TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES
Ayala Boulevard, Ermita, Manila
CIT-ELECTRONICS DEPARTMENT

CPET11L-M – Microprocessor and Microcontroller Systems, Lab
1st Semester, SY 2-24-2025

"ATOM THE RECYCABOT"

A SUMO-BOT PROJECT

MEMBERS:

ARENAS, JOSEPH C,

ESTRADA, ADRIENE CYRUZ

GUTIERREZ, GEO KENTZER

PACIS, LIAN GIL B.

RECAÑA, JORDAN

NOVEMBER 2025

I. OBJECTIVES

- To apply the ESP32 microcontroller for controlling an autonomous mobile robot.
- To design an opponent detection system using Ultrasonic sensors.
- To leverage the ESP32's processing power and connectivity (Wi-Fi/Bluetooth) for real-time telemetry or advanced strategies.
- To enhance skills in C++ programming for dual-core microcontrollers, mechanical assembly, and real-time logic.

II. EQUIPMENT AND MATERIALS

HARDWARE

- Any flat metal plate
- Barrel Jack
- Cellphone
- DC Motor 12V; 250 RPM (x2)
- ESP32
- Jumper Wires (Male to Male and Male to Female)
- Junction Box
- L298N Motor Driver
- Laptop
- LED (one red, one blue)
- Lithium Battery 3.7V; 2200 mAh (x3)
- Nuts
- Philip’s Screw Drivers
- Sintra Board
- SPDT Power Switch
- Tact Switch (x2)
- Ultrasonic Sensor
- Rubber Wheel (x2)
- Wood Blocks

SOFTWARE

- Arduino IDE
- Dabble Application (must be downloaded through google browser – for android /app store – for IOS)

