UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

IMPLEMENTACIÓN DE SISTEMAS IOT UTILIZANDO TÉCNICAS DE PROGRAMACION VISUAL

FERNANDEZ QUIÑONEZ SUSY LISSETH
INGENIERA DE SISTEMAS

MACHALA
2022

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

IMPLEMENTACIÓN DE SISTEMAS IOT UTILIZANDO TÉCNICAS DE PROGRAMACION VISUAL

FERNANDEZ QUIÑONEZ SUSY LISSETH
INGENIERA DE SISTEMAS

MACHALA
2022

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

TRABAJO TITULACIÓN
PROPUESTAS TECNOLÓGICAS

IMPLEMENTACIÓN DE SISTEMAS IOT UTILIZANDO TÉCNICAS DE PROGRAMACION VISUAL

FERNANDEZ QUIÑONEZ SUSY LISSETH
INGENIERA DE SISTEMAS

HERNANDEZ ROJAS DIXYS LEONARDO

MACHALA, 20 DE SEPTIEMBRE DE 2022

MACHALA
2022

Tesis Susy Fer2

INFORME DE ORIGINALIDAD

5%	4%	1%	1%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	asignaturas.diatel.upm.es Fuente de Internet	1%
2	repositorio.utmachala.edu.ec Fuente de Internet	1%
3	Submitted to Consorcio CIXUG Trabajo del estudiante	<1%
4	Submitted to Universidad Técnica de Machala Trabajo del estudiante	<1%
5	idoc.pub Fuente de Internet	<1%
6	hdl.handle.net Fuente de Internet	<1%
7	www.studocu.com Fuente de Internet	<1%
8	repositorio.ucv.edu.pe Fuente de Internet	<1%
9	gacetasanitaria.org Fuente de Internet	<1%

CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

La que suscribe, FERNANDEZ QUIÑONEZ SUSY LISSETH, en calidad de autora del siguiente trabajo escrito titulado IMPLEMENTACIÓN DE SISTEMAS IOT UTILIZANDO TÉCNICAS DE PROGRAMACION VISUAL, otorga a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tiene potestad para otorgar los derechos contenidos en esta licencia.

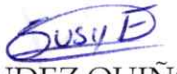
La autora declara que el contenido que se publicará es de carácter académico y se enmarca en las dispociones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

La autora como garante de la autoría de la obra y en relación a la misma, declara que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asume la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.

Machala, 20 de septiembre de 2022



FERNANDEZ QUIÑONEZ SUSY LISSETH
0706718905

DEDICATORIA

Este trabajo está dedicado a mis padres, Ruth Amada Quiñonez Álvarez y Genaro Sixto Fernández Domínguez por darme la oportunidad de seguir mis estudios, por su motivación para alcanzar mis metas y por todos los sacrificios que hicieron por sacar adelante a toda la familia.

Srta. Susy Lisseth Fernández Quiñonez

AGRADECIMIENTO

Agradezco a Jehová por darme salud y sabiduría para tomar decisiones, a mi prometido Israel Rendón por animarme a seguir adelante en mis estudios y a mis profesores por el conocimiento que me impartieron a lo largo de la carrera los cuales me han permitido formarme profesionalmente.

Srta. Susy Lisseth Fernández Quiñonez

RESUMEN

El internet de las cosas sigue en crecimiento con el pasar de los años, cada vez encontramos más dispositivos adaptados a IoT en todos las áreas . En términos generales IoT es una red de dispositivos interconectados que transmiten información para ser procesada y realizar una función específica. Existen muchas aplicaciones para programar sistemas IoT, todas ellas requieren tener gran conocimiento en los fundamentos de la programación, es por ello que se han desarrollado nuevas aplicaciones para que los desarrolladores novatos puedan sumergirse en el mundo de los sistemas IoT. La programación visual contribuye en gran medida a que los estudiantes y programadores que desean realizar sistemas IoT puedan desarrollarlos de manera sencilla, de tal manera que los incentive a continuar aprendiendo acerca de esta nueva tecnología. La implementación de sistemas IoT con ESP32 y MICROBIT utilizando técnicas de programación visual, ayuda a comprender el funcionamiento de las aplicaciones basadas en la programación en bloques. Se utilizaron las herramientas de programación ArduinoBlocks, MakeCode, Appinventor que permiten la creación de sistemas IoT ya sea mediante comunicación WIFI utilizando protocolos de comunicación como MQTT o controlar los dispositivos por bluetooth. A través de prácticas divididas en 4 grupos: Grupo 1 se realizan prácticas básicas como encendido y apagado de leds, relé, lectura de datos con sensores DTH22, LDR, sensor de ultrasonido y sensor IR conectados a la placa ESP32. Grupo 2 prácticas basadas en la comunicación wifi utilizando el protocolo de comunicación MQTT que permitan controlar los componentes de las prácticas del grupo 1 con la placa ESP32, debido a que la placa MICROBIT no posee un módulo WiFi. Grupo 3 comunicación a través del módulo bluetooth de la placa ESP32 y MICROBIT. Grupo 4 creación de aplicaciones móviles para el control de los componentes del grupo 1 para la placa ESP32 y MICROBIT utilizando la herramienta Appinventor que se basa en la programación visual. La evaluación del funcionamiento de las herramientas se realizó tomando en cuenta tres de las características definidas por la norma ISO 9126. La usabilidad se la obtuvo realizando una encuesta a estudiantes y profesionales en donde da como resultado que el lenguaje de programación visual es fácil de comprender, ya que la mayoría de los encuestados lograron entender a simple vista cuál era el resultado de la ejecución de los códigos en bloques por lo que les gustaría trabajar con estas

herramientas. Las herramientas utilizadas en las prácticas resultan eficientes ya que al estar basadas en la web no consumen recursos a diferencia de utilizar el IDE de Arduino para programar la placa ESP32. En cuanto a la portabilidad estas herramientas no necesitan instalación por lo que solo hace falta tener acceso a internet y un navegador web para utilizar estas herramientas. Por lo tanto se determina que las herramientas ArduinoBlocks, MakeCode, Appinventor son óptimas para la creación de sistemas IoT ya que gracias al entorno de programación con bloques hacen que sea sencillo programar aplicaciones que pueden ser utilizadas en distintas áreas y sería un gran apoyo para los estudiantes que desean sumergirse en el Internet de las Cosas.

PALABRAS CLAVES: Internet de las Cosas, Programación Visual, ESP32, MICROBIT, MQTT, ArduinoBlocks, Appinventor, CodeSkool.

ABSTRACT

The internet of things continues to grow over the years, we find more and more devices adapted to IoT in all areas. In general terms, IoT is a network of interconnected devices that transmit information to be processed and perform a specific function. There are many applications to program IoT systems, all of them require great knowledge in the fundamentals of programming, which is why new applications have been developed so that novice developers can immerse themselves in the world of IoT systems. Visual programming greatly contributes to the fact that students and programmers who want to make IoT systems can develop them in a simple way, in such a way that it encourages them to continue learning about this new technology. The implementation of IoT systems with ESP32 and MICROBIT using visual programming techniques helps to understand the operation of applications based on block programming. The ArduinoBlocks, MakeCode, Appinventor programming tools were used that allow the creation of IoT systems either through WIFI communication using communication protocols such as MQTT or controlling devices via bluetooth. Through practices divided into 4 groups: Group 1 basic practices are carried out such as turning on and off LEDs, relay, data reading with DTH22 sensors, LDR, ultrasound sensor and IR sensor connected to the ESP32 board. Group 2 practices based on Wi-Fi communication using the MQTT communication protocol that allow controlling the components of group 1 practices with the ESP32 board, since the MICROBIT board does not have a WiFi module. Group 3 communication through the bluetooth module of the ESP32 and MICROBIT board. Group 4 creation of mobile applications to control the components of group 1 for the ESP32 and MICROBIT board using the Appinventor tool that is based on visual programming. The evaluation of the operation of the tools was carried out taking into account three of the characteristics defined by the ISO 9126 standard. The usability was obtained by conducting a survey of students and professionals, which resulted in the visual programming language being easy to understand. , since most of the respondents managed to understand at a glance what the result of the execution of the codes in blocks was, so they would like to work with these tools. The tools used in the practices are efficient since, being web-based, they do not consume resources, unlike using the Arduino IDE to program the ESP32 board. In

terms of portability, these tools do not need installation, so you only need to have access to the Internet and a web browser to use these tools. Therefore, it is determined that the ArduinoBlocks, MakeCode, Appinventor tools are optimal for the creation of IoT systems since, thanks to the programming environment with blocks, they make it easy to program applications that can be used in different areas and would be a great support for users. students who want to immerse themselves in the Internet of Things.

KEY WORDS: Internet of Things, Visual Programming, ESP32, MICROBIT, MQTT, ArduinoBlocks, Appinventor.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

Internet de las cosas o IoT es un término muy utilizado actualmente, el cual no es más que conectar un objeto inteligente de uso cotidiano (cafetera, lavadora, microondas, un automóvil, entre otros) a internet. Estos objetos a su vez cuentan con circuitos integrados que permiten el intercambio de datos a través de la red. [1] Se está adaptando la tecnología de tal manera que transformará la forma de vida de las personas.

IoT son objetos cotidianos que con ayuda de una mini computadora se conectan a la red, por ejemplo uno de las áreas donde esta tecnología está inmersa es en la agricultura ya que permite mejorar la productividad de los cultivos, ahorrando costos, mano de obra y recursos. [2]

Trabajar con aplicaciones IoT requiere tener una amplia experiencia en la programación, existen un sinnúmero de lenguajes de programación para trabajar, los cuales conlleva conocer cada uno de sus componentes, variables, sintaxis, comandos y muchos aspectos que se necesita conocer para trabajar con un lenguaje de programación.

Al aumentar la demanda del internet y al estar todo conectado a la red los desarrolladores han creado facilidades para que todos puedan ser partícipe de esta tecnología por lo que se han creado nuevas técnicas de programación que permiten que desde los más pequeños puedan ir sumergiéndose en este mundo digital mediante la programación visual.

La programación visual consiste en realizar un flujo de trabajo mediante bloques diseñados de manera gráfica, por lo que no se necesita tener grandes conocimientos de programación para hacer uso de estos. Actualmente este lenguaje está haciendo utilizado en el ámbito educativo de tal manera que los más pequeños vayan desarrollando habilidades de programación sin tener profundos conocimientos sobre códigos. Se utilizan los elementos de programación los cuales ya están definidos según el lenguaje de programación visual que se esté utilizando.

Estos elementos de programación están diseñados como piezas de puzle, se agrupan en colores de acuerdo a la función que desempeñan.

Capítulo 1: Describe las necesidades y los requerimientos para la implementación de las técnicas de la programación visual.

Capítulo 2: Describe el desarrollo del proyecto, fundamentación teórica, diseño del prototipo con la ejecución y pruebas.

Capítulo 3: Describe los resultados de las pruebas obtenidas, conclusiones del proyecto y recomendaciones.

1. DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS

1.1 Ámbito de Aplicación: descripción del contexto y hechos de interés

El IoT (Internet of Things) es uno de los avances más importantes para la sociedad. Esta tecnología es una red de objetos integrados con electrónica, circuitos, software, sensores y conexiones de red que permiten la recopilación e intercambio de datos. A través de una infraestructura de red se controlan de forma remota los objetos, creando oportunidades para integrar el mundo físico a los sistemas informáticos. [3]

El internet de las cosas está inmerso en diferentes áreas de estudio ya sea en la medicina, agricultura, educación, en los hogares y en muchas otras áreas en las cuales se ha logrado automatizar procesos. Trabajar con IoT no es una tarea fácil pero existen una gran cantidad de artículos relacionados que permiten conocer más acerca de esta tecnología, ya que en el periodo entre 2006 y 2018 se han publicado unos 8.510 artículos de revistas y 16.775 artículos de actas de congreso referentes al internet de las cosas. [4]

Existen muchos lenguajes de programación que permiten trabajar con dispositivos inteligentes IoT, el que impera en la lista es el lenguaje de programación C y C++, ya que se los encuentra en la mayoría de microcontroladores para estos dispositivos. Para hacer uso de este lenguaje de programación se requiere tener un amplio conocimiento de cada uno de sus componentes, teniendo que invertir tiempo y recursos en cursos extensos.

Al ser IoT el área más emergente de la informática han surgido nuevas técnicas de programación como lo es la programación visual. El lenguaje programación visual (VPL) permite que el usuario cree y manipule programas gráficamente en lugar de codificarlos textualmente. [5] Actualmente este lenguaje está siendo utilizado con fines educativos. Por ejemplo, Scratch es un lenguaje de programación libre y gratuito el cual está diseñado para que los niños creen a través de una interfaz sencilla juegos y animaciones, promoviendo así el pensamiento computacional y la habilidad de resolver problemas de una manera creativa. [6]

Varias organizaciones han desarrollado los VPL para que de una manera fácil y fluida se pueda programar aplicaciones con solo hacer unos cuantos clics, arrastrar y soltar, sin saber a profundidad sobre el lenguaje. A partir de estas aplicaciones existentes se realizan prácticas utilizando el lenguaje de programación visual para realizar aplicaciones basadas en IoT.

1.2 Establecimiento de requerimientos

Los lenguajes de programación visual permiten realizar diferentes aplicaciones de manera fácil y con una interfaz amigable a través de bloques por lo que en este trabajo se realizarán diferentes prácticas utilizando lenguajes de programación visual diferentes de tal manera de demostrar el alcance que puede tener cada uno en la tecnología IoT.

A través del uso del microcontrolador ESP32 y la placa microbit, se mostrará como realizar un sistema IoT utilizando lenguajes de programación visual, partiendo desde prácticas básicas y terminando en prácticas estableciendo conexiones IoT.

1.3 Justificación de requerimiento a satisfacer

El Internet de las cosas va en aumento cada día, es por eso que se debe educar a las nuevas generaciones de ingenieros. Los nuevos empleos requieren habilidades para trabajar con las nuevas tecnologías, es por ello que la educación se vuelve un pilar importante para mejorar el potencial de los estudiantes para diseñar y construir sistemas tecnológicos.

Trabajar con IoT requiere tener conocimientos de programación avanzados por lo que puede resultar tedioso para los programadores novatos realizar aplicaciones

con esta tecnología. Sin embargo, para llamar la atención de las nuevas generaciones se han diseñado plataformas de hardware como herramientas de programación para expandir el desarrollo de aplicaciones IoT. Estas aplicaciones basadas en la programación en bloques han logrado que los programadores novatos mejoraran su pensamiento algorítmico y desarrollaran mayor interés por la informática. [7]

En este trabajo se utilizaran herramientas basadas en la programación en bloques, mediante las cuales se crearán prácticas utilizando las placas de ESP32 y microbit para crear prácticas básicas y prácticas para crear un sistema IoT. De esta manera mostrar el uso del lenguaje de programación visual facilita el desarrollo de sistemas IoT y evaluar el funcionamiento de las herramientas de acuerdo a las prácticas realizadas.

2. DESARROLLO DE PROTOTIPO

2.1 Definición del prototipo

Desarrollar sistemas IoT conlleva tener conocimientos profundos en los lenguajes de programación por ejemplo, para trabajar con las placas de arduino se debe conocer el lenguaje c, así también para elaborar aplicaciones móviles y web que permitan controlar los dispositivos conectados más lenguajes que aprender.

Los lenguajes de programación visual permiten implementar sistemas de una forma sencilla a través de bloques, es por ello que la propuesta tecnológica tiene como propósito mostrar las funcionalidades de las aplicaciones que trabajan con la programación basada en bloques. Se elaborarán prácticas de laboratorio para demostrar como trabaja la programación en bloques para la elaboración de sistemas IoT.

Para la elaboración de las prácticas se utilizarán las placas ESP32 y MICROBIT, las cuales serán programadas utilizando el lenguaje de programación visual. Dichas prácticas serán divididas en cuatro grupos: El primer grupo consta de prácticas básicas donde se muestre la conexión de periféricos a las placas; el segundo grupo consta de prácticas donde se muestre la comunicación con los periféricos del primer grupo a través de la comunicación WiFi utilizando el protocolo de comunicación MQTT; el tercer grupo será mediante la comunicación bluetooth; el cuarto grupo se

mostrará aplicaciones móviles para el control de los dispositivos de la práctica con la conexión bluetooth.

2.2 Fundamentación teórica

2.2.1 IoT

Existen muchas definiciones para IoT una de ellas lo define como "*un ecosistema físico de sensores y actuadores interconectados*" [8], otros como "*dispositivos interconectados que intercambian información a través de internet, información que una vez procesada se convierte en punto clave para la automatización de procesos"*. [9] Los sistemas IoT actualmente se los puede encontrar en diferentes aplicaciones como hogares inteligentes, agricultura inteligente, hospitales inteligentes entre otras. [10]

Mediante las aplicaciones IoT se realiza el monitoreo de datos y eventos en tiempo real, se capturan los datos enviados por los sensores inalámbricos, esta información se procesa para la toma de decisiones y control de actuadores. [11]

2.2.2 Arquitectura IoT

La arquitectura IoT está compuesta por dispositivos inteligentes, sensores, actuadores, elementos para establecer las conexiones inalámbricas como son los protocolos de comunicación, bases de datos en donde se almacena la información, software de monitoreo y control de los dispositivos Smart, además de implementar fuentes de información externas e integración a otros sistemas de negocio. [12]

Existen un sinnúmero de arquitecturas propuestas para IoT, para este trabajo se ha tomado como referencia la arquitectura de 3 capas: Dominio de Aplicación, Dominio de Red y Dominio de sensores. [13]

Figura 1 Arquitectura IoT

Fuente: Elaboración Propia

2.2.3 Lenguaje de Programación Visual

Visual Programming Language (VPL), permiten el desarrollo de aplicaciones utilizando elementos gráficos visuales en lugar de código textual, es decir son lenguajes de programación textuales con un generador de interfaz gráfica que permite al usuario programar de manera más sencilla.

Este lenguaje de programación utiliza un entorno de programación basada en bloques, los cuales están diseñados para que encajan como piezas de puzles. Actualmente están siendo utilizados en el ámbito educativo debido a su interfaz intuitiva permitiendo que programadores novatos se introduzcan en el mundo de la programación. [8]

La programación basada en bloques se está introduciendo en la educación a gran escala, países como Chile, Inglaterra, Sudáfrica, Japón, están implementando estrategias educativas y pedagógicas para que los estudiantes puedan aprender a desarrollar sus habilidades y su razonamiento lógico de manera creativa. [14] Con este método de programación se pretende atraer a personas de todas las edades a interesarse en codificar.

Existen un sinnúmero de aplicaciones basadas en la programación en bloques, para la realización de este estudio se tomarán en cuenta aquellas que posean la capacidad de programar en circuitos integrados para realizar la detección y el procesamiento de señales las cuales permitirán acceder a dispositivos IoT. En la ilustración 2 se pueden observar algunas aplicaciones basadas en la programación en bloques las cuales fueron tomadas en cuenta para la realización de un estudio de sistemas distribuidos con aplicaciones IA.

Name	Open Source	Web-based	Generated Code visual	Code gen. for Arduino	Arduino lib. usable	support of Microcontr.	Offline	Project start
Ardublockly	✓	✓	✓	✓	✓	✓	✓	22.04.2012
BlocklyDuinoReboot	✓	✓	✓	✓	✓	✓	✓	27.10.2013
Oxocard	✓	✓	✓	✓	✓	✗	✓	06.07.2017
Open Roberta	✓	✓	✓	✓	✗	✗	✗	02.03.2014
Blockly	✓	✓	✓	✗	✗	✗	✓	27.10.2013
MakeCode	✓	✓	✓	✗	✗	✗	✗	24.01.2016
Snap4Arduino	✓	✓	✗	✓	?	?	✓	31.01.2016
Scratch	✓	✓	✗	✗	✗	✗	✓	11.09.2016
ScratchJr	✓	✓	✗	✗	✗	✗	✓	03.01.2016
ToonTalk	✓	✓	✗	✗	✗	✗	✓	02.02.2014
BlockPy	✓	✓	✗	✗	✗	✗	✗	27.10.2013
NetsBlox	✓	✓	✗	✗	✗	✗	✗	10.03.2013
Snap!	✓	✓	✗	✗	✗	✗	✗	10.03.2013
ArduBlock	✓	✗	✓	✓	✓	✓	✓	06.02.2011
miniBloq	✓	✗	✓	✓	✓	✓	✓	25.08.2013
mBlock	✓	✗	✓	✓	✗	✗	✓	08.03.2015
S4A	✓	✗	✗	✓	✗	✗	✓	08.10.2013
Squeak	✓	✗	✗	✗	✗	✗	✓	?

Figura 2 Aplicaciones para la Programación basada en bloques

Fuente: [15]

Para la selección de las aplicaciones a utilizar también se considerarán los criterios como: [16] Usabilidad y soporte en las cuales se tomará en cuenta el idioma y el sistema operativo que puede ser ejecutado, además se considerará la capacidad del lenguaje de programación, es decir hasta qué punto se puede crear con esa

aplicación. [17] Su interfaz debe ser intuitiva, amigable y fácil de usar para la elaboración de proyectos con tecnología IoT.

2.2.3.1 Scratch

Desarrollado por el grupo de MIT Media Lab Lifelong Kindergarten Group y el Equipo Scratch de la Fundación Scratch, el cual fue lanzado el mayo del 2007, se podría descargar de forma gratuita como una aplicación de escritorio. [18] [19] Desde su lanzamiento el uso de Scratch sigue incrementando con el pasar del tiempo, debido a que proporciona variedad de proyectos que se encuentran de forma gratuita en la plataforma oficial.

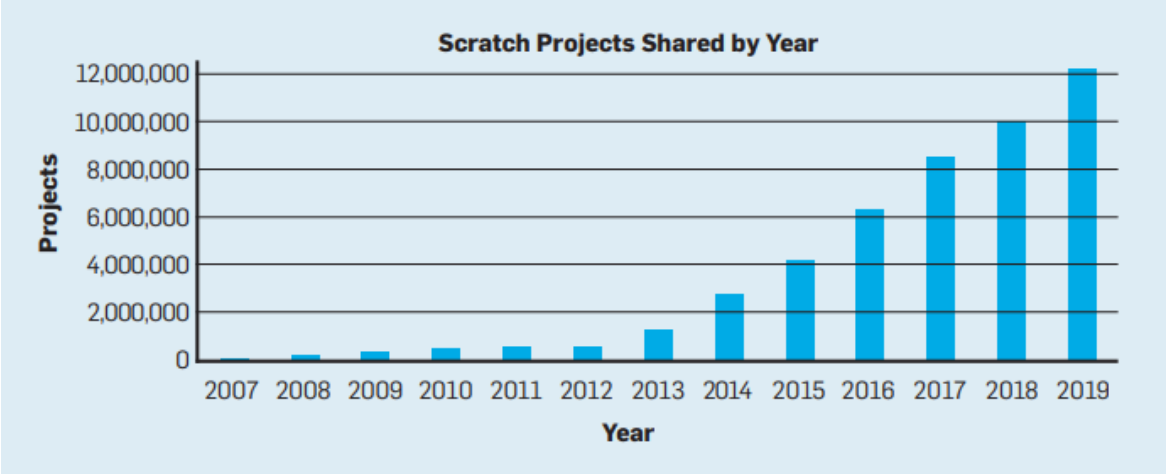


Figura 3 Proyectos compartidos en la comunidad en línea de Scratch

Fuente: [14]

Scratch es un herramienta de código abierto y gratuito, el cual tiene como sus precedentes a micromundo de Logo, e-toys de Squeak y LogoBlocks los cuales fueron la inspiración para su creación. Scratch puede ejecutarse desde el navegador web o descargado como aplicación. [14]

A pesar de ser una herramienta que fue ideada para jóvenes es una gran herramienta para la introducción a la programación, ya que permite aprender conceptos importantes utilizados en muchos lenguajes de programación como son las iteraciones, flujos de control, condicionales, variables, eventos y procesos. [20]

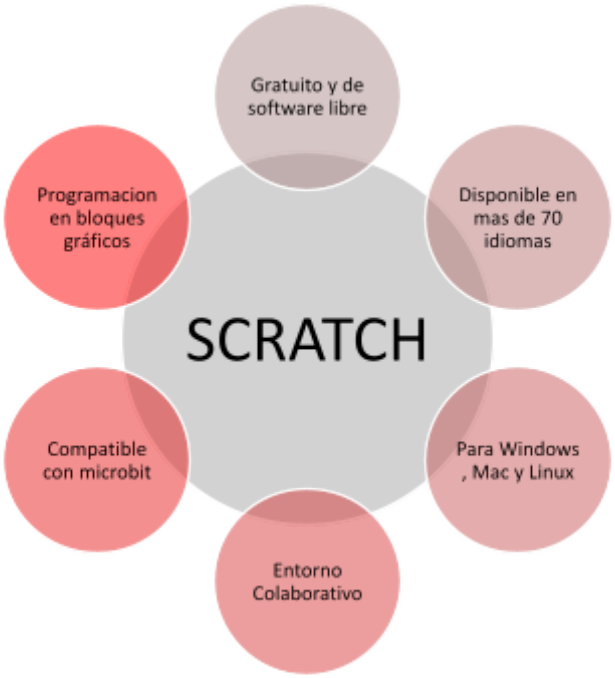


Figura 4 Características de Scratch

Fuente: Elaboración propia

A pesar de ser uno de los lenguajes más utilizados dentro de la programación en bloque, únicamente se puede trabajar con 11 extensiones que posee la aplicación web y la de escritorio, por lo que está limitado a la hora de realizar trabajos IoT. Tiene cinco extensiones que permiten realizar programación de software y seis extensiones para realizar la programación en hardware.

Tabla 1 Extensiones de SCRATCH

EXTENSIONES DE SCRATCH	
Música	Permite crear nuevas melodías utilizando 31 tipos de instrumentos musicales.
Lápiz	Permite realizar diferentes tipos de trazos, en donde se manipula el color, saturación, brillo, transparencia y el tamaño del lápiz.
Sensor de video	Permite interactuar con la webcam en la cual detecta los movimientos.

Texto a voz	Permite insertar voz y sonido a las aplicaciones que se estén desarrollando.
Traducir	Permite traducir texto a muchos idiomas.
Makey Makey	Permite incorporar controles convirtiendo cualquier tecla en un mando interactivo,
Micro:bit	Permite establecer conexión y programar la placa de micro:bit.
LEGO MINDSTORMS EV3	Permite realizar la programación con los robots de Lego.
LEGO BOOST	Permite realizar la programación de los robots de Lego.
LEGO Education WeDo 2.0	Permite realizar la conexión mediante bluetooth para controlar los sensores de Lego.
Go Direct Force y Acceleration	Permite la conexión a un sensor de fuerza y aceleración.

Fuente: Elaboración Propia

Actualmente Scratch lanzó su versión 3.29.1 la cual tiene como novedad que no requiere flash a la hora de ejecutarlo. Además de que se puede migrar los proyectos de la versión 2.0 a la versión actual sin ningún inconveniente.

2.2.3.2 CodeSkool

Basada en la web es un editor de código basado en bloques, su entorno es muy parecido a la de Scratch. Se lo encuentra en el link https://ide.codeskool.cc/, y mediante este es posible trabajar con un sinnúmero de extensiones, permitiendo crear proyectos para Arduino, Machine Learning e Inteligencia Artificial.

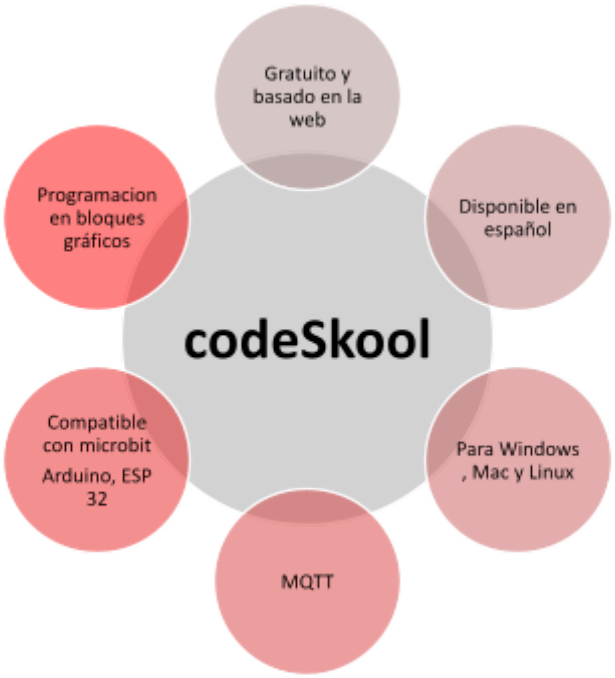


Figura 5 Características de codeSkool

Fuente: Elaboración Propia

2.2.3.3 ArduinoBlocks

Plataforma web en línea que permite escribir código utilizando bloques, funciona a través del navegador, lo que permite utilizarlo en cualquier sistema operativo sin inconvenientes. Para hacer uso de este editor es necesario realizar la descarga de un paquete llamado ArduinoBlocks-Connector, lo que permite que se pueda trabajar con la placa a través del navegador. [21]

Esta plataforma compila y sube la programación a la placa a través de la conexión USB. Se puede trabajar con un sinnúmero de sensores y actuadores modulares. Todas las funciones están predefinidas para hacer uso de estas con solo arrastrar un bloque y realizar un algoritmo ordenado que permite ejecutar un acción dentro de la placa que se está utilizando.

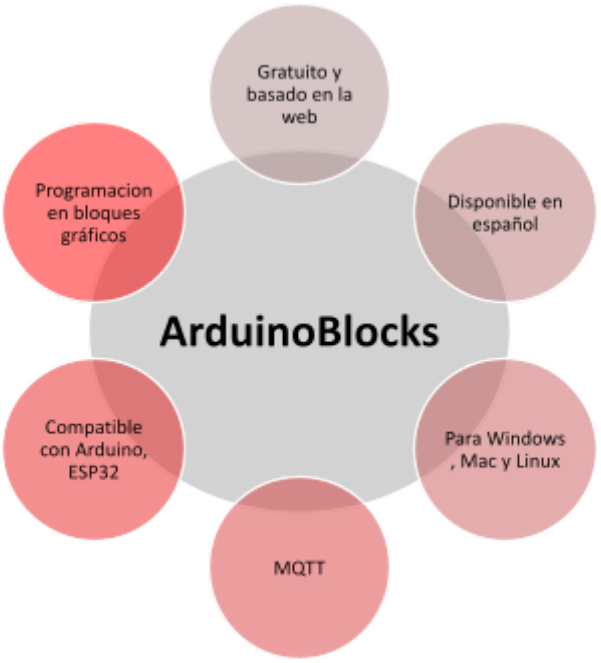


Figura 6 Características de ArduinoBlocks

Fuente: Elaboración Propia

2.2.3.4 MakeCode

Plataforma web gratuita de código abierto que permite crear proyectos utilizando la programación basada en bloques. Su aplicación web posee editor de código de bloques como Blockly y Monaco, código JavaScript y Python. Además, posee un simulador para realizar pruebas antes de transferirlos al dispositivo físico. No es necesaria su instalación y es compatible con cualquier navegador web.

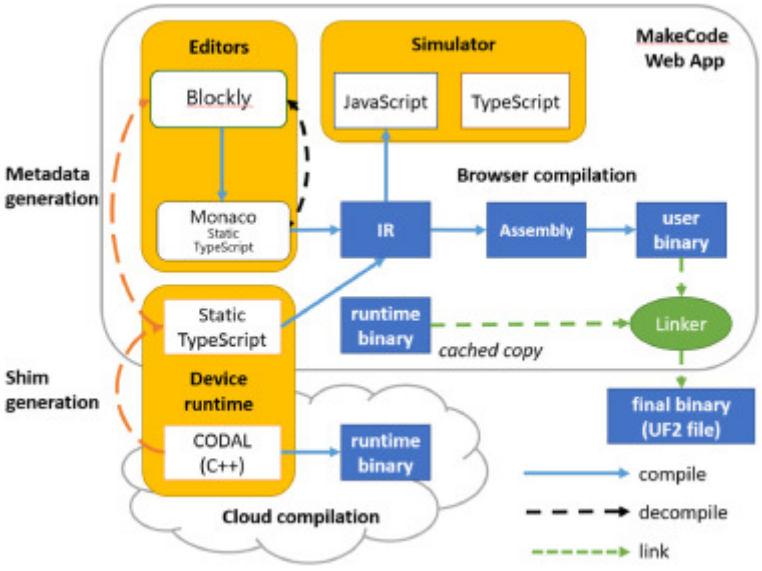


Figura 7 Arquitectura de MakeCode

Fuente: [22]

MakeCode está implementado en TypeScript que es un lenguaje de programación orientado a objetos superior a JavaScript, posee mejoras adicionales que permite desarrollar aplicaciones con menos errores de código y más sencillo, es importante señalar que todo código que esté escrito en JS es válido por TS.

Utilizando esta plataforma se pueden crear proyectos con la placa micro:bit, programar con Circuit Playground Express, Minecraft, LEGO MINDSTORMS, Cue, Arcade y Chini Chip. Esta herramienta será utilizada para la programación con la placa micro:bit.

2.2.3.5 App inventor

Plataforma de desarrollo en línea para la creación de aplicaciones móviles Android y sistemas operativos iOS. Posee dos editores principales: editor de diseño que permite crear la interfaz de manera sencilla, solo arrastrar y soltar componentes para crear nuestra aplicación. El editor de bloques basado en Blockly , al igual que Scratch codificados en bloques por colores que funcionan como piezas de rompecabezas. Contiene un emulador llamado App Inventor Companion que permite realizar pruebas en tiempo real. [23]

2.2.4 ESP32

Módulo de bajo costo que permite la conectividad Wifi y Bluetooth con frecuencia 2.4GHz, en la figura 8 se puede observar todos los componentes que integran la placa ESP32. Gracias a estos componentes se pueden realizar sistemas IoT con bajo costo e integrando funciones avanzadas. [24] [25] Este módulo se puede programar con el Arduino IDE, el programa se almacenará en la memoria que posee.

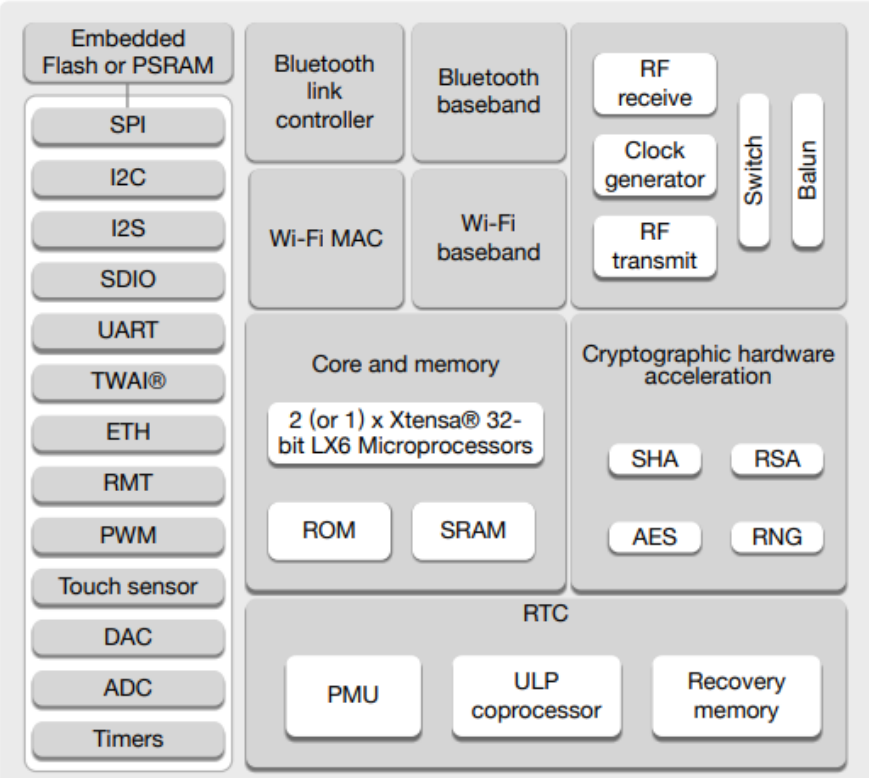


Figura 8 Diagrama de bloques ESP32

Fuente: [26]

El dispositivo contiene interfaces periféricas las cuales incluyen , I2S, I2C, UART, interfaz SPI, LED, PWM, LCD, sensor táctil, DAC. Contiene un CPU de dos núcleos y un microcontrolador de 32 bits. [27]

2.2.5 BBC MICRO:BIT

Placa programable que posee un procesador de 32 bits, con 16k de RAM y funciona a 16 MHz. Es una pequeña placa que puede ser programada en línea, tiene conexión con puerto USB. A través de su página oficial https://microbit.org/ se puede encontrar un sinnúmero de ejemplos y tutoriales que su sitio web ofrece. Se puede programar en diferentes entornos de desarrollo en línea cómo MakeCode, Scratch, Open Roberta Lab entre otros, los cuales se basan en la programación en bloques.

Tabla 2 Características de micro:bit

MICRO:BIT		
	**CPU**	32-bit ARM
	Tamaño	4x5 cm
	Pines GPIO	3
	Antena	Bluetooth
	Voltaje de operación	3V
	RAM	16KB
	BOTONES	2

Fuente: [28]

Se pueden utilizar pinzas de cocodrilo para poder conectar componentes externos. Esta pequeña placa puede trabajar con motores, sensores y altavoces para crear un sinnúmero de tecnologías para ser controladas a través de IoT. [29]

2.2.6 Sensores

Los sensores son componentes que pueden medir una magnitud de acuerdo a la sensibilidad que posean a ciertas propiedades. Existen diferentes tipos de sensores, estos varían de acuerdo a la propiedad física o química que se busca calcular.

Tipos de sensores según variable física a medir	De posición, velocidad y aceleración
	De nivel y proximidad
	De humedad y temperatura
	De fuerza y deformación
	De flujo y presión
	De color, luz y visión
	De gas y pH
	Biométricos
	De corriente

Figura 9 Clasificación de sensores por el tipo de variable medida
Fuente: [30]

2.2.6.1 DTH 22

Dispositivo digital que detecta la temperatura y humedad relativa, mediante el cual se mide el aire circundante, ya que contiene un sensor capacitivo de humedad y termistor.

Este sensor puede ser utilizado en plataformas como Raspberry Pi, Arduino entre otras, presentando facilidades a la hora de aplicarlas tanto a nivel de software como hardware. En el ámbito del software, existen gran variedad de librerías para ser aplicadas en Arduino IDE como en las otras plataformas. En cuanto al hardware se refiere, el dispositivo tiene un pin VCC de alimentación de 3-5V para conectar a la corriente eléctrica, un pin GND para la conexión a Tierra (0V),un pin nulo que no puede ser utilizado y un pin de datos que se conecta al pin digital en Arduino. Para utilizar distintos sensores DHT22 con un mismo Arduino, se necesita que cada sensor se conecte mediante su propio pin de datos.

Se recomienda proteger el sensor de la luz directa del sol. La única desventaja del sensor es que sólo muestra nuevos datos cada 2 segundos.

2.2.6.2 SENSOR ULTRASONIDO HC-SR04

El sensor de ultrasonido permite medir la distancia en la que se encuentra un objeto, debido a que posee un emisor y receptor de ultrasonidos.

Tabla 3 Características del Sensor HC-SR04

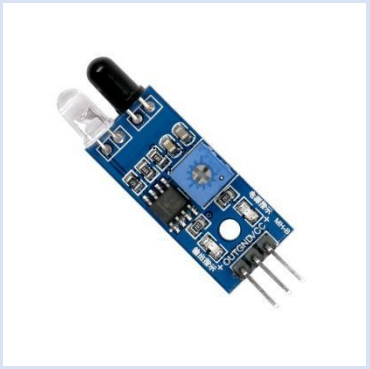
SENSOR ULTRASONIDO		
	**Voltaje de Operación**	5v
	Ángulo de apertura	15°
	Rango de medición	2cm a 450cm
	Frecuencia de ultrasonido	40KHz

Fuente: Elaboración propia

Sin duda una de las mejoras alternativas a utilizar para la detección de movimiento debido a la facilidad que proporciona al momento de trabajar con este módulo, además de su bajo costo.

2.2.6.3 SENSOR IR

Dispositivo electrónico que contiene un LED infrarrojo y un fototransistor, el primero se encarga de lanzar señales infrarrojas y el fototransistor es el encargado de receptar las señales que se encuentren en su campo de visión.

SENSOR IR		
	**Voltaje de Operación**	3.3v-5v
	Ángulo de apertura	35°
	Rango de medición	20mm a 300mm

Fuente: Elaboración propia

2.2.7 Actuadores

Dispositivos capaces de generar fuerza con la finalidad de lograr un cambio de posición, velocidad o estado la cual funciona a través de la transformación de energía. [30]

Tipos de actuadores	Neumáticos
	Hidráulicos
	Eléctricos

Figura 10 Tipos de Actuadores

Fuente: [30]

2.2.7.1 RELÉ

Componente electromagnético que es utilizado como interruptor eléctrico, está compuesto por una bobina que al recibir corriente, crea un campo magnético, el cual dependiendo del tipo de relé, permitirá abrir o cerrar el paso de la corriente eléctrica.

2.2.7.2 LED

Componente electrónico que funciona cuando se transmite una tensión a la terminal positiva y como resultado este emite luz . Existen de diferentes colores o RGB, son de bajo costo y fácil de utilizar.

2.2.7.3 LCD

LCD que hace referencia a Pantalla de Cristal Líquido por sus siglas en inglés, su característica principal es que al estar formada por píxeles a través de las moléculas que están contenidas en el cristal líquido permite el paso de luz, la cantidad de luz variará de acuerdo al estímulo eléctrico que esta reciba.

2.2.8 Tecnologías inalámbricas para IoT

IoT es un nuevo paradigma que tiene como objetivo conectar cualquier dispositivo a través de la red ya sea esta inalámbrica o cableada. Para realizar las conexiones inalámbricas se pueden utilizar Wifi, Bluetooth ZigBee y LoRa, las cuales son las más utilizadas. [13]

Para realizar las conexiones IoT en las prácticas de laboratorio se utilizarán las conexiones inalámbricas a través de Wifi y Bluetooth.

2.2.8.1 Wifi

Al estar conectados a internet se transmite información a través de ondas de radio. Mediante esta tecnología se construyen WLANs(Wireless Local Area Networks) la cual es un punto de acceso para poder conectarnos desde diferentes dispositivos.

La tecnología inalámbrica está basada en el estándar IEEE 802.11, en sus inicios permitía una transmisión de 2Mbps, conforme ha evolucionado la tecnología ha mejorado en muchos aspectos como la calidad del servicio, seguridad, administración de energía, entre otros. Hoy en día se puede enviar datos con una velocidad de transmisión de 6,75 Gbps con un rango superior a 382 Km. [31]

2.2.8.2 Bluetooth

Tecnología inalámbrica que permite el intercambio de información con un corto alcance a través de ondas de radio. Basado en el estándar IEEE 802.15.1, emplea el modelo de comunicación maestro-esclavo. Para establecer la comunicación ambos dispositivos deben estar previamente autenticados, la cual realizan un

emparejamiento donde se le asigna a un dispositivo el papel de maestro, los demás dispositivos aceptan el emparejamiento convirtiendo a los maestros en esclavos. [31]

Bluetooth Low Energy (BLE) está siendo muy utilizado por los desarrolladores IoT, ya que todos los teléfonos inteligentes cuentan con esta tecnología. Además de ser compatible con los sistemas operativos (Android, Linux, OSX y Windows). [32] [33]

2.2.9 Protocolo de comunicación

2.2.9.1 MQTT

Protocolo de Publicación/suscripción (pub/sub) desarrollado por IBM, permite la comunicación de máquina a máquina (M2M). Es un protocolo muy popular dentro de la tecnología IoT debido a su sencillez, funciona de tal manera que un cliente realiza la publicación de un tema y otros clientes tienen la opción de suscribirse a este tema, de esta manera se pueden intercambiar mensajes entre los clientes que publican y se suscriben por lo que es un excelente protocolo para las conexiones con dispositivos IoT . [34] [35]

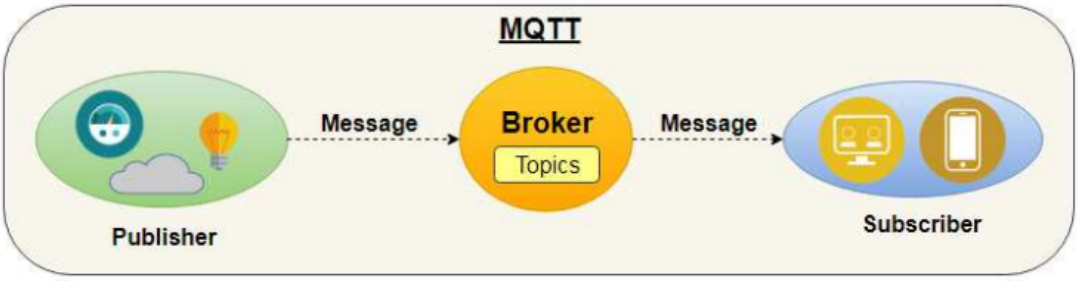


Figura 11 Arquitectura MQTT

Fuente: [34]

Trabaja con TCP/IP y actúa sobre SSL/TSL para establecer una conexión y cifrado de datos seguro. Soporta tres clases de QoS para garantizar la entrega de mensajes. Una de las grandes ventajas de MQTT es su gran compatibilidad con dispositivos pequeños de recursos limitados y que poseen bajo ancho de banda, lo cual lo convierte en la mejor opción para trabajar con dispositivos IoT. [36] [37]

Tabla 4 Paquete de control MQTT

PAQUETE DE CONTROL	DESCRIPCIÓN
CONNECT	Solicitud conexión al servidor
CONNACK	Reconocimiento de la solicitud de conexión
PUBLISH	Mensaje de publicación
PUBACK	Publicar reconocimiento
PUBREC	Publicación recibida (Qo2)
PUBREL	Liberar publicación (Qo2)
PUBCOMP	Publicación completa (Qo2)
SUBSCRIBE	Cliente suscrito al tópico
SUBACK	Servidor reconoce la solicitud de suscripción del cliente
UNSUBSCRIBE	Cliente cancela la suscripción
UNSUBACK	Servidor reconoce la solicitud de cancelar la suscripción del cliente
PINSREQ	Cliente hace PING con el servidor
PINGRESP	Servidor responde el PING al cliente
DISCONNECT	Cliente envía notificación de desconexión para que el servidor borre la conexión

Fuente: [38]

Para la realización de prácticas IoT se hará uso de este protocolo de comunicación ya que es uno de los más utilizados para IoT debido a que es un protocolo ligero para realizar la trasmisión de los datos a través de la red entre los dispositivos IoT y el cual trabaja con dispositivos de capacidades limitadas.

2.3 Objetivos del Prototipo

2.3.1 Objetivo General

Implementar técnicas de programación visual mediante el uso del microcontrolador ESP32 y la placa Microbit para el desarrollo de sistemas IoT.

2.3.2 Objetivo Específicos

- Investigar artículos científicos sobre la programación visual utilizada para la creación de sistemas IoT.
- Seleccionar aplicaciones de programación visual para el desarrollo de prácticas de laboratorio.
- Realizar prácticas de laboratorio usando técnicas de programación visual utilizando la placa ESP32.
- Realizar prácticas de laboratorio usando técnicas de programación visual utilizando microbit.
- Realizar prácticas de laboratorio utilizando el protocolo de comunicación MQTT para la creación de sistemas IoT.
- Realizar prácticas de laboratorio utilizando el módulo de bluetooth de la placa ESP32 y MICRO:BIT.
- Elaborar una aplicación móvil utilizando técnicas de programación visual para el manejo de los sistemas IoT.
- Analizar el funcionamiento de las aplicaciones utilizadas a través de las prácticas de laboratorio.

2.4 Diseño del Prototipo

Estará dividido en 4 grupos, en el cual el grupo 1 serán prácticas básicas con la Placa ESP32, en el grupo 2 se harán prácticas con las conexiones del grupo 1 utilizando la comunicación wifi para ello se utilizará el protocolo de comunicación MQTT con la EPS32, en vista que la placa microbit solo tiene conexión bluetooth, el grupo 3 se crearán prácticas utilizando la conexión bluetooth en la ESP32 y la microbit, y por último el grupo 4 se realizarán las prácticas de control de las placas a través de aplicaciones móviles.

2.4.1 GRUPO 1 : PERIFÉRICOS

Tabla 5 Listado de prácticas

No. PRACTICA	Nombre de la practica	Objetivo
PRACTICA 1	ENCENDIDO DE LED	Encender y apagar leds utilizando los sensores táctiles capacitivos de la ESP32.
PRACTICA 2	LCD y DTH22	Leer la temperatura y humedad del sensor DTH22 y mostrar los resultados a través de la pantalla LCD.
PRÁCTICA 3	SENSOR ULTRASONICO – RGB – BUZZER	Crear una alarma de detección de objetos mediante el uso del sensor ultrasónico.
PRÁCTICA 4	SENSOR LDR	Encender o apagar un led de acuerdo al nivel de luz que incide sobre el sensor LDR.
PRÁCTICA 5	RELÉ Y PULSADOR	Enviar un pulso y activar el circuito para encender un foco.
PRACTICA 6	RECEPTOR INFRARROJO	Receptar las señales del control remoto y encender las luces del led RGB de acuerdo al botón presionado.

Fuente: Elaboración propia

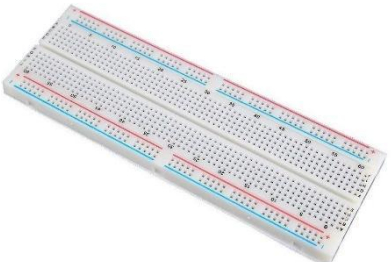
2.4.1.1 PRÁCTICA 1: ENCENDIDO DE LED

Objetivos: Encender y apagar leds utilizando los sensores táctiles capacitivos de la ESP32.

Descripción: Para la realización de la práctica se utiliza la herramienta de programación basada en bloques arduinoblocks y se hace uso del pin 14 para tener una lectura de capacitancia.

- ESP32 posee 10 pines que permiten obtener valores de capacitancia.

Tabla 6 Materiales para Práctica 1

DESCRIPCIÓN	PRESENTACIÓN
ESP 32	
Cable USB	
Cables puente para protoboard	
Leds	
Resistencia 220 Ohm	
Protoboard	

Fuente: Elaboración Propia

En la tabla 6 se muestran los materiales a utilizar para la práctica con la placa EPS32, a continuación se muestra el diagrama de conexiones. Para la elaboración del diagrama de conexiones se utilizó la herramienta fritzing.

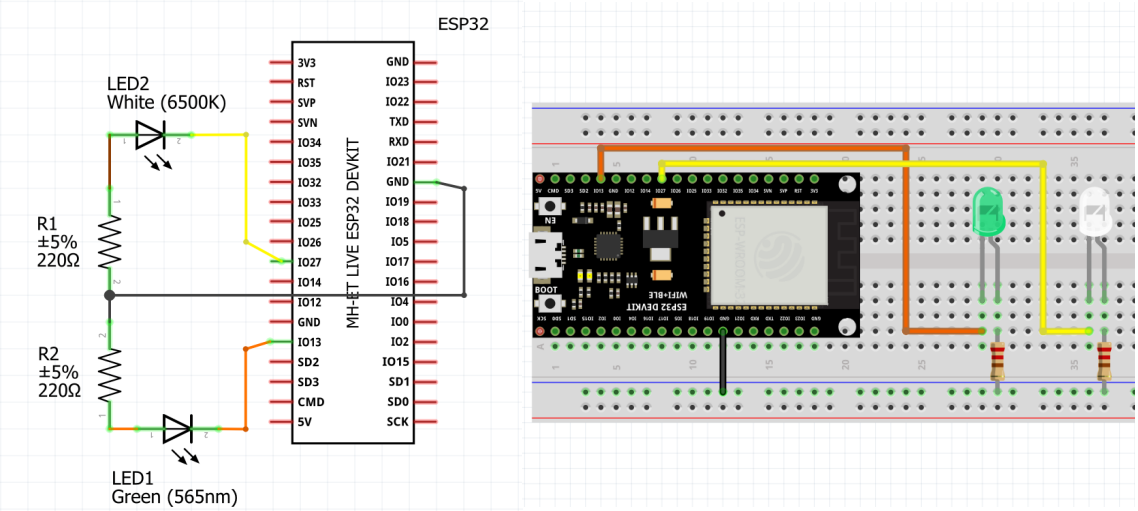


Figura 12 Diagrama de conexiones práctica 1

Fuente: Elaboración propia

Conexión de los componentes a la placa ESP32

El pin 13 estará conectado al positivo del led Blanco y el negativo a la resistencia de 220 que va a tierra.

El pin 27 estará conectado al positivo del led Verde y el negativo a la resistencia de 220 que va a tierra.

El pin 14 el cual será utilizado como un lector de entrada capacitiva para el encendido de los leds.

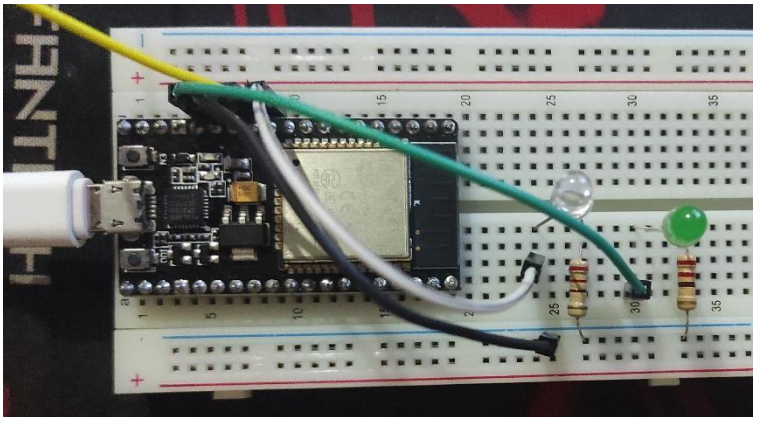


Figura 13 Conexión de la Práctica 1 en la protoboard

Fuente: Elaboración propia

2.4.1.1.1 CodeSkool

Para dirigirnos a CodeSkool lo encontramos en el siguiente enlace https://ide.codeskool.cc/. Creamos una cuenta de codeSkool para poder guardar los prácticas creadas.

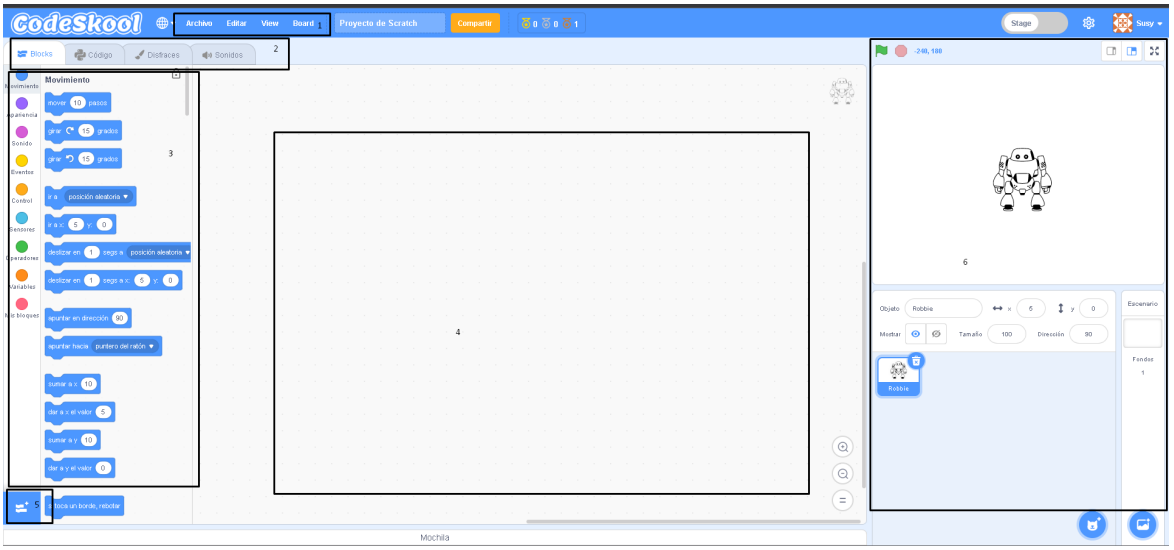


Figura 14 Pantalla principal de CodeSkool

Fuente: Elaboración Propia

En la sección 1 nos encontraremos con el menú desplegable de archivo en el cual podemos crear un nuevo proyecto, buscar ejemplos, abrir un proyecto desde un link específico, cargar un proyecto desde el ordenador y guardar el proyecto en el ordenador. Editar encontramos la opción de restaurar y activar el modo Turbo. En View encontramos tutoriales acerca de CodeSkool, cursos, visor de gráficos y abrir la pantalla de la consola. Board nos permite agregar las placas con las que vamos a trabajar, establecer conexión con las placas.

En la sección 2 tenemos cuatro opciones la primera Blocks muestra los bloques que se usan para la programación. Código que permite agregar código Python. Disfraces permite editar los objetos o imágenes que se estén utilizando. La opción sonidos permite agregar sonidos o editarlos.

La sección 3 Blocks muestra los bloques que están divididos por colores de acuerdo a su categoría de color azul los bloques de movimiento que permiten controlar un objeto, morado controla la apariencia de un objeto, rosado control de sonido, amarillo control de eventos para disparar los scripts, naranja contiene los bloques de control de flujo para ejecutar scripts, celeste control de sensores, verde operadores,

naranja para crear variables y color rojo permite crear bloques con funciones propias.

La sección 4 es el área de trabajo en la que se arrastran los bloques con los que se va a programar.

La sección 5 es la opción que permite agregar nuevas extensiones para trabajar con CodeSkool, programación, hardware, internet, juegos, robótica, machine Learning. Al elegir las extensiones se presentarán los bloques en la sección de blocks en la parte final.

Por último la sección 6 que nos muestra la imagen del objeto sobre el que estamos trabajando.

Para empezar a realizar la práctica con el ESP32 nos dirigimos al codeSkool para agregar la placa en parte de Board y damos clic en install software, para ello debemos tener la placa conectada con nuestro cable USB a la computadora.

