Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0624 – Laboratorio de Microcontroladores
II ciclo 2023

Proyecto Final

Arduino UNO: Carro por control IoT

Kenny Wu Wen C08592
Oscar Fallas B92861
Grupo 01

Profesor: MSc. Marco Villalta Fallas

30 de noviembre de 2023

Índice

Índice de figuras

1. Resumen

Para este proyecto se va a realizar un carro por control IoT que se pueda controlar en cualquier parte del mundo y además se controle por una aplicación en el teléfono inteligente Android. Se va a utilizar Arduino UNO, modulo de Bluetooth, controlador de motores, un modulo de carga, paneles solares y se utilizara buildozer kivy y pygame para crear la aplicación para Android.

El repositorio para el proyecto es el siguiente: GIT

2. Nota teórica

2.1. Arduino UNO

El microcontrolador Arduino UNO esta basado en el ATmega328P. Este tiene 14 pines de entrada/salida, de los cuales 6 son salidas PWM, 6 salidas analógicas, una frecuencia de 16 MHz, coenxión por usb, fuente de energía por jack y un boton de reset. En la imagen de la figura 1 podemos ver los pines de salidas/entradas.

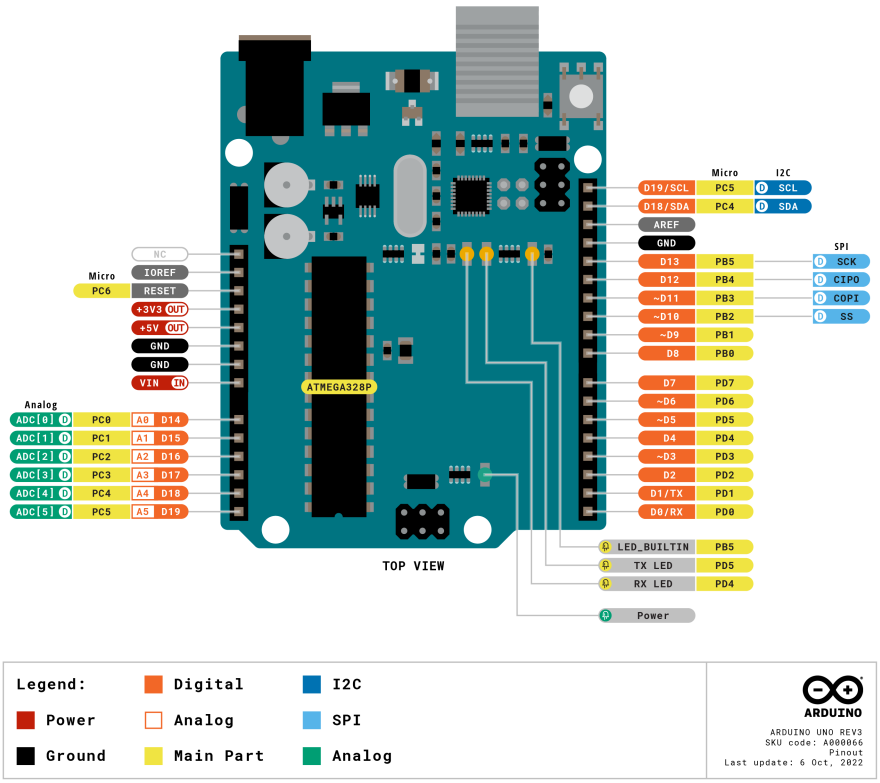

Figura 1: Pines del Arduino UNO [1]

En la figura 2, observamos las especificaciones de la hoja de fabricante sobre consumo de potencia y condiciones de operación. Además en la tabla 1, se observa las especificaciones técnicas de los pines [1].

2.1 Recommended Operating Conditions

Symbol	Description	Min	Max
	Conservative thermal limits for the whole board:	-40 °C (-40°F)	85 °C (185°F)

NOTE: In extreme temperatures, EEPROM, voltage regulator, and the crystal oscillator, might not work as expected.

2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
VINMax	Maximum input voltage from VIN pad	6	-	20	V
VUSBMax	Maximum input voltage from USB connector		-	5.5	V
PMax	Maximum Power Consumption	-	-	xx	mA

Figura 2: Consumo de potencia y condiciones de operación recomendadas [1]

Cuadro 1: Especificaciones técnicas

Board	**Name**	Arduino UNO R3
	SKU	A000066
Microcontroller	ATmega328P	
USB connector	USB-B	
Pins	**Built-in LED Pin**	13
	Digital I/O Pins	14
	Analog input pins	6
	PWM pins	6
Communication	**UART**	Yes
	I2C	Yes
	SPI	Yes
Power	**I/O Voltage**	5V
	Input voltage (nominal)	7-12V
	DC Current per I/O Pin	20 mA
	Power Supply Connector	Barrel Plug
Clock speed	**Main Processor**	ATmega328P 16 MHz
	USB-Serial Processor	ATmega16U2 16 MHz
Memory	**ATmega328P**	2KB SRAM, 32KB FLASH, 1KB EEPROM
Dimensions	**Weight**	25 g
	Width	53.4 mm
	Length	68.6 mm

2.2. Bluetooth HM-10

El modulo HM-100 es un dispositivo que nos permite tener acceso a bluetooth low energy v4.0. Este se puede controlar por el puerto serial/UART mediante comandos AT+UUID. Este viene de una familia de modulos bluetooth y tiene una memoria flash de 256Kb y aguanta una tensión de entrada de 2.2-3.7V. En la figura 3, se aprecia el esquemático del HM-100 [2].

También, en la figura 4 se aprecia los valores que utiliza el HM-10 y otros modelos de la familia HM y en la figura 5 en donde apreciamos que aguantan una corriente de 500mA.

1.1 HM-10 Schematic

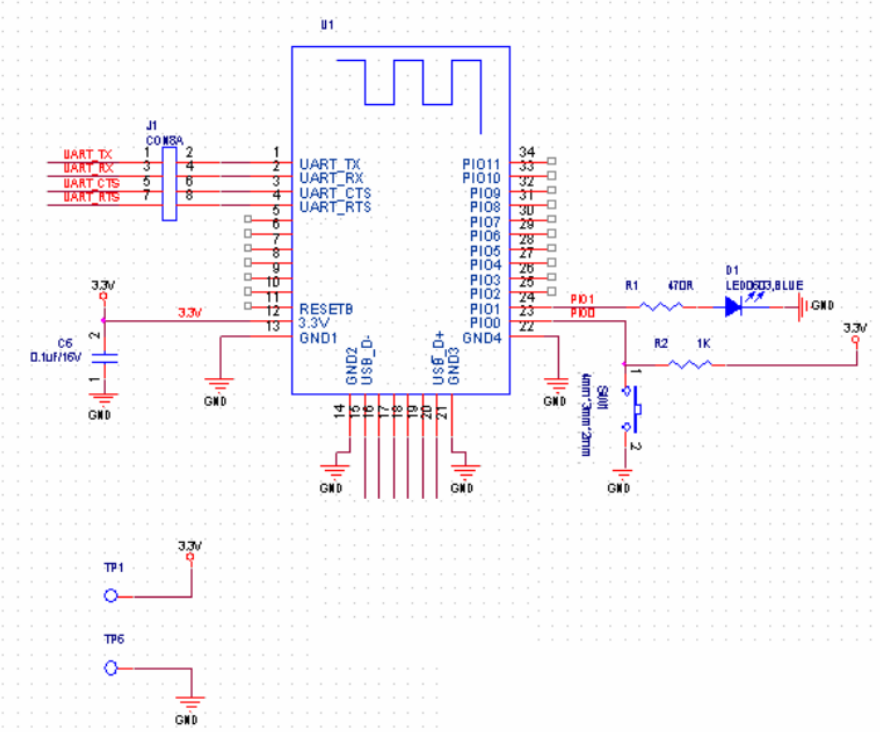

Figura 3: Esquemático del HM-10 [2]

