**IMPLEMENTACIÓN DE UNA PLATAFORMA DE PROGRAMACIÓN VISUAL CON BLOCKLY COMO PARTE DEL PROYECTO MADI**

**CARLOS STEVEN ORTIZ COPETE**

**ESCUELA COLOMBIANA DE CARRERAS INDUSTRIALES**

**FACULTAD DE INGENIERÍA MECATRÓNICA**

**PROGRAMA INGENIERÍA MECATRÓNICA**

**BOGOTÁ D.C.**

**AÑO 2019**

**IMPLEMENTACIÓN DE UNA PLATAFORMA DE PROGRAMACIÓN VISUAL CON BLOCKLY COMO PARTE DEL PROYECTO MADI**

**CARLOS STEVEN ORTIZ COPETE**

**Proyecto de investigación**

**DOCENTE**

**FERNEY ALBERTO BELTRAN MOLINA**

**ESCUELA COLOMBIANA DE CARRERAS INDUSTRIALES**

**FACULTAD DE INGENIERÍA MECATRÓNICA**

**PROGRAMA INGENIERÍA MECATRÓNICA**

**BOGOTÁ D.C.**

**AÑO 2019**

**CONTENIDO**

**Pág.**

[1 Definición del problema 1](#_Toc41335616)

[2 Objetivos 2](#_Toc41335617)

[2.1 Objetivo general 2](#_Toc41335618)

[2.2 Objetivos específicos 2](#_Toc41335619)

[2.2.1 Construir bloques básicos de programación visual bajo el entorno de Blockly. 2](#_Toc41335620)

[2.2.2 Construir un prototipo inicial de hardware que permita probar la integración de la programación visual. 2](#_Toc41335621)

[2.2.3 Validar la interface visual de programación con la tarjeta de procesamiento. 2](#_Toc41335622)

[2.2.4 Implementar la web e integrar el sistema con un diseño centrado en el usuario. 2](#_Toc41335623)

[3 Justificación 2](#_Toc41335624)

[4 Marco conceptual 4](#_Toc41335625)

[4.1 Pensamiento computacional 4](#_Toc41335626)

[4.2 Programación por bloques 5](#_Toc41335627)

[4.3 Blockly 6](#_Toc41335628)

[4.3.1 App Inventor 6](#_Toc41335629)

[4.3.2 Micro:bit 7](#_Toc41335630)

[4.3.3 CODE 8](#_Toc41335631)

[4.3.4 AutoBlocks for Jira 8](#_Toc41335632)

[4.3.5 NOVA Labs 9](#_Toc41335633)

[4.4 ESP32 9](#_Toc41335634)

[4.5 MicroPython 10](#_Toc41335635)

[4.6 Wi-Fi 10](#_Toc41335636)

[4.7 Bluetooth 10](#_Toc41335637)

[4.8 PWM 10](#_Toc41335638)

[5 Metodología 11](#_Toc41335639)

[5.1 Objetivo 1 – Construir bloques básicos de programación visual bajo el entorno de Blockly 11](#_Toc41335640)

[5.1.1 Actividad 1.1 - Selección de bloques funcionales y la metodología de construcción 11](#_Toc41335641)

[5.1.2 Actividad 1.2 – Descripción de bloques funcionales en Python 13](#_Toc41335642)

[5.1.3 Actividad 1.3 – Integración de bloques con JavaScript 13](#_Toc41335643)

[5.2 Objetivo 2 - Construir un prototipo inicial de hardware que permita probar la integración de la programación visual 13](#_Toc41335644)

[5.2.1 Actividad 2.1 – Selección de componentes electrónicos 13](#_Toc41335645)

[5.2.2 Actividad 2.2 – Diseño de la tarjeta PCB 13](#_Toc41335646)

[5.2.3 Actividad 2.3 - Ensamblaje y testeo del diseño de hardware 13](#_Toc41335647)

[5.3 Objetivo 3 - Validar la interface visual de programación con la tarjeta de procesamiento 14](#_Toc41335648)

[5.3.1 Actividad 3.1 – Instalación de MicroPython en el procesador ESP32 14](#_Toc41335649)

[5.3.2 Actividad 3.2 – Prueba de MicroPython en el procesador con la tarjeta de desarrollo ESP32 SparkFun 14](#_Toc41335650)

[5.3.3 Actividad 3.3 - Integración de los bloques con MicroPython 14](#_Toc41335651)

[5.3.4 Actividad 3.4 - Prueba del hardware básico y su funcionalidad 15](#_Toc41335652)

[5.4 Objetivo 4 - Implementar la web e integrar el sistema con un diseño centrado en el usuario 15](#_Toc41335653)

[5.4.1 Actividad 4.1 – Configuración del web server 15](#_Toc41335654)

[5.4.2 Actividad 4.2 - Diseño del HTML e integración de los bloques 15](#_Toc41335655)

[6 Resultados 15](#_Toc41335656)

[6.1 Software 15](#_Toc41335657)

[6.1.1 Instalación de MicroPython en el procesador ESP32 15](#_Toc41335658)

[6.1.2 Selección de bloques funcionales 17](#_Toc41335659)

[6.1.3 Creación de bloques bajo el entorno de Blockly 17](#_Toc41335660)

[6.1.4 Programación 19](#_Toc41335661)

[6.1.5 Descargar Código generado en Python. 23](#_Toc41335662)

[6.1.6 Envío Por WebSocket. 24](#_Toc41335663)

[6.2 Hardware 27](#_Toc41335664)

[6.2.1 Selección de componentes eléctricos 27](#_Toc41335665)

[7 Cronograma 31](#_Toc41335666)

[8 Bibliografía 31](#_Toc41335667)

**Tabla de ilustraciones**

[Ilustración 1 Interfaz App Inventor (Tomada de: App Inventor interfaz de programación) 7](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335512)

[Ilustración 2 micro:bit (Tomada de:https://microbit.org/code/) 8](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335513)

[Ilustración 3 Interfaz Code.org (Tomada de: CODE interfaz de programación) 8](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335514)

[Ilustración 4 Interfaz de AutoBlock (Tomada de https://marketplace.atlassian.com/apps/1219915/autoblocks-for-jira?hosting=server&tab=overview) 9](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335515)

[Ilustración 5 Diagrama de bloques de funciones (Tomada de: ESP32 página 12) 11](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335516)

[Ilustración 6 Herramienta para elaborar bloques (Tomado de: Blockly Developer Tools) 12](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335517)

[Ilustración 7 Instalación de Esptool 16](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335518)

[Ilustración 8 Configuración del WebSocket 16](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335519)

[Ilustración 9 Generadores de Blockly Developer Tools 17](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335520)

[Ilustración 10 Bloque de paso de información 18](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335521)

[Ilustración 11 Bloque de recibir información 18](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335522)

[Ilustración 12 Bloque Combinado 19](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335523)

[Ilustración 13 Bloque tipo condicional, bucle y función 19](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335524)

[Ilustración 14 Definición del Bloque, archivo JavaScript 20](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335525)

[Ilustración 15 Carpeta de definición de bloques 20](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335526)

[Ilustración 16 Generador Python 21](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335527)

[Ilustración 17 Carpeta con generadores Python 21](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335528)

[Ilustración 18 Llamado de bloques a través de HTML 22](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335529)

[Ilustración 19 Compilación de Blockly 22](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335530)

[Ilustración 20 Comprobación del bloque 23](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335531)

[Ilustración 21 Programación de un Blink con Blockly 23](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335532)

[Ilustración 22 Código Python generado por Blockly 24](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335533)

[Ilustración 23 Código JavaScript para enviar archivos Python por WebSocket 25](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335534)

[Ilustración 24 Proceso para conexión WebSocket 25](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335535)

[Ilustración 25 Ingreso de Dirección IP de la tarjeta ESP32 26](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335536)

[Ilustración 26 ESP32 realizando Blink con un Led 26](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335537)

[Ilustración 27 Lectura de archivos internos de la tarjeta ESP32 27](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335538)

[Ilustración 28 Distancias del HC-SR04 en centimetros 28](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335539)

[Ilustración 29 Datos obtenidos del MPU6050 29](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335540)

[Ilustración 30 Prueba con NeoPixel 29](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335541)

[Ilustración 31 Buzzer conectado a una ESP32 30](file:///C:\Users\Steve\github\blocky_madi\MADI\MADI\tesis.docx#_Toc41335542)

# Definición del problema

El arte, como lo es la música requiere tiempo, persistencia y un gusto por lo que se hace. Cuando se aprende a tocar un instrumento tradicional, se mejoran las habilidades en varias áreas: mayor creatividad, mejor coordinación corporal, mejora la memoria y la inteligencia sensorial, entre otras.