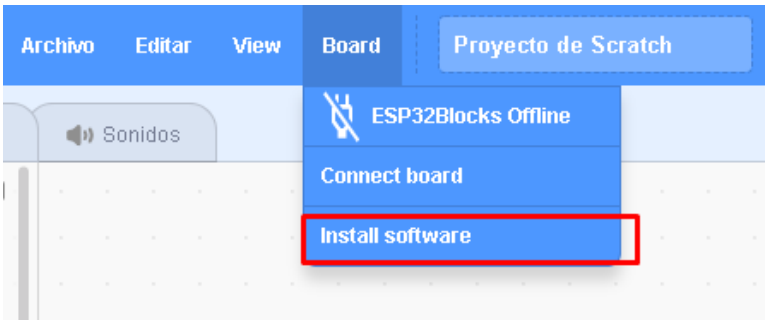


Figura 15 Instalar placa

Fuente: Elaboración propia

Se presentará una pantalla en donde seleccionaremos la placa ESP32, además nos mostrará las instrucciones que debemos seguir para realizar la instalación. Se debe mantener presionado el botón boot para actualizar el firmware de la placa.

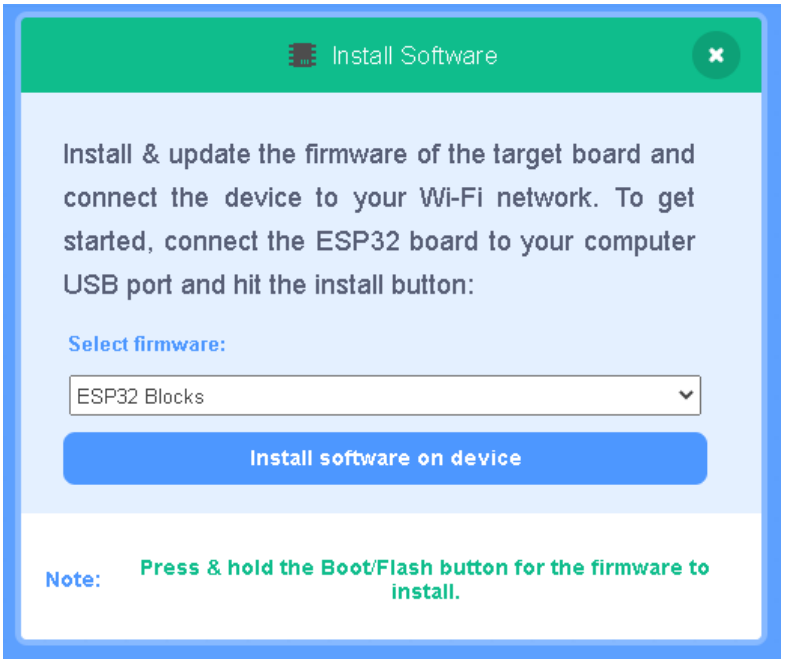


Figura 16 Ventana de instalación de la placa ESP32

Fuente: Elaboración propia

Se presentará una nueva ventana en donde se debe seleccionar el puerto en el que se encuentra la placa.

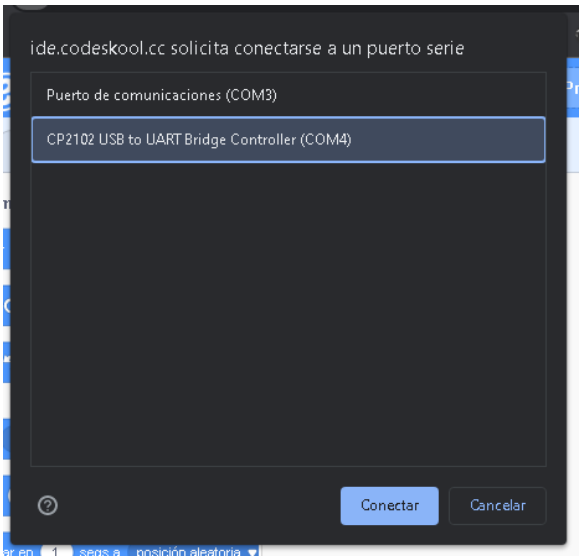


Figura 17 Selección de puerto

Fuente: Elaboración propia

Se mostrará un mensaje en donde se elegirá la instalación del diagrama de bloques de ESP32 para poder cargar los datos a la placa.

Figura 18 Instalación de ESP32 Blocks

Fuente: Elaboración propia

Por último mostrará un mensaje de advertencia en el que nos indica que se perderán todos los datos del dispositivo. Marcamos la opción de borrar dispositivo y damos clic en siguiente. En el momento de dar clic en siguiente no olvidar tener presionado el botón de boot para que se pueda realizar la instalación.

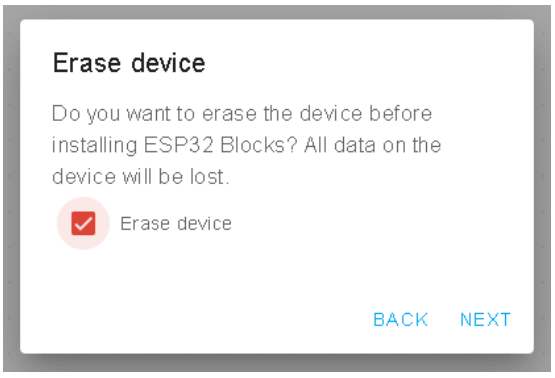


Figura 19 Mensaje de advertencia

Fuente: Elaboración propia

Con el botón de boot presionado esperaremos que se complete al 100% la instalación.

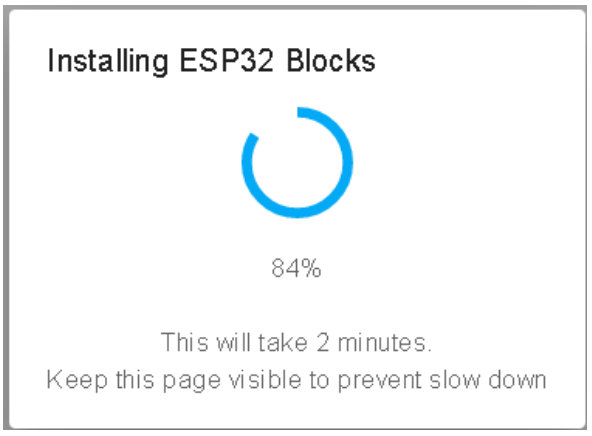


Figura 20 Instalando ESP32 Blocks

Fuente: Elaboración propia

Al terminar la instalación se mostrará un mensaje de instalación completa y damos clic en siguiente.

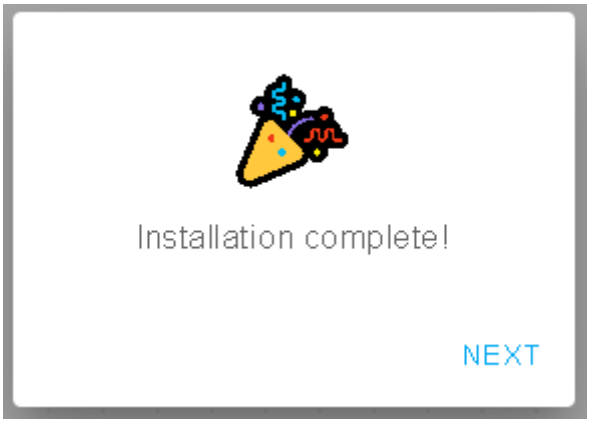


Figura 21 Instalación completa

Fuente: Elaboración propia

En la siguiente ventana se procede a configurar la red de WIFI de la placa ESP32 en donde colocaremos los datos de nuestra red local y damos clic en siguiente.

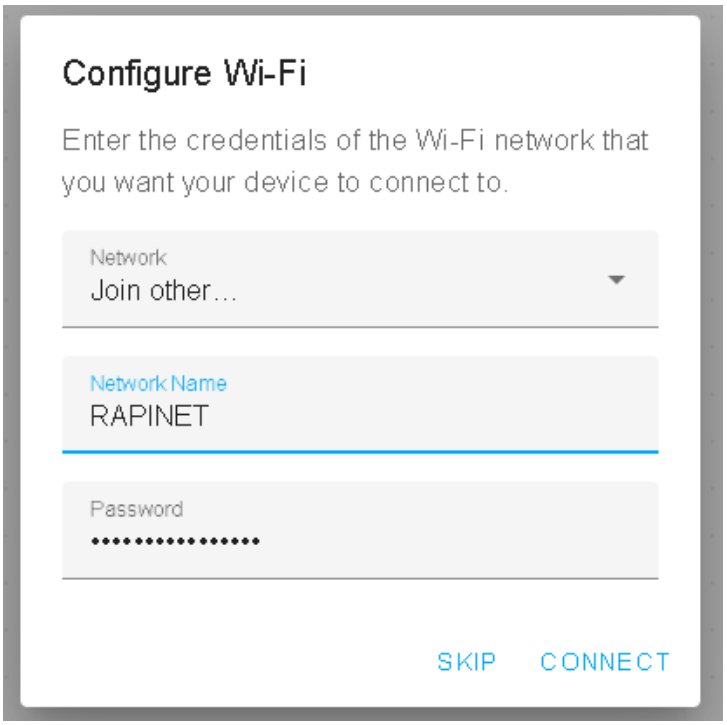


Figura 22 Configuración WIFI

Fuente: Elaboración propia

Comenzará la configuración del WIFI y al estar completa nos mostrará un mensaje de dispositivo conectado.

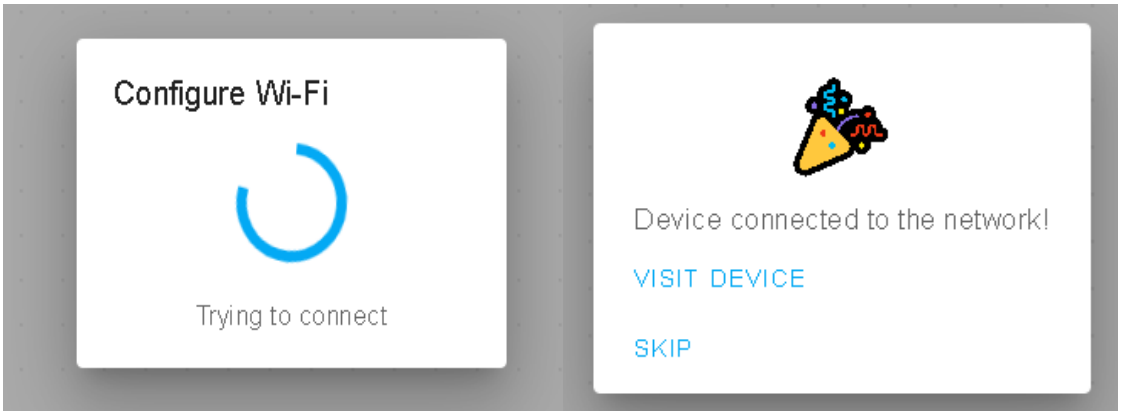


Figura 23 Conectado a la red

Fuente: Elaboración propia

Damos clic en visitar dispositivo y colocamos los datos de nuestra cuenta de codeskool para agregar el dispositivo a la cuenta creada.

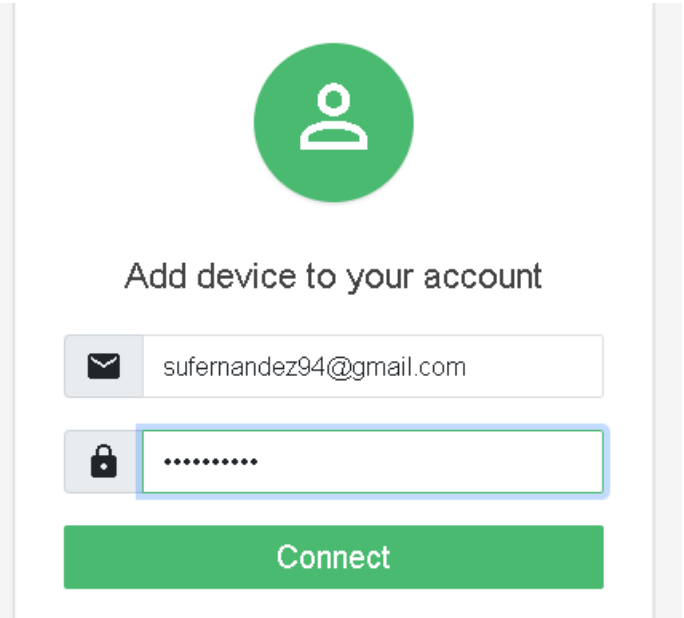


Figura 24 Agregar dispositivo a la cuenta de CodeSkool

Fuente: Elaboración propia

Una vez agregado a la cuenta nos dirigimos a CodeSkool y nos mostrará un último mensaje de lo que hemos instalado nuestra placa ESP32. Cerramos esa ventana dando clic en la x.

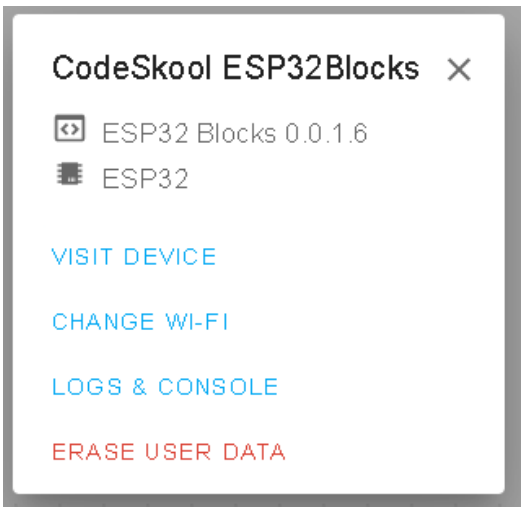


Figura 25 ESP32 instalada

Fuente: Elaboración propia

Antes de realizar la conexión a la placa debemos de dirigirnos a la configuración del sitio.

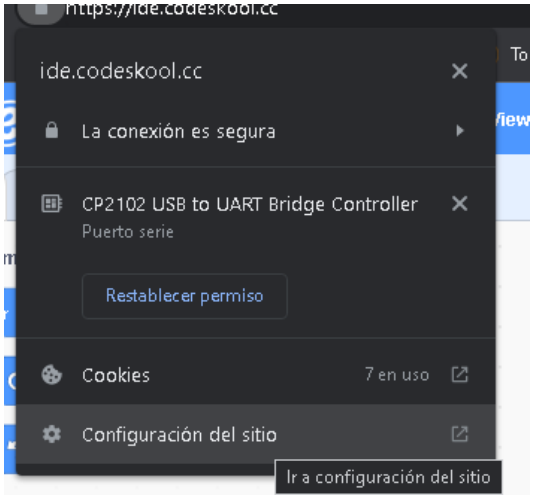


Figura 26 Configuración del sitio CodeSkool

Fuente: Elaboración propia

Para poder conectar la placa debemos configurar la página web en la que indicamos que el contenido es seguro para que el navegador pueda detectar la placa ESP32 y poder trabajar con esta. Actualizamos la página web y reseteamos la placa.

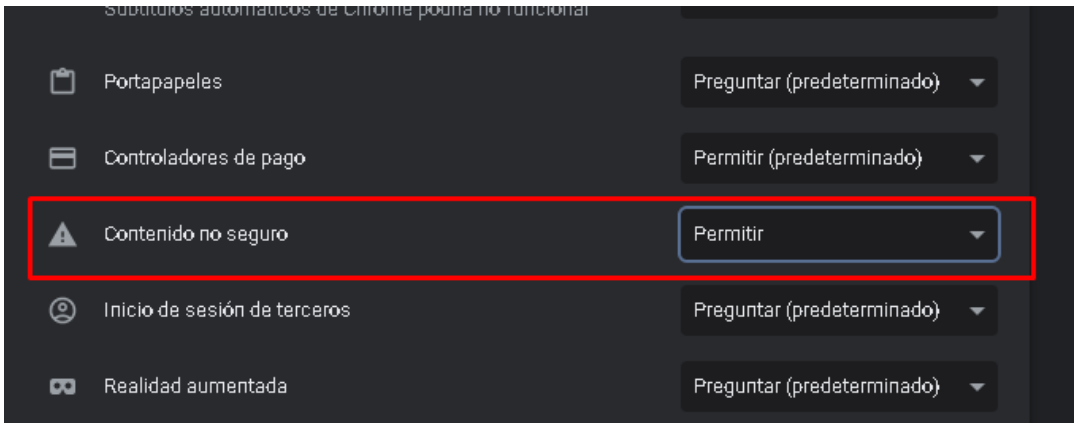


Figura 27 Configuración de sitio

Fuente: Elaboración propia

Una vez que se han realizado los pasos anteriores de configuración nos dirigimos a Board y conectar Board.

Figura 28 Conectar placa

Fuente: Elaboración propia

Nos mostrará un mensaje que nuestro dispositivo se ha conectado, procedemos a cerrar la ventana.

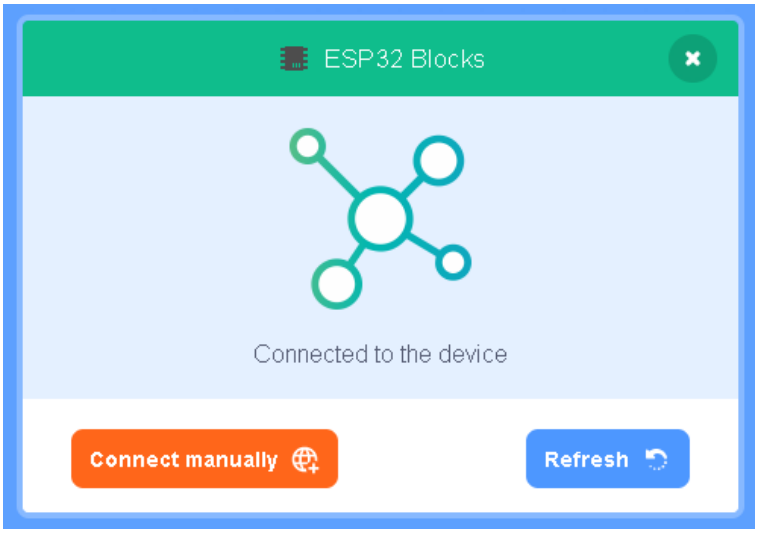


Figura 29 ESP32 conectada

Fuente: Elaboración propia

Como último paso para verificar si nuestra placa está conectada nos dirigimos a Board en donde se mostrará el ESP32Blocks online.

Figura 30 ESP32Blocks online

Fuente: Elaboración propia

Una vez que se ha conectado la placa procedemos agregar la extensión de Arduino en para poder realizar la práctica.

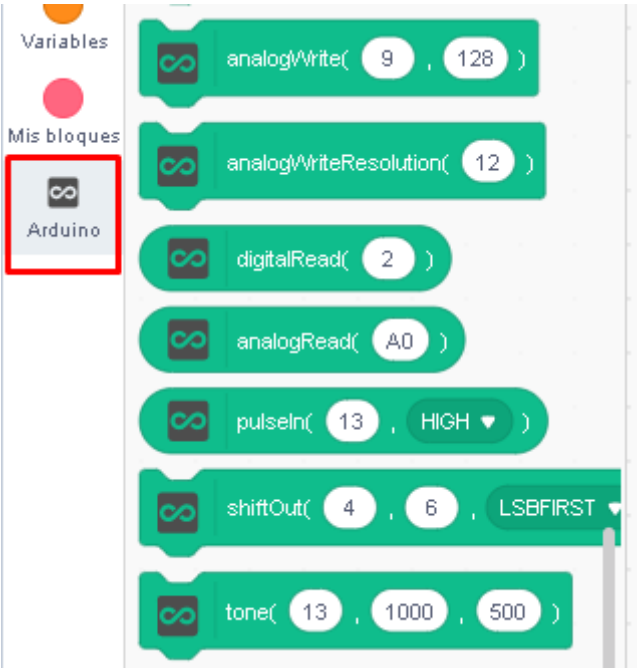


Figura 31 Extensión de Arduino

Fuente: Elaboración propia

Esta práctica consiste en realizar el apagado y encendido del led mediante el cual se ha utilizado el pin 14 que es uno de los pines táctiles capacitivos que posee la ESP32, estos pines detectan la carga eléctrica de los objetos por lo que al estar en contacto con la piel procede a enviar valores que permiten encender el led verde y cuando no detecta ningún valor permanece encendido el led blanco.

Esta extensión posee las funciones que son utilizadas para la programación con Arduino. Se crearán dos variables led1 y led2 las cuales se les asignará el valor de pin 12 y 27 de la placa del ESP32 donde estarán conectados los leds. Con la función pinMode se especifica si será un valor de entrada o salida por lo que ambos serán OUTPUT valores de salida. Entrará en un bucle repetitivo con un delay de 500 ms en donde la condición será que si el valor del sensor táctil T06 que corresponde al pin 14 es menor a 20 se enviará un salida digital al led 1 con valor de alto que enciende el led y el led 2 con valor bajo para apagar el led caso contrario al tener valores superiores a 20 el led 1 procede a estar apagado y el led 2 encendido.

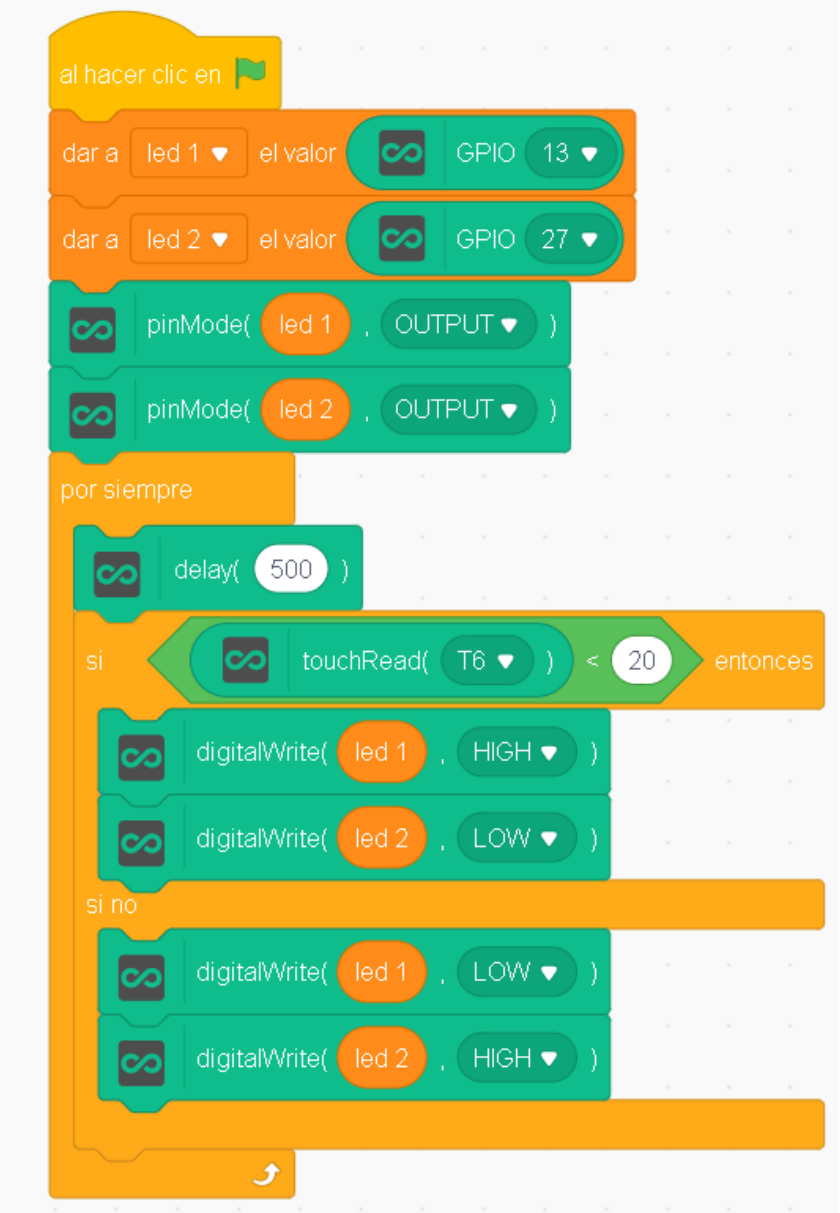


Figura 32 Código de práctica 1 ESP32 y CodeSkool

Fuente: Elaboración propia

2.4.1.1.2 ArduinoBlocks

Arduinoblocks lo encontramos en el siguiente enlace www.arduinoblocks.com/ , crearemos una cuenta para poder almacenar nuestros proyectos.

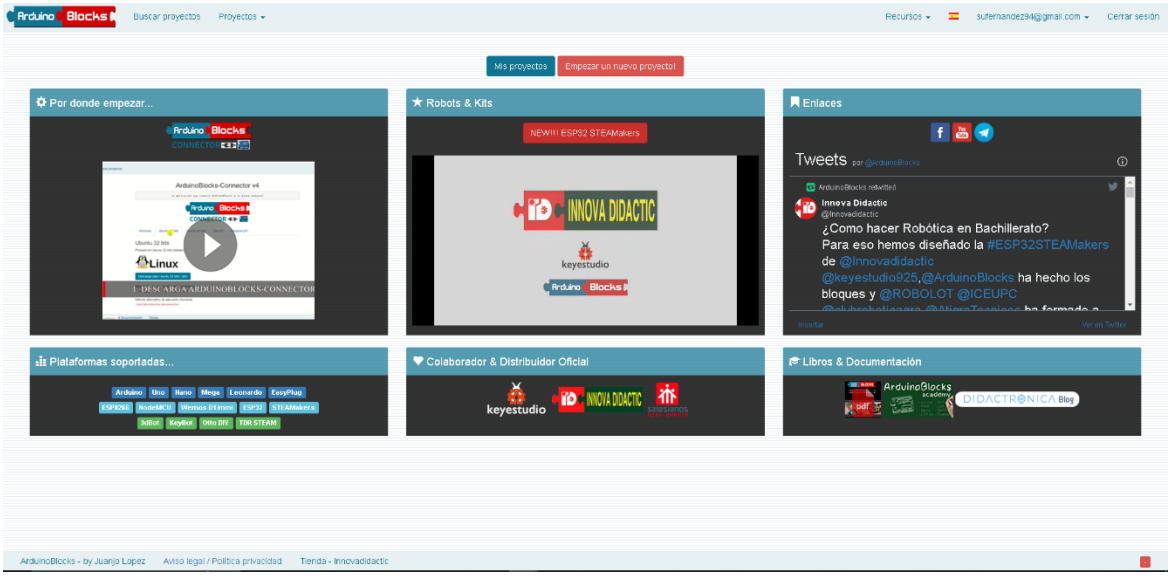


Figura 33 Página principal de arduinoblocks

Fuente: Elaboración propia

Al iniciar sesión tendremos acceso a nuestros proyectos en la que podemos seguir editando y comenzar a crear nuevos. Al hacer clic en nuevos proyectos nos dará tres opciones de proyectos, crear proyectos personales en la que solo es de acceso privado. Proyecto como profesor en la que pueden invitar a los estudiantes a participar en el proyecto compartido y la opción de alumno es donde se coloca el código que ha proporcionado en el proyecto como profesor para poder tener acceso.

Figura 34 Opciones de proyectos

Fuente: Elaboración propia

Para crear un nuevo proyecto personal se debe especificar el tipo de proyecto en donde se coloca la placa con la que vamos a trabajar. En este caso será la placa

ESP32/WROOM. Además de proporcionar una breve descripción y los componentes que utilizaremos.

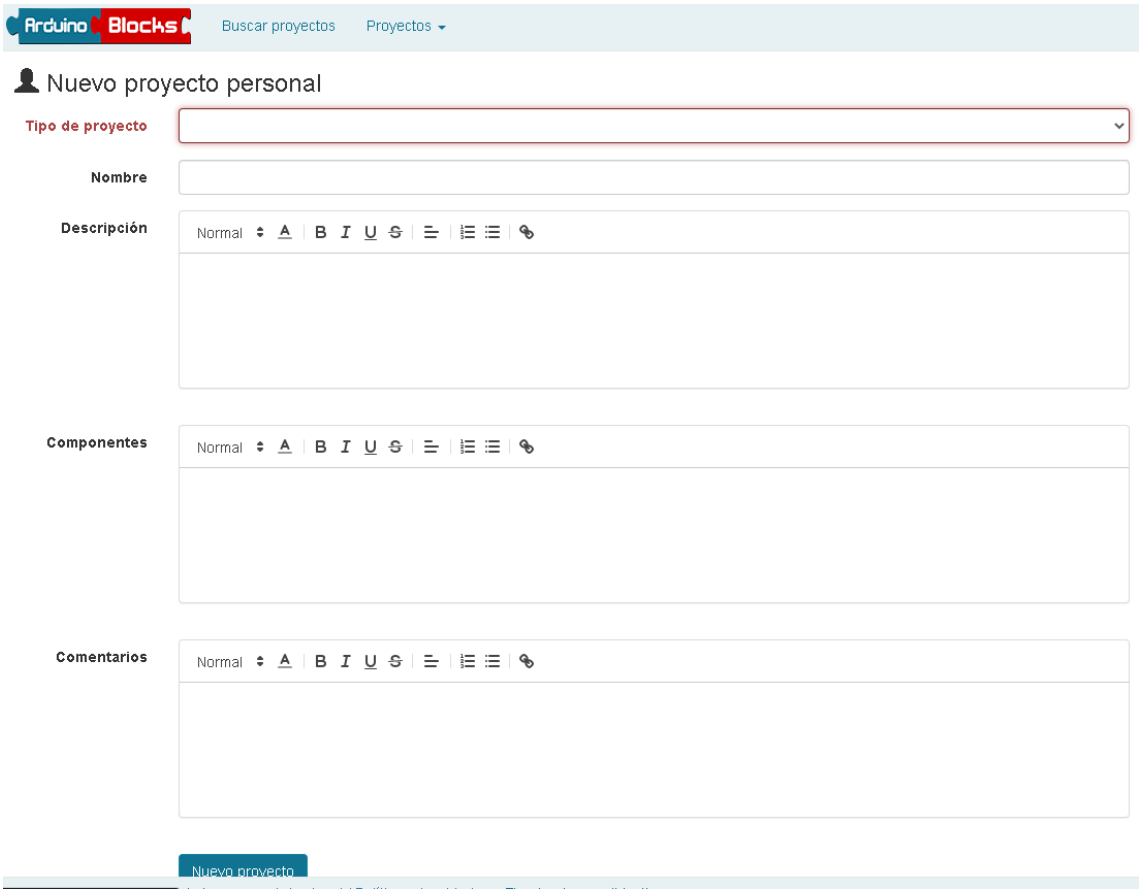


Figura 35 Crear nuevo proyecto

Fuente: Elaboración propia

A continuación se presentará la página de trabajo de Arduinoblocks en donde se mostrará las secciones de trabajo que posee.

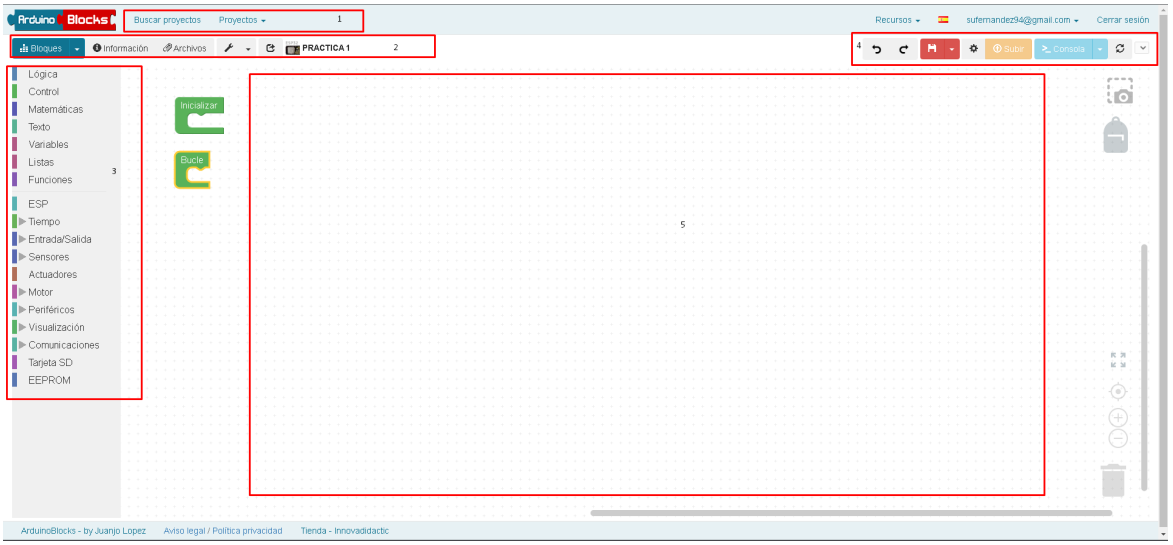


Figura 36 Ventana principal de ArduinoBlocks

Fuente: Elaboración propia

En la sección 1 tenemos las opciones de buscar proyectos en donde saldrán una lista de ejemplos a las que tendremos acceso con solo dar clic en cada uno de ellos. En la opción de proyectos podremos visualizar nuestra lista de proyectos, crear un nuevo proyecto e importar un archivo desde nuestro computador.

La sección 2 tendremos el menú de bloques donde podemos ver el código que se va generando en lenguaje textual de Arduino a partir de nuestros diagrama de bloques, tenemos la opción de descargar el código que se descarga con la extensión .ino para poder utilizar en el Arduino IDE y descargar todo en un archivo zip.

En la sección 3 encontramos los bloques con los que nos permite trabajar el Arduinoblocks, al igual que codeSkool este separa cada uno de los categorías por colores. Como se puede observar Arduinoblocks trabaja con las funciones predeterminadas por la que no se pueden agregar nuevas funciones.

La sección 4 es el área de trabajo donde arrastraremos los bloques a utilizar para la creación de las prácticas con las placas.

Y por último la sección 5 tiene las opciones de guardar el proyecto, generar un nuevo proyecto como alumno y exportar el archivo que estamos trabajando. El botón de subir permite cargar el programa a la placa y nos permite visualizar los datos por la consola. Se debe seleccionar el puerto donde está conectado la placa.

Para poder empezar a trabajar con el Arduinoblocks se debe descargar una extensión que permite que la página detecte la placa y el puerto donde se encuentra conectada. Al dirigirse al menú de recursos podemos encontrar la opción para descargar ArduinoBlocks Connector.

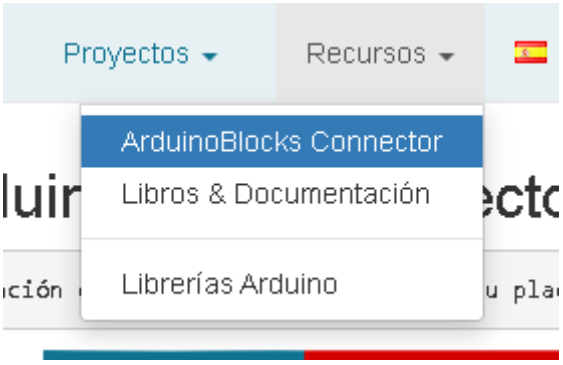


Figura 37 Descarga de ArduinoBlocks-Connector

Fuente: Elaboración propia

Seleccionamos la opción de descarga para nuestro Windows y una vez descargada procedemos a instalarla en nuestro ordenador. El proceso de instalación tarda alrededor de unos 3 a 4 minutos.

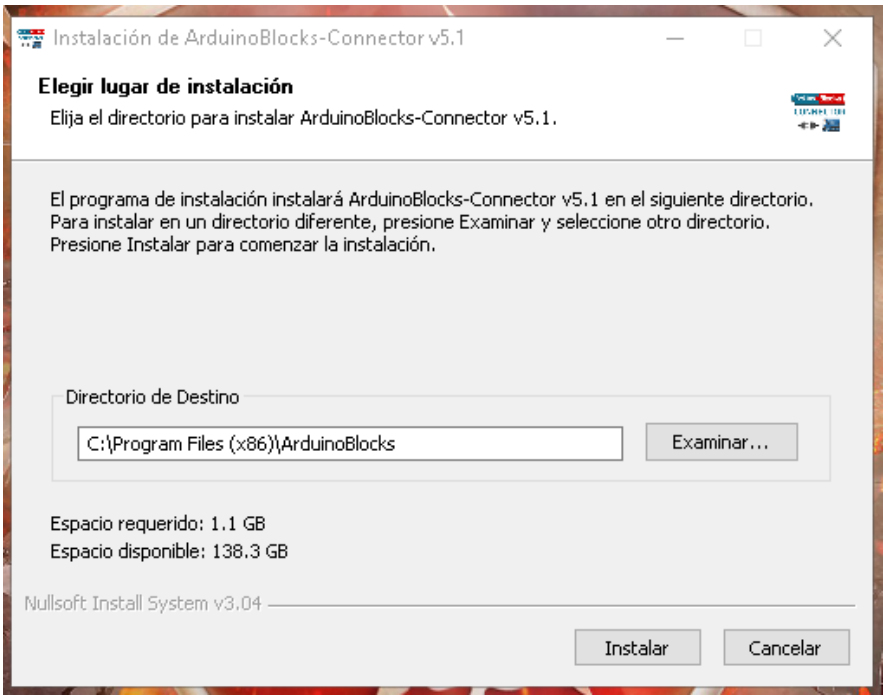


Figura 38 Instalación de ArduinoBlocks-Connector

Fuente: Elaboración propia

Al terminar la instalación damos clic en cerrar y procedemos a ejecutar, se mostrará la pantalla de ejecución.

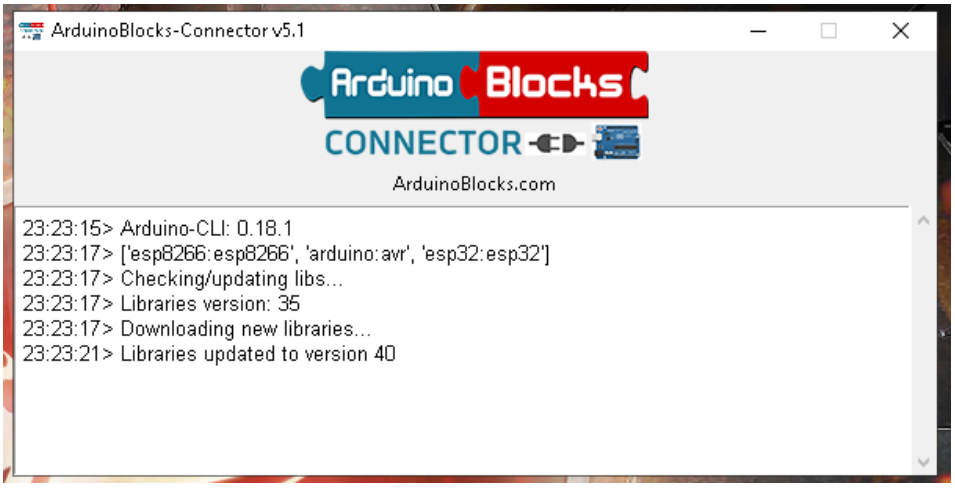


Figura 39 ArduinoBlocks-Connector

Fuente: Elaboración propia

Al dirigirnos a la página del área de trabajo de nuestro proyecto podemos observar la placa y la configuración de ArduinoBlocks.

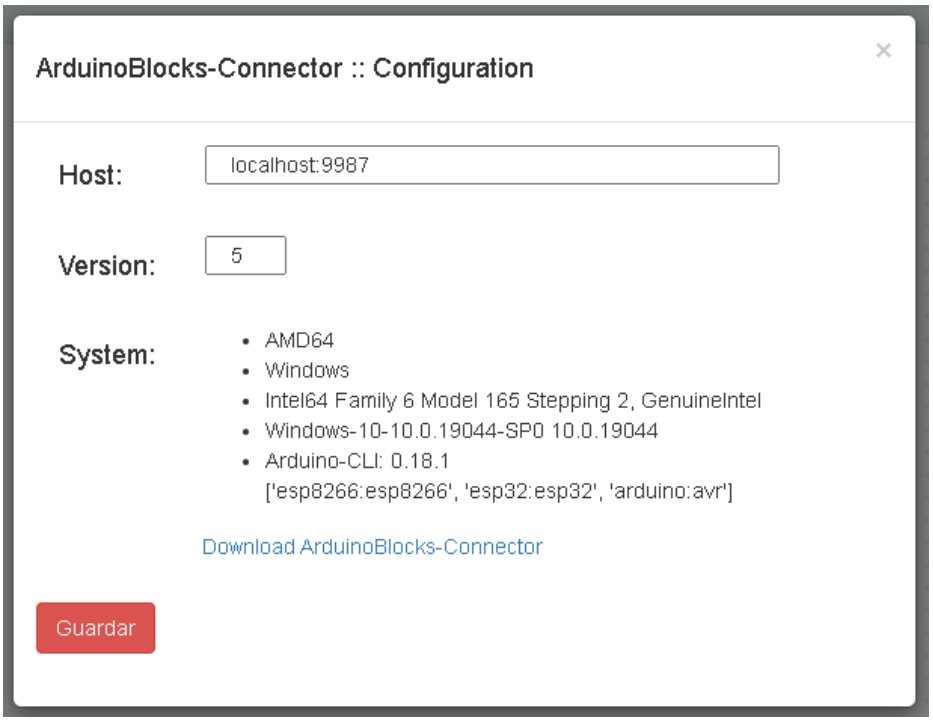


Figura 40 Configuración de ArduinoBlocks-Connector

Fuente: Elaboración propia

Empezamos con la práctica 1 encendido de leds, la placa debe estar en modo reset con el led azul encendido y al dar clic en subir se cargará el programa a la placa, al terminar el proceso se mostrará un mensaje si se ha cargado correctamente.

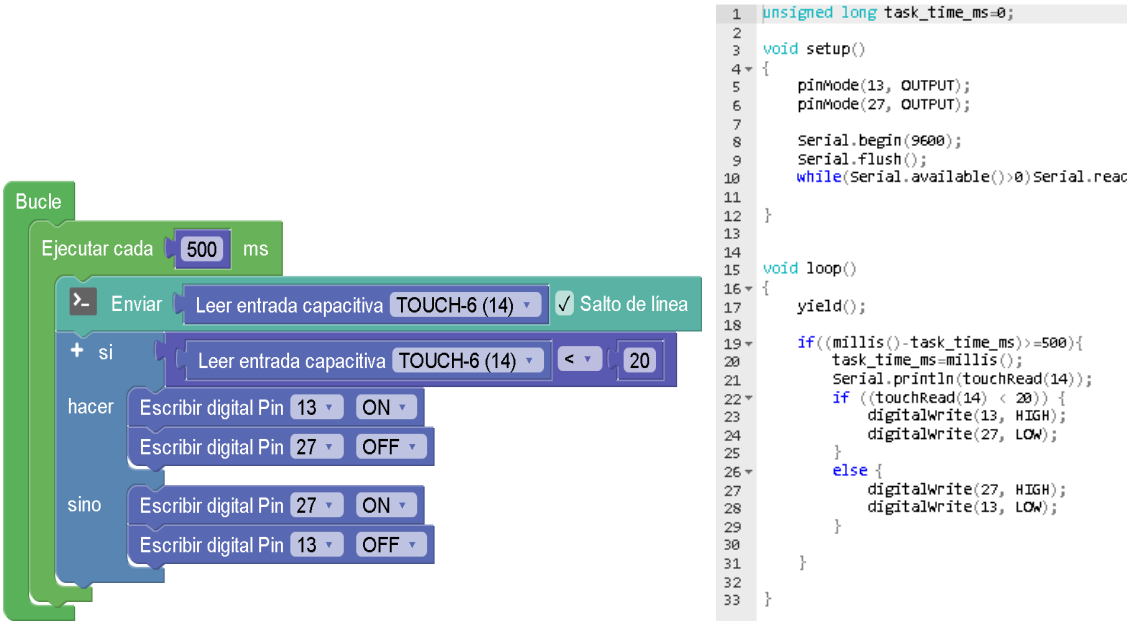


Figura 41 Código de práctica 1 ESP32 y ArduinoBlocks

Fuente: Elaboración propia

Es importante tener seleccionado el puerto correcto donde se encuentra seleccionado la placa, para verificar el puerto se puede dirigir a **administrador de dispositivos**, en la opción de **puertos** podrá visualizar el puerto de la placa.

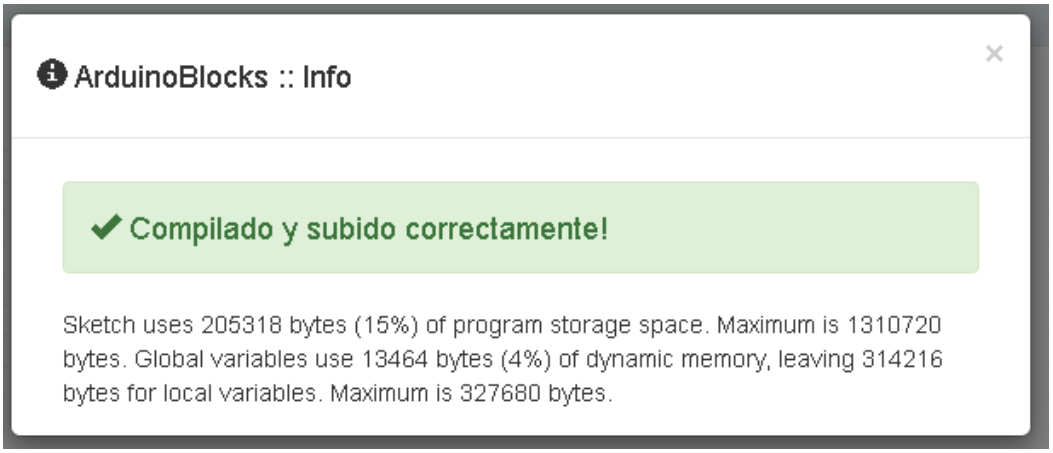


Figura 42 Código cargado correctamente

Fuente: Elaboración propia

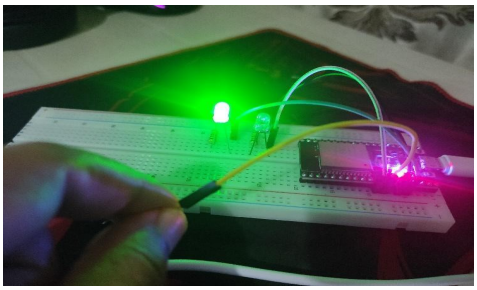
En Arduinoblocks podemos mostrar datos a través de la consola por medio de la línea enviar nos mostrará el valor de la entrada capacitiva que está captando el pin 14.

Figura 43 Comunicación serial

Fuente: Elaboración propia

2.4.1.1.3 Ejecución Práctica 1

Al estar en contacto el pin 14 aumenta la latencia por lo que el led del pin 13 se enciende y al tener un valor bajo este led se apaga, permanecerá encendido el led del pin 27 cuando los valores sean menores a 20. Como se puede observar en la consola de la ilustración 43 los valores que recibe cuando entra en contacto el pin 14.



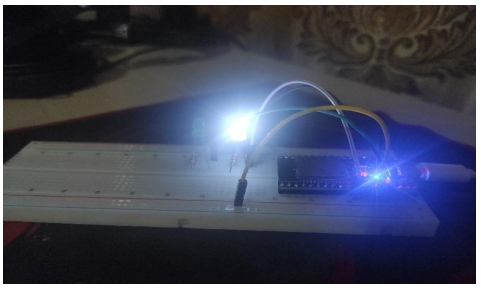


Figura 44 Ejecución de la práctica 1
Fuente: Elaboración propia

2.4.1.2 PRÁCTICA 2: LCD y DTH22

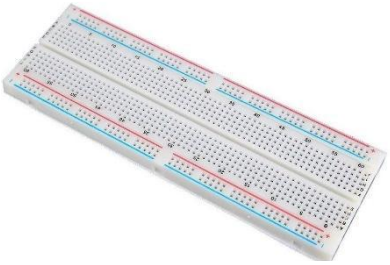
Objetivos: Leer la temperatura y humedad del sensor DTH22 y mostrar los resultados a través de la pantalla LCD.

Descripción: Para la realización de la práctica se utiliza el sensor DTH22 y la pantalla LCD.

- A través del sensor DTH22 se mostrará la temperatura y humedad que permite medir el aire alrededor del sensor y arrojará los datos mediante el pin digital que posee los cuales serán mostrados en la LCD.

Tabla 7 Materiales para Práctica 2

DESCRIPCIÓN	PRESENTACIÓN
ESP 32	
Cable USB	

Cables puente para protoboard	
DHT22	
Resistencia 220 Ohm	
Protoboard	
LCD con módulo I2C	

Fuente: Elaboración propia

En la tabla número 7 se establecen los materiales a utilizar para la práctica 2. En la figura 45 se muestra el diagrama de conexiones de la ESP32 con la pantalla LCD y el sensor DTH22.

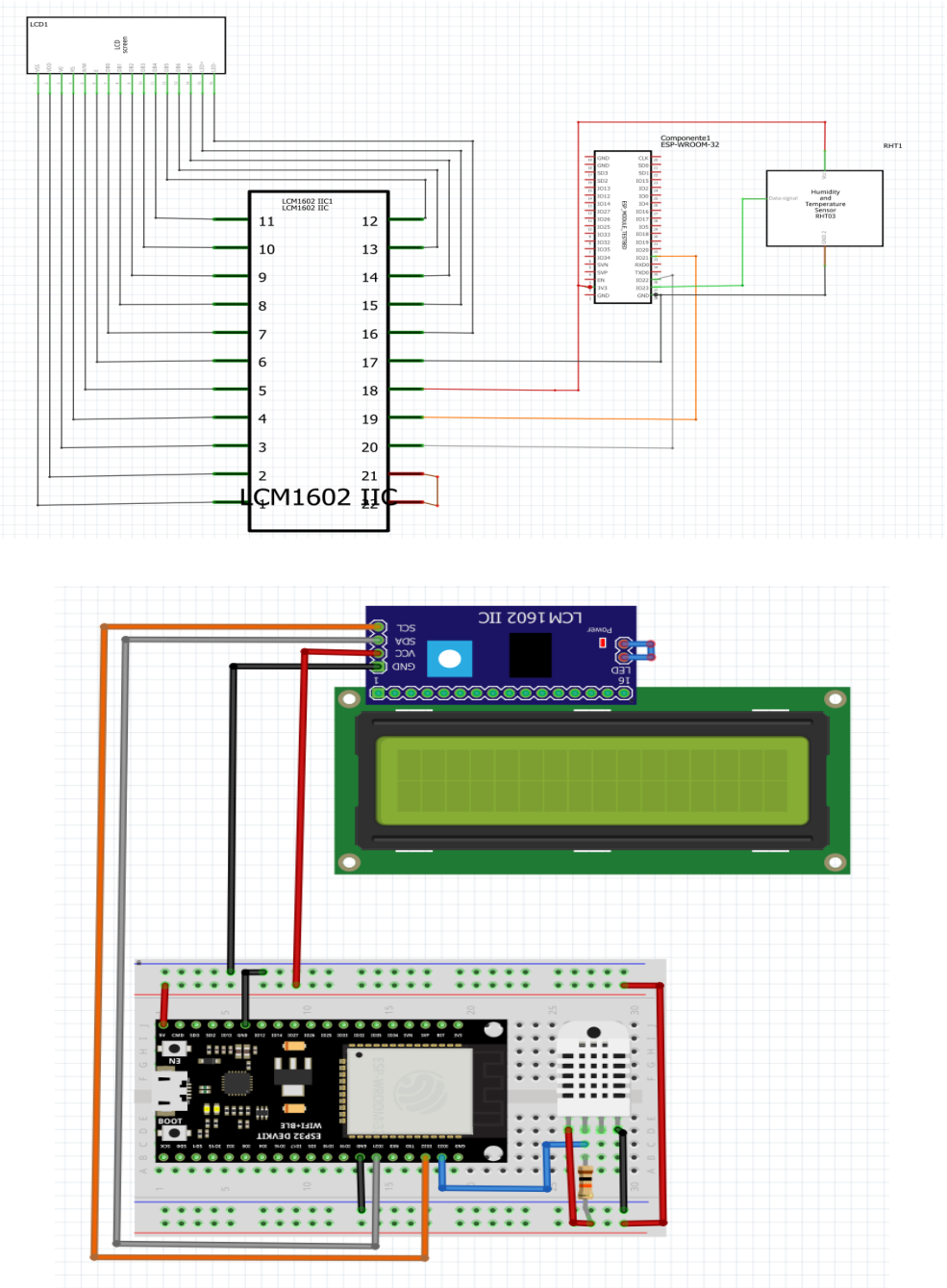


Figura 45 Diagrama de conexiones práctica 2

Fuente: Elaboración propia

Conexión de los componentes a la placa ESP32

La lectura de la señal digital del DTH22 será a través del pin 23.

La pantalla LCD el pin de datos SDA de la LCD estará conectada al pin 21 de la ESP32 y el pin 22 conectado al pin de señal del reloj SCL.

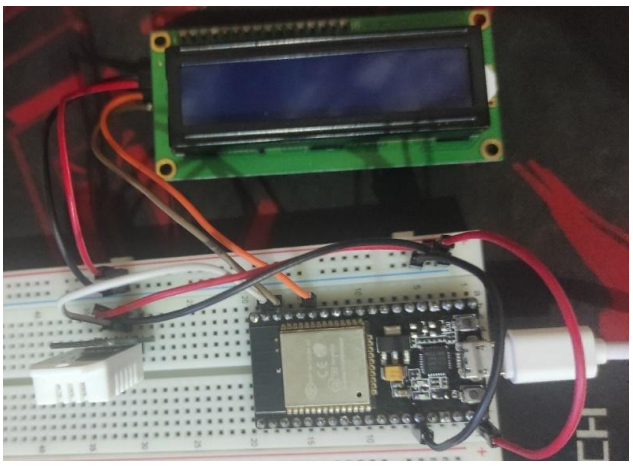


Figura 46 Conexión de la Práctica 2 en la protoboard

Fuente: Elaboración propia

2.4.1.2.1 CodeSkool

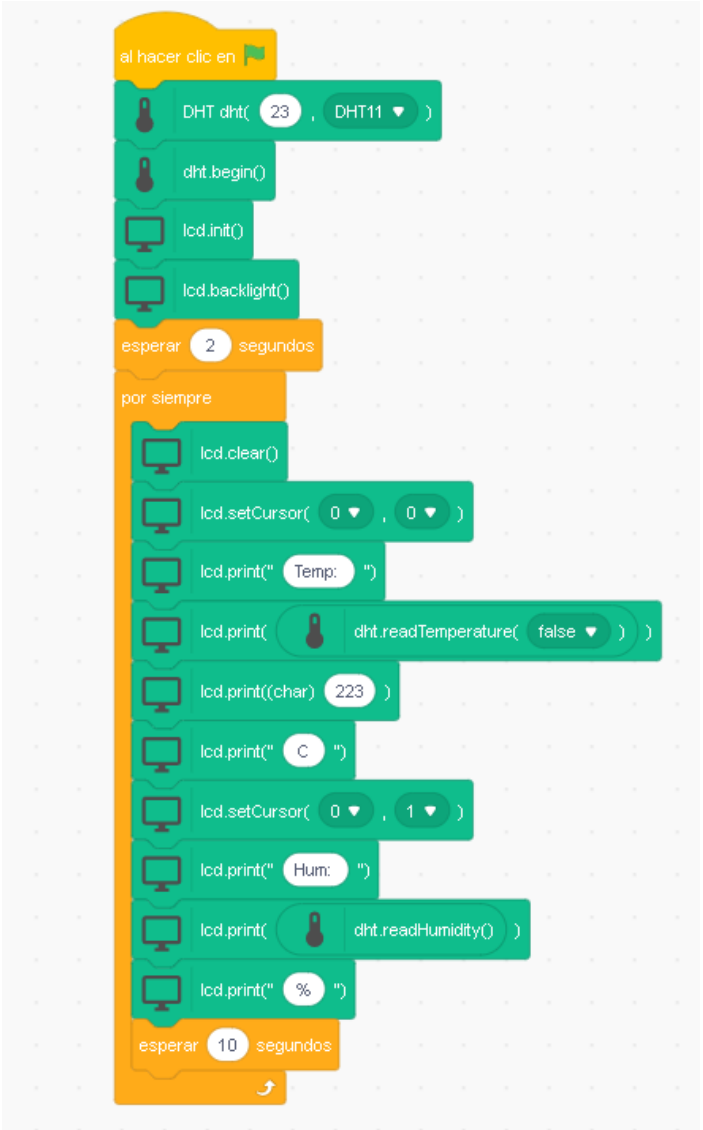


Figura 47 Código de Practica 2 CodeSkool
Fuente: Elaboración propia

2.4.1.2.2 ArduinoBlocks

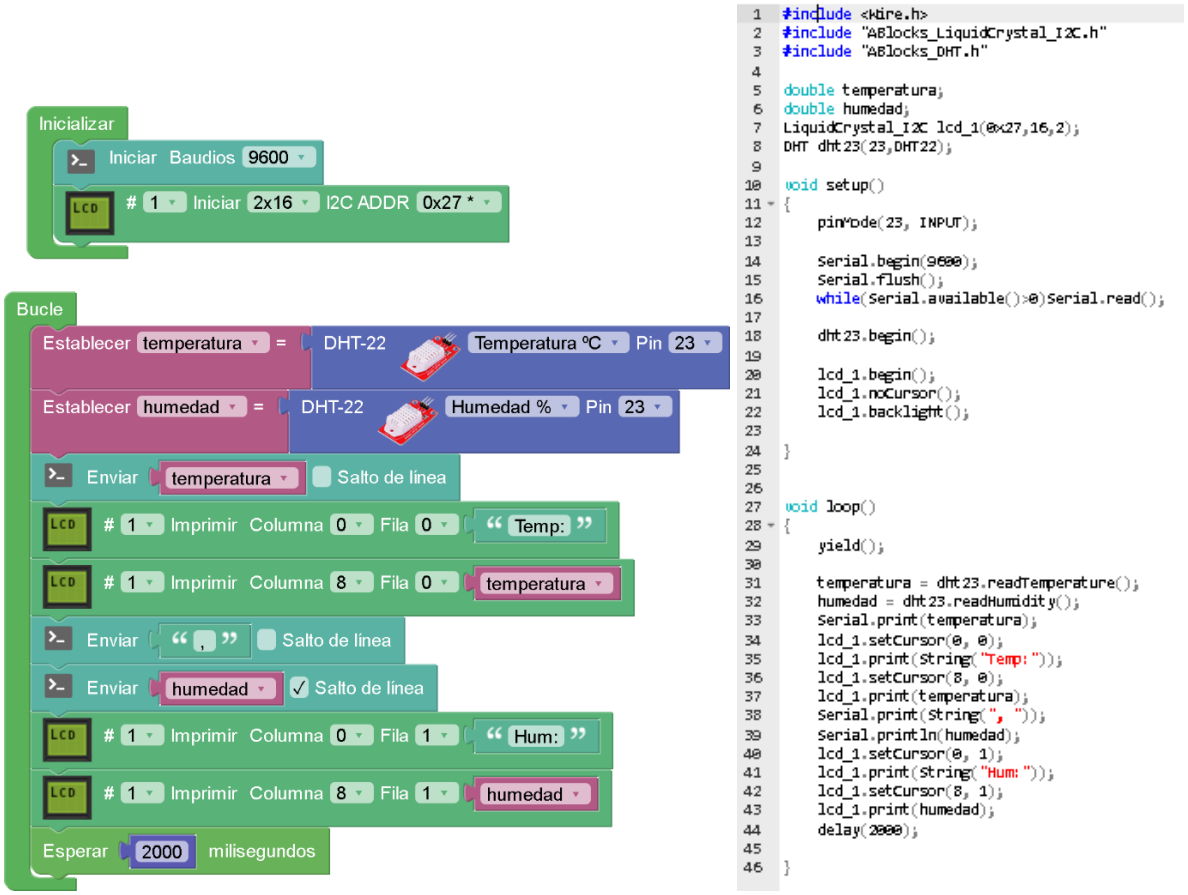


Figura 48 Código de Práctica 2 ArduinoBlocks

Fuente: Elaboración propia

2.4.1.2.3 Ejecución Práctica 2

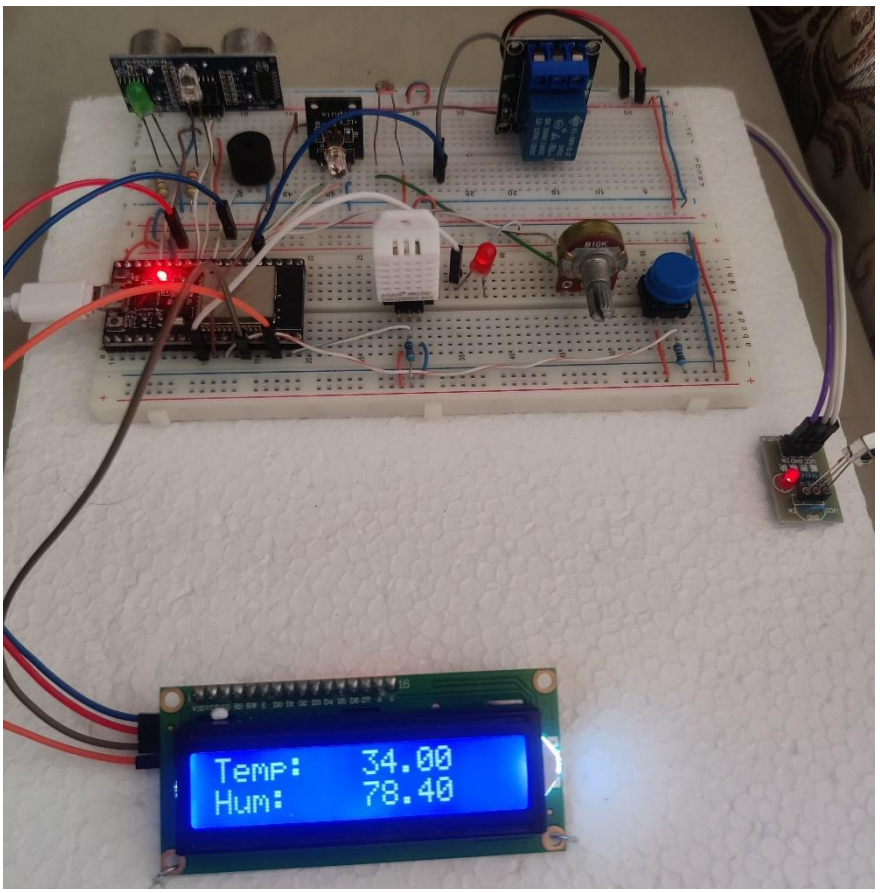


Figura 49 Ejecución práctica 2

Fuente: Elaboración propia

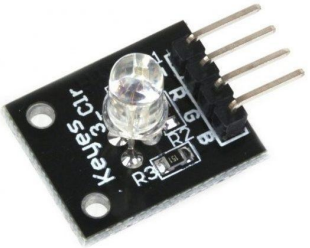
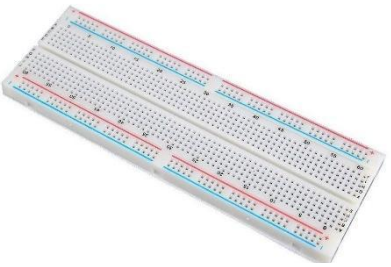
2.4.1.3 PRÁCTICA 3: SENSOR ULTRASÓNICO – RGB – BUZZER

Objetivos: Crear una alarma de detección de objetos mediante el uso del sensor ultrasónico.

