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

CARLOS STEVEN ORTIZ COPETE

UNIVERSIDAD ECCI
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA MECATRÓNICA
BOGOTÁ D.C.
AÑO 2020

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

UNIVERSIDAD ECCI
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA MECATRÓNICA
BOGOTÁ D.C.
AÑO 2020

CONTENIDO

Pág.

Tabla de ilustraciones

1 Resumen

El presente trabajo de grado consiste en el diseño de una interfaz de programación visual web con el código abierto de Blockly de Google, con el fin de que las personas se puedan familiarizar con el mundo de la programación, ayudándoles a dar el primer paso sin que se tengan que enfrentar al aprendizaje de algún lenguaje de programación en primera instancia. El propósito está focalizado a la música algorítmica bajo el concepto de crear tu propio instrumento musical, con la funcionalidad IoT de poder conectar tu tarjeta de desarrollo a través de internet y programarla remotamente. El proyecto se divide en nueve capítulos en donde se encontrará la definiendo del problema a tratar, el planteamiento de los objetivos a alcanzar, la justificación, la investigación de trabajos anteriores y definición de conceptos, las actividades a realizar, el tiempo de desarrollo, la obtención de resultados y el análisis de estos para así llegar a una serie de conclusiones.

2 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 realizar pruebas para ello, que se adapte a los proyectos MADI y K3OS.

3 Objetivos

3.1 Objetivo general

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

3.2 Objetivos específicos

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

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

3.2.3 Validar la interfaz visual de programación con la tarjeta de procesamiento.

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

4 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, blockly 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 con 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.

5 Marco conceptual

5.1 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,

2015) “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.

5.2 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.

5.3 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;

5.3.1 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 interesante, 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)

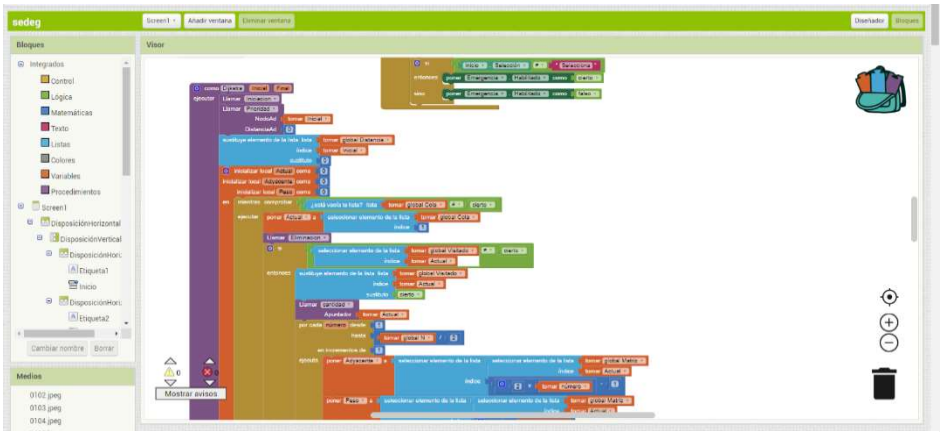


Ilustración 1 Interfaz App Inventor (Tomada de: App Inventor interfaz de programación)

5.3.2 Micro: bit

BBC micro: bit es un microcomputador 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.)

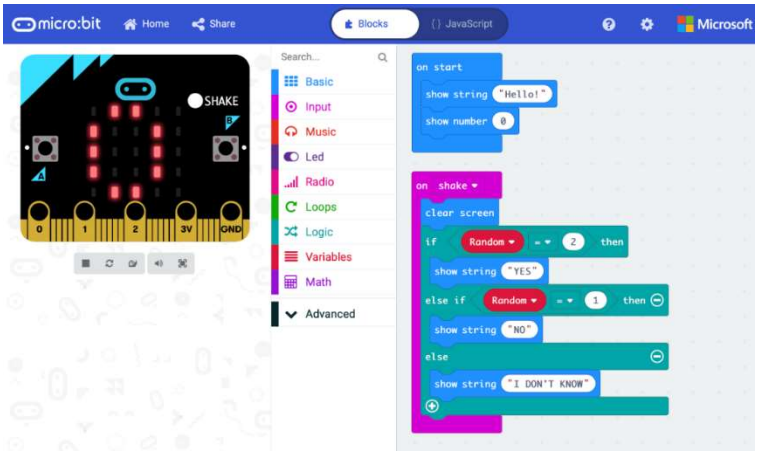


Ilustración 2 micro:bit (Tomada de: https://microbit.org/code/)

5.3.3 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 subrepresentadas. 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)

5.3.4 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)

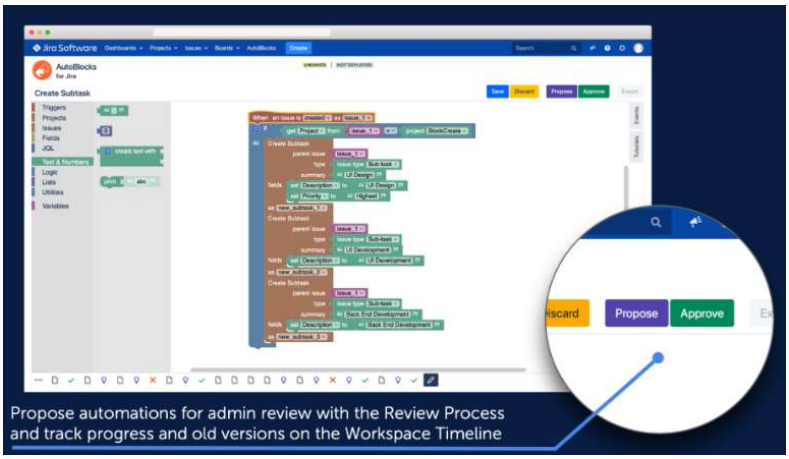


Ilustración 4 Interfaz de AutoBlock (Tomada de '/marketplace.atlassian.com/apps/1219915/autoblocks-for-jira?hosting=server&tab=overview)

5.3.5 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)

5.4 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 Bluetooth 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.

5.5 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)

5.6 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.

5.7 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).

6 Metodología

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

6.1.1 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.

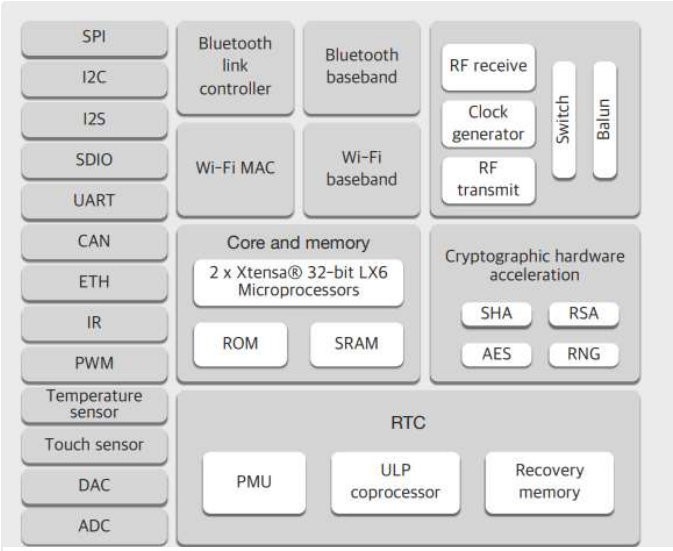


Ilustración 5 Diagrama de bloques de funciones (Tomada de: ESP32 página 12)

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.

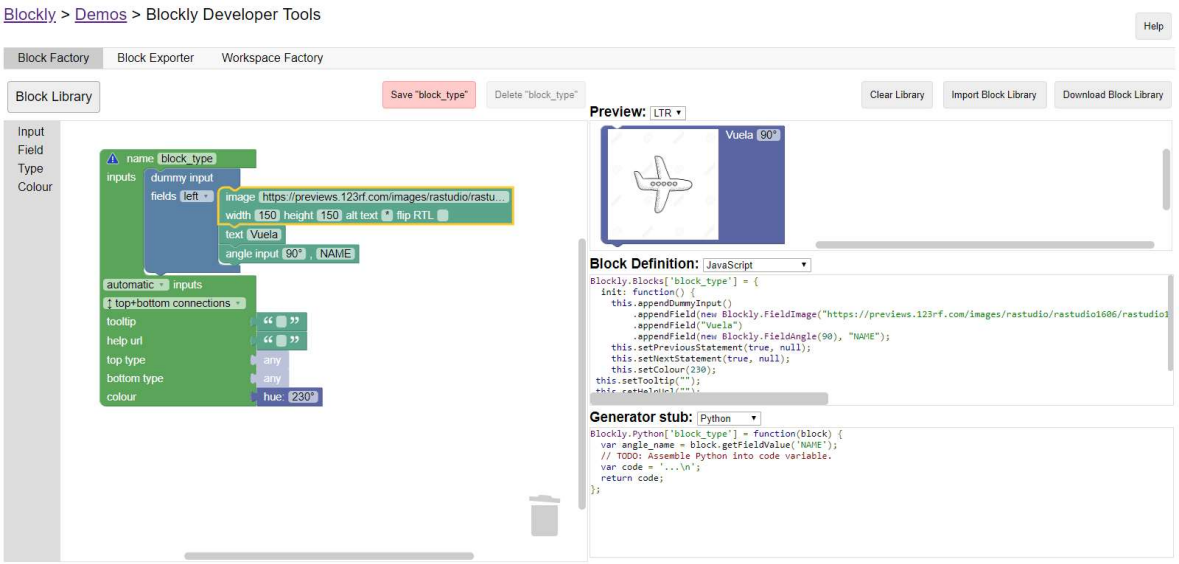


Ilustración 6 Herramienta para elaborar bloques (Tomado de: Blockly Developer Tools)

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.

