Established – 1961 Subject: _________

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Mr./Ms. ___________________________________ of S.Y. Computer Science (SYCS) Roll No. ________________________________ has satisfactorily completed The Internet Of Thing Mini Project entitled___________________
____
during the academic year 2025 – 2026, as a part of the practical requirement. The project work is found to be satisfactory and is approved for submission.

PROF. INCHARGE **HEAD OF DEPT**

_____________ ______________

INDEX

INTRODUCTION

Smart Helmet For Accident Detection- Introduction :

Road accidents are one of the major causes of fatalities worldwide. Many accident victims fail to receive immediate medical assistance because no one is aware of the accident in time. A smart solution is required that can automatically detect accidents and notify others instantly.

The **Smart Helmet Accident Detection System** is an IoT-based safety device designed to detect sudden impacts or abnormal motion using sensors. When an accident is detected, the system activates an alarm and generates an alert. The project integrates embedded systems, sensors, and communication technologies to provide real-time accident monitoring.

- **Motivation :**

The main motivation behind this project is to enhance rider safety using modern technology. Traditional **helmets** provide physical protection but lack intelligent safety features.

Reasons for selecting this project:

- Increasing road accident rates
- Delay in emergency medical response
- Lack of automatic alert systems
- Need for affordable safety solutions
- Educational value for IoT learning

- **Problem Definition :**
- The core problem addressed by this project is the lack of immediate emergency response during road accidents involving two-wheeler riders. In many cases, accidents occur in isolated areas or during late hours, where victims may become unconscious and unable to seek help. Delays in assistance significantly increase the risk of fatal injuries.
- This project tackles challenges such as detecting sudden impacts, identifying abnormal motion patterns, and triggering an alert system automatically without requiring manual input. By integrating motion sensors, a microcontroller, and an alert mechanism, the system ensures that accidents can be detected instantly and appropriate warning signals can be generated.
- The goal is to create an intelligent safety helmet capable of monitoring rider motion and responding automatically during emergencies.

How it works :

The Smart Helmet Accident Detection System operates using an embedded microcontroller that integrates motion sensing and alert components. An MPU6050 accelerometer sensor continuously monitors acceleration and tilt along three axes (X, Y, Z). Under normal riding conditions, these values remain within a stable range.

When a sudden impact or abnormal tilt occurs, the sensor detects rapid changes in motion values. The microcontroller processes these readings and compares them with predefined threshold values. If the threshold is exceeded, the system assumes that an accident has occurred.
Once an accident is detected, the buzzer is activated immediately to produce an audible alarm.

This alert can help nearby people notice the emergency situation quickly. A manual push button is also included, allowing the rider to disable the alarm if it was triggered accidentally.

This combination of sensing, processing, and alert mechanisms enables the helmet to function as an intelligent safety system capable of real-time accident detection.

Key Features :

1.Accident Detection :
Uses motion and acceleration data to detect sudden impacts or abnormal helmet movement, enabling real-time accident identification.

2. Automatic Alert System :
Activates a buzzer instantly when an accident is detected, helping nearby people recognize emergency situations quickly.

3. Motion Monitoring :
Continuously monitors rider head movement and tilt using a high-precision motion sensor.

4. Compact Embedded Design :
All components are integrated into a compact structure that can be mounted inside a helmet without affecting comfort.

5. Manual Override Button :
Allows the rider to stop the alarm manually if the alert was triggered accidentally.

6. Real-Time Processing :
Processes sensor data instantly using a microcontroller to ensure rapid response during emergencies.

7. Educational Value :
Serves as an excellent learning platform for students to understand embedded systems, sensors, and IoT-based safety technologies.

Scope of the Project :

The Smart Helmet Accident Detection System has wide potential applications across safety, education, and research domains. Its ability to detect accidents automatically makes it useful in real-world safety scenarios as well as academic demonstrations. The system showcases how embedded electronics and sensors can be used to solve practical problems.

---

1. Academic Institutions :
● The system can be used as a hands-on educational tool for learning about sensors, embedded programming, and safety system design. Students can assemble, program, and test the device, gaining practical technical skills.

---

2. Transportation and Road Safety :
● The smart helmet can enhance rider safety by providing automatic accident detection and alert mechanisms, which can reduce response time during emergencies.

---

3. Research and Development :
● Researchers can use this system as a base model for developing advanced safety devices such as smart wearables or intelligent transportation systems.

---

4. Industrial Safety :
● The concept can be adapted for workers in hazardous environments, such as construction or mining, where impact detection helmets could improve worker safety.

Objective :

The objective of the Smart Helmet Accident Detection System project is to design and develop an intelligent helmet capable of detecting accidents automatically using motion sensors and generating alerts instantly. The project aims to enhance safety, demonstrate embedded system applications, and provide a practical solution to a real-world problem.

Key Goals include :

Improve Rider Safety :
● Reduce response time during accidents by detecting impacts automatically.

Enhance System Reliability :
● Ensure accurate detection using sensor-based motion analysis.

Enable User Interaction :

● Provide manual control through a push button for alarm cancellation.

Implement Embedded Technology :
● Utilize microcontroller and sensor modules to build a real-time safety system.

Promote Learning :
● Encourage students to explore embedded systems, IoT concepts, and hardware programming.

Support Future Innovation :
● Provide a foundation for future upgrades such as GPS tracking, wireless alerts, and health monitoring sensors.

REQUIREMENT SPECIFICATION

Hardware Requirements

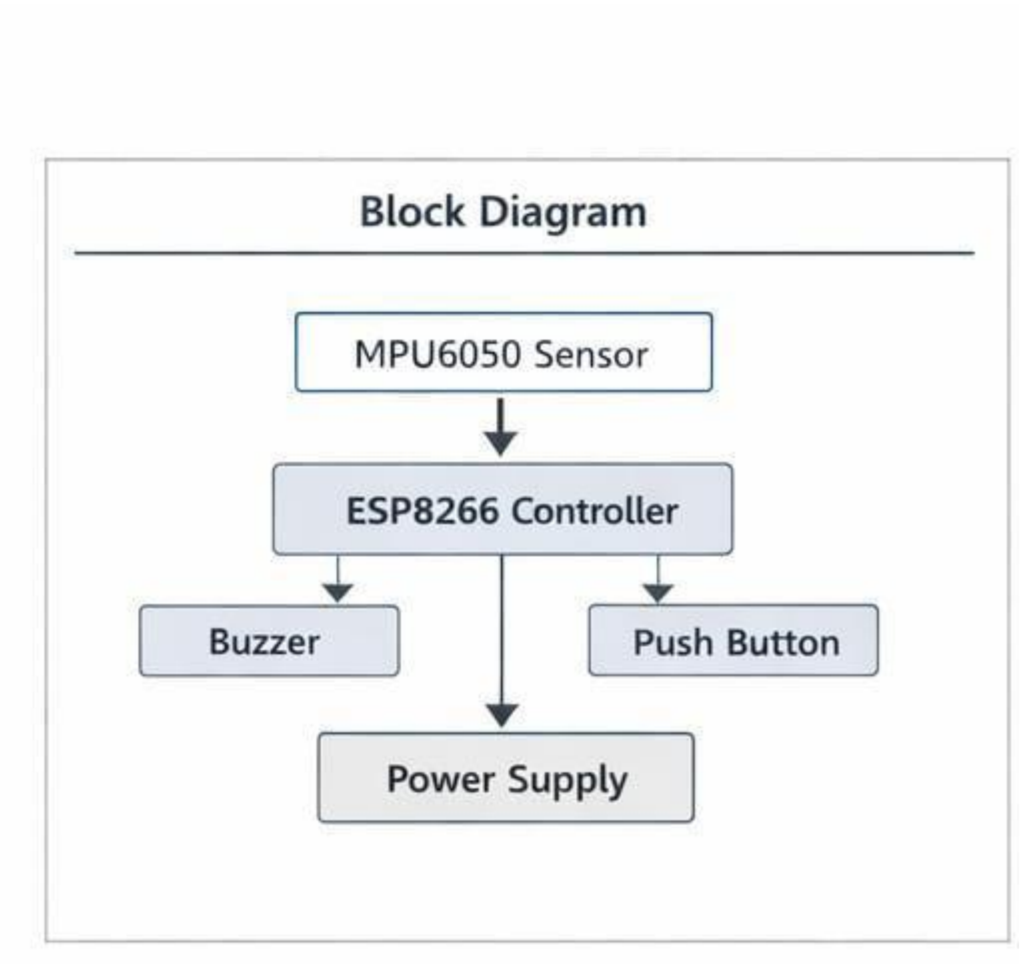
- ESP8266 WiFi microcontroller
- MPU6050 accelerometer and gyroscope sensor
- Buzzer
- Push button
- Battery
- Connecting wires
- Helmet casing

Software Requirements

- Arduino IDE
- Embedded C programming
- ESP8266 board drivers
- Serial monitor for debugging

SYSTEM DESIGN

Block Diagram :

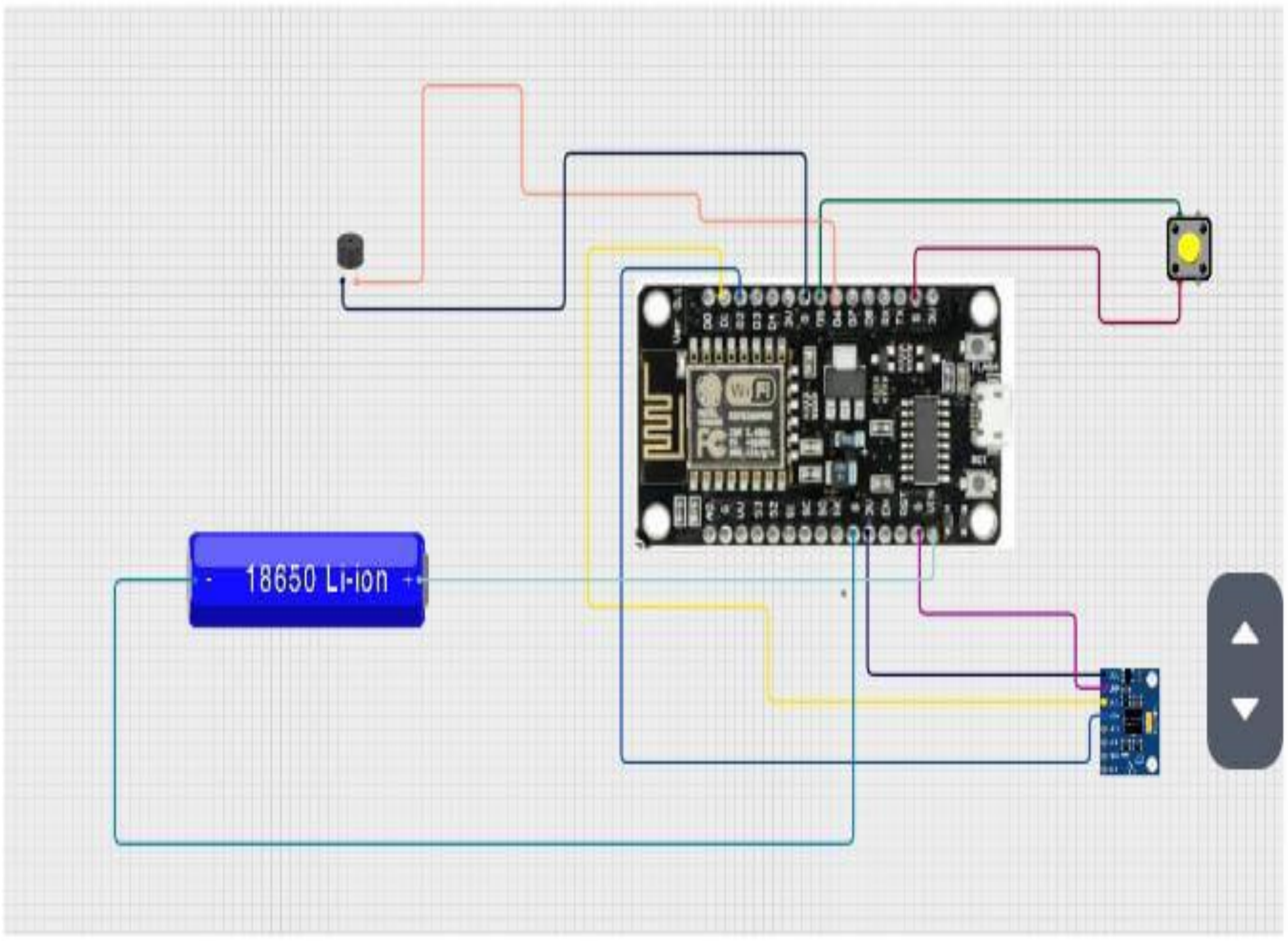


The block diagram outlines the high-level interaction between the core components of the Smart Helmet Accident Detection System. It demonstrates how different modules (sensor, controller, alert system, and input control) communicate and function together.

