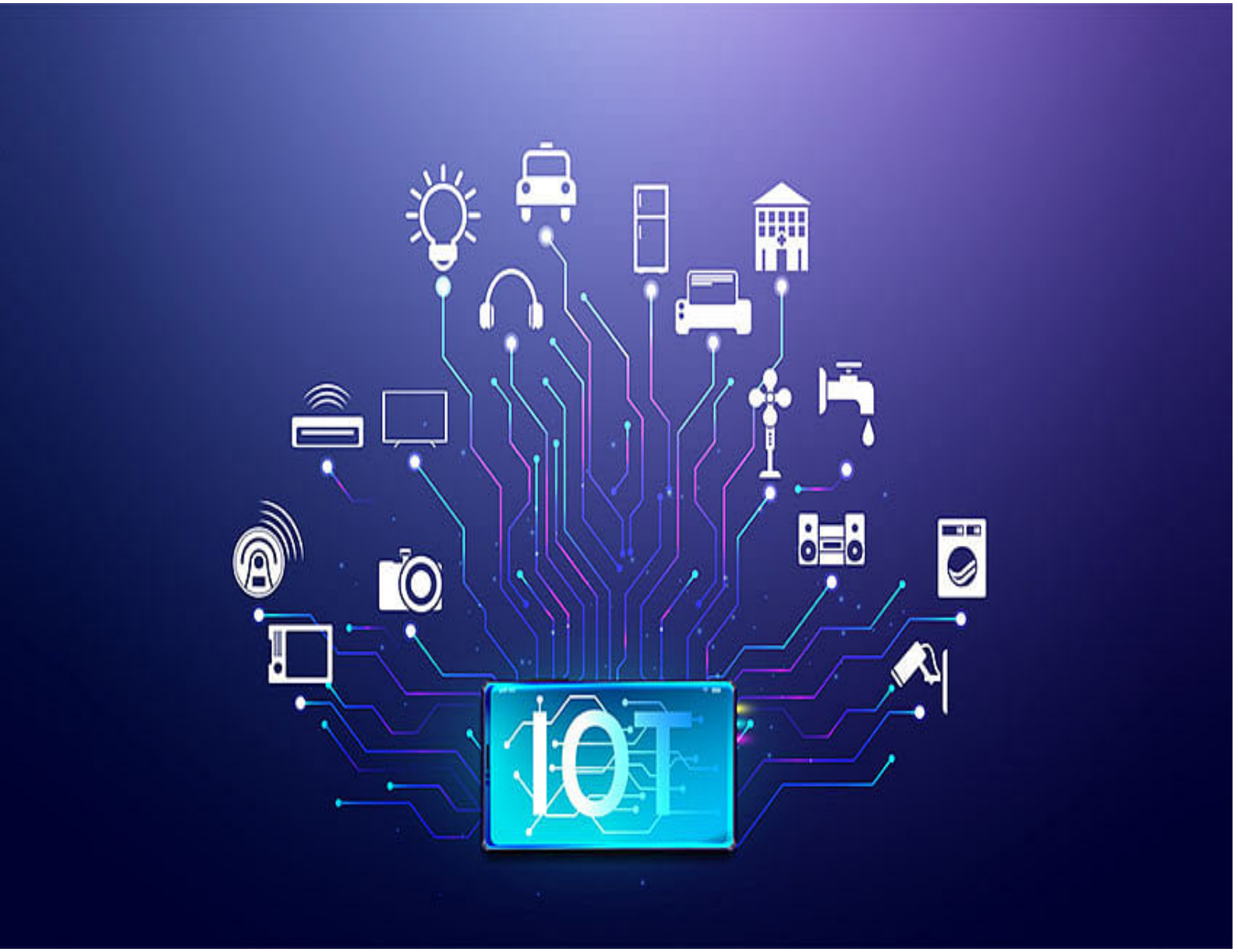
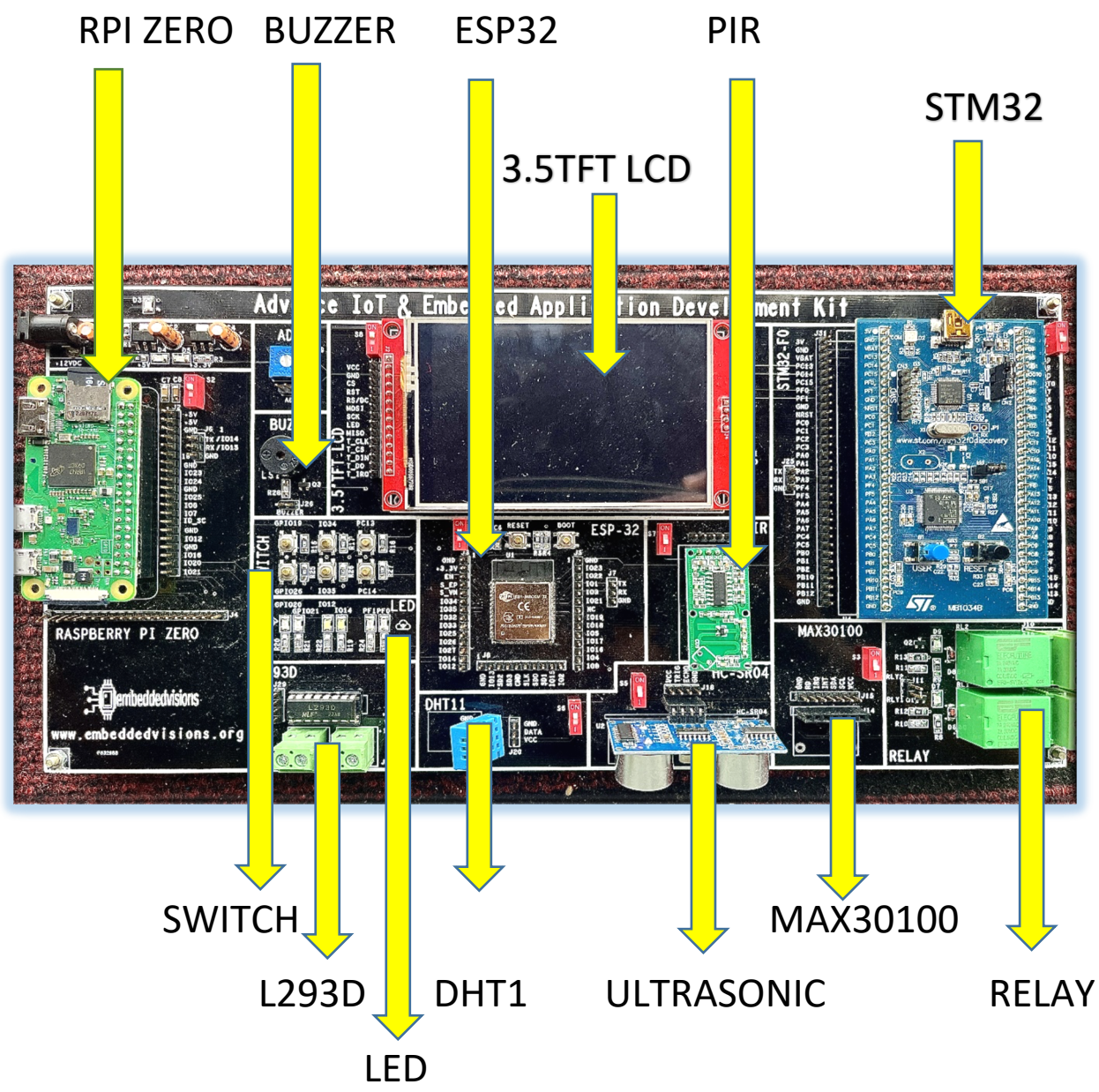
ADVANCE IOT & EMBBEDED APPLICATION DEVELOPMENT KIT



