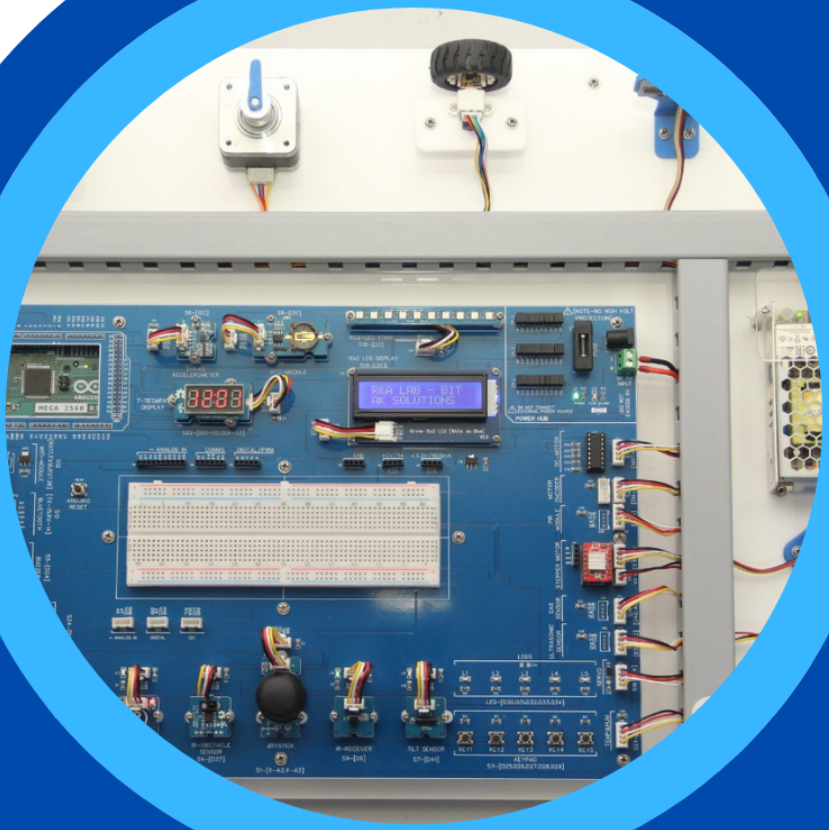
E-GROOTS

TRAINER KIT V.1



USER MANUAL

egroots.in@gmail.com | 80152-21905 | http://egroots.in

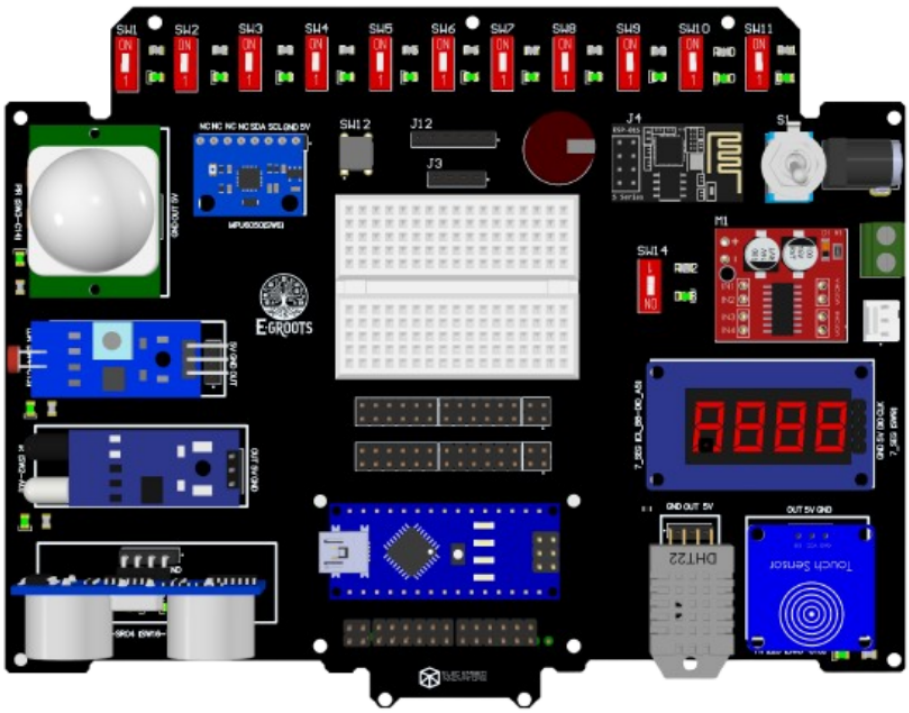
E-GROOTS

TRAINER KIT V.1

USER MANUAL

egroots.in@gmail.com | 80152-21905 | http://egroots.in

TRAINER KIT MINI



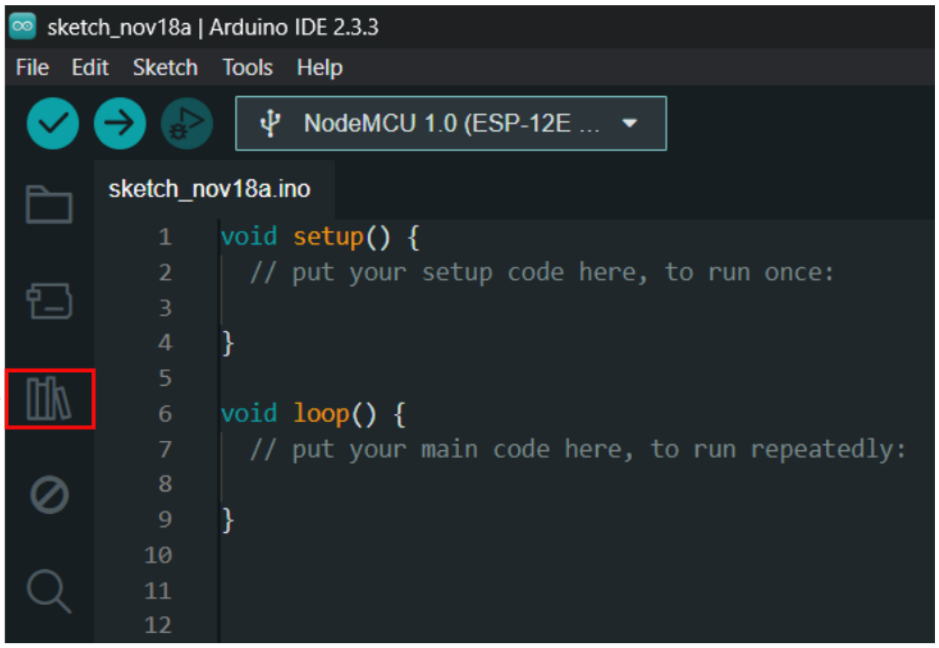
ARDUINO IDE LIBRARY INSTALLATION STEPS

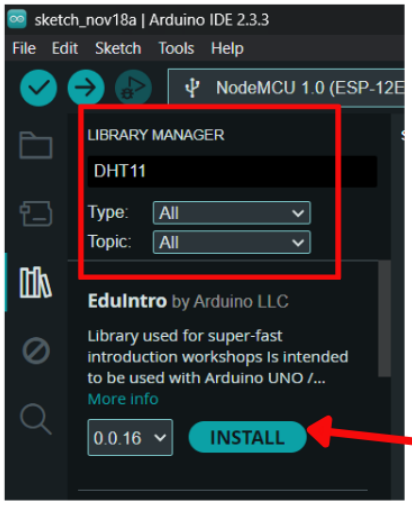
- **Preferred version** : Arduino IDE 2.3.2

01

STEP

Library Manager





Search the library

Click Install

02

STEP

WHAT'S IN THE BOX?

1 SENSORS

1. LDR
2. PIR
3. HC-SR04
4. TTP223
5. DHT11
6. IR

2 DISPLAY

1. 7 - SEGMENT DISPLAY

3 MODULES

1. MPU6050
2. BLUETOOTH MODULE
3. ESP8266 WIFI MOODULE

4 DRIVERS

1. MX1508

5 ACTUATORS AND INDICATORS

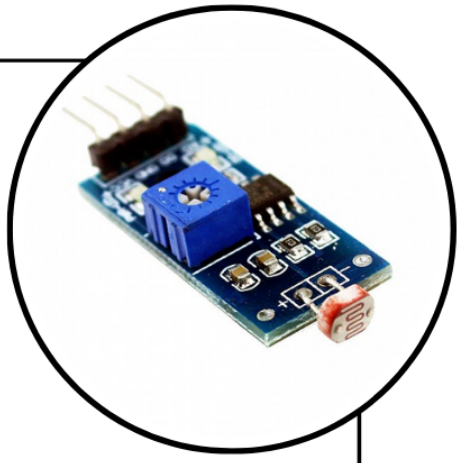