6.1.2 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.

6.1.3 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.

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

6.2.1 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.

6.2.2 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.

6.2.3 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.

6.3 Objetivo 3 - Validar la interfaz visual de programación con la tarjeta de procesamiento

6.3.1 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.

6.3.2 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.

6.3.3 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.

6.3.4 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.

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

6.4.1 Actividad 4.1 – Configuración del web server

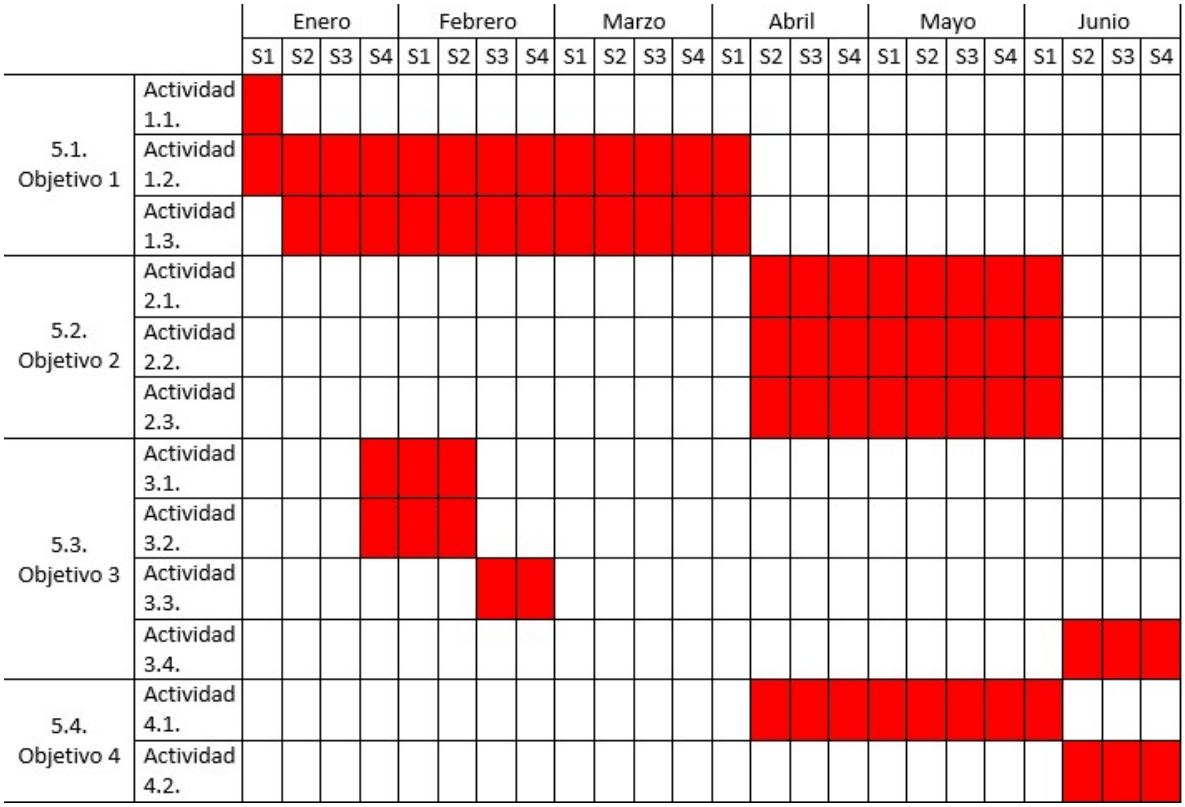
La tarjeta ESP32 cuenta con la funcionalidad WebSocket, el cual es un canal de comunicación full-duplex (Pardo, 2019). Esto puede servir como medio para enviar el código generado por Blockly sin tener que conectar la tarjeta físicamente al dispositivo con la interfaz.

6.4.2 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.

7 Cronograma

Las tareas mostradas en Tabla 1, son aquellas mostradas en la Metodología.



8 Resultados

En este capítulo se muestran los resultados obtenidos de las actividades planteadas en la metodología para lograr el diseño de la interfaz de programación visual en plataformas con soporte Micropython, se destacarán aquellos resultados que estén directamente relación con el objetivo general, esto porque se obtuvieron resultados de pruebas realizadas a la interfaz de programación respecto a mejoras de rendimiento y resolución de errores.

8.1 Creación de bloques funcionales

Se definieron unos bloques funcionales que sirvan para crear instrumentos musicales a partir de bloques funcionales escritos en Python estos están divididos en categorías y bloques, ver Ilustración 7, 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, algunos bloques importadores contienen un controlador el cual presenta una interfaz codificada que se encarga de manejar la comunicación con el dispositivo. El funcional que es el que activara las funciones del objeto anterior y por último un bloque extra para alguna otra funcionalidad o configuración.

En la Ilustración 8 se puede evidenciar la construcción del bloque importador del bloque ***notas***, además de 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. Estos códigos se agregan al programa base. La construcción de estos bloques se realizó siguiendo la documentación de Blockly para cada tipo de bloques (Google, 2020) y la herramienta Blockly Developer Tools.

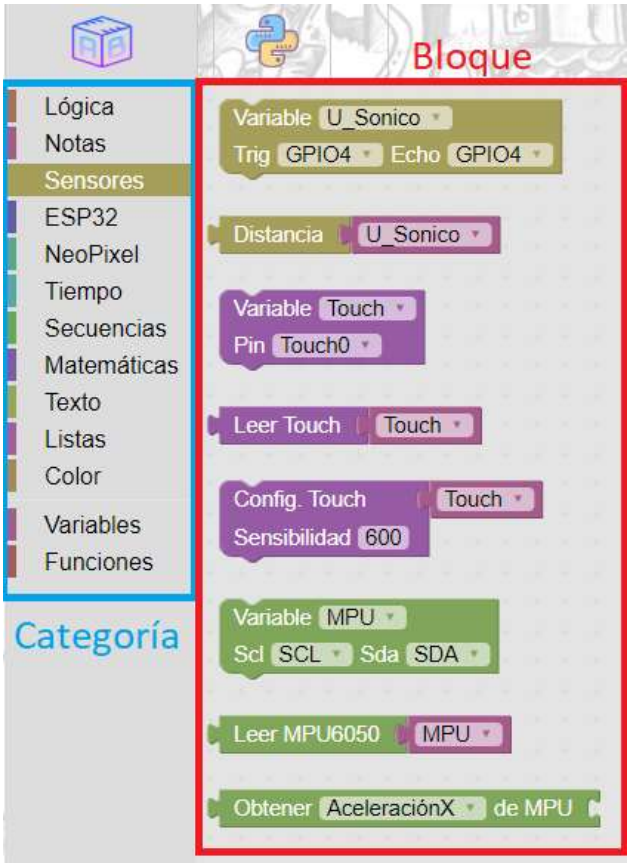


Ilustración 7 Distribución de bloques (Categoría y bloque)

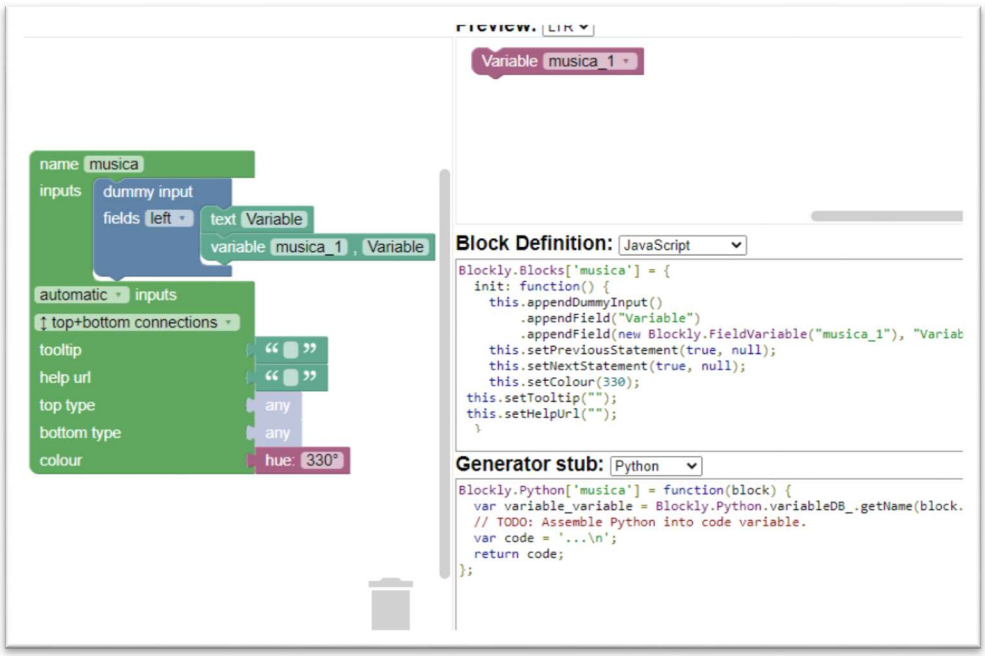


Ilustración 8 Creación del bloque importador musical, con su vista previa, su definición y su generador

8.1.1 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 número ocho 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 9, el importador se encarga de la configuración inicial para la utilización del PWM, para lo cual, el usuario puede modificar el nombre de la variable la cual trae como predeterminado “musica_1”, bajo este nombre creará el objeto con este nombre asignándole al Pin 25 un PWM y ver Ilustración 10 del funcional (flauta), este bloque se encarga de asignarle al PWM la frecuencia y el ciclo útil de este, además de darle un tiempo de espera, para lo cual el usuario puede modificar el tipo de nota (dependiendo de la nota cambia la frecuencia), el tiempo que se reproducirá la nota, y podrá agregarle el bloque con la variable que tiene consigo el objeto creado con el bloque importador, que es donde se le asignará todas las configuraciones establecidas en este bloque funcional.

