LEBANESE UNIVERSITY

DEPARTMENT ELECTRICAL AND ELECTRONIC ENGINEERING

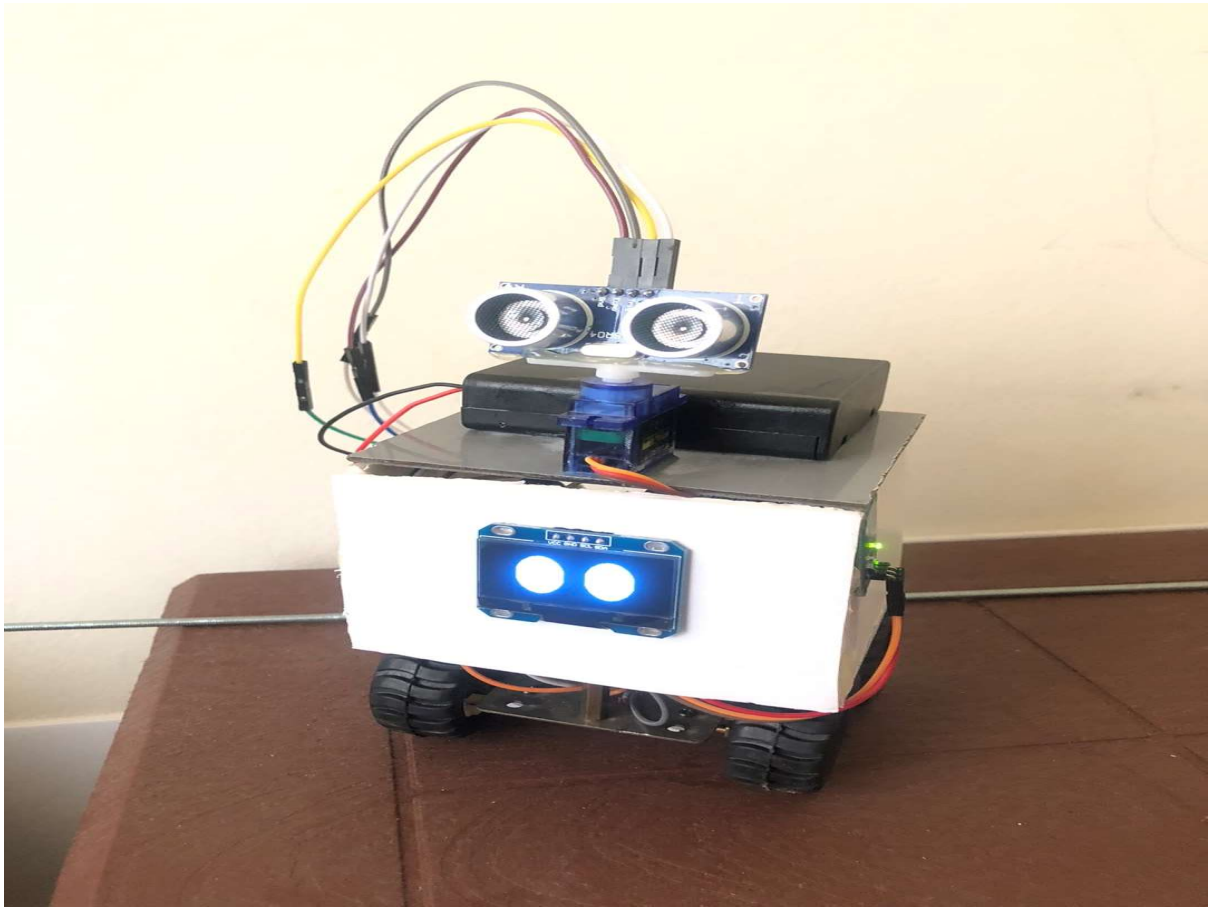
TELECOMMUNICATION

ROBOT PROJECT REPORT

Mini-Projet – CTE

NAME:ABDEL AZIZ MOHAMAD / TAREK ISMAIL

Proton: Arduino Obstacle Avoidance Robot Controlled by Smartphone



INTRODUCTION

The Proton project is an Arduino-based robot designed to avoid obstacles and be controlled via a smartphone using the HC-05 Bluetooth module. This project explores the integration of various electronic components and sensors to create an autonomous vehicle capable of navigating its environment with minimal human intervention. The goal is to develop a robot that not only avoids obstacles but can also be directed through a smartphone app, making it an excellent project for learning about robotics, electronics, and wireless communication.

Overview

The primary objectives of the Proton project are:

- To design and build a robot capable of avoiding obstacles autonomously.
- To enable remote control of the robot via a smartphone using the HC-05 Bluetooth module.
- To integrate various electronic components, including sensors, motor drivers, and displays, to achieve the desired functionality.

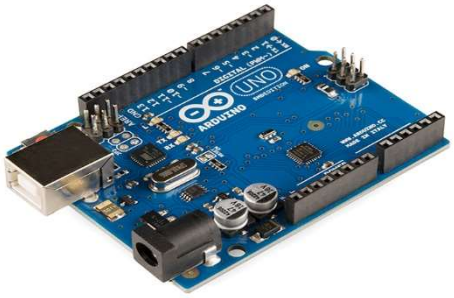
Project Scope

The project encompasses the following:

- Selection and integration of appropriate components.
- Designing the wiring and assembly of the robot.
- Developing the software necessary to control the robot and process sensor data.
- Testing the robot's performance and troubleshooting any issues encountered.

