FUTURE GURUKULS

A Step Towards Future

AI & ROBOTICS HAND BOOK

Class 8

SUBROTO CHATTERJEE

(AI & ROBOTICS TRAINER)

FUTURE GURUKULS EDUTECH PRIVATE LIMITED.

IMPORTANT LINKS:

https://www.futuregurukuls.in
https://in.linkedin.com/company/futuregurukuls
https://www.instagram.com/futuregurukuls
https://www.youtube.com/@futuregurukuls
https://kitvit.in

ACKNOWLDGEMENT

I am deeply grateful and honored to present this *AI & Robotics Handbook* designed for students from Class 4 to 12. As a first-time author, this book represents not only my professional experience but also my passion for empowering young minds with practical knowledge in Artificial Intelligence, Robotics, IoT, and emerging technologies.

I sincerely thank **Future Gurukuls Edutech Private Limited** for providing me with the opportunity, platform, and continuous support to develop this comprehensive STEM learning resource. Their vision of delivering quality technical education to students has been a constant source of inspiration throughout this journey.

I also extend my heartfelt gratitude to my students, whose curiosity, enthusiasm, and innovative ideas motivated me to structure this handbook in a simple, practical, and activity-based format. Their questions and creativity helped shape the clarity and approach of this content.

Special appreciation goes to my mentors, colleagues, and well-wishers who encouraged me to take this step as a new author and guided me during the preparation of this handbook.

This book is a sincere effort to make AI and Robotics education accessible, practical, and engaging for school students. I hope it inspires learners to explore, experiment, and innovate in the field of technology.

Subroto Chatterjee
AI & Robotics Trainer
Future Gurukuls Edutech Private Limited

INTRODUCTION

IMPORTANT DEFINITION

(COMMON FOR ALL CLASS)

1- What is a Robot?

Robot is a smart machine which can take decision, it works like a human and makes our works easier.

2- What is IoT?

IoT Stands for *Internet Of Things.*

Connecting objects with internet and controlling it from anywhere around the world is called IoT.

3- What is Computer programming?

Set of instructions or command by which we can communicate with computer, machine or robot is called programming.
There are various types of programming languages like C, C++, JAVA, Python, Scratch, Arduino etc.

4- What is programming language?

A programming language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks.

Example - C. C++, C#, Arduino, Java, Python, Scratch, Cobol etc.

5- What is Input Device?

Such device which collects data from outside and send it inside of any brain, are called input devices.
Ex- IR Sensor, LDR Sensor, Flame Sensor, Ultrasonic Sensor etc.
Basically, all types of sensors are input devices.

6- What are output devices?

Such devices which execute the final task according to the given input or give the final result, we called them output devices.
Ex- LED Module, Buzzer, DC Motor, Servo Motor etc.

7- What are Sensors?

Sensors are the sensing device ***which detect any change in environment*** and send signals to the microcontroller.

Ex- IR Sensor, LDR Sensor, Flame Sensor, Ultrasonic Sensor etc.

Basically, all types of sensors are input devices.

8- What are Actuators?

Actuators are the electronic device ***which makes any change in environment*** by signals from microcontroller.

Ex- LED Module, Buzzer, DC Motor, Servo Motor etc.

Basically, all types of sensors are Output devices.

9- What is Microcontroller?

A **Microcontroller** is a small computer on a single chip that is used to ***control devices and machines***. It has a **processor, memory, and input/output pins** inside one chip.

Ex- Arduino, ESP8266 / ESP32, Raspberry Pi etc.

10- What is AI?

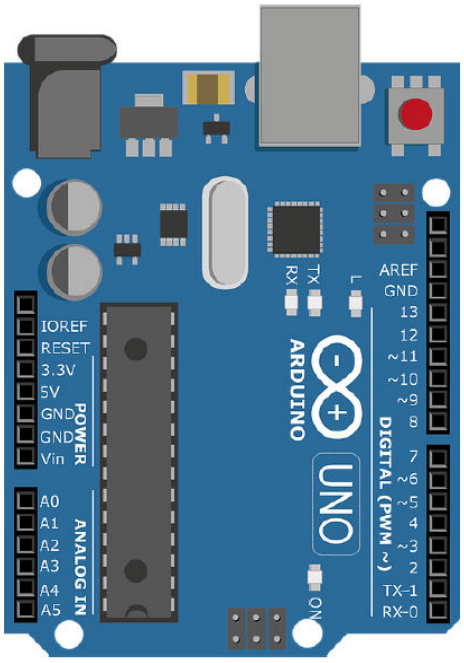
AI Stands for *Artificial Intelligence* Machines That Think

AI means *making machines smart* so they can *think and learn like humans*. It helps computers and robots to *learn from data and past experiences*. AI can see patterns, make decisions, and understand language.

ACTIVITIES FOR CLASS 8		
ACTIVITIES:	**CURRICULUM ACTIVITIES**	**PAGE**
ACTIVITY 1:	About Arduino, Analog Value of Sensor – MQ Series Gas Detection System	6
ACTIVITY 2:	IR Sensor & Line Follower Robot	11
ACTIVITY 3:	Advanced IoT – Wi-Fi-Controlled Car	17
ACTIVITY 4:	AI + IoT = AIoT – Smart Helmet Integrated with Bike	29
ACTIVITY 5:	Concept of Satellites & Simulated Satellite Launch	38
ACTIVITIES:	**EXTRA ACTIVITIES**	**PAGE**
ACTIVITY 1:	Introduction to Arduino - LED Blinking	
ACTIVITY 2:	Multiple GPIO Control	
ACTIVITY 3:	Servo Motor Control (Smart Dustbin)	
ACTIVITY 4:	Welcome Robot	
ACTIVITY 5:	Prosthetic Hand	
ACTIVITY 6:	Smart Bridge	

ACTIVITY 1

About Arduino, Analog Value of Sensor – MQ Series Gas Detection System



What is Arduino UNO?

Arduino uno is a programmable board which works like a artificial brain. We can call Arduino a computer for physical world. With the help of this board one can build 1000s of projects.

- **DC Power Jack:**
  Used to provide power to Arduino.
- **USB Jack:**
  Used to upload code and provide power supply.

Pins of Arduino UNO

Pin Name	Pin Number/Range	Total Pins
Digital Input/output Pin (GPIO PIN)	0 to 13 pin	14 pins
Analog Output Pin (PWM ~)	3,5,6,9,10,11 pin	6 pins
Analog Input Pin	A0 to A5 pin	6 pin
Power Pin	Vin, 5V, 3.3V	3 pin
Ground Pin	GND	3 pin

GPIO Stands for :
G - General.
P - Purpose.
I - Input.
O - Output.

PWM Stands for :
P - Pulse.
W - Width
M - Modulation

INPUT

- pinMode(pin, INPUT);
- pinMode(pin, INPUT_PULLUP);
- digitalRead(pin)
- Returns 0 or 1

Example:
Press and release pushbutton

DIGITAL

OUTPUT

- pinMode(pin, OUTPUT);
- digitalWrite(pin, 0 or 1)
- Write 0 or 1

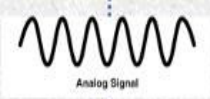
Example:
Turn the LED on and OFF

INPUT

- analogRead(pin)
- Returns 0 - 1023

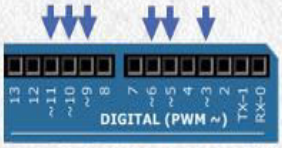
Example:
Rotate the rod of the potentiometer

ANALOG



OUTPUT

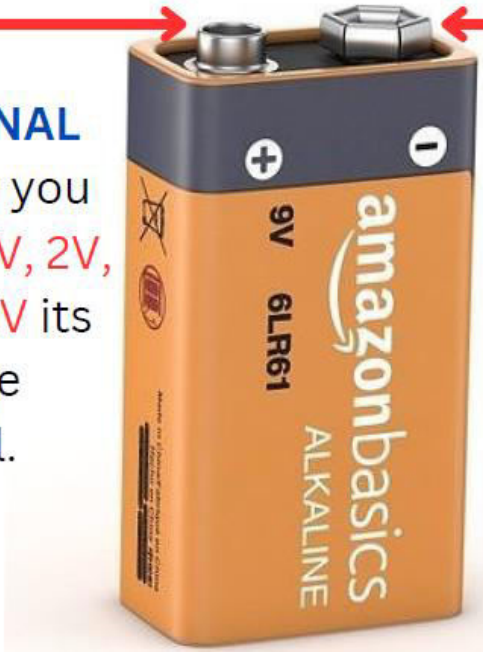
- analogWrite(pin, 0-255)
- Write 0 - 255



Example:
Fade the LED's brightness on and off

POSITIVE TERMINAL

Note - Wherever you see +, Vin, Vcc, 1V, 2V, 3V, 4V, 5V..... NV its clarly indicate the Positive Terminal.