III. DIAGRAM

A. Block Diagram

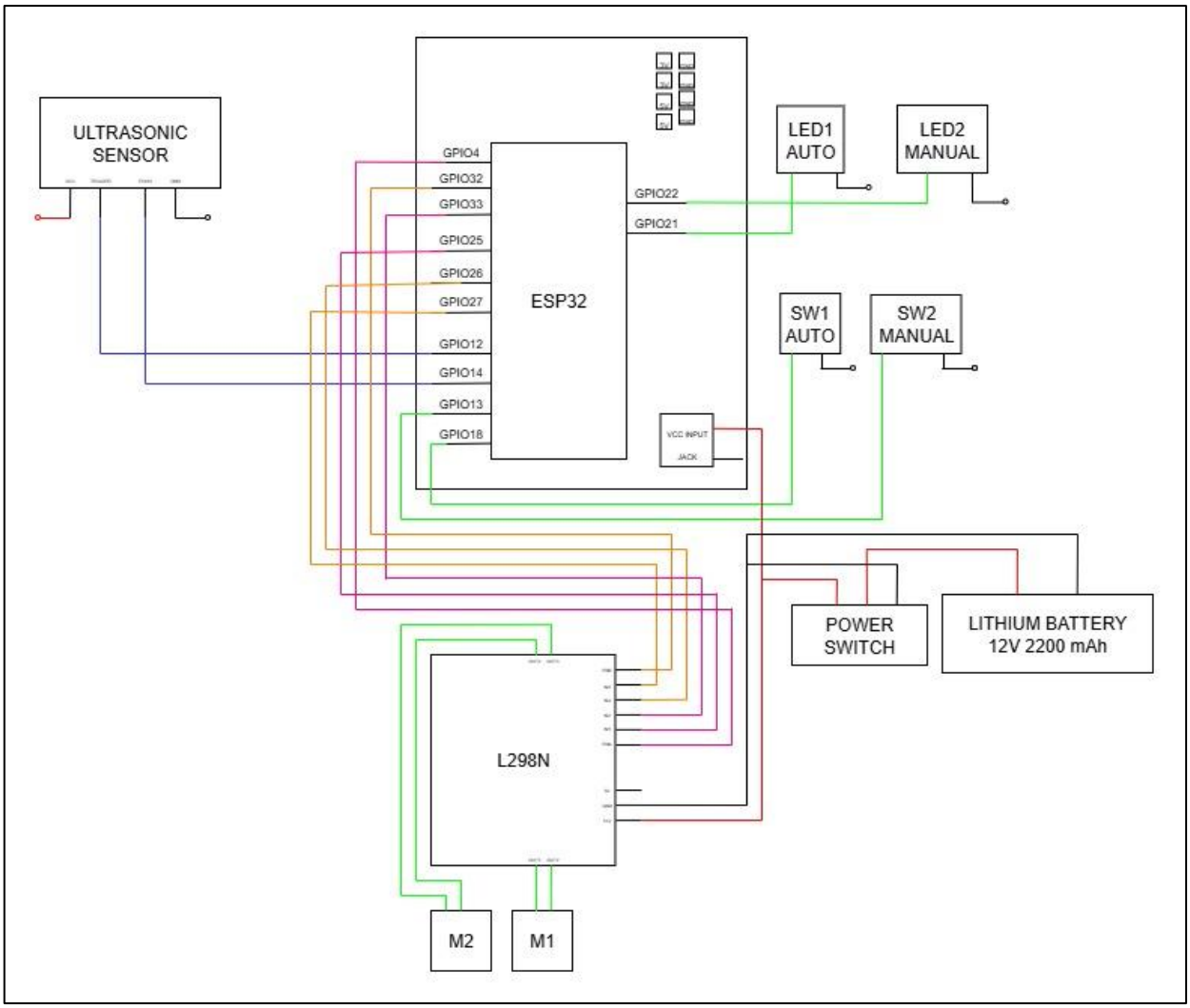

FIGURE 1

B. Pictorial Diagram

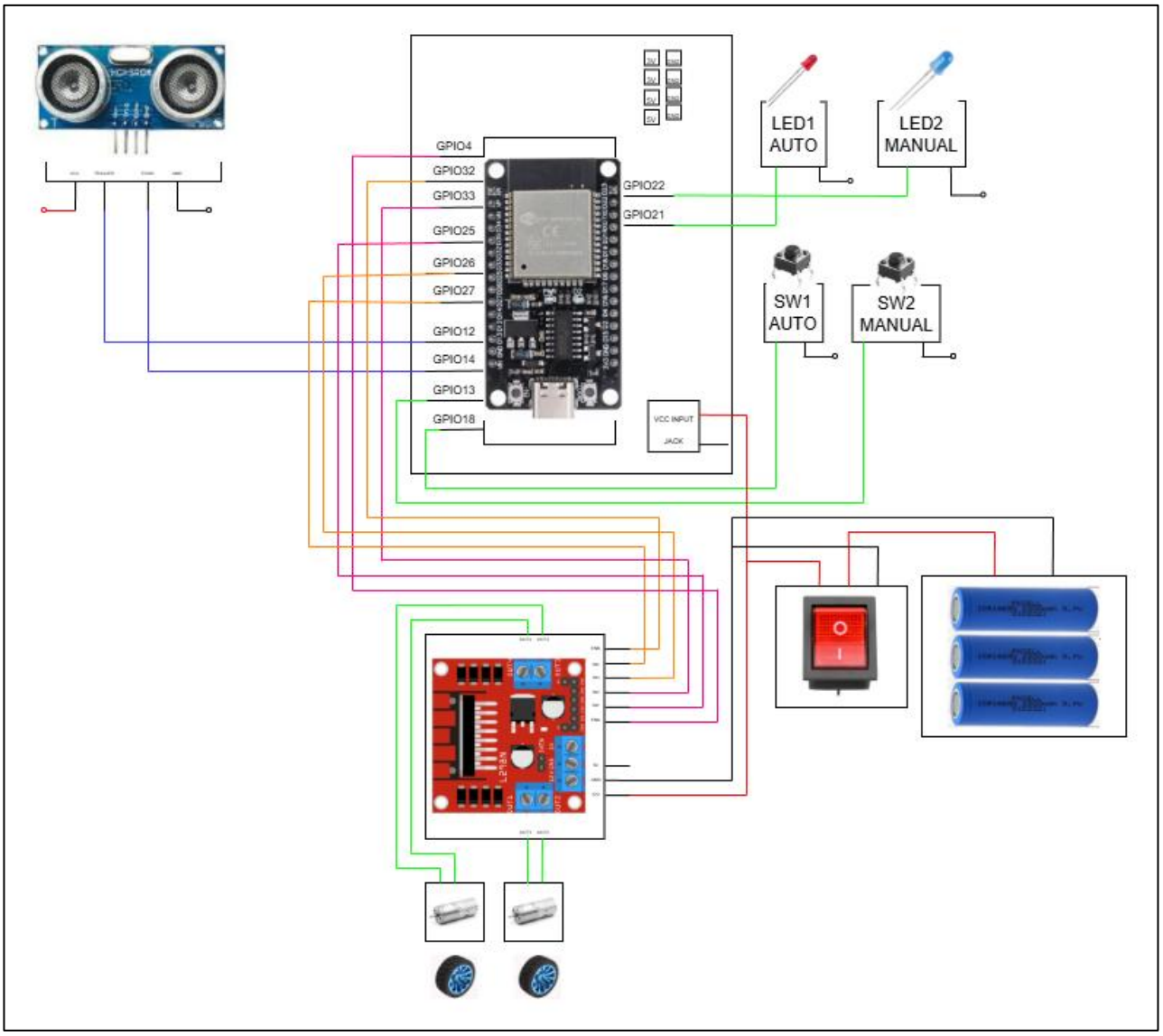

FIGURE 2

C. Actual Sumo Bot

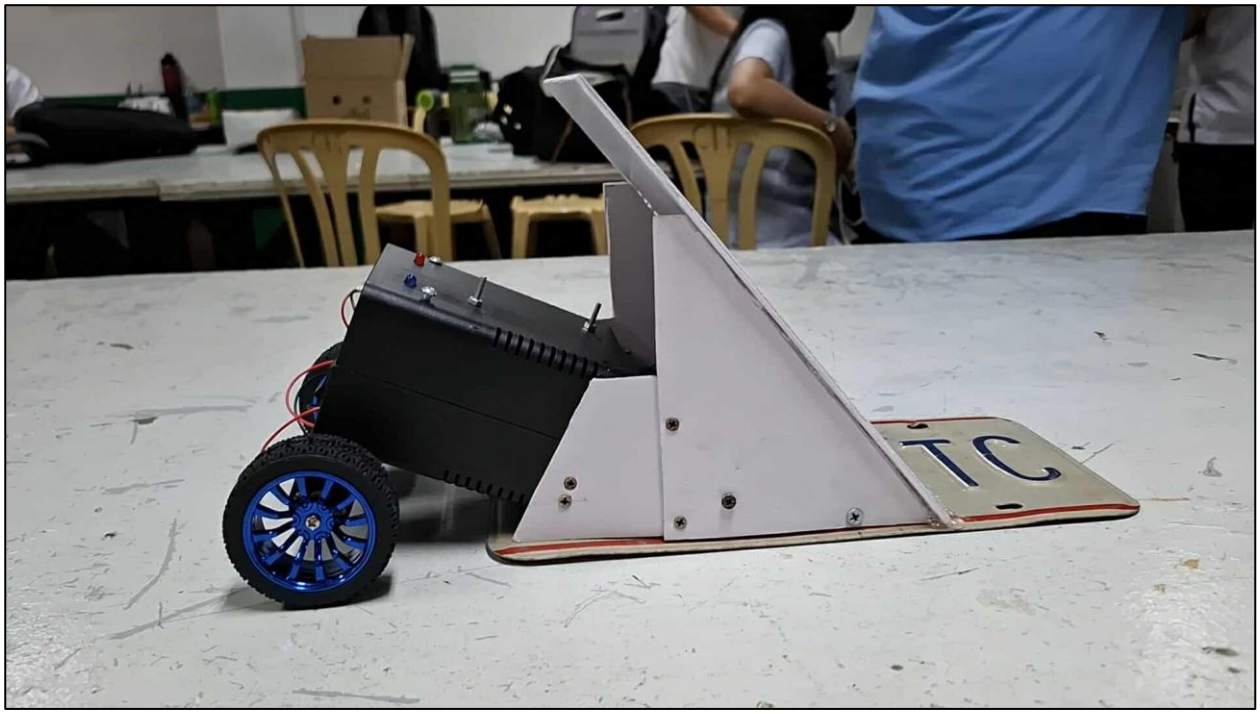

FIGURE 3

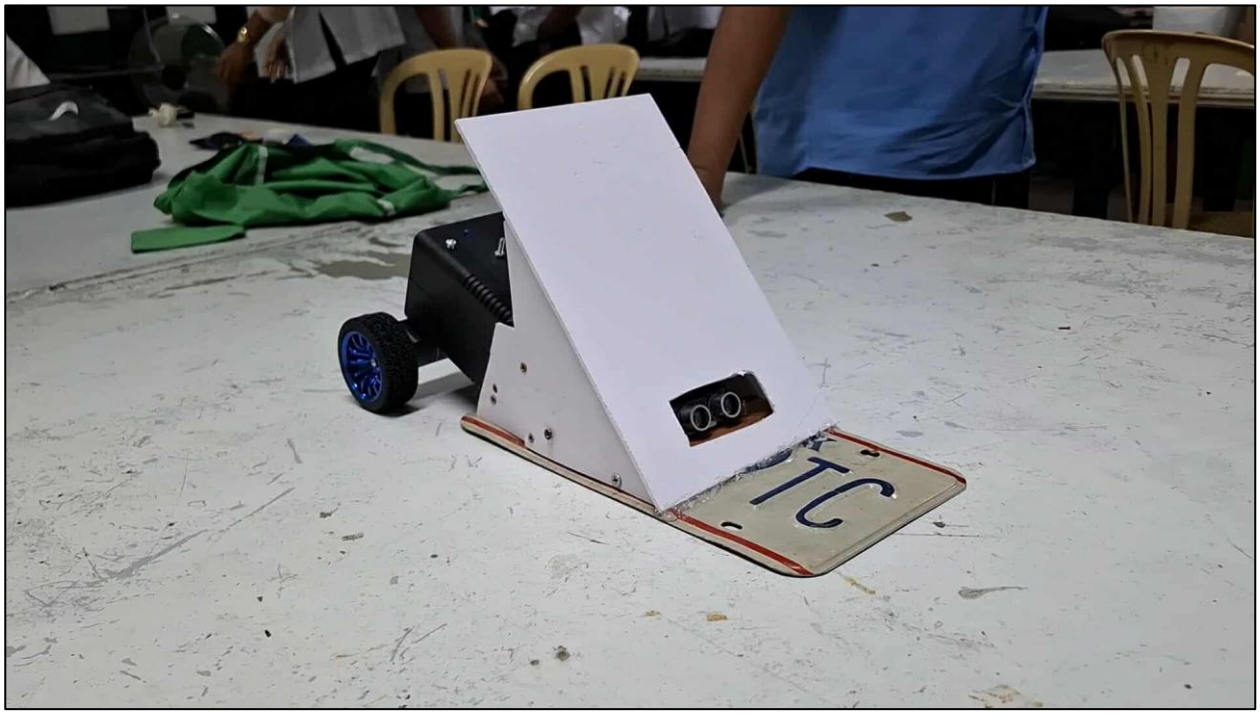

FIGURE 3.1

D. Source Code

```
#define CUSTOM_SETTINGS
#define INCLUDE_GAMEPAD_MODULE
#include <DabbleESP32.h>
#include <BluetoothSerial.h>

// Ultrasonic
#define trigPin 12
#define echoPin 14

// Right Motor
#define enableRightMotor 32
#define rightMotorPin1 26
#define rightMotorPin2 27

// Left Motor
#define enableLeftMotor 4
#define leftMotorPin1 25
#define leftMotorPin2 33

// Buttons
#define buttonAuto 18
#define buttonManual 13

// LEDs
#define ledAuto 21
#define ledManual 22

bool autoMode = false;
bool manualMode = false;

long duration;
int distance;

void setup() {
 Serial.begin(115200);

 // Motor pins