Por su parte la educación científica y tecnológica, brinda herramientas para desarrollar un pensamiento crítico y aporta destrezas para la solución de problemas. En esta medida, se desarrolla un pensamiento computacional y estructural, que permite plasmar la ciencia en la cotidianidad.

Por lo general, el aprendizaje de estas disciplinas se realiza de forma separada, para lo cual, MADI es un proyecto de la convocatoria interna financiada por la universidad ECCI donde busca diseñar, implementar y evaluar una plataforma de desarrollo electrónico basado en la interacción de la música algorítmica y la computación física (sensores y actuadores), en un contexto académico y hobbista; que opere en tiempo real con lenguajes de programación visuales, concurrentes y editables sobre la marcha.

En este sentido, ¿la programación por bloques es la herramienta más eficaz para el aprendizaje de la computación física y la música algorítmica, aún sin que tengan el mínimo conocimiento de programación?, para ello es necesario generar una herramienta desarrollada en la universidad ECCI con todas las condiciones y requerimiento que permita llevar a cabo implementar herramientas de hardware y programación por bloques, y pruebas para ello, que se adapte a los proyectos MADI y K3OS.

# OBJETIVOS

## Objetivo general

Diseñar e implementar la interfaz web de programación visual y de música algorítmica tipo blockly, para plataformas con soporte MicroPython.

## Objetivos específicos

### Construir bloques básicos de programación visual bajo el entorno de Blockly.

### Construir un prototipo inicial de hardware que permita probar la integración de la programación visual.

### Validar la interface visual de programación con la tarjeta de procesamiento.

### Implementar la web e integrar el sistema con un diseño centrado en el usuario.

# Justificación

Gracias a que la programación visual fortalece el pensamiento estructural y computacional, varios autores e investigadores han trabajado en el aula de clase la relación de la programación con la electrónica, en especial la robótica junto con IoT; que algunos llaman la cuarta revolución Industrial. En este sentido, el pensamiento computacional, el diseño y la construcción de artefactos conectados entre sí, permite dar las herramientas necesarias a las nuevas generaciones que afrontan los vertiginosos avances tecnológicos (Harms, Balzuweit, Chen, & Kelleher, 2016).

En esta línea argumentativa, se mueve la educación STEM, que es la forma de enseñar con el objetivo de integrar las cuatro grandes áreas: ciencia, tecnología, ingeniería y matemáticas, donde, las personas tienen un aprendizaje significativo, colaborativo y vivencial. En este sentido, otros autores integran las Artes al modelo de aprendizaje STEAM (Science, Technology, Engineering, Arts and Mathematics). Esta filosofía de aprendizaje busca las intersecciones entre las 5 áreas de conocimientos y permitir al estudiante un enfoque interdisciplinar en los procesos de enseñanza-aprendizaje (Sanders, 2009).

En esta filosofía, se evidencia la gran aceptación de lenguajes de programación visuales como Scratch, blocky y tarjetas o sistemas electrónicos como Arduino, Raspberry Pi, Microbit, Lego Mindstorm y robot NAO, como herramientas que apoyan el modelo STEAM. Microbit es una de las más recientes propuestas de sistemas electrónicos para fomentar la educación STEAM, liderada por la BBC (British Broadcast Coorporation). Los estudios realizados por la King’s College de Londres en relación a la validación de la tarjeta evidencia que los estudiantes que usan Microbit tienen una mejor disposición para el estudio de las 5 áreas (STEAM). Dichas herramientas de hardware y de software son muy funcionales en ambientes controlados como aulas de clase y laboratorios.

Por otra parte, se evidencia que en los últimos años se desarrollaron iniciativas que buscan incluir la música en el marco filosófico STEAM. Es así como han surgido lenguajes de programación musical, como es el caso de Sonic Pi y Cruck. Estos entornos de música algorítmica se centran en explorar a partir de la síntesis y generación de sonidos musicales (Aaron & Blackwell, 2013). Ahora bien, integrar la música algorítmica al aula de clase y el mundo hobbista, es un reto que actualmente se está desarrollando.

En este contexto, los actuales desarrollos de música algorítmica se centran en el uso de un computador, dejando a un lado la interacción con el mundo físico. Por lo que, se evidencia una línea de desarrollo e investigación poco estudiada, como lo es, la interacción de lenguajes de programación musical en tiempo real y concurrente con tarjetas de desarrollo electrónico.

Teniendo como evidencia lo anterior, el proyecto MADI deberá contar con una plataforma de programación la cual integre las características mencionadas anteriormente.

# Marco conceptual

## Pensamiento computacional

El ser humano tiene la capacidad de resolver problemas a través de las experiencias que ha adquirido en su vida y ha solucionado problemas a través de la historia para su supervivencia, estas se han ido transmitiendo de generación en generación evolucionando las herramientas para resolverlos. Junto con las herramientas de la informática y sus limitaciones, el pensamiento computacional brinda el poder para resolver problemas y diseñar sistemas que no se podría abordar de manera individual. Según (Jeannette M. Wing) el pensamiento computacional es una habilidad fundamental para todos, no sólo para los científicos informáticos, sino también para fortalecer la lectura, escritura y la aritmética.

Actualmente se está evidenciando un mundo de constante cambio, donde siempre van a surgir necesidades adicionales, el mundo se está digitalizando y es imposible vivir separados de las herramientas tecnológicas. Estas se deben aprovechar preguntándose ¿Hasta qué punto un ser humano puede llegar a realizar una tarea?, ¿Hasta qué punto una maquina puede solucionar un problema? (Jeannette M. Wing, 2006). El pensamiento computacional permite la solución de estos ideando estrategias o sistemas que permitan llegar a una solución donde ni el uno ni el otro puedan hacerlo por separado.

Esto hace que sea necesario que a temprana edad se pueda perfeccionar esta habilidad, permitiendo así que las personas puedan desarrollar sus ideas enfrentándose a varias adversidades que trae consigo la vida cotidiana y teniendo en cuenta que se están trabajando las competencias abstracto-matemático con el pragmático-ingenieril (Berrocoso, Rosa, Sánchez, Del Carmen, & Arroyo), sin necesidad de ser bueno en ciertas áreas del conocimiento como las matemáticas y dándole importancia a las ideas. Según (Zapata-Ros) “lo importante es saber cómo se representa la realidad, el mundo de objetivos y expectativas” haciendo que sea importante lo que piensan y como lo piensan, como abstraen el problema y solucionar cada parte de este para finalmente integrarlo.

## Programación por bloques

La programación es el medio por el cual se puede llegar a dar vida a las ideas y a través de este el ser humano se comunica con la máquina dándole pasos a seguir. La programación es una extensión de la escritura (Sáez-López & Cózar-Gutiérrez, 2017), por lo tanto, al escribir código las personas plasman sus ideas de forma que la máquina entienda lo que se va a realizar y según (Berrocoso, Rosa, Sánchez, Del Carmen, & Arroyo) se aprende a organizar un proceso donde reconoce rutinas o repeticiones y además de ello, se perfecciona el pensamiento computacional descubriendo sus errores cuando el programa no funciona según se tenía planeado o no alcanza la expectativa deseada. Aquí el problema es ¿en qué lenguaje programar?, cuando las personas empiezan a programar se enfrentan a varios obstáculos, dos de ellos son la lógica de programación y el propio lenguaje de programación (sintaxis), haciendo que el proceso pueda llegar a ser tedioso e incluso aburrido para algunas personas, ya que se les dificulta aprender la sintaxis de este lenguaje dado que es como aprender un nuevo idioma. Pero saber la composición de algún lenguaje de programación no es saber programar, porque como anteriormente se mencionó, es un tipo de idioma más y sí se toma de esta forma no necesariamente los poliglotas saben programar.