● **ESP8266 Microcontroller:** Acts as the central processing unit controlling all operations.
● **MPU6050 Sensor:** Detects motion, tilt, and sudden acceleration to identify possible accidents.
● **Buzzer:** Produces an audible alarm when an accident is detected.
● **Push Button:** Allows manual override to stop the alarm if triggered accidentally.
● **Power Supply:** Provides electrical power to the entire system.

The diagram visually shows how these components are connected to the controller and how they interact during operation.

Circuit Diagram :



Description :

The circuit diagram demonstrates the interaction between key components of the system: ESP8266, motion sensor, buzzer, and button. The controller receives data from the MPU6050 sensor and analyzes motion values. If abnormal movement is detected, it activates the buzzer. The push button allows manual control, and power is supplied through a battery source.

Key Steps of Circuit Diagram :

1. **Controller Setup:**
   Connect the ESP8266 microcontroller as the main processing unit. All modules communicate through this controller.
2. **MPU6050 Sensor:**
   Connect SDA → D2 and SCL → D1 pins. This sensor measures motion and acceleration.
3. **Buzzer Connection:**
   Connect buzzer positive terminal → D5 and negative → GND. It generates an alarm signal.
4. **Push Button:**
   Connect button → D7. This acts as a manual reset switch.
5. **Power Supply:**
   Connect battery to VIN and GND pins. Ensure proper voltage regulation to prevent damage.

These steps ensure proper integration of components for reliable system performance.

Benefits of Circuit Diagram :

1. Clear Visualization:
   Shows how all components are interconnected.
2. Error Detection:
   Helps identify wiring mistakes before assembly.
3. Documentation:
   Serves as reference for future replication.
4. Ease of Assembly:
   Makes hardware assembly simple and organized.

This is essential for embedded safety systems to ensure correct functionality and prevent hardware failure.

When to Use a Circuit Diagram ?

Design Phase:
Helps visualize system architecture before building.
Troubleshooting:
Useful for diagnosing wiring issues.
Documentation:
Important for reports, publications, and presentations.
Education:
Helps students understand electronics and system design.
Assembly Guidance:
Provides a roadmap for assembling hardware correctly.

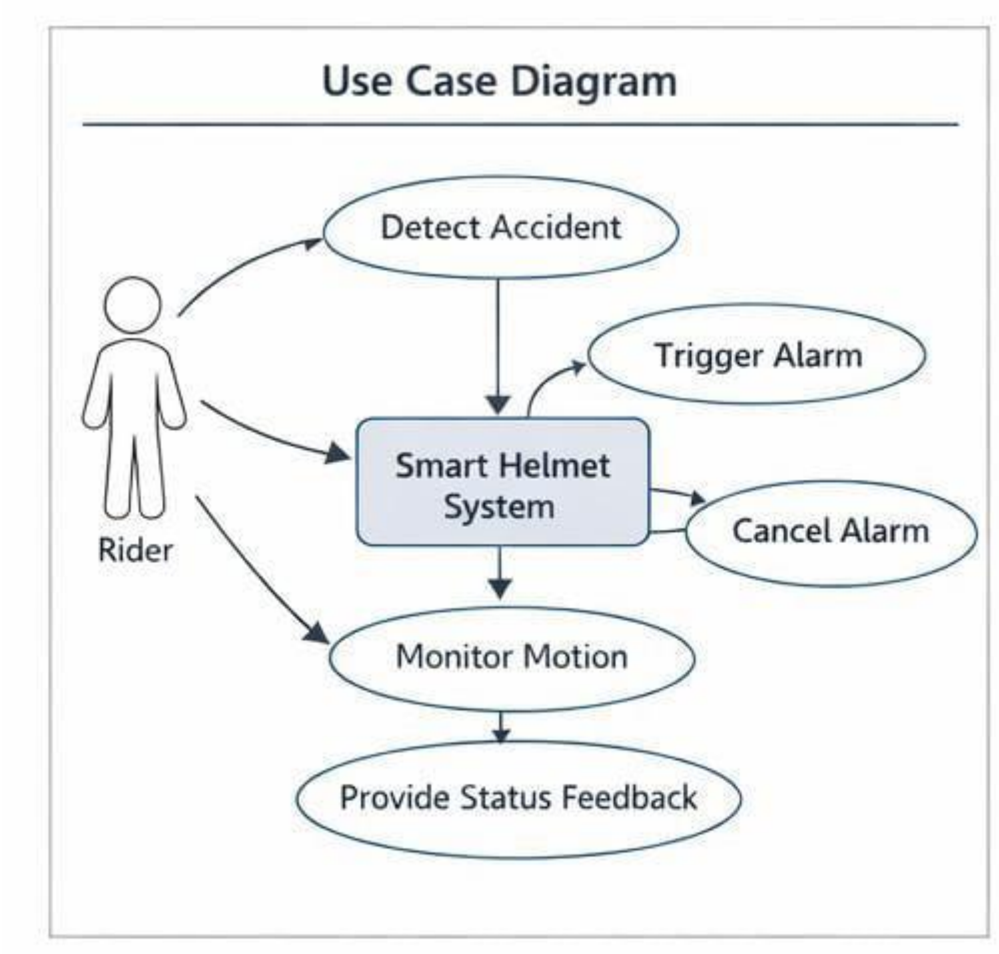
Circuit Model Testing Methods :

1. Simulation Testing:
   Use simulation tools to test logic before building hardware.
2. Continuity Testing:
   Verify connections using a multimeter.
3. Functional Testing:
   Check sensor readings and buzzer response.
4. Load Testing:

Test system performance for extended operation.

5. Debugging:
   Identify and resolve unexpected behavior.

<u>Use Case Diagram</u> :



Description :

The Use Case Diagram illustrates interactions between users and the Smart Helmet system. It represents actions performed by the user and responses of the system. The main actors are Rider and Smart Helmet System.

Detect Accident

●Actor:Smart_Helmet

●Description:

The system continuously monitors motion data. If sudden acceleration or abnormal tilt is detected, it identifies a possible accident.

Trigger Alarm

●Actor:_Smart_Helmet

●Description:

When an accident is detected, the buzzer is activated automatically to alert nearby people.

Cancel Alarm

●Actor:Rider

●Description:

The rider can press the button to stop the alarm if it was triggered accidentally.

Monitor Motion

●Actor:Smart_Helmet

●Description:

The system constantly monitors head movement and acceleration values in real time.

Provide Status Feedback

●Actor:Smart_Helmet

●Description:

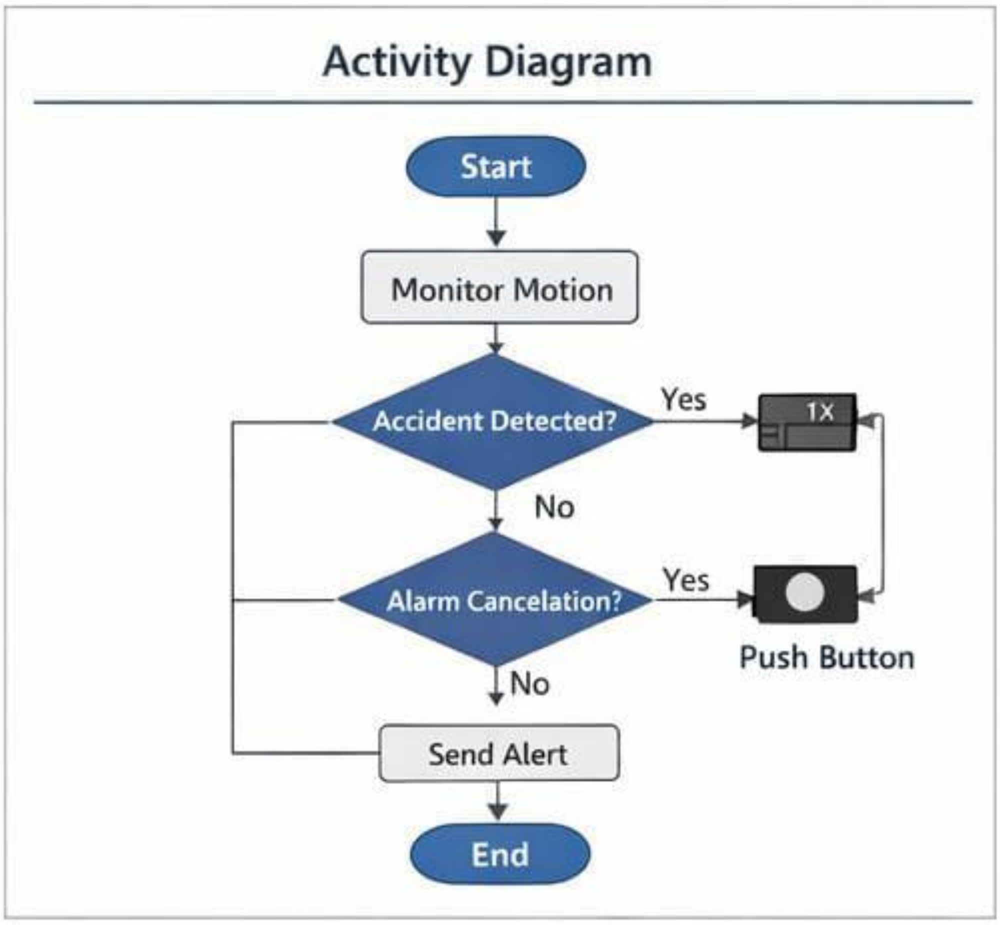
The system provides feedback through sound signals indicating alert or normal status.

Purpose of Use Case Diagrams :

1. Visualizing system interactions
2. Defining system requirements
3. Facilitating communication

4. Guiding system design
5. Documenting functionality
6. Supporting testing
7. Identifying user needs
8. Simplifying complex systems

Activity Diagram :



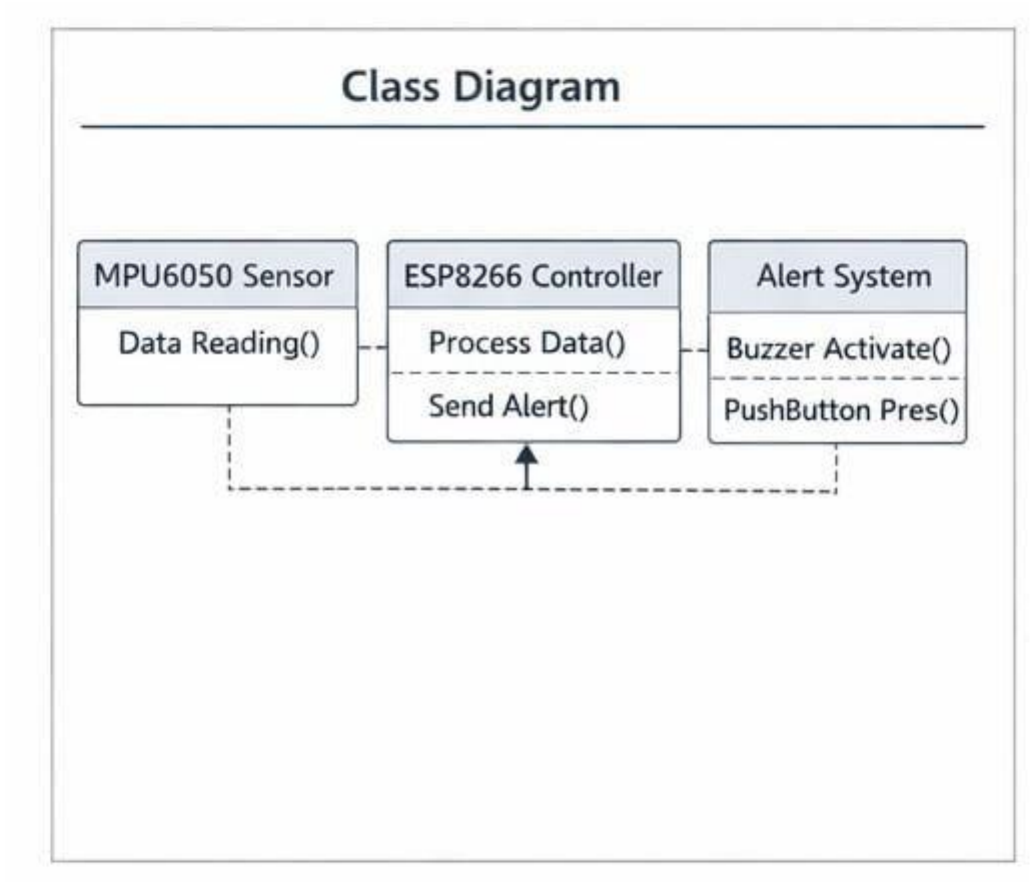
Description :

Activity diagrams represent workflow and process flow inside the system. They illustrate the sequence of operations performed by the Smart Helmet during accident monitoring.

