UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: IRID: Infraestructura para la recolección inalámbrica de datos provistos por sensores en el marco de robótica educativa en nivel secundario

AUTORES: Juan Pablo Lozano Arce

DIRECTOR/A: Alejandra Beatriz Lliteras, Andrés Rodríguez

CODIRECTOR/A:

ASESOR/A PROFESIONAL:

Resumen

En este trabajo se propone e implementa una infraestructura -basada en el uso de placas Micro:Bit, ESP, sensores y conectividad WIFI- que permite a docentes y estudiantes de nivel secundario, mediante programación basada en bloques: configurar una conexión vía WIFI a un servidor, recolectar y enviar datos a un servidor REST, que los almacena y disponibiliza; y desde el cual es posible visualizar los datos. Para lo anterior, se implementó un protocolo de comunicación entre las placas y el servidor REST, por otro lado se definió una extensión para el entorno de programación MakeCode (para Micro:BIT) que simplifica aspectos técnicos tanto de conectividad como de envío de datos. Además, se personalizó una maqueta física y bloques específicos para llevar adelante una prueba con estudiantes de nivel secundario.

Palabras Clave

Arduino, Micro:Bit, ESP, Programación Basada en Bloques (PBB), Sensores, Transmisión Inalámbrica de Datos, Protocolo de Comunicación, WIFI, Servidor REST, Docker, Grafana, Python, MongoDB, JavaScript, JSON, Swagger, MakeCode, Educación de Nivel Secundario

Conclusiones

En este trabajo se presentó, por un lado, un protocolo de comunicación, luego, una extensión para el entorno MakeCode para trabajar el envío de datos obtenidos desde sensores de manera inalámbrica (vía Wifi) a un servidor. Se implementó un servidor REST que está diseñado para que los datos recolectados puedan ser almacenados y visualizados mediante cierto tipo de gráficos. Se diseñaron guías para que sea posible desplegar un servidor propio y para preconfigurar la placa ESP (por única vez). Se diseñó un caso de estudio a modo de secuencia didáctica y se puso en práctica con estudiantes de cuarto año de una escuela secundaria. La propuesta fue presentada en el congreso CACIC del año 2024. Y previamente por dos años consecutivos en los Proy. de Innov. y Aplic. con Alumnos en la Facultad de informática, UNLP.

Trabajos Realizados

- *Análisis de tecnologías de conectividad inalámbrica*
- *Análisis y pruebas con placas Arduino, Micro:Bit y ESP32*
- *Análisis de entornos de PBB*
- *Protocolo de comunicación que permite empaquetar datos, para su envío vía WiFi a un servidor REST*
- *Extensión para MakeCode, entorno de PBB para placas Micro:Bit, para lo que se usó JavaScript*
- *Diagramación de una maqueta física para pruebas y personalización de la extensión a dicha maqueta (bloques y config. de ayuda con info. para el uso de pines)*
- *Implementación de un servidor REST usando, Python, MongoDB, y Docker. Visualizador usando Grafana*
- *Guías para despliegue y uso por parte de docentes*
- *Guía de actividades para estudiantes*
- *Prueba de la propuesta integral con estudiantes*
- *Proy. de Innovación y Aplic. 2022-2023. UNLP, Fac. de Inf.*
- *Dos artículos presentados y aceptados en CACIC2024*

Trabajos Futuros

- *Ampliar la variedad de tipos de gráficos disponibles para la visualización de los datos.*
- *Proponer nuevas maquetas de trabajo para que se puedan hacer kits de trabajo*
- *Mejorar las guías en base al feedback recibido durante el caso de estudio*
- *Ampliar las pruebas con docentes y estudiantes*
- *Simplificar la configuración inicial de la placa ESP*
- *Proveer de guías que usen el protocolo de comunicación propuesto con placas Arduino*
- *Explorar el protocolo MQTT y generar una versión del código de la ESP que lo utilice.*
- *Explorar el uso de Raspberry Pi*
- *Analizar aspectos de seguridad en el protocolo bajo la premisa de facilidad para usuarios no expertos*

Fecha de la presentación: Diciembre 2024

Tabla de contenido

Agradecimientos

A mi familia y a mi novia que me apoyaron pacientemente todos estos años y creyeron en mí y a mis directores que me guiaron durante el proceso.

Capítulo 1: Introducción

En este capítulo se exponen las motivaciones, un resumen general de los capítulos siguientes y el objetivo de esta tesina.

1.1 Motivación

Ante un mundo cada vez más tecnológico y considerando que los trabajos del siglo XXI implican nuevas competencias, es que se resalta la importancia de desarrollar pensamiento computacional en los jóvenes que transitan el nivel secundario tanto en áreas de STEAM (siglas en inglés del conjunto de disciplinas de "*science, technology, engineering, arts and mathematics" -ciencias, tecnología, ingeniería, arte y matemáticas*) como en la de Humanidades y Ciencias Sociales. En este trabajo, se considera al pensamiento computacional como una forma de resolver y analizar problemas de manera que sean ejecutadas por una computadora y que, como mencionan Dening y Tedre, involucra habilidades y conceptos como la abstracción, la descomposición, la generalización, los algoritmos (y su diseño) así como, el diseño, la recolección y el análisis de datos [Denning & Tedre, 2021].

El desarrollo del Pensamiento Computacional puede ser abordado de diferentes formas, por ejemplo, mediante la visualización de datos [Özkök, 2021], la programación [Ezeamuzie & Leung, 2022], la robótica educativa [Noh & Lee, 2020], IoT (de la sigla en inglés para *Internet of Things*) [Udvaros et al., 2023] y la Inteligencia Artificial [Yim & Su, 2024].

Una posibilidad para incorporar el desarrollo del Pensamiento Computacional en el nivel secundario es proveer a los docentes y estudiantes de herramientas accesibles y simples de usar y que involucren diferentes habilidades y conceptos de este tipo de pensamiento, así como que consideren diferentes formas de abordaje. En particular, al considerar la programación como forma posible, para sus primeros pasos se suelen usar lenguajes de programación basados en bloques ya que estos bajan la complejidad que

presentan los lenguajes de programación basados en textos y favorecen la adquisición de habilidades del Pensamiento Computacional [Sun et al., 2024].

Cuando se piensa en el desarrollo del Pensamiento Computacional considerando temas de robótica o IoT en el área de educación, en general se hace en términos de kits que ya vienen pre armados y con instrucciones concretas para que los docentes puedan trabajar en el aula usando placas como, por ejemplo, Micro:Bit[1] o Arduino[2]. Las placas Arduino si bien están ampliamente difundidas para diversas actividades educativas, pueden presentar desafíos al momento de su uso tanto para docentes como para estudiantes de escuelas no técnicas principalmente, a quienes le puede resultar complejo el cableado de circuitos que se necesita para trabajar con ellas. Por lo anterior y debido a que ya además traen integrados algunos componentes es que Pech y Novák proponen el uso de placas Micro:Bit para estos contextos educativos [Pech & Novák, 2020].

Por otro lado, al trabajar con sensores y placas en el contexto educativo, puede surgir la posibilidad de sumar conceptos a ser introducidos en la propuesta educativa, como el de conectividad inalámbrica y el de visualización de datos recolectados por los sensores. La conectividad inalámbrica (ZigBee, Bluetooth, Wifi, etc.) a utilizar en la propuesta debe ser analizada en el contexto de la placa que se disponga en el aula (por ejemplo, Aduino o Micro:Bit). Adicionalmente, dado que se espera trabajar con aspectos de programación basada en bloques para estudiantes secundarios (por lo mencionado anteriormente en relación con su facilidad de uso durante los primeros pasos) es importante establecer que entorno de programación se usará en base, principalmente, a la placa que se adopte para la propuesta.

Esta tesina está organizada de la siguiente manera:

En el *Capítulo 2: Marco de trabajo* se presenta el marco de trabajo analizando tecnologías de conectividad inalámbrica, placas usadas en robótica educativa e IOT, así como los lenguajes de programación basado en bloques que pueden usarse para programar dichas placas.

[1] https://microbit.org/
[2] https://www.arduino.cc/

En el *Capítulo 3: Trabajos relacionados* se presentan y analizan trabajos relacionados.

En el *Capítulo 4: Pruebas sobre placas* se presentan pruebas de conectividad y transmisión realizadas usando las placas analizadas en el segundo capítulo y se establece el contexto de trabajo para esta tesina: por un lado, el uso de placa Micro:Bit y ESP8266 y por otro, conectividad WIFI.

El *Capítulo 5: Propuesta de Solución* presenta la propuesta de solución donde por un lado se presenta el protocolo de comunicación y por otro, una extensión para el entorno de programación basado en bloques llamado MakeCode.

El *Capítulo 6: Implementación de componentes* aborda la implementación sobre las placas Micro:Bit y ESP8266, la implementación del servidor REST en Python, la implementación de una herramienta de visualización de datos y la Dockerización de los servicios.

En el *Capítulo 7: Caso de Estudio* se presenta una maqueta de trabajo y una adecuación de la extensión que facilita y personaliza la programación basada en bloques para el caso de estudio, se describen las guías de configuración de la ESP y los servicios, se describe el caso de estudio y presenta una discusión preliminar.

En el *Capítulo 8: Conclusiones y Trabajos Futuros* se presentan conclusiones y trabajos futuros.

Adicionalmente se presentan cinco anexos en donde se presentan:

El *Anexo I: Guía de Configuración de la ESP* se explica paso a paso como cargar el código necesario para la comunicación con la Micro:Bit y el envío de datos vía WiFi en la ESP.

En el *Anexo II: Instructivo para el despliegue de servicios* se detallan los pasos para el despliegue de todos los servicios necesarios para la recepción, almacenamiento y visualización de datos.

El *Anexo III: Guías para las actividades prácticas* incluye las guías utilizadas en las prácticas con alumnos durante el caso de estudio en una escuela secundaria.

El *Anexo IV: Artículo Presentado en III SPA -CACIC 2024* correspondiente al track *Short Papers de Alumnos*

Y el Anexo V: *Artículo Presentado en XXIII WTIAE -CACIC 2024* correspondiente al track *Workshop Tecnología Informática aplicada en Educación*.

Por último, es importante mencionar que el trabajo presentado en esta tesina de grado se enmarca en temas más generales que forman parte de una tesis doctoral [Lliteras, 2020] la cual involucra aspectos de tecnología informática y educación y por ello, como parte de la introducción, se mencionan ciertos aspectos que permiten encuadrar la relevancia de la propuesta de este trabajo. Dejado lo anterior como base, se deja constancia que el tesista hará foco para su trabajo en los aspectos propios de su carrera de grado, quedando fuera de su alcance cualquier consideración propia del área de educación.

1.2 Objetivo

El objetivo de esta tesina es el de proponer una infraestructura que mediante la combinación de software y hardware permita, la recolección, y envío de datos inalámbricamente a un servidor desde el cual se pueda realizar visualización de los datos. Siendo el destinario de esta propuesta estudiantes y docentes de nivel secundario (ciclo superior).

Capítulo 2: Marco de trabajo

En este capítulo se introducen los conceptos que conforman el marco de trabajo de esta tesina, relacionados a la conectividad inalámbrica, a algunas de las placas usadas en educación para robótica educativa y para IoT, y en relación con los entornos de programación de las placas descriptas.

2.1 Conectividad inalámbrica

En esta sección se analizan algunas de las tecnologías inalámbricas más difundidas actualmente, ponderando sus pros y contras, teniendo en cuentas tanto consideraciones generales como su adaptabilidad a las necesidades del proyecto de esta tesina. El análisis antes mencionado se da en el ámbito de lo tecnológico, de la documentación técnica y lo económico.

En la actualidad existe una gran variedad de tecnologías de conectividad inalámbrica factibles de ser utilizadas por este proyecto [Srinivasulu et al., 2020]. Este estudio se centra en aquellas que son implementadas mediante radiofrecuencia, tienen mediano o largo alcance y son asequibles en Argentina al momento de la realización de este. Siendo ZigBee, LoRa, Bluetooth y WiFi las que cumplen con las características antes mencionadas. Sin embargo, para este proyecto también se tendrán que considerar otros aspectos tales como la facilidad de acceso a la documentación para el desarrollo sobre dichas tecnologías y la factibilidad de interoperar con otros dispositivos tales como celulares o computadoras.

2.1.1 ZigBee

ZigBee es un protocolo de comunicación inalámbrico de baja potencia y corto alcance utilizado principalmente en aplicaciones IoT.

Su tecnología de transmisión de baja potencia permite un muy bajo consumo y una duración de batería prolongada. Debido a esa misma baja potencia de transmisión su alcance está limitado a unos 10 a 15mts, lo que circunscribe su uso redes de área personal (WPAN).

Una característica de ZigBee es su capacidad para funcionar bajo una topología de red en malla, lo que significa que cada dispositivo puede actuar como un nodo y transmitir señales a otros dispositivos ZigBee, aumentando la cobertura y la fiabilidad de la red y evitando la necesidad de utilizar un hub central.

Su velocidad de transmisión es relativamente baja (250 kbps) y utiliza cifrado AES-128 como medida de seguridad.

2.1.2. LoRa

LoRa (Long Range) es un estándar de comunicación inalámbrica de larga distancia y baja potencia diseñado para aplicaciones IoT. Tiene un alcance de entre 10km en áreas urbanas y 20km en áreas rurales, dependiendo de las condiciones, sin necesidad de repetidores. Consume muy poca energía, lo que lo hace ideal para dispositivos que funcionan con baterías y que se encuentran en regiones remotas o de difícil acceso, como estaciones meteorológicas o balizas.

Debido a su topología en estrella necesita un hub central para operar. Su velocidad de transmisión es relativamente baja, aproximadamente unos 255bytes/s.

Utiliza encriptación de extremo a extremo mediante un cifrado con AES-128 para garantizar la seguridad de la red.

2.1.3. Bluetooth

La tecnología Bluetooth es un estándar de comunicación inalámbrica que permite la transmisión de datos entre dispositivos cercanos. Fue lanzado en el año 1998 por el

Bluetooth Special Interest Group, formado originalmente por Ericsson, Intel, Nokia, IBM y Toshiba. Más tarde se sumarian al Bluetooth SIG Microsoft, Lenovo y Apple. Esta tecnología funciona en la banda de 2.4 GHz, al igual que otras tecnologías inalámbricas, es una banda que no necesita licencia y puede usarse libremente a nivel mundial. Su consumo de energía es muy bajo y su tasa de transferencia ronda los 3 Mbit/s.

Su principal uso es el envío de audio entre dispositivos como auriculares, dispositivos móviles, estéreos de autos y computadoras. También se utiliza para el envío de datos desde dispositivos médicos como medidores de ritmo cardiaco o glucosa, smartwatches y dispositivos weareables en general.

Bluetooth utiliza esquemas de cifrado para proteger la confidencialidad de los datos transmitidos. Hasta la especificación Bluetooth 4.2, se utilizaba principalmente el algoritmo de cifrado simétrico de flujo denominado E0, actualmente considerado vulnerable. Sin embargo, a partir de Bluetooth 5.0, se recomienda el uso del algoritmo de cifrado simétrico AES-CCM (Advanced Encryption Standard-Counter with Cipher Block Chaining Message Authentication Code). AES-CCM proporciona una mayor seguridad y es ampliamente utilizado en otras aplicaciones criptográficas.

El protocolo Bluetooth ofrece diferentes modos de seguridad que pueden configurarse durante el emparejamiento. Estos modos incluyen "Ninguno" (sin seguridad), "Nivel de servicio autenticado" (requiere autenticación) y "Nivel de servicio cifrado" (requiere cifrado). Los dispositivos pueden negociar el nivel de seguridad adecuado durante el proceso de emparejamiento.

Junto con Wifi es la tecnología inalámbrica con mayor base instalada. La Bluetooth SIG estima que la cantidad de dispositivos fabricados con bluetooth creció un 12% anual durante los últimos 10 años, llegado a fabricarse cinco mil doscientos millones de dispositivos en el año 2022[3].

[3] https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_Market_Update_2018.pdf

2.1.4. WiFi

Es una tecnología de conexión inalámbrica definida en el estándar IEEE 802.11[4] en el año 1997. Se encuentra madura, en constante evolución[5] y respaldada por el organismo de estándares más importante del mundo en lo que respecta a tecnología, la IEEE. Debido a todo esto es utilizada por una gran variedad de dispositivos y cuenta con una gran base instalada.

En la *Figura 1* se puede ver un resumen de su historia, desde sus antecedentes a mediados de los 80 hasta proyecciones para el 2023.

Como se aprecia en

Tabla *1* el estándar IEEE 802.11 fue evolucionando a lo largo del tiempo en sus sucesivas versiones, agregando más frecuencias y canales de mayor amplitud y por sobre todas las cosas mejorando notablemente el ancho de banda que fue de 2Mbps en su primera versión a 2.4Gbps en la última.

Aunque quede fuera del alcance de este estudio cabe destacar que las actualizaciones también trajeron mejoras en la seguridad, alcance y fiabilidad de las conexiones.

Tabla 1: Comparativa revisiones WiFi [Fuente: propia]

Protocolo	Frecuencia	Ancho del Canal	Ancho de banda teórico
802.11ax	2.4 / 5GHz	20, 40, 80, 160MHz	2.4 Gbps
802.11ac wave2	5 GHz	20, 40, 80, 160MHz	1.73 Gbps
802.11ac wave1	5 GHz	20, 40, 80MHz	866.7 Mbps
802.11n	2.4 / 5 GHz	20, 40MHz	450 Mbps

[4] http://standards.ieee.org/getieee802/802.11.html
[5] https://www.intel.com/content/www/us/en/support/articles/000005725/wireless/legacy-intel-wireless-products.html

802.11g	2.4 GHz	20 MHz	54 Mbps
802.11a	5 GHz	20 MHz	54 Mbps
802.11b	2.4 GHz	20 MHz	11 Mbps
Legacy 802.11	2.4 GHz	20 MHz	2 Mbps

Figura 1: Historia de Wifi

[Fuente: http://wififorward.org/news/timeline-the-wi-fi-success-story/]

Actualmente la presencia de alguna variante de las tecnologías WiFi mencionadas previamente resulta ubicua, desde dispositivos móviles como laptops y celulares hasta electrodomésticos, lámparas y vehículos, entre otros. Siendo 802.11n y 802.11ac las más utilizadas. Versiones anteriores como 802.11g o 802.11b fueron quedando en desuso debido a sus limitaciones mientras que los dispositivos con versiones más nuevas como 802.11ax todavía no constituyen un volumen crítico como para considerarlos relevantes.

Una de las claves de su éxito se encuentra en la retro compatibilidad de los dispositivos de hardware que permiten la interoperabilidad entre distintas versiones del protocolo WiFi evitando tener que contar un dispositivo por cada revisión.

Debido a ser un estándar libre, a su madurez tecnológica y a su omnipresencia se puede encontrar mucha documentación, ejemplos, implementaciones y librerías que utilizan WiFi.

2.1.5 Algunas reflexiones acerca de las tecnologías analizadas

Luego de haber presentado las tecnologías, se puede destacar que en el caso de elegir LoRa o ZigBee se tendría la ventaja del bajo consumo energético por cualquiera de las dos tecnologías. Por otro lado, al no ser tecnologías de uso masivo no están incluidas en aparatos de uso habitual como computadoras o celulares, por lo cual se deberían adquirir dispositivos tanto para la transmisión como para la recepción de los datos. Esto las hace inviables para el proyecto.

Por otro lado, Bluetooth y WiFi son dos tecnologías ampliamente difundidas con una gran base instalada. Presentes en la mayoría de los dispositivos electrónicos de uso cotidiano, que permitirían la conexión del dispositivo de sensado de datos sin la necesidad de hardware extra en el lado de la recepción. A pesar de las similitudes se decide optar por la utilización de WiFi debido a que es un standard abierto, posee mejor acceso a la documentación y mayor cantidad de proyectos de código abierto.

2.2 Comparativa de Placas

En esta sección se analizan las principales características de las placas candidatas a ser la base de hardware del sistema de recolección de datos inalámbricos IRID, ellas son Arduino y Micro:Bit.

Si bien a priori eran las únicas dos placas por analizar y comparar, posterior a realizar pruebas con las mismas (Capítulo 4) se detecta la necesidad de trabajar con la placa ESP32. Por este motivo, se suma esta placa en el análisis de esta sección.

2.2.1 Características principales de las placas Arduino, Micro:Bit y ESP32

Considerando las placas Arduino, Micro:Bit y ESP32, en la *Tabla* 2 se muestran diferentes aspectos placas analizadas de las mismas, a modo comparativo. En la columna “modelo” se listan los 3 modelos de placas analizadas, la columna “incluye WiFi” indica si la placa posee conectividad WiFi, las columnas “facilidad para programarlo”, “facilidad de conexión de sensores” y “poder de cómputo” puntúan a las placas del 1 al 5 teniendo en cuenta aspectos técnicos de los procesadores de las placas y por último tenemos el precio en pesos argentinos y en dólares. Esta tabla permite ver de manera concisa y breve las diferencias de las principales características que se tuvieron en cuenta al momento de estudiar las placas.

Tabla 2: Comparativa de placas [Fuente: estudio propio]

Modelo	Incluye Wifi	Facilidad para programarlo	Facilidad de Coneccion de Sensores	Poder de Computo	Precio en pesos*	Precio en dolares**
Micro:Bit V2	No	5/5	4/5	3/5	$65,000.00	$50.00
Arduino Mega Rev3 (Compatible)	No	4/5	5/5	2/5	$37,000.00	$20.00
ESP 32	Si	3/5	1/5	4/5	$12,000.00	$10.00
*Precios promedio con envio en pesos Argentinos al 28/11/2024, fuente MercadoLibre						
**Precios promedio con envio en dolares al 28/11/2024, fuente Ebay						

A continuación, se presenta un análisis de algunas de las columnas presentadas en la Tabla 3.

En la columna “incluye WiFi” se puede observar que de las 3 placas elegidas solo una posee conectividad WiFi. La falta de conectividad de la Micro:Bit y el Arduino no representa un problema técnico ya que se les pueden agregar módulos WiFi para solucionar esta carencia. Sin embargo, esto traería algunos problemas aparejados tales como: aumento del costo del hardware, incremento de la complejidad de las conexiones electrónicas, incremento de la complejidad del software al tener que agregar las librerías específicas de los módulos y disminución de los pines disponibles para la conexión de sensores.

En la columna “Facilidad para programarlo” se tuvo en cuenta que tan fácil es programar la placa para una persona con pocos o nulos conocimientos de informática. La mejor puntuada fue la Micro:Bit ya que posee un entorno[6] oficial de programación en bloques listo para utilizar desde un navegador web. En segundo lugar, se encuentra Arduino que posee un entorno oficial de programación, pero no soporta programación en bloques de manera nativa. En el último lugar tenemos la ESP que no posee un entorno de programación oficial, pero puede ser programada mediante el IDE de Arduino previa instalación de plugins. Cabe aclarar que el hardware ESP actualmente en venta es bastante heterogéneo y en algunos casos se vende sin firmware, en esos casos el puntaje de 3/5 bajaría a 1/5 debido a la dificultad generada por tener que grabarle el firmware correspondiente.

6 https://makecode.microbit.org/

Existen alternativas fuera de los entornos oficiales para programar Arduino mediante programación en bloques, cada una con distintos enfoques y características. ArduBlock se agrega como pluggin al IDE de Arduino y agrega la capacidad de programación en bloques dentro del mismo. BlockDuino y Mblock son entornos de programación en bloques que soportan Arduino y otras placas, ambos entornos poseen versiones online y offline. Mblock también soporta ESP. Los entornos mencionados se analizan en profundidad en la siguiente sección de "2.3 Comparativa de entornos de programación".

En la columna "Facilidad de conexión de sensores" se tuvo en cuenta si se pueden conectar sensores a la placa sin ningún adaptador y cuan confiable es esa conexión. La mejor puntuada en esta categoría es Arduino, sus conectores Dupont garantizan una conexión firme con los sensores y hacen que conectar y desconectar los mismos sea muy sencillo y seguro. En segundo lugar se encuentra la Micro:Bit con sus conectores pensado para ser utilizados con pinzas cocodrilo que hacen que la conexión sea fácil pero más frágil y propensa a falsos contactos que la del Arduino. Si se quiere generar conexiones más seguras existen adaptadores del conector tipo cartucho de la Micro:Bit a conectores Dupont machos pero no se incluyen con la placa. En tercer lugar, encontramos a la ESP la cual requiere que se le suelden conectores antes de poder utilizarse con módulos externos.

2.2.2 Posibilidades de expansión

Las tres placas mencionadas anteriormente soportan una amplia variedad de sensores que agregan todo tipo de funcionalidades.

Arduino posee placas de expansión o shields, ejemplificados en la *Figura 2*, que se conectan directamente a las placas Arduino para proporcionar funcionalidades adicionales y ampliar sus capacidades. Estos shields están diseñados para encajar fácilmente en los pines del Arduino, brindando una solución plug-and-play, sin la necesidad de realizar conexiones complicadas o soldaduras adicionales.

Figura 2: Ejemplo de Arduino Shield – Placa Ethernet

[Fuente: https://www.mouser.com/ProductDetail/Arduino/A000024?qs=tbM5a8K56PWYRqjqIIHKUA%3D%3D]

A pesar de ser fáciles de usar, su compatibilidad se ve reducida debido a que dependen de la disposición de los pines según el modelo de Arduino que se decida usar. Esta incompatibilidad se puede saltear utilizando cables para conectar los pines del shield a los pines que corresponda del Arduino, pero se pierda la facilidad de uso que los caracteriza.

Por otro lado, se venden sensores sueltos (*Figura 3)* para cualquiera de las tres placas mencionadas. Estos sensores son un poco más complejos de utilizar debido a que se deben conectar mediante cables, lo que puede llevar a conexiones defectuosas y lecturas erróneas.

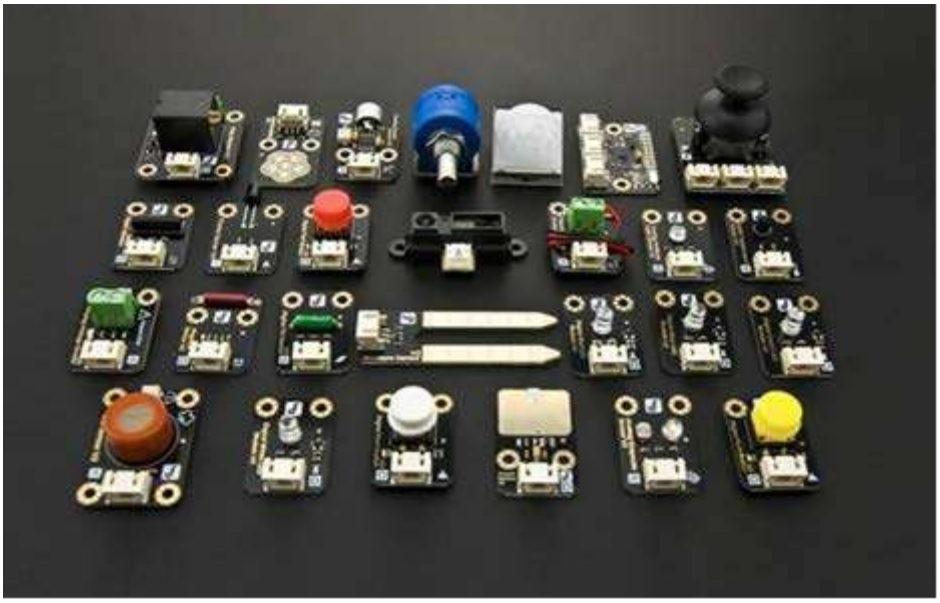


Figura 3: Sensores varios

[Fuente: https://www.dfrobot.com/product-725.html]

La principal diferencia entre los sensores destinados a las distintas placas son los circuitos adicionales que la mayoría de las veces se comprenden de alguna resistencia y algún capacitor para compatibilizar las señales del sensor y las de la placa. La mayoría de los sensores funcionan a 5v o 3.3v y pueden utilizarse con cualquiera de las placas agregando una fuente externa de energía o alguna resistencia para limitar el flujo de corriente.

2.2.3. Soporte

Tanto Arduino como Micro:Bit poseen documentación oficial[6y7] para guiar a los usuarios en el desarrollo de proyectos utilizando dichas placas. Esta documentación es de acceso libre y gratuito, incluye ejemplos de proyectos de varios niveles de complejidad

[7] https://www.arduino.cc/

en los que se detalla tanto el código a implementar en las placas como la forma de realizar las conexiones de estas con los módulos necesarios para cada uno de los proyectos.

Otra fuente muy importante de información son los foros[8y9] de Arduino y Micro:Bit. En estos espacios los usuarios comparten sus experiencias, hacen preguntas, documentan problemas comunes y se ayudan mutuamente.

Ambas fuentes de información se complementan para formar un extenso catálogo de documentos que abarcan desde la implementación de ejemplos básicos hasta la resolución de problemas específicos de compatibilidad de librerías, configuración de módulos o configuración de los entornos de programación en distintos SO, por nombrar algunos.

En cuanto a la ESP no existe una documentación oficial tan amigable como las de Arduino o Micro:Bit, pero si posee un foro[10] oficial donde se puede encontrar información similar a la de los foros mencionados anteriormente.

2.2.4 Algunas reflexiones acerca de las placas analizadas

Una vez presentado el análisis de las placas Arduino, Micro:Bit y ESP, en el presente trabajo se adopta el uso de la placa Micro:Bit por considerarse de mayor facilidad de uso (por sobre el costo) y de la placa ESP para trabar con la conectividad a WiFi.

2.3 Comparativa de entornos de programación

En esta sección se compararán los distintos entornos de programación compatibles con las placas propuestas. Si bien existen una gran cantidad de entornos y formas de programar las placas, en esta sección se hace foco en las más populares, los entornos oficiales y los que dispongan de herramientas que faciliten su uso por personas no vinculadas a las ciencias de la computación.

[8] https://forum.makecode.com/c/microbit
[9] https://forum.arduino.cc/
[10] https://www.esp32.com/viewforum.php?f=23

2.3.1 Criterios para la elección de un entorno de programación

Una de las características que se considerará con más énfasis para la selección de un entorno de programación, en este trabajo de tesina, es la programación basada en bloques. La programación basada en bloques consiste en que se puedan construir algoritmos uniendo bloques gráficos como si fueran piezas de un rompecabezas. Los comandos y los tipos de datos están representados por bloques de diferentes formas, con piezas que encajan entre sí sólo en formas sintácticamente correctas. Este enfoque elimina los errores de sintaxis (que han demostrado ser un gran obstáculo para aprender lenguajes de programación basados en texto), lo que permite a los jóvenes concentrarse en los problemas que quieren resolver, y no en la sintaxis específica de la programación.

Otra característica imprescindible del entorno de programación basado en bloques es el nivel de interoperabilidad con la placa Arduino o Micro:Bit. El escenario ideal es que el entorno soporte la placa nativamente, es decir, que no exista la necesidad de instalar librerías externas, drivers, ni cargar programas previamente en la placa para poder utilizarla junto con el entorno. En este escenario la utilización del entorno junto con la placa se reduce a escribir el programa en dicho entorno, conectar la placa vía usb y subir el código a la placa desde el entorno.

Otros criterios para tener en cuenta son la posibilidad de ejecutar el entorno sin la necesidad de una conexión constante a internet, que disponga de una versión en español, que sea gratuito, open source y soporte la mayor cantidad de sistemas operativos posibles.

2.3.2 Entornos para Micro:Bit

A continuación se presentan entornos de programación para las placas Micro:Bit.

2.3.2.1 MakeCode

Mediante el entorno de programación MakeCode se puede programar la placa en JavaScript, Python y Programación en Bloques, pudiendo intercambiar de lenguaje "en vivo" y traduciendo automáticamente el código. También provee librerías y funciones para sacar máximo provecho de todos los componentes de la placa, un amplio catálogo de ejemplos y tutoriales, la posibilidad de compartir creaciones propias con facilidad, subir el código generado a la placa y un emulador interactivo que muestra en tiempo real los resultados del código en una simulación de la placa. Se encuentra disponible en varios idiomas, incluido el español. En la *Figura 4* se puede apreciar la pantalla de inicio donde se encuentran los tutoriales mencionados.

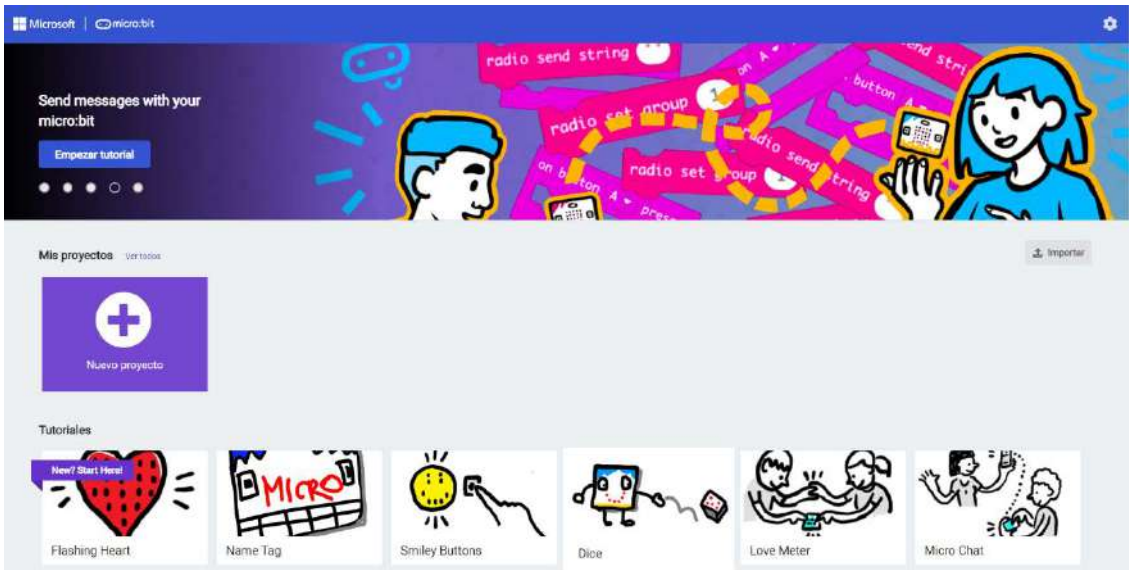


Figura 4: Pantalla de inicio de MakeCode

[Fuente: fuente propia]

En la *Figura 5* se puede observar el entorno de programación MakeCode en modo de programación en bloques. En la barra superior podemos ver de izquierda a derecha el nombre de la aplicación, el selector de modo de programación en bloques, el selector desplegable de lenguajes de programación textuales e iconos de inicio, compartir, ayuda y configuración. Al centro a la izquierda se encuentra el emulador interactivo de la placa

Micro:Bit, en él se pueden visualizar los efectos que tendrá el código sobre los componentes de hardware antes se subirlo a la placa. También se puede interactuar con los componentes de la Micro:Bit de manera virtual, simular que se presiona un botón o activar el sensor de agitación, entre otros. Luego se encuentra una columna con los grupos de bloques disponibles con un buscador en su parte superior. Por último, se encuentra el área de programación donde se pueden arrastrar los bloques para crear programas.

Continuando con la descripción de la Figura 5, en la parte inferior también de izquierda a derecha se encuentra el botón de Descargar que permite descargar el código a un archivo o directamente a la placa, un campo de entrada que permite cambiar el nombre del proyecto junto a un botón de guardado rápido y por último botones de deshacer y rehacer y controles para aumentar o disminuir el tamaño de los bloques.

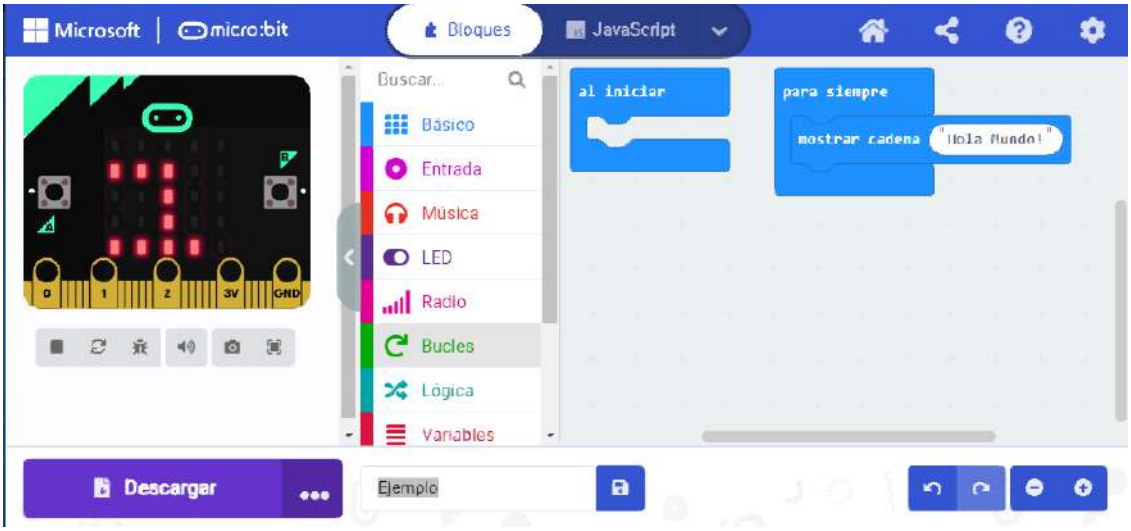


Figura 5: Entorno de programación en bloques de MakeCode

[Fuente: fuente propia]

En la *Figura 6* se puede apreciar la misma instancia de MakeCode vista en la Figura 5 pero en modo JavaScript. En este modo el código generado mediante bloques se convierte automáticamente en código JavaScript que puede ser editado dentro de la misma área de programación mencionada en el párrafo anterior.

Otra característica de este modo es el navegador los archivos de código que se encuentra debajo del emulador de la placa. Este navegador de archivos permite inspeccionar la implementación de todos los archivos de código dentro del proyecto y editar algunos de ellos.

El modo Python es exactamente igual al modo JavaScript pero con todo el código transformado a Python.

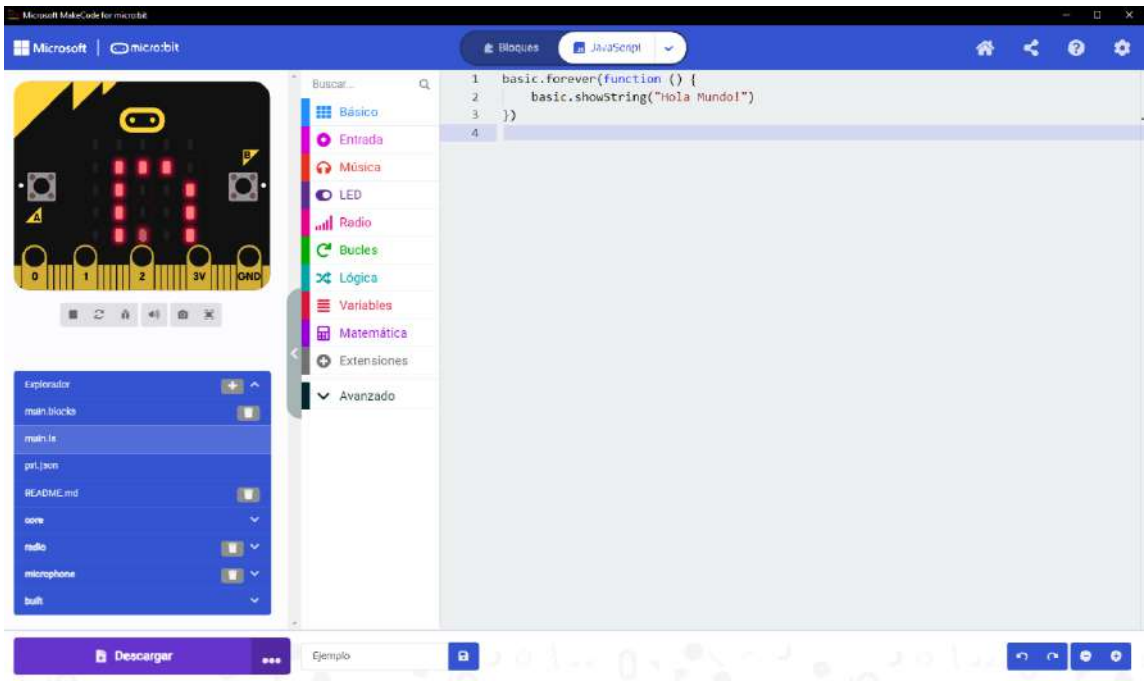


Figura 6: Entorno de programación MakeCode en modo JavaScript

[Fuente: fuente propia]

El entorno posee una versión online e instaladores para entornos Windows y Mac. Tanto la versión online[11] como las versiones instalables, o versiones offline, poseen las mismas funcionalidades que se mencionaron en el primer párrafo, aunque algunas de ellas como los ejemplos o los tutoriales con video y proyectos con dependencias de librerías externas requieren una conexión a internet.

MakeCode también brinda la posibilidad de extender los bloques por defecto con nuevos bloques generados por el usuario. Los nuevos bloques se crean mediante código

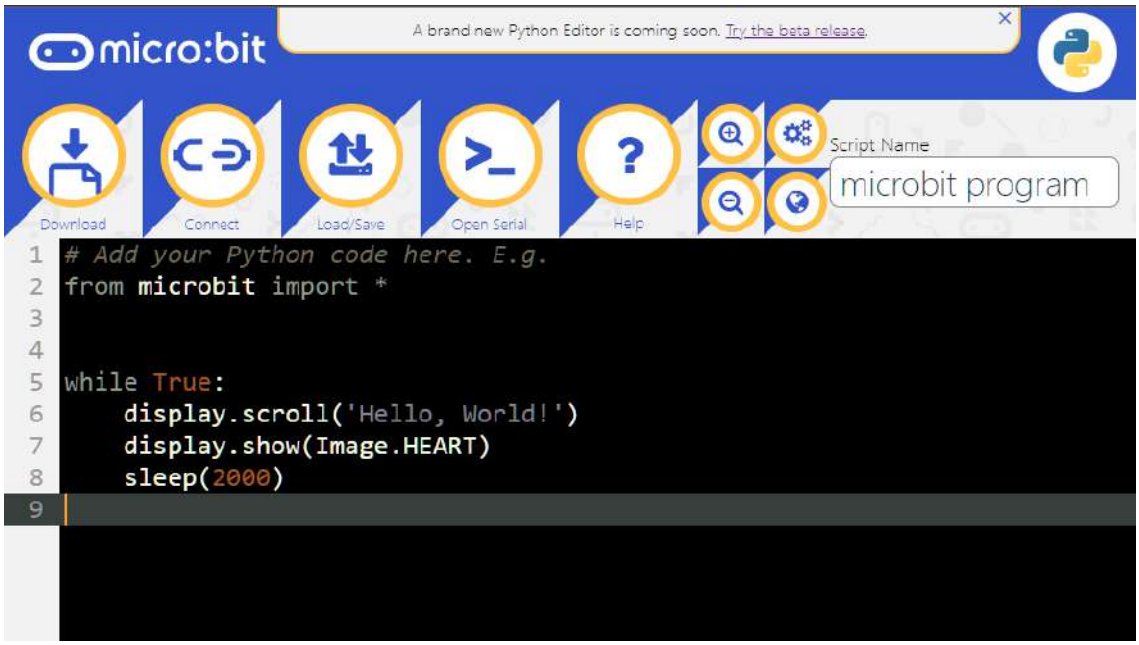
[11] https://makecode.microbit.org/

JavaScript, no pudiendo generar bloques nuevos mediante código Python o basándose en bloques prexistentes en el modo de programación en bloques. Esta es una característica avanzada por lo cual no se encuentran referencias directas a su existencia dentro de MakeCode, para utilizarla se debe investigar la documentación online[12] del programa.

2.3.2.2 MicroPython

MicroPython es un entorno de programación online oficial de Micro:Bit. No posee versiones offline pero una vez cargado en el navegador puede ser utilizado sin necesidad de una conexión a internet[13].

Este entorno posee una versión Legacy (



) que provee funcionalidades básicas como un editor para escribir código en Python, comunicación vía serie con la placa, algunos fragmentos de código de ejemplo y soporte para grabar el programa en la Micro:Bit. A pesar de desincentivarse su uso, se hace mención a esta versión porque al momento de iniciar la investigación para este trabajo era la única versión estable.

[12] https://makecode.com/defining-blocks
[13] https://microbit.org/new-microbit-python-editor/

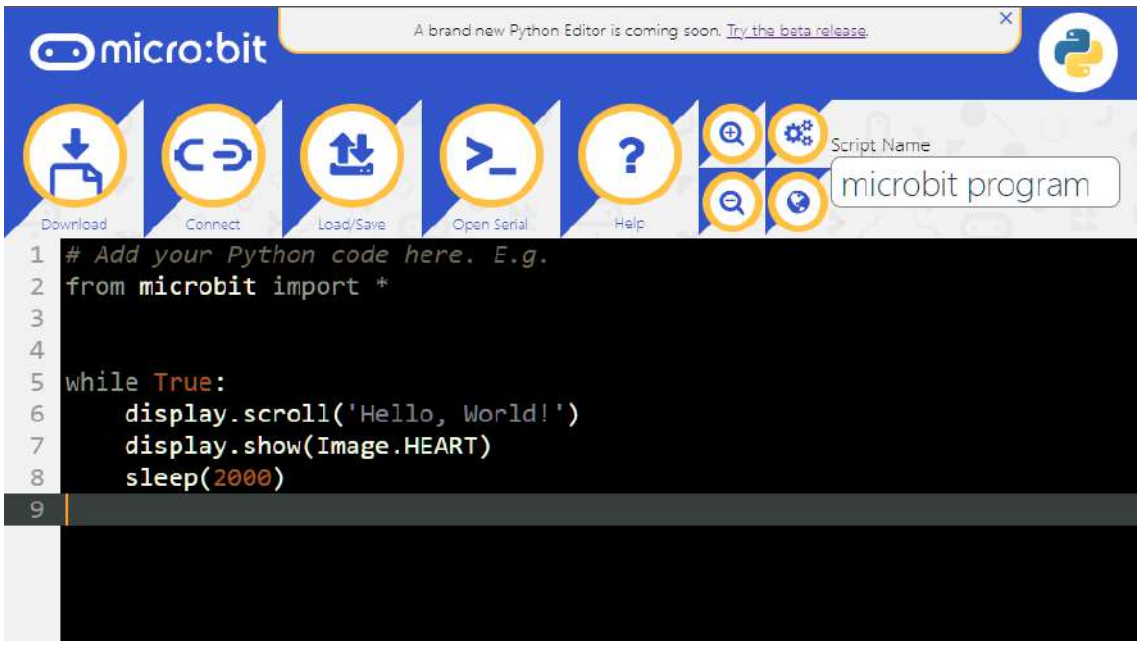


Figura 7: MicroPython V2 - Legacy

[Fuente: fuente propia]

La versión actual de MicroPython (V3), mostrada en la *Figura 8*, implementa una interfaz más amigable muy similar a la vista en MakeCode.

A la izquierda de la Figura 8, se visualiza una columna con una guía de referencia que explica el funcionamiento de la placa mediante gráficos y ejemplos de código, una lista de proyectos de ejemplo bajo la pestaña de Ideas y una pestaña con la documentación completa de las funciones de la API para controlar los componentes de la Micro:Bit. En el medio se encuentra el editor donde se puede escribir código Python a mano o arrastrar desde la pestaña de API o Referencia y debajo de él se encuentran botones para enviar el código a la placa, guardarlo o abrir un proyecto. A la derecha se encuentra un emulador interactivo de la placa muy parecido al de MakeCode que muestra los efectos que tendrá el código sobre la placa y permite interactuar con la misma mediante el grafico del emulador y mediante controles específicos para cada uno de los componentes ubicados en la parte inferior. En esta misma columna de la derecha también se encuentra un emulador de la consola serie que nos permite enviar y recibir mensajes hacia y desde la placa.

Ambas versiones se encuentran disponible en varios idiomas, incluido el español.

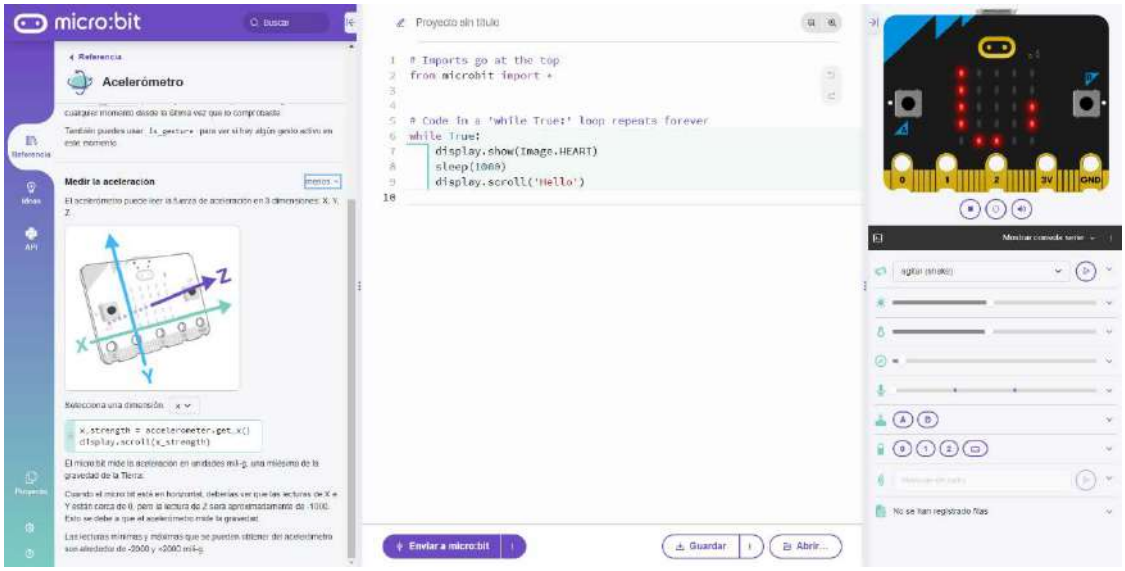


Figura 8: MicroPython V3

[Fuente: fuente propia]

También puede programarse con una gran variedad de *2.3.4 Entornos no* Oficiales, tanto online como offline, o compilar los programas en formato UF2 y correrlos en la placa simplemente copiando el archivo resultante vía USB.

2.3.3 Entornos para Arduino

A continuación, se presentan algunos entornos de programación para la placa Arduino.

2.3.3.1 Arduino IDE

El Arduino IDE (sigla del inglés, *Integrated Development Environment)* es un entorno de desarrollo integrado diseñado para programar y cargar código en placas

Arduino. Fue desarrollado con la idea de facilitar la programación de la placa, por lo que posee una interfaz gráfica sencilla y fácil de usar, accesible para usuarios poco experimentados. Se encuentra disponible en varios idiomas, incluido el español.

Posee una versión online[14], para la cual es necesario registrarse (Figura 9), y dos versiones offline, Arduino 1.X (Legacy) (Figura 10) y Arduino 2.X (Figura 11) , con instaladores para Windows, Mac y Linux.

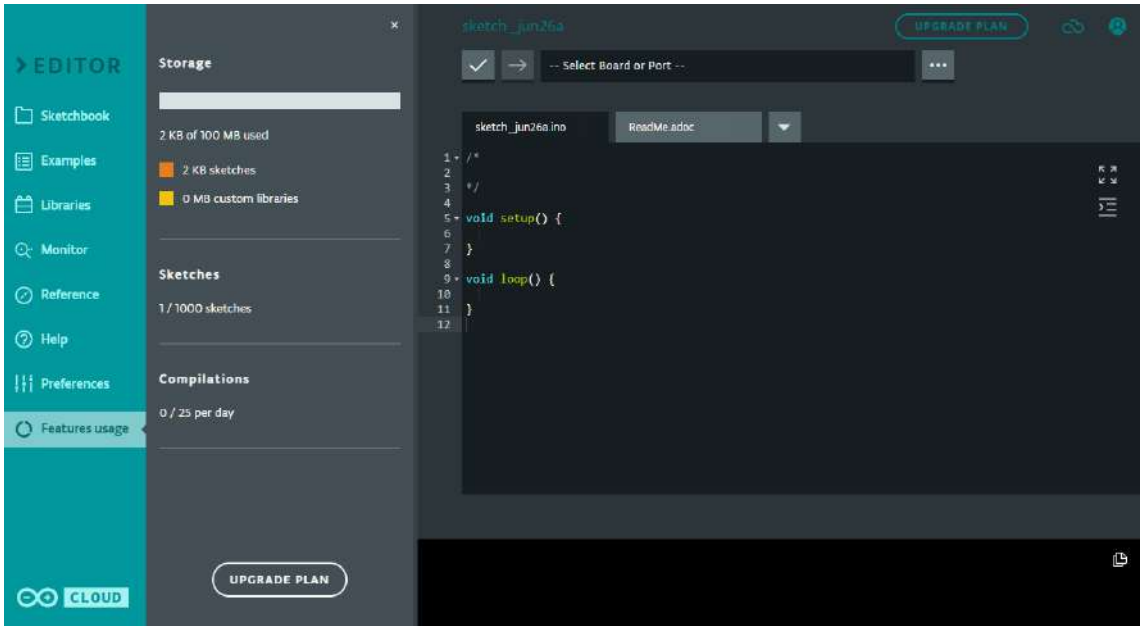


Figura 9: Arduino Online

[Fuente: fuente propia]

Entre sus componentes más destacados, comunes a las tres versiones, se encuentran el editor de código, el monitor serie, la biblioteca de funciones, el compilador y el cargador de código.

El editor de código es un editor de texto que permite a los usuarios escribir, editar y guardar su código. Su utilización es bastante sencilla, admite la organización de archivos mediante pestañas y en la versión Legacy carece de funciones comúnmente encontradas en otros editores como autocompletado inteligente o análisis del código.

[14] https://create.arduino.cc/editor/

El monitor serie permite la vista de datos y comunicación en tiempo real con la placa, facilitando la depuración y la comunicación con la misma.

Incluye una amplia biblioteca que permite a los programadores acceder a funciones comunes sin tener que escribir el código desde cero, agilizando el proceso de desarrollo y minimizando la complejidad de la utilización de los módulos de hardware.

El compilador se encarga de compilar el código escrito en lenguaje de programación Arduino (basado en C y C++) en un lenguaje de máquina que la placa Arduino pueda entender.

El cargador de código ofrece una forma sencilla de cargar el código compilado en la placa Arduino a través de una conexión USB para posteriormente ser ejecutado por la misma.

Como se ve en la *Figura 9* la versión online del IDE de Arduino tiene algunas limitaciones en cuanto a la cantidad de proyectos que se pueden generar, llamados sketches, y a la cantidad de compilaciones por día que se pueden ejecutar. Para sortear estas limitaciones se ofrecen distintos tipos de planes[15] que van desde los dos dólares hasta los veinte dólares por mes.

[15] https://cloud.arduino.cc/plans/

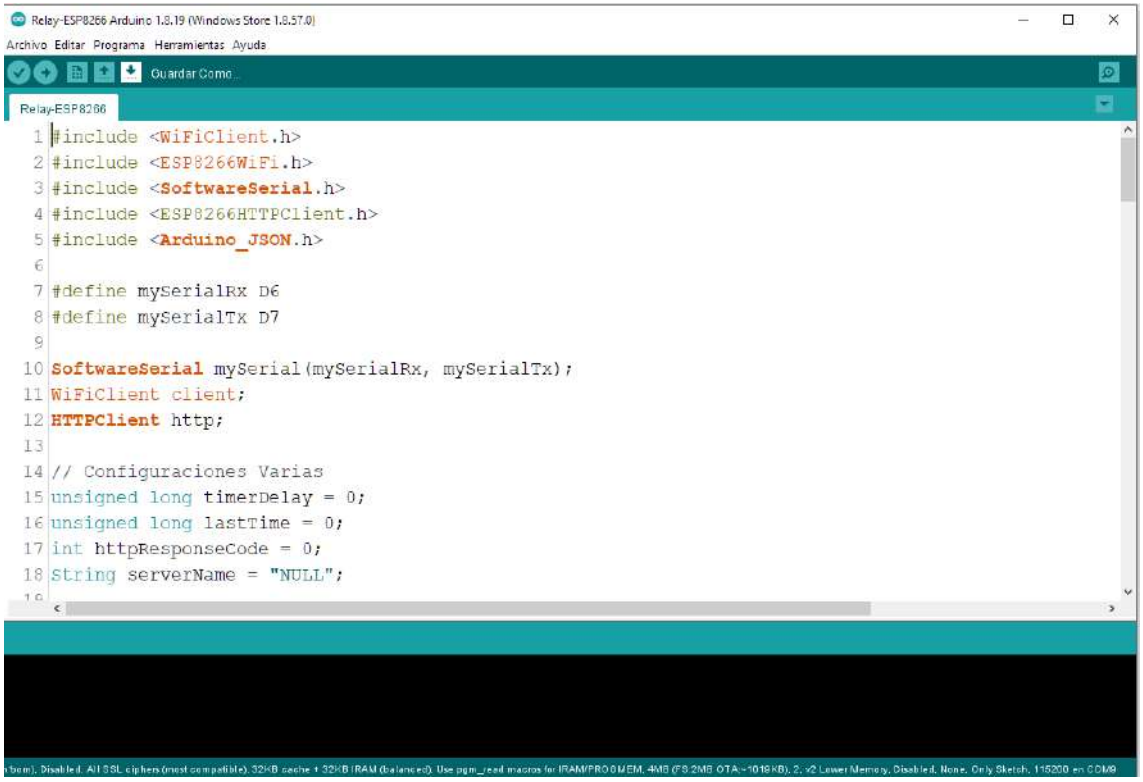


Figura 10: Arduino 1.X (Legacy)

[Fuente: fuente propia]

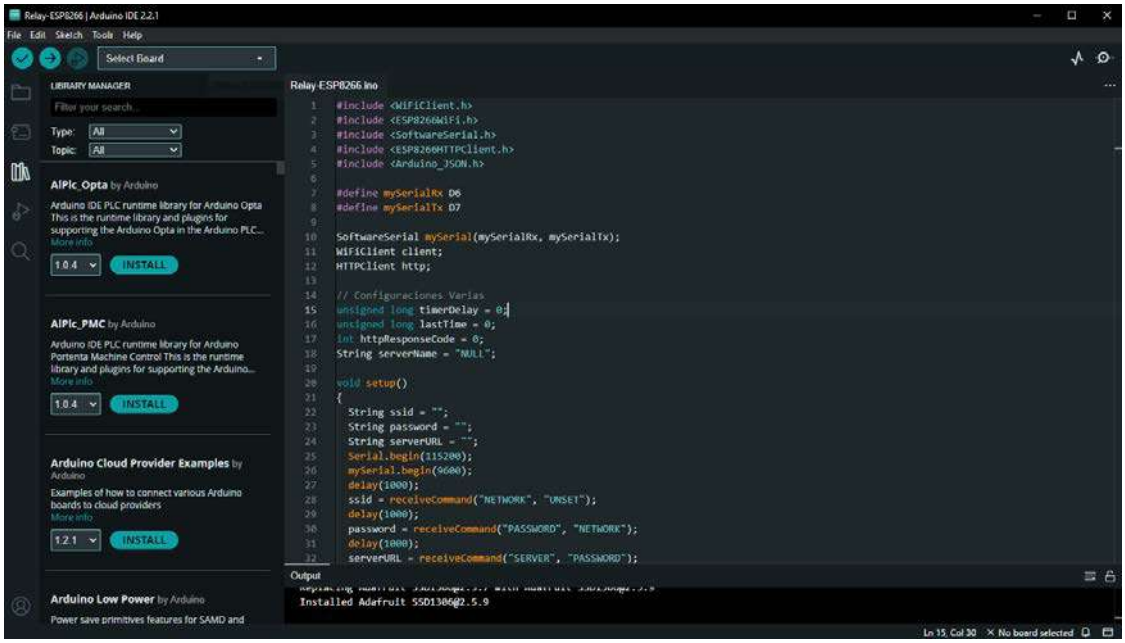


Figura 11: Arduino 2.X

[Fuente: fuente propia]

Las versiones offline, mostradas en la *Figura 10* y en la *Figura 11*, no poseen ningún tipo de limitación en cuanto a su uso, son gratuitas y open source[16]. Ambas versiones poseen las funcionalidades mencionadas, pero en la versión 2.X se agregan autocompletado y autoformato de código. El resto de las diferencias son meramente estéticas, siendo la más destacable la barra de la izquierda (Figura 11) que permite acceder a funciones que en la versión 1.X están dentro de submenús.