La idea al empezar a programar es poder desarrollar el pensamiento computacional y entre más joven se comience a trabajar esta habilidad será mejor. Teniendo en cuenta lo anterior, enseñar a programar puede llegar a ser aburrido para alguien que centra la mayoría de su atención en algo que le parezca más didáctico.

La programación por bloques rompe el paradigma de la programación estructurada, ya que a diferencia de esta, la programación por bloques es más visual que escrita, haciendo que no sea muy elemental aprender la sintaxis de un lenguaje, logrando que la persona se enfoque más en saber qué está haciendo y asegurándose que el programa este bien escrito, este enfoque permite que se desarrolle mejor el pensamiento computacional ya que pasará la mayor parte del tiempo en el proceso de plasmar sus ideas.

## Blockly

Para Google, Blockly es una biblioteca que agrega un editor de código visual a aplicaciones web y móviles. Este editor utiliza bloques gráficos entrelazados para representar conceptos de código como variables, expresiones lógicas, bucles y más. Este editor presenta la gran ventaja de permitir a los usuarios aplicar principios de programación sin tener que preocuparse por la sintaxis. (Google, 2018).

Blockly permite exportar código a diferentes tipos de lenguajes estructurados tales como JavaScript, Python, PHP, Lua y Dart, gracias a esto lo convierte en una excelente herramienta y punto de partida para aplicaciones tales como lo son;

### App Inventor

MIT App Inventor es un entorno de programación visual e intuitiva que permite a todos, incluso niños, crear aplicaciones totalmente funcionales para teléfonos inteligentes y tabletas. Los principiantes en App Inventor pueden tener una primera aplicación simple en funcionamiento en menos de 30 minutos. Y lo que es más, esta herramienta basada en bloques facilita la creación de aplicaciones complejas y de alto impacto en mucho menos tiempo que los entornos de programación tradicionales. El proyecto MIT App Inventor busca democratizar el desarrollo de software al empoderar a todas las personas, especialmente a los jóvenes, para pasar del consumo de tecnología a la creación de tecnología. (MIT, 2012)

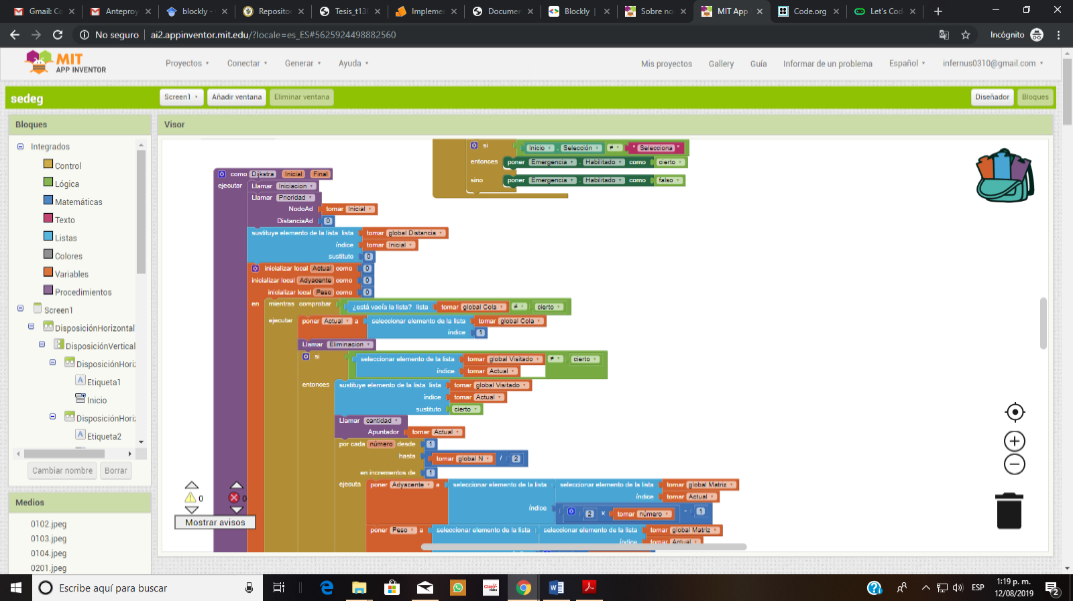


Ilustración 1 Interfaz App Inventor (Tomada de: App Inventor interfaz de programación)

### Micro:bit

BBC micro:bit es un micro-computador programable que cabe en la mano y que puede usarse para todo tipo de fantásticas invenciones: desde robots a instrumentos musicales.

Se puede programar desde cualquier navegador web en Bloques, Javascript, Python, Scratch y más; no se requiere ningún otro software. (Micro:Bit Educational Foundation, s.f.)

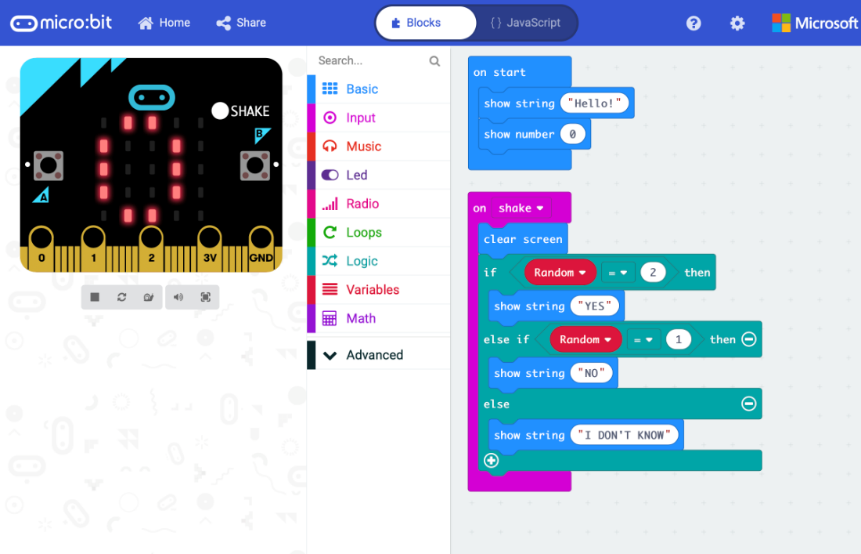


Ilustración 2 micro:bit (Tomada de:https://microbit.org/code/)

### CODE

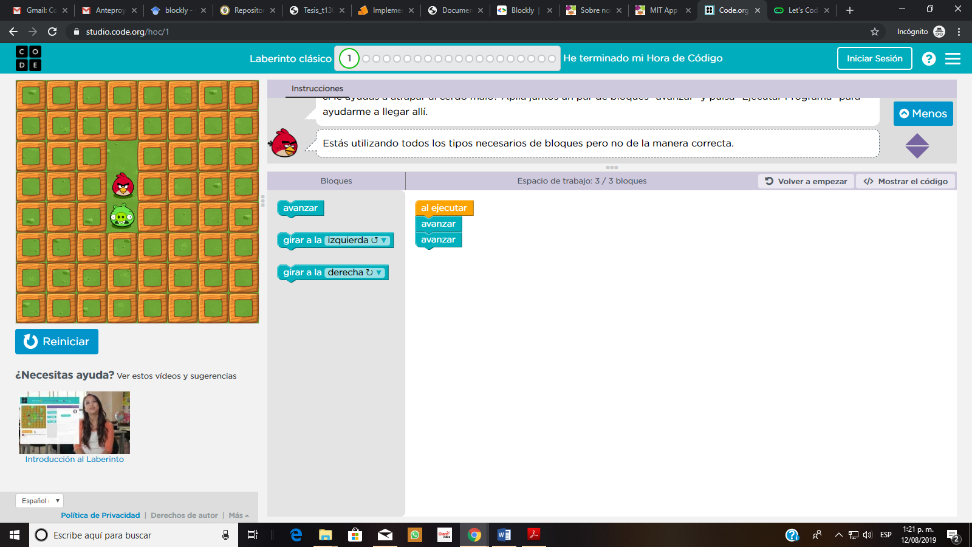
Code.org es una organización sin fines de lucro, dedicada a expandir el acceso a Ciencias de la Computación; haciéndola disponible en más escuelas y a aumentar la participación de las mujeres y minorías sub-representadas. Su visión es que cada estudiante en cada escuela tenga la oportunidad de aprender informática, de la misma manera que biología, química o álgebra. (Code, 2013)

Ilustración 3 Interfaz Code.org (Tomada de: CODE interfaz de programación)

### AutoBlocks for Jira

Basado en el Google Blockly Framework: AutoBlocks es una pizarra virtual que democratiza la automatización para los usuarios de Jira, permitiendo la personalización de Jira con tecnología fácil de arrastrar y soltar.