Purpose of Activity Diagrams :

- Visualize Workflow — Shows operation sequence clearly
- Model Processes — Represents system logic step-by-step
- Identify Parallel Activities — Detect simultaneous processes
- Clarify Responsibilities — Defines system tasks
- Support Communication — Helps explain system logic
- Facilitate Documentation — Provides workflow reference
- Enhance System Design — Helps improve logic efficiency
- Testing Support — Helps design test cases

Class Diagram :



Class diagrams visually represent system components and their relationships. For this project, they illustrate how the sensor, controller, and alert modules interact.

Purpose of Class Diagrams :

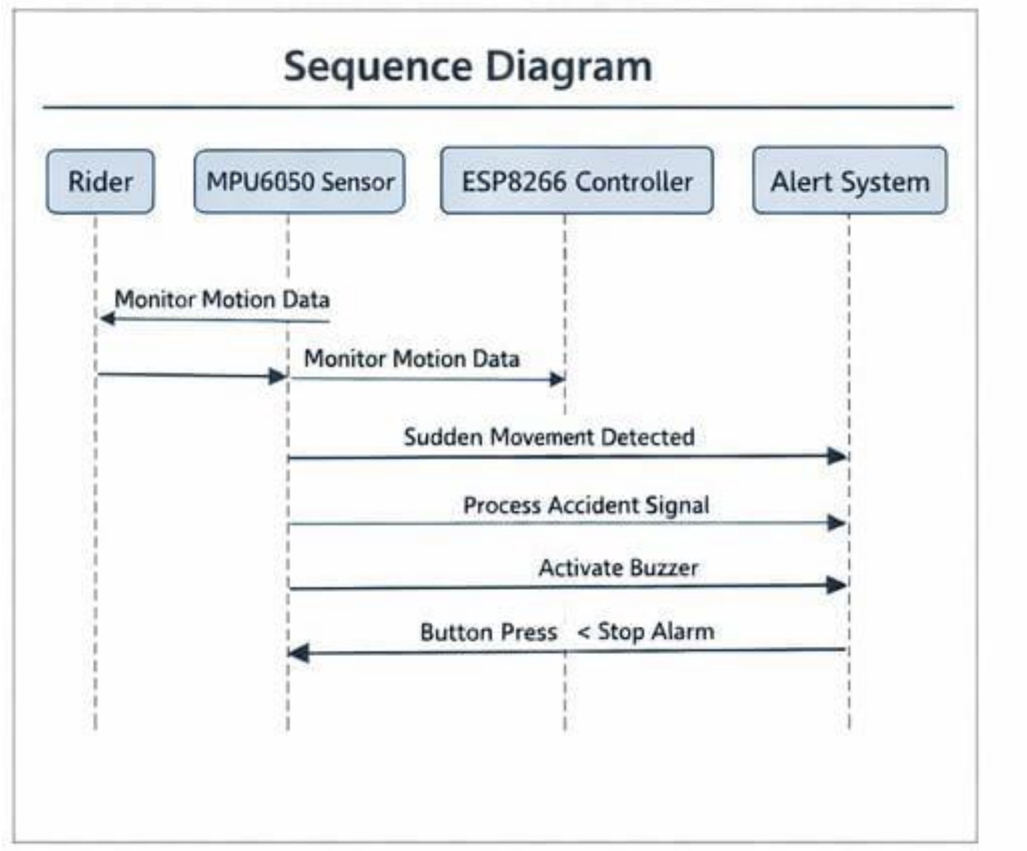
●Modeling_System_Structure
Shows relationship between modules like sensor, controller, buzzer, and button.

●Understanding_Relationships
Clarifies data flow between components.

●Design_Blueprint
Provides a reference for coding and wiring.

●Documentation
Helps future maintenance and upgrades.

Sequence Diagram :



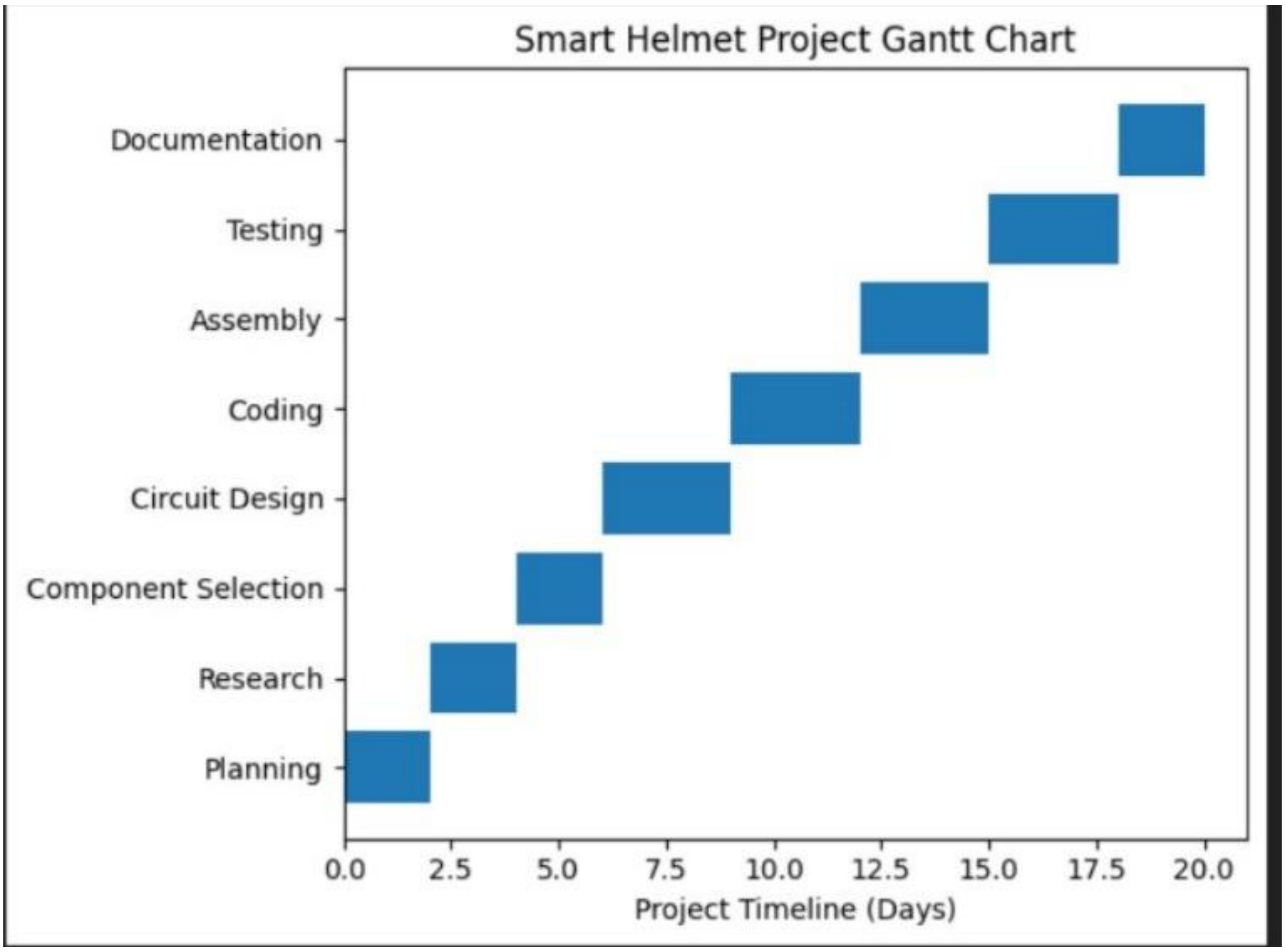
Description :

A Sequence Diagram illustrates how system components interact with each other in a time sequence. It shows how data flows between the sensor, controller, and alert system when an accident occurs.

Purpose of Sequence Diagrams :

- Visual representation of interactions
- Clarifies message flow between modules
- Helps developers understand execution order
- Guides programming logic
- Provides system documentation

Gantt Chart :



Description :

A Gantt Chart is a type of bar chart that represents a project schedule. It visually outlines the tasks involved in a project along a timeline, showing when each task starts, how long it will take, and when it should be completed. Gantt charts are widely used in project management to plan, coordinate, and track specific tasks or activities, providing a clear overview of the project's progress and deadlines.

The above Gantt chart provides a visual representation of the timeline for the multi function arduino robot project, with tasks scheduled between **July 10, 2024**, and **October 10, 2024**. The chart breaks down the project into key phases, displaying when each phase begins and ends .

Advantages of Using Gantt Charts :

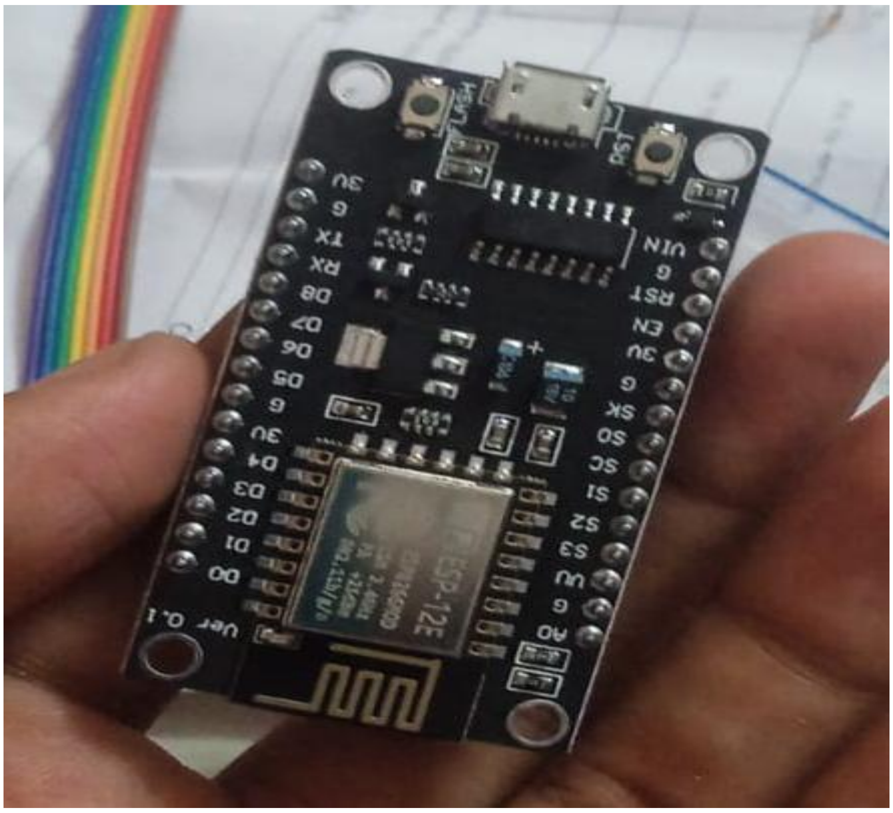
1. Clear Visualization
2. Effective Planning
3. Tracking Progress
4. Collaboration

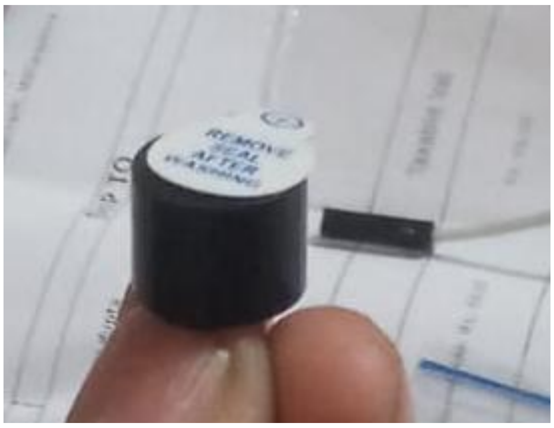
SYSTEM IMPLEMENTATION

Step-by-Step Assembly:

Step 1 :

Firstly, identify these components.