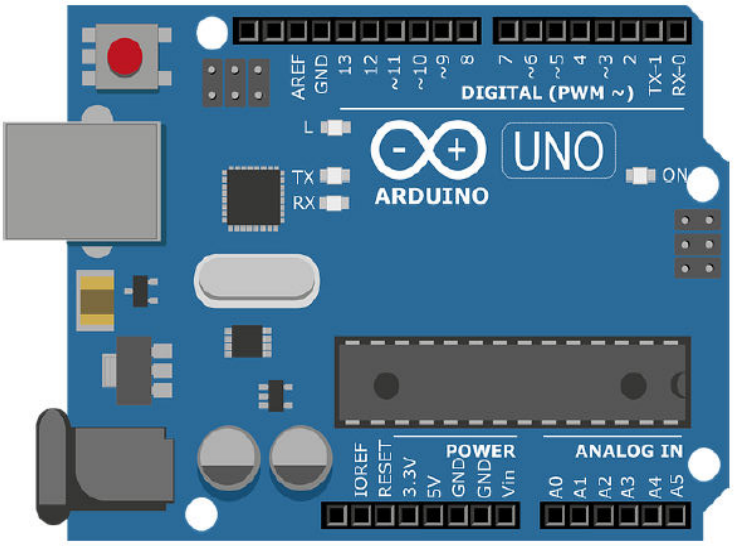


NEGATIVE TERMINAL

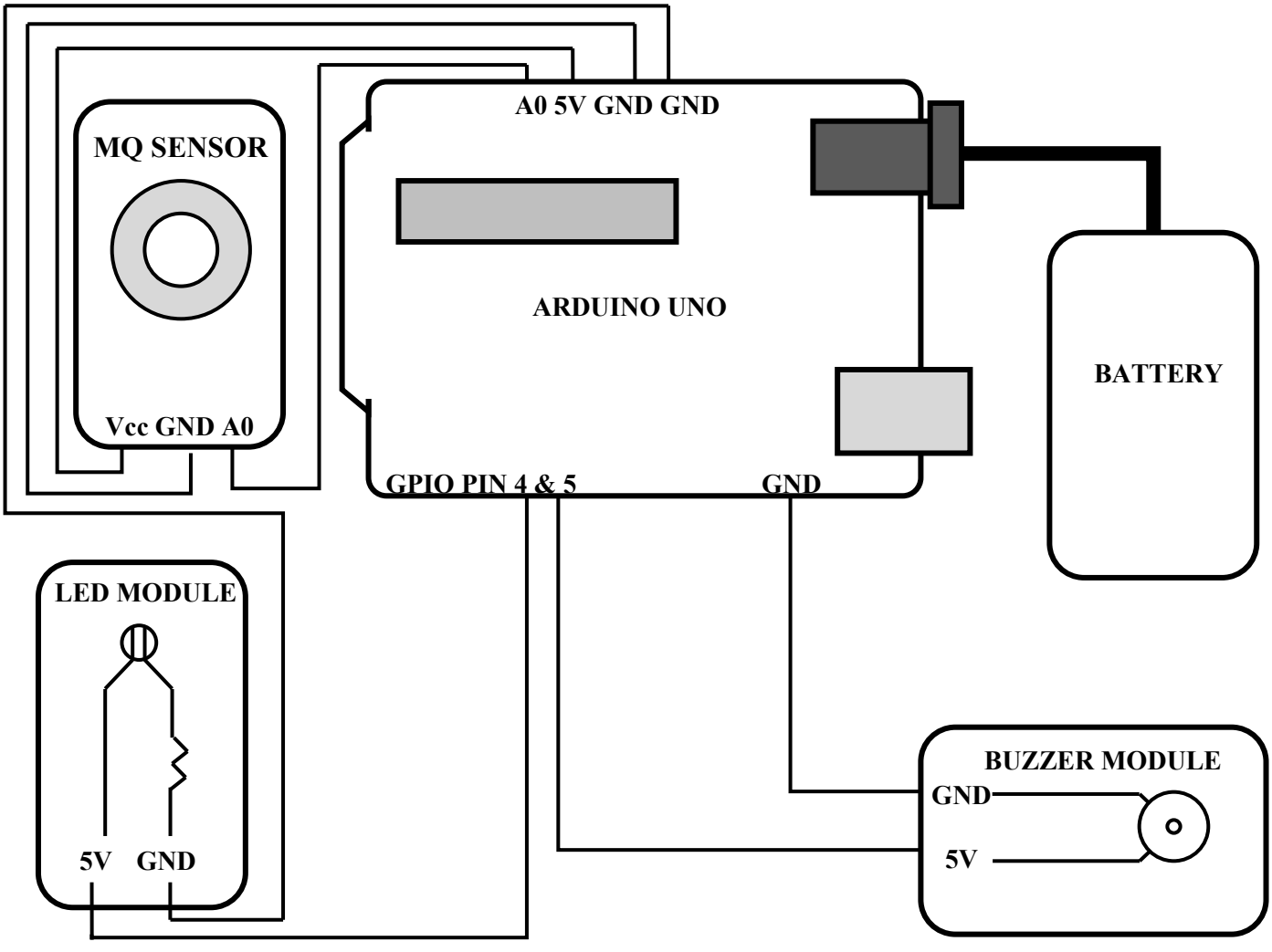
Note - Wherever you see -, GND, Ground its clarly indicate the Negative Terminal.

REQUIRED COMPONENTS:

1. Arduino Uno
2. LED Module
3. Buzzer Module
4. MQ Sensor
5. 1-Pin Jumper Wires
6. Battery
7. Battery Cap



CONNECTION DIAGRAM OF MQ SERIES GAS DETECTION SYSTEM:



PROGRAM CODE OF MQ SERIES GAS DETECTION SYSTEM:

```
// Define the pins we will use
const int sensorPin = A0;      // The analog pin connected
to the sensor's A0 pin
const int warningPin = 13;     // The digital pin connected
to the LED/buzzer
const int LedPin = 10;     // The digital pin connected to
the LED/buzzer
// Define the threshold value. You can change this later!
// This is the point where we decide the gas level is too
high.
const int gasThreshold = 400;

void setup() {
  // Set the warning pin as an output
  pinMode(warningPin, OUTPUT);

  // Start the serial communication so we can see the
sensor values
  // 9600 is the communication speed (baud rate)
  Serial.begin(9600);
}

void loop() {
  // Read the analog value from the sensor
  // The value will be between 0 and 1023
  int sensorValue = analogRead(sensorPin);
```

```
  // Print the value to the Serial Monitor so we can see it
  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);

  // Check if the sensor value is above our threshold
  if (sensorValue > gasThreshold) {
    // If it is, turn on the warning LED/buzzer
    Serial.println("GAS DETECTED! WARNING!");
    digitalWrite(warningPin, HIGH);
    digitalWrite(LedPin, HIGH);
    delay(1000);
    digitalWrite(warningPin, LOW);
    digitalWrite(LedPin, LOW);
    delay(1000);
  } else {
    // If it's not, keep the warning LED/buzzer off
    digitalWrite(warningPin, LOW);
  }

  // Wait a little bit before checking again
  delay(100);
}
```