Empoderar a los usuarios de Jira para construir sus propias automatizaciones, una vez que estén listas, enviarlas a Jira Admin para su revisión y aprobación. (Atlassian Marketplace, 2019)

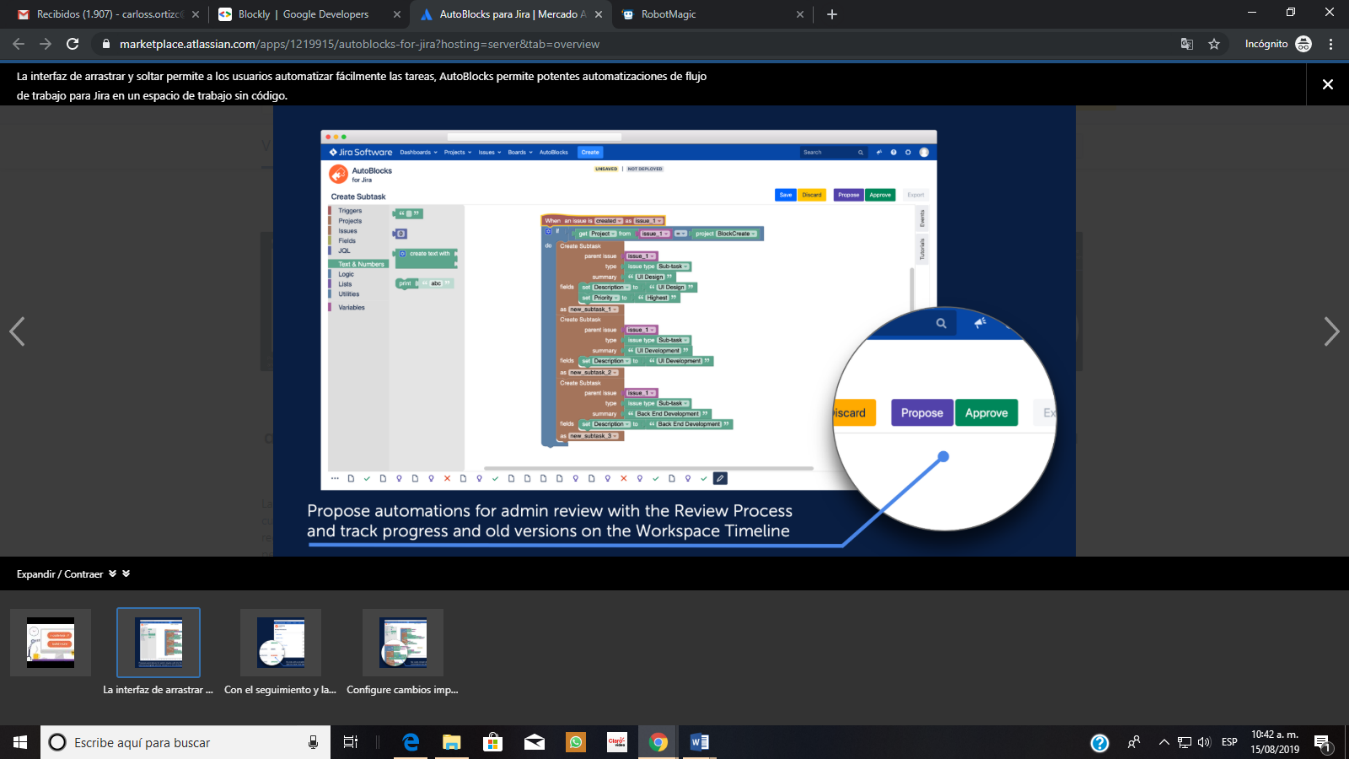


Ilustración 4 Interfaz de AutoBlock (Tomada de https://marketplace.atlassian.com/apps/1219915/autoblocks-for-jira?hosting=server&tab=overview)

### NOVA Labs

NOVA Labs es una plataforma digital gratuita que involucra a adolescentes y estudiantes en juegos interactivos que fomentan la exploración científica auténtica. Desde la predicción de tormentas solares y la construcción de sistemas de energía renovable hasta el seguimiento del movimiento de las nubes y el diseño de moléculas de ARN, los participantes de NOVA Labs pueden realizar investigaciones visualizando, analizando y compartiendo los mismos datos que usan los científicos. (NOVA Labs, 2019)

## ESP32

Es un procesador elaborado por Espressif Systems, cuenta con capacidades IoT, y que permite la conexión mediante Wi-Fi (802.11 b/g/n/e/i) y Bluetooth versión 4.2 y Bluethooth de baja energía (BLE). (Systems, 2016)

De fábrica trae el firmware para ser programado desde el IDE de Arduino, pero este puede ser cambiado por el firmware de MicroPython.

## MicroPython

Este es un compilador de Python, el cual obtiene un mensaje interactivo (REPL) para ejecutar comandos desde una consola. Con capacidad de ejecutar e importar archivos integrados.

MicroPython según sus desarrolladores es bastante compatible con Python. (MicroPython, 2018)

## Wi-Fi

Es una marca comercial de Wi-Fi Alliance, tiene como objetivo fomentar las conexiones inalámbricas y la compatibilidad entre equipos. (Alliance, 2020)

Actualmente se ha popularizado a tal grado que teniendo un dispositivo compatible con Wi-Fi se puede conectar a la red inalámbricamente desde cualquier parte del mundo, esto es debido a la gran ventaja de esta tecnología en cuanto a la ausencia de cables.

## Bluetooth

Es una especificación tecnológica a través de redes de área personal inalámbricas (WPAN) mediante una radio frecuencia de 2.4 GHz permitiendo la transmisión de datos bajo el estándar de comunicación inalámbrica IEEE 802.15.1. (Definición , 2009) (Bluetooth, 2020)

## PWM

La modulación por ancho de pulsos es una seña cuadrada la cual se puede variar cuánto dura esta señal en el nivel alto (generalmente VCC) y nivel bajo (generalmente GND), durante un periodo de tiempo determinado. Lo que se logra con esto es variar la tensión media, esta variación puede hacer que el comportamiento en un componente cambie, por ejemplo, el brillo en un LED o la velocidad en un motor de corriente continua (Gómez, 2018).