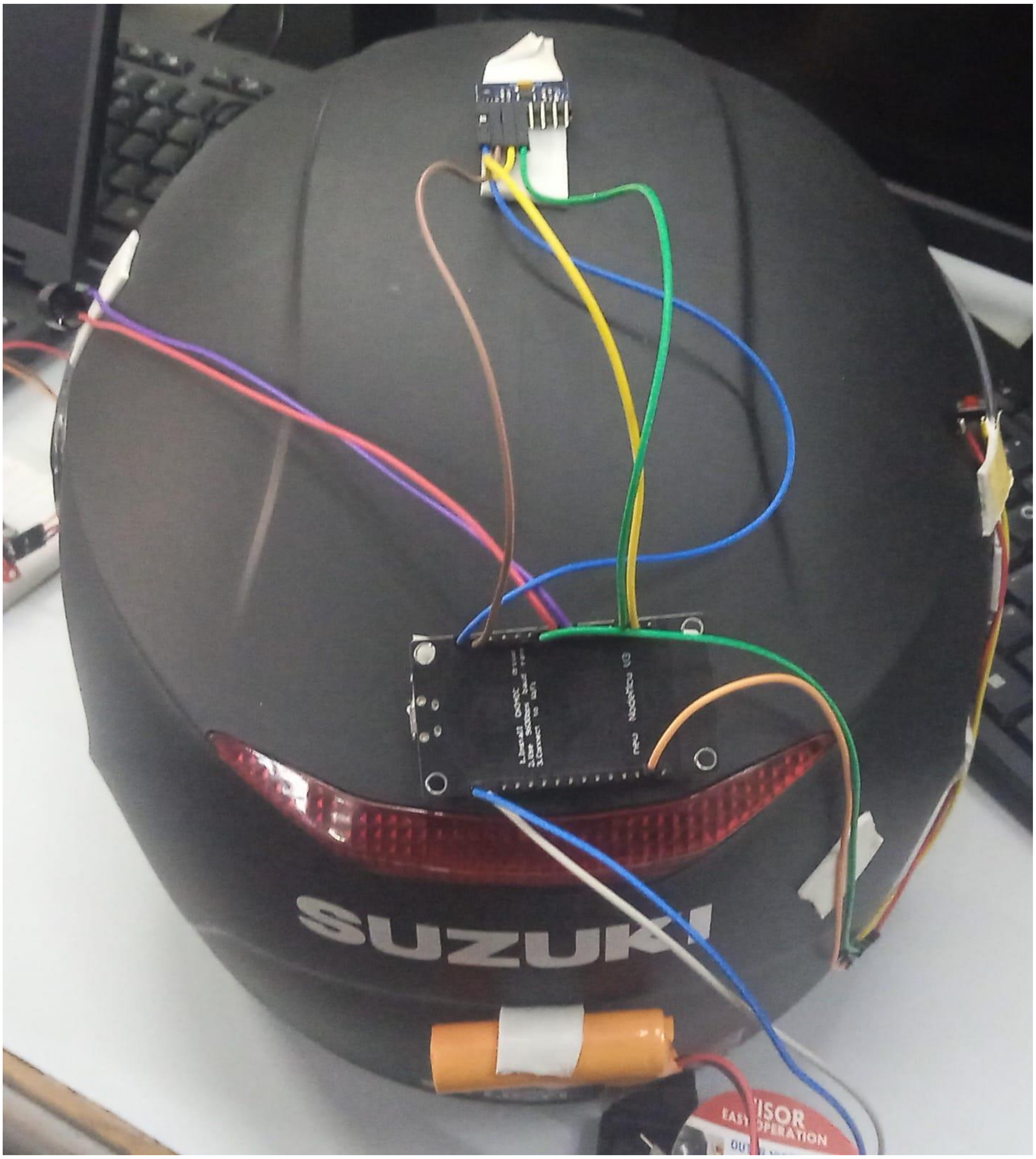


Step 2 :

Secondly, cut the foam helmet as follows :

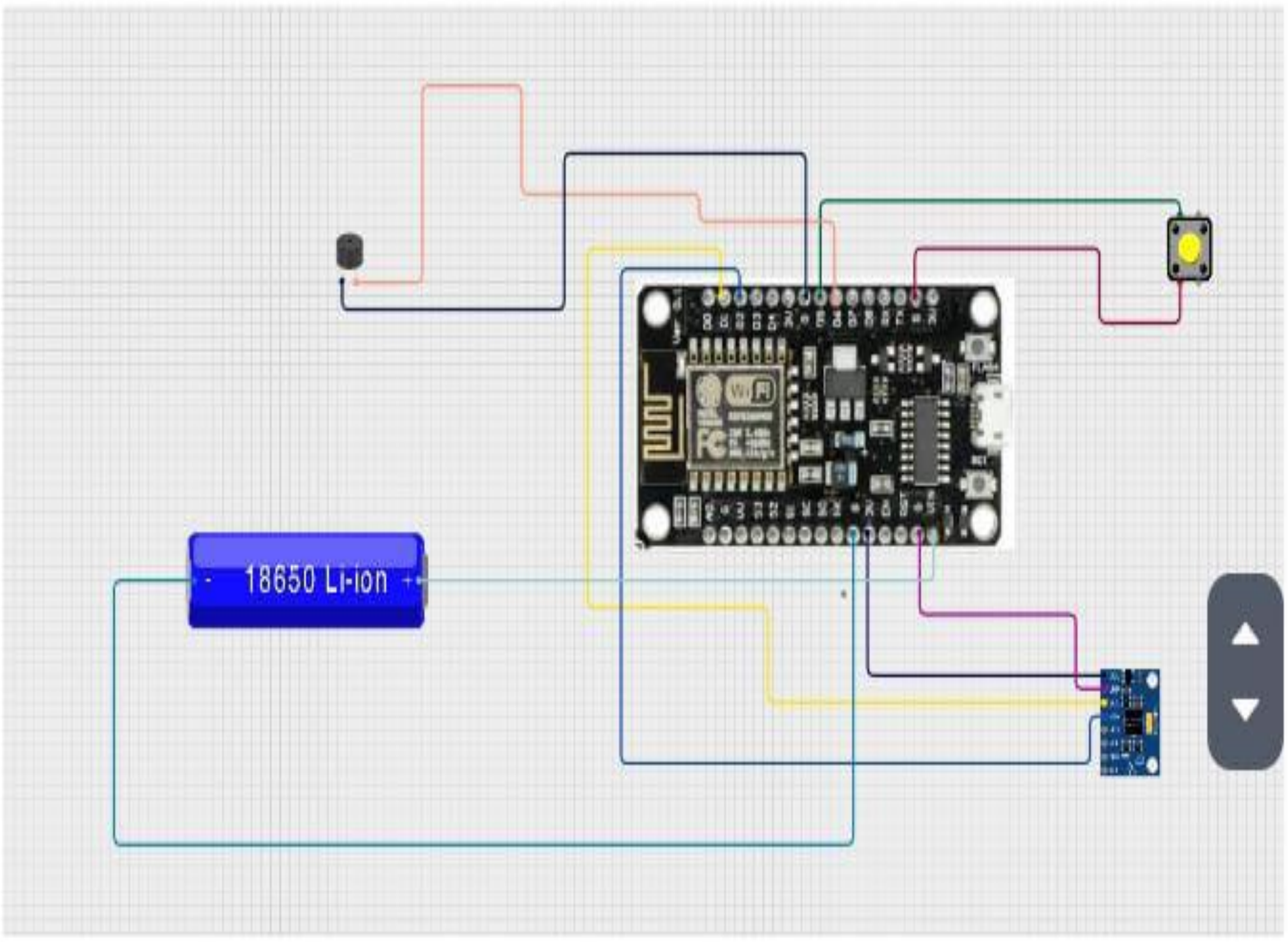
Step 3 :

Thirdly, connect the component on helmet



Step 6 :

Then, connect the esp8266,mpu6050 on helmet to detect accident its block diagra



```
 Smart helmet for accident detection using wifi , recived msg when accident is detected*/
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

const char* ssid = "Pgl.♡▯";
const char* password = "MH050553";

String botToken = "8354725551:AAHKeKvUvCn95ZoeZE3BhAIwHD2vNZeNUDs";
String chatID = "8570744271";

WiFiClientSecure client;

const int buzzer = D5;
const int button = D7;
const int MPU_addr = 0x68;

int16_t AcX, AcY, AcZ;

bool accidentSent = false;
bool alarmDisabled = false;
int triggerCount = 0;

// double tap variables
unsigned long lastTapTime = 0;
int tapCount = 0;
float tapThreshold = 2.2;   // adjust sensitivity

// universal buzzer function (works for all buzzers)
void beep(int duration){
  unsigned long start = millis();
  while(millis() - start < duration){
    digitalWrite(buzzer,HIGH);
    delayMicroseconds(500);
    digitalWrite(buzzer,LOW);
    delayMicroseconds(500);
  }
}

// telegram function
void sendTelegram(String message) {
  client.setInsecure();

  if (!client.connect("api.telegram.org", 443)) {
    Serial.println("Connection failed");
    return;
  }

  String url = "/bot" + botToken + "/sendMessage?chat_id=" + chatID + "&text=" + message;

  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: api.telegram.org\r\n" +
               "Connection: close\r\n\r\n");
}
```

```
void setup() {
  Wire.begin(D2, D1);
  Serial.begin(9600);

  pinMode(buzzer, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  // MPU6050 start
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);

  // WiFi connect
  WiFi.begin(ssid, password);
  Serial.print("Connecting WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected!");
}

void loop() {

  // read sensor
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 6, true);

  AcX = Wire.read() << 8 | Wire.read();
  AcY = Wire.read() << 8 | Wire.read();
  AcZ = Wire.read() << 8 | Wire.read();

  float x = AcX / 16384.0;
  float y = AcY / 16384.0;
  float z = AcZ / 16384.0;

  Serial.print("X: "); Serial.print(x);
  Serial.print(" Y: "); Serial.print(y);
  Serial.print(" Z: "); Serial.println(z);

  float magnitude = sqrt(x*x + y*y + z*z);

  // ---------- DOUBLE TAP DETECT ----------
  if (magnitude > tapThreshold) {

    unsigned long currentTime = millis();

    if (currentTime - lastTapTime < 500) {
      tapCount++;
    } else {
      tapCount = 1;
    }
    lastTapTime = currentTime;

    if (tapCount == 2) {
      alarmDisabled = !alarmDisabled;

      Serial.println("DOUBLE TAP DETECTED");

      beep(400);

      tapCount = 0;
      delay(300);
    }
  }

  // ---------- BUTTON DISABLE ----------
  if (digitalRead(button) == LOW) {
    alarmDisabled = true;
    Serial.println("Alarm Disabled by Button");
    delay(500);
  }

  // ---------- ACCIDENT DETECT ----------
  if (!alarmDisabled) {

    if (abs(x) > 0.50 || abs(y) > 0.50) {
      triggerCount++;
    } else {
      triggerCount = 0;
    }

    // confirm accident
    if (triggerCount >= 3) {

      beep(1000);

      if (!accidentSent) {
        sendTelegram("🚨 Accident Detected!");
        accidentSent = true;
      }

    } else {
      accidentSent = false;
    }
  }

  delay(150);
}
```

🔍 Code Explanation (Line-by-Line Concept)

📚 1. Libraries

```
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
```

- **Wire.h → for I2C communication (MPU6050 sensor)**
- **ESP8266WiFi.h → connect ESP8266 to WiFi**
- **WiFiClientSecure.h → send HTTPS request (Telegram message)**

🌐 2. WiFi + Telegram Credentials

```
const char* ssid = "Pgl.♡□";
const char* password = "MH050553";
String botToken = "...";
String chatID = "...";
```

- **WiFi name + password → for internet connection**
- **Bot token + chat ID → to send alert to Telegram**

🔊 3. Pin + Sensor Setup

```
const int buzzer = D5;
const int button = D7;
const int MPU_addr = 0x68;
```

- **Buzzer → D5**
- **Button → D7**
- **MPU6050 I2C address → 0x68**

📊 4. Sensor Variables

```
int16_t AcX, AcY, AcZ;
```

Stores raw accelerometer data from MPU6050.