2.3.4 Entornos no Oficiales

A continuación, se describen algunos entornos no oficiales de las placas que sin embargo proveen la posibilidad de programación para las mismas.

2.3.4.1 Scratch

Scratch es un entorno de programación en bloques diseñado por el MIT para personas entre los 8 y 16 años, pero su uso actual comprende un rango más amplio de edades. Es utilizado en una gran variedad de entornos, incluyendo hogares, escuelas, museos, bibliotecas y centros comunitarios. Se encuentra disponible en varios idiomas, incluido el español[17]. Scratch permite crear historias, juegos y animaciones interactivas, y compartir creaciones con otras personas en todo el mundo. A medida que se crean y comparten proyectos Scratch, se aprende a pensar de manera creativa, razonar sistemáticamente y trabajar colaborativamente[18]. Aparte del entorno en sí mismo posee una gran variedad de tutoriales y proyectos de ejemplo, y recursos para estudiantes y educadores[19], como se muestra en la *Figura 12*.

[16] https://github.com/arduino/
[17] https://scratch.mit.edu/about
[18] https://scratch.mit.edu/faq
[19] https://scratch.mit.edu/educators

Figura 12: Pagina de recursos de Scratch

[Fuente: fuente propia]

En la *Figura 13* se puede ver la interfaz de Scratch que posee una estructura similar a los entornos de programación en bloques MakeCode y MicroPython. De izquierda a derecha se puede encontrar los bloques disponibles para crear los programas, el editor donde se arrastran los bloques en el medio y a la derecha un emulador para previsualizar el resultado del programa. A diferencia de los dos entornos mencionados, que están diseñados para interactuar con hardware, Scratch hace énfasis en la animación y la creación de juegos proveyendo bloques que facilitan estas tareas.

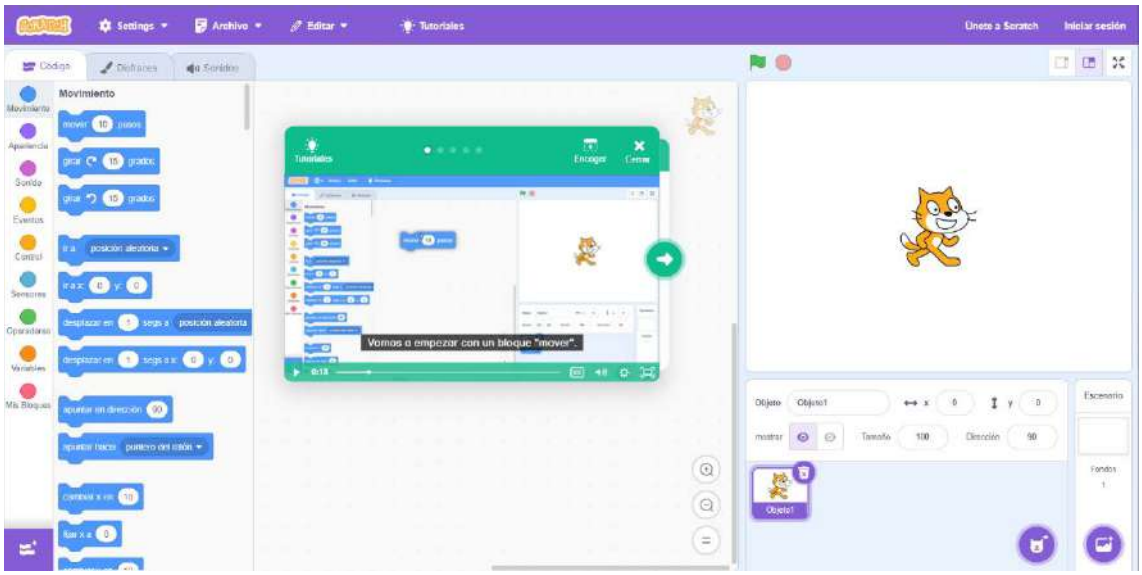


Figura 13: Scratch

[Fuente: fuente propia]

El entorno posee versiones online[20] e instaladores[21] para Windows, Mac y Linux.

Scratch no soporta de manera nativa las placas Arduino ni Micro:Bit pero existen proyectos que intentan integrar estas placas dentro del mismo.

2.3.4.2 S4A

Para utilizar Arduino con Scratch existe S4A[22] (Scratch for Arduino) que provee bloques específicos para manejar sensores y actuadores conectados a la placa. El proyecto fue desarrollado en el Citilab[23] por el equipo de investigación Edutech, Smalltalk.cat[24] e integrantes de la Universitat Politècnica de Catalunya y la Universitat Autònoma de Barcelona. Solamente se puede usar de modo offline y posee instaladores para Windows, Mac y Linux. Como se aprecia en la *Figura 14* la interfaz del S4A tiene la misma disposición que la de Scratch original, pero con un aspecto menos refinado.

[20] https://scratch.mit.edu/projects/editor/
[21] https://www.scratch.school/aprender/descargar-scratch-3-0-a-nuestro-pc/
[22] https://s4a.cat/
[23] http://citilab.eu/
[24] http://smalltalk.cat/

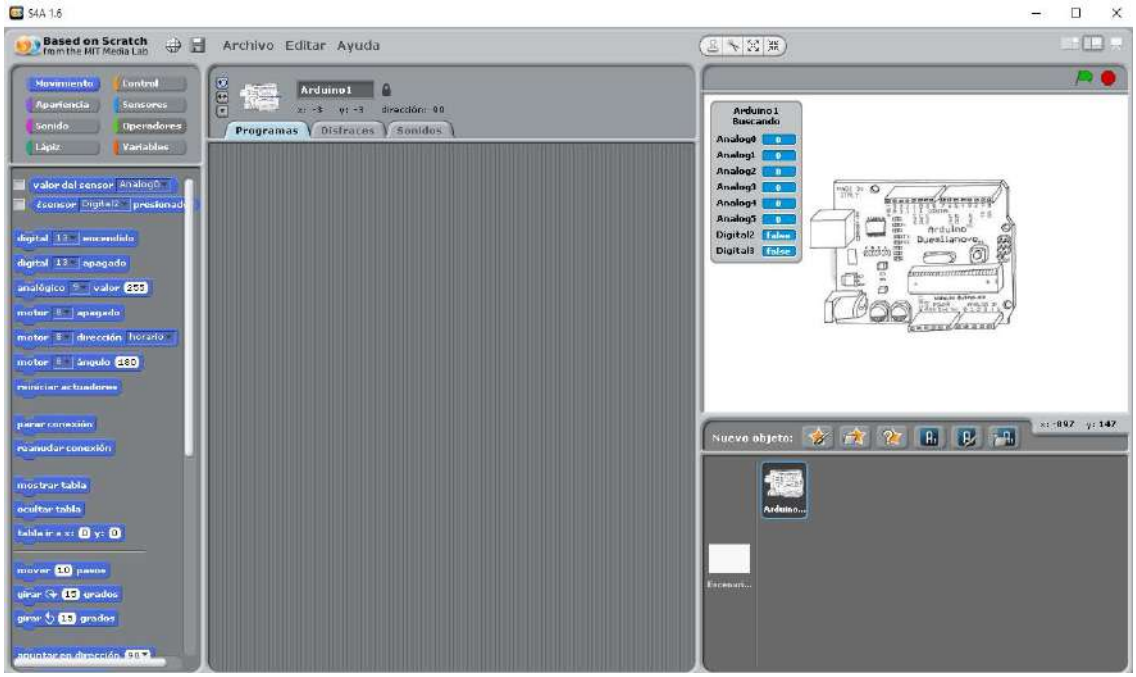


Figura 14: S4A

[Fuente: fuente propia]

Este entorno tiene algunas limitantes como son las placas soportadas oficialmente (Arduino Diecimila, Duemilanove y Uno), imposibilidad de expandir los bloques para soportar hardware que requiera de librerías, limitación en la forma en la que se pueden conectar los componentes a la placa. Según la documentación[25] los componentes deben conectarse de una forma determinada. S4A habilita 6 entradas analógicas (pines analógicos), 2 entradas digitales (pines digitales 2 y 3), 3 salidas analógicas (pines digitales 5, 6 y 9), 3 salidas digitales (pines 10, 11 i 13) y 4 salidas especiales para conectar servomotores de rotación continua (pines digitales 4, 7, 8 y 12). Sumado a todo esto es necesario cargar un programa en la Arduino mediante el entorno de Arduino previo a la utilización de S4A.

El proyecto no recibe actualizaciones desde el 2016 y la lista de correo parece estar abandonada desde el 2021.

[25] https://s4a.cat/

2.3.4.3 Extensión de Micro:Bit para Scratch

Existe una extensión de Micro:Bit para Scratch que agrega bloques capaces de controlar algunos componentes de la placa dentro de Scratch y la posibilidad de subir el código generado a la misma. La extensión se instala con dos simples clics y se puede utilizar tanto en la versión online de Scratch como en las versiones instalables. En la *Figura 15* se puede ver que los bloques agregados por la extensión no cubren la totalidad de las funciones de la Micro:Bit ni agrega el emulador de la placa que encontramos en los editores oficiales. Tampoco es posible agregar bloques complejos de manera sencilla desde el editor, para ello se necesitaría crear una extensión para Scratch desde cero.

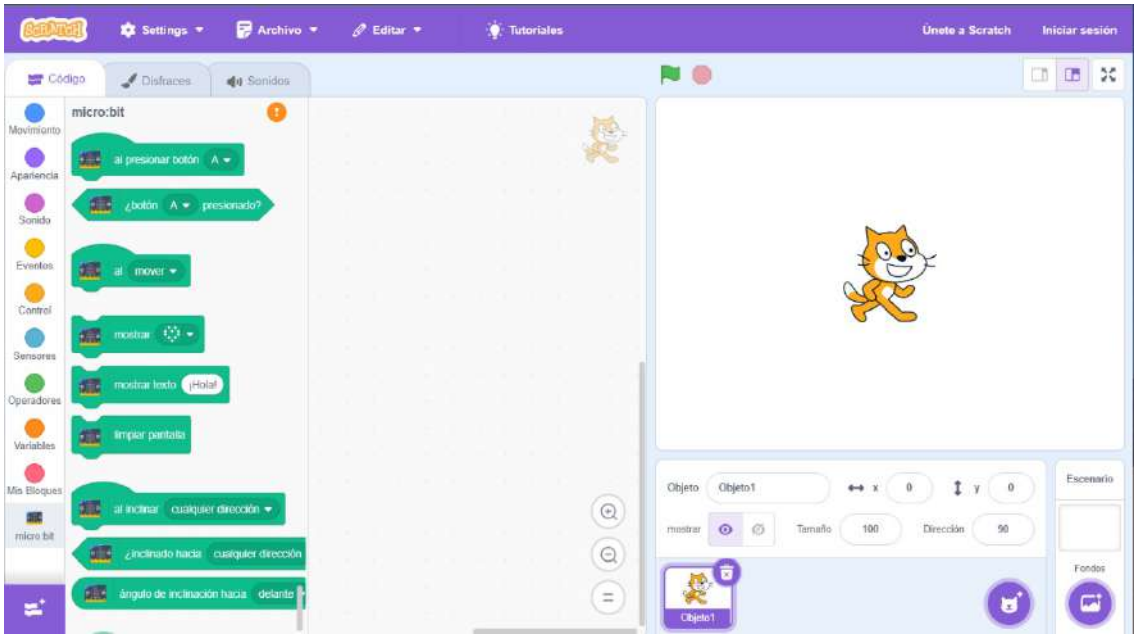


Figura 15: Extensión de Micro:Bit para Scratch

[Fuente: fuente propia]

2.3.4 Algunas reflexiones acerca de los entornos de programación

Una vez presentados los diferentes entornos de programación y al haber seleccionado para el uso a la placa Micro:Bit en la sección anterior, en este trabajo se toma como base de trabajo al entorno de programación MakeCode ya que además de completo y robusto, permite su ampliación mediante programación en JavaScript.

Capítulo 3: Trabajos relacionados

En este capítulo se analizan trabajos relacionados con la presente propuesta. Su análisis, cuando sea posible, se realiza considerando las siguientes dimensiones:

- Tipo de placa.
- Método de conexión de periféricos con las placas.
- Tecnología inalámbrica que se utiliza.
- En qué nivel del sistema educativo se recomienda su uso.
- Lenguaje y entorno de programación.
- Capacidad de programarla en bloques.
- Protocolo para enviar los datos al servidor.
- Se muestra un caso de estudio con docentes y/o estudiantes.

Luego de presentar el análisis de cada trabajo relacionado, se muestra una tabla que los compara desde las dimensiones analizadas y posteriormente, se indican los aspectos a considerar para el presente trabajo de tesina.

3.1 Análisis de los trabajos

Respecto a la relevancia del uso de placas y conectividad inalámbrica en educación, Pramod Abichandani [Abichandani et al., 2022] indica en su trabajo la existencia de quince mil papers relacionados al uso de IOT en educación, término más abarcativo pero estrechamente relacionado al tema en cuestión. El estudio se focaliza en el análisis de sesenta papers luego de un filtro exhaustivo y arroja que en la mayoría de los casos se usó alguna placa de la familia Arduino (68.5%), seguido por la familia de las Raspberry Pi (26.5%), las Micro:Bit (3%) y otras (2%). Por otro lado, también afirma que el uso de WiFi o Ethernet es casi ubicuo (90%) seguido por Bluetooth (6%) y Zigbee (4%) [Abichandani et al., 2022].

Por otro lado, Peech y Novák [Peech and Novák, 2020] comparan en su trabajo a las placas Arduino y Micro:Bit. Los autores se concentran en el análisis del impacto que tiene la utilización de dichas placas para la enseñanza STEM (*Science, Technology, Engineering and Mathematics*) en escuelas secundarias técnicas y no técnicas. Los autores mencionan que los dos principales problemas con los que se toparon los estudiantes y docentes utilizando Arduino en el aula, principalmente por parte de los estudiantes sin background técnico-electrónico, fueron, por un lado, la dificultad para realizar la conexión de la placa con los componentes, y por otro, la dificultad de escribir código en el entorno de Arduino. Como solución a esto, los autores proponen la utilización de Micro:Bit o ESP8266 para este tipo de audiencia, pero finalmente se deciden por la primera de las placas mencionadas. Entre los beneficios de usar la placa Micro:Bit mencionan su gran variedad de periféricos embebidos en la placa, lo que quita la necesidad de conectar componentes externos, y su capacidad de soportar varios lenguajes de programación: MakeCode, JavaScript y MicroPython. También mencionan la ventaja de poder conectar componentes (sensores) mediante cables con pinzas cocodrilo. Por último, a pesar de enfocarse principalmente en la enseñanza STEM, al final del trabajo mencionan que la enseñanza de programación con Micro:Bit se superpone con otras materias no STEM como por ejemplo, inglés, música, geografía y artes visuales [Peech and Novák, 2020].

Luego, Aneesh Pradeep [Pradeep, 2023] toma como eje central de su trabajo una placa ESP32. Propone el uso de esta placa como una plataforma IoT asequible para su uso en entornos educativos, enfocado en la recolección y análisis de datos. Este autor indica que la ESP32 es una placa de bajo costo y bajo consumo, con la capacidad de controlar sensores y actuadores mediante puertos de entrada/salida y capaz de comunicarse vía WiFi y Bluetooth mediante distintos protocolos. Aneesh además cita dos experiencias donde la utilización de la ESP32 para la recolección y análisis de datos en el aula intenta crear un entorno de aprendizaje más interactivo y personalizado, basado en la investigación, con feedback inmediato y que permiten a los estudiantes visualizar su progreso, fomentar su participación y el pensamiento crítico. En cuanto al entorno de programación menciona muy brevemente que se puede utilizar el IDE de Arduino, pero

no explica cómo en el desarrollo de su trabajo. Por último, propone la utilización de un kit de robótica para la enseñanza STEM [Pradeep, 2023].

Susy Lisseth Fernandez Quiñonez [Fernández Quiñonez, 2022] en su trabajo, destaca la importancia de la utilización de entornos de programación en bloque en contextos educativos y propone la realización de varias prácticas utilizando estos entornos en conjunto con placas Micro:Bit y ESP32. El primer grupo de prácticas las realiza utilizando los entornos de programación en bloques CodeSkool y ArduinoBlocks en conjunto con la ESP32. Dichas prácticas van desde simples ejercicios, donde solo se prende un led, hasta la recolección de datos de varios sensores y su envío vía WiFi a un servidor local mediante el protocolo MQTT y su posterior visualización mediante una interfaz web, o el encendido y apagado remoto de los leds conectados a la placa desde una interfaz web utilizando el protocolo HTTP. También se realizan prácticas de envió de datos y control remoto similares a las vistas utilizando Bluetooth en lugar de WiFi y una aplicación móvil creada con App Inventor (en lugar de un servidor web). El segundo grupo de prácticas se realiza utilizando la placa Micro:Bit junto al entorno MakeCode. Estas prácticas se centran en la utilización de los componentes embebidos de la Micro:Bit y la comunicación vía Bluetooth con la misma aplicación móvil antes mencionada. Por último, la autora realiza una encuesta y llega a la conclusión de que "el lenguaje de programación basado en bloques es un lenguaje fácil de comprender. Ya que al mostrar el código de bloques pueden comprender lo que está realizando el código sin necesidad de una explicación previa, es decir los bloques que se utilizan están detallados de tal manera que se hace comprensible a la vista del usuario resultando fácil de comprender y aprender" [Fernández Quiñonez, 2022].

Por último se analiza la extensión [pxt-iot-environment-kit] para MakeCode llamada "Pxt-iot-environment-kit" que permite agregar capacidades WiFi a la Micro:Bit mediante la utilización de una placa ESP8266. Esta extensión agrega bloques a MakeCode que permiten conectarse a una red WiFi y enviar mensajes a la plataforma de recolección y análisis de datos online ThingSpeak[26]. La forma en que se realiza es conectado la Micro:Bit y la ESP físicamente mediante sus puertos serie y enviando

[26] https://thingspeak.com/

comandos AT[27] por los mismos. Vale aclarar que para que esto funcione la ESP8266 debe tener previamente cargado un firmware que le permita entender los comandos AT. La conexión entre la placa ESP8266 y el servidor se da a través del protocolo MQTT[28]. La principal limitación de esta extensión es su atadura a la plataforma ThingSpeak, lo que impide su utilización con un servidor dentro de una red local sin conexión a internet o su uso con otras plataformas de recolección de datos en general. Por otro lado, los comandos[29] AT proporcionan funcionalidad limitada y poseen una sintaxis compleja y poco intuitiva [pxt-iot-environment-kit].

3.2 Comparación de los trabajos relacionados

Una vez presentada la descripción de los trabajos relacionados, se presenta una tabla comparativa entre ellos (*Tabla 3),* dejando de lado el estudio de Pramod Abichandani [Abichandani et al., 2022] debido a que solamente analiza la utilización de distintas tecnologías en ámbitos educativos y no contiene los parámetros utilizados para la comparación.

Tabla 3: Comparación de trabajos relacionados [Fuente: fuente propia]

TR\dimensiones analizadas	[Peech and Novák, 2020]	[Pradeep, 2023]	[Fernández Quiñonez, 2022]	[pxt-iot-environment-kit]
Tipo de placa	Micro:Bit Arduino	ESP32	Micro:Bit ESP32	ESP8266
Método de conexión de periféricos con las placas	No especifica	No especifica	Protoboard	No Aplica
Tecnología inalámbrica que se utiliza	No usa (Soporta	No especifica (Soporta	WiFi Bluetooth	WiFi

[27] https://es.wikipedia.org/wiki/Conjunto_de_comandos_Hayes
[28] https://www.ibm.com/docs/en/mas-cd/maximo-monitor/continuous-delivery?topic=messaging-mqtt-standards-requirements
[29] https://www.espressif.com/sites/default/files/4a-esp8266_at_instruction_set_en_v1.5.4_0.pdf

TR\dimensiones analizadas	[Peech and Novák, 2020]	[Pradeep, 2023]	[Fernández Quiñonez, 2022]	[pxt-iot-environment-kit]
	Bluetooth)	WiFi y Bluetooth)		
En qué nivel del sistema educativo se recomienda su uso	Secundario (Escuelas de Gramática y Técnicas Checas)	No especifica	No especifica	No especifica
Lenguaje y entorno de programación	MicroPhyton C++ MakeCode IDE Arduino	C++ IDE Arduino con plugins	Lenguaje de Bloques C++ MakeCode, CodeSkool ArduinoBlock Appinventor	Lenguaje de Bloques MicroPython JavaScript MakeCode
Capacidad de programarla en bloques	Si	No especifica	Si	Si
Protocolo para enviar los datos al servidor	No Aplica	No especifica	MQTT Bluetooth HTTP	MQTT

Capítulo 4: Pruebas sobre placas

En este capítulo se presentan pruebas realizadas con las placas Micro:Bit y Arduino y luego sumando la placa ESP.

4.1 Prueba usando Micro:Bit

Las placas Micro:Bit v2 (*Figura 16*) son microcomputadoras diseñadas por la BBC con la intención de alentar el interés de los niños en la programación y la electrónica[30]. Sus creadores la describen como "una computadora de bolsillo que te introduce en cómo trabajan el software y el hardware en conjunto. Poseen una pantalla led, botones, sensores y varias características de entrada/salidas programables con las que se puede interactuar. La última versión agrega las capacidades de sensar y reproducir sonido."[31]

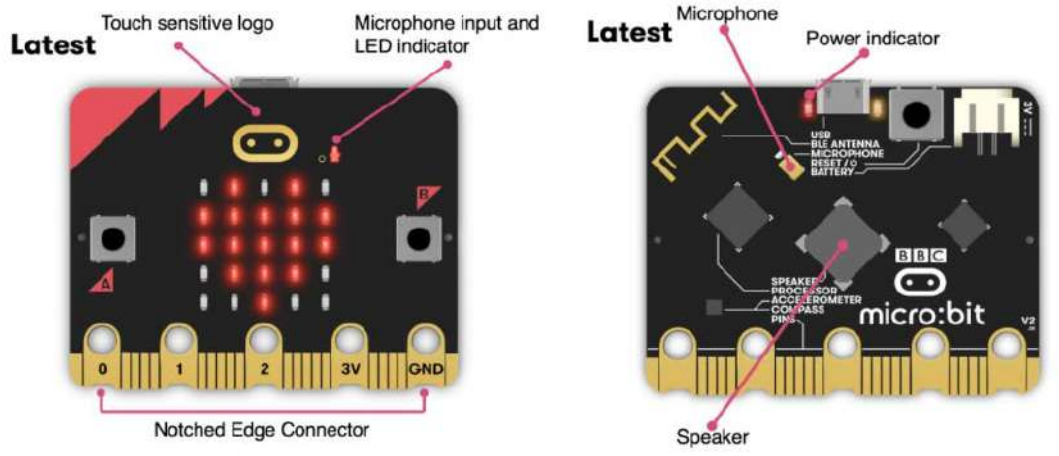


Figura 16: Imágenes frontal (der.) y posterior (izq.) de la placa Micro:Bit V2 [Fuente: https://support.microbit.org/]

[30] https://www.theguardian.com/technology/2015/jul/07/bbc-give-away-1m-microbit-computers-schoolchildren

[31] https://support.microbit.org/support/solutions/articles/19000013983-what-is-a-micro-bit-

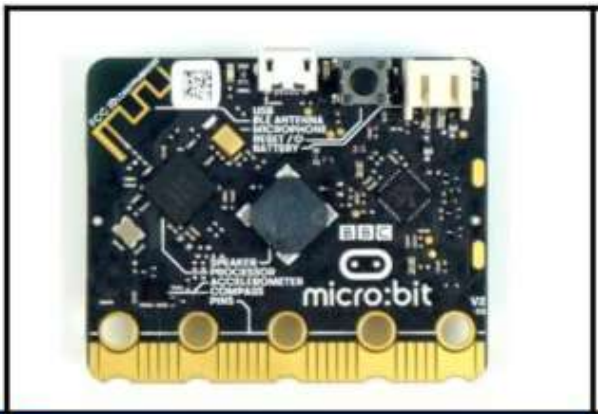
Como se muestra en la *Figura 16* y en la Tabla 4, su hardware está compuesto por un procesador ARM Cortex-M4 a 64 MHz, 128 KB de RAM, una matriz de led de 5x5, acelerómetro, compás, bluetooth, micrófono, parlante, dos botones físicos y uno capacitivo. También posee 25 pines de entrada salida en forma de cartucho, de los cuales solo 4 son para entrada salida de propósito general (GPIO), el resto son utilizados por los componentes de la placa[32].

Todos los componentes mencionados hacen que se puedan implementar una gran variedad de proyectos interactivos con la Micro:Bit sin la necesidad de conectar y configurar módulos externos.

Por el lado del software existen dos plataformas de codificación oficiales: Microsoft MakeCode[33] y MicroPython[34] las cuales se describen en profundidad en la sección de comparativa de entornos de programación en el Capítulo 2. Ambas plataformas facilitan la programación de la Micro:Bit proporcionando librerías con las cuales se puede interactuar de manera sencilla con todos los componentes de la misma. MakeCode es particularmente interesante debido a implementar programación en bloques lo que la hace muy adecuada para su uso por personas no familiarizadas con la programación tradicional.

32 https://makecode.microbit.org/device/pins
33 https://makecode.microbit.org/
34 https://python.microbit.org/v/2

Tabla 4: Tabla de componentes de la Micro:Bit V2[Fuente: https://www.mybotic.com.my/microbit?product_id=3561]



Features/Specs	micro:bit V2
Release Date	13th Oct 2020
MCU or Processor	Nordic Semiconductor **nRF52833**
MCU Core Architecture	ARM **Cortex-M4** 32-bit **(FPU)**
MCU Flash Size	**512KB**
RAM Size	**128KB**
MCU Clock	**64MHz**
USB Interface Processor	NXP **KL27Z**, **32KB RAM**
Microphone, MIC	**MEMS Microphone, LED indicator**
Speaker	**Onboard Piezo Buzzer**
Touch Sensitive Logo	**Touch Sensitive Logo Pad**
Wireless	2.4GHz micro:bit radio/BLE **Bluetooth 5.1**
Power	5V via USB, 3V via edge connector or battery port
Power indicator LED	**Onboard Power Indicator LED**
Power Off Button	**Onboard Power Button (Push and Hold*)**
Current for External	3V, **200mA**
Motion Sensor	ST LSM303
Edge Connector	25-pin, **4 dedicated GPIO**, PWM, I2C, SPI, Power, and etc
Ring Connector	3 (GPIO) + 2 (Power) ring connectors, **notched edge**
I2C	**Dedicated I2C Bus** for peripherals
Software/IDE	C++, makecode, Python, Scratch
Size	5cm (w) x 4cm (h)

4.1.1 Análisis de Factibilidad usando Micro:Bit

Se eligió esta plataforma de hardware para comenzar las pruebas de factibilidad por estar diseñada para ser utilizada en educación y por sus capacidades de entrada salida inalámbricas incorporadas en la propia placa: RADIO y BLE.

4.1.1.1 Comunicación vía RADIO

Un primer análisis sobre las funciones provistas por el entorno de programación en bloques MakeCode revela una manera muy sencilla y pragmática de enviar mensajes inalámbricamente desde la placa. Esto da la idea de que debería ser relativamente sencillo leer los datos de un sensor conectado a los pines de expansión y enviarlos a un servidor.

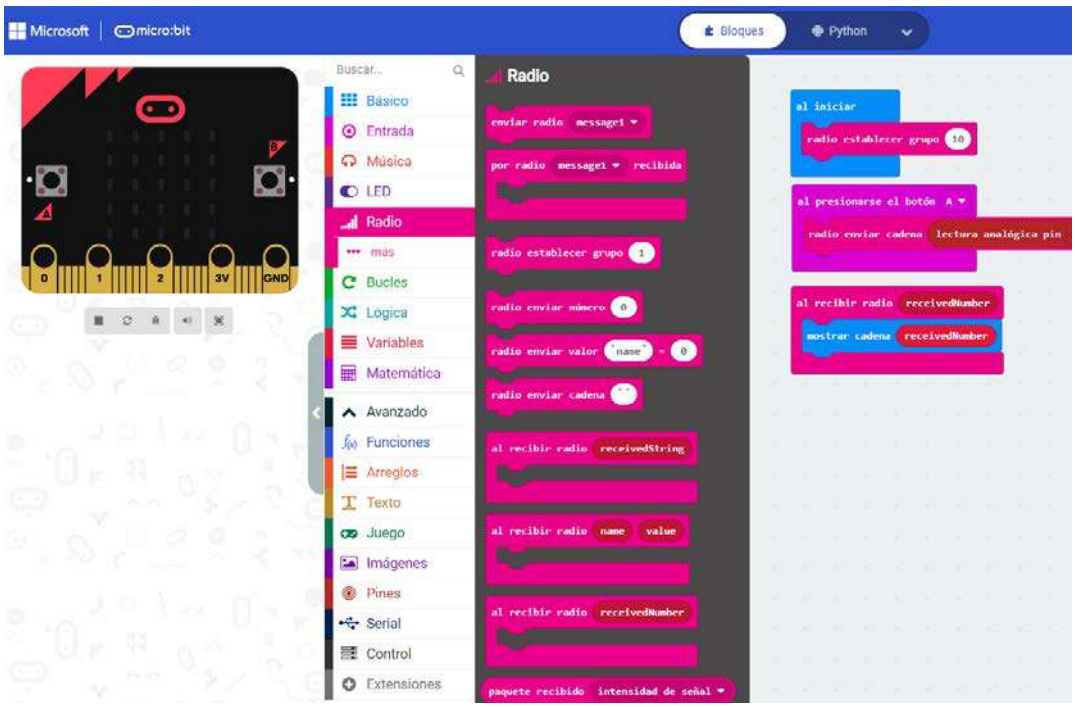


Figura 17: Ejemplo en MakeCode de lectura de un pin analógico y envió de datos por protocolo "radio" de Micro:Bit [Fuente: fuente propia]

En el ejemplo de la *Figura* 17 podemos ver que lo único necesario para comunicar varias placas vía radio es que estén configuradas todas dentro del mismo grupo. Una vez realizada la configuración cualquier placa que transmita vía radio emitirá un mensaje vía broadcast capaz de ser escuchado por cualquier otra placa dentro del mismo grupo.

El problema radica cuando queremos enviar la información a otro dispositivo que no sea una placa Micro:Bit. El protocolo radio es un protocolo privativo de Nordic Semiconductor y Lancaster University[35] y su stack ya viene incorporado en las placas Micro:Bit, por lo cual implementar un servidor compatible en un hardware distinto sería, como mínimo, bastante difícil.

Descartado el protocolo radio pasamos a la siguiente opción de entrada/salida inalámbrica que ofrece Micro:Bit: BLE (Bluetooth Low Energy)

4.1.1.2 Comunicación vía Bluetooth Low Energy (BLE)

El BLE[36], sigla de *Bluetooth Low Energy*, es un protocolo privativo de comunicación inalámbrica introducido en el año 2011 por el *Bluetooth Special Interest Group* (SIG). Su principal objetivo es operar con la mayor eficiencia energética posible[37], manteniendo un rango de comunicación similar a Bluetooth. Se diseñó con la finalidad de ser utilizado en dispositivos que deban operar con baterías por largos periodos (monitores de dispositivos médicos, smartwatches, sensores IoT, etc.). Soporta varias topologías de red (point-to-point, broadcast y mesh), lo que lo hace especialmente atractivo en aplicaciones machine-to-machine[38]. Otra característica importante de este protocolo es la inclusión de funciones que permiten que un dispositivo determine la

[35] https://tech.microbit.org/bluetooth/

[36] https://web.archive.org/web/20170310111443/https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/low-energy

[37] https://www.researchgate.net/publication/304102769_BLE_or_IEEE_802154_Which_Home_IoT_Communication_Solution_is_more_Energy-Efficient

[38] M. I. Jameel and J. Dungen, "Low-power wireless advertising software library for distributed M2M and contextual IoT," 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 597-602, doi: 10.1109/WF-IoT.2015.7389121.

presencia, la distancia y la dirección de otro dispositivo, lo que permite la implementación de aplicaciones de posicionamiento de alta precisión en interiores[39]. Cabe mencionar que BLE no puede interoperar con Bluetooth estándar.

A pesar de que en principio las características mencionadas podrían cubrir los requisitos de comunicación inalámbrica propuestos al inicio del estudio, se optó por descartar este protocolo por las siguientes razones:

- Es un protocolo privativo diseñado e implementado por un conjunto de empresas (SIG), lo que dificulta el acceso a la documentación, no garantiza la idéntica implementación por parte de los fabricantes ni nos permite auditoria de las mimas.
- No es compatible con el stack clásico de bluetooth. Lo que puede llevar a cierta confusión por parte de los usuarios finales que pueden tener un dispositivo con capacidades bluetooth que no implemente el stack BLE y no funcione con nuestro Micro:Bit.
- Por su naturaleza privativa se encuentra menor cantidad de documentación, librerías y ejemplos para el desarrollo del servidor para la recolección de datos.
- Se encontraron problemas de seguridad tanto en el protocolo Bluetooth Clásico como en BLE[40].

4.1.2 Resultados

A pesar de que la placa Micro:Bit y su entorno de programación tienen características que los hacen muy atractivos para el uso en entornos educativos, el carácter privativo de los protocolos de comunicación inalámbrica embebidos en la placa dificultan la implementación del sistema planteado por este estudio.

[39] https://www.bluetooth.com/learn-about-bluetooth/tech-overview/
[40] https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/blurtooth/

4.2 Pruebas usando Arduino

Para la segunda prueba se utilizó un Arduino Mega V3. Esta placa es una evolución de la primera placa de Arduino, la Arduino Uno (una evolución en el tiempo de este tipo de placas puede ser visto en la Figura 18). A pesar de no estar ideadas exclusivamente para entornos educativos, estas placas tienen una relevancia especial porque fueron las primeras en tratar de hacer más accesible el ingreso al mundo de la electrónica a gente con poco o nulo conocimiento de esta. Para lograr esto se creó una placa de bajo costo que pudiera ser programable vía USB sin ningún hardware adicional, alimentada eléctricamente por el mismo cable USB con varios conectores Dupont para poder conectar fácilmente periféricos sin la necesidad de soldar.

Paralelamente a la placa se creó un IDE para facilitar su programación y configuración. Este IDE posee varias herramientas que permiten, por ejemplo: editar, compilar y subir código, seleccionar parámetros de comunicación como el puerto y el baud-rate, enviar y recibir información desde el pc a la placa y viceversa, a través de una terminal virtual.

Tanto el hardware como el software se publicaron bajo el modelo de hardware y software libre respectivamente, lo que permitió la proliferación de placas similares y compatibles con el IDE. Por el lado del software este modelo permitió la expansión del IDE mediante infinidad de pluggins y librerías. Todo esto dio como resultado el amplio ecosistema de software y hardware disponible actualmente apoyado por una comunidad colaborativa que le da soporte y busca soluciones a las inquietudes de los usuarios.

Como se detalla en la *Tabla 5*, la Arduino Mega posee un microprocesador ATmega2560 de un solo núcleo que corre a 16MHz, 8KB de memoria SRAM y 256KB de memoria Flash disponibles para guardar los programas generados por los usuarios. También dispone de 54 pines los cuales se pueden usar en modo E/S digital, dentro de esos encontramos 16 pines de entrada analógicos y 15 pines de salida analógica o PWD (Pulse Width Modulation).

Como conclusión de las características mencionadas se puede decir que a diferencia de la primera placa que se probó, la Arduino Mega posee menos capacidad de

procesamiento y memoria, más puertos de entrada/salida y no posee capacidad de comunicación inalámbrica.

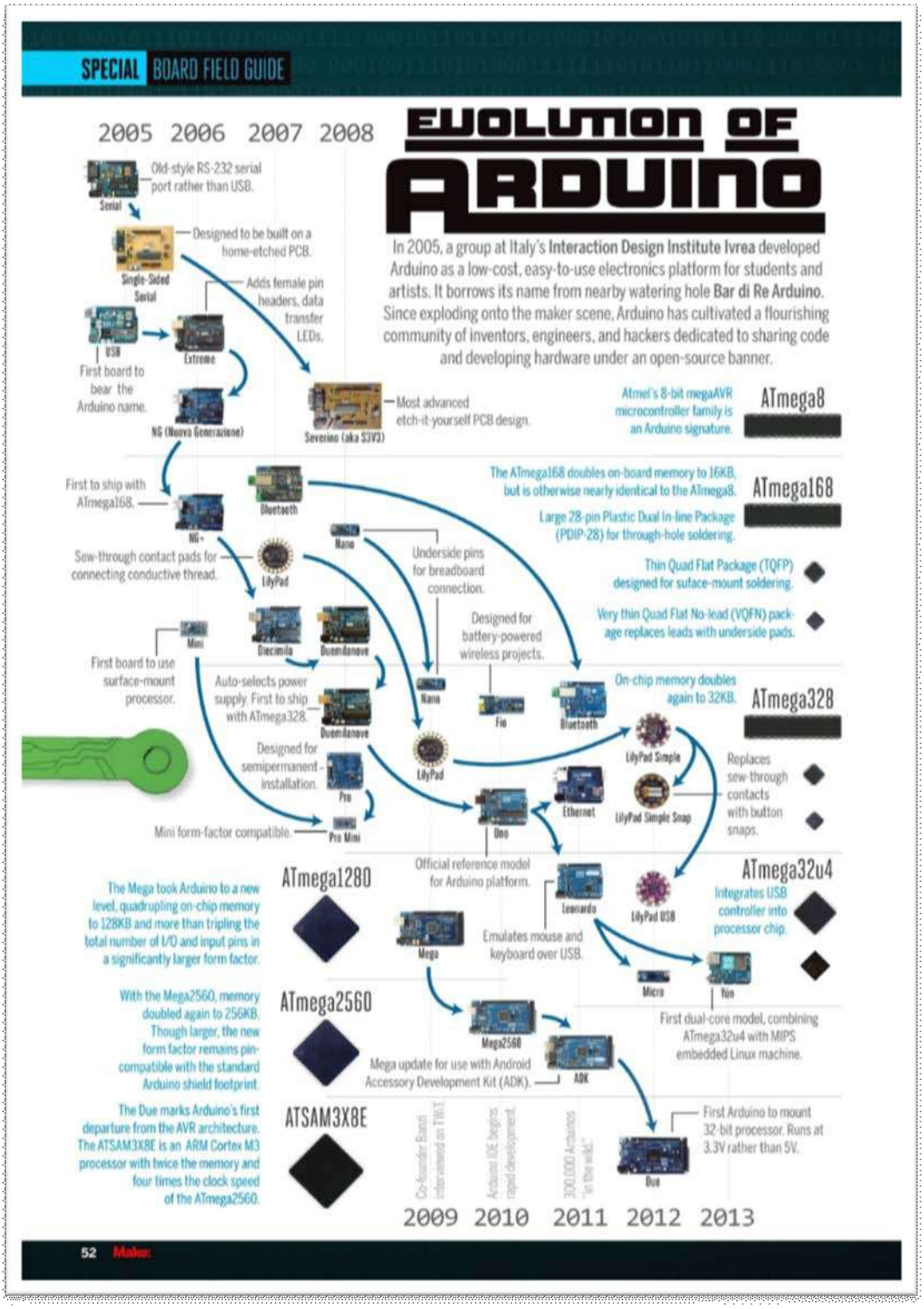


Figura 18: Evolución de placas Arduino

[Fuente: https://blog.arduino.cc/2013/11/02/evolution-of-arduino-the-family-tree/]

Tabla 5: Tabla de componentes Arduino Mega V3 [Fuente: https://docs.arduino.cc/hardware/mega-2560]

Here you will find the technical specifications for the Arduino® Mega 2560 Rev3.

Board	**Name**	Arduino® Mega 2560 Rev3
	SKU	A000067
Microcontroller	ATmega2560	
USB connector	USB-B	
Pins	**Built-in LED Pin**	13
	Digital I/O Pins	54
	Analog input pins	16
	PWM pins	15
Communication	**UART**	Yes, 4
	I2C	Yes
	SPI	Yes
Power	**I/O Voltage**	5V
	Input voltage (nominal)	7-12V
	DC Current per I/O Pin	20 mA
	Supported battery	9V battery
	Power Supply Connector	Barrel Plug
Clock speed	**Main Processor**	ATmega2560 16 MHz
	USB-Serial Processor	ATmega16U2 16 MHz
Memory	**ATmega2560**	8KB SRAM, 256KB FLASH, 4KB EEPROM
Dimensions	**Weight**	37 g
	Width	53.3 mm
	Length	101.5 mm

4.3 Prueba usando Arduino con Shiel WiFly

Para la comunicación inalámbrica se agregó un shield Wifi Sparkfun “WiFly”. Este Shield basado en el chip RN-131C de Roving Networks soporta únicamente la conexión a redes WiFi 802.11b/g. El motivo de la elección de estas versiones de Arduino y de shield WiFi fue simplemente por ser el hardware al que se pudo acceder al momento de iniciar la prueba.

En un primer intento, luego de conectar el shield al Arduino y subir el código correspondiente, no se pudo hacer andar el hardware. Posteriormente se encontró la solución en el foro de Arduino: Al ser un shield diseñado para Arduino Uno debemos re cablear 4 puertos para su correcto funcionamiento. Una vez hecho esto se pudo hacer andar las placas correctamente.

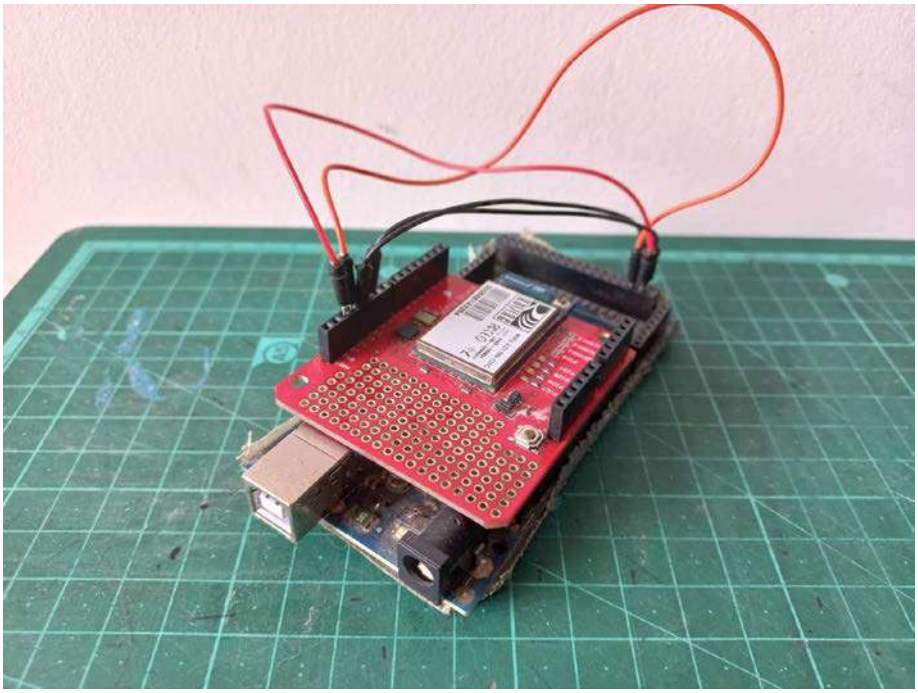


Figura 19: Arduino Mega (abajo) con Shield Sparkfun "WiFly" (arriba) recableado [Fuente: Imagen propia]

Luego de lograr que las placas funcionaran se procedió a buscar información técnica para agregar a la documentación y se descubrió que el shield está obsoleto, Sparkfun discontinuo su venta y el último commit importante en el github[41] de sus librerías fue en el 2015.

Se descartó el shield WiFly porque a pesar de cumplir con los requerimientos que se tenían en mente su obsolescencia impide que el resultado final pueda replicarse por otras personas.

41 https://github.com/sparkfun/WiFly-Shield

4.4 Búsqueda de un módulo WiFi alternativo

Descartado el primer shield se procedió a buscar una alternativa para agregar capacidad de conexión inalámbrica al Arduino. Para esto se realizó una búsqueda con las palabras clave "Arduino+Wifi" en proveedores internacionales como Amazon y SparkFun, y nacionales como Patagonia Tec y Mercado Libre, a partir de lo anterior se determinó que el 87% de las placas ofrecidas para satisfacer nuestro propósito son alguna variante basada en Chips ESP de Espressif. Se pueden ver estos resultados con más detalle en la *Tabla 6* donde se muestra, para cada proveedor, el tamaño de la muestra expresada en cantidad de elementos hallados (muestra), la cantidad de ítems que no cumplen con el requerimiento de poseer la capacidad de agregar conexión inalámbrica al Arduino (No Cumple), la cantidad de ítems que cumplen este requerimiento y además utilizan un chip fuera de la familia ESP (Wifi No-ESP), la cantidad de ítems que utilizan un chip de la familia ESP (Wifi ESP) y el porcentaje de chips ESP sobre el total de chips WiFi (Porcentaje ESP).

Tabla 6: Comparativa de hardware WiFi compatible con Arduino [Fuente: Estudio propio]

Pagina	Muestra	No cumple	Wifi No-ESP	Wifi ESP	Porcentaje ESP
Amazon	16 elementos	28	2	18	90%
SparkFun	46 elementos	21	10	15	60%
Patagonia Tec	24 elementos	4	0	20	100%
Mercado Libre	54 elementos	10	1	43	99%

El tamaño de la muestra de cada proveedor, visualizado en la Tabla 6, corresponde a distintos criterios. En Amazon se optó por tomar las 3 primeras páginas de búsqueda dado que en las siguientes páginas se empezaba a repetir el patrón de elementos encontrados. SparkFun posee una página de carga dinámica por ende se decidió cortar donde empezaron a aparecer varias líneas seguidas donde ningún producto cumplía con el requerimiento. Con Patagonia Tec se tomó la totalidad de los ítems que devolvió la

búsqueda. Y por último en Mercado Libre se tomó la primera página por contener una cantidad de elementos suficientes.

En la *Tabla 7* se presenta la sumatoria de los valores expresados en cada columna de la *Tabla 6*. En la Tabla 7, de derecha a izquierda se puede ver en el total de elementos muestreados (muestra), la cantidad total de ítems que no cumplen con el requerimiento (No Cumplen), la cantidad total de elementos que cumplen con el requerimiento (Cumplen), la cantidad total de elementos que cumplen con el requerimiento pero no poseen chips de la familia ESP (Wifi No-ESP), la cantidad total de elementos que cumplen con el requerimiento y poseen chips de la familia ESP (Wifi ESP) y por último el porcentaje promedio de elementos que cumplen el requerimiento y aparte poseen chips de la familia ESP (Porcentaje ESP).

Tabla 7: Cantidades totales de los elementos de la Tabla 1 y porcentaje promedio de placas que utilizan ESP [Fuente: Estudio propio]

Muestra	No cumplen	Complen	Wifi No-ESP	Wifi ESP	Porcentaje ESP
140 elementos	63	109	13	96	87%

Dentro de los ítems que se consideran que cumplen con los requisitos (Columna "Cumplen" de la *Tabla 7*) podemos encontrar módulos WiFi, kits y placas Arduino compatible con conectividad WiFi incorporada. Fuera de la familia de los chips ESP se encuentra el 13% de módulos y placas relevados que utilizan alguno de los siguientes chips WiFi: NINA-W10, AP 6212, ATSA MW25, Murata 1DX y DA 16200.

En la *Figura* 20 se presenta visualmente la diferencia en porcentaje entre las placas de la familia ESP y las que no pertenecen a dicha familia.

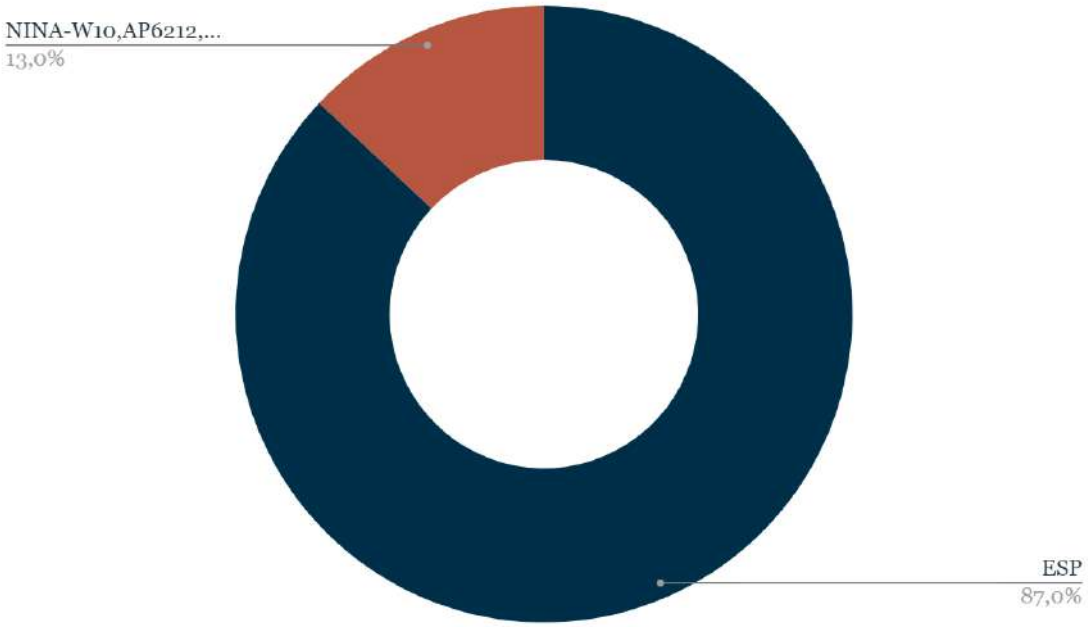


Figura 20: Grafico comparativo de chips WiFi en módulos para Arduino

[Fuente: Estudio propio]

En la *Figura* 20 se destaca que las placas de la familia ESP predominan por sobre las que no pertenecen a esta familia.

Dados los resultados del estudio antes mencionado se optó por adquirir una placa ESP32 y otra ESP8266 para agregar conectividad wifia al Arduino y avanzar con las pruebas en este sentido, lo que se describe en la siguiente sección del capítulo.

4.5 Pruebas usando ESP

Las placas ESP8266 y ESP32 son módulos fabricados por Espressif cuya principal función es proporcionar conectividad inalámbrica. El ESP8266, lanzado inicialmente como un módulo WiFi de bajo consumo, se ha convertido en una opción popular para el desarrollo de prototipos debido a su tamaño compacto y bajo costo. Por otro lado, el ESP32 es una versión más avanzada que combina conectividad WiFi y Bluetooth,

ofreciendo capacidades mejoradas de procesamiento y E/S, lo que lo convierte en una elección más adecuada para aplicaciones más exigentes. Ambas placas son ampliamente utilizadas en la comunidad de desarrollo de hardware y software debido a su accesibilidad y funcionalidades, facilitando la implementación de proyectos que necesitan conectividad inalámbrica.

4.5.1 Presentación de los módulos

En la Figura 21, se presentan ambos módulos, y se observa que cada uno de ellos tienen características físicas similares midiendo aproximadamente 6cm de largo por 3 de ancho, poseen el chip montado en la parte superior de la placa y un conector micro USB en la parte inferior.

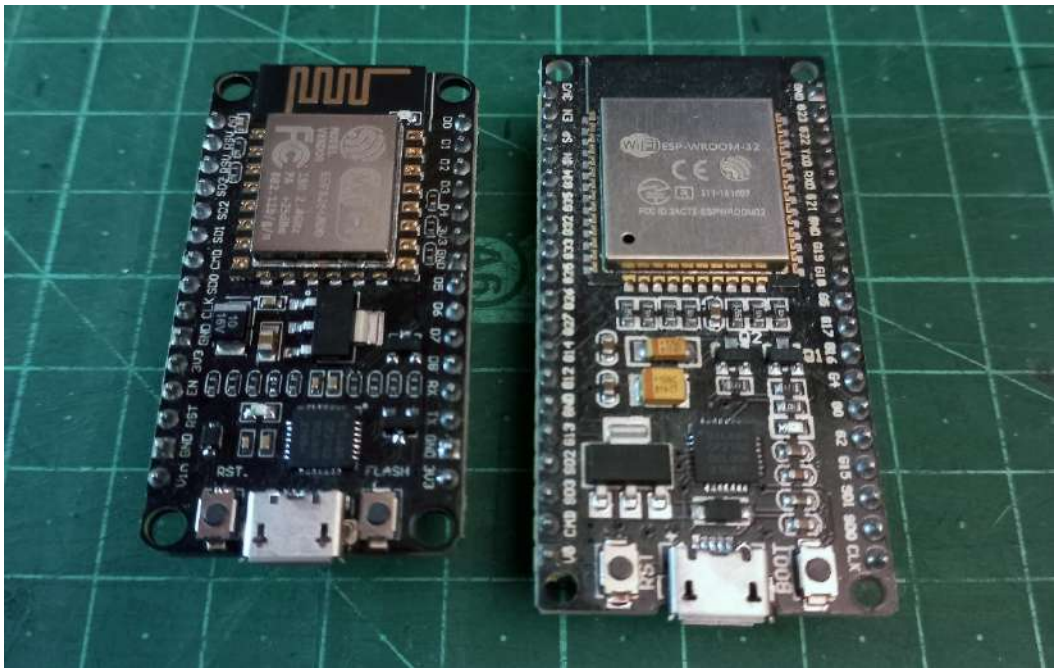


Figura 21: Módulo ESP8266 (izq.) y ESP32 (der)

[Fuente: Imagen propia]

Aparte de las versiones mostradas en la *Figura 21* se los puede encontrar en otras versiones con distintas características como, por ejemplo: sin pines de E/S soldados, distintos tamaños de placa, mayor o menor cantidad de memoria RAM, con o sin firmware instalado.

4.5.2 Características de los módulos analizados

A fines prácticos ambos módulos cumplen la función de agregar conectividad WiFi (b/g/n) a las placas Arduino y Micro:Bit. Por otra parte, se puede deducir del cuadro comparativo en la *Tabla 8* que el ESP32 posee algunas ventajas sobre el ESP8266 entre las que podemos encontrar: mayor poder de procesamiento, dado por una mayor cantidad de cores funcionando al doble de velocidad, incorporación de conectividad Bluetooth y mayor cantidad de puertos de entrada/salida.

Tabla 8: Cuadro comparativo entre los módulos ESP8266 y ESP32 [Fuente: Estudio propio]

	ESP8266	ESP32
Procesador	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6 with 600 DMIPS
802.11 b/g/n Wi-Fi	Si	Si
Bluetooth	No	Bluetooth 4.2 and BLE
Frecuencia del Procesador	80 MHz	160 MHz
GPIO (E/S)	17	34
Hardware /Software PWM	None / 8 channels	None / 16 channels
Sensor Touch	No	Si
Sensor de Temperatura	No	Si
Sensor Hall effect	No	Si
Precio	3 a 6 u$d	6 a 12 u$d

La Tabla 8 se confeccionó mediante el análisis de los datasheets[42 y 43] de cada una de las placas.

[42] https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
[43] https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

4.5.3 ESP + Arduino

Una vez adquiridos los módulos ESP32 y ESP8266 se procedió a investigar la manera de conectarlos a la placa Arduino. Luego de analizar varias entradas en el foro[44] oficial de Arduino y las características técnicas en los datasheets[45 y 46] de ambos módulos se llegó a la conclusión de que los sensores pueden ser conectados directamente a los puertos de entrada salida del ESP sin la necesidad de utilizar la placa Arduino, simplificando así las conexiones de hardware y disminuyendo el código necesario al tener que programar solamente el ESP. Dado que el IDE oficial de Arduino no provee soporte para la programación con bloques y ambas placas se programan con él, la utilización de una placa Arduino con la ESP es redundante y no genera ningún beneficio.

Como prueba concepto se decide conectar el ESP32 a un sensor de humedad y temperatura DTH22 e intentar enviar los datos recolectados vía WiFi. El primer paso para llevar a cabo esta prueba es configurar el entorno de programación de Arduino para que reconozca la placa ESP32, instalando las librerías necesarias y posteriormente eligiendo la placa con la que se usará el entorno. Luego se conecta la placa al sensor como se muestra en la *Figura 23*. Con el hardware y el entorno de programación listo se procede a implementar en la ESP32 un programa sencillo que muestra las lecturas del sensor en el monitor serial del entorno, para probar la compatibilidad entre el sensor y el módulo.

[44] https://forum.arduino.cc/
[45] https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
[46] https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

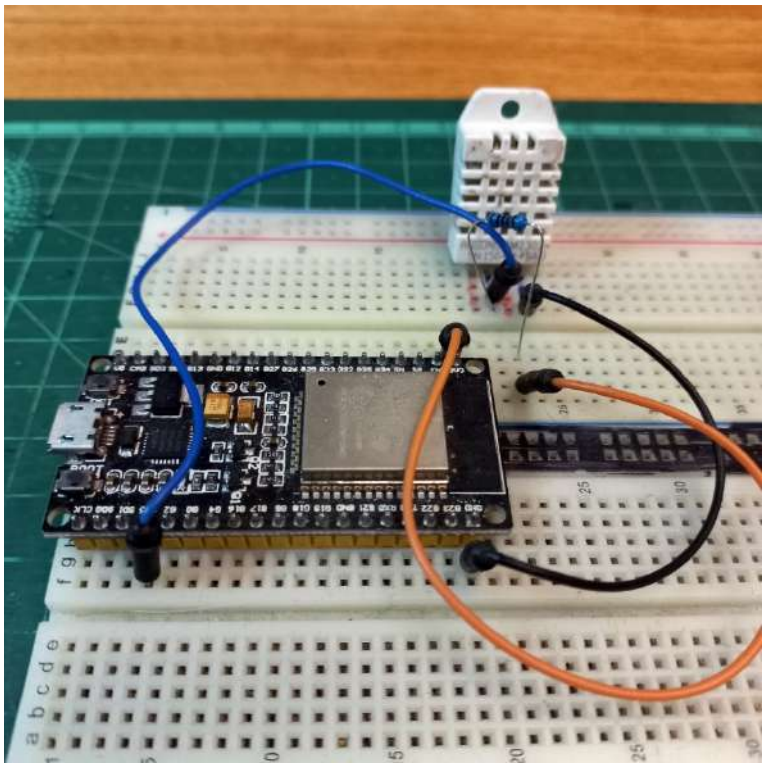


Figura 22: Modulo ESP32 conectado al sensor DTH22

[Fuente: Imagen propia]

4.5.4 ESP + Micro:Bit

El primer intento se realizó conectado puertos de entrada/salida analógicos de la Micro:Bit con puertos de entrada/salida analógicos de la ESP, dando un resultado fallido. Luego de investigar varios foros se descubre que la forma de conectar ambas placas es mediante el puerto serie de estas. Se crean dos programas simples para probar dicha comunicación. El primero, realizado en MakeCode para la Micro:Bit, para realizar la comunicación serie. El segundo programa, realizado en el entorno de Arduino para la ESP, consiste en la configuración del baud-rate y de los puertos Tx y Rx para la comunicación serie, un bucle para la escucha del canal y la impresión de los datos recibidos en la consola del entorno de Arduino.

Una vez creados y cargados los programas en las placas se procede a conectarlas con los elementos con los que se disponía en dicho momento, como se muestra en la *Figura 23*: una pinza cocodrilo, filamentos de cable UTP y broches de madera miniatura.

Los resultados de la comunicación son exitosos pero la conexión resulta inestable cuando se mueve cualquier componente de la misma.