# METODOLOGÍA

## Objetivo 1 – Construir bloques básicos de programación visual bajo el entorno de Blockly

### Actividad 1.1 - Selección de bloques funcionales y la metodología de construcción

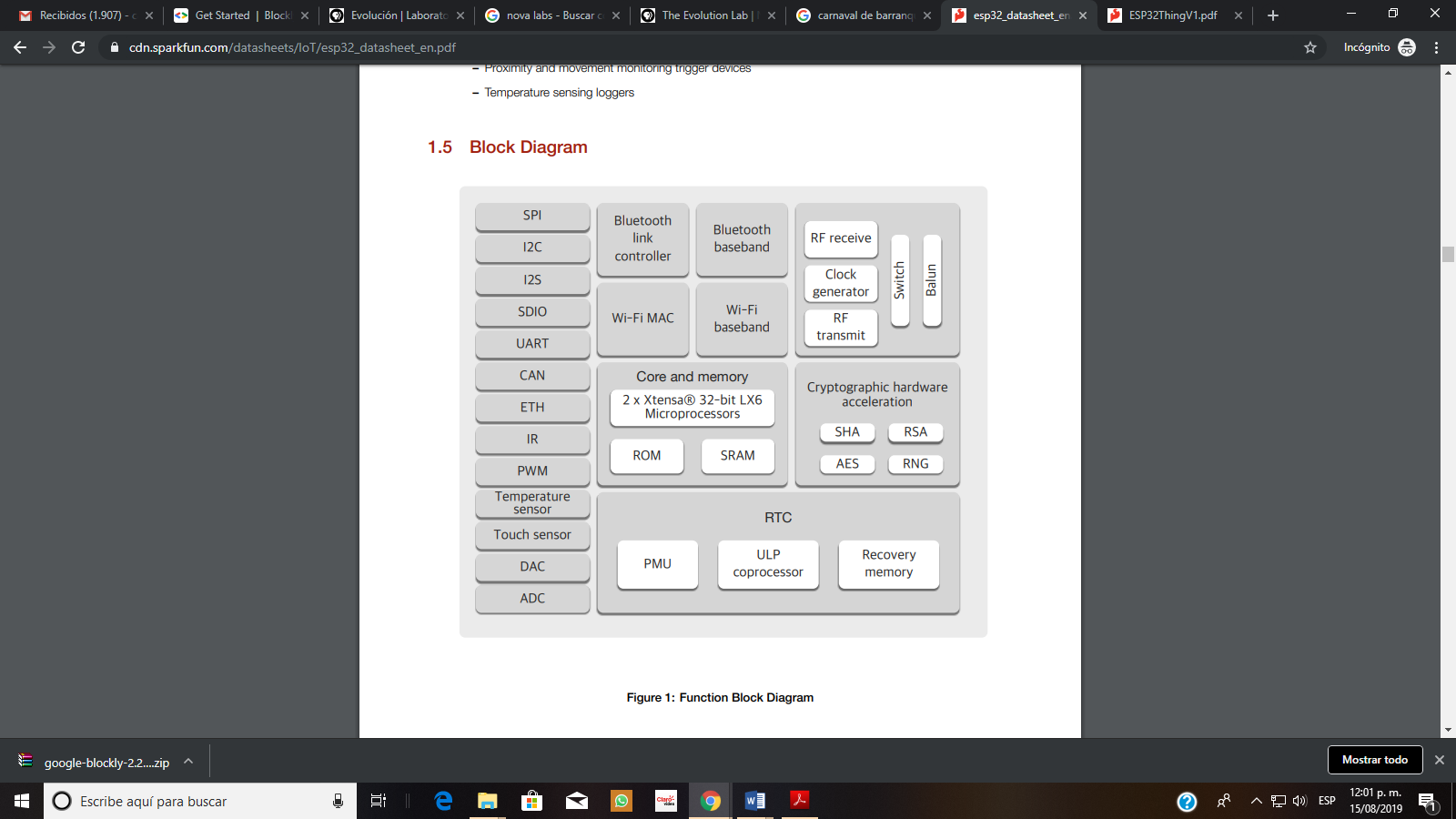
Se trabajará la tarjeta ESP32 con el Firmware de MicroPython, para lo cual se deben seleccionar el tipo de bloques que necesita y las especificaciones de la tarjeta de desarrollo (véase Ilustración 5). Estos bloques pueden ser WI-FI, Bluetooth, PWM, entre otros.

Ilustración 5 Diagrama de bloques de funciones (Tomada de: ESP32 página 12)

El paquete de demostración de Blockly (Google), incluye el Blockly Developer Tools (véase Ilustración 6), el cual permite diseñar bloques dependiendo de la función que va a cumplir.

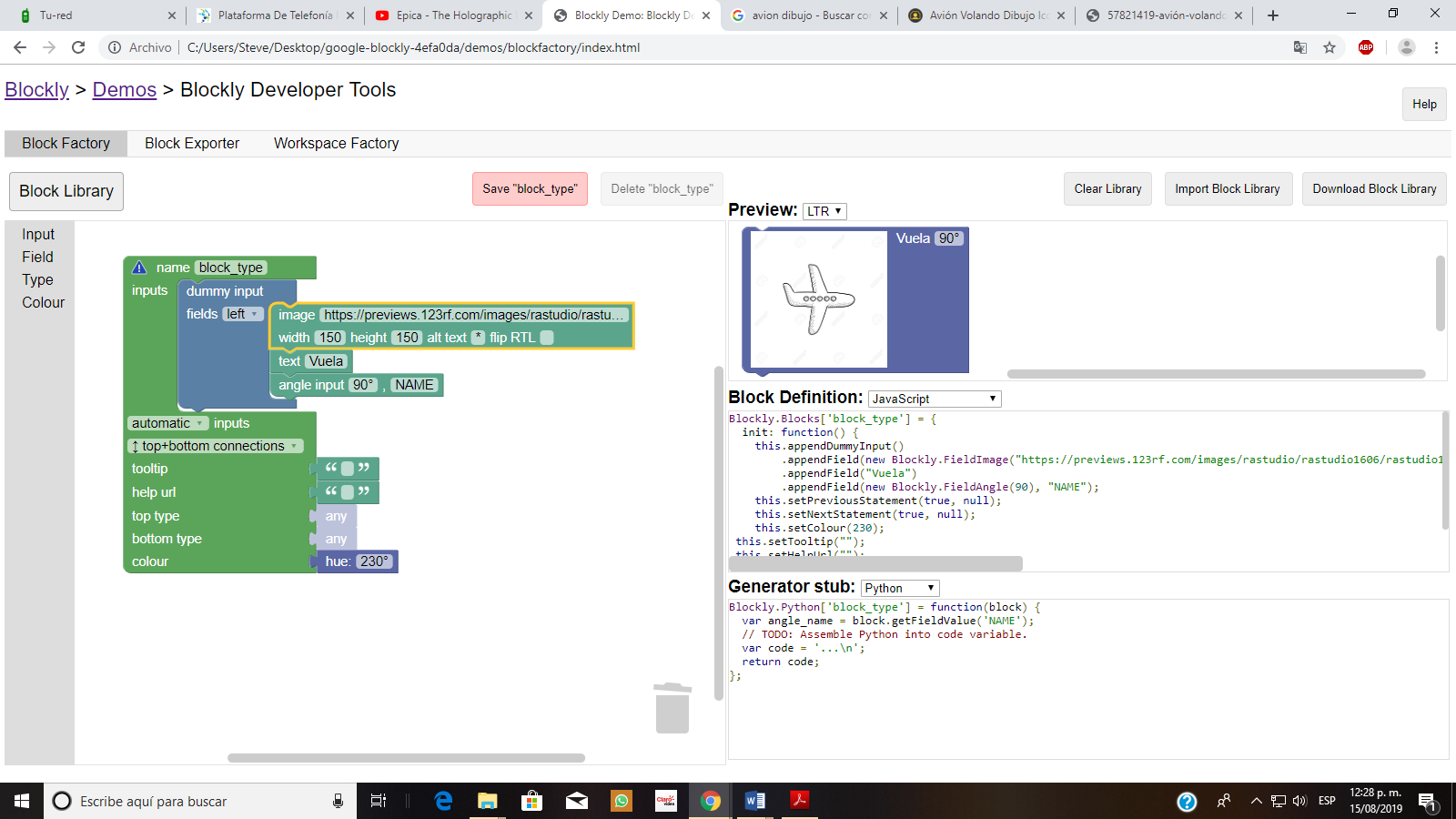


Ilustración 6 Herramienta para elaborar bloques (Tomado de: Blockly Developer Tools)

Dependiendo del tipo de bloque, este deberá cumplir algunas especificaciones de construcción, tales como;

* Paso de información: Los bloques con esta funcionalidad tienen información contenida y se podrá pasar a otros bloques.
* Recibir información: Estos bloques reciben la información de los anteriores. Estos bloques pueden procesar esa información y hacer algo en específico, un ejemplo de estos son los que llaman las funciones y envío por puerto serial.
* Combinados: Son una mezcla de las dos anteriores características, estos bloques reciben información de otros bloques, la procesan y pasan la información a otros bloques. Un ejemplo de estos bloques son los ingresos a funciones con retorno.
* Condicionales, bucles y funciones: Estos bloques pueden recibir información y poderla pasar, y en su interior pueden ingresar más bloques para realizar procesos.