Descripción: Para la realización de la práctica se utiliza el sensor ultrasónico, led RGB y el buzzer.

- El sensor ultrasónico mide la proximidad con un rango mínimo de 2 cm a 400 cm.
- Al detectar un objeto en el sensor se ejecutan los bloques de encendido del led RGB y el buzzer emite un sonido para alertar del movimiento.

Tabla 8 Materiales para Práctica 3

DESCRIPCIÓN	PRESENTACIÓN
ESP 32	
Cable USB	
Cables puente para protoboard	
Modulo LED RGB	
Protoboard	

Buzzer	
Sensor ultrasónico HC-SR04	

Fuente: Elaboración propia

En la tabla 8 se pueden observar los materiales a utilizar para la práctica con la placa EPS32, a continuación se muestra el diagrama de conexiones. Para la elaboración del diagrama de conexiones se utilizó la herramienta fritzing.

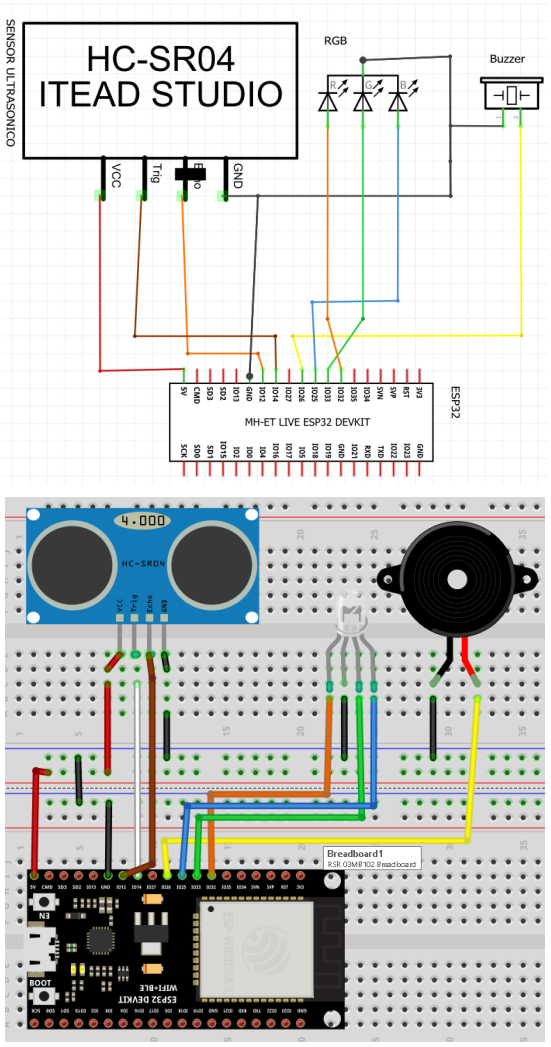


Figura 50 Diagrama de conexión practica 3

Fuente: Elaboración propia

Conexión de los componentes a la placa ESP32

Para la conexión del sensor ultrasónico se utilizará el Pin 14 de la placa que estará conectado al Pin Trigger y el Pin 12 conectado al Pin Echo del sensor.

El led RGB el PIN R estará conectado al PIN 32 de la placa, el pin G estará conectado al Pin 33 y el pin B conectado al Pin 25.

El buzzer positivo del buzzer estará conectado al Pin 26 de la placa.

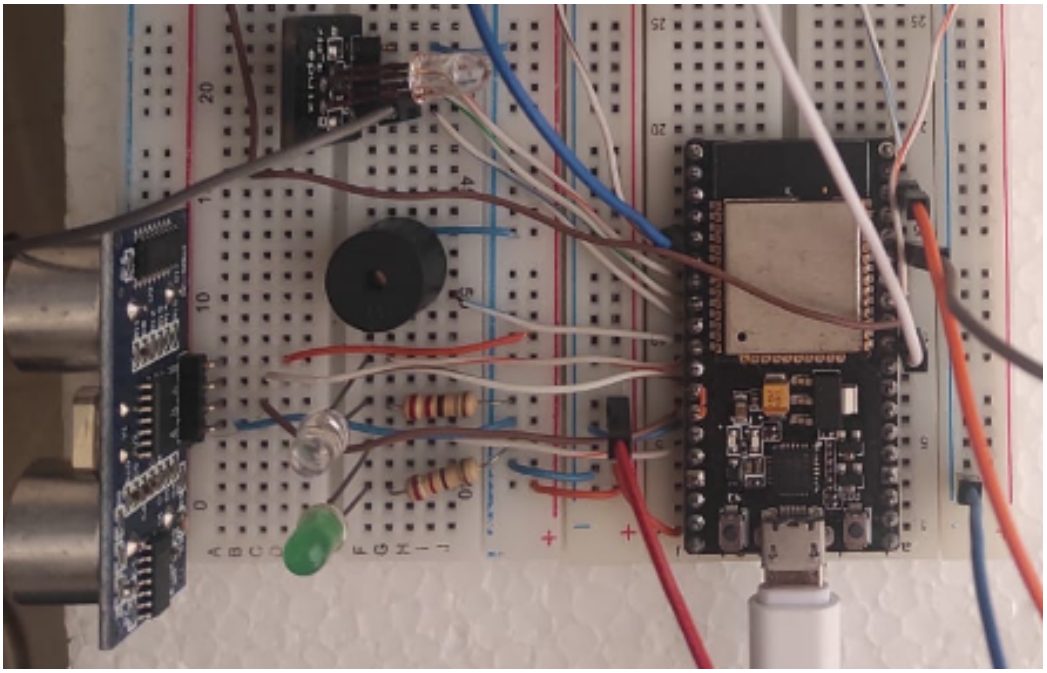


Figura 51 Conexión de la Práctica 3 en la protoboard

Fuente: Elaboración propia

2.4.1.3.1 CodeSkool

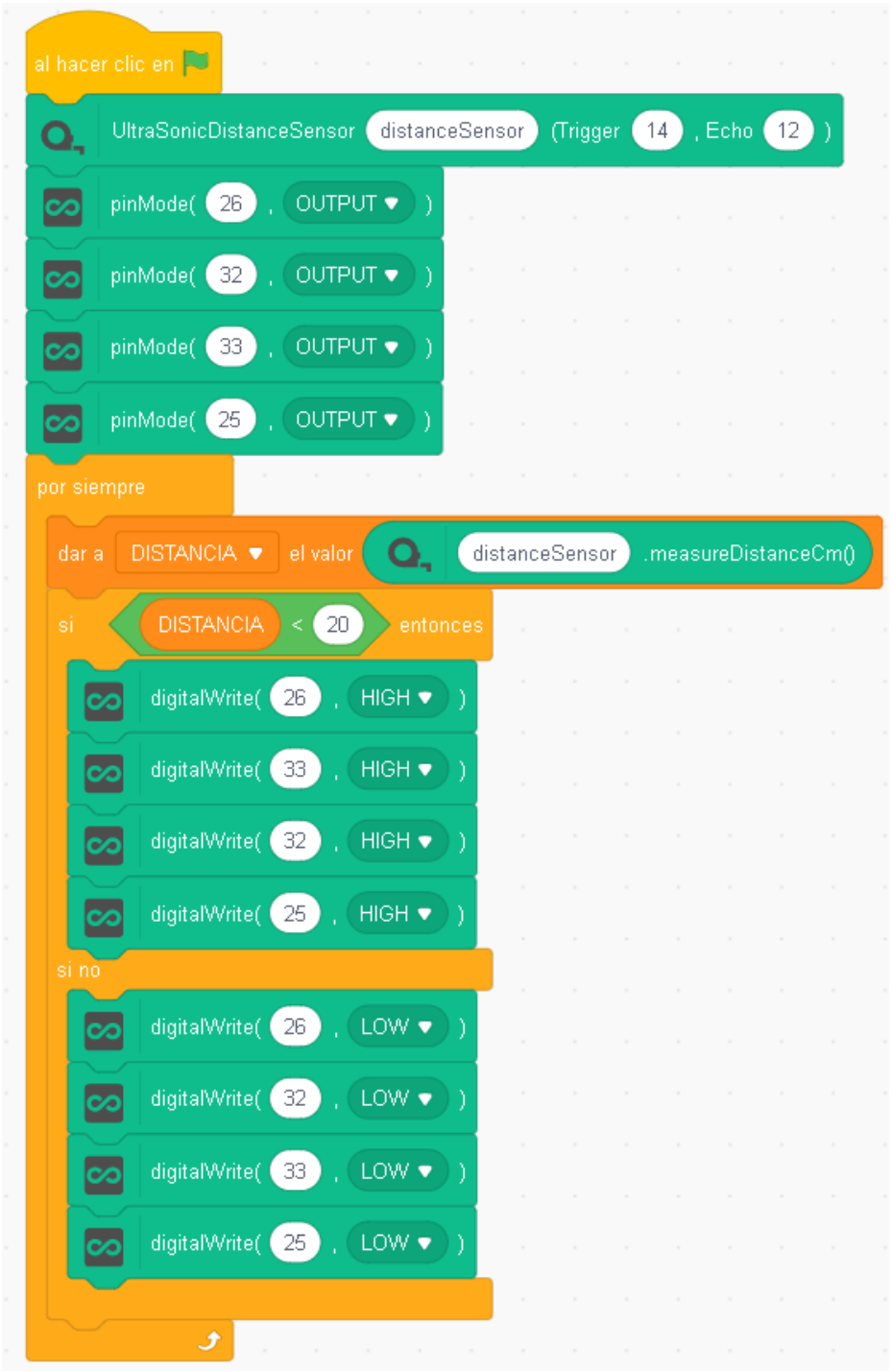
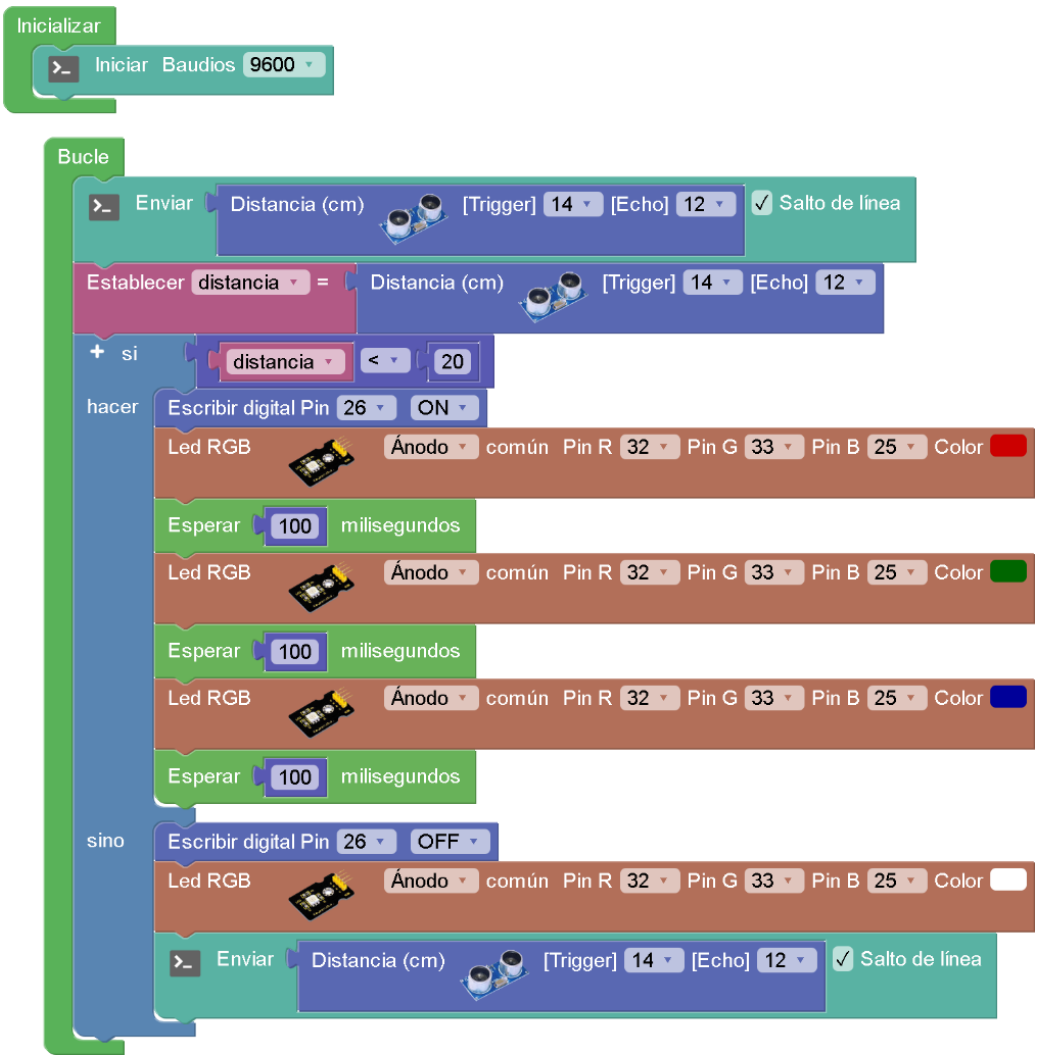


Figura 52 Código Practica 3 CodeSkool

Fuente: Elaboración propia

2.4.1.3.2 ArduinoBlocks




```ino
#include <analogWrite.h>

double distancia;

double fnc_ultrasonic_distance(int _t, int _e){
    unsigned long dur=0;
    digitalWrite(_t, LOW);
    delayMicroseconds(5);
    digitalWrite(_t, HIGH);
    delayMicroseconds(10);
    digitalWrite(_t, LOW);
    dur = pulseIn(_e, HIGH, 10000);
    if(dur==0)return 999.0;
    return (dur/57);
}

void setup()
{
    pinMode(14, OUTPUT);
    pinMode(12, INPUT);
    pinMode(26, OUTPUT);
    pinMode(32, OUTPUT);
    pinMode(33, OUTPUT);
    pinMode(25, OUTPUT);

    Serial.begin(9600);
    Serial.flush();
    while(Serial.available()>0)Serial.read();

}

void loop()
{
    yield();

    Serial.println(fnc_ultrasonic_distance(14,12));
    distancia = fnc_ultrasonic_distance(14,12);
    if ((distancia < 20)) {
        digitalWrite(26, HIGH);
        analogWrite(32,255-(204));
        analogWrite(33,255-(0));
        analogWrite(25,255-(0));
        delay(100);
        analogWrite(32,255-(0));
        analogWrite(33,255-(102));
        analogWrite(25,255-(0));
        delay(100);
        analogWrite(32,255-(0));
        analogWrite(33,255-(0));
        analogWrite(25,255-(153));
        delay(100);
    }
    else {
        digitalWrite(26, LOW);
        analogWrite(32,255-(255));
        analogWrite(33,255-(255));
        analogWrite(25,255-(255));
        Serial.println(fnc_ultrasonic_distance(14,12));
    }
```

Figura 53 Código Practica 3 ArduinoBlocks

Fuente: Elaboración propia

2.4.1.3.3 Ejecución Práctica 3

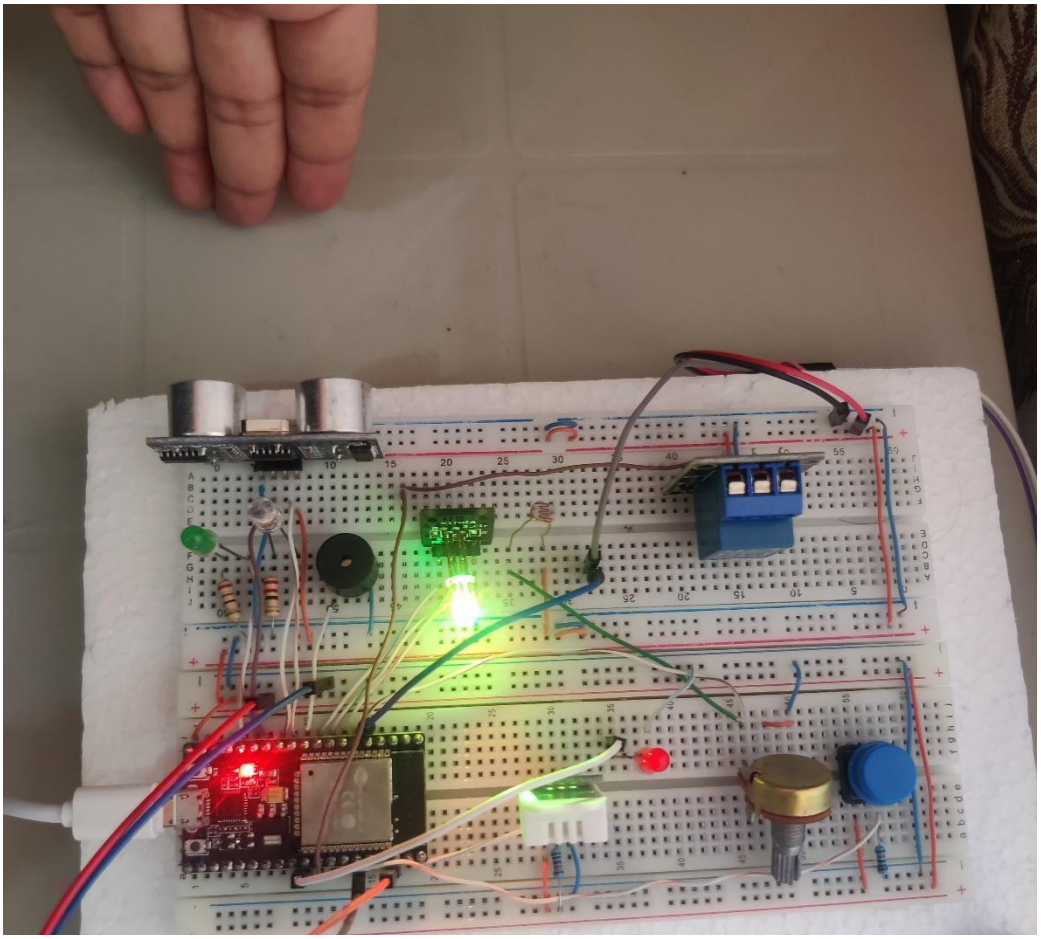


Figura 54 Ejecución práctica 3

Fuente: Elaboración propia

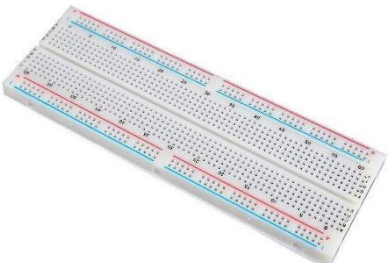
2.4.1.4 PRÁCTICA 4: SENSOR LDR

Objetivos: Encender o apagar un led de acuerdo al nivel de luz que incide sobre el sensor LDR.

Descripción: Para la realización de la práctica se utiliza el sensor LDR.

- El valor del sensor LDR varía según la luminosidad que percibe, es decir mientras más luz detecta menos resistencia eléctrica y al recibir menos luz tendrá mayor resistencia eléctrica.

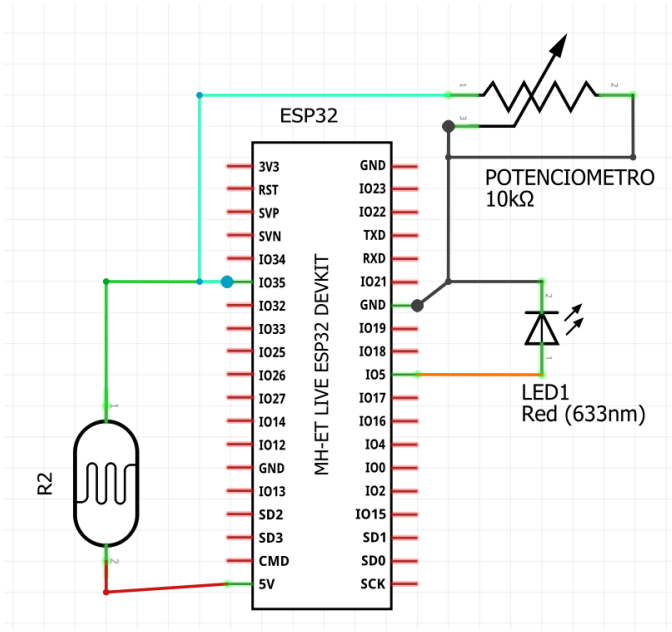
Tabla 9 Materiales para Práctica 4

DESCRIPCIÓN	PRESENTACIÓN
ESP 32	
Cable USB	
Cables puente para protoboard	
LED	
Protoboard	
Resistencia LDR	

Resistencia 10k	

Fuente: Elaboración propia

En la tabla 9 se muestran los materiales a utilizar para la práctica con la placa EPS32, a continuación se muestra el diagrama de conexiones. Para la elaboración del diagrama de conexiones se utilizó la herramienta fritzing.



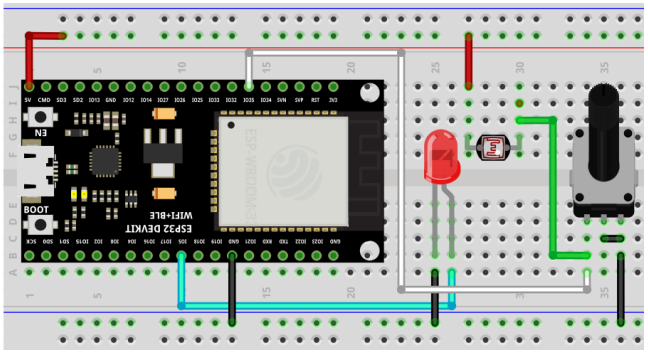


Figura 55 Diagrama de conexión practica 4

Fuente: Elaboración propia

Conexión de los componentes a la placa ESP32

El sensor LDR estará conectado una terminal a positivo y la otra irá al terminal 3 del potenciómetro.

El potenciómetro estará conectado el pin 1 y 2 a tierra y el terminal 3 al pin 35 de la placa.

El led estará conectado en el pin 5 de la placa.

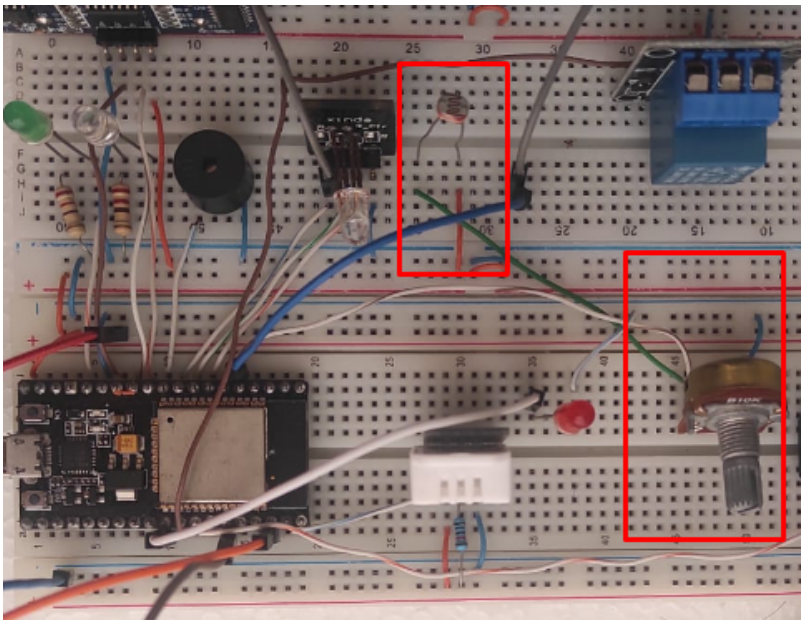


Figura 56 Conexión de la Práctica 4 en la protoboard

Fuente: Elaboración propia

2.4.1.4.1 CodeSkool

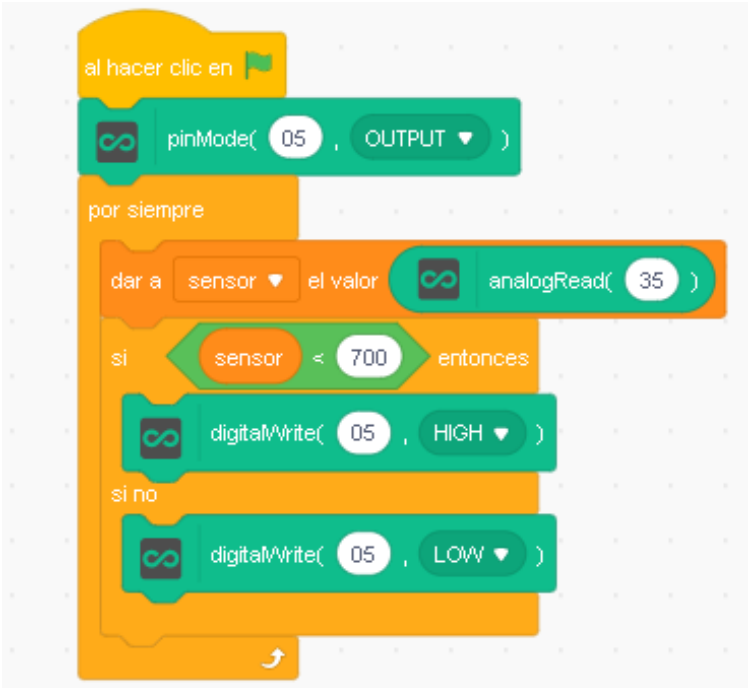
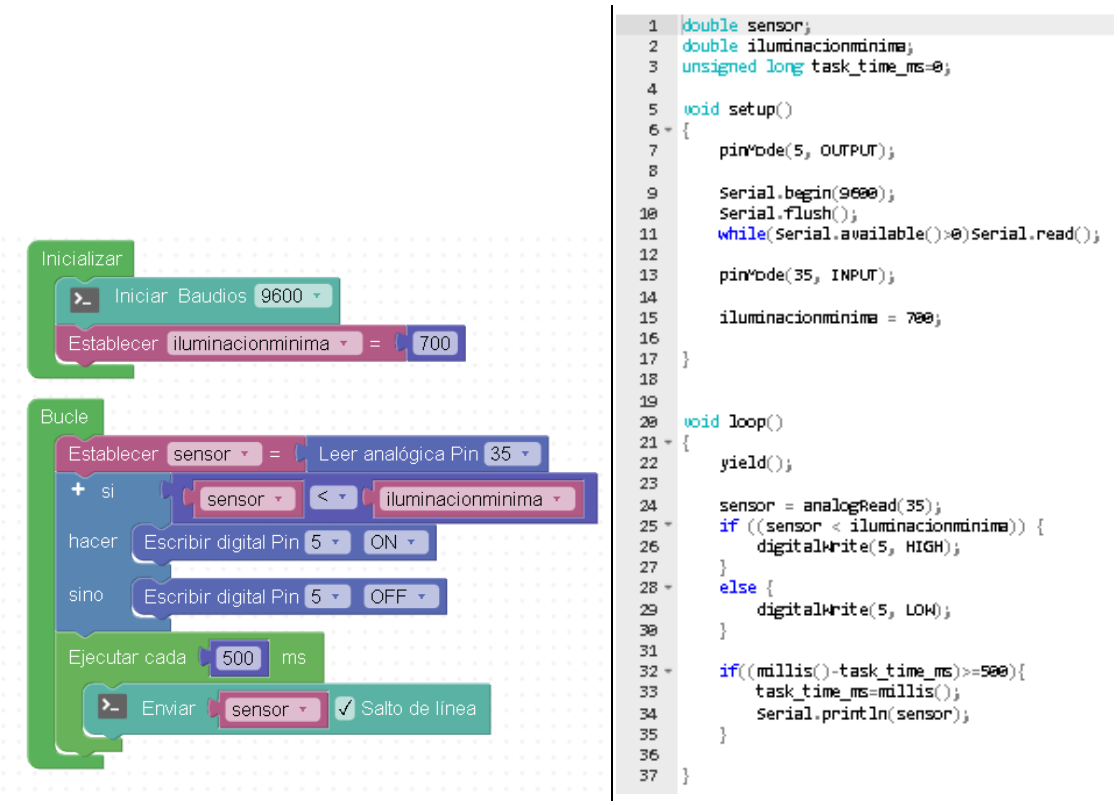


Figura 57 Código Practica 4 CodeSkool

Fuente: Elaboración propia

2.4.1.4.2 ArduinoBlocks




```
double sensor;
double iluminacionminima;
unsigned long task_time_ms=0;

void setup()
{
    pinMode(5, OUTPUT);

    Serial.begin(9600);
    Serial.flush();
    while(Serial.available()>0)Serial.read();

    pinMode(35, INPUT);

    iluminacionminima = 700;

}


void loop()
{
    yield();

    sensor = analogRead(35);
    if ((sensor < iluminacionminima)) {
        digitalWrite(5, HIGH);
    }
    else {
        digitalWrite(5, LOW);
    }

    if((millis()-task_time_ms)>=500){
        task_time_ms=millis();
        Serial.println(sensor);
    }

}
```

Figura 58 Código Practica 4 ArduinoBlocks

Fuente: Elaboración propia

2.4.1.4.3 Ejecución Práctica 4

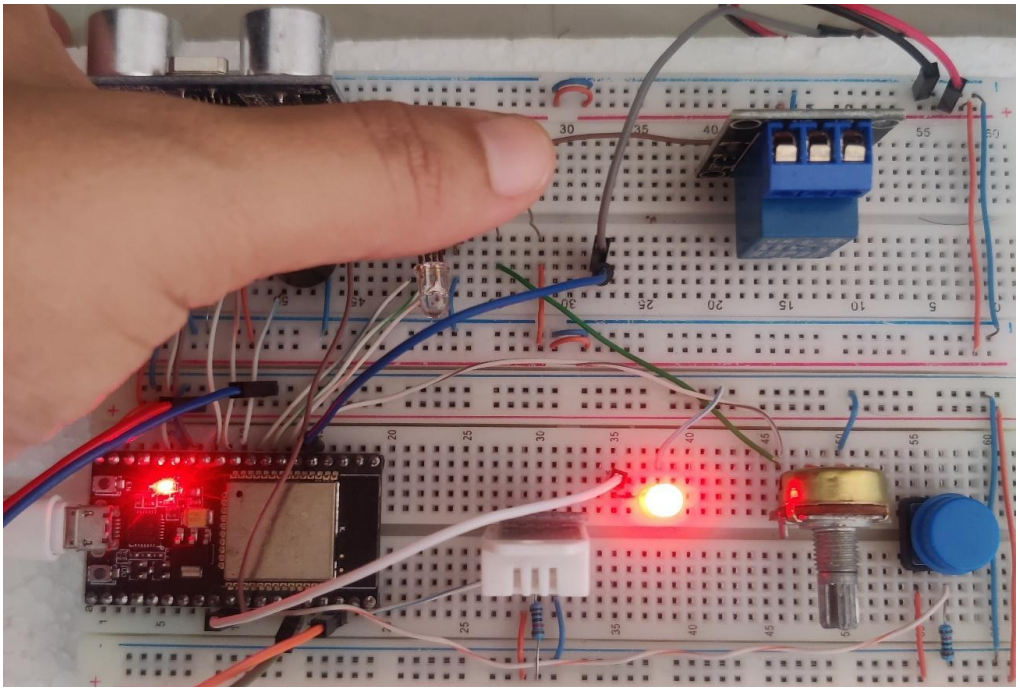


Figura 59 Ejecución práctica 4

Fuente: Elaboración propia

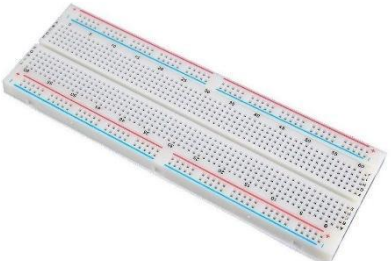
2.4.1.5 PRÁCTICA 5: RELÉ Y PULSADOR

Objetivos: Enviar un pulso y activar el circuito para encender un foco.

Descripción: Para la realización de la práctica se utiliza un pulsador y un relé.

- El relé posee una bobina que está conectada a una corriente, cuando esta se activa hace que el circuito que estaba abierto se cierre y pase la corriente para realizar el encendido del foco.

Tabla 10 Materiales para Práctica 5

DESCRIPCIÓN	PRESENTACIÓN
ESP 32	
Cable USB	
Cables puente para protoboard	
Protoboard	

Relé de 1 canal	
Pulsador	

Fuente: Elaboración propia

En la tabla 10 se muestran los materiales a utilizar para la práctica con la placa EPS32, a continuación se muestra el diagrama de conexiones. Para la elaboración del diagrama de conexiones se utilizó la herramienta fritzing.

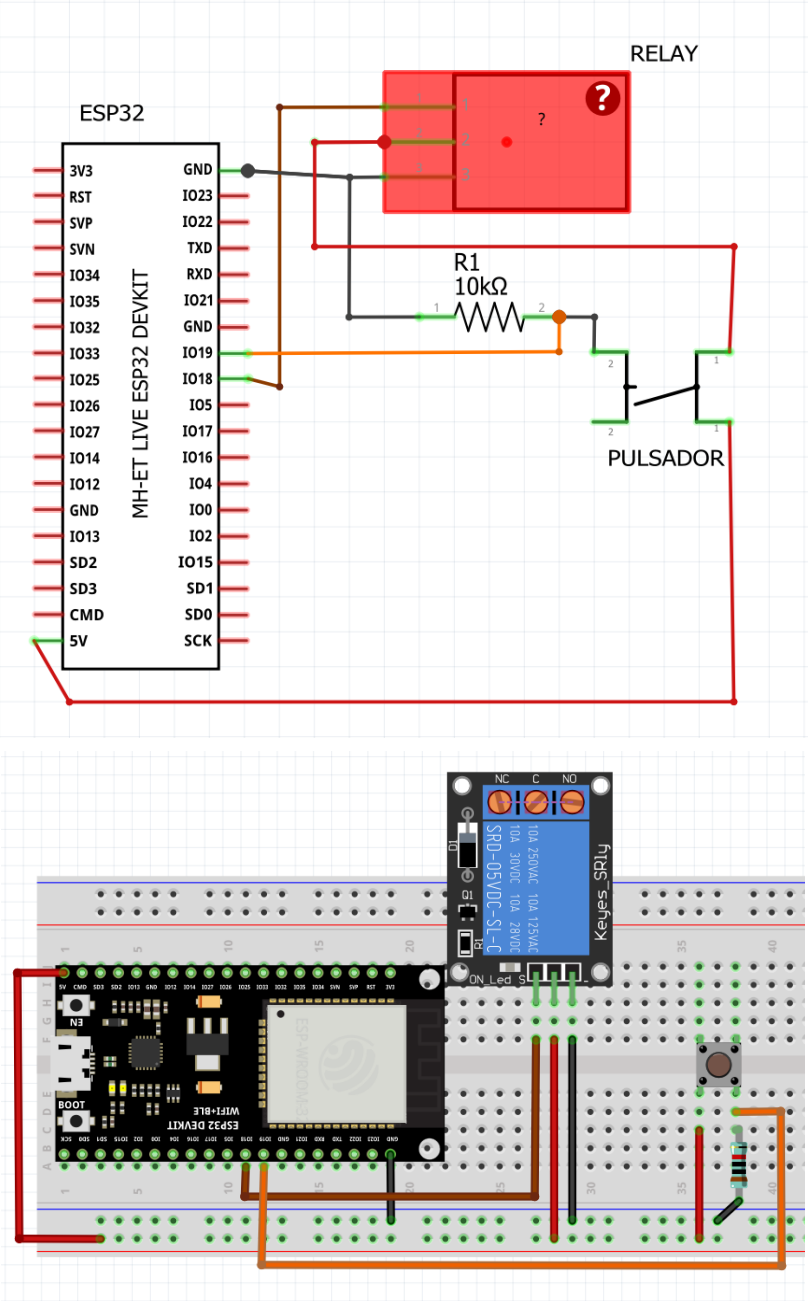


Figura 60 Diagrama de conexión practica 5

Fuente: Elaboración propia

Conexión de los componentes a la placa ESP32

El pulsador estará conectado al pin 19 de la placa ESP32 y su terminal 2 a tierra.

La señal de activación del pin de relé estará conectado al pin 18, y sus terminales a tierra y voltaje respectivamente.

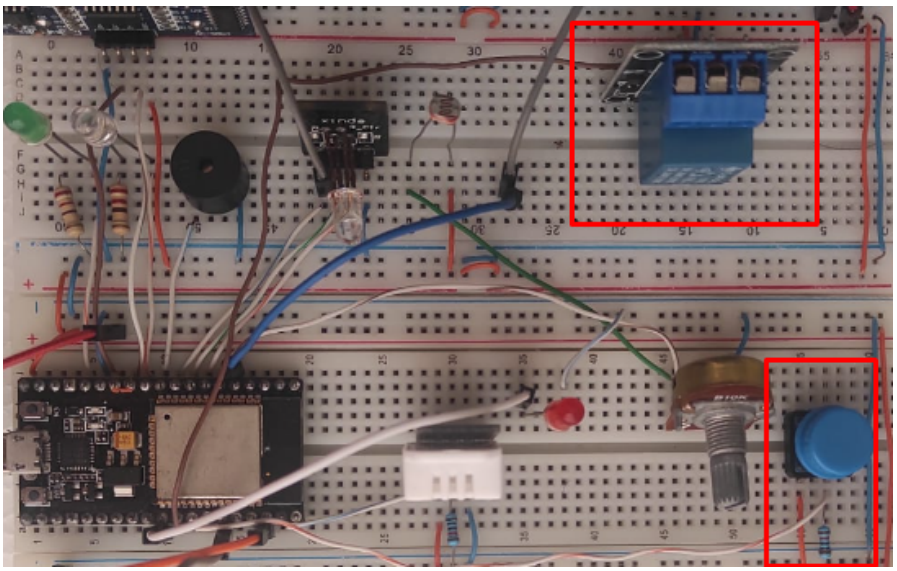


Figura 61 Conexión de la Práctica 5 en la protoboard

Fuente: Elaboración propia

2.4.1.5.1 CodeSkool

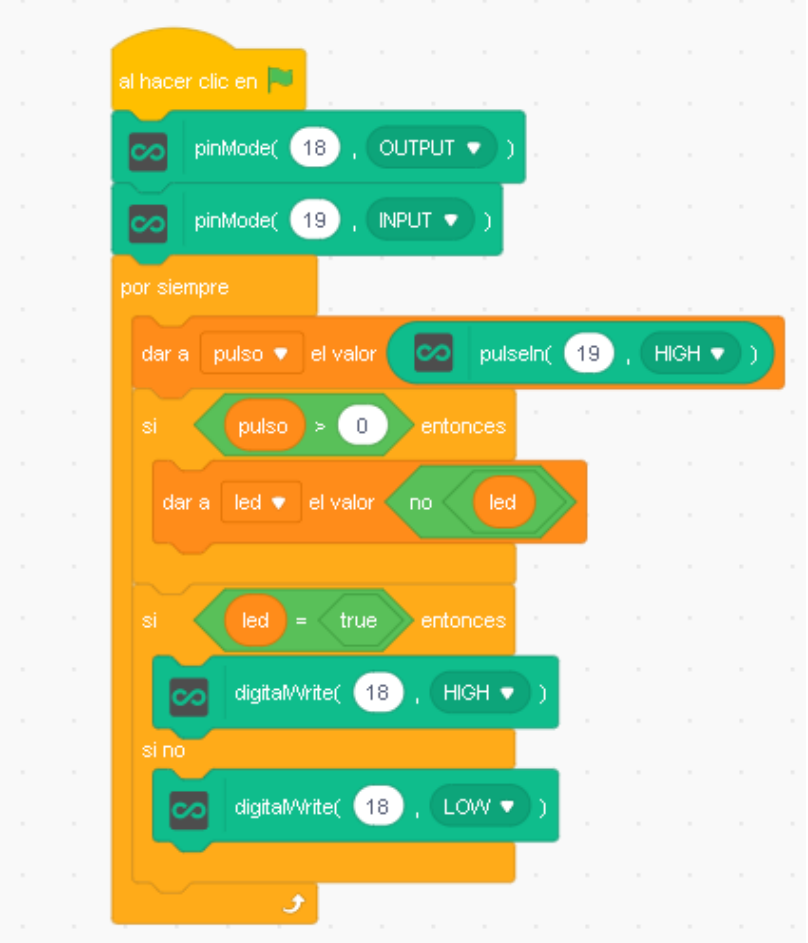
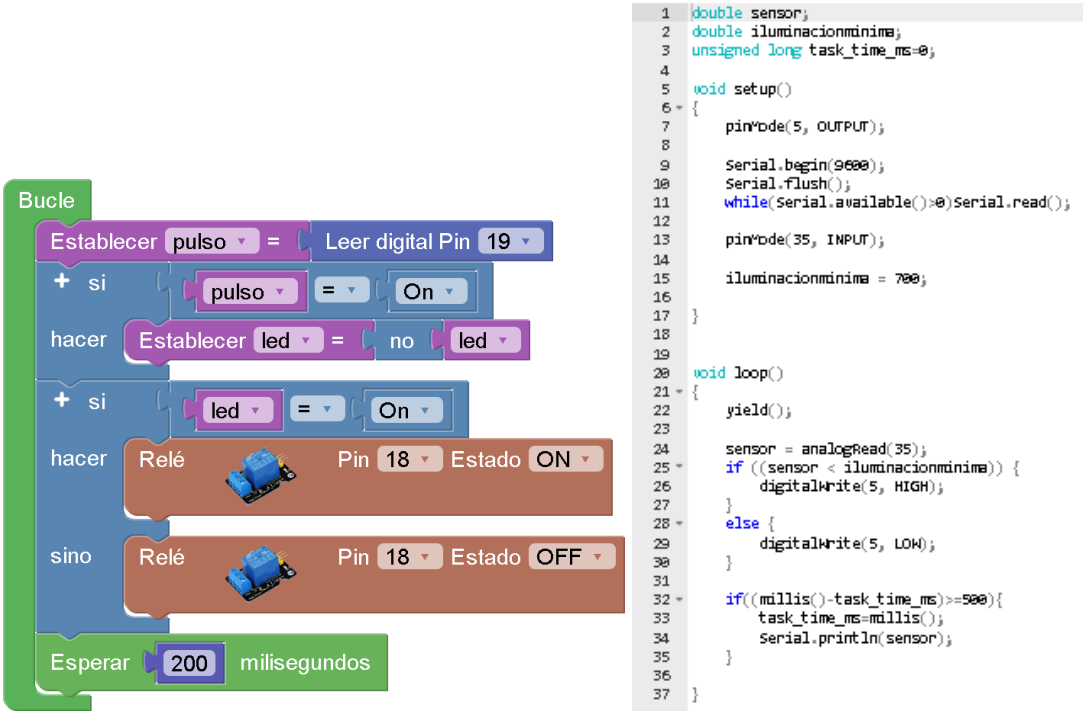


Figura 62 Código Práctica 5 CodeSkool

Fuente: Elaboración propia

2.4.1.5.2 ArduinoBlocks




```
double sensor;
double iluminacionminima;
unsigned long task_time_ms=0;

void setup()
{
    pinMode(5, OUTPUT);

    Serial.begin(9600);
    Serial.flush();
    while(Serial.available()>0)Serial.read();

    pinMode(35, INPUT);

    iluminacionminima = 700;

}


void loop()
{
    yield();

    sensor = analogRead(35);
    if ((sensor < iluminacionminima)) {
        digitalWrite(5, HIGH);
    }
    else {
        digitalWrite(5, LOW);
    }

    if((millis()-task_time_ms)>=500){
        task_time_ms=millis();
        Serial.println(sensor);
    }

}
```

Figura 63 Código Práctica 5 ArduinoBlocks

Fuente: Elaboración propia

2.4.1.5.3 Ejecución Práctica 5

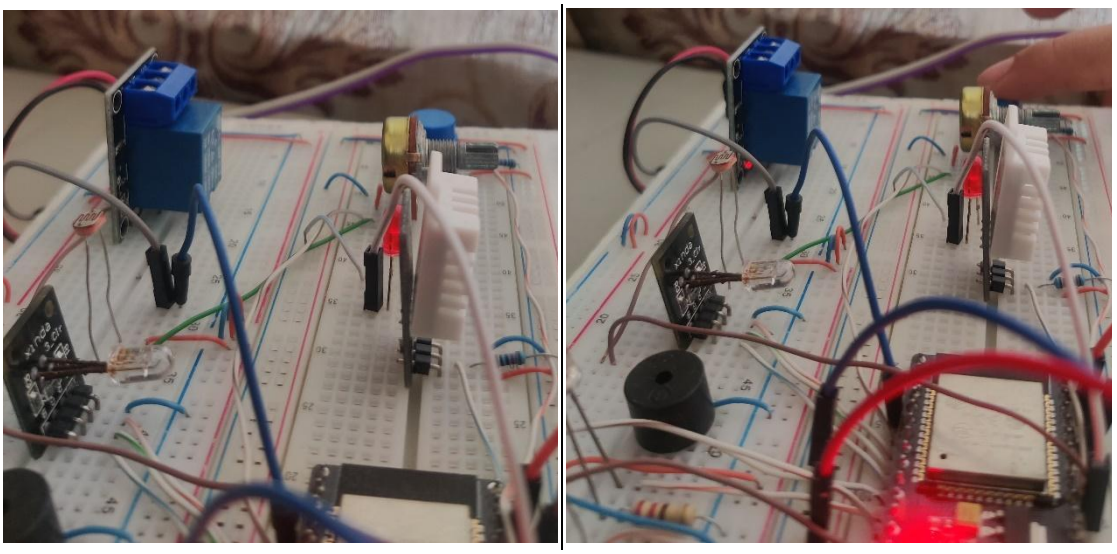


Figura 64 Ejecución práctica 5

Fuente: Elaboración propia

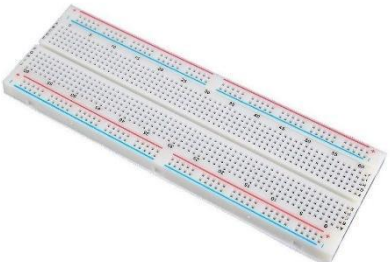
2.4.1.6 PRÁCTICA 6: RECEPTOR INFRARROJO

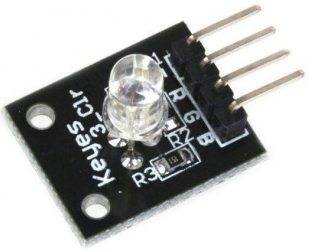
Objetivos: Receptar las señales del control remoto y encender las luces del led RGB de acuerdo al botón presionado.

Descripción: Para la realización de la práctica se utiliza el receptor ir y el control remoto.

- El Receptor IR permite detectar las señales del control remoto y señales infrarrojas, posee un fotodiodo que convierte la señal infrarroja en pulsos eléctricos.

Tabla 11 Materiales para Práctica 6

DESCRIPCIÓN	PRESENTACIÓN
ESP 32	
Cable USB	
Cables puente para protoboard	
Protoboard	
Receptor infrarrojo IR	 

Led RGB	

Fuente: Elaboración propia

En la tabla 11 se muestran los materiales a utilizar para la práctica con la placa EPS32, a continuación se muestra el diagrama de conexiones. Para la elaboración del diagrama de conexiones se utilizó la herramienta fritzing.

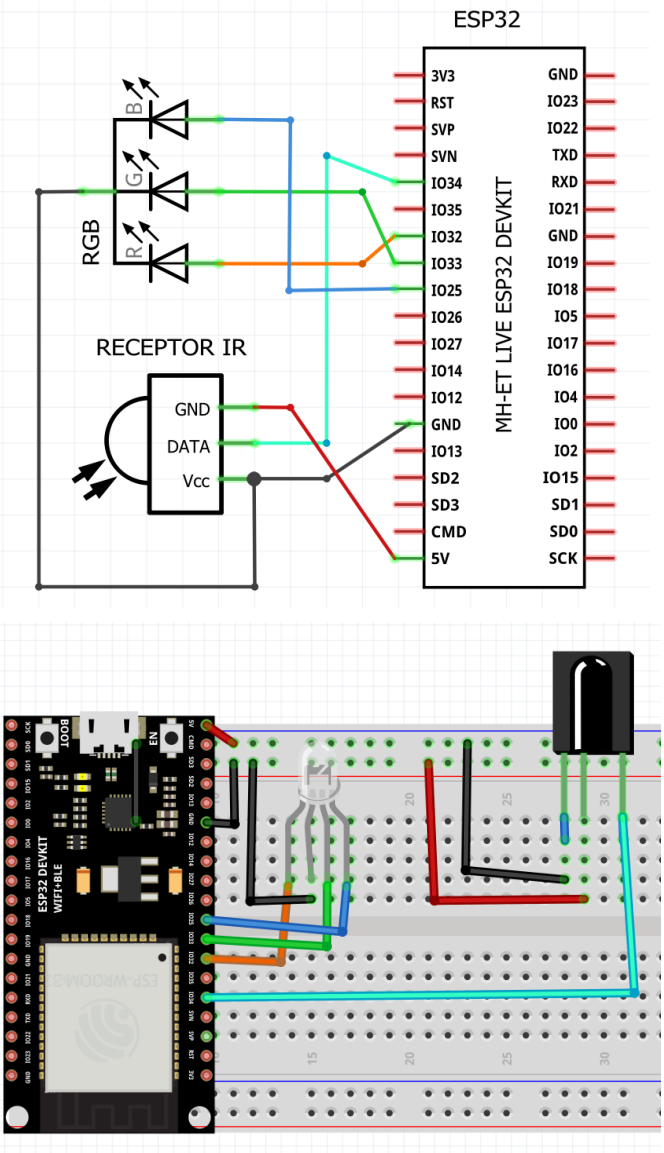


Figura 65 Diagrama de conexión practica 6

Fuente: Elaboración propia

Conexión de los componentes a la placa ESP32

El receptor IR estará conectado al pin 34 de placa y se utilizara el control remoto de arduino.

