en entornos académicos y profesionales). Esto le aporta cierta robustez y soporte. El editor Monaco tiene una serie de características bastante extensa, pero en esta versión para MakeCode se han reducido:
    - Sugerencias.
    - Autocompletado.
    - Información de parámetros.
    - Información rápida.
    - Finalización de código (Generación de llaves, paréntesis).
    - Errores de código.
    - Zoom.
    - Comandos de búsqueda y reemplazo.
    - Menú contextual.
    - Comandos y atajos rápidos.
    - Coloreo con el sistema de colores de la representación por bloques.

4. Propuesta didáctica.

Esta propuesta didáctica se plantea como una serie de actividades prácticas a desarrollar en la asignatura de TIC de 1º de Bachillerato y cuyo objetivo es el de lograr un paso adecuado de la programación visual a la programación textual. Si rescatamos los contenidos de la asignatura la programación conforma un bloque completo. Nuestra propuesta didáctica no pretende presentar una unidad didáctica completa que resuma la acción en este bloque sino ayudar en la elaboración de determinadas actividades prácticas que encajen dentro de ésta. Los motivos de esta reducción son, básicamente, temporales ya que debido a la introducción de todos los aspectos técnicos la extensión del trabajo se excedería, por mucho, lo establecido en la guía del TFM. Aún así, intentaremos, enlazar cada una de las actividades propuestas con sus objetivos, metodologías, materiales, evaluación, etc.

Es decir, nuestra aportación sigue una modalidad de clases prácticas (De Miguel, 2005). Estas prácticas dispondrán de un guion lo que permitirá que el profesor actúe con un papel de apoyo. Además, se plantean actividades dentro de cada sesión de prácticas que encajen en un modelo de aprendizaje cooperativo donde las competencias sociales y organizativas suben de nivel. Todos los detalles y más de cada una de las sesiones prácticas y actividades serán descritos en los epígrafes siguientes.

Ubicación y temporalización

La presente propuesta se encuadra dentro de los contenidos del bloque 5: *Programación* de la asignatura de Tecnologías de la Información y la Comunicación I de 1º de Bachillerato. Teniendo en cuenta que el índice es orientativo si resulta interesante estudiar los anteriores bloques antes de empezar con programación. Es por esto por lo que mantenemos el orden presentado en el currículo de Cantabria. Otra razón para dejarlo al finalizar el bloque es la gran importancia que recobra en Tecnologías de la Información y la Comunicación II de 2º de Bachillerato donde representa el primer bloque. Aún así esta propuesta puede ser transparente en su ubicación. Nuestro planteamiento concreto la sitúa en el tercer bloque con 5 sesiones de 1 hora que

pudieran intercalarse con sesiones teóricas ya que esta propuesta no es una propuesta completa de unidad didáctica.

Actividades propuestas

Las actividades propuestas a continuación están secuenciadas de forma incremental. Las primeras nos ayudarán a familiarizarnos con el entorno micro:bit a la par que descubrimos de forma activa las posibilidades que nos presenta. Tras cada práctica el nivel de dificultad se verá incrementado y el enfoque irá orientándose hacia la programación textual (sin dejar de lado el soporte por bloques que nos puede ayudar a afianzar elementos). Cada una de las prácticas propuestas estará relacionada con uno o más contenidos descritos en el currículo y, además, tratará de abordar diferentes metodologías.

En la tabla siguiente se presenta una visión resumida de las prácticas propuestas.

Práctica	*Resumen*
P0	Primeros pasos con micro:bit y JavaScript.
P1	Variables y constantes.
P2	Funciones y métodos.
P3	Estructuras de control I.
P4	Estructuras de control II.
P5	Proyecto.

A continuación, se va a presentar cada una de las prácticas relacionando sus objetivos, contribución a las competencias, satisfacción de estándares de aprendizaje, etc.

P0. Primeros pasos con micro:bit y JavaScript.

En esta práctica descubriremos la placa micro:bit y el lenguaje de programación JavaScript. Al inicio de la práctica se realizará una breve descripción de la placa y el editor MakeCode. A continuación, seguiremos el guion para realizar un pequeño programa a través de la programación por bloques. Tras la finalización

de esta parte se introducirá JavaScript y se explorará la traducción generada por el editor.

Esta práctica tiene un objetivo claramente introductorio donde el alumno tomará contacto con la placa y del entorno de edición MakeCode. En un primer lugar experimentará las opciones por programación en bloques y finalizará con una pequeña introducción a JavaScript y al editor correspondiente.

Los principales contenidos de esta práctica son de tipo instructivo ya que se trata de una primera práctica introductoria. Se aprenderá a conectar la placa, a programarla, a ejecutar los programas. Respecto al lenguaje se aprenderán aspectos básicos como su sintaxis básica (llaves, identación, nomenclatura).

Los conceptos previos necesarios son bajos. En general sería una buena iniciación que conocieran programación por bloques (Google, s. f.; MIT, s. f.). Los elementos propios de la placa son sencillos y no requieren mucho conocimiento previo.

Las metodologías seguidas en esta práctica podrían encajarse dentro de una instrucción tradicional sobre la descripción de las tareas, pero fuertemente influenciado por metodologías como “aprender haciendo” (se va a conocer el funcionamiento de la placa trabajando sobre la propia placa y realizando pruebas). Además, tendremos a la vista metodologías de resolución de problemas de forma guiada, pero en la que el alumno tiene el control y cierta gamificación. En general estas y muchas otras metodologías serán vistas a lo largo de nuestras prácticas como veremos posteriormente.

El material necesario para esta práctica:

- Ordenador y conexión a internet. La conexión a internet podría no ser imprescindible si descargáramos la versión offline de MakeCode (aunque si altamente recomendable).
- Placa micro:bit y cable micro-usb. (Batería opcional)
- Guion de prácticas.

P1. Variables y constantes.

Al comienzo de la clase se dará una pequeña descripción magistral sobre el significado de variable y constante haciendo uso de conocimientos relacionados por los alumnos. En este caso se entablará la relación con las matemáticas. A continuación, se realizarán dos programas guiados y se propondrá manipular el resultado final con ideas propias por el alumno.

Los contenidos a desarrollar son los conceptos de variables y constante (entendiendo esta última como un caso restringido de la primera). Se aprenderá a declarar variables y dar una utilidad a este concepto con el desarrollo de dos programas. Intentaremos limitar el uso de elementos a estudiar en prácticas posteriores, pero si no fuera posible se explicará brevemente por parte del profesor en el transcurso de la práctica. Con esto intentaremos no ser demasiado planos en los ejemplos propuestos y despertar cierta motivación por el aprendizaje de los nuevos elementos aparecidos.

Los conceptos previos necesarios serán los logrados en las prácticas anteriores y conceptos multidisciplinares relacionados (concepto de variable o constante en matemáticas o física y química).

Las metodologías usadas son las mismas ya expresadas en la práctica anterior haciendo un mayor hincapié en el aprendizaje basado en proyectos y la gamificación (ya que incluimos un juego en uno de los programas).

El material necesario para esta práctica:

- Ordenador y conexión a internet. La conexión a internet podría no ser imprescindible si descargáramos la versión offline de MakeCode (aunque si altamente recomendable).
- Placa micro:bit y cable micro-usb. (Batería opcional)
- Guion de prácticas.

P2. Funciones y métodos.

El objetivo de esta práctica es aprender el concepto de función o método. Para ello se realizará una breve exposición magistral sobre el concepto y la sintaxis

general de JavaScript para las funciones. De la misma forma que en la práctica anterior se hará hincapié en la relación presente con las matemáticas.