Models	VDD	Size(mm)	Flash	Chip	BT Version
HM-01	3.3V	26.9*13*2.2	8M	BC417143	V2.1+EDR
HM-02A	2.5-3.7V	26.9*13*2.2	6M	BC31A223	V2.1
HM-02B	2.5-3.7V	26.9*13*2.2	6M	BC41C671	V2.1+EDR
HM-03A	2.5-3.7V	27.4*12.5*4.3	6M	BC31A223	V2.1
HM-03B	2.5-3.7V	27.4*12.5*4.3	6M	BC41C671	V2.1+EDR
HM-04A	3.3V	Not for sale			
HM-04B	3.3V	Not for sale			
HM-05/06A	2.5-3.7V	13.5*18.5*2.3	6M	BC31A223	V2.1
HM-05/06B	2.5-3.7V	13.5*18.5*2.3	6M	BC41C671	V2.1+EDR
HM-07	2.5-3.7V	13.5*18.5*2.3	8M		V2.1+EDR
HM-08	3.3V	26.9*13*2.5	8M	Class 1	V2.1+EDR
HM-09	2.5-3.7V	26.9*13*2.2	8M		V2.1+EDR
HM-10	2-3.7V	26.9*13*2.2	256Kb	CC2540	V4.0 BLE
HM-11	2.5-3.7V	13.5*18.5*2.2	256Kb	CC2540	V4.0 BLE

Figura 4: Familia de modulos HM [2]

Figura 5: Parámetros del HM-10 [2]

2.3. TP4056 modulo de carga

El TP4056 es un modulo de carga de corriente/tensión constante linear para un bateŕia. Además, tiene una entrada por USB y adaptor a pared. La tensión de carga está fija en 4.2V y la corriente puede ser controlada por una resistencia. Una carga completa a una batería de 1000mAh se puede apreciar los valores de tensión y corriente de entrada a este en la figura 6, en donde la corriente tiene un comportamienta parabólico y la tensión exponencial y llega a un valor máximo/constante [11].

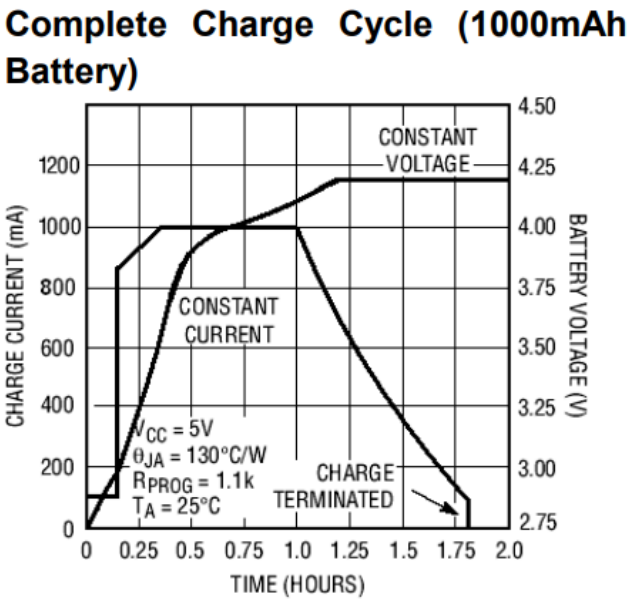

Figura 6: Un ciclo de carga completa Alldatasheet2023

La ecuación para controlar la corriente a la batería es:

$$I_{BAT} = \frac{V_{PROG}}{R_{PROG]}} \cdot 1200 \tag{1}$$

Asumiendo un $V_{PROG} = 1$V.
Tenemos la hoja de características eléctricas en la figura 7

ELECTRICAL CHARACTERISTICS

The ● denotes specifications which apply over the full operating temperature range, otherwise specifications are at T_A=25℃，V_{CC} =5V，unless otherwise noted.

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
V_{CC}	Input Supply Voltage		●	4.0	5	8.0	V
I_{CC}	Input Supply Current	Charge Mode, R_{PROG} = 1.2k	●		150	500	μA
		StandbyMode(Charge Terminated)	●		55	100	μA
		Shutdown Mode (R_{PROG} Not Connected, V_{CC} < V_{BAT}, or V_{CC} < V_{UV})	●		55	100	μA
V_{FLOAL}	Regulated Output (Float) Voltage	0℃≤T_A≤85℃，I_{BAT}=40mA		4.137	4.2	4.263	V
I_{BAT}	BAT Pin Current	R_{PROG} = 2.4k, Current Mode	●	450	500	550	mA
	Text condition:VBAT=4.0V	R_{PROG} = 1.2k, Current Mode	●	950	1000	1050	mA
		Standby Mode, V_{BAT} = 4.2V	●	0	−2.5	−6	μA
I_{TRIKL}	Trickle Charge Current	V_{BAT}<V_{TRIKL}, R_{PROG}=1.2K	●	120	130	140	mA
V_{TRIKL}	Trickle Charge Threshold Voltage	R_{PROG}=1.2K, V_{BAT} Rising		2.8	2.9	3.0	V
V_{TRHYS}	Trickle Charge Hysteresis Voltage	R_{PROG}=1.2K		60	80	100	mV
T_{LIM}	Junction Temperature in Constant Temperature Mode				145		℃

Figura 7: Características eléctricas del TP4056 Alldatasheet2023

indicator light state

Charge state	Red LED $\overline{CHRG}$	Greed LED $\overline{STDBY}$
charging	bright	extinguish
Charge Termination	extinguish	bright
Vin too low; Temperature of battery too low or too high; no battery	extinguish	extinguish
BAT PIN Connect 10u Capacitance; No battery	Greed LED bright，Red LED Coruscate T=1-4 S	

Figura 8: Indicador de estado por color de LED Alldatasheet2023

Un ejemplo de uso:

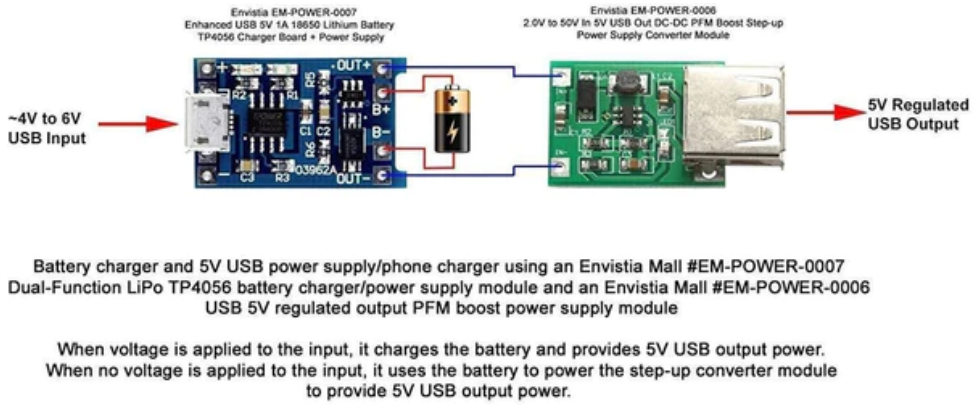

Figura 9: Ejemplo de uso del tp4056 [3]

2.4. Baterías Recargables 3.7V

El tp4056 es recomendable utilizar baterías de 3.7V, por lo que se va a utilizar baterías con esta tensión y se pueden encontrar en muchas tiendas.

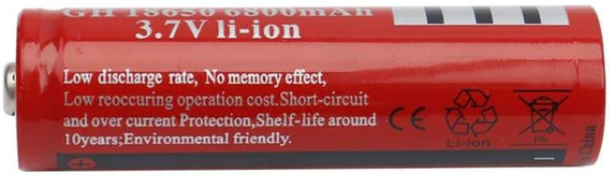

Figura 10: Batería recargable de 3.7V [4]

2.5. L293D controlador de motores

2.5.1. L293D

El circuito integrado L293D es un dispositivo monolítico que permite el control de alto voltage y corriente cuatro canales del estándar DTL o TTL, es posible su uso en motores DC, relés, solenoides, motores tipo step [5].