🚨 5. Logic Flags

```
bool accidentSent = false;
bool alarmDisabled = false;
int triggerCount = 0;
```

- **accidentSent → prevents multiple messages**
- **alarmDisabled → disables alarm**
- **triggerCount → confirms accident after multiple shakes**

👆 6. Double Tap Detection Variables

```
unsigned long lastTapTime = 0;
int tapCount = 0;
float tapThreshold = 2.2;
```

Used to detect double tap on helmet.

🔔 7. Buzzer Function
void beep(int duration)
Creates sound by rapidly switching buzzer HIGH/LOW.
This works for:

- **active buzzer**
- **passive buzzer**

📩 8. Telegram Message Function
void sendTelegram(String message)
Steps:

1. **Connects to Telegram server**
2. **Sends HTTP request**
3. **Telegram sends message to your phone**

⚙ 9. Setup Function
Runs once when device starts.
✓ Start I2C
Wire.begin(D2, D1);

- **D2 → SDA**
- **D1 → SCL**

✓ Start Serial Monitor
Serial.begin(9600);
For debugging values.

✓ Pin Modes
pinMode(buzzer, OUTPUT);
pinMode(button, INPUT_PULLUP);
Button uses internal pull-up resistor.

✓ Wake MPU6050
Wire.write(0x6B);
Wire.write(0);
This removes MPU6050 from sleep mode.

✓ Connect WiFi
WiFi.begin(ssid, password);
Loops until connected.

🔁 10. Loop Function (Main Brain)
Runs repeatedly.

📡 Step 1 — Read Sensor Data
Wire.write(0x3B);

Wire.requestFrom(MPU_addr, 6, true);
Reads accelerometer X Y Z values.

Convert raw values:
float x = AcX / 16384.0;
This converts into g-force units.

📏 Step 2 — Calculate Motion Strength
float magnitude = sqrt(x*x + y*y + z*z);
This gives total motion strength.

👆 Step 3 — Double Tap Detection
if (magnitude > tapThreshold)
If helmet is tapped hard enough → detect tap.
If two taps within 500 ms:
tapCount == 2
Then:
✔ Toggle alarm ON/OFF
✔ Buzzer beep confirmation

🔘 Step 4 — Button Disable
if (digitalRead(button) == LOW)
If button pressed → disable alarm manually.

🚑 Step 5 — Accident Detection
if (abs(x) > 0.50 || abs(y) > 0.50)
If strong tilt or shake → possible crash.

To confirm:
triggerCount >= 3
Needs 3 consecutive readings → avoids false alarm.

✔ If Accident Confirmed
beep(1000);
sendTelegram("🚨 Accident Detected!");

- **buzzer sounds**
- **message sent to phone**

✔ Reset Condition
If motion becomes normal:
triggerCount = 0;

⏱ Loop Delay
delay(150);
System checks every 150 ms

▯ Overall Working Logic

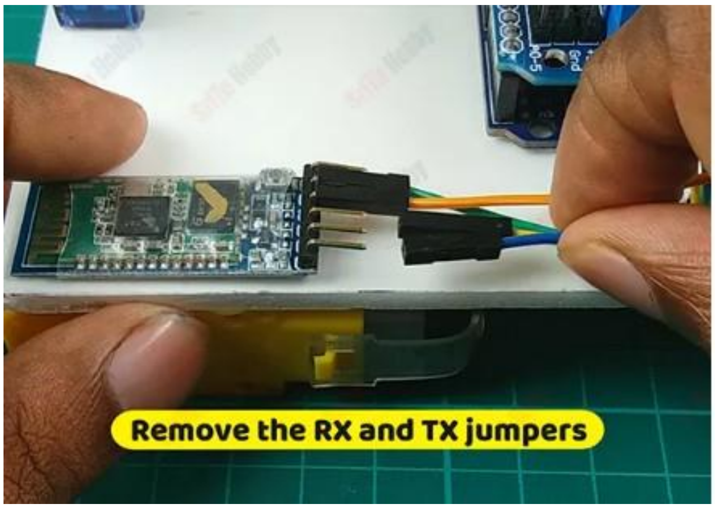
```
Start
↓
Connect WiFi
↓
Read Motion
↓
Double tap?
 ├ YES → Toggle alarm
 └ NO
↓
Button pressed?
 ├ YES → Disable alarm
 └ NO
↓
Accident motion detected?
 ├ YES → Buzzer + Telegram
 └ NO → Continue
```

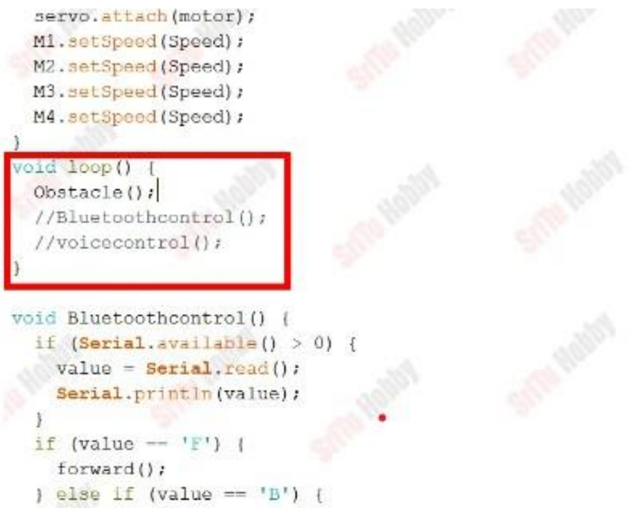
SYSTEM TESTING AND RESULT

Step 13 :

Obstacle avoidance program

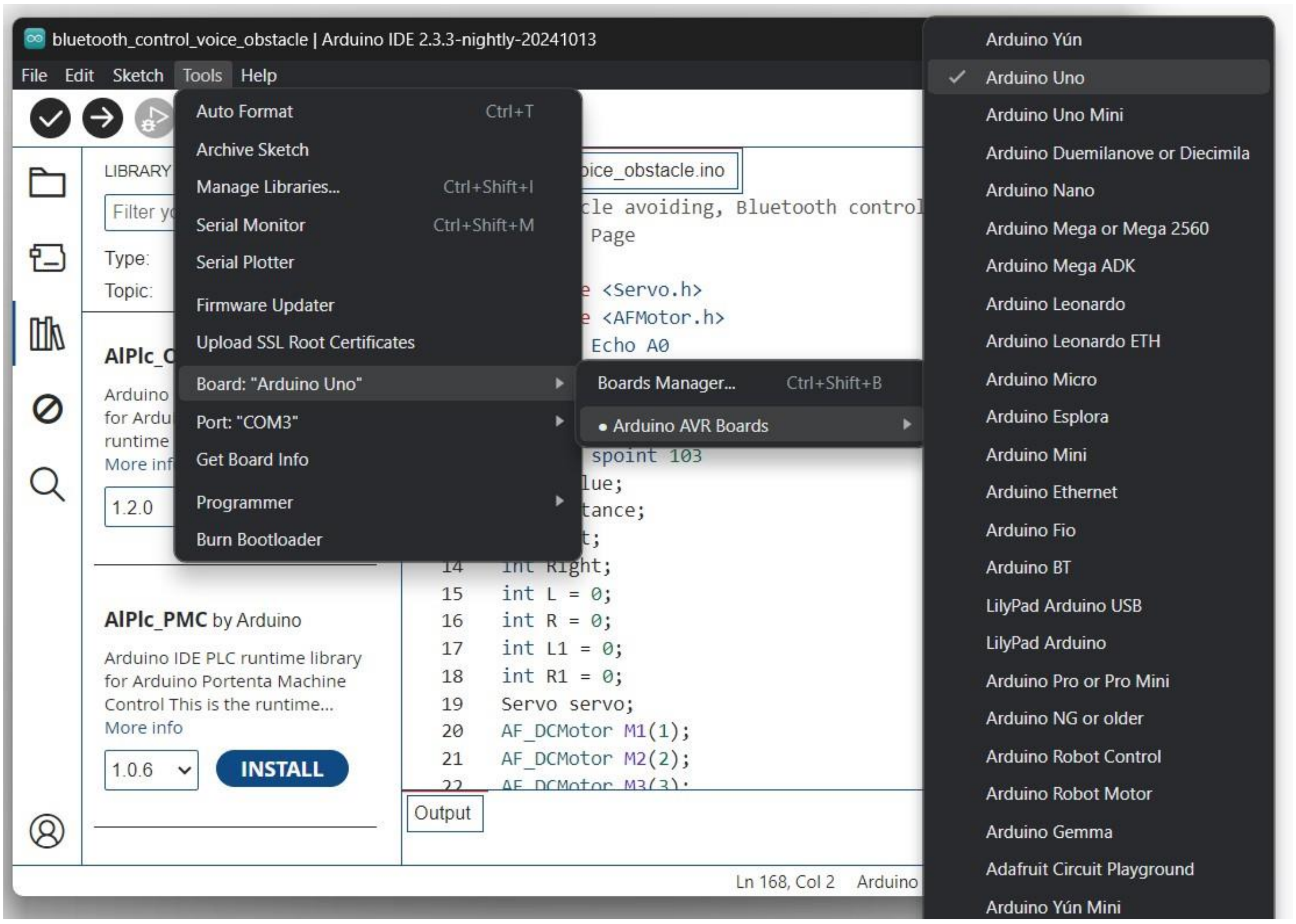
OK, now connect this robot car to the computer. Then, remove the two forward slashes in front of the "obstacle" function. Next, remove the RX and TX jumper wires connected to the Bluetooth module.





```
  servo.attach(motor);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);
}
void loop() {
  Obstacle();
  //Bluetoothcontrol();
  //voicecontrol();
}

void Bluetoothcontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
  }
  if (value == 'F') {
    forward();
  } else if (value == 'B') {
```

Now, select board and port. After, upload this code to the robot and reconnect the RX and TX jumper wires.