Los contenidos a desarrollar relacionados con las variables van desde la declaración de las funciones, tipos de funciones, estructura general de la sintaxis y uso práctico de las funciones. De forma más breve se comentarán aspectos avanzados de las funciones presentes en el entorno micro:bit aunque no se profundizará en ellos.

Los conceptos previos necesarios serán los logrados en las prácticas anteriores y conceptos multidisciplinares relacionados (concepto de función en matemáticas).

Las metodologías usadas para esta práctica se alejan un poco de la gamificación de la práctica anterior para mantener un mismo enfoque aprender haciendo, aprendizaje basado en proyectos y aprendizaje colaborativo.

El material necesario para esta práctica:

- Ordenador y conexión a internet. La conexión a internet podría no ser imprescindible si descargáramos la versión offline de MakeCode (aunque si altamente recomendable).
- Placa micro:bit y cable micro-usb. (Batería opcional)
- Guion de prácticas.

P3. Estructuras de control I

El objetivo de esta práctica es estudiar un tipo básico de estructura de control: los condicionales. Estas estructuras han sido usadas en las prácticas anteriores, pero aquí se detallará brevemente su sintaxis y significado y se aprenderá a usar en dos programas sencillos.

Los contenidos a desarrollar son el uso de estas estructuras de control dentro de programas para comprender su funcionamiento. Haremos uso de breves descripciones iniciales inspiradas en la documentación básica del lenguaje y el trabajo sobre dos prácticas guiadas. Las descripciones iniciales requerirán de

una gran cantidad de ejemplos basados en aspectos más o menos cotidianos para rebajar la dificultad abstracta del concepto.

Los conceptos previos necesarios serán los logrados en las prácticas anteriores y conceptos multidisciplinares relacionados como pudieran ser en el campo de las matemáticas, circuitos lógicos, etc.

La metodología vuelve a ser similar a la de las prácticas anteriores. Comenzamos con un poco de clase magistral con ejemplos de sintaxis y manejo de las estructuras en el lenguaje para dar paso al uso de prácticas guiadas para que el alumno aprenda a manejar los conceptos. Dentro de estas prácticas guiadas hemos rescatado la utilización de un juego simple y reconocer las metodologías aprender haciendo que tanto estamos usando en esta propuesta.

El material necesario para esta práctica:

- Ordenador y conexión a internet. La conexión a internet podría no ser imprescindible si descargáramos la versión offline de MakeCode (aunque si altamente recomendable).
- Placa micro:bit y cable micro-usb. (Batería opcional)
- Guion de prácticas.

P4. Estructuras de control II

Continuando con la práctica anterior se seguirá con el concepto de estructura de control. En este caso avanzaremos con estructuras de repetición. De la misma forma que en el anterior caso estas estructuras se han ido viendo de una u otra forma en las anteriores prácticas, pero en esta sesión serán tratadas de una forma más exhaustiva determinando su definición, su sintaxis básica y sus posibles usos.

Los contenidos de la práctica serán el aprendizaje a través de la creación de dos programas de los bucles for y while en el lenguaje JavaScript. Se comenzará trabajando sobre los conceptos básicos inspirados en la documentación del lenguaje y se acabará usando estas dos prácticas guiadas para afianzar los conceptos.

Los conceptos previos necesarios serán los logrados en las prácticas anteriores y conceptos multidisciplinares relacionados como pudieran ser en el campo de las matemáticas, circuitos lógicos, etc.

La metodología consistirá en una exposición teórica inspirada en la documentación del lenguaje y la aportación de numerosos ejemplos cotidianos para entender los conceptos. Una vez esta parte se concluye se comienza con el desarrollo de las prácticas guiadas que vuelven a incidir en varias metodologías anteriormente expuestas. Centrándonos en una metodología "aprender haciendo", aprendizaje basado en resolución de problemas, aprendizaje basado en proyectos y gamificación.

El material necesario para esta práctica:

- Ordenador y conexión a internet. La conexión a internet podría no ser imprescindible si descargáramos la versión offline de MakeCode (aunque si altamente recomendable).
- Placa micro:bit y cable micro-usb. (Batería opcional)
- Guion de prácticas.

P5. Proyecto

El objetivo de esta práctica consiste en que colaborando en pequeños grupos de dos alumnos construyan un sensor de noche. La idea es sencilla pero la clave de esta práctica es la introducción del pseudocódigo y con el uso de éste poder desarrollar el algoritmo deseado. Además, se permitirá cierta creatividad en los detalles de la práctica como se puede apreciar en el guion de ésta. Al finalizar la práctica se realizará una breve exposición comentando el funcionamiento y el código realizado para poder ver las diferencias entre grupos.

Los contenidos de esta sesión recogen todos los anteriores y añaden una fase de diseño de software y el concepto de pseudocódigo tan utilizado para describir algoritmos con independencia del lenguaje de programación.

Como se ha comentado los conceptos previos necesarios serán todos los adquiridos en las prácticas anteriores y conocimientos multidisciplinares en otros ámbitos.

Las metodologías usadas seguidas serán las mismas que en prácticas anteriores añadiendo un mayor peso a la creatividad del alumno, así como al aprendizaje basado en proyectos y el trabajo en equipo. Además, se incluye una parte final de exposición oral ante la clase. De forma más concreta y dado que resulta de la práctica más compleja el profesor presentará conceptos relacionados con el diseño de software y el uso del pseudocódigo para enlazar con la presentación de la práctica. Esta se presentará como una problemática a resolver y se dejará a los grupos trabajar en sus posibles soluciones. En un primer lugar sin aportar el pseudocódigo. En el transcurso de la prueba se presentará un pseudocódigo que resuelve el problema (aunque se hace hincapié que podría haber distintas soluciones) y se vuelve a dejar trabajar a los equipos. Una vez finalizado el programa cada grupo deberá documentar su programa para elaborar una exposición oral y realizar un pseudocódigo asociado (con el que presentar su solución al resto de la clase).

El material necesario para esta práctica:

- Ordenador y conexión a internet. La conexión a internet podría no ser imprescindible si descargáramos la versión offline de MakeCode (aunque si altamente recomendable).
- Placa micro:bit y cable micro-usb. (Batería opcional)
- Guion de prácticas.

A pesar de que micro:bit permite el uso de sensores externos hemos querido eliminarlos de las ecuaciones de estas prácticas para hacer aún más accesible el trabajo con la placa. Prácticamente se ha convertido en un sistema “plug and play” que permite un mayor acceso a una mayor cantidad de los alumnos. El uso más avanzado de sensores externos o de placas más complejas se deja como motivación al alumno y se trabajará en la asignatura del año siguiente: Tecnología de la Información y Comunicación II (2º de Bachillerato).

5. Competencias involucradas

En este apartado se discute como son trabajadas las competencias en la propuesta didáctica presentada. En esta propuesta se ven reflejadas de una u

otra forma las 7 competencias clave. En la siguiente tabla se muestra la relación de la contribución de la propuesta a las diferentes competencias.

Competencias	*Resumen*
Comunicación lingüística	• Vocabulario especifico. Lectura y comprensión de los guiones. • Análisis y diseño de código. • Expresión oral ante una audiencia.
Competencia matemática y competencias básicas en ciencia y tecnología	• Pensamiento lógico y abstracto. Algoritmia básica. • Sensores y actuadores en robótica.
Competencia digital	• Manejo de software basado en web y escritorio. Manejo de editores de código. • Trabajo con hardware propio de robótica como sensores y actuadores. • Uso de herramientas TIC como internet para buscar información.
Aprender a aprender	• Capacidad de evolucionar el contenido teórico para solucionar problemas mediante el desarrollo de programas informáticos.
Competencias sociales y cívicas	• Trabajo en grupo. • Desarrollo de habilidades sociales como exposiciones ante un público. • Implicación del desarrollo de software en beneficio social.
Sentido de iniciativa y espíritu emprendedor	• Diseño de una solución basada en programa informático para solucionar un problema.
Conciencia y expresiones culturales	• Posibilidad de realizar expresiones creativas a través de las TIC.