COMPONENTS USED

- Arduino Uno

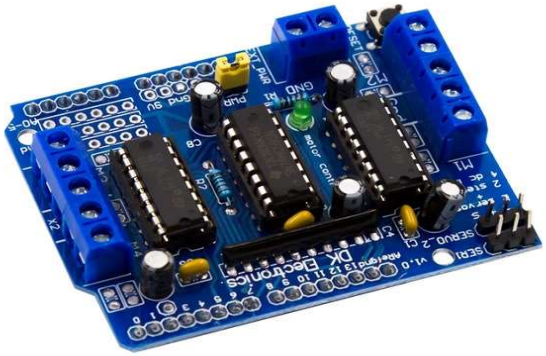


The Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins, 6 analog inputs, a USB connection, a power jack, and a reset button. It is the brain of the Proton robot, handling all processing and control functions.

- OLED 128x128 display

The OLED 128x128 I2C display is used to show the eyes of the robot. It communicates with the Arduino using the I2C protocol, providing a simple interface for displaying graphics.

- L293D Motor Driver Shield



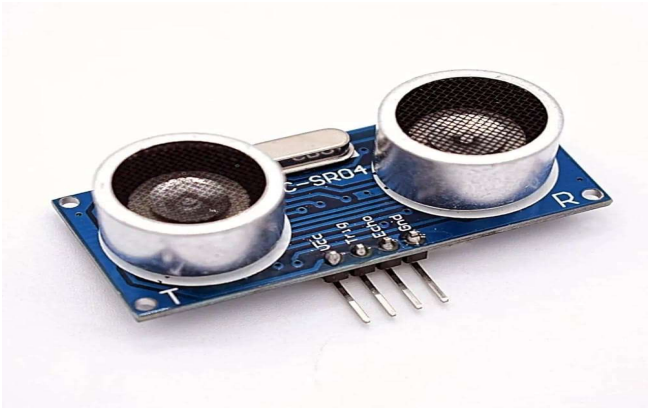
The L293D Motor Driver Shield allows the Arduino to control the motors of the robot. It can drive up to four DC motors, making it ideal for the four-wheel chassis of the Proton robot.

There's a drawback in this motor driver which is this driver is a power consumer.

- Servo Motor

A servo motor is used to position the ultrasonic sensor, enabling it to look right and left. This allows the robot to detect obstacles in different directions.

- Ultrasonic Sensor



The ultrasonic sensor measures the distance to obstacles in the robot's path. It sends out ultrasonic waves and measures the time it takes for the waves to bounce back, providing accurate distance measurements.

- Battery Holder with Switch

The battery holder provides power to the robot. It includes a switch for easily turning the robot on and off.

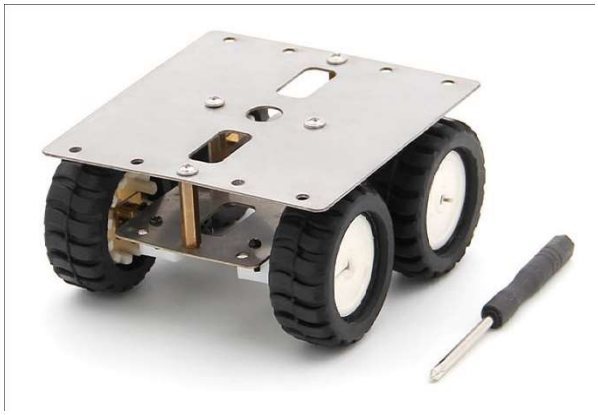
- Batteries

Four AA batteries are used to power the robot, provides 6v.

- HC-05 Bluetooth Module

The HC-05 Bluetooth module enables wireless communication between the robot and a smartphone. It allows the user to send commands to the robot remotely.

- Car Chassis



The car chassis consists of four wheels with four N20 geared motors. It provides the structural base and mobility for the Proton robot.