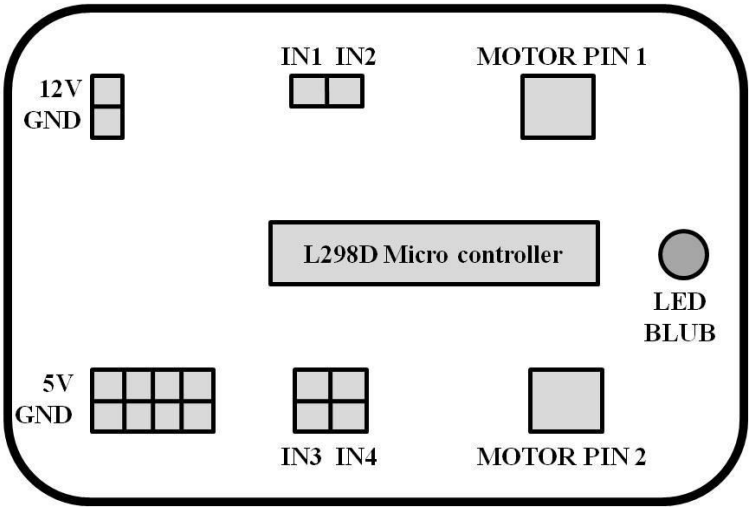
ACTIVITY 2

IR Sensor & Line Follower Robot

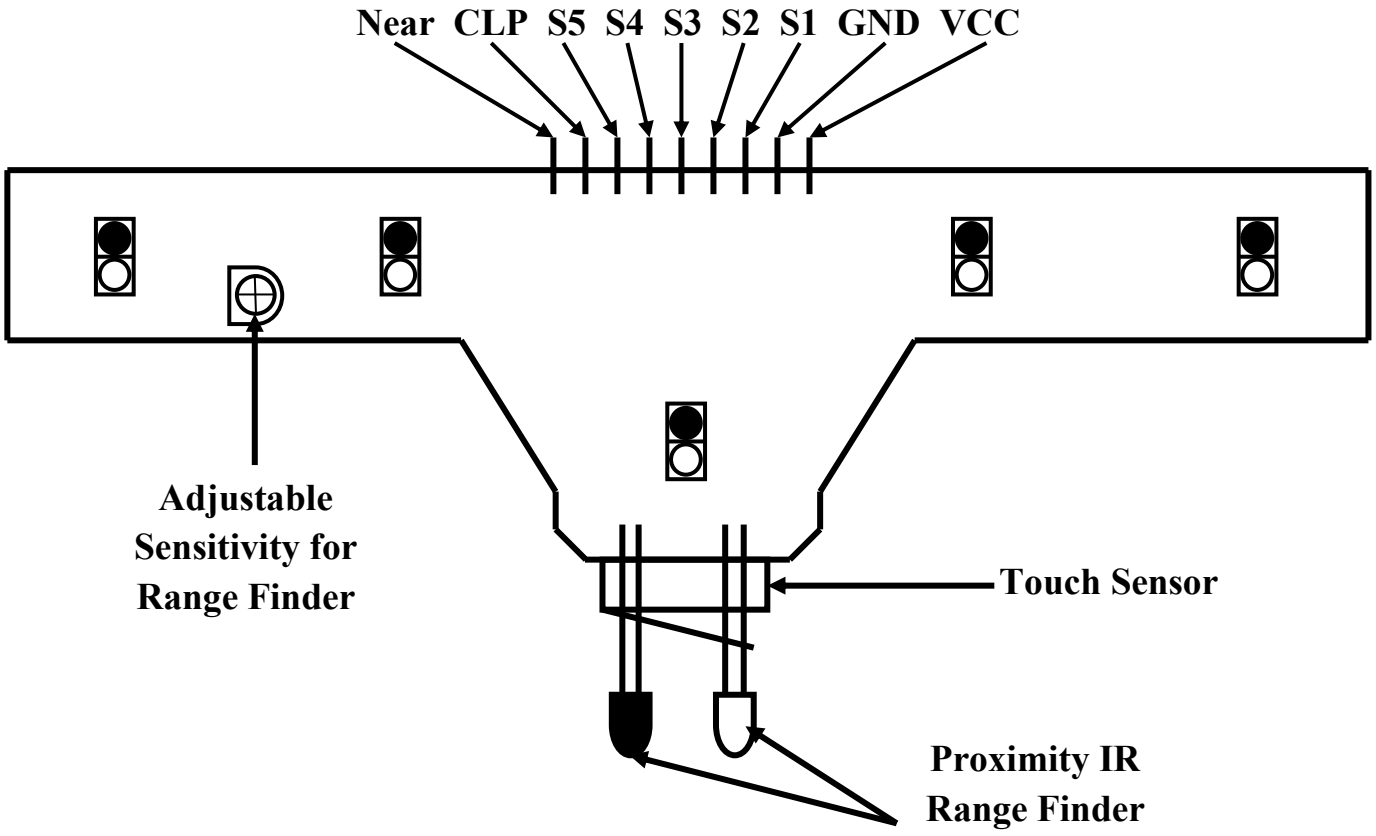
REQUIRED COMPONENTS:

1. Car Frame
2. Wheels
3. Castor Wheel
4. Screw & Nuts
5. Geared Motors
6. Motor Driver L298D
7. Power Board
8. 0 & 1-PIN Jumper Wire
9. Rechargeable Battery
10. Arduino Uno
11. 5-Array IR Sensor

Diagram of Motor Driver



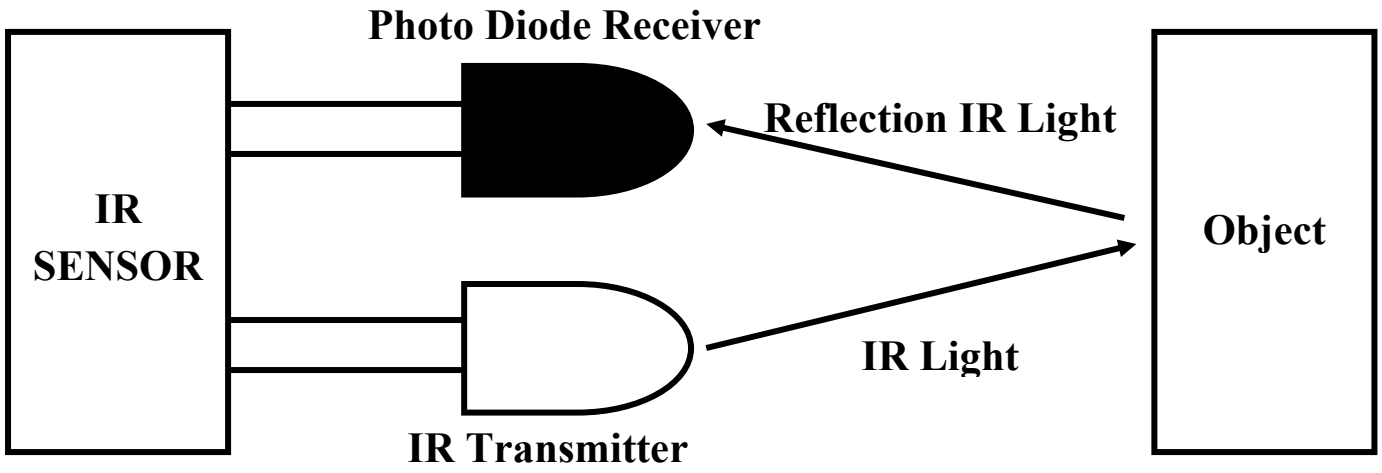
5-Array IR Sensor Diagram:



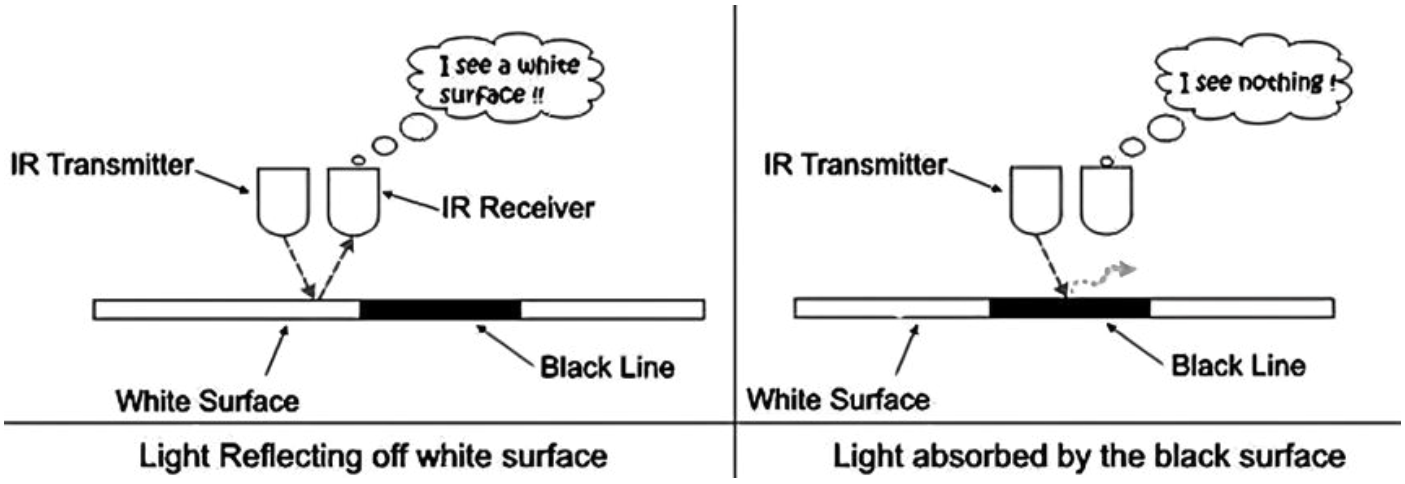
Working of IR (Infra-Red) Sensor:

What is an IR Sensor?

IR sensor is an electronic device that emits the light in order to sense some object of the surroundings. An **IR sensor** can measure the heat of an object as well as detects the motion. Usually, in the **infrared spectrum**, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations.