• Pertenencia a un grupo diverso en los trabajos en grupo.

6. Evaluación

Debido a que esta propuesta no conforma una unidad didáctica completa se recomienda integrarla en los criterios de evaluación y calificación del bloque. De igual forma recordaremos los criterios de evaluación y estándares de aprendizaje que atañen a este bloque.

Criterio de evaluación	***Estándar de aprendizaje***
Aplicar algoritmos a la resolución de los problemas más frecuentes que se presentan al trabajar con estructuras de datos. • *Competencia lingüística*	1.1 Desarrolla algoritmos que permitan resolver problemas aritméticos sencillos elaborando sus diagramas de flujo correspondientes.
Analizar y resolver problemas de tratamiento de información dividiéndolos en subproblemas y definiendo algoritmos que los resuelven. • *Competencia digital*	2.1 Escribe programas que incluyan bucles de programación para solucionar problemas que implique la división del conjunto en parte más pequeñas.
Analizar la estructura de programas informáticos, identificando y relacionando los elementos propios del lenguaje de programación utilizado. • *Competencia digital*	3.1 Obtiene el resultado de seguir un pequeño programa escrito en un código determinado, partiendo de determinadas condiciones.
Conocer y comprender la sintaxis y la semántica de las construcciones básicas de un lenguaje de programación. • *Competencia lingüística*	4.1 Define qué se entiende por sintaxis de un lenguaje de programación proponiendo ejemplos concretos de un lenguaje determinado.

Realizar pequeños programas de aplicación en un lenguaje de programación determinado aplicándolos a la solución de problemas reales. • *Competencia para aprender a aprender*	5.1 Realiza programas de aplicación sencillos en un lenguaje determinado que solucionen problemas de la vida real.

Se ha asociado en cada uno de los criterios de evaluación su relación con la competencia más relevante (teniendo en cuenta que normalmente siempre hay varias presentes).

Instrumentos de evaluación.

Los instrumentos de evaluación empleados que proponemos seguir para la realización de esta propuesta son los siguientes:

1. Actitud durante la clase, valorando los siguientes aspectos:
    a. Participación en clase.
    b. Actitud positiva y favorable hacia el aprendizaje.
    c. Comportamiento adecuado en los grupos de trabajo.
2. Actividades entregables:
    a. Orden y estructuración.
    b. Claridad en la expresión.
    c. Resultados.

Esta propuesta no lleva incluida la valoración de una prueba escrita, pero si conlleva la entrega y evaluación de cada una de las sesiones de trabajo propuestas.

Criterios de calificación.

Criterios de calificación	
Actitud y Participación	**10 %**
Actividades	**90 %**

La nota mínima para las actividades será de un 4 y serán recuperables.

VII. Conclusiones

El mundo ha evolucionado y está haciéndolo en estos mismos instantes. El proceso es imparable y durante él las disciplinas de carácter tecnológico están recogiendo cada vez más peso. Esta visión está prácticamente consensuada a nivel social y es por esto por lo que la introducción de disciplinas afines al mundo de las tecnologías de la información está siendo un aspecto clave en los currículos. Si tratáramos estos cambios a nivel laboral o profesional nos quedaríamos en la superficie ya que estos van más allá. Sin ir más lejos, en plena pandemia se ha propuesto una app del ministerio de sanidad llamada Radar COVID con la que ayudar en la tarea de los rastreos de contagiados. Este es uno de cientos de ejemplos de cómo el desarrollo tecnológico puede servir a la sociedad y está cada vez más presente en nuestras vidas.

En el presente trabajo hemos querido introducir una disciplina dentro de las ciencias de la computación clave: la programación. Lo hemos realizado para el curso de 1º de Bachillerato y con la ayuda de un soporte físico en el que trabajar los aspectos prácticos de la programación.

En esta línea se ha presentado a lo largo del trabajo la situación actual de la disciplina en los currículos de enseñanza de Europa, España y Cantabria. Describiendo brevemente aquellos más acertados y las deficiencias encontradas en el caso español. A partir de ahí se ha establecido una comparativa de las opciones que el mercado nos ofrece para la realización de la propuesta didáctica. Se ha escogido como soporte de robótica la placa micro:bit y el lenguaje de programación textual JavaScript. Las razones han sido descritas en los epígrafes anteriores, pero, en resumen, la placa resultaba la opción más accesible con un editor de código online en el que poder trabajar desde programación visual y textual al mismo tiempo. Escogimos JavaScript y no python por tratarse de la solución nativa del editor de micro:bit y asegurarnos un mejor soporte. Además,

se trata de un lenguaje con gran popularidad y un estándar en la programación web.

Tras esto se elaboró la propuesta didáctica referida al aprendizaje de la programación textual (a través del lenguaje JavaScript) con el soporte de la placa micro:bit. Esta combinación resulta beneficiosa ya que nos permite valorar nuestros programas mediante funcionamiento físico que podemos ver y tocar. Esta propuesta integra una serie de sesiones prácticas en las que variamos las metodologías usadas. Podríamos decir que gracias a este enfoque trabajamos metodologías tan diversas como el aprendizaje basado en proyectos, aprendizaje basado en problemas, aprendizaje cooperativo, gamificación, aprendizaje basado en competencias, aprender haciendo, etc.

A pesar de lo diverso y la cantidad de sesiones propuestas no se ha podido ejecutar en la práctica debido a la situación de suspensión de clases presenciales a raíz de la pandemia de COVID-19. El siguiente paso consistiría en ejecutar esta propuesta en el centro para evaluar y observar los resultados. A raíz de la buena valoración o de los cambios necesarios para mejorar la propuesta se podría motivar a los estudiantes con la participación en certámenes de robótica como un paso futuro.

Una vez exploradas las líneas futuras referidas a la ejecución práctica de la propuesta nos queda establecer los siguientes pasos hacía la integración con 2º de Bachillerato. En este curso se introducen aún más conceptos de la programación textual (siempre basada en lenguajes orientados a objetos) con el que el enfoque debería evolucionar en complejidad. Mi particular propuesta es la utilización de una nueva plataforma: la Raspberry Pi y mantener el lenguaje de programación JavaScript. En cuanto a lenguaje me presenta más dudas ya que deberíamos introducir el entorno servidor node.js y podría complicar la abstracción de código. Otras opciones típicas serían trabajar con lenguajes más apropiados como Python.