1. BUZZER
2. PUSH BUTTON
3. N20 MOTOR
4. RELAY CIRCUIT

6 OTHERS

1. MICROCONTROLER
2. BREAD BOARD

SENSORS

LDR (Light Dependent Resistor)



The LDR is a light based sensor where the amount of light present in the surrounding is measured by the element in the sensor. When the amount of the light increases the resistence offered by the element also increases.

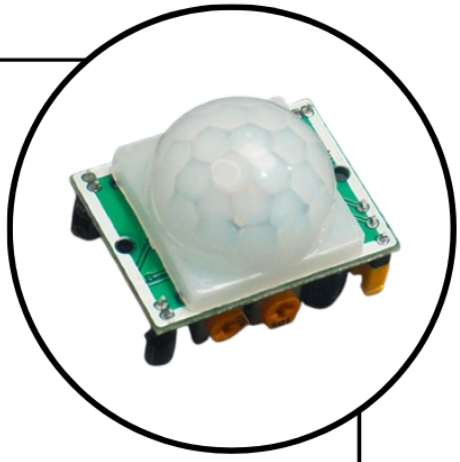
SPECIFICATION:

- **Input Voltage:** DC 3.3V to 5V
- **Operating Current:** 10mA to 20mA
- **Output:** Analog and Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 8

PIR (Passive Infrared Sensor)



PIR sensor detects motion by sensing the infrared radiation(heat) emitted by humans and animals .When they move in front of the sensor, the infrared radiation changes and the sensor detects movement.