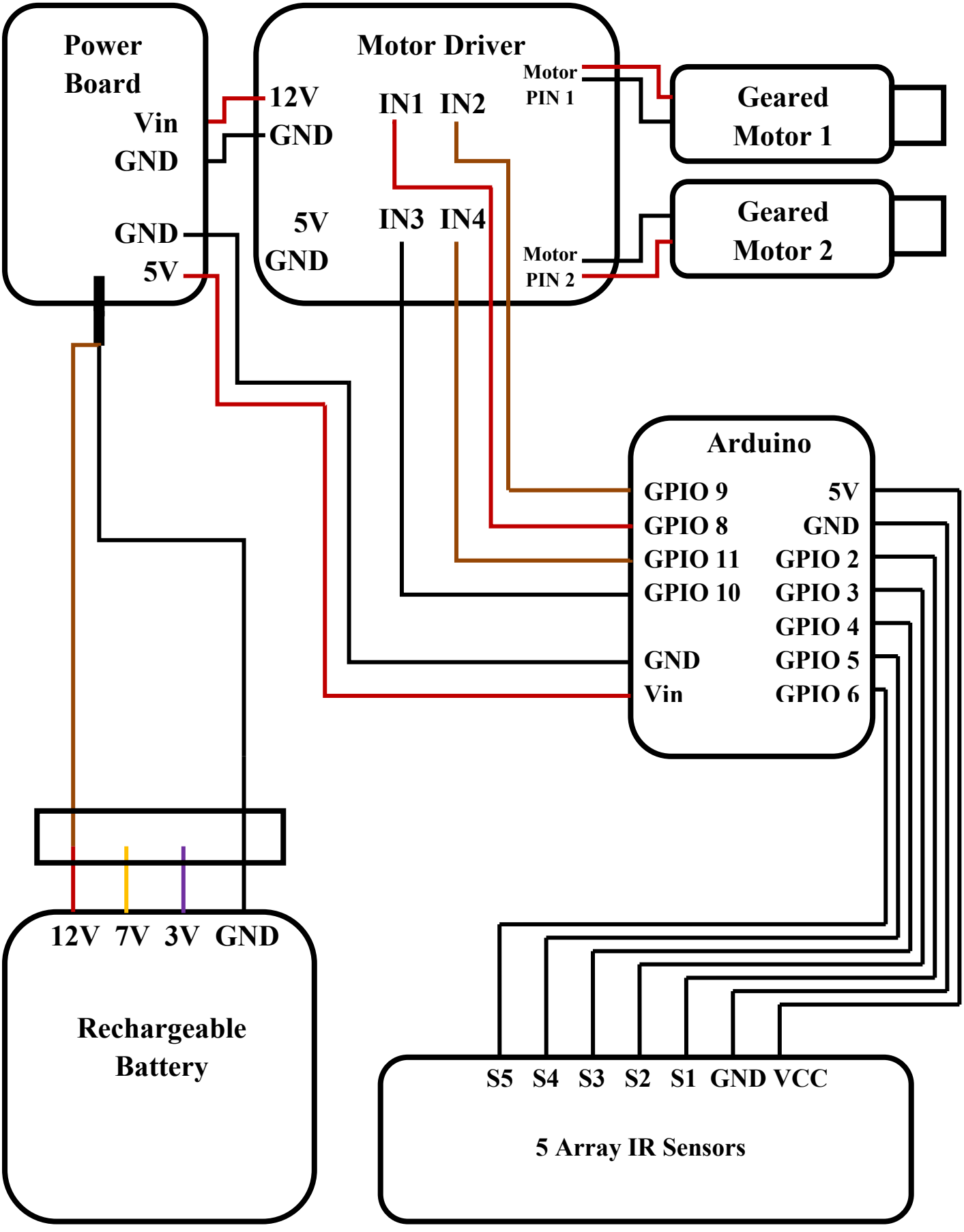


IR reflective sensors have emitter (IR LED) and receiver (Photo-Transistor or photo diode). If white surface is present beneath the IR LED or if there is any object, IR rays are reflected and are sensed by the receiver, while in case of no object receiver does not sense IR rays and same is in case of black surface as light gets absorbed and hence receiver does not sense IR rays.



Signal at **White Surface is 1(one)** and Signal at **Black Surface is 0(zero)**.

CONNECTION DIAGRAM OF LINE FOLLOWER ROBOT



PROGRAM CODE OF LINE FOLLOWER ROBOT

```
// ===== Line Follower Robot (5 IR Sensor + 2 Motor) =====
// Motor driver pins
#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 11
// IR sensor pins
#define S1 2
#define S2 3
#define S3 4
#define S4 5
#define S5 6

void setup() {
  // Motor pins
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  // IR sensor pins
  pinMode(S1, INPUT);
  pinMode(S2, INPUT);
  pinMode(S3, INPUT);
  pinMode(S4, INPUT);
  pinMode(S5, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Read sensor values (0 = black, 1 = white)
  int s1 = digitalRead(S1);
  int s2 = digitalRead(S2);
  int s3 = digitalRead(S3);
  int s4 = digitalRead(S4);
  int s5 = digitalRead(S5);
```

```
  Serial.print(s1);
  Serial.print(" ");
  Serial.print(s2);
  Serial.print(" ");
  Serial.print(s3);
  Serial.print(" ");
  Serial.print(s4);
  Serial.print(" ");
  Serial.println(s5);

  // ===== Movement Logic =====
  // Case 1: Center on line
  if ((s2 == 0 && s3 == 0 && s4 == 0) || (s3 == 0)) {
    moveForward();
  }
  // Case 2: Slight Left
  else if (s1 == 0 || s2 == 0) {
    turnLeft();
  }
  // Case 3: Slight Right
  else if (s4 == 0 || s5 == 0) {
    turnRight();
  }
  // Case 4: Lost line (All white)
  else {
    stopCar();
  }

  delay(10);
}

// ===== Movement Functions =====
void moveForward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
```

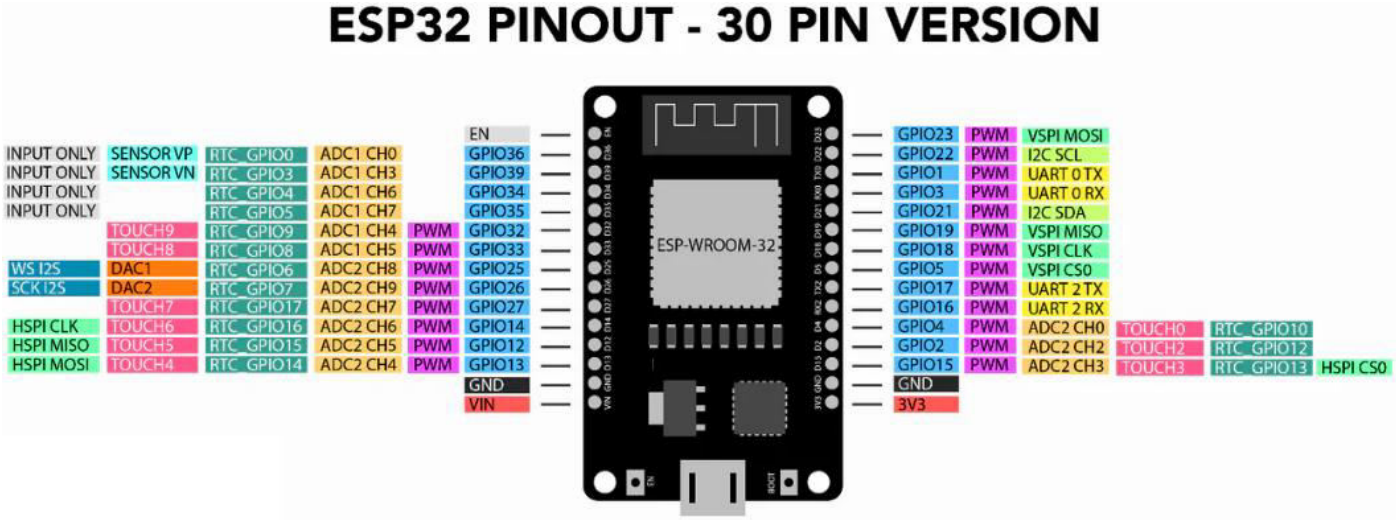
```
void turnLeft() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

void turnRight() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void stopCar() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
```

ACTIVITY 3

Advanced IoT – Wi-Fi-Controlled Car



ESP32 DEV MODULE – COMPLETE PIN DETAILS (TABULAR FORMAT)

Board Pin (Written on Board)	GPIO No. (Full Form)	Pin Name / Type	Main Function	Other Functions (If Any)	Important Notes / Details
VIN	—	Power Input	External power input (5V)	—	Can power ESP32 from 5V source (regulated on board to 3.3V).
GND	—	Ground	Common ground	—	Must be connected to ground of circuit.
3V3	—	Power Output	Provides 3.3V	—	Output from onboard regulator, max ~600mA (varies).
EN	—	Enable Pin (Enable Reset)	Chip Enable	Reset when pulled LOW	Must be HIGH for ESP32 to run.
VP (GPIO36)	GPIO36	Input Only	ADC1 Channel 0	Hall Sensor Input	Cannot be used as output.
VN (GPIO39)	GPIO39	Input Only	ADC1 Channel 3	Hall Sensor Input	Input only, no pull-up.
D34 (GPIO34)	GPIO34	Input Only	Digital Input	ADC1 Channel 6	Cannot output signal.
D35 (GPIO35)	GPIO35	Input Only	Digital Input	ADC1 Channel 7	Input only.

Board Pin (Written on Board)	GPIO No. (Full Form)	Pin Name / Type	Main Function	Other Functions (If Any)	Important Notes / Details
D32 (GPIO32)	GPIO32	I/O Pin	Digital I/O	Touch Sensor T9, ADC1 Ch 4	Safe for most uses.
D33 (GPIO33)	GPIO33	I/O Pin	Digital I/O	Touch Sensor T8, ADC1 Ch 5	Can be used for LED, sensor, etc.
D25 (GPIO25)	GPIO25	I/O Pin	Digital I/O	DAC1 (Digital to Analog Converter)	Can generate analog voltage.
D26 (GPIO26)	GPIO26	I/O Pin	Digital I/O	DAC2, ADC2 Ch 9	Good for audio output.
D27 (GPIO27)	GPIO27	I/O Pin	Digital I/O	Touch Sensor T7, ADC2 Ch 7	Can be used normally.
D14 (GPIO14)	GPIO14	I/O Pin	Digital I/O	Touch T6, ADC2 Ch 6	Also supports SPI clock in some modes.
D12 (GPIO12)	GPIO12	I/O Pin	Digital I/O	Touch T5, ADC2 Ch 5	⚠ Boot-strapping pin (avoid critical use).
D13 (GPIO13)	GPIO13	I/O Pin	Digital I/O	Touch T4, ADC2 Ch 4	Normal use OK.
D9 (GPIO9)	GPIO9	Flash SPI (Do not use)	Connected to internal Flash	—	❌ Do NOT use.
D10 (GPIO10)	GPIO10	Flash SPI (Do not use)	Connected to internal Flash	—	❌ Do NOT use.
D11 (GPIO11)	GPIO11	Flash SPI (Do not use)	Connected to internal Flash	—	❌ Do NOT use.
D8 (GPIO8)	GPIO8	Flash SPI (Do not use)	Connected to internal Flash	—	❌ Do NOT use.
D7 (GPIO7)	GPIO7	Flash SPI (Do not use)	Connected to internal Flash	—	❌ Do NOT use.
D6 (GPIO6)	GPIO6	Flash SPI (Do not use)	Connected to internal Flash	—	❌ Do NOT use.