 pinMode(enableRightMotor, OUTPUT);
 pinMode(rightMotorPin1, OUTPUT);
 pinMode(rightMotorPin2, OUTPUT);
 pinMode(enableLeftMotor, OUTPUT);
 pinMode(leftMotorPin1, OUTPUT);
 pinMode(leftMotorPin2, OUTPUT);

 // Ultrasonic pins
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);

 // Buttons
 pinMode(buttonAuto, INPUT_PULLUP);
 pinMode(buttonManual,
INPUT_PULLUP);

 // LEDs
 pinMode(ledAuto, OUTPUT);
 pinMode(ledManual, OUTPUT);

 // Bluetooth
 Dabble.begin("SumoBot_Manual");

 Serial.println(" SumoBot Ready! Use
buttons to select mode.");
}

void rotateMotor(int rightMotorSpeed, int
leftMotorSpeed) {
 digitalWrite(rightMotorPin1,
rightMotorSpeed > 0);
 digitalWrite(rightMotorPin2,
rightMotorSpeed < 0);
 digitalWrite(leftMotorPin1,
leftMotorSpeed > 0);
 digitalWrite(leftMotorPin2,
leftMotorSpeed < 0);

 analogWrite(enableRightMotor,
abs(rightMotorSpeed));
 analogWrite(enableLeftMotor,