embeddedvisions

Content:

INTRODUCTION OF HARDWARE INTERFACE:



FEATURES AND SPECIFICATIONS:

- On Board Power Supply
- On Board three MCU ESP32, Raspberry pi zero, STM32
- On Board Peripherals LED, BUZZER, PIR, ULTRASONIC, RELAY, Touch LCD, MAX30100, L293D
- Dual-Core Processor
- Wi-Fi Connectivity, Bluetooth
- Low Power Consumption
- Integrated Sensors
- **ESP32**
    - Xtensa® Dual-Core 32-bit LX6 microprocessors, up to 600 DMIPS
    - 448 KByte ROM
    - 520 KByte SRAM
    - 16 KByte SRAM in RTC
    - QSPI Flash/SRAM, up to 4 x 16 Mbytes
- **Raspberry Pi ZERO**
    - Broadcom BCM2710A1, quad-core 64-bit SoC (Arm Cortex-A53 @ 1GHz)
    - 2.4GHz IEEE 802.11b/g/n wireless LAN, Bluetooth 4.2, BLE, onboard antenna
    - 5V DC 2.5A
    - microSD card slot
    - HAT-compatible 40-pin I/O header footprint
    - CSI-2 camera connector
- **STM32**
    - Core: Arm® 32-bit Cortex®-M3 CPU (120 MHz max) with Adaptive real-time accelerator (ART Accelerator™)
    - Up to 1 Mbyte of Flash memory
    - 8- to 14-bit parallel camera interface (48 Mbyte/s max.)
    - 2 × 12-bit D/A converters

SAFETY GUIDELINES:

- Always use the recommended power supply or power ratings.
- Do not remove any component while the power is switched on
- Do not remove the microcontroller from the programmer during the burning process Doing so may end up in damaging the controller permanently.
- While mounting the microcontroller on the ZIF sockets (of both development board and Programmer), keep in mind the correct direction of the microcontroller.