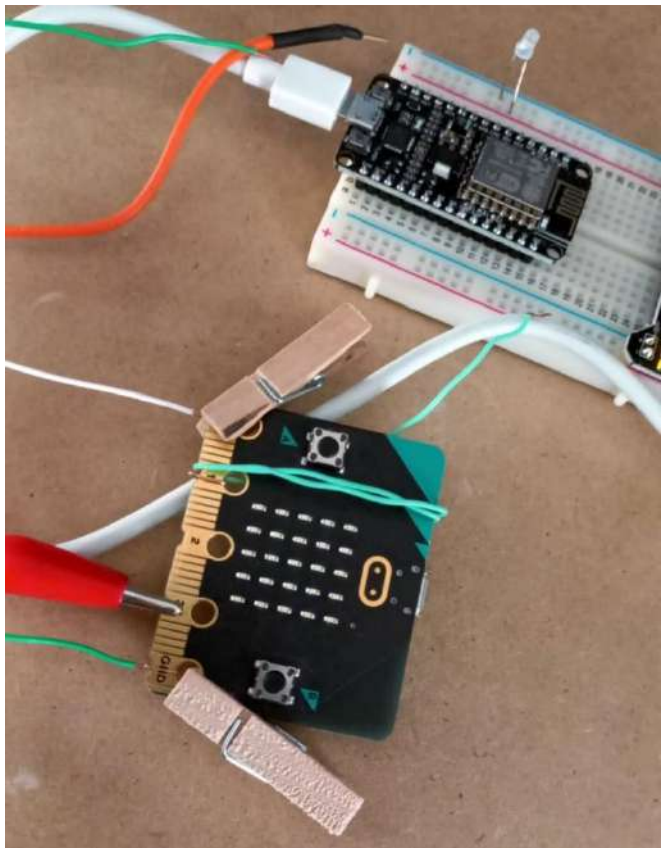


Figura 23: Micro:Bit conectada a módulo ESP

[Fuente: Imagen propia]

Para resolver el problema de inestabilidad en la conexión se recurre a un módulo expansor de puertos como el que se muestra en la *Figura 24*, que permite el acceso a todos los puertos de E/S de la Micro:Bit mediante conectores DuPont y brinda la posibilidad de alimentarla mediante un transformador.

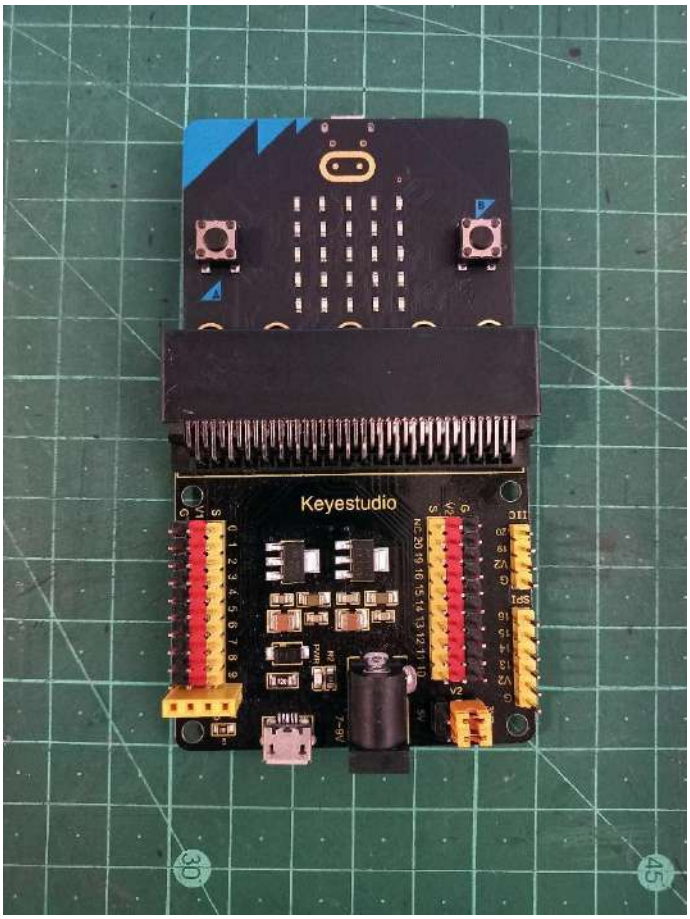


Figura 24: Micro:Bit conectado a expansor de puertos

[Fuente: Imagen propia]

Se debe tener especial cuidado cuando se utiliza este módulo ya que habilita el acceso a puertos de E/S utilizados por los componentes embebidos en la Micro: Bit, por lo que su uso indebido puede llevar a resultados inesperados. Por ejemplo, si utilizamos un pin que está conectado a la matriz de led para leer información desde la Micro: Bit mientras intentamos escribir en pantalla podría pasar que se escriba la información que queremos leer en la pantalla o que leamos el estado de una porción de la pantalla en lugar de la información que deseamos leer.

Con el problema de la inestabilidad de las conexiones resuelto se procede a conectar un sensor de humedad y temperatura al expansor de puertos y se modifica el código de la Micro:Bit para que envíe los datos del sensor a la ESP. El resultado es exitoso probando así la posibilidad de conexión entre la Micro:Bit y la ESP.

4.5.5 Prueba de envío de datos mediante WiFi

Una vez chequeado el correcto funcionamiento de los componentes se sigue por implementar un servidor REST en Python capaz de recibir la información que se envía desde el ESP, el cual se abordará con más detalle en el *Capítulo 6: Implementación de componentes de la propuesta de solución*. Por último, se modifica el programa dentro del módulo ESP para que deje de mostrar los datos sensados por el monitor serial y los envíe al servidor REST.

Con esta pequeña prueba se demuestra la factibilidad del sensado y envío de datos vía wifi y se sientan las bases para la creación de la infraestructura propuesta al principio de la tesina.

4.6 Diagrama conceptual de la evolución de las pruebas realizadas

Una vez presentadas las pruebas realizadas con las tres placas analizadas (Arduino, Micro:Bit y ESP), se procede a mostrar mediante un diagrama la lógica seguida para destacar como se llega a proponer en los capítulos sucesivos un modelo de software y hardware desacoplado.

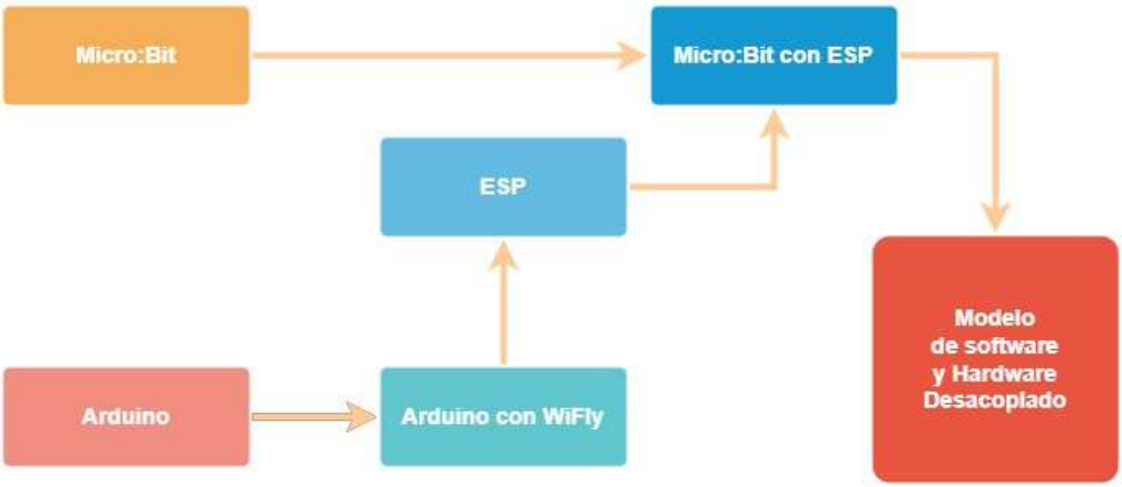


Figura 25: Evolución de las pruebas

[Fuente: Diagrama propio]

4.7 Conclusiones

Luego de las sucesivas pruebas utilizando las placas Micro:Bit, Arduino, WiFly y ESP, junto con algunas combinaciones de ellas se llegó a la conclusión de que el hardware más adecuado para resolver el problema planteado es la combinación de la Micro:Bit en conjunto con la ESP. La selección de la placa Micro:Bit se debe a tener el entorno de programación en bloques más amigable de todos los analizados en esta tesina y la elección de la placa ESP se debe a su amplio uso, gran cantidad de documentación y ejemplos y su bajo costo.

Capítulo 5: Propuesta de Solución

En este capítulo se exponen dos soluciones que implementadas en conjunto intentan atacar la problemática de este estudio. Por un lado se propone la creación de un protocolo de comunicación entre la Micro:Bit y la ESP que permita simplificar el uso de las placas abstrayendo lo más posible la implementación para el usuario final y dando suficiente flexibilidad para su utilización con diferentes sensores sin tener que cambiar el código. Adicionalmente se propone la creación de bloques personalizados en MakeCode para la configuración de red y el envió de datos sensados abstrayendo todo el trabajo subyacente hecho por ambas placas.

5.1 Descripción general del contexto de trabajo

A partir de todos los análisis realizados, se decide proponer una infraestructura de trabajo que considera el uso de placa Micro:Bit que tendrá conectados los sensores responsables de tomar los datos, una placa ESP para comunicación vía WiFi y un servidor Rest para recibir y almacenar los datos y luego poder visualizarlos desde el mismo. El esquema de propuesta de solución se muestra en la *Figura 26*.

Figura 26: Esquema de propuesta de solución [Fuente: Ilustración propia]

Como se mencionó en la sección *4.7 Conclusiones*, después de realizar las pruebas con Micro:Bit, Arduino y ESP se decidió tomar como base de hardware para el IRID a la placa Micro:Bit en conjunto con la placa ESP.

Esta decisión se tomó debido a que Micro:Bit posee el entorno de programación oficial más adecuado para el propósito propuesto por este trabajo y no presenta ninguna desventaja significativa en cuanto a capacidades de hardware frente al Arduino. Como se mencionó en la sección *2.3 Comparativa de entornos de programación*, MakeCode, el

entorno de Micro:Bit, soporta la programación en bloques, lo que lo hace un excelente candidato para el uso en entornos educativos con personas sin conocimientos avanzados en ciencias informáticas.

El uso de la ESP como módulo WiFi demostró ser la alternativa más viable debido a su amplia disponibilidad, precio, cantidad de documentación y su gran comunidad de usuarios que le dan soporte mediante proyectos open source y foros.

5.2 Propuesta de un Protocolo de Comunicación entre Micro:Bit y ESP

Ante la problemática de conectar las placas Micro:Bit y ESP surge la necesidad de generar un protocolo de comunicación que dé soporte a esa comunicación. En esta sección se explica el desarrollo y funcionamiento de dicho protocolo.

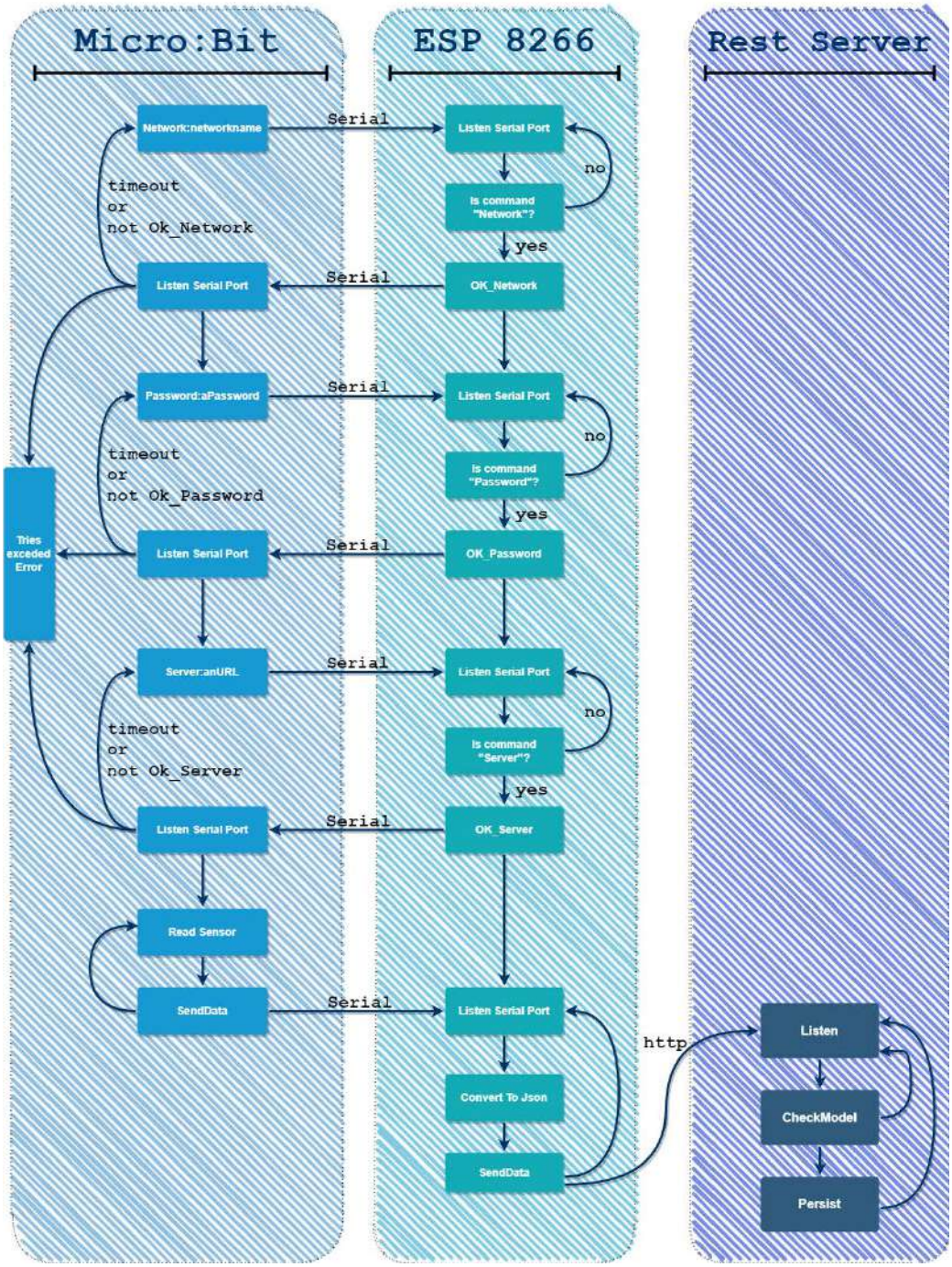


Figura 27: Protocolo de comunicación

[Fuente: Ilustración propia]

El obstáculo que da inicio al desarrollo del protocolo es la configuración los parámetros de red y del servidor. Hasta ese momento para modificar dichos parámetros

debía editarse el código de la ESP en el entorno de Arduino. Se decidió mover esa configuración a la Micro:Bit y se creó un protocolo de comunicación entre ambas placas para hacer posible dicha configuración y diferenciarla de la etapa del envío de datos, como se muestra en la *Figura 27*. Además de permitir la comunicación entre las placas y el servidor, la definición de este protocolo permite desacoplar los componentes del sistema IRID y reemplazarlos en caso de ser necesario. Por ejemplo se podría reemplazar la Micro:Bit por una placa Arduino siempre que esta última genere los mensajes definidos en el protocolo. Lo mismo aplica para la ESP o el servidor REST.

Este protocolo surge de la necesidad de organizar la manera en la que se comunican los distintos componentes del sistema, fue evolucionando a partir de la implementación de las primeras mejoras en el código y da una base sólida para futuras mejoras.

5.2.1 Consideraciones sobre el protocolo de comunicación

El protocolo expuesto en la *Figura 28* es el resultado de varias iteraciones de desarrollo e implementación de software para las placas Micro:Bit, ESP y para el servidor REST, tratando de facilitar el uso del entorno y desacoplar sus componentes en cada una de dichas iteraciones.

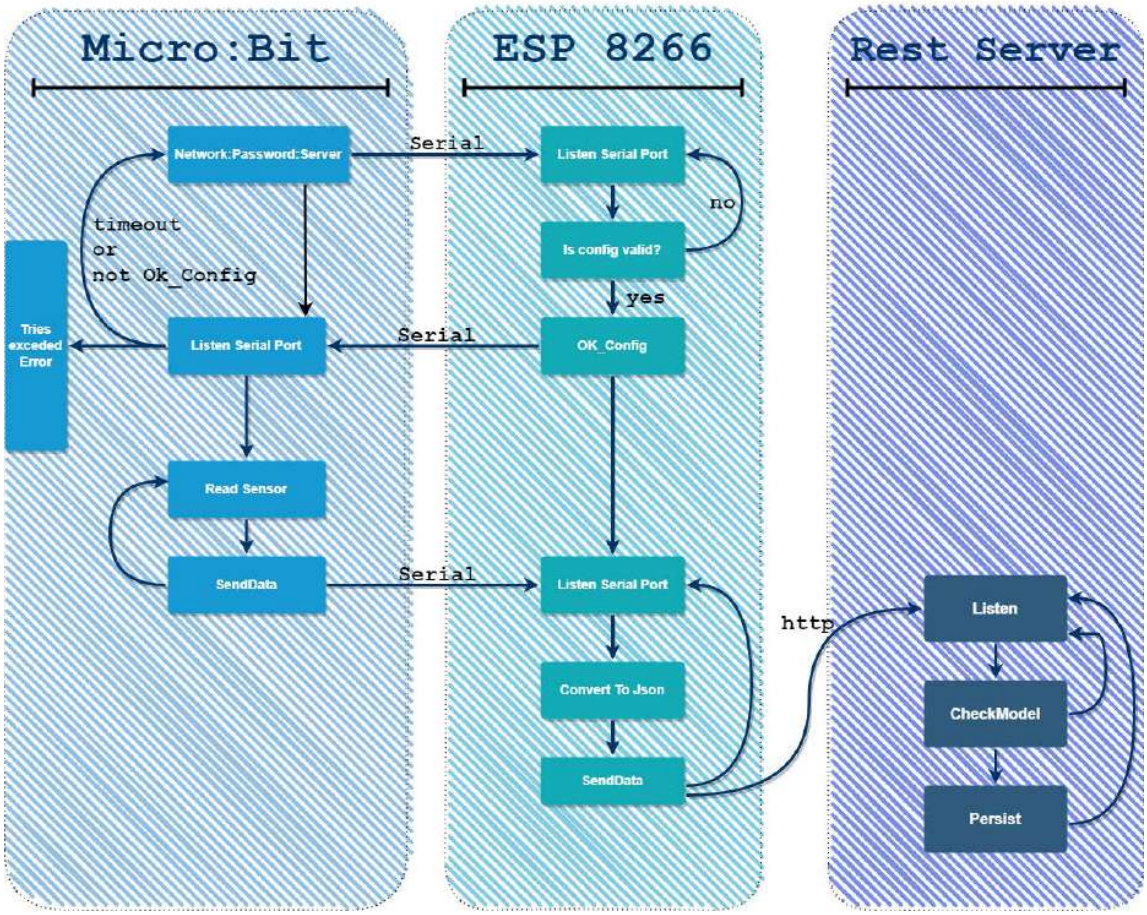


Figura 28: Protocolo de comunicación reducido

[Fuente: Ilustración propia]

Durante la etapa de configuración hay retroalimentación entre la Micro:Bit y la ESP para garantizar que los parámetros sean correcto y este todo en condiciones de funcionar. Para la comunicación serie entre las placas se utiliza un mecanismo de entrega de paquetes de "mejor esfuerzo", parecido al utilizado en el protocolo IP, en el cual el emisor envía los datos y se desentiende de si llegan o no. Para la comunicación WiFi entre la ESP y el server REST se utiliza un método HTTP POST estándar pero no se chequea si el paquete llega a destino, aunque el protocolo lo permita. La utilización de un mecanismo que controle el envío de datos vía serie y WiFi cargaría el hardware limitado del que disponemos y agregaría complejidad al código.

Un tema que se decidió dejar fuera del alcance de la tesis es lo relacionado a la seguridad. Al estar orientado a un entorno educativo tener conexiones no cifradas podría ayudar a ver cómo se realizan las mismas y entender un poco más el trasfondo de la tecnología subyacente. Por el mismo motivo se entiende que no deberían transmitirse datos sensibles usando esta herramienta, por lo que la seguridad queda fuera del alcance de este trabajo.

Junto al aumento de la simplicidad, la otra ventaja que aporta la implementación de este protocolo es el desacoplamiento de componentes. Esto hace que sea posible intercambiar la Micro:Bit por una gran variedad de placas con capacidad de comunicarse vía serie que le envíen los datos en el formato que espera la ESP, como pueden ser las placas de la familia Arduino o Raspberry. Del otro lado, al utilizar REST, es posible reemplazar el servidor Python por uno en Java, .Net o cualquier otra tecnología que soporte REST, del mismo modo que se puede reemplazar la base de datos donde se persisten las lecturas.

5.3 Propuesta de Bloques Personalizados considerando el Protocolo propuesto

Otro problema detectado, es la eventual complejidad para usuarios finales al momento de configurar parámetros necesarios para el sensado mencionado en el capítulo *6.1 Generación de código en bloques en MakeCode* Debido a eso se decide simplificar el uso de la herramienta mediante la creación de nuevos bloques en MakeCode tanto para la configuración como para el sensado. Como se puede apreciar en la *Figura 29* se creó un bloque llamado “setup esp” que agrupa todos los parámetros de configuración de red y servidor y otro bloque llamado “send data”, mostrado en la *Figura 30* , que permite el envío de datos desde cualquier sensor conectado, identificando el origen, qué se está sensando y el valor del dato en cuestión.

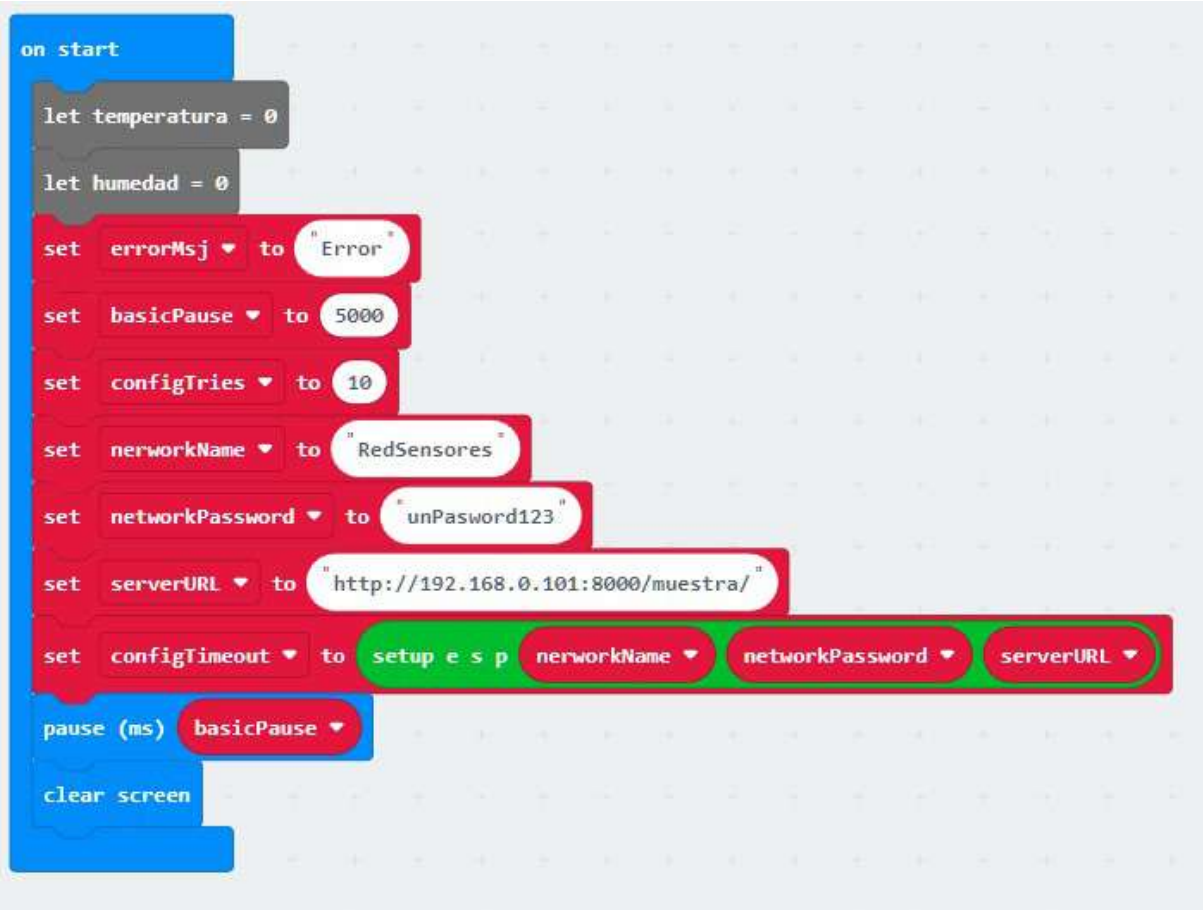


Figura 29: Creación de bloques de código de configuración

[Fuente: Captura propia]

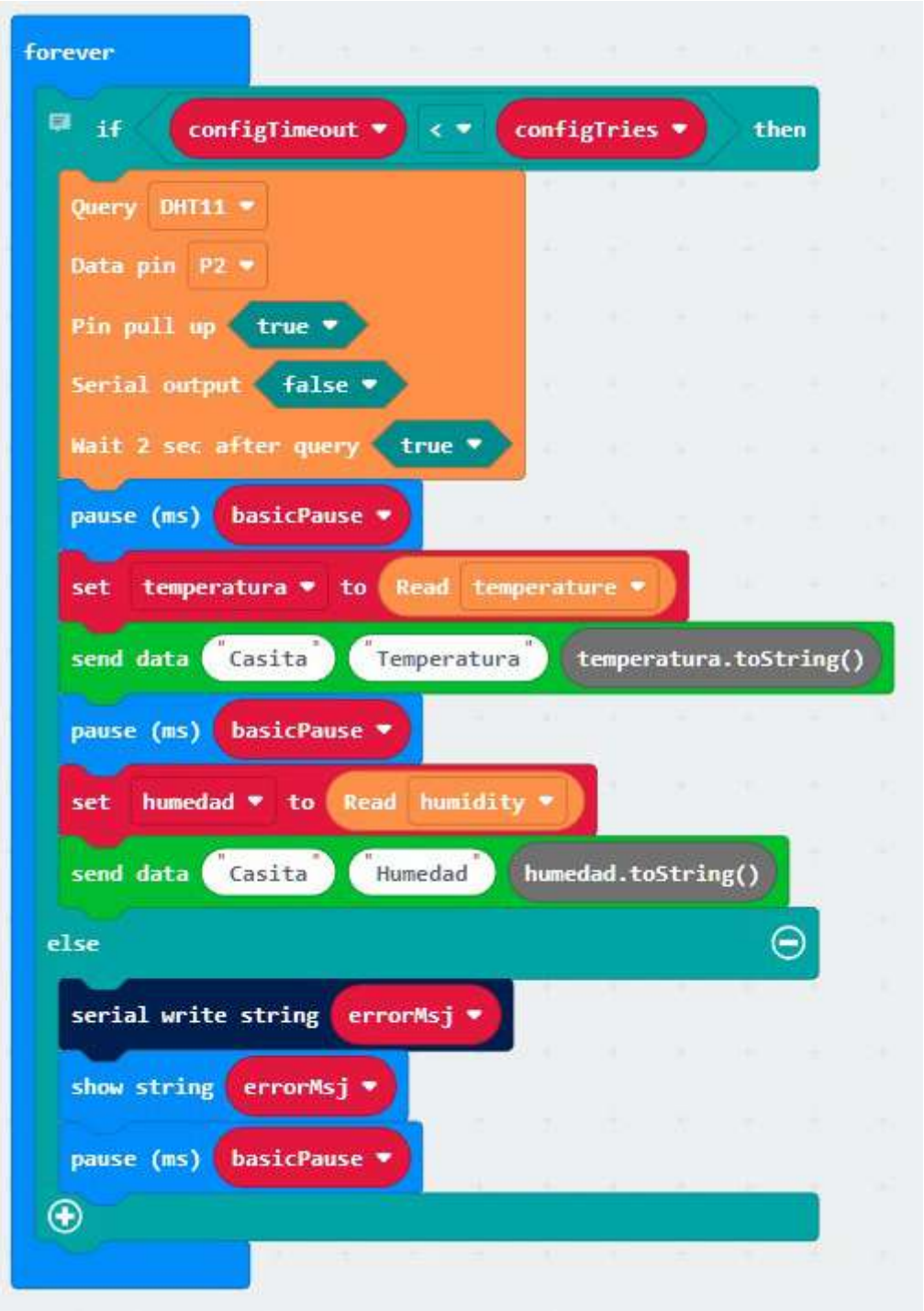


Figura 30: Creación de bloques de código de envío de datos

[Fuente: Captura propia]

La generación de estos bloques permite acercar el objetivo de simplificar lo más posible el uso de la herramienta, permitiendo configurar todos los parámetros del entorno en un solo bloque y enviar los datos de un determinado sensor también en un solo bloque. Quedando solamente por afuera configuraciones básicas como el intervalo entre lecturas y las configuraciones propias necesarias por cada sensor.

Luego de la creación de los nuevos bloques el protocolo de comunicación mencionado en la *Figura 27* se puede reducir a sus elementos básicos como lo muestra la *Figura 28*.

Capítulo 6: Implementación de componentes de la propuesta de solución

En este capítulo se presenta el proceso de implementación de los distintos componentes de la infraestructura propuesta (*Figura 31*), que se dio en paralelo a la creación y refinamiento del protocolo de comunicación (presentado en el *Capítulo 5: Propuesta de Solución*). Ambas actividades fueron evolucionando iterativamente y retroalimentándose a medida que avanzaba el proyecto, siendo el desarrollo de cada una de ellas indispensable para el desarrollo de la otra.

Este capítulo detalla las distintas etapas de implementación de:

- El código en bloques que ejecuta la Micro:Bit para el sensado y envío de datos.
- El código ejecutado por la ESP para la comunicación con la Micro:Bit y el envío de datos vía WiFi.
- La implementación del servidor REST encargado de la recepción, validación y persistencia de los datos.
- La configuración de la plataforma de visualización de datos.
- Y por último la Dockerización de todos los servicios (server REST, BD, visualizador de datos).

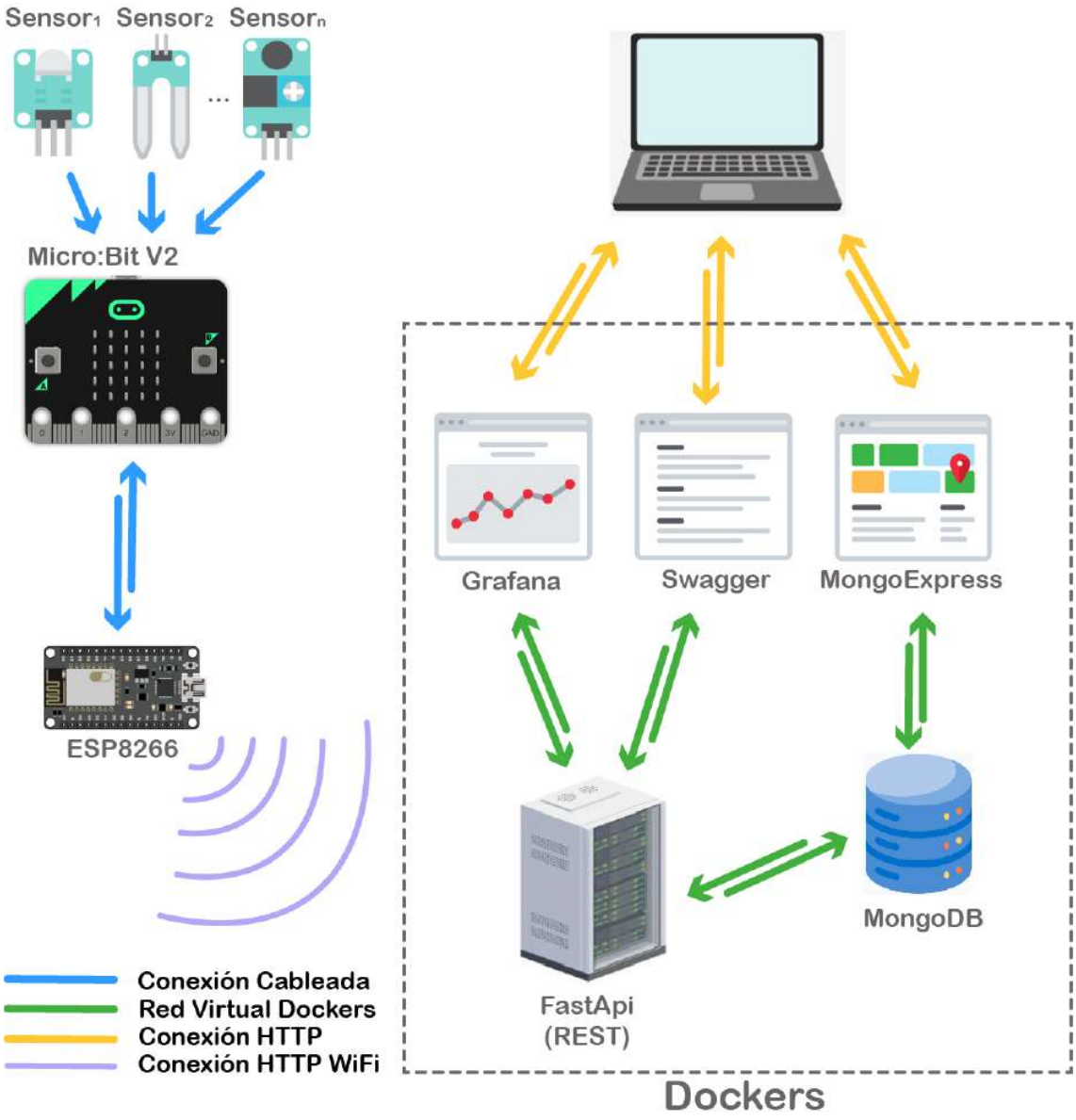


Figura 31: Diagrama de implementación

[Fuente: fuente propia]

Empezando desde arriba a la izquierda, la *Figura 31* presenta sensores de distintos tipos conectados a una placa Micro:Bit a través de una conexión cableada, por debajo se aprecia la Micro:Bit conectada a una placa ESP8266 también mediante conexión cableada, en este caso a través del puerto serie. En la comuna de la derecha podemos ver una computadora de un usuario final que se conecta a los distintos servicios brindados por IRID a través de una conexión HTTP. Por debajo se pueden ver todos los servicios

dockerizados del frontend: Grafana para la visualización de datos, Swagger y MongoExpress para el desarrollo, testeo y debugging del proyecto. Estos servicios del frontend se comunican a los servicios del backend, FastApi (REST) y MongoDB, para poder acceder a los datos mediante la red virtual de Dockers. Por último, podemos apreciar en el centro del gráfico como la placa ESP8266 se conecta mediante una conexión WiFi al servidor REST para enviarle los datos sensados.
El código fuente de las implementaciones realizadas para la presente tesina son de código abierto disponibles en github[47].

6.1 Generación de código en bloques en MakeCode

Tomando como base el código utilizado en el capítulo *4.5 Pruebas usando* ESP se procede a buscar la manera de simplificar la forma de configurar y programar la placa Micro:Bit de manera que sea lo más amigable posible para personas con pocos conocimientos informáticos.

Lo primero que se identifica como un punto de complejidad para usuarios con pocos conocimientos informáticos, factible de atacar, es la configuración de red en la ESP. En las primeras pruebas, tanto de la ESP sola como de la ESP junto a la Micro:Bit, la configuración de los parámetros de red estaba embebida dentro del código de la ESP. Bajo este escenario si se quería cambiar de red debía modificarse el código de la ESP, compilarlo y subirlo a la placa mediante un cable usb, algo sumamente impráctico, pero funcional para la realización de esas pruebas iniciales.

En la siguiente iteración se decidió modificar el código de la ESP para permitirle recibir los parámetros de red desde la Micro:Bit mediante el puerto serie. Esta modificación hizo que se tenga que cargar el código en la ESP solo una vez mejorado mucho la usabilidad y el flujo de trabajo, e incluso dando la posibilidad de entregar las placas con el código previamente cargado. Del lado de la Micro:Bit se creó dentro de MakeCode la función sendConfigParamerter que se utiliza llamándola tres veces para la configuración del nombre red, la contraseña y la url del server. Como se puede apreciar

[47] https://github.com/zeeno77/Tesina

en la *Figura 32* esta función posee código engorroso que debe ser adjuntado a cada proyecto que el usuario realice en MakeCode dificultando la legibilidad de las funciones principales.

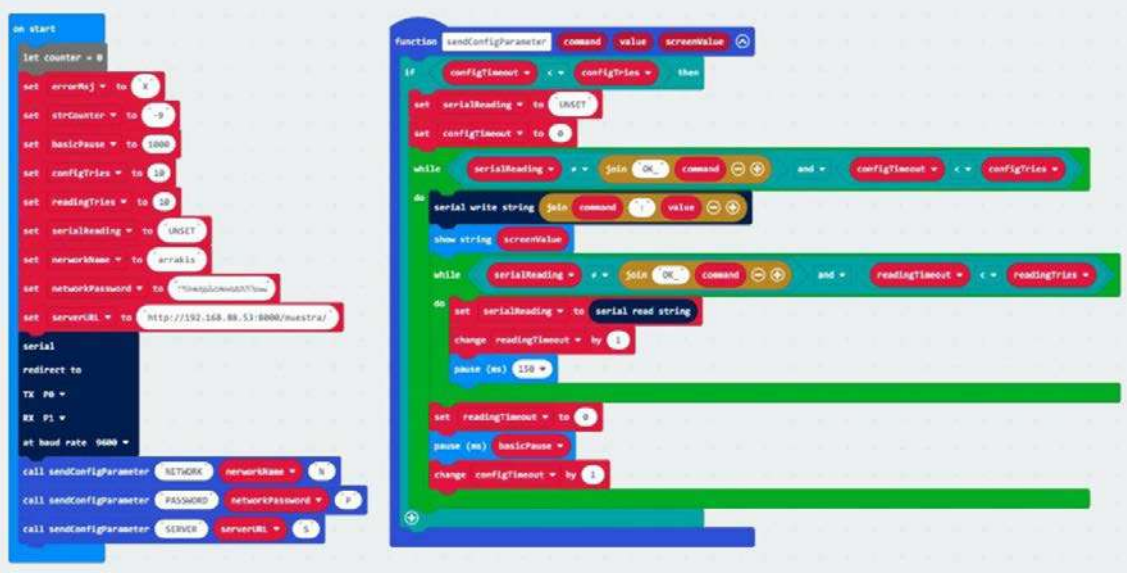


Figura 32: Bloques de configuración de parámetros de red – primera iteración
[Fuente: fuente propia]

En una siguiente iteración se decide abstraer la funcionalidad de sendConfigParamerter mediante la creación del bloque personalizado “setup_esp”. En la *Figura 33* se puede apreciar el bloque mencionado junto a los bloques “ping” y “send data” dentro de grupo personalizado “Irid”. A la derecha se puede ver el bloque “setup esp” en uso.

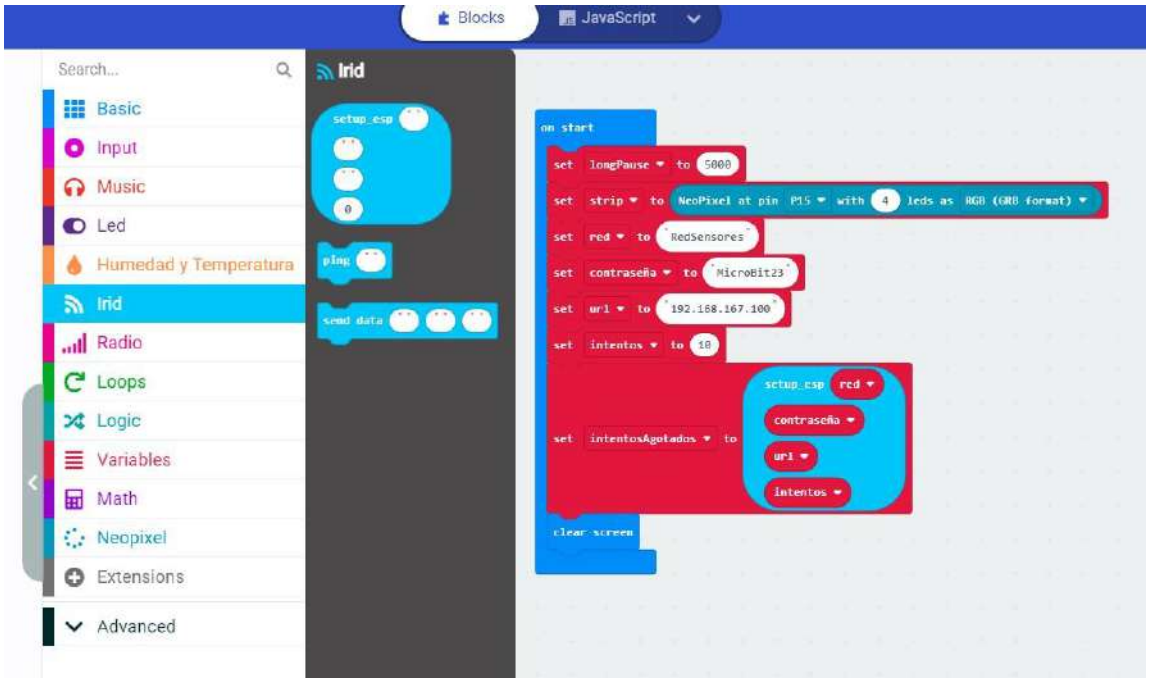


Figura 33: Bloques de configuración de parámetros de red – segunda iteración

[Fuente: fuente propia]

La creación de estos tipos de bloques especiales se hace mediante la edición de un archivo JavaScript titulado “custom.ts” dentro del mismo MakeCode. Este archivo se crea por defecto al momento de la creación de los proyectos y se accede mediante la pestaña de modo JavaScript. En la *Figura 34* se puede apreciar la implementación del grupo y los bloques antes mencionados. Aparte de la funcionalidad propia de cada bloque también se pueden definir dentro del archivo el nombre, icono y color del grupo, se pueden hacer comentarios y definir ayudas contextuales que se despliegan al pasar el cursor sobre el bloque, *Figura 35*.

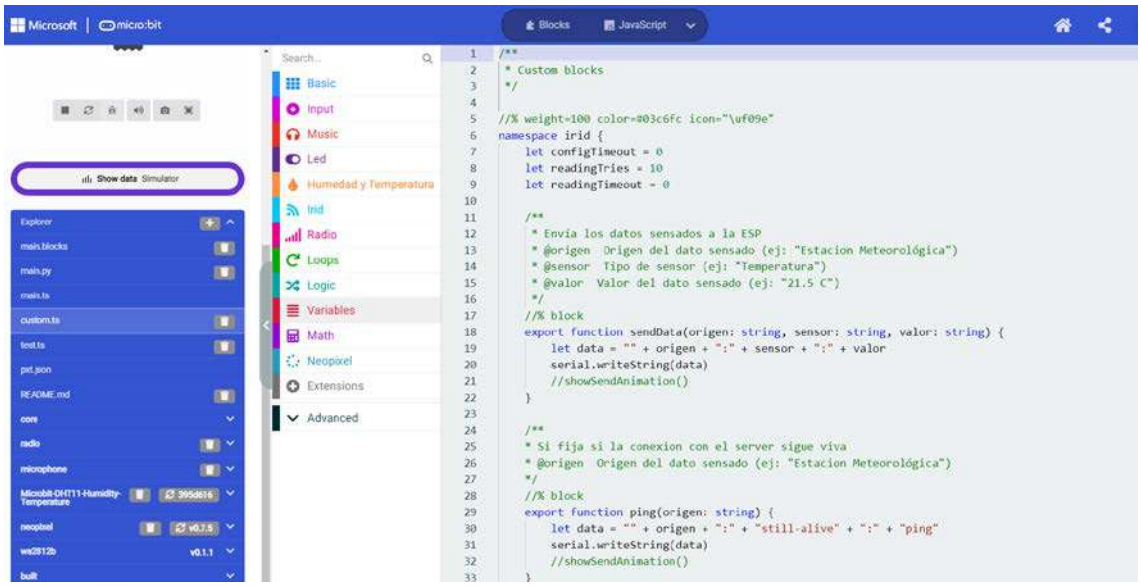


Figura 34: Implementación de bloques personalizados en custom.ts

[Fuente: fuente propia]

Figura 35: Ayuda contextual de un bloque personalizado

[Fuente: fuente propia]

El bloque “send data” se basa en una función con el mismo nombre utilizada para enviar los datos recolectados por los sensores conectados a la Micro:Bit a la ESP mediante el puerto serie.

La principal ventaja de la implementación de bloques personalizados es la abstracción del código ajeno a lo que el usuario quiere implementar, en este caso sensado y envío de datos.

Tomando como ejemplo el bloque “setup_esp” se puede ver cómo se pasa de tener alrededor de 20 líneas de código dedicadas exclusivamente a configurar los parámetros de red y del servidor dentro del programa principal a solo tener una línea y un par de variables. Esto hace que sea mucho más fácil enfocarse en la actividad principal al no tener código que ensucie innecesariamente el entorno.

Otra ventaja es la posibilidad de guardar los bloques personalizados en un repositorio de GitHub e importarlos para su reutilización desde cualquier proyecto de MakeCode.

6.2 Implementación en la ESP

Para la ESP se desarrolló un código capaz de entender los comandos enviados desde la Micro:Bit para configurar la red, el servidor de destino y los datos sensados. Dicho código debe ser subido por única vez desde el IDE de Arduino (*Figura 36*). Este proceso esta detallado en el *Anexo I: Guía de Configuración de la ESP*.

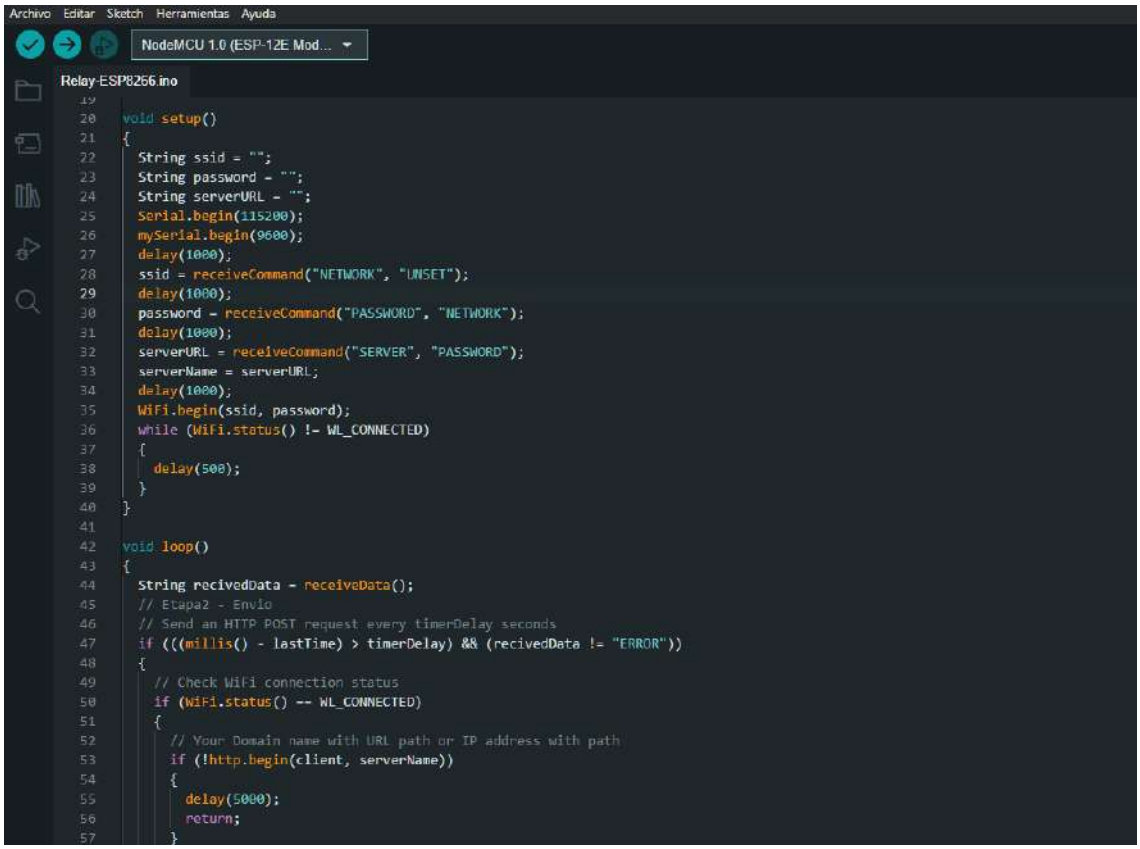


Figura 36: Código de la ESP en el IDE de Arduino

[Fuente: fuente propia]

6.3 Implementación del Servidor REST

Para la implementación del servidor encargado de recibir los datos enviados por las placas se decidió utilizar Python con el framework FastApi. La selección de estas tecnologías se hizo principalmente por haber tenido muy buenos resultados utilizándolas para la implementación de un sistema en un trabajo previo.

FastApi permite crear interfaces REST a partir de modelos de datos de manera eficiente e intuitiva, se encarga de la conversión de objetos Python a Json y viceversa de manera transparente y genera documentación automática en Swagger. Como se puede apreciar en la *Figura 37* y en la *Figura 38*, el código resultante es muy simple, legible y carece de conversión de tipos o anotaciones presentes en otras tecnologías.

```python
@muestraRouter.get("/muestras/", response_description="List all muestras")
async def list_muestras(request: Request):
    muestras = []
    async for doc in request.app.mongodb["muestras"].find().sort("fecha", -1): # -1 = Descending
        muestras.append(doc)
    return muestras


@muestraRouter.get("/muestras/{id}", response_description="Get a single muestra")
async def show_muestra(id: str, request: Request):
    if (muestra := await request.app.mongodb["muestras"].find_one({"_id": id})) is not None:
        return muestra

    raise HTTPException(status_code=404, detail=f"Muestra {id} not found")
```

Figura 37: Controller implementado en FastApi

[Fuente: fuente propia]

```python
class MuestraModel(BaseModel):
    id: str = Field(default_factory=uuid.uuid4, alias="_id")
    fecha : datetime = Field(default_factory=datetime.now)
    origen: str = Field(...)
    sensor: str = Field(...)
    valor: str = Field(...)

    ##Example for FastApi
    class Config:
        allow_population_by_field_name = True
        schema_extra = {
            "example": {
                "id": "00010203-0405-0607-0809-0a0b0c0d0e0f",
                "origen": "deDondeViene",
                "sensor": "queSensorEs",
                "valor": "unValor"
            }
        }
```

Figura 38: Modelo de datos

[Fuente: fuente propia]

6.4 Implementación de la visualización

En una primera instancia se analizó la posibilidad de implementar la visualización de datos mediante alguna librería en JavaScript pero al analizar las opciones existentes, esta idea se descartó y se optó por utilizar Grafana.

Grafana es un framework para el manejo y visualización de datos capaz de consumir los mismos mediante una gran variedad de protocolos, tratarlos y mostrarlos utilizando distintos tipos de gráficos.

Para este trabajo se configuro un servidor Grafana local para que consuma los datos sensados mediante REST y muestre distintos gráficos mediante una interfaz web. Como se puede apreciar en la *Figura 39* un mismo tipo de dato puede ser visualizado de varias maneras de manera clara y sencilla.

Figura 39: Gráficos IRID en Grafana

[Fuente: fuente propia]

La configuración del servidor Grafana se realiza mediante el seteo del endpoint desde donde se van a consumir los datos y la posterior configuración de qué tipo de gráficos se van a generar para ese dato en particular. Dentro de la configuración de los gráficos hay una gran cantidad de parámetros para personalizarlos como el tamaño, la disposición, el tipo de gráfico y distintos colores para indicar valores bajos, normales o altos, entre otros.

6.5 Dockerización

Teniendo el entorno andando y testeado se manifestó el inconveniente de tener que levantar cada uno de los servicios a mano cada vez que se quería utilizar el sistema, por lo que se decidió automatizarlo usando Dockers. Para esto se creó un DockerCompose que reúne el servidor REST, la base de datos MongoDB y Grafana bajo un mismo paraguas y permite ejecutarlos a todos juntos bajo un solo comando. Otras ventajas colaterales de la Dockerización son la resolución automática de las dependencias de software necesarias para ejecutar cada uno de los servicios mencionados y la independencia del sistema operativo donde se ejecute, la misma configuración de Dockers sirve para hacer el deploy en Windows, Linux o Mac.

Capítulo 7: Caso de Estudio

Se decidió armar una maqueta especial de trabajo para el caso de estudio, y en consecuencia, crear bloques específicos para esta maqueta junto a la ayuda dentro de MakeCode para facilitar el posterior uso en el aula.

A continuación, se describe la maqueta de trabajo, los bloques personalizados para la maqueta y la diagramación específica del caso de estudio

7.1 Maqueta de trabajo

Para un uso más motivador de la actividad, se decidió utilizar y armar un kit comercial de una maqueta en forma de casa, en la que se encuentran conectados diversos sensores (ej. Temperatura, humedad y gas) y actuadores (ej. luces led, motor que permite abrir puertas y un motor con un ventilador). A esta maqueta la llamamos “Casita”. La ***¡Error! No se encuentra el origen de la referencia.*** muestra la maqueta armada.

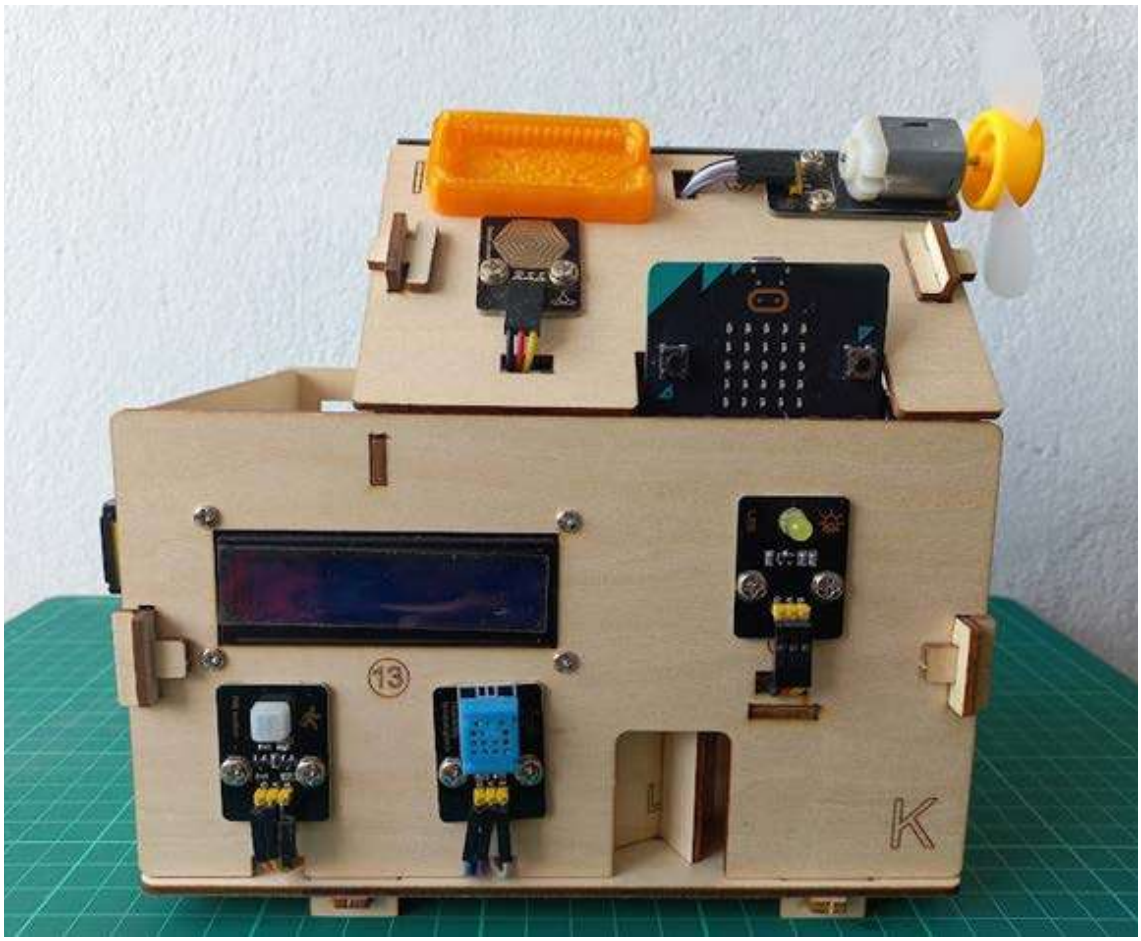


Figura 40: Maqueta de la "Casita"

[Fuente: Imagen propia]

7.2 Personalización de bloques para la maqueta

En base a la maqueta, y usando la misma metodología para crear la extensión propuesta en la sección *5.3 Propuesta de Bloques Personalizados considerando el Protocolo propuesto,* se creó una nueva familia de bloques llamado "*Casita*" en el que se sumaron los bloques a usar de la familia "*IRID*" y se creó un bloque especial para que puedan sumarse comentarios. A esta nueva familia de bloques se le configuró la ayuda que permite MakeCode para que, al usar la maqueta, los estudiantes pudieran consultar en que Pin estaba conectado cada sensor o actuador. Lo que haría que funcionara como un kit. La nueva familia de bloques y la ayuda desplegada se visualizan en la *Figura 41*.

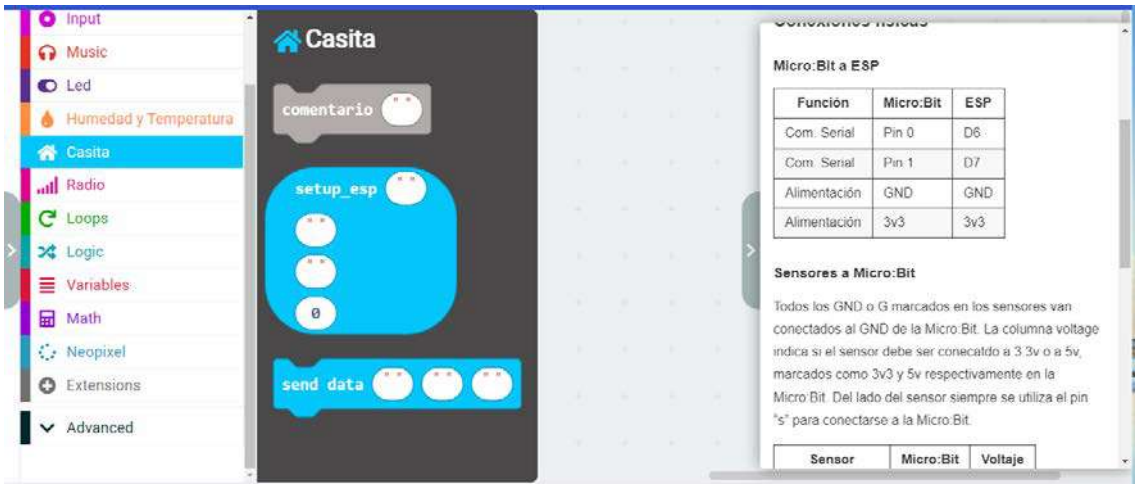


Figura 41: Familia de bloques para trabajar con la "Casita"

[Fuente: fuente propia]

7.3 Guías de configuraciones

Dos guías fueron creadas para facilitar el despliegue de todos los componentes técnicos de software del proyecto.

El *Anexo I: Guía de Configuración de la ESP* detalla los pasos necesarios para la carga del software necesario para el funcionamiento de la placa WiFi ESP.

El *Anexo II: Instructivo para el despliegue de servicios* describe los pasos para el despliegue desde cero del servidor REST, la base de datos y la configuración de Grafana para la visualización de datos.

7.4 Diagramación general del caso de estudio

El diseño del caso de estudio se diseñó para estudiantes de cuarto año de una escuela secundaria técnica, con orientación en programación de la ciudad de La Plata, Buenos Aires, Argentina. Dada la disponibilidad ofrecida desde la escuela, se planificaron tres encuentros de dos horas treinta minutos cada uno. Considerando el uso de "la casita" introducida previamente

Los encuentros se programaron de la siguiente manera:

- En el primer encuentro, se presenta el entorno de programación MakeCode se introducen conceptos básicos de la programación en bloque usando la plataforma MakeCode para placas Micro:Bit. En este encuentro, se espera que los estudiantes adquieran habilidades básicas.
- El segundo encuentro, suma el uso de sensores externos. Se presenta el esquema de la infraestructura de trabajo y se abordan aspectos de comunicación de datos usando ESP8266, se envían datos a un servidor web usando la programación en bloques y finalmente se visualizan los datos generados por sensores.
- El tercer encuentro, retoma la actividad del encuentro anterior y se realiza una charla de cierre con los estudiantes a modo de relevamiento.