Board Pin (Written on Board)	GPIO No. (Full Form)	Pin Name / Type	Main Function	Other Functions (If Any)	Important Notes / Details
D5 (GPIO5)	GPIO5	I/O Pin	Digital I/O	VSPI CS (Chip Select)	Safe pin for LED.
D4 (GPIO4)	GPIO4	I/O Pin	Digital I/O	Touch T0, ADC2 Ch 0	Commonly used for sensors.
D0 (GPIO0)	GPIO0	Boot Button / I/O	Must be LOW for flashing	Touch T1, ADC2 Ch 1	Press BOOT while uploading code.
D2 (GPIO2)	GPIO2	I/O + Onboard LED	Controls onboard LED	Touch T2, ADC2 Ch 2	LED is active LOW.
D15 (GPIO15)	GPIO15	I/O Pin	Digital I/O	Touch T3, ADC2 Ch 3	Boot-strapping pin (be careful).
RX2 (GPIO16)	GPIO16	UART2 RX	Serial Receive	—	Used for Serial2 communication.
TX2 (GPIO17)	GPIO17	UART2 TX	Serial Transmit	—	Used for Serial2 communication.
RX0 (GPIO3)	GPIO3	UART0 RX	USB Serial Receive	—	Used by Serial Monitor.
TX0 (GPIO1)	GPIO1	UART0 TX	USB Serial Transmit	—	Used by Serial Monitor.
D22 (GPIO22)	GPIO22	I/O Pin	I2C SCL (Clock)	General I/O	Common for I2C sensors.
D21 (GPIO21)	GPIO21	I/O Pin	I2C SDA (Data)	General I/O	Common for I2C sensors.
D19 (GPIO19)	GPIO19	I/O Pin	VSPI MISO	General I/O	SPI communication.
D18 (GPIO18)	GPIO18	I/O Pin	VSPI SCK (Clock)	General I/O	SPI communication.
D23 (GPIO23)	GPIO23	I/O Pin	VSPI MOSI	General I/O	SPI communication.

Important Terms & Their Full Forms (Easy Explanation)

Term	Full Form	Meaning (Simple)
GPIO	General Purpose Input Output	Pins that can act as input or output.
ADC	Analog to Digital Converter	Converts analog voltage to digital value.
DAC	Digital to Analog Converter	Converts digital signal to analog voltage.
SPI	Serial Peripheral Interface	High-speed communication protocol.
I2C	Inter-Integrated Circuit	Two-wire communication (SDA, SCL).
UART	Universal Asynchronous Receiver Transmitter	Serial communication (TX, RX).
VIN	Voltage Input	External power supply input.
GND	Ground	0V reference.
EN	Enable	Turns ESP32 ON/OFF.
BOOT	Boot Mode	Used while uploading code.

Best Pins to Use in Projects (Recommended)

Use these safely:

- D2, D4, D5, D13, D14, D18, D19, D21, D22, D23, D25, D26, D27, D32, D33

Avoid using (if possible):

- GPIO0, GPIO2, GPIO12, GPIO15 (Boot-related pins)
- GPIO6–GPIO11 (Flash pins)

ESP32 – Explanation (with Full Form)

Full Form of ESP32:

Espressif Systems Processor 32-bit

(It is commonly just called **ESP32**.)

What is ESP32?

ESP32 is a **microcontroller + Wi-Fi + Bluetooth module** made by **Espressif Systems (a company in China).**

It is used in:

- IoT (Internet of Things) projects
- Robotics
- Home automation
- Smart devices
- Wireless communication projects

Main Features of ESP32

Feature	Explanation
Processor	Dual-core 32-bit CPU (faster than Arduino)
Clock Speed	Up to 240 MHz
Wi-Fi	Yes (Built-in)
Bluetooth	Yes (Classic + BLE)
GPIO Pins	Around 30
ADC Pins	Many analog input pins
DAC Pins	2 (for audio, etc.)
Uses	IoT, automation, robotics, smart systems
Programming	Arduino IDE, ESP-IDF, MicroPython

Why is ESP32 Powerful?

Because it has:

- More speed than Arduino
- Built-in Wi-Fi
- Built-in Bluetooth
- More memory
- More pins and features

NodeMCU

Full Form of NodeMCU:

Node MicroController Unit

What is NodeMCU?

NodeMCU is **not a microcontroller itself** — it is a **development board** that uses the **ESP8266 chip**.

So **NodeMCU = Board + ESP8266 chip + USB + voltage regulator**

It is mainly used for **Wi-Fi based IoT projects.**

Main Features of NodeMCU

Feature	Explanation
Microcontroller Chip	ESP8266
Wi-Fi	Yes (Built-in)
Bluetooth	❌ No
GPIO Pins	About 15
ADC Pins	Only 1
Speed	Slower than ESP32
Programming	Arduino IDE, Lua
Uses	Basic IoT projects

What is an MCU? (Simple Definition)

An **MCU (Microcontroller Unit)** is a **small computer on a single chip** that is designed to control electronic devices.

It contains:

- Processor (CPU)
- Memory (RAM + Flash)
- Input/Output (I/O) pins all inside **one single integrated circuit (IC).**

ESP32 vs NodeMCU

Feature	ESP32	NodeMCU (ESP8266)
Company	Espressif	Espressif
Wi-Fi	✅ Yes	✅ Yes
Bluetooth	✅ Yes	❌ No
Processor	Dual-core	Single-core
Speed	Faster	Slower
GPIO Pins	More (~30)	Less (~15)
ADC Pins	Many	Only 1
Best for	Advanced IoT	Simple IoT

Comparison between NodeMCU (ESP32 Dev 30-Pin) & Arduino Uno R3:

Feature	NodeMCU (ESP32 Dev 30-Pin)	Arduino Uno R3
Microcontroller	ESP32	ATmega328P
Board Name	ESP32 Dev Module	Arduino Uno R3
Architecture	32-bit Dual Core	8-bit
Clock Speed	Up to 240 MHz	16 MHz
Operating Voltage	3.3V	5V
Input Voltage (Recommended)	5V (via USB) or 5–9V (Vin)	7–12V
Digital I/O Pins	~30 GPIO (some restricted)	14
Analog Input Pins	18 (ADC)	6
Analog Output (DAC)	2 DAC pins	No true DAC
PWM Pins	Almost all GPIO	6
WiFi	✅ Built-in	❌ External module required
Bluetooth	✅ Built-in	❌ Not available
SRAM	520 KB	2 KB
Flash Memory	4 MB (typical)	32 KB
USB-to-Serial	CP2102 / CH340	ATmega16U2

Feature	NodeMCU (ESP32 Dev 30-Pin)	Arduino Uno R3
Programming Language	Arduino C/C++, MicroPython	Arduino C/C++
Power Consumption	Higher	Lower
Price	Usually cheaper	Usually slightly higher

REQUIRED COMPONENTS:

1. Car Frame
2. Wheels
3. Castor Wheel
4. Screw & Nuts
5. Geared Motors
6. Motor Driver L298D
7. Power Board
8. 0 & 1-PIN Jumper Wire
9. Rechargeable Battery
10. ESP 32

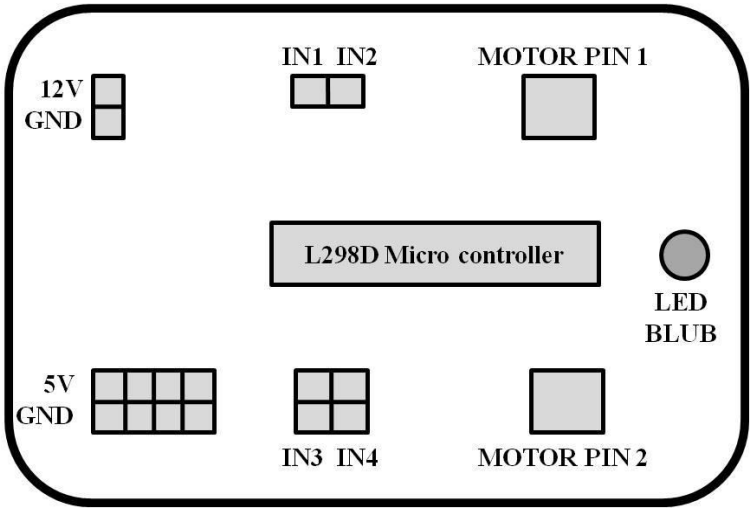
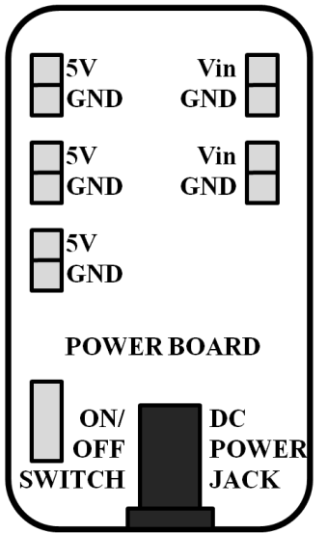


Diagram of Motor Driver

Motor Driver L298D:

A motor driver is an electronic circuit or module that controls the speed, direction, and other parameters of an electric motor.