### Actividad 1.2 – Descripción de bloques funcionales en Python

Teniendo el tipo de bloque que se va a realizar, se usa la herramienta constructora de bloques. Se crea el bloque dependiendo del tipo y el Blockly Developer Tools devuelve un código JavaScript con espacio para un programa y aquí se le asigna un código en Python dependiendo de su funcionalidad.

### Actividad 1.3 – Integración de bloques con JavaScript

El programa base de Blockly está escrito en JavaScript y HTML, y es aquí donde se debe integrar cada bloque que se construya, esto se hace agregando el código el bloque al programa base y compilándolo, esto ya añadirá el bloque construido a Blockly.

## Objetivo 2 - Construir un prototipo inicial de hardware que permita probar la integración de la programación visual

### Actividad 2.1 – Selección de componentes electrónicos

Con las funcionalidades en hardware ya definidas, se selecciona cada componente electrónico que cumpla con esta funcionalidad, una de estas funcionalidades es la música algorítmica, por tanto, se debe seleccionar un componente que permita reproducir esta música programada.

### Actividad 2.2 – Diseño de la tarjeta PCB

Ya con cada componente definido, se realiza el diseño de la tarjeta PCB con el circuito que permita el funcionamiento de cada uno de ellos y la tarjeta de desarrollo.

### Actividad 2.3 - Ensamblaje y testeo del diseño de hardware

Con la tarjeta ya hecha, se procede a ensamblarla con cada uno de sus componentes. Luego de tenerla lista, se realizan las pruebas de funcionamiento, las cuales consisten en realizar un programa, enviar el código a la tarjeta y se verifica que cada componente este funcionando correctamente.

## Objetivo 3 - Validar la interface visual de programación con la tarjeta de procesamiento

### Actividad 3.1 – Instalación de MicroPython en el procesador ESP32

Las tarjetas ESP32 traen de fábrica el firmware para ser programados por el IDE de Arduino, por tanto, se debe cambiar para ser programado por MicroPython. Para poder realizar este procedimiento se debe descargar el firmware para MicroPython y seguir las instrucciones del fabricante (estas instrucciones se encuentran en la página de SparkFun) las cuales consisten en eliminar el firmware actual e instalar el de MicroPython y luego verificar por una terminal que fue instalado correctamente.

### Actividad 3.2 – Prueba de MicroPython en el procesador con la tarjeta de desarrollo ESP32 SparkFun

Luego de haber instalado el Firmware se procede a realizar pruebas con MicroPython corriendo en la ESP32, para esto se realiza un Blink. Se usa el programa Atom para realizar el código correspondiente y se carga en la tarjeta de desarrollo, y se valida que el led integrado en la ESP32 parpadee.

### Actividad 3.3 - Integración de los bloques con MicroPython

Para poder integrar los bloques con MicroPython, se realiza un programa tipo Blink en Blockly y el programa resultante se copea y se pega en un archivo en Atom y se carga el programa a la tarjeta para luego verificar su funcionamiento. Luego de lo anterior se le agrega la funcionalidad a Blockly para poder descargar el archivo .py, para así descargarlo y poder abrirlo en Atom y cargar el programa para revisar su funcionamiento.

### Actividad 3.4 - Prueba del hardware básico y su funcionalidad

Se hace la integración entre el hardware y el Blockly, para esto, se realiza un programa en bloques con todas las funcionalidades agregadas en el hardware, con esto se envía a la tarjeta de desarrollo y se ejecuta este programa y se revisa que todas las funcionalidades se estén realizando correctamente según lo que se programó.

## Objetivo 4 - Implementar la web e integrar el sistema con un diseño centrado en el usuario

### Actividad 4.1 – Configuración del web server

La tarjeta ESP32 cuenta con la funcionalidad WebSocket, el cual es un canal de comunicación full-duplex (Pardo, 2019). Esto puede servir como medio para enviar el código generado por Blockly sin tener que conectar la tarjeta físicamente al dispositivo desde donde se está utilizado la plataforma.

### Actividad 4.2 - Diseño del HTML e integración de los bloques

La plataforma debe ser amigable con el usuario, para lo cual, la distribución de bloques, botones y área de trabajo debe ser ordenada y que el usuario pueda familiarizarse con la plataforma de forma eficaz.

# RESULTADOS

## Creación de bloques funcionales

Se definieron unos bloques funcionales que sirvan para crear instrumentos musicales, estos están divididos en categorías y cada funcionalidad puede tener hasta 3 bloques, ya sea el importador, que es el que crea el objeto con los respectivos pines para su funcionamiento, el funcional que es el que activara las funciones del objeto anterior y por ultimo un bloque extra para alguna otra funcionalidad o configuración:

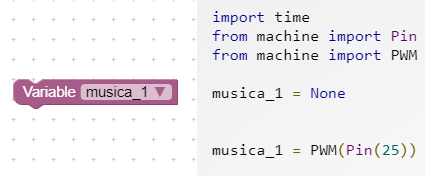
### Notas

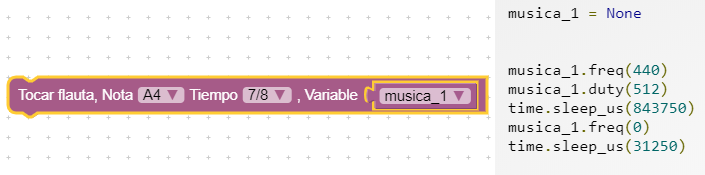
En esta categoría se encuentran todos los bloques que manejan la parte musical. Se simuló el sonido con un PWM y un BUZZER, una flauta, un piano, un xilófono, un instrumento de percusión, y una guitarra eléctrica.

Para poder generar una mayor gama de sonidos se tomaron las notas musicales (Do, Do#,Re, Re#,Mi,Fa,Fa#,Sol, Sol#,La, La# y Si) para diferentes instrumentos musicales, y desde su segunda octava Do2 hasta su octava octava Do8, cada nota musical en cada una de sus escalas tiene una frecuencia característica, según (Web Archive Org, 2006).

Para poder diferenciar un instrumento del otro se le asignó un porcentaje para el ciclo útil del PWM, esta asignación se realizó de manera empírica y se relacionaron los sonidos generados con instrumentos existentes, de esta manera se obtuvieron los instrumentos nombrados anteriormente.

Ver ilustración tal del importador, y ver ilustración tal del funcional (flauta).

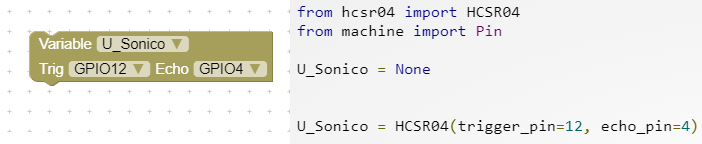




### Sensores

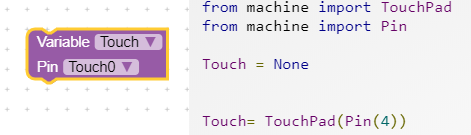
Debido a la situación con el Covid-19, la categoría “Sensores” tiene consigo bloques con módulos que se habían usado con anterioridad en otros proyectos siendo estos el MPU6050 y el HC-SR04, además de ello se agregó un bloque para controlar el TouchPad que trae la tarjeta ESP32.

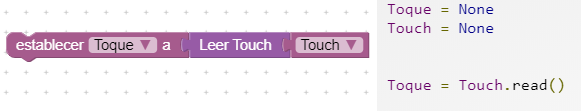
Ver ilustración tal del importador, y ver ilustración tal del funcional del sensor de ultrasonido.





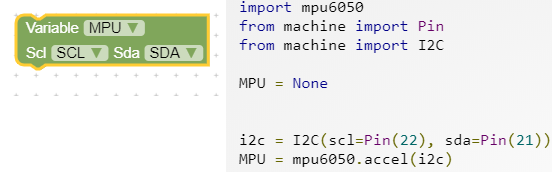
Ver ilustración tal del importador, ver ilustración tal del funcional y la ilustracion tal del configurador del Touchpad.

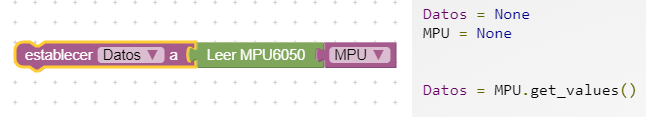






Ver ilustración tal del importador, ver ilustración tal del funcional y la ilustracion tal del extra del MPU6050.