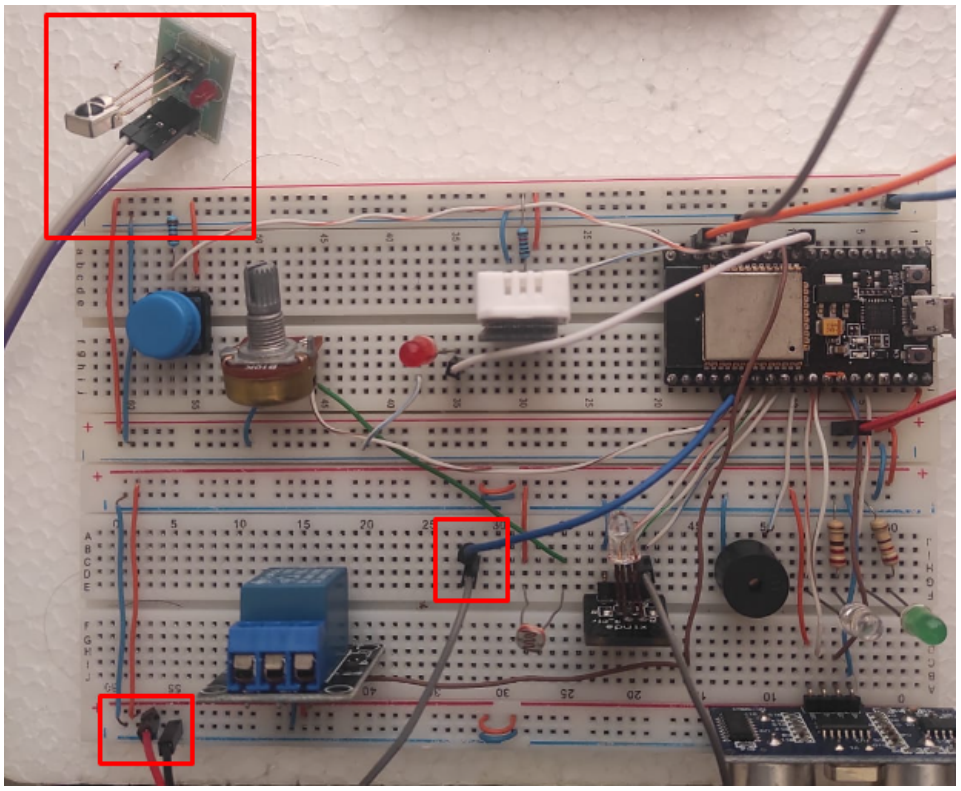


Figura 66 Conexión de la Práctica 6 en la protoboard

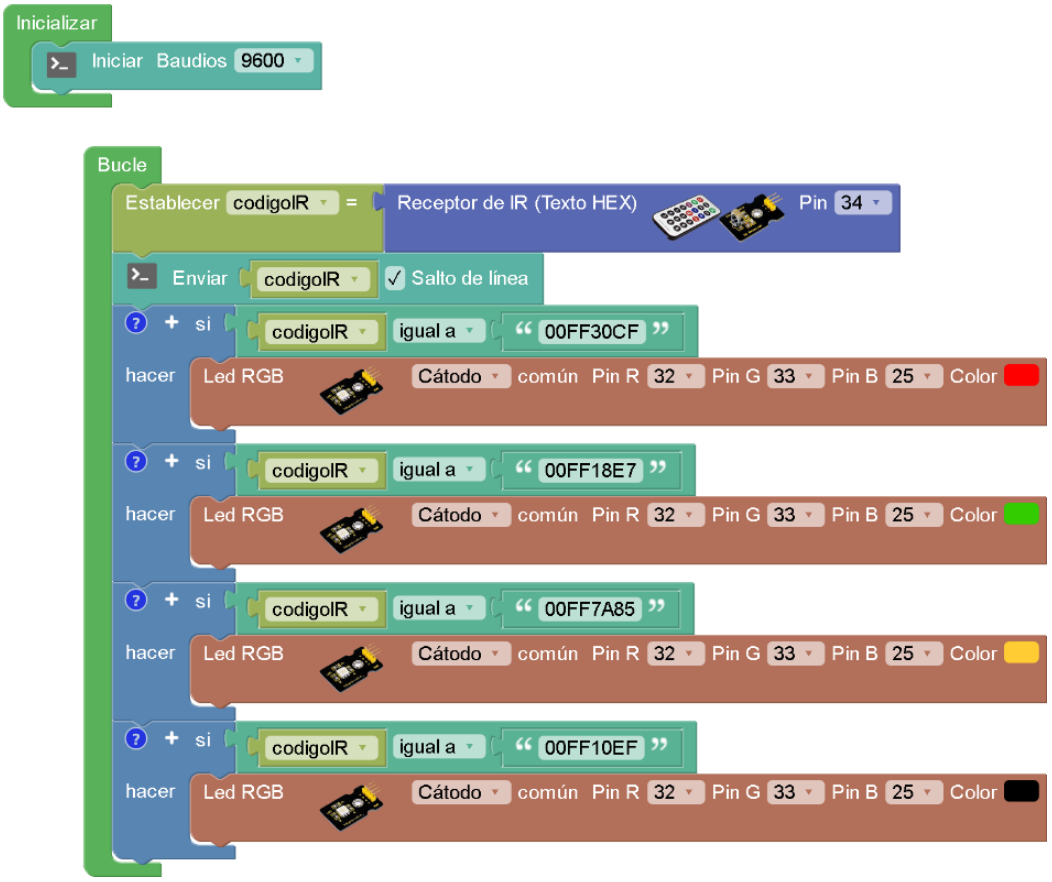
Fuente: Elaboración propia

2.4.1.6.1 CodeSkool

Figura 67 Código Práctica 6 CodeSkool

Fuente: Elaboración propia

2.4.1.6.2 ArduinoBlocks




```
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>

#include <analogWrite.h>

String s_codigoIR;
IRrecv ir_rx(34);
decode_results ir_rx_results;

String fnc_ir_rx_decode_txt()
{
    char buff[16];
    buff[0]=0;
    if(ir_rx.decode(&ir_rx_results))
    {
        sprintf(buff,"%08lx",(unsigned long)ir_rx_results.value);
        ir_rx.resume();
    }
    return String(buff);
}

void setup()
{
    pinMode(32, OUTPUT);
    pinMode(33, OUTPUT);
    pinMode(25, OUTPUT);

    Serial.begin(9600);
    Serial.flush();
    while(Serial.available()>0)Serial.read();

    ir_rx.enableIRIn();

}

void loop()
{
    yield();

    s_codigoIR = fnc_ir_rx_decode_txt();
    Serial.println(s_codigoIR);
    if (String(s_codigoIR).equals(String("00FF30CF"))) {
        analogWrite(32,255);
        analogWrite(33,0);
        analogWrite(25,0);
    }

    if (String(s_codigoIR).equals(String("00FF18E7"))) {
        analogWrite(32,51);
        analogWrite(33,204);
        analogWrite(25,0);
    }

    if (String(s_codigoIR).equals(String("00FF7A85"))) {
        analogWrite(32,51);
        analogWrite(33,51);
        analogWrite(25,255);
    }

    if (String(s_codigoIR).equals(String("00FF10EF"))) {
        analogWrite(32,255);
        analogWrite(33,204);
        analogWrite(25,0);
    }

}
```

Figura 68 Código Práctica 6 ArduinoBlocks

Fuente: Elaboración propia

2.4.1.6.3 Ejecución Práctica 5

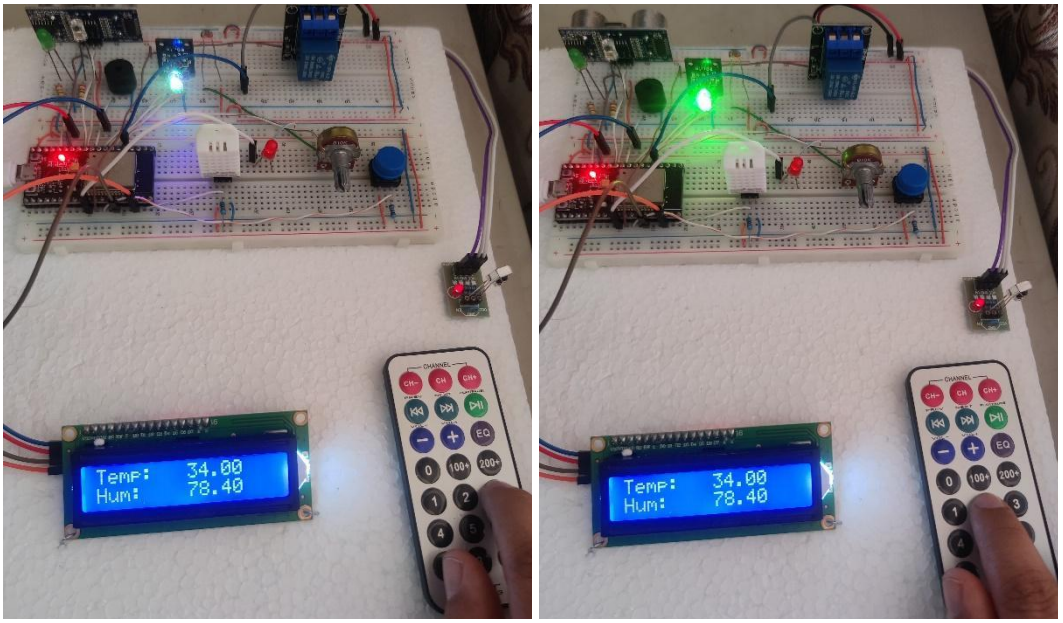


Figura 69 Ejecución práctica 6

Fuente: Elaboración propia

2.4.2 GRUPO 2 : COMUNICACIÓN POR WIFI ESP32

2.4.2.1 MQTT

Objetivos:

- Utilizar el protocolo MQTT mediante la modalidad de suscriptor-publicador para la comunicación IOT.
- Utilizar las prácticas del grupo 1 para realización de comunicación con el protocolo MQTT y la ESP32.

Descripción: Para la realización de la práctica se utiliza el bróker Mosquitto y la herramienta de programación visual Node red para la comunicación cliente servidor.

- MQTT es un protocolo que permite la comunicación M2M (machine to machine) . Funciona a través de un servidor central llamado bróker en donde se crean tópicos o temas, los clientes que realizan la conexión al bróker pueden publicar y suscribirse a los tópicos.
- Node red permite la creación de sistemas IOT mediante la programación basada en bloques. Diseñada para facilitar la comunicación entre hardware y los servicios de internet, su programación está basada en lenguaje de JavaScript.

- Mosquitto es un servidor de mensajes que permite la conexión de los clientes los cuales pueden publicar y suscribirse a un tópico.

A continuación se detallan las prácticas que se realizaron con la placa ESP32 utilizando el protocolo MQTT.

Tabla 12 Prácticas Grupo 2

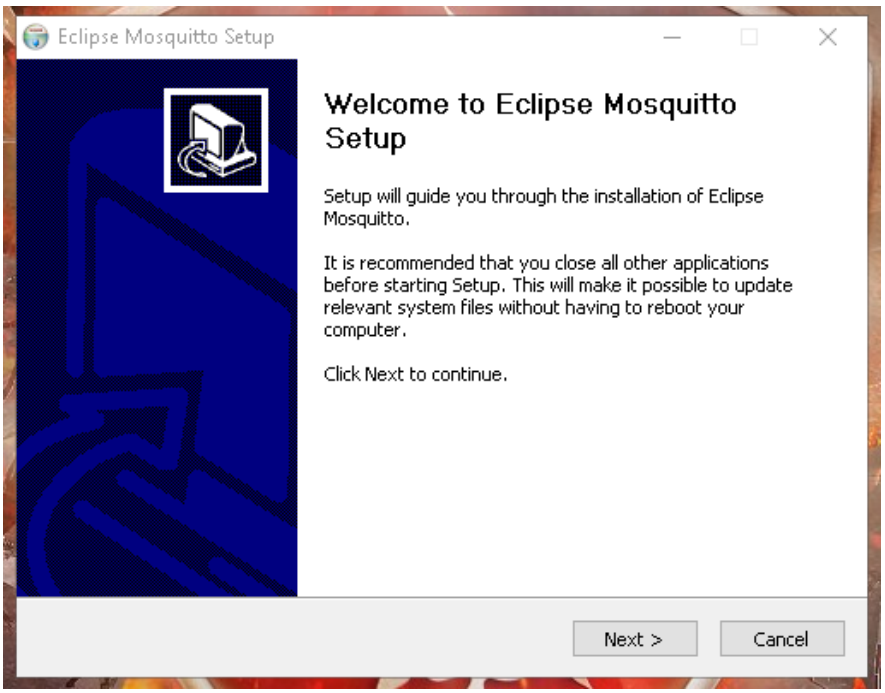
No. PRACTICA	Nombre de la practica	Objetivo
PRACTICA 1	ENCENDIDO DE LED	Encender y apagar leds mediante un servidor web utilizando el protocolo de comunicación MQTT
PRACTICA 2	LCD y DTH22	Leer la temperatura y humedad del sensor DTH22, mostrar los resultados a través de la pantalla LCD y dashboard en la página web.
PRÁCTICA 3	SENSOR ULTRASONICO – RGB – BUZZER	Crear una alarma de detección de objetos mediante el uso del sensor ultrasónico y enviar una alerta a través de telegram.
PRÁCTICA 4	SENSOR LDR	Encender o apagar un led de acuerdo al nivel de luz que incide sobre el sensor LDR.
PRÁCTICA 5	RELÉ	Activar el módulo relé a través del uso de un botón utilizando el protocolo de comunicación MQTT

Fuente: Elaboración propia

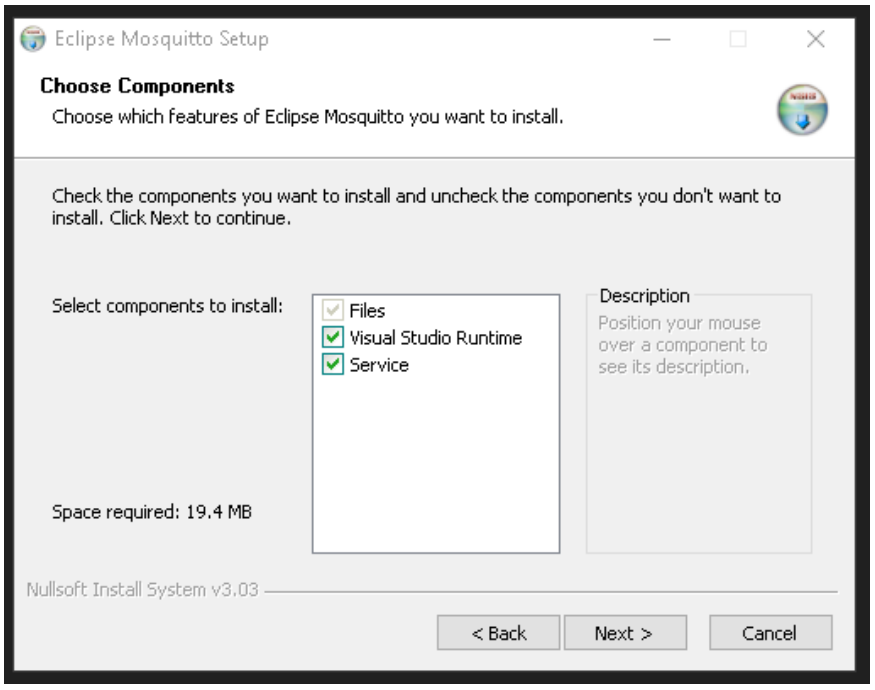
2.4.2.1.1 Instalación Mosquitto

Paso 1: Damos doble click sobre el instalador de Mosquitto el cual podremos descargarlo de la página oficial.

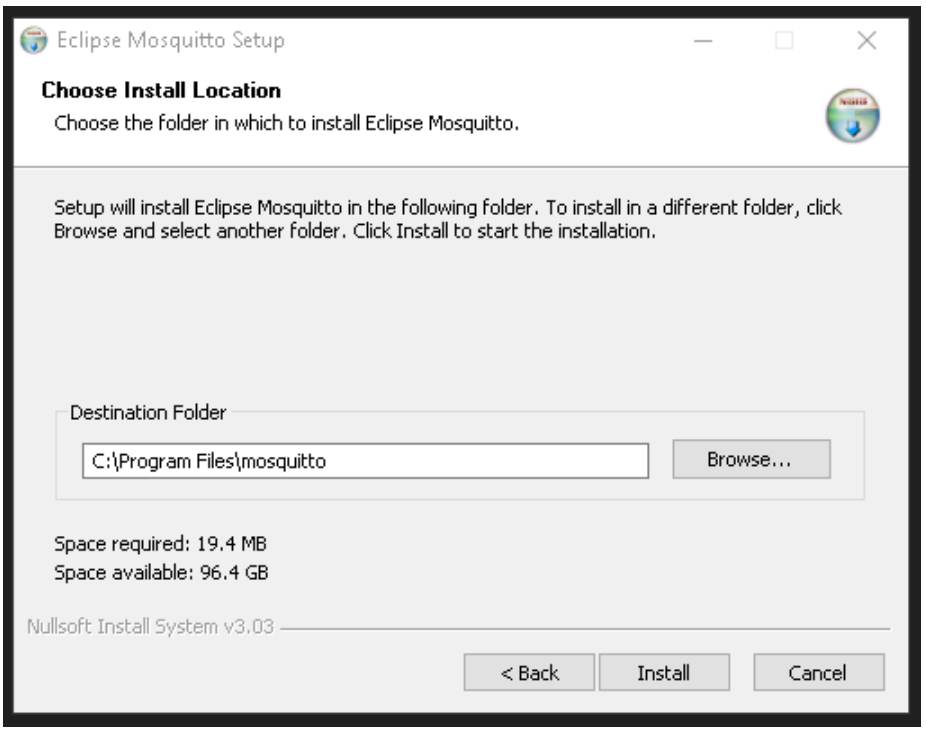
Paso 2: Damos doble click en Next para continuar con la instalación.



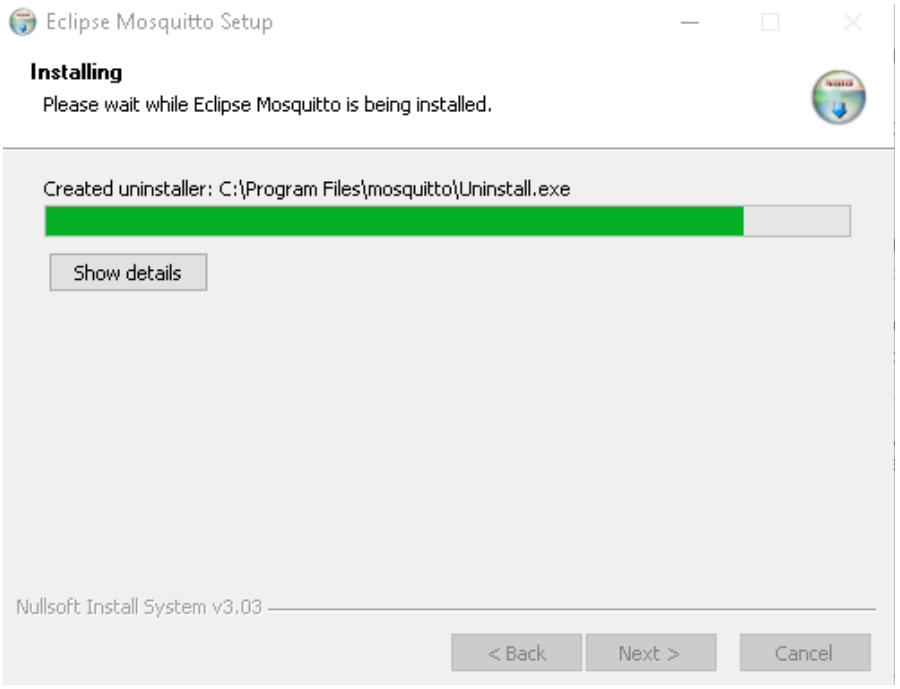
Paso 3: Damos doble click en Next .



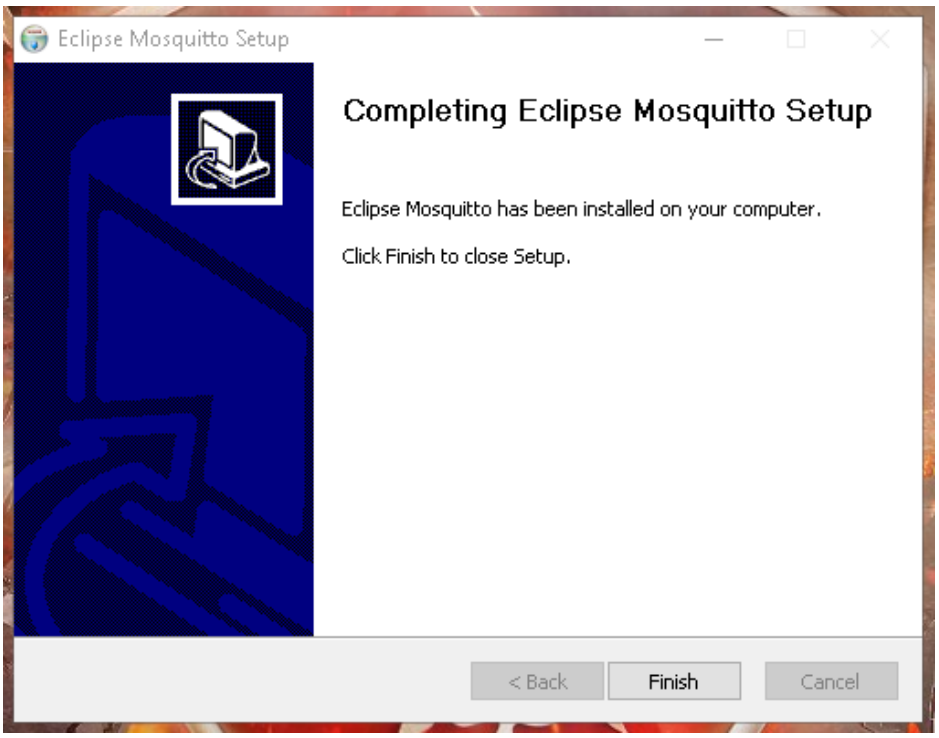
Paso 4: Seleccionamos la ruta donde deseamos instalar el programa y click en install.



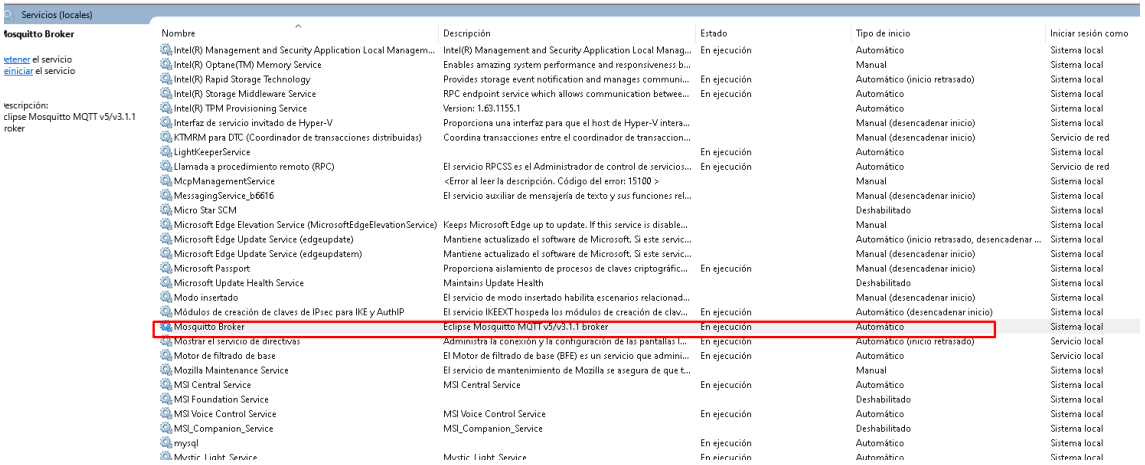
Paso 5: Esperamos que se complete la instalación.



Paso 6: Damos click en Finish.



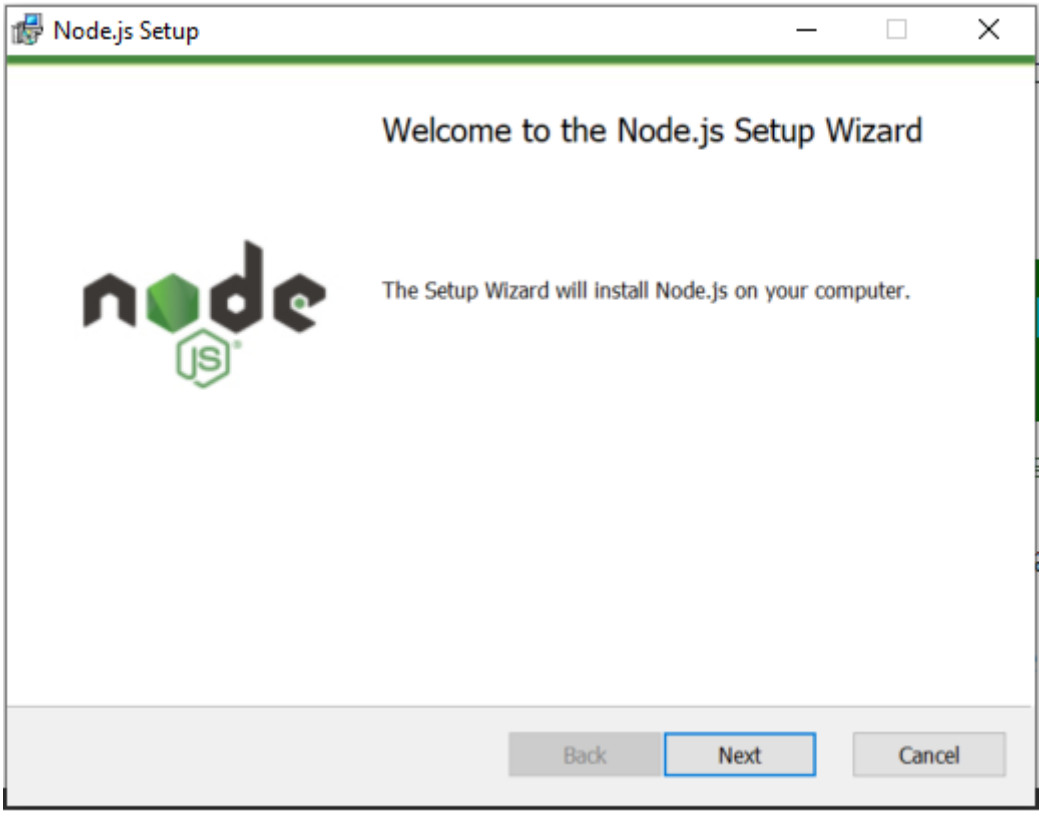
Paso 7: Para verificar que Mosquitto está instalado nos dirigimos a servicios y verificamos si el servicio está ejecutándose.



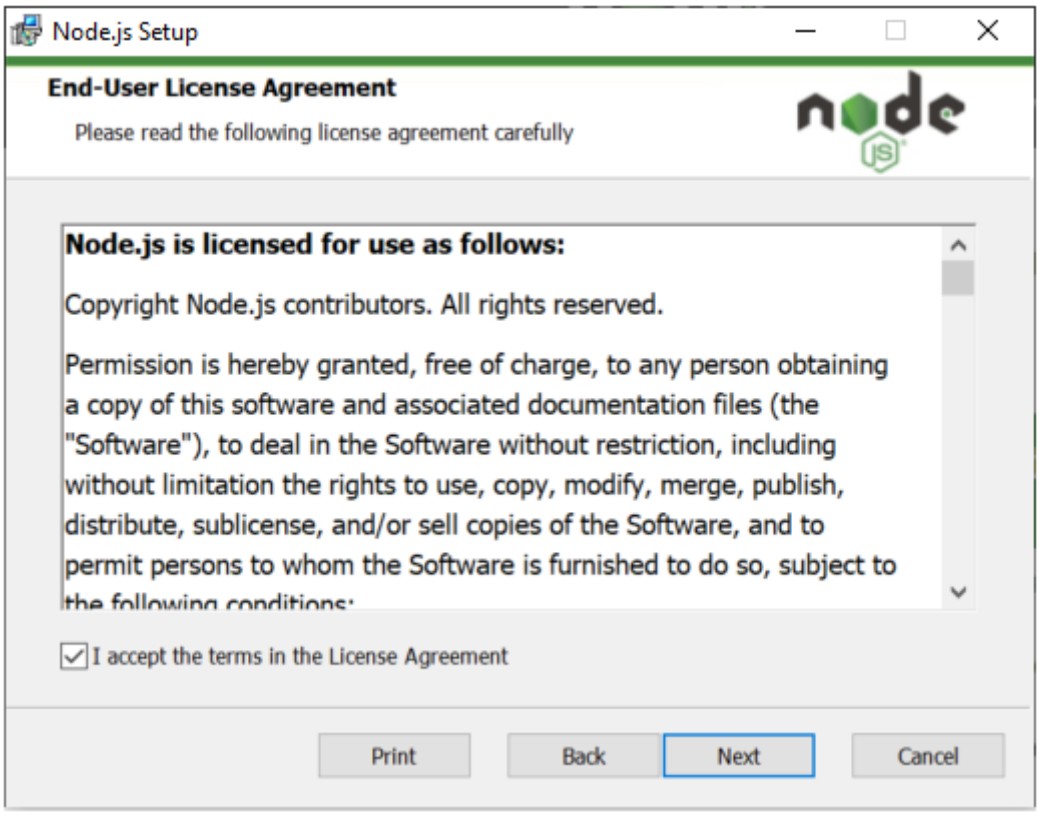
2.4.2.1.2 Instalación Node-red

Paso 1: Click sobre el instalador de Node-RED.

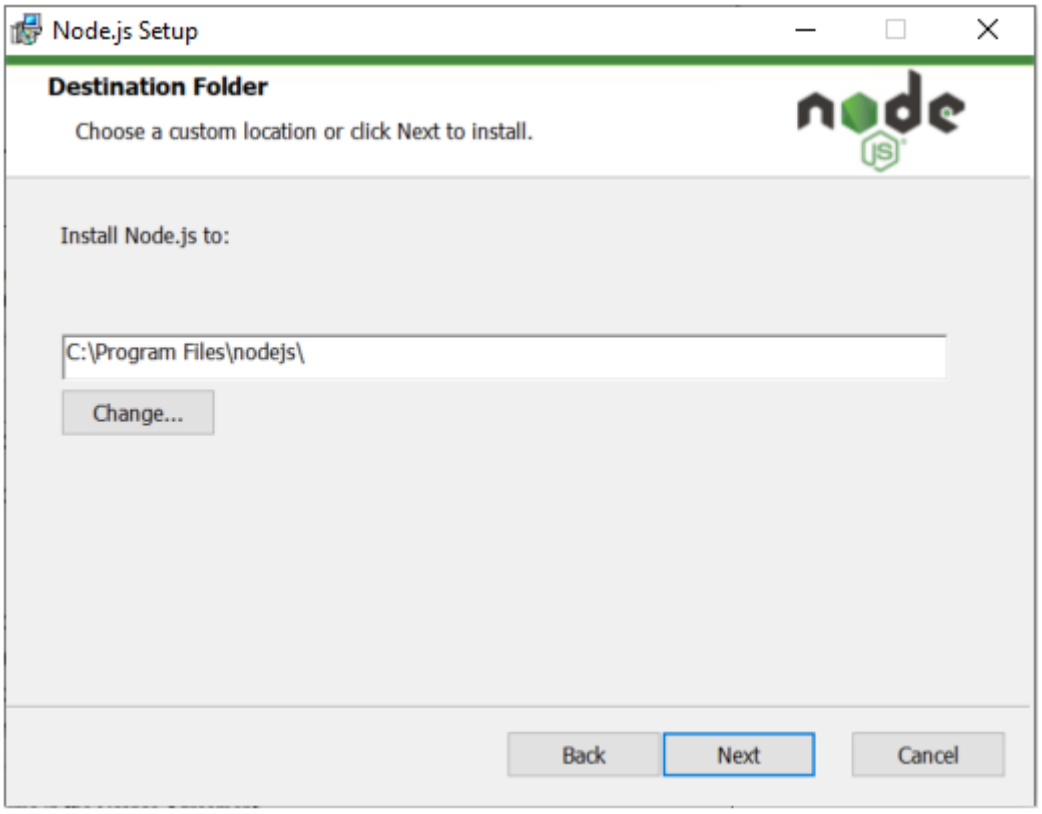
Paso 2: Click en next.



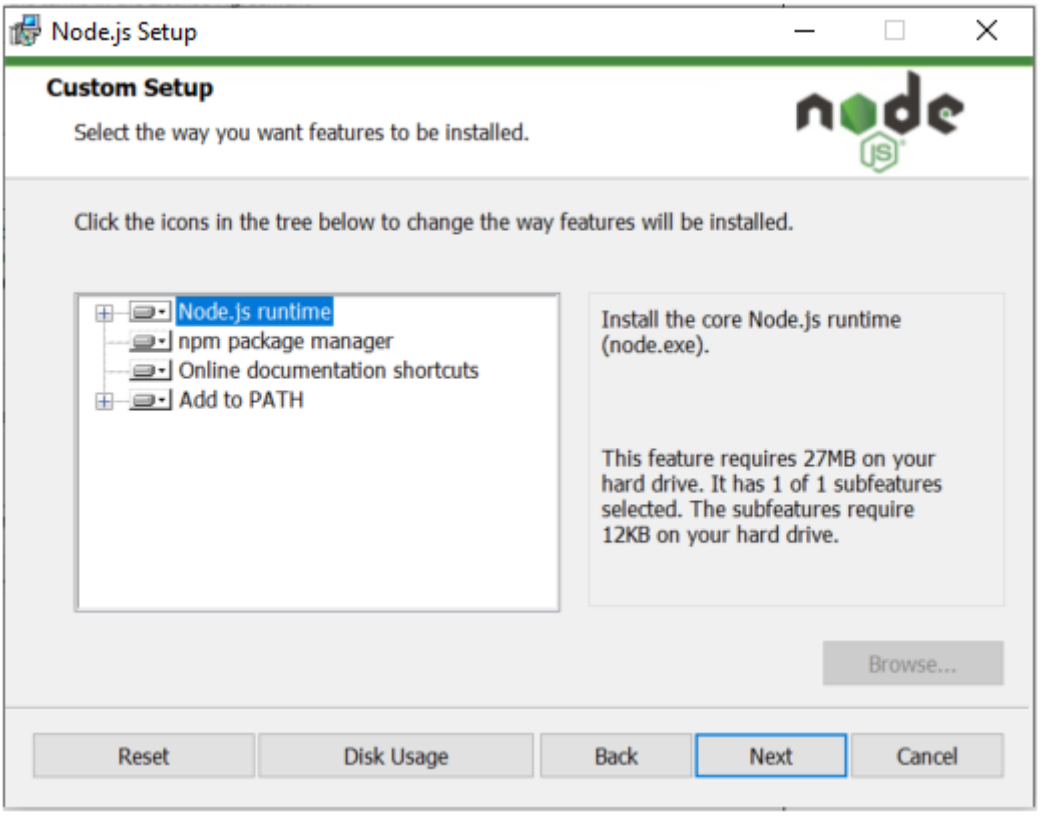
Paso 3: Seleccionamos la casilla de aceptar términos y Click en next.



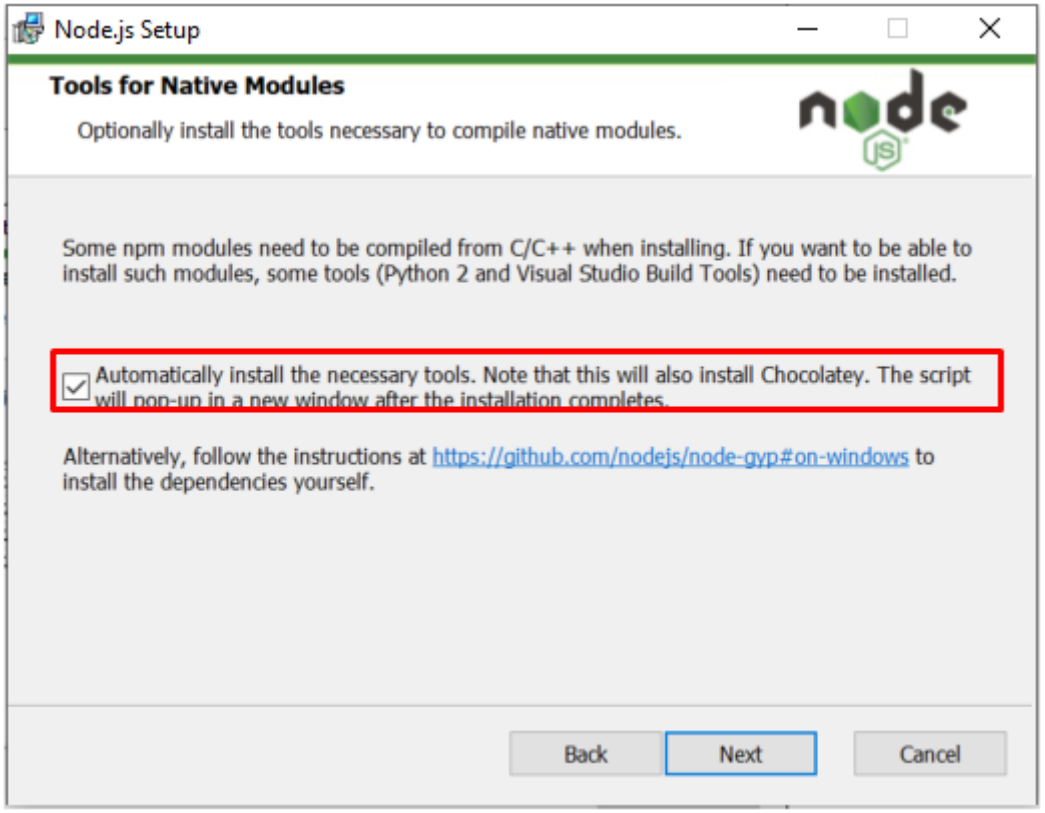
Paso 4: Seleccionamos la ubicación y click en Next.



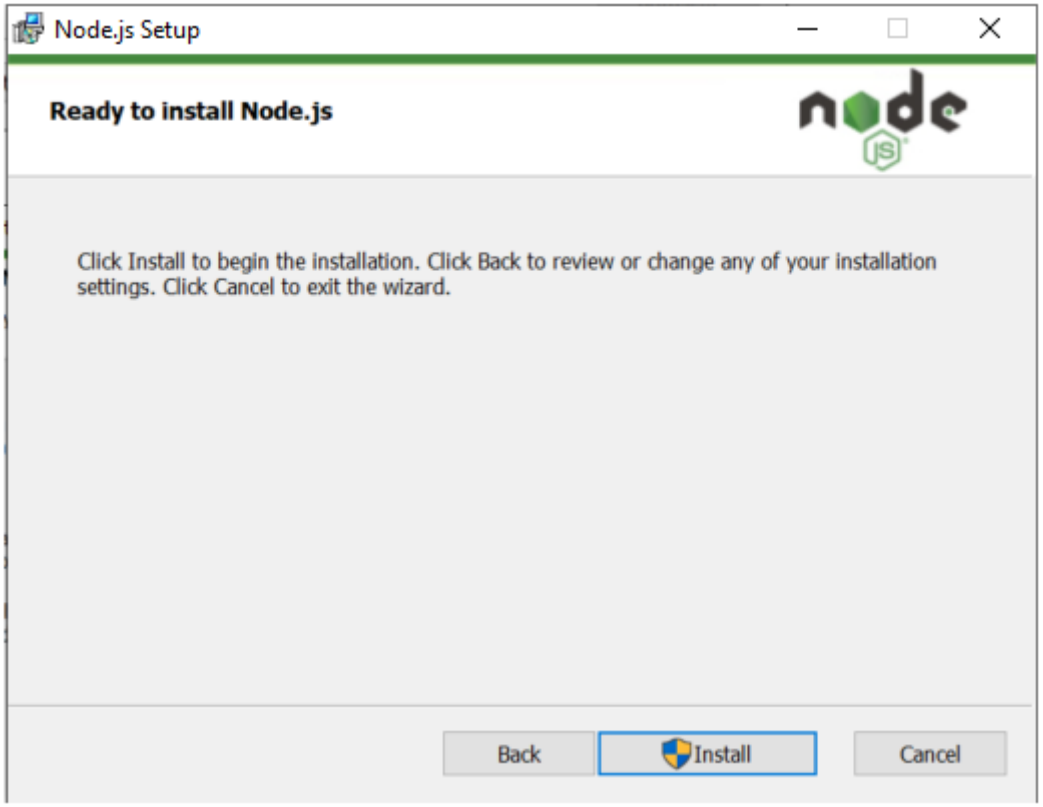
Paso 6: Click en next.



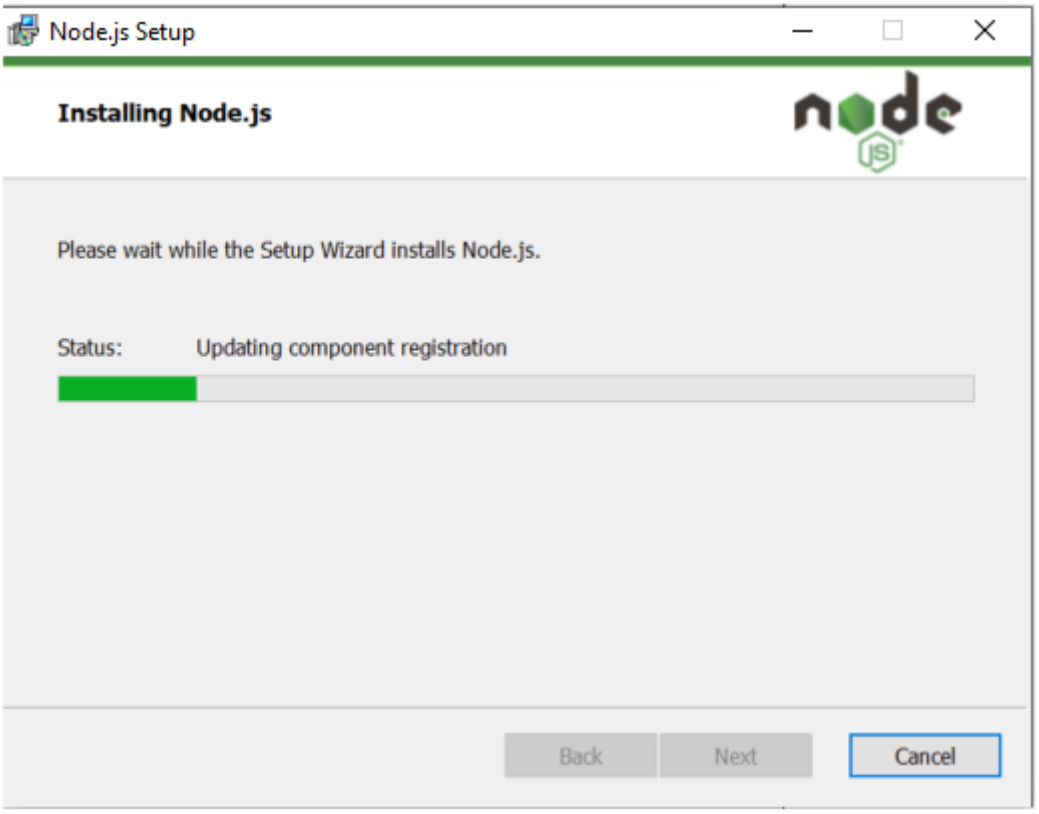
Paso 7: Click en Next y seleccionamos la casilla.



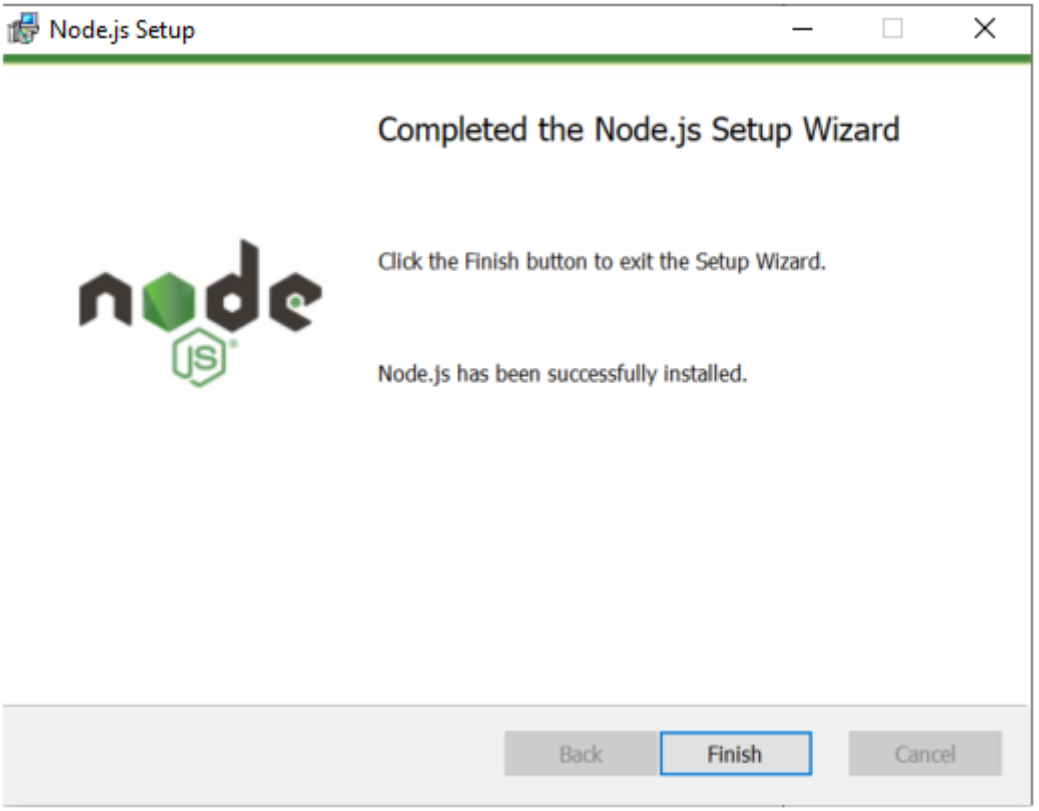
Paso 8: Click en Install.



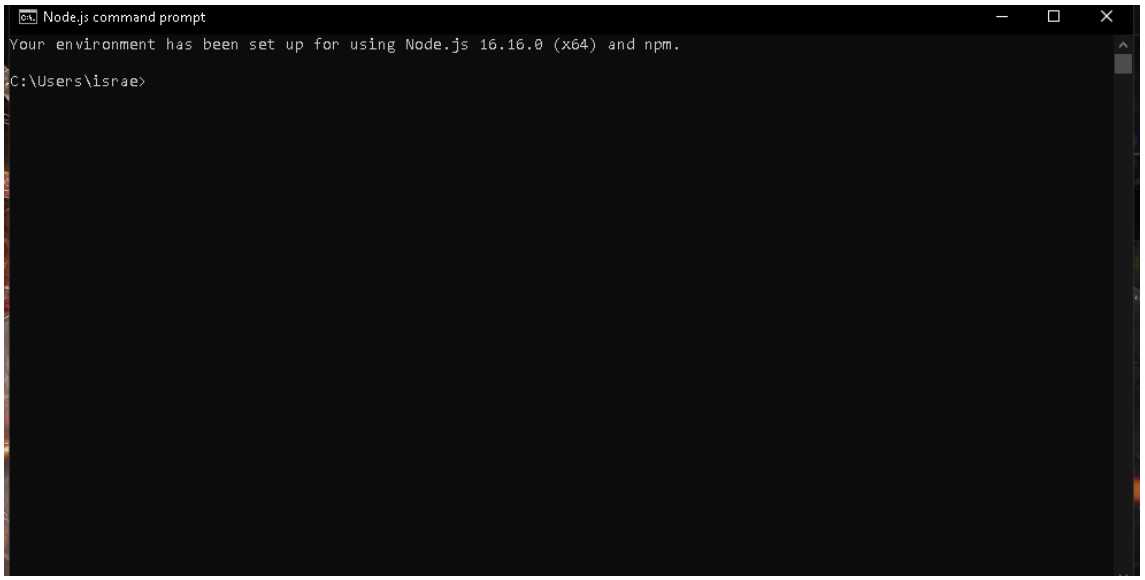
Paso 9: Esperamos que se complete la instalación.



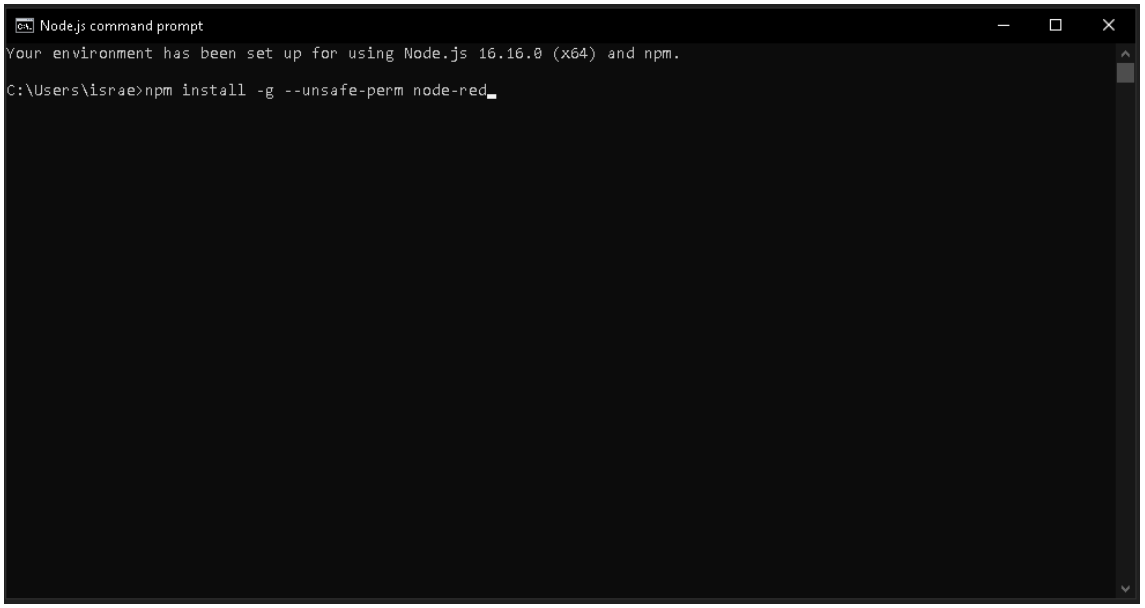
Paso 8: Click en Finish.



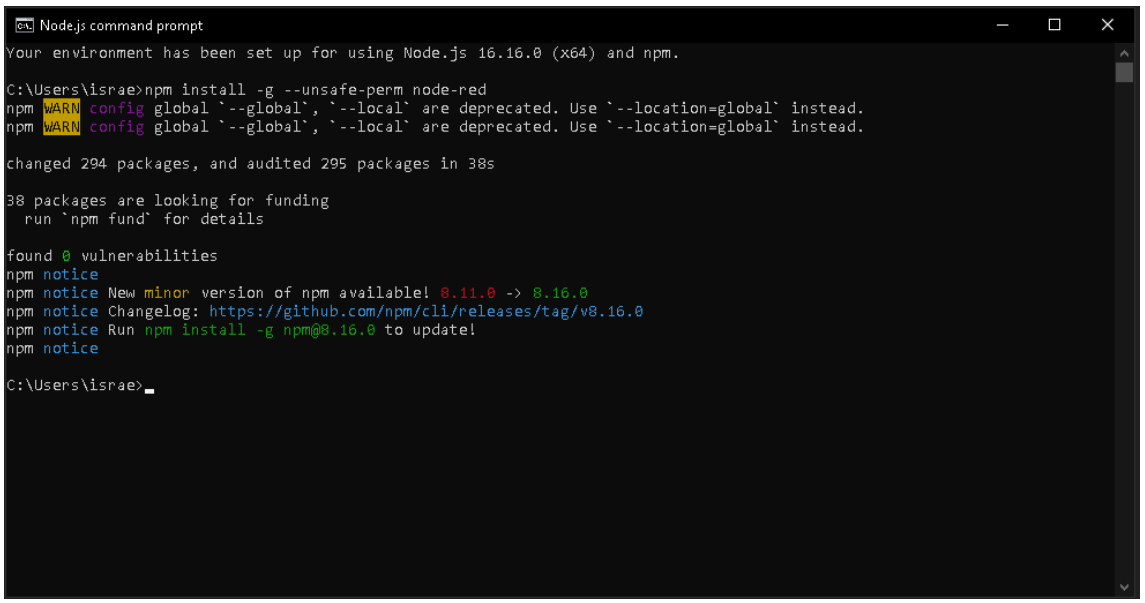
Paso 9: Abrimos el command prompt de node.js



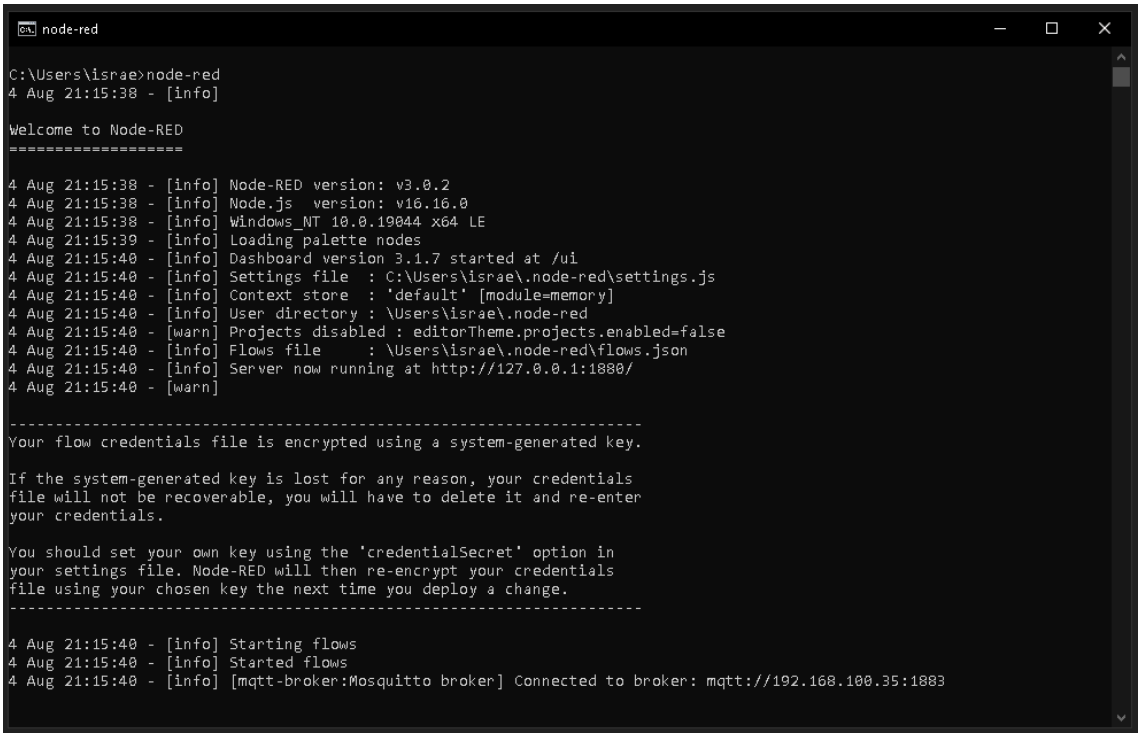
Paso 10: Escribimos el siguiente comando: *npm install -g --unsafe-perm node-red*.



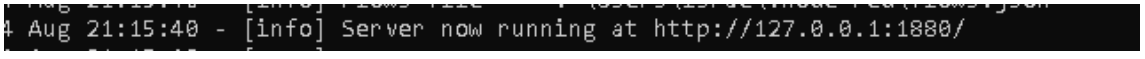
Paso 11: Esperamos que se complete la instalación.

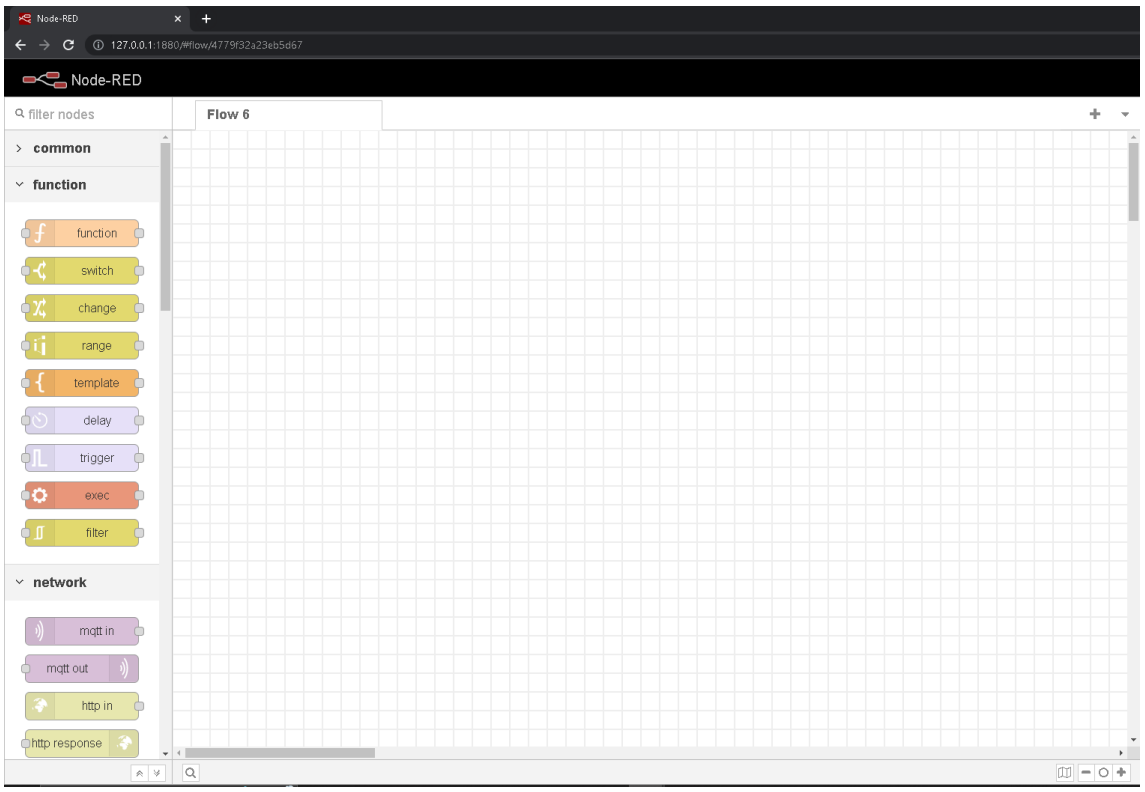


Paso 12: Ejecutamos escribiendo el node-red.



Paso 12: Colocamos la dirección donde está corriendo nuestro servidor en el navegador.





2.4.2.1.3 PRÁCTICA 1: ENCENDIDO DE LED

Objetivos: Encender y apagar leds mediante un servidor web utilizando el protocolo de comunicación MQTT.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 1 del grupo de periféricos.

- Se crearán dos tópicos de encender/apagar/led para que la ESP32 se suscriba.
- Utilizando un switch en node-red se publicará en ese tópico, un 1 para encender y un 0 para apagar el led.

Código Arduinoblocks

El código C equivalente a este código se encuentra en el Anexo A.

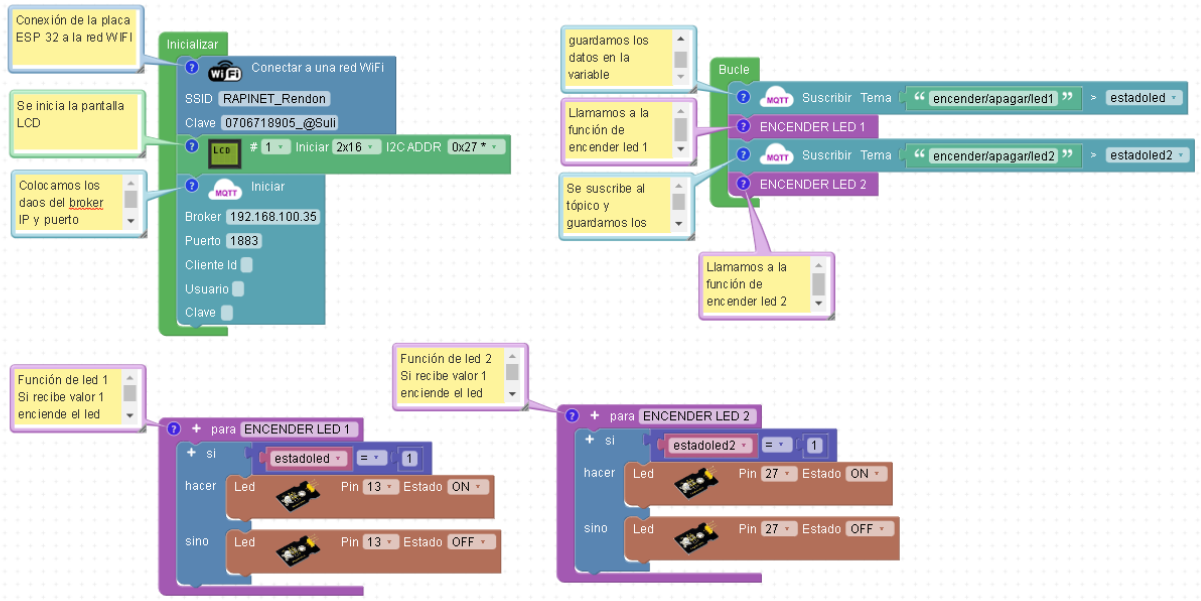


Figura 70 Código PRÁCTICA 1 MQTT

Fuente: Elaboración propia

Node-red

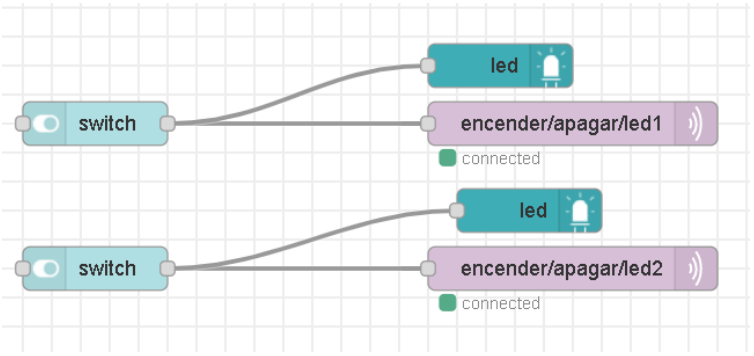


Figura 71 Node-red Práctica 1

Fuente: Elaboración propia

Ejecución

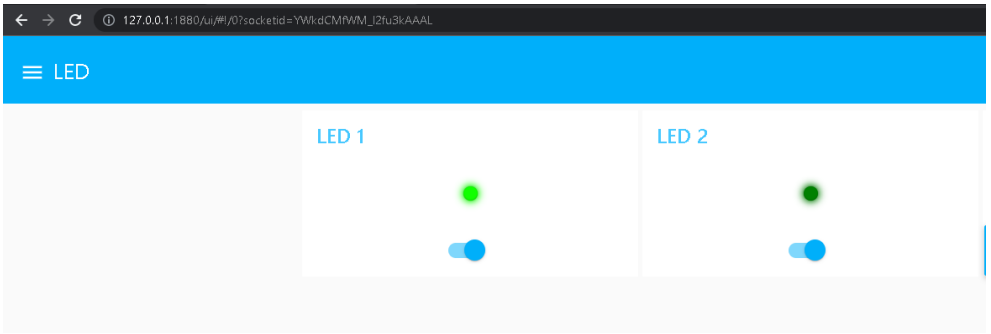


Figura 72 Ejecución Práctica 1

Fuente: Elaboración propia

2.4.2.1.4 PRÁCTICA 2: LCD y DTH22

Objetivos: Leer la temperatura y humedad del sensor DTH22, mostrar los resultados a través de la pantalla LCD y dashboard en la página web.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 2 del grupo de periféricos. Se utilizará el protocolo de comunicación MQTT para la lectura de los datos del sensor.