Diagram of Power Board



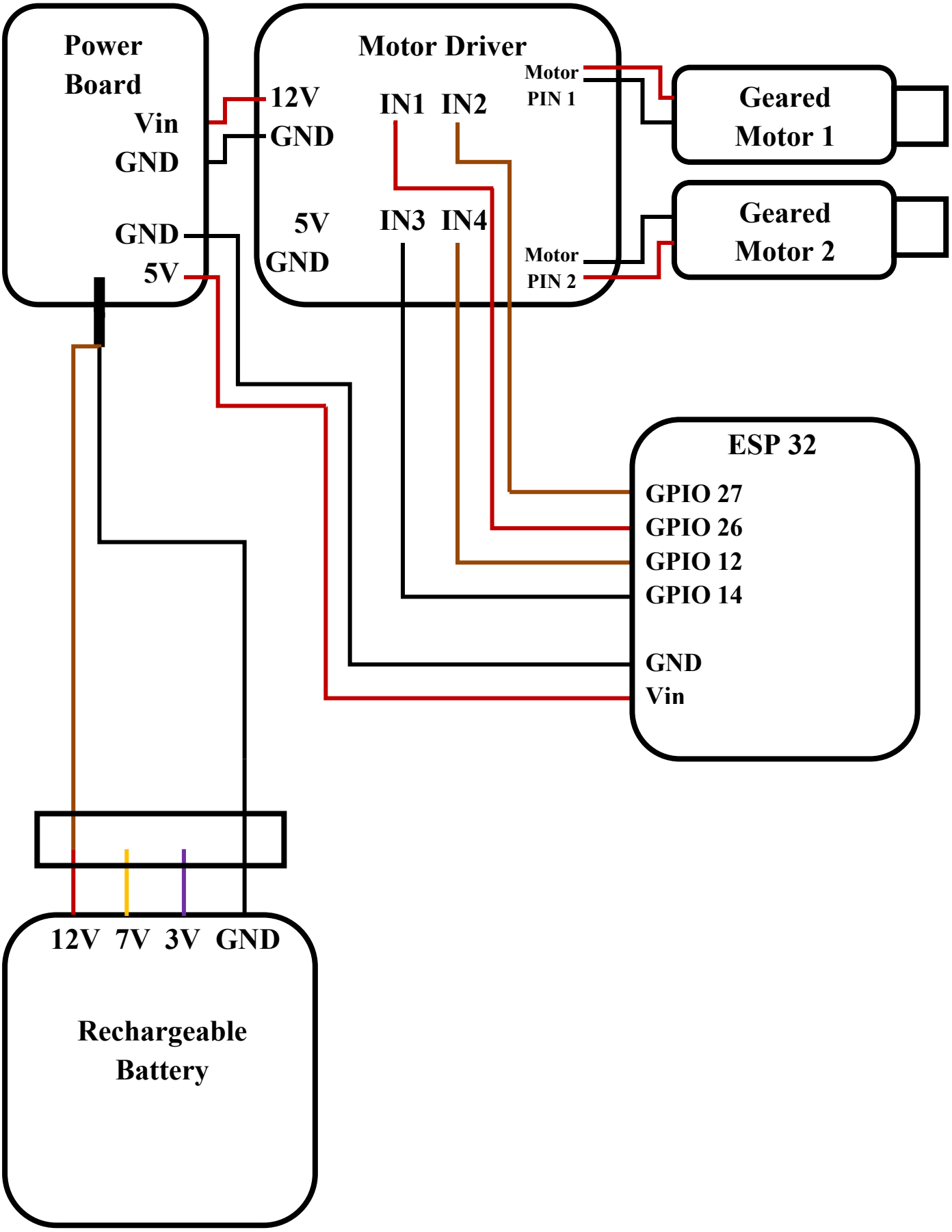
Power Board:

A power board is an electronic circuit or module that distributes the power supply to all components from battery. It can handle maximum supply of 12V.

Power board has pins:

- Two: Vin pins
- Three: 5V pins
- Five: GND pins

CONNECTION DIAGRAM OF WI-FI-CONTROLLED CAR



PROGRAM CODE OF WI-FI-CONTROLLED CAR

```
#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "ESP32_CAR";       // WiFi name
const char* password = "12345678";    // WiFi password

WebServer server(80);

// Motor Pins
#define IN1 26
#define IN2 27
#define IN3 14
#define IN4 12

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

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  WiFi.softAP(ssid, password);
  Serial.println("WiFi Started");
  Serial.println(WiFi.softAPIP());

  server.on("/", handleRoot);
  server.on("/forward", forward);
  server.on("/backward", backward);
  server.on("/left", left);
  server.on("/right", right);
  server.on("/stop", stop);

  server.begin();
}
```

```
void loop() {
  server.handleClient();
}

void handleRoot() {
  String page = "<h1>ESP32 WiFi Car</h1>"
                "<a
href='/forward'><button>FORWARD</button></a><br><br>"
                "<a
href='/backward'><button>BACKWARD</button></a><br><br>"
                "<a
href='/left'><button>LEFT</button></a><br><br>"
                "<a
href='/right'><button>RIGHT</button></a><br><br>"
                "<a
href='/stop'><button>STOP</button></a>";

  server.send(200, "text/html", page);
}

void forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  server.send(200, "text/html", "Forward");
}

void backward() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  server.send(200, "text/html", "Backward");
}
```

```
void left() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  server.send(200, "text/html", "Left");
}

void right() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  server.send(200, "text/html", "Right");
}

void stop() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  server.send(200, "text/html", "Stopped");
}
```

ACTIVITY 4

AI + IoT = AIoT – Smart Helmet Integrated with Bike

REQUIRED COMPONENTS FOR HELMET:

1. ESP32
2. Ulrasonic Sensor
3. MQ 2
4. Power Board
5. 1-PIN Jumper Wire
6. Battery

REQUIRED COMPONENTS FOR CAR:

1. Car Frame
2. Wheels
3. Castor Wheel
4. Screw & Nuts
5. Geared Motors
6. Motor Driver L298D
7. Power Board
8. 0 & 1-PIN Jumper Wire
9. Rechargeable Battery
10. ESP 32

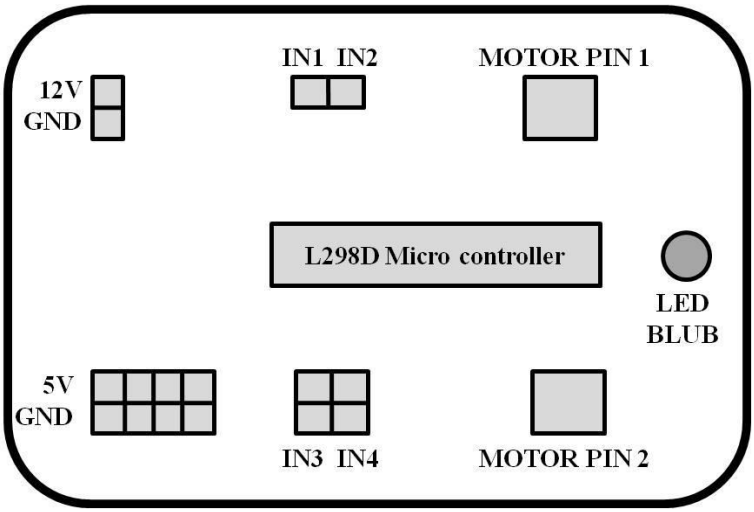
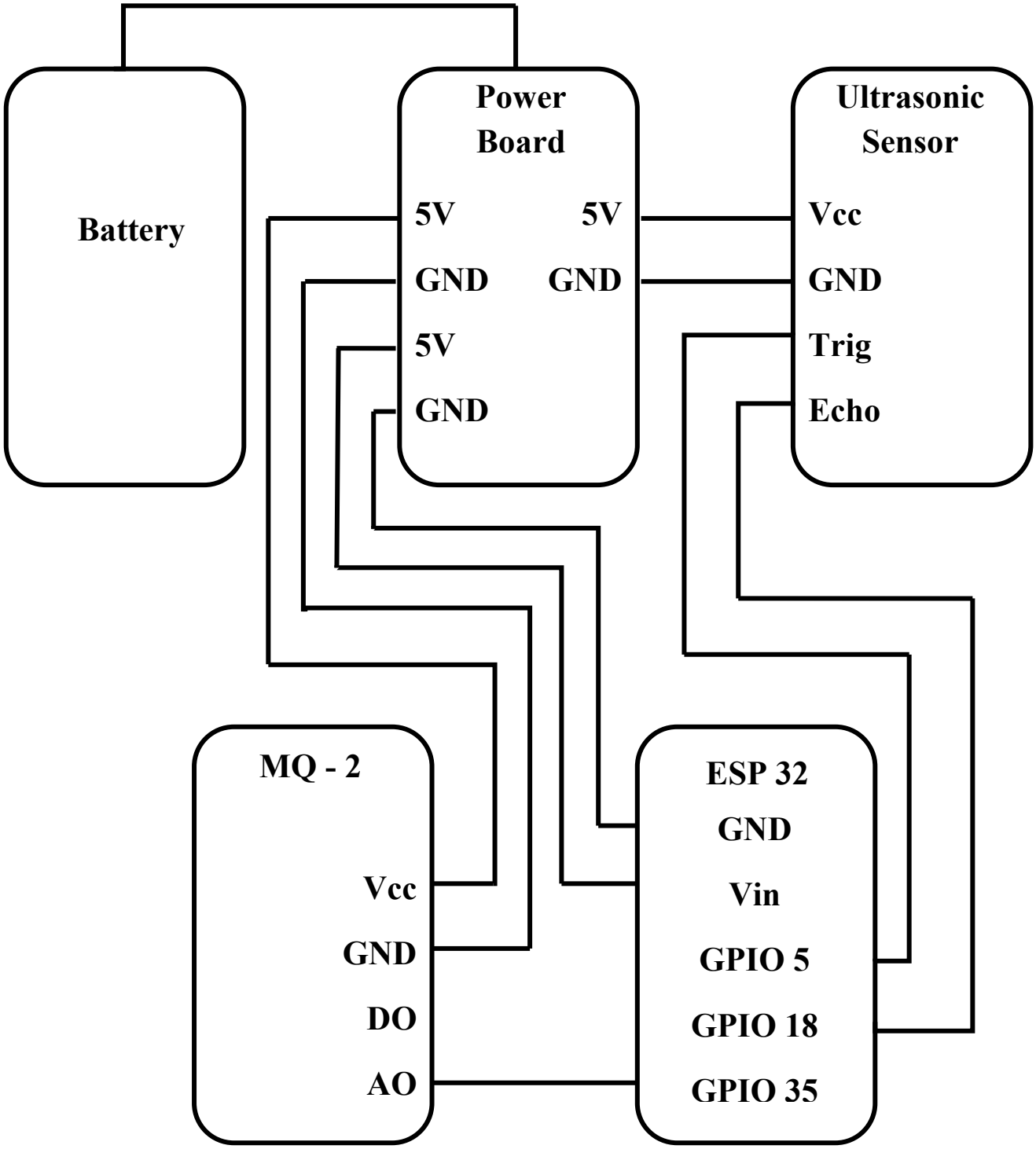