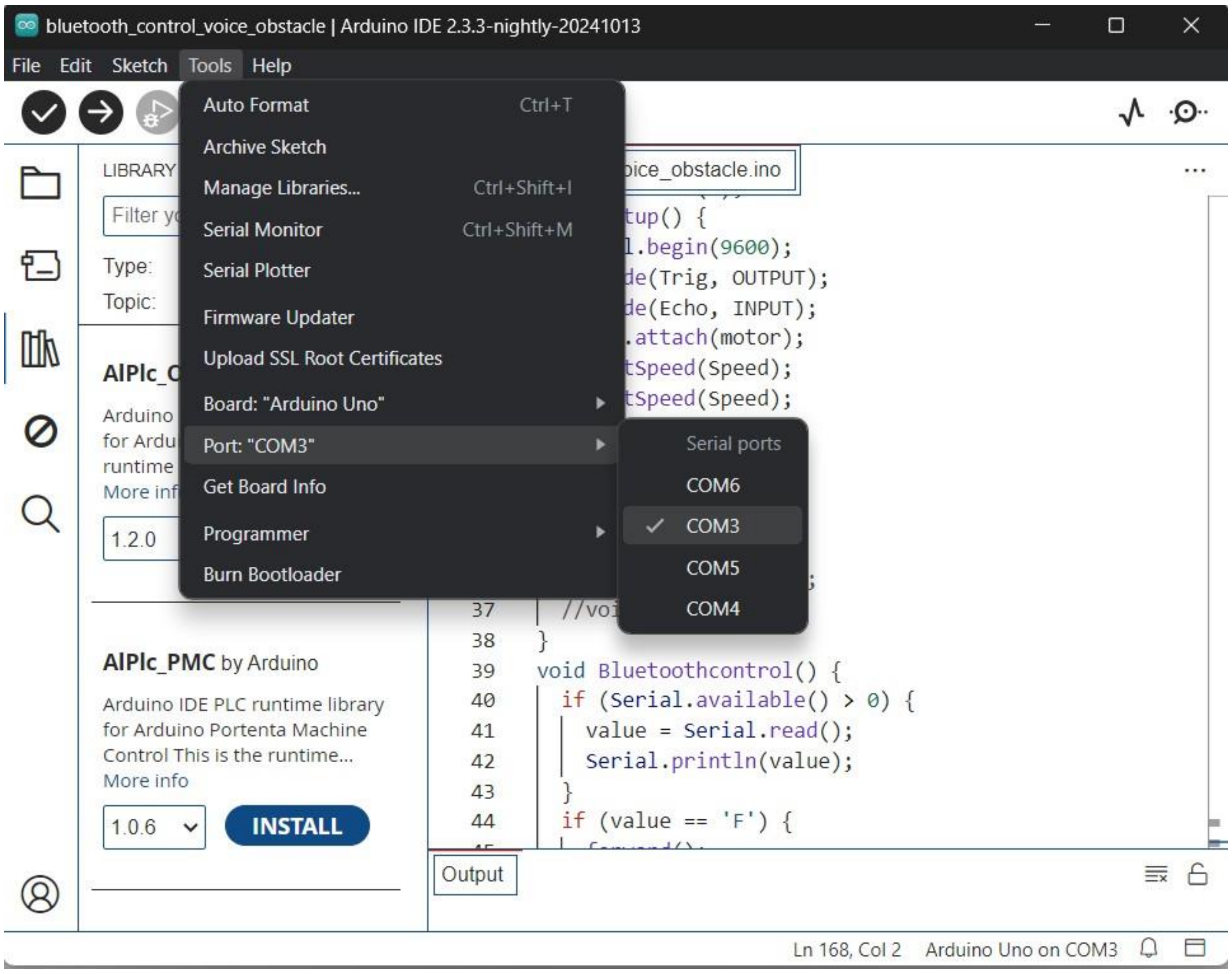
bluetooth_control_voice_obstacle | Arduino IDE 2.3.3-nightly-20241013

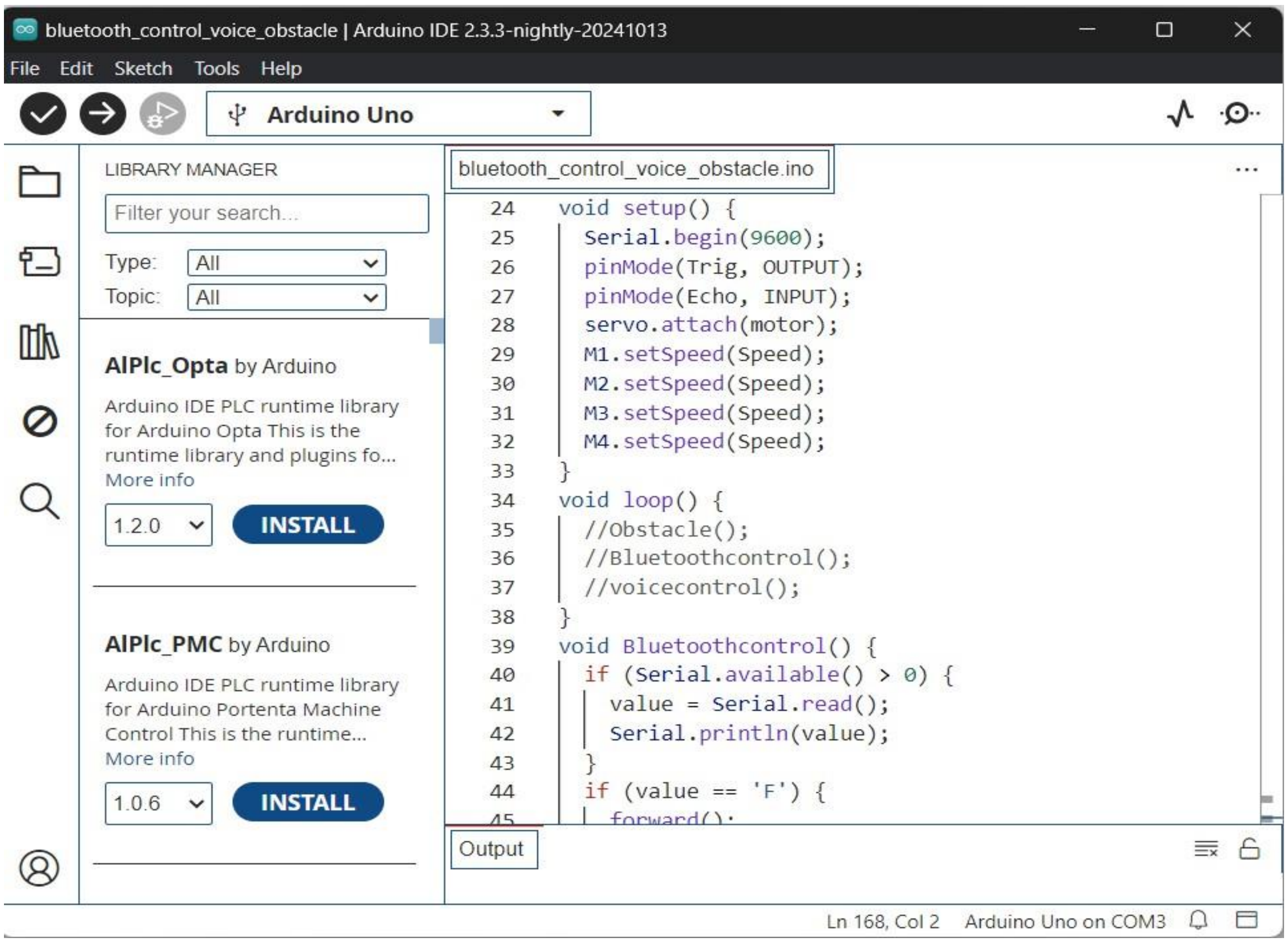
Board: "Arduino Uno"

Port: "COM3"

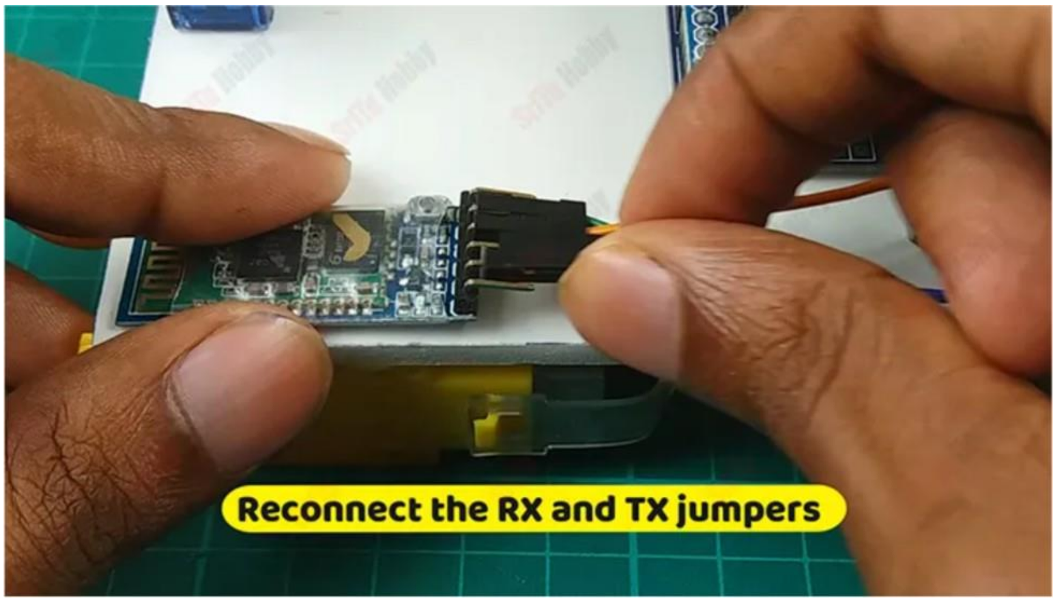
```
int L = 0;
int R = 0;
int L1 = 0;
int R1 = 0;
Servo servo;
AF_DCMotor M1(1);
AF_DCMotor M2(2);
```

Ln 168, Col 2 Arduino

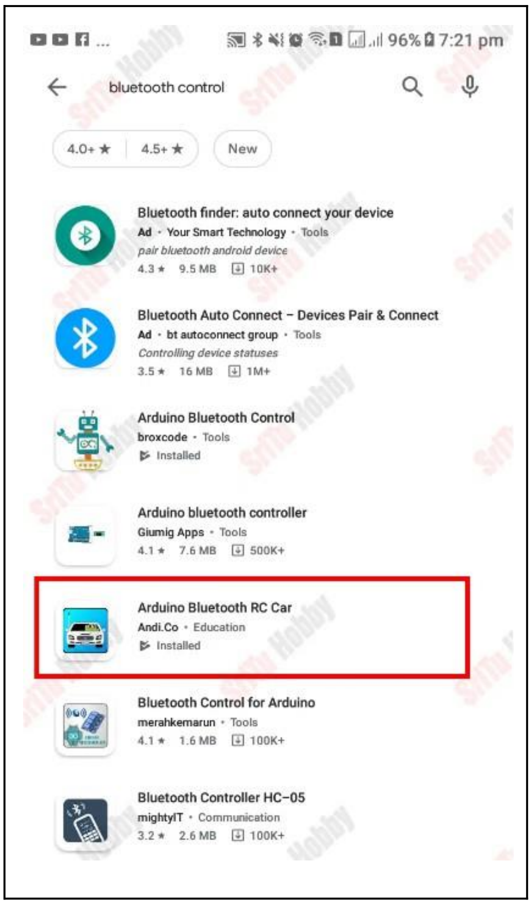

bluetooth_control_voice_obstacle | Arduino IDE 2.3.3-nightly-20241013
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Uno"
Port: "COM3"
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM6
✓ COM3
COM5
COM4
Output
Ln 168, Col 2 Arduino Uno on COM3


bluetooth_control_voice_obstacle | Arduino IDE 2.3.3-nightly-20241013
File Edit Sketch Tools Help
Arduino Uno
LIBRARY MANAGER
Filter your search...
Type: All
Topic: All
AIPlc_Opta by Arduino
Arduino IDE PLC runtime library for Arduino Opta This is the runtime library and plugins fo...
More info
1.2.0 INSTALL
AIPlc_PMC by Arduino
Arduino IDE PLC runtime library for Arduino Portenta Machine Control This is the runtime...
More info
1.0.6 INSTALL
bluetooth_control_voice_obstacle.ino
Output
Ln 168, Col 2 Arduino Uno on COM3

```
void setup() {
  Serial.begin(9600);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  servo.attach(motor);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);
}
void loop() {
  //Obstacle();
  //Bluetoothcontrol();
  //voicecontrol();
}
void Bluetoothcontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
  }
  if (value == 'F') {
    forward();
```



OK, now download and install the app below. Then, follow the steps below.