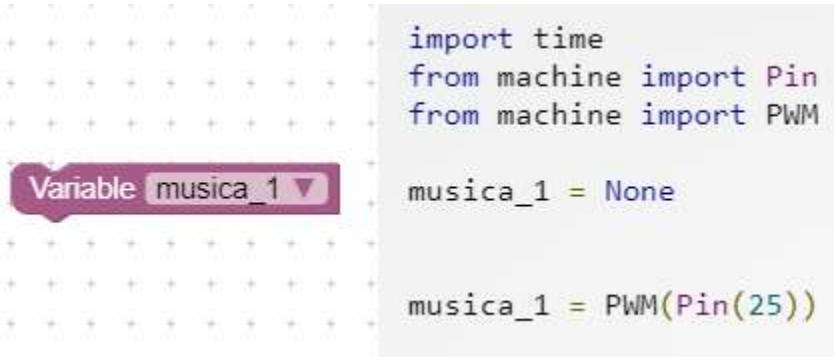


Ilustración 9 Importador musical

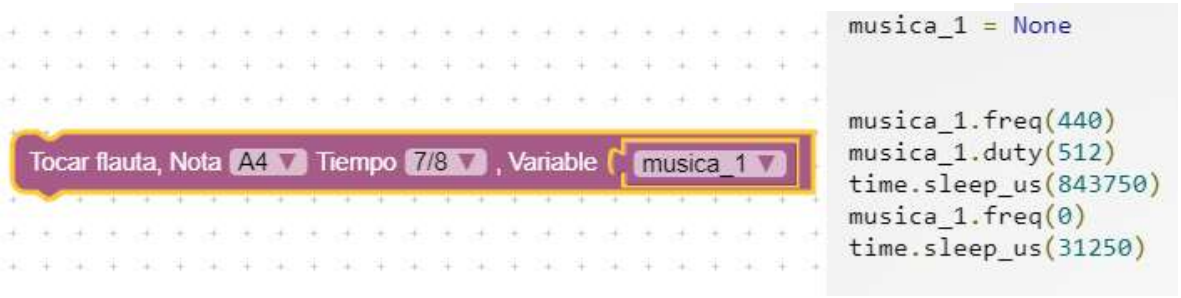


Ilustración 10 Funcional musical (Flauta)

8.1.2 Sensores

- Ultrasonido HC-SR04:

Ver Ilustración 11 del importador, este bloque se encarga de importar los controladores para el sensor HC-SR04 y de la configuración inicial, para lo cual el usuario puede cambiar el nombre de la variable que predeterminado es “U_Sonico”, bajo este nombre se creará el objeto con los pines que el usuario asigne en “Trig” y “Echo” y ver Ilustración 12 del funcional del sensor de ultrasonido, ese bloque llama a la función encargada de obtener los datos del sensor, procesarlas y asignarlas a una variable, tiene como entrada la variable con el objeto creado en el importador del ultrasonido y como salida un valor flotante de la distancia en centímetros.

El controlador para el manejo del sensor de ultrasonido se tomó de (Hirwing & Roberto, 2017).

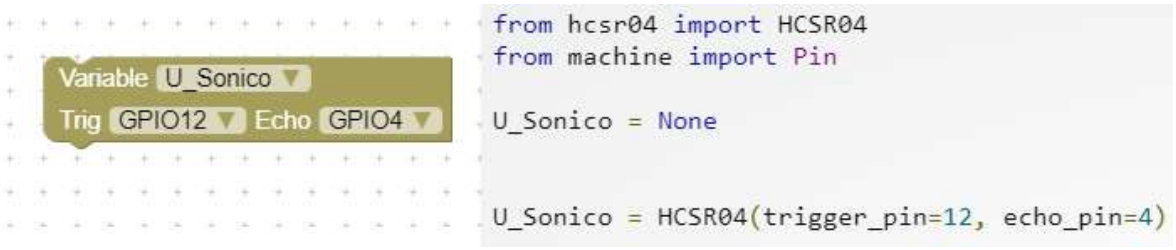


Ilustración 11 Importador sensor HC-SR04

Ilustración 12 Funcional sensor HC-SR04

- TouchPad:

Ver Ilustración 13 del importador, este bloque se encarga de la configuración inicial, el usuario puede cambiar el nombre de la variable que predeterminada es “Touch”, en esta variable se crea el objeto con el pin configurado, el usuario puede seleccionar el “Pin” que tendrá la funcionalidad touch, ver Ilustración 14 del funcional, este funcional obtiene los valores del touchpad, tiene como entrada la variable con el objeto creado con el bloque importador, y como salida un valor entero de 0 a 600, este rango puede cambiarse con el bloque configurador y la Ilustración 15 configurador Touchpad, este bloque permite al usuario cambiar el rango de la sensibilidad del touchpad.

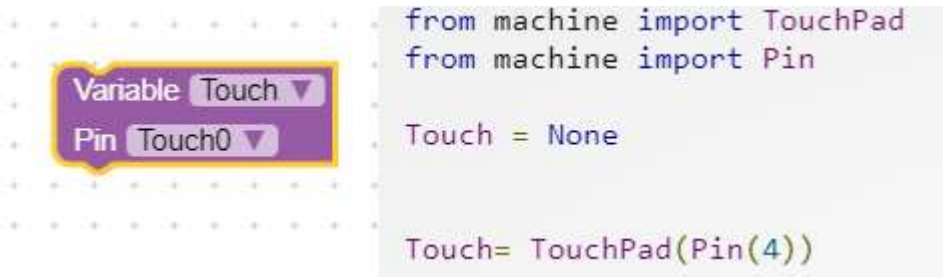


Ilustración 13 Importador Touch

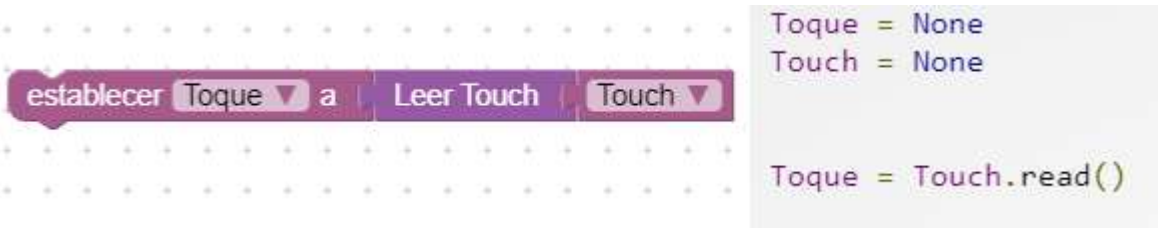


Ilustración 14 Funcional Touch

Ilustración 15 Configuración Touch

- IMU MPU6050:

Ver Ilustración 16 del importador, este bloque se encarga de la configuración inicial del sensor, el usuario puede cambiar el nombre de la variable que predeterminada es “MPU” en esta variable se creará el objeto que tendrá la configuración inicial, el usuario puede modificar los pines SCL y SDA pero solo se tiene un pin para cada uno, ver Ilustración 17 del funcional, este bloque permite obtener los valores del sensor, tiene como entrada la variable con el objeto configurado con el bloque importador y como salida un arreglo con los valores del acelerómetro, giroscopio y temperatura. y la Ilustración 18 del extra del MPU6050, este bloque extra permite separar los valores que se obtuvieron con el bloque funcional, por tanto, tiene como entrada el arreglo con los datos del sensor y como salida un valor numérico, el usuario puede seleccionar lo que desea obtener, en este caso se obtiene la aceleración del eje x.