VIII. Bibliografía

Adell Segura María Ángeles Llopis Nebot Francesc Esteve Mon María Gracia Valdeolivas Novella, J. 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

Anzai, Y., y Simon, H. A. (1979). The theory of learning by doing. *Psychological Review*, *86*(2), 124-140. https://doi.org/10.1037/0033-295X.86.2.124

Arduino. (s. f.). *Arduino*. Recuperado 11 de agosto de 2020, de https://www.arduino.cc/

Aru-Chabilan, H. (2020). Tiger Leap for digital turn in the Estonian education. *Educational Media International*, *57*(1), 61-72. https://doi.org/10.1080/09523987.2020.1744858

Bargury, I. Zur, Haberman, B., Cohen, A., Muller, O., Zohar, D., Levy, D., y Hotoveli, R. (2012). Implementing a new Computer Science Curriculum for middle school in Israel. *Proceedings - Frontiers in Education Conference, FIE*. https://doi.org/10.1109/FIE.2012.6462365

BBC. (2015). *Micro:bit*. https://microbit.org/

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., y Punie, Y. (2016). Developing Computational Thinking in Compulsory Education - Implications for policy and practice. En *Joint Research Centre (JRC)* (Número June). https://doi.org/10.2791/792158

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113-124. https://doi.org/10.1109/TIT.1956.1056813

Cristina Delgado. (2019, abril 25). Más del 20% de los empleos de España pueden acabar en manos de robots y máquinas | Economía | EL PAÍS. *El País*. https://elpais.com/economia/2019/04/25/actualidad/1556181336_514611.html

De Miguel, M. (2005). *Modalidades de enseñanza centradas en el desarrollo de competencias: orientaciones para promover el cambio metodológico en el espacio europeo de educación superior*.

Denning, P. J. (2003). Great principles of computing. En *Communications of the ACM* (Vol. 46, Número 11, pp. 15-20). https://doi.org/10.1145/948383.948400

Domènech-Casal, J., Lope, S., y Mora, L. (2019). Which projects design and which difficulties express on Project-Based Learning Secondary Education teachers. Analysis of 87 project proposals. *Revista Eureka*, *16*(2), 2203. https://doi.org/10.25267/Rev_Eureka_ensen_divulg_cienc.2019.v16.i2.2203

Gal-Ezer, J., y Stephenson, C. (2014). A tale of two countries: Successes and challenges in K-12 computer science education in Israel and the United States. *ACM Transactions on Computing Education*, *14*(2), 1-18. https://doi.org/10.1145/2602483

García-Peñalvo, F. (2016). *TACCLE 3, O5: An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers KA2 project " TACCLE 3 – Coding " (2015-1-BE02-KA201-012307)*. 2016. https://doi.org/10.5281/zenodo.165123

Google. (s. f.). *Blockly*. Recuperado 13 de agosto de 2020, de https://developers.google.com/blockly

Hazzan, O., Gal-Ezer, J., y Blum, L. (2008). A model for high school computer science education. *ACM SIGCSE Bulletin*, *40*(1), 281-285. https://doi.org/10.1145/1352322.1352233

Hernando Barragán. (2003). *Wiring*. http://wiring.org.co/

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., Pal, Y., Jackova, J., y Jasute, E. (2015). A global snapshot of computer science education in K-12 schools. *ITiCSE-WGP 2015 - Proceedings of the 2015 ITiCSE Conference on Working Group Reports*, 65-

83. https://doi.org/10.1145/2858796.2858799

Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado. (2018). *Programación, robótica y pensamiento computacional en el aula*. http://code.intef.es/wp-content/uploads/2017/09/Pensamiento-Computacional-Fase-1-Informe-sobre-la-situación-en-España.pdf

Kidscode. (s. f.). *Overview of MakeCode*. Recuperado 13 de agosto de 2020, de https://docs.kidscode.co.za/microbit-beginners/overview-of-makecode

Lego. (s. f.). *MINDSTORMS*. Recuperado 11 de agosto de 2020, de https://www.lego.com/es-es/themes/mindstorms

MakeCode. (s. f.-a). *Bluetooth Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/reference/bluetooth

MakeCode. (s. f.-b). *Button Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/reference/input/on-button-pressed

MakeCode. (s. f.-c). *Compass Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/reference/input/compass-heading

MakeCode. (s. f.-d). *LED Screen Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/device/screen

MakeCode. (s. f.-e). *Light Level Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/reference/input/light-level

MakeCode. (s. f.-f). *On Gesture Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/reference/input/on-gesture

MakeCode. (s. f.-g). *Pins Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/reference/pins

MakeCode. (s. f.-h). *Radio Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/reference/radio

MakeCode. (s. f.-i). *Temperature Reference*. Recuperado 12 de agosto de 2020, de https://makecode.microbit.org/reference/input/temperature

Mercedes Martín López. (2017). Programación: diseño curricular actual en las etapas no universitarias del sistema educativo español. *IE Comunicaciones: Revista Iberoamericana de Informática Educativa, ISSN-e 1699-4574, Nº. 26, 2017 (Ejemplar dedicado a: Julio-Diciembre), págs. 38-45*, *26*, 38-45. https://dialnet.unirioja.es/servlet/articulo?codigo=6231882&info=resumen&idioma=SPA

Micro:bit. (s. f.). *Set up | micro:bit*. Recuperado 12 de agosto de 2020, de https://microbit.org/get-started/first-steps/set-up/

Micro:bit Foundation. (s. f.). *Hardware Micro:bit*. Recuperado 12 de agosto de 2020, de https://tech.microbit.org/hardware/

Microsoft. (s. f.-a). *Lego Makecode*. Recuperado 11 de agosto de 2020, de https://makecode.mindstorms.com/

Microsoft. (s. f.-b). *microsoft/monaco-editor: A browser based code editor*. Recuperado 13 de agosto de 2020, de https://github.com/microsoft/monaco-editor

MIT. (s. f.). *Scratch - Imagine, Program, Share*. Recuperado 13 de agosto de 2020, de https://scratch.mit.edu/

Mozilla Foundation. (s. f.). *JavaScript | MDN*. Recuperado 11 de agosto de 2020, de https://developer.mozilla.org/es/docs/Web/JavaScript

Peña, R. (2015). Python como primera aproximación a la programación. *ReVisión, Vol. 8, Nº. 2, 2015*, *8*(2), 1. https://dialnet.unirioja.es/servlet/articulo?codigo=5830314&info=resumen&idioma=SPA

Python Foundation. (s. f.-a). *PEP 20 -- The Zen of Python | Python.org*. Recuperado 11 de agosto de 2020, de https://www.python.org/dev/peps/pep-0020/

Python Foundation. (s. f.-b). *Python Software Foundation | Python Software Foundation*. Recuperado 11 de agosto de 2020, de https://www.python.org/psf/

Raspberry Pi Foundation. (s. f.). *Raspberry Pi* . Recuperado 11 de agosto de 2020, de https://www.raspberrypi.org/

Riesco Albizu, M., Díaz Fondón, M. Á., Álvarez Gutiérrez, D., López Pérez, B., Cernuda del Río, A., y Juan Fuente, A. A. (2014). Informática: materia esencial en la educación obligatoria del siglo XXI. *ReVisión, Vol. 7, Nº. 3, 2014 (Ejemplar dedicado a: Revista de Investigación en Docencia Universitaria de la Informática)*, *7*(3), 6. https://dialnet.unirioja.es/servlet/articulo?codigo=5828258&info=resumen&idioma=SPA

Sanders, M. E. (2008). *STEM, STEM Education, STEMmania*. Technology Teacher. https://vtechworks.lib.vt.edu/handle/10919/51616

Stack Overflow. (s. f.). *Stack Overflow Developer Survey 2020*. Recuperado 11 de agosto de 2020, de https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages

U.K. (s. f.). *National curriculum in England: computing programmes of study*. Recuperado 8 de agosto de 2020, de https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

Webb, M., Davis, N., Bell, T., Katz, Y., Reynolds, N., Chambers, D. P., y Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, *22*(2), 445-468. https://doi.org/10.1007/s10639-016-9493-x

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33. https://doi.org/10.1145/1118178.1118215

IX. Anexo: Prácticas

1. P0: Primeros pasos con micro:bit y JavaScript.

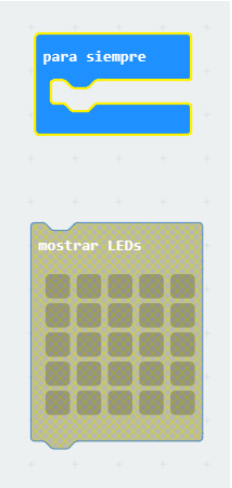
Introducción.