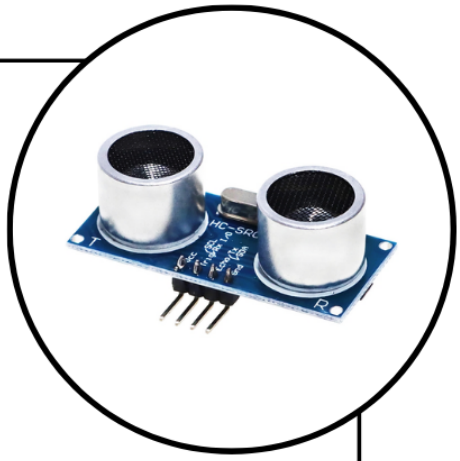
SPECIFICATION:

- **Input Voltage:** DC 4.5V- 12V
- **Operating Current:** 5mA to 15mA
- **Output:** Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 9

HC-SR04 (Ultrasonic Sensor)



An ultrasonic sensor measures an object's distance by using ultrasonic sound wave. The sensor emits sound wave to the object and recieves the reflected wave from the object. The time taken to emit and recieve the wave is used to calculate the distance.

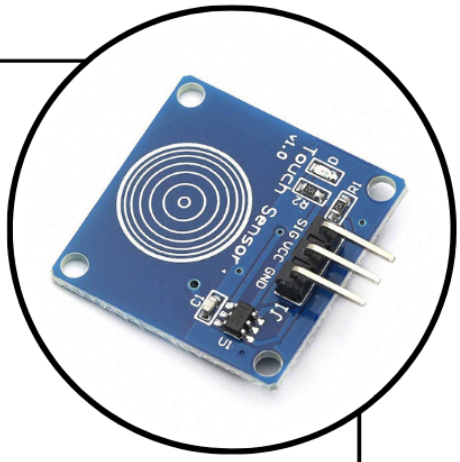
SPECIFICATION:

- **Input Voltage:** DC 5V
- **Operating** Current: 15mA
- **Output:** Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 10

TTP223 (Touch Sensor)



TTP223 is a touch sensor that detects the touch on its surface. When an conductive object touches the sensor, the charge present in the sensor and object changes which is detected by the touch sensor.

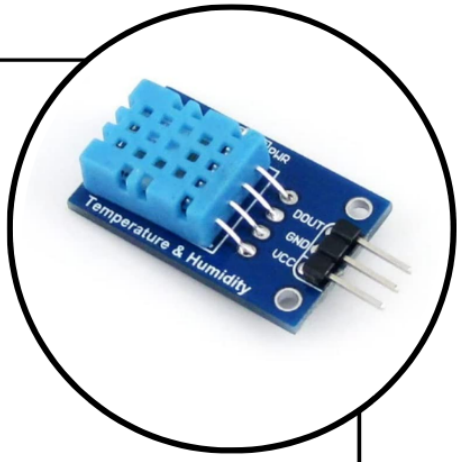
SPECIFICATION:

- **Input Voltage:** DC 3.3V to 5V
- **Operating Current:** 2mA
- **Output:** Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 12

DHT11 (Temperature Sensor)



Temperature sensor is used to measure the temperature and humidity of an object or surroundings.The sensor has a thermistor which is a type of resistor that increases resistence when the temperature increase. It also has a moisture absorbing material that senses any changes in the humidity.

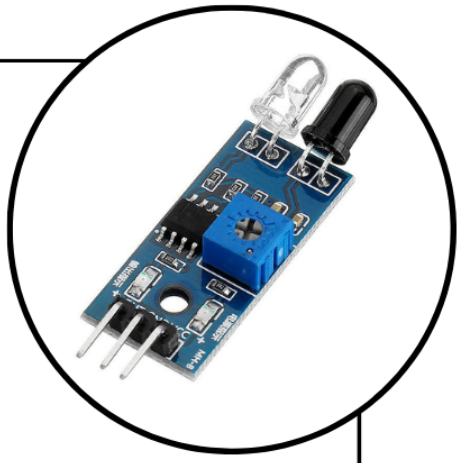
SPECIFICATION:

- **Input Voltage:** DC 3.3V to 5V
- **Operating Current:** 2.5mA
- **Output:** Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 13

IR SENSOR (Infrared)



An ir sensor measures an object's distance by using infrared light wave. The sensor emits light wave to the object and recieves the reflected wave from the object. The time taken to emit and recieve the wave is used to calculate the distance.