La guía de actividades para los estudiantes esta detallada en el *Anexo III: Guías para las actividades prácticas*

Los ejemplos y ejercicios se dejaron implementados y disponibles en una carpeta[48], a la que, en caso de ser requerido, se les daba acceso a los estudiantes.

7.5 Propuesta en acción

Para este caso de estudio, considerando la cantidad de encuentros disponibles, si bien se entregaron las guías para preconfiguración de la ESP (*Anexo I: Guía de Configuración de la ESP*) y para el despliegue de un servidor propio (*Anexo II: Instructivo para el despliegue de servicios*), se decidió llevar la placa ESP preconfigurada, y un servidor ya levantado para que pudiesen directamente acceder al mismo. Los tres encuentros se realizaron durante el mes de noviembre de 2023.

La cantidad de asistentes varió en cada encuentro, en el primer encuentro se presentaron 12 estudiantes y cuatro docentes, en el segundo 15 estudiantes y dos docentes y en el tercero 14 estudiantes tres docentes. Se sumó además de la docente responsable del aula, una docente de apoyo técnico, un docente que imparte tecnología informática en sexto año del secundario y un docente de otra comisión de cuarto año.

[48] https://drive.google.com/drive/folders/1XWY3Cx4_8Ahvg2UKsqT68Q1ZJ9lRSZc5?usp=drive_link

Varios de los estudiantes del curso tenían experiencia en desarrollo de placas Arduino y con lenguajes de programación textual. No todos los estudiantes poseían el mismo nivel de expertise en programación. Cuatro estudiantes, aun no programaban.

El aula disponía de mesas con pcs de escritorio y además se contaba con notebooks que la escuela recibió por el programa Conectar Igualdad[49] y luego con el plan Juana Manso[50]. Lo anterior permitía que cada estudiante dispusiera de un equipo para trabajar, sin embargo, al momento de desarrollar las prácticas, se permitió el trabajo en grupo, con agrupación natural por parte de los estudiantes.

A continuación, la *Figura 42* muestra la manera en la que se toman los datos del sensor de temperatura y se le envían a la ESP para que se encargue de transmitirlos al servidor usando el bloque "*send data*".

Figura 42: Ejemplo de envío de datos de temperatura
[Fuente: fuente propia]

Por último la *Figura 43* muestra la visualización de los datos cuando se accede desde un cliente al servidor.

[49] https://conectarigualdad.edu.ar/inicio
[50] https://www.educ.ar/noticias/200546/educacioacuten-presenta-la-nueva-versioacuten-de-huayra-50nbsp

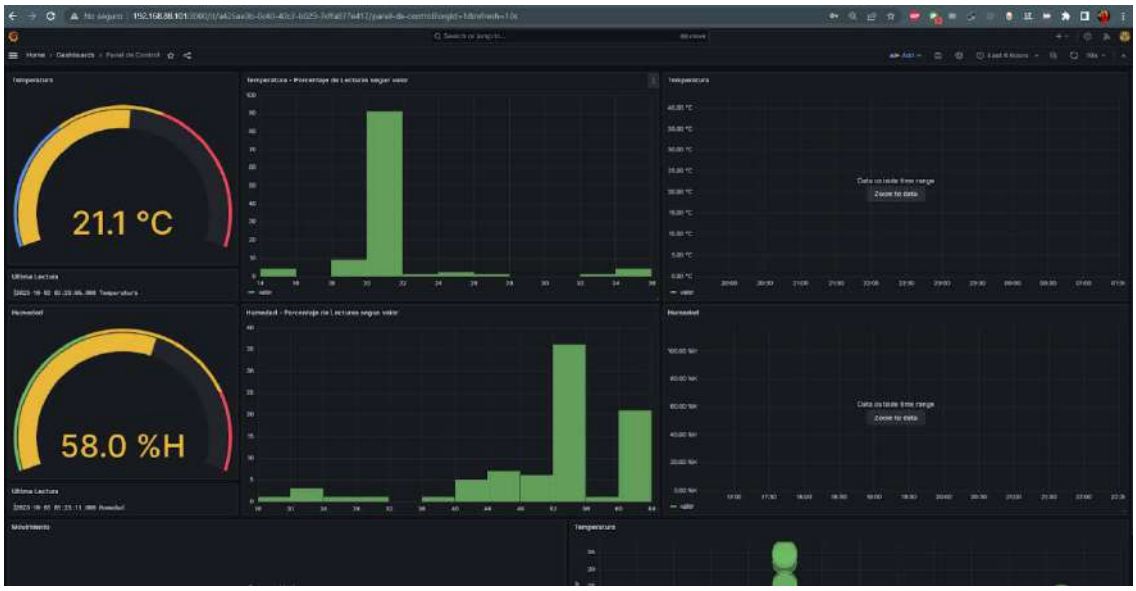


Figura 43: Visualización de los datos almacenados en el servidor

[Fuente: fuente propia]

7.6 Discusión

En esta primera validación de conceptos, el público destinatario estuvo conformado por estudiantes y si bien había docentes participando, éstos no guiaron los encuentros. El curso presentó algunas particularidades tales como la gran disparidad de conocimientos previos entre los estudiantes del mismo curso, así como la actitud frente a las actividades propuestas. En el segundo encuentro, debido a la falta de tiempo, no fue posible trabajar con la visualización de datos, motivo por el cual se retomó en el tercer encuentro.

De todos los estudiantes presentes, se mantuvo constante el trabajo solamente con siete de ellos, quienes realizaron las actividades propuestas con interés.

El relevamiento realizado durante el último encuentro con los siete estudiantes que trabajaron en la propuesta mostró su entusiasmo para este tipo de iniciativas quienes destacaron la facilidad de realizar la actividad con programación basada en bloques para la parte de conexión al servidor y el envío de datos. Esta apreciación se debe a que los

estudiantes previamente habían programado con la IDE de Arduino durante un taller de robótica educativa.

Capítulo 8: Conclusiones y Trabajos Futuros

En este trabajo se presentó una infraestructura que mediante la combinación de software y hardware permite, la recolección, y envío de datos inalámbricamente a un servidor desde el cual se puede realizar la visualización de los datos.

Luego de realizar pruebas sobre una serie de placas y analizar los trabajos relacionados, se estableció como escenario de trabajo el uso de placa Micro:Bit y ESP para conectividad WIFI. También se estableció el uso de servidor REST, el uso de programación basada en bloques y el entorno de programación MakeCode para placas Micro:Bit.

Se presentó e implementó un protocolo de comunicación entre la placa Micro:Bit y la ESP, que permite el empaquetamiento y envío de datos a un servidor REST.

Se implementó un servidor REST responsable de almacenar los datos recibidos y permitir su posterior visualización, mediante un visualizador provisto para tal fin.

Para bajar el nivel de complejidad que implica configurar la placa ESP con el servidor de destino, se definió e implementó una extensión para el entorno de programación basado en bloques MakeCode.

Se diseño un caso de estudio para poder ser realizado con estudiantes secundarios de nivel superior, en una escuela técnica de la ciudad de La Plata. Para ello se eligió y armó una maqueta con sensores y actuadores específicos. A partir de lo anterior, usando la extensión a MakeCode propuesta, se realizó una personalización de bloques a fin de que se brindara lo necesario para la maqueta definida. A esta personalización se le sumo la implementación de un tipo de bloque especial para trabajar comentarios y se configuró la ayuda del entorno para que se supiera en que pin estaba conectado cada sensor y actuador.

Se generó una guía detallada para que personas ajenas a esta tesina pudiesen configurar por única vez la placa ESP y para que se pudiese desplegar un servidor sin necesidad de usar el servidor del proyecto. Luego, se creó una guía de actividades para que los estudiantes usaran durante cada una de las clases planificadas y se les facilitó en una carpeta online las soluciones a las problemáticas propuestas.

Las pruebas realizadas con los estudiantes y docentes voluntarios resultaron relevantes para la obtención de feedback de la propuesta de esta tesina y considerar posibles trabajos a futuro.

Los trabajos a futuro pueden ser categorizados como a corto, mediano y largo plazo. En el corto plazo se espera mejorar las guías en base al feedback recibido durante el caso de estudio haciéndolas más comprensibles y completas, así como simplificar la configuración inicial de la placa ESP mediante la creación de un script o un ejecutable que permita realizarla de una manera semi automatizada. Una vez realizado lo anterior, se espera realizar nuevas pruebas con docentes y estudiantes.

A mediano plazo se espera ampliar la variedad de tipos de gráficos disponibles para la visualización de los datos, ya sea profundizando en la implementación con Grafana, buscando nuevas herramientas de visualización de datos o integrando este tipo de datos a la plataforma Alfadatizando [Lliteras et al., 2024_a]. Se espera proponer nuevas maquetas de trabajo que incentiven la utilización de la infraestructura y permitan generar nuevos kits de trabajo.

A largo plazo se espera proveer de guías que usen el protocolo de comunicación propuesto empleando placas Arduino. Así como, explorar el protocolo MQTT y generar una versión del código de la ESP que lo utilice. También se espera explorar el uso de Raspberry Pi, analizar aspectos de seguridad en el protocolo bajo la premisa de facilidad para usuarios no expertos, así como considerar aspectos de accesibilidad en la extensión analizando la posibilidad de incluir modos de alto contraste y asistencias visuales y auditivas.

Por último y no por ello menos importante, se deja constancia que, durante la evolución del presente trabajo, se participó en los Proyectos de Innovación y Aplicación con alumnos, en el año 2022 [Lliteras et al., 2022] y en el año 2023 [Lliteras et al., 2023].

Finalmente, en octubre de 2024, se presentaron durante el Congreso Argentino de Ciencias de la Computación (CACIC) dos artículos de la temática de esta tesina [Lozano Arce, 2024] [Lliteras et al., 2024_b].

Referencias

[Abichandani et al., 2022] Abichandani, P., Sivakumar, V., Lobo, D., Iaboni, C., & Shekhar, P. (2022). Internet-of-things curriculum, pedagogy, and assessment for stem education: A review of literature. *IEEE Access*, *10*, 38351-38369.

[Denning & Tedre, 2021] Denning, P. J., & Tedre, M. (2021). Computational thinking: A disciplinary perspective. Informatics in Education, 20(3), 361.

[Ezeamuzie & Leung, 2022] Ezeamuzie, N. O., & Leung, J. S. (2022). Computational thinking through an empirical lens: A systematic review of literature. Journal of Educational Computing Research, 60(2), 481-511.

[Fernández Quiñonez, 2022] Fernández Quiñonez, S. L. (2022). Implementación de sistemas iot utilizando técnicas de programación visual.

[Lliteras et al., 2022] Lliteras A., Grigera J., Garrido A., & Gardey J. (2022). Plataforma para la visualización de datos con fines eductivos en nivel secundario para ciencias sociales y humanidades. Proyecto de Desarrollo de Aplicaciones e Innovación con Alumnos. UNLP, Facultad de Informática. Resol. HCD Nro 51/22.

[Lliteras et al., 2023] Lliteras A., Grigera J., Garrido A., Gardey J., & Perez G. (2023). Nuevas tecnologías informáticas en el proceso de enseñanza -aprendizaje con datos. Proyecto de Desarrollo de Aplicaciones e Innovación con Alumnos. UNLP, Facultad de Informática. Resol. HCD Nro 91/23.

[Lliteras et al., 2024_a] Lliteras A., Artopoulos A., Ger J., & Boza G. (2024). Alfadatizando 2.0 applied to data visualization at high school level and for digital humanities. Empowering digital citizens. In: XIX Conferencia Latinoamericana de Tecnologías de Aprendizaje (LACLO) En prensa.

[Lliteras et al., 2024_b] Lliteras A., Lozano Arce, J.P, & Rodriguez A. (2024). Recolección, envío, almacenamiento y visualización de datos considerando programación basada en bloques de placas MicroBit, ESP, sensores y Wifi. In Congreso Argentino de Ciencias de la Computación (CACIC), XXIII WTIAE. En prensa.

[Lliteras, 2020] Lliteras A. (2020) Aprendo Sociales con Datos. Propuesta doctoral. UNLP. Facultad de Informática.

[Lozano Arce, 2024] Lozano Arce, J.P. (2024). IRID: Infraestructura para la recolección inalámbrica de datos provistos por sensores en el marco de robótica educativa en nivel secundario. In Congreso Argentino de Ciencias de la Computación (CACIC), III SPA. En prensa.

[Noh & Lee, 2020] Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. Educational technology research and development, 68(1), 463-484.

[Özkök, 2021] Ozkök, G. A. (2021). Fostering Computational Thinking Through Data Visualization and Design on Secondary School Students. J. Univers. Comput. Sci., 27(3), 285-302.

[Pech & Novák, 2020] Pech, J., & Novák, M. (2020). Use Arduino and micro: bit as teaching platform for the education programming and electronics on the stem basis. V International Conference on Information Technologies in Engineering Education (pp. 1-4). IEEE.

[Peech and Novák, 2020] Pech, J., & Novák, M. (2020, April). Use Arduino and micro: bit as teaching platform for the education programming and electronics on the stem basis. In *2020 V International Conference on Information Technologies in Engineering Education (Inforino)* (pp. 1-4). IEEE.

[Pradeep, 2023] Pradeep, A. (2023, August). Enabling IoTs with ESP32 for Affordable Education. In *2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA)* (pp. 1368-1373). IEEE.

[pxt-iot-environment-kit] Extensión pxt-iot-environment-kit. Disponible desde la plataforma MakeCode, accediendo a https://github.com/tinkertanker/pxt-iot-environment-kit/tree/master. Fecha de último acceso: 2024/12/01.

[Srinivasulu et al., 2020] Srinivasulu, S., Kavitha, & M., Kolli, C.S (2020). Wireless Technology over Internet of Things. In: Venkata Krishna, P., Obaidat, M. (eds) Emerging Research in Data Engineering Systems and Computer Communications. Advances in Intelligent Systems and Computing, vol 1054. Springer, Singapore. https://doi.org/10.1007/978-981-15-0135-7_54

[Sun et al., 2024] Sun, D., Looi, C. K., Li, Y., Zhu, C., Zhu, C., & Cheng, M. (2024). Block-based versus text-based programming: a comparison of learners' programming behaviors, computational thinking skills and attitudes toward programming. Educational technology research and development, 72(2), 1067-1089.

[Udvaros et al., 2023] Udvaros, J., Forman, N., & Avornicului, M. (2023). Developing computational thinking with microcontrollers in Education 4.0.

[Yim & Su, 2024] Yim, I. H. Y., & Su, J. (2024). Artificial intelligence (AI) learning tools in K-12 education: A scoping review. Journal of Computers in Education, 1-39.

Anexo I: Guía de Configuración de la ESP

Objetivo del documento

El objetivo de este documento es facilitar la programación de la placa ESP8266 en el IDE de Arduino con el código provisto para el funcionamiento en conjunto con la placa Micro:Bit. Esto se debe a que la placa ESP8266 viene desprovista de código y es necesario cargarle el código capaz de entender los mensajes enviados desde la Micro:Bit por el puerto serie. Por otro lado, el código cargado también proporciona la lógica necesaria para convertir los mensajes recibidos desde la Micro:Bit en formato json, crear los paquetes http y enviarlos al servidor correspondiente.

Instalación del IDE de Arduino

Consideraciones previas a la instalación:

- La versión 2 de Arduino para Windows solo funciona desde Windows 10 en adelante.
- En caso de tener un Windows más antiguo se pueden seguir los mismos pasos de esta guía con la versión 1.8.x de Arduino

Desde una computadora con sistema operativo windows y acceso a internet:

Ingresar a la página que se menciona a continuación para descargar Arduino en su equipo desde la opción "**Windows** Win 10 and newer, 64 bits" (Figura 1). Arduino 2.3.2 desde https://www.arduino.cc/en/software

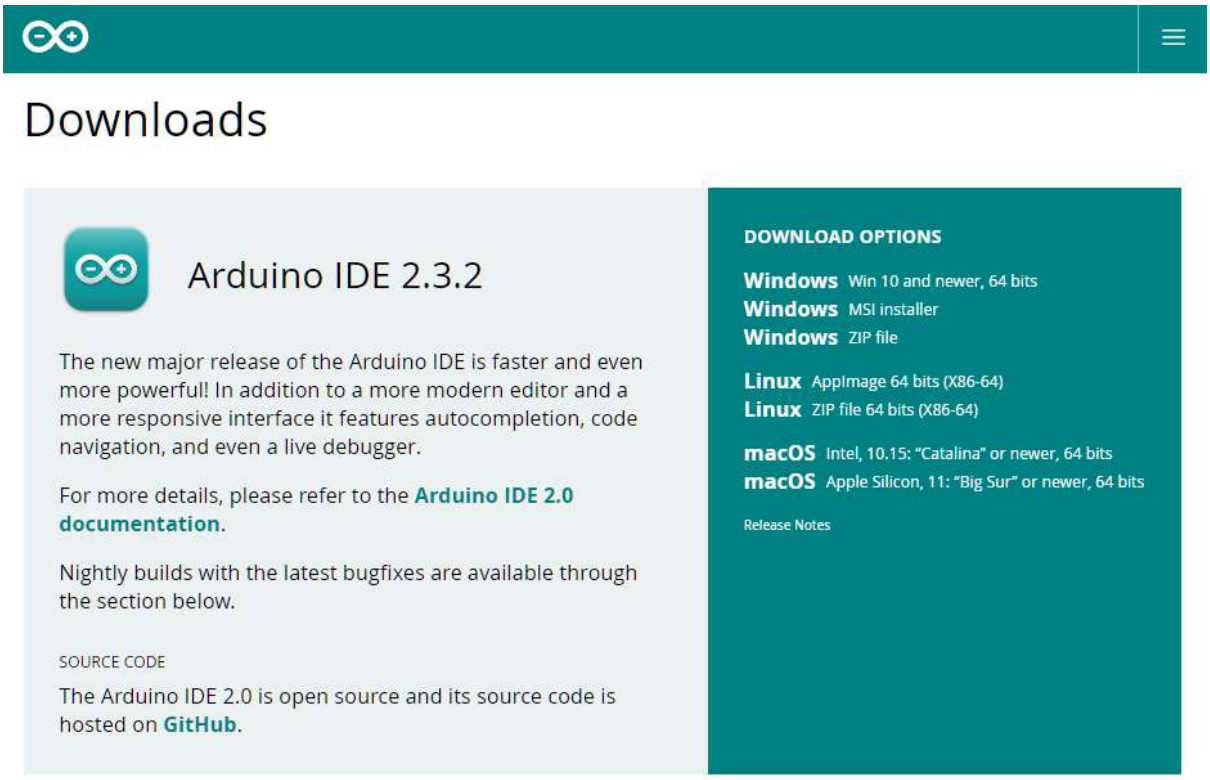


Figura 1: Página de descarga

[Fuente: fuente propia]

Luego Cliquear en "Just Download" como se muestra en la Figura 2

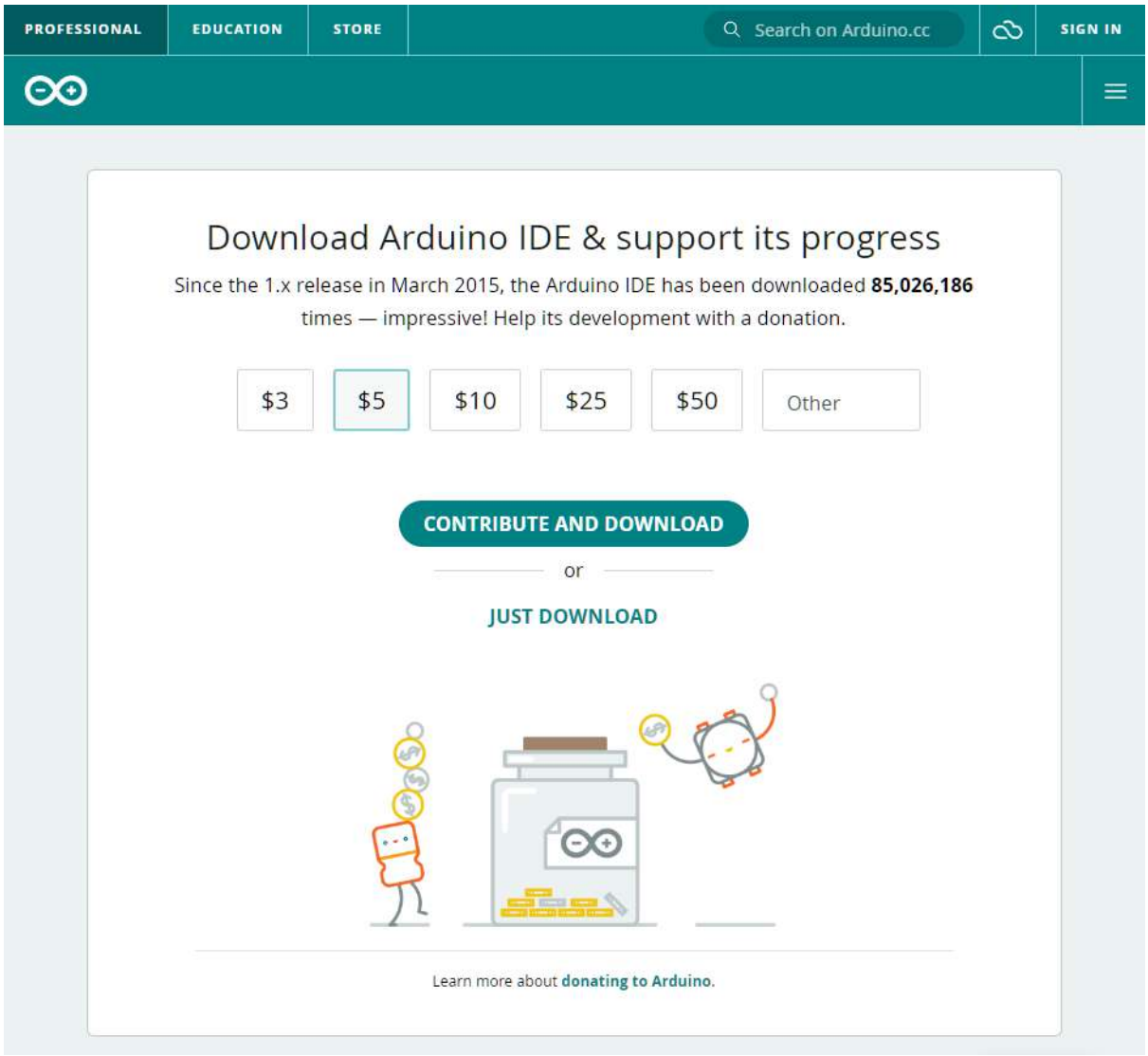


Figura 2: Página de descarga, segundo paso

[Fuente: fuente propia]

Deberá indicar dónde guardar el archivo que se descarga (Figura 3).

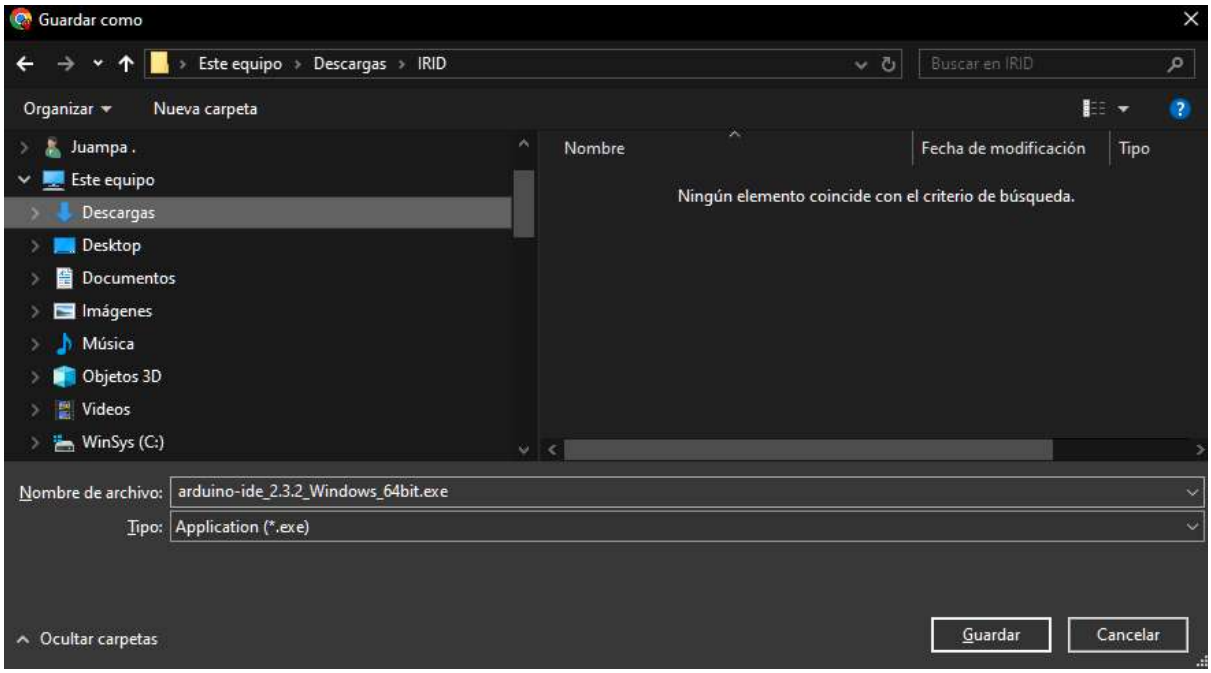


Figura 3: Guardar archivo

[Fuente: fuente propia]

Una vez descargado el archivo, deberá proceder a su instalación.

Haga doble clic sobre el archivo y siga los pasos propuestos por el asistente guiado. Se sugiere la instalación por defecto propuesta por el asistente.

Es posible que se visualice una pantalla de advertencia de seguridad (Figura 4).

Seleccionar la opción de Ejecutar.

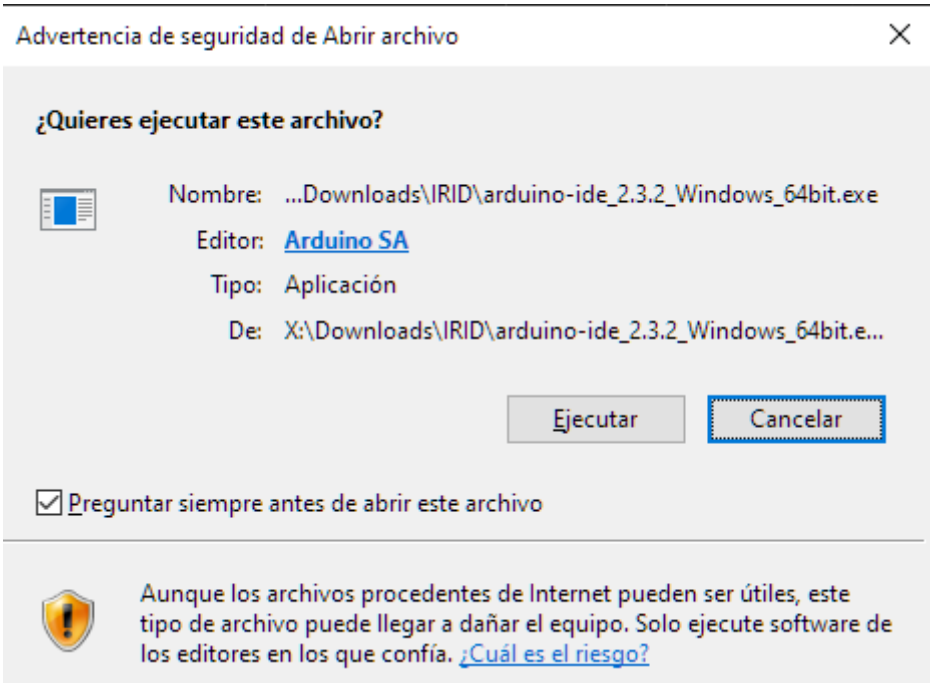


Figura 4: Advertencia de seguridad

[Fuente: fuente propia]

Instalación del plugin de Arduino para ESP8266

Bajar el archivo Relay-ESP8266.ino desde el siguiente repositorio de github, haciendo click en los tres puntos de la parte superior derecha y luego en Download, como lo muestra la Figura 5

https://github.com/zeeno77/Tesina/blob/main/placas/microbitConESP/Relay-ESP8266/Relay-ESP8266.ino

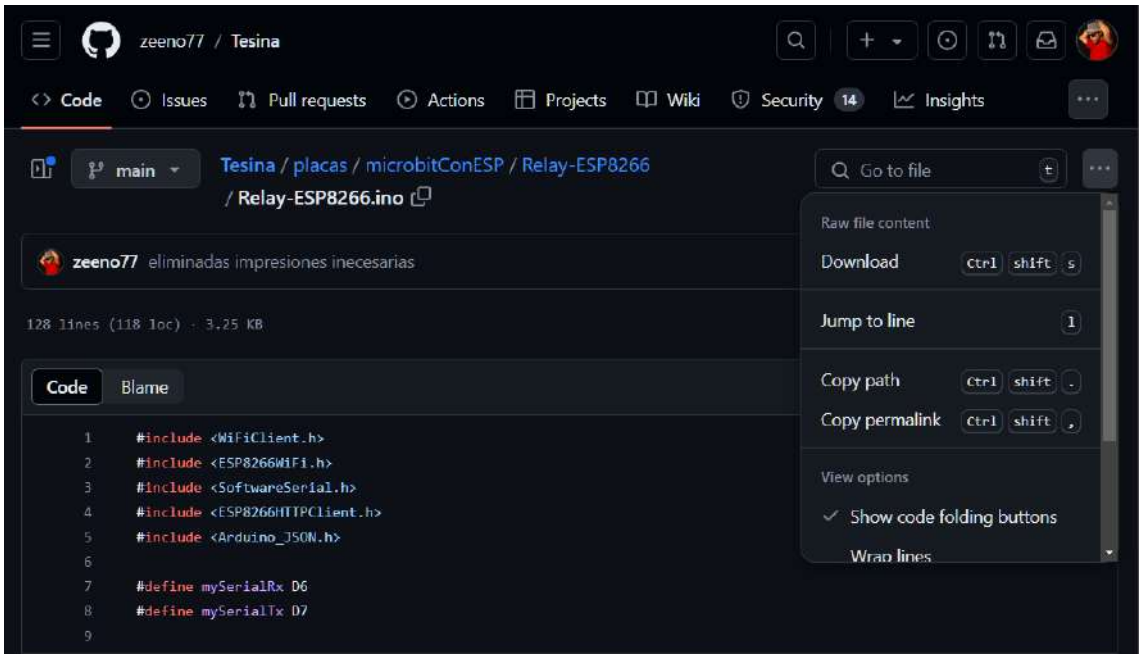


Figura 5: Repositorio github

[Fuente: fuente propia]

Una vez descargado deberá navegar hasta la carpeta donde guardó el archivo y hacer doble click sobre el mismo. Si tiene correctamente instalado el IDE de Arduino el archivo se abrirá dentro del mismo.

Dentro del IDE de Arduino navegar hasta Archivo -> Preferencias como muestra la Figura 6

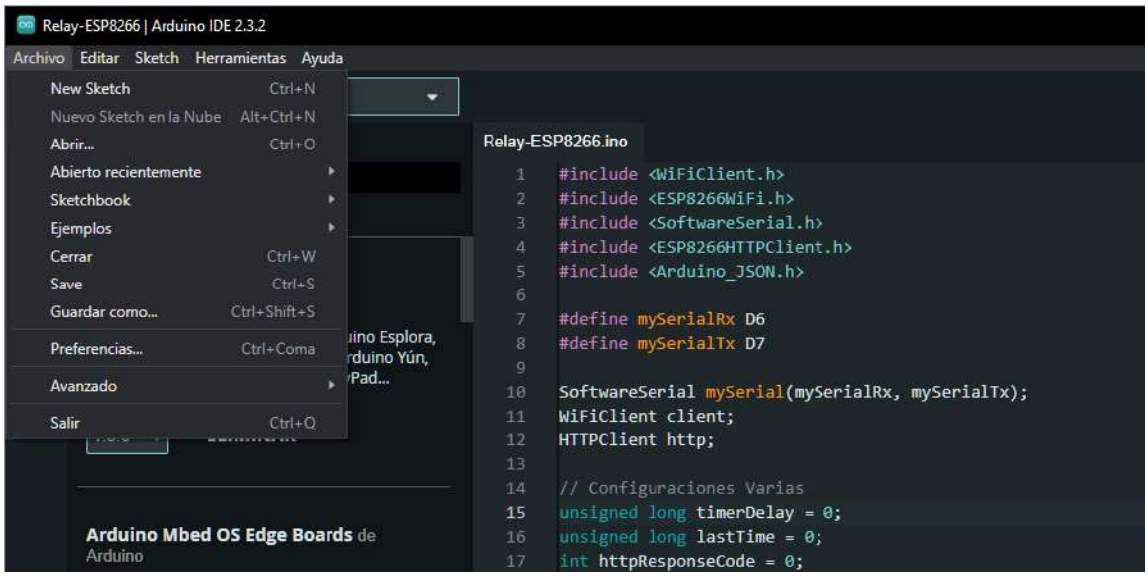


Figura 6: Arduino IDE

[Fuente: fuente propia]

Dentro de la pantalla de preferencias agregar lo siguiente dentro de “URLs adicionales de gestor de placas” como se muestra en la Figura 7:

https://arduino.esp8266.com/stable/package_esp8266com_index.json

y tocar aceptar

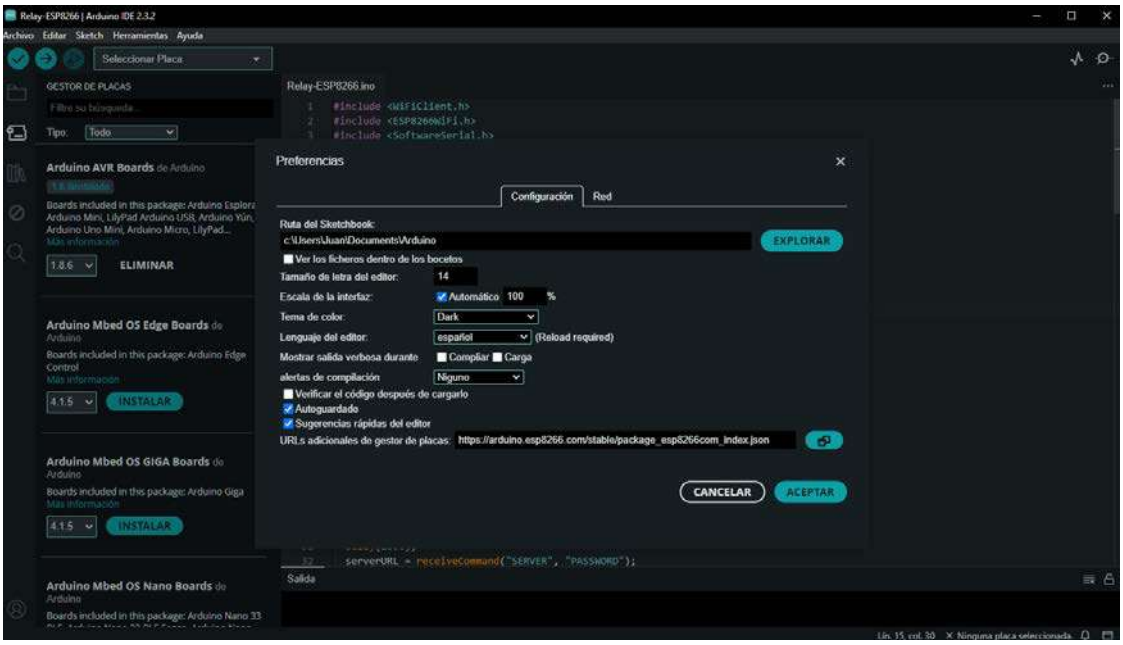


Figura 7: Arduino IDE – Preferencias

[Fuente: fuente propia]

De vuelta en la pantalla principal, navegar hasta Herramientas -> Placas -> esp8266 -> NodeMCU 1.0 (ESP-12E Module) como se muestra en la Figura 8

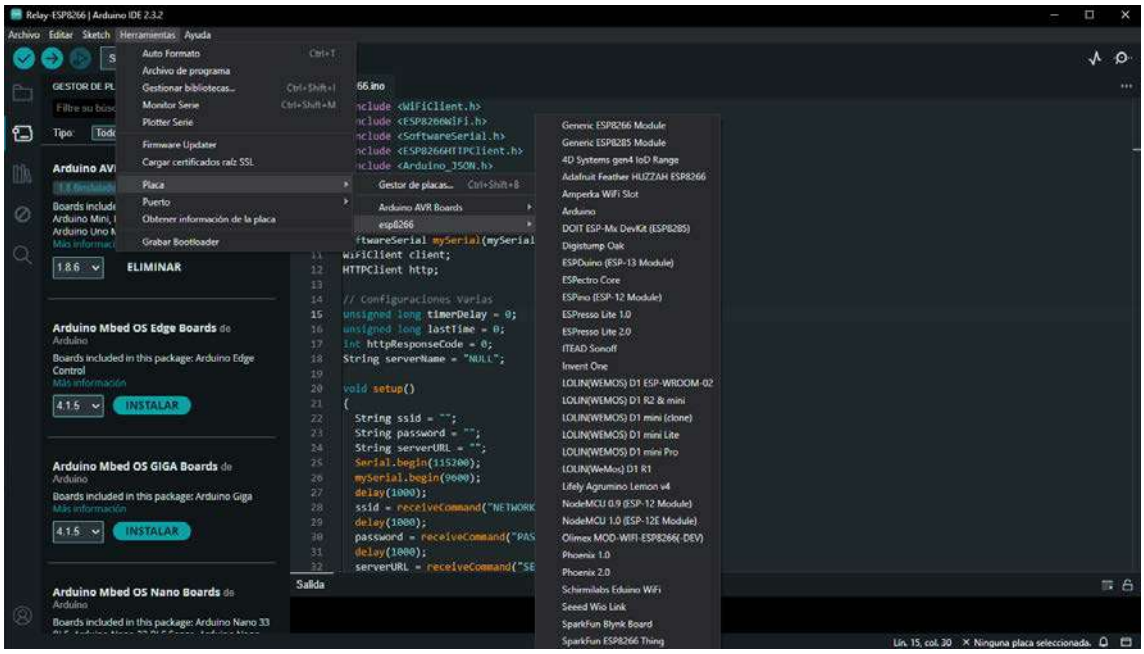


Figura 8: Arduino IDE - Selección de placa

[Fuente: fuente propia]

Carga del código en la ESP

Conectar la placa ESP8266 vía USB y tocar el botón verde con una flecha apuntando (cargar) a la derecha ubicado en la parte superior izquierda de la pantalla. Esto hará que se cargue el código en la placa y quede lista para su utilización con la Micro:Bit.

Anexo II: Instructivo para el despliegue de servicios

Objetivo del documento

El objetivo de este documento es facilitar la personalización en la instalación del servidor Para ello se presenta por un lado la instalación de Docker y por otro lado su despliegue.

Instalación de Dockers

Consideraciones previas a la instalación:

- Dockers no funciona en la versión Windows Home.
- Asegurarse de tener habilitada la virtualización desde el Bios.
- Previo a la instalación actualizar Windows.

Desde una computadora con sistema operativo windows y acceso a internet:

Ingresar a la página que se menciona a continuación (Figura 1) para descargar Docker en su equipo.

Docker Desktop desde https://docs.docker.com/desktop/install/windows-install/

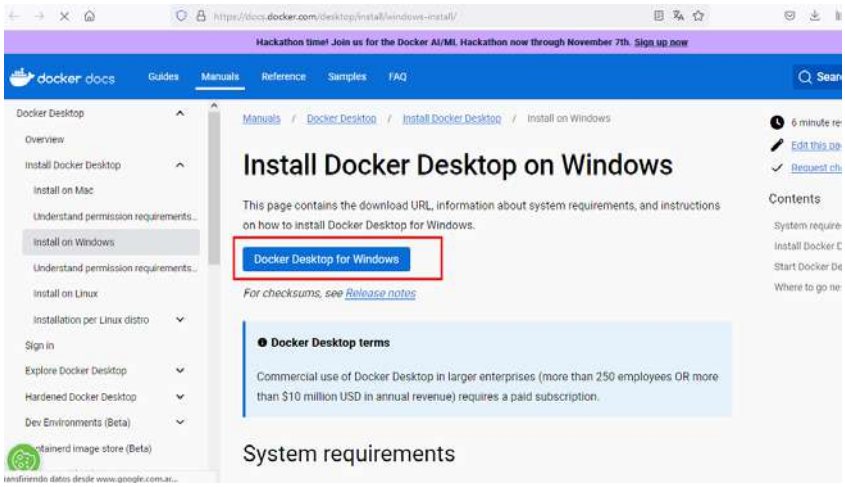


Figura 1: Página de descarga

[Fuente: fuente propia]

Deberá indicar dónde guardar el archivo que se descarga (Figura 2).

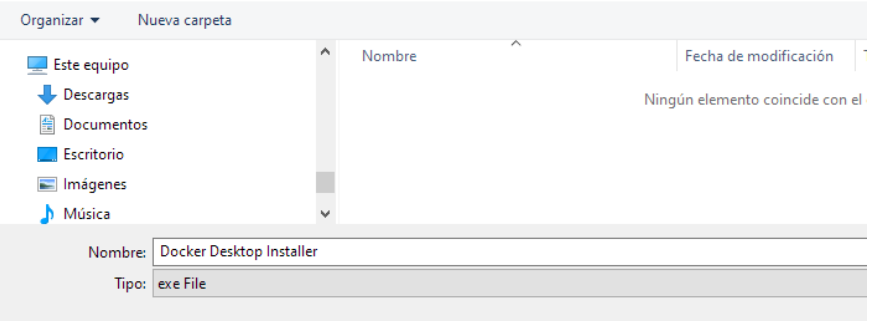


Figura 2: Guardar archivo

[Fuente: fuente propia]

Una vez descargado el archivo, deberá proceder a su instalación.

Haga doble clic sobre el archivo y siga los pasos propuestos por el asistente guiado. Se sugiere la instalación por defecto propuesta por el asistente.

Es posible que se visualice una pantalla de advertencia de seguridad (Figura 3).

Seleccionar la opción de Ejecutar.

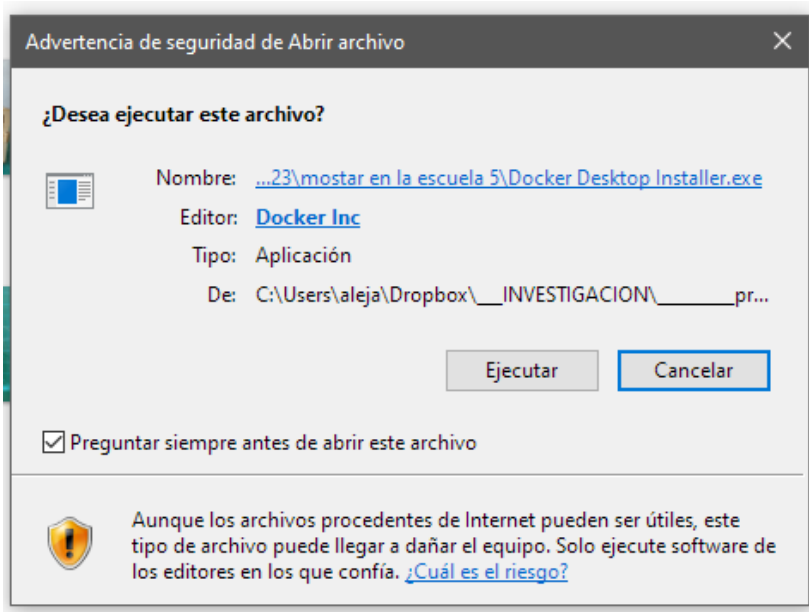


Figura 3: Advertencia de seguridad

[Fuente: fuente propia]

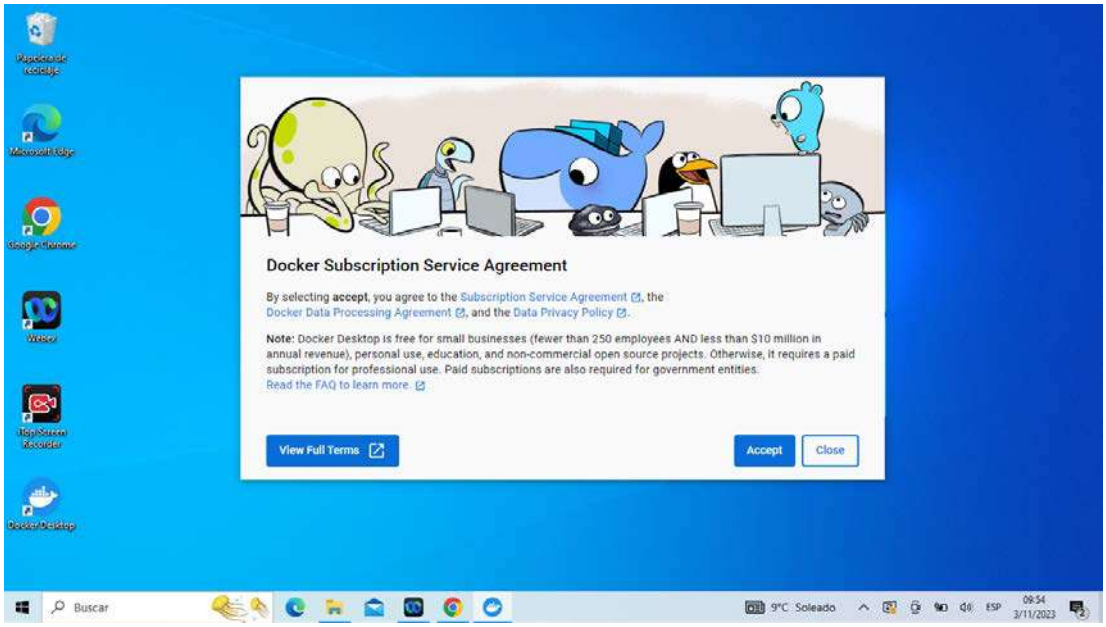
El proceso de instalación puede demorar varios minutos.

Nota: En caso de tener algún problema durante esta instalación referirse a consultar la página del primer punto que tiene documentación para solucionar todos los posibles escenarios.

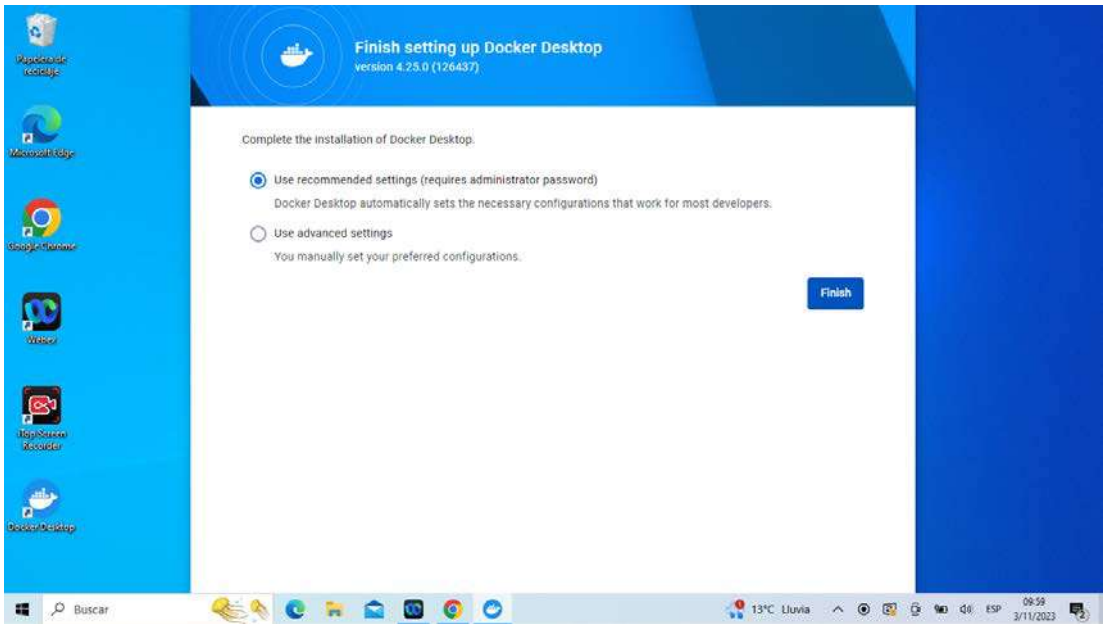
Cuando finalice el proceso, el asistente le pedirá reiniciar el equipo. Deberá reiniciarlo.

Iniciar Docker Desktop y terminar la configuración:

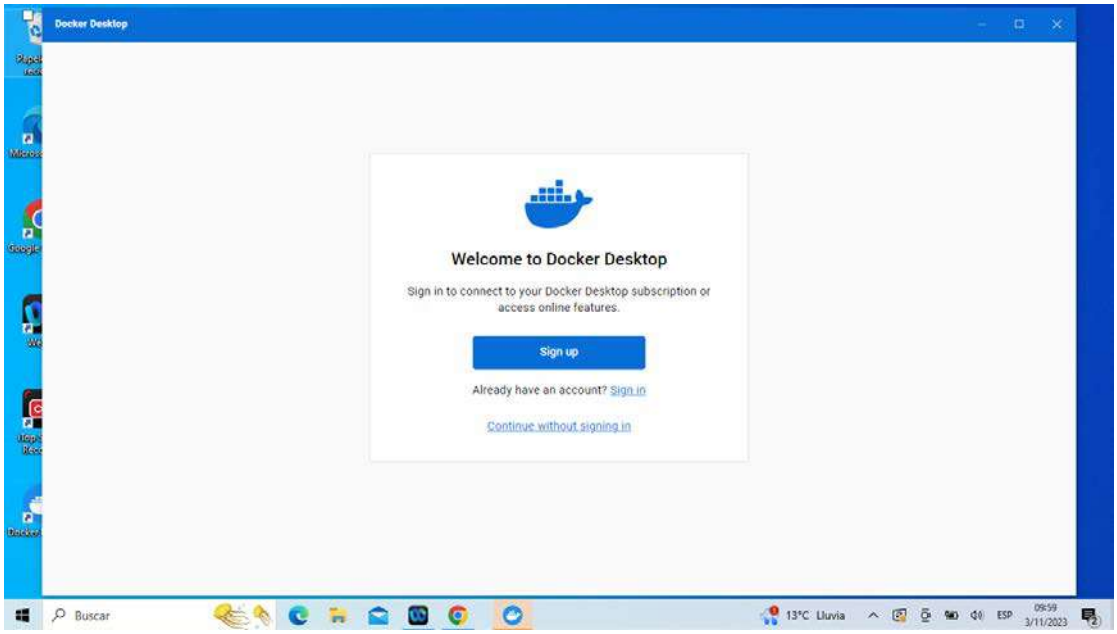
1- Tocar Aceptar:



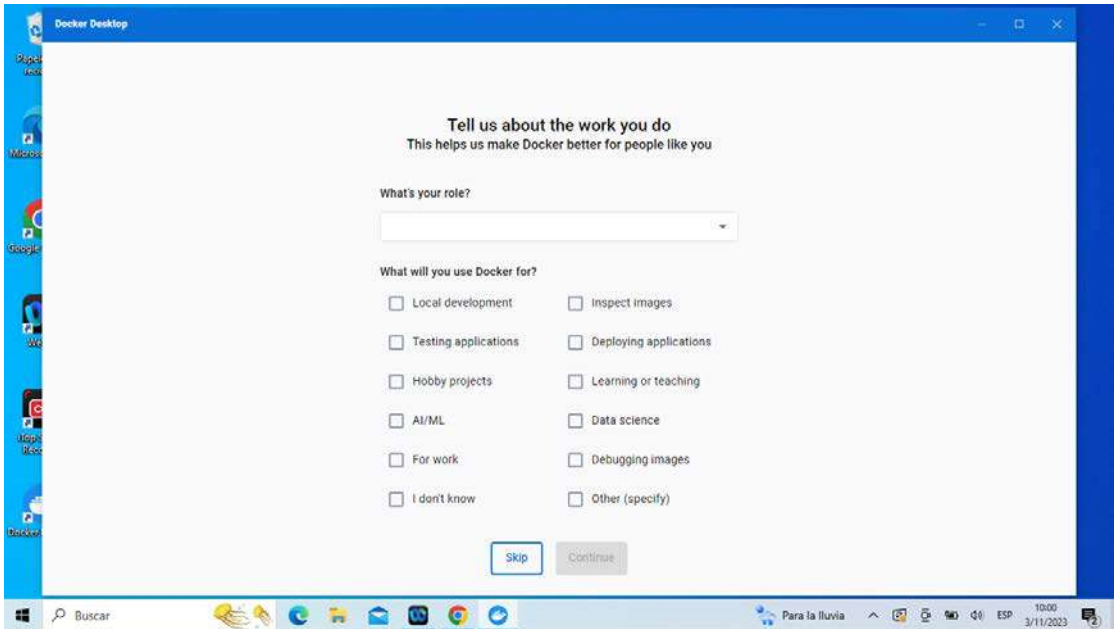
2- Tocar Finish:



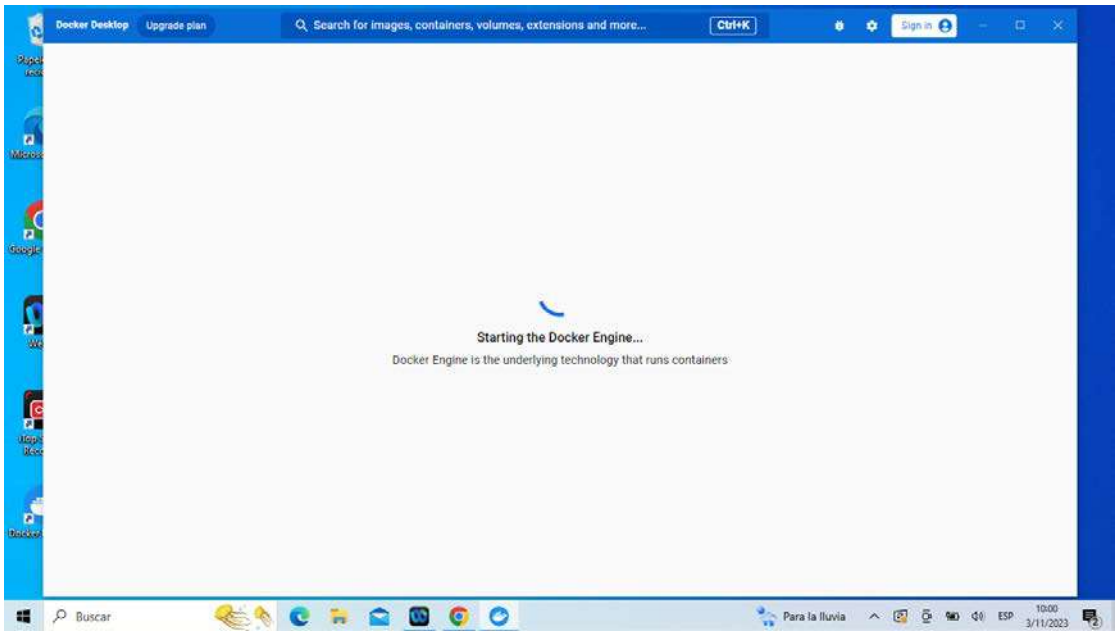
3- Tocar “Continue without signing in”



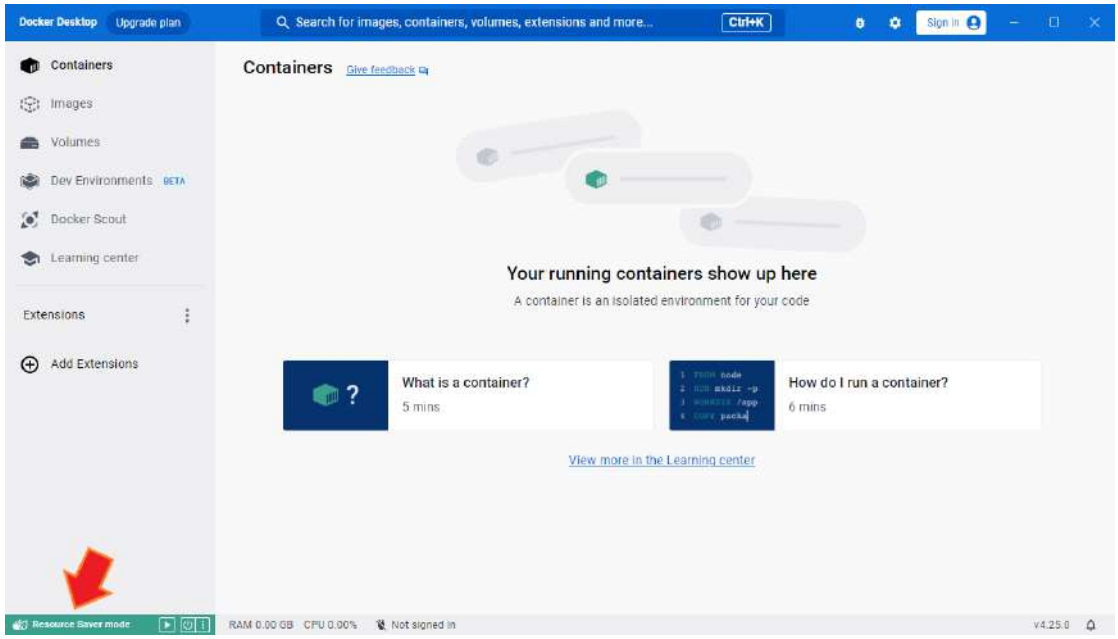
4- Tocar Skip:



5- Esperar a que termine de arrancar:



6- Una vez finalizado corroborar que el servicio de Dockers esté andando.



Despliegue de Dockers

Bajar el repositorio del proyecto de este trabajo en formato zip desde https://github.com/zeeno77/Tesina/archive/refs/heads/main.zip

Deberá elegir dónde descargar el archivo Tesina-main.zip mostrado en la Figura 4 (No usar carpetas con espacios en los nombres)

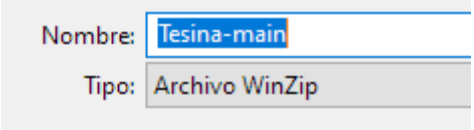


Figura 4: Descarga del archivo Tesina-main

[Fuente: fuente propia]

Una vez descargado deberá descomprimirlo en una carpeta sin espacios en su nombre.

Ingresar a la carpeta llamada "*Tesina-main*" recientemente descomprimida, e ingresar a la carpeta llamada "*backendAPI*". Una vez allí dentro, copiar la ruta donde se encuentra el archivo docker-compose.yml (como se muestra a continuación en la Figura 5)

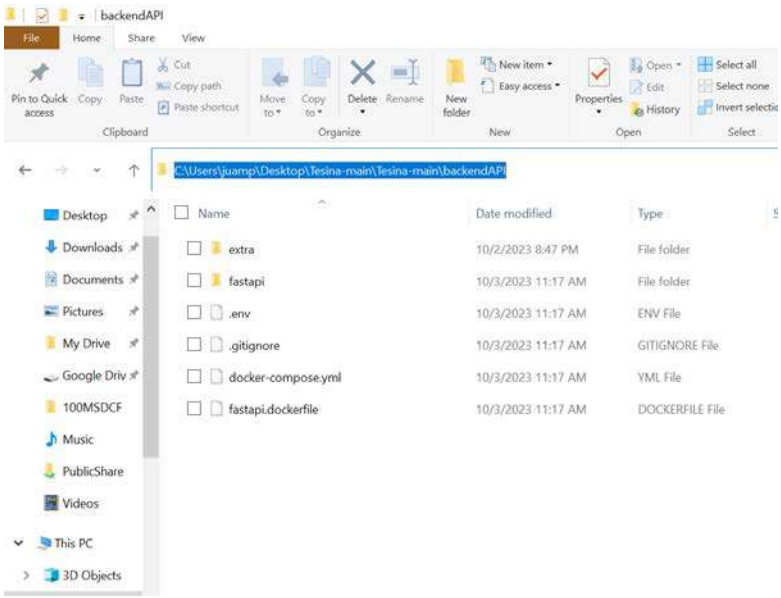


Figura 5: Copiar ruta del archivo docker-compose.yml

[Fuente: fuente propia]

A continuación deberá abrir en su equipo la aplicación Windows Powershell y ejecutarla como administrador . La Figura 6 muestra cómo buscar la aplicación es su equipo y una forma de acceder a ejecutarla como administrador.