El controlador para el manejo del MPU6050 se tomó de (Ježek & Kuethe, 2017)

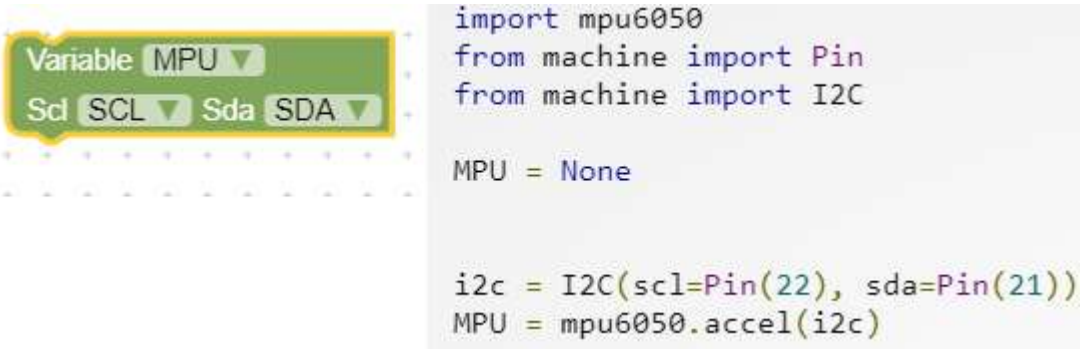


Ilustración 16 Importador MPU6050

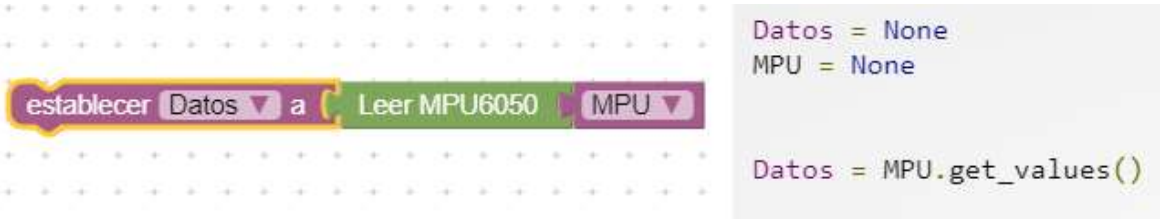


Ilustración 17 Funcional MPU6050

Ilustración 18 Extra MPU6050

8.1.3 ESP32

Esta categoría presenta bloques de funcionalidades para la tarjeta ESP32, esta categoría está creada para que se puedan agregar nuevas funcionalidades aparte de las ya establecidas, ya que tiene bloques de programación de pines, conversor análogo digital y PWM.

- GPIO (pines digitales)

Ver Ilustración 19 e Ilustración 20 del importador, estos bloques permiten la configuración inicial del pin, cada uno permite al usuario cambiarle el nombre de la variable que predeterminada es Entrada y Salida respetivamente, además permite seleccionar el pin. El bloque de entrada permite configurar el pin con una resistencia ya sea pull up o pull down,

ver Ilustración 21 e Ilustración 22 del funcional de los pines, estos bloques funcionales son diferentes respecto si es entrada o salida, sí es salida permite configurar el valor como encender o apagar (1 y 0 lógico), tiene como entrada la variable creada con el importador de salida, y el bloque funcional como entrada, permite leer el valor que tiene el pin, tiene como entrada la variable con el objeto creada en el importador entrada y como salida un valor booleano.

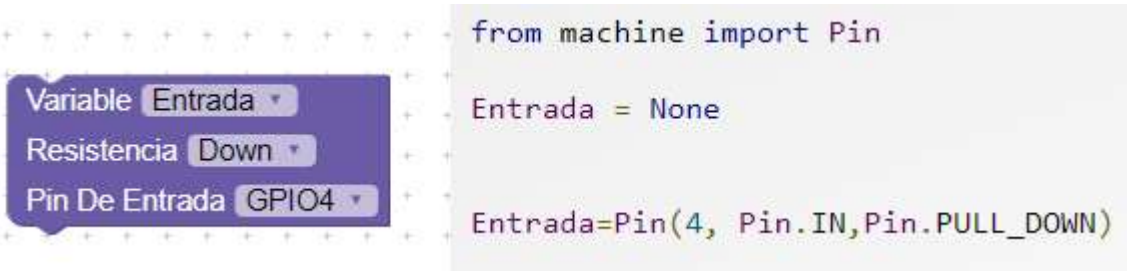


Ilustración 19 Importador GPIO como entrada

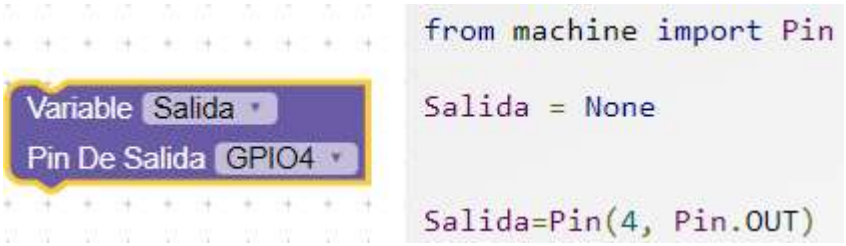


Ilustración 20 Importador GPIO como salida

Ilustración 21 Funcional GPIO como salida

Ilustración 22 Funcional GPIO como entrada

- Conversor análogo digital.

Ver Ilustración 23 del importador, este bloque permite la configuración inicial del conversor, permite al usuario modificar el nombre de la variable que predefinida es “Conversor” y seleccionar el pin que desea usar como conversor ver Ilustración 24 del funcional del conversor análogo digital, este funcional obtiene el valor numérico obtenido por el conversor teniendo como entrada la variable configurada en el bloque importador y como salida en valor del conversor.

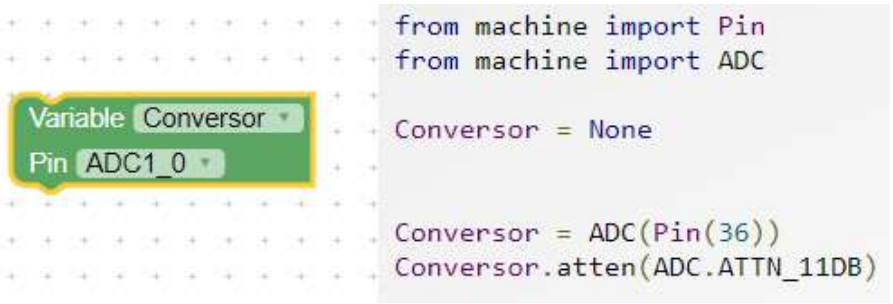


Ilustración 23 Importador Conversor

Ilustración 24 Funcional Conversor

- PWM:

Este PWM es igual al usado con los bloques musical, con la diferencia del pin y que este permite modificar la frecuencia y el ciclo útil numéricamente.

Ver Ilustración 25 del importador, este bloque se encarga de crear la configuración inicial, permitiendo al usuario cambiar el nombre de la variable que es “pwm1” y el pin, ver Ilustración 26 del configurador, este bloque permite modificar la configuración del pwm

permitiendo al usuario modificar la frecuencia y el ciclo útil, como entrada recibe la variable configurada en el bloque importador, y ver Ilustración 27 del bloque extra del PWM, este bloque permite detener el pwm tiene como entrada la variable que se configuro en el bloque importador.

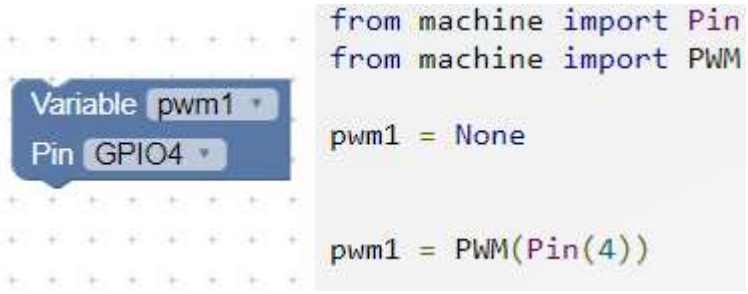


Ilustración 25 Importador PWM

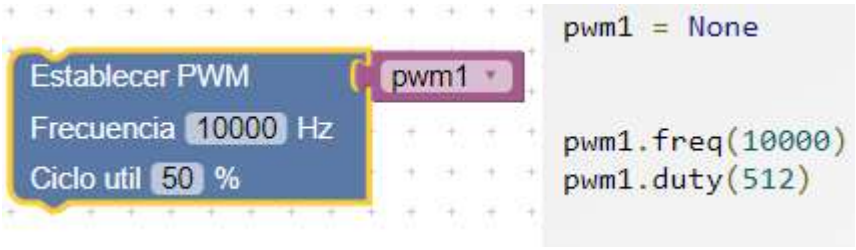


Ilustración 26 Configurador PWM

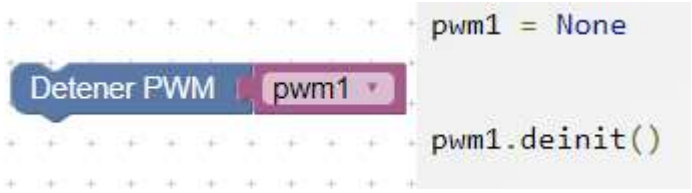


Ilustración 27 Extra PWM

8.1.4 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. Cada led tiene un integrado que almacena 3 bytes que corresponde a los 3 colores RGB, cuando un led recibe un flujo de bytes, almacena los últimos bytes recibidos y transmite los

que contenía al siguiente Led, para lo último con una señal de reset cada Led muestra el ultimo valor almacenado (Llamas, 2016).

Ver Ilustración 28 del importador, este bloque realiza la configuración inicial donde el usuario puede modificar el nombre de la variable que predeterminado es “neo_pixel” de igual forma puede seleccionar el pin para el control del dispositivo y la cantidad de pixeles que se tiene conectado, en este caso se usó un anillo con 24 pixeles, ver Ilustración 29 del funcional, este bloque configura el color a un pixel, por tanto, el usuario puede indicarle el color que desea configurar y asignarle ese color a un pixel, la entrada es la variable creada con el bloque importador, y la Ilustración 30 del extra del NeoPixel, este bloque es el encargado de realizar el reset, sin este bloque, las configuraciones realizadas en el bloque funcional no se verán reflejadas en el dispositivo NeoPixel.