abs(leftMotorSpeed));
}

int getDistance() {
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
```

```
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  return distance;
}

void runAutomaticMode() {
  int dist = getDistance();
  Serial.print("Distance: ");
  Serial.println(dist);

  if (dist <= 62 && dist > 0) {
    rotateMotor(200, 200); // Move forward
  } else {
    rotateMotor(0, 0); // Stop
  }
}

void runManualMode() {
  Dabble.processInput();

  int rightMotorSpeed = 0;
  int leftMotorSpeed = 0;

  if (GamePad.isUpPressed()) {
    rightMotorSpeed = 255;
    leftMotorSpeed = 255;
  } else if (GamePad.isDownPressed()) {
    rightMotorSpeed = -255;
    leftMotorSpeed = -255;
  } else if (GamePad.isLeftPressed()) {
    rightMotorSpeed = 255;
    leftMotorSpeed = -255;
  } else if (GamePad.isRightPressed()) {
    rightMotorSpeed = -255;
    leftMotorSpeed = 255;
  } else {
    rightMotorSpeed = 0;
    leftMotorSpeed = 0;
  }
```

```
  rotateMotor(rightMotorSpeed,
leftMotorSpeed);
}

void loop() {
   Dabble.processInput();
  // Read buttons (LOW = pressed)
  bool buttonAutoState =
!digitalRead(buttonAuto);
  bool buttonManualState =
!digitalRead(buttonManual);

  // Mode switching
  if (buttonAutoState) {
    autoMode = true;
    manualMode = false;
    digitalWrite(ledAuto, HIGH);
    digitalWrite(ledManual, LOW);
    Serial.println("Auto Mode Activated");
    delay(500);
  } else if (buttonManualState) {
    manualMode = true;
    autoMode = false;
    digitalWrite(ledAuto, LOW);
    digitalWrite(ledManual, HIGH);
    Serial.println("Manual Mode Activated");
    delay(500);
  }

  // Run active mode
  if (autoMode && !manualMode) {
    runAutomaticMode();
  } else if (manualMode && !autoMode) {
    runManualMode();
  } else {
    rotateMotor(0, 0);
  }
}
```