Esp32

1. INTRODUCTION OF ESP32

• Xtensa® Dual-Core 32-bit LX6 microprocessors, up to 600 DMIPS

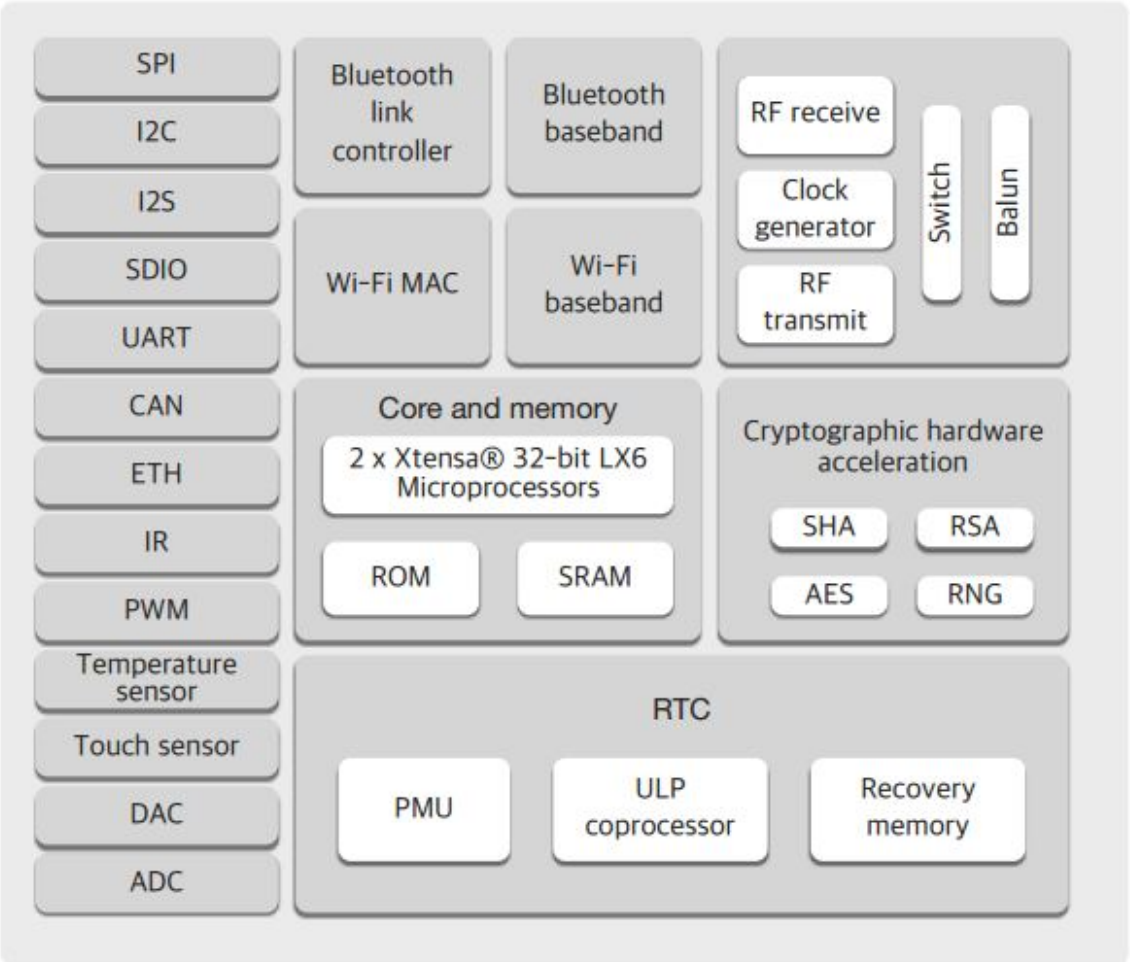
• 448 KByte ROM

• 520 KByte SRAM

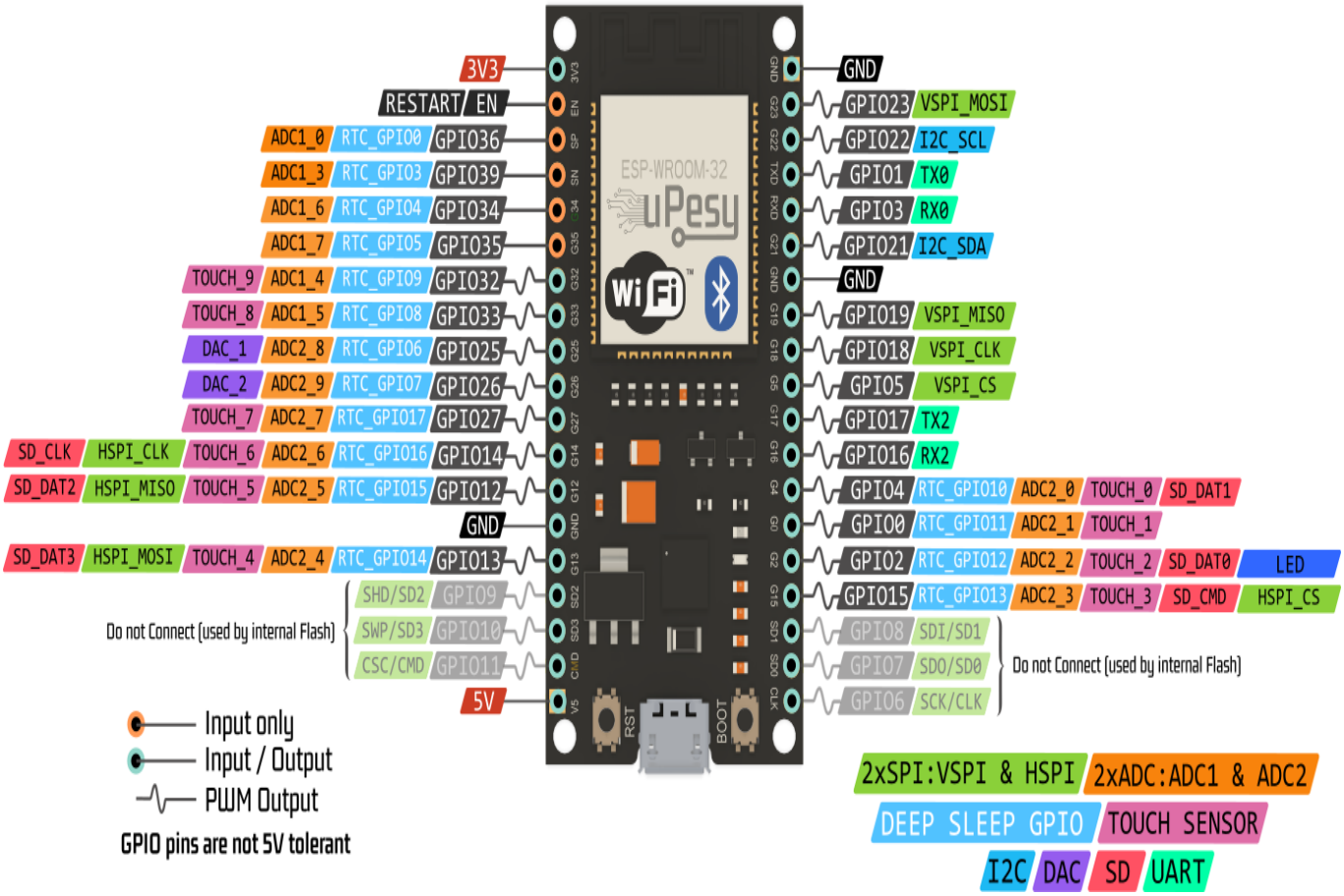
• 16 KByte SRAM in RTC

• QSPI Flash/SRAM, up to 4 x 16 MBytes

• Power supply: 2.2 V to 3.6 V



PIN DIADRAM OF ESP32:



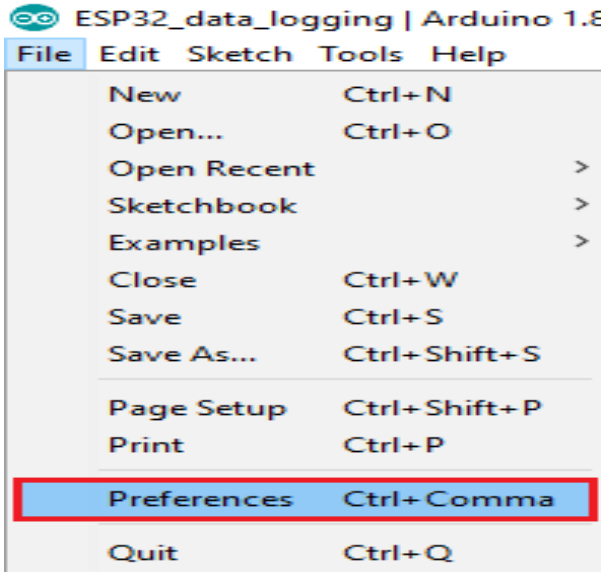
INSTRALLATION SOFTWARE AUDUINO IDE FOR PROGRAMING OF ESP32:

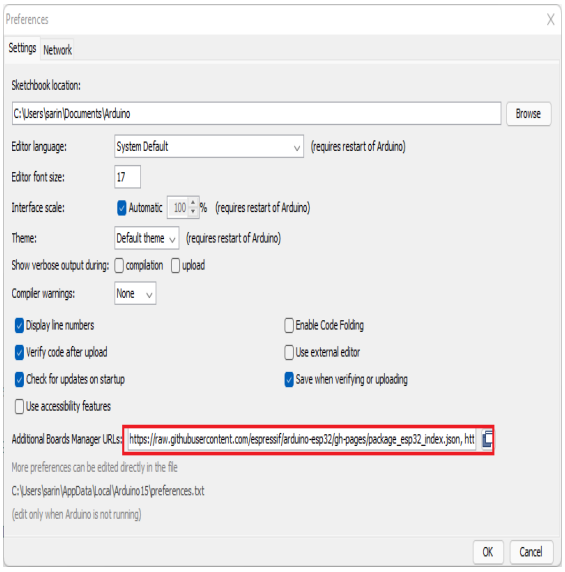
Download and Install Arduino IDE:

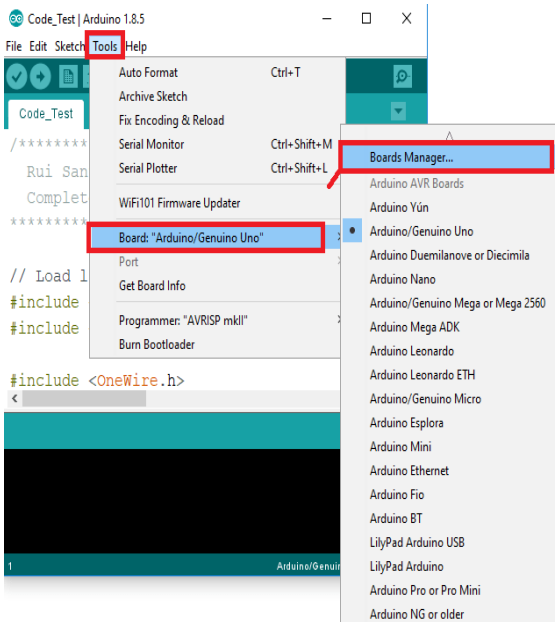
• Go to the official Arduino website: https://www.arduino.cc/en/software

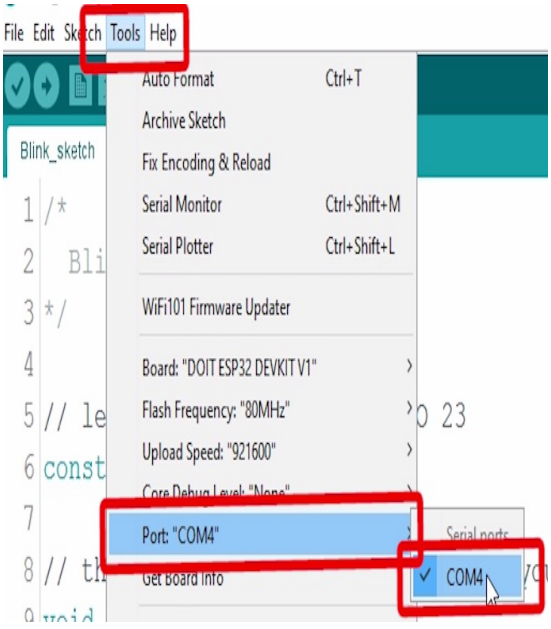
Enter Bootloader Mode (if necessary):

Some ESP32 boards require you to press and hold the "BOOT" button or "FLASH" button while uploading code. Refer to your board's documentation for specific instructions.