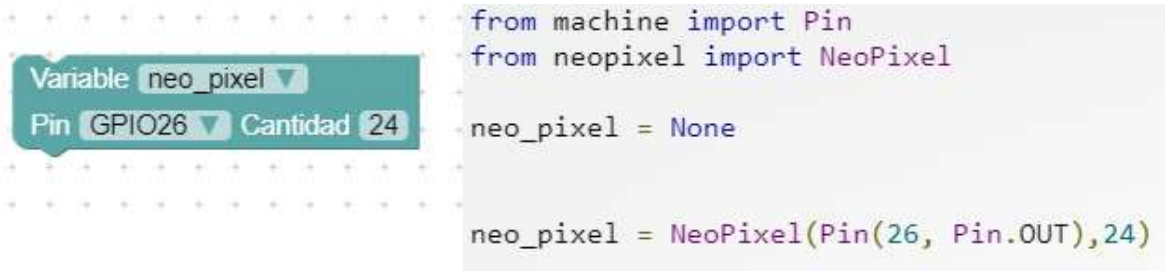


Ilustración 28 Importador NeoPixel

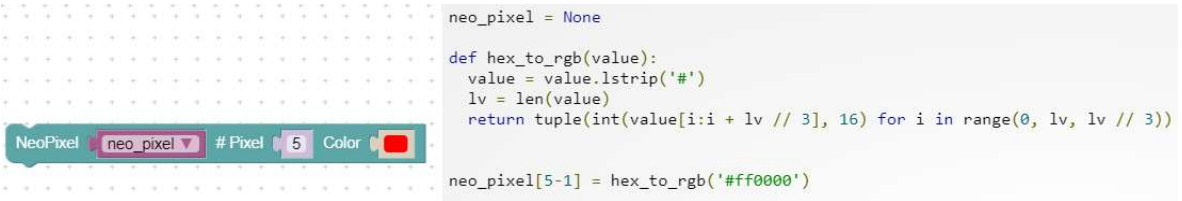


Ilustración 29 Funcional NeoPixel

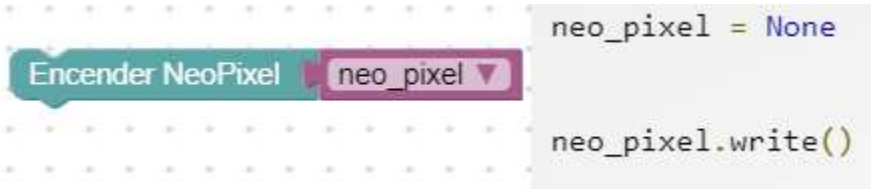


Ilustración 30 Extra NeoPixel

8.1.5 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.

Ver Ilustración 31 del funcional para un tiempo de retraso.

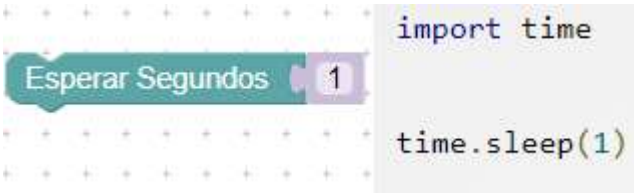


Ilustración 31 Funcional retraso

- Fecha:

Ver Ilustración 32 del importador, este bloque se encarga de realizar la configuración inicial donde el usuario puede modificar el nombre de la variable que por defecto es “Fecha”, ver Ilustración 33 del configurador, este bloque realiza la sincronización del reloj con un servidor en línea, tiene una entrada que es la variable creada con el bloque importador, la Ilustración 34 del funcional permite obtener la fecha, tiene una entrada que es la variable creada con el bloque importador y una salida que es un arreglo con la fecha (día, mes, año, hora, minuto, segundo) y la Ilustración 35 del extra de la fecha, este bloque permite separar los valores obtenidos del bloque funcional, el usuario puede seleccionar el dato que desea obtener y este bloque tiene una entrada que es la variable creada en el bloque importador y una salida que es el valor seleccionado.

El controlador para la actualización del RTC vía internet se tomó de (ESPloradores, 2019)

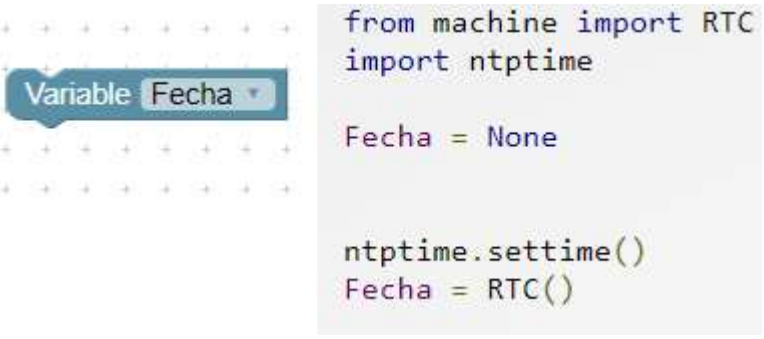


Ilustración 32 Importador fecha

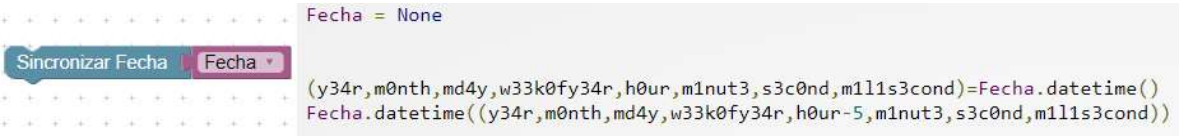


Ilustración 33 Configurador fecha

Ilustración 34 Funcional fecha

Ilustración 35 Extra fecha

- Temporizador:

Ver Ilustración 36 del configurador, este bloque permite configurar el temporizador, permite al usuario cambiar el nombre de la variable que por defecto es “Temp”, seleccionar el temporizador de 0 a 3, el modo ya sea periódico o único y el periodo al cual se activará el temporizador, tiene una entrada que es el nombre de la función que realizará cuando el temporizador finalice en este caso la función es “hacer_algo” y ver Ilustración 37 del bloque

extra para una interrupción por temporizador, este bloque detendrá el temporizador en caso de que el temporizador sea periódico.

Ilustración 36 Configurador de interrupción por temporizador

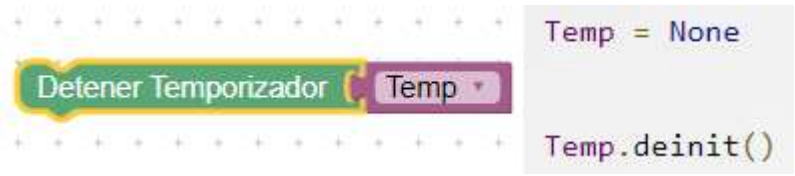


Ilustración 37 Extra de interrupción por temporizador

8.2 Diseño de la PCB

Para el diseño se tuvo en cuenta que se usó el HC-SR04, MPU6050, BUZZER, TouchPad y el NeoPixel, y se pensó para que fuera un instrumento musical programable, por lo tanto, se utilizó el programa gratuito llamado KiCad para realizar el diseño. Ver Ilustración 38 para observar el diagrama de conexión. Ver Ilustración 39 del diseño de la PCB y ver Ilustración 40 del 3D y sus componentes.

Los criterios de diseño están direccionados a poder conectar y desconectar los dispositivos en cualquier momento, de esta forma se usan regletas de pines, por tanto, cada zona de contacto es through hole, caminos con un ancho de 1mm, esto porque es recomendado para una corriente máxima de 3 amperios (Bellido Díaz , 2015), y por experiencia propia con la fabricación. Caminos a 45° sin ángulos de 90°. Doble capa, jaula de Faraday en ambas capas conectadas a GND, para evitar el ruido electromagnético y otras interferencias. De esta forma se alcanzó el objetivo de construir un prototipo inicial de hardware que permita probar la integración de la programación visual.