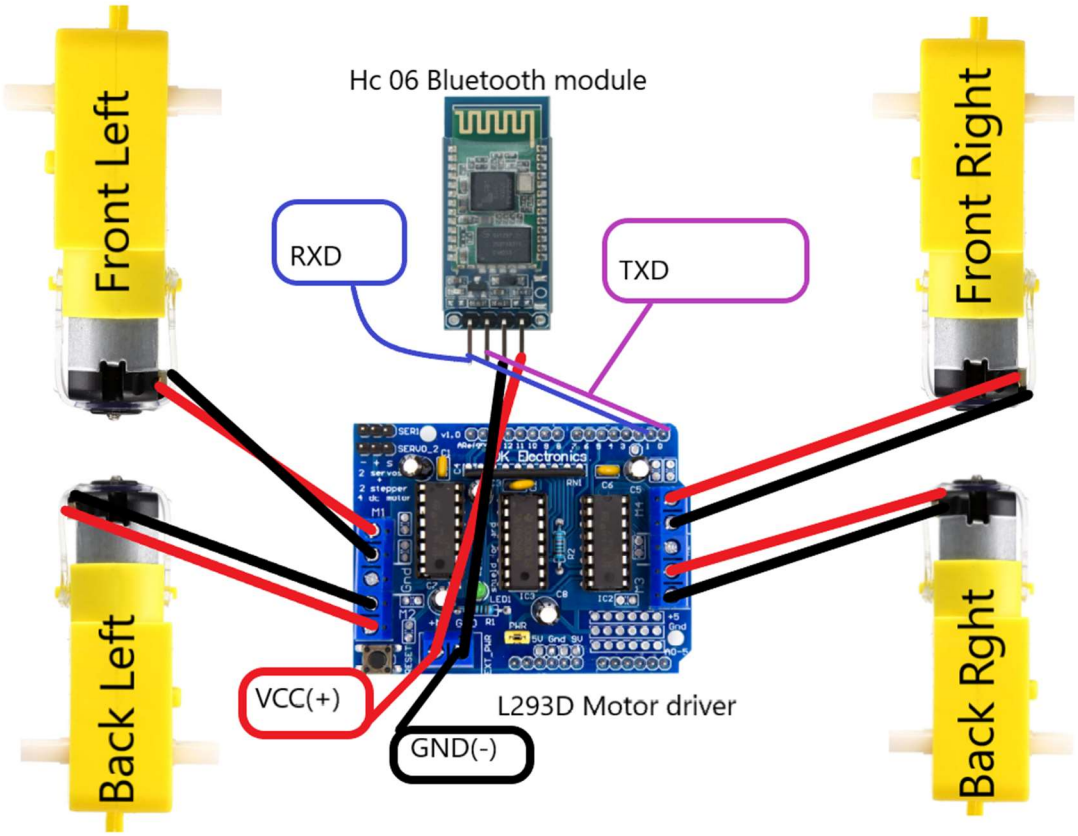
WIRING AND ASSEMBLY

In this section, we will discuss the detailed wiring and assembly process of our Arduino-based obstacle avoidance robot, Proton. We collaborated closely to wire the robot, addressing various challenges and solving them together. The following steps outline our journey in assembling the robot and connecting its components.

Motor Shield Setup

The first step in our assembly was to attach the L293D motor driver shield to the Arduino Uno. The motor shield simplifies the process of controlling the robot's motors and provides convenient pin access for other components. By stacking the shield on top of the Arduino, we efficiently utilized the available space, leaving room for additional connections.

Look at the figure below to see how we started by placing the motor shield on the Arduino and soldering the TX and RX pins of the HC-05 module to the RX and TX pins of the motor shield.



HC-05 Bluetooth Module

Connecting the HC-05 Bluetooth module was our next task. This module allows us to control the robot wirelessly via a smartphone.

Wiring:

- HC-05 RX → Arduino TX
- HC-05 TX → Arduino RX
- HC-05 VCC → Arduino 5V
- HC-05 GND → Arduino GND

During the wiring process, we faced a challenge with the Bluetooth module's connectivity. The initial connections were unstable, causing intermittent communication drops. After troubleshooting, we realized the issue was due to loose solder joints. We re-soldered the connections, ensuring a secure and stable link.

OLED display

Next, we connected the OLED 128x128 I2C display, which displays the robot’s "eyes".

Wiring:

- OLED VCC → Arduino 5V
- OLED GND → Arduino GND
- OLED SDA → Arduino A4
- OLED SCL → Arduino A5

Integrating the OLED display was straightforward, but we encountered a minor issue with the I2C address. The display did not show any output initially. After some research, we discovered that the default I2C address needed to be modified in the code to match our display's address. Updating the code resolved the issue.

Ultrasonic Sensor

The ultrasonic sensor is a crucial component for obstacle detection. We mounted it on a servo motor to allow it to scan the surroundings.

Wiring:

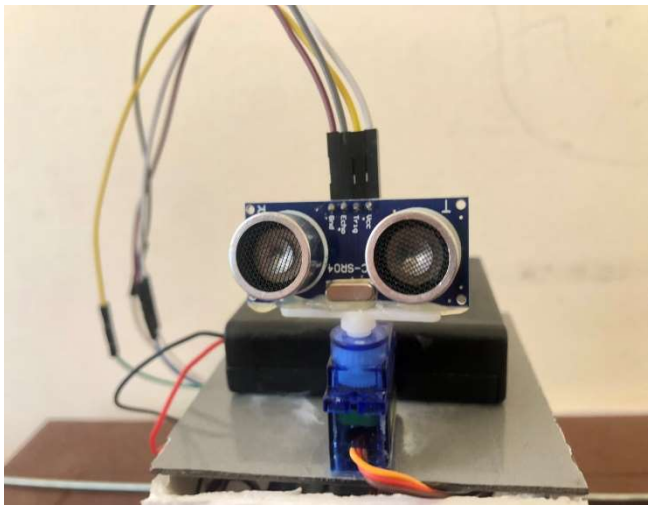
- Ultrasonic Sensor VCC → Arduino 5V
- Ultrasonic Sensor GND → Arduino GND
- Ultrasonic Sensor TRIG → Arduino A0
- Ultrasonic Sensor ECHO → Arduino A1

Servo Motor

The servo motor is controlled by the motor shield. Connecting the servo motor to the motor shield was seamless.

Wiring:

- Servo Motor → Motor Shield



Power Supply

The robot is powered by four AA batteries connected to the motor shield. These batteries simultaneously power the motor driver and the Arduino, which is a powerful setup. However, this configuration does have a drawback: the robot runs a bit slowly due to the relatively low power supply. We decided to make a tradeoff by using a single power supply for simplicity and convenience.

Wiring:

- 4xAA Batteries → Motor Shield VIN and GND

Touch Sensor

Although the touch sensor did not function as expected, we still wired it to the Arduino for potential future use.

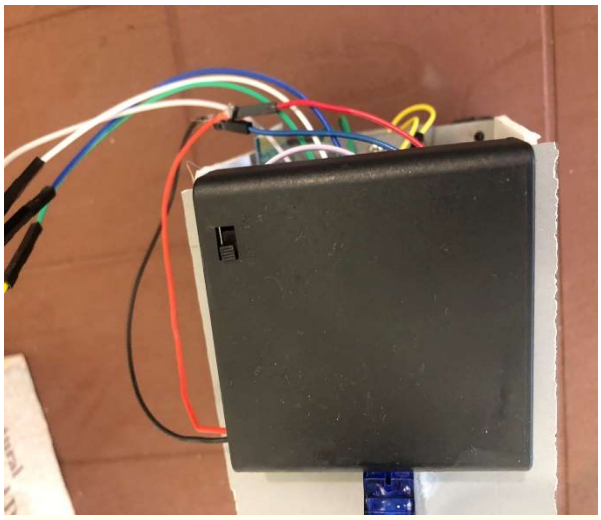
Wiring:

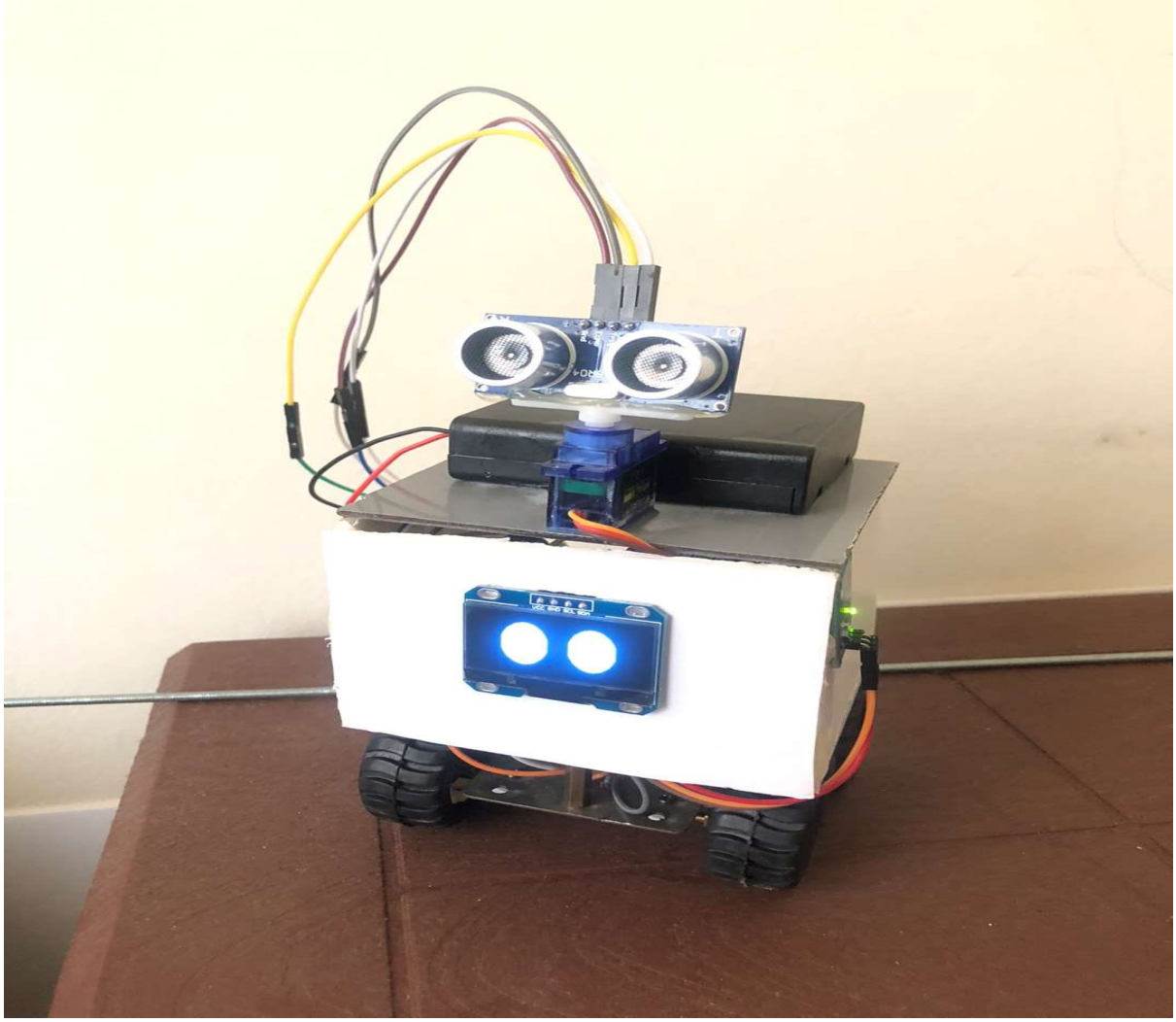
- Touch Sensor VCC → Arduino 5V
- Touch Sensor GND → Arduino GND
- Touch Sensor OUT → Arduino A2

We attempted to debug the touch sensor by checking the connections and reviewing the code, but it remained unresponsive.

Final Assembly

After connecting all the components, we carefully arranged the wiring to avoid any interference and ensure a clean assembly. This step required meticulous attention to detail to prevent short circuits and maintain a tidy setup.





CODE AND IMPLEMENTATION

In this section, we will delve into the code and implementation of the Arduino-based obstacle avoidance robot, Proton. The code is written to control the robot's movements, handle obstacle detection, and communicate with a smartphone via Bluetooth. Below is a detailed explanation of the key components and functions within the code.

Libraries and Definitions

To begin with, we included several libraries essential for the robot's operation:

- **AFMotor.h**: This library is used to control the motors via the motor driver shield.
- **NewPing.h**: This library is for interfacing with the ultrasonic sensor.
- **Servo.h**: This library allows us to control the servo motor.
- **ScreenImages.h**: This header file contains image arrays and functions to animate the eyes on the OLED display (We will talk more about the header file later on).

```
#include <AFMotor.h>
#include <NewPing.h>
#include <Servo.h>
#include "ScreenImages.h"

#define TRIG_PIN A0
#define ECHO_PIN A1
#define MAX_DISTANCE 200
#define MAX_SPEED 190 // sets speed of DC motors
#define MAX_SPEED_OFFSET 20
```