Micro:bit es un pequeña placa programable (similar a un ordenador) que tiene a su disposición un amplio abanico de sensores (y se pueden añadir nuevos). A lo largo de la serie de prácticas trabajaremos con la placa aprendiendo a programar en JavaScript.

En primer lugar, se seguirá el guion para la programación por bloques y, posteriormente, se comprobará la traducción a JavaScript generada. Así como rehacer la práctica de forma textual.

Desarrollo.

- Conecta la placa por micro-usb al ordenador y dirígete a la siguiente web https://makecode.microbit.org/
- Explora los distintos menús y componentes que se nos presentan. Lo más destacado es la presencia de un simulador (nos dará la vista previa del funcionamiento de nuestro código), un espacio de trabajo donde alojar los bloques o código y un menú de herramientas.
- El menú de herramientas es amplio y está bien familiarizarse con todas las familias de elementos que tenemos.
- Estas tres fases irán guiadas por el profesor para poder realizar nuestro primer programa a continuación.



Corazón intermitente.

Nuestro primer programa consistirá en un corazón intermitente. Para ello necesitaremos usar dos bloques. El bloque mostrar LEDs en un bucle para siempre.

Podemos comprobar su buen funcionamiento desde el simulador. Una vez veamos que se comporta como deseamos vamos a transferir el programa a la placa. Para ello

descargamos el programa desde la web y lo arrastramos a la ruta de micro:bit.

Este programa ha sido muy sencillo, ahora vamos a ver la estructura del código JavaScript generado y probaremos a manipularlo un poco. El código JavaScript que deberíamos observar sería el siguiente.

```
basic.forever(function () {
    basic.showLeds(`
        . # . # .
        # # # # #
        # # # # #
        . # # # .
        . . # . .
        `)
    basic.showLeds(`
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        `)
})
```

Este es el código que representa nuestro programa. Vamos a explicar brevemente que significa cada parte. En un primer lugar se invoca una función llamada basic.forever() que representa el bloque para siempre. Es decir, lo que vaya dentro de esta función se ejecutará dentro de un bucle infinito. Lo que realizamos a continuación es un concepto un poco más complejo que es el de función anónima. En apartados posteriores se entenderá de una mejor forma. Esta función solo se puede ejecutar si es llamada por otra función y a su vez se compone de dos llamadas a la misma función basic.showLeds() que es la encargada de representar los LEDs encendidos y apagados (‘.’ Significa apagado y ’#’ significa encendido).

El siguiente paso es manipular un poco el código generado. En este caso vamos a cambiar la representación LED de la segunda llamada a la función basic.showLeds(). Podemos cambiarlo a nuestro antojo, como ejemplo vamos a dibujar un corazón más pequeño.

```
basic.forever(function () {
    basic.showLeds(`
        . # . # .
        # # # # #
        # # # # #
        . # # # .
        . . # . .
        `)
    basic.showLeds(`
        . . . . .
        . # # # .
        . # . # .
        . . # . .
        . . . . .
        `)
})
```

Finalmente podemos aprovechar una funcionalidad del editor y es la programación híbrida. Si buscas en el cajón de herramientas el bloque que deseas y lo despliegas en el editor textual lo sustituirá por código.

2. P1: Variables y constantes.

Introducción

Una variable en programación recoge la misma idea que la variable matemática. Una variable es un elemento que se emplea para almacenar y hacer referencia a otro valor. La idea principal entre variable y constante es que la variable puede ser reasignada o producirse un cambio en su valor y una constante se inicializa en un valor y este no puede ser modificado. Podríamos decir que una constante es una variable inmutable.

Para declarar una variable en JavaScript debemos usar las palabras reservadas *var* o *let* seguida del nombre que queremos usar. A su vez podemos inicializar o no esta variable. A continuación, podemos ver un código de ejemplo.

```
01. var miVariable = 10;
02. let miNombre = "Carlos";
```

La diferencia entre usar *var* o *let* residen en la historia de JavaScript. En una primera instancia solo existía *var* pero a medida que el lenguaje fue evolucionando se encontraron problemas para usarlo en determinados bloques. En general es algo que escapa en los objetivos del curso, pero la más destacable es que *let* reúne ciertas características que convierten la sintaxis del lenguaje en algo más ordenado y lógico. En JavaScript moderno se recomienda el uso de let siempre y cuando sea posible.

Las variables no deben confundirse con los valores, sino más bien entender que una variable es una referencia o identificador a una “caja” donde puedes almacenar muchos tipos de datos. En los ejemplos anteriores se debe entender de la siguiente forma: Declaro la variable miNombre y la inicializo con la cadena de texto “Carlos”. Este aspecto es importante ya que podemos actualizar el contenido de la variable a nuestro antojo.

```
01. miVariable = 15;
02. miNombre = "Alberto";
```

Otro aspecto importante en la declaración de variables es el tipado dinámico con el que trabaja JavaScript. Esto significa que al declarar la variable no debo

asignarle un tipo a la variable, lo que permite que la variable cambie su tipo de forma dinámica (en función de lo que contenga).

```
01. let myNumber = '500'; // Vaya, esto sigue siendo una cadena
02.
03. myNumber = 500; // mucho mejor - ahora este es un número
```

Las constantes en los inicios de JavaScript tampoco existían, pero de igual forma se introdujeron porque aportan una funcionalidad muy interesante a la hora de elaborar nuestro código. Para declarar la constante solo debemos usar la palabra reservada *const* y como se ha explicado anteriormente no se puede actualizar el valor de la constante.

```
01. const daysInWeek = 7;
02. daysInWeek = 8;
```

Se recomienda seguir la documentación de mozilla para estos y otros aspectos: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Variables

Desarrollo

En este apartado vamos a trabajar estos conceptos con la ayuda de micro:bit y siguiendo los dos siguientes guiones referidos a dos programas sencillos.

Calculadora

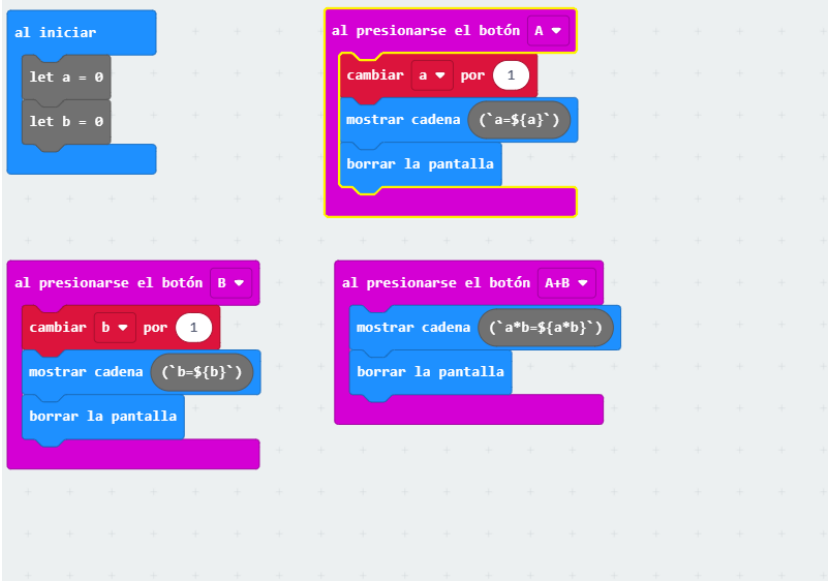
En este primer programa vamos a crear una calculadora en la que trabajaremos con los conceptos de variables. En este caso usaremos los dos botones para almacenar dos variables que finalmente multiplicaremos. A continuación, se presenta el código JavaScript comentado.

```
//Declaramos las dos variables inicializadas en 0
let a = 0
let b = 0
//Llamamos a la función encargada de actualizar el valor de la
variable a
//Cada vez que presionamos el boton se actualiza el valor de la
variable sumando 1
input.onButtonPressed(Button.A, function () {
    a += 1
    basic.showString(`a=${a}`)
    basic.clearScreen()
})
```

```
//Llamamos a la función encargada de actualizar el valor de la
variable b
//Cada vez que presionamos el boton se actualiza el valor de la
variable sumando 1
input.onButtonPressed(Button.B, function () {
    b += 1
    basic.showString(`b=${b}`)
    basic.clearScreen()
})
//Finalmente mostramos el producto de las dos variables por la
pantalla
input.onButtonPressed(Button.AB, function () {
    basic.showString(`a*b=${a*b}`)
    basic.clearScreen()
})
```