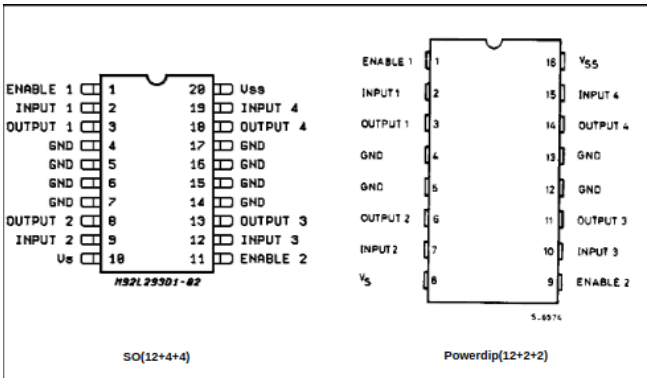

Figura 11: Diagrama L293D [5]

2.5.2. Arduino Motor Shield

Ahora bien, Arduino tiene un módulo especial que funciona como extensión de la placa que brinda un manejo más simple del IC L293D.

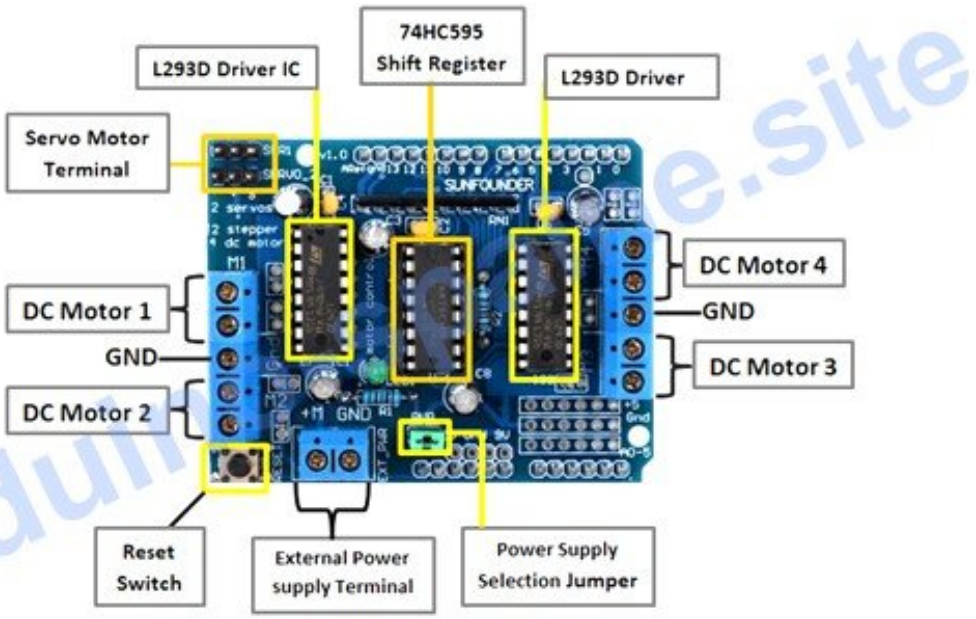

Figura 12: Modulo controlador de Motores [6]

Entre sus características más importantes se encuentran las siguientes:

- Alimentación de 4.5 a 24V
- Compatibilidad con Arduino Uno, Mega y Due

- Corriente por canal de 600mA
- Corriente pico por canal de 1.2A
- Posibilidad de controlar hasta 4 motores de corriente directa por control del velocidad y dos motores a pasos.

2.6. Paneles solares 0.5W

Los paneles solares tipo mini son especiales para el desarrollo de proyectos pequeños de electrónica, con la capacidad de proveer de autonomía al diseño. Los paneles solares de 0.5W son capaces de proveer hasta 5V con una corriente de 100mA.

Figura 13: Panel solar tipo MINI de 0.5W.[3]

2.7. ESP8266 EP-01

El ESP8266 es un microcontrolador de la familia ESP que funciona especialmente como módulo de WiFi. El integrado se basa el protocolo IEEE 802.11b/g/n soportando 2.4GHz.

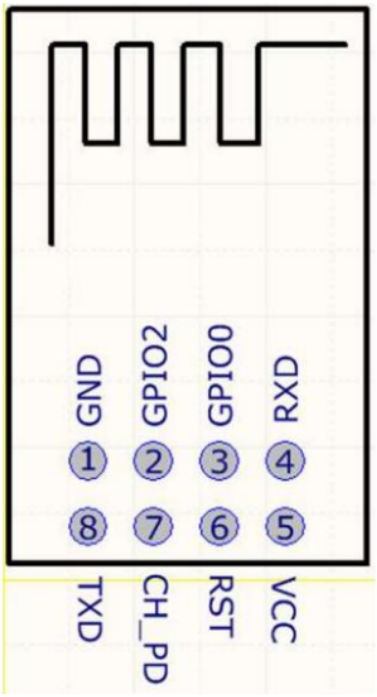

Figura 14: Diagrama de pines ESP8266-01 [7]

Características ESP8266-01	
MCU	ESP8266 es un dispositivo incrustado con Tensilica L106 de 32-bit, con características de extra bajo consumo de potencia basado en arquitectura RISC.
Tensión de Operación	3.0 V - 3.6 V
Entrada lógica en bajo	-0.3 - 0.25 VDD
Entrada lógica en alto	0.75VDD - VDD+0.3
Interfaces disponibles	HSPI, PWM, IR Remote Control, ADC, I2C, UART, I2S
External SPI Flash	El modulo integra un 1 MB externo para la programación externa usuario

Cuadro 2: Tabla de especificaciones ESP8266-01. Información tomada de [7]

2.8. IoT

El Internet de las Cosas(IoT) es una red conectada a dispositivos físicos, instrumentos, vehículos, o cualquier otro dispositivo incrustado que permite la recolección e intercambio de datos. El internet de las cosas permite a los dispositivos ser controlados a distancia con una infraestructura de red, creando así la oportunidad de una integración e interacción más directa al mundo físico de parte de las computadoras. [12]

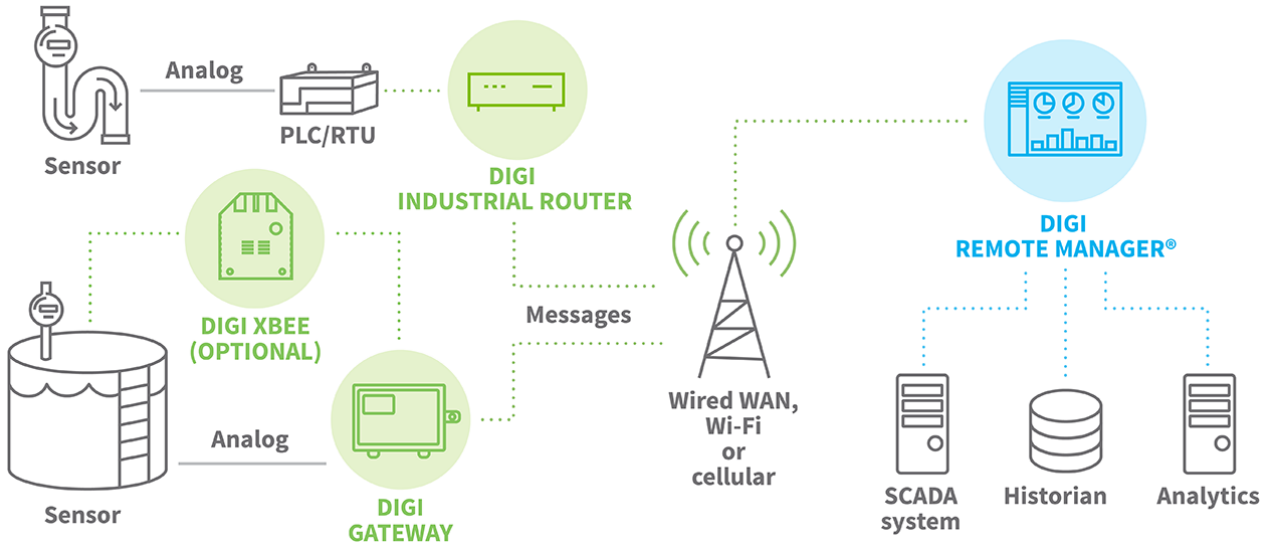

Figura 15: Arquitectura de ejemplo de IoT[8].