Diagram of Motor Driver

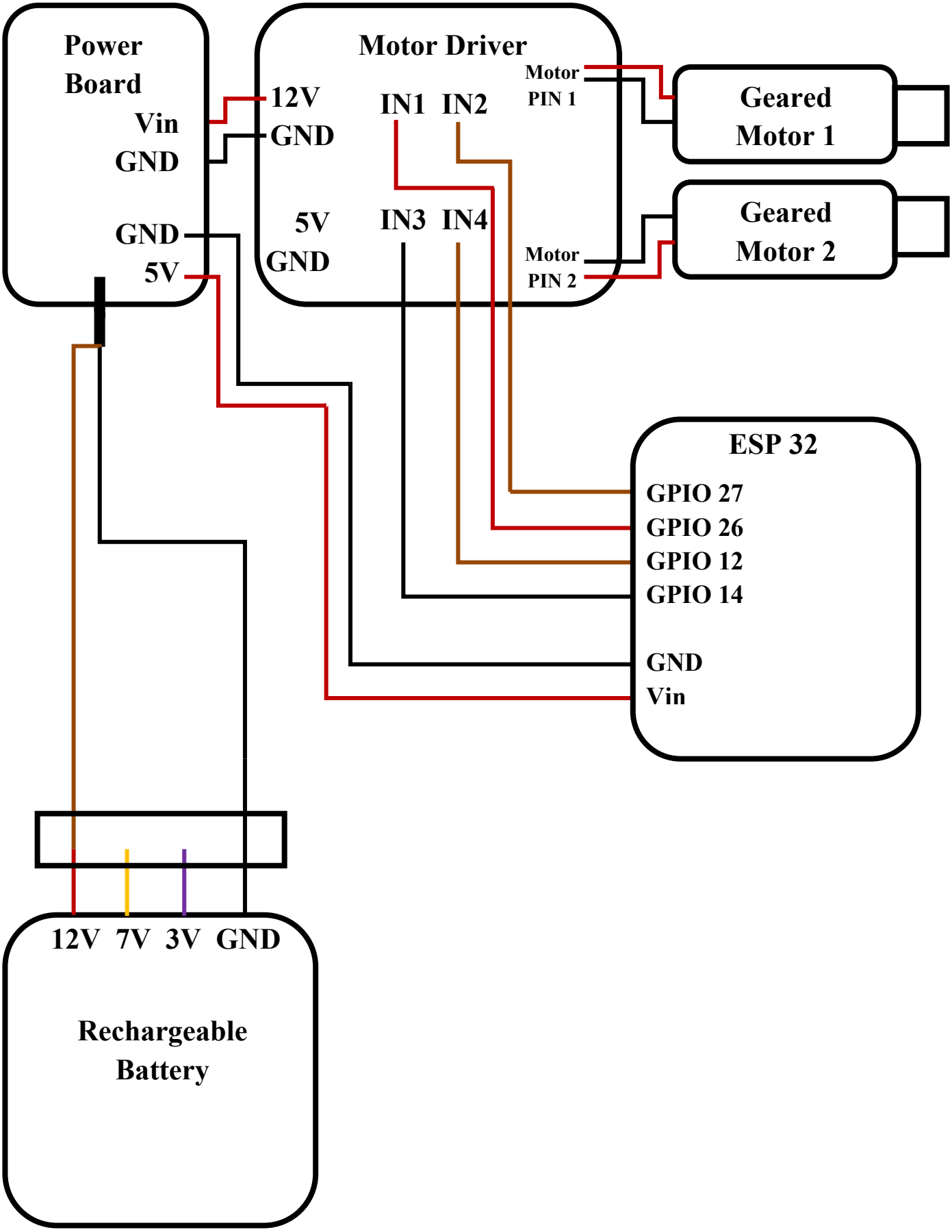
Working of Smart Helmet:

- When Helmet is not wearied by user the car will not start which mean ultrasonic sensor need to detect someone in helmet to send signal to car.
- When any alcohol detected by MQ-2 Sensor then car will not start.
- Both signals transmission and receiving is done between both ESP-32 with knowing there MAC Address.

CONNECTION DIAGRAM OF HELMET TRANSMITTER



CONNECTION DIAGRAM OF HELMET VEHICLE RECEIVER



PROGRAM CODE OF SMART HELMET

Code A: Helmet Transmitter

```
#include <WiFi.h>
#include <esp_now.h>

// === Pins ===
#define TRIG_PIN 5
#define ECHO_PIN 18
#define MQ3_PIN 34

// === Thresholds ===
#define SAFE_DISTANCE 20
#define ALCOHOL_LIMIT 1100

// === Data Structure ===
typedef struct struct_message {
  float distance;
  int alcohol;
  bool safe;
} struct_message;

struct_message dataToSend;

// Receiver MAC Address (ESP32 #2)
uint8_t receiverAddress[] = {0x38, 0x18, 0x2B, 0x2E, 0xFC,
0x50};  // <-- CHANGE THIS!

// ✅ UPDATED CALLBACK (for new ESP-NOW version)
void onDataSent(const wifi_tx_info_t *info,
esp_now_send_status_t status) {
  Serial.print("Send Status: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Success"
: "Fail");
}
```

```
void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(MQ3_PIN, INPUT);
  // Initialize WiFi in STA mode
  WiFi.mode(WIFI_STA);
  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW init failed!");
    return;
  }

// ✅ Register new callback
  esp_now_register_send_cb(onDataSent);
  // Register peer
  esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, receiverAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }
}

float getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  float distance = duration * 0.034 / 2;
  return distance;
}
```

```
void loop() {
  float distance = getDistance();
  int alcohol = analogRead(MQ3_PIN);

  bool isSafe = (distance < SAFE_DISTANCE && alcohol < 
ALCOHOL_LIMIT);

  dataToSend.distance = distance;
  dataToSend.alcohol = alcohol;
  dataToSend.safe = isSafe;

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.print(" cm, Alcohol: ");
  Serial.print(alcohol);
  Serial.print(" -> Safe: ");
  Serial.println(isSafe ? "YES" : "NO");

  esp_now_send(receiverAddress, (uint8_t *)&dataToSend, 
sizeof(dataToSend));

  delay(1000);
}
```