Este código presenta algún concepto avanzado que veremos en prácticas posteriores, pero podemos tomarlo como una guía permitiendo algún cambio menor en el código. (Por ejemplo se ha escrito el código que está incluido en los basic.showString() mediante interpolación de cadenas de texto).

Por otro lado, a continuación, se ve la vista resultante en el visor por bloques. Es importante destacar que se produce una “traducción” automática que modifica algo el código, pero en general son aspectos de ordenación y estructura.



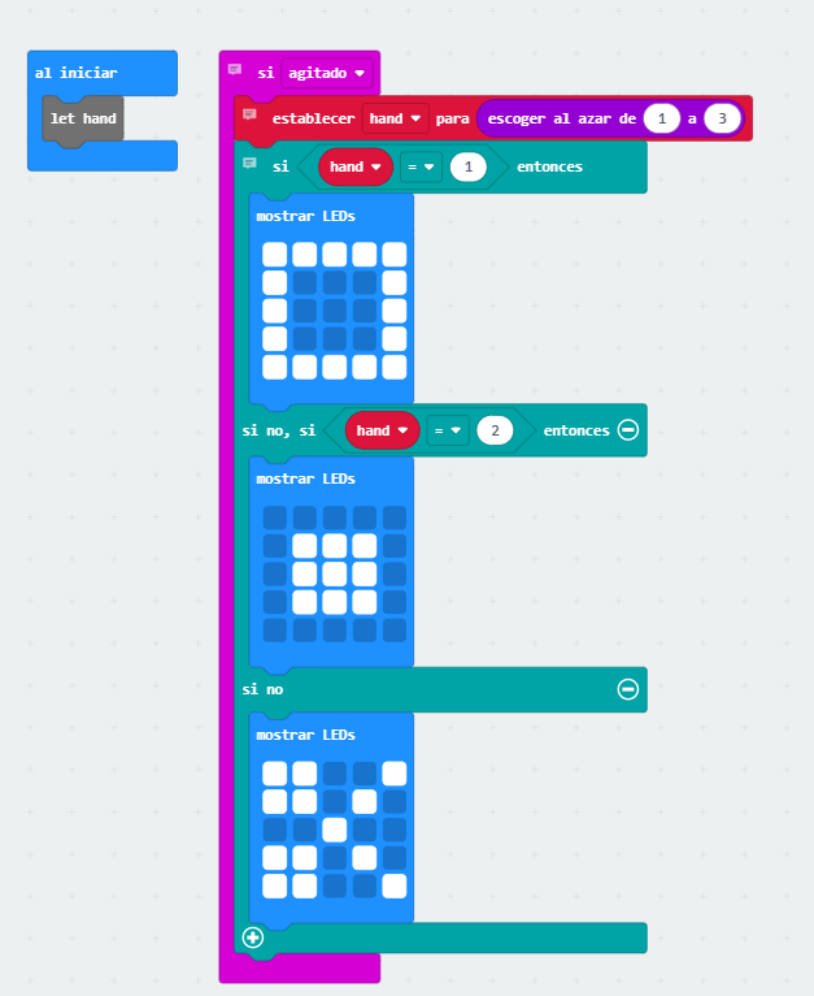
Piedra, papel o tijera

En este programa vamos a elaborar el juego piedra, papel o tijera. Para ello la idea básica es que la placa micro:bit sea un participante del juego. Para esto

declararemos una variable y la placa al ser agitada generará un número aleatorio del 1 al 3. Cada uno de estos números representará cada estado (1->papel, 2->piedra, 3->tijera). Esta parte de la práctica está pensada para hacer en parejas y así poder realizar pruebas de funcionamiento del juego completo (2 participantes). El código JavaScript comentado es el siguiente.

```
//Declaramos una variable
let hand
//Llamamos a la función encargada de ejecutar la selección de
piedra,papel o tijera
input.onGesture(Gesture.Shake, function () {
    //Una vez se ha dado el estado agitado la variable se actualiza
con un número aleatorio
    hand = randint(1, 3)
    //Si el número aleatorio es 1 la opción elegida es papel
    if (hand == 1) {
        basic.showLeds(`
            # # # # #
            # . . . #
            # . . . #
            # . . . #
            # # # # #
            `)
    } //Si el número aleatorio es 1 la opción elegida es piedra
    else if (hand == 2) {
        basic.showLeds(`
            . . . . .
            . # # # .
            . # # # .
            . # # # .
            . . . . .
            `)
    } //Si el número aleatorio es 1 la opción elegida es tijera
    else {
        basic.showLeds(`
            # # . . #
            # # . # .
            . . # . .
            # # . # .
            # # . . #
            `)
    }
})
```

De nuevo en este código se ha tenido que utilizar elementos más avanzados de estructuras de control, pero son fácilmente entendibles y explicables a lo largo de la práctica. La representación por bloques es la siguiente.


al iniciar
let hand
si agitado
establecer hand para escoger al azar de 1 a 3
si hand = 1 entonces
mostrar LEDs
si no, si hand = 2 entonces
mostrar LEDs
si no
mostrar LEDs

3. P2: Funciones y métodos.

Introducción

Las funciones son uno de los pilares en JavaScript. Una función es un procedimiento en JavaScript. Es decir, se trata de un conjunto de sentencias que realizan una tarea.

La definición de una función consiste en la palabra function, seguida por:

- El nombre de la función (opcional).
- Una lista de argumentos para la función, encerrados entre paréntesis y separados por comas.
- La sentencia de la función encerrada por llaves.

A continuación podemos ver un ejemplo de función denominada square().

```
01. function square(number) {
02.   return number * number;
03. }
```

La función square toma un argumento llamado number. La función consiste en retornar el cuadrado del argumento. La sentencia return especifica el valor que devuelve la función (opcional).

A continuación comentamos un aspecto importante dentro de JavaScript y es la expresión de función que tanto se usa en el editor de JavaScript de micro:bit. Por ejemplo, la función square podría haber sido definida de la siguiente forma:

```
01. let square = function(number) {return number * number};
02. let x = square(4) //x obtiene el valor 16
```

En este ejemplo se ha definido una variable por expresión de función. Además, la función es anónima (no tiene nombre).