2.9. MQTT

Mosquitto o sus siglas en ingles MQTT, es un protocolo de estándar de mensajería para el internet de las cosas (IoT). MQTT es ligero, fácil de usar, se envía y recibe mensajes con publicaciones o suscripciones [9].

Para este proyecto se utilizara tanto la plataforma IoT de Thingsboard de EIE y EMQX que nos provee un broker público y gratis [10].

En el apéndice se encuentra la tabla de errores rc de MQTT.

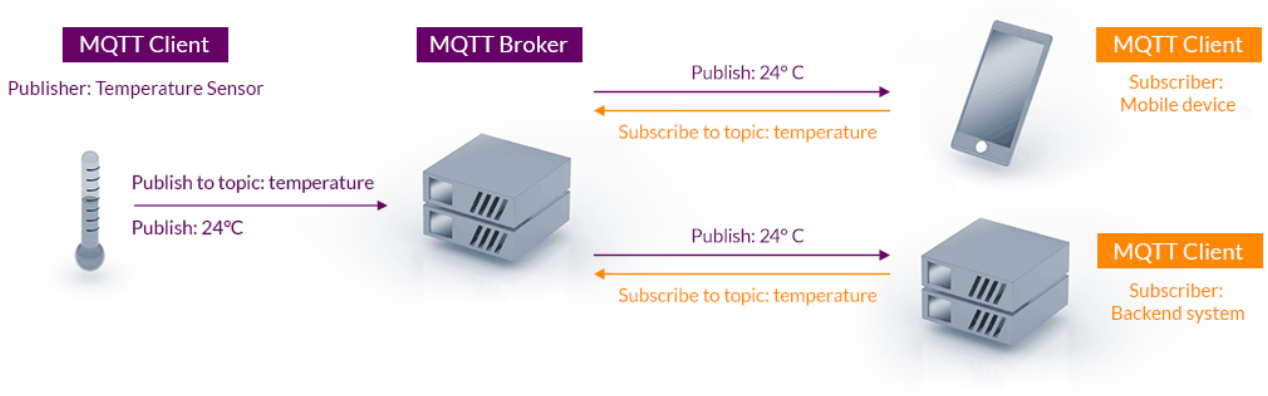

Figura 16: Arquitectura MQTT de suscripciones y publicaciones [9]

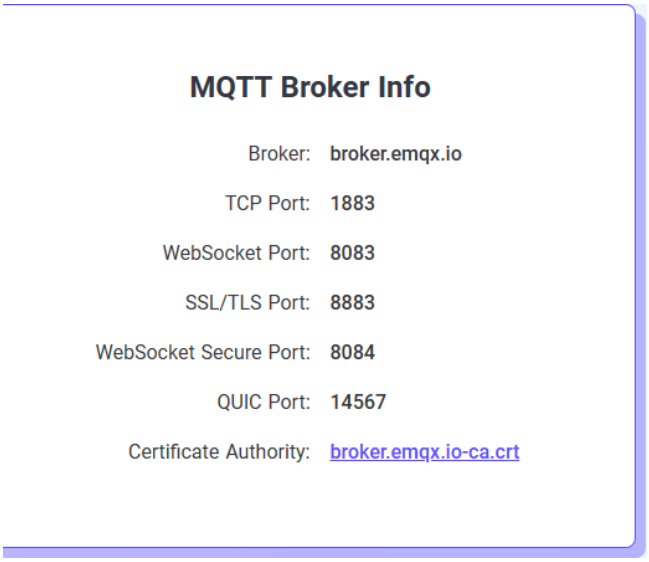

Figura 17: EMQX broker publico gratis [10]

2.10. Buildozer Kivy

Buildozer es una herramienta que apunta a empaquetar aplicaciones móviles fácilmente. Este automatiza el proceso de construir y descargar todos los requisitos necesarios para que un programa de python corra en Android [13].

Se puede configurar en el archivo buildozer.spec las preferencias de la aplicación a construir, como pantalla completa, librerías necesarias, carpeta de imágenes, icono, nombre de la aplicación, acceso a internet, permisos, entre otros.

2.11. Herramienta BLE-serial

El concepto de BLE se refiere a conexión Bluetooth en dispositivos de bajo consumo de energía. Cuando un dispositivo envía una pequeña cantidad de datos por un corto periodo de tiempo se considera de bajo consumo de energía. Por lo general, esto es difícil de detectar para los dispositivos como teléfonos y computadores por lo que su conexión en muchos casos debe realizarse de manera diferente [14].

2.11.1. BLE Serial

Esta es una herramienta que conecta Bluetooth 4.0+ Low Energy entre sus modulo UART y PC, laptops o RaspberryPi. Esta disponible tanto en sistemas Linux como Windows, para el caso de crea un puerto virtual entre rfcomm y uno serial. Para Windows lo hace de manera similar conectando de manera virtual uno de los puertos COM con el Standard Serial over Bluetooth de Microsoft. La instalación y uso es sencilla y amigable con el usuario encontrandose en el siguiente link https://github.com/Jakeler/ble-serial

2.12. Pygame

Pygame es un conjunto de módulos ed Python que son creadas para crear video juegos. Sus ventajas son que, además se utilizar Python como lenguaje de programación, sus librerias son fáciles de usar y hay una gran comunidad detrás de todo por lo cual es fácil aprender el utilizar los módulos de Pygame [15].

2.13. Comm.py

Este script de python de auditoria propia se implementó dada la necesidad de recepción de datos a través de MQTT, su análisis y envío por medio de un puerto serial virtualizado anteriormente.

Se inicia el software importando las líbrerias de Python necesarias e inicializando las variables de MQTT y de la conexión serial

```
import random
import serial
from paho.mqtt import client as mqtt_client
from re import search as re_search


broker = 'broker.emqx.io'
port = 1883
topic = "carrito"
# Generate a Client ID with the subscribe prefix.
client_id = f'subscribe-{random.randint(0, 100)}'
command = ''
# username = 'emqx'
# password = 'public'

# Obtain data
ser = serial.Serial(
port = 'COM9',\
baudrate = 9600,\
timeout=1\
)
```

Se realiza la función que permite la conexión al broker mqtt, asignandolo como cliente, para posteriormente crear una función que permite la recepción del mensaje enviado, decodificación y envío a través de la librería serial.

```
def connect_mqtt() -> mqtt_client:
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
```

```
        else:
            print("Failed to connect, return code %d\n", rc)

    client = mqtt_client.Client(client_id)
    # client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client


def subscribe(client: mqtt_client):
    def on_message(client, userdata, msg):
        print(f"Received {msg.payload.decode()} from {msg.topic} topic")
        command = msg.payload.decode()
        reg = r'messages:\s+(?P<comando>\w+)'
        reg_result = re_search(reg,command).group('comando')
        if(reg_result == 'ARRIBA'):
            ser.write(b'F')
        elif(reg_result == 'ABAJO'):
            ser.write(b'B')
        elif(reg_result == 'IZQUIERDA'):
            ser.write(b'L')
        elif(reg_result == 'DERECHA'):
            ser.write(b'R')
        elif(reg_result == 'STOP'):
            ser.write(b'S')


    client.subscribe(topic)
    client.on_message = on_message
```

Finalmente, se crea una función que conecta, llama a los métodos y crea un bucle infinito para que el proceso se repita.

```
def run():
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()

if __name__ == '__main__':
    run()
```

2.14. Interfaz para Android

Para la interfaz vamos a utilizar Python con Pygame. En la figura 18 se aprecia el como funciona el código en el anexo.