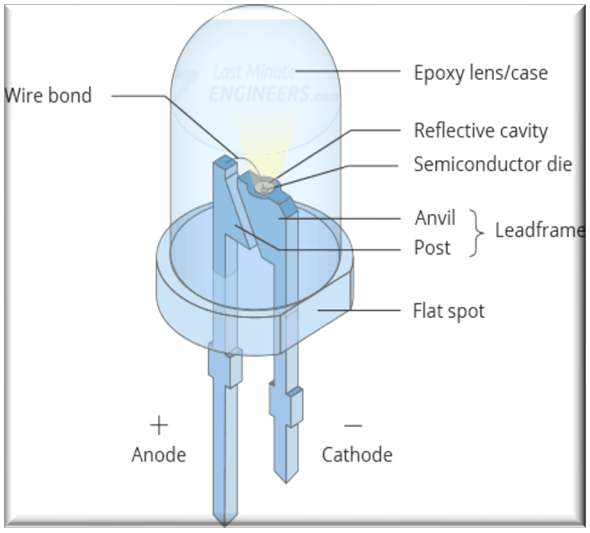




SENSOR INTERFACE & CODE WITH ESP32

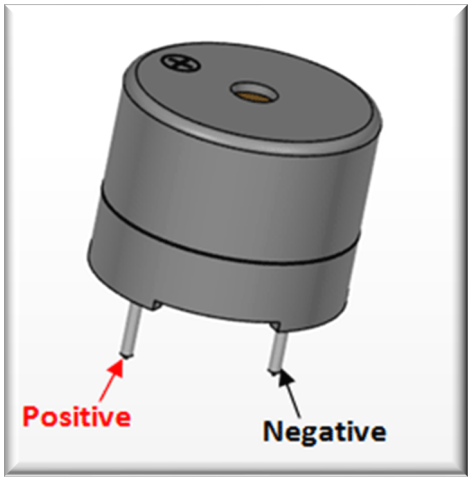
1.1 LED BLINK CODE

```
#define LED_BUILTIN 12
void setup()
      {
        pinMode(LED_BUILTIN,  OUTPUT);
      }
void loop()
  {
    digitalWrite(LED_BUILTIN,  HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN,  LOW);
    delay(1000);
  }
```



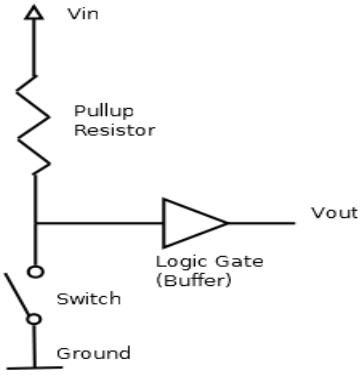
1.2 BUZZER INTERFACE

```
void setup()
        {
          pinMode(23,OUTPUT);
        }
void loop()
        {
          digitalWrite(23,HIGH);
          delay(100);
          digitalWrite(23,LOW);
          delay(1000);
        }
```



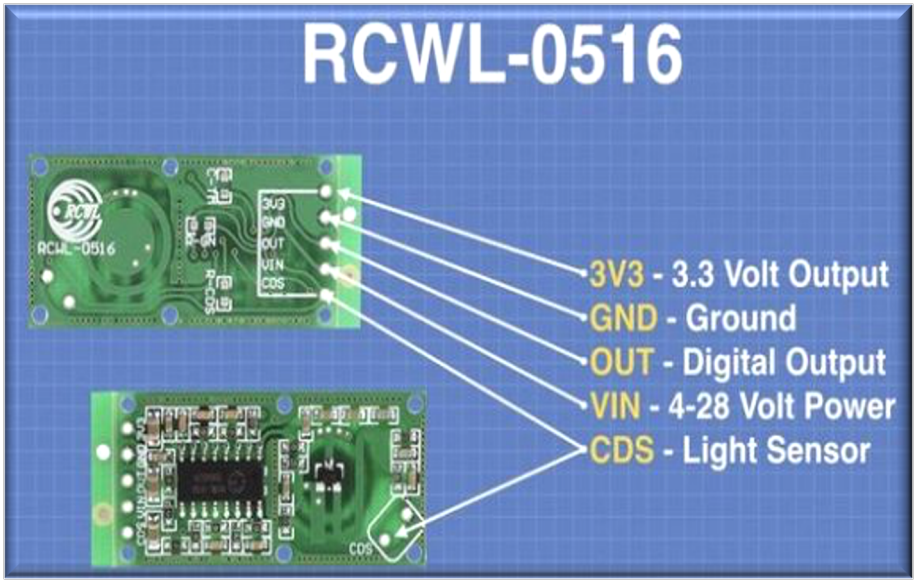
1.3 LED AND SWITCH

```
#define LED 12
#define SWITCH 34
bool ledstate = false;
void setup(){

          Serial.begin(115200);
          pinMode(LED,OUTPUT);
          pinMode(SWITCH,INPUT);
        }
void loop() {
             int switchstate = digitalRead(SWITCH);
            if(switchstate==LOW)
       {
             ledstate =!ledstate;
             digitalWrite(LED,ledstate);
       }

    }
```

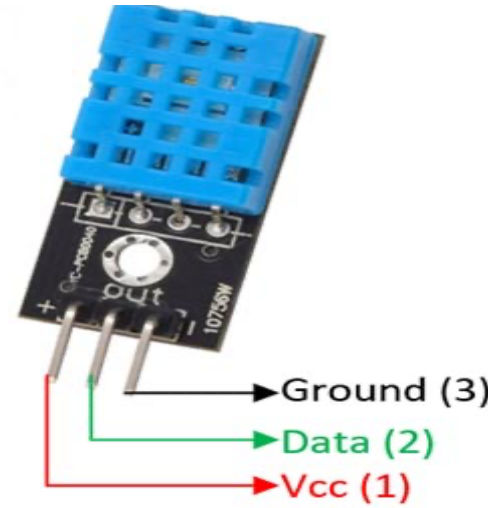


1.4. PIR SENSOR RCWL-0561

```
#define PIR_PIN 23
#define LED 12
void setup() {
  Serial.begin(115200);
  pinMode(PIR_PIN, INPUT);
  pinMode(12, OUTPUT);
}
void loop() {
  int motionDetected = digitalRead(PIR_PIN);

  if (motionDetected == HIGH){

    Serial.println("Motion detected");
    digitalWrite(LED,HIGH);

  } else
  {
    Serial.println("0");
    digitalWrite(LED,LOW);

  }
}
```