Initial Setup

In the setup function, we initialize serial communication, attach the servo motor, set initial motor speeds, and initialize the OLED display. The ultrasonic sensor's distance measurement is also initialized.

```
void setup() {
Serial.begin(9600);
myservo.attach(10);
myservo.write(115);
motor1.setSpeed(0);
motor1.run(RELEASE);
motor2.setSpeed(0);
motor2.run(RELEASE);
motor3.setSpeed(0);
motor3.run(RELEASE);
motor4.setSpeed(0);
motor4.run(RELEASE);
display.begin(i2c_Address, true); // Address 0x3C default
Sleep();
delay(2000);
distance = readPing();
delay(100);
distance = readPing();
delay(100);
distance = readPing();
delay(100);
distance = readPing();
delay(100);
}
```

Main Loop

The loop function is where the robot's core functionality is implemented. The robot reads commands from the serial port to move forward, backward, turn left, right, or stop. Additionally, it processes obstacle detection and avoidance.

```
void loop() {
  if (Serial.available() > 0) { // Checks whether data is coming from the serial port
    state = Serial.read(); // Reads the data from the serial port
    if (state == 'f') {
      isMoving = true;
    } else if (state == 'b') {
      moveBackward();
    } else if (state == 'r') {
      turnRight();
    }else if (state == 'l') {
      turnLeft();
    } else if (state == 'w') {
      WakeUp();
    }else if (state == 'n') {
      Sleep();
    }else if (state == 'h') {
      Happy();
    }else if (state == 's') {
      isMoving = false;
      moveStop();
    } else if (state == 'a') {
      Angry();
    } else if (state == 'c') {
      Sad();
    }
  }
  if (isMoving) {
    int distanceR = 0;
    int distanceL =  0;
    delay(40);
    if (distance <= 15) {
      moveStop();
      delay(100);
      moveBackward();
      delay(300);
      moveStop();
      delay(200);
      distanceR = lookRight();
      delay(200);
      distanceL = lookLeft();
      delay(200);

      if (distanceR >= distanceL) {
        turnRight();
        moveStop();
      } else {
        turnLeft();
        moveStop();
      }
    } else {
      moveForward();
    }
    distance = readPing();
  }
}
```

Functions for Movement and Obstacle Detection

The code includes several functions to control the robot's movement and handle obstacle detection using the ultrasonic sensor and servo motor. lookRight() and lookLeft(): These functions rotate the servo motor to scan the right and left directions and measure distances using the ultrasonic sensor.

readPing(): This function measures the distance to an obstacle using the ultrasonic sensor.

Movement Functions: These functions control the robot's motors to move forward, backward, turn right, turn left, and stop.

```
int lookRight() {
  myservo.write(50);
  delay(500);
  int distance = readPing();
  delay(100);
  myservo.write(115);
  return distance;
}

int lookLeft() {
  myservo.write(170);
  delay(500);
  int distance = readPing();
  delay(100);
  myservo.write(115);
  return distance;
  delay(100);
}
```

```
int readPing() {
  delay(70);
  int cm = sonar.ping_cm();
  if (cm == 0) {
    cm = 250;
  }
  return cm;
}
```

```
void moveStop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}
```

```
void moveForward() {
  if (!goesForward) {
    goesForward = true;
    motor1.run(FORWARD);
    motor2.run(FORWARD);
    motor3.run(FORWARD);
    motor4.run(FORWARD);
    for (speedSet = 0; speedSet < MAX_SPEED; speedSet += 2) {
      motor1.setSpeed(speedSet);
      motor2.setSpeed(speedSet);
      motor3.setSpeed(speedSet);
      motor4.setSpeed(speedSet);
      delay(5);
    } }}
```

```
void moveBackward() {
  goesForward = false;
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  for (speedSet = 0; speedSet < MAX_SPEED; speedSet += 2) {
    motor1.setSpeed(speedSet);
    motor2.setSpeed(speedSet);
    motor3.setSpeed(speedSet);
    motor4.setSpeed(speedSet);
    delay(5);
  }
}
```

```
void turnRight() {
  motor3.run(FORWARD);
  motor2.run(FORWARD);
  motor1.run(BACKWARD);
  motor4.run(BACKWARD);
  delay(550);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
```

```
void turnLeft() {
  motor3.run(BACKWARD);
  motor2.run(BACKWARD);
  motor1.run(FORWARD);
  motor4.run(FORWARD);
  delay(550);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
```

Animation Functions:

The robot's OLED display shows various animations (happy, sad, angry, etc.) which are implemented in functions such as Happy(), Sad(), Angry(), etc., using the images defined in the "ScreenImages.h" file. These functions are called based on the commands received via Bluetooth.

APP DEVELOPMENT WITH MIT APP INVENTOR

Let’s describe how we developed the mobile application for controlling Proton using MIT App Inventor. This application allows us to send commands to the robot via the HC-05 Bluetooth module, enabling various movements and animations.