1. Primero se importa las librerias necesarias como MQTT y Pygame.
2. Luego definimos y configuramos las variables tanto para Pygame y MQTT.
3. Después, se va a llamar la función main(), el cual va a crear los botones
4. Al crear los botones, se entra en un loop sin fin el cual va a dibujar los botones en la pantalla.

5. Si detecta que se presiono algún botón, se va a publicar por MQTT un mensaje dependiendo del botón que se presione.

6. Si no se presiona ningún botón simplemente no ocurre nada.

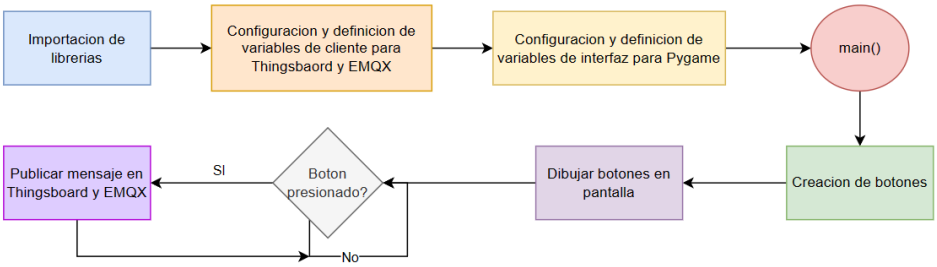

Figura 18: Diagrama de flujo de la interfaz de Pygame

El resultado final de la interfaz es la mostrada en la figura 19.

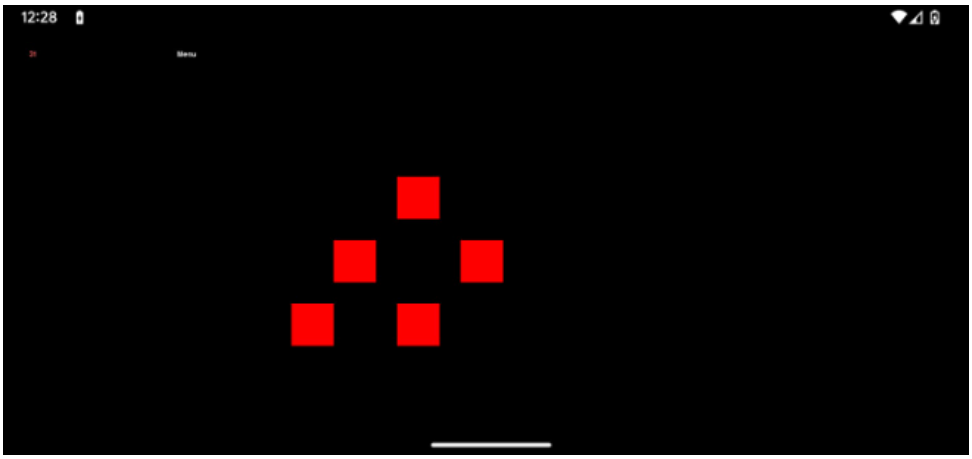

Figura 19: Interfaz funcionando en telefono Android

2.15. Firmware Arduino UNO

Nuestro microcontrolador principal es un Arduino UNO, que a pesar que se tiene un módulo controlador de motores, el Arduino será el encargado de indicar como se deben manipular. Para el control del módulo se utilizó la libería Adafruit Motor Shield library.

Se incia incia importando la librería, y asignando dos variables heredadas de la libería que asignan los motores a utilizarse.

```
#include <AFMotor.h>
char direccion;

AF_DCMotor motor(1);
AF_DCMotor motor2(4);
```

Se configura la conexión serial, esta es importante ya que la recepción y envío de datos hacía la tarjeta bluetooth se harán de manera serial por los puerto Tx y Rx del Arduino. Además, se inicializan los motores quietos.

```
void setup() {
  Serial.begin(9600);            // set up Serial library at 9600 bps
  Serial.println("READY to Drive");


  motor.run(RELEASE);
  motor2.run(RELEASE);
}
```

Se crean cuatro funciones que designarán el tipo de movimiento que se quiere en los motores: adelante, atrás, izquierda, derecha y detenido.

```
 void forward()
{
  motor.setSpeed(255);
  motor.run(FORWARD);
  motor2.setSpeed(255);
  motor2.run(FORWARD);
}

void back()
{
  motor.setSpeed(255);
  motor.run(BACKWARD);
  motor2.setSpeed(255);
  motor2.run(BACKWARD);
}

void left()
{
  motor2.setSpeed(255);
  motor2.run(FORWARD);
}

void right()
{
  motor.setSpeed(255);
  motor.run(FORWARD);
}

void Stop()
{
  motor.setSpeed(0);
  motor.run(RELEASE);
  motor2.setSpeed(0);
  motor2.run(RELEASE);
}
```

Finalmente, el lazo principal que mediante la lectura serial clasificará el dato y llmará al tipo de movimiento indicado.

```
void loop() {

  if(Serial.available() > 0){
```

```
    direccion = Serial.read();
    if(direccion == 'F'){
      Stop();
      forward();
      }
    else if(direccion == 'B'){
      Stop();
      back();
      }
    else if(direccion == 'L'){
      Stop();
      left();
      }
    else if(direccion == 'R'){
      Stop();
      right();
    }
    else if(direccion == 'S'){
      Stop();
    }
    }
}
```

3. Desarrollo/Análisis de resultados

3.1. Análisis de funcionalidad electrónica

Inicialmente se construyó el carro únicamente alimentado por una batería cuadrada y las batería recargables previamente cargadas. En la imagen se puede ver las conexiones y el funcionamiento de la tarjeta bluetooth

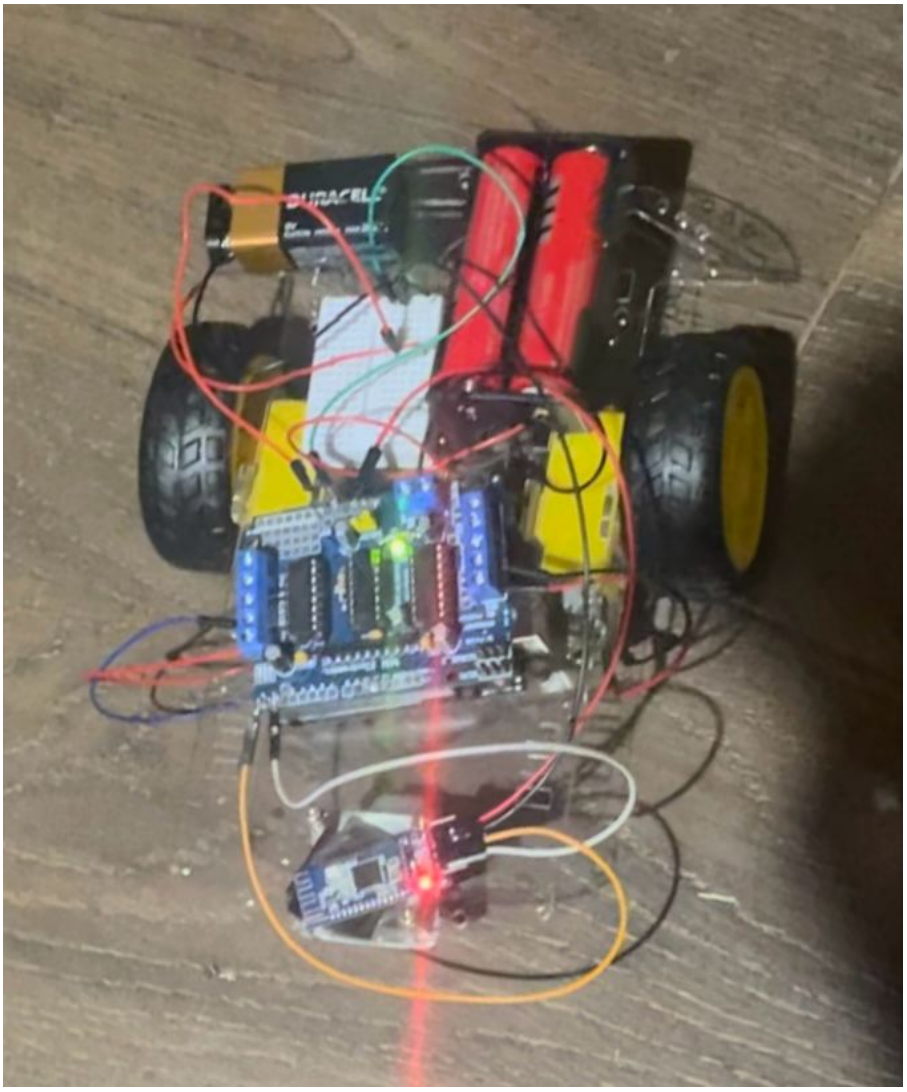

Figura 20: Modelo inicial del carro

Posteriormente, se le implementó un segundo piso que contendría los paneles solares para que el carro pueda utilizarse y cargarse al mismo tiempo cuando se encuentre en exteriores.