- Se crearán dos tópicos en la ESP32 llamados sensor/dth22/temp y sensor/dth22/hum.
- Utilizando un dashboard se mostrarán los valores de temperatura y humedad.

Código Arduinoblocks

El código C equivalente a este código se encuentra en el Anexo B.

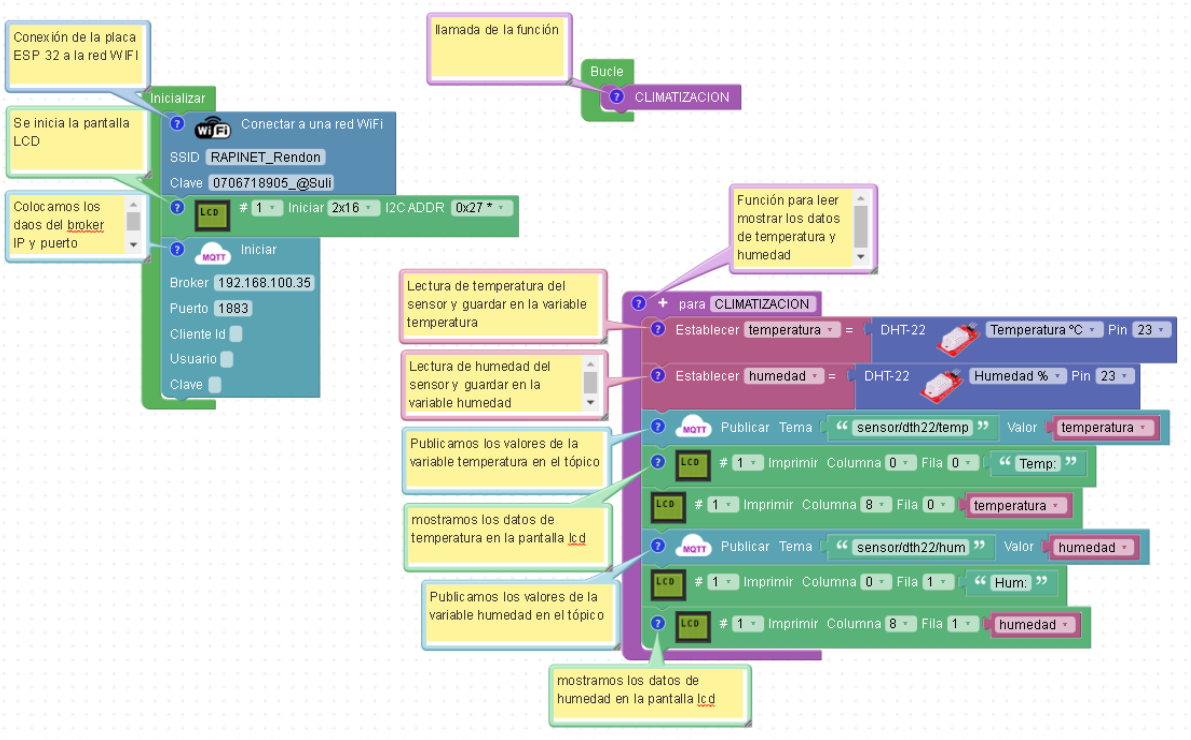


Figura 73 Código PRACTICA 2 MQTT

Fuente: Elaboración propia

Node-red

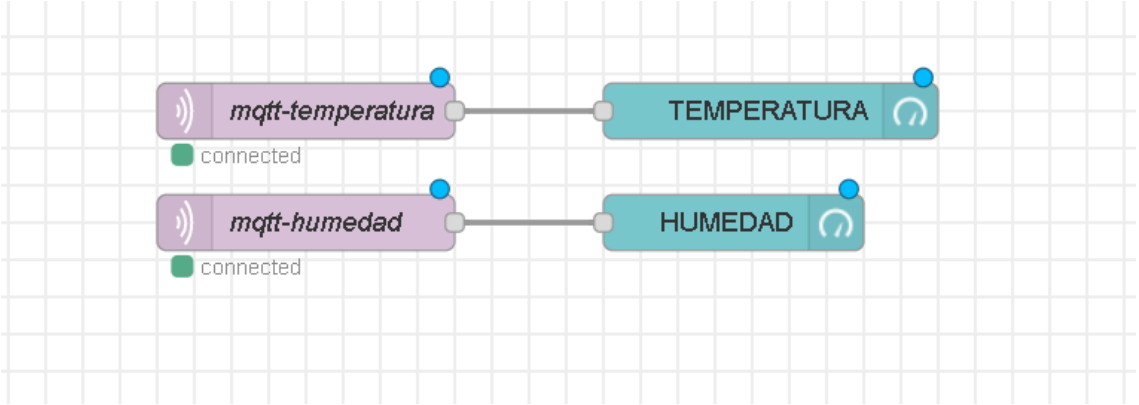


Figura 74 Node-red Práctica 2

Fuente: Elaboración propia

Ejecución

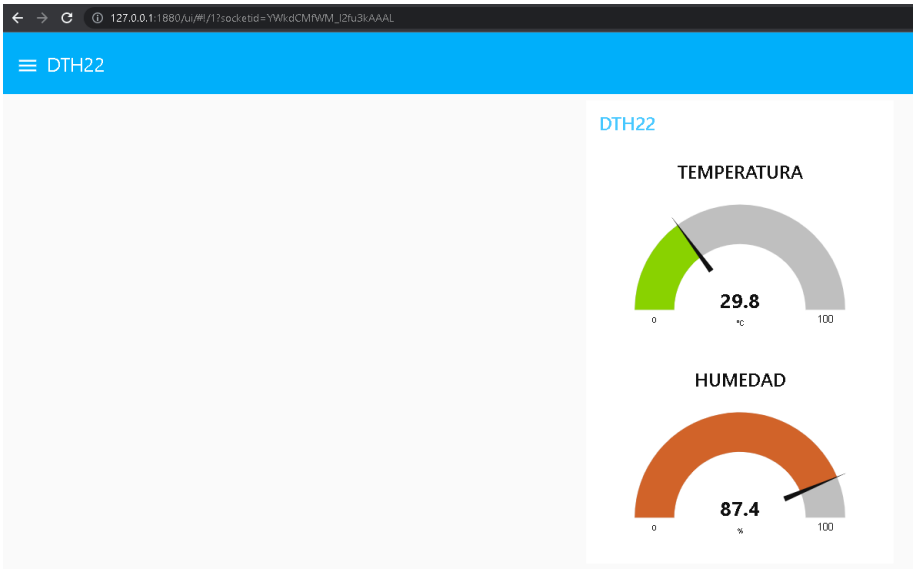


Figura 75 Ejecución Práctica 2

Fuente: Elaboración propia

2.4.2.1.5 PRÁCTICA 3: SENSOR ULTRASÓNICO – RGB – BUZZER

Objetivos: Crear una alarma de detección de objetos mediante el uso del sensor ultrasónico y enviar una alerta de mensaje a través de telegram.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 3 del grupo de periféricos. A través del protocolo de comunicación MQTT se realizará la lectura de los datos, además se establecerá el valor de la distancia para la detección.

- Se creará un tópico en llamados distancia/establecida para el valor de la

distancia mínima que detecte el movimiento.

- Se publicarán los datos del sensor en el tópico sensor/ultrasónico.
- Si detecta movimiento en la distancia establecida se publicará un mensaje de movimiento detectado en el tópico de enviar/alarma, este mensaje será enviado a nuestra cuenta de telegram.

Código Arduinoblocks

El código C equivalente a este código se encuentra en el Anexo C.

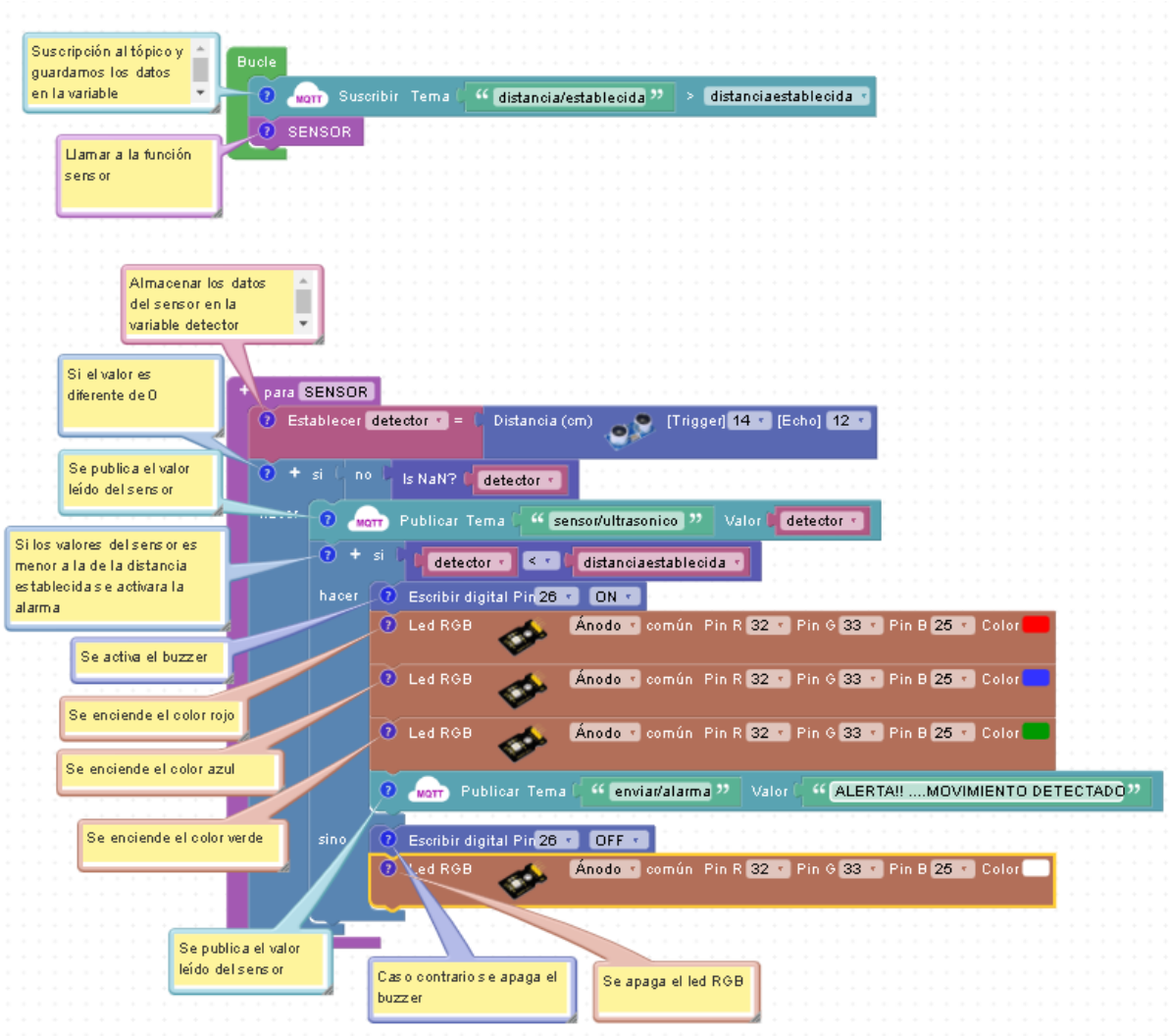


Figura 76 Código Práctica 3

Fuente: Elaboración propia

Node-red

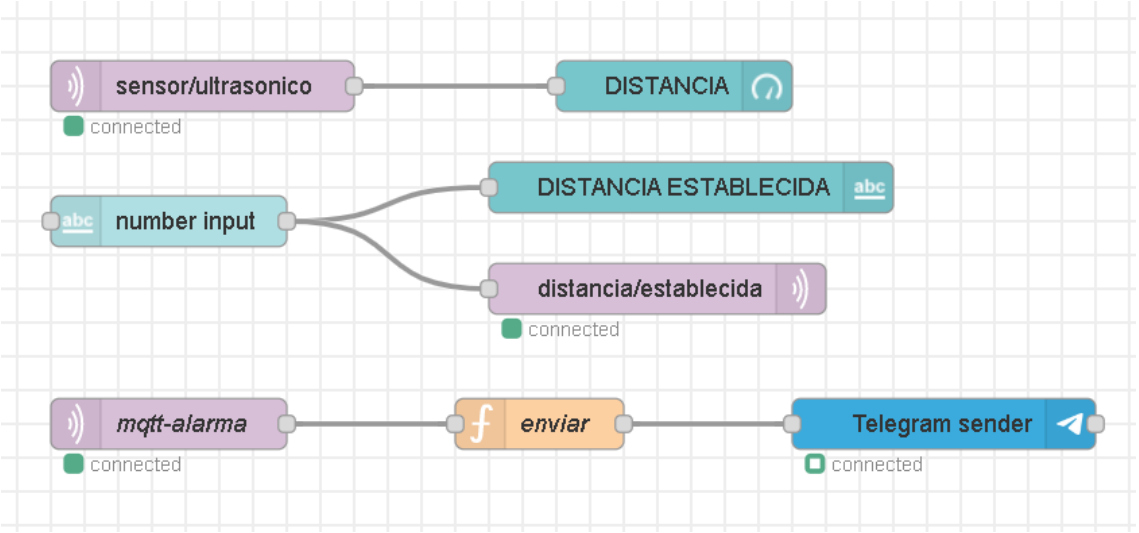


Figura 77 Node-red Práctica 3

Fuente: Elaboración propia

Ejecución

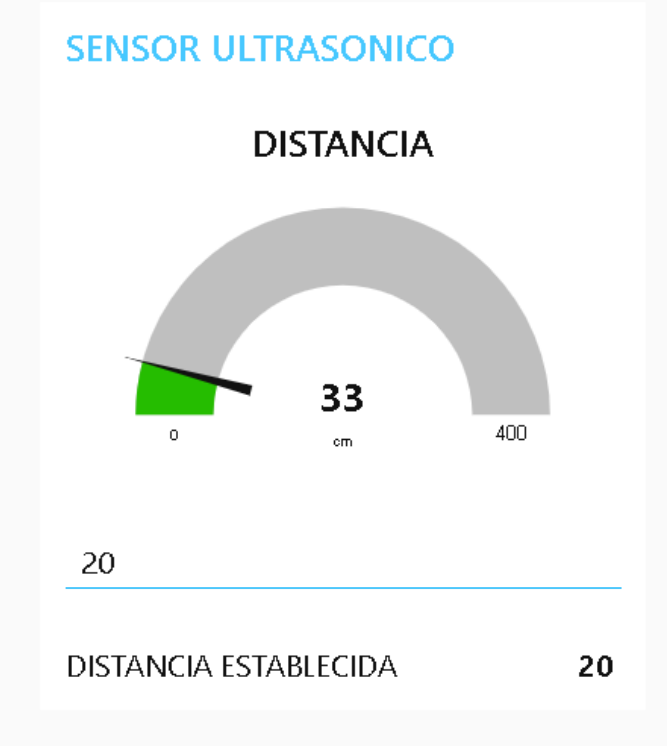


Figura 78 Ejecución Práctica 3

Fuente: Elaboración propia

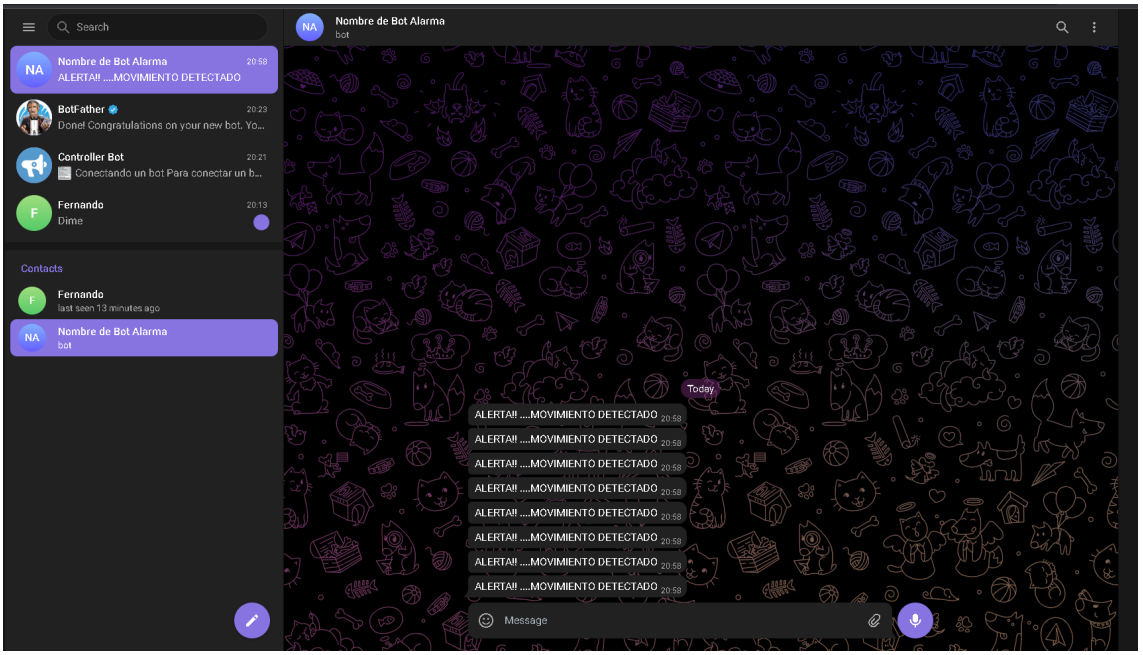


Figura 79 Mensaje de Alerta Telegram

Fuente: Elaboración propia

2.4.2.1.6 PRÁCTICA 4: SENSOR LDR

Objetivos: Encender o apagar un led de acuerdo al nivel de luz que incide sobre el sensor LDR.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 4 del grupo de periféricos. Se utilizará el protocolo de comunicación MQTT para la lectura del valor del LDR y se emplea un led.

- Se creará un tópico llamado intensidad/establecida para suscribirse a este tema y obtener el valor de la intensidad el cual se establecerá a través del uso de un slider en la página web .
- Utilizando un dashboard donde se mostrarán los valores del LDR.

Código Arduinoblocks

El código C equivalente a este código se encuentra en el Anexo D.

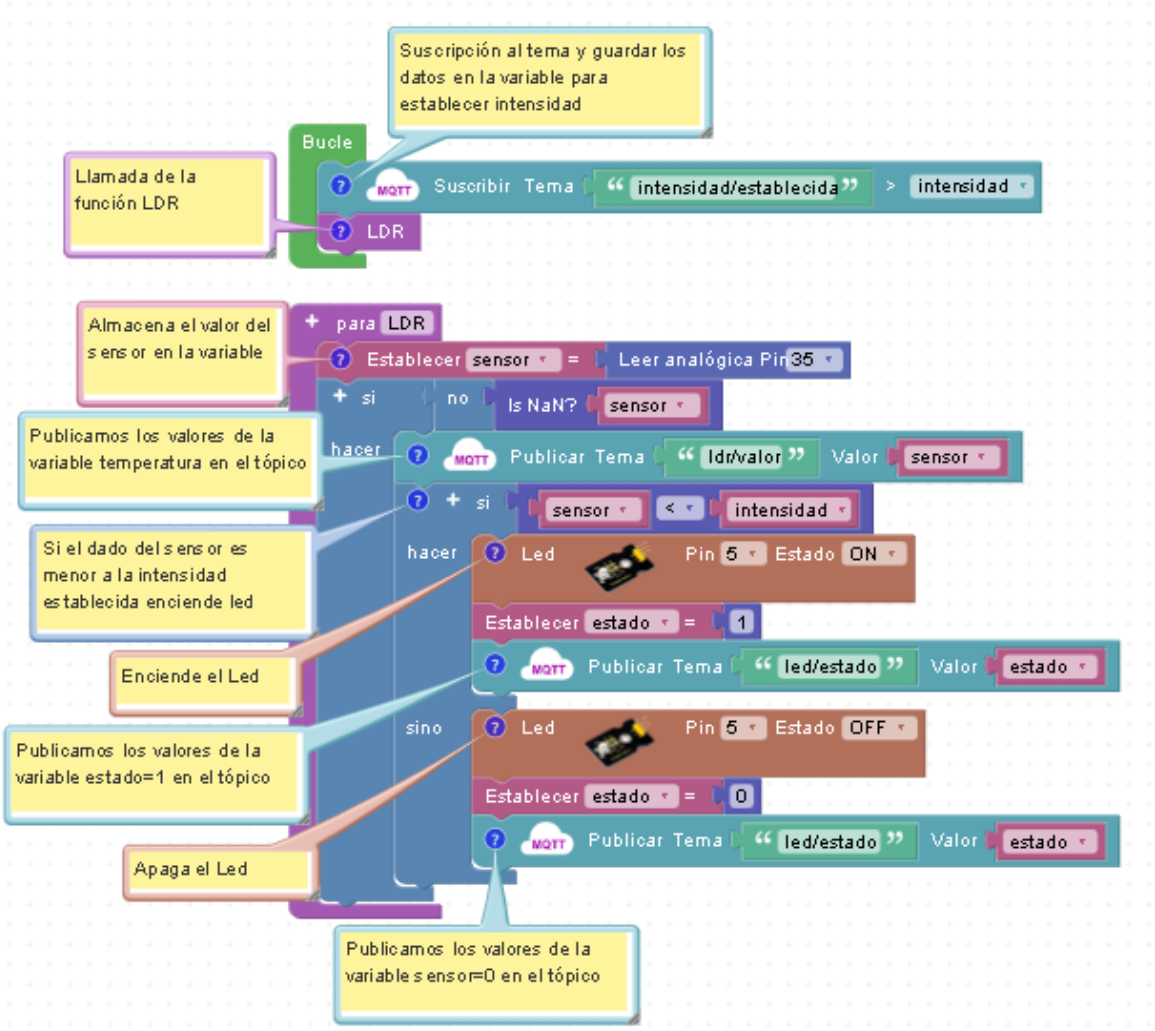


Figura 80 Código Practica 4

Fuente: Elaboración propia

Node-red

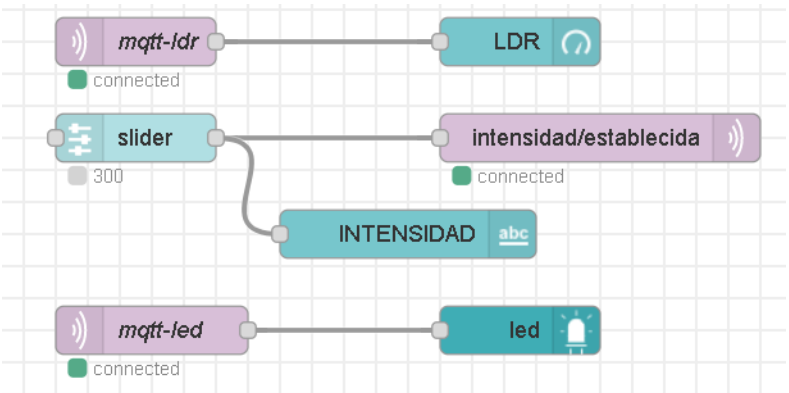


Figura 81 Node-red Práctica 4

Fuente: Elaboración propia

Ejecución

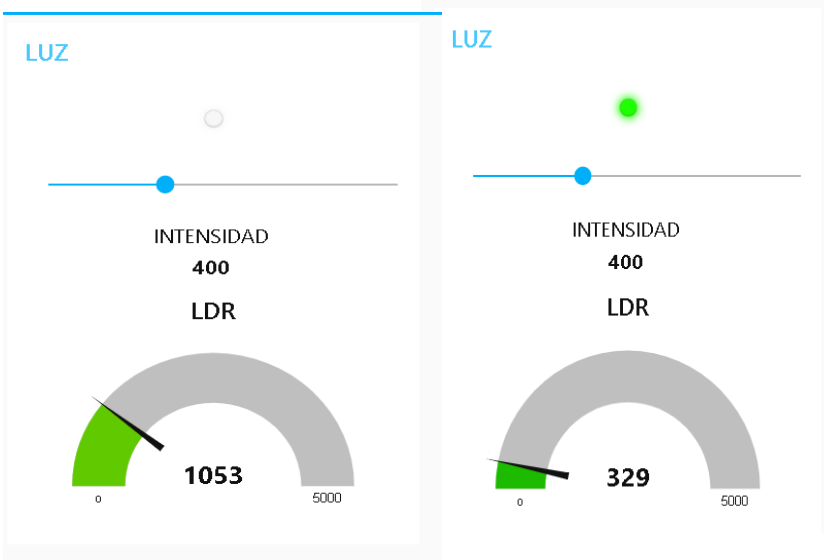


Figura 82 Ejecución Práctica 4

Fuente: Elaboración propia

2.4.2.1.7 PRÁCTICA 5: RELÉ

Objetivos: Activar el módulo relé a través del uso de un botón utilizando el protocolo de comunicación MQTT.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 5 del grupo de periféricos. Se utilizará el protocolo de comunicación MQTT para el encendido y apagado del relé.

- Se creará un tópico llamado encender/apagar/relay para suscribirte a este tema y encender en la ESP32.

Código Arduinoblocks

El código C equivalente a este código se encuentra en el Anexo E.

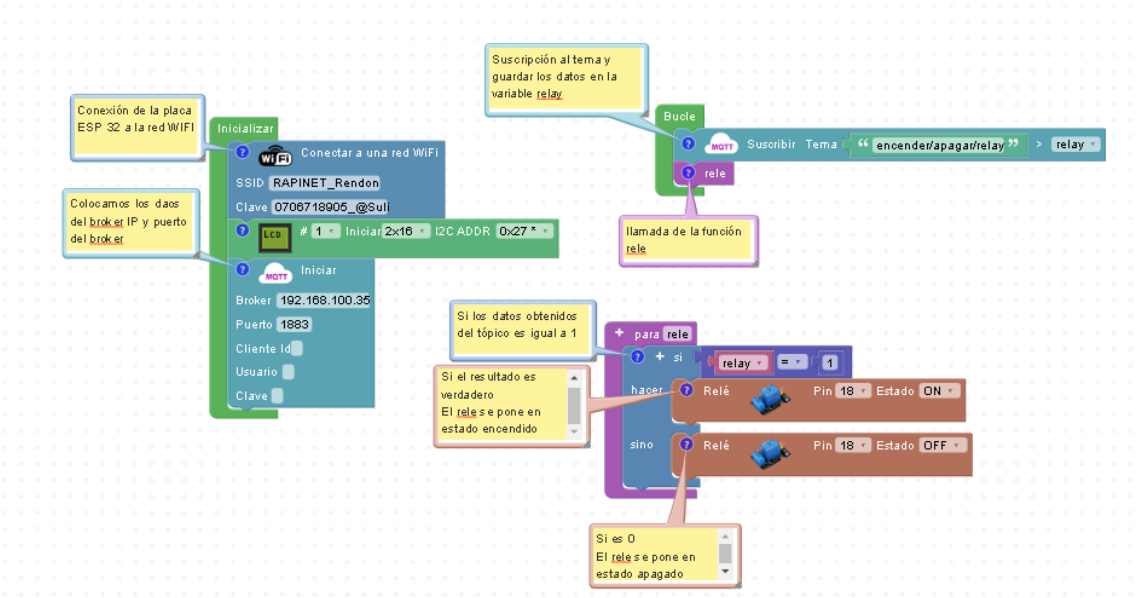


Figura 83 Código Practica 5

Fuente: Elaboración propia

Node-red

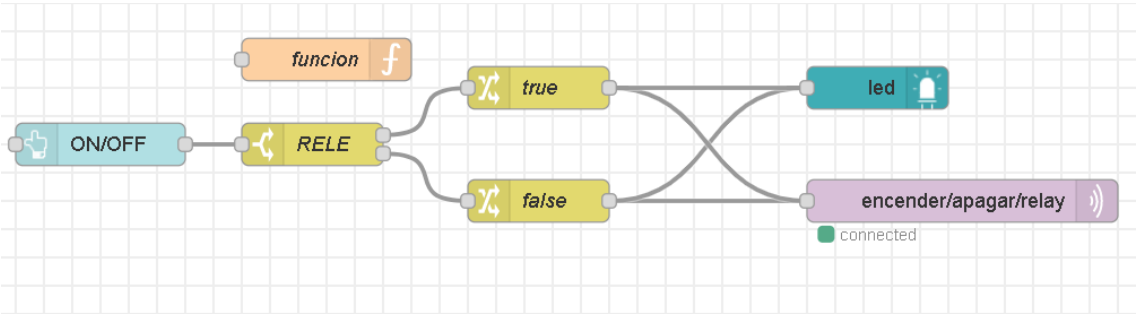


Figura 84 Node-red Práctica 5

Fuente: Elaboración propia

Ejecución

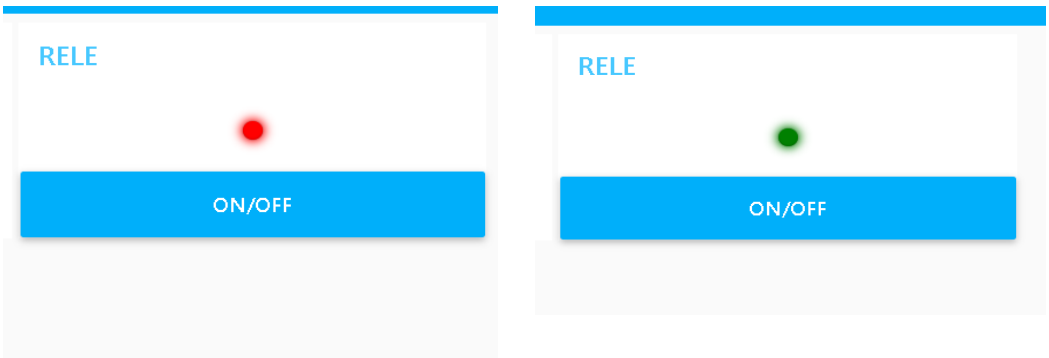


Figura 85 Ejecución Práctica 5

Fuente: Elaboración propia

2.4.2.2 SERVIDOR – GET/POST

Objetivos: Crear un servidor web sencillo en la ESP32, que permita el encendido y apagado de led y muestre los datos de los valores del DTH22 y el sensor Ultrasónico.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 1,2,3 y 5 del grupo de periféricos.

- Se realizaron peticiones para el encendido del led y el relé.
- Se realizarán peticiones para mostrar los valores de los sensores.

Arduinoblocks

Como primer paso debemos conectarnos a una red wifi, inicializamos la pantalla LCD e iniciamos el servidor en el puerto 80.

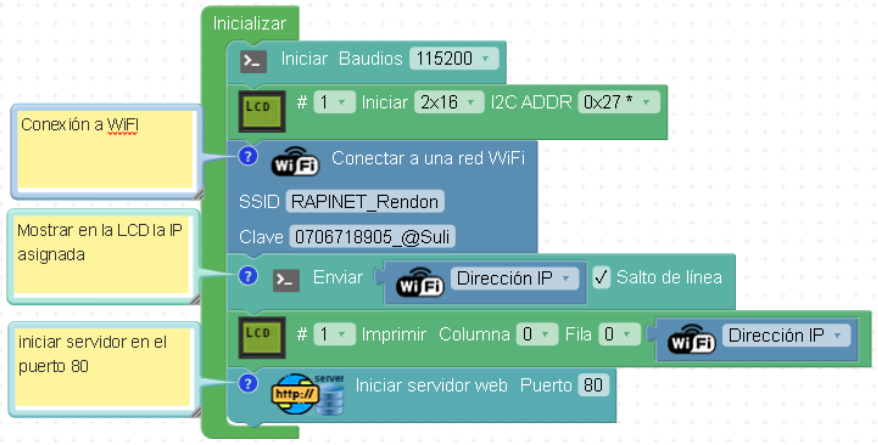


Figura 86 Conexión a la red e iniciar servidor

Fuente: Elaboración propia

Ahora procedemos a realizar la página web que mostrará el servidor cuando el cliente haga una solicitud, utilizando el método get para llamar al formulario html.

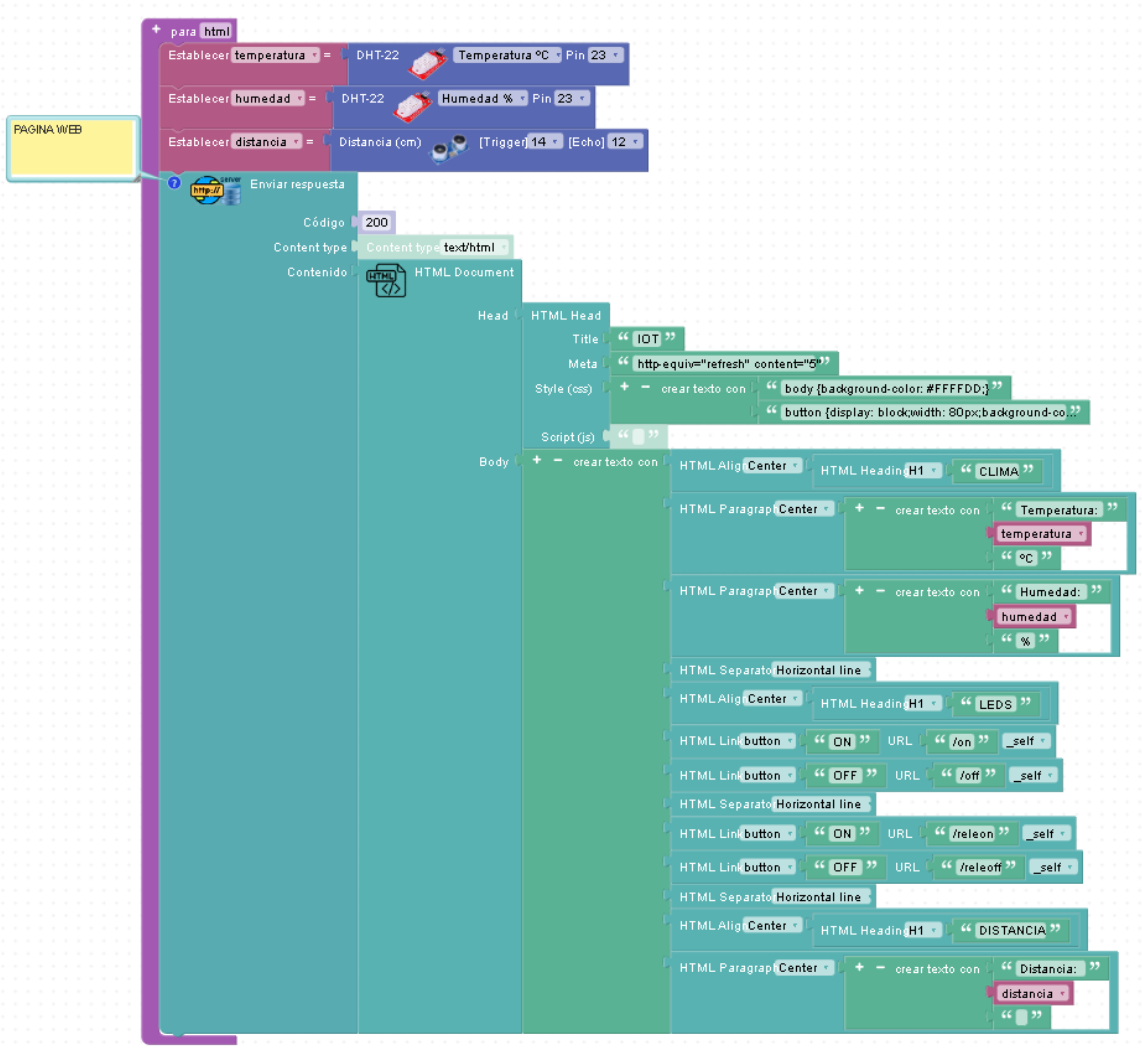


Figura 87 Página Web del servidor

Fuente: Elaboración propia

Una vez realizada la página web procedemos a realizar las peticiones de encendido y lectura de datos.

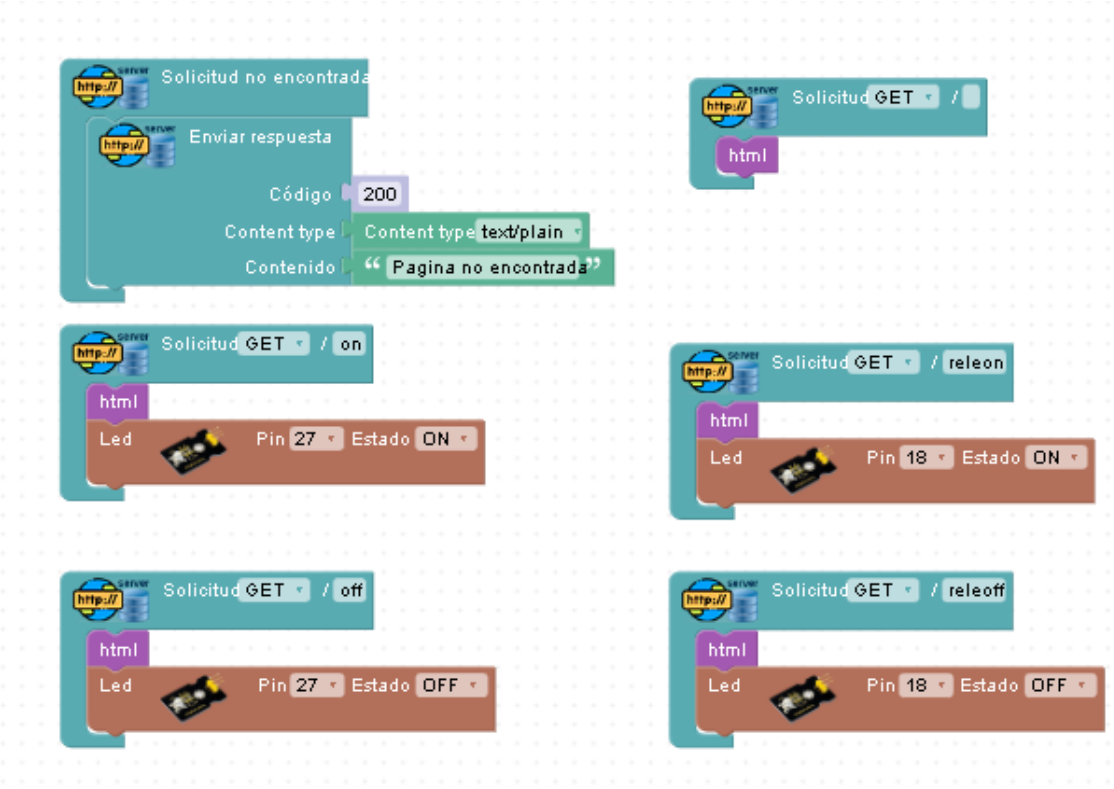


Figura 88 Peticiones HTTP – GET

Fuente: Elaboración propia

Para verificar los datos del sensor que esté tomando la lectura correcta mostramos por la pantalla LCD. El código C equivalente a este código se encuentra en el Anexo F.

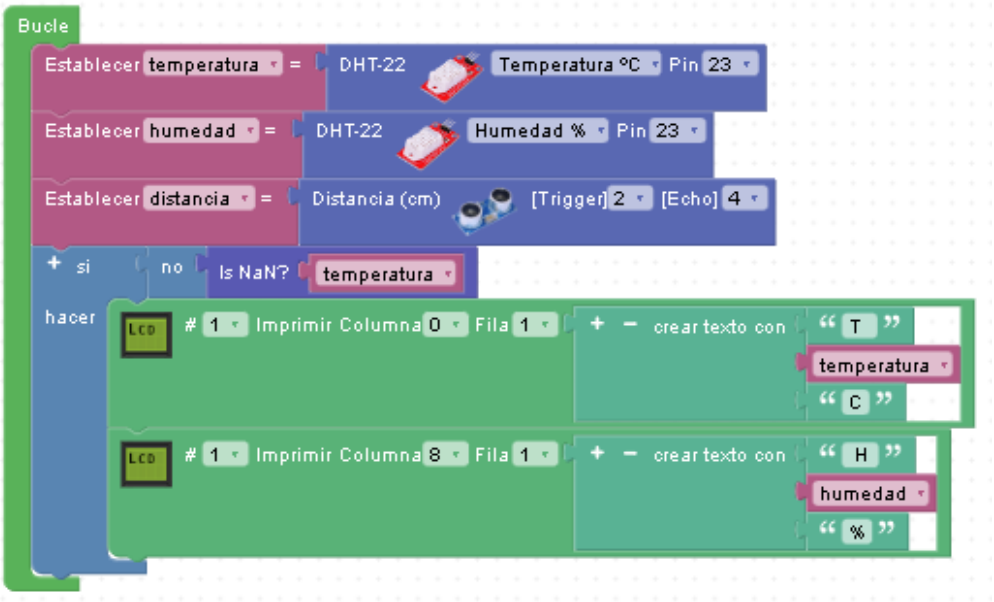


Figura 89 Mostrar datos LCD

Fuente: Elaboración propia

2.4.2.2.1 Ejecución

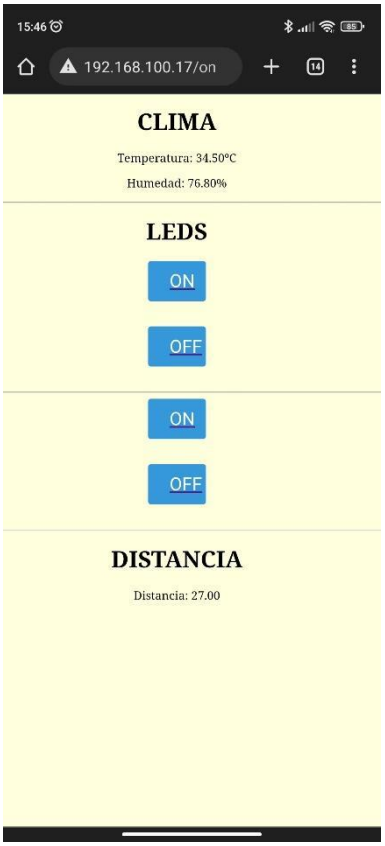


Figura 90 Ejecución servidor

Fuente: Elaboración propia

2.4.3 GRUPO 3 : COMUNICACIÓN POR BLE

2.4.3.1 ESP32

Tabla 13 Practicas Grupo 3 bluetooth ESP32

No. PRACTICA	Nombre de la practica	Objetivo
PRACTICA 1	ENCENDIDO DE LED	Encender y apagar leds a través del módulo de bluetooth
PRACTICA 2	DTH22 – SENSOR ULTRASONICO	Leer la temperatura, humedad del sensor DTH22 y la distancia del sensor ultrasónico, enviar los datos a través de bluetooth
PRÁCTICA 3	RELÉ	Encender y apagar relé a través de bluetooth

PRÁCTICA 4	INTENSIDAD DE LED	Controlar la intensidad de un led a través de bluetooth

Fuente: Elaboración propia

El código C equivalente a este grupo se encuentra en el Anexo G.

2.4.3.1.1 PRÁCTICA 1: ENCENDIDO DE LED

Objetivos: Encender y apagar leds a través del módulo de bluetooth.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 1 del grupo de periféricos.

Código Arduinoblocks

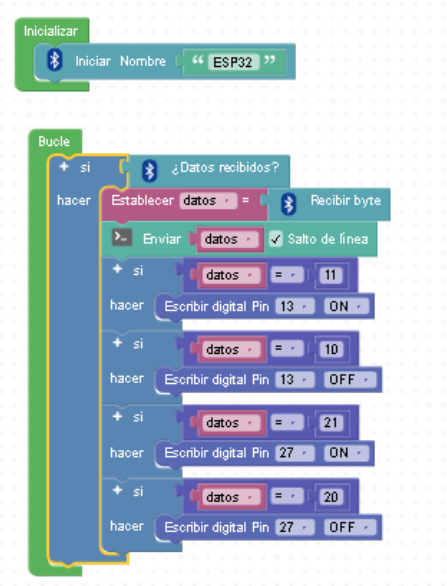


Figura 91 Código BLE Práctica 1

Fuente: Elaboración propia

2.4.3.1.2 PRÁCTICA 2: DTH22 – SENSOR ULTRASÓNICO

Objetivos: Leer la temperatura, humedad del sensor DTH22 y la distancia del sensor ultrasónico, enviar los datos a través de bluetooth.

Descripción: Para la realización de la práctica se utilizarán las mismas conexiones de la práctica 2 y 3 del grupo de periféricos.

Código Arduinoblocks

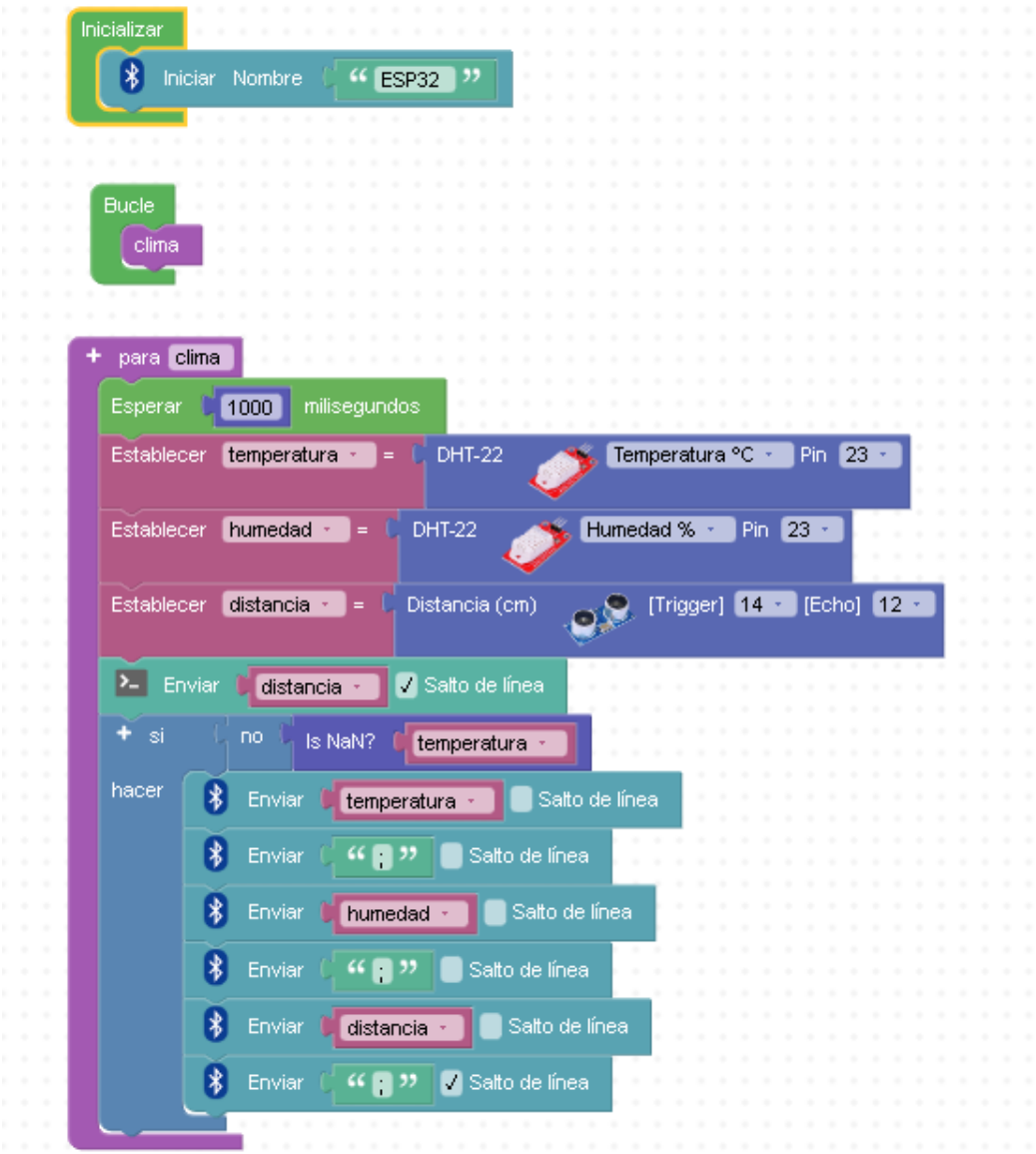


Figura 92 Código BLE Práctica 2

Fuente: Elaboración propia

2.4.3.1.3 PRÁCTICA 3: RELE

Objetivos: Encender y apagar el relé a través de bluetooth.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 5 del grupo de periféricos.

Código Arduinoblocks

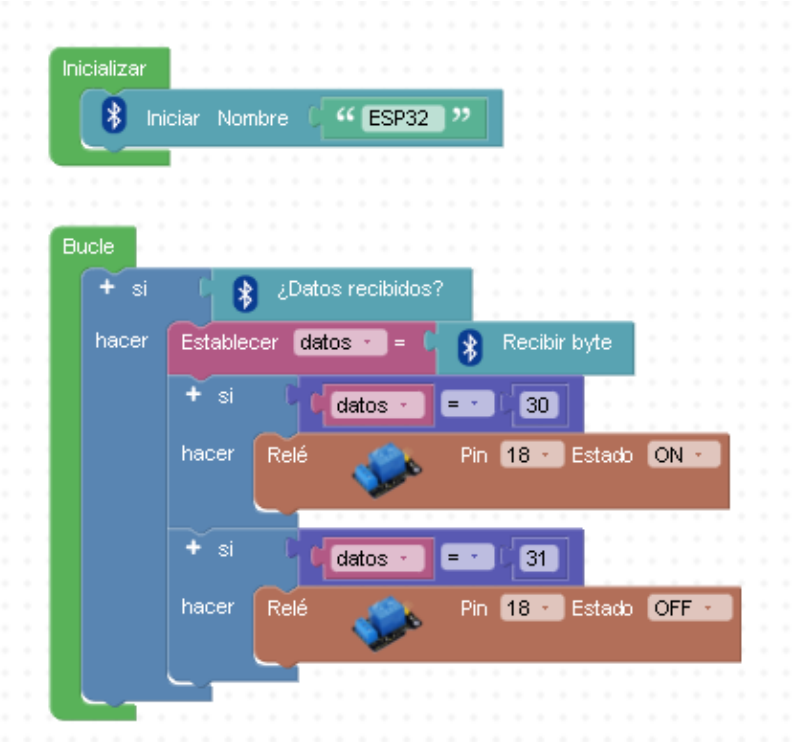


Figura 93 Código BLE Práctica 3

Fuente: Elaboración propia

2.4.3.1.4 PRÁCTICA 4: INTENSIDAD DE LED

Objetivos: Controlar la intensidad de un led a través de bluetooth.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 1 del grupo de periféricos.

Código Arduinoblocks

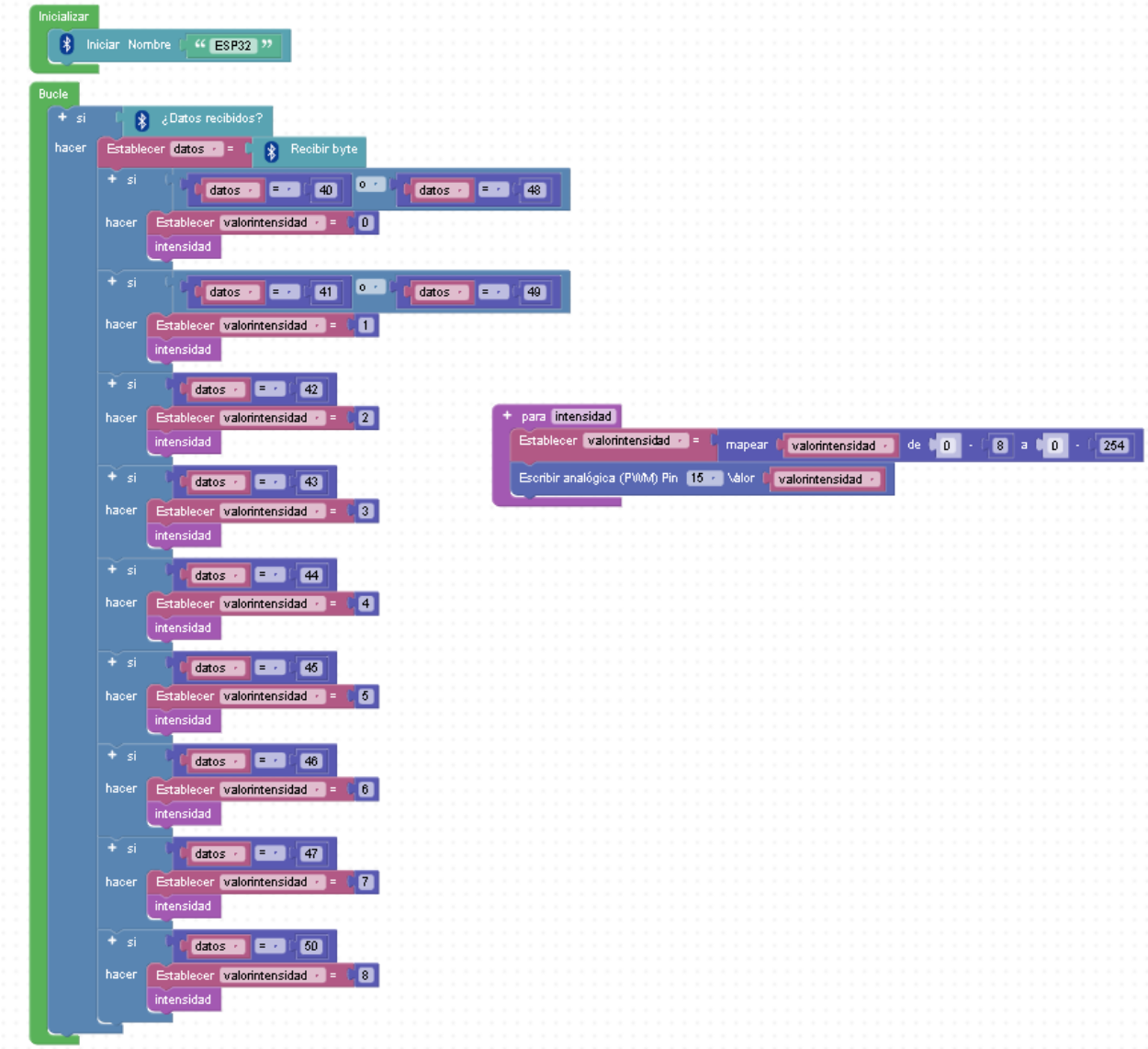


Figura 94 Código BLE Práctica 4

Fuente: Elaboración propia

2.4.3.2 MICRO:BIT

Objetivos: Crear una aplicación que permita controlar la placa MICRO:BIT mediante bluetooth.

Descripción: Para la codificación de la placa MICRO:BIT se utiliza la herramienta MakeCode es un editor online que permite la programación basada en bloques.

Tabla 14 Materiales

DESCRIPCIÓN	PRESENTACIÓN
MICRO:BIT	
Cable USB	

Fuente: Elaboración propia

En la tabla 14 se detallan los materiales a utilizar para la elaboración de las prácticas bluetooth con la microbit.

Código makecode

Para empezar a codificar la placa MICRO:BIT nos dirigimos a la plataforma de makecode.microbit.org y creamos un nuevo proyecto, le damos un nombre al proyecto.

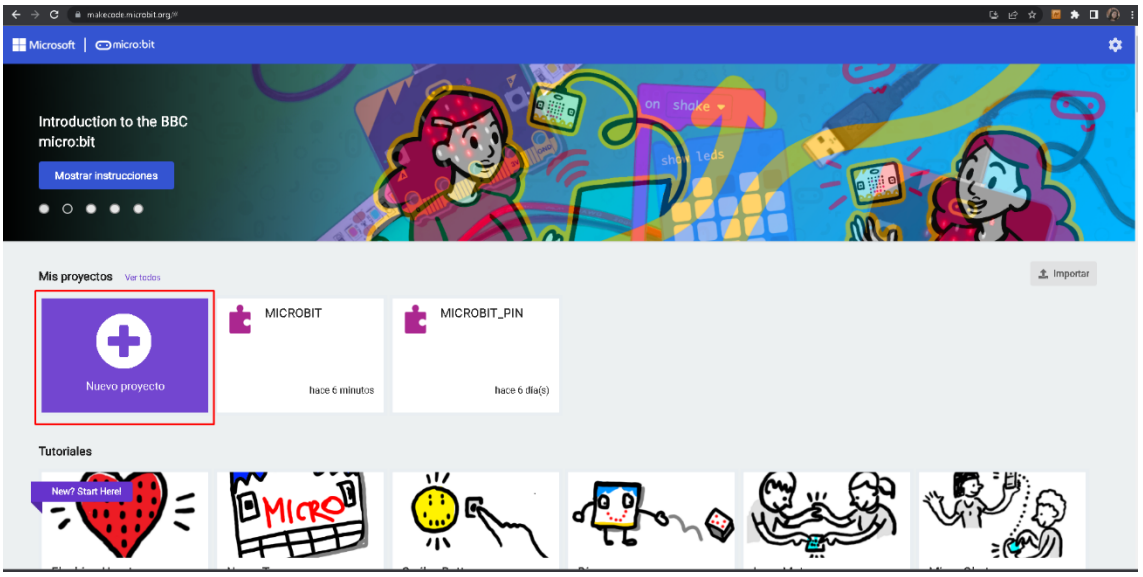


Figura 95 Crear nuevo proyecto

Fuente: Elaboración propia

Se nos mostrará la pantalla principal para empezar la programación de la placa. Para empezar a trabajar con el módulo de bluetooth debemos agregar la extensión. Para ello, nos dirigimos a la parte superior derecha en el icono de configuración, escogemos la opción **extensiones**.