96% 7:21 pm

bluetooth control

4.0+ ★ 4.5+ ★ New

Bluetooth finder: auto connect your device
Ad · Your Smart Technology · Tools
pair bluetooth android device
4.3 ★ 9.5 MB 10K+

Bluetooth Auto Connect – Devices Pair & Connect
Ad · bt autoconnect group · Tools
Controlling device statuses
3.5 ★ 16 MB 1M+

Arduino Bluetooth Control
broxcode · Tools
Installed

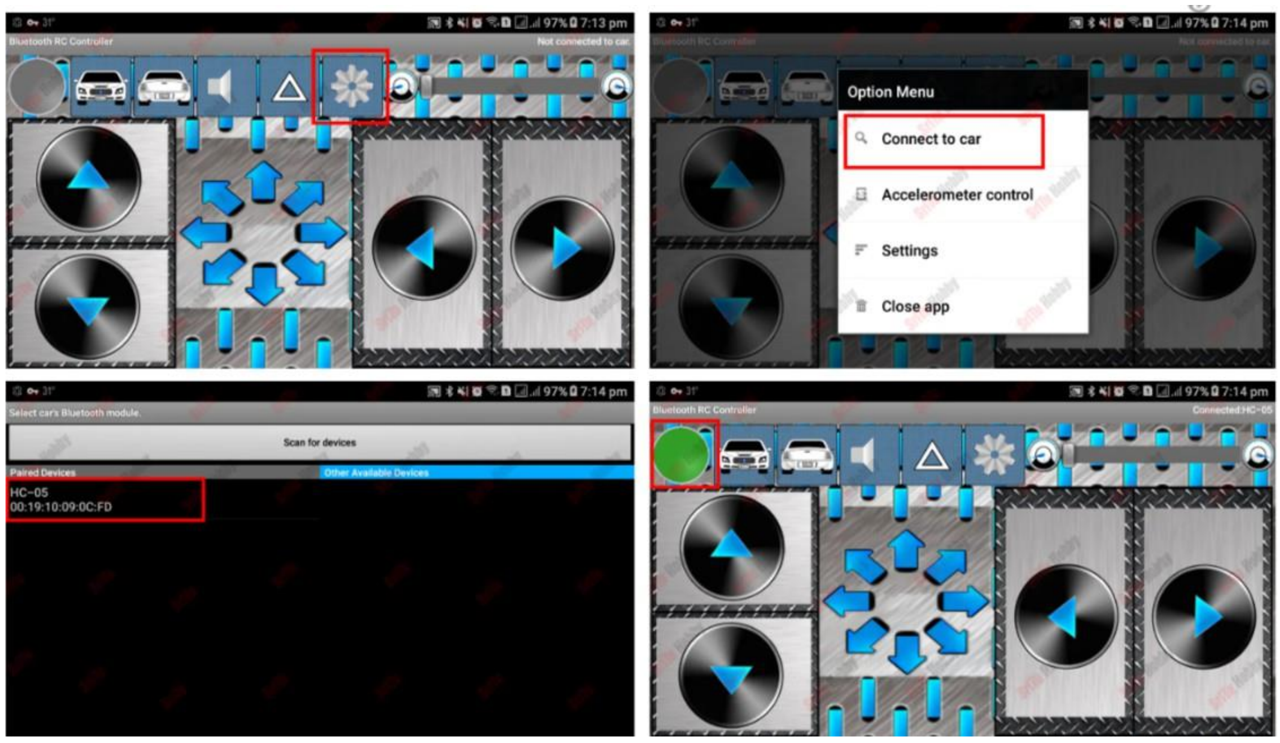
Arduino bluetooth controller
Giumig Apps · Tools
4.1 ★ 7.6 MB 500K+

Arduino Bluetooth RC Car
Andi.Co · Education
Installed

Bluetooth Control for Arduino
merahkemarun · Tools
4.1 ★ 1.6 MB 100K+

Bluetooth Controller HC-05
mightyIT · Communication
3.2 ★ 2.6 MB 100K+

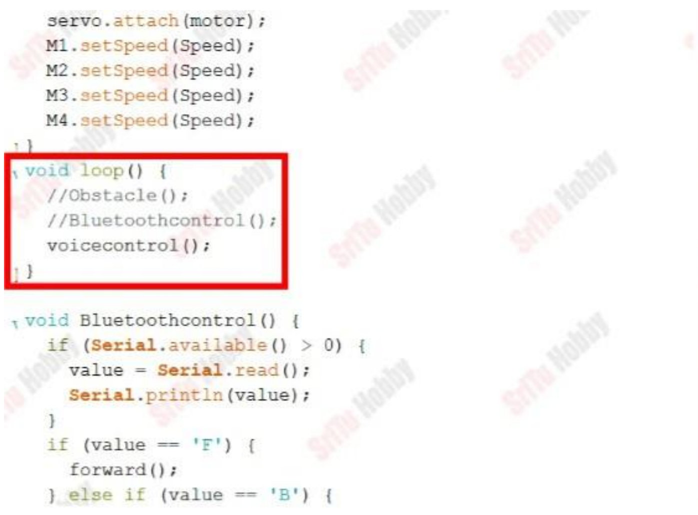
After, run this application and click the Settings button. Then, click the “Connect to Car” button and select the name of the Bluetooth module. Now, you can see the green bulb in the corner.



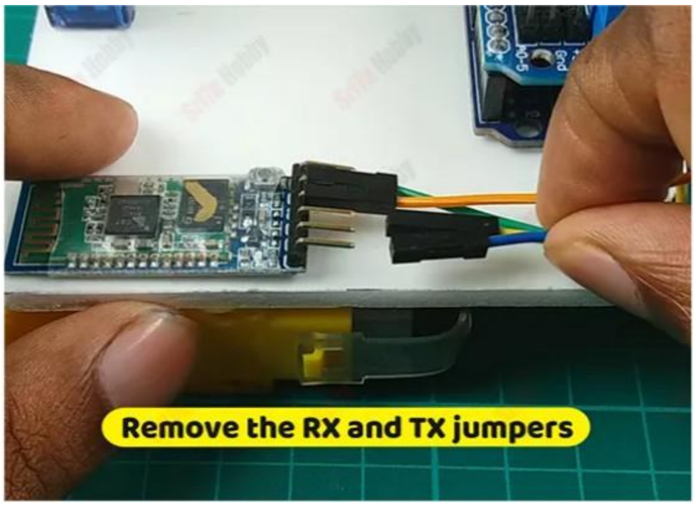
OK, now click the controller buttons and enjoy it.

Voice control program

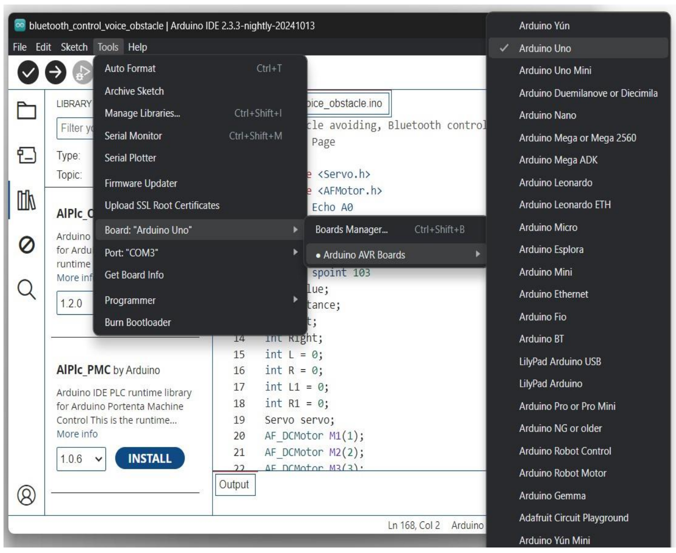
OK, now connect this robot car to the computer. Then, remove the two forward slashes in front of the “voicecontrol” function. Next, remove the RX and TX jumper wires connected to the Bluetooth module.



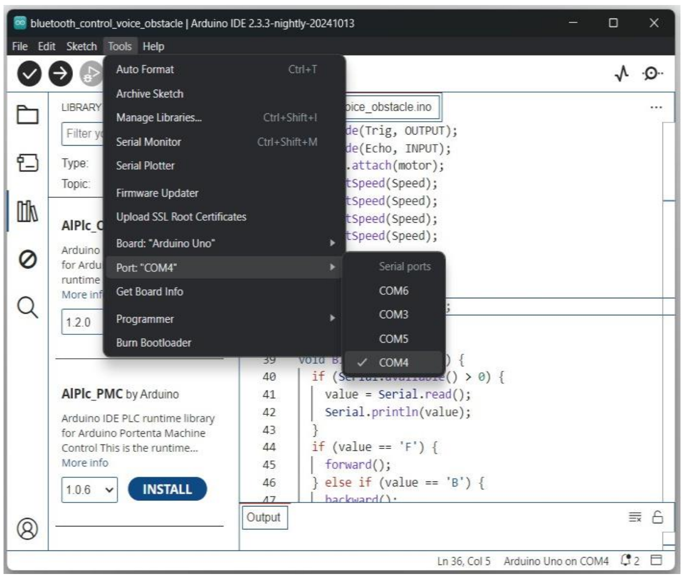
```
  servo.attach(motor);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);
}
void loop() {
  //Obstacle();
  //Bluetoothcontrol();
  voicecontrol();
}

void Bluetoothcontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
  }
  if (value == 'F') {
    forward();
  } else if (value == 'B') {
```



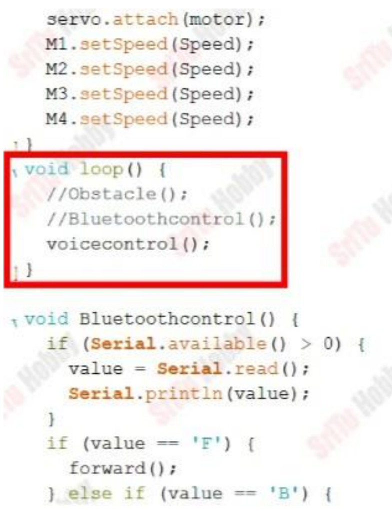
Now, select board and port. After, upload this code to the robot and reconnect the RX and TX jumper wires.Now, select board and port. After, upload this code to the robot and reconnect the RX and TX jumper wires.