Finalmente, está la figura 22 demuestra la completa construcción del carro. Además, en el siguiente video

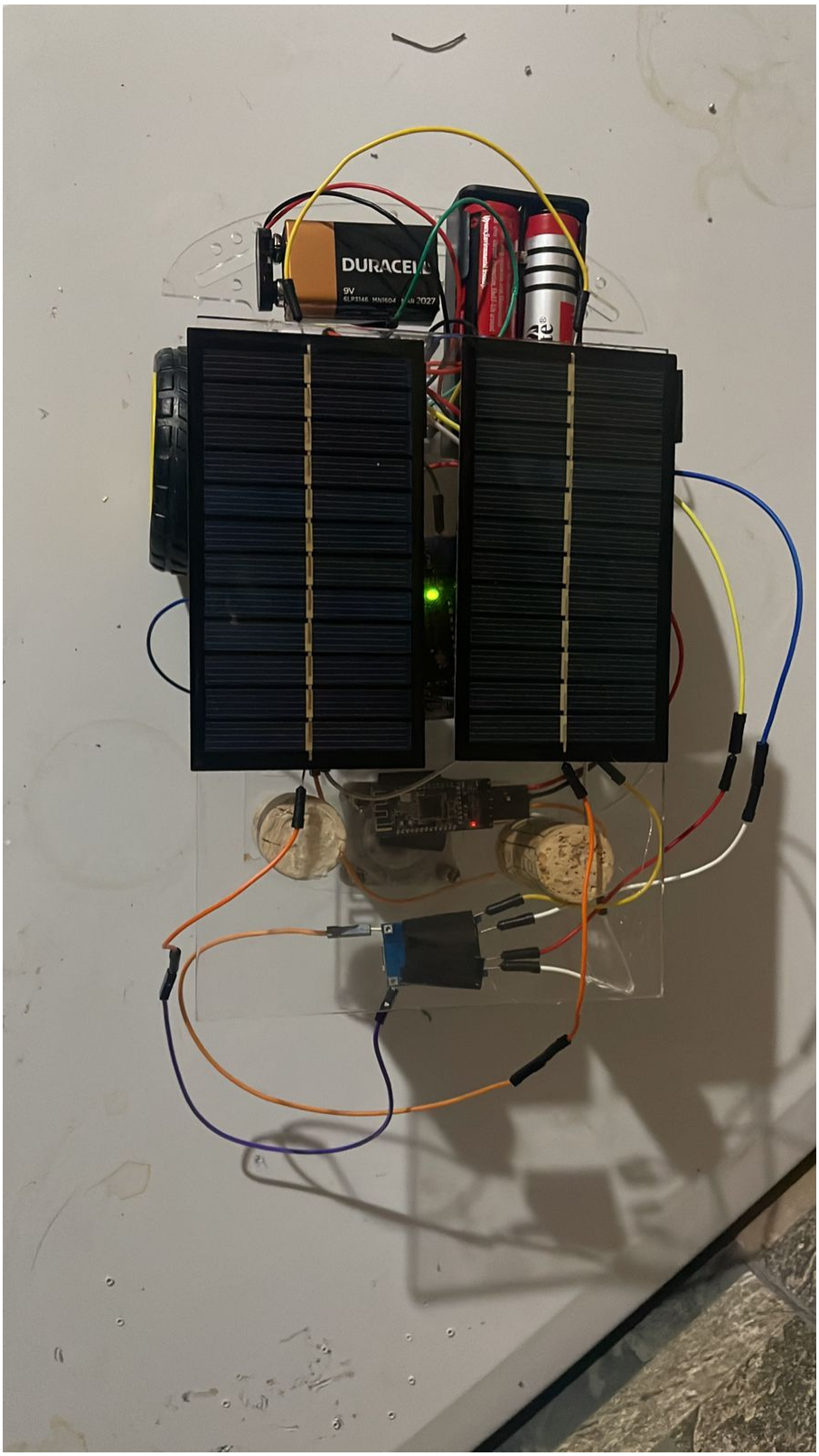

Figura 21: Modelo inicial del carro

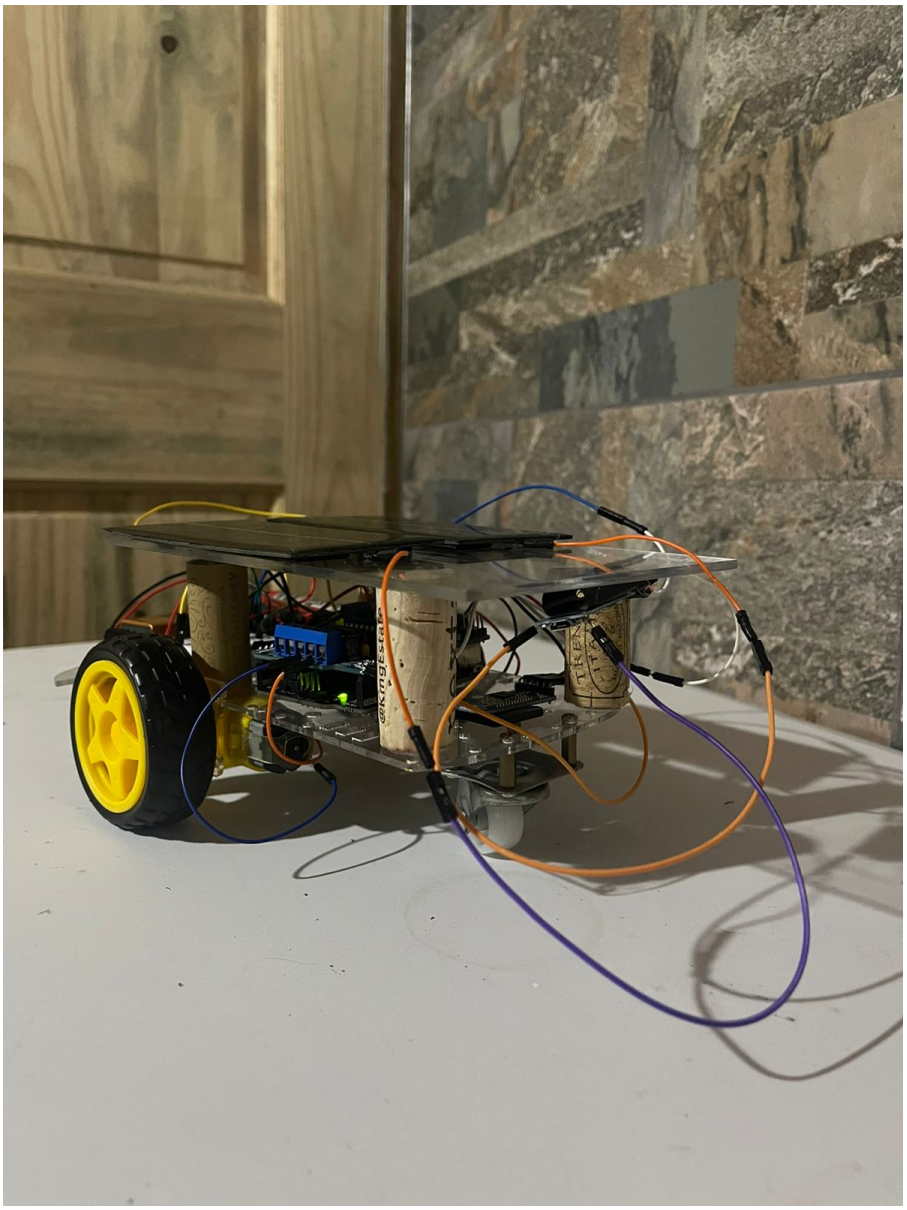

Figura 22: Modelo inicial del carro

3.2. Análisis de la funcionalidad del programa

Para probar la funcionalidad se puede observar en la imagen de la figura 24, que el Thingsboard recibe el mensaje izquierda al presionar el botón izquierda de la interfaz en Android y además en la figura 25 se aprecia la publicación a ambos plataformas IoTs.