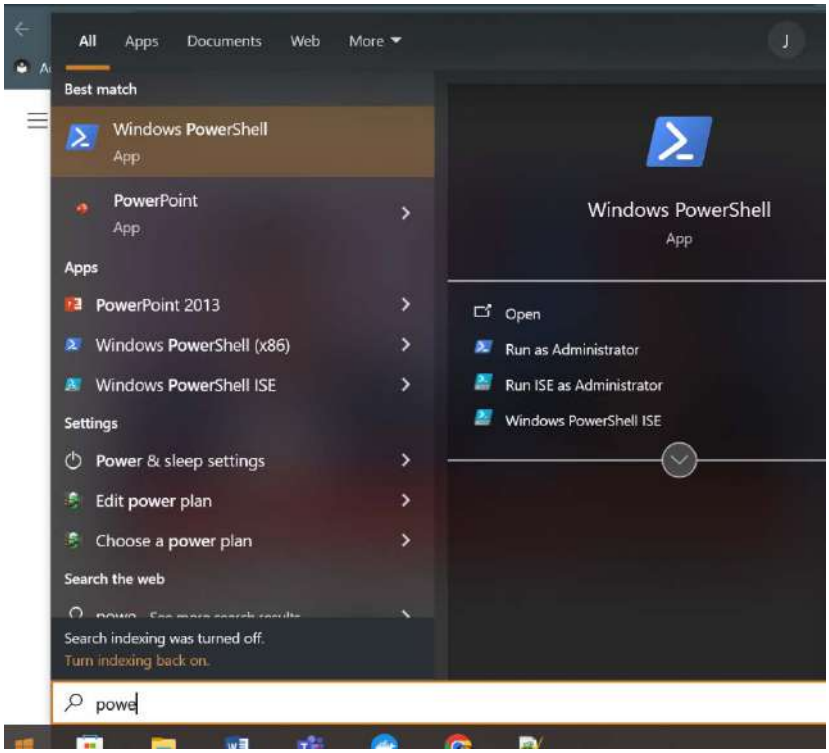


Figura 6: Aplicación Windows PowerShell

[Fuente: fuente propia]

Desde la consola que habilita la aplicación PowerShell, ir a la carpeta donde se encuentra descomprimido el archivo docker-compose.yml. Recuerde que el nombre de la carpeta no debe tener espacios en blanco ya que de ser así no podrá acceder desde esta aplicación (la Figura 7 muestra un ejemplo de ruta, en su caso, es la ruta que copió anteriormente)

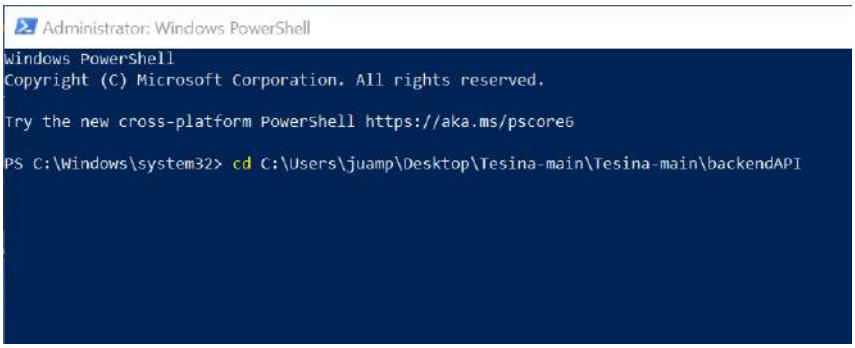


Figura 7: Línea de comando para acceder a la carpeta backendApi

[Fuente: fuente propia]

Luego de estar en la ruta adecuada, ejecutar el comando '*docker-compose build*'. A continuación la Figura 8 muestra la línea de comando la cual se ejecuta al presionar la tecla Enter..

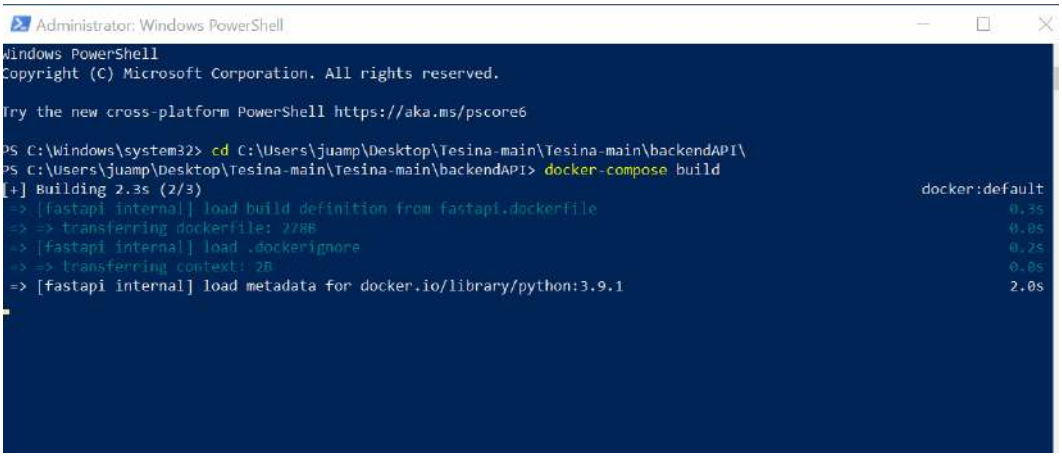


Figura 8: Ejecución del comando build sobre el archivo docker-compose

[Fuente: fuente propia]

A continuación, ejecutar el comando '*docker-compose up*' y esperar a que se bajen todas las imágenes (notar que este proceso puede tardar varios minutos).

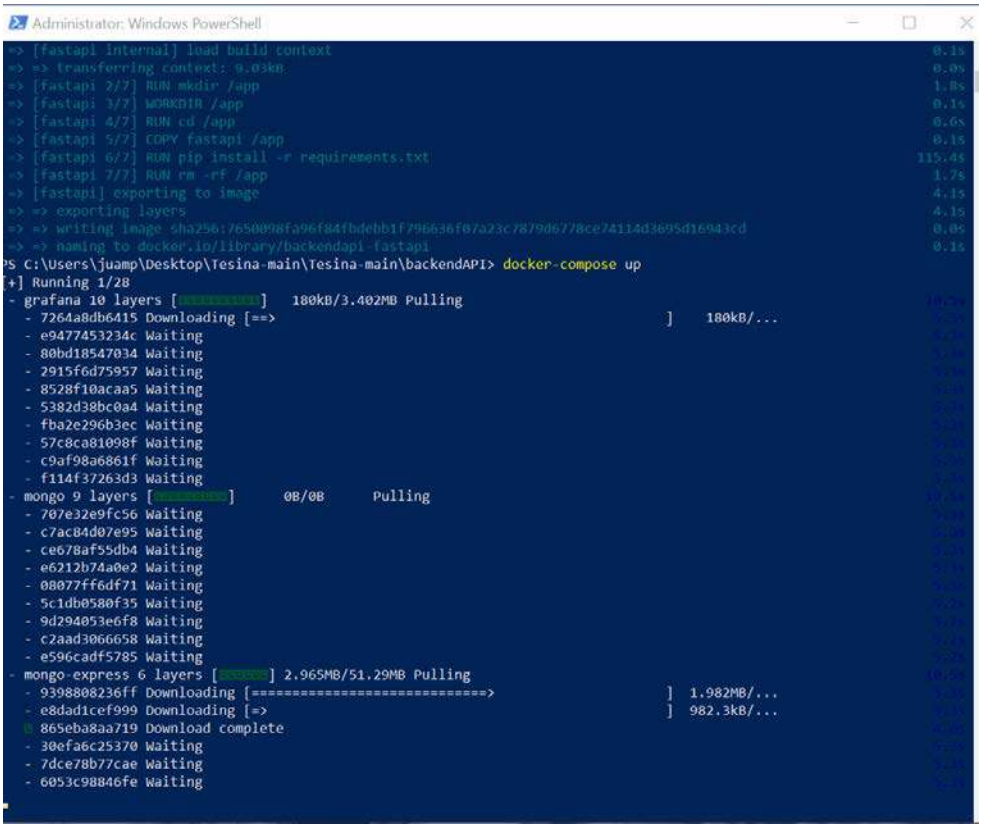
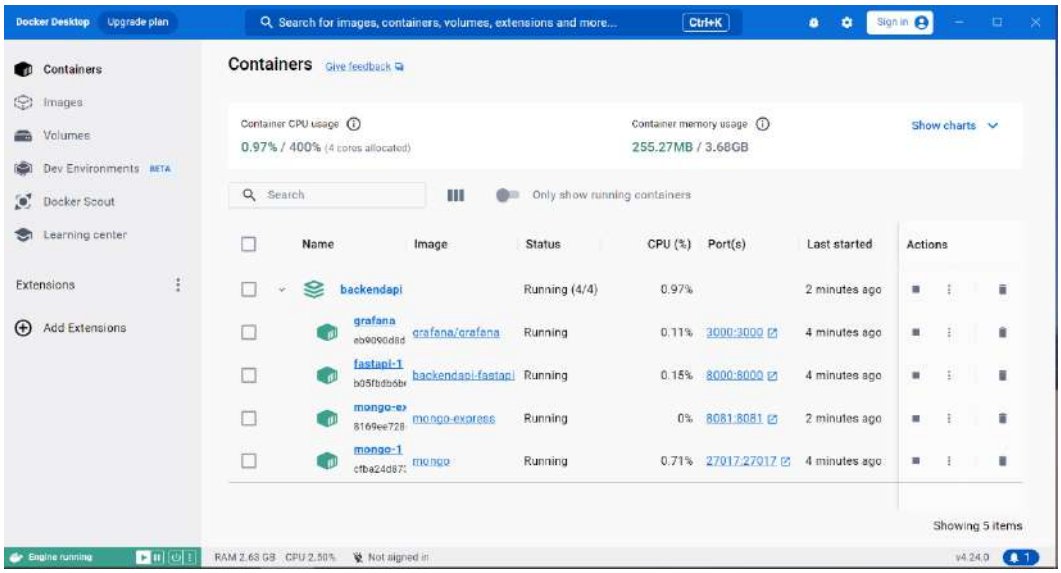


Figura 9: Ejecución del comando up sobre el archivo docker-compose

[Fuente: fuente propia]

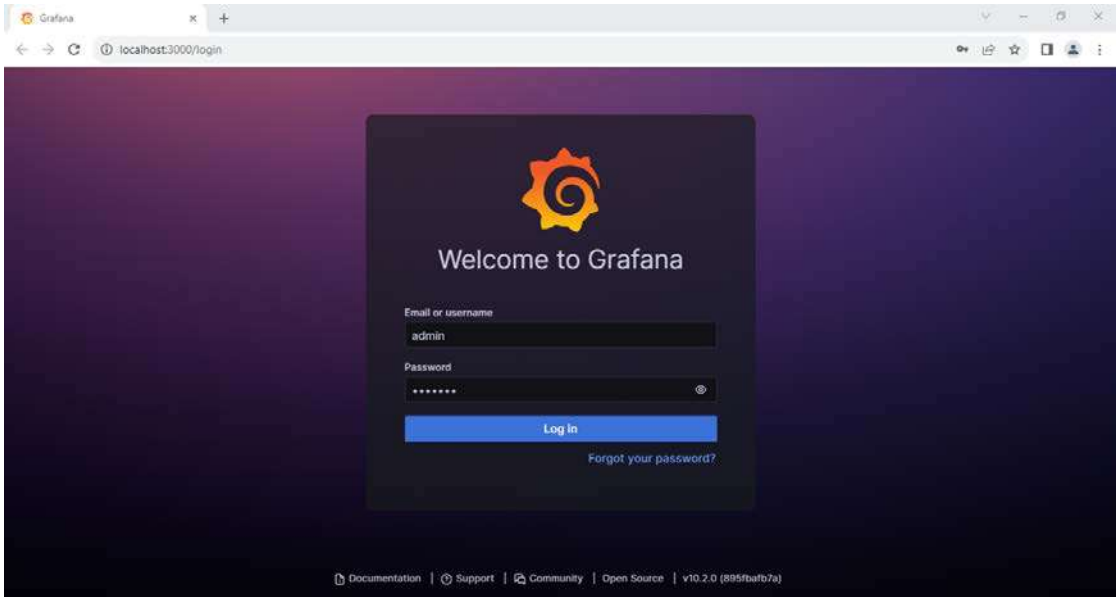
Una vez terminado el proceso cerrar la ventana de Powershell.

Proceder a levantar los containers desde Docker Desktop. Para ello,

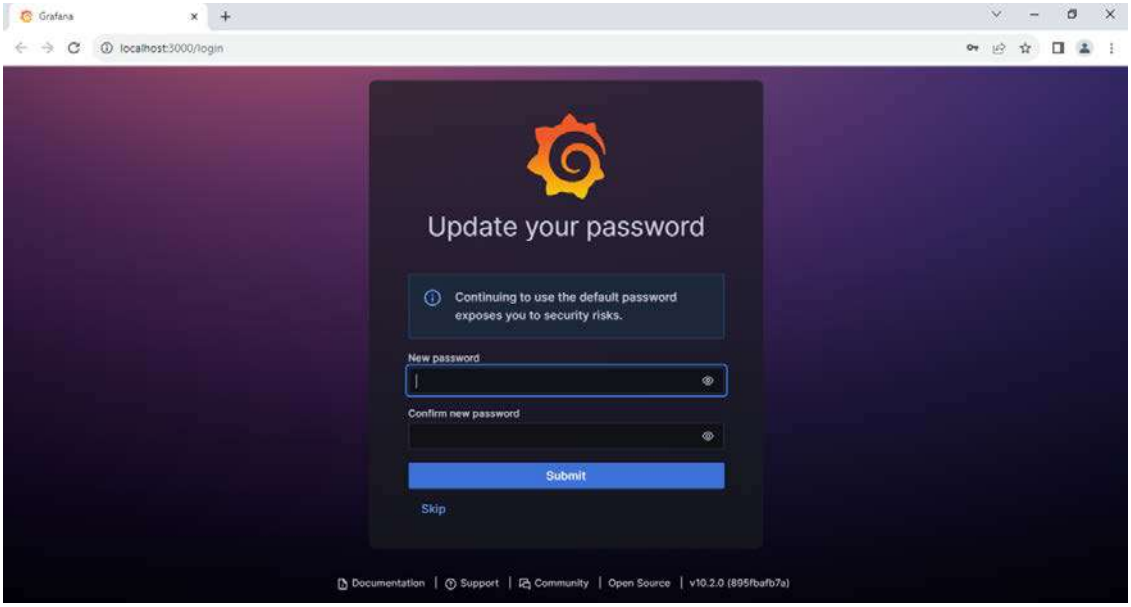


Configuración de Grafana para la visualización de datos

Entrar en http://localhost:3000/login (user: admin pass: admin)

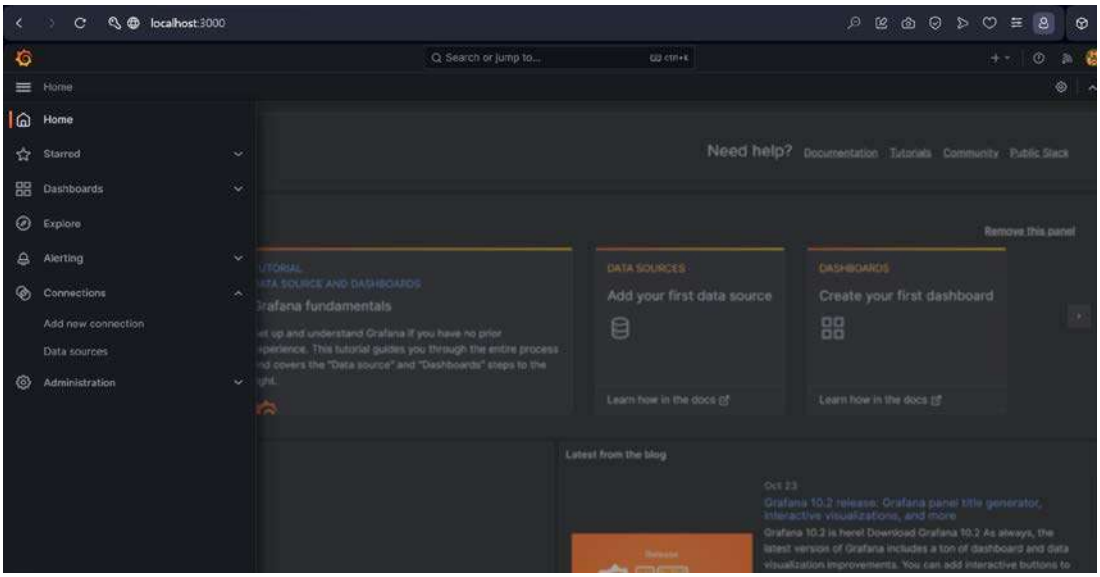


Nos va a pedir que actualicemos la contraseña, poner una nueva y anotarla.

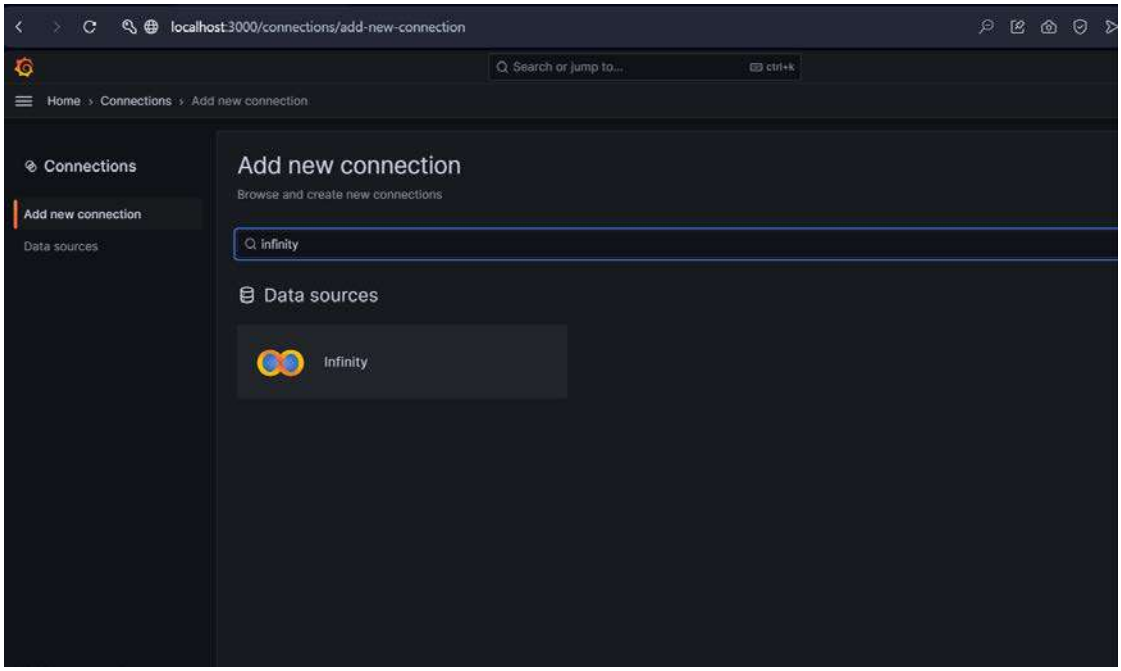


Crear un datasource infinity:

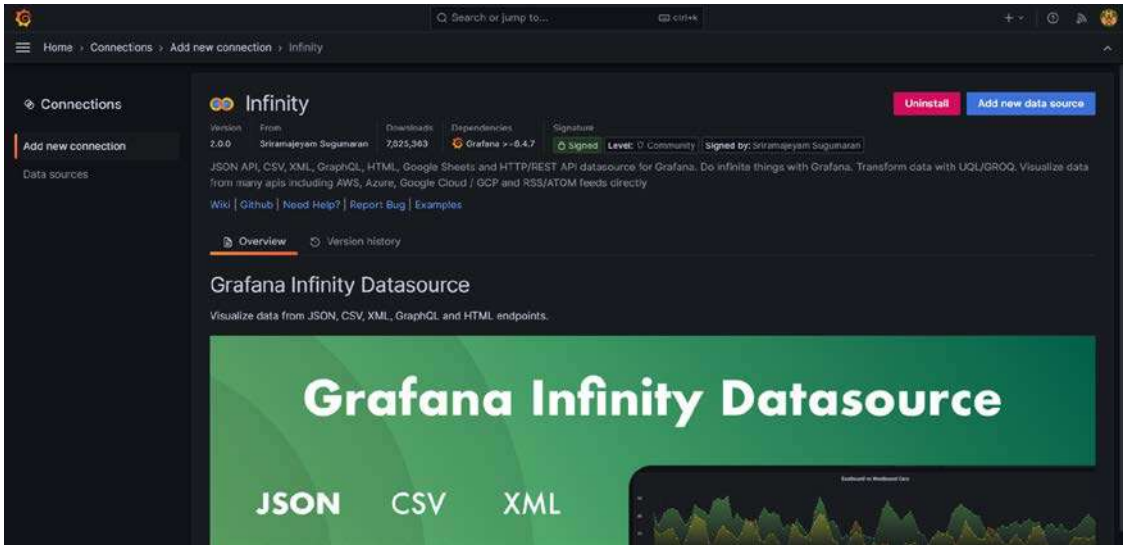
1-Ingresar a Home/Connections/Add new datasource



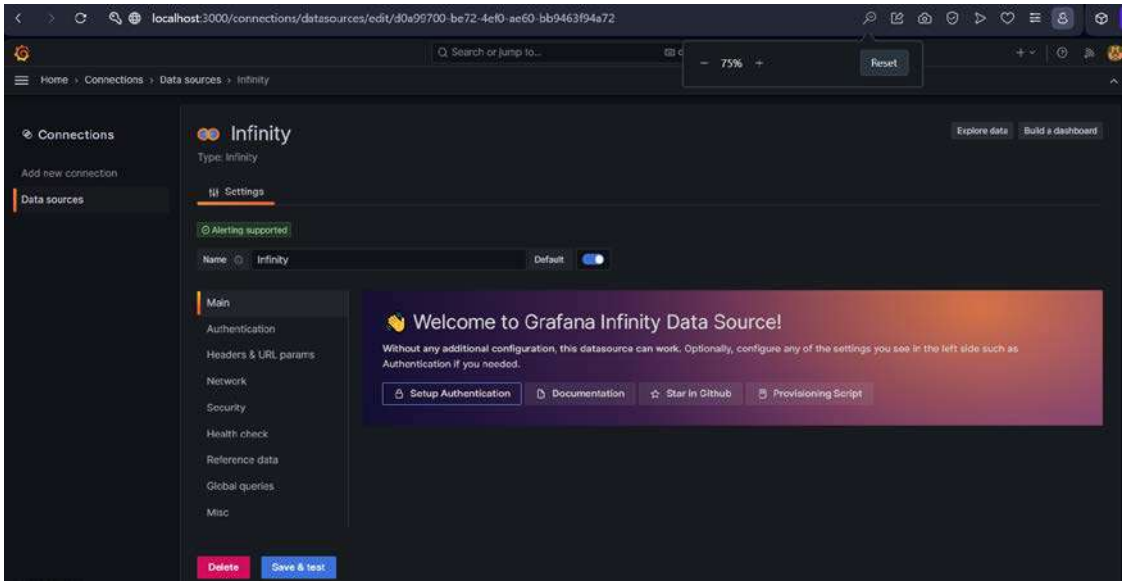
2-Buscar infinity para filtrar los tipos de conexiones y tocar en único icono que nos devuelve la búsqueda.



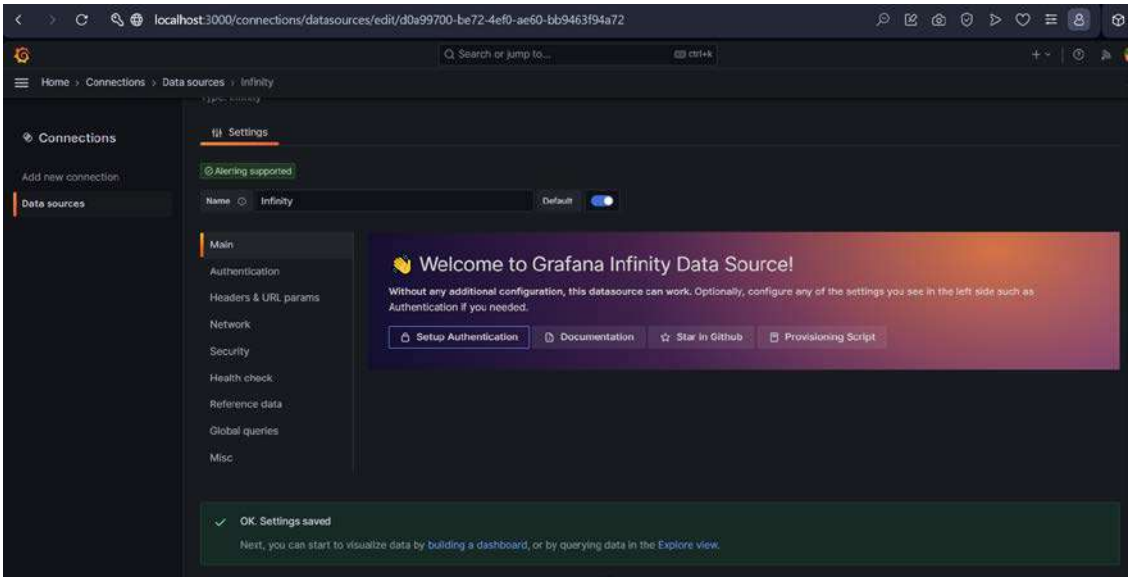
3-Tocar Add new datasource



4-Tocar Save and Test

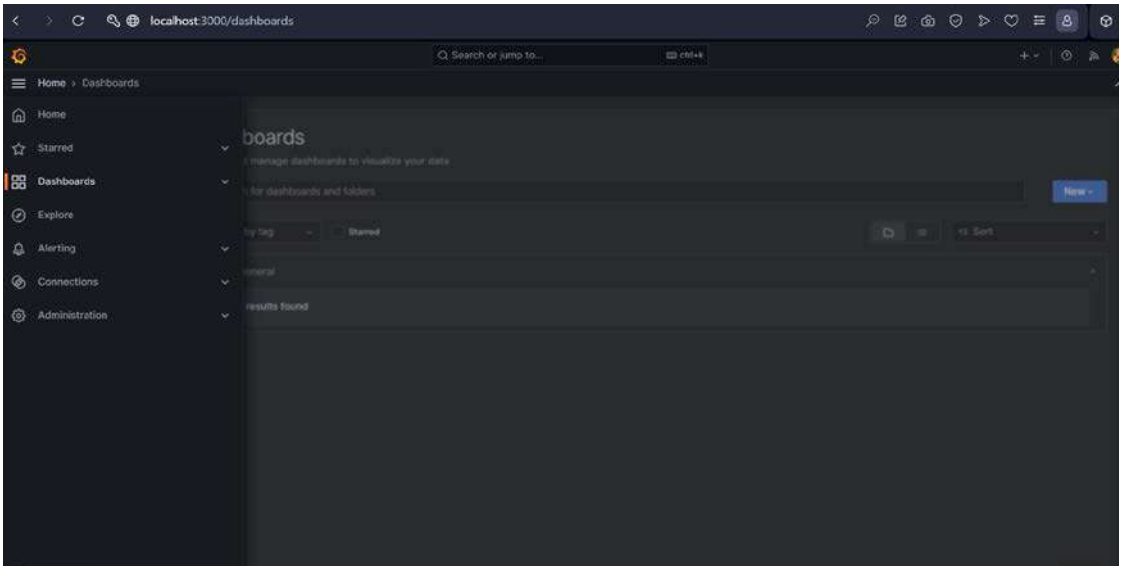


5- Si todo sale bien va a tirar el cartel verde que se muestra en la figura de abajo.

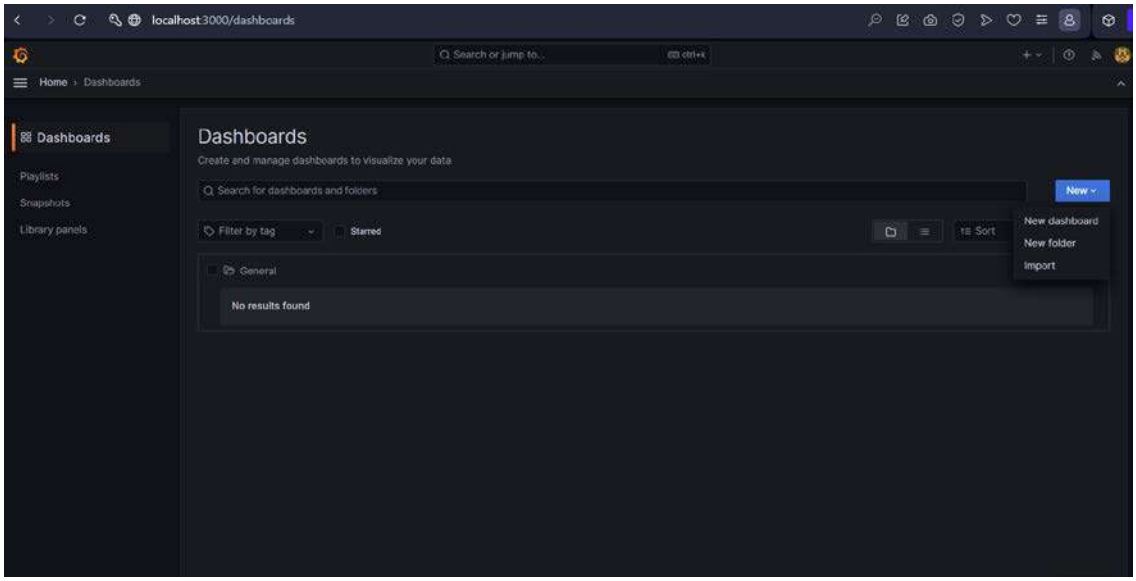


Importar la configuracion del dashboard.

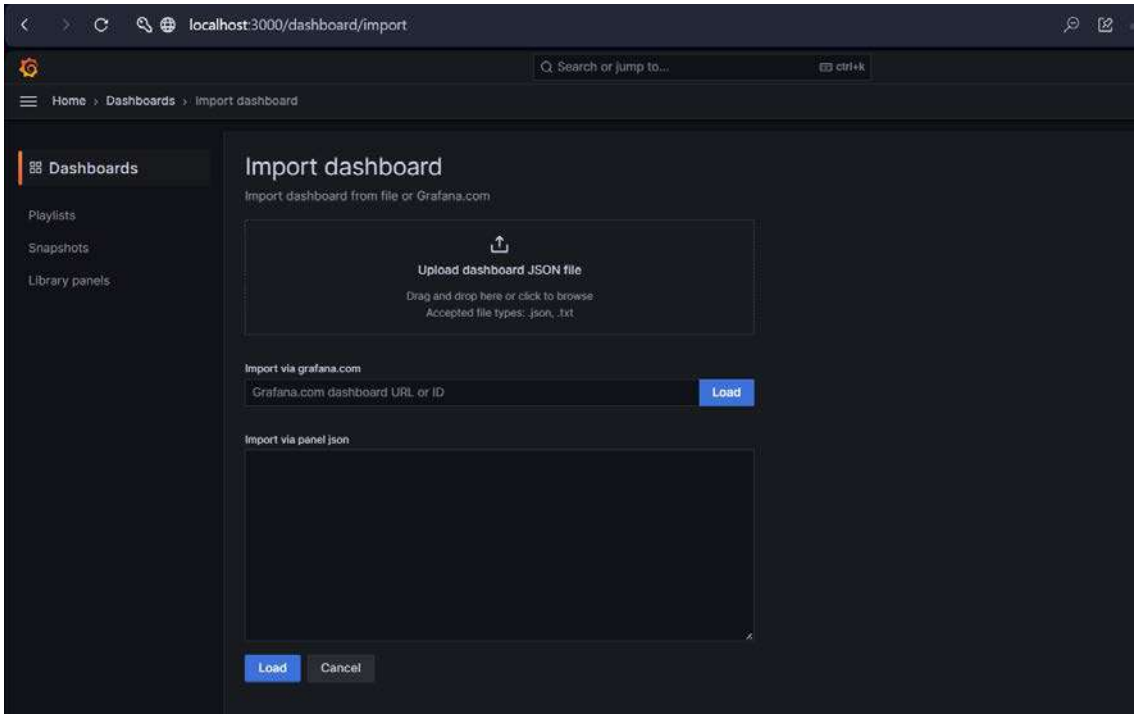
1- Ingresar a Home/Dashboars



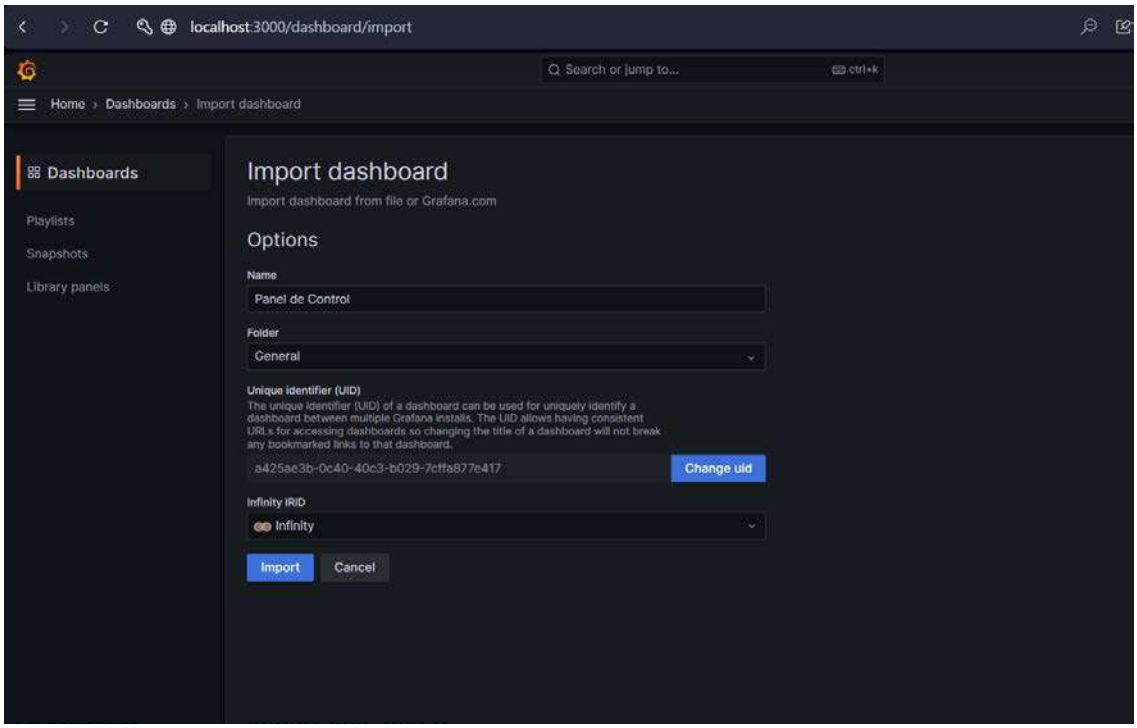
2- Ingresar a New/import



3- Ingresar a Upload dashboard JSON file y seleccionar el archivo dentro de Tesina-main\backendAPI\extra\grafana_exports\Panel de Control-xxxxxxxxxxx.json



4- Seleccionar el datasource que creamos antes y tocar import



Anexo III: Guías para las actividades prácticas

Primer encuentro

Fecha de realización: Lunes 06 de noviembre de 2023

Objetivos del encuentro

Introducir conceptos básicos de la programación en bloque usando la plataforma MakeCode para placas MicroBit.
Adquirir las habilidades básicas con el fin de comprender aspectos básicos necesarios para transmitir datos usando una arquitectura particular (a presentarse en el segundo encuentro)

Introducción

MakeCode es un entorno de programación visual desarrollado por Microsoft diseñado para enseñar programación a principiantes de todas las edades. Utiliza bloques de código que representan diferentes comandos y operaciones, como bucles, condicionales,

variables y eventos. Los usuarios pueden arrastrar y soltar estos bloques para construir programas de manera intuitiva.
Posee una función de simulación, permite a los usuarios probar sus programas antes de cargarlos en hardware real. Esto facilita la depuración y la verificación del comportamiento de un programa.
La comunidad de MakeCode comparte proyectos, tutoriales y recursos educativos. También ofrece una biblioteca de proyectos y actividades listas para usar que abarcan desde la programación de videojuegos hasta la automatización de tareas del mundo real.

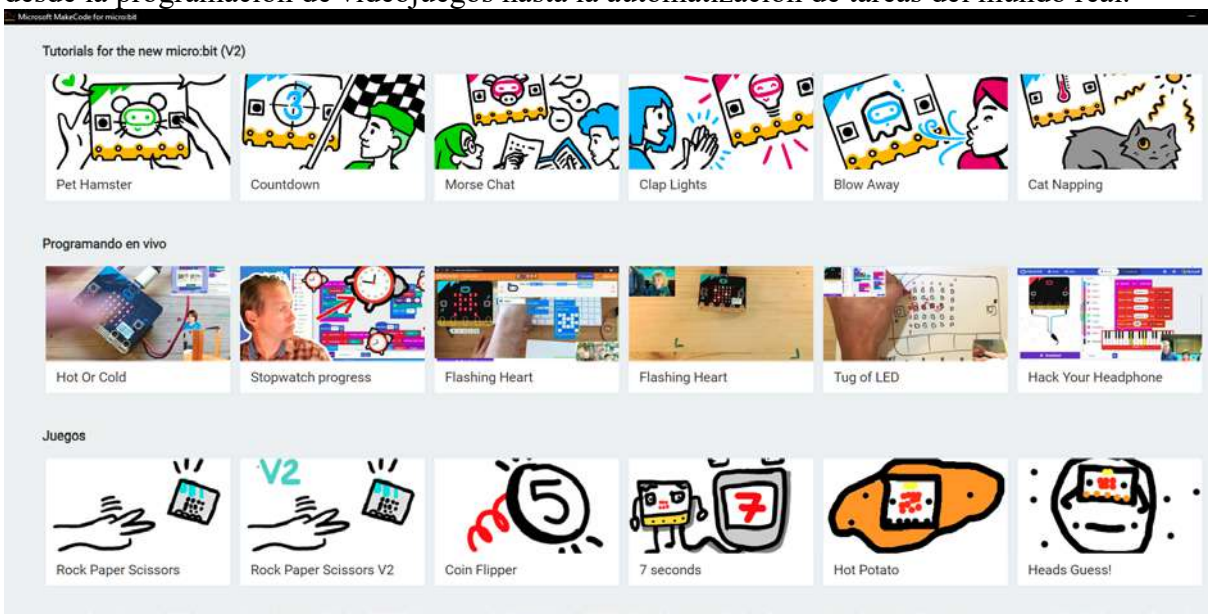


Figura 1: Biblioteca de MakeCode [Fuente: captura propia]

Se puede utilizar de manera online mediante un navegador desde cualquier plataforma y también cuenta con instaladores para Windows y Mac.
También cuenta con opciones para usuarios más avanzados, como la capacidad de convertir el código en bloques a Python y JavaScript y viceversa, y la creación de bloques personalizados.

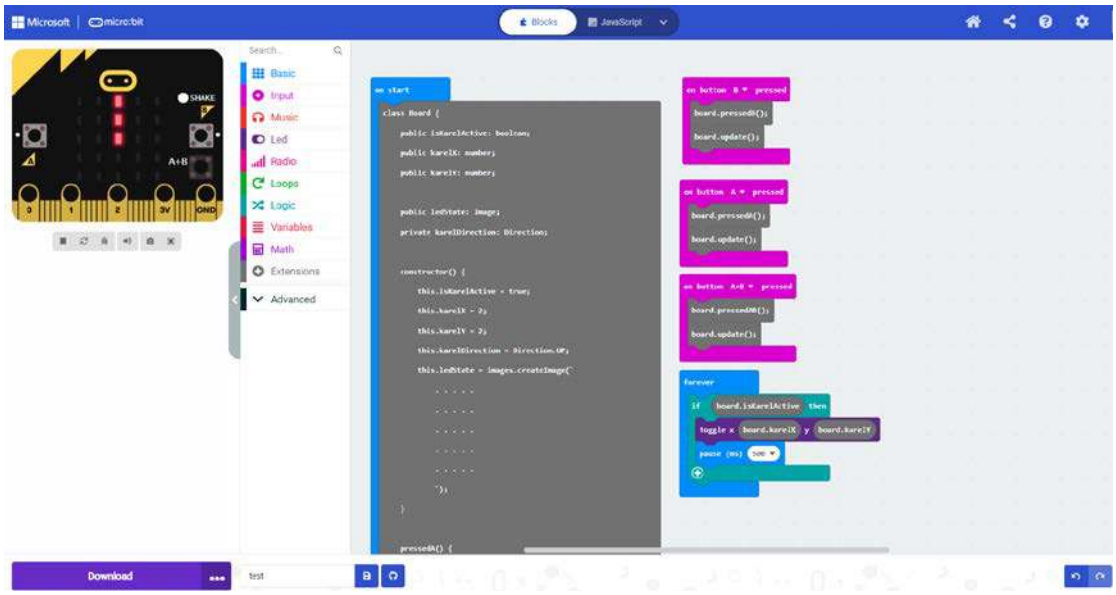


Figura 2: Ejemplo programa en editor MakeCode [Fuente: captura propia]

Ejercicios prácticos

A continuación se presentan tres ejercicios prácticos con nivel progresivo de complejidad.

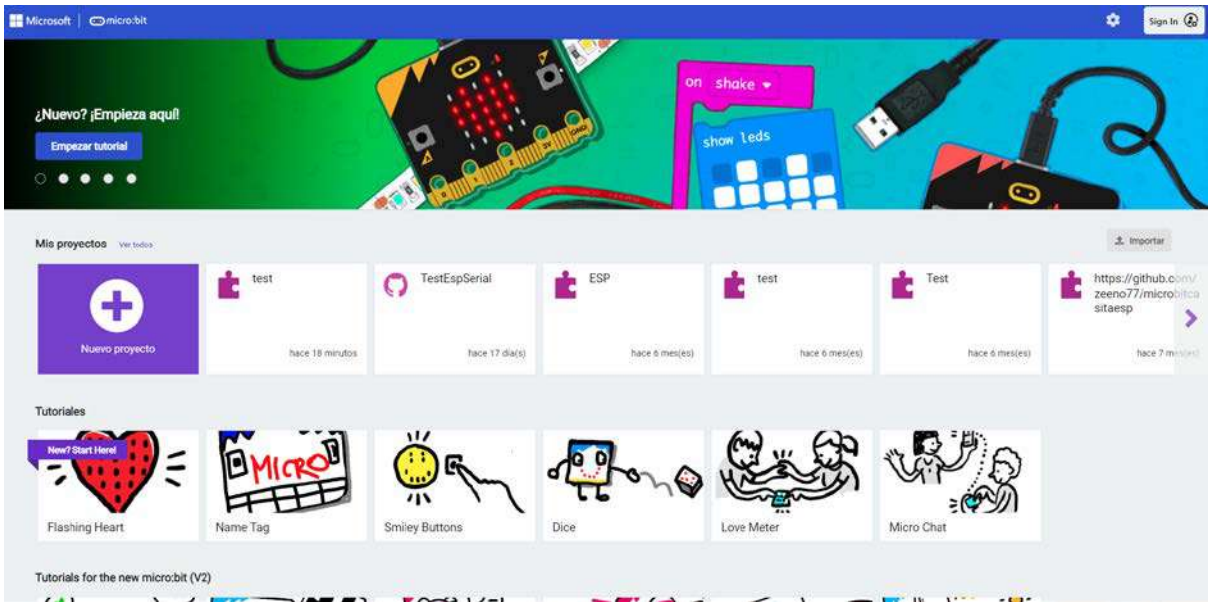
Ejercicio A

Vamos a crear un programa que reproduzca un sonido al tocar un botón.

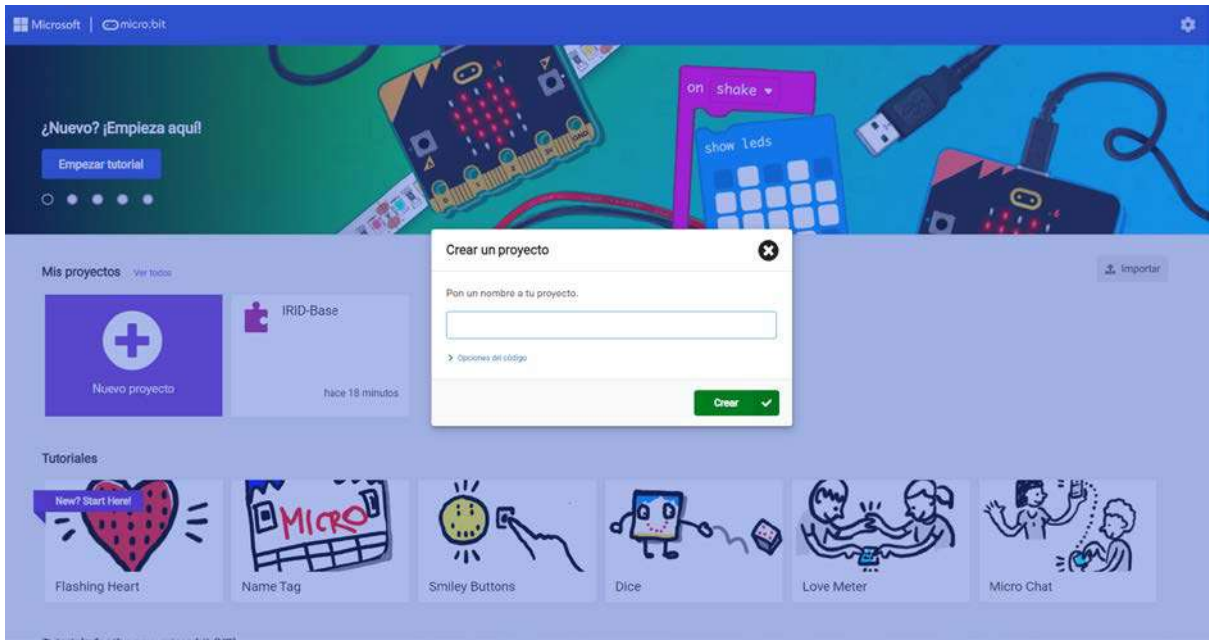
Conceptos a aplicar

- programación en bloque
- evento

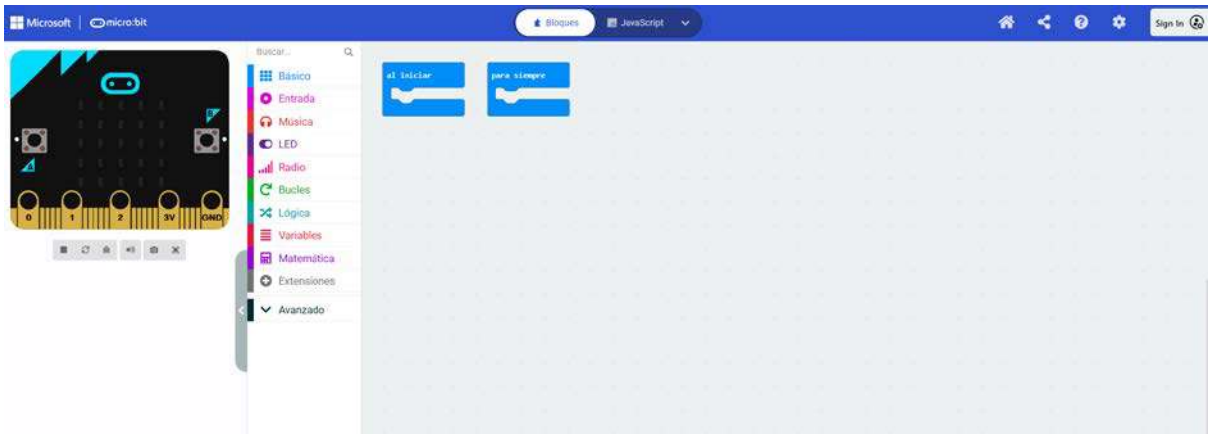
1- Ingresar a https://makecode.microbit.org/ y seleccionar **Nuevo proyecto**



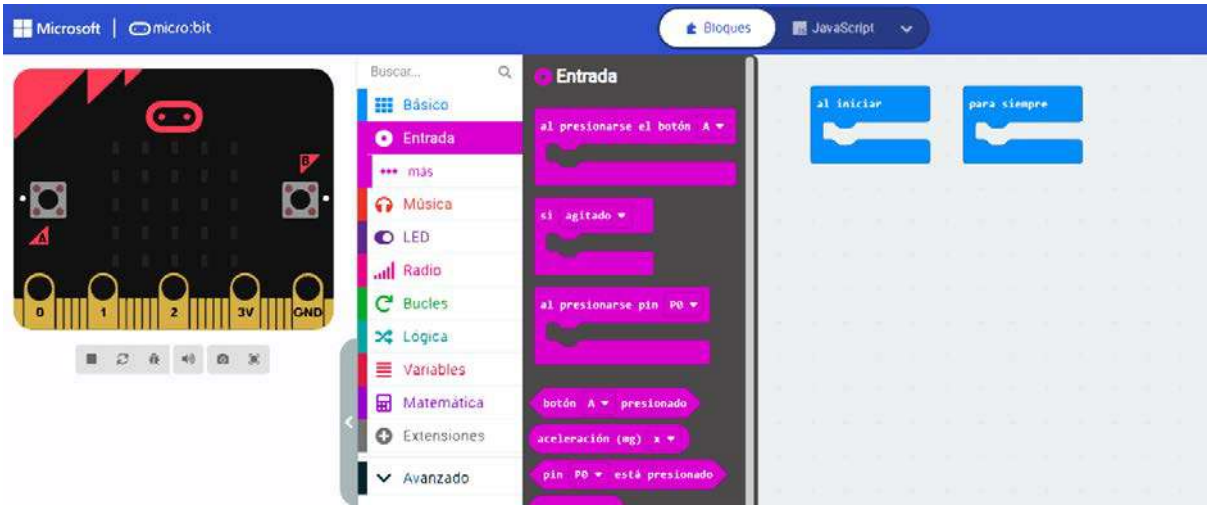
2- Ponerle un nombre al proyecto (por ejemplo “PrimerProyecto”) y presionar **Crear**



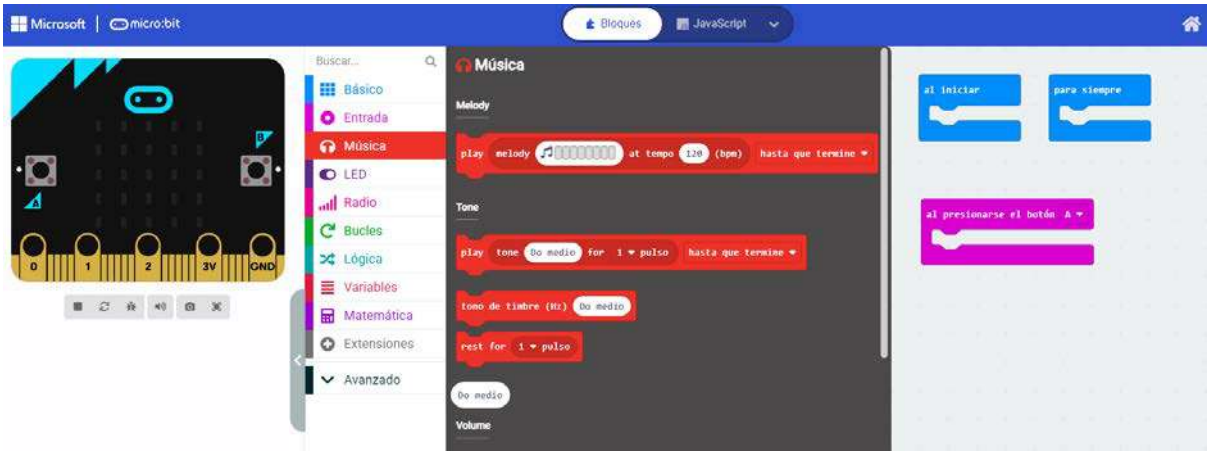
3- El bloque “**al iniciar**” se ejecuta una sola vez al prenderse la placa y el bloque “**para siempre**” se repite en bucle mientras la placa esté encendida.



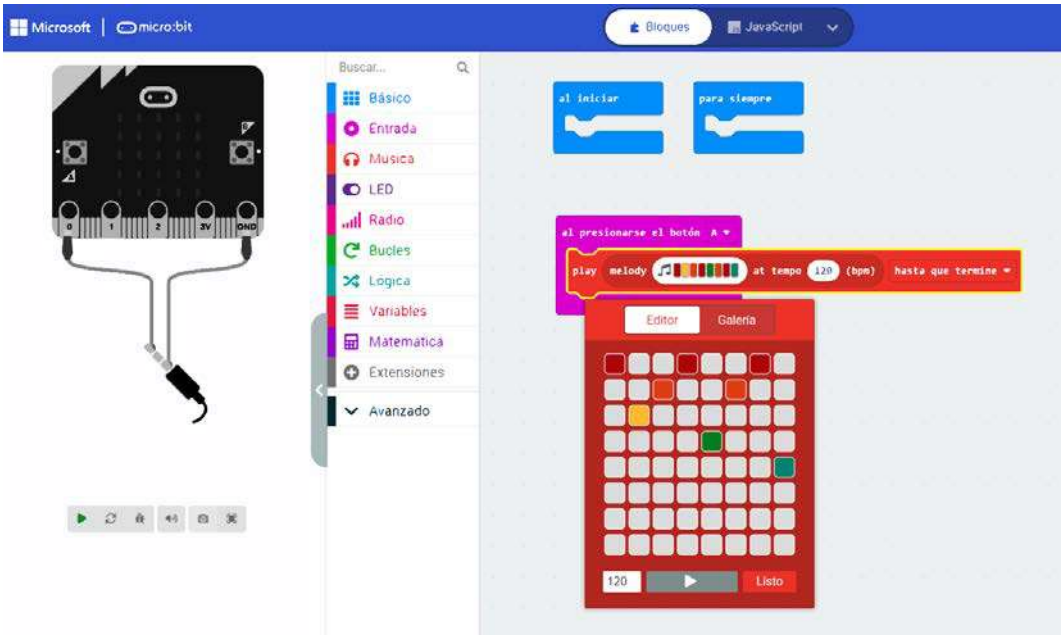
4- Arrastrar el bloque "**Entrada/Al presionar el botón A**" dentro del entorno de programación (parte gris a la derecha).



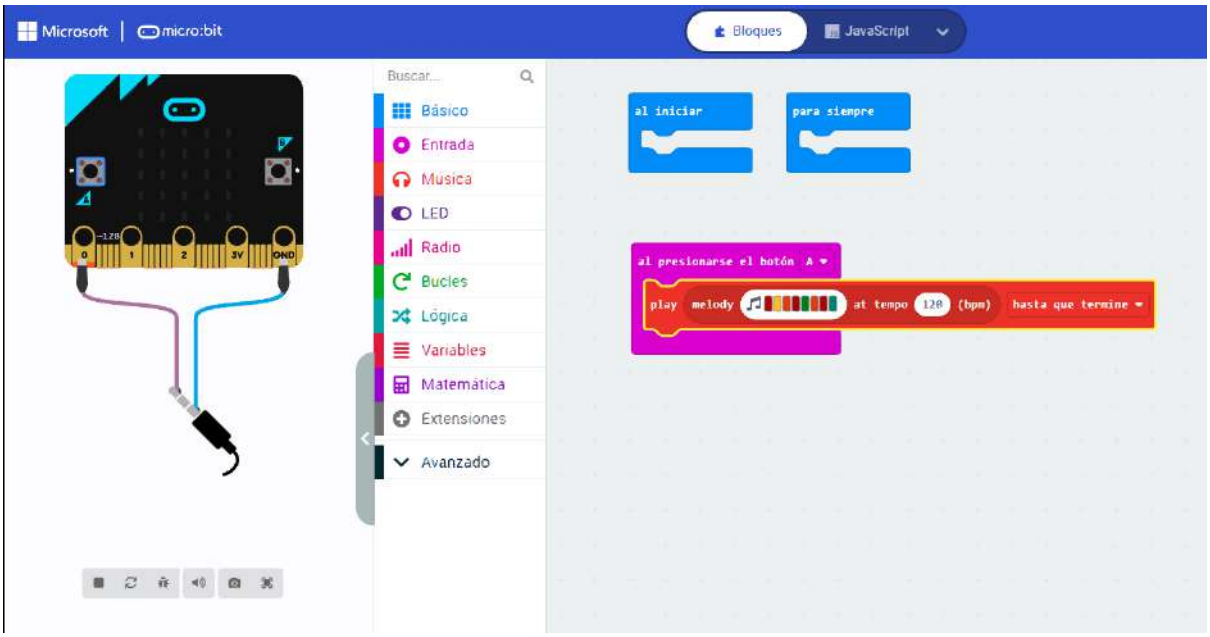
5- Arrastrar el bloque "**Música/ Play melody at tempo hasta que termine**" dentro del bloque "**Al presionar el botón A**".



6- Tocar el símbolo de música a la derecha de **melody** y editar la melodía que se quiera reproducir. Una vez editado tocar **Listo**.



7- Una vez terminado probar la melodía tocando el botón A del simulador.

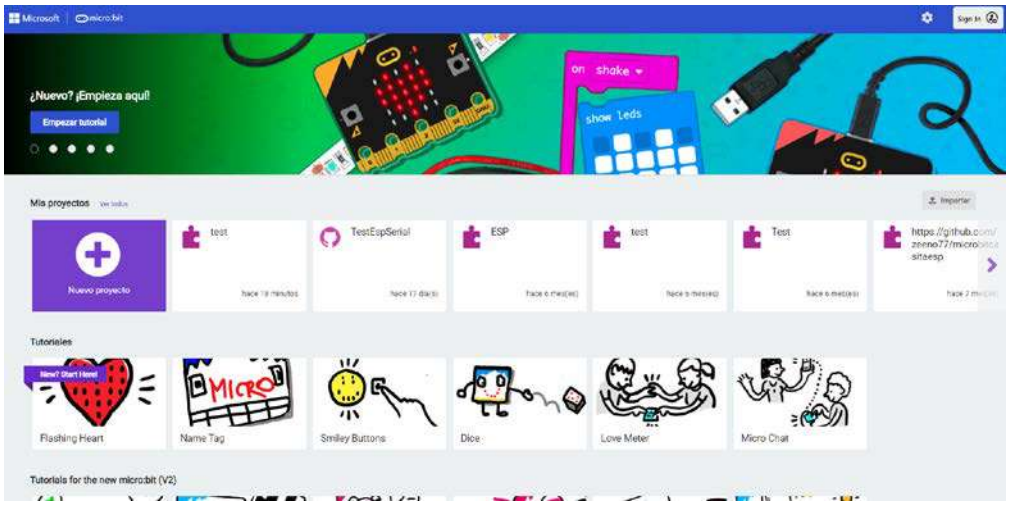


Ejercicio B

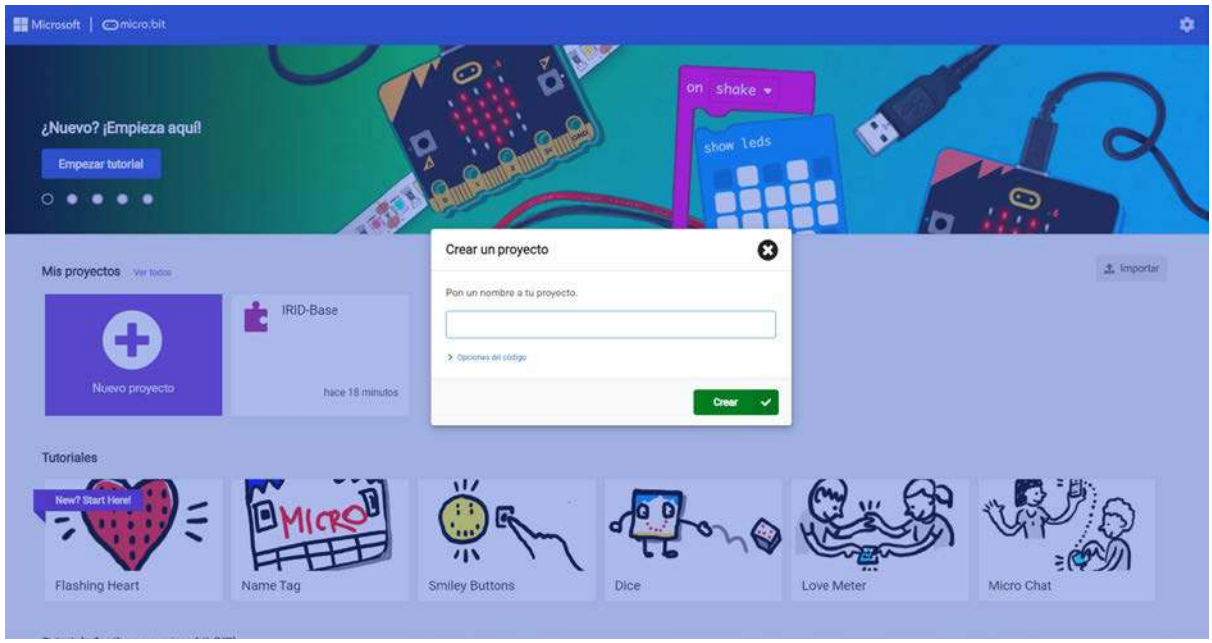
Vamos a crear un dado digital en MicroBit usando el sensor de agitación y la pantalla de led.