SPECIFICATION:

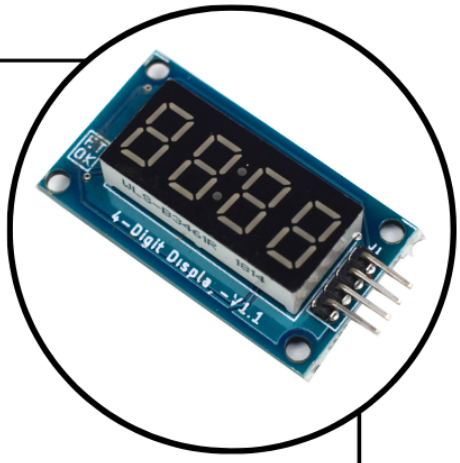
- **Input Voltage:** DC 3.3V to 5V
- **Operating Current:** 10mA to 50mA
- **Output:** Analog and Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 15

DISPLAY

7 SEGMENT LCD



The 7 segment LCD is a display device used to display some numbers and alphabets. Its consists of 7 induvidual LCDs or LEDs that can be turned on or off to form various numbers and characters.

SPECIFICATION:

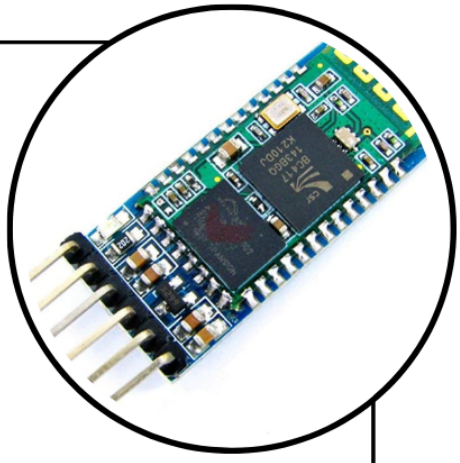
- **Input Voltage:** DC 3.3V to 5V
- **Operating Current:** 1mA to 30mA

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 16

MODULES

BLUETOOTH MODULE



This module enables wireless communication to transmit and receive data between devices using bluetooth technology.

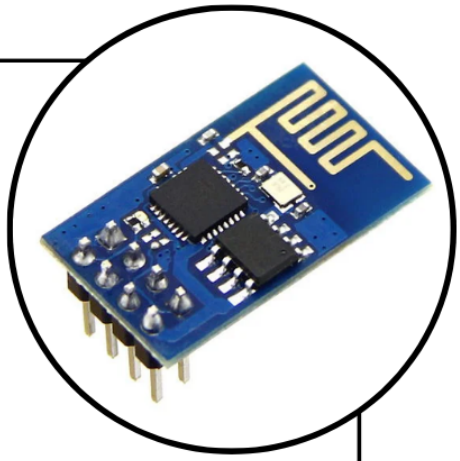
SPECIFICATION:

- **Input Voltage:** DC 3.3V- 5V
- **Operating Current:** 30mA

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 18

ESP8266 WIFI MODULE



The ESP8266 Wi-Fi Module is a highly integrated, cost-effective device for enabling Wi-Fi connectivity in embedded systems. It is widely used in IoT applications due to its ability to connect microcontrollers to Wi-Fi networks for wireless communication and data exchange.

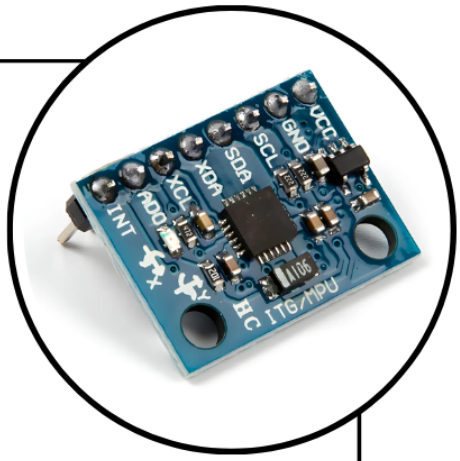
SPECIFICATION:

- **Input Voltage:** DC 5V
- **Operating Current**: 15mA
- **Output:** Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 19

MPU6050



The MPU6050 is a 6 axis motion tracking sensor that is widely used in applications like in drones, robotics, wearable and gaming devices for precise motion detection and orientation. It measures acceleration (in X, Y, and Z axes) and angular velocity.

SPECIFICATION:

- **Input Voltage:** DC 3.3V to 5V
- **Operating Current:** 1mA to 4mA
- **Output:** Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 21

ACTUATORS AND INDICATORS

BUZZER

Buzzers are devices that produce sound when an electrical signal is applied. It converts electrical energy in mechanical energy(vibration) which in turn produces sound. Its is used in applications like alarm, timer, notification etc,.

SPECIFICATION:

- **Input Voltage:** DC 4V to 8V
- **Operating Current:** **20**mA to 40mA
- **Sound level:** 90dB to 110 dB

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 23

PUSH BUTTON

A push button is a simple switch used to control the electrical circuit. By default the switch are open, meaning the circuit is open. When the buttin is pressed the switch is closed, completeing the ciruit and allowing the current to flow.