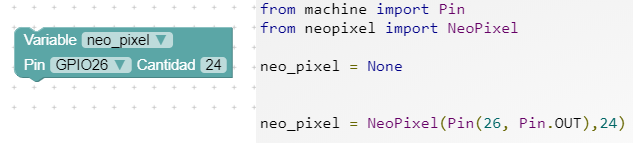


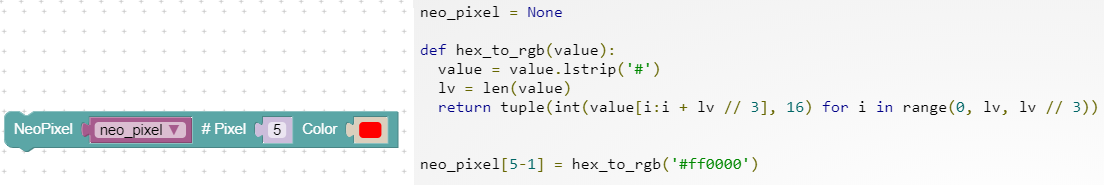


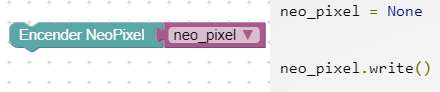
### NeoPixel

Como parte visual se agregó esta categoría para controlar los NeoPixel, estos son fabricados por Adafruit, y son diodos LED de tipo 5050 con un controlador WS2812 integrado en cada píxel (BricoGeek, s.f.), esto permite controlarlos mediante un solo hilo.

Ver ilustración tal del importador, ver ilustración tal del funcional y la ilustracion tal del extra del NeoPixel.







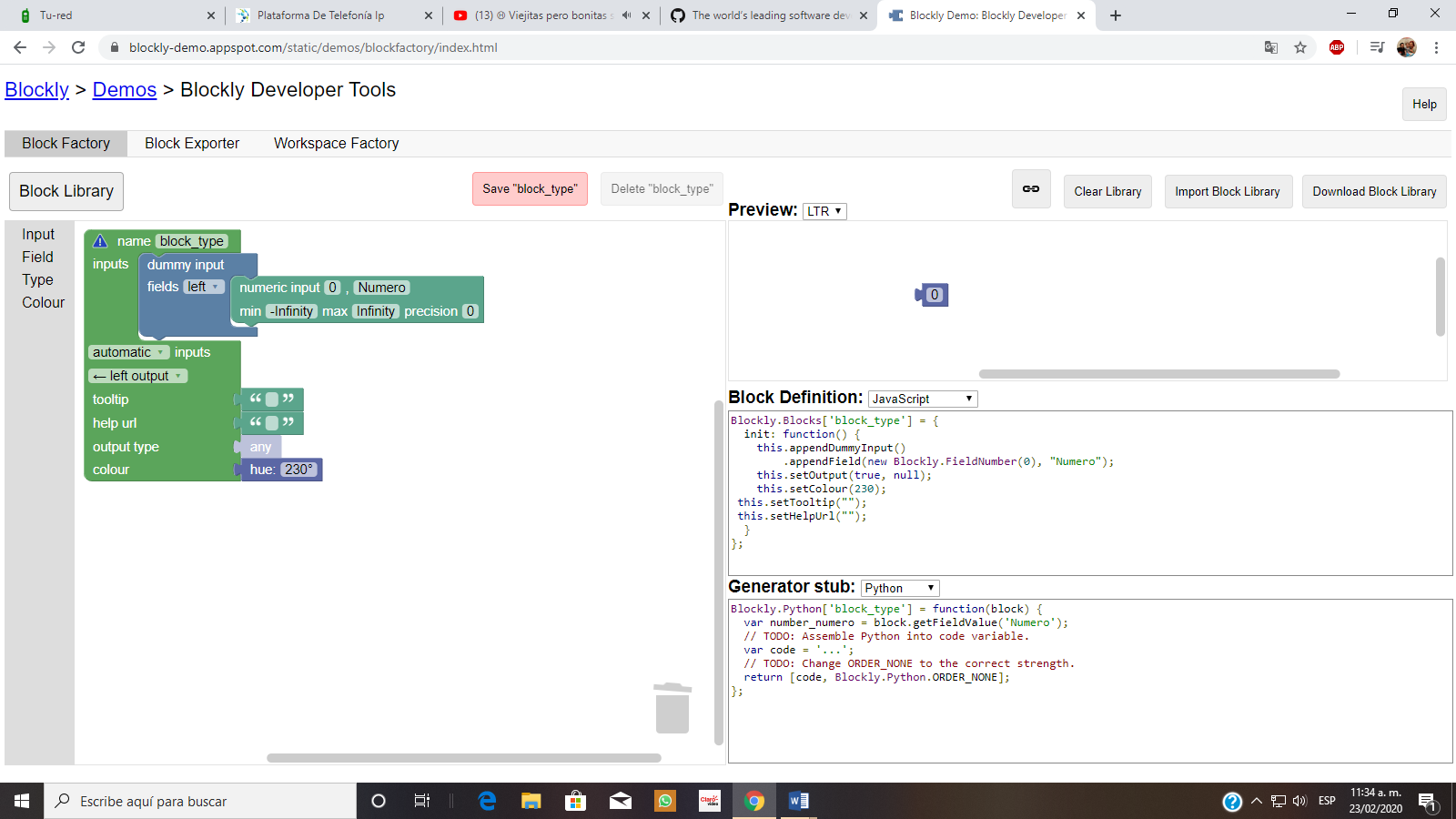
### Tiempo

En la música el tiempo es una parte importante, para lo cual, la categoría tiempo cuenta con bloques de retrasos, además de ello tiene bloques para funcionalidades de tiempo real y bloques con funcionalidades de crear interrupciones por tiempo.

## Construcción de bloques

Usando Blockly Developer Tools permite crear bloques según sea su tipo.

En la Ilustración 9 se puede evidenciar la definición del bloque en JavaScript y su generador en Python. Los dos códigos generados al crear cada uno de los bloques son fundamentales al momento de hacer la integración con el programa base, la definición del bloque contiene su forma, y el generador contiene el fragmento de código que corresponde al bloque dependiendo de su funcionalidad