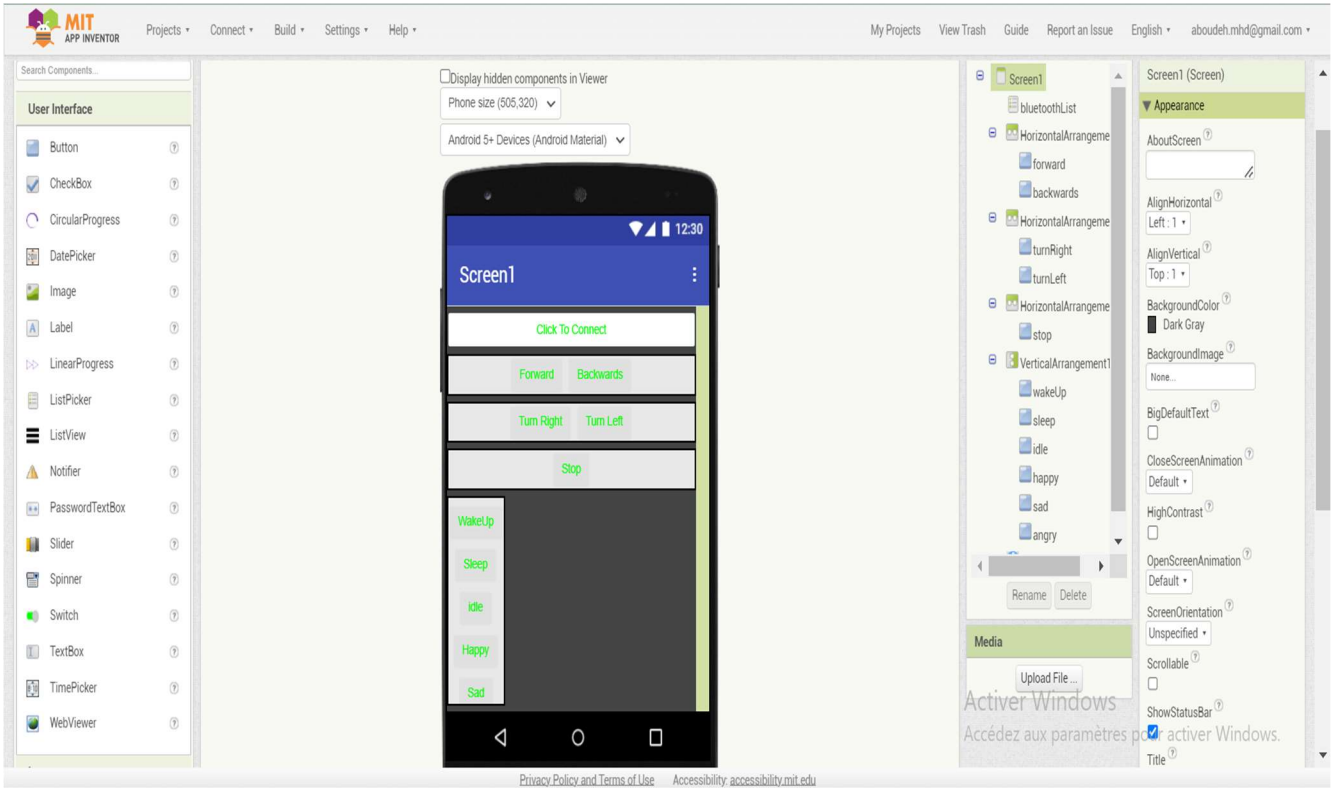
Introduction to MIT App Inventor

MIT App Inventor is a user-friendly platform that allows anyone to create fully functional apps for smartphones and tablets. It uses a block-based programming language that simplifies app development, making it accessible even to those without extensive coding experience.

Creating the App Interface

We started by designing the interface of our app. The app includes buttons to control the robot's movements and animations. Here's a step-by-step overview of the process:

1. Setting Up MIT App Inventor:
   1.1. We registered an account on the MIT App Inventor website.
   1.2. Created a new project for the Proton robot.
2. Designing the User Interface:
   2.1. We dragged and dropped various components from the palette onto the viewer to design the app's layout.
   2.2. The interface includes buttons for forwarding, backward, turn right, turn left, stop, sleep, wake up, happy, sad, and angry commands.



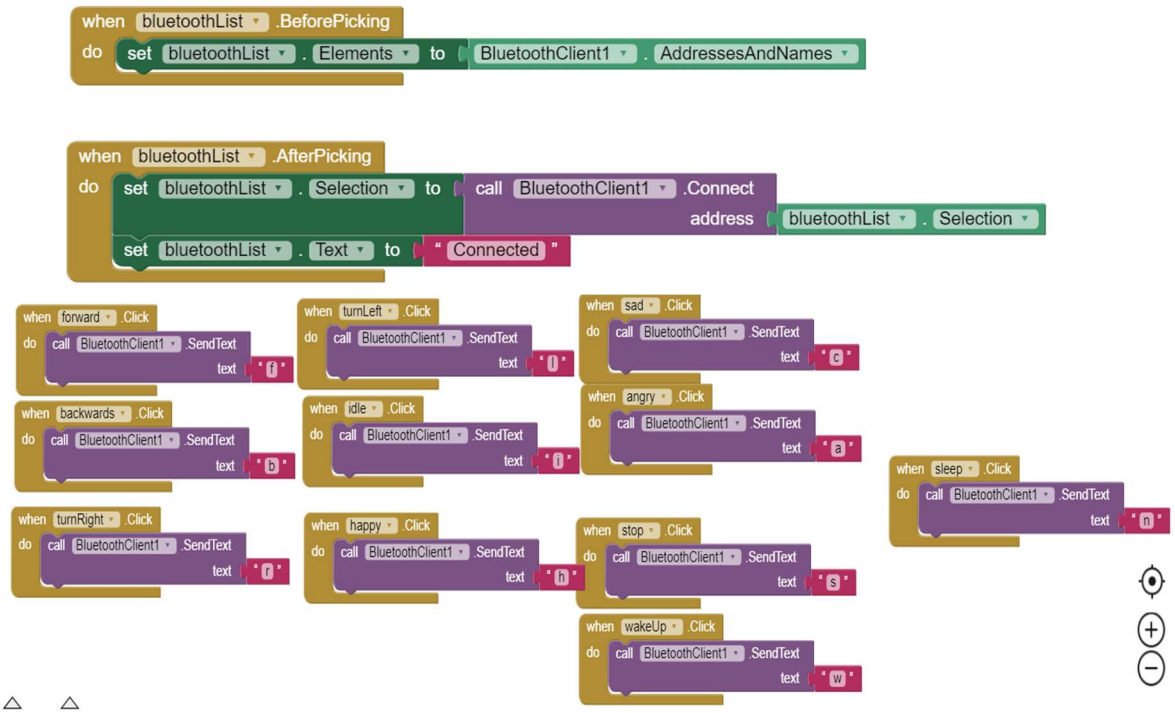
List of Buttons and their Functions:

- Forward: Moves the robot forward.
- Backward: Moves the robot backward.
- Turn Right: Turns the robot to the right.
- Turn Left: Turns the robot to the left.
- Sleep: Puts the robot in sleep mode.
- Wake Up: Wakes the robot up from sleep mode.
- Happy: Displays a happy animation on the robot's OLED screen.
- Sad: Displays a sad animation on the robot's OLED screen.
- Angry: Displays an angry animation on the robot's OLED screen.