SPECIFICATION:

- **Input Voltage:** DC 3.3V- 5V
- **Operating Current:** 1mA to 10mA

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 24

N20 MOTOR

N20 motor is a small cylindrical DC gear motor widely used in robotics. The gearbox attached to it helps it to spin slowly but with more force(torque).

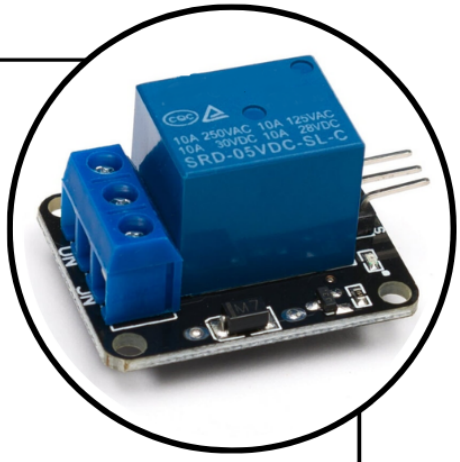
SPECIFICATION:

- **Input Voltage:** DC 3V to 12V
- **Operating Current:** 50mA to 300mA

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 25

RELAY CIRCUIT



A relay is an electrically operated switch that uses a small eletrical signal to turn on or off high power circuits. There is a small coil of wire called eletrcomagnet that creates mangmetic field when current passes through it. This magnetic field pulls a small lever to either on or off the circuit

SPECIFICATION:

- **Input Voltage:** DC 3.3V to 5V
- **Operating Current:** 10mA to 50mA
- **Output:** Analog and Digital values

REFERENCE: ***Datasheet Link***

EXAMPLE CODE - PAGE NO : 26

CODE SNIPPETS

LDR SENSOR:

ARDUINO

```
const int ldrPin = A0;
void setup() {
  Serial.begin(9600);
  pinMode(ldrPin, INPUT);
}
void loop() {
  int ldrStatus = analogRead(ldrPin);
  Serial.println(ldrStatus);
}
```

ESP32

```
const int ldrPin = 34;
void setup() {
  Serial.begin(9600);
  pinMode(ldrPin,INPUT);
}
void loop() {
  int ldrStatus = analogRead(ldrPin);
  Serial.println(ldrStatus);
}
```

RASPBERRY PI PICO

```
const int ldrPin = 26;
void setup() {
  Serial.begin(9600);
  pinMode(ldrPin,INPUT);
}
void loop() {
  int ldrStatus = analogRead(ldrPin);
  Serial.println(ldrStatus);
}
```

STM32

```
const int ldrPin = PA0;
void setup() {
  Serial.begin(9600);
  pinMode(ldrPin,INPUT);
}
void loop() {
  int ldrStatus = analogRead(ldrPin);
  Serial.println(ldrStatus);
}
```

PIR SENSOR:

ARDUINO

```
const int pin = 2;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Motion Detected!");
  } else {
    Serial.println("No Motion");
  }
}
```

ESP32

```
const int pin = 15;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Motion Detected!");
  } else {
    Serial.println("No Motion");
  }
}
```

RASPBERRY PI PICO

```
const int pin = 0;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Motion Detected!");
  } else {
    Serial.println("No Motion");
  }
}
```

STM32

```
const int pin = PA0;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Motion Detected!");
  } else {
    Serial.println("No Motion");
  }
}
```

HC-SR04 (Ultrasonic Sensor):

ARDUINO

```
const int trigPin = 9;
const int echoPin = 10;
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
}
```

ESP32

```
const int trigPin = 5;
const int echoPin = 18;
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
}
```

HC-SR04 (Ultrasonic Sensor):

RASPBERRY PI PICO

```
const int trigPin = 2;
const int echoPin = 3;
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
}
```

STM32

```
const int trigPin = PA8;
const int echoPin = PA9;
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
}
```

TTP223 (Touch Sensor):

ARDUINO

```
const int pin = 2;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Touched!");
  } else {
    Serial.println("Not Touched");
  }
}
```

ESP32

```
const int pin = 15;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Touched!");
  } else {
    Serial.println("Not Touched");
  }
}
```

RASPBERRY PI PICO

```
const int pin = 0;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Touched!");
  } else {
    Serial.println("Not Touched");
  }
}
```

STM32

```
const int pin = PA0;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Touched!");
  } else {
    Serial.println("Not Touched");
  }
}
```

DHT11 SENSOR (Temperature Sensor):

ARDUINO

```
#include<DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);
void setup(){
 Serial.begin(9600);
 dht.begin();
}
void loop(){
 float humidity=dht.readHumidity();
 float temperature=dht.readTemperature();
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.print("Temperature: ");
 Serial.print(temperature);
}
```