- **Conceptos a aplicar**
  - programación en bloque
  - uso de variables
  - tablero led para dibujar
  - sensor de agitación (‘shake’ en el emulador)
  - evento
  - estructura condicional
  - uso de diferentes familias de bloques
    - por ejemplo:
      - función de generación de números pseudoaleatorios (Bloque nativo)
      - cadenas de strings

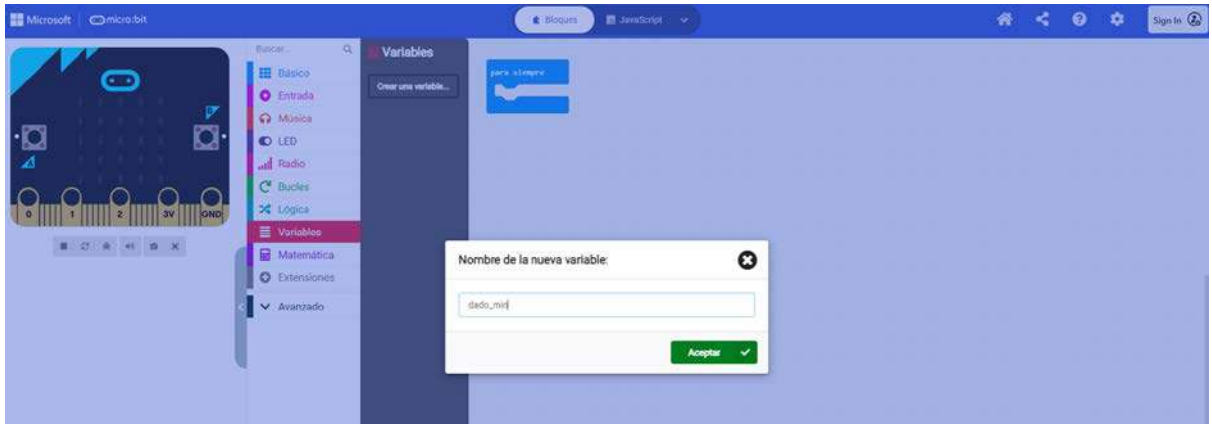
1- Ingresar a https://makecode.microbit.org/ y seleccionar **Nuevo proyecto**



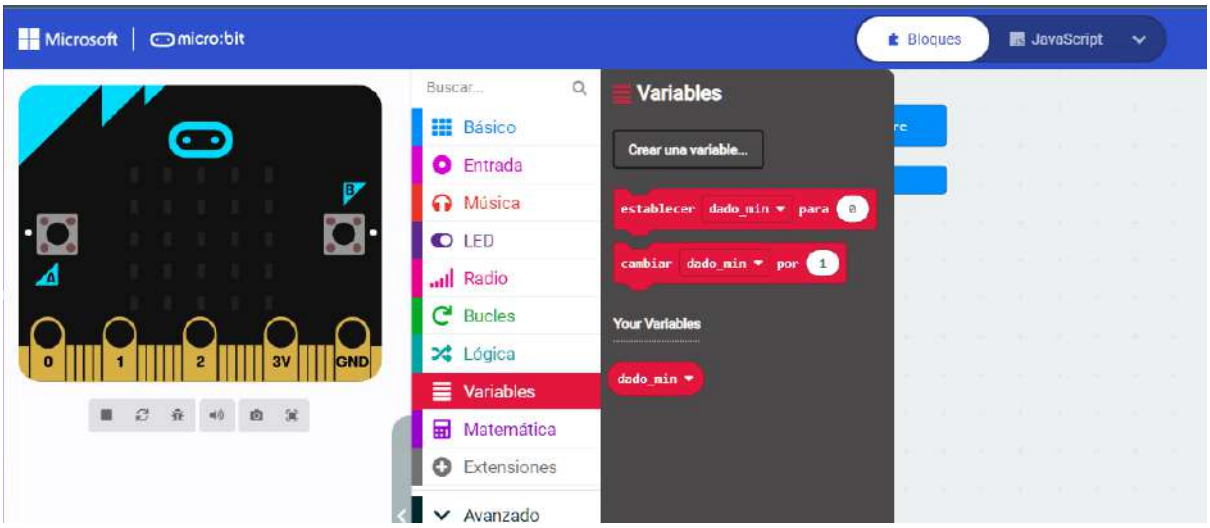
2- Ponerle un nombre al proyecto (por ejemplo “PrimerProyecto”) y presionar **Crear**



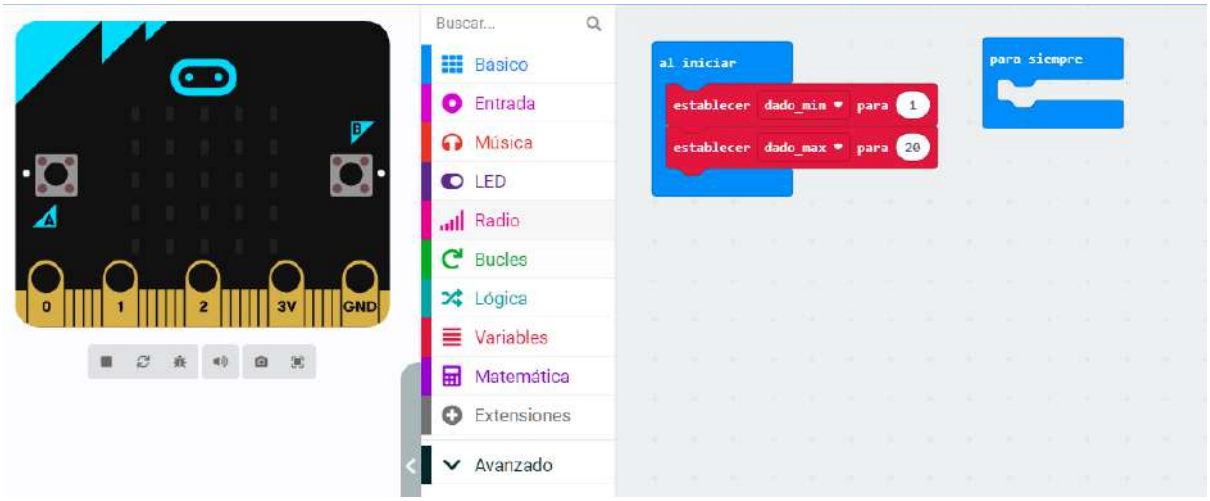
3- Crear 2 variables: **dado_min**, **dado_max** y le vamos a asignar 1 a **dado_min** y 6 a **dado_max**



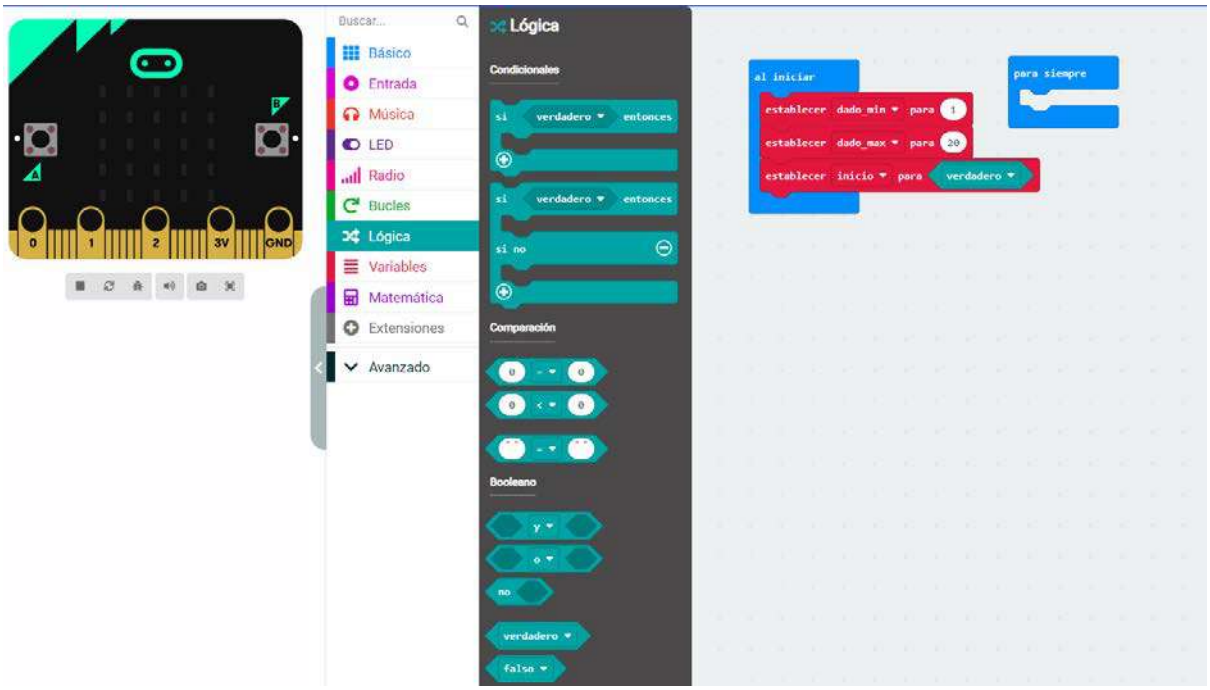
4.1- Asignar 1 a **dado_min** y 20 a **dado_max**



4.2 – Arrastrar los bloques de "**establecer dado_min**" en **1** y "**establecer dado_max"** para **20** al bloque "**al iniciar**"

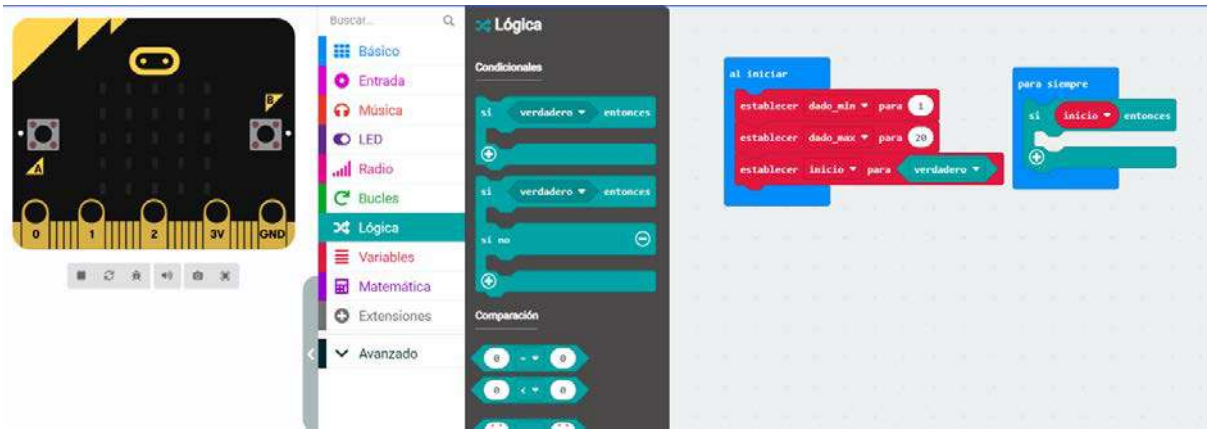


5 – Crear una variable "**numero_actual**" (no asignarle nada) y otra variable inicio y asignarle el bloque "**Lógica/verdadero**"

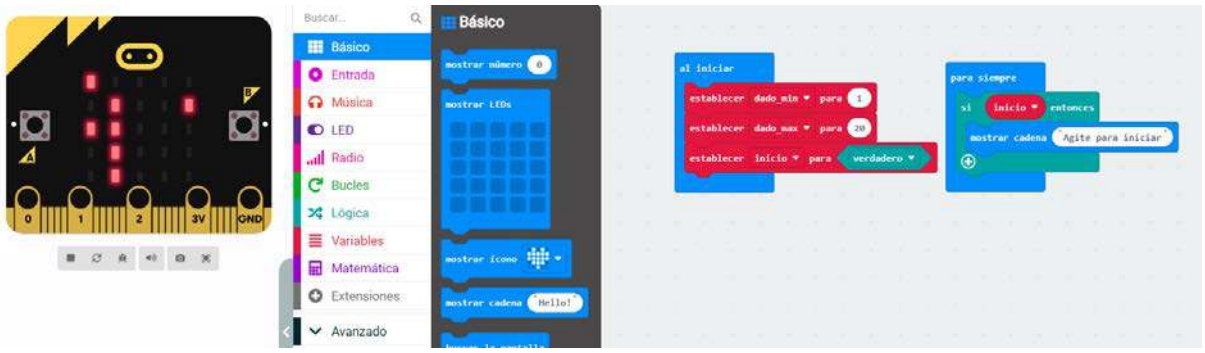


6 - Arrastrar el bloque "**Lógica/si verdadero entonces**" al bloque "**para siempre**"

6.1 - Arrastrar el bloque "**variables/inicio**" al bloque "**para siempre/si verdadero entonces**"

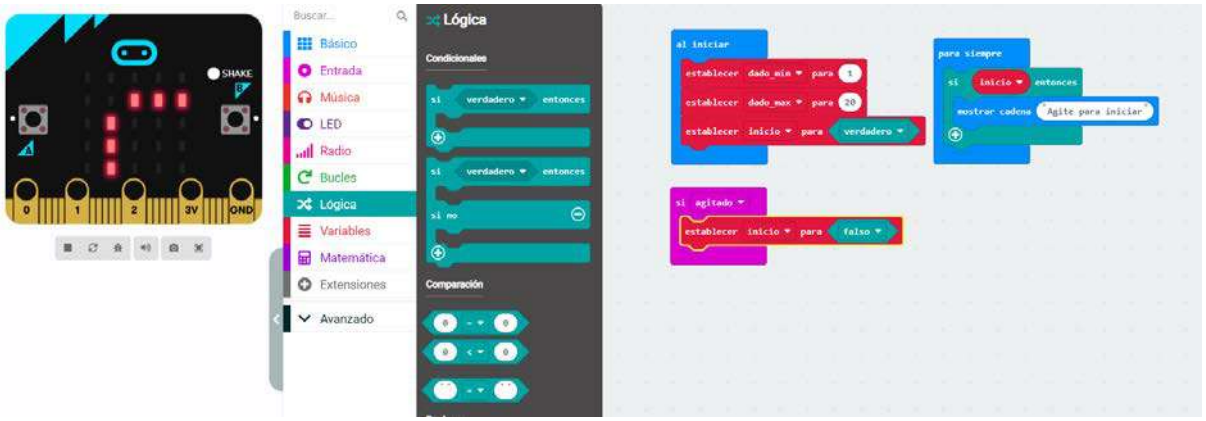


7 – Arrastrar el bloque "**basico/mostar cadena**" dentro del bloque "**para siempre/si verdadero entonces**" y cambiar el contenido por "Agite para iniciar"

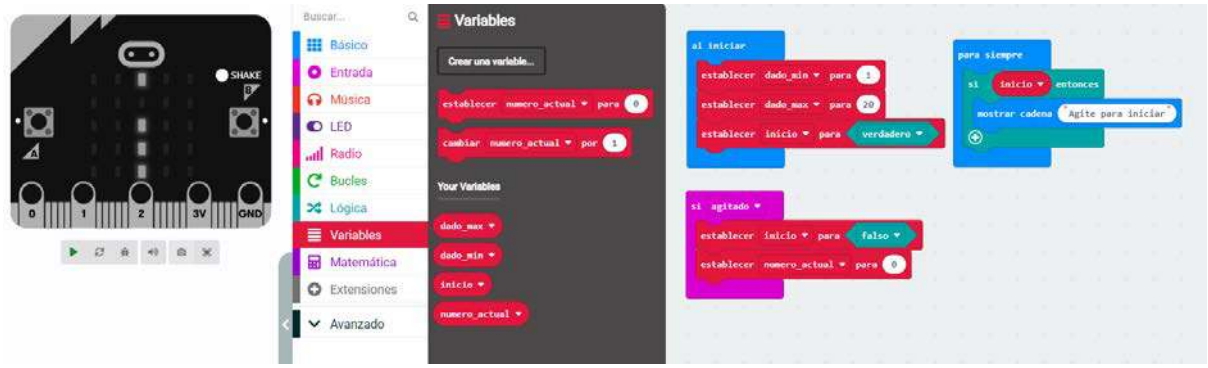


8 – Arrastrar el bloque "**entrada/si agitado**" al espacio de programación

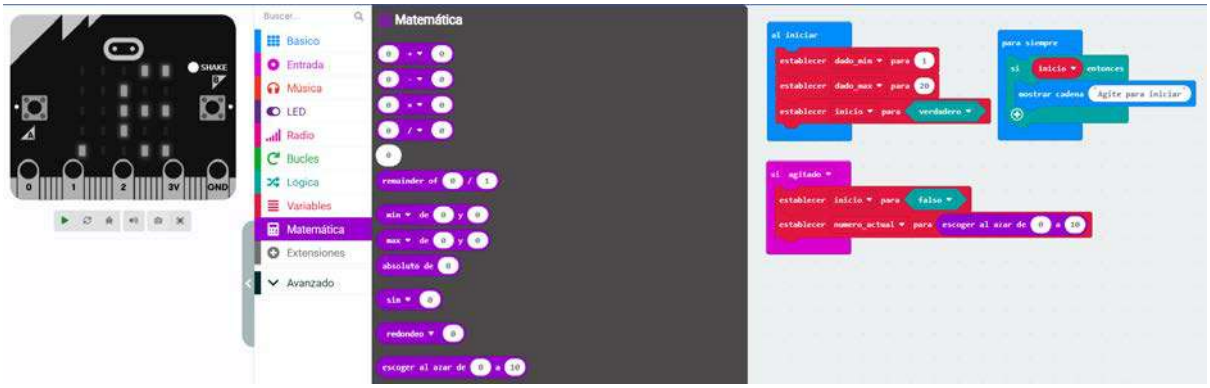
8.1 – Arrastrar el bloque "**variables/establecer inicio para..**" dentro del bloque "**si agitado**" y establecerlo en "**falso**"



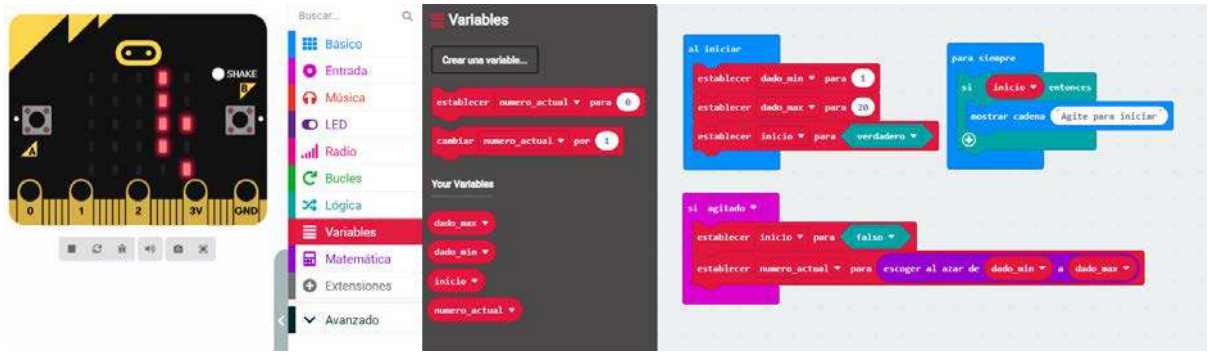
8.2 – Arrastrar el bloque "**variables/establecer número actual para..**" dentro del bloque "**si agitado**"



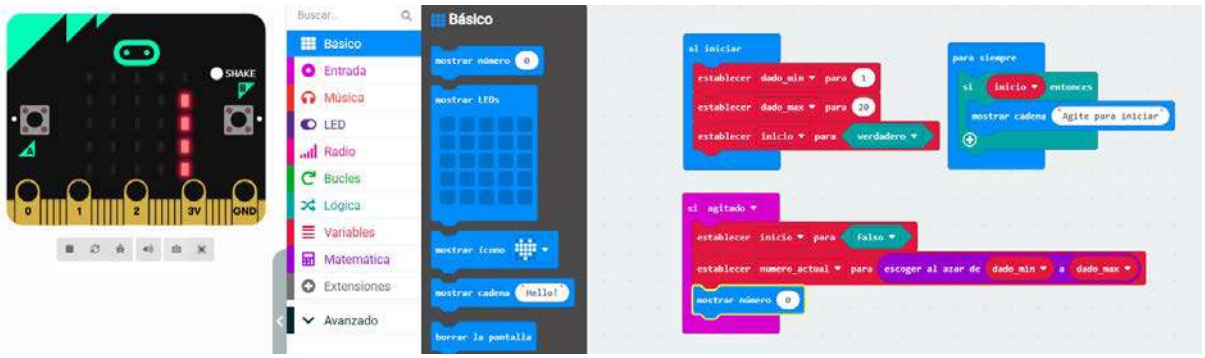
8.3 – Arrastrar el bloque "**matemática/escoger al azar de …**" dentro del bloque "**si agitado /establecer numero_actual para…**"



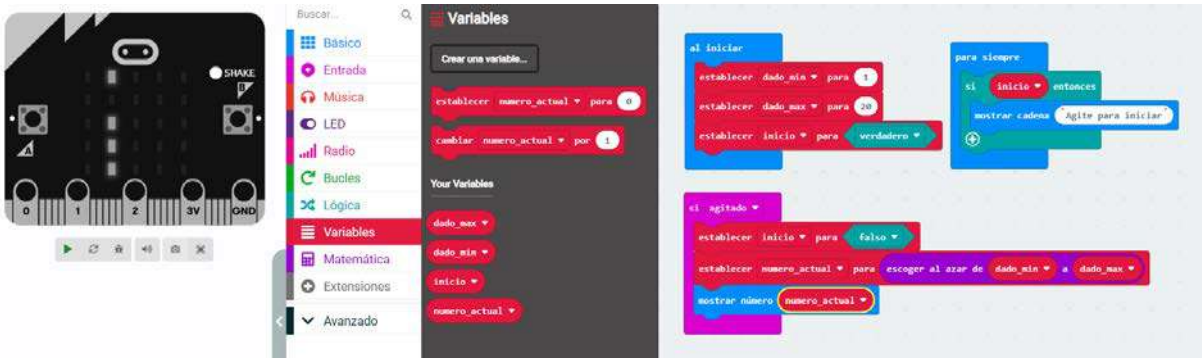
8.4 - Arrastrar los bloque "**variables/dado_min**" y "**variables/dado_max**" dentro del bloque "**si agitado /establecer numero_actual para…/escoger al azar de..**"



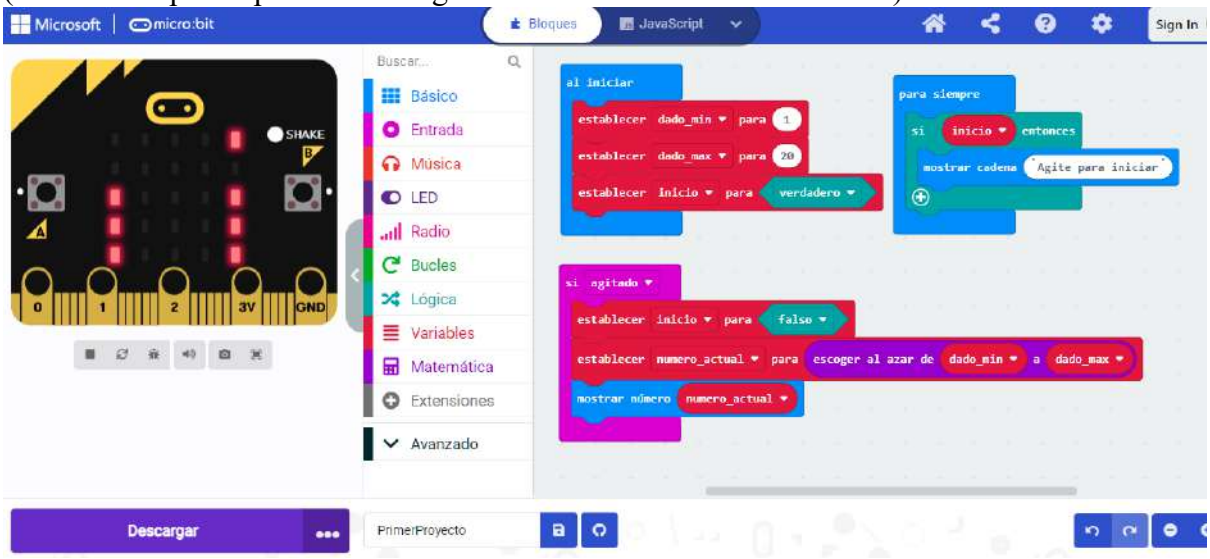
9 – Arrastrar el bloque "**básico/mostrar número**" dentro del bloque "**si agitado**"



9.2 – Arrastrar el bloque "**variables/numero_actual**" dentro del bloque "**si agitado/mostrar número**"



10 – Conectar la MicroBit vía usb y subir el código con el botón "**Descargar**"
(También se puede probar el código tocando "**shake**" en el simulador)



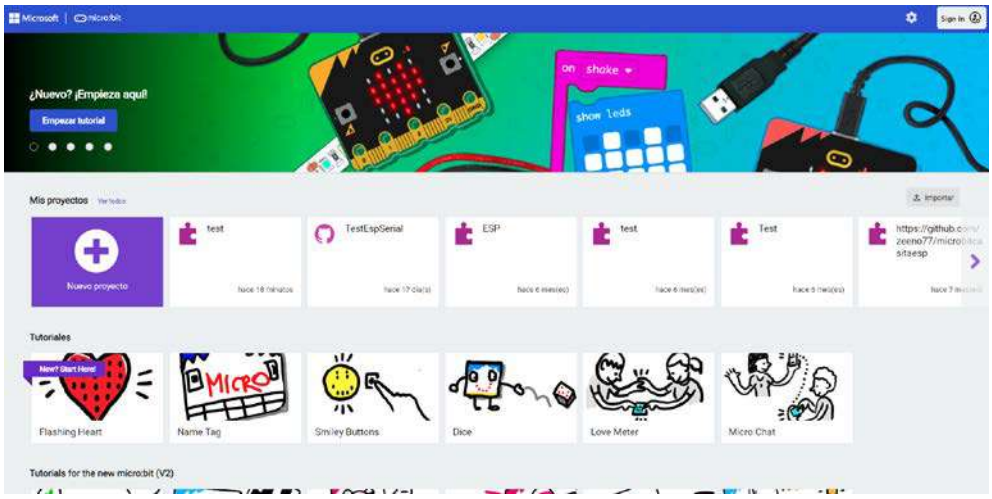
Ejercicio C

Vamos a utilizar la MicroBit con sensores externos para crear una puerta que se abre automáticamente al detectar movimiento.

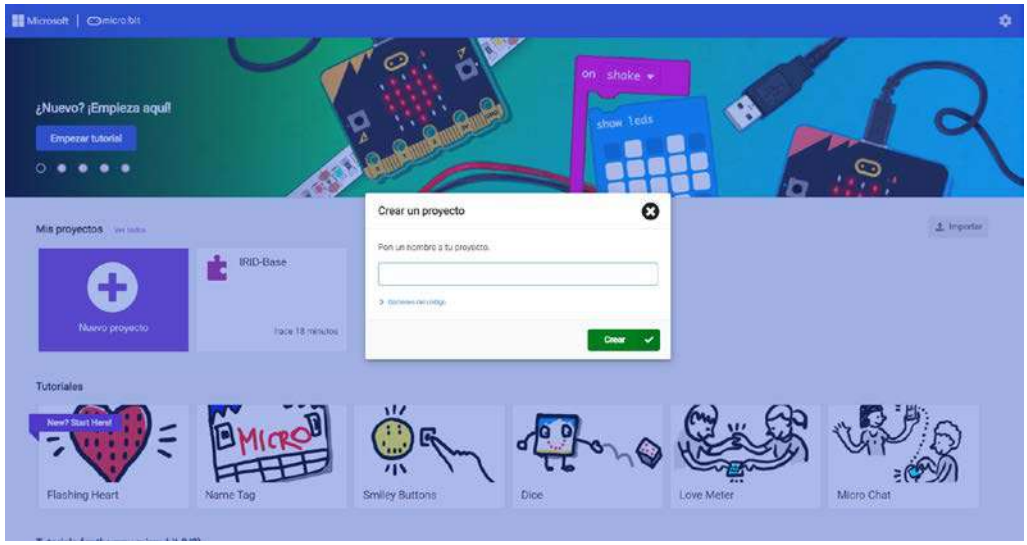
Conceptos por aplicar

- programación en bloque
- uso de variables
- estructura condicional
- uso de diferentes familias de bloques
  - por ejemplo:
    - bloques de control de servo

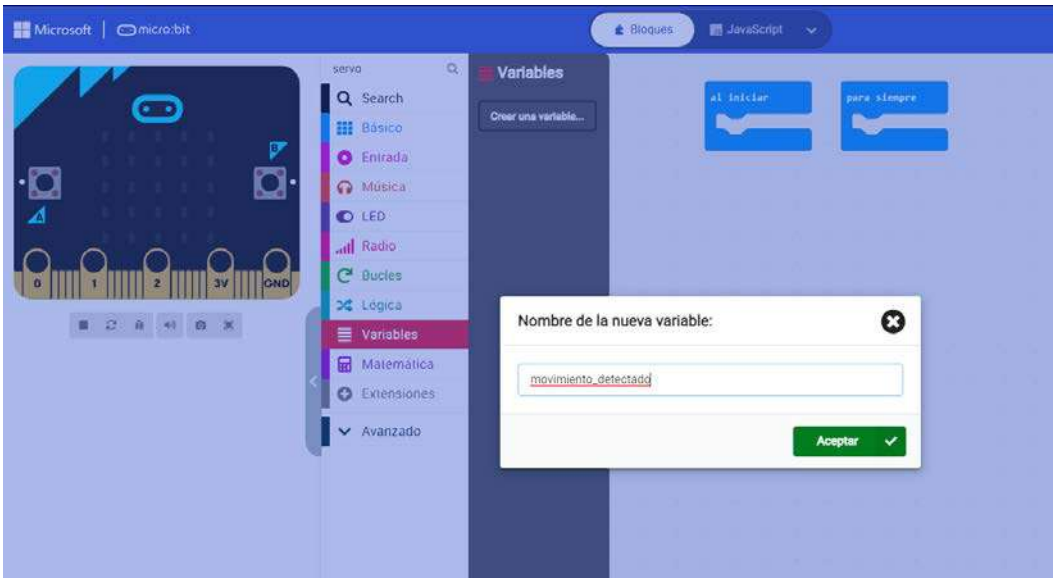
1- Ingresar a https://makecode.microbit.org/ y seleccionar **Nuevo proyecto**



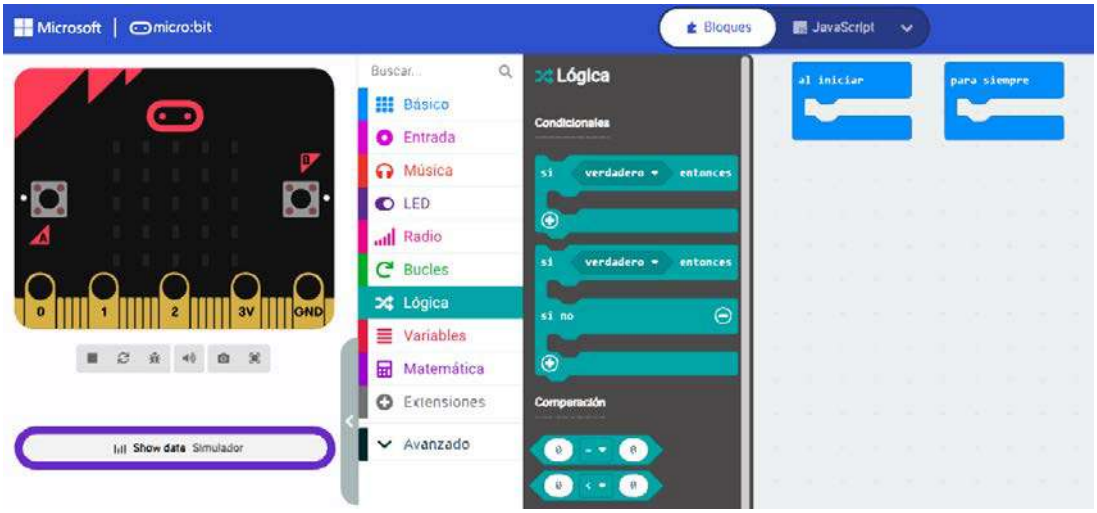
2- Ponerle un nombre al proyecto (por ejemplo “PrimerProyecto”) y presionar **Crear**



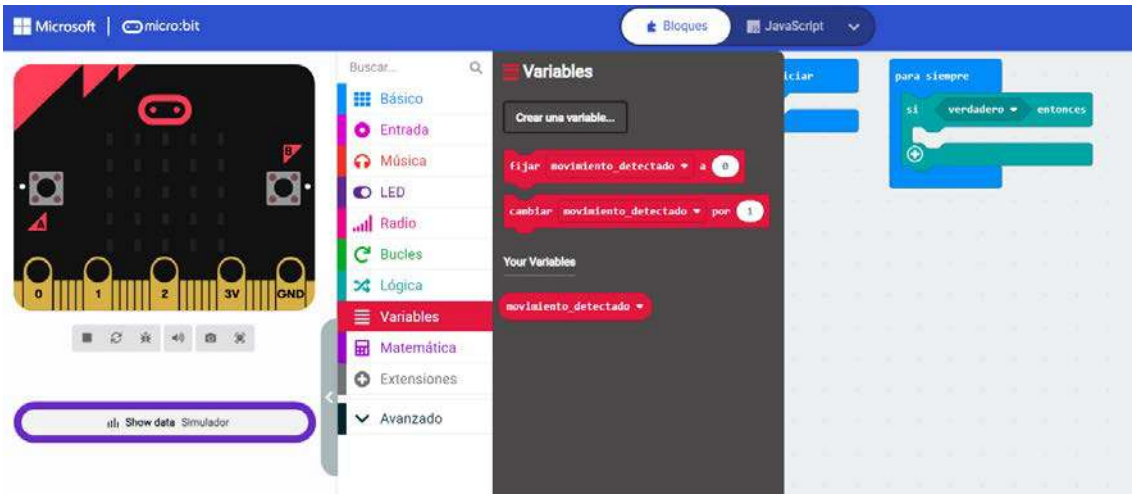
3- Crear una variable llamada **movimiento_detectado**



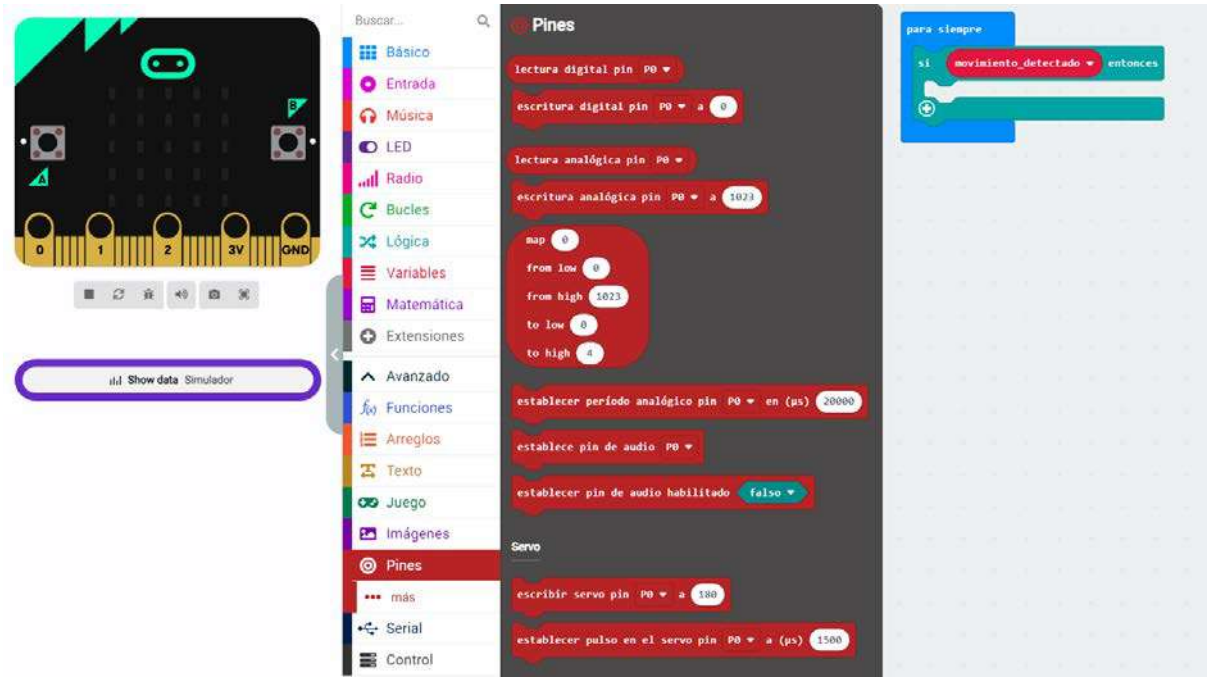
4- Arrastrar el bloque "**Logica/Si verdadero entonces**" dentro del bloque "**para siempre**"



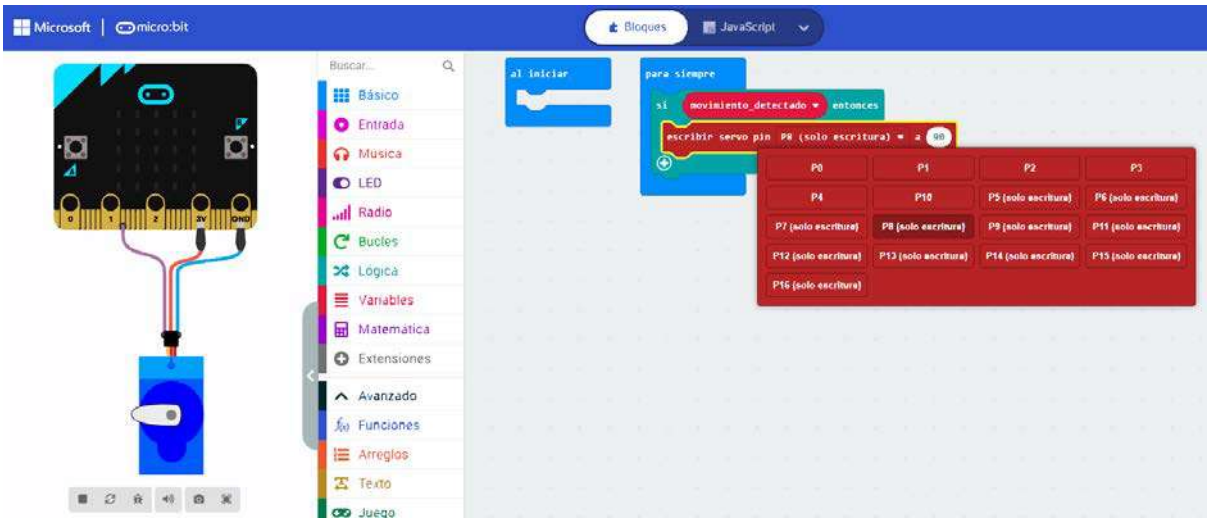
5- Arrastrar el bloque "**Variables/movimiento_detectado**" dentro del bloque "**para siempre/ si verdadero entonces/verdadero**"



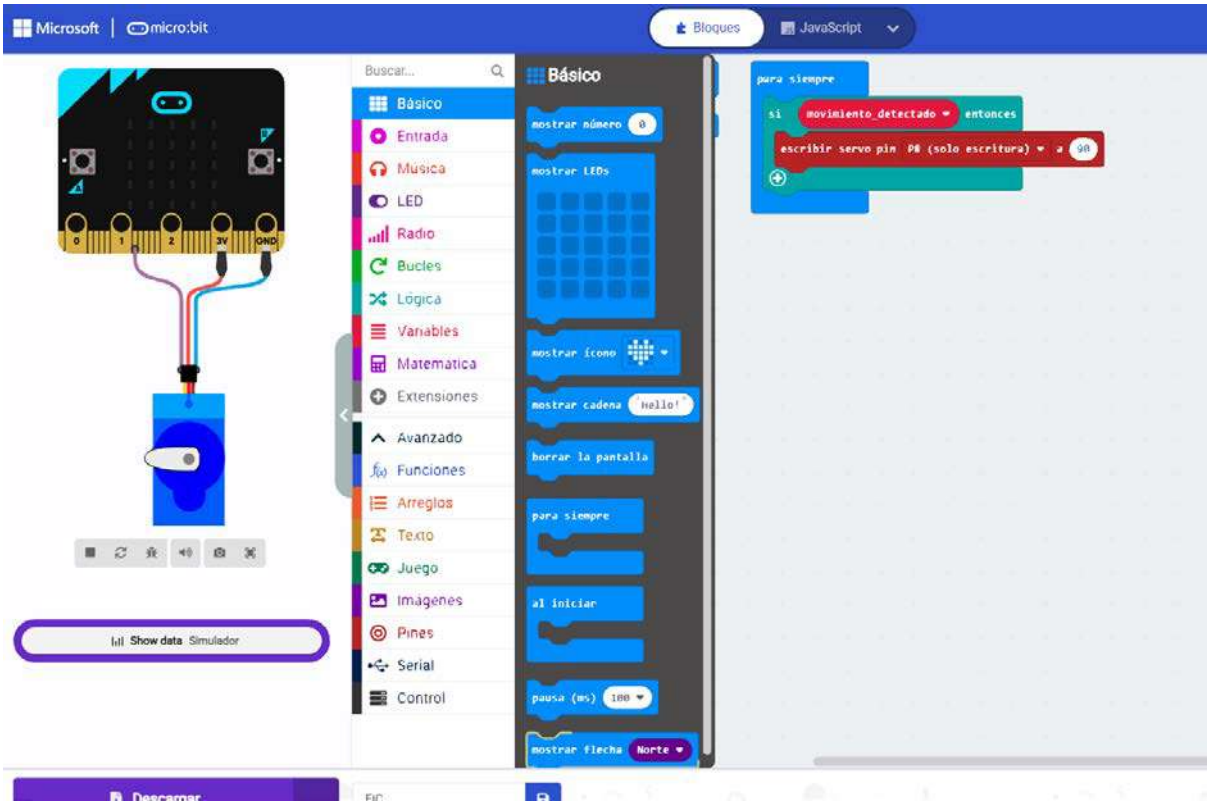
6-Arrastrar el bloque "**avanzado/pines/escribir servo pin**" dentro de "**para siempre/ si verdadero entonces/**"



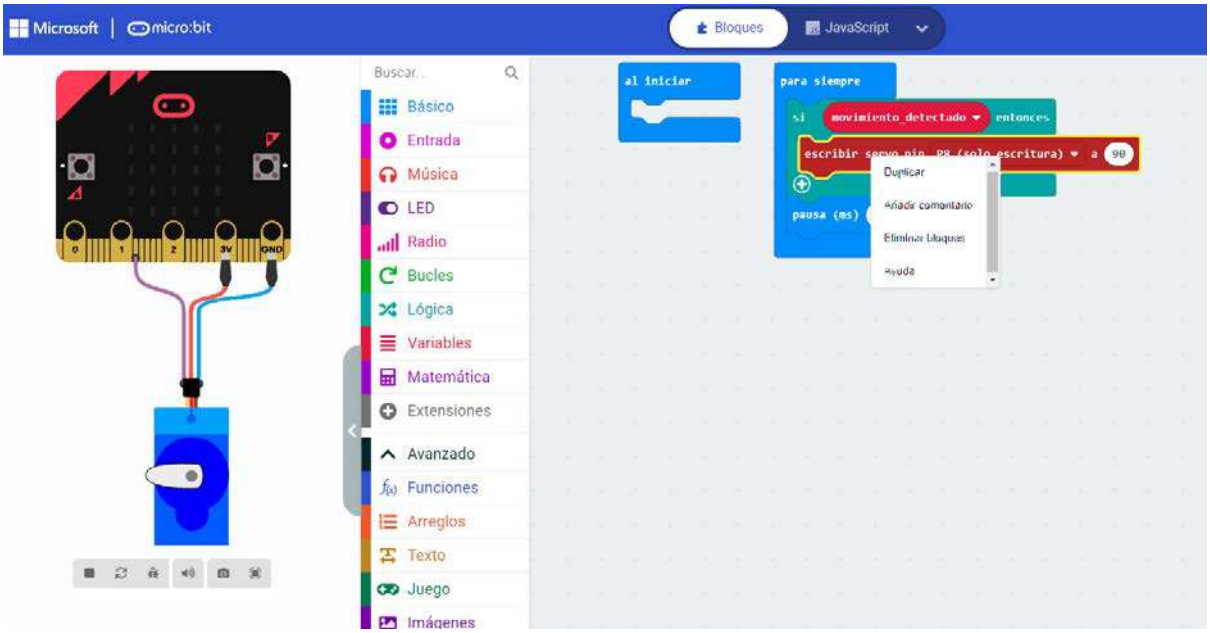
7- Establecer el pin en 8 y el ángulo a 90



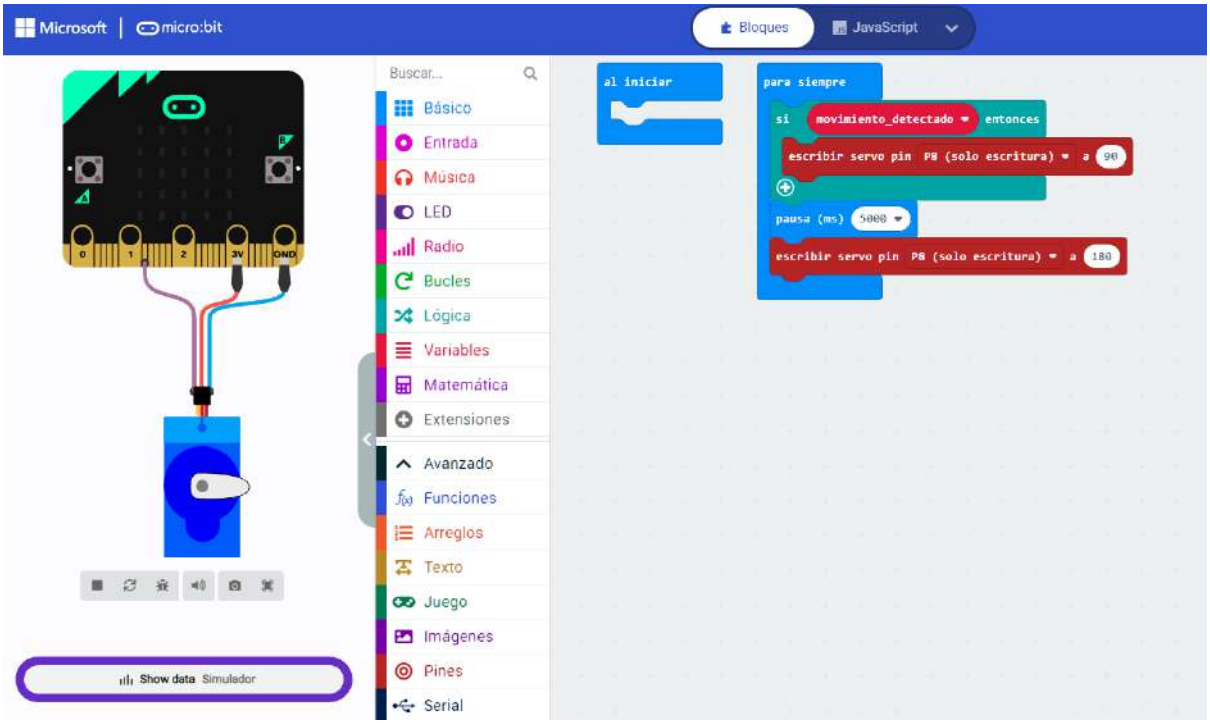
8- Arrastrar el bloque “**Basico/pausa(ms)** ” al final del bloque “**para siempre**” y configurar la pausa en 5 segundos.



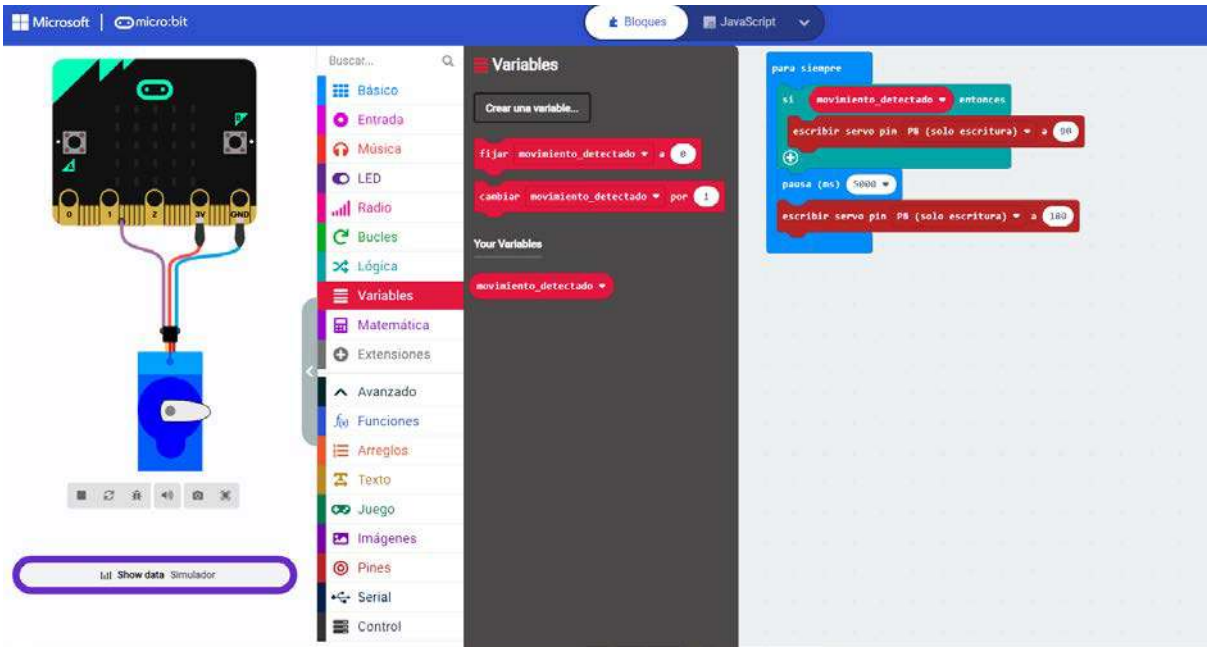
9- Tocar botón derecho sobre el bloque “**para siempre/ si movimiento_detectado entoces/ escribir servo p8 a 90**” y duplicarlo



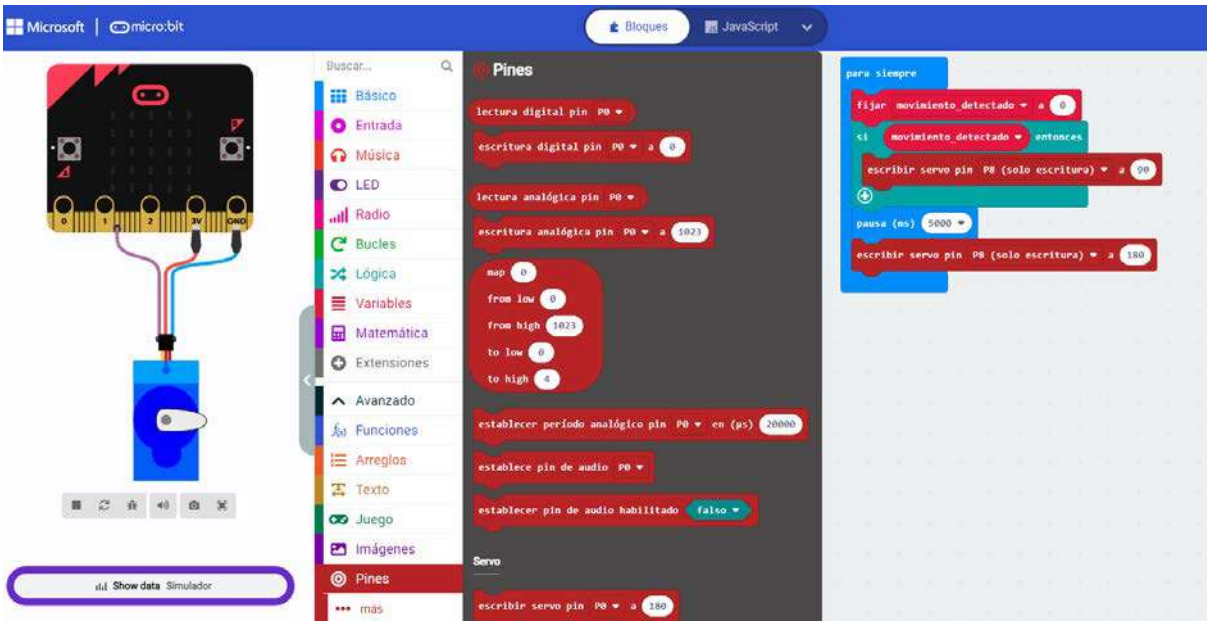
10- Arrastrar el bloque duplicado al final del bloque “**para siempre**” y cambiar el valor de 90 a 180.

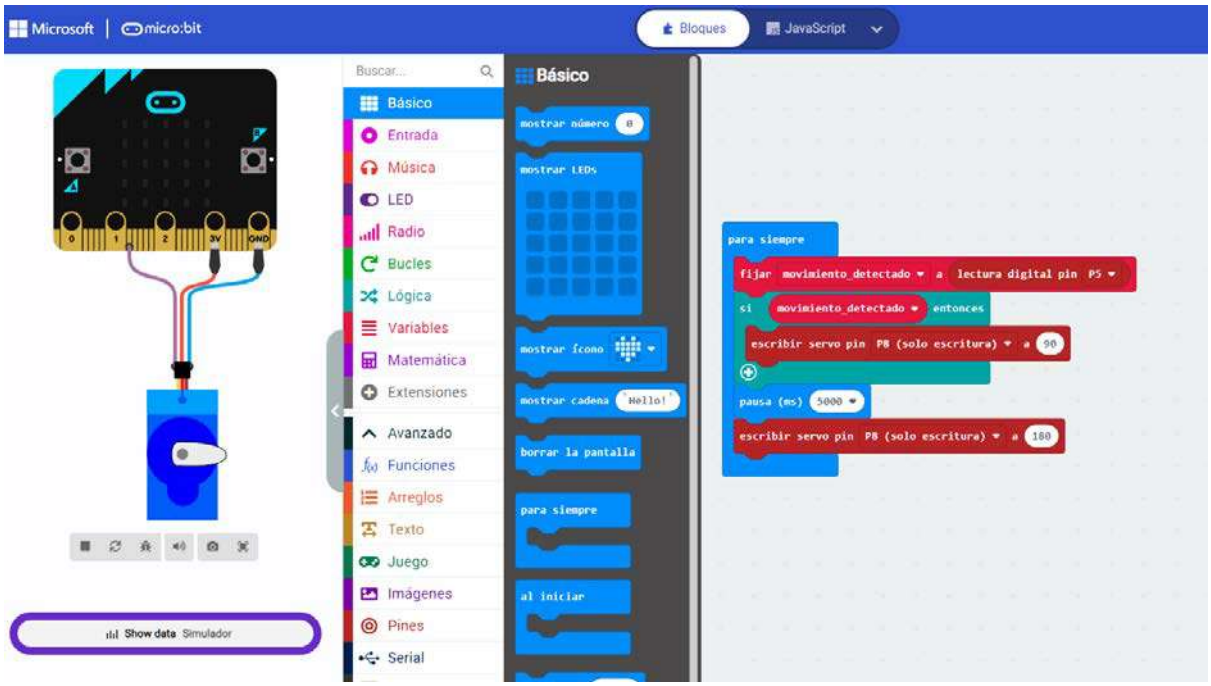


11- Arrastrar el bloque “**variables/ fijar movimiento_detectado a**” al principio del bloque “**para siempre**”



12- Arrastrar el bloque "**avanzado/pines/ lectura digital pin**" dentro de "**para siempre/ fijar movimiento_detectado/0**" y cambiar **P0** a **P5**





Segundo encuentro

Fecha de realización: martes 07 de noviembre de 2023

Objetivos del encuentro

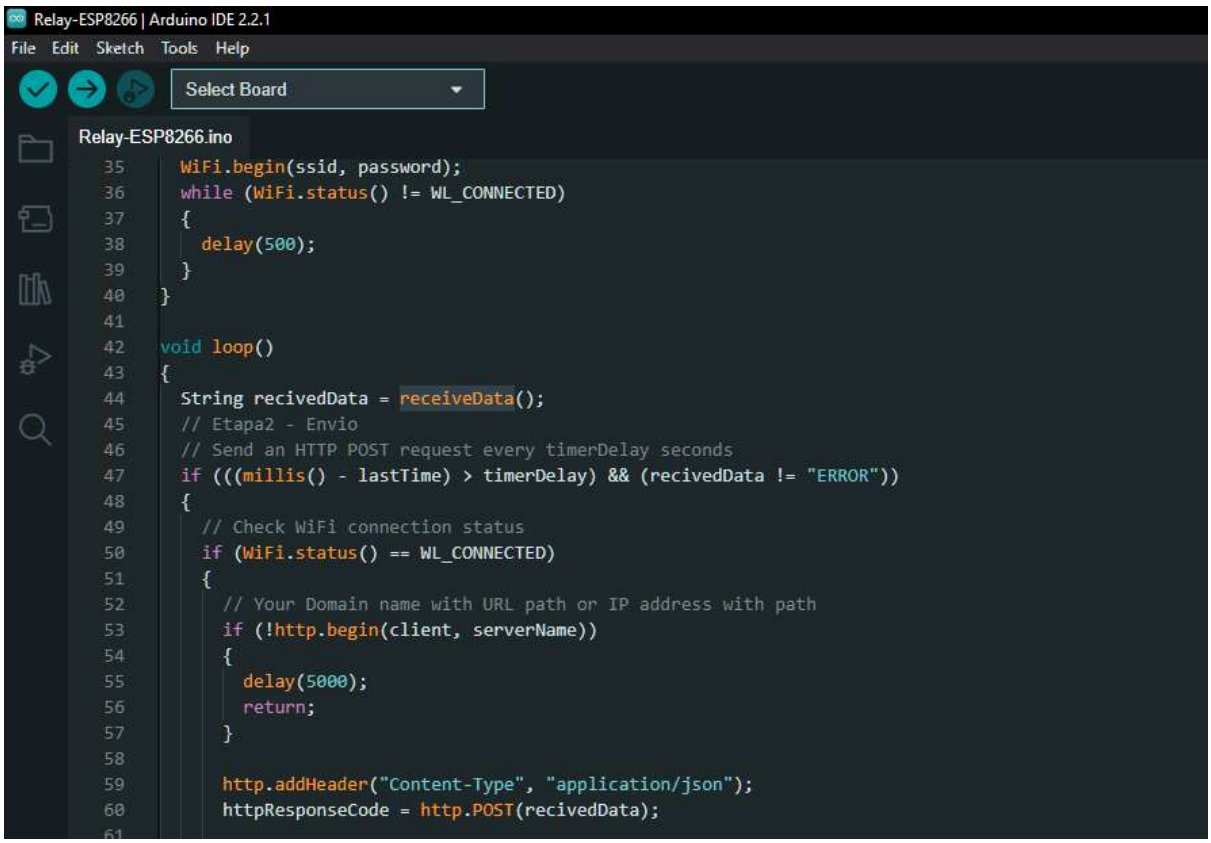
- Poner en práctica conceptos de programación e bloques usando sensores externos
- Trabajar aspectos de comunicación de datos usando ESP8266
- Enviar datos a un servidor web usando la programación en bloques
- Visualización de datos generados por sensores

Primeros pasos

Para poder llevar adelante la actividad de este encuentro, es necesario realizar primero algunas acciones.