Para ejecutar una función solo tenemos que nombrarla y fijar los argumentos necesarios para su ejecución. En los ejemplos anteriores hemos visto esta ejecución. En JavaScript las funciones pueden estar definidas por debajo de la línea donde son llamadas. Esto se define como función izada y es algo que no encontraremos en Python, por ejemplo.

Registro de temperatura.

El objetivo de este programa es el trabajo sobre funciones (anónimas y no) con las que vamos a registrar las temperaturas máximas y mínimas de la zona con la ayuda del sensor de temperatura de la placa micro:bit. La función principal encajará dentro de nuestro bucle infinito (también es una función en el editor de microbit, basic.forever()) y se verá como almacenamos en variables nuestros datos. El código de ejemplo es el siguiente.

```
// Se declaran las variables que vamos a necesitar. Inicializamos
el valor de minTemp y maxTemp
let minTemp = input.temperature();
let maxTemp = input.temperature();
let currentTemp: number;

// Bucle infinito donde se ejecutan nuestras acciones
basic.forever(function () {
    //Se actualiza la variable currentTemp
    currentTemp = input.temperature();
    //Se comprueba si la temperatura actual es menor que la minima
registrada.
    //Si es así se actualiza la temperatura minima
    if (currentTemp < minTemp){
        minTemp = currentTemp;
    }
    //Se comprueba si la temperatura actual es mayor que la maxima
registrada.
    //Si es así se actualiza la temperatura maxima
    if (currentTemp>maxTemp){
        maxTemp = currentTemp;
    }
    //Se muestra la temperatura minima al presionar el boton A
    input.onButtonPressed(Button.A, function () {
        basic.showNumber(minTemp);
    })
    //Se muestra la temperatura maxima al presionar el boton B
    input.onButtonPressed(Button.B, function () {
        basic.showNumber(maxTemp);
    })
    //Se muestra la temperatura actual al agitar la placa
    input.onGesture(Gesture.Shake, function () {
        basic.showNumber(currentTemp);
    })

})
```

A continuación se muestra una variación de este código en el que obviamos el uso de funciones anónimas por un enfoque más básico en el que definimos una función main() que es llamada en el bucle basic.forever().

```
// Se declaran las variables que vamos a necesitar. Inicializamos
el valor de minTemp y maxTemp
let minTemp = input.temperature();
let maxTemp = input.temperature();
let currentTemp: number;

function main(){
    //Se actualiza la variable currentTemp
    currentTemp = input.temperature();
    //Se comprueba si la temperatura actual es menor que la minima
registrada.
    //Si es así se actualiza la temperatura minima
    if (currentTemp < minTemp){
        minTemp = currentTemp;
    }
    //Se comprueba si la temperatura actual es mayor que la maxima
registrada.
    //Si es así se actualiza la temperatura maxima
    if (currentTemp>maxTemp){
        maxTemp = currentTemp;
    }
    //Se muestra la temperatura minima al presionar el boton A
    input.onButtonPressed(Button.A, function () {
        basic.showNumber(minTemp);
    })
    //Se muestra la temperatura maxima al presionar el boton B
    input.onButtonPressed(Button.B, function () {
        basic.showNumber(maxTemp);
    })
    //Se muestra la temperatura actual al agitar la placa
    input.onGesture(Gesture.Shake, function () {
        basic.showNumber(currentTemp);
    })
}


// Bucle infinito donde se ejecutan nuestras acciones
basic.forever(main)
```

Cuéntame un secreto

Esta práctica está preparada para ser realizada en parejas lo que nos va a permitir establecer una comunicación entre dos placas micro:bit. El objetivo del programa es elaborar un código único para el emisor y receptor (ambas placas serán emisor y receptor al mismo tiempo) en el que al presionar un botón en una

de las placas envíe un mensaje a la otra y pinten la misma figura en la matriz de led.

```
input.onButtonPressed(Button.A, function () {
    radio.sendString("yes")
    basic.showIcon(IconNames.Yes)
    basic.pause(500)
    basic.clearScreen()
})
radio.onReceivedString(function (receivedString) {
    if (receivedString == "yes") {
        basic.showIcon(IconNames.Yes)
        basic.pause(500)
        basic.clearScreen()
    } else if (receivedString == "no") {
        basic.showIcon(IconNames.No)
        basic.pause(500)
        basic.clearScreen()
    }
})
input.onButtonPressed(Button.B, function () {
    radio.sendString("no")
    basic.showIcon(IconNames.No)
    basic.pause(500)
    basic.clearScreen()
})
radio.setGroup(7)
```

En este ejemplo hay un detalle nuevo y es la llamada de la función de forma no anónima. Esto se debe realizar así ya que lo usamos como una condición para el bucle if.

4. P3 Estructuras de control I

Introducción.

En todo lenguaje de programación el código necesita realizar decisiones y llevar a cabo distintas acciones de acuerdo con diferentes condiciones. Por ejemplo, en un juego, si el número de vidas de un jugador se reduce a 0, entonces el juego se termina. A continuación, exploraremos las estructuras de control condicionales en JavaScript.

La sintaxis básica de una estructura de control condicional es el clásico *if…else* que se escribe de la siguiente forma en JavaScript.

```
if (condición) {
  código a ejecutar si la condición es verdadera
} else {
  ejecuta este otro código si la condición es falsa
}
```

- Declaración de la palabra *if* seguida de unos paréntesis.
- Los paréntesis rodean una condición. Esta condición retornará un true o false y determinará que código se ejecuta.
- Un conjunto de llaves, en las cuales tenemos un bloque de código a ejecutar si la condición es evaluada como true.
- La palabra clave *else*.
- Un conjunto de llaves, en las cuales tenemos un bloque de código a ejecutar si la condición es evaluada como algo distinto de true.

En general la introducción del *else* es opcional, pero es una buena práctica encerrar todos los valores posibles de la condición.

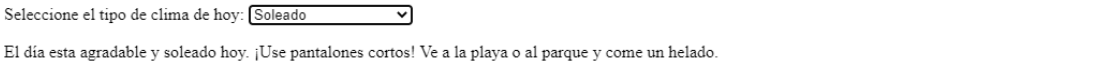
Existe una variante de la estructura en la cual podemos tener más posibles resultados (más de dos). Esto se logra con las sentencias *else if*. A continuación, vemos un ejemplo.

```
let seleccionar = document.querySelector('select');
let parrafo = document.querySelector('p');

seleccionar.addEventListener('change', establecerClima);

function establecerClima() {
  let eleccion = seleccionar.value;

  if (eleccion === 'soleado') {
    parrafo.textContent = 'El día esta agradable y soleado hoy. ¡Use pantalones cortos! Ve a la playa o al pa
  } else if (eleccion === 'lluvioso') {
    parrafo.textContent = 'Está lloviendo, tome un abrigo para lluvia y un paraguas, y no se quede por fuera
  } else if (eleccion === 'nevando') {
    parrafo.textContent = 'Está nevando — ¡está congelando! Lo mejor es quedarse en casa con una taza calient
  } else if (eleccion === 'nublado') {
    parrafo.textContent = 'No está lloviendo, pero el cielo está gris y nublado; podría llover en cualquier m
  } else {
    parrafo.textContent = '';
  }
}
```



El día esta agradable y soleado hoy. ¡Use pantalones cortos! Ve a la playa o al parque y come un helado.

En este ejemplo se ve el código JavaScript asociado a un elemento <select> de html. Esto no es importante, pero si lo es como se ha usado las estructuras condicionales en ella para enviar distintos mensajes en función del clima del día actual.