Programming The App

After designing the interface, we programmed the buttons to send corresponding commands to the robot via Bluetooth.

1. Adding Bluetooth Connectivity:
   - We included the Bluetooth Client component to manage the Bluetooth connection between the app and the robot.
2. Setting Up Button Functions:
   - We used the Blocks Editor in MIT App Inventor to define the behavior of each button.
   - Each button sends a specific character to the robot when pressed. For example, the forward button sends the character 'f', the backward button sends 'b', and so on.



TESTING

we will discuss the testing and troubleshooting process that Tarek and I went through to ensure the smooth operation of our Arduino-based obstacle avoidance robot, Proton. This phase was crucial to identify and resolve any issues, fine-tune the robot's performance, and ensure its reliability.

Initial Testing

After completing the wiring and coding, our first step was to power up the robot and perform initial tests. We aimed to verify that all components were functioning correctly and the robot could follow basic commands.

Power and Connectivity Check:

- We began by turning on the power switch to ensure the robot received adequate power from the 4xAA batteries.
- We then checked the Bluetooth connection by pairing the HC-05 module with a smartphone and sending basic movement commands through a Bluetooth terminal app.

Initial Observations:

- Initial Observations: The motors responded to the commands, indicating that the motor shield and motors were correctly wired.
- The OLED display powered on and showed the initial animation, confirming the display was functioning.

Obstacle Detection and Avoidance Testing

Next, we focused on testing the robot's ability to detect and avoid obstacles using the ultrasonic sensor and servo motor.

Ultrasonic Sensor Check:

We used the serial monitor to print distance readings from the ultrasonic sensor to ensure it accurately measured distances. Initially, the readings were inconsistent. After rechecking the connections and adjusting the sensor's position, the readings stabilized.

Movement and Avoidance: We placed objects at varying distances in front of the robot to test its obstacle detection. The robot successfully detected obstacles and made decisions to turn left or right based on the distance readings from the ultrasonic sensor.

Bluetooth Control Testing

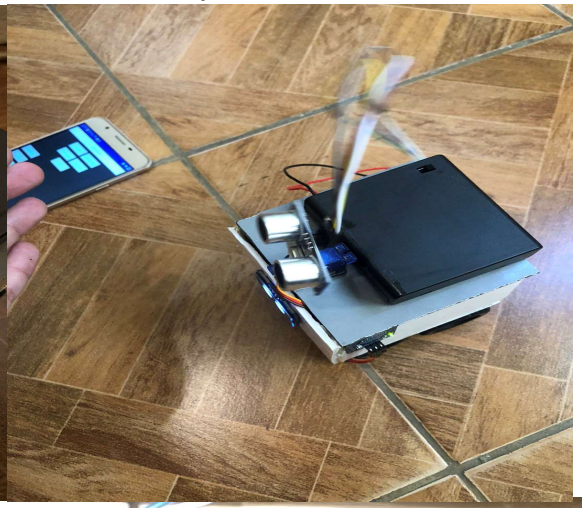
Testing the Bluetooth control functionality involved verifying that the robot could receive and execute commands sent from a smartphone.

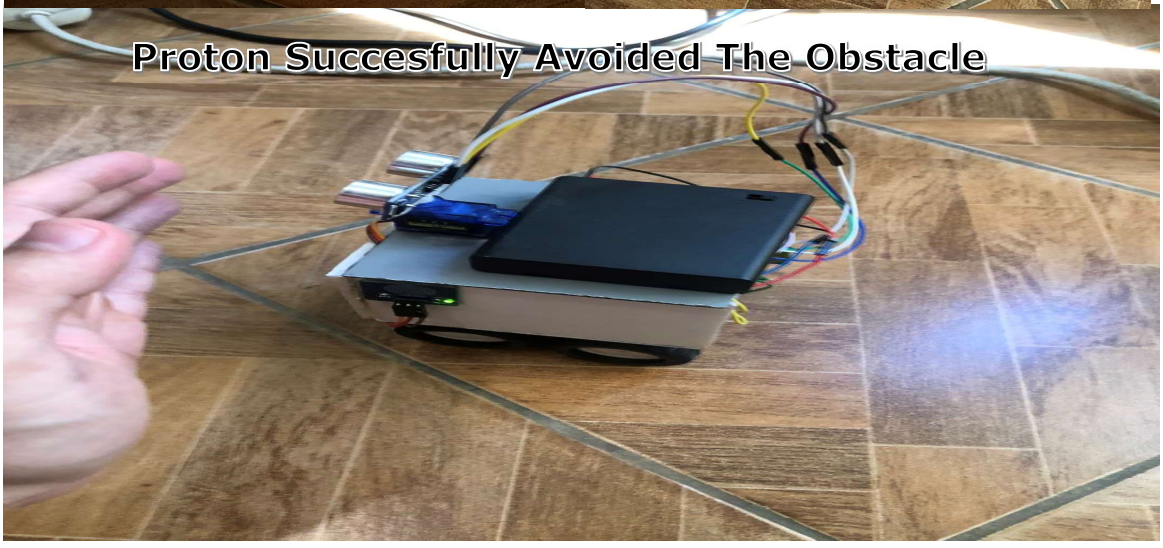
Command Verification:

We tested each command ('f' for forward, 'b' for backward, 'r' for right, 'l' for left, 's' for stop, etc.) to ensure the robot responded correctly. The robot's movements were consistent with the commands, indicating that the Bluetooth module was correctly integrated and the serial communication was reliable.