E. Dabble Application

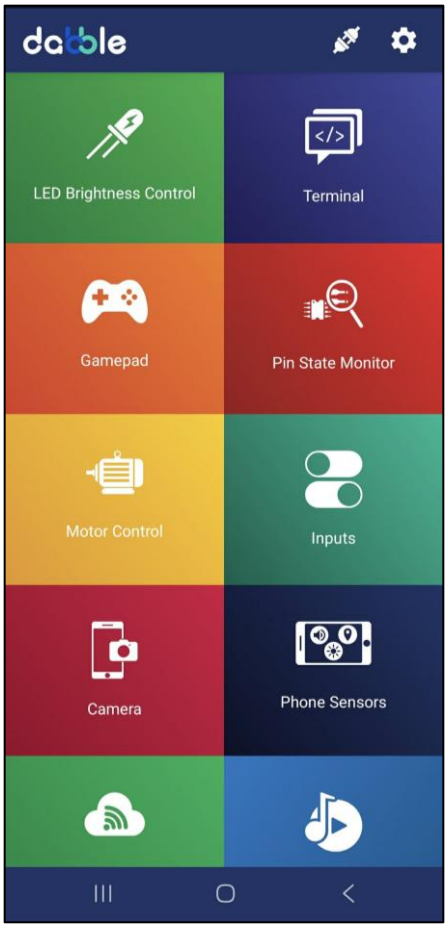

FIGURE 4

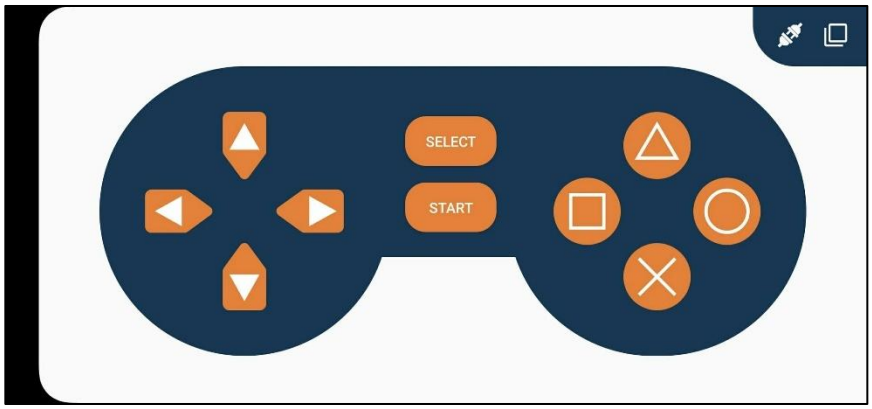

FIGURE 4.1

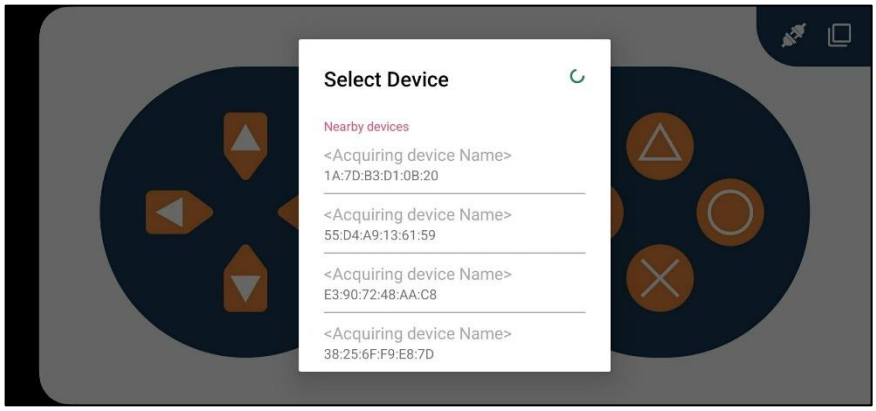

FIGURE 4.2

IV. PROCEDURE

A. Preparation

1. **Plan Your Sumo Bot**
   - Decide what size of sumo bot you will make (small, medium, or big). In our case the requirement needs to be a maximum of 2kgs.
   - Think about how it will move, usually with two wheels for balance.
   - Choose what brain (controller) you will use like an Arduino board. In this project we will be using ESP32.
   - Make a simple sketch of your robot design.

2. **Gather All the Needed Parts**
   - Please refer to section 2 of this document.

3. **Check Each Part Before Building**

   Before putting everything together, test each component one by one:

   Battery:

   - Use a multimeter to check voltage.
   - Example: A three 3.7V battery that we have used should read around 11–12 volts.
   - Make sure the battery is not hot, swollen, or damaged.
   - Charge it fully before use.

   Motors:

   - Take one motor and connect the two wires directly to the positive and negative side of the battery for a few seconds.
   - The motor should spin smoothly.
   - Reverse the wires — the motor should spin in the opposite direction.
   - Do the same for the second motor.

Motor Driver:

- You can test if your motor driver works using just the battery and a 5V power source.

Example: Testing L298N motor driver

Connections:

- VCC / +12V → Battery positive (+)
- GND → Battery negative (-)
- Connect a motor to OUT1 and OUT2 terminals.
- IN1 and IN2 are the control pins — you can tap them manually with a 5V wire from the battery or a USB 5V source.

How to test:

- Tap IN1 to 5V, leave IN2 unconnected or at GND → motor spins one way.
- Tap IN2 to 5V, leave IN1 unconnected or at GND → motor spins the other way.
- If the motor spins when you tap and stop when released → the motor driver is working.

Tips:

- Do not tap both IN1 and IN2 to 5V at the same time.
- Just quick taps (1–2 seconds) are enough.
- Check the driver is not getting too hot.

Sensors (Ultrasonic Sensor):

- Connect VCC → 5V, GND → GND, Trig → Arduino pin 9, Echo → pin 10.
- Upload a simple distance-measuring code (from Arduino examples).
- Open Serial Monitor — you should see the distance numbers change when you move your hand closer/farther.

Wires and Connections:

- Check that there are no broken wires.
- Label each wire for easy connection later.

4. Final Check Before Assembly

- Make sure all parts are working and complete.
- Check your battery is charged and safe to use.
- Review your plan or sketch one last time before starting to build.

B. Actual

Step 1: Layout all the components as shown on Section III and connect them according to the pictorial diagram/block or one could refer to the following table for the pin and power supply connections.

Power Supply and Power Switch Connections

Component	Power Supply and Power Switch Polarity
SPDT Switch - Common	Positive (Anode)
SPDT Switch - NC	Negative (Cathode)
LS298N Motor Driver VS (12V)	SPDT Switch - NO
LS298N Motor Driver GND	SPDT Switch - NC
ESP32 Shield Board - External Power Input (Barrel Jack)	SPDT Switch - NO

L298N Motor Driver - Motor Connections

Components	Motor Driver Ports
DC Motor 1 - Positive (Anode)	OUT1
DC Motor 1 – Negative (Cathode)	OUT2
DC Motor 2 – Positive (Anode)	OUT3
DC Motor 2 – Negative (Cathode)	OUT4