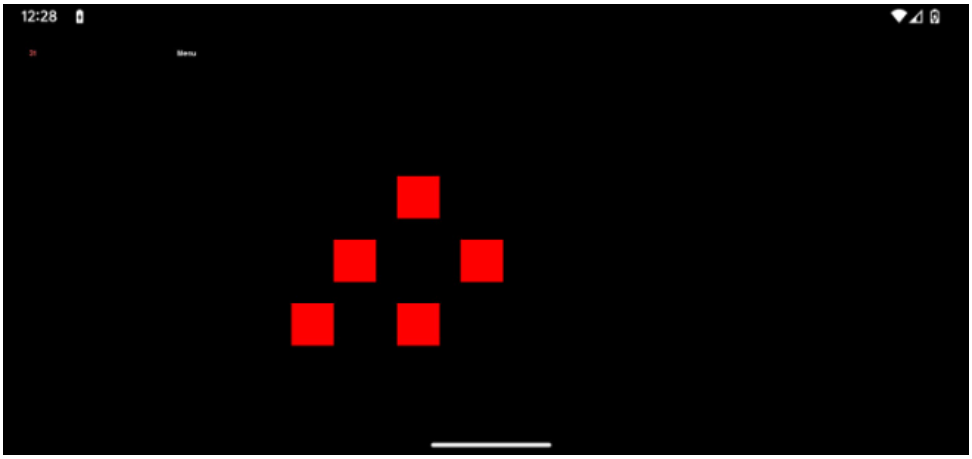

Figura 23: Interfaz final funcionando en Android

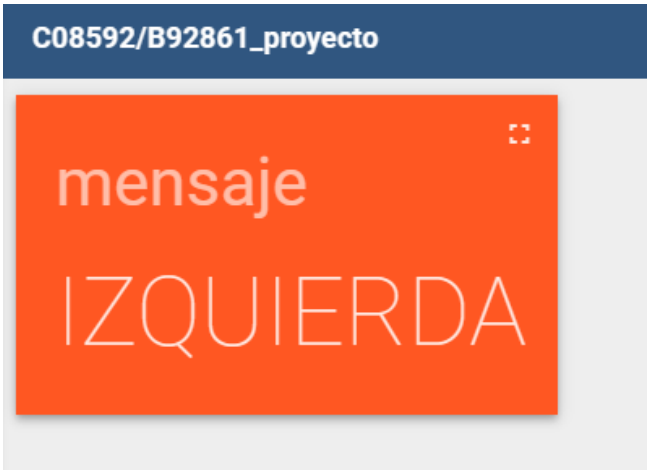

Figura 24: Thingsboard EIE recepción de publicación del interfaz de Android

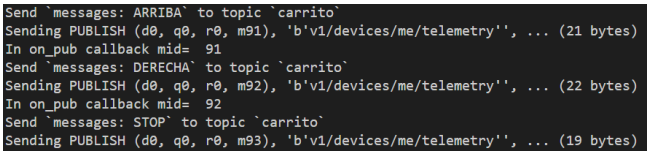

Figura 25: Publicación de los dos clientes, Thingsboard y EMQX

4. Conclusiones y recomendaciones

- Se logró implementar un modelo de control inalámbrico para el carro.
- A pesar de que hubo complicaciones con el diseño inicial dada que la tarjeta WiFi no funcionó por problemas de hardware de fábrica, se ideó una solución muy parecida con el uso de la tarjeta bluetooth, IoT y protocolo MQTT.

- Se mejoraron los conocimiento en electrónica básica y manejo del cautín.
- Se logró implementar una aplicación exitosa utilizando PyGame para posteriormente utilizar Buildozer para convertirlo a una aplicación Android.
- Se aprendió sobre Bluetooth LE.
- Como recomendación, utilizar un modulo de wifi ESP32 y no el ESP8266 al ser el ESP32 más estable

Referencias

[1] Arduino, "Uno r3 — arduino documentation," 2023, accesado: 2023-12-07. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3

[2] J. T. Company, "Bluetooth 4.0 ble module datasheet v507," 2023, accesado: 2023-12-07. [Online]. Available: https://components101.com/sites/default/files/component_datasheet/HM10%20Bluetooth%20Module%20Datasheet.pdf

[3] Centroniks, "Módulo tp4056 — cargador de baterías de litio — 5v / 1a — ce-car-03," 2023, accesado: 2023-12-07. [Online]. Available: https://centroniks.com/products/modulo-tp4056-cargador-de-baterias-de-litio-5v-1a-ce-car-03

[4] ——, "Pila / batería 18650 recargable — 3.7v / 1300mah — ce-bat-15," 2023, accesado: 2023-12-07. [Online]. Available: https://centroniks.com/collections/electronica/products/pila-bateria-18650-recargable-3-7v-1300mah-ce-bat-15

[5] Alldatasheet, "L293d datasheet — alldatasheet," 2023, accesado: 2023-12-07. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/22432/STMICROELECTRONICS/L293D.html

[6] A. Spain, "Conectar l293d arduino motor shield, datasheet," 2023, accesado: 2023-12-07. [Online]. Available: https://arduino-spain.site/shield-l293d/

[7] AIThinker, "Esp-01 wifi module," 2015, accesado: 2023-12-07. [Online]. Available: https://www.microchip.ua/wireless/esp01.pdf

[8] DIGI, "The 4 stages of iot architecture," 2023, accesado: 2023-12-07. [Online]. Available: https://www.digi.com/blog/post/the-4-stages-of-iot-architecture

[9] MQTT, "Mqtt - the standard for iot messaging," 2023, accesado: 2023-12-07. [Online]. Available: https://mqtt.org/

[10] EMQX, "The free global public mqtt broker — try now - emqx," 2023, accesado: 2023-12-07. [Online]. Available: https://www.emqx.com/en/mqtt/public-mqtt5-broker

[11] Alldatasheet, "Tp4056 datasheet — alldatasheet," 2023, accesado: 2023-12-07. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/download/1487471/ETC2/TP4056.html

[12] P. Gokhale, O. Bhat, and S. Bhat, "Introduction to iot," *International Advanced Research Journal in Science, Engineering and Technology*, vol. 5, no. 1, pp. 41–44, 2018.