Cara o cruz

En este programa vamos a programar un lanzamiento de moneda. Para ello trabajaremos de nuevo con la generación de un valor aleatorio (recordar prácticas anteriores) y en función de su valor retornaremos la imagen de un cuadrado o una calavera desde la matriz de LEDs.

```
input.onButtonPressed(Button.A, function () {
    basic.showIcon(IconNames.Diamond)
    basic.showIcon(IconNames.SmallDiamond)
    basic.showIcon(IconNames.Diamond)
    basic.showIcon(IconNames.SmallDiamond)
    if (Math.randomBoolean()) {
        basic.showIcon(IconNames.Skull)
    } else {
        basic.showIcon(IconNames.Square)
    }
})
```

Brújula

En este programa vamos a hacer uso de la brújula incluida en la placa. Para ello haremos uso de las estructuras condicionales como en el ejemplo anterior, pero será necesario usar más opciones. Por ello hará falta el uso de *else if*. Las condiciones serán las siguientes.

- Si el ángulo retornado por la brújula es menor de 45º retornaremos Norte.
- Si no y si el ángulo retornado por la brújula es menor de 135º retornaremos Este.
- Si no y si el ángulo retornado por la brújula es menor de 225º retornaremos Sur.
- Si no y si el ángulo retornado por la brújula es menor de 315º retornaremos Oeste.
- Si no es ninguna de las anteriores retornaremos, de nuevo, Norte.

A continuación, se presenta un código de ejemplo con este funcionamiento.

```
basic.forever(function () {
    let degrees: number = input.compassHeading()
    if (degrees < 45) {
        basic.showString("N")
    } else if (degrees < 135){
        basic.showString("E")
    } else if (degrees < 225){
        basic.showString("S")
    } else if (degrees < 315){
        basic.showString("W")
    } else {
        basic.showString("N")
    }
})
```

5. P4 Estructuras de control II.

Introducción

A menudo querremos realizar alguna tarea repetidamente. Esto es lo que conocemos como estructuras de control de repetición. De forma más coloquial se conocen como bucles y en JavaScript disponemos de varios bucles distintos. A pesar de sus diferencias en esencia hacen lo mismo, repetir una acción un número de veces. Las diferencias radican en cómo se determina el inicio y el final del bucle y esto hace que alguna situación se resuelva de una forma más sencilla con un bucle u otro. Veremos los dos bucles más básicos a continuación.

El bucle *for* se repite hasta que la condición especificada se evalúa como *false*.

```
for ([expresionInicial]; [condicion]; [expresionIncremento])
  sentencia
```

Cuando un bucle *for* se ejecuta sucede lo siguiente:

- La expresión de inicialización *expresionInicial*, si existe, se ejecuta. Esta expresión habitualmente inicializa uno o más contadores del bucle. Esta expresión puede también declarar variables.
- Se evalúa la expresión *condicion*. Si el valor de *condicion* es *true*, se ejecuta la sentencia del bucle. Si el valor de *condicion* es *false*, el bucle finaliza.
- Se ejecuta la sentencia. Para ejecutar múltiples sentencias, use un bloque de sentencias ({ ... }) para agruparlas.
- Se ejecuta la expresión *expresionIncremento*, si hay una, y el control vuelve al paso 2.

El bucle *while* ejecuta la sentencia de su interior mientras la condición sea evaluada como verdadera.

```
while (condicion)
  sentencia
```

En el momento en que la condición camba a *false* la sentencia deja de ejecutarse y se sale del bucle y el control pasa a la sentencia inmediatamente posterior al bucle.

Música con los bucles.

En este primer programa se propone generar música con la placa micro:bit. Para ello se pide realizar algunos bucles for como se puede comprobar en el código siguiente.

```
input.onButtonPressed(Button.A, function () {
    music.setTempo(120)
    for (let index = 0; index < 2; index++) {
        music.playTone(262, music.beat(BeatFraction.Whole))
        music.playTone(294, music.beat(BeatFraction.Whole))
        music.playTone(330, music.beat(BeatFraction.Whole))
        music.playTone(262, music.beat(BeatFraction.Whole))
    }
    for (let index = 0; index < 2; index++) {
        music.playTone(330, music.beat(BeatFraction.Whole))
        music.playTone(349, music.beat(BeatFraction.Whole))
        music.playTone(392, music.beat(BeatFraction.Double))
    }
})
```

Distancia entre dos placas.

En el siguiente programa vamos a medir la distancia entre dos placas micro:bit a través de su radio. Para ello necesitaremos trabajar en equipo y usar dos placas. Una actuará como emisor de la señal de radio y la otra como receptor. La emisora enviará la señal con una intensidad y desde cierta distancia y la receptora pintará una gráfica en función de la intensidad de la onda de radio que esté recibiendo. El código del emisor se puede ver a continuación.

```
radio.setGroup(1)
radio.setTransmitPower(1)
basic.forever(function () {
    radio.sendString("1")
    basic.pause(200)
})
```

Y el código para el receptor:

```
radio.onReceivedString(function (receivedString) {
    signal =
radio.receivedPacket(RadioPacketProperty.SignalStrength)
    led.plotBarGraph(
    Math.map(signal, -95, -42, 0, 9),
    9
    )
})
let signal = 0
radio.setGroup(1)
```

6. P5 Proyecto.

Introducción

Esta sesión será algo más libre y creativa y consistirá en debatir sobre posibles soluciones tecnológicas ante problemas de seguridad vial.

En un primer lugar los estudiantes se pondrán en parejas y comentarán los principales problemas viales que tienen los peatones. Sobre todo, para jóvenes con algún tipo de vulnerabilidad particular.

Una vez se han identificado los problemas se propone que busquen ideas de como la tecnología puede ayudarnos a paliarlos. También será un trabajo en grupo, pero está vez en grupos de 4 (2 parejas de 2 estudiantes). Elaborad una lista de ideas y exponerla a los demás grupos.

Tras este primer trabajo en el que nos acercamos al diseño de soluciones tecnológicas daremos un paso más hacia el diseño de software y se propondrá encontrar posibles ayudas que nos puede dar la placa micro:bit (pensando ya en la forma de programar).

Finalmente se propone la solución ante la baja visibilidad nocturna a través de la placa micro:bit de un detector de noche basado en el sensor de luminosidad integrado en la matriz LED de la placa. Antes de ello se introduce un concepto nuevo: el pseudocódigo. Este código es una descripción de alto nivel más compacta e informal de un algoritmo o programa informático. La idea es que el psuedocódigo es más legible para un humano y nos da la base para entender los algoritmos que después deberemos programar.

En este ejemplo se presenta el siguiente pseudocódigo:

```
Algorithm 1: Detector luminosidad baja
  Result: Melodía de alerta y texto por pantalla si luminosidad menor de 100
  luminosidad = 0;
  while True do
      luminosidad = lightLevel();
      if luminosidad < 100 then
          startMelody();
          ShowString();
      end
  end
```

Figura 10: Pseudocódigo.

Donde los argumentos de las funciones se dejan a la elección del estudiante. Pueden experimentar con la batería de posibilidades que presenta la placa con sus melodías instaladas, así como definir que texto se ve en la placa.

Al finalizar la implementación del programa se propondrán hacia la clase explicando su código y mostrando los resultados obtenidos.

El código siguiente sería un código JavaScript correcto, pero no debe ser el seguido por los alumnos.

```
let light_ = 0
basic.forever(function () {
    light_ = input.lightLevel()
})
basic.forever(function () {
    if (light_ < 100) {
        music.startMelody(music.builtInMelody(Melodies.Funk),
MelodyOptions.Once)
        basic.showString("BE SAFE, BE SEEN!")
    }
})
```