1) Poner en marcha el servidor
    a) Para poner en marcha el **servidor** para la recepción, persistencia y visualización de datos es necesario instalar y configurar Dockers y Grafana siguiendo los pasos detallados en el documento llamado "**IRID Instructivo para el despliegue de ambiente".**
2) Configurar ESP8266

a) Una vez que se tenga el **servidor** andando procederemos a subir el código necesario para la comunicación con la MicroBit a la ESP8266 mediante el entorno de programación de Arduino siguiendo los pasos del documento llamado *Anexo I: Guía de Configuración de la ESP.*



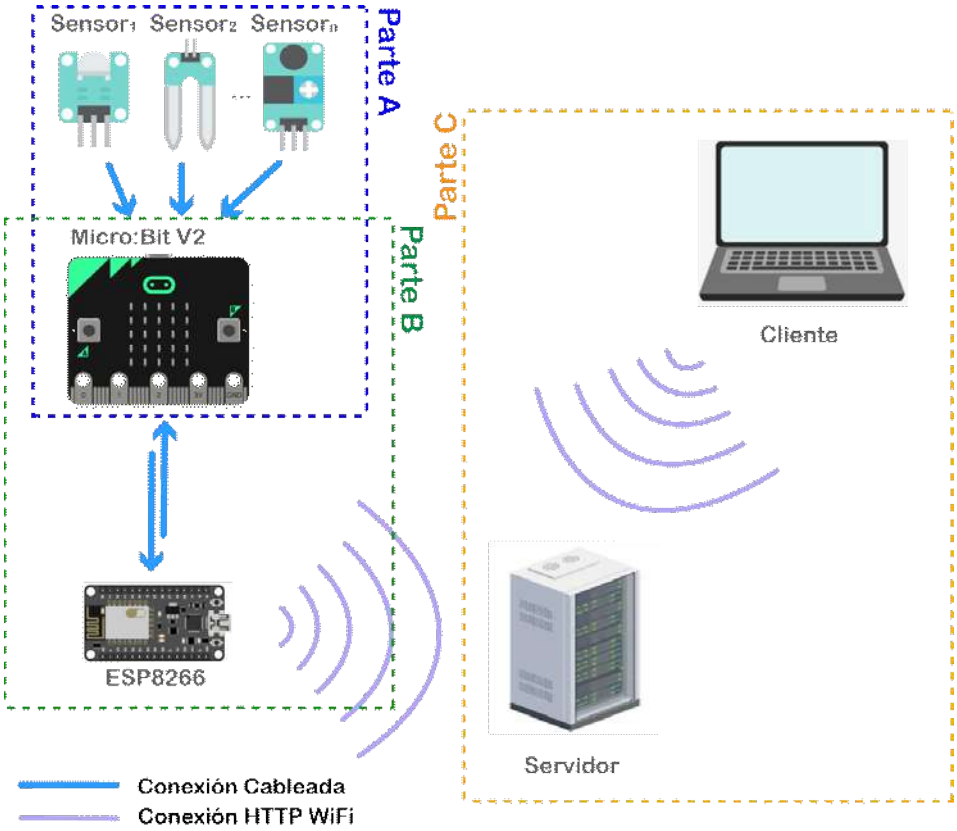
Una vez subido el código a la ESP8266 no es necesario volver a modificarlo, ***a partir de ese punto toda la programación se realizará mediante los bloques de MakeCode.***

Actividades

La actividad se dividirá en 3 partes: A,B y C.

- La parte A consistirá en la programación de distintos sensores conectados a la MicroBit.
- La parte B de la configuración de la ESP8266 mediante bloques y el envío de los datos sensados al servidor.
- La parte C de la visualización de los datos enviados.

A continuación, se muestra una Figura con el hardware que involucra cada parte mencionada

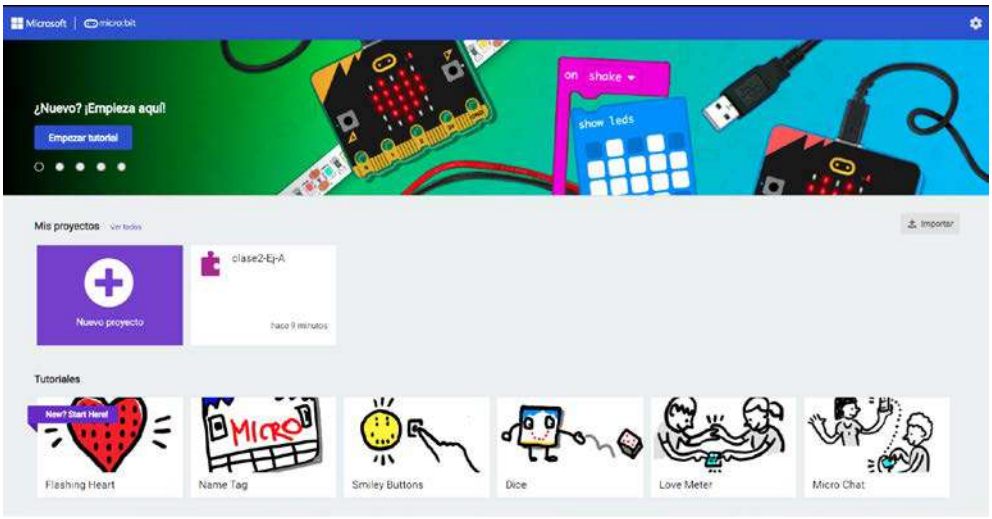


Parte A

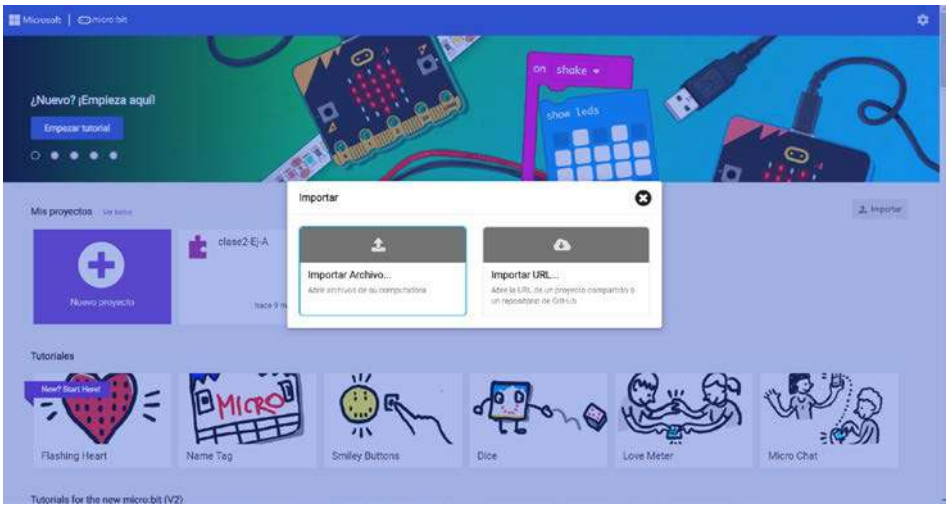
Programación de sensores.
En esta sección vamos a programar acciones de los distintos sensores que posee "la casita." A continuación, iremos describiendo los pasos a realizar

- Importar el proyecto **microbit-IRID-Base.hex** tocando el botón ubicado a la derecha.

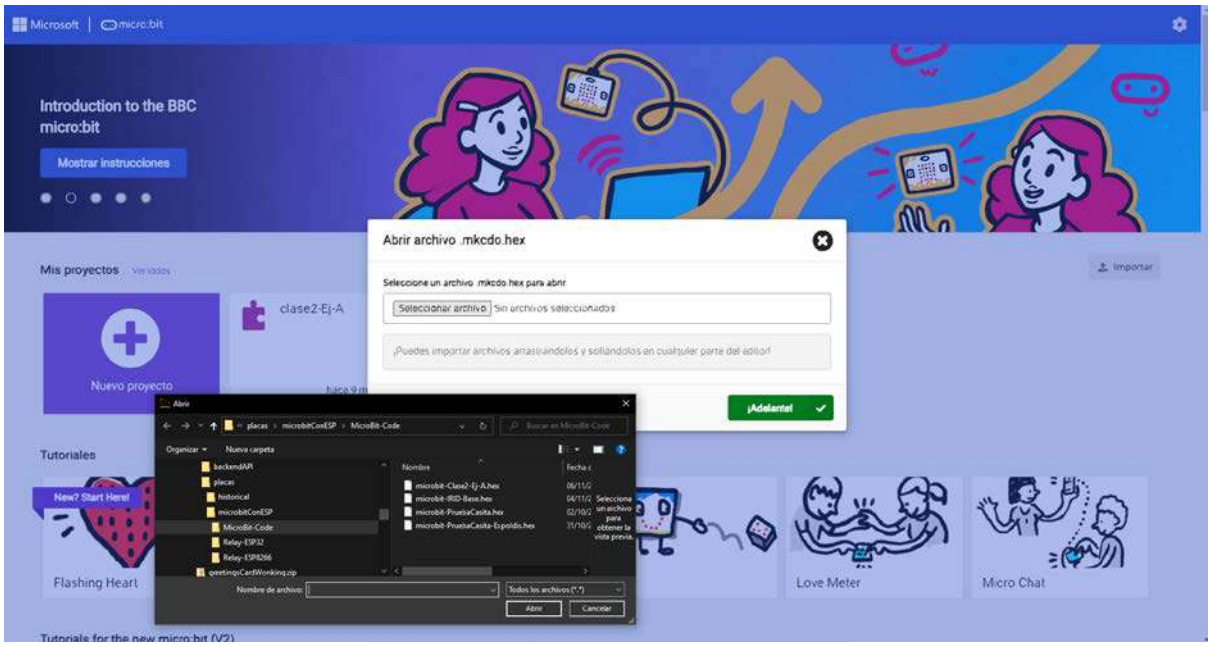
  Esta acción nos dará acceso a bloques personalizados que usaremos en la parte B.



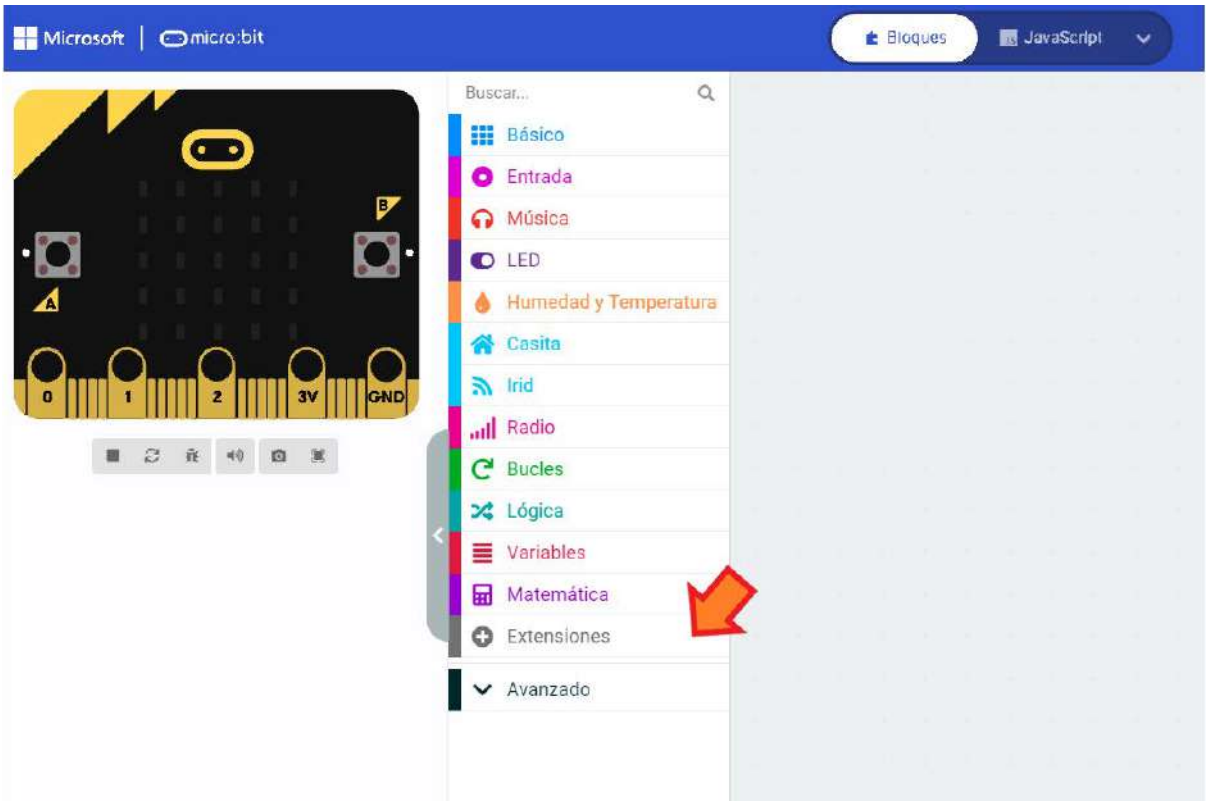
- De las opciones, hacemos clic en “Importar Archivo”



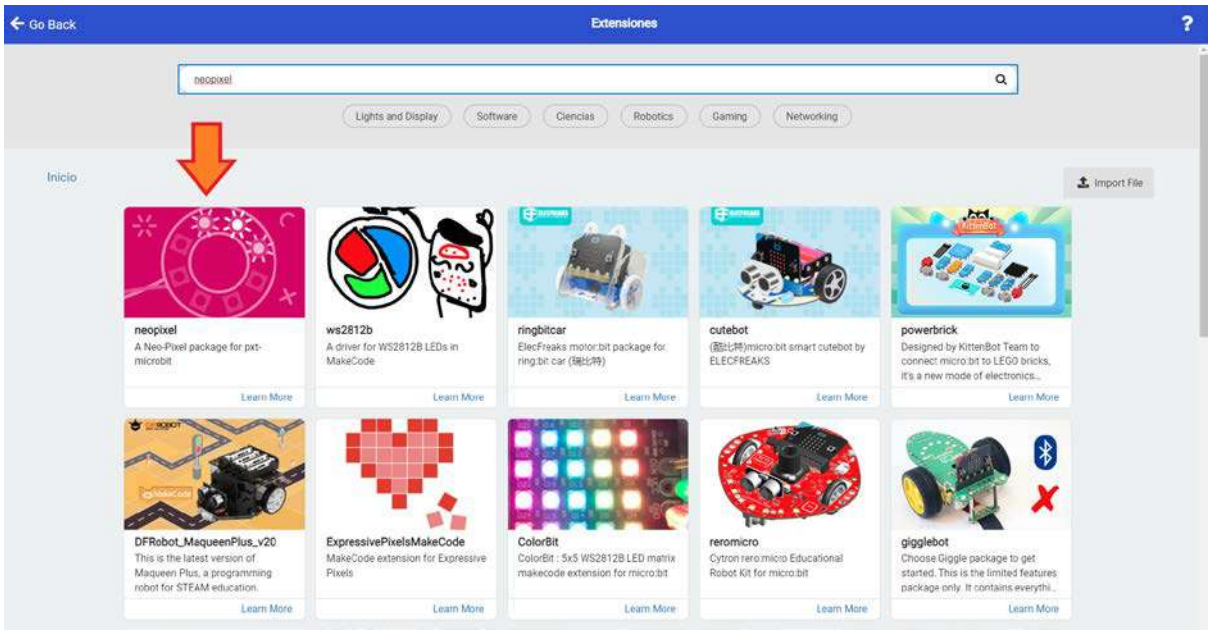
- Seleccionar archivo una vez más, buscamos la ubicación de **microbit-IRID-Base.hex** y presionamos “**adelante**!” para terminar de importar el proyecto.



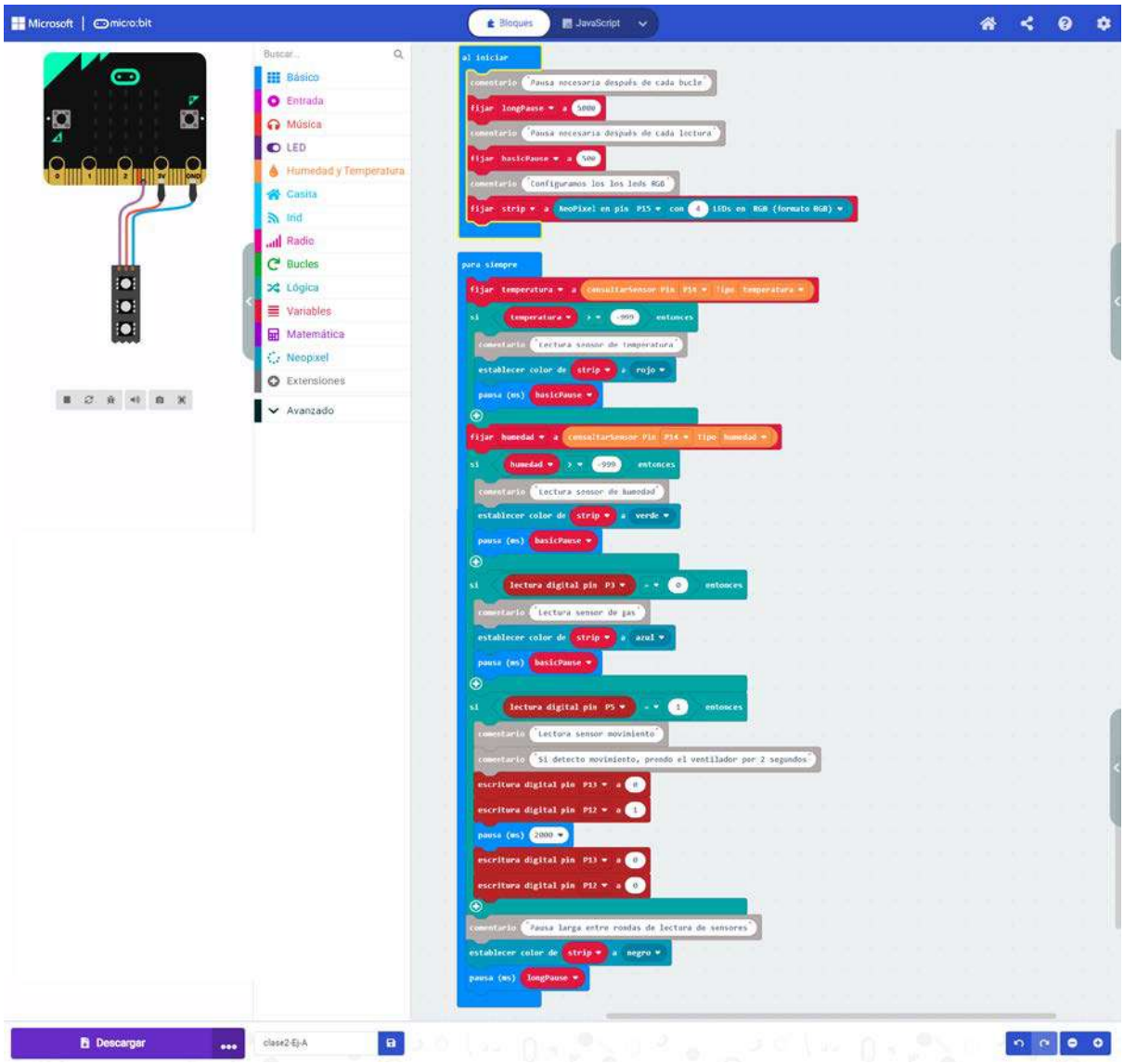
- Luego vamos a importar una librería necesaria para el uso de los led RGB.
    - Vamos a ingresar a Extensiones



    - Escribimos **neopixel** en la barra de búsqueda e importamos la librería tocando sobre el icono



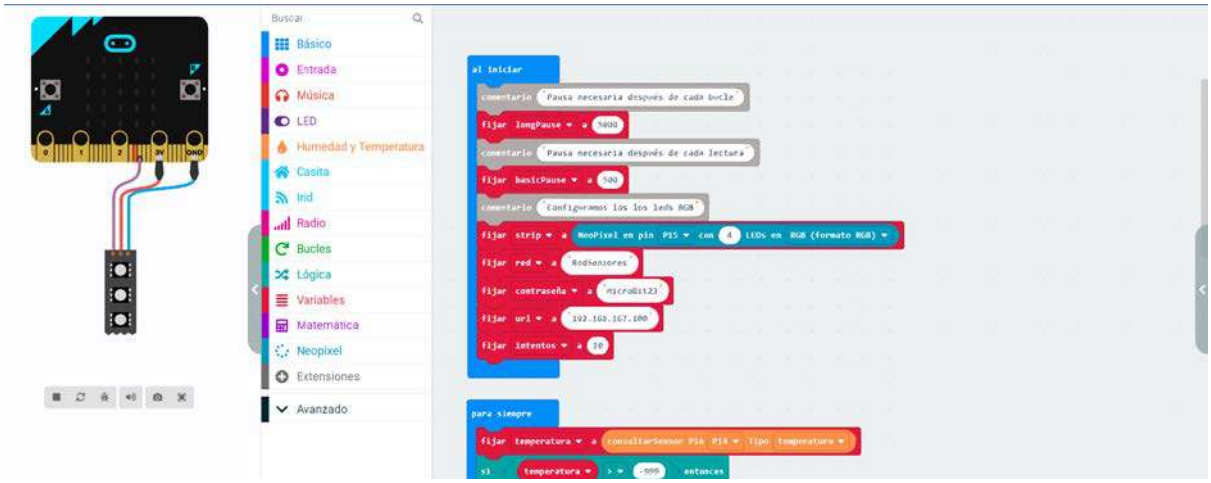
- Copiar el código arrastrando los bloques que correspondan.



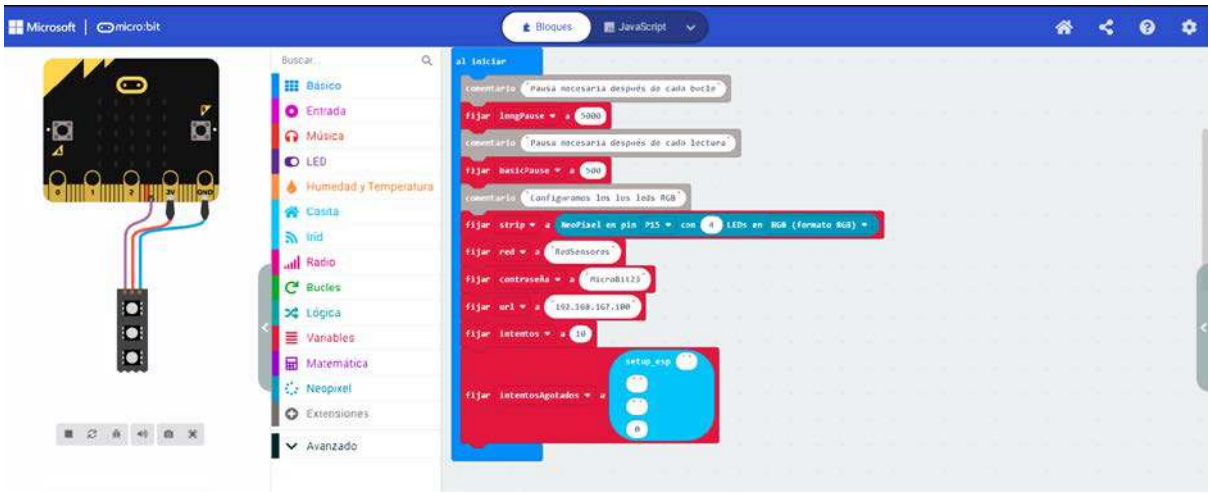
Parte B

En esta parte vamos a conectar la placa ESP8266 encargada de enviar los datos sensados por la MicroBit hacia el servidor.

1. Primero vamos a configurar 4 variables llamadas: red, contraseña, url e intentos.
    - **red**: red wifi a la cual nos vamos a conectar
    - **contraseña**: contraseña de la red wifi.
    - **url**: dirección del servidor al que le vamos a enviar los datos.
    - **intentos**: cantidad de intentos de conexión.

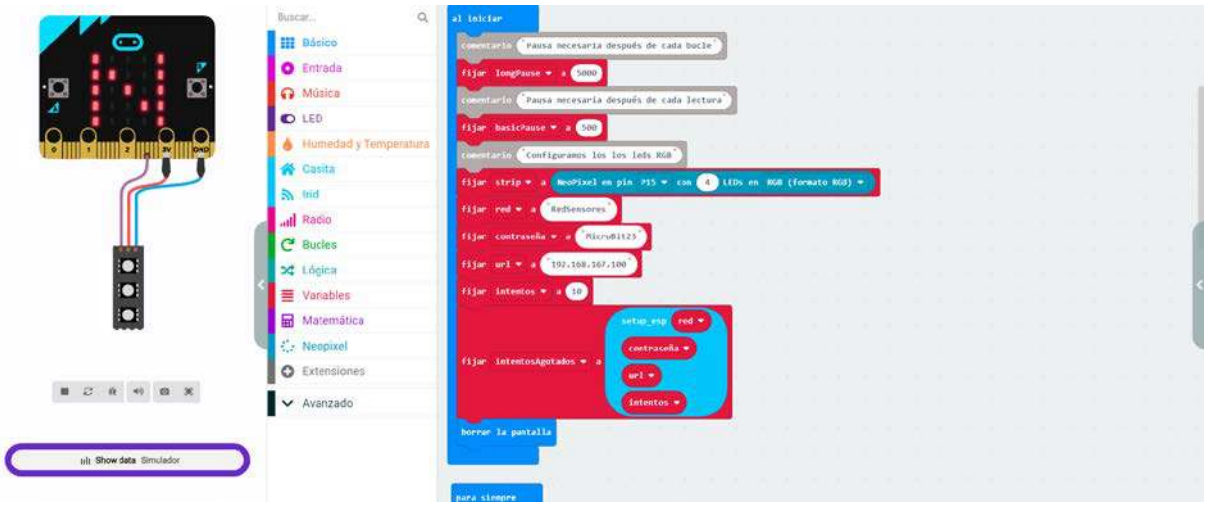


2. Vamos a crear otra variable que se llame ***intentosAgotados*** y vamos a fijarla con el bloque **Irid/setup_esp**



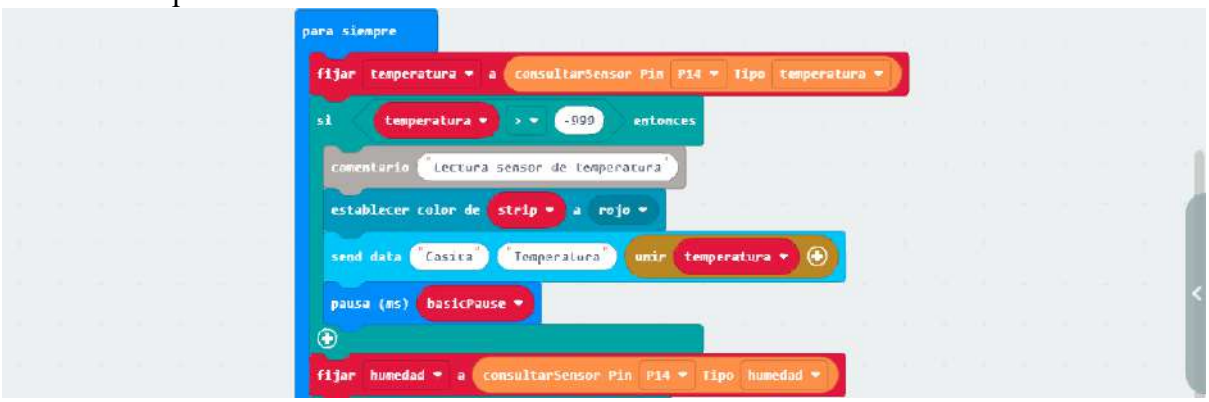
3. Arrastrar las variables red, contraseña, url e intentos, en ese orden, dentro de los campos del bloque **"setup_esp".**

4. Borrar la pantalla al final del bloque "**al iniciar**"



5. Enviar los datos sensados utilizando el bloque "**Irid/send data**".
    - El primer campo va a contener de donde vienen los datos,
    - El segundo campo que sensor tomó la medición y
    - El tercer campo es el valor sensado.

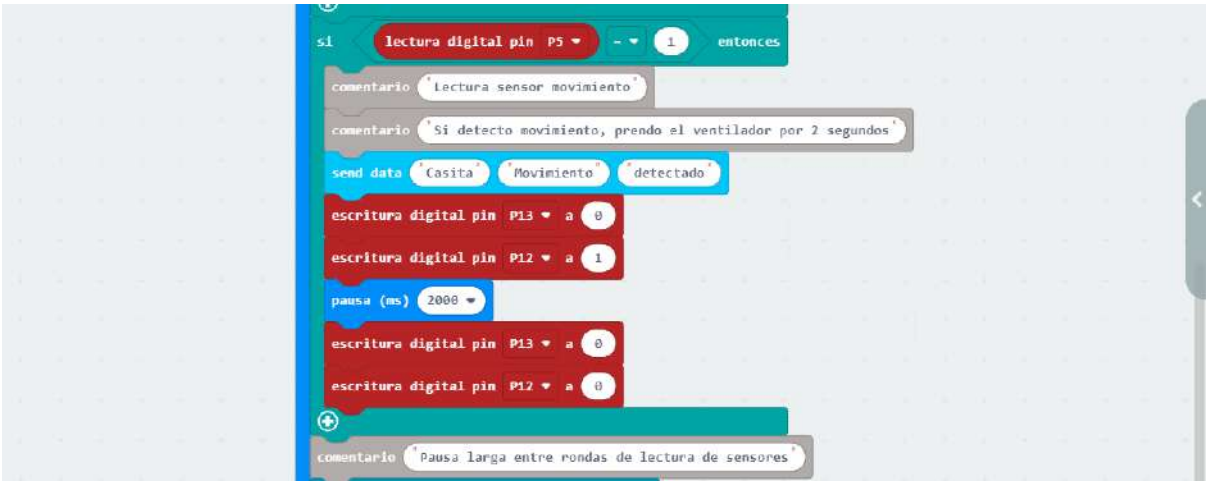
Envío de temperatura



Envío de humedad

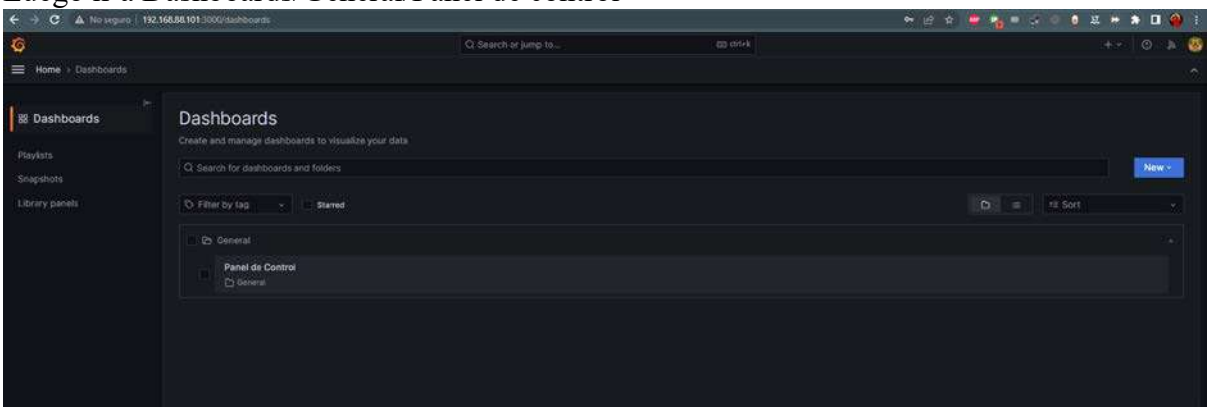
Envío de detección de gas

Envío de detección de movimiento

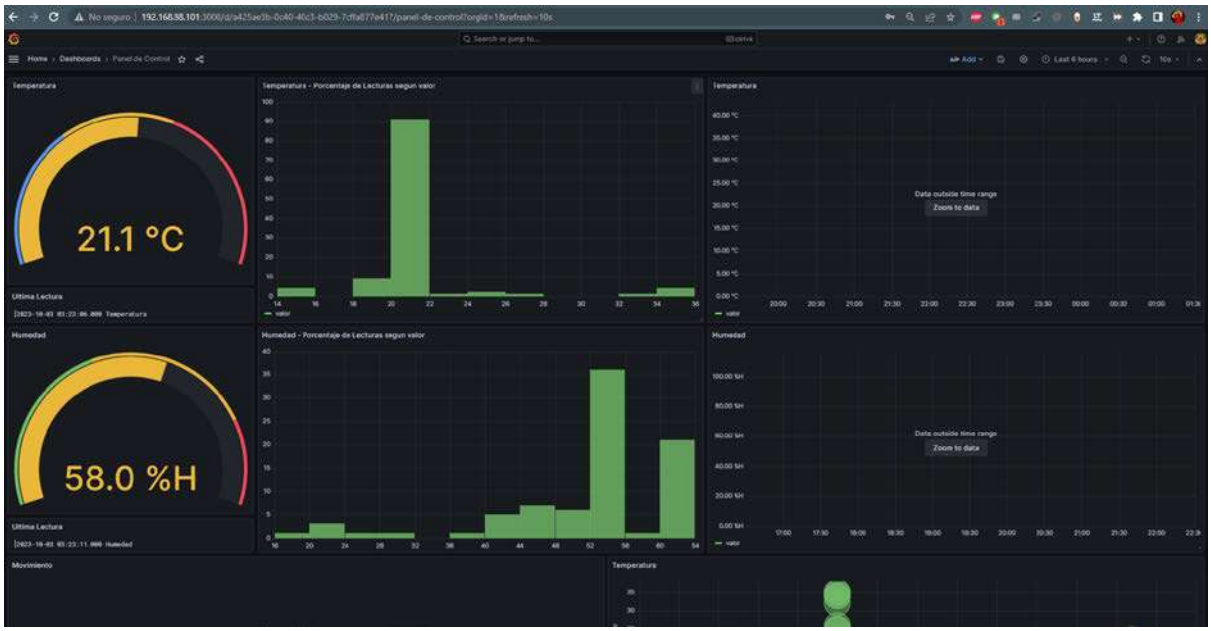


Parte C

Para la visualización de datos vamos a ingresar la dirección del servidor en un navegador en el puerto 300.
En el caso del ejemplo sería: http://192.168.167.100:3000/
Luego ir a Dashboards/General/Panel de control



Desde este panel se podrán visualizar los datos en tiempo real



Tercer encuentro

Fecha de realización: lunes 13 de noviembre de 2023

Objetivos del encuentro

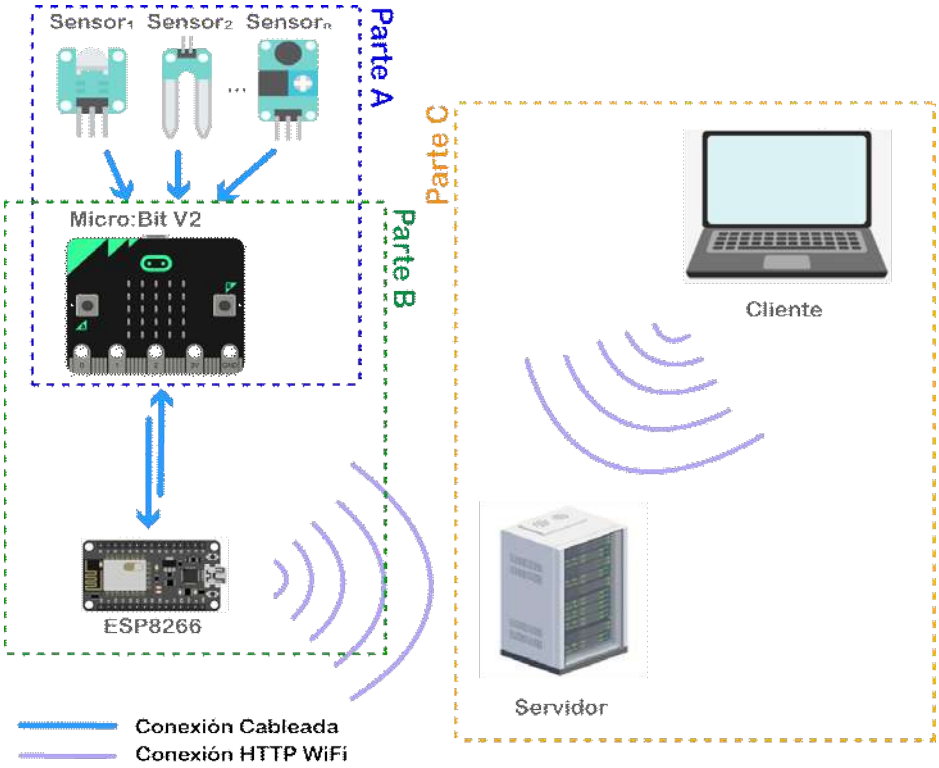
- Finalizar las actividades iniciadas en el segundo encuentro
- Realizar encuestas individuales sobre las actividades propuestas en el segundo encuentro
- Presentar aspectos técnicos de la arquitectura usada
- Cierre del taller

Actividades propuestas en el segundo encuentro

La actividad se dividirá en 3 partes: A,B y C.

- La **parte A** consistirá en la programación de distintos sensores conectados a la MicroBit.
- La **parte B** de la configuración de la ESP8266 mediante bloques y el envío de los datos sensados al servidor.
- La **parte C** de la visualización de los datos enviados.

A continuación, se muestra una Figura con el hardware que involucra cada parte mencionada



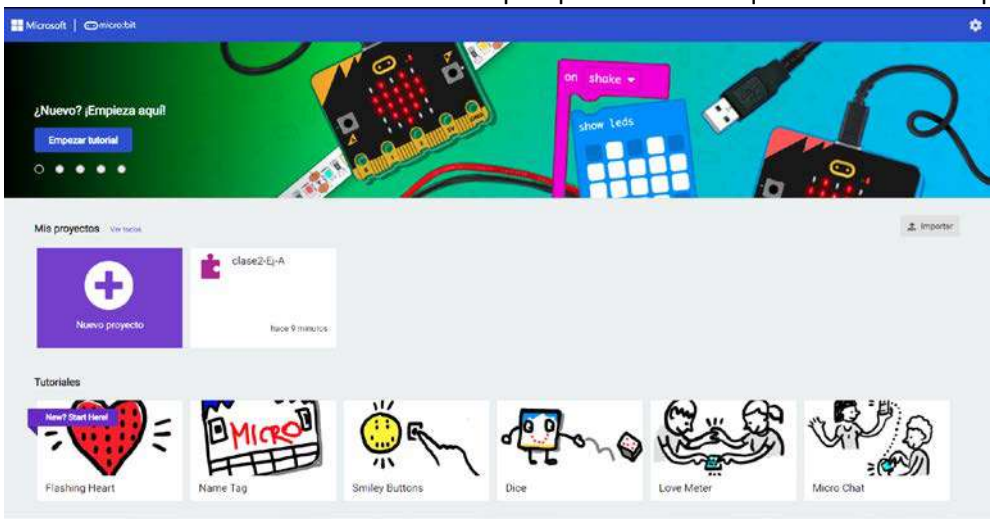
Parte A

Programación de sensores.

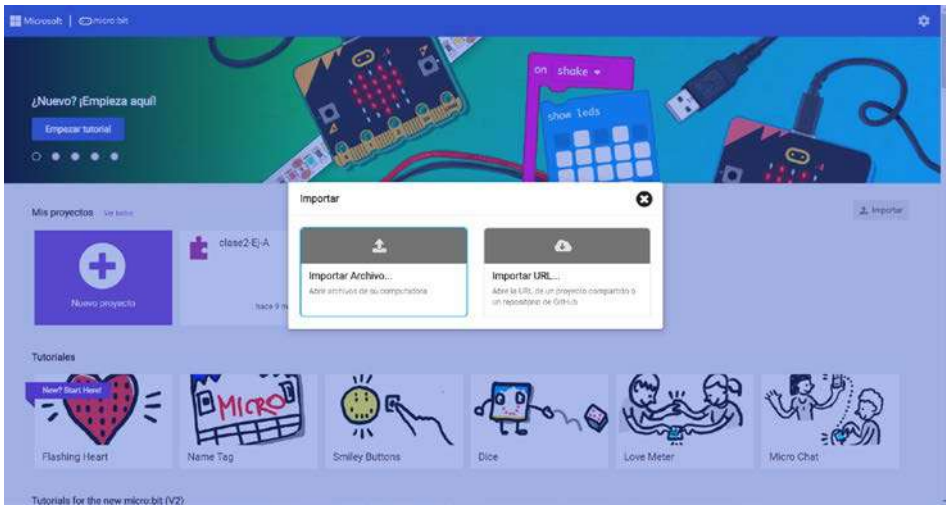
En esta sección vamos a programar acciones de los distintos sensores que posee “la casita.” A continuación, se irán describiendo los pasos a realizar

- Importar el proyecto **microbit-IRID-Base.hex** tocando el botón ubicado a la derecha.

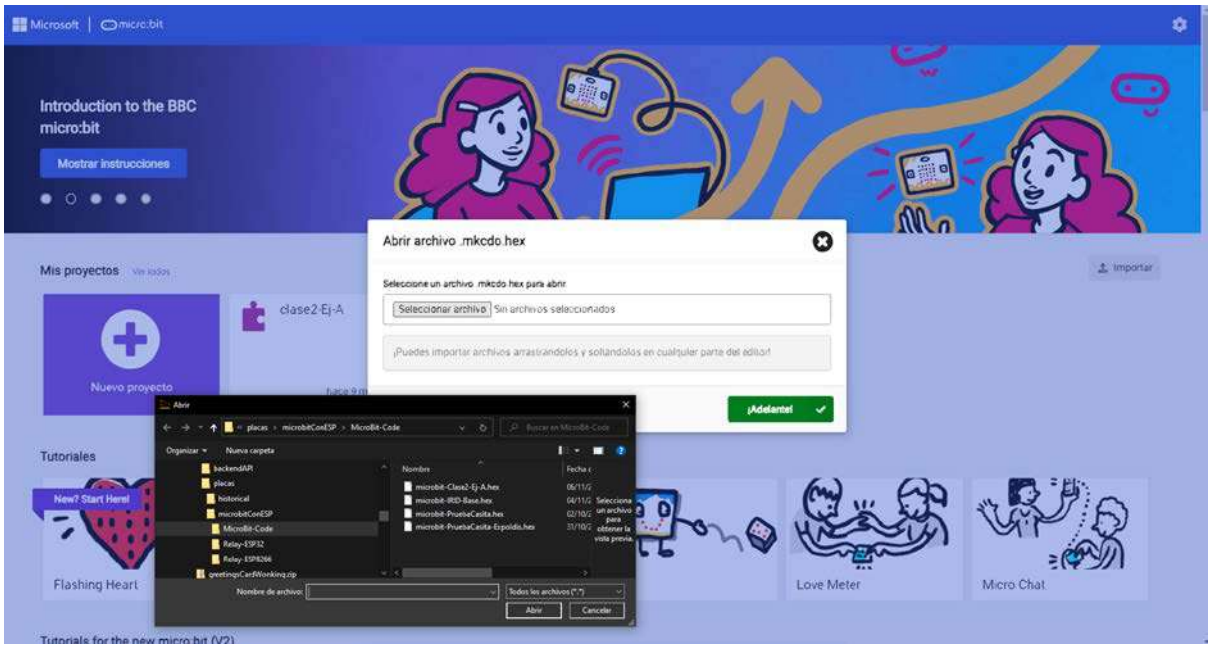
  Esta acción nos dará acceso a bloques personalizados que usaremos en la parte B.



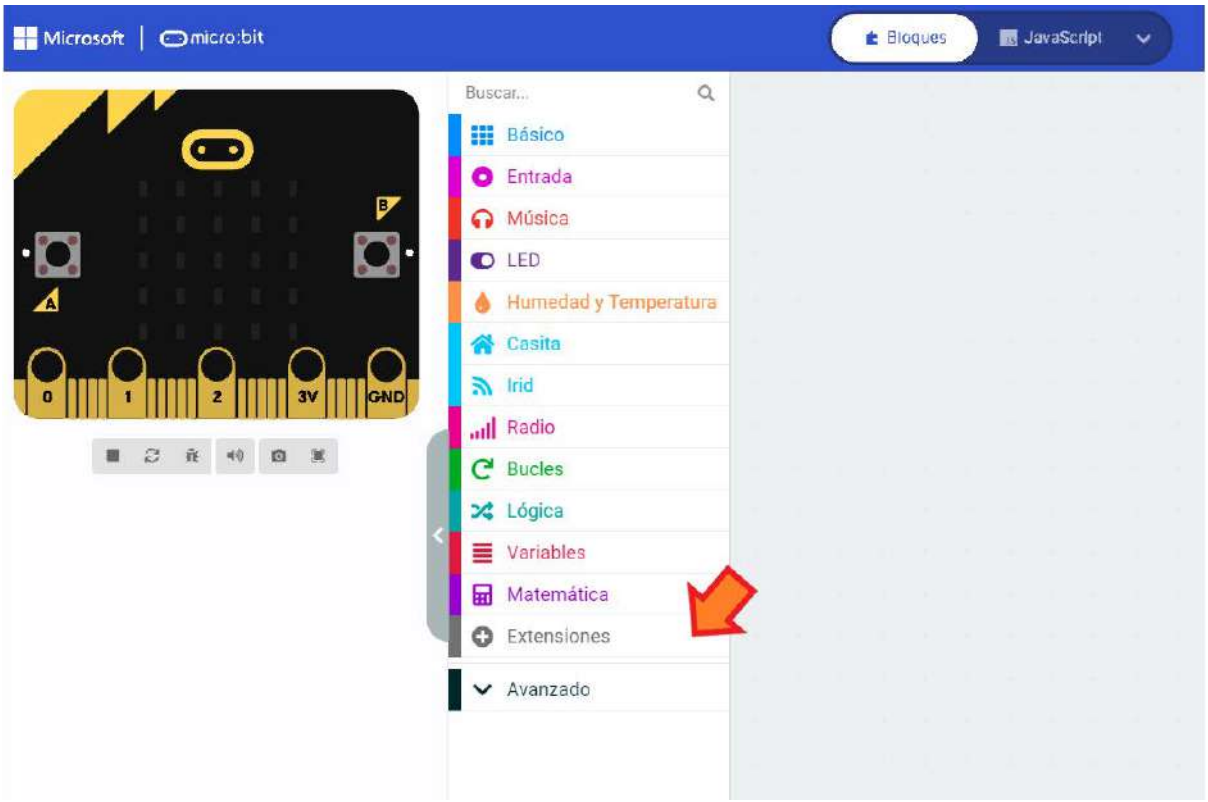
- De las opciones, hacemos clic en “Importar Archivo”



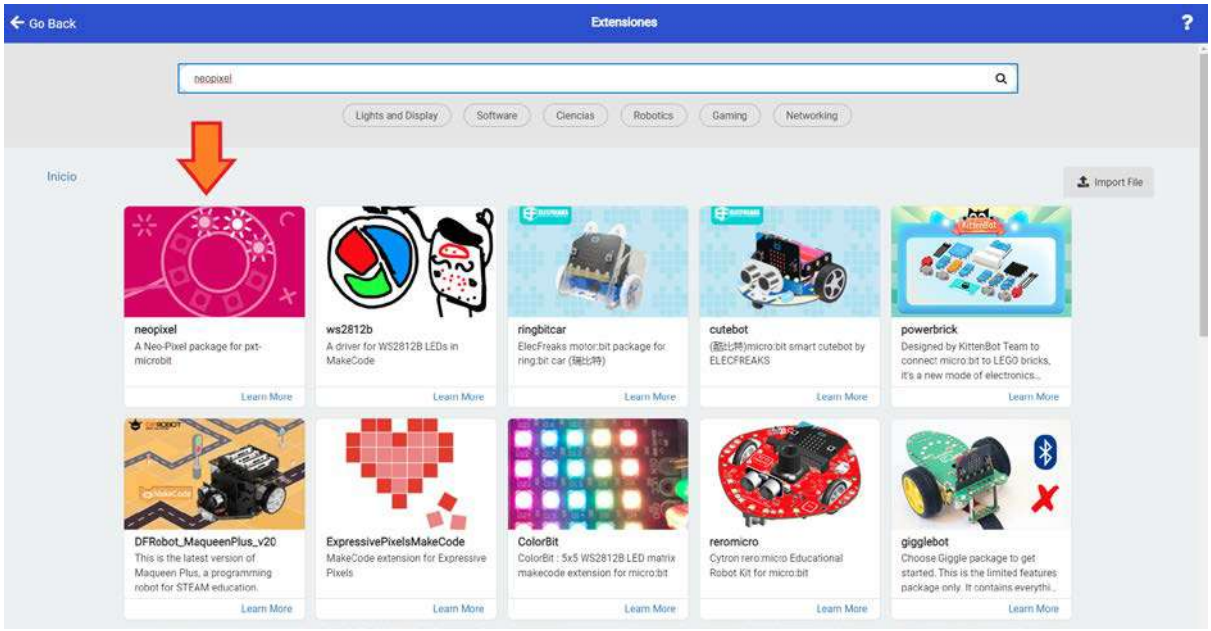
- Seleccionar archivo una vez más, buscamos la ubicación de **microbit-IRID-Base.hex** y presionamos "**adelante!**" para terminar de importar el proyecto.



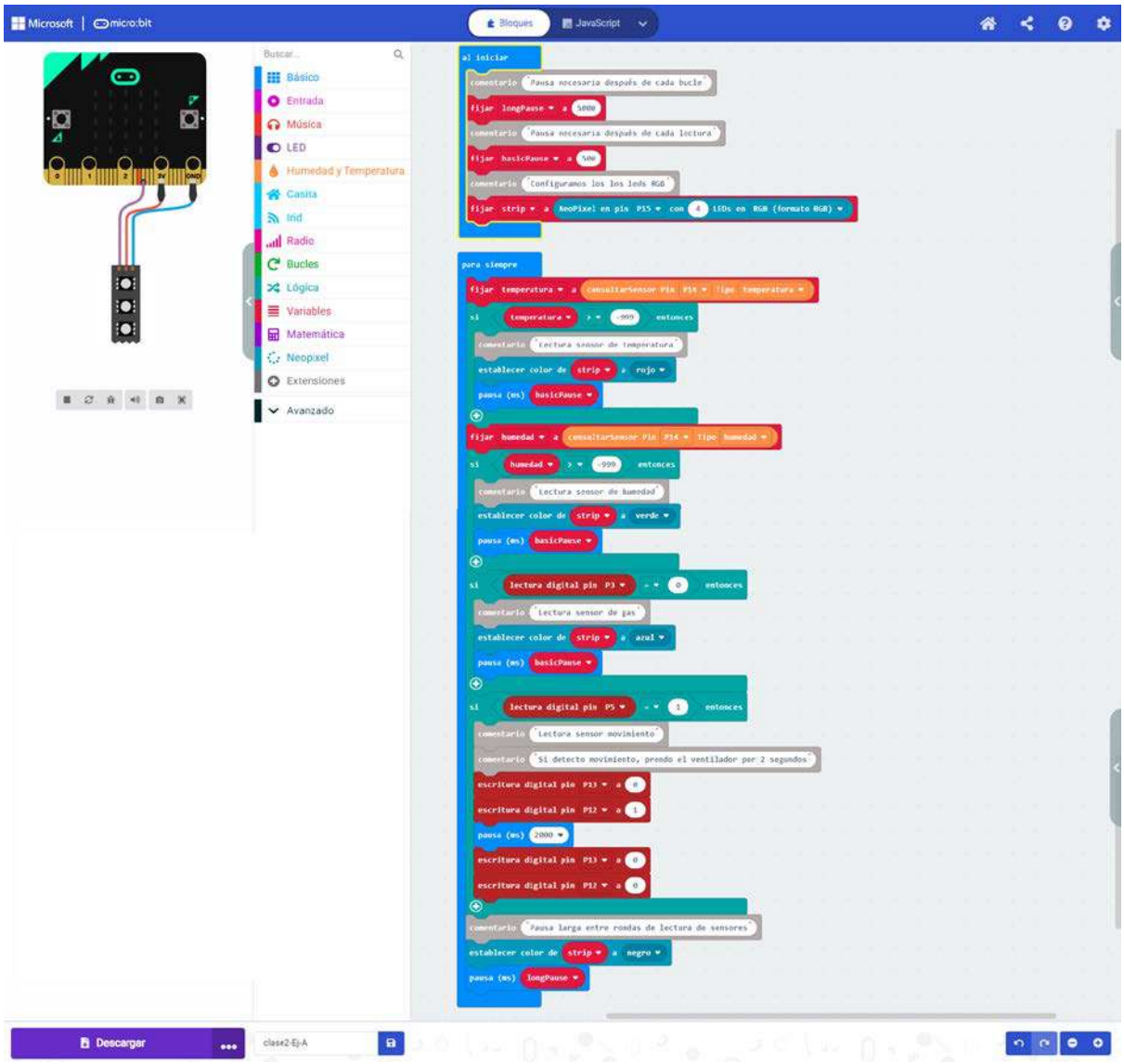
- Luego vamos a importar una librería necesaria para el uso de los led RGB.
    - Vamos a ingresar a Extensiones



- Escribimos **neopixel** en la barra de búsqueda e importamos la librería tocando sobre el icono



- Copiar el código arrastrando los bloques que correspondan.

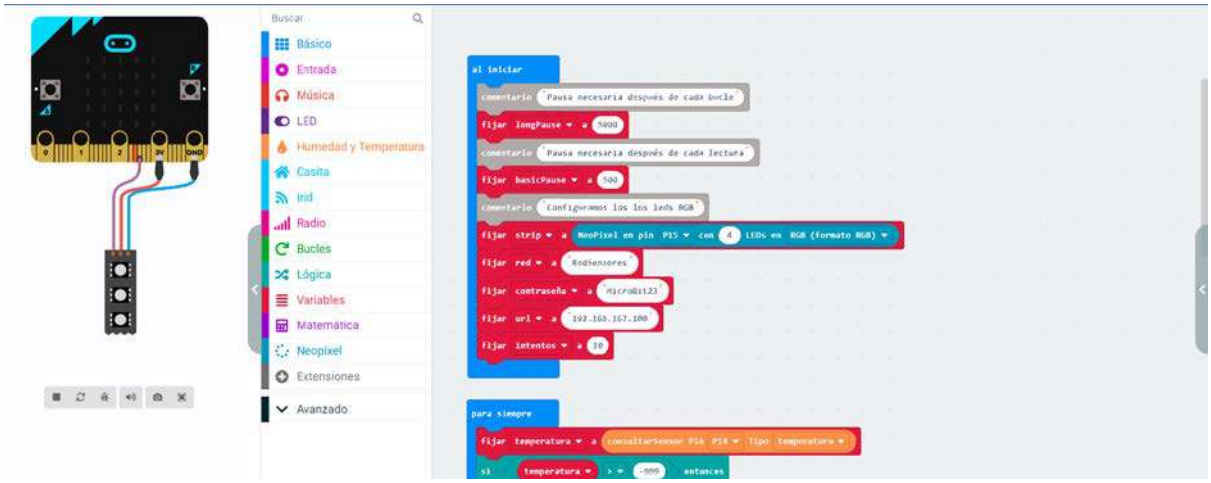


Parte B

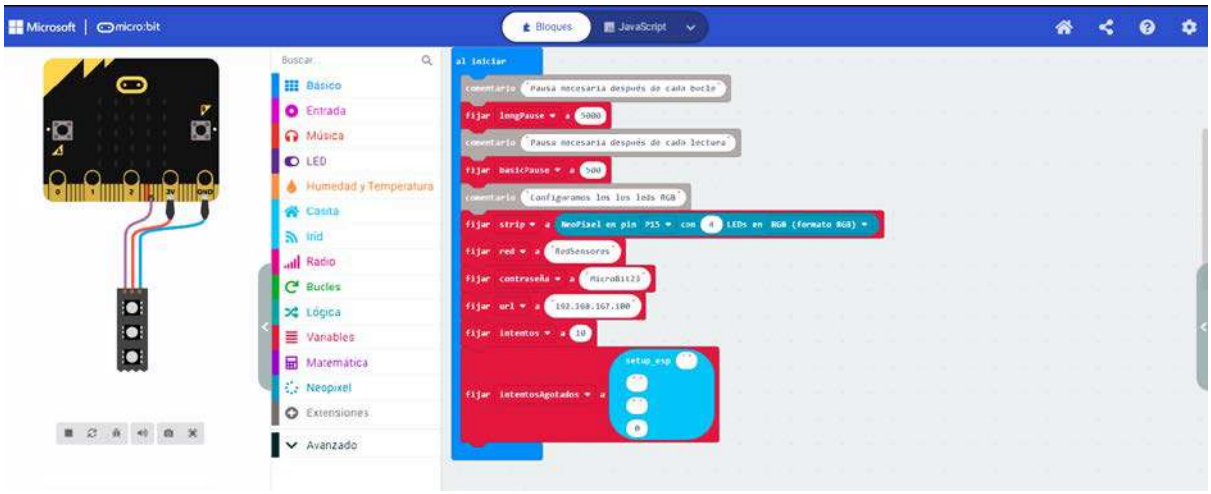
En esta parte vamos a conectar la placa ESP8266 encargada de enviar los datos sensados por la MicroBit hacia el servidor.

1. Primero vamos a configurar 4 variables llamadas: red, contraseña, url e intentos.

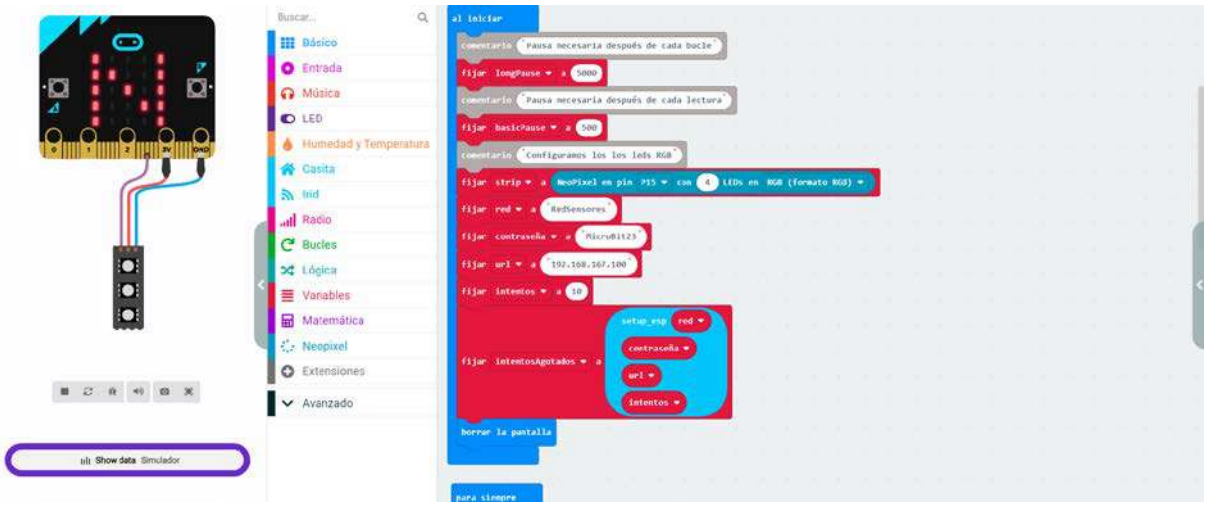
    - **red**: red wifi a la cual nos vamos a conectar
    - **contraseña**: contraseña de la red wifi.
    - **url**: dirección del servidor al que le vamos a enviar los datos.
    - **intentos**: cantidad de intentos de conexión.