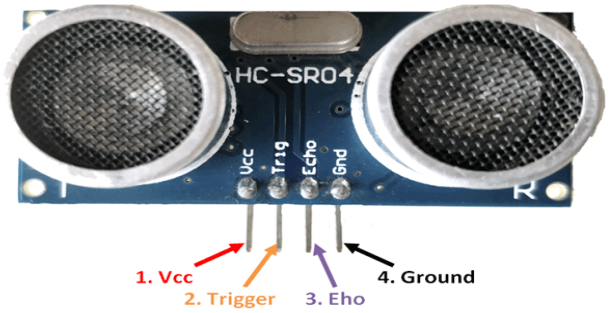


1.5 TEMPRETURE & HUMIDITY SENSOR:




```
#include <Bonezegei_DHT11.h>

#include <DHT.h>

#define DHTPIN 2 // Digital pin connected to the
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

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

void loop() {
  delay(200);  // Delay between sensor readings

  float temperature = dht.readTemperature(); // Read temperature as
Celsius
  float humidity = dht.readHumidity(); // Read humidity

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Temperature: ");// Print temperature and humidity
  Serial.print(temperature);
  Serial.print(" °C\t");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");
}
```

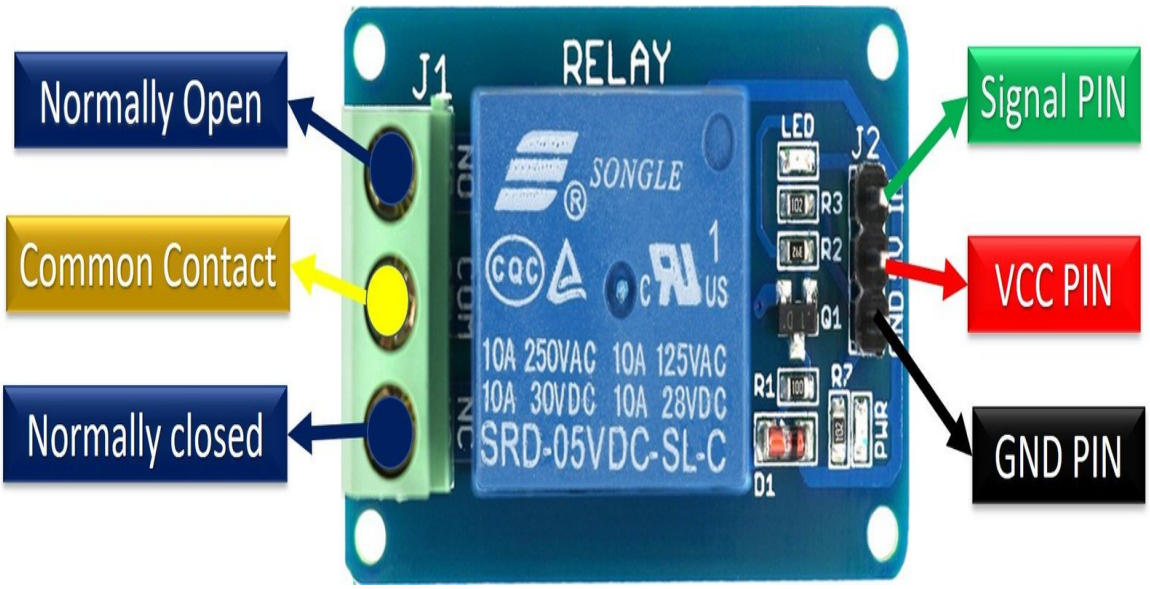
1.6 ULTRASONIC SENSOR:




```
#define TRIG_PIN 23
#define ECHO_PIN 22
#define BUZZER_PIN 18

void setup() {
          Serial.begin(115200);
          pinMode(TRIG_PIN, OUTPUT);
          pinMode(ECHO_PIN, INPUT);
          pinMode(BUZZER_PIN, OUTPUT);
       }
void loop() {
            long duration, distance;
            digitalWrite(TRIG_PIN, LOW); // Clear the trigger pin
            delayMicroseconds(2);

            digitalWrite(TRIG_PIN, HIGH);// Send a 10 microsecond
pulse to trigger pin
            delayMicroseconds(10);
            digitalWrite(TRIG_PIN, LOW);

            duration = pulseIn(ECHO_PIN, HIGH);// Measure the pulse
duration on the echo pin


            distance = duration * 0.034 / 2; // Calculate the
distance

            Serial.print("Distance: ");
            Serial.print(distance);
            Serial.println(" cm");

  if (distance < 20) {
              tone(BUZZER_PIN, 1000);
              delay(1000);
              noTone(BUZZER_PIN);
            }
  delay(500);
}
```