[13] R. the Docs, "Buildozer — read the docs," 2023, accessed: 2023-12-07. [Online]. Available: https://buildozer.readthedocs.io/en/latest/#indices-and-tables

[14] B. L. Wiki, "Bluetooth le wiki: The unofficial repository all about bluetooth le," 2023, accesado: 2023-12-07. [Online]. Available: https://bluetoothle.wiki/overview

[15] Pygame, "About - pygame wiki," 2023, accessed: 2023-12-07. [Online]. Available: ˆ1ˆ

5. Apéndice

5.1. Precios de componentes

Cuadro 3: Precios de cada componente

Componente	**Precio (Colones)**	**Cantidad**
Arduino UNO	12300	1
L293D	3700	1
TP4056	3700	1
Chassis de Carro 2 Ruedas	11000	1
HM-10	7500	1
Panel solar	3000	2
ESP8266-01	5000	1
Baterías de Litio Recargables	4990	2
Holder 2 Slots de Batería	800	1

5.2. Código de interfaz

```
import pygame, sys, time, random, json
from pygame.locals import *
from paho.mqtt import client as mqtt_client
import paho.mqtt.client as mqtt
#CLIENT METHODS
def connect_mqtt():
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
        else:
            print("Failed to connect, return code %d\n", rc)

    client = mqtt_client.Client(client_id)
    # client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client


def publish(client, msg):
        msg = f"messages: {msg}"
        result = client.publish(topic, msg)
        # result: [0, 1]
        status = result[0]
        if status == 0:
                print(f"Send `{msg}` to topic `{topic}`")
        else:
                print(f"Failed to send message to topic {topic}")

def on_connect(client, userdata, flags, rc):
    if not rc:
        client.is_connected = True
        print('Connection established. Working')
    else:
        print('Connection Faild', rc)
        client.loop_stop()

def on_disconnect(client, userdata, rc):
    if(rc == 0):
        print("Client disconneted OK")
    else:
        print("System disconnected via code: ", rc)

def on_log(client, userdata, level, buf):
   print(buf)

def on_publish(client, userdata, mid):
    print("In on_pub callback mid= ", mid)

#SETUP
broker = 'broker.emqx.io'
```

```
port = 1883
topic = "carrito"
# Generate a Client ID with the publish prefix.
client_id = f'publish-{random.randint(0, 1000)}'
# username = 'emqx'
# password = 'public'

client = connect_mqtt()

client2 = mqtt.Client('B928611')
client2.on_connect    = on_connect
client2.on_disconnect = on_disconnect
client2.on_publish    = on_publish
client2.on_log        = on_log
client2.is_connected  = False
port        =   1883
broker      =   "iot.eie.ucr.ac.cr"
topic2       =   "v1/devices/me/telemetry"
username    =   'C08592/B92861'
password    =   'brjxmov6h994bpdea2fr'
client2.username_pw_set(password)
client2.connect(broker, port)
while not client2.is_connected:
    client2.loop()
    time.sleep(0.5)

# Parametros
ANCHO = 800
ALTURA = 600
NOMBRE_VENTANA = "Interfaz"
FONT_SIZE = 25
# INICIAMOS PYGAME
pygame.init()
# Fotogramas por segundo
clock = pygame.time.Clock()
# Nombre de nuestra ventana
pygame.display.set_caption(NOMBRE_VENTANA)
# Dimensiones de la ventana
screen = pygame.display.set_mode((ANCHO, ALTURA), pygame.RESIZABLE)
#Fuente de las letras
FONT = pygame.font.SysFont(None, FONT_SIZE)


# Metodo escribir texto en ventana
def draw_text(text, font, color, surface, x, y):
    textobj = font.render(text, 1, color)
    textrect = textobj.get_rect()
    textrect.center = (x, y)
    surface.blit(textobj, textrect)

#funcion para crear botones
def dibujar_boton(nombre, coor_x, coor_y, constante):
    boton = pygame.image.load(nombre).convert()
```

```
    boton.set_colorkey([255, 255, 255])
    boton = pygame.transform.scale(boton,
                                   (220 + constante // 2, 85 + constante
                                      // 2))
    screen.blit(boton, [(ANCHO / 2 - coor_x - constante // 4),
                        (ALTURA / 2 - coor_y - constante // 4)])

#funcion para dibujar en pantalla
def dibujar_pant(screen, color, objeto):
    pygame.draw.rect(screen, color, objeto)


def main():
        click = False
        msg = ""
        start_time = time.localtime(time.time()).tm_sec
        # Creamos los botones
        botton_up = pygame.Rect(float(ANCHO / 2 - 50),
                                float(ALTURA / 2 - 200),
                                100, 100)
        botton_dwn = pygame.Rect(float(ANCHO / 2 - 50),
                                float(ALTURA / 2 + 100),
                                100, 100)
        botton_left = pygame.Rect(float(ANCHO / 2 + 100),
                                float(ALTURA / 2 - 50),
                                100, 100)
        botton_right = pygame.Rect(float(ANCHO / 2 - 200),
                                float(ALTURA / 2 - 50),
                                100, 100)
        botton_bt = pygame.Rect(float(ANCHO / 2 - 300),
                                                float(ALTURA / 2 + 100),
                                                100, 100)
        while True:
                screen.fill(pygame.Color("black"))
    # Dibujar en la ventana
                dibujar_pant(screen, (255, 0, 0), botton_up)
                dibujar_pant(screen, (255, 0, 0), botton_dwn)
                dibujar_pant(screen, (255, 0, 0), botton_left)
                dibujar_pant(screen, (255, 0, 0), botton_right)
                dibujar_pant(screen, (255, 0, 0), botton_bt      )
                draw_text('Menu', FONT, (255,255,255), screen, 412, 50)
                draw_text(msg, FONT, (255,255,255), screen, ANCHO/2,
                   ALTURA/2)
                draw_text(str(time.localtime(time.time()).tm_sec), FONT,
                   (255,100,100), screen, 50, 50)
                # Logica de clicks
                mx, my = pygame.mouse.get_pos()
                if click:
                        if botton_up.collidepoint((mx, my)):
                                msg = "ARRIBA"
                        elif botton_dwn.collidepoint((mx,my)):
                                msg = "ABAJO"
                        elif botton_left.collidepoint((mx,my)):
```

```
                        msg = "IZQUIERDA"
                    elif botton_right.collidepoint((mx,my)):
                        msg = "DERECHA"
                    elif botton_bt.collidepoint((mx,my)):
                        msg = "STOP"
                    publish(client, msg)
                    client2.publish(topic2, json.dumps({"mensaje":
                        msg}))
                click = False

                # Eventos
                for event in pygame.event.get():
                        # Si es una tecla y es Esc
                        # cierre todo
                        if event.type == KEYDOWN and event.key ==
                            K_ESCAPE:
                                pygame.quit()
                                sys.exit()
                        # Si es un click del mouse izquierdo
                        # click se vuelve true
                        if event.type == MOUSEBUTTONDOWN:
                                if event.button == 1:
                                        click = True
                        # Ventana dinamica
                        if event.type == pygame.VIDEORESIZE:
            # There's some code to add back window content here.
                                surface = pygame.display.set_mode((event.
                                    w, event.h),
                                                                  pygame.RESIZABLE)
                pygame.display.update()
                clock.tick(60)

main()
```

Cuadro 4: Tabla de errores de MQTT

Error code (Decimal)	**Error code (Hex)**	**Meaning**
0	0x0	No Error
1	0x1	Connection Refused: Unacceptable protocol version
10	0xa	Timeout waiting for SUBACK
11	0xb	Timeout waiting for UNSUBACK
12	0xc	Timeout waiting for PINGRESP
13	0xd	Malformed Remaining Length
14	0xe	Problem with the underlying communication port
15	0xf	Address could not be parsed
16	0x10	Malformed received MQTT packet
17	0x11	Subscription failure
18	0x12	Payload decoding failure
19	0x13	Failed to compile a Decoder
2	0x2	Connection Refused: Identifier rejected
20	0x14	The received MQTT packet type is not supported on this client
21	0x15	Timeout waiting for PUBACK
22	0x16	Timeout waiting for PUBREC
23	0x17	Timeout waiting for PUBCOMP
3	0x3	Connection Refused: Server Unavailable
4	0x4	Connection Refused: Bad username or password
5	0x5	Connection Refused: Authorization error
6	0x6	Connection lost or bad
7	0x7	Timeout waiting for Length bytes
8	0x8	Timeout waiting for Payload
9	0x9	Timeout waiting for CONNACK