ESP32 Connections

Component Pins	ESP32 GPIO Pins
L298N Motor Driver - ENA	GPIO4
L298N Motor Driver – IN1	GPIO25
L298N Motor Driver – IN2	GPIO33
L298N Motor Driver – IN3	GPIO26
L298N Motor Driver – IN4	GPIO27
L298N Motor Driver - ENB	GPIO32

HC-SR04 Ultrasonic Sensor - VCC	5v VCC Out
HC-SR04 Ultrasonic Sensor – GND	Ground (GND) Pin
HC-SR04 Ultrasonic Sensor – Trig	GPIO12
HC-SR04 Ultrasonic Sensor – Echo	GPIO14
Red LED – Positive (Anode)	GPIO21
Red LED – Negative (Cathode)	Ground (GND) Pin
Red LED – Positive (Anode)	GPIO22
Blue LED – Negative (Cathode)	Ground (GND) Pin
Push Button 1 – A	GPIO18
Push Button 1 - C	Ground (GND) Pin
Push Button 2 – A	GPIO13
Push Button 2 - C	Ground (GND) Pin

Step 2: Upload the program to the ESP32. To check if both the program and hardware is working, please refer to the following section.

Step 3: Download the "Dabble" mobile application from Aptoide or another third-party APK installer for android phones (**https://dabble.en.aptoide.com/app**), for iPhone, you can install it directly from the app store.

Step 4: Open Dabble mobile application then select "Gamepad" as shown in Figure 4. You should see something like in Figure 4.1. Then press the connect button and pair with your ESP32 as shown in Figure 4.2.

Step 5: Test the controls and you're good to go.

C. Checking

After connecting all necessary components to the microcontroller and uploading the program, verify that each component functions correctly by following these steps:

Motor Functionality:

- Ensure both motors respond properly to the controller:
- Forward Button: Both motors should rotate forward.
- Backward Button: Both motors should rotate backward.
- Right Button: The right motor should rotate backward, while the left motor rotates forward.
- Left Button: The left motor should rotate backward, while the right motor rotates forward.

Wheels:

- Check if the wheels are securely attached. Tighten the screws if they are loose.

Ultrasonic Sensor:

- Verify that the ultrasonic sensor can detect and respond to objects within the programmed distance:
- When an object is detected, both motors should rotate forward.
- When no object is within the programmed distance, both motors should stop rotating.

Buttons (Modes):

- Check if the buttons function properly when pressed.
- When you press the button near the ramp, the automatic mode (Ultrasonic Sensor) should activate, and the corresponding LED indicator should light up.
- When you press the button located behind the first button, the manual mode (controlled via the Dabble app) should activate. The sumo bot should respond to commands from the app, and the corresponding LED indicator for this mode should also light up.

D. Uploading

Upload the final code. Once uploaded, check the behavior of the overall Sumo-Bot if its functionality correlates with the algorithm of the program. Make sure to test both Auto-mode and Manual-mode to properly examine its performance and to have a grasp of its control through the app. If the output is emitting unexpected results, it is suggested to initially check the hardware components—mainly for loose wires or improper soldered components before heading towards the software to check for bugs in the program.

V. CONCLUSION

The project, Atom the recycabot, was a complete success. It not only fulfilled its purpose as an effective sumo bot but also achieved this by being constructed primarily from recycled materials. This approach demonstrated great resourcefulness, compromising with available components to avoid excessive spending. The ESP32 successfully demonstrated its robust capability to manage the complex, real-time autonomous robot. Each subsystem showcased a critical robotics concept: detecting opponents with ultrasonic sensors, avoiding boundaries with reflectance sensors, and implementing precise movement with a motor driver. The integration of these components, powered by the ESP32's fast processing, highlighted how to fuse sensor data into an effective, autonomous combat strategy. This project deepened our understanding of sensor interfacing, motor control, and the development of cost-effective algorithms for competitive robotics.

VI. PHOTO DOCUMENTATION