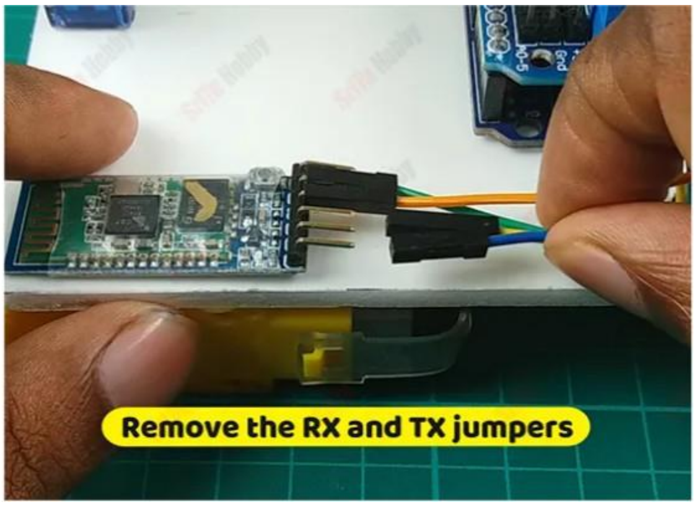
```
  int Right;
  int L = 0;
  int R = 0;
  int L1 = 0;
  int R1 = 0;
  Servo servo;
  AF_DCMotor M1(1);
  AF_DCMotor M2(2);
```


bluetooth_control_voice_obstacle | Arduino IDE 2.3.3-nightly-20241013
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Uno"
Port: "COM4"
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM6
COM3
COM5
✓ COM4
AIPlc_PMC by Arduino
INSTALL
Output
Ln 36, Col 5 Arduino Uno on COM4

```
    if (Serial.available() > 0) {
      value = Serial.read();
      Serial.println(value);
    }
    if (value == 'F') {
      forward();
    } else if (value == 'B') {
      backward();
```

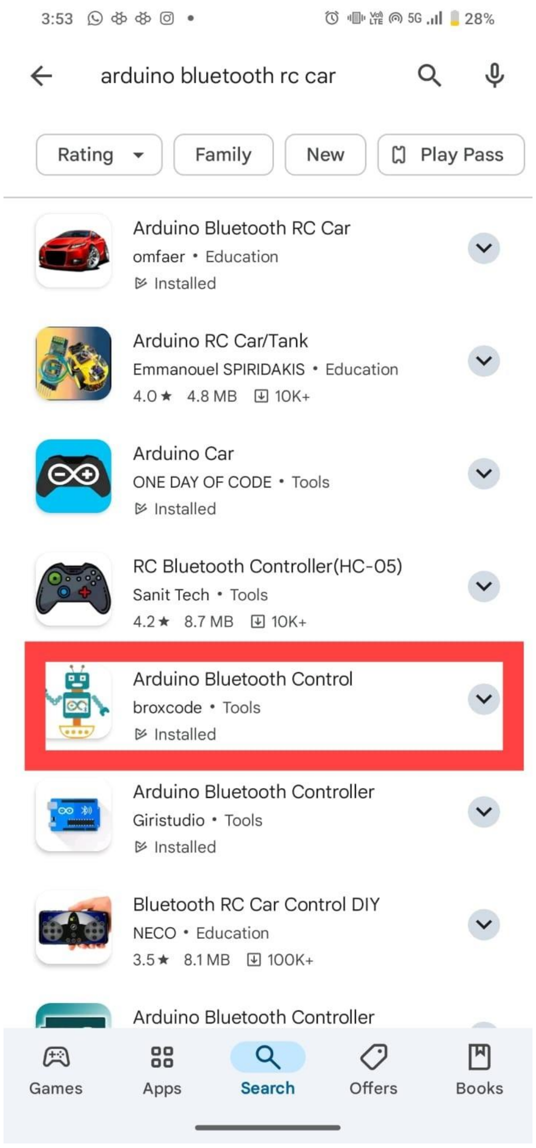


```
  servo.attach(motor);
  M1.setSpeed(Speed);
  M2.setSpeed(Speed);
  M3.setSpeed(Speed);
  M4.setSpeed(Speed);
}
void loop() {
  //Obstacle();
  //Bluetoothcontrol();
  voicecontrol();
}

void Bluetoothcontrol() {
  if (Serial.available() > 0) {
    value = Serial.read();
    Serial.println(value);
  }
  if (value == 'F') {
    forward();
  } else if (value == 'B') {
```



Now, select board and port. After, upload this code to the robot and reconnect the RX and TX jumper wires.

OK, now download and install the app below. Then, follow the steps belowOK, now download and install the app below. Then, follow the steps below .


3:53 28%

arduino bluetooth rc car

Rating Family New Play Pass

Arduino Bluetooth RC Car
omfaer • Education
Installed

Arduino RC Car/Tank
Emmanouel SPIRIDAKIS • Education
4.0★ 4.8 MB 10K+

Arduino Car
ONE DAY OF CODE • Tools
Installed

RC Bluetooth Controller(HC-05)
Sanit Tech • Tools
4.2★ 8.7 MB 10K+

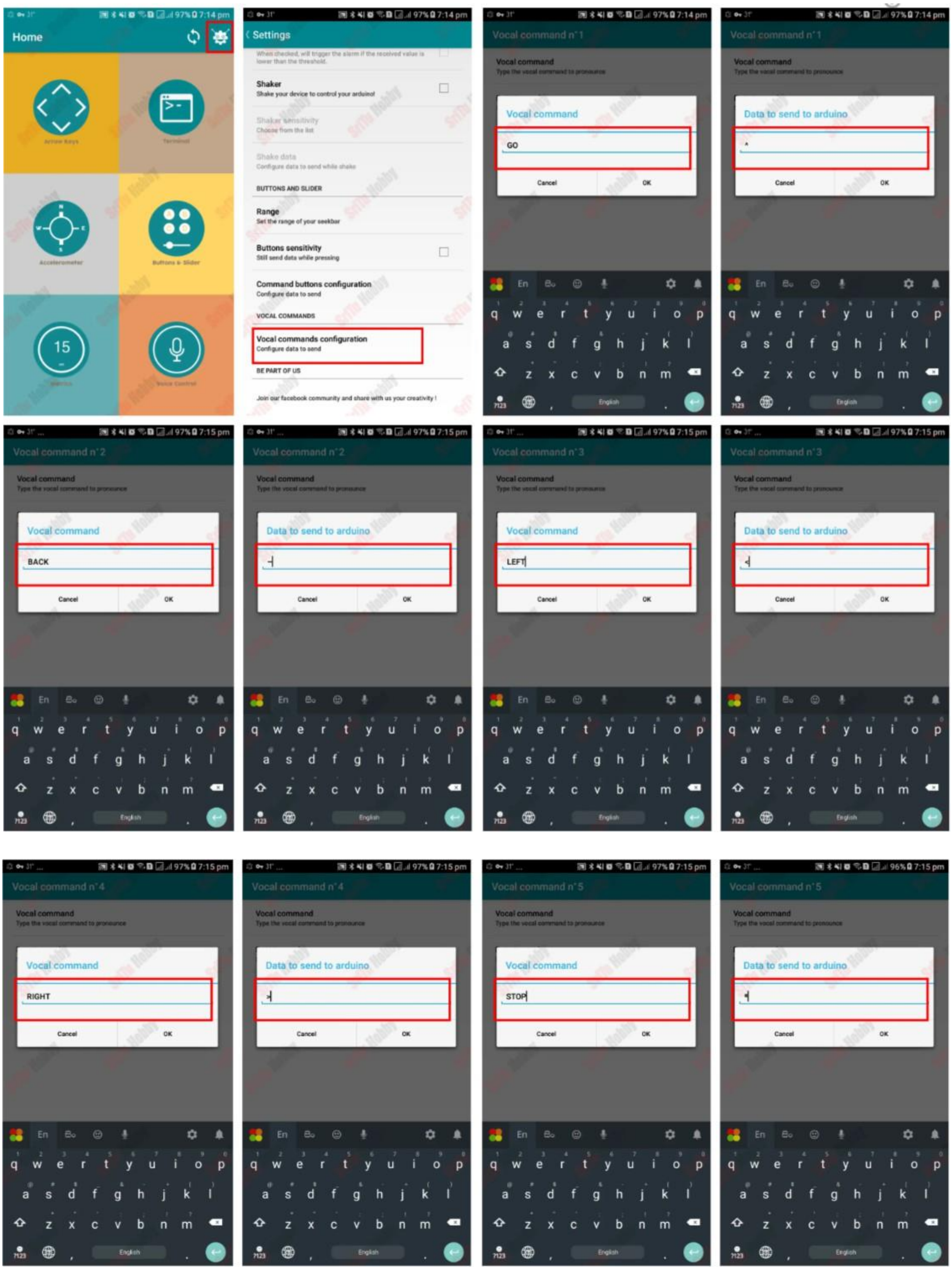
Arduino Bluetooth Control
broxcode • Tools
Installed

Arduino Bluetooth Controller
Giristudio • Tools
Installed

Bluetooth RC Car Control DIY
NECO • Education
3.5★ 8.1 MB 100K+

Arduino Bluetooth Controller

After, run this application and click the setting button. Then, click the “voice commands configuration ” button and include the commands one by one. These are as follows.



Settings

When checked, will trigger the alarm if the received value is lower than the threshold.

Shaker
Shake your device to control your arduino!

Shaker sensitivity
Choose from the list

Shake data
Configure data to send while shake

BUTTONS AND SLIDER

Range
Set the range of your seekbar

Buttons sensitivity
Still send data while pressing

Command buttons configuration
Configure data to send

VOCAL COMMANDS

Vocal commands configuration
Configure data to send

BE PART OF US

Join our facebook community and share with us your creativity !

Vocal command n°1
Vocal command
Type the vocal command to pronounce
Vocal command: GO
Cancel OK

Vocal command n°1
Data to send to arduino: ^
Cancel OK

Vocal command n°2
Vocal command: BACK
Cancel OK

Vocal command n°2
Data to send to arduino: -
Cancel OK

Vocal command n°3
Vocal command: LEFT
Cancel OK

Vocal command n°3
Data to send to arduino: <
Cancel OK

Vocal command n°4
Vocal command: RIGHT
Cancel OK

Vocal command n°4
Data to send to arduino: >
Cancel OK

Vocal command n°5
Vocal command: STOP
Cancel OK

Vocal command n°5
Data to send to arduino: *
Cancel OK

OK, now click the voice control button and enjoy this project.

FUTURE SCOPE AND CONCLUSION

Future Scope

The Smart Helmet Accident Detection System has significant potential for future enhancements and real-world deployment. The current prototype demonstrates accident detection and alert functionality, but it can be further improved with advanced technologies and additional safety features.
Possible Future Improvements

- GPS_Integration
  Adding a GPS module would allow the system to send the rider's live location along with the accident alert message for faster emergency response.

- GSM_Module_Support
  Instead of Wi-Fi dependency, a GSM module could be used to send SMS alerts directly, making the system usable anywhere without internet.

- Real-Time_Health_Monitoring
  Sensors like heart-rate or pulse sensors could be added to monitor the rider's health condition during accidents.

- Mobile_Application_Integration
  A dedicated mobile app could display live helmet status, motion data, and emergency alerts.

- Cloud_Data_Logging
  Accident data can be stored online for analysis, insurance claims, or accident statistics.

- Voice_Alert_System
  Voice prompts could be added to warn the rider about dangerous movements or impacts.

- Automatic_Emergency_Call_Feature
  The system could automatically call emergency contacts if the rider does not respond within a set time after accident detection.

Conclusion

The Smart Helmet Accident Detection System successfully demonstrates how embedded systems and sensors can be used to improve road safety. By integrating an ESP8266 microcontroller, MPU6050 motion sensor, buzzer alert system, and WiFi communication, the helmet is capable of detecting sudden impacts or abnormal motion that may indicate an accident.

When such motion is detected, the system triggers an alarm and sends an alert notification to a predefined contact through an online messaging service. The push button and double-tap feature allow the rider to cancel false alarms, preventing unnecessary alerts.

This project proves that low-cost electronic components can be combined to build an effective safety device. It highlights how IoT technology can play a vital role in saving lives by reducing emergency response time after road accidents.

Overall, the system is:

- Cost-effective
- Reliable
- Expandable
- Practical for real-world implementation

With further development, this smart helmet system can become a commercial safety product for two-wheeler riders.

REFERENCES

References

The following resources were used during the development of this project:

Technical Resources

- ESP8266 WiFi Module Datasheet and Programming Guides
- MPU6050 Sensor Documentation and I2C Communication Reference
- Arduino IDE Official Documentation
- Embedded C Programming References

Online Learning Sources

- Electronics tutorials for MPU6050 interfacing
- IoT communication examples using ESP8266
- Sensor calibration and motion detection tutorials

Components Source
The hardware components such as ESP8266 board, MPU6050 sensor, buzzer, push button, jumper wires, and battery module were obtained from:

- Online electronics suppliers
- Local electronics stores

These references were essential for understanding circuit connections, programming logic, and system testing.

GLOSSARY

ESP8266

A WiFi-enabled microcontroller used to process sensor data and send alerts through the internet.

MPU6050 Sensor

A motion tracking sensor containing a gyroscope and accelerometer used to detect movement, tilt, and sudden impacts.

Accelerometer
A sensor that measures acceleration forces and detects motion or vibration.

Gyroscope
A sensor that measures angular rotation and orientation.

Buzzer
An electronic sound device used to generate alarm alerts.

Push Button
A manual switch used to cancel alarms or control system actions.

I2C Communication
A communication protocol used for data transfer between microcontrollers and sensors.

IoT (Internet of Things)
A system where electronic devices communicate over the internet to send and receive data.

Embedded System
A small computer system designed to perform a specific task within a larger device.