2. Vamos a crear otra variable que se llame ***intentosAgotados*** y vamos a fijarla con el bloque **Irid/setup_esp**



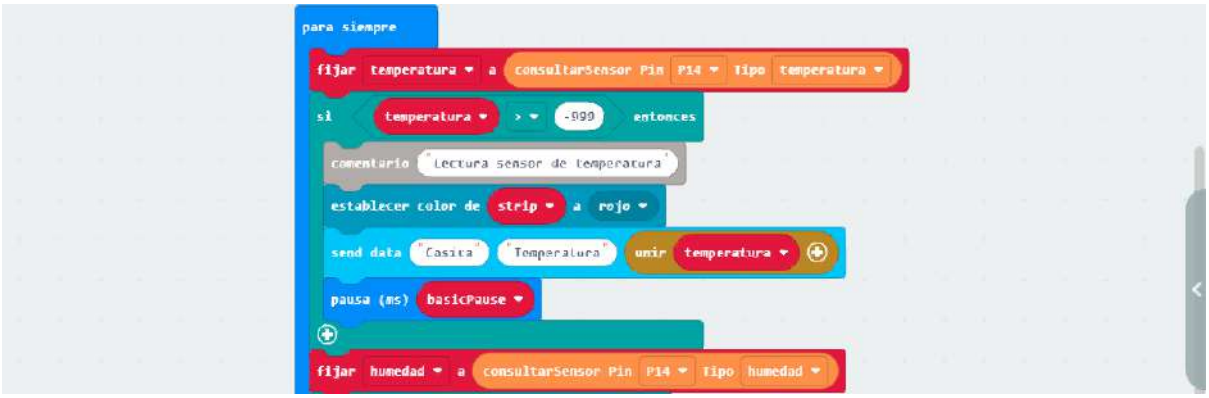
3. Arrastrar las variables red, contraseña, url e intentos, en ese orden, dentro de los campos del bloque **"setup_esp".**
4. Borrar la pantalla al final del bloque "**al iniciar**"



5. Enviar los datos sensados utilizando el bloque "**Irid/send data**".

- El primer campo va a contener de dónde vienen los datos,
- El segundo campo que sensor tomó la medición y
- El tercer campo es el valor sensado.

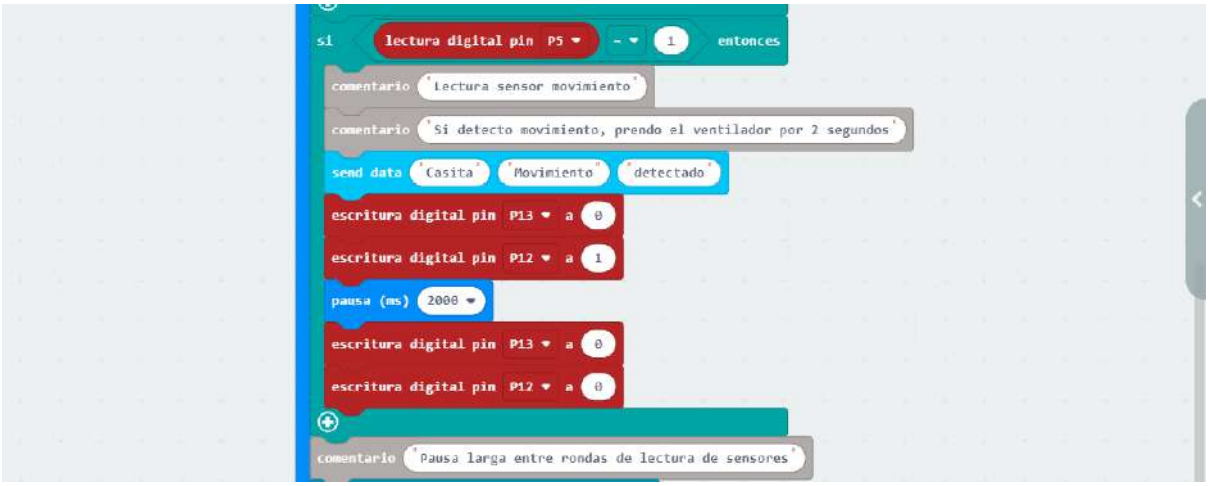
Envío de temperatura



Envío de humedad

Envío de detección de gas

Envío de detección de movimiento

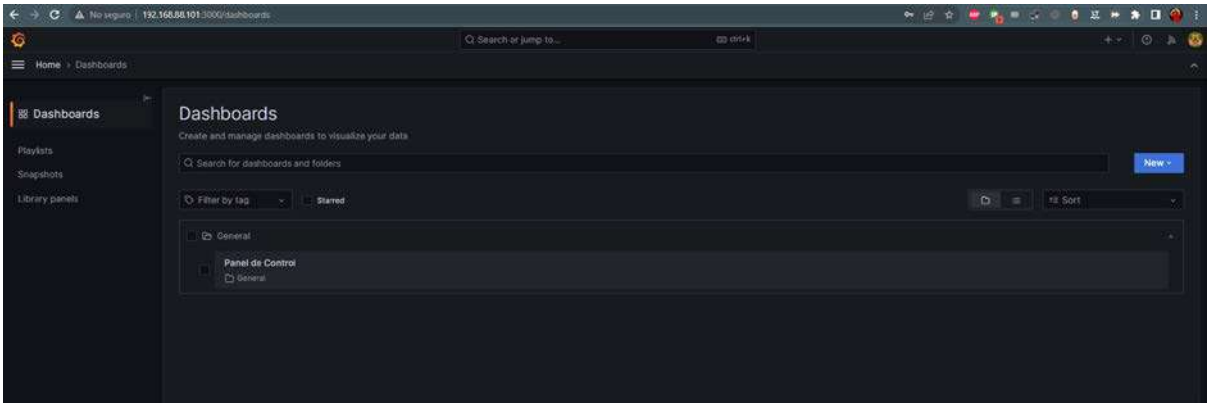


Parte C

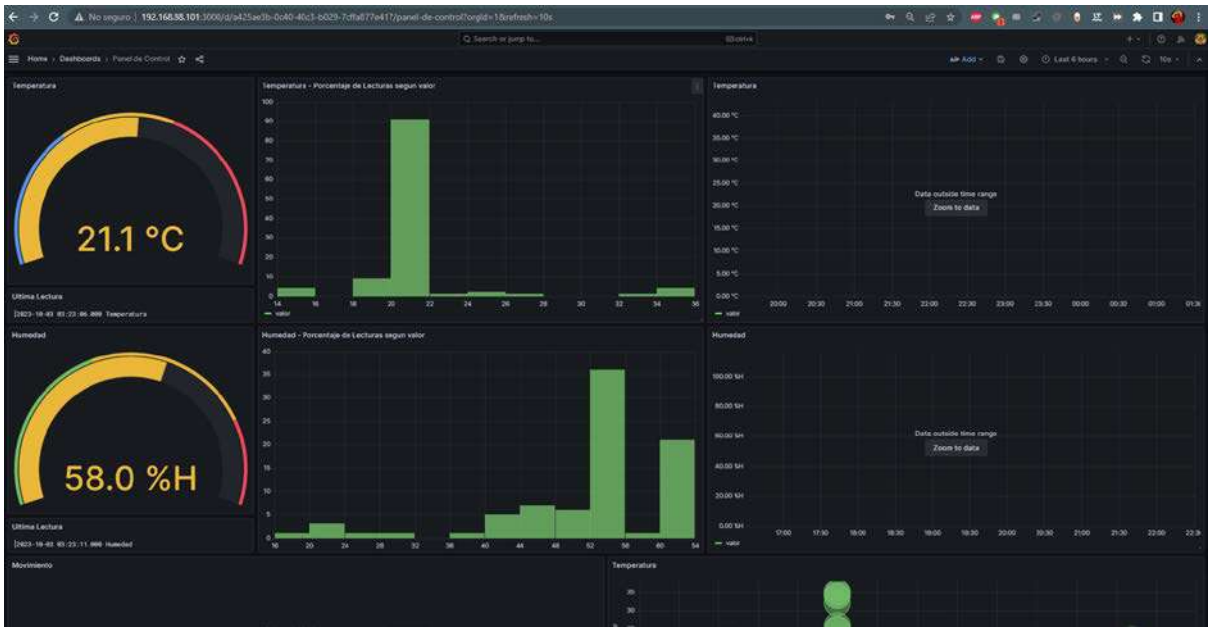
Para la visualización de datos vamos a ingresar la dirección del servidor en un navegador en el puerto 300.

En el caso del ejemplo sería: http://192.168.167.100:3000/

Luego ir a Dashboards/General/Panel de control



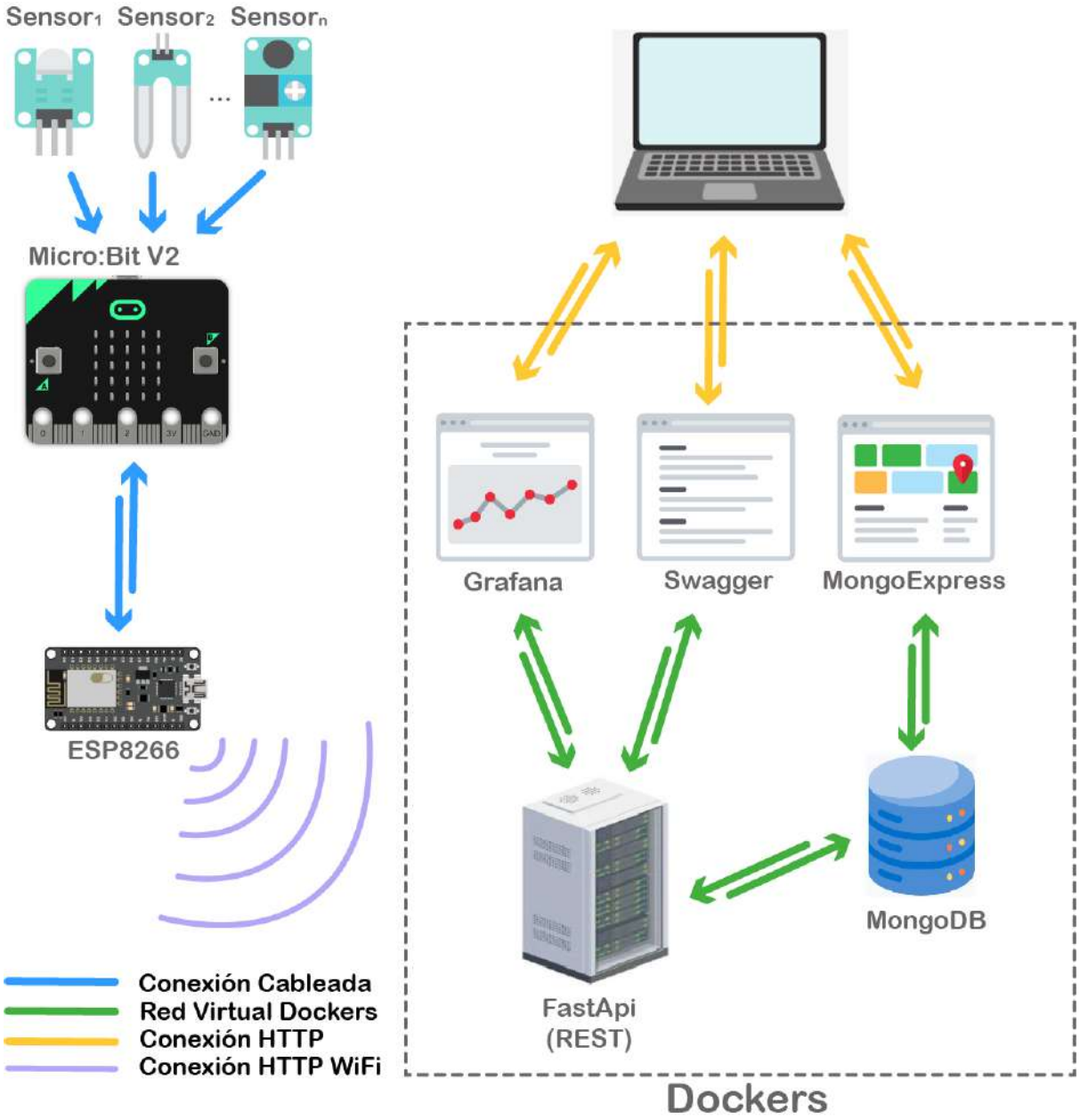
Desde este panel se podrán visualizar los datos en tiempo real



Aspectos Técnicos de la Arquitectura

En la figura de abajo se pueden apreciar los componentes internos de lo que denominamos **Servidor** al principio del documento.

Todos los componentes utilizados son de código abierto y gratuitos. Una vez depositada la tesina objeto de este trabajo, el código quedará liberado para su uso



A continuación, se detallan los componentes visualizados en la figura anterior.

FastApi (REST) es el componente encargado de recibir, analizar y tratar la información enviada por la ESP8266 vía wifi. Explicado de manera sencilla, este componente pone a disposición ciertas direcciones http para la recepción de información, si esa información cumple con los parámetros predefinidos por el componente se guardan en una base de datos. Aparte de la dirección http antes mencionada, también pone a disposición direcciones para consultar la información almacenada. Estas direcciones se verán en más detalle más adelante. Está desarrollado en el lenguaje de programación Python.

MongoDB es el componente encargado de guardar la información recibida por **FastApi (REST)**. Es una base de datos no relacional, por lo cual no necesita que definamos como va a estar

estructurada la información antes de empezar a usarla. Para su utilización solo hizo falta instalarla y configurar las credenciales de acceso, el nombre de la base y la dirección de la misma dentro de **FastApi (REST).**

Grafana es el componente encargado de la visualización de los datos. Soporta varios tipos distintos de orígenes de datos. En este proyecto se utilizan las direcciones http de consulta de **FastApi (REST)** para obtener los datos a mostrar.

Swagger y **MongoExpress** son dos componentes destinados a la utilización durante el desarrollo del proyecto.

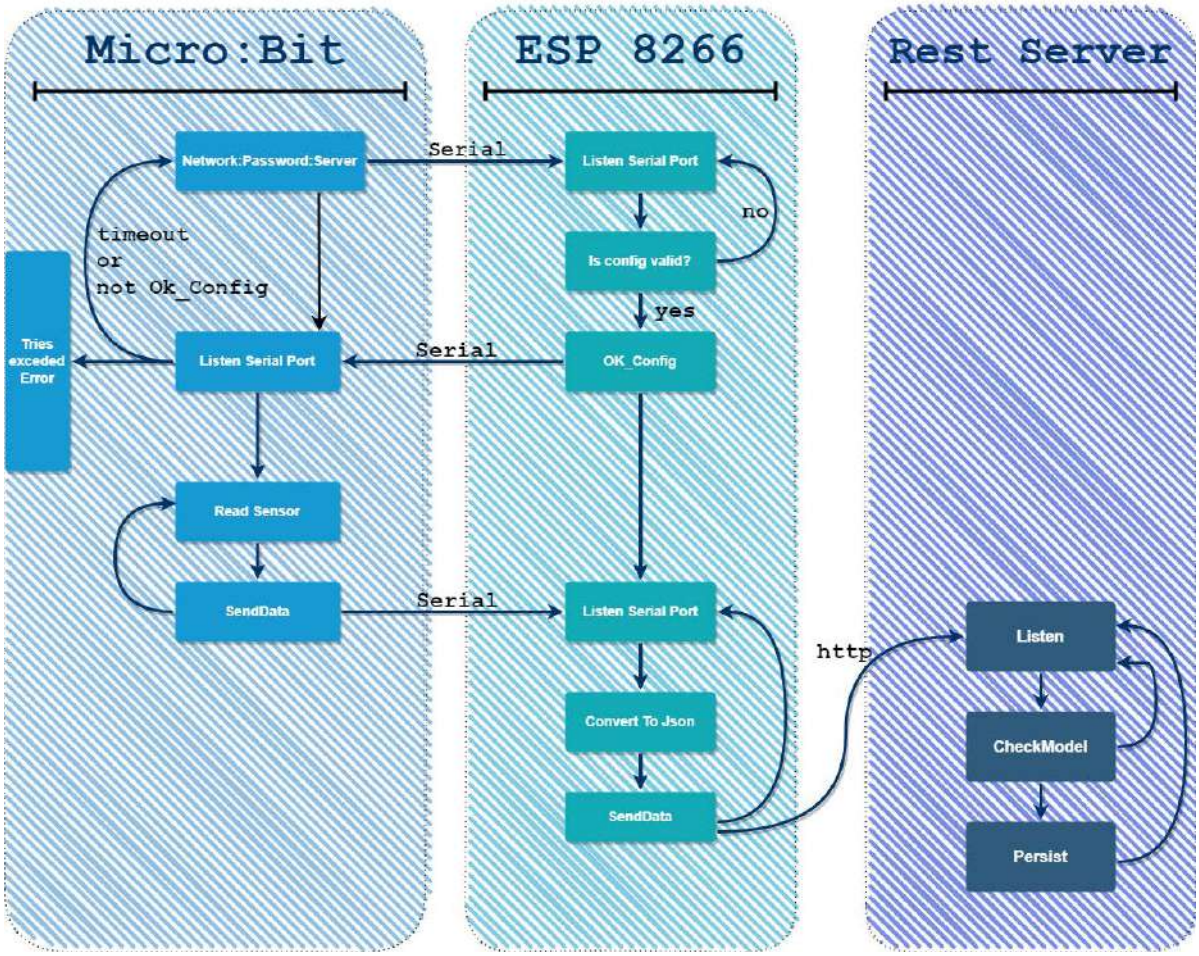
Swagger provee una interfaz web para probar el correcto funcionamiento de las direcciones http de **FastApi (REST).**

MongoExpress provee una interfaz web para ver y manipular los datos almacenados por **MongoDB** en crudo.

Todos los componentes están unificados bajo un entorno de virtualización llamado **Dockers** que permite automatizar las configuraciones, instalación de software necesario para el funcionamiento de cada componente y el funcionamiento en varios sistemas operativos distintos (Windows, Mac, Linux).

Otro aspecto técnico importante es el protocolo de comunicación tanto entre la **MicroBit** y la **ESP8266** como entre la **ESP8266** y el **FastApi (REST)** o Rest Server.

El siguiente gráfico muestra como es el flujo de comunicación entre los componentes mencionados.



En un principio la MicroBit envía los parámetros de configuración de red a la ESP8266. La ESP8266 los recibe, si los parámetros son correctos le envía OK a la MicroBit y empieza a esperar que la MicroBit le envíe datos de los sensores. Si los parámetros no son correctos vuelve a esperar a que la MicroBit le envíe los parámetros de red.

Una vez que la MicroBit le envía los parámetros de red se queda esperando que la ESP8266 le diga si los parámetros son válidos, si los parámetros están bien la MicroBit empieza la tarea de leer los sensores y enviar los datos a la ESP8266, seguirá haciendo esto hasta que se apague. Si los parámetros no son correctos y no agotó las chances de configurar la ESP8266 le enviará nuevamente los parámetros, si agotó las chances mostrará un mensaje de error.

Una vez que se chequearon los parámetros y la MicroBit está en etapa de lectura de sensores y envío de datos, la ESP8266 se queda en un bucle de esperar los datos de la MicroBit, convertirlos a formato json y enviarlos a **FastApi (REST)** o Rest Server en el gráfico.

Por último, **FastApi (REST)** está esperando conexiones desde que empieza a funcionar, cuando recibe una conexión verifica que los datos cumplan con ciertos parámetros y de ser así los guarda en la base **MongoDB**, este proceso se conoce como *persistir los datos*. En caso de no cumplir los parámetros no guarda los datos en la base y vuelve a quedarse esperando nuevas conexiones.

Anexo IV: Artículo Presentado en III SPA -CACIC 2024

IRID: Infraestructura para la recolección inalámbrica de datos provistos por sensores en el marco de robótica educativa en nivel secundario

Alumno: Juan Pablo Lozano Arce[1]

Directores: Alejandra Beatriz Lliteras [1,2] y Andrés Rodriguez [1]

[1] UNLP. Facultad de Informática, Centro LIFIA [2] CICPBA

Contacto {jlozano, alejandra.lliteras, andres.rodriguez}@lifia.info.unlp.edu.ar

Tipo de Trabajo: Tesina de grado en curso

Palabras Claves: MicroBit, ESP, WiFi, Protocolo, Servidor, API Rest, Programación basada en Bloques, Comunicación Inalámbrica, Recolección de Datos, Visualización de Datos

Síntesis

En este trabajo se propone una infraestructura que combina software y hardware para que sea posible, por un lado, recolectar datos mediante sensores conectados a placas MicroBit (V2), y por otro, enviar dichos datos, vía Wifi, a un servidor para su almacenamiento y posterior visualización. El objetivo es que esta infraestructura pueda ser usada de manera sencilla y guiada por docentes y estudiantes de nivel secundario, y que permita trabajar aspectos de pensamiento computacional empleando programación basada en bloques. En este trabajo se propone: a) el diseño e implementación de un servidor que recepciona, valida, almacena y permite visualizar datos, b) un protocolo de comunicación entre placas MicroBit, ESP8266 y un servidor Web, c) una extensión al entorno MakeCode de programación basada en bloques para placas MicroBit d) guías para la preconfiguración de la placa ESP y para desplegar el servidor

Motivación

El desarrollo del Pensamiento Computacional al considerar temas de robótica o IoT en el área de educación, suele trabajarse a partir de kits que ya vienen pre armados y con instrucciones concretas para que los docentes puedan trabajar en el aula usando placas como, por ejemplo, MicroBit[51] o Arduino[52]. Para bajar la complejidad que implican las placas Arduino para estudiantes de secundarios no técnicos, Pech y Novák sugieren el uso de placas MicroBit por sobre las otras ya que, entre otros aspectos, traen componentes integrados.

Al conectar sensores a una placa MicroBit y querer enviar vía Wifi los datos recolectados, surge la necesidad de conectarla a una placa ESP para que sea esta quien realice el envío. En particular, se espera

[51] https://microbit.org/

[52] https://www.arduino.cc/

visualizar mediante gráficos los datos recolectados y que tanto la recolección de los datos, como su envío vía Wifi pueda ser programado de manera intuitiva usando programación basada en bloques. Es objetivo de este trabajo que la infraestructura formada por sensores, placa MicroBit, ESP y servidor Web, pueda ser replicados en diferentes espacios de trabajo, como por ejemplo escuelas, sin necesidad de expertos en tecnología.

Aporte

El trabajo presenta diferentes aportes, a continuación, se detalla cada uno de ellos

Servidor:

Para este trabajo se propone un servidor web. La Fig. 1 muestra en esquema de la solución del trabajo, en esta sección se hace foco en el servidor. A continuación, se describen los diferentes componentes.
FastApi (REST) es el componente encargado de recibir, analizar y tratar la información enviada por la ESP8266 vía wifi. Está desarrollado en el lenguaje de programación Python. *MongoDB* es el componente encargado de guardar la información recibida por FastApi (REST). *Grafana* es el componente encargado de la visualización de los datos. Soporta varios tipos distintos de orígenes de datos. En este proyecto se utilizan las direcciones http de consulta de FastApi (REST) para obtener los datos a mostrar. *Swagger* provee una interfaz web para probar el correcto funcionamiento de las direcciones http de FastApi (REST). *MongoExpress* provee una interfaz web para ver y manipular los datos almacenados por MongoDB.
Todos los componentes están unificados bajo Docker, un entorno de virtualización que permite automatizar las configuraciones, instalación de software necesario para el funcionamiento de cada componente y el funcionamiento en varios sistemas operativos distintos (Windows, Mac, Linux).

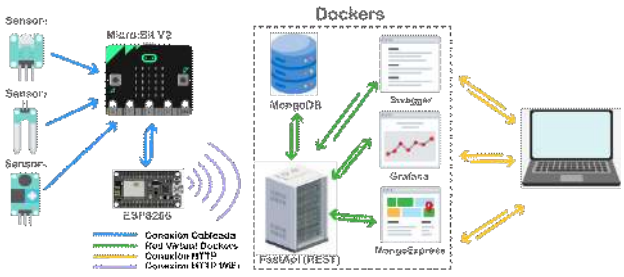


Fig. 1: Detalle del Servidor

Si bien el foco de la Fig.1 es mostrar el esquema del servidor, también puede apreciarse la manera en la que los sensores se conectan a la `placa MicroBit y ésta a la ESP8266 quien envía los datos via Wifi.

Protocolo de Comunicación

En esta sección se muestra el protocolo de comunicación entre la placa MicroBit y la placa ESP8266 y entre la placa ESP8266 y el FastApi (REST) o Rest Server.
La Fig. 2 muestra como es el flujo de comunicación entre los componentes mencionados.

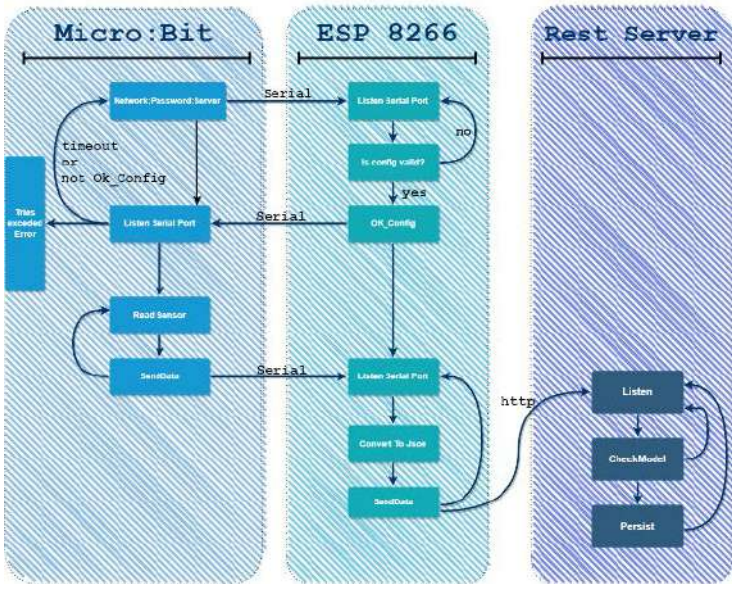


Fig. 2:Protocolo de comunicación

En un principio la MicroBit envía los parámetros de configuración de red a la ESP8266. La ESP8266 los recibe, si los parámetros son correctos le envía OK a la MicroBit y empieza a esperar que la MicroBit le envíe datos de los sensores. Si los parámetros no son correctos vuelve a esperar a que la MicroBit le envíe los parámetros de red. Una vez que la MicroBit le envía los parámetros de red a la ESP, la MicroBit se queda esperando que la ESP8266 le diga si los parámetros son válidos, si los parámetros están bien la MicroBit empieza la tarea de leer los sensores y enviar los datos a la ESP8266, seguirá haciendo esto hasta que se apague. Si los parámetros no son correctos y no agotó las chances de configurar la ESP8266 le enviará nuevamente los parámetros, si agotó las chances mostrará un mensaje de error. Una vez que se chequearon los parámetros y la MicroBit está en etapa de lectura de sensores y envío de datos, la ESP8266 se queda en un bucle de esperar los datos de la MicroBit, convertirlos a formato json y enviarlos a FastApi (REST) o Rest Server (Fig. 2). Por último, FastApi (REST) está esperando conexiones desde que empieza a funcionar, cuando recibe una conexión verifica que los datos cumplan con ciertos parámetros y de ser así los guarda en la base MongoDB. En caso de no cumplir los parámetros no guarda los datos en la base y vuelve a quedarse esperando nuevas conexiones.

Notar que están fuera del alcance de este trabajo los aspectos de seguridad y de validación de arribo correcto de los datos.

Extensión al entorno MakeCode

Con el fin de encapsular la complejidad relacionada a configurar la placa ESP y al envío de datos, se creó una extensión con una familia de bloques para el entorno MakeCode. El bloque "*setup_esp"* permite setear el nombre de la red, la contraseña de acceso, la url del servidor y un número entero que representa la cantidad de intentos a realizar antes de avisar que hay un error. El bloque de "*send*

data" envía los datos avisando desde donde se toma, que tipo de sensor y el dato recolectado por el sensor. Por último, el bloque "*ping*" sirve para validar el estado del servidor. Adicionalmente, se sumó una ayuda contextual que se accede al pasar por sobre cada bloque. Lo anterior puede visualizarse en la Fig.3.

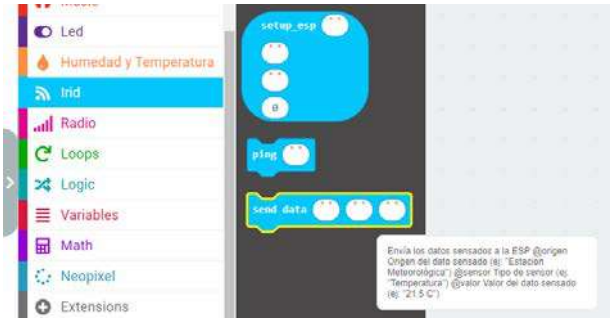


Fig. 3: Familia de bloques de la extensión para MakeCode

La creación de estos tipos de bloques especiales se hace mediante la edición de un archivo JavaScript titulado "custom.ts" dentro del entorno MakeCode (Fig. 4). Este archivo se crea por defecto al momento de la creación de los proyectos y se accede mediante la pestaña de modo JavaScript.

Guías de preconfiguración de la ESP y despliegue del servidor

Se desarrollaron dos *guías*, una relacionada al servidor y otra para preconfigurar la placa ESP. Una primera guía que explica paso a paso como poner en marcha el servidor para la recepción, almacenamiento y visualización de datos (llamada "*IRID Instructivo para el despliegue de ambiente* [53]"). La segunda guía, dada que la forma de comunicar la placa MicroBit con la placa ESP8266 es mediante el puerto serie de estas, es necesario realizar preconfiguraciones en la placa ESP para usarlos, por tal motivo y por única vez es necesario cargarle esa preconfiguración a la placa. Para este fin se creó una guía con el paso a paso (llamada "*IRID Instructivo de programación de la ESP*[54]").

[53] *"IRID Instructivo para el despliegue de ambiente"* está disponible en *https://zenodo.org/records/13234008*

[54] *"IRID Instructivo de programación de la ESP"* está disponible en https://zenodo.org/records/13240796

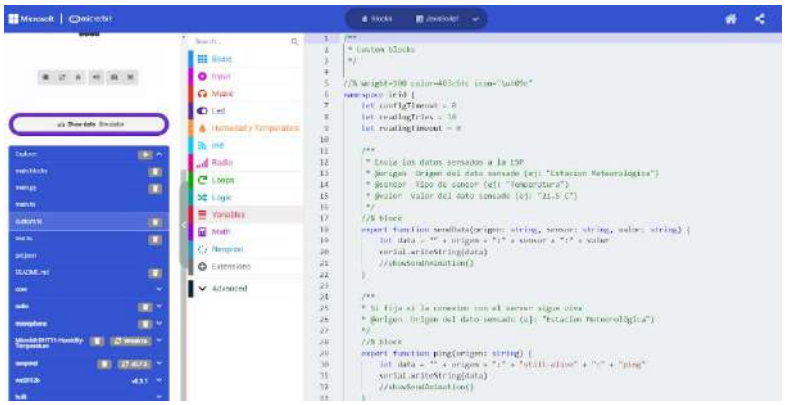


Fig. 4: Archivo custom.ts

Líneas de Investigación Futura

Se proponen como futuras líneas de investigación:

- Analizar otro tipo de protocolos para el envío de datos como alternativa a REST, como por ejemplo MQTT.
- Ampliar el protocolo de comunicación para que exista una validación bidireccional respecto al arribo de los datos y reenviar en caso de falla.
- Ampliar a más cantidad de gráficos la visualización de datos
- Mejorar la usabilidad de las guías destinadas a usuarios finales
- Conformar un kit de solución al estilo “Hágalo usted Mismo”

Bibliografía Básica

- Fernández Quiñonez, S. L. (2022). Implementación de sistemas iot utilizando técnicas de programación visual.
- Pech, J., & Novák, M. (2020). Use Arduino and micro: bit as teaching platform for the education programming and electronics on the stem basis. V International Conference on Information Technologies in Engineering Education (pp. 1-4). IEEE.
- Pradeep, A. (2023). Enabling IoTs with ESP32 for Affordable Education. 5th International Conference on Inventive Research in Computing Applications (pp. 1368-1373). IEEE.
- https://www.alldatasheet.es/datasheet-pdf/pdf/1132995/ESPRESSIF/ESP8266.html
- https://support.microbit.org/support/solutions/articles/19000119052-details-of-micro-bit-v2

Anexo V: Artículo Presentado en XXIII WTIAE - CACIC 2024

Recolección, envío, almacenamiento y visualización de datos considerando programación basada en bloques de placas MicroBit, ESP, sensores y Wifi

Alejandra Beatriz Lliteras[1,2][0000-0002-4148-1299] , Juan Pablo Lozano Arce[1][0009-0001-7428-9455], Andrés Rodriguez[1][0000-0002-7600-6041]

[1] UNLP. Facultad de Informática, Centro LIFIA, Argentina
[2] CICPBA
{alejandra.lliteras, jlozano,andres.rodriguez}@lifia.info.unlp.edu.ar

Abstract. El Pensamiento Computacional es una de las habilidades a desarrollar desde la escuela secundaria y así brindar igualdad de oportunidades a los estudiantes para una participación plena como ciudadanos digitales. En este trabajo se presenta una infraestructura para poner en práctica aspectos de recolección de datos con sensores conectados a una placa Microbit la que conectada a una ESP permite enviarlos vía Wifi a un servidor Web. Se usa programación basada en bloques en MakeCode proponiendo una extensión. Se brinda además un servidor que permite recibir paquetes de datos, validarlos, almacenarlos y dejarlos disponibles para su posterior visualización. Se brindan guías para el despliegue del servidor propuesto y para la preconfiguración de la placa ESP para que puedan ser usados en diferentes contextos. Se presenta un caso de estudio usando la extensión propuesta y el servidor y se realizó una primera validación con estudiantes de cuarto año de secundario.

Keywords: Pensamiento Computacional, Ciudadanía Digital, Robótica Educativa, IoT, Recolección de Datos, Visualización de Datos, MicroBit, ESP, Sensor, WiFI, Comunicación Inalámbrica, Nivel Secundario, Programación Basada en Bloques, MakeCode

1 Introducción

El Pensamiento Computacional ha cobrado relevancia en el ámbito educativo en los últimos años. Si bien el término fue acuñado por Papert [1] hace más de tres décadas, en el año 2006 fue ampliamente difundido por Wing [2] y desde entonces su vigencia permanece. La autora sostiene que ese tipo de pensamiento es necesario para todos los individuos, no solo para quienes se dedican a las ciencias de la computación, sino que también para quienes se desempeñan en otras disciplinas y profesiones. Su objetivo, y el de otros autores en el tema, es agregar e integrar el Pensamiento Computacional como un saber fundamental en el currículum de la educación básica (inicial, primaria y secundaria) y de esta forma avanzar hacia una actualización de contenidos que no solo esté basada en

habilidades informáticas sino en conocimientos de campos disciplinares curriculares en general.

Si bien existen diversas definiciones del Pensamiento Computacional y se ha trabajado en este tema durante muchos años, aun no existe una definición consensuada [3]. En este trabajo se considera que el Pensamiento Computacional es una forma de resolver y analizar problemas de manera que sean ejecutadas por una computadora y que, como mencionan Dening y Tedre, involucra habilidades y conceptos como la abstracción, la descomposición, la generalización, los algoritmos (y su diseño) así como, el diseño, la recolección y el análisis de datos [3].

Para desarrollar el Pensamiento Computacional existen diferentes esfuerzos, en general, asociados a temas relacionados a la programación y a las ciencias de la computación. Por ejemplo, en el año 2013 se creó Code.org[55] con origen en Estados Unidos y de alcance internacional. Por otro lado, en el mismo año, pero en Argentina, se suma la iniciativa Program.ar[56] de la mano de la fundación Sadosky[57] [4]. Si bien estos programas implican avances significativos en cuanto a la enseñanza de la programación, en la actualidad, uno de los desafíos más grandes radica en la manera de promover este tipo de pensamiento a toda la currícula de la escuela secundaria en forma transversal y trascendiendo la enseñanza de la programación por sí sola.

En el año 2018 se sancionaron en Argentina los Núcleos de Aprendizaje Prioritario (NAP) de Educación Digital, Programación y Robótica[58] que plantean una integración del Pensamiento Computacional en forma transversal a las diferentes áreas, evitando circunscribirla a la asignatura de Informática, sin embargo, su difusión y adopción aún hoy es escasa. Adicionalmente, en 2021 Argentina lanzó un plan Federal de Robótica Educativa e Internet de las Cosas en el que participan 14 provincias.

En este trabajo se destaca la importancia de desarrollar el Pensamiento Computacional en los estudiantes secundarios considerando el vertiginoso avance tecnológico [5]. Ante esta realidad es importante trabajar en post de que los estudiantes puedan comprender como funcionan las tecnologías, usarlas adecuadamente y pensar en cómo proponer nuevas alternativas, con una actitud responsable y crítica de modo que puedan comprender el impacto que éstas tienen en la sociedad, lo que en palabras de Ribble no es otra cosa más que formar a los estudiantes como ciudadanos digitales [6].

El desarrollo del Pensamiento Computacional puede ser abordado de diferentes formas, por ejemplo, mediante la visualización de datos [7], la programación [8], la robótica educativa [9], IoT (de la sigla en inglés para *Internet of Things,* previamente nombrado en castellano como Internet de las Cosas) [10] y la Inteligencia Artificial [11]. En este trabajo, se hará foco en el desarrollo de este tipo de pensamiento desde la programación de aspectos de robótica educativa y de IoT para la posterior visualización de datos.

Un primer paso hacia la solución de la incorporación del Pensamiento Computacional de manera transversal en la currícula es proveer a los docentes y alumnos de herramientas accesibles y simples de usar que engloben las diferentes aristas de este tipo de pensamiento y para la programación. En particular, para facilitar los primeros pasos en la

[55] https://code.org/
[56] https://program.ar/
[57] https://fundacionsadosky.org.ar/
[58] https://www.educ.ar/recursos/150123/nap-de-educacion-digital-programacion-y-robotica

programación, se suelen usar lenguajes de programación basados en bloques ya que estos bajan la complejidad que presentan los lenguajes de programación basados en textos y favorecen la adquisición de habilidades del Pensamiento Computacional [12].

Cuando se piensa en el desarrollo del Pensamiento Computacional considerando temas de robótica o IoT en el área de educación, en general se hace en términos de kits que ya vienen pre armados y con instrucciones concretas para que los docentes puedan trabajar en el aula usando placas como, por ejemplo, MicroBit[59] o Arduino[60]. Las placas Arduino si bien están ampliamente difundidas para diversas actividades educativas, pueden presentar desafíos al momento de su uso tanto para docentes como para estudiantes de escuelas no técnicas principalmente, a quienes le puede resultar complejo el cableado de circuitos que se necesitan para trabajar con ellas. Por lo anterior y debido a que ya además traen integrados algunos componentes es que Pech y Novák proponen el uso de placas MicroBit [13]. A raíz de lo anterior es que para este trabajo se considera el uso de placas MicroBit, en particular, en su versión 2 y de aquí en adelante, estas placas serán nombradas simplemente como MicroBit asumiendo la versión. Estas placas tienen la opción de almacenar datos, por ejemplo, provenientes de sensores, o bien de enviarlos usando el Bluetooth que trae incorporado. En cambio, cuando el envío se quiere hacer inalámbricamente vía Wifi es necesario sumar un componente de hardware, como por ejemplo una placa ESP. En particular, en este trabajo se usará una placa ESP8266 para tal fin.

Los datos una vez recolectados por los sensores pueden ser luego cargados y analizados desde diversas plataformas de visualización, como por ejemplo Alfadatizando [14], una plataforma diseñada para realizar actividades educativas de visualización de datos para desarrollar el Pensamiento Computacional o bien pueden ser visualizados desde una plataforma específica diseñada ad hoc.

Como se mencionó anteriormente es importante proveer a los docentes y alumnos de herramientas accesibles y simples de usar que permitan desarrollar las diferentes aristas de este tipo de pensamiento. En este sentido, Fernandez Quiñonez propone diferentes maneras para programar sistemas IOT usando programación basada en bloques, cuando presenta el uso de placas MicroBit al que le conecta sensores para la toma de datos, almacena localmente los datos para su posterior visualización desde una aplicación móvil que muestra cómo desarrollar usando AppInventor[61]. En este caso, el dispositivo móvil desde el cual se accede a los datos posee activa la conexión vía Bluetooth al igual que la placa MicroBit [15]. Por otro lado, Aneesh Pradeep propone el uso de una placa ESP32 con fines educativos para trabajar aspectos de IOT, recolectando y analizando datos provenientes de sensores. Justifica su elección de esta placa por sobre otras, dado su bajo costo. Sugiere a los profesores que usen proyectos ya existentes para enseñar de IOT con la placa elegido y de cómo realizar conexiones a sensores. Ya que, como menciona el autor, esos proyectos suelen tener instrucciones paso a paso y están diseñados para ser fáciles de seguir. El autor no muestra de manera concreta como se programa la placa ni la manera en la que propone visualizar los datos [16].

[59] https://microbit.org/
[60] https://www.arduino.cc/
[61] https://appinventor.mit.edu/

En este trabajo se propone una infraestructura que combina software y hardware para la toma de datos y su posterior visualización. Con foco en que sea de simple armado y configuración para usuarios sin conocimientos avanzados de informática y para que pueda ser usado en diferentes lugares, por diferentes personas y de manera autónoma. Permitiéndole de este modo diseñar actividades para el desarrollo del Pensamiento Computacional mediante programación basada en bloques, visualización de datos, el uso de placas, sensores y comunicación inalámbrica vía Wifi. El trabajo está organizado de la siguiente manera: en la Sección 2 se presenta el escenario de trabajo, en la Sección 3 un posible esquema de solución, en la Sección 4 se presenta un caso de estudio. En la Sección 5 se presenta una discusión sobre el caso de estudio y finalmente en la Sección 6 conclusiones y trabajos futuros.

2 Escenario de trabajo

En esta sección en primer lugar se esquematiza la problemática a abordar. Como se mencionó previamente, el objetivo de este trabajo es facilitar una infraestructura de trabajo que permita desarrollar habilidades de Pensamiento Computacional abordando aspectos de programación basada en bloques, visualización de datos, manejo de placas, sensores y comunicación inalámbrica. Para poder realizar esto se tomó como primer requerimiento que el envío de datos recolectados se realice vía Wifi a un servidor para tal fin, desde el cual es posible conectarse desde un cliente Web para visualizar los datos obtenidos. La Fig. 1 muestra un esquemático del escenario de trabajo, en donde se consideran diferentes sensores conectados a la placa MicroBit usando cable, una placa EP8266 conectada por cable a la placa MicroBit, la cual es responsable de trasmitir datos vía Wifi a un Servidor que los almacena. Luego es posible acceder desde cualquier cliente web al servidor para visualizar gráficamente los datos.

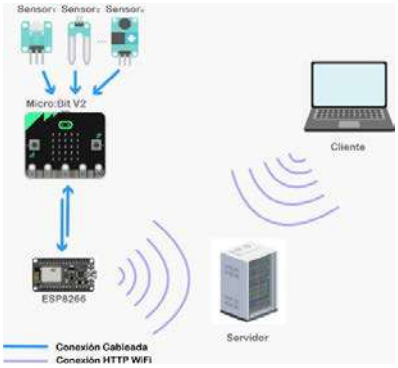


Fig. 5: Esquema del escenario de trabajo

3 Propuesta de solución

Dado que se decidió utilizar placas MicroBit, se tomó como entorno de programación a MakeCode que permite programación usando bloques, Phyton y JavaScript. El entorno

provee familias de bloques para la programación y es posible ampliarlo mediante extensiones. En el caso del uso de Wifi como mecanismo de comunicación inalámbrico, a la fecha de iniciar con esta propuesta, no había una extensión disponible y por tal motivo se decidió proponer una nueva para usar la placa ESP.
Por otro lado, al querer enviar los datos recolectados a un servidor para su posterior visualización, una opción era dejar un servidor dedicado, sin embargo, es objetivo de este trabajo que se use en diferentes lugares, por diferentes personas y de manera autónoma, por esto es por lo que se diseñó e implementó un servidor y visualizador que puede ser desplegado donde el usuario desee siguiendo una serie de pasos detallados.
Se propone en este trabajo a) una extensión para el entorno MakeCode sumando una nueva familia de bloques para encapsular la complejidad técnica de configurar la placa ESP b) un servidor para la recepción, almacenamiento y visualización de datos c) guías para el despliegue del servidor propuesto y para la preconfiguración de la placa ESP
Respecto a la *extensión para el entorno MakeCode*, se logró simplificar la propuesta con los bloques que visualizan en la Fig. 2. Se creó una nueva familia de bloques llamada IRID donde para cada nuevo bloque al posar el cursor sobre el mismo es posible acceder a información de lo que hace y se muestra un ejemplo de valor para sus parámetros.
El bloque "*setup_esp"* permite configurar el nombre de la red a utilizar, la contraseña, la url del servidor y un número que indica la cantidad de veces que se espera la placa intente conectarse al servidor antes de dar un error y avisar que no es posible. El bloque "*send data"* se usa para el envío de datos al servidor. El primer parámetro indica desde donde se reciben los datos, el segundo con que tipo de sensor se toman y por último el dato del valor sensado. Finalmente, se deja disponible un bloque para "*ping*" que puede ser usado para analizar si el servidor está respondiendo.

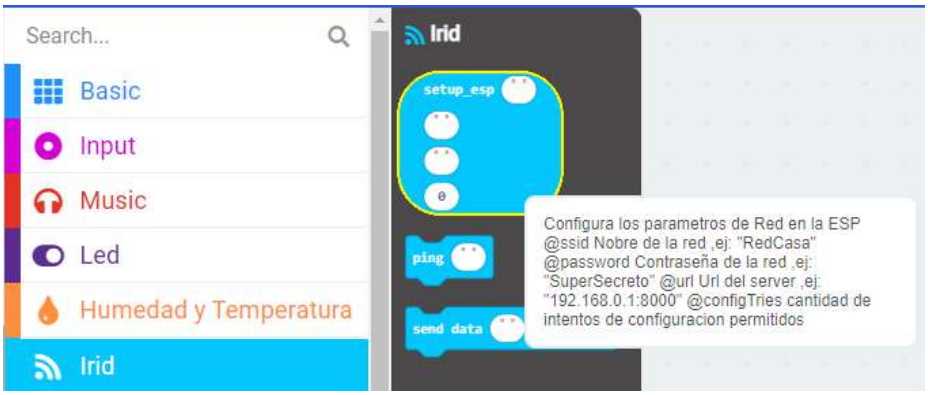


Fig. 6: Familia de bloques para trabajar con la ESP

Notar que en este trabajo no se hace foco en aspectos de seguridad del servidor, por ello quedarán datos sensibles expuestos.
En cuanto al *Servidor para la recepción, almacenamiento y visualización de datos*, el mismo fue desarrollado con herramientas de código abierto. Cuenta con componente encargado de recibir, analizar y tratar la información enviada por la placa ESP8266 vía

Wifi (desarrollado en Phyton[62]). El servidor al recibir datos, si éstos cumplen con el formato esperado, los almacena (en MongoDB[63]). Los datos quedan disponibles para ser visualizados mediante un componente implementado en el servidor para tal fin (usando Grafana[64]).
Por último, se desarrollaron dos *guías*, una relacionada al servidor y otra para preconfigurar la placa ESP. Una primera guía que explica paso a paso como poner en marcha el servidor[65] para la recepción, almacenamiento y visualización de datos. La segunda guía[66], dada que la forma de comunicar la placa MicroBit con la placa ESP8266 es mediante el puerto serie de estas, es necesario realizar preconfiguraciones en la placa ESP para usarlos, por tal motivo y por única vez es necesario cargarle esa preconfiguración a la placa.

4 Caso de Estudio

El diseño del caso de estudio se realizó para estudiantes de cuarto año de una escuela secundaria técnica, con orientación en programación de la ciudad de La Plata, Buenos Aires, Argentina. Se decidió organizar el material siguiendo el esquema de usar-modificar-crear para apoyar y profundizar la adquisición de Pensamiento Computacional [17]. Dada la disponibilidad ofrecida desde la escuela, se diseñaron tres encuentros de dos horas treinta minutos cada uno. Para un uso más motivador de la actividad, se cuenta con una maqueta en forma de casa, en la que se encuentran conectados diversos sensores (ej. Temperatura, humedad y gas) y actuadores (ej. luces led, motor que permite abrir puertas y un motor con un ventilador). A esta maqueta la llamamos “Casita”. La Fig. 3 muestra la maqueta armada.

Fig. 7:Maqueta de la “Casita”

En el *primer encuentro*, se presenta el entorno de programación MakeCode se introducen conceptos básicos de la programación en bloque usando la plataforma MakeCode para placas MicroBit. En este encuentro, se espera que los estudiantes adquieran habilidades básicas.
El *segundo encuentro*, suma el uso de sensores externos. Se presenta el esquema de la infraestructura de trabajo y se abordan aspectos de comunicación de datos usando

[62] https://www.python.org/
[63] https://www.mongodb.com/
[64] https://grafana.com/
[65] Disponible en https://zenodo.org/records/13234008
[66] Disponible en https://zenodo.org/records/13240796

ESP8266, se envían datos a un servidor web usando la programación en bloques y finalmente se visualizan los datos generados por sensores.
El *tercer encuentro*, retoma la actividad del encuentro anterior y se realiza una charla de cierre con los estudiantes a modo de relevamiento.
Para simplificar la programación durante los encuentros, se creó una nueva familia de bloques llamado "*Casita*" en el que se sumaron los bloques a usar de la familia "*IRID*" y se creó un bloque especial para que puedan sumarse comentarios. A esta nueva familia de bloques se le configuró la ayuda que permite MakeCode para que, al usar la maqueta, los estudiantes pudieran consultar en que Pin estaba conectado cada sensor o actuador. Lo que haría que funcionara como un kit. La nueva familia de bloques y la ayuda desplegada se visualizan en la Fig.4.
Los ejemplos y ejercicios se dejaron implementados y disponibles en una carpeta, a la que, en caso de ser requerido, se les daba acceso a los estudiantes.

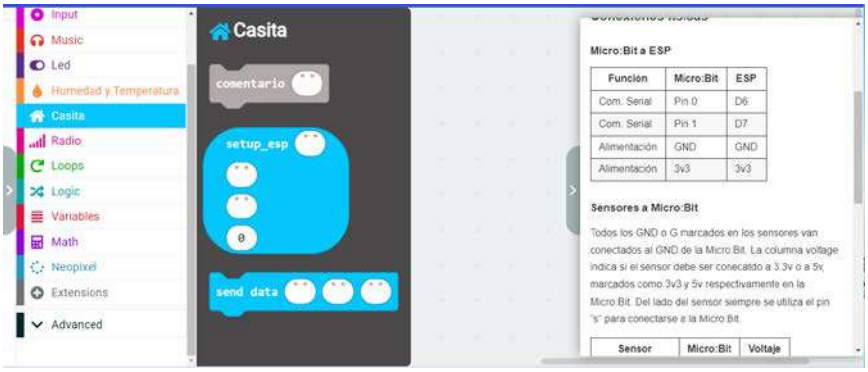


Fig. 8: Familia de bloques para trabajar con la "Casita"

4.1 Propuesta en acción

Para este caso de estudio, considerando la cantidad de encuentros disponibles, si bien se entregaron las guías para preconfiguración de la ESP y para el despliegue de un servidor propio, se decidió llevar la placa ESP preconfigurada, y un servidor ya levantado para que pudiesen directamente acceder al mismo. Los tres encuentros se realizaron durante el mes de noviembre de 2023.
La cantidad de asistentes varió en cada encuentro, en el primer encuentro se presentaron 12 estudiantes y cuatro docentes, en el segundo 15 estudiantes y dos docentes y en el tercero 14 estudiantes tres docentes. Se sumó además de la docente responsable del aula, una docente de apoyo técnico, un docente que imparte tecnología informática en sexto año del secundario y un docente de otra comisión de cuarto año.
Varios de los estudiantes del curso tenían experiencia en desarrollo de placas Arduino y con lenguajes de programación textual. No todos los estudiantes poseían el mismo nivel de expertise en programación. Cuatro estudiantes, aun no programaban.

El aula disponía de mesas con pcs de escritorio y además se contaba con notebooks que la escuela recibió por el programa Conectar Igualdad[67] y luego con el plan Juana Manso[68]. Lo anterior permitía que cada estudiante dispusiera de un equipo para trabajar, sin embargo, al momento de desarrollar las prácticas, se permitió el trabajo en grupo, con agrupación natural por parte de los estudiantes.
A continuación, la Fig. 5. muestra la manera en la que se toman los datos del sensor de temperatura y se le envían a la ESP para que se encargue de transmitirlos al servidor usando el bloque "*send data*".

Fig. 9: Ejemplo de envío de datos de temperatura

Por último la Fig. 6 muestra la visualización de los datos cuando se accede desde un cliente al servidor.

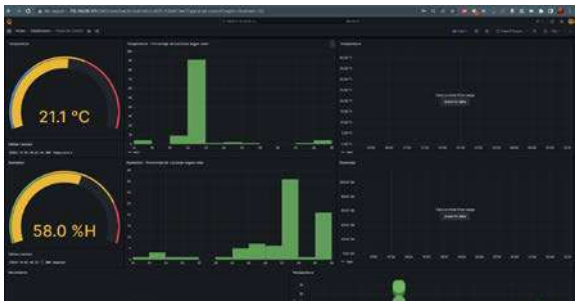


Fig. 10: Visualización de los datos almacenados en el servidor

5 Discusión

En esta primera validación de conceptos, el público destinatario estuvo conformado por estudiantes y si bien había docentes participando, éstos no guiaron los encuentros. El curso presentó algunas particularidades tales como la gran disparidad de conocimientos previos entre los estudiantes del mismo curso, así como la actitud frente a las actividades propuestas. En el segundo encuentro, debido a la falta de tiempo, no fue posible trabajar con la visualización de datos, motivo por el cual se retomó en el tercer encuentro. No hubo tiempo suficiente para cumplir con todo el ciclo de usar-modificar-crear propuesto, solo se pudo usar y modificar.

[67] https://conectarigualdad.edu.ar/inicio
[68]https://www.educ.ar/noticias/200546/educacioacuten-presenta-la-nueva-versioacuten-de-huayra-50nbsp

De todos los estudiantes presentes, se mantuvo constante el trabajo solamente con siete de ellos, quienes realizaron las actividades propuestas con interés.
El relevamiento realizado durante el último encuentro con los siete estudiantes que trabajaron en la propuesta mostró su entusiasmo para este tipo de iniciativas quienes destacaron la facilidad de realizar la actividad con programación basada en bloques para la parte de conexión al servidor y el envío de datos. Esta apreciación se debe a que los estudiantes previamente habían programado con la IDE de Arduino durante un taller de robótica educativa.

6 Conclusiones y Trabajos Futuros

En este trabajo se presenta, por un lado, una extensión para el entorno MakeCode para trabajar el envío de datos obtenidos desde sensores de manera inalámbrica (vía Wifi) a un servidor. Se usa una placa MicroBit, una placa ESP8266 y sensores, y se presenta un servidor que está diseñado para que los datos recolectados puedan ser almacenados y visualizados mediante cierto tipo de gráficos. Se diseñaron guías para que sea posible desplegar un servidor propio y para preconfigurar la placa ESP (por única vez). Se diseñó un caso de estudio y se puso en práctica con estudiantes de cuarto año de una escuela secundaria técnica, habiendo, por un lado, obtenido resultados favorables y por otro, recolectado evidencia para mejorar la propuesta.
Si bien se realizaron los tres encuentros planificados, en base a la experiencia resulta recomendable para futuras puestas en práctica, poder realizar una entrevista previa para conocer a los participantes y así refinar el detalle y la cantidad de encuentros, el que, ante la evidencia, debería haber sido al menos uno más para llegar a la etapa en la que los estudiantes crean por sí mismos.
Como trabajo futuro, se espera en el corto plazo sumar otro tipo de gráficos para la visualización de datos, en el mediano plazo, dejar disponibles los datos recolectados para que puedan ser visualizados desde otras plataformas, refinar las guías de trabajo, y las actividades de cada encuentro, sumando, al menos uno de carácter introductorio y opcional que permita reforzar aspectos generales de la programación basada en bloques para el caso de audiencias sin conocimientos previos. Luego de esto se espera realizar nuevas pruebas con estudiantes y docentes de nivel secundario. Se espera también sumar actividades que involucren otras maquetas además de la "Casita". A largo plazo y una vez alcanzado un grado de madures aceptable de las guías de trabajo y de las actividades de los encuentros, se espera proponer este trabajo como una actividad de "Hágalo usted mismo" (conocido por la sigla DIY en inglés de *Do it Yourself*), donde el equipo docente pueda configurar la infraestructura con el andamiaje adecuado provisto por las guías y llevar adelante por sí mismo los encuentros con sus estudiantes.
Los autores consideran que la propuesta conforma una posibilidad, no solo para trabajar aspectos de Pensamiento Computacional y visualización de datos con los estudiantes, sino que, además, una manera de contribuir en su formación como ciudadanos digitales.

Agradecimientos. El equipo de trabajo agradece a Jeziel Paladino por ponernos en contacto con la Profesora Sara Neiret de la Escuela de Educación Técnica N°5, General Manuel N. Savio, de la ciudad de La Plata (Buenos Aires, Argentina) a quien también agradecemos por ser quien, de manera muy generosa

y comprometida, nos brindó un espacio en una de sus materias a cargo. Sumamos a este agradecimiento a sus estudiantes de 4to año (año 2023).

Referencias

1. Papert, S. (1980). " Mindstorms" Children. Computers and powerful ideas.
2. Wing, J. (2006). Computational thinking. Communications of ACM, 49(3), 33-35.
3. Denning, P. J., & Tedre, M. (2021). Computational thinking: A disciplinary perspective. Informatics in Education, 20(3), 361.
4. Pereiro, E., Montaldo, M., Koleszar, V., & Urruticoechea, A. (2022). Computational thinking, artificial intelligence and education in Latin America.
5. Lliteras, A. B. (2024). Alfadatizando como parte de la Ciudadanía Digital. In XXVI Workshop de Investigadores en Ciencias de la Computación (Puerto Madryn, 18 y 19 de abril de 2024). En Prensa.
6. Ribble, M. (2008). Passport to digital citizenship. Learning & leading with technology, 36(4), 14-17.
7. Özkök, G. A. (2021). Fostering Computational Thinking Through Data Visualization and Design on Secondary School Students. J. Univers. Comput. Sci., 27(3), 285-302.
8. Ezeamuzie, N. O., & Leung, J. S. (2022). Computational thinking through an empirical lens: A systematic review of literature. *Journal of Educational Computing Research*, *60*(2), 481-511.
9. Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational technology research and development*, *68*(1), 463-484.
10. Udvaros, J., Forman, N., & Avornicului, M. (2023). Developing computational thinking with microcontrollers in Education 4.0.
11. Yim, I. H. Y., & Su, J. (2024). Artificial intelligence (AI) learning tools in K-12 education: A scoping review. Journal of Computers in Education, 1-39.
12. Sun, D., Looi, C. K., Li, Y., Zhu, C., Zhu, C., & Cheng, M. (2024). Block-based versus text-based programming: a comparison of learners' programming behaviors, computational thinking skills and attitudes toward programming. Educational technology research and development, 72(2), 1067-1089.
13. Pech, J., & Novák, M. (2020). Use Arduino and micro: bit as teaching platform for the education programming and electronics on the stem basis. V International Conference on Information Technologies in Engineering Education (pp. 1-4). IEEE.
14. AlfaDatizando: a Data Visualization Platform to work Computational Thinking in Digital Humanities. Lliteras Alejandra Beatriz, Artopoulos Alejandro, Fernandez Alejandro, Huarte Jimena. LACLO, octubre 2022. In *2022 XVII Latin American Conference on Learning Technologies (LACLO)* (pp. 1-6). IEEE.
15. Fernández Quiñonez, S. L. (2022). Implementación de sistemas iot utilizando técnicas de programación visual.
16. Pradeep, A. (2023). Enabling IoTs with ESP32 for Affordable Education. 5th International Conference on Inventive Research in Computing Applications (pp. 1368-1373). IEEE.
17. Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... & Werner, L. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32-37.