ESP32

```
#include<DHT.h>
#define DHTPIN 15
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);
void setup(){
 Serial.begin(9600);
 dht.begin();
}
void loop(){
 float humidity=dht.readHumidity();
 float temperature=dht.readTemperature();
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.print("Temperature: ");
 Serial.print(temperature);
}
```

DHT11 SENSOR:

RASPBERRY PI PICO

```
#include<DHT.h>
#define DHTPIN 0
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);
void setup(){
 Serial.begin(9600);
 dht.begin();
}
void loop(){
 float humidity=dht.readHumidity();
 float temperature=dht.readTemperature();
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.print("Temperature: ");
 Serial.print(temperature);
}
```

STM32

```
#include<DHT.h>
#define DHTPIN PA1
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);
void setup(){
 Serial.begin(9600);
 dht.begin();
}
void loop(){
 float humidity=dht.readHumidity();
 float temperature=dht.readTemperature();
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.print("Temperature: ");
 Serial.print(temperature);
}
```

IR SENSOR:

ARDUINO

```
const int pin = 2;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Object Detected!");
  } else {
    Serial.println("No Object Detected");
  }
}
```

ESP32

```
const int pin = 15;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Object Detected!");
  } else {
    Serial.println("No Object Detected");
  }
}
```

RASPBERRY PI PICO

```
const int pin = 0;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Object Detected!");
  } else {
    Serial.println("No Object Detected");
  }
}
```

STM32

```
const int pin = PA0;
void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}
void loop() {
  int sensorState = digitalRead(pin);
  if (sensorState == HIGH) {
    Serial.println("Object Detected!");
  } else {
    Serial.println("No Object Detected");
  }
}
```

7 SEGMENT DISPLAY

ARDUINO

```
const int a = 2;
const int b = 3;
const int c = 4;
const int d = 5;
const int e = 6;
const int f = 7;
const int g = 8;
int segments[] = {a, b, c, d, e, f, g};
byte digitPattern = B1011011;
void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(segments[i], OUTPUT);
  }
  displayNumber();
}
void loop() {
}
void displayDigit(int digit) {
  for (int i = 0; i < 7; i++) {
    digitalWrite(segments[i], bitRead(digitPattern, 6 - i));
  }
}
```

ESP32

```
const int a = 13;
const int b = 12;
const int c = 14;
const int d = 27;
const int e = 26;
const int f = 25;
const int g = 33;
int segments[] = {a, b, c, d, e, f, g};
byte digitPattern = B1011011;
void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(segments[i], OUTPUT);
  }
  displayNumber();
}
void loop() {
}
void displayDigit(int digit) {
  for (int i = 0; i < 7; i++) {
    digitalWrite(segments[i], bitRead(digitPattern, 6 - i));
  }
}
```

7 SEGMENT DISPLAY

RASPBERRY PI PICO

```
const int a = 2;
const int b = 3;
const int c = 4;
const int d = 5;
const int e = 6;
const int f = 7;
const int g = 8;
int segments[] = {a, b, c, d, e, f, g};
byte digitPattern = B1011011;
void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(segments[i], OUTPUT);
  }
  displayNumber();
}
void loop() {
}
void displayDigit(int digit) {
  for (int i = 0; i < 7; i++) {
    digitalWrite(segments[i], bitRead(digitPattern, 6 - i));
  }
}
```

STM32

```
const int a = PB0;
const int b = PB1;
const int c = PA8;
const int d = PA9;
const int e = PA10;
const int f = PB10;
const int g = PB11;
int segments[] = {a, b, c, d, e, f, g};
byte digitPattern = B1011011;
void setup() {
  for (int i = 0; i < 7; i++) {
    pinMode(segments[i], OUTPUT);
  }
  displayNumber();
}
void loop() {
}
void displayDigit(int digit) {
  for (int i = 0; i < 7; i++) {
    digitalWrite(segments[i], bitRead(digitPattern, 6 - i));
  }
}
```

BLUETOOTH MODULE:

ARDUINO

```
#include<SoftwareSerial.h>
SoftwareSerial bluetooth(2,3);
void setup() {
  bluetooth.begin(9600);
  Serial.begin(9600);
}
void loop() {
    if (bluetooth.available())
    {
     Serial.write(bluetooth.read());
    }
}
```

ESP32

```
#define RX_PIN 16
#define TX_PIN 17
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
}
void loop() {
  if (Serial1.available()) {
    Serial.write(Serial1.read());
  }
}
```

RASPBERRY PI PICO

```
#define RX_PIN 0
#define TX_PIN 1
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
}
void loop() {
  if (Serial1.available()) {
    Serial.write(Serial1.read());
  }
}
```

STM32

```
#define RX_PIN PA9
#define TX_PIN PA10
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
}
void loop() {
  if (Serial1.available()) {
    Serial.write(Serial1.read());
  }
}
```

ESP8266 WIFI MODULE:

ARDUINO

```
#include <SoftwareSerial.h>
SoftwareSerial softSerial(3, 2);
void setup()
{
  uint32_t baud = 9600;
  Serial.begin(baud);
  softSerial.begin(baud);
  Serial.print("SETUP!! @");
  Serial.println(baud);
}
 void loop()
{
    while(softSerial.available() > 0)
    {
      char a = softSerial.read();
      if(a == '')
        continue;
      if(a != 'r' && a != 'n' && (a < 32))
        continue;
      Serial.print(a);
    }
    while(Serial.available() > 0)
    {
      char a = Serial.read();
      Serial.write(a);
      softSerial.write(a);
    }
}
```

ESP32

```
#define RX_PIN 16
#define TX_PIN 17
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
  Serial.println("ESP32 UART Setup Complete!");
}
void loop() {
  while (Serial1.available() > 0) {
    char c = Serial1.read();
    Serial.print(c);
  }
  while (Serial.available() > 0) {
    char c = Serial.read();
    Serial1.write(c);
  }
}
```

ESP8266 WIFI MODULE:

RASPBERRY PI PICO

```
#include "hardware/uart.h"
#include "hardware/gpio.h"
#define RX_PIN 5
#define TX_PIN 4
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
  Serial.println("Raspberry Pi Pico UART Setup Complete!");
}
void loop() {
  while (Serial1.available() > 0) {
    char c = Serial1.read();
    Serial.print(c);
  }
  while (Serial.available() > 0) {
    char c = Serial.read();
    Serial1.write(c);
  }
}
```

STM32

```
#define RX_PIN PA10
#define TX_PIN PA9
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600, SERIAL_8N1, RX_PIN, TX_PIN);
  Serial.println("STM32 UART Setup Complete!");
}
void loop() {
  while (Serial1.available() > 0) {
    char c = Serial1.read();
    Serial.print(c);
  }
  while (Serial.available() > 0) {
    char c = Serial.read();
    Serial1.write(c);
  }
}
```

MPU6050 MODULE:

ARDUINO

```
#include <Wire.h>
#include <MPU6050.h>
MPU6050 mpu;
void setup() {
  Serial.begin(9600);
  Wire.begin();
  Serial.println("Initializing MPU6050...");
  if (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G)) {
    Serial.println("Could not find MPU6050 sensor. Check connections.");
    while (1);
  }
  Serial.println("MPU6050 Initialized!");
  delay(1000);
}
void loop() {
  Vector rawAccel = mpu.readRawAccel();
  Vector rawGyro = mpu.readRawGyro();
  Serial.print("Accel: ");
  Serial.print(rawAccel.XAxis);
  Serial.print(", ");
  Serial.print(rawAccel.YAxis);
  Serial.print(", ");
  Serial.print(rawAccel.ZAxis);
  Serial.print(" | ");
  Serial.print("Gyro: ");
  Serial.print(rawGyro.XAxis);
  Serial.print(", ");
  Serial.print(rawGyro.YAxis);
  Serial.print(", ");
  Serial.print(rawGyro.ZAxis);
  Serial.println();
}
```

ESP32

```
#include <Wire.h>
#include <MPU6050.h>
MPU6050 mpu;
void setup() {
  Serial.begin(9600);
  Wire.begin(21,22);
  Serial.println("Initializing MPU6050...");
  if (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G)) {
    Serial.println("Could not find MPU6050 sensor. Check connections.");
    while (1);
  }
  Serial.println("MPU6050 Initialized!");
  delay(1000);
}
void loop() {
  Vector rawAccel = mpu.readRawAccel();
  Vector rawGyro = mpu.readRawGyro();
  Serial.print("Accel: ");
  Serial.print(rawAccel.XAxis);
  Serial.print(", ");
  Serial.print(rawAccel.YAxis);
  Serial.print(", ");
  Serial.print(rawAccel.ZAxis);
  Serial.print(" | ");
  Serial.print("Gyro: ");
  Serial.print(rawGyro.XAxis);
  Serial.print(", ");
  Serial.print(rawGyro.YAxis);
  Serial.print(", ");
  Serial.print(rawGyro.ZAxis);
  Serial.println();
}
```

MPU6050 MODULE:

RASPBERRY PI PICO

```
#include <Wire.h>
#include <MPU6050.h>
MPU6050 mpu;
void setup() {
  Serial.begin(9600);
  Wire.begin(4,5);
  Serial.println("Initializing MPU6050...");
  if (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G)) {
    Serial.println("Could not find MPU6050 sensor. Check connections.");
    while (1);
  }
  Serial.println("MPU6050 Initialized!");
  delay(1000);
}
void loop() {
  Vector rawAccel = mpu.readRawAccel();
  Vector rawGyro = mpu.readRawGyro();
  Serial.print("Accel: ");
  Serial.print(rawAccel.XAxis);
  Serial.print(", ");
  Serial.print(rawAccel.YAxis);
  Serial.print(", ");
  Serial.print(rawAccel.ZAxis);
  Serial.print(" | ");
  Serial.print("Gyro: ");
  Serial.print(rawGyro.XAxis);
  Serial.print(", ");
  Serial.print(rawGyro.YAxis);
  Serial.print(", ");
  Serial.print(rawGyro.ZAxis);
  Serial.println();
}
```