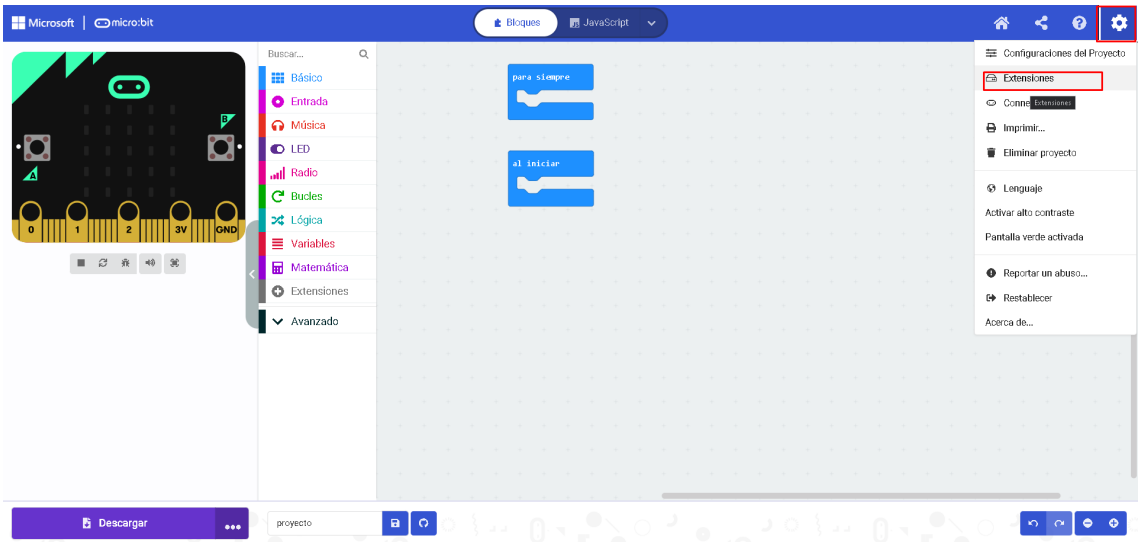


Figura 96 Insertar Extensión bluetooth

Fuente: Elaboración propia

Se mostrará una página donde podremos observar todas las extensiones disponibles para trabajar con nuestra placa, buscamos Bluetooth y damos clic sobre la extensión.

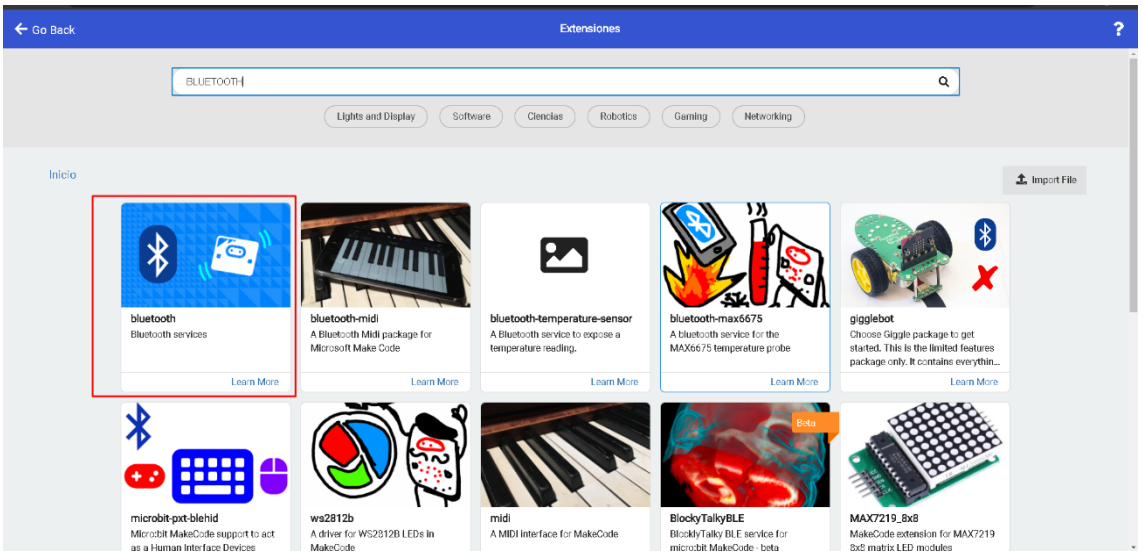


Figura 97 Extensión bluetooth

Fuente: Elaboración propia

Automáticamente se nos mostrará un mensaje, el cual nos informa que al añadir la extensión de bluetooth se desactiva la extensión de radio y se elimina ya que es incompatible con la nueva extensión. Damos clic en quitar y añadir bluetooth.

Figura 98 Insertar Extensión de bluetooth

Fuente: Elaboración propia

Una vez agregada la extensión empezamos la programación en la placa. En la sección de bloques elegimos la opción al conectar bluetooth. Este bloque es el que se mostrará cuando un dispositivo se haya conectado a la placa, mostrando el icono de visto en los leds.

Figura 99 Código de conexión a bluetooth

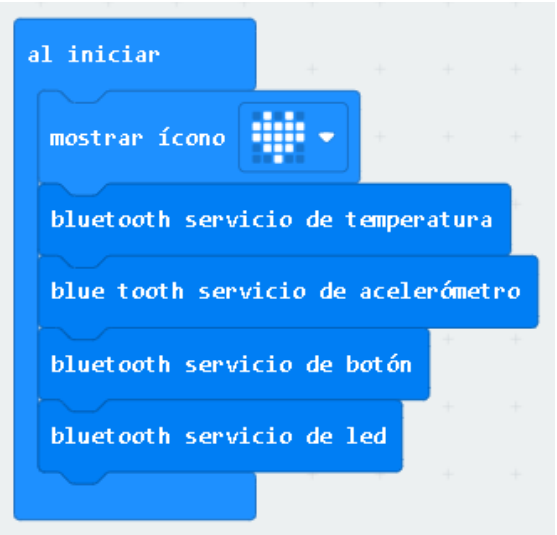
Fuente: Elaboración propia

Ahora agregamos un bloque para que muestre el icono de x cuando se haya desconectado el bluetooth.

Figura 100 Código de desconectar a bluetooth

Fuente: Elaboración propia

Al iniciar la placa se mostrará un icono de corazón en donde se muestra que el dispositivo está listo para emparejarse. Para agregar un bloque para que muestre el icono de x cuando se haya desconectado el bluetooth.




```
def on_bluetooth_connected():
    basic.show_icon(IconNames.YES)
bluetooth.on_bluetooth_connected(on_bluetooth_connected)

def on_bluetooth_disconnected():
    basic.show_icon(IconNames.NO)
bluetooth.on_bluetooth_disconnected(on_bluetooth_disconnected)

basic.show_icon(IconNames.HEART)
bluetooth.start_temperature_service()
bluetooth.start_accelerometer_service()
bluetooth.start_button_service()
bluetooth.start_led_service()
```

Figura 101 Código al iniciar MICRO:BIT

Fuente: Elaboración propia

Insertamos los servicios que vamos a utilizar vía bluetooth, el primero que insertamos es el servicio de temperatura ya que la placa posee el sensor de temperatura integrado. El segundo es el servicio de acelerómetro el cual permite

medir la fuerza en los tres ejes X, Y, Z. El tercero es el servicio de botón que permite reconocer los estados que se encuentran los botones A y B de la placa. Y por último tenemos el servicio de led la cual permite controlar los 25 led que tiene integrados la placa MICRO:BIT.

Para descargar el código a nuestra placa procedemos a conectar nuestra placa con el cable USB. Nos dirigimos a la parte inferior izquierda, elegimos la opción de Connect device.

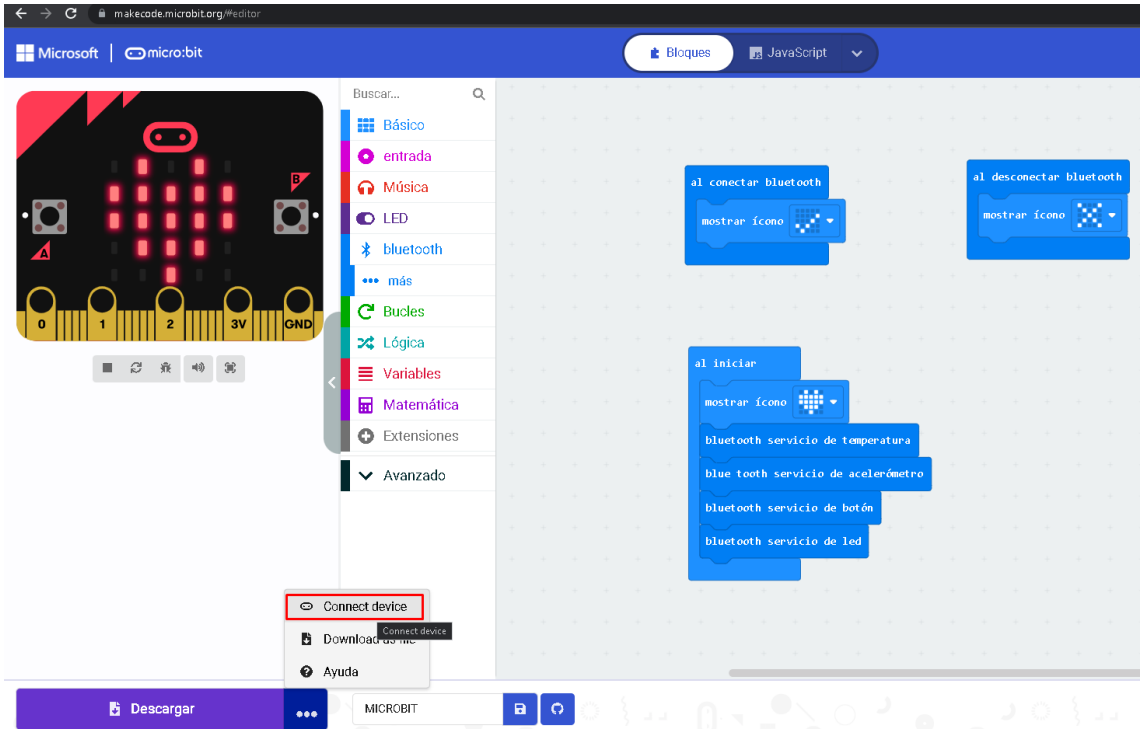


Figura 102 Conexión de la placa

Fuente: Elaboración propia

Se nos presentará unos pasos para proceder a conectar la placa. Como primer paso debemos tener previamente conectado la placa MICRO:BIT.

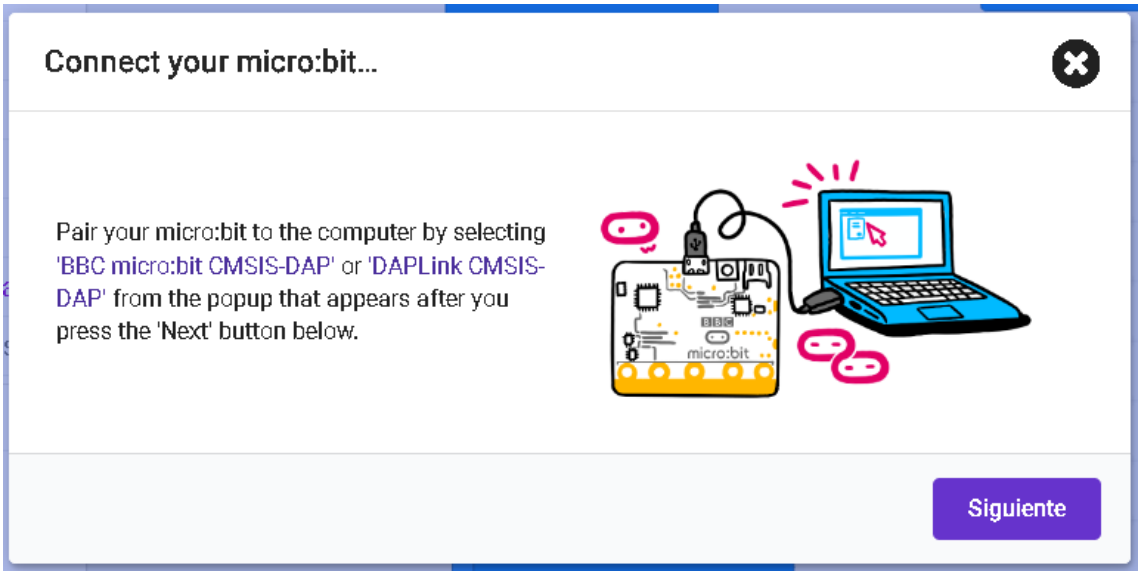


Figura 103 Conectar placa Paso 1

Fuente: Elaboración propia

La siguiente pantalla mostrará el nombre de la placa micro:bit que se encuentra conectada, seleccionamos la placa y damos click en conectar.

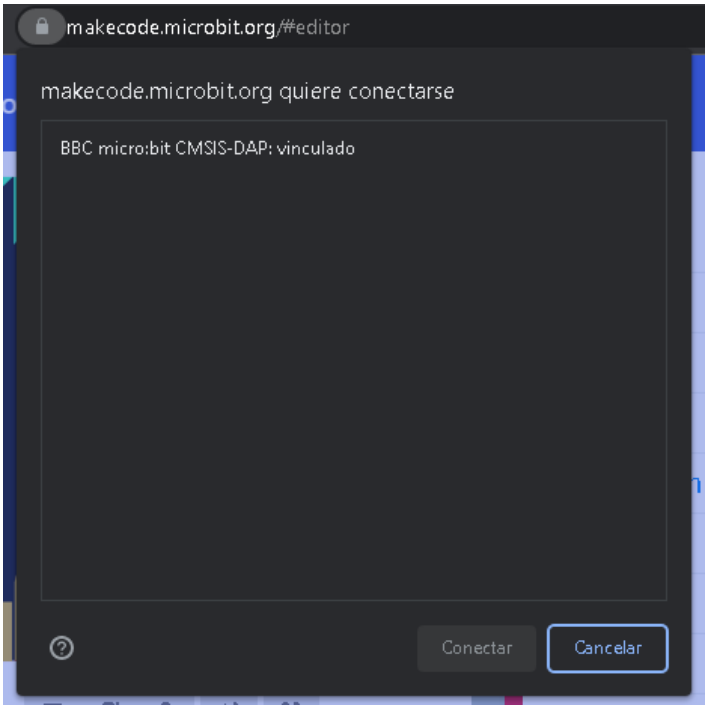


Figura 104 Seleccionar placa MICRO:BIT Paso 2

Fuente: Elaboración propia

Por último nos mostrará un mensaje que nuestra placa ha sido conectada y damos click en listo.

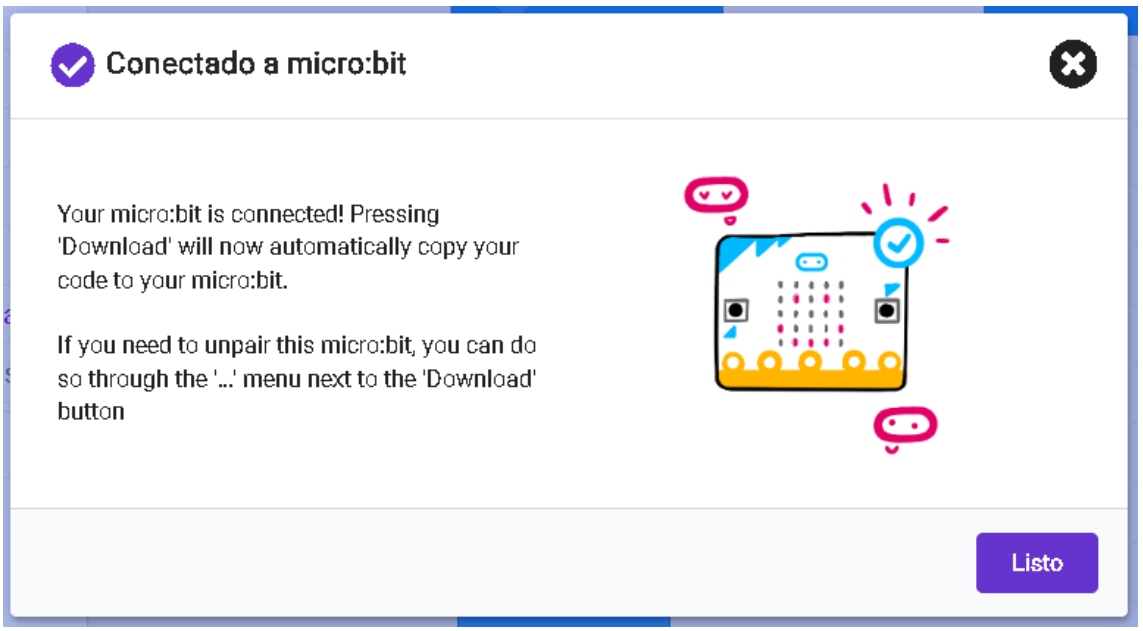


Figura 105 Placa conectada

Fuente: elaboración propia

Una vez conectada la placa solo damos clic en descargar y automáticamente se descarga nuestro código en la MICRO:BIT.

Ejecución

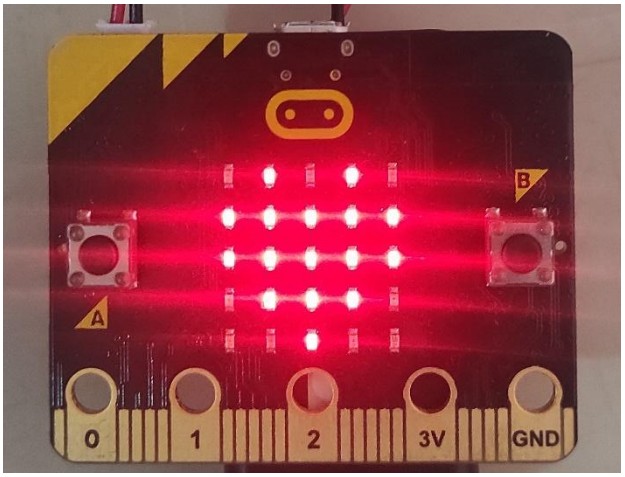


Figura 106 Ejecución

Fuente: Elaboración propia

Ahora procedemos a vincular la placa microbit al nuestro celular. Una vez que nos hayamos conectado correctamente aparecerá la ilustración de visto.

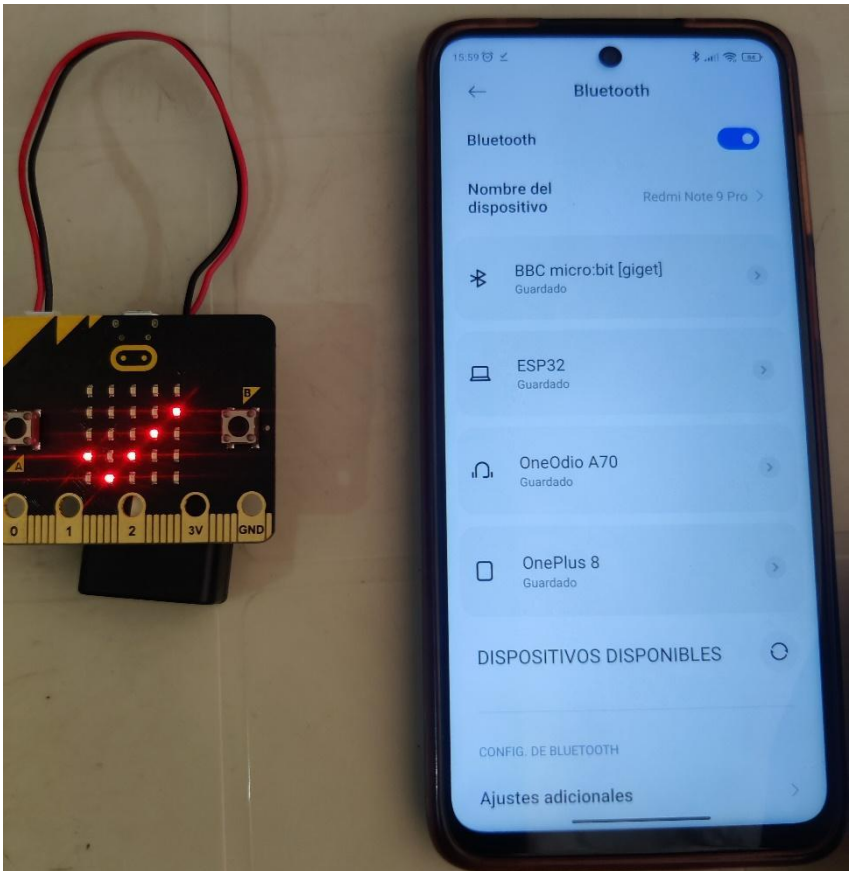


Figura 107 Vincular microbit

Fuente: Elaboración propia

2.4.4 GRUPO 4 : CELULAR

2.4.4.1 ESP32

Agregamos la extensión de bluetooth para poder hacer uso de las funciones y conectar mediante bluetooth la placa.

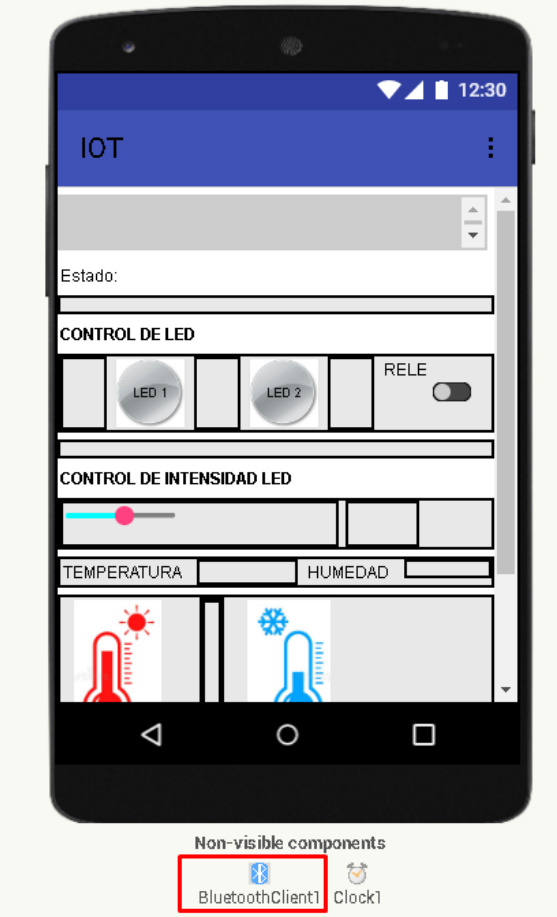


Figura 108 Extensión Bluetooth

Fuente: Elaboración propia

Primero se realizará la conexión bluetooth, para ello agregamos un listview que mostrarán los dispositivos visibles para poder conectarnos como la placa ESP32. En la parte izquierda en la sección **user interface** encontraremos los componentes con las que nos permite trabajar App inventor. Seleccionamos la opción Listview y agregamos a nuestra pantalla.

En la sección de bloques nos dirigimos al lado izquierdo donde se listan los bloques con los que podemos trabajar. Primero se mostrarán los bloques predeterminados que son una lista de 9 tipos que contienen funciones diferentes. Luego se mostrarán los componentes que hemos agregado al área de trabajo o a nuestra Screen 1. Al dar click sobre el componente de Screen 1 nos aparecerán las funciones que podemos usar con este componente. Seleccionamos la opción de **cuándo Screen 1 se inicializa** y la arrastramos al área de trabajo.

Ahora insertamos las funciones de bluetooth para poder escanear los dispositivos que están visibles y poder conectarnos. Se mostrarán los dispositivos en la listview .

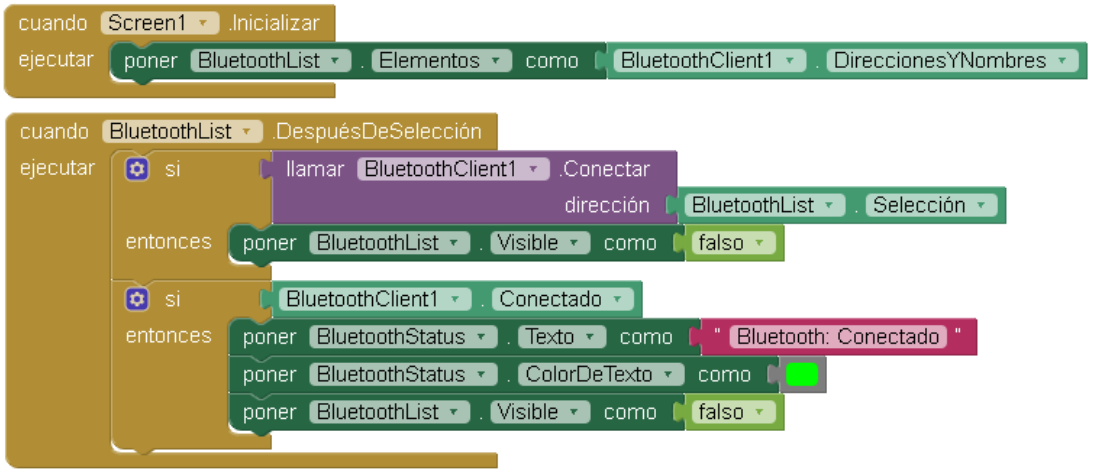


Figura 109 Código de inicializar bluetooth

Fuente: Elaboración propia

Cuando el dispositivo se haya conectado el celular al bluetooth de la placa procedemos a codificar el funcionamiento de la aplicación.

2.4.4.1.1 PRÁCTICA 1: ENCENDIDO DE LED

Objetivos: Elaborar una aplicación móvil que permite encender y apagar leds a través del módulo de bluetooth.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 1 del grupo de periféricos, se realizará el manejo de los led a través de dos botones creados en appinventor para el encendido y apagado.

App inventor

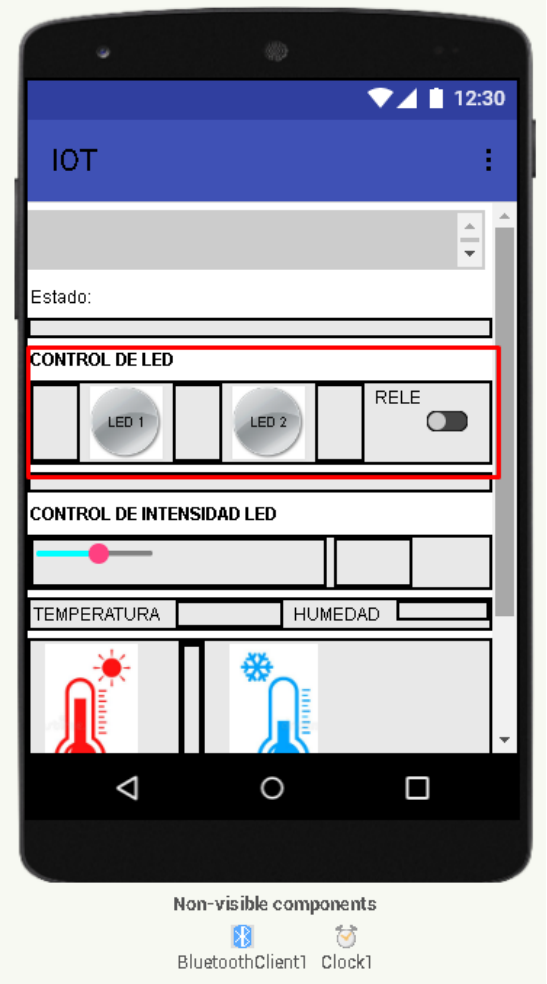


Figura 110 Interfaz gráfica Practica 1

Fuente: Elaboración propia

Código

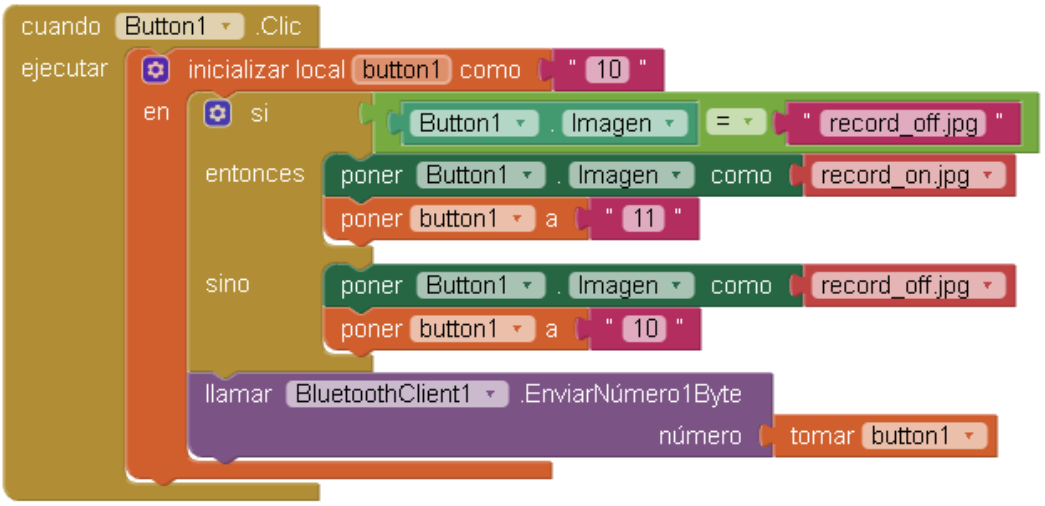


Figura 111 Código Practica 1

Fuente: Elaboración propia

Ejecución

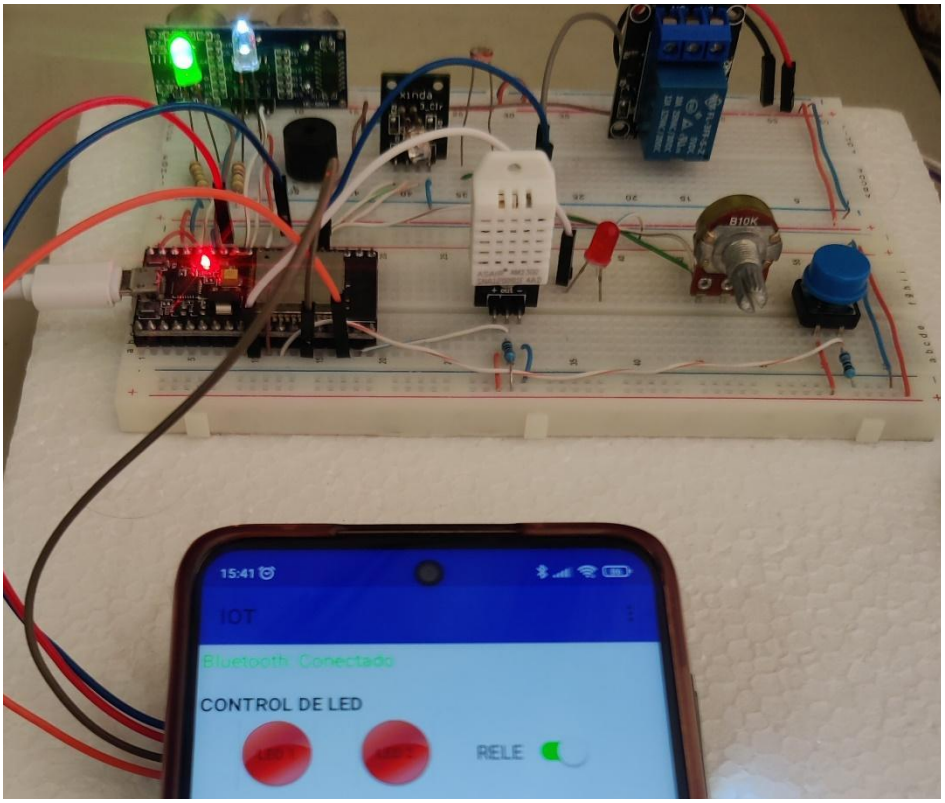


Figura 112 Ejecución práctica 1

Fuente: Elaboración propia

2.4.4.1.2 PRÁCTICA 2: DTH22 – SENSOR ULTRASÓNICO

Objetivos: Leer la temperatura, humedad del sensor DTH22 y la distancia del sensor ultrasónico, enviar los datos a través de bluetooth y mostrar los datos en la aplicación móvil.

Descripción: Para el desarrollo de la práctica se utilizarán las mismas conexiones de la práctica 2 y 3 del grupo de periféricos. Se utilizarán etiquetas para mostrar los datos obtenidos de la placa.

App inventor

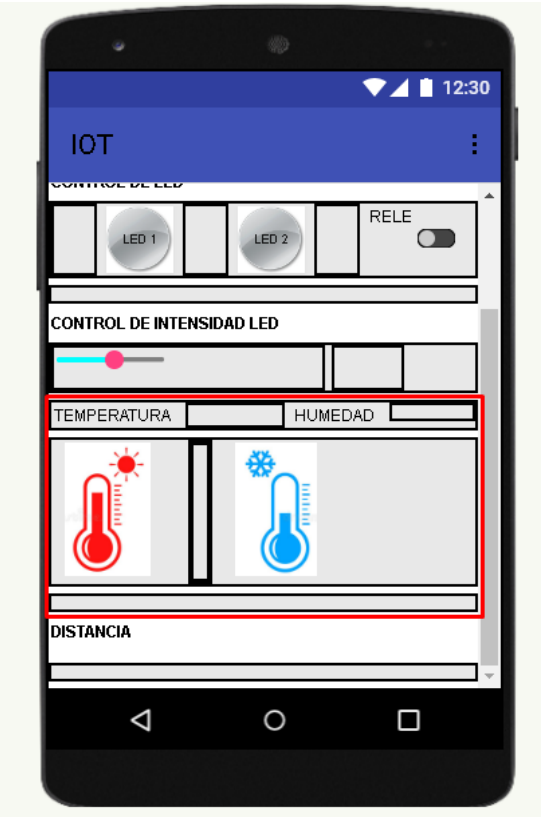


Figura 113 Interfaz gráfica Práctica 2

Fuente: Elaboración propia

Código

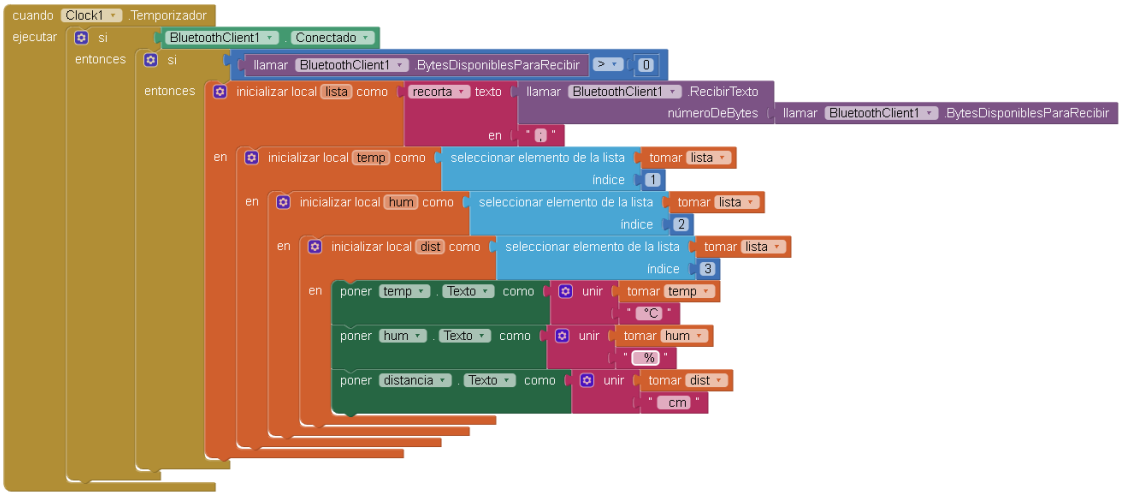


Figura 114 Código Practica 2

Fuente: Elaboración propia

Ejecución

Se mostrarán los valores del sensor ultrasónico cuando detecta un objeto cerca de la distancia establecida.

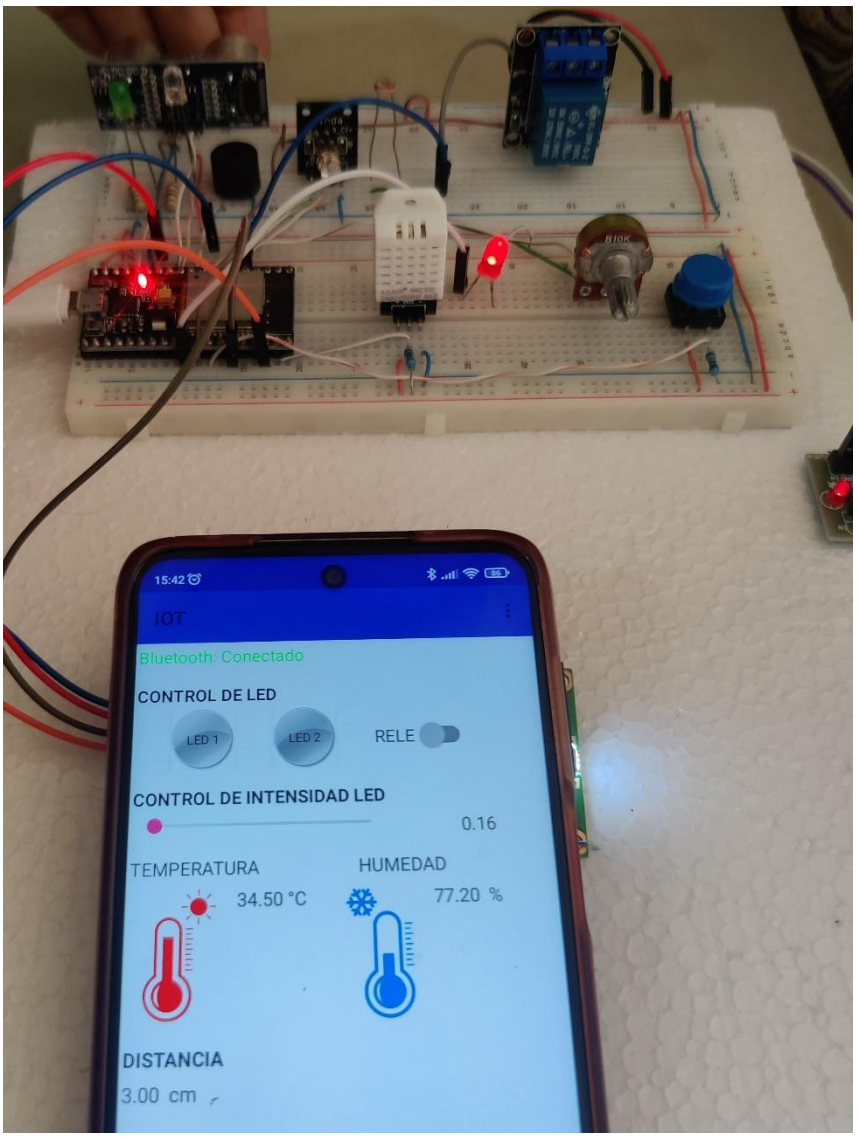


Figura 115 Ejecución práctica 2

Fuente: Elaboración propia

2.4.4.1.3 PRÁCTICA 3: RELE

Objetivos: Encender y apagar el relé a través de bluetooth mediante una aplicación móvil.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 5 del grupo de periféricos. Se utilizará un Switch para encender y apagar el relé.

App inventor

Figura 116 Interfaz gráfica Práctica 3

Fuente: Elaboración propia

Código

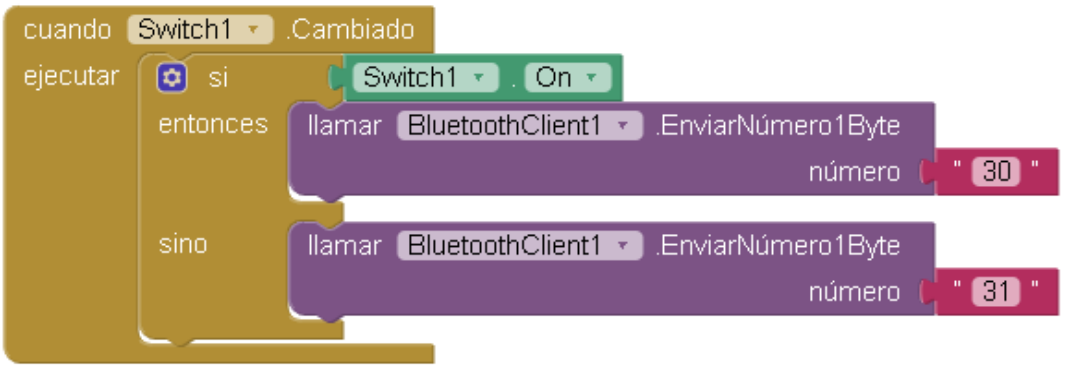


Figura 117 Código Practica 3

Fuente: Elaboración propia

Ejecución

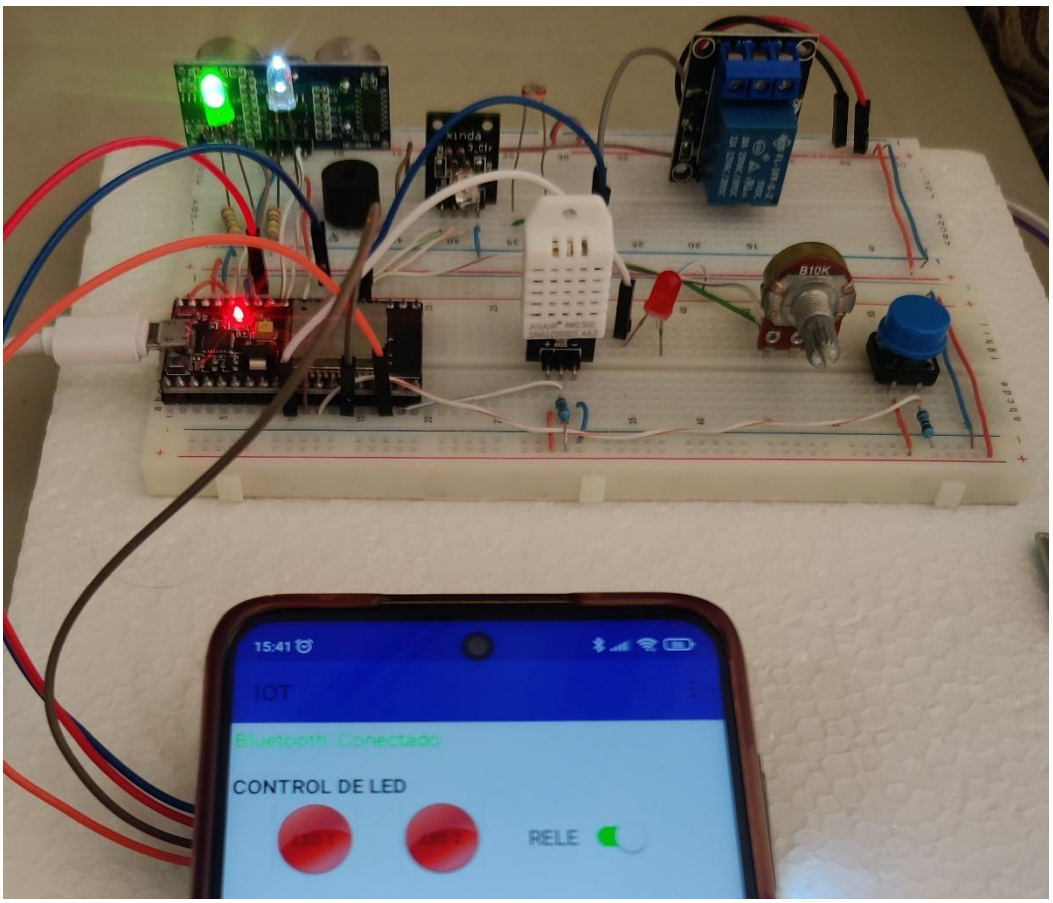


Figura 118 Ejecución de practica 3

Fuente: Elaboración propia

2.4.4.1.4 PRÁCTICA 4: INTENSIDAD DE LED

Objetivos: Controlar la intensidad de un led a través de aplicación móvil mediante bluetooth.

Descripción: Para la realización de la práctica se utilizará las mismas conexiones de la práctica 1 del grupo de periféricos. Se utilizará un Slider para establecer la intensidad de un led.

App inventor

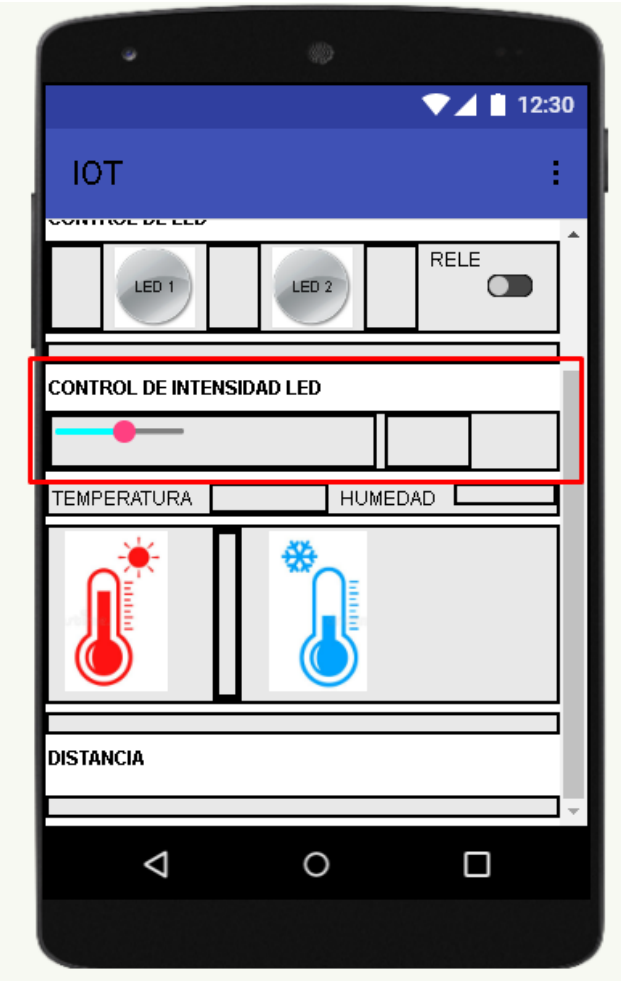


Figura 119 Interfaz gráfica Práctica 4

Fuente: Elaboración propia

Código

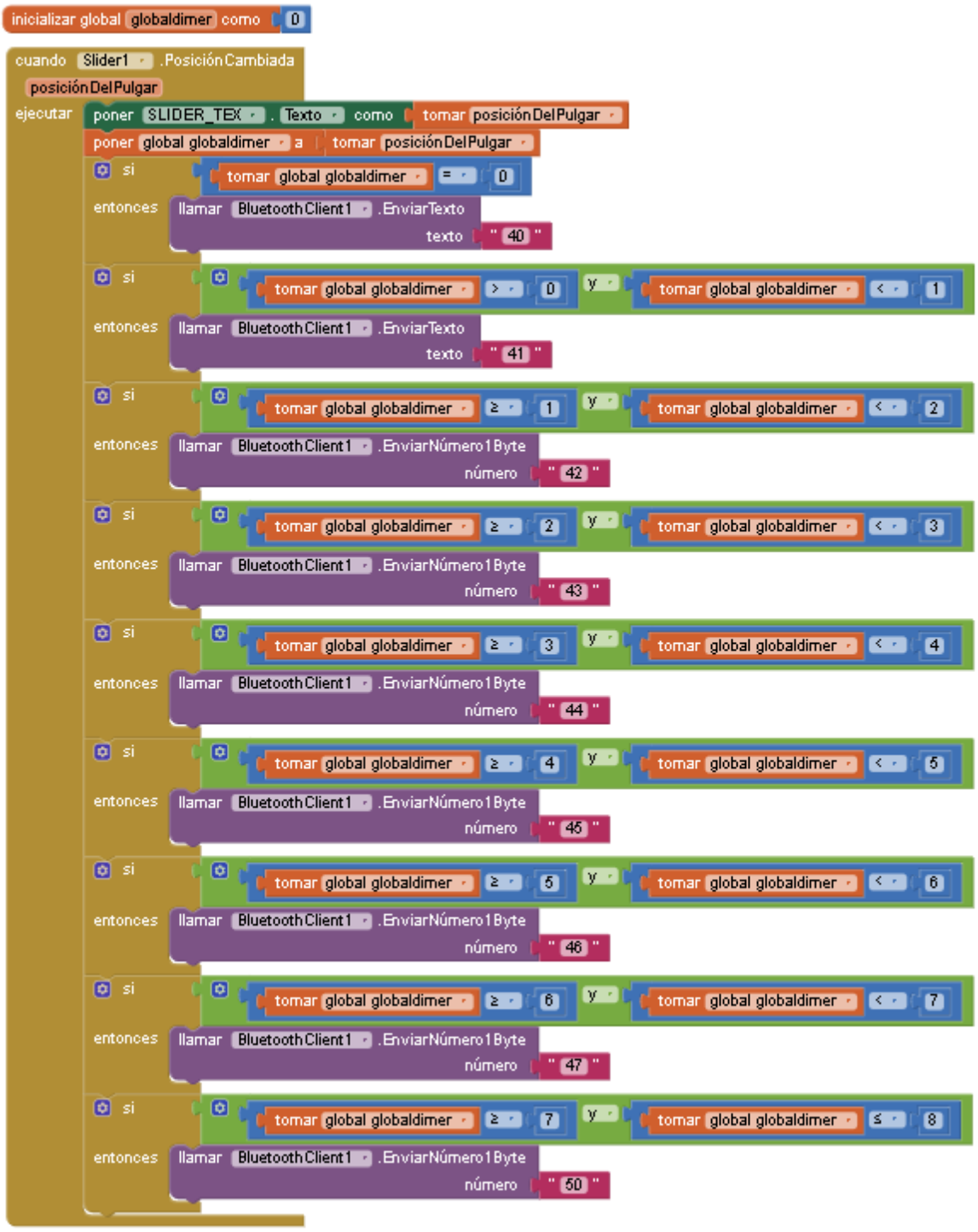


Figura 120 Código Practica 4

Fuente: Elaboración propia

Ejecución

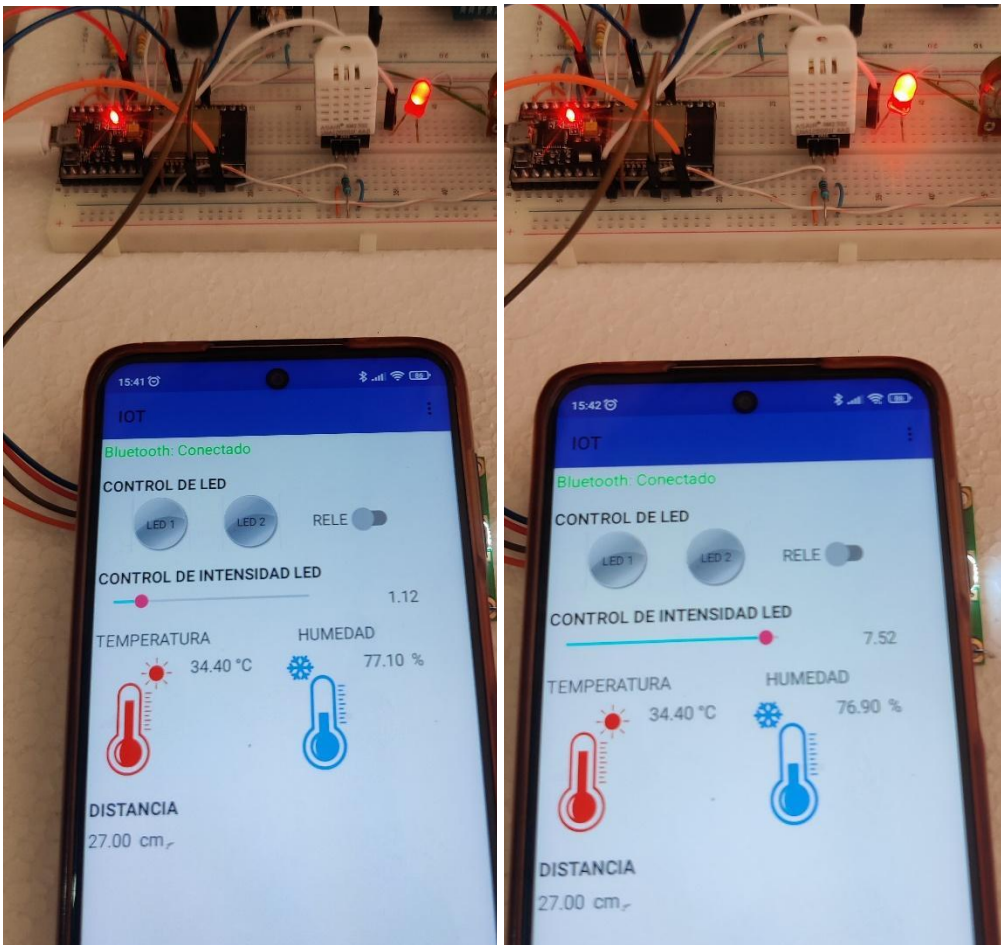


Figura 121 Ejecución práctica 4

Fuente: Elaboración propia

2.4.4.2 MICRO:BIT

Objetivos: Crear una aplicación que permita controlar la placa MICRO:BIT mediante bluetooth utilizando App Inventor.

Descripción: Para la codificación de la aplicación móvil se utiliza la herramienta App inventor que permite crear aplicaciones móviles mediante la programación basada en bloques.

- Se creará una aplicación móvil que permite controlar los servicios inicializados en la microbit en la práctica del grupo 3 utilizando la extensión de microbit que posee app inventor.
- Mediante el uso de una etiqueta se mostrará el valor de la temperatura proporcionada por el servicio de temperatura.
- A través de etiquetas se mostrarán los valores que tengan los ejes X,Y,Z al mover la placa con el servicio de acelerómetro.

- Mediante el uso de check list se encenderán los led que posee la placa de microbit a través del servicio de led.
- Se mostrará el estado de los botones de la placa microbit proporcionado a través del servicio de botón.

A continuación se detallan las prácticas que se elaboran con esta placa.

Tabla 15 Prácticas Grupo 4 MICRO:BIT

No. PRACTICA	Nombre de la practica	Objetivo
PRACTICA 1	LECTURA DE TEMPERATURA	Mostrar los datos del sensor de temperatura de la placa microbit a través de la aplicación móvil.
PRACTICA 2	LECTURA DE DATOS DEL ACELERÓMETRO	Mostrar los datos obtenidos del servicio de acelerómetro de la microbit a través de la aplicación móvil.
PRÁCTICA 3	LECTURA DEL ESTADO DE LOS BOTONES A Y B	Mostrar el estado de los botones de la placa microbit mediante la aplicación móvil.
PRÁCTICA 4	CONTROL DE LEDS	Crear una aplicación móvil que permita el control de los leds integrados en la placa microbit.
PRÁCTICA 5	ENCENDIDO Y APAGADO DE LEDS	Crear una aplicación móvil que permite encender y apagar un led utilizando el pin de la microbit a través de bluetooth.

Fuente: Elaboración propia

Extensiones App Inventor

Para trabajar con la App inventor y poder tener acceso a placa debemos agregar la extensión de micro:bit y bluetooth BLE.

Paso 1: Damos clic derecho y copiamos el enlace que está en la página oficial de App inventor.

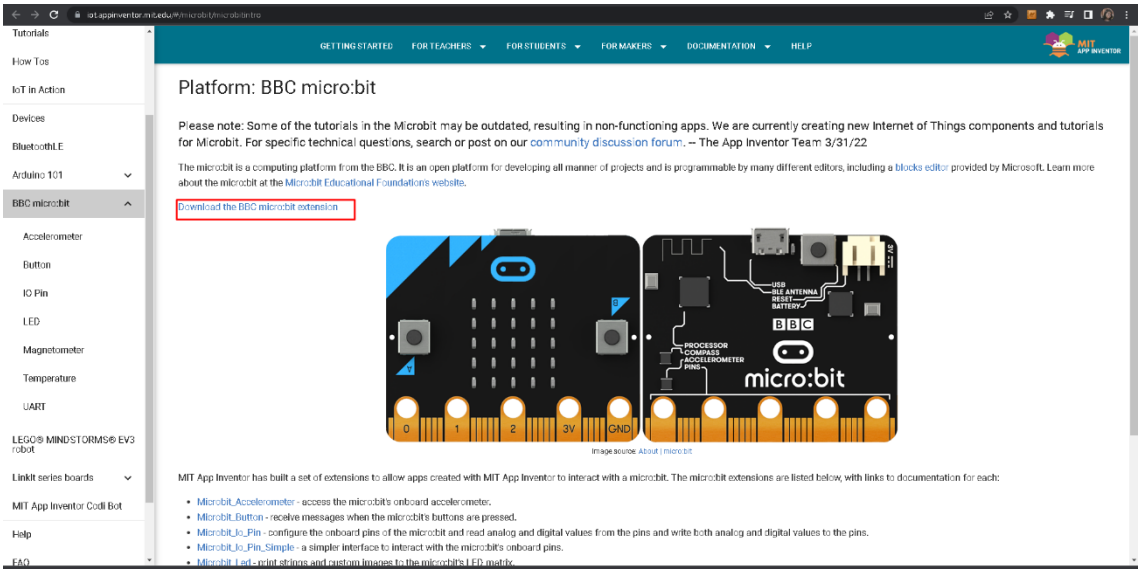


Figura 122 Extensión MICRO:BIT

Fuente: Elaboración propia

Paso 2: Nos dirigimos a App inventor en la opción extensión damos clic en import extensión.