1.7 RELAY SENSOR:

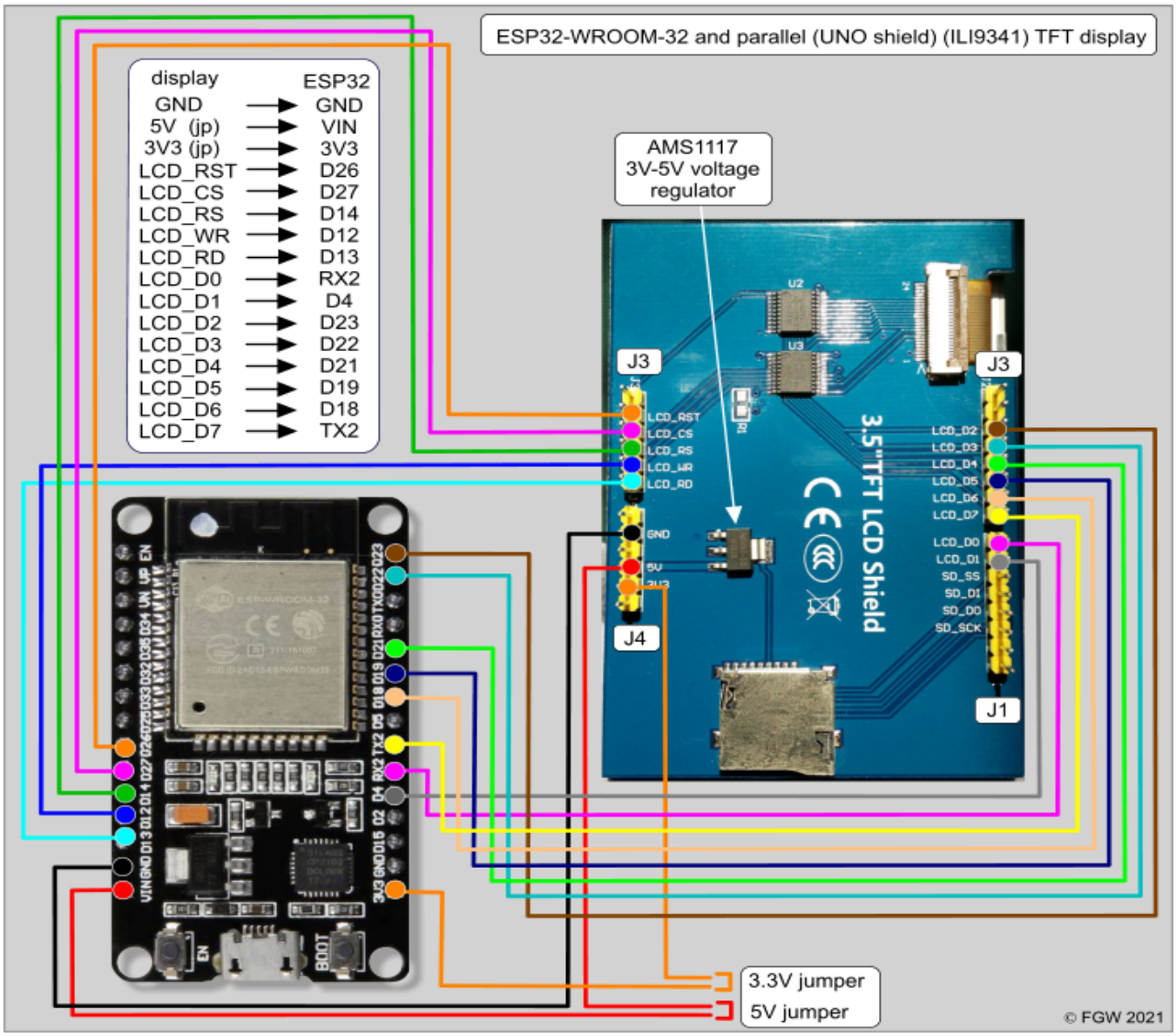
```
const int relay = 3;

void setup(){
            pinMode(2,OUTPUT); //relay satup
        }
void loop(){
          digitalWrite(relay,HIGH);  // realY HIGH
          delay(1000);
          digitalWrite(relay,LOW);
          delay(1000);
        }
```



1.8 Wi-fi connection code

```
#include <WiFi.h>

const char* ssid = "Embeddedvisions";
const char* password = "Swizz@79";

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

  Serial.println("Connecting to WiFi...");

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting...");
  }
  Serial.println("Connected to the WiFi network");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}
void loop() {     // Your code here
}
```