STM32

```
#include <Wire.h>
#include <MPU6050.h>
MPU6050 mpu;
void setup() {
  Serial.begin(9600);
  Wire.begin(PB7,PB6);
  Serial.println("Initializing MPU6050...");
  if (!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G)) {
    Serial.println("Could not find MPU6050 sensor. Check connections.");
    while (1);
  }
  Serial.println("MPU6050 Initialized!");
  delay(1000);
}
void loop() {
  Vector rawAccel = mpu.readRawAccel();
  Vector rawGyro = mpu.readRawGyro();
  Serial.print("Accel: ");
  Serial.print(rawAccel.XAxis);
  Serial.print(", ");
  Serial.print(rawAccel.YAxis);
  Serial.print(", ");
  Serial.print(rawAccel.ZAxis);
  Serial.print(" | ");
  Serial.print("Gyro: ");
  Serial.print(rawGyro.XAxis);
  Serial.print(", ");
  Serial.print(rawGyro.YAxis);
  Serial.print(", ");
  Serial.print(rawGyro.ZAxis);
  Serial.println();
}
```

BUZZER:

ARDUINO

```
int buzzerPin = 9;
void setup() {
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  digitalWrite(buzzerPin, HIGH);
  delay(1000);
  digitalWrite(buzzerPin, LOW);
  delay(1000);
}
```

ESP32

```
int buzzerPin = 14;
void setup() {
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  digitalWrite(buzzerPin, HIGH);
  delay(1000);
  digitalWrite(buzzerPin, LOW);
  delay(1000);
}
```

RASPBERRY PI PICO

```
int buzzerPin = 0;
void setup() {
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  digitalWrite(buzzerPin, HIGH);
  delay(1000);
  digitalWrite(buzzerPin, LOW);
  delay(1000);
}
```

STM32

```
int buzzerPin = PA0;
void setup() {
  pinMode(buzzerPin, OUTPUT);
}
void loop() {
  digitalWrite(buzzerPin, HIGH);
  delay(1000);
  digitalWrite(buzzerPin, LOW);
  delay(1000);
}
```

PUSH BUTTON:

ARDUINO

```
int buttonPin = 2;
int buttonState = 0;
void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    Serial.println("Button Pressed");
  } else {
    Serial.println("Button Released");
  }
  delay(100);
}
```

ESP32

```
int buttonPin = 13;
int buttonState = 14;
void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    Serial.println("Button Pressed");
  } else {
    Serial.println("Button Released");
  }
  delay(100);
}
```

RASPBERRY PI PICO

```
int buttonPin = 2;
int buttonState = 0;
void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    Serial.println("Button Pressed");
  } else {
    Serial.println("Button Released");
  }
  delay(100);
}
```

STM32

```
int buttonPin = PA0;
int buttonState = PA1;
void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    Serial.println("Button Pressed");
  } else {
    Serial.println("Button Released");
  }
  delay(100);
}
```

N20 MOTOR

ARDUINO

```
int motorPin1 = 2;
int motorPin2 = 0;
void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
}
void loop() {
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  delay(2000);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  delay(1000);
}
```

ESP32

```
int motorPin1 = 13;
int motorPin2 = 14;
void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
}
void loop() {
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  delay(2000);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  delay(1000);
}
```

RASPBERRY PI PICO

```
int motorPin1 = 2;
int motorPin2 = 0;
void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
}
void loop() {
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  delay(2000);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  delay(1000);
}
```

STM32

```
int motorPin1 = PA0;
int motorPin2 = PA1;
void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
}
void loop() {
  digitalWrite(motorPin1, HIGH);
  digitalWrite(motorPin2, LOW);
  delay(2000);
  digitalWrite(motorPin1, LOW);
  digitalWrite(motorPin2, LOW);
  delay(1000);
}
```

RELAY CIRCUIT:

ARDUINO

```
int relayPin = 8;
void setup() {
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
}
void loop() {
  digitalWrite(relayPin, HIGH);
  delay(2000);
  digitalWrite(relayPin, LOW);
  delay(2000);
}
```

ESP32

```
int relayPin = 13;
void setup() {
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
}
void loop() {
  digitalWrite(relayPin, HIGH);
  delay(2000);
  digitalWrite(relayPin, LOW);
  delay(2000);
}
```

RASPBERRY PI PICO

```
int relayPin = 0;
void setup() {
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
}
void loop() {
  digitalWrite(relayPin, HIGH);
  delay(2000);
  digitalWrite(relayPin, LOW);
  delay(2000);
}
```

STM32

```
int relayPin = PA0;
void setup() {
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
}
void loop() {
  digitalWrite(relayPin, HIGH);
  delay(2000);
  digitalWrite(relayPin, LOW);
  delay(2000);
}
```