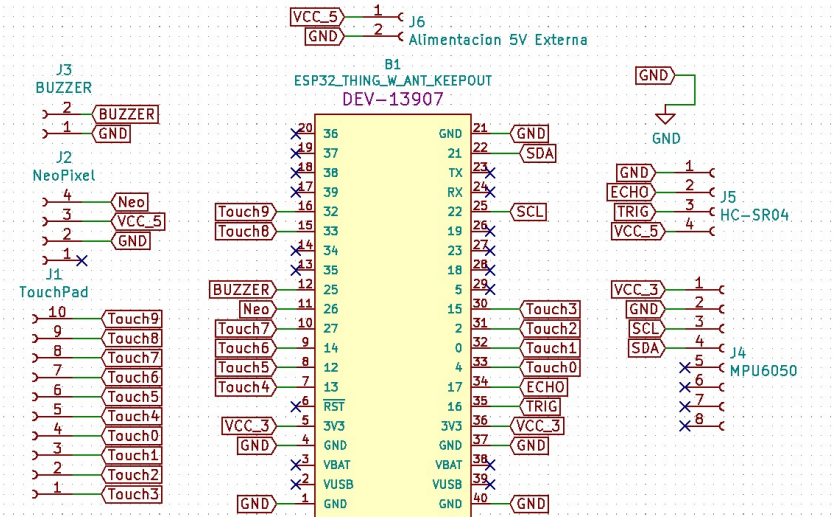


Ilustración 38 Diagrama de conexión

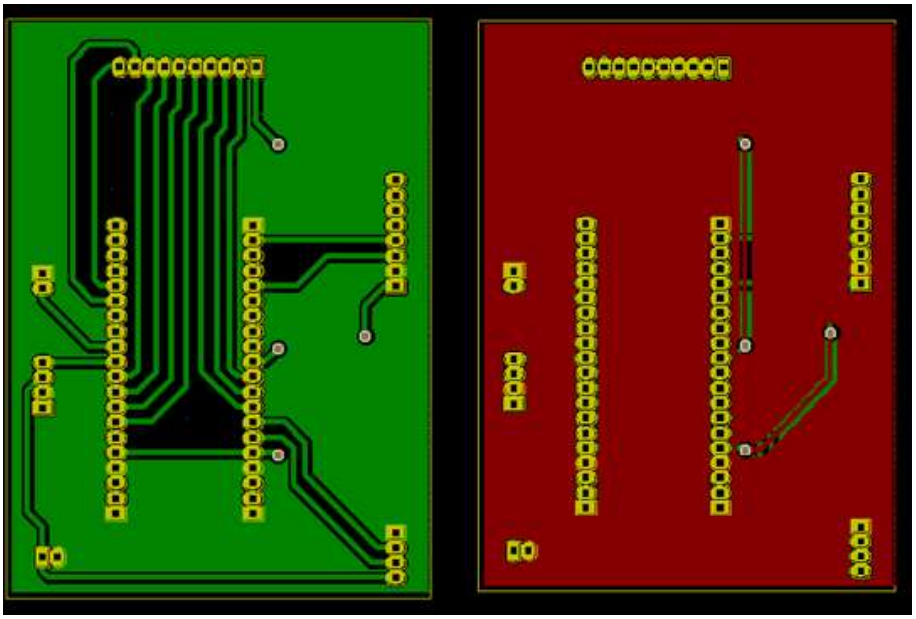


Ilustración 39 Diseño de la PCB

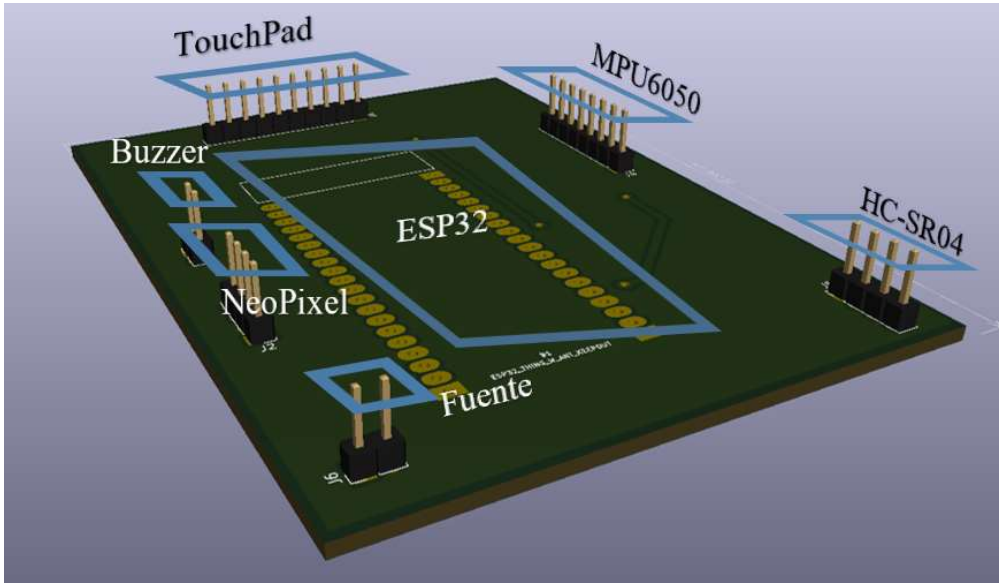


Ilustración 40 3D de la PCB

8.3 Validación de la interfaz visual y la tarjeta

Esta validación se realiza con el fin de poder programar la tarjeta de forma remota, de este modo los bloques ya creados se prueban con la tarjeta de desarrollo y los dispositivos conectados.

8.3.1 Creación del WebSocket

Según (Mozilla and individual contributors, 2019), es una tecnología que hace posible abrir una sesión de comunicación entre el navegador del usuario y un servidor, de esta forma se pueden enviar mensajes a un servidor y recibir respuestas. Esta tecnología es soportada por la tarjeta de desarrollo, además de permitir enviar de archivos de la interfaz a la tarjeta de forma segura, permitiendo de esta forma programar remotamente la tarjeta.

Para habilitar WebSocket en la tarjeta ESP32 se siguió las instrucciones encontradas en la documentación de MicroPython en la tarjeta ESP32 (MicroPython.org, 2019).

JavaScript permite la creación de un WebSocket, este se configurará para que se conecte al de la tarjeta y así se comuniquen. Como el WebSocket de la tarjeta está conectado al REPL (Bucle de lectura-escritura-impresión), desde la página web se puede programar la

tarjeta, pero estos programas que se realizan desde aquí no se almacenan en la memoria de la tarjeta, para ello se usa el gestor de archivos de la tarjeta enviando el programa que se quiere tener en la tarjeta como un archivo Python.

Para lo cual, el código que se genera al usar los bloques es una cadena de texto y como Python no maneja corchetes ni llaves, su estructura debe ser ordenada, para lo cual esta debe convertirse a UTF-8, la web nos genera un código con algunos "errores de organización" y al convertirlo a UTF-8 algunos espacios son borrados a la mitad, por ejemplo, sí una línea tenía 4 espacios (2 tabulaciones del teclado) terminaba con 2 espacios (1 tabulación del teclado) esto se arregla buscando estos espacios y agregando los que le faltan, esto es importante ya que sin ello la tarjeta lo toma como un error y no ejecuta el código.

Se empieza enviando la trama de la Ilustración 41. Esta trama se obtuvo analizando el programa de muestra para el WebSocket con la tarjeta de desarrollo (Micropython, 2019)

Inicio de trama		Put									Tamaño del archivo				Longitud del nombre		Nombre del archivo		
W	A	1	0	0	0	0	0	0	0	0	XX	XX	XX	XX	XX	XX	XX	...	XX

Ilustración 41 Trama de envío de archivos por WebSocket

Cuando se reciba respuesta, se empieza a enviar el archivo por bytes, y cuando se termine de enviar y no se ha recibido respuesta de algún error, el archivo quedo bien enviado. Sí por el WebSocket se envía el comando "os.listdir()\r" se recibirá respuesta de la tarjeta con un arreglo con los nombres de los archivos que tiene la tarjeta, de esa forma se puede verificar sí el archivo llegó.

8.3.2 Validación con el Software-Hardware

- Buzzer: para verificar el funcionamiento del Buzzer, se crean los bloques de la Ilustración 42, y se envía el código a la tarjeta y esté sonó con la nota establecida.

Ilustración 42 Prueba Buzzer

- Ultrasonido: se crearon los bloques de la Ilustración 43 el cual permite programar el sensor y obtener las medidas del ultrasonido en centímetros.

Ilustración 43 Prueba HC-SR04

- Touch: para programar la funcionalidad touch de la esp32, se crearon los bloques de la Ilustración 44, y se imprimió los valores obtenidos.

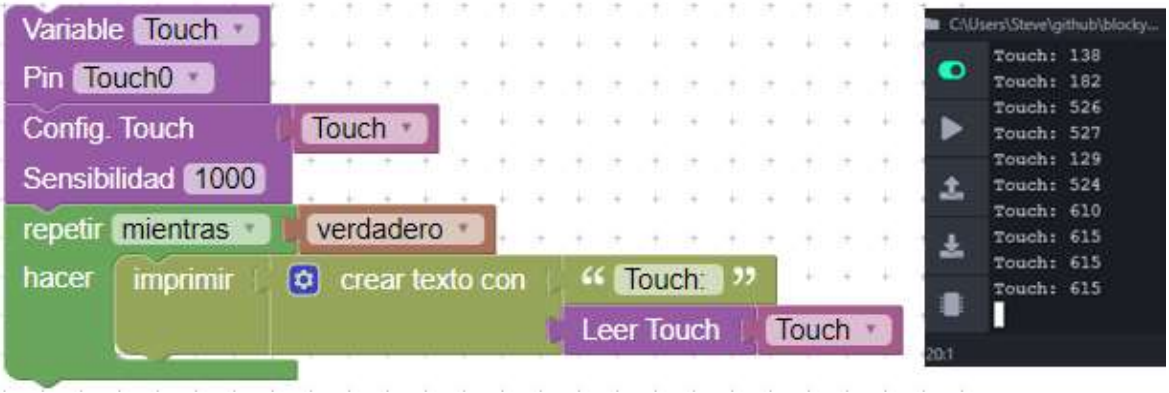


Ilustración 44 Prueba Touch

- MPU6050: usando los bloques de la Ilustración 45, se obtienen los valores que arroja el módulo, sí se desean tener los valores por separado, se podría usar el bloque de la Ilustración 18.