1.9 3.5 TFT LCD INTERFACE



Note: contact us for code

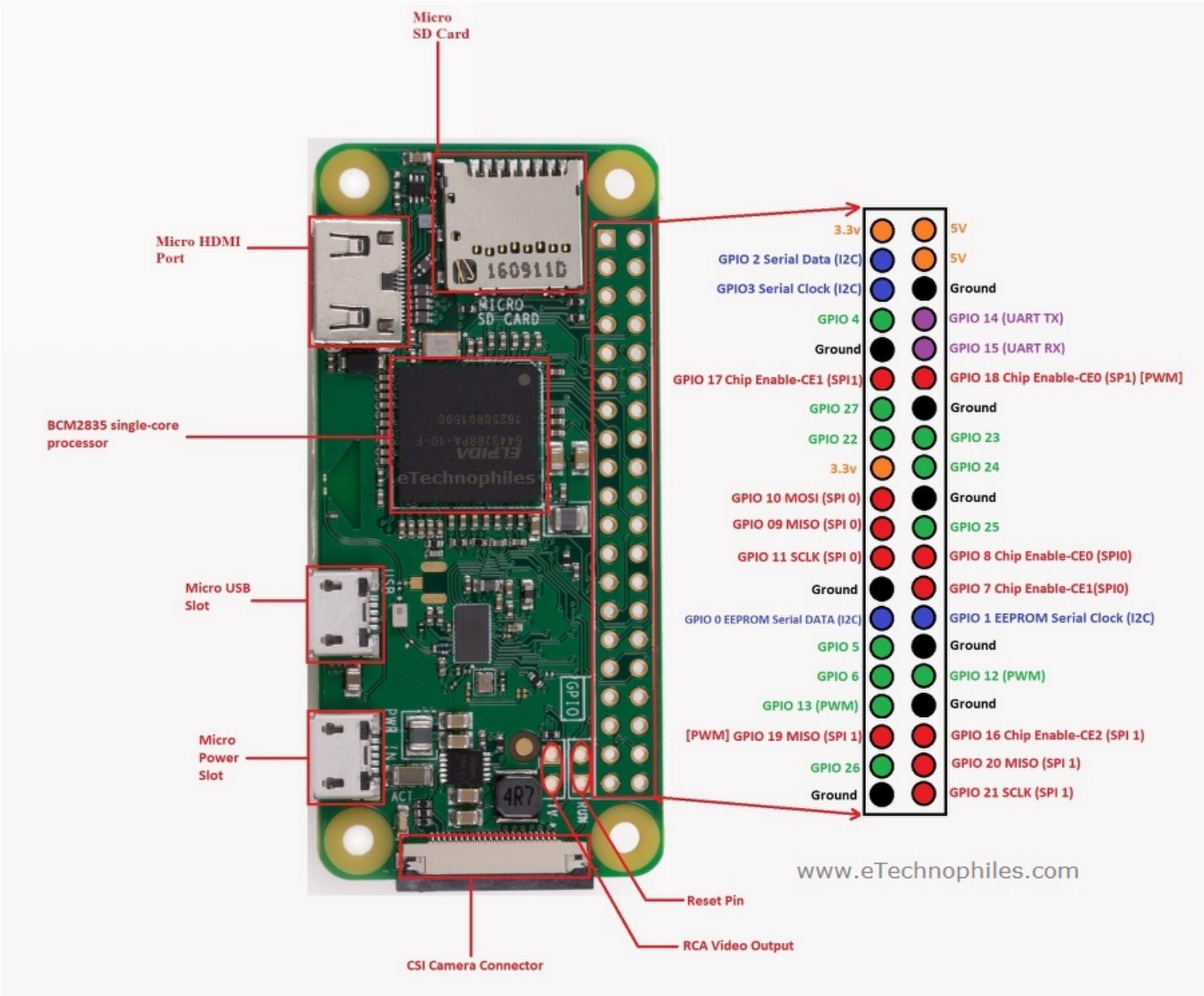
Contact No. : 8982754201

Mail id : embeddedvisions@gmail.com

2. INTRODUCTION OF RASPBERRY PI ZERO:

The board incorporates a quad-core 64-bit Arm Cortex-A53 CPU, clocked at 1GHz. At its heart is a Raspberry Pi RP3A0 system-in-package (SiP), integrating a Broadcom BCM2710A1 die with 512MB of LPDDR2 SDRAM.

PIN DAIGRAM



Get the Necessary Hardware:

- Raspberry Pi board (such as Raspberry Pi 4)
- MicroSD card (at least 8GB, class 10 recommended)
- Power adapter (compatible with the Raspberry Pi)
- Keyboard and mouse
- Monitor or TV with HDMI input
- HDMI cable
- Optional: Raspberry Pi case, cooling fan/heatsinks

2. Download Operating System:

- Go to the official Raspberry Pi website (https://www.raspberrypi.org/) and download the latest version of the Raspberry Pi OS (formerly known as Raspbian).

3. Write OS Image to MicroSD Card:

- Use a tool like Etcher (https://www.balena.io/etcher/) to write the downloaded OS image to the microSD card.

4. Boot Up Raspberry Pi:

- Insert the microSD card into the Raspberry Pi.
- Connect the HDMI cable from the Raspberry Pi to your monitor or TV.
- Connect the keyboard and mouse to the USB ports.
- Plug in the power adapter to turn on the Raspberry Pi.

5. Initial Setup:

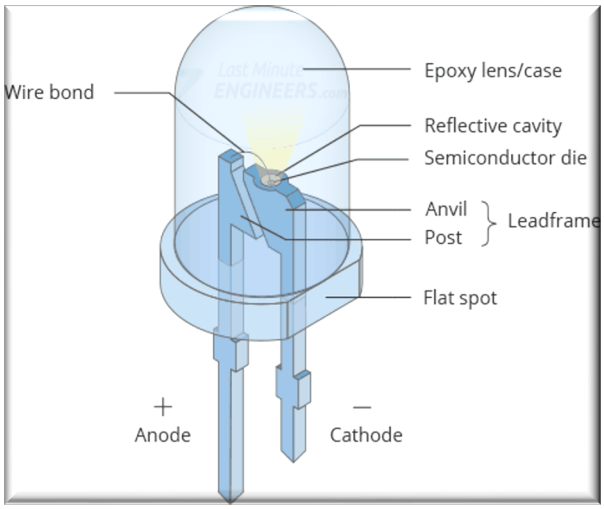
- Follow the on-screen instructions to complete the initial setup, including language selection, password setup, and updating the system.
- If you didn't set up Wi-Fi earlier, you can do it through the desktop environment by clicking on the network icon in the taskbar.

6. Update Software:

- Open a terminal and run the following commands to update the software:
- sudo apt update
- sudo apt upgrad

SENSOR INTERFSCE WITH RPI ZERO

2.1LED BLINK




```
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<time.h>
#include<signal.h>
#include<stdio.h>
int main()
{       int fd;
        int i;
        char buf[128];
        int gpio = 20;
        fd=open("/sys/class/gpio/export", O_WRONLY);
        sprintf(buf,"%d",gpio);
        write(fd, buf, strlen(buf));
        close(fd);
        sprintf(buf,"/sys/class/gpio/gpio%d/direction",gpio);
        fd=open(buf, O_WRONLY);
        write(fd,"out",3);
        close(fd);
        sprintf(buf,"/sys/class/gpio/gpio%d/value",gpio);
        fd=open(buf,O_WRONLY);
        for(i=0;i<=10;i++)
        {       write(fd,"1",1);
                sleep(1);
                write(fd,"0",1);
                sleep(1);}
        close(fd);}
```

STM32

Introduction

The STM32F405xx and STM32F407xx family is based on the high-performance Arm® Cortex®-M4 32-bit RISC core operating at a frequency of up to 168 MHz. High-speed embedded memories (Flash memory up to 1 Mbyte, up to 192 Kbytes of SRAM), up to 4 Kbytes of backup SRAM

Pin Diagram