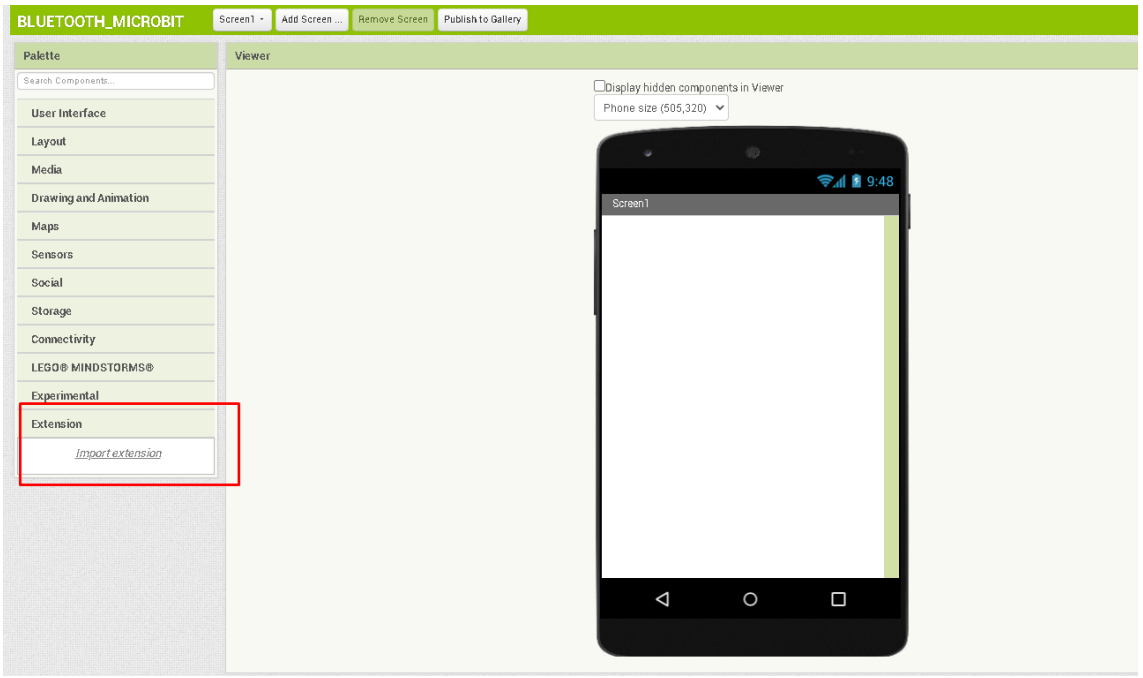


Figura 123 Importar extensión al App inventor

Fuente: Elaboración propia

Paso 3: Seleccionamos la opción URL y pegamos el enlace. Damos click en import.

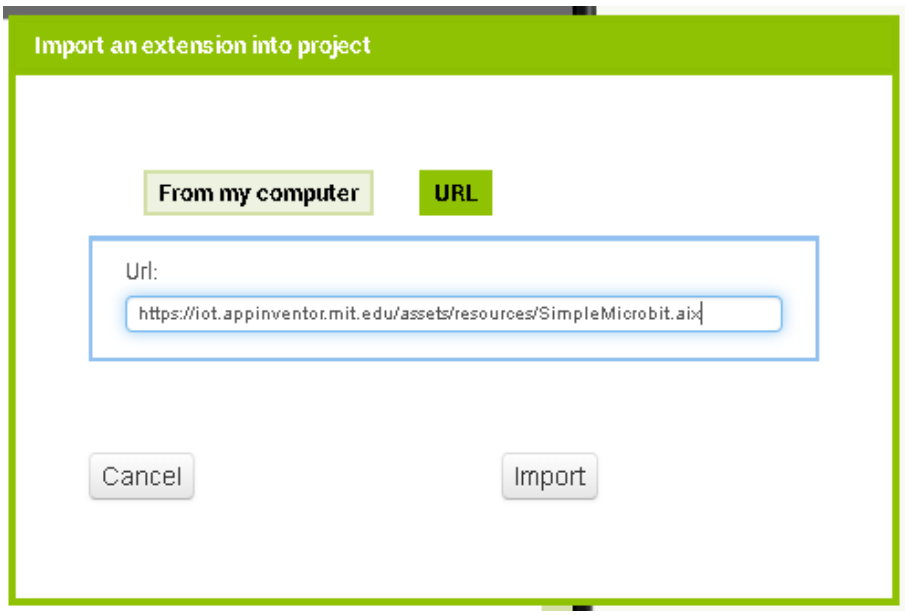


Figura 124 Agregar URL de extensión

Fuente: Elaboración propia

Paso 4: Se nos mostrará las 8 extensiones con las que podemos trabajar para controlar la placa micro:bit desde App inventor.

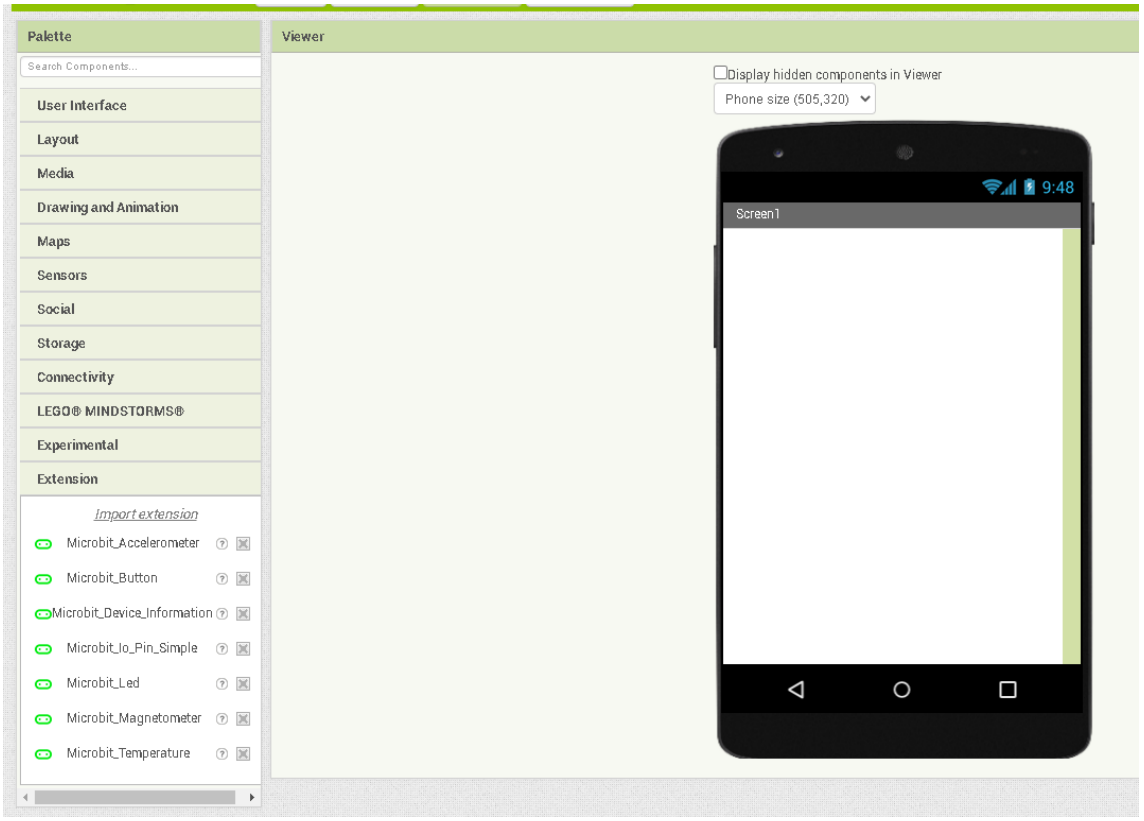


Figura 125 Extensión agregada

Fuente: Elaboración propia

Paso 5: Realizamos los mismos pasos para agregar la extensión de bluetooth.

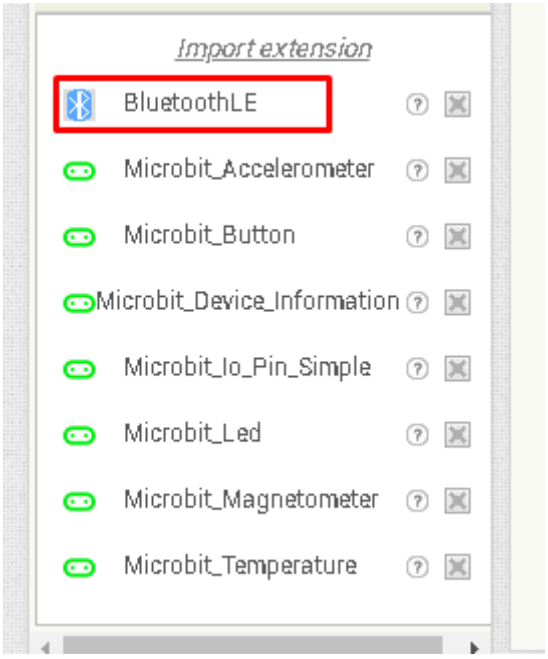


Figura 126 Extensión de Bluetooth agregada

Fuente: Elaboración propia

Una vez agregadas las extensiones que se van a utilizar procedemos a realizar la aplicación móvil. Agregamos la extensión de bluetooth para poder hacer uso de las funciones y conectar mediante bluetooth la placa.

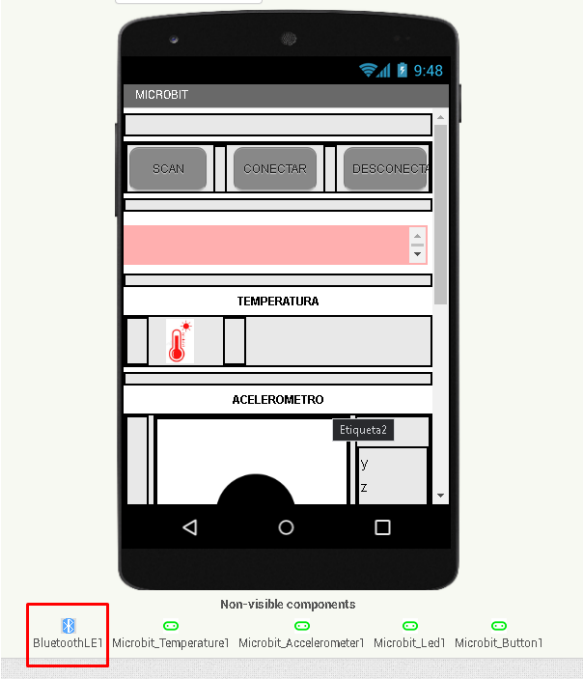


Figura 127 Extensión Bluetooth

Fuente: Elaboración propia

Primero se realizará la conexión bluetooth, para ello agregamos 3 botones los cuales serán: Scan, Conectar y Desconectar. En la parte izquierda en la sección **user interface** encontraremos los componentes con las que nos permite trabajar App inventor. Seleccionamos la opción botones y colocamos tres botones.

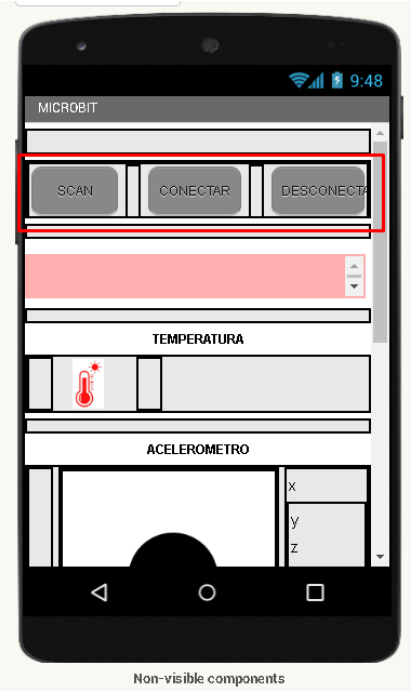


Figura 128 Botones conexión bluetooth

Fuente: Elaboración propia

Para el botón de SCAN, en la sección de bloques nos dirigimos al lado izquierdo donde se listan los bloques con los que podemos trabajar. Primero se mostrarán los bloques predeterminados que son una lista de 9 tipos que contienen funciones diferentes. Luego se mostrarán los componentes que hemos agregado al área de trabajo o a nuestra Screen 1. Al dar click sobre el botón de scan nos aparecerán las funciones que podemos usar con este componente. Seleccionamos la opción de **cuando damos clic** y la arrastramos al área de trabajo.

Ahora insertamos las funciones de bluetooth para poder escanear los dispositivos que están visibles y poder conectarnos. Se mostrarán los dispositivos en una listview y al estar conectado se llamarán los métodos de temperatura, acelerómetro y botón para poder mostrar los datos obtenidos de la placa.

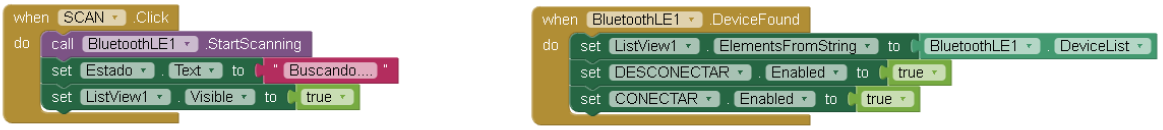


Figura 129 Código de botón scan

Fuente: Elaboración propia

Repetimos los mismos pasos para el botón de conectar, insertar la opción de **cuando damos clic** y ahora colocamos la función de **Stop Scanning** que detiene el escaneado de los dispositivos. **Connect index** toma el dispositivo que está seleccionado de la lista y se conecta al dispositivo previamente seleccionado de la lista.

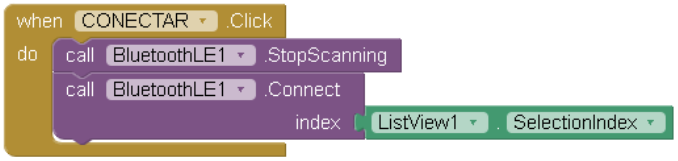


Figura 130 Código botón conectar

Fuente: Elaboración propia

Para el botón de desconectar, insertamos la opción de **cuando damos clic** y ahora colocamos la función de bluetooth **Disconnect** que desconecta el dispositivos y se procede a bloquear los botones de Conectar y Desconectar.

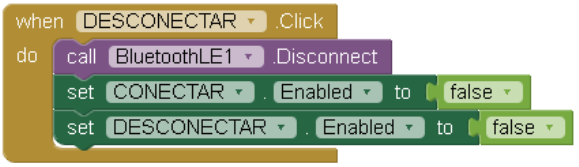


Figura 131 Código botón desconectar

Fuente: Elaboración propia

Cuando el dispositivo se haya conectado el celular al bluetooth de la placa procedemos a codificar el funcionamiento de la aplicación cuando esté conectada. Para ello seleccionamos en el componente bluetooth la opción de **cuando la placa esté conectada** y mostramos a través de una etiqueta que el dispositivo está conectado, luego procedemos a llamar a los datos de la placa para publicarlos a través de etiquetas. Es importante tener agregados nuestros componentes de la extensión microbit para poder hacer el llamado de temperatura, acelerómetro, botón.

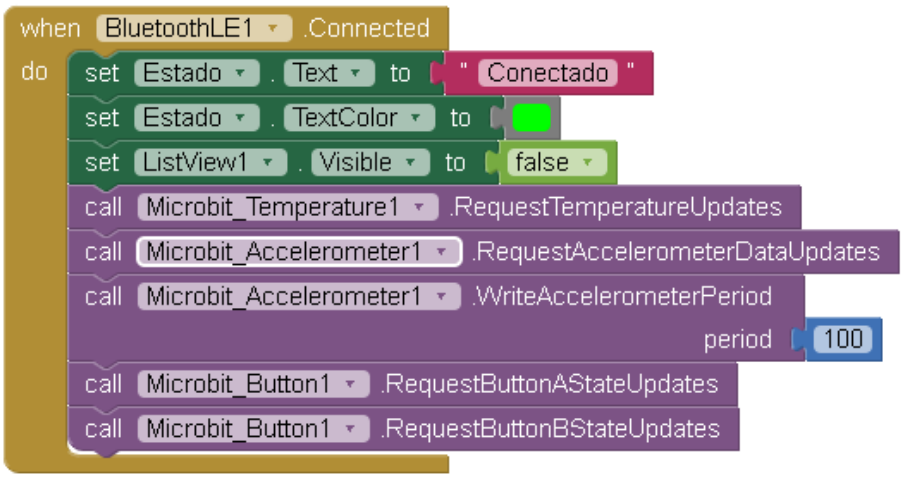


Figura 132 Código Bluetooth conectado

Fuente: Elaboración propia

Al insertar los componentes de la extensión de microbit es importante dirigirse a propiedades y seleccionar el bluetooth que hemos agregado previamente como se muestra a continuación.

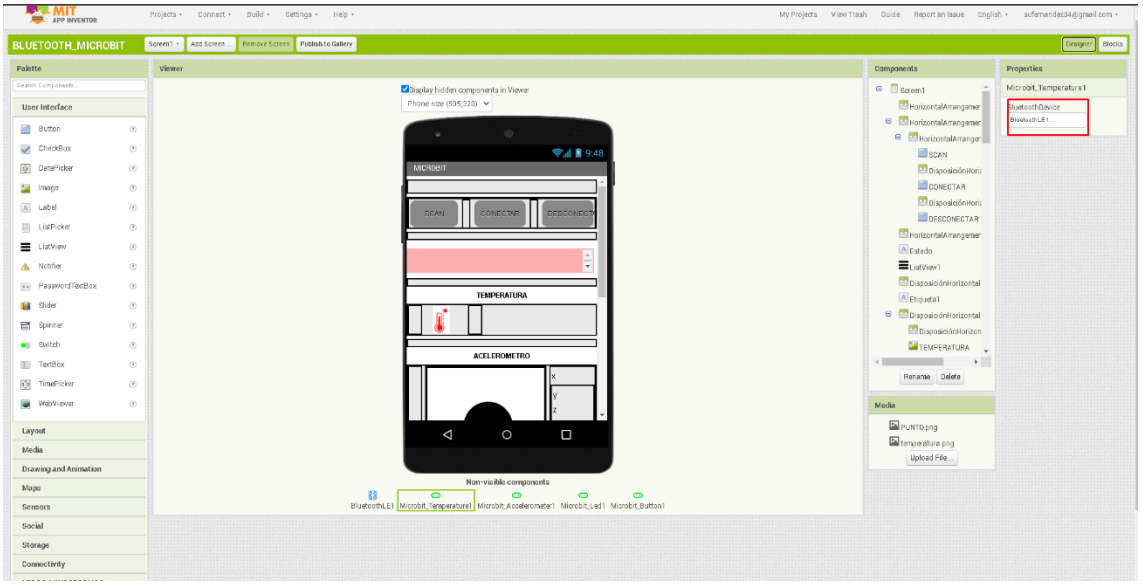


Figura 133 Configuración de extensiones microbit

Fuente: Elaboración propia

2.4.4.2.1 PRÁCTICA 1: LECTURA DE TEMPERATURA

Objetivos: Mostrar los datos del sensor de temperatura de la placa microbit a través de la aplicación móvil.

Descripción: Para la realización de la práctica se utilizará la herramienta app inventor. Mediante una etiqueta se mostrarán los valores obtenidos de la microbit.

App inventor

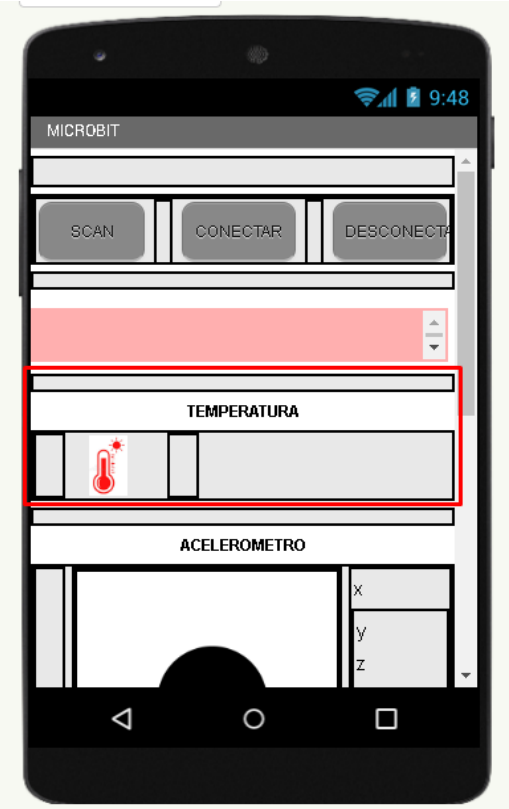


Figura 134 Interfaz gráfica Practica 1

Fuente: Elaboración propia

Código

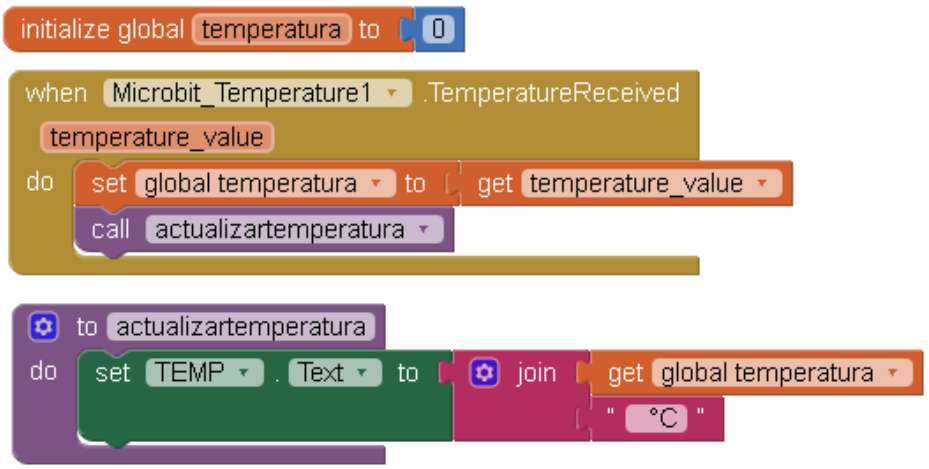


Figura 135 Código appinventor Práctica 1

Fuente: Elaboración propia

Ejecución

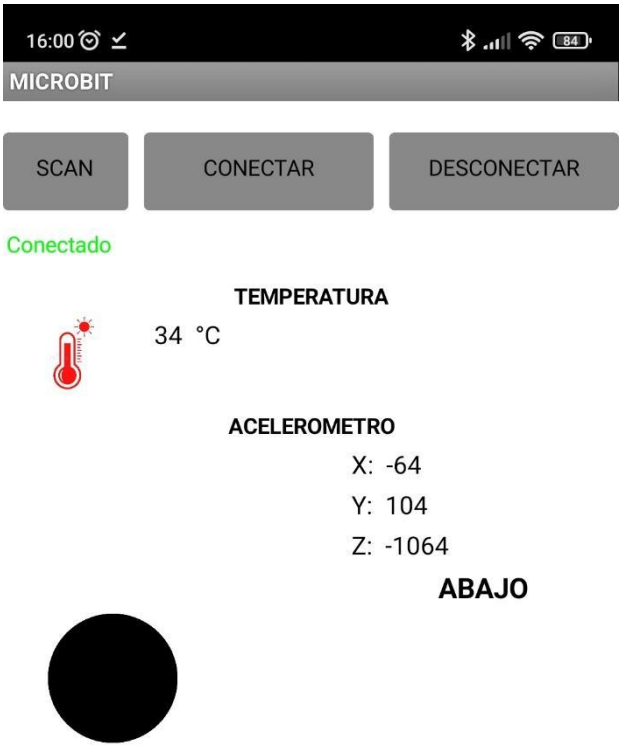


Figura 136 Ejecución práctica 1

Fuente: Elaboración propia

2.4.4.2.2 PRÁCTICA 2: LECTURA DE DATOS DEL ACELERÓMETRO

Objetivos: Mostrar los datos obtenidos del servicio de acelerómetro de la microbit a través de la aplicación móvil.

Descripción: Para la realización de la práctica se utilizará la herramienta app inventor. Mediante una etiqueta se mostrarán los valores X,Y,Z obtenidos de la microbit. Se utilizará una imagen con movimiento para simular los movimientos de acuerdo a los datos de las variables X,Y,Z.

App inventor

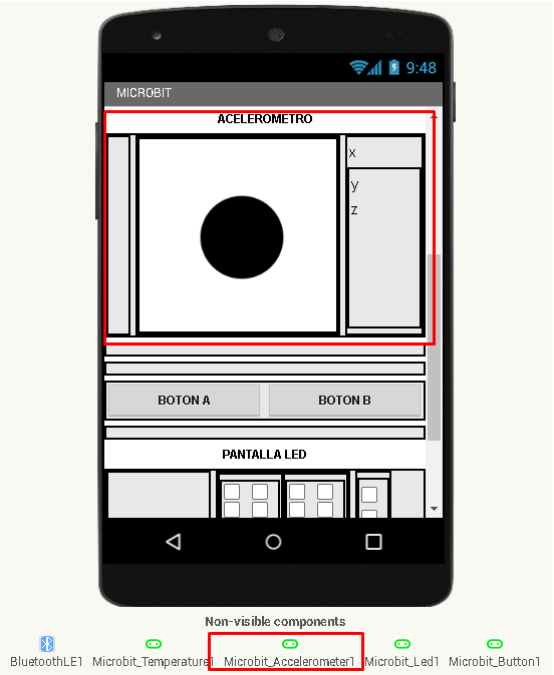


Figura 137 Interfaz gráfica Práctica 2

Fuente: Elaboración propia

Código

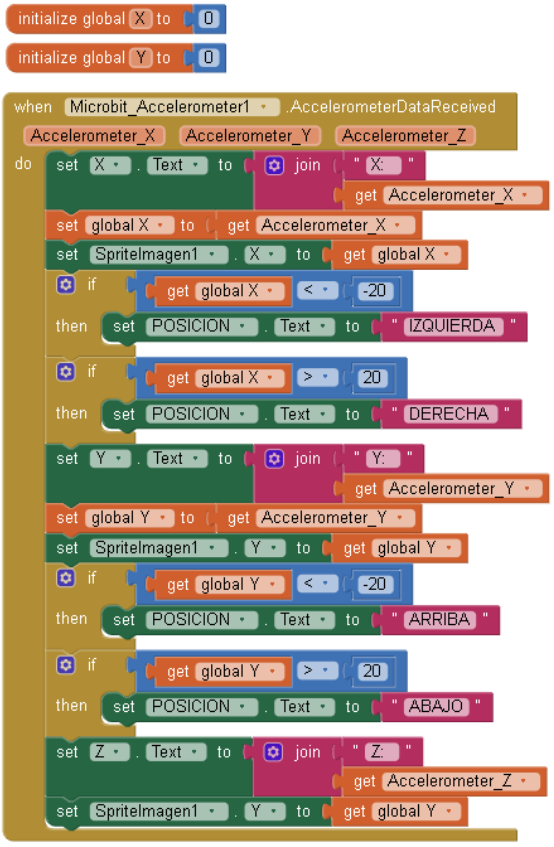


Figura 138 Código appinventor Práctica 2

Fuente: Elaboración propia

Ejecución

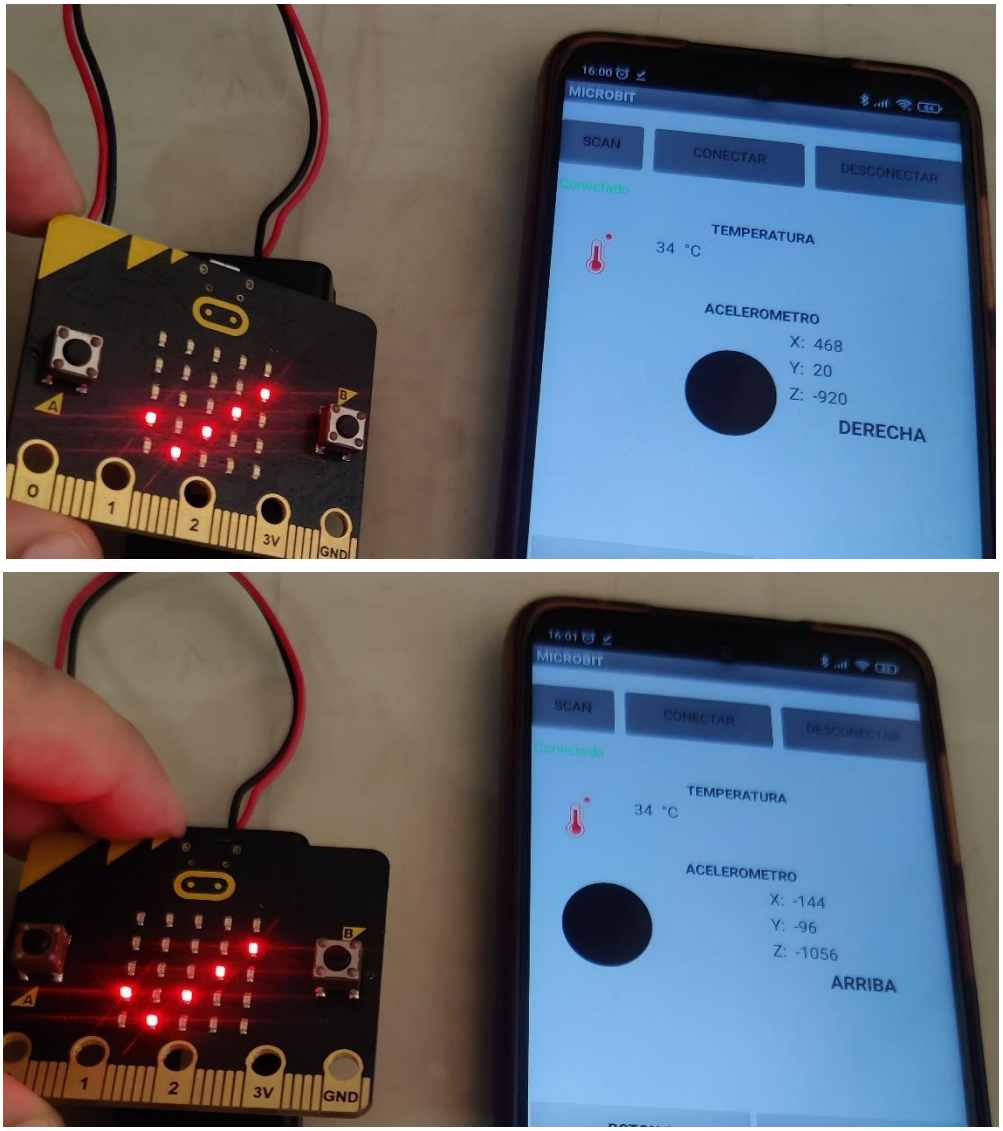


Figura 139 Ejecución práctica 2

Fuente: Elaboración propia

2.4.4.2.3 PRÁCTICA 3: LECTURA DEL ESTADO DE LOS BOTONES A Y B

Objetivos: Mostrar el estado de los botones de la placa microbit mediante la aplicación móvil.

Descripción: Para la realización de la práctica se utilizará la herramienta app inventor. Mediante 2 botones se mostrará el estado en que se encuentran los botones A y B de la placa. Cuando el valor recibido del botón es 0 se mostrara de color plomo, cuando el valor es 1 porque ha sido presionado mostraremos el

background del botón de color verde y cuando es 2 porque está siendo presionado de forma continua se mostrara de color rojo.

App inventor

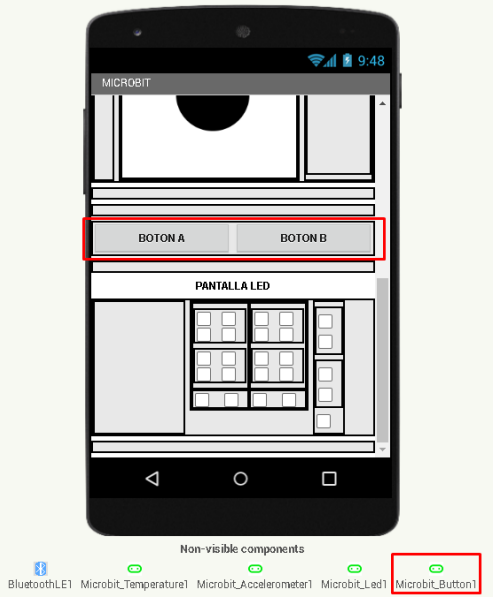


Figura 140 Interfaz gráfica Práctica 3

Fuente: Elaboración propia

Código

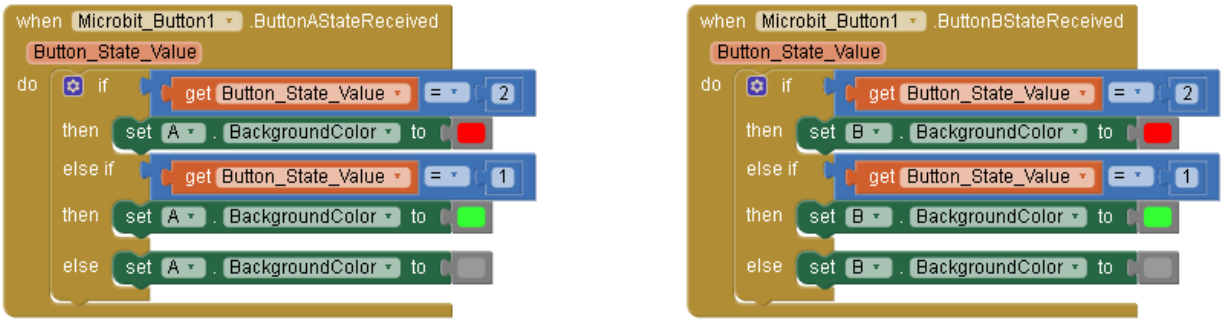


Figura 141 Código appinventor Práctica 3

Fuente: Elaboración propia

Ejecución

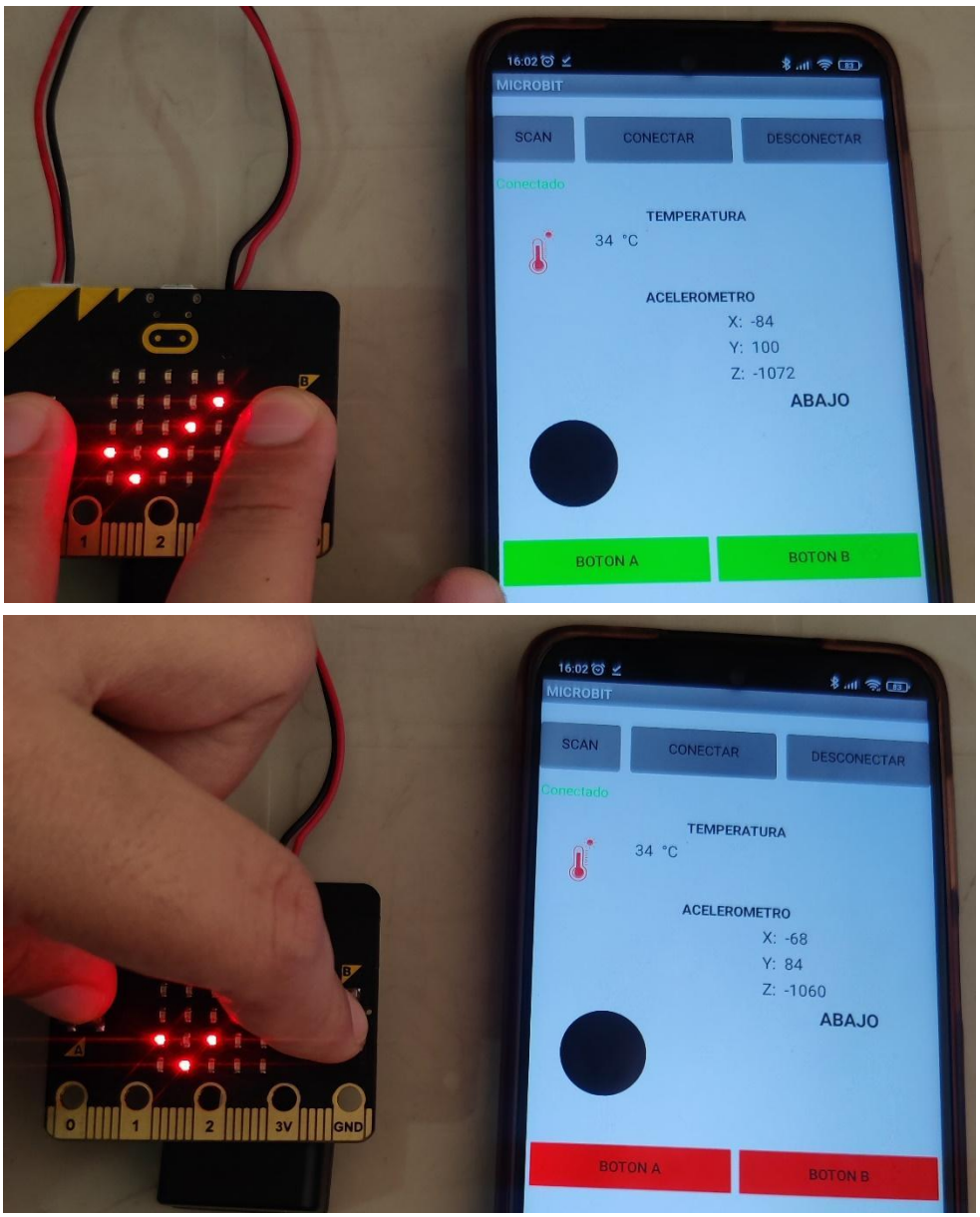


Figura 142 Ejecución práctica 3

Fuente: Elaboración propia

2.4.4.2.3 PRÁCTICA 4: CONTROL DE LEDS

Objetivo: Crear una aplicación móvil que permita el control de los leds integrados en la placa microbit.

Descripción: Para la realización de la práctica se utilizará la herramienta app inventor. Mediante el uso de 25 checkbox se utilizará para encender el led correspondiente.

App inventor

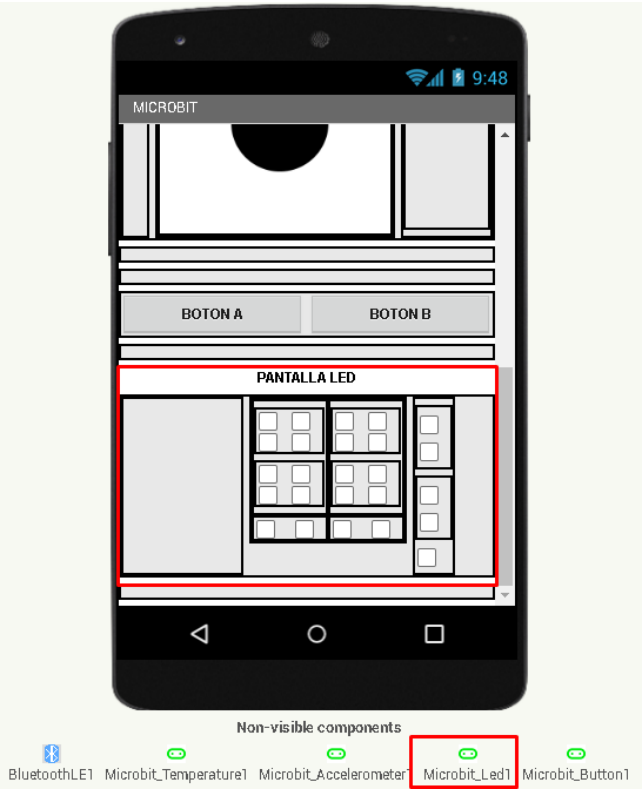


Figura 143 Interfaz gráfica Práctica 4

Fuente: Elaboración propia

Código

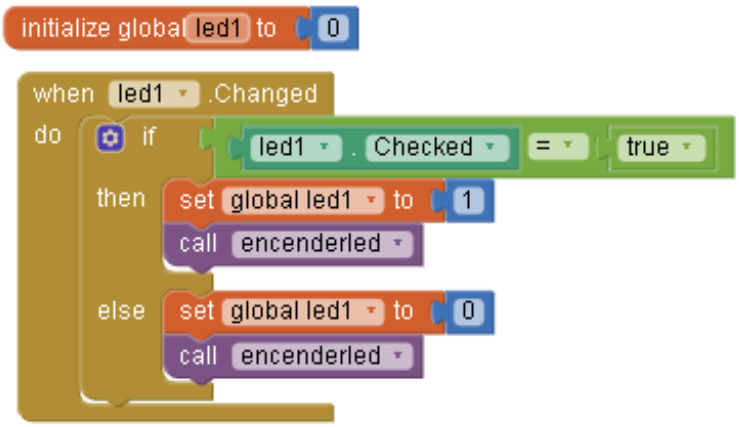


Figura 144 Código appinventor Práctica 4

Fuente: Elaboración propia

Para el código del encendido y apagado de led se utilizaron 25 checkbox y variables desde led1 hasta led 25 en donde cada checkbox al ser habilitado almacenará en la variable led un 1 y 0 al ser deshabilitado. Se creó una función llamada encender led

para enviar los datos en el método LED_Matrix_State y se convirtió las variables en base 10 para que pueda ser leído por la microbit.

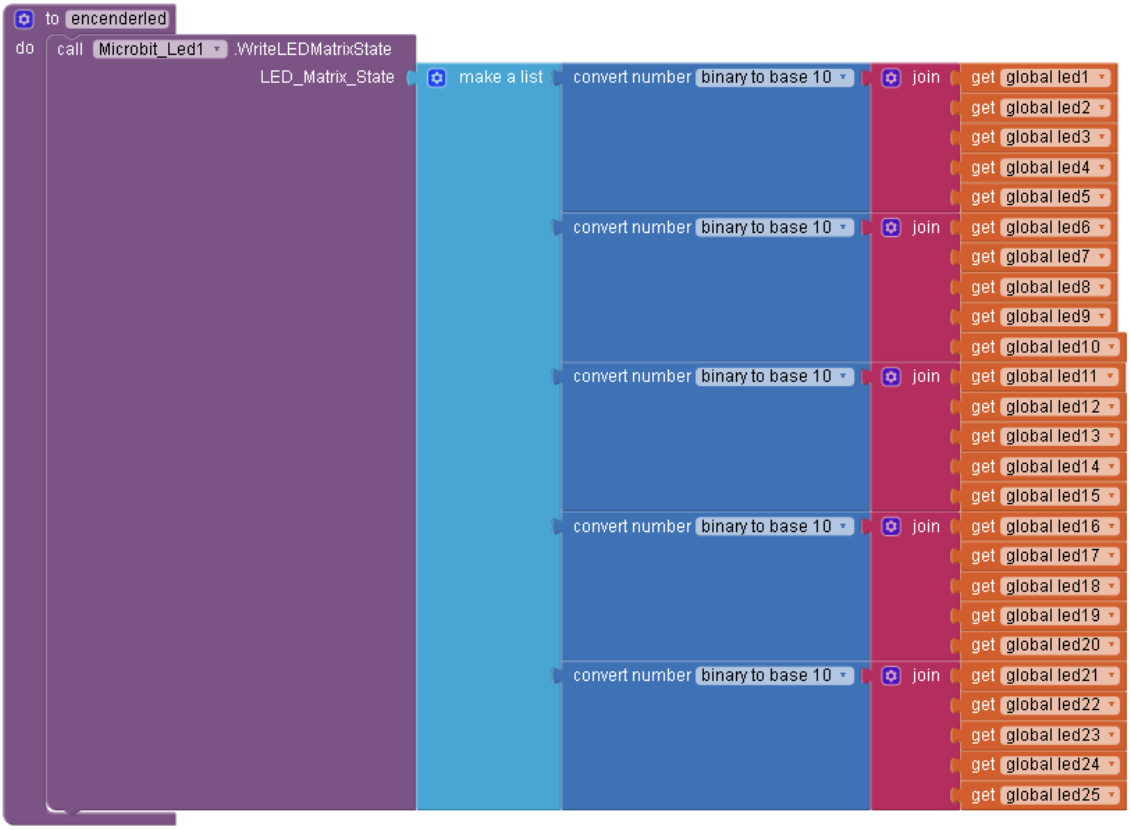


Figura 145 Código appinventor Práctica 4

Fuente: Elaboración propia

Ejecución

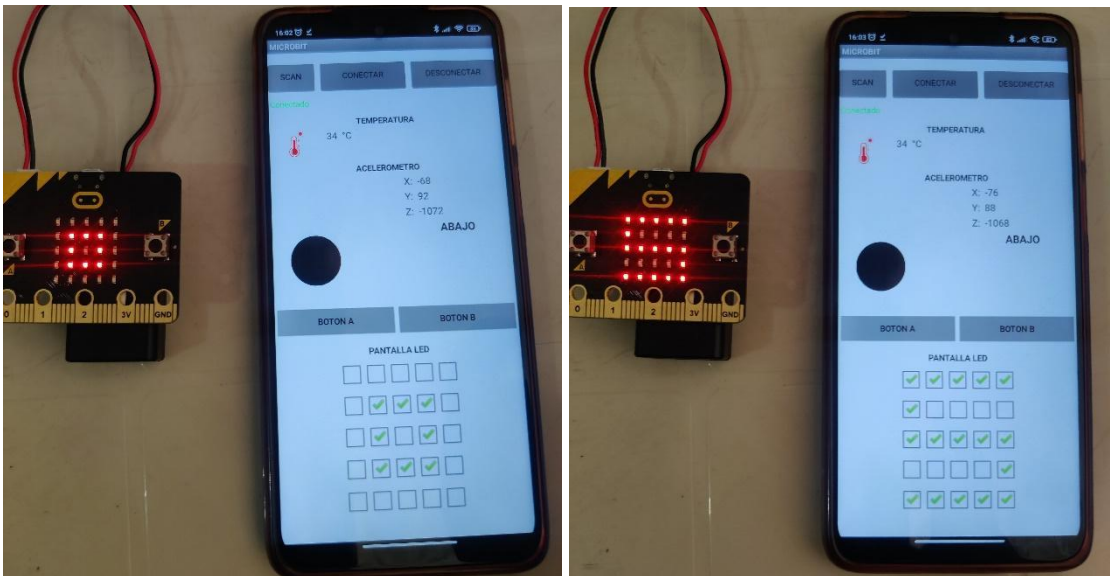


Figura 146 Ejecución práctica 4

Fuente: Elaboración propia

2.4.4.2.4 PRÁCTICA 5: ENCENDIDO Y APAGADO DE LEDS

Objetivos: Crear una aplicación móvil que permite encender y apagar un led utilizando el pin de la microbit a través de bluetooth.

Descripción: Para la realización de la práctica se utilizará la herramienta app inventor. Mediante el uso de dos botones se realizará el encendido y apagado del led.

App inventor

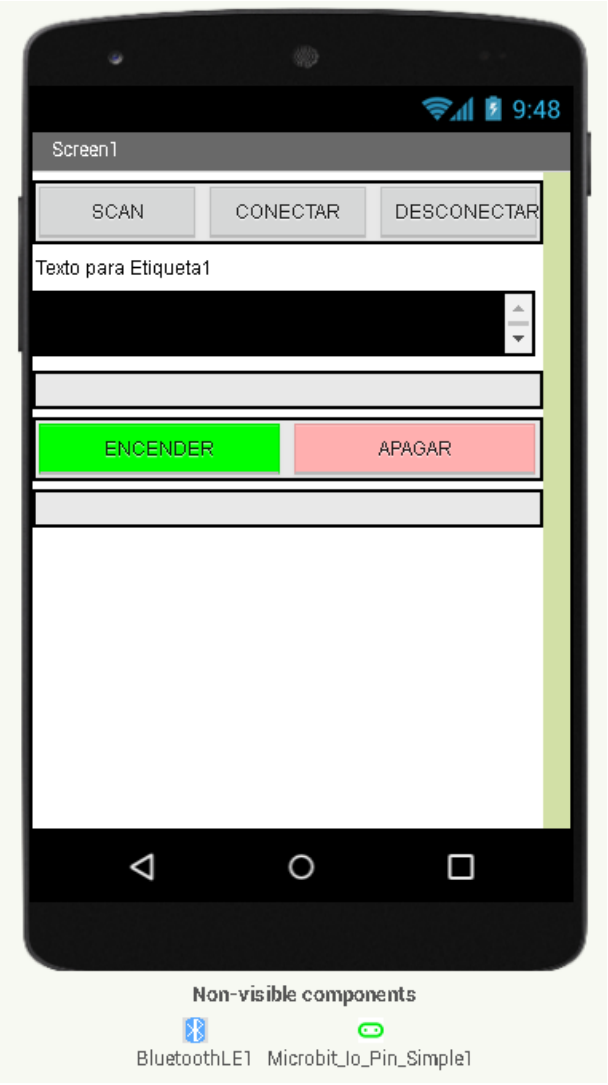


Figura 147 Interfaz gráfica Práctica 5

Fuente: Elaboración propia

Código

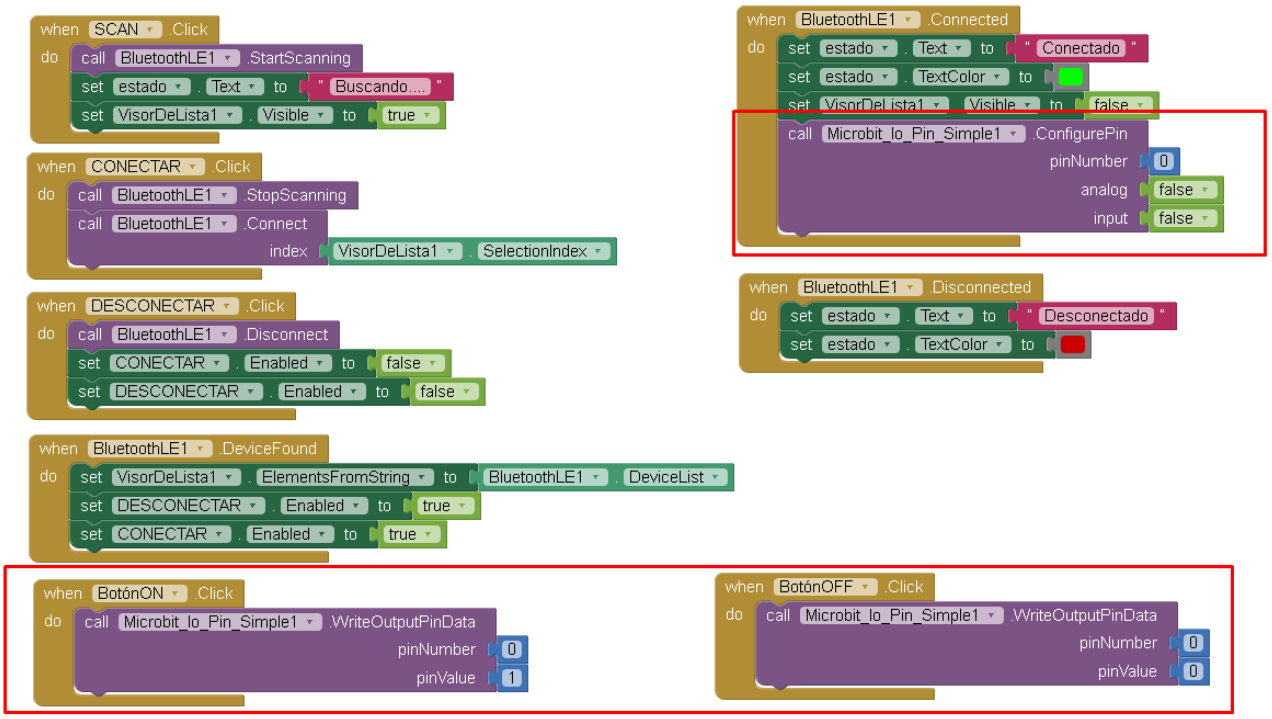


Figura 148 Código appinventor Práctica 5

Fuente: Elaboración propia

Ejecución

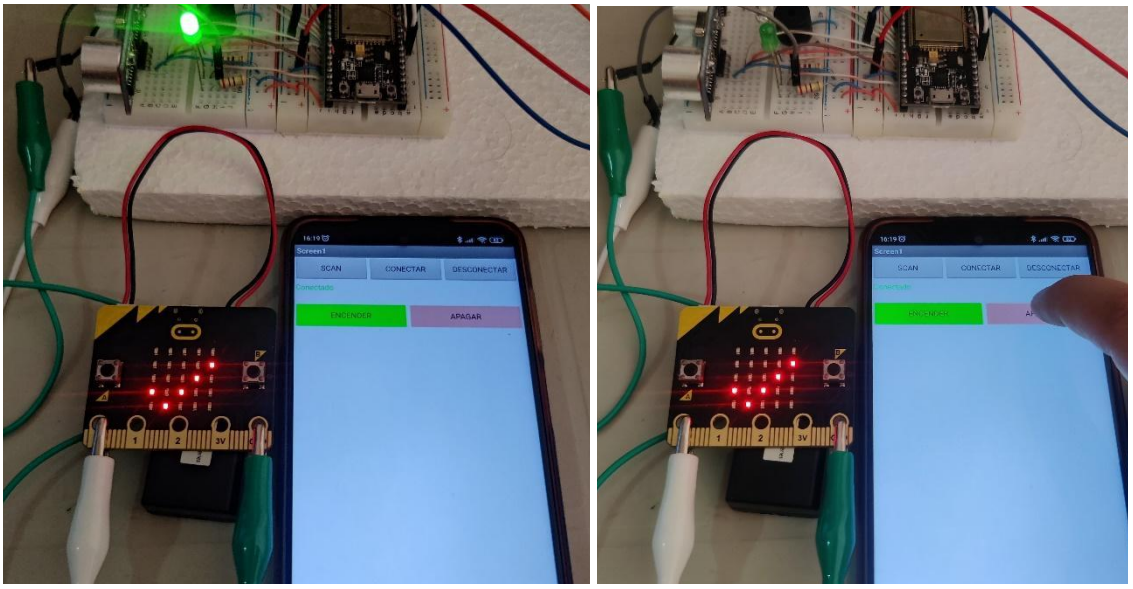


Figura 149 Ejecución práctica 5

Fuente: Elaboración propia

3. EVALUACIÓN DEL PROTOTIPO

3.1 Plan de evaluación

Para evaluar el funcionamiento de las aplicaciones se tomará en cuenta las características definidas en el modelo de calidad de software basadas en ISO 9126. El estándar ISO 9126 identifica características claves de calidad y los lineamientos para su uso. [39] [40] En la tabla 16 se muestran los parámetros para realizar la evaluación de las aplicaciones utilizadas en las prácticas.

Tabla 16 Parámetros de Evaluación

Características	Subcaracterísticas
Usabilidad	Facilidad de comprensión
	Facilidad de aprendizaje
	Operatividad
Eficiencia	Tiempo de uso
	Recursos utilizados
Portabilidad	Facilidad de ajuste
	Adaptabilidad
	Facilidad de instalación

Fuente: Elaboración propia

3.1.1 Usabilidad

Permite conocer el grado de esfuerzo que debe invertir el usuario para hacer uso de la aplicación.Para evaluar esta característica se realizaron encuestas a 27 personas entre estudiantes y profesionales.

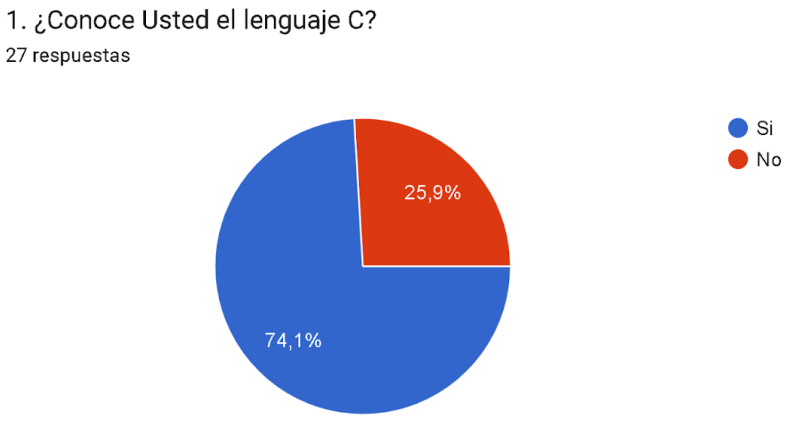


Figura 150 Pregunta 1. ¿Conoce usted el lenguaje C?

Figura 150 Fuente: Elaboración propia

Análisis de la pregunta 1: El 74.1 % de los encuestados tienen conocimiento del lenguaje C, mientras que un 25,9% no conoce este lenguaje.

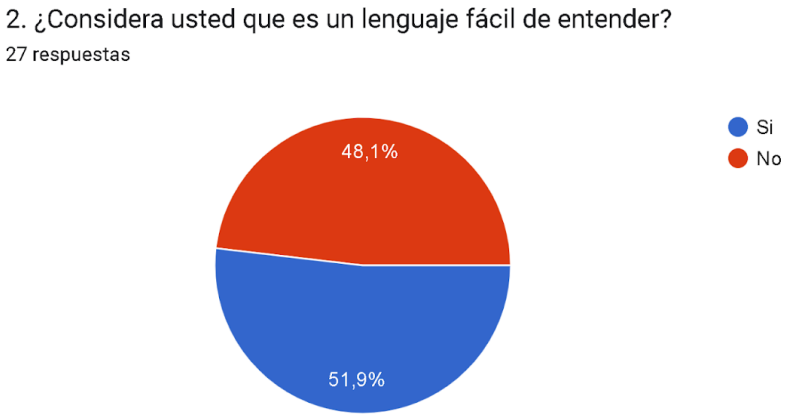


Figura 151 Pregunta 2. ¿Considera usted que es un lenguaje fácil de entender?

Fuente: Elaboración propia

Análisis de la pregunta 2: El 51,9% de los encuestados piensa que el lenguaje C es fácil de entender, mientras que un 48,1% considera que este lenguaje no es fácil de entender.

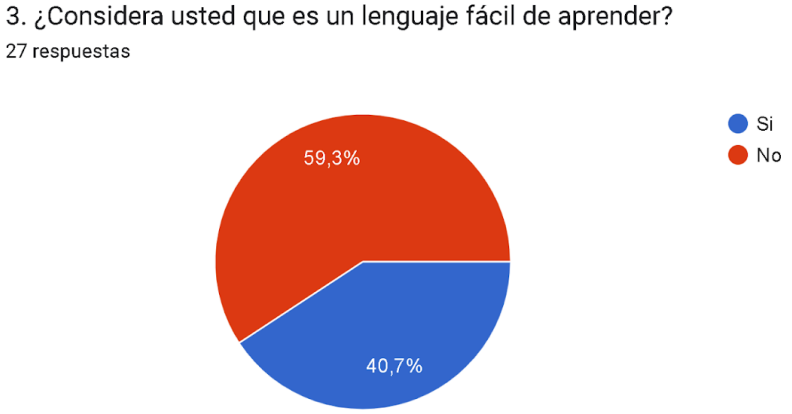


Figura 152 Pregunta 3 ¿Considera usted que es un lenguaje fácil de aprender?