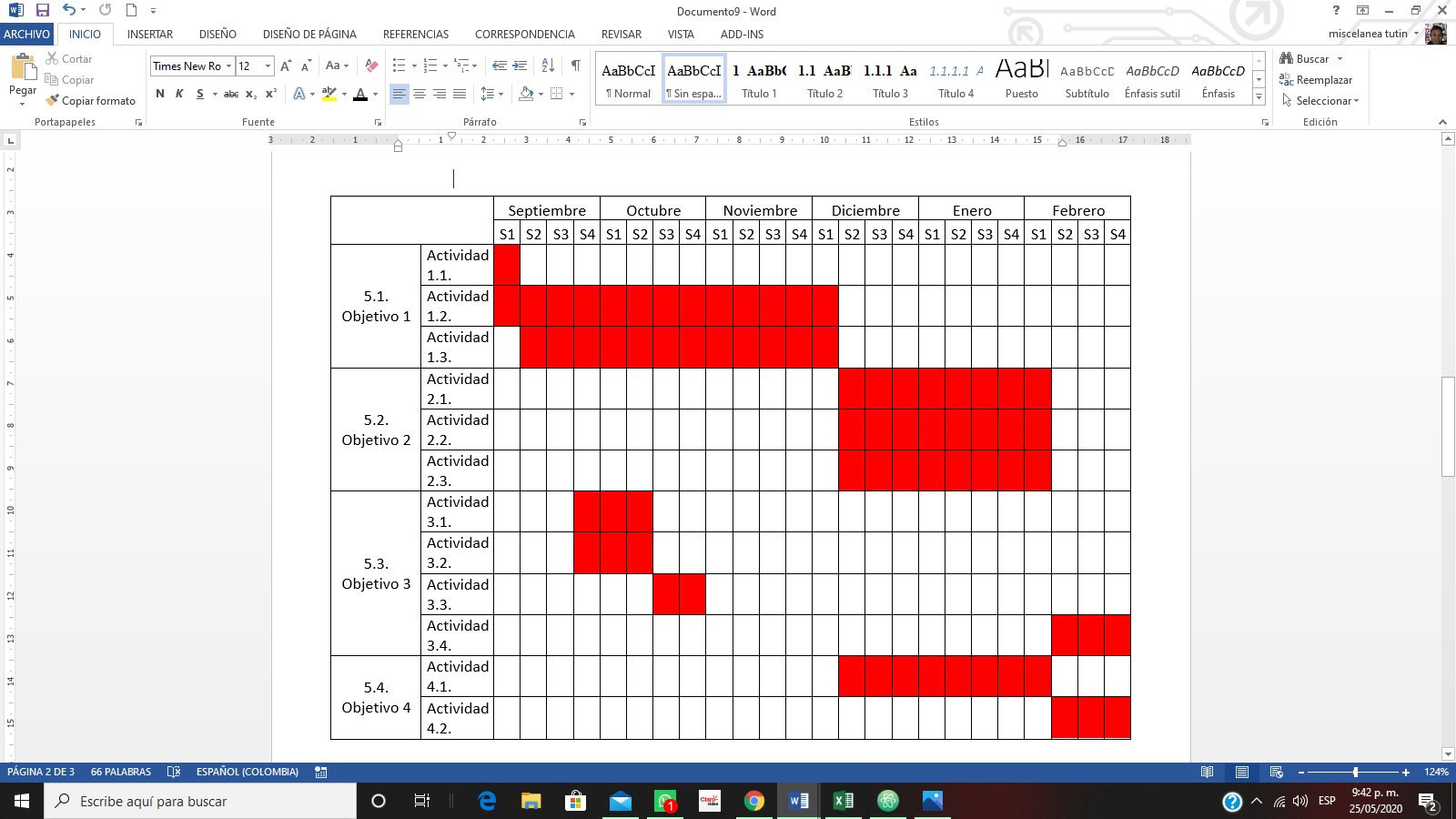
# CONCLUSIONES

* MicroPython es un lenguaje de programación muy versátil, pero apenas está teniendo su auge en el caso de las tarjetas ESP32, por lo tanto, para algunas de las funcionalidades de la tarjeta no hay mucha documentación o soporte.
* Como herramienta para la enseñanza de la programación estructurada, este proyecto podría ayudar a facilitar el aprendizaje de esta habilidad ya que no debe aprender la sintaxis de algún lenguaje para utilizarla.
* Además de facilitar el aprendizaje de la programación estructurada, crea un puente para el aprendizaje de Python ya que la plataforma tiene la opción de mostrar el código en Python de lo que se está programando con los bloques.
* La funcionalidad de poder enviar el código de la plataforma a la tarjeta de desarrollo presenta una gran ventaja a la hora de realizar proyectos con un enfoque IoT, ya que se puede realizar a través de la red.
* Puede crearse cualquier tipo de instrumento musical, el límite lo pone la imaginación.
* De igual manera como Blockly permite crear bloques y tener en su interior un código predefinido en Python, puede llegar a contener cualquier otro tipo de lenguaje de programación desde que su sintaxis sea escrita, por tanto, cumple con el objetivo principal.

# Cronograma

Las tareas mostradas en Tabla 1 Cronograma, son aquellas mostradas en el Diseño Metodológico.

Tabla 1 Cronograma



# BIBLIOGRAFÍA

Aaron, S., & Blackwell, A. F. (2013). From Sonic Pi to Overtone: Creative Musical Experiences with Domain- Specific and Functional Languages. *ACM SIGPLAN Workshop on Functional Art, Music, Modeling, and Design*, 35-46.

Alliance, W.-F. (2020). *Wi-Fi Alliance*. Obtenido de Who We Are: https://www.wi-fi.org/who-we-are

Atlassian Marketplace. (2019). *AutoBlocks for Jira*. Obtenido de More details: https://marketplace.atlassian.com/apps/1219915/autoblocks-for-jira?hosting=server&tab=overview

Berrocoso, J., Rosa, M., Sánchez, F., Del Carmen, M., & Arroyo, G. (s.f.). *RED-Revista de Educación a Distancia El pensamiento computacional y las nuevas ecologías del aprendizaje Computacional thinking and new learning ecologies.*

Bluetooth. (2020). *Bluetooth*. Obtenido de Bluetooth: https://www.bluetooth.com/

BricoGeek. (s.f.). *LED NeoPixel*. Obtenido de BricoGeek: https://tienda.bricogeek.com/110-led-neopixel

Code. (2013). *Code*. Obtenido de About Us: https://code.org/international/about

Definición . (2009). *Definición* . Obtenido de Bluetooth: https://definicion.de/bluetooth/

Gómez, E. (2018). *Rincon Ingenieril*. Obtenido de PWM: https://www.rinconingenieril.es/que-es-pwm-y-para-que-sirve/

Google. (20 de Septiembre de 2018). *Google for Education - Blockly*. Obtenido de Introduction to Blockly: https://developers.google.com/blockly/guides/overview

Harms, K. J., Balzuweit, E., Chen, J., & Kelleher, C. (2016). Learning programming from tutorials and code puzzles: Children's perceptions of value. In Visual Languages and Human-Centric Computing (VL/HCC). *IEEE Symposium*, 59-67.

Jeannette M. Wing. (2006). Computational Thinking. *Viewpoint, 49*(3).

Keyence Corporation. (s.f.). *¿Qué es un sensor ultrasónico?* Obtenido de Keyence: https://www.keyence.com.mx/ss/products/sensor/sensorbasics/ultrasonic/info/

Maggiolo, D. (s.f.). *Propagación del sonido*. Obtenido de Eumus: https://www.eumus.edu.uy/docentes/maggiolo/acuapu/prp.html

Micro:Bit Educational Foundation. (s.f.). *Micro:Bit*. Obtenido de Start your micro:bit adventure!: https://microbit.org/guide/

MicroPython. (2018). *MicroPython*. Obtenido de Proper Python with hardware-specific modules : https://micropython.org/

MIT. (2012). *MIT App Inventor*. Obtenido de About Us: http://appinventor.mit.edu/about-us

NayLamp Mechatronics. (2016). *Tutorial MPU6050, Acelerómetro y Giroscopio*. Obtenido de NayLamp Mechatronics: https://naylampmechatronics.com/blog/45\_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html

NOVA Labs. (2019). *NOVA Labs*. Obtenido de ABOUT: https://www.pbs.org/wgbh/nova/labs/about/

Pardo, D. (11 de Abril de 2019). *Ehorus*. Obtenido de ¿Ser o no ser? ¡No! ¿Qué es websocket? Esa es la cuestión: https://ehorus.com/es/que-es-websocket/

ROTOTAON. (1 de Marzo de 2008). *ROTOTAON*. Obtenido de Raspberry Pi ESP32 MicroPython Touch & Sound Tutorial: https://www.rototron.info/raspberry-pi-esp32-micropython-touch-sound-tutorial/

Sáez-López, J., & Cózar-Gutiérrez, R. (2017). Programación visual por bloques en Educación Primaria: Aprendiendo y creando contenidos en Ciencias Sociales. *Revista Complutense de Educacion, 28*(2), 409-426.

Sanders, M. (Diciembre/Enero de 2009). *The Technology Teacher.* Obtenido de STEM, STEM Education, STEMmania: https://vtechworks.lib.vt.edu/bitstream/handle/10919/51616/STEMmania.pdf?sequence=1&isAllowed=y

Systems, E. (2016). *The Internet of Things with ESP32*. Obtenido de Features & Specifications: http://esp32.net/

Web Archive Org. (27 de Julio de 2006). *Almost a shape/technology/Piano frequencies/Piano frequency table.pl*. Obtenido de Web Archive Org: https://web.archive.org/web/20070305040009/http://wiki.highinbcgallery.com/index.php/Almost\_a\_shape/technology/Piano\_frequencies/Piano\_frequency\_table.pl

Zapata-Ros, M. (2015). *Pensamiento computacional: Una nueva alfabetización digital Computational Thinking: A New Digital Literacy.*