Challenges Faced

- The robot moved slower than expected due to the relatively low power from the AA batteries. We confirmed that this was a tradeoff we had accepted for simplicity.
- At times, the robot did not turn in time to avoid collisions. We realized this was due to the delay in distance readings and the servo motor's response time. We adjusted the delay intervals in the code, which improved the robot's responsiveness.
- Occasionally, the robot did not respond to commands. We identified that this issue was due to Bluetooth interference and resolved it by ensuring a clear line of sight between the smartphone and the robot.





RESULTS AND DISCUSSION

Proton, our Arduino-based obstacle avoidance robot controlled via a smartphone, has been a significant project that showcases the potential of integrating robotics with smart technology. Throughout this project, we have gained invaluable experience in electronics, programming, and problem-solving.

Positive Aspects

Obstacle Avoidance and Navigation:

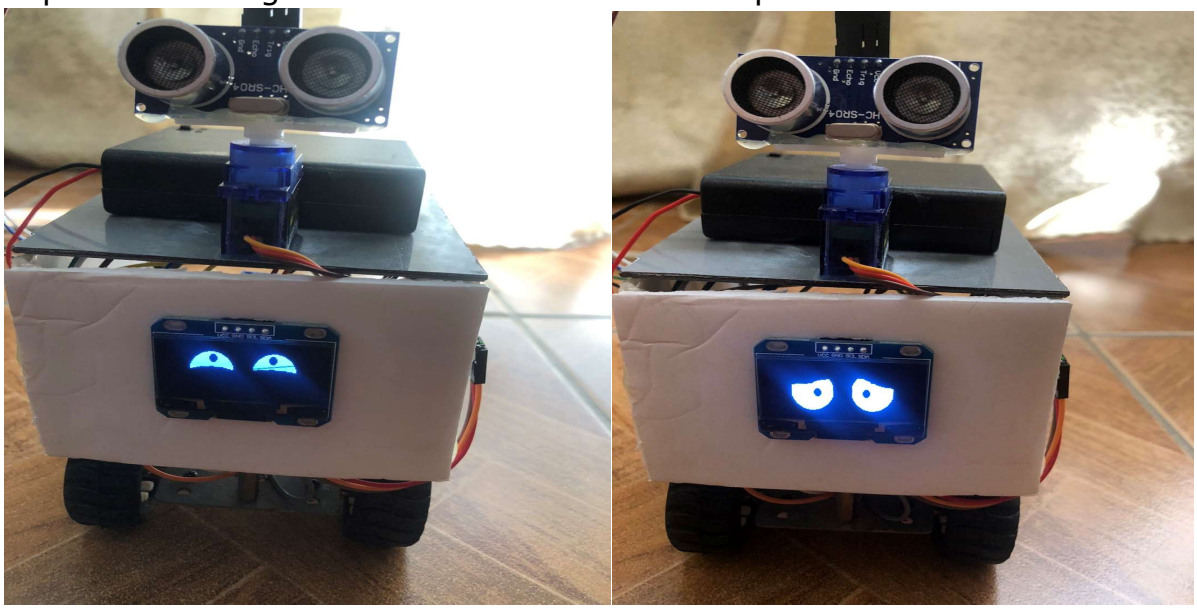
Proton successfully detects and avoids obstacles using the ultrasonic sensor and servo motor. This functionality demonstrates its potential for smart car applications, where obstacle avoidance is crucial for safety and efficiency.

Bluetooth Control:

The ability to control Proton using a smartphone app developed with MIT App Inventor adds a modern touch, making it easy to navigate and interact with the robot remotely. This feature highlights the convenience of integrating Bluetooth technology in robotics.

Visual Feedback:

The OLED display provides an interactive visual feedback system, making Proton appear more animated and engaging. This enhances the user experience and gives the robot a more lifelike presence.



Challenges and Drawbacks

Despite its strengths, Proton does have some limitations that we encountered during the project:

- Speed and Power: One of the main drawbacks is the robot's slow movement. This is due to the limited power supply from the 4xAA batteries. While it is a tradeoff for simplicity, the robot’s performance could be improved with a higher voltage battery pack.
- Incomplete Features: The touch sensor, intended to make Proton more interactive, did not function as expected. This limited the robot's ability to respond to physical touches, reducing its interactivity. Additionally, we had planned to include an accelerometer sensor to make Proton react to hits and a gyroscope to ensure precise 90-degree turns. These features would have made Proton more responsive and accurate in its movements. Unfortunately, due to time and technical constraints, we were unable to implement these components fully.

Future Improvements

Looking ahead, there are several enhancements we envision for Proton to realize its full potential:

- Improved Power Supply: Upgrading to a more powerful battery pack would increase the robot's speed and performance.
- Enhanced Interactivity: Successfully integrating the touch sensor and adding an accelerometer would make Proton more interactive and responsive to user inputs.
- Precision Movements: Incorporating a gyroscope would enable Proton to make precise turns, improving its navigation accuracy.

These improvements would make Proton not just a powerful obstacle avoidance robot, but also a more interactive and engaging smart device. Our ultimate goal is to create a versatile robot that can serve as a prototype for future smart cars and other intelligent machines.

Conclusion

In conclusion, Proton is a promising project that has taught us a great deal about robotics and smart technology. Despite some limitations, the successful implementation of obstacle avoidance, Bluetooth control, and visual feedback demonstrates its potential. We are excited about the possibilities for future enhancements and the continued development of this powerful robot.