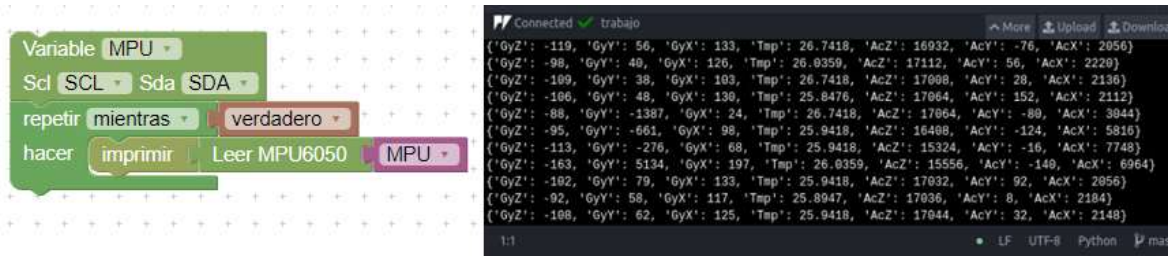


Ilustración 45 Prueba MPU6050

- NeoPixel: para probar las funcionalidades de NeoPixel, se programaron los bloques de la Ilustración 46 dando como resultado lo visto en la misma ilustración.

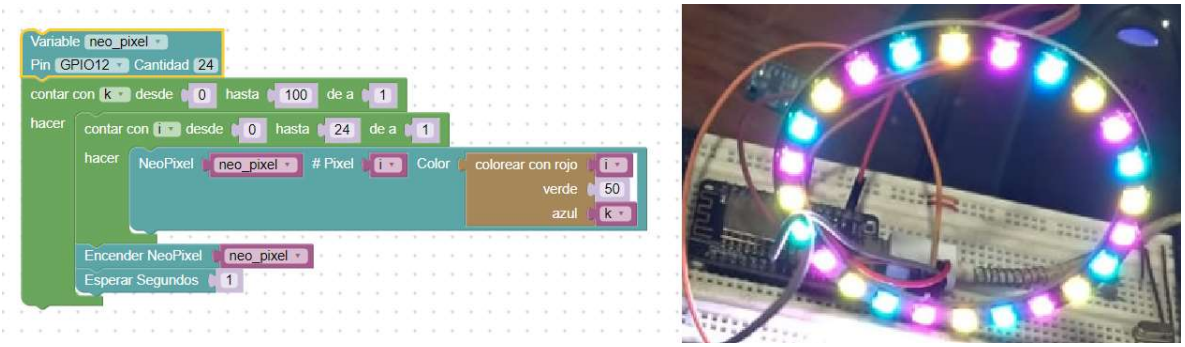


Ilustración 46 Prueba NeoPixel

8.4 Diseño centrado en el usuario

El programa base de Blockly, cuenta con bloques de programación básico del tipo Lógica, Secuencias, Matemáticas, Texto, Listas, Color, Variables y Funciones como los vistos en la Ilustración 47, además generadores para otros lenguajes como lo es, Lua, XML, PHP entre otros.

Ilustración 47 Interfaz de Blockly base

Para mejorar la experiencia del usuario se creó el concepto de la Ilustración 48, con una sección musical, botones, área de trabajo y visualización de tarjeta.

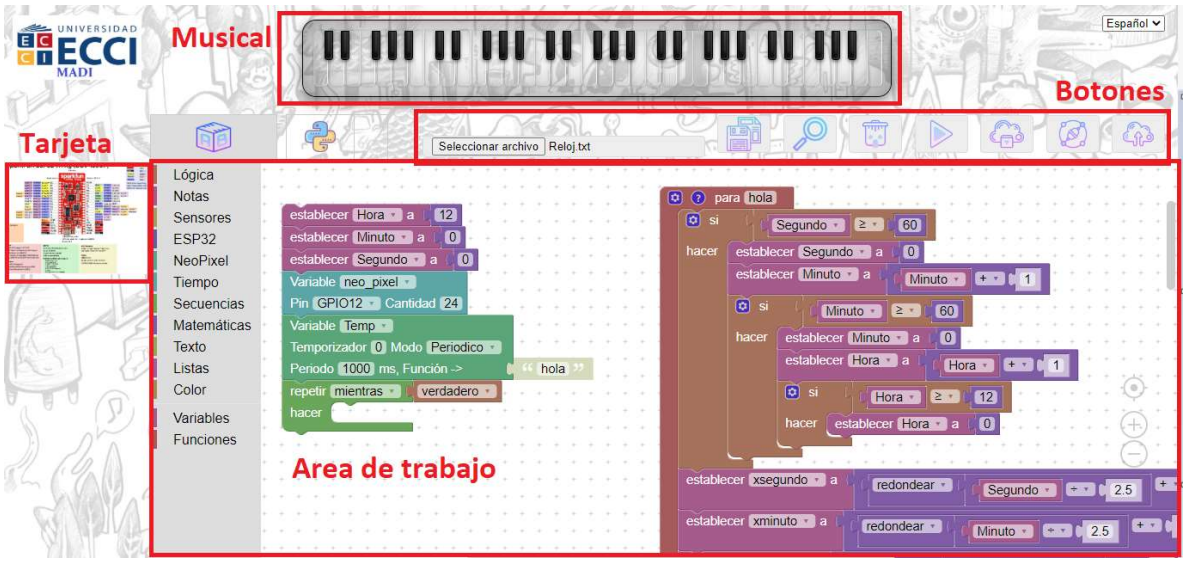


Ilustración 48 Concepto para la interfaz de Blockly MADI

8.4.1 Área musical

Esta área se presenta en forma de piano funcional, este piano contiene todas las notas que se usaron en los bloques musicales, con el fin de que el usuario pudiera buscar alguna nota, oírla y saber que nomenclatura musical tenía.

8.4.2 Área de botones

Cada botón trae consigo la siguiente funcionalidad;

- Carga: este botón permite cargar un programa guardado con anterioridad, este archivo es de tipo texto. Este botón es el número 1 en la Ilustración 49.
- Guardar: este botón permite guardar el trabajo que se haya realizado, este botón genera un archivo tipo texto. Número 2 en la Ilustración 49.
- Buscar: este botón funciona con la tarjeta de desarrollo conectar mediante WebSocket, sirve para ver los archivos que tiene la tarjeta de desarrollo. Número 3 en la Ilustración 49.
- Descartar: borra todos los bloques que se encuentren en el área de trabajo. Número 4 en la Ilustración 49.
- Ejecutar: permite ejecutar lo que se encuentre en el área de trabajo, esta ejecución es realizada por JavaScript, y no todos los bloques realizan acciones en la página. Este botón además le avisa al usuario si algún pin de la tarjeta está siendo usado más de una vez y le generará una advertencia avisándole el nombre y el número del pin para que sea revisado en el área de tarjeta y los bloques que tenga programados. Ver Ilustración 51. Número 5 en la Ilustración 49.
- Descargar: Funciona para descargar el código generado en Python de lo que se encuentre en el área de trabajo. Número 6 en la Ilustración 49.
- Conectar: Permite establecer la conexión por WebSocket con la tarjeta de desarrollo. Número 7 en la Ilustración 49.
- Enviar: Funciona con la tarjeta de desarrollo conectada por WebSocket, y sirve para enviar el archivo de lo que se haya programado en el área de trabajo. El

archivo enviado es de tipo Python con el nombre “main.py”. Número 8 en la Ilustración 49.

Ilustración 49 Distribución de botones de la interfaz Blockly MADI

8.4.3 Área de trabajo

En esta área es donde se pueden manipular los bloques, organizarlos y trabajar con ellos para programar. Sí se le da clic en el símbolo de Python ver Ilustración 50, la página presentará en el área de trabajo el código en Python de los bloques que se encontraban en el área de trabajo. Sí se le da clic en el bloque ver Ilustración 50, el área de trabajo volverá a mostrar el espacio para trabajar con bloques.

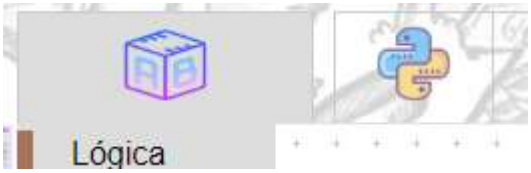


Ilustración 50 Botones del área de trabajo

8.4.4 Área de Tarjeta

Esta área tiene la funcionalidad de mostrar el Pinout de la tarjeta ESP32, con el fin de mostrar los pines que pueden ser usados, y para verificar junto al área de trabajo sí algún pin ya está en uso.

Ilustración 51 Mensaje de advertencia de pines repetidos

8.5 Prueba de integración

Para la prueba de integración, se usaron los componentes del apartado 6.2.1 Para ello, se creó un instrumento musical cada sensor activado genera el sonido de algún instrumento, ver Ilustración 52.