Code B: Helmet Vehicle Receiver

```
#include <WiFi.h>
#include <esp_now.h>

// === Motor Pins ===
#define IN1 26
#define IN2 27
#define IN3 14
#define IN4 12
// === Data Structure (same as sender) ===
typedef struct struct_message {
  float distance;
  int alcohol;
  bool safe;
} struct_message;

struct_message receivedData;

void moveForward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void stopMotor() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

// === UPDATED CALLBACK for ESP-IDF v5.x ===
void onDataRecv(const esp_now_recv_info *info, const
uint8_t *incomingData, int len) {
  memcpy(&receivedData, incomingData,
sizeof(receivedData));
```

```
  Serial.print("Distance: ");
  Serial.print(receivedData.distance);
  Serial.print(" cm, Alcohol: ");
  Serial.print(receivedData.alcohol);
  Serial.print(", Safe: ");
  Serial.println(receivedData.safe ? "YES" : "NO");

  if (receivedData.safe) {
    moveForward();
  } else {
    stopMotor();
  }
}
void setup() {
  Serial.begin(115200);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  stopMotor();

  WiFi.mode(WIFI_STA);

  if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW init failed!");
    return;
  }
  // ✅ Register callback (new signature)
  esp_now_register_recv_cb(onDataRecv);
}

void loop() {
  // Optional safety
  delay(1000);
}
```

Code C: ESP MAC Checker

```
#include <WiFi.h>
#include "esp_mac.h"   // Needed to get hardware MAC
address

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Start WiFi in Station Mode
  WiFi.mode(WIFI_STA);

  // 1 Get WiFi MAC Address
  Serial.print("WiFi MAC Address: ");
  Serial.println(WiFi.macAddress());

  // 2 Get Hardware (Base) MAC Address
  uint8_t mac[6];   // Array to store MAC address
  esp_efuse_mac_get_default(mac);

  Serial.print("Hardware MAC Address: ");
  Serial.printf("%02X:%02X:%02X:%02X:%02X:%02X\n",
                mac[0], mac[1], mac[2],
                mac[3], mac[4], mac[5]);
}

void loop() {
  // Nothing here
}
```

ACTIVITY 5

Concept of Satellites & Simulated Satellite Launch

REQUIRED COMPONENTS:

1. ESP32
2. DHT 11
3. MQ 2
4. Power Board
5. 1-PIN Jumper Wire
6. Battery

What is a Satellite?

A **satellite** is an object that moves around (orbits) a planet.

There are two types:

1. **Natural Satellite** – Example: Moon (moves around Earth naturally)
2. **Artificial Satellite** – Made by humans and sent into space.
   Example: Aryabhata (India's first satellite)

Why Do We Use Satellites?

Satellites help us in:

- TV and Mobile Communication
- Weather Forecast
- GPS Location
- Space Research
- Earth Observation

Simulated Satellite Launch Activity (Using ESP32)

In this activity, students will **simulate a satellite launch** and understand how satellites send data to Earth.

Instead of real space, we will:

- Use **ESP32** as our "Satellite"
- Use **Sensors** to collect data

- Use **WiFi** to send data
- Use **Laptop/Mobile screen** as "Ground Station"

Components Used (Easy Explanation)

ESP32 (Main Brain)

Full Form: **ESP = Espressif**, 32-bit Microcontroller

- Works like the **satellite computer**
- Has built-in **WiFi**
- Collects data from sensors
- Sends data wirelessly

It acts as our **mini satellite**

DHT11 Sensor

Measures Temperature, Humidity

In real satellites:

- Temperature sensors check space temperature
- Weather satellites measure atmospheric data

In our project:

- DHT11 gives temperature & humidity

MQ Gas Sensor

Detects gases like Smoke, LPG, Pollution gases

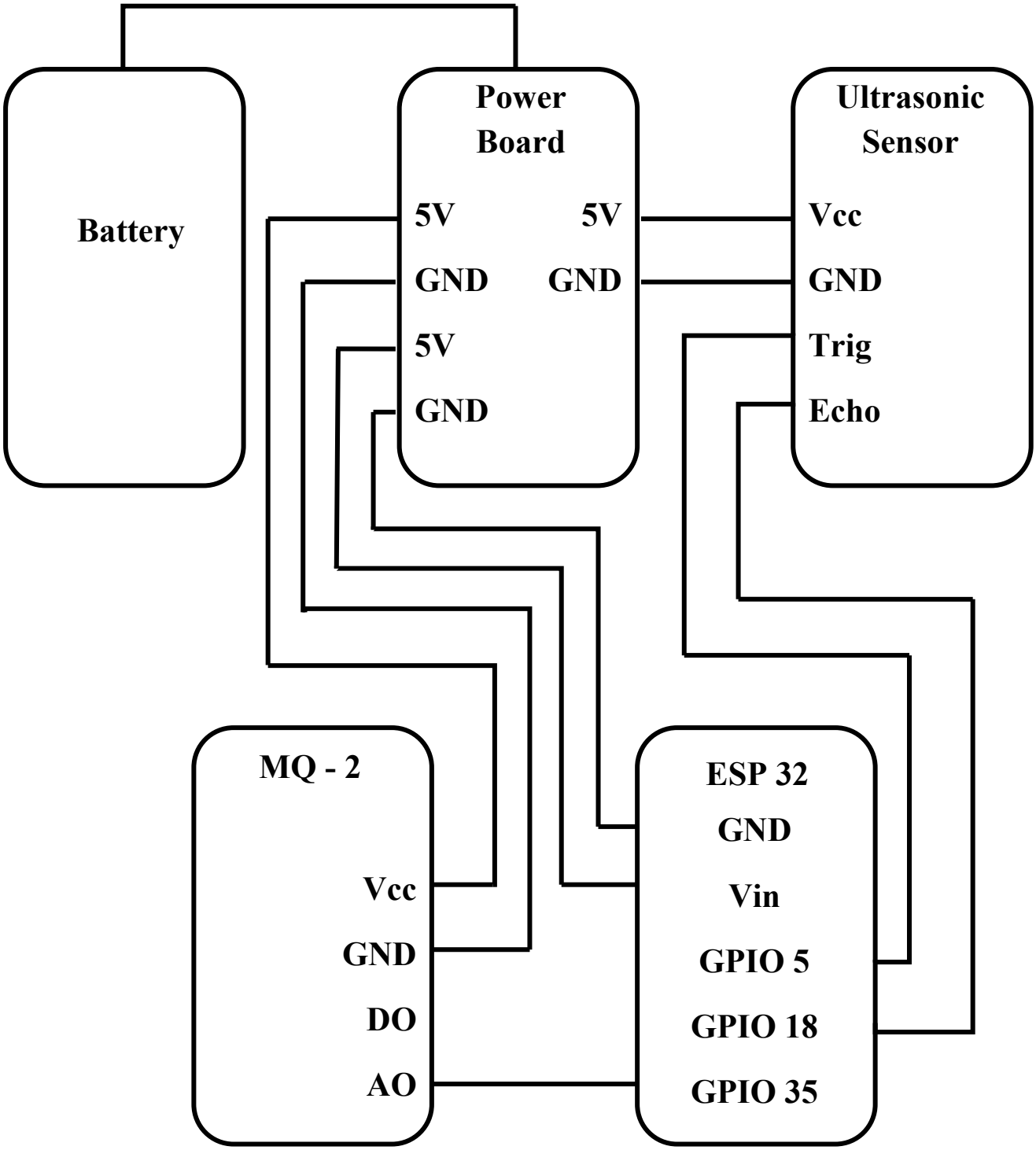
In real satellites:

- They detect air pollution & harmful gases

In our project:

- MQ sensor simulates atmosphere monitoring

CONNECTION DIAGRAM OF SATELLITES SYSTEM



PROGRAM CODE OF SATELLITES SYSTEM

```
#include <WiFi.h>
#include <WebServer.h>
#include <DHT.h>
// ===== Pin Definitions =====
#define DHTPIN 4
#define DHTTYPE DHT11
#define MQ3_PIN 34
#define WIFI_LED 2
// ===== WiFi Details =====
const char* ssid = "Redmi Note 11 Pro+ 5G";
const char* password = "12345678";
// ===== Objects =====
WebServer server(80);
DHT dht(DHTPIN, DHTTYPE);
// ===== HTML Web Page =====
String webpage = "";

void setup() {
  Serial.begin(115200);
  pinMode(WIFI_LED, OUTPUT);
  digitalWrite(WIFI_LED, LOW);
  dht.begin();
  // Connect WiFi
  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
       digitalWrite(WIFI_LED,  !digitalRead(WIFI_LED));  //
Blink LED
  }

  digitalWrite(WIFI_LED, HIGH);
  Serial.println("\nWiFi Connected!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
```

```
  // Webpage route
  server.on("/", handleRoot);
  server.begin();
}
void loop() {
  server.handleClient();
}
// ===== Function to Handle Web Page =====
void handleRoot() {

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  int airValue = analogRead(MQ3_PIN);
  // If sensor fails
  if (isnan(temperature) || isnan(humidity)) {
    temperature = 0;
    humidity = 0;
  }
  // Simple HTML Page
  webpage = "<html>";
       webpage      +=      "<head><title>ESP32      Sensor
Data</title></head>";
    webpage   +=   "<body   style='text-align:center;   font-
family:Arial;'>";
  webpage += "<h1>ESP32 Sensor Dashboard</h1>";
  webpage += "<h2>Temperature: " + String(temperature) + "
°C</h2>";
   webpage  +=  "<h2>Humidity:  "  +  String(humidity)  +  "
%</h2>";
  webpage += "<h2>Air Quality (MQ3): " + String(airValue) +
"</h2>";
  webpage += "<br><br>";
    webpage   +=   "<a   href='/'><button   style='padding:10px
20px;'>Refresh</button></a>";
  webpage += "</body></html>";

  server.send(200, "text/html", webpage);
}
```