Fuente: Elaboración propia

Análisis de la pregunta 3: El 59.3% de los encuestados considera que el lenguaje C es fácil de aprender, mientras que un 40,7% considera que este lenguaje es difícil de aprender.

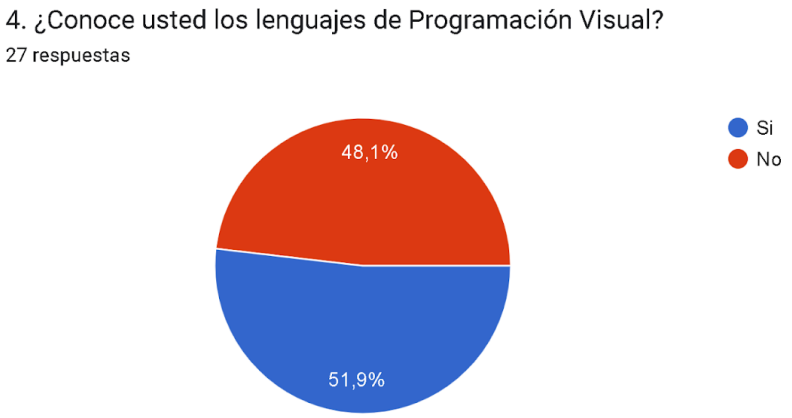


Figura 153 Pregunta 4 ¿Conoce usted los lenguajes de Programación Visual?

Fuente: Elaboración propia

Análisis de la pregunta 4: El 51,9% de los encuestados tiene conocimiento de los lenguajes de programación visual, mientras que 48,1% de los encuestados no conoce lo que son los lenguajes de programación.

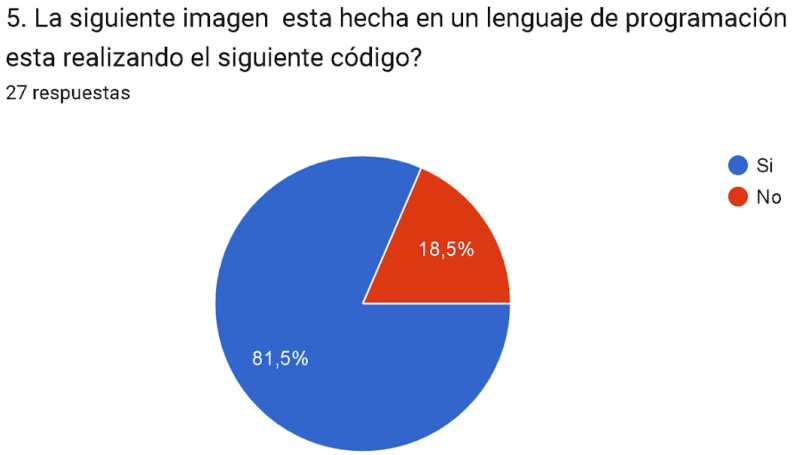


Figura 154 Pregunta 5 La siguiente imagen está hecha en un lenguaje de programación de bloques ¿Entiende lo que está realizando el siguiente código?

Fuente: Elaboración propia

Análisis de la pregunta 5: El 81,5% de los encuestados pudo entender el código basado en la programación de bloques mostrado sin necesidad de tener una explicación previa del código, mientras que un 18,5% de los encuestados no logró entender el código.

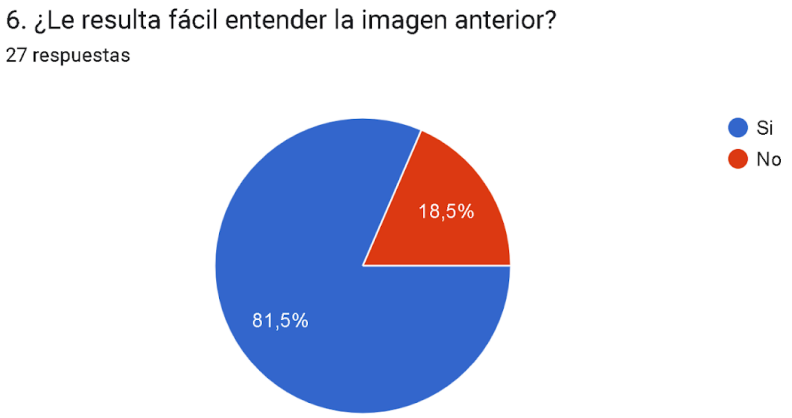


Figura 155 Pregunta 6 ¿Le resulta fácil entender la imagen anterior?

Fuente: Elaboración propia

Análisis de la pregunta 6: El 81,5% de los encuestados resultó fácil entender el código basado en la programación de bloques mostrado sin necesidad de tener una explicación previa del código, mientras que un 18,5% de los encuestados no logró entender el código.

Figura 156 Pregunta 7 La siguiente imagen muestra el código en lenguaje C de la imagen anterior basado en bloques. ¿Con cuál de los dos lenguajes le gustaría trabajar ?

Fuente: Elaboración propia

Análisis de la pregunta 7: El 88,9% de los encuestados les gustaría trabajar con el lenguaje de programación de bloques, mientras que un 11,1% de los encuestados trabajaría con el Lenguaje C.

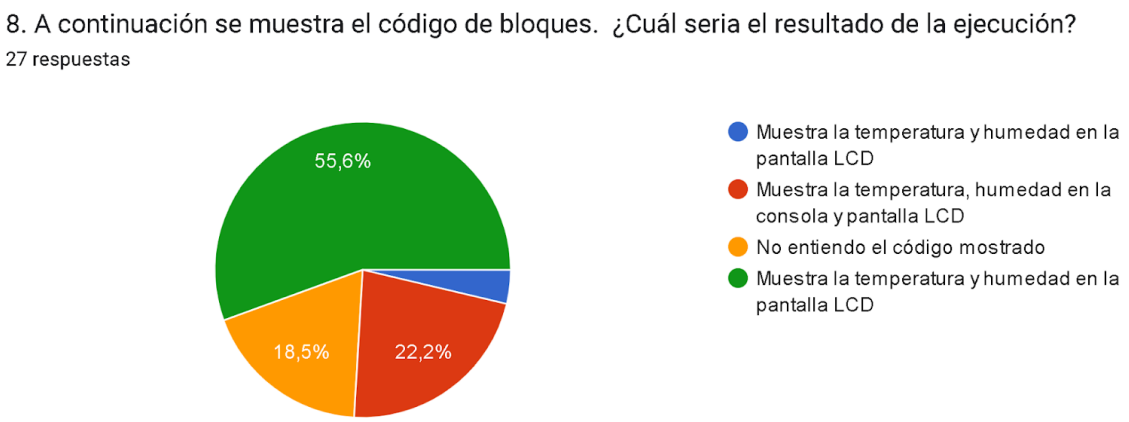


Figura 157 Pregunta 8 A continuación se muestra el código de bloques. ¿Cuál sería el resultado de la ejecución?

Fuente: Elaboración propia

Análisis de la pregunta 8: El 55,6% de los encuestados creen que al ejecutar el código se muestra la temperatura y humedad en la pantalla LCD, un 18,5% no entendió el código mostrado y solo un 22,2% elige la respuesta correcta.

3.1.2 Eficiencia

La eficiencia permite establecer el tiempo y la cantidad de recursos que se consumen al utilizar las aplicaciones.

Tabla 17 Evaluación de eficiencia

Subcaracterísticas	ESP32		MICROBIT
	RAM	Memoria Flash	
Recursos utilizados	39,42 Kbytes	743,99 Kbytes	No se visualiza

Fuente: Elaboración propia

Arduinoblocks permite visualizar los recursos que se consumen cuando se compila el programa a la ESP32, en la figura 158 se muestra cuánto se ha utilizado para el funcionamiento del programa en la práctica de SERVIDOR-GET del Grupo 2, con un 56% correspondiente a 743,99 Kbytes de almacenamiento, el máximo es de 1,31 MB. Los porcentajes de almacenamiento y de consumo de memoria varían de

acuerdo a las necesidades del programa como son variables que se van a utilizar dentro de la aplicación.

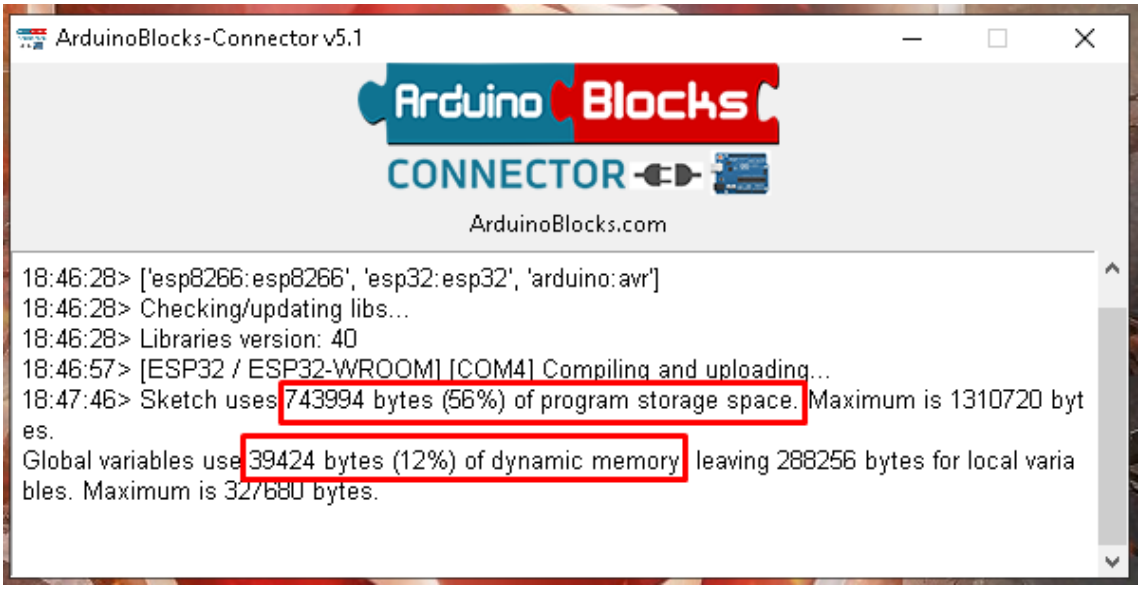


Figura 158 Recursos utilizados por ArduinoBlocks

Fuente: Elaboración propia

3.1.3 Portabilidad

Es la habilidad que posee el software para ser transferido de una plataforma a otra.

Tabla 18 Evaluación de portabilidad

Subcaracterísticas	ArduinoBlocks		MakeCode		Appinventor	
	Si	No	Si	No	Si	No
Facilidad de ajuste	X		X		X	
Adaptabilidad	X		X		X	
Facilidad de instalación	X		X		X	

Fuente: Elaboración propia

3.2 Resultado de la evaluación

3.2.1 Usabilidad

Según los resultados obtenidos de las encuestas realizadas se puede concluir 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 de tal manera que sea fácil de comprender y aprender.

3.2.2 Eficiencia

Los recursos que se utilizan en la ESP32 en base al programa que se está ejecutando y guardando se puede visualizar en el Arduinoblocks conector que nos muestra el total que ha ocupado el programa que se ha compilado, tal como lo hace en el Arduino IDE.

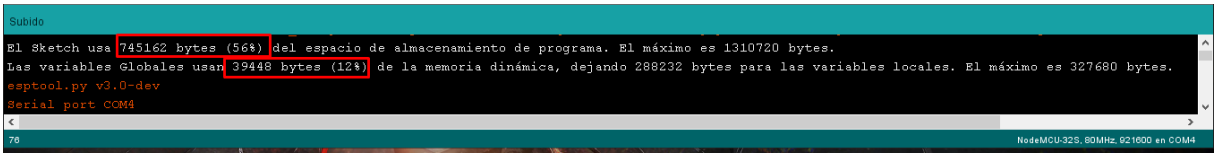


Figura 159 Recursos utilizados en la ESP32 con Arduino IDE

Fuente: Elaboración propia

En la figura 159 se muestra el mismo código compilado con el IDE de Arduino, el espacio de almacenamiento y de la memoria dinámica varía con unos cuantos bytes superiores a los utilizados con ArduinoBlocks. Además, consume recursos de nuestro ordenador aunque no implique que decaiga el rendimiento del mismo. En cuanto al ámbito de eficiencia se concluye que ArduinoBlocks posee cierta ventaja sobre el IDE de Arduino, ya que al estar basado en la web los recursos que consume son mínimos.

Para MICROBIT las aplicaciones utilizadas para la programación de la placa no visualiza los recursos que consumen dentro de la placa

3.2.3 Portabilidad

No es necesaria una instalación ya que están basadas en la web, ahorrándonos recursos como se mencionó en el parámetro de eficiencia. Estas herramientas son adaptables a cualquier sistema operativo que contenga un navegador web con acceso a internet.

Conclusiones

- Con la investigación bibliográfica se puede determinar que es una ayuda, la utilización de las herramientas de programación visual para la elaboración de las prácticas con las placas ESP32 y MICRO:BIT, el protocolo de comunicación MQTT y el módulo bluetooth.
- Las herramientas utilizadas en la elaboración de las prácticas de laboratorio son muy conocidas dentro del campo de la programación visual es por ello que fueron seleccionadas, ya que proporcionan una gran variedad de información y tutoriales para su uso.
- Con el desarrollo de las prácticas utilizando la placa ESP32, se puede concluir que ArduinoBlocks es una buena opción para crear sistemas IoT, ya que permite trabajar con el módulo MQTT, HTTP Servidor y Cliente, todo a través de la programación en bloques. La interfaz es fácil de manejar y los bloques son fáciles de entender.
- Con el desarrollo de las prácticas de laboratorio realizadas con la placa MICROBIT se concluye que MakeCode es una herramienta sencilla de utilizar y comprender, posee una gran variedad de tutoriales que permiten aprender la codificación de la placa y una gran comunidad que ayudarán a solventar los problemas que se susciten con la placa.
- Con el desarrollo de las prácticas del grupo 4 se concluye que la herramienta Appinventor permite trabajar con el módulo bluetooth de la placa ESP32 y MICRO:BIT sin ningún inconveniente y de manera sencilla, debido a que posee una interfaz gráfica que permite el desarrollo de las aplicaciones móviles para el control de los sistemas IoT a través de bluetooth.
- Las herramientas de programación visual permitieron la elaboración de proyectos basados en sistemas IOT, de manera sencilla ya que gracias a la programación basada en bloques solo se necesita conocer el funcionamiento principal de los bloques, los puertos que utilizaremos de las placas y que es lo que queremos que nuestro programa haga para crear un proyecto. Su interfaz es intuitiva y podemos encontrar información para aprender a utilizarlas en las páginas oficiales de ArduinoBlocks, MakeCode y Appinventor.

Recomendaciones

- En la selección de la herramienta de programación se debe tener en cuenta la placa que se utilizara en el proyecto ya que existen herramientas que solo permiten trabajar con placas específicas.
- Integrar las herramientas de programación visual dentro de los cursos de estudio para que los estudiantes puedan interesarse más en la programación de los sistemas IOT, gracias a la sencillez que proporcionan estas herramientas para la elaboración de los proyectos.
- Al momento de realizar los circuitos eléctricos tener en cuenta de conectar los pines correctamente para evitar los datos en la placa y además no se visualizarán los resultados correctos.

Bibliografía

[1] D. Hernández, T. Fernández y P. Fraga, «A Plug-and-Play Human-Centered Virtual TEDS Architecture for the Web of Things,» *Sensors,* vol. 18, nº 7, p. 2052, 2018.

[2] J. Berrú-Ayala, D. Hernandez-Rojas, P. Morocho-Díaz, J. Novillo-Vicuña, B. Mazon-Olivo y A. Pan, «SCADA system based on IoT for intelligent control of banana crop irrigation,» *International Conference on Applied Technologies,* pp. 243-256, 2019.

[3] P. Gokhale, O. Bhat y B. Sagar, «Introduction to IOT,» *IARJSET,* vol. 5, pp. 41-44, 2018.

[4] D. M, T. Y. M. Zagloel y R. S. L, «Knowledge growth and development: internet of things (IoT) research, 2006–2018,» *Elsevier,* vol. 5, pp. 1-14, 2019.

[5] P. Pratim Ray, «A Survey on Visual Programming Languages in the Internet of Things,» *Hindawi,* vol. 2017, pp. 1-6, 2017.

[6] J. Sáez y R. Cózar, «Pensamiento computacional y programación visual por bloques en el aula de Primaria,» *Complutense de Educación,* vol. 28, nº 2, pp. 409-426, 2017.

[7] I. Ruiz, J. M. Mota, T. Person y J. M. Rodríguez, «Block-Based Development of Mobile Learning Experiences for the Internet of Things,» *Sensors,* vol. 19, nº 24, pp. 19-24, 2019.

[8] J. Cartuche-Calva, D. Hernández-Rojas, R. Morocho-Román y C. Radicelli-García, «Seguridad IoT: Principales amenazas en una taxonomía de activos,» *HAMUT'AY,* vol. 1, nº 7, pp. 51-59, 2020.

[9] A. Campoverde, D. L. Hernández y B. Mazón, «Cloud computing con herramientas open-source para Internet de las cosas,» *Maskana,* vol. 6, pp. 173-182, 2015.

[10] F. M, W. Muhammad, M. Sadia, K. Anjum y K. Talha, «A Review on Internet of Things (IoT),» *International Journal of Computer Applications,* vol. 113, nº 1, pp. 1-7, 2015.

[11] B. Mazon-Olivo, D. Hernández-Rojas, J. Maza-Salinas y A. Pan, «Rules engine and complex event processor in the context of internet of things for precision agriculture,» *Computers and Electronics in Agriculture,* vol. 154, pp. 347-360, 2018.

[12] D. Hernández, B. Mazón y A. Campoverde, «Cloud Computing para el Internet de las Cosas. Caso de estudio orientado a la agricultura de precisión: I Congreso Internacional de Ciencia y tecnología,» *UTMACH,* pp. 47-53, 2015.

[13] D. Hernandez, B. Mazon-Olivo y C. Escudero, Internet de las cosas (IoT), Machala: Universidad Técnica de Machala, 2018, pp. 74-100.

[14] M. Resnick y N. Rusk, «Coding at a Crossroads» *Communications of the ACM,* vol. 11, nº 63, pp. 120-127, 2020.

[15] M. Hauck, R. Machhamer, L. Czenkusch, K.-U. Gollmer y G. Dartmann, «Node and Block-Based Development Tools for Distributed Systems With AI Applications,» *IEEE,* vol. 7, pp. 143109-143119, 2019.

[16] K. Radoslava y K. VELIN, «A Methodology for the Analysis of Block-Based Programming Languages Appropriate for Children '', Journal *of Computing Science and Engineering,* vol. 13, nº 1, pp. 1-10, 2019.

[17] C. Avella, E. Sandoval y C. Montañez, «Selección de herramientas web para la creación de actividades de aprendizaje en Cibermutua,» *Revista de investigación, Desarrollo e Innovación,* vol. 8, nº 1, pp. 107-120, 2017.

[18] M. Resnick, J. Maloney, A. Monroy Hernández, N. Rusk y E. Eastmond, «Scratch: programming for all,» *Communications of the ACM,* vol. 52, nº 11, pp. 60-67, 2009.

[19] Fundación Scratch, «Fundación Scratch,» [En línea]. Available: https://www.scratchfoundation.org/our-story. [Último acceso: 20 06 2022].

[20] Ç. Ünal, S. Sude, K. Yılmaz B, A. S. Y. Suheda y Ö. Mücahit, «Exploring perceived cognitive load in learning programming via Scratch,» *Research in Learning Technology,* vol. 26, 2018.

[21] F. Vera, F. Beltrán y J. D. D. Lugo, «Software Intérprete para la Interfaz Gráfica (Arduino Blocks),» *Revista de Cómputo Aplicado,* vol. 2, nº 6, pp. 6-10, 2018.

[22] J. Devine, J. Finney, P. Halleux, M. Moskal, T. Ball y S. Hodges, «MakeCode and CODAL: Intuitive and efficient embedded systems programming for education,» *Journal of Systems Architecture,* vol. 98, pp. 468-463, 2019.

[23] E. W. T. M. &. H. F. Patton, «Computational Thinking Education» de *MIT app inventor: Objectives, design, and development*, Singapore, Springer, 2019, pp. 31-49.

[24] P. Bertoleti, Proyectos con ESP32 y LoRa, Sao Paulo: INSTITUTO NEWTON C BRAGA., 2019.

[25] P. V y P. Ch Rajendra, «Internet of Things Based Home Monitoring and Device Control Using Esp32,» *International Journal of Recent Technology and Engineering,* vol. 8, nº 1, pp. 58-62, 2019.

[26] ESPRESSIF SYSTEMS, «ESPRESSIF,» 2022. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Último acceso: 13 06 2022].

[27] A. B. Mirza Jabbar, I. M. Tariq, J. Mohsin y K. Jahangir, «Design and implementation of an open-Source IoT and blockchain-based peer-to-peer energy trading platform using ESP32-S2, Node-Red and, MQTT protocol,» *Energy Reports,* vol. 7, pp. 5733-5746, 2021.

[28] A. Schmidt, «Increasing Computer Literacy with the BBC micro:bit,» *IEEE,* vol. 15, nº 2, pp. 5-7, 2016.

[29] A. Cederqvist, «An exploratory study of technological knowledge when pupils are designing a programmed technological solution using BBC Micro:bit,» *International Journal of Technology and Design Education,* pp. 1-27, 2020.

[30] L. Corona, G. Abarca y J. Mares, Sensores y actuadores, México: Grupo Editorial Patria, 2014.

[31] K. Lounis y M. Zulkernine, «Attacks and Defenses in Short-Range Wireless Technologies for IoT,» *IEEE,* vol. 8, pp. 88892-88932, 2020.

[32] D. Hernández-Roja, T. Fernández-Caramés, P. Fraga-Lamas y C. Escudero, «Design and practical evaluation of a family of lightweight protocols for heterogeneous sensing through BLE beacons in IoT telemetry applications,» *Sensors,* vol. 18, nº 1, p. 57, 2017.

[33] D. Hernandez-Rojas, B. Mazon-Olivo, J. Novillo-Vicuña, C. Escudero-Cascon, A. Pan-Bermudez y G. Belduma-Vacacela, «IoT Android Gateway for Monitoring and Control a WSN,» *International Conference on Technology Trends. Springer,* pp. 18-32, 2017.

[34] D. Bilal, R. Atteeq Ur y M. Rizwan, «Internet of Things (IoT) Protocols: A Brief Exploration of MQTT and CoAP,» *International Journal of Computer Applications,* vol. 179, nº 27, pp. 9-14, 2018.

[35] M. Kashyap, V. Sharma y N. Gupta, «Taking MQTT and NodeMcu to IOT: Communication in Internet of Things,» *Procedia Computer Science,* vol. 132, pp. 1611-1618, 2018.

[36] D. Silva, L. Carvalho, J. Soares y S. Rute, «A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA,» *Applied Sciences,* vol. 11, nº 11, p. 4879, 2021.

[37] A. Haripriya y K. Kulothungan «Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things,» *Energy Reports,* vol. 2019, nº 1, pp. 1-15, 2019.

[38] V. Aleksandar, M. Aleksandar y W. Steffen, «Covert channels in the MQTT-based Internet of Things,» *IEEE ,* vol. 7, pp. 161899-161915, 2019.

[39] S. Aizprua, A. Ortega y L. Von Chong, «Calidad de Software una Perspectiva Continua,» *CENTROS,* vol. 8, nº 2, pp. 120-134, 2019.

[40] L. Z. Castillo González y L. R. Guanipa Maluenga, «NORMAS ISO 9126 E ISO 14598,» *LAS TIC Y LA EDUCACIÓN, ante el reto de la Innovación,* pp. 78-74, 2018.

[41] K. Kyungbin, L. Sang Joon y C. Jaehwa, «Computational Concepts Reflected on Scratch Programs,» *International Journal of Computer Science Education in Schools,* vol. 2, nº 3, 2018.

[42] C. P. Millahual, Descubriendo Arduino, Buenos Aires: RedUsers, 2020.

[43] N. Zade, S. Deshpande y D. Sita, «Analysis of Passive Infrared Detectors for Target Detection in an IOT Based Outdoor Environment» *Proceedings of the International Conference on Recent Advances in Computational Techniques,* 2020.

[44] J. Novillo, D. Hernandez, O. Mazón, R. Molina y O. Villavicencio, «Arduino y el internet de las cosas,» vol. 45, 2018.

Anexos

Anexo A. GRUPO 2: Práctica 1

```
#include "WiFi.h"
#include <Wire.h>
#include "ABlocks_LiquidCrystal_I2C.h"
#include "PubSubClient.h"

double estadoled;
double estadoled2;
const char espwifi_ssid[]="RAPINET_Rendon";
const char espwifi_pass[]="0706718905_@Suli";
LiquidCrystal_I2C lcd_1(0x27,16,2);
const char espmqtt_broker[]="192.168.100.35";
const int espmqtt_port=1883;
const char espmqtt_user[]="";
const char espmqtt_pass[]="";
const char espmqtt_clientid[]="";
WiFiClient espmqtt_wifiClient;
PubSubClient espmqtt_client(espmqtt_wifiClient);
String espmqtt_topic="";
String espmqtt_msg_str="";
double espmqtt_msg_number=0;
char espmqtt_payload[128];

void espwifi_setup(){
	WiFi.mode(WIFI_STA);
	WiFi.begin(espwifi_ssid,espwifi_pass);
	while (WiFi.status() != WL_CONNECTED) delay(500);
}

void espmqtt_onreceive(){};

void espmqtt_setup(){
	delay(10);;
	randomSeed(micros());
	espmqtt_client.setServer(espmqtt_broker, espmqtt_port);
	espmqtt_client.setCallback(espmqtt_callback);
	espmqtt_subscribe();
}
void espmqtt_loop(){
	if (!espmqtt_client.connected()) {
		espmqtt_client.connect(espmqtt_clientid,espmqtt_user,espmqtt_pass);
		espmqtt_subscribe();
	}
	if (espmqtt_client.connected()) {
		espmqtt_client.loop();
	}
}
double espmqtt_payload2double(unsigned char *_payload, int _length){
```

```
	int i;
	for (i = 0; i<_length && i<128; i++){
		espmqtt_payload[i] = _payload[i];
	}
	espmqtt_payload[i] = 0;
	return atof(espmqtt_payload);
}

String espmqtt_payload2string(unsigned char *_payload, int _length){
	int i;
	for (i = 0; i<_length && i<128; i++){
		espmqtt_payload[i] = _payload[i];
	}
	espmqtt_payload[i] = 0;
	return String(espmqtt_payload);
}
void espmqtt_callback(char* _topic, unsigned char* _payload, unsigned int
_payloadlength){
	espmqtt_topic=String(_topic);
	espmqtt_msg_str=espmqtt_payload2string(_payload,_payloadlength);
	espmqtt_msg_number=espmqtt_payload2double(_payload,_payloadlength);

if(espmqtt_topic==String("encender/apagar/led1"))estadoled=espmqtt_msg_number;

if(espmqtt_topic==String("encender/apagar/led2"))estadoled2=espmqtt_msg_numbe
r;
	espmqtt_onreceive();
}

void espmqtt_subscribe(){
	espmqtt_client.subscribe(String("encender/apagar/led1").c_str());
	espmqtt_client.subscribe(String("encender/apagar/led2").c_str());
}

void ENCENDER_LED_2() {
	if ((estadoled2 == 1)) {
		digitalWrite(27,HIGH);
	}
	else {
		digitalWrite(27,LOW);
	}

}
void ENCENDER_LED_1() {
	if ((estadoled == 1)) {
		digitalWrite(13,HIGH);
	}
	else {
		digitalWrite(13,LOW);
	}
```

```
}
void LDR() {
}
void setup()
{
	pinMode(27, OUTPUT);
	pinMode(13, OUTPUT);

	espwifi_setup();
	lcd_1.begin();
	lcd_1.noCursor();
	lcd_1.backlight();
	espmqtt_setup();

}

void loop()
{
	yield();

	espmqtt_loop();
	ENCENDER_LED_1();
	ENCENDER_LED_2();

}
```

Anexo B. GRUPO 2: Práctica 2

```
#include "WiFi.h"
#include <Wire.h>
#include "ABlocks_LiquidCrystal_I2C.h"
#include "PubSubClient.h"
#include "ABlocks_DHT.h"

double temperatura;
double humedad;
const char espwifi_ssid[]="RAPINET_Rendon";
const char espwifi_pass[]="0706718905_@Suli";
LiquidCrystal_I2C lcd_1(0x27,16,2);
const char espmqtt_broker[]="192.168.100.35";
const int espmqtt_port=1883;
const char espmqtt_user[]="";
const char espmqtt_pass[]="";
const char espmqtt_clientid[]="";
WiFiClient espmqtt_wifiClient;
PubSubClient espmqtt_client(espmqtt_wifiClient);
String espmqtt_topic="";
String espmqtt_msg_str="";
double espmqtt_msg_number=0;
char espmqtt_payload[128];
```

```
DHT dht23(23,DHT22);

void espwifi_setup(){
	WiFi.mode(WIFI_STA);
	WiFi.begin(espwifi_ssid,espwifi_pass);
	while (WiFi.status() != WL_CONNECTED) delay(500);
}

void espmqtt_onreceive(){};

void espmqtt_setup(){
	delay(10);;
	randomSeed(micros());
	espmqtt_client.setServer(espmqtt_broker, espmqtt_port);
	espmqtt_client.setCallback(espmqtt_callback);
	espmqtt_subscribe();
}

void espmqtt_loop(){
	if (!espmqtt_client.connected()) {
		espmqtt_client.connect(espmqtt_clientid,espmqtt_user,espmqtt_pass);
		espmqtt_subscribe();
	}
	if (espmqtt_client.connected()) {
		espmqtt_client.loop();
	}
}

double espmqtt_payload2double(unsigned char *_payload, int _length){
	int i;
	for (i = 0; i<_length && i<128; i++){
		espmqtt_payload[i] = _payload[i];
	}
	espmqtt_payload[i] = 0;
	return atof(espmqtt_payload);
}

String espmqtt_payload2string(unsigned char *_payload, int _length){
	int i;
	for (i = 0; i<_length && i<128; i++){
		espmqtt_payload[i] = _payload[i];
	}
	espmqtt_payload[i] = 0;
	return String(espmqtt_payload);
}

void espmqtt_callback(char* _topic, unsigned char* _payload, unsigned int
_payloadlength){
	espmqtt_topic=String(_topic);
	espmqtt_msg_str=espmqtt_payload2string(_payload,_payloadlength);
```

```
	espmqtt_msg_number=espmqtt_payload2double(_payload,_payloadlength);
	espmqtt_onreceive();
}

void espmqtt_subscribe(){
}
void CLIMATIZACION() {
	temperatura = dht23.readTemperature();
	humedad = dht23.readHumidity();
	if ((!isnan(temperatura))) {

espmqtt_client.publish(String("sensor/dth22/temp").c_str(),String(temperatura).c_str(
));
		lcd_1.setCursor(0, 0);
		lcd_1.print(String("Temp:"));
		lcd_1.setCursor(8, 0);
		lcd_1.print(temperatura);

espmqtt_client.publish(String("sensor/dth22/hum").c_str(),String(humedad).c_str());
		lcd_1.setCursor(0, 1);
		lcd_1.print(String("Hum:"));
		lcd_1.setCursor(8, 1);
		lcd_1.print(humedad);
	}
}
void setup()
{
	pinMode(23, INPUT);

	dht23.begin();

	espwifi_setup();
	lcd_1.begin();
	lcd_1.noCursor();
	lcd_1.backlight();
	espmqtt_setup();

}
void loop()
{
	yield();

	espmqtt_loop();
	CLIMATIZACION();

}
```

Anexo C. GRUPO 2: Práctica 3

```
#include "WiFi.h"
#include <Wire.h>
#include "ABlocks_LiquidCrystal_I2C.h"
```

```
#include "PubSubClient.h"
#include <analogWrite.h>

double distanciaestablecida;
double detector;
const char espwifi_ssid[]="RAPINET_Rendon";
const char espwifi_pass[]="0706718905_@Suli";
LiquidCrystal_I2C lcd_1(0x27,16,2);
const char espmqtt_broker[]="192.168.100.35";
const int espmqtt_port=1883;
const char espmqtt_user[]="";
const char espmqtt_pass[]="";
const char espmqtt_clientid[]="";
WiFiClient espmqtt_wifiClient;
PubSubClient espmqtt_client(espmqtt_wifiClient);
String espmqtt_topic="";
String espmqtt_msg_str="";
double espmqtt_msg_number=0;
char espmqtt_payload[128];

void espwifi_setup(){
        WiFi.mode(WIFI_STA);
        WiFi.begin(espwifi_ssid,espwifi_pass);
        while (WiFi.status() != WL_CONNECTED) delay(500);
}

void espmqtt_onreceive(){};

void espmqtt_setup(){
        delay(10);;
        randomSeed(micros());
        espmqtt_client.setServer(espmqtt_broker, espmqtt_port);
        espmqtt_client.setCallback(espmqtt_callback);
        espmqtt_subscribe();
}

void espmqtt_loop(){
        if (!espmqtt_client.connected()) {
                espmqtt_client.connect(espmqtt_clientid,espmqtt_user,espmqtt_pass);
                espmqtt_subscribe();
        }
        if (espmqtt_client.connected()) {
                espmqtt_client.loop();
        }
}

double espmqtt_payload2double(unsigned char *_payload, int _length){
        int i;
        for (i = 0; i<_length && i<128; i++){
                espmqtt_payload[i] = _payload[i];
```

```
        }
        espmqtt_payload[i] = 0;
        return atof(espmqtt_payload);
}

String espmqtt_payload2string(unsigned char *_payload, int _length){
        int i;
        for (i = 0; i<_length && i<128; i++){
                espmqtt_payload[i] = _payload[i];
        }
        espmqtt_payload[i] = 0;
        return String(espmqtt_payload);
}

void espmqtt_callback(char* _topic, unsigned char* _payload, unsigned int
_payloadlength){
        espmqtt_topic=String(_topic);
        espmqtt_msg_str=espmqtt_payload2string(_payload,_payloadlength);
        espmqtt_msg_number=espmqtt_payload2double(_payload,_payloadlength);

if(espmqtt_topic==String("distancia/establecida"))distanciaestablecida=espmqtt_msg
_number;
        espmqtt_onreceive();
}

void espmqtt_subscribe(){
        espmqtt_client.subscribe(String("distancia/establecida").c_str());
}

double fnc_ultrasonic_distance(int _t, int _e){
        unsigned long dur=0;
        digitalWrite(_t, LOW);
        delayMicroseconds(5);
        digitalWrite(_t, HIGH);
        delayMicroseconds(10);
        digitalWrite(_t, LOW);
        dur = pulseIn(_e, HIGH, 18000);
        if(dur==0)return 999.0;
        return (dur/57);
}

void SENSOR() {
        detector = fnc_ultrasonic_distance(14,12);
        if ((!isnan(detector))) {

espmqtt_client.publish(String("sensor/ultrasonico").c_str(),String(detector).c_str());
                if ((detector < distanciaestablecida)) {
                        digitalWrite(26, HIGH);
                        analogWrite(32,255-(255));
                        analogWrite(33,255-(0));
```

```
                        analogWrite(25,255-(0));
                        analogWrite(32,255-(51));
                        analogWrite(33,255-(51));
                        analogWrite(25,255-(255));
                        analogWrite(32,255-(0));
                        analogWrite(33,255-(153));
                        analogWrite(25,255-(0));

espmqtt_client.publish(String("enviar/alarma").c_str(),String("ALERTA!!
....MOVIMIENTO DETECTADO").c_str());
                }
                else {
                        digitalWrite(26, LOW);
                        analogWrite(32,255-(255));
                        analogWrite(33,255-(255));
                        analogWrite(25,255-(255));
                }

        }

}

void setup()
{
        pinMode(14, OUTPUT);
        pinMode(12, INPUT);
        pinMode(26, OUTPUT);
        pinMode(32, OUTPUT);
        pinMode(33, OUTPUT);
        pinMode(25, OUTPUT);

        SENSOR();

}


void loop()
{
        yield();

        espmqtt_loop();
        espwifi_setup();
        lcd_1.begin();
        lcd_1.noCursor();
        lcd_1.backlight();
        espmqtt_setup();

}
```

Anexo D. GRUPO 2: Práctica 4

```
#include "WiFi.h"
```

```
#include <Wire.h>
#include "ABlocks_LiquidCrystal_I2C.h"
#include "PubSubClient.h"

double intensidad;
double sensor;
double estado;
const char espwifi_ssid[]="RAPINET_Rendon";
const char espwifi_pass[]="0706718905_@Suli";
LiquidCrystal_I2C lcd_1(0x27,16,2);
const char espmqtt_broker[]="192.168.100.35";
const int espmqtt_port=1883;
const char espmqtt_user[]="";
const char espmqtt_pass[]="";
const char espmqtt_clientid[]="";
WiFiClient espmqtt_wifiClient;
PubSubClient espmqtt_client(espmqtt_wifiClient);
String espmqtt_topic="";
String espmqtt_msg_str="";
double espmqtt_msg_number=0;
char espmqtt_payload[128];

void espwifi_setup(){
	WiFi.mode(WIFI_STA);
	WiFi.begin(espwifi_ssid,espwifi_pass);
	while (WiFi.status() != WL_CONNECTED) delay(500);
}

void espmqtt_onreceive(){};

void espmqtt_setup(){
	delay(10);;
	randomSeed(micros());
	espmqtt_client.setServer(espmqtt_broker, espmqtt_port);
	espmqtt_client.setCallback(espmqtt_callback);
	espmqtt_subscribe();
}

void espmqtt_loop(){
	if (!espmqtt_client.connected()) {
		espmqtt_client.connect(espmqtt_clientid,espmqtt_user,espmqtt_pass);
		espmqtt_subscribe();
	}
	if (espmqtt_client.connected()) {
		espmqtt_client.loop();
	}
}

double espmqtt_payload2double(unsigned char *_payload, int _length){
	int i;
```

```
        for (i = 0; i<_length && i<128; i++){
                espmqtt_payload[i] = _payload[i];
        }
        espmqtt_payload[i] = 0;
        return atof(espmqtt_payload);
}

String espmqtt_payload2string(unsigned char *_payload, int _length){
        int i;
        for (i = 0; i<_length && i<128; i++){
                espmqtt_payload[i] = _payload[i];
        }
        espmqtt_payload[i] = 0;
        return String(espmqtt_payload);
}

void espmqtt_callback(char* _topic, unsigned char* _payload, unsigned int
_payloadlength){
        espmqtt_topic=String(_topic);
        espmqtt_msg_str=espmqtt_payload2string(_payload,_payloadlength);
        espmqtt_msg_number=espmqtt_payload2double(_payload,_payloadlength);

if(espmqtt_topic==String("intensidad/establecida"))intensidad=espmqtt_msg_number
;
        espmqtt_onreceive();
}

void espmqtt_subscribe(){
        espmqtt_client.subscribe(String("intensidad/establecida").c_str());
}

void LDR() {
        sensor = analogRead(35);
        if ((!isnan(sensor))) {
                espmqtt_client.publish(String("ldr/valor").c_str(),String(sensor).c_str());
                if ((sensor < intensidad)) {
                        digitalWrite(5,HIGH);
                        estado = 1;

espmqtt_client.publish(String("led/estado").c_str(),String(estado).c_str());
                }
                else {
                        digitalWrite(5,LOW);
                        estado = 0;

espmqtt_client.publish(String("led/estado").c_str(),String(estado).c_str());
                }

        }
```

```
}
void setup()
{
	pinMode(5, OUTPUT);

	pinMode(35, INPUT);

	espwifi_setup();
	lcd_1.begin();
	lcd_1.noCursor();
	lcd_1.backlight();
	espmqtt_setup();

}

void loop()
{
	yield();

	espmqtt_loop();
	LDR();

}
```

Anexo E. GRUPO 2: Práctica 5

```
#include "WiFi.h"
#include <Wire.h>
#include "ABlocks_LiquidCrystal_I2C.h"
#include "PubSubClient.h"

double relay;
const char espwifi_ssid[]="RAPINET_Rendon";
const char espwifi_pass[]="0706718905_@Suli";
LiquidCrystal_I2C lcd_1(0x27,16,2);
const char espmqtt_broker[]="192.168.100.35";
const int espmqtt_port=1883;
const char espmqtt_user[]="";
const char espmqtt_pass[]="";
const char espmqtt_clientid[]="";
WiFiClient espmqtt_wifiClient;
PubSubClient espmqtt_client(espmqtt_wifiClient);
String espmqtt_topic="";
String espmqtt_msg_str="";
double espmqtt_msg_number=0;
char espmqtt_payload[128];

void espwifi_setup(){
	WiFi.mode(WIFI_STA);
	WiFi.begin(espwifi_ssid,espwifi_pass);
	while (WiFi.status() != WL_CONNECTED) delay(500);
}
```

```
void espmqtt_onreceive(){};

void espmqtt_setup(){
	delay(10);;
	randomSeed(micros());
	espmqtt_client.setServer(espmqtt_broker, espmqtt_port);
	espmqtt_client.setCallback(espmqtt_callback);
	espmqtt_subscribe();
}

void espmqtt_loop(){
	if (!espmqtt_client.connected()) {
		espmqtt_client.connect(espmqtt_clientid,espmqtt_user,espmqtt_pass);
		espmqtt_subscribe();
	}
	if (espmqtt_client.connected()) {
		espmqtt_client.loop();
	}
}

double espmqtt_payload2double(unsigned char *_payload, int _length){
	int i;
	for (i = 0; i<_length && i<128; i++){
		espmqtt_payload[i] = _payload[i];
	}
	espmqtt_payload[i] = 0;
	return atof(espmqtt_payload);
}

String espmqtt_payload2string(unsigned char *_payload, int _length){
	int i;
	for (i = 0; i<_length && i<128; i++){
		espmqtt_payload[i] = _payload[i];
	}
	espmqtt_payload[i] = 0;
	return String(espmqtt_payload);
}

void espmqtt_callback(char* _topic, unsigned char* _payload, unsigned int
_payloadlength){
	espmqtt_topic=String(_topic);
	espmqtt_msg_str=espmqtt_payload2string(_payload,_payloadlength);
	espmqtt_msg_number=espmqtt_payload2double(_payload,_payloadlength);
	espmqtt_onreceive();
}

void espmqtt_subscribe(){
}
```

```
void rele() {
	if ((relay == 1)) {
		digitalWrite(18,HIGH);
	}
	else {
		digitalWrite(18,LOW);
	}

}

void setup()
{
	pinMode(18, OUTPUT);

	espwifi_setup();
	lcd_1.begin();
	lcd_1.noCursor();
	lcd_1.backlight();
	espmqtt_setup();

}


void loop()
{
	yield();

	espmqtt_loop();
	rele();

}
```

Anexo F. GRUPO 2: SERVIDOR - GET/POST

```
#include <Wire.h>

#include "ABlocks_LiquidCrystal_I2C.h"

#include "WiFi.h"

#include <WebServer.h>

#include "ABlocks_DHT.h"


double temperatura;

double humedad;

double distancia;

LiquidCrystal_I2C lcd_1(0x27,16,2);

const char espwifi_ssid[]="RAPINET_Rendon";

const char espwifi_pass[]="0706718905_@Suli";
```

```
WebServer http_server;
DHT dht23(23,DHT22);

void espwifi_setup(){
	WiFi.mode(WIFI_STA);
	WiFi.begin(espwifi_ssid,espwifi_pass);
	while (WiFi.status() != WL_CONNECTED) delay(500);
}

double fnc_ultrasonic_distance(int _t, int _e){
	unsigned long dur=0;
	digitalWrite(_t, LOW);
	delayMicroseconds(5);
	digitalWrite(_t, HIGH);
	delayMicroseconds(10);
	digitalWrite(_t, LOW);
	dur = pulseIn(_e, HIGH, 18000);
	if(dur==0)return 999.0;
	return (dur/57);
}
void html() {
	temperatura = dht23.readTemperature();
	humedad = dht23.readHumidity();
	distancia = fnc_ultrasonic_distance(14,12);
	http_server.send(200,String("text/html"),String("<!DOCTYPE
html><HTML><HEAD>")+String("<META ")+String("http-equiv=\"refresh\"
content=\"5\"")+String(" >")+String("<STYLE>")+String("body {background-color:
#FFFFDD;}")+String("button {display: block;width: 80px;background-color:
#3498db;border: none;color: white;padding: 13px 30px;text-decoration:
none;font-size: 25px;margin: 0px auto 35px;cursor: pointer;border-radius:
4px;}")+String("</STYLE>")+String("<TITLE>")+String("IOT")+String("</TITLE>")+Str
ing("</HEAD><BODY>")+String("<div
align=\"center\">")+String("<H1>")+String("CLIMA")+String("</H1>")+String("</div>")
+String("<p align=\"center\">")+String("Temperatura:
")+String(temperatura)+String("ºC")+String("</p>")+String("<p
align=\"center\">")+String("Humedad:
")+String(humedad)+String("%")+String("</p>")+String("<HR>")+String("<div
```

```
align=\"center\">")+String("<H1>")+String("LEDS")+String("</H1>")+String("</div>")+
String("<A href=\"")+String("/on")+String("\"
target=\"")+String("_self")+String("\">")+String("<button>")+String("ON")+String("</bu
tton>")+String("</A>")+String("<A href=\"")+String("/off")+String("\"
target=\"")+String("_self")+String("\">")+String("<button>")+String("OFF")+String("</b
utton>")+String("</A>")+String("<HR>")+String("<A
href=\"")+String("/releon")+String("\"
target=\"")+String("_self")+String("\">")+String("<button>")+String("ON")+String("</bu
tton>")+String("</A>")+String("<A href=\"")+String("/releoff")+String("\"
target=\"")+String("_self")+String("\">")+String("<button>")+String("OFF")+String("</b
utton>")+String("</A>")+String("<HR>")+String("<div
align=\"center\">")+String("<H1>")+String("DISTANCIA")+String("</H1>")+String("</d
iv>")+String("<p align=\"center\">")+String("Distancia:
")+String(distancia)+String("")+String("</p>")+String("</BODY></HTML>"));
}
void _http_server_on__NOTFOUND__(){
        http_server.send(200,String("text/plain"),String("Pagina no encontrada"));
}

void http_server_on_(){
        html();
}
void http_server_on_on(){
        html();
        digitalWrite(27,HIGH);
}
void http_server_on_releon(){
        html();
        digitalWrite(18,HIGH);
}
void http_server_on_off(){
        html();
        digitalWrite(27,LOW);
}
void http_server_on_releoff(){
        html();
        digitalWrite(18,LOW);
```

```
}
void setup()
{
	pinMode(23, INPUT);
	pinMode(14, OUTPUT);
	pinMode(12, INPUT);
	pinMode(2, OUTPUT);
	pinMode(4, INPUT);
	pinMode(27, OUTPUT);
	pinMode(18, OUTPUT);
	Serial.begin(115200);
	Serial.flush();
	while(Serial.available()>0)Serial.read();
	dht23.begin();
	lcd_1.begin();
	lcd_1.noCursor();
	lcd_1.backlight();
	espwifi_setup();
	Serial.println(WiFi.localIP().toString());
	lcd_1.setCursor(0, 0);
	lcd_1.print(WiFi.localIP().toString());
	http_server.begin(80);
	http_server.onNotFound(_http_server_on__NOTFOUND__);
	http_server.on("/",HTTP_GET,http_server_on_);
	http_server.on("/on",HTTP_GET,http_server_on_on);
	http_server.on("/releon",HTTP_GET,http_server_on_releon);
	http_server.on("/off",HTTP_GET,http_server_on_off);
	http_server.on("/releoff",HTTP_GET,http_server_on_releoff);
}
void loop()
{
	yield();
```

```
	http_server.handleClient();
	temperatura = dht23.readTemperature();
	humedad = dht23.readHumidity();
	distancia = fnc_ultrasonic_distance(2,4);
	if ((!isnan(temperatura))) {
		lcd_1.setCursor(0, 1);
		lcd_1.print(String("T  ")+String(temperatura)+String("C"));
		lcd_1.setCursor(8, 1);
		lcd_1.print(String(" H ")+String(humedad)+String("%"));
	}
}
```

Anexo G. GRUPO 3

```
#include "BluetoothSerial.h"
#include <analogWrite.h>
#include "ABlocks_DHT.h"
double valorintensidad;
double temperatura;
double datos;
double humedad;
double distancia;
BluetoothSerial bt_serial;
DHT dht23(23,DHT22);

void intensidad() {
	valorintensidad = map(valorintensidad, 0,8,0,254);
	analogWrite(5, valorintensidad);
}
double fnc_ultrasonic_distance(int _t, int _e){
	unsigned long dur=0;
	digitalWrite(_t, LOW);
	delayMicroseconds(5);
	digitalWrite(_t, HIGH);
```

```
        delayMicroseconds(10);
        digitalWrite(_t, LOW);
        dur = pulseIn(_e, HIGH, 18000);
        if(dur==0)return 999.0;
        return (dur/57);
}
void clima() {
        delay(1000);
        temperatura = dht23.readTemperature();
        humedad = dht23.readHumidity();
        distancia = fnc_ultrasonic_distance(14,12);
        Serial.println(distancia);
        if ((!isnan(temperatura))) {
                bt_serial.print(temperatura);
                bt_serial.print(String(";"));
                bt_serial.print(humedad);
                bt_serial.print(String(";"));
                bt_serial.print(distancia);
                bt_serial.println(String(";"));
        }
}
void setup(){
        pinMode(13, OUTPUT);
        pinMode(27, OUTPUT);
        pinMode(18, OUTPUT);
        pinMode(5, OUTPUT);
        pinMode(23, INPUT);
        pinMode(14, OUTPUT);
        pinMode(12, INPUT);

bt_serial.begin(String("ESP32").c_str());
        dht23.begin();
```

```
        Serial.begin(9600);
        Serial.flush();
        while(Serial.available()>0)Serial.read();
}

void loop(){
        yield();
        clima();
        if ((bt_serial.available()>0)) {
                datos = bt_serial.read();
                if ((datos == 11)) {
                        digitalWrite(13, HIGH);
                }

                if ((datos == 10)) {
                        digitalWrite(13, LOW);
                }

                if ((datos == 21)) {
                        digitalWrite(27, HIGH);
                }

                if ((datos == 20)) {
                        digitalWrite(27, LOW);
                }

                if ((datos == 30)) {
                        digitalWrite(18,HIGH);
                }

                if ((datos == 31)) {
                        digitalWrite(18,LOW);
                }
```

```
if (((datos == 40) || (datos == 48))) {
        valorintensidad = 0;
        intensidad();
}

if (((datos == 41) || (datos == 49))) {
        valorintensidad = 1;
        intensidad();
}

if ((datos == 42)) {
        valorintensidad = 2;
        intensidad();
}

if ((datos == 43)) {
        valorintensidad = 3;
        intensidad();
}

if ((datos == 44)) {
        valorintensidad = 4;
        intensidad();
}

if ((datos == 45)) {
        valorintensidad = 5;
        intensidad();
}

if ((datos == 46)) {
        valorintensidad = 6;
        intensidad();
}
```

```
                if ((datos == 47)) {
                        valorintensidad = 7;
                        intensidad();
                }

                if ((datos == 50)) {
                        valorintensidad = 8;
                        intensidad();
                }
        }
}
```

Anexo H. Encuestas

ENCUESTA

Programación Visual

slfernandez_est@utmachala.edu.ec (no compartidos)
Cambiar de cuenta

*Obligatorio

1. ¿Conoce Usted el lenguaje C? *

- Si
- No

2. ¿Considera usted que es un lenguaje fácil de entender? *

- Si
- No

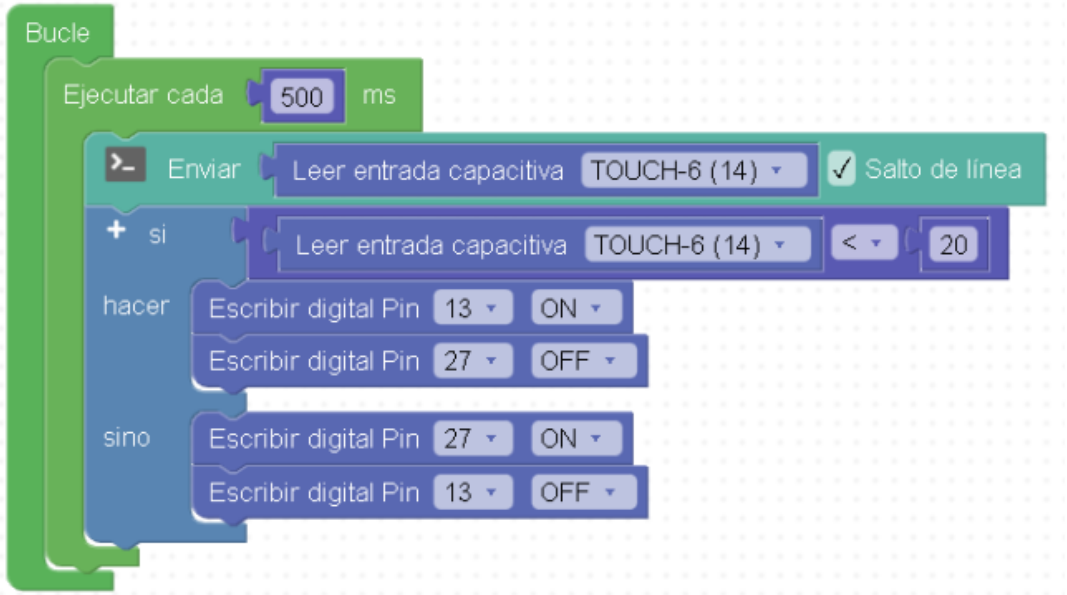
3. ¿Considera usted que es un lenguaje fácil de aprender? *

- Si
- No

4. ¿Conoce usted los lenguajes de Programación Visual? *

- ◯ Si
- ◯ No

5. La siguiente imagen esta hecha en un lenguaje de programación de bloques * ¿Entiende lo que esta realizando el siguiente código?



- ◯ Si
- ◯ No

6. ¿Le resulta fácil entender la imagen anterior? *

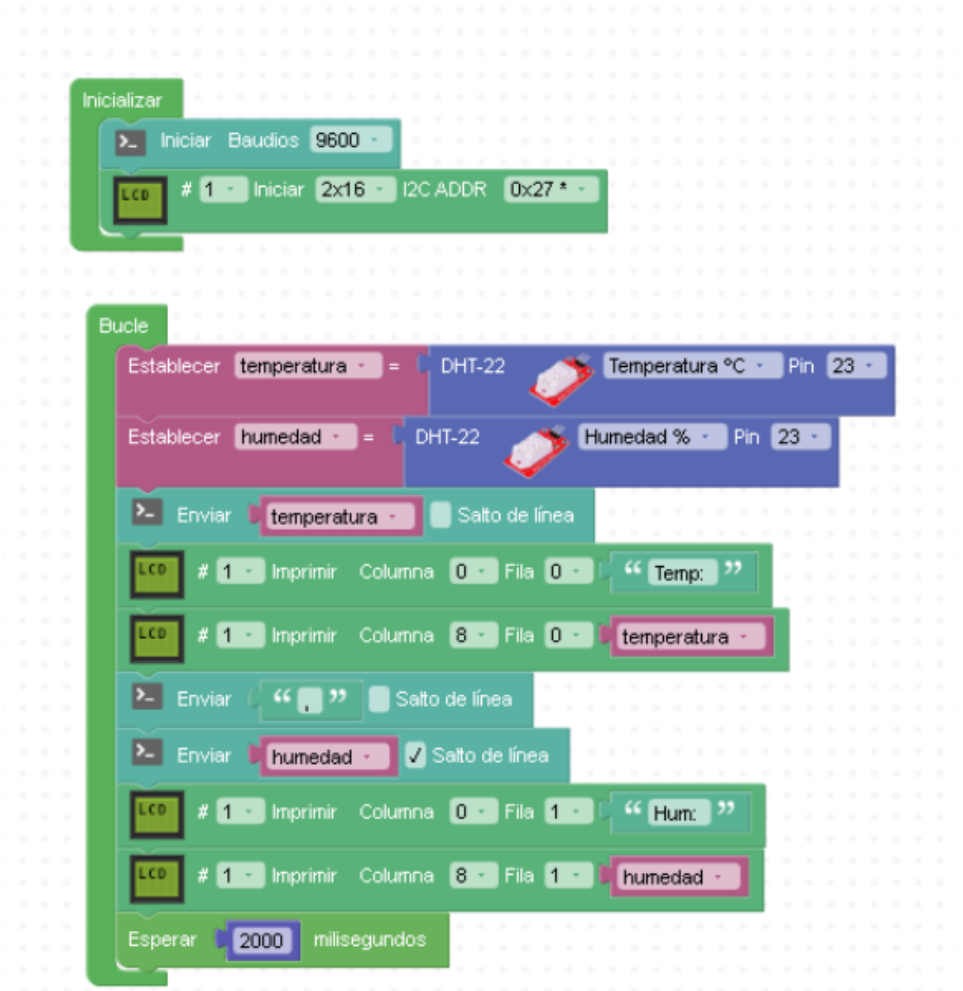
- ◯ Si
- ◯ No

7. La siguiente imagen muestra el código en lenguaje C de la imagen anterior basado en bloques. ¿Con cual de las dos lenguajes le gustaría trabajar ? *

```
unsigned long task_time_ms=0;

void setup()
{
    pinMode(13, OUTPUT);
    pinMode(27, OUTPUT);

    Serial.begin(9600);
    Serial.flush();
    while(Serial.available()>0)Serial.read();

}


void loop()
{
    yield();

    if((millis()-task_time_ms)>=500){
        task_time_ms=millis();
        Serial.println(touchRead(14));
        if ((touchRead(14) < 20)) {
            digitalWrite(13, HIGH);
            digitalWrite(27, LOW);
        }
        else {
            digitalWrite(27, HIGH);
            digitalWrite(13, LOW);
        }

    }

}
```

- ◯ Lenguaje C
- ◯ Programación Visual

8. A continuación se muestra el código de bloques. ¿Cuál seria el resultado de la ejecución? *



- ◯ Muestra la temperatura y humedad en la pantalla LCD
- ◯ Muestra la temperatura, humedad en la consola y pantalla LCD
- ◯ No entiendo el código mostrado