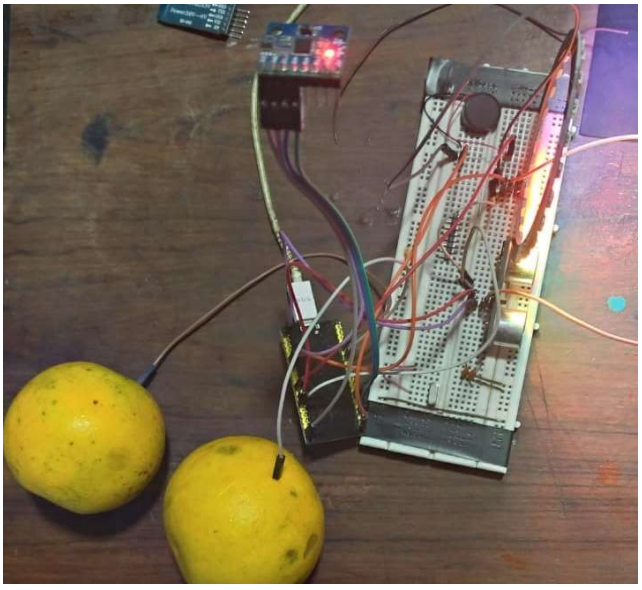


Ilustración 52 Instrumento musical inventado

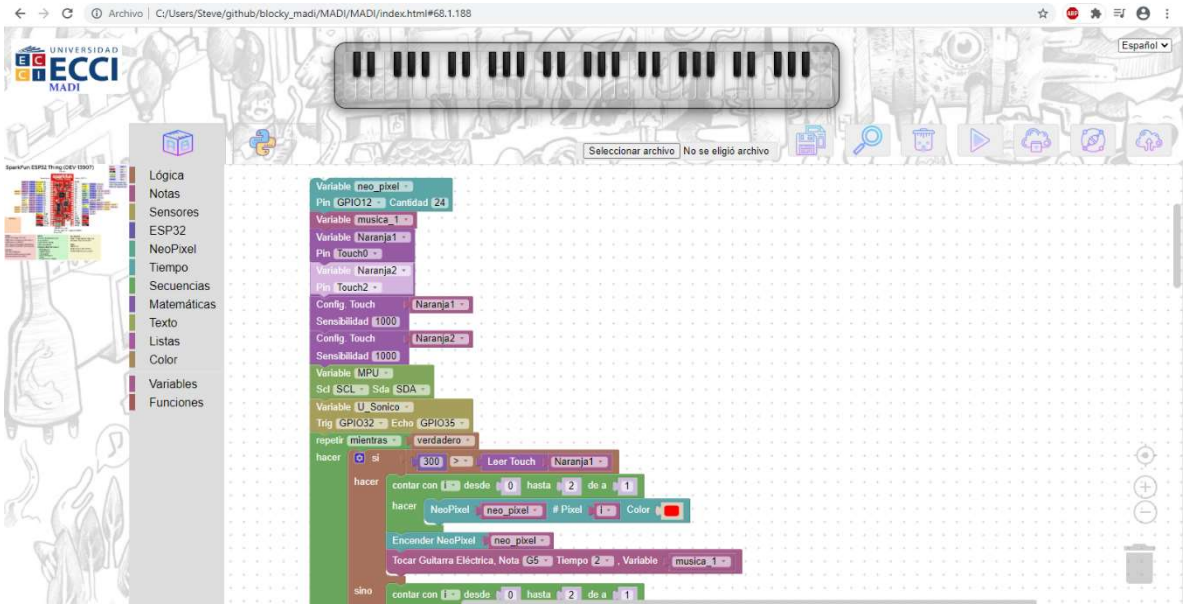


Ilustración 53 Programación de instrumento musical

El programa es realizado con bloques y enviado a través de WebSocket ver Ilustración 53. La prueba de integración está dividida en dos partes, la primera en el software y la segunda en el hardware. La parte del software contiene el código desarrollado con la interfaz, este programa en su inicio tiene los bloques importadores del NeoPixel, música, dos touch y su bloque configurador, la MPU6050, y el sensor de ultrasonido HC-SR04, siguiéndole un bucle infinito donde se revisa constantemente los sensores y cuando estos son activados generan un sonido, en el caso de la MPU contiene un sonido para la inclinación sobre el eje X y Y, el HC-SR04 permite modular la frecuencia de un instrumento dependiendo de la distancia que devuelva este sensor, los touch contienen un sonido dependiendo de si este fue tocado, y el NeoPixel dependiendo del sonido enciende una parte del anillo con un color

En la parte del hardware vista en la Ilustración 52, se realiza la conexión de todos los dispositivos en los respectivos pines asignados en el software.

Para ver el instrumento dirigirse al enlace https://youtu.be/-TZP49mcGTE

9 Análisis de resultados

Para lograr el objetivo planteado al inicio de esta tesis, se realizaron cada una de las anteriores actividades y se completaron logrando el objetivo, ya que la plataforma permite la programación a través de bloques, esto lo hace que sea visual y no verse involucrado en la labor de saber o tener conocimientos de un lenguaje de programación, además permite que el usuario pueda construir sus propios instrumentos musicales.

Entre los objetivos específicos se encuentra realizar un prototipo de hardware que permita probar la integración visual. Esta PCB tendría la conexión para los diferentes sensores y dispositivos que se conectarían con la tarjeta de desarrollo, pero debido a la pandemia del Covid19 y el lugar de residencia del autor de la tesis, no se pudo realizar esta parte, por esta razón se realizaron estas pruebas de integración en una protoboard como se evidencia en los resultados.

Entre las plataformas de programación visual encontradas en el apartado 5.3, este proyecto de tesis se diferencia de la mayoría ya que permite la programación de una tarjeta de desarrollo, solo pareciéndose a la encontrada en el apartado 5.3.2 Micro: bit, ya que esta se puede realizar casi lo mismo para la tarjeta MicroBit, pero a diferencia de la anterior, esta presenta soporte para cualquier tarjeta que soporte MicroPython, además pudiéndose conectar a la tarjeta a través de una conexión por WebSocket.

Aunque cualquier programa que se escriba en MicroPython puede ser corrido en gran variedad de tarjetas, este proyecto se realizó centrándose en la utilización de la tarjeta ESP32 y se debe tener en cuenta que las demás tarjetas no presentan la misma distribución de pines, componentes y funciones específicas de la tarjeta. Además, MicroPython es bastante parecido a Python, pero no contiene todas las funciones y librerías que Python puede tener.

Este proyecto de tesis presenta gran expectativa, ya que podría utilizarse para enseñar programación como en las escuelas y universidades, y en el caso de las universidades podría utilizarse tanto para adentrar a los aprendices en el mundo de la programación entre los primeros semestres y en el manejo de lenguajes de programación como lo es Python, como en el manejo de tarjetas de desarrollo.

Como recomendaciones para futuros proyectos que surjan a partir de este, se encuentran, poder agregar animaciones a la plataforma cuando se corra un programa realizado en la página. Poder generar sonidos utilizando I2S y lograr una mejoría en el mismo. Y expandir la plataforma agregando nuevos dispositivos y funciones.

10 Conclusiones

Este capítulo presenta las conclusiones del trabajo. Por tanto, se concluye que, el uso del buzzer permite una gran gama de tonos musicales, pero por la morfología de este, no dejará de sonar como un buzzer. Respecto a los sensores utilizados para la creación de instrumentos musicales (MPU6050, HC-SR04 y Touch) se genera una nueva perspectiva ya que un instrumento musical que genere ciertos sonidos tocando objetos no convencionales en la musical (como una fruta), de su inclinación o la distancia que se tenga de un objeto.

Trabajar con bloques para concebir código ayuda a la comprensión de este, esto como experiencia propia, ya que a través de su desarrollo y las pruebas realizadas se notó que su estructuración facilita la lectura de este y frente a la programación convencional, resulta más amigable ver formas y colores que simple texto. Además, realizar código sin tener que pensar en la sintaxis resultaría en la gran ventaja de centrarse en el problema a resolver, evidenciando un ahorro de tiempo al programar ya que solo se están arrastrando bloques que tienen consigo un código predefinido.

Implementar este tipo de programación con la tarjeta de desarrollo ESP32 abre la puerta a la curiosidad, ya que de igual forma como se utilizó la tarjeta para la creación de instrumentos musicales se pueden desarrollar múltiples dispositivos. Esto da pie a la continuación, ya que hay elementos que pueden ser mejorados.

Al terminar el presente trabajo, con interesantes puntos expuestos se alcanza el objetivo principal con la interfaz de programación por bloques para plataformas con soporte MicroPython como lo es la ESP32.

11 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

Bellido Díaz , M. (Octubre de 2015). *dte.* Obtenido de Normas Básicas y Recomendaciones en el Diseño de PCBs: https://www.dte.us.es/docencia/etsii/gii-ic/laboratorio-de-desarrollo-hardware/temas/Tema5NormasPCB/at_download/file

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.*

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

ESPloradores. (7 de Abril de 2019). *ESPloradores*. Obtenido de MICROPYTHON ESP32 – Reloj en tiempo real: https://www.esploradores.com/micropython_rtc/

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

Google. (11 de Junio de 2020). *Blockly*. Obtenido de 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.

Hirwing, E., & Roberto. (7 de Enero de 2017). *Github*. Obtenido de HC-SR04 Sensor driver in micropython: https://github.com/rsc1975/micropython-hcsr04

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

Ježek, A., & Kuethe, C. (2017). *Github*. Obtenido de MPU6050-ESP8266-MicroPython: https://github.com/adamjezek98/MPU6050-ESP8266-MicroPython

Llamas, L. (7 de Febrero de 2016). *Luis Llamas*. Obtenido de ¿CÓMO FUNCIONA UN WS2812B?: https://www.luisllamas.es/arduino-led-rgb-ws2812b/

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/

Micropython. (2019). *MicroPython*. Obtenido de WebREPL: http://micropython.org/webrepl/

MicroPython.org. (20 de Diciembre de 2019). *MicroPython 1,12.* Obtenido de WebREPL (web browser interactive prompt): https://docs.micropython.org/en/latest/esp32/quickref.html#webrepl-web-browser-interactive-prompt

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

Mozilla and individual contributors. (23 de Marzo de 2019). *MDN web docs.* Obtenido de WebSockets: https://developer.mozilla.org/es/docs/Web/API/WebSockets_API

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/

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.*