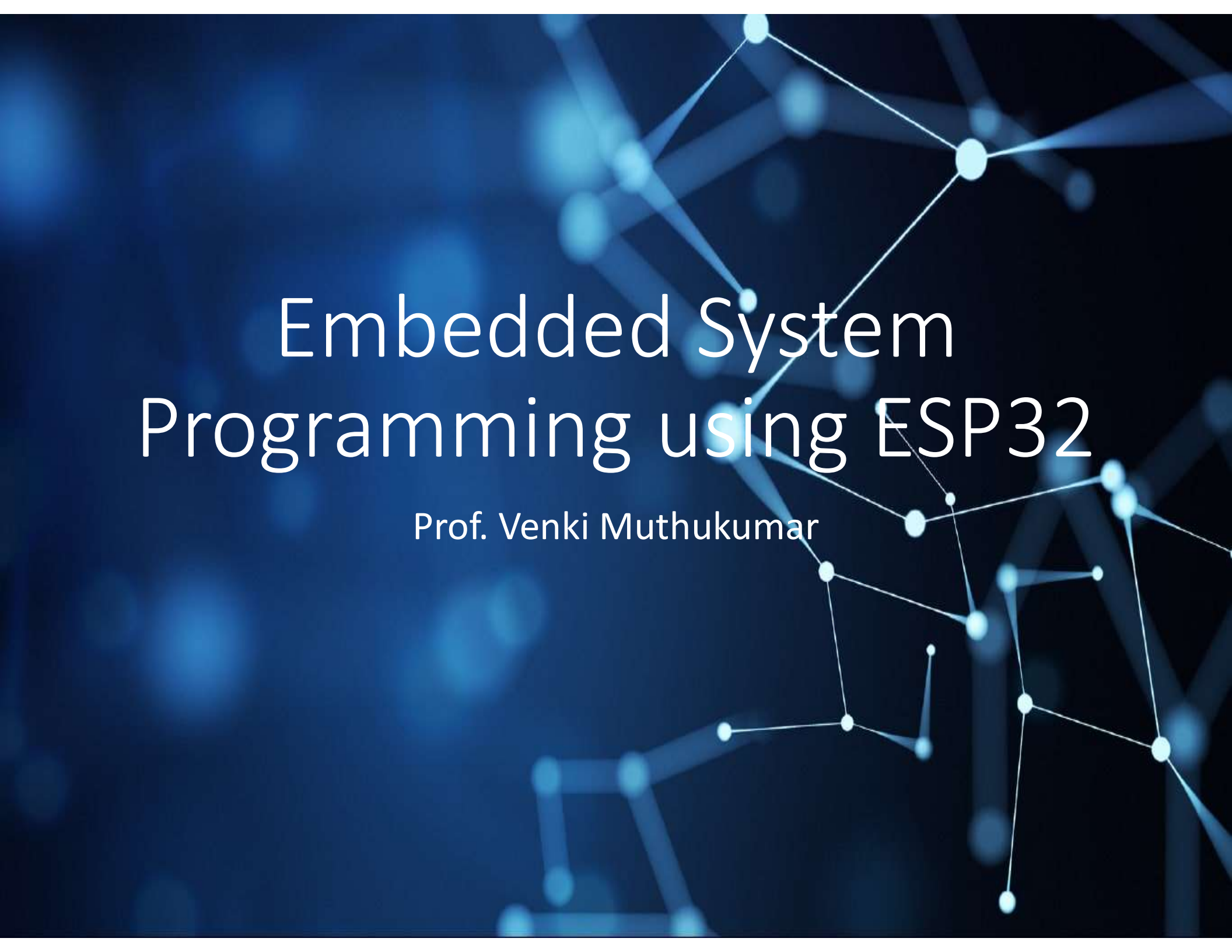


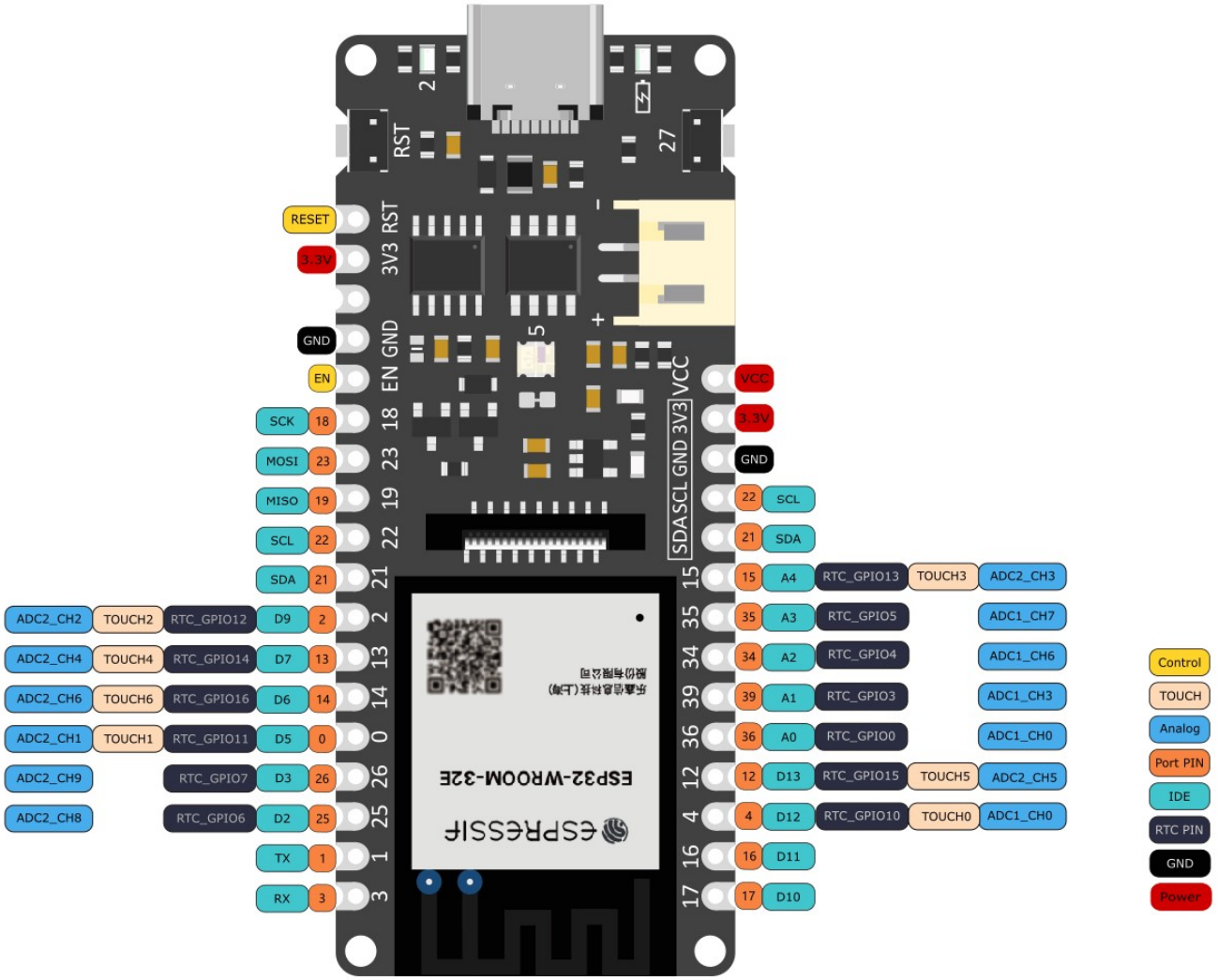
Embedded System Programming using ESP32

Prof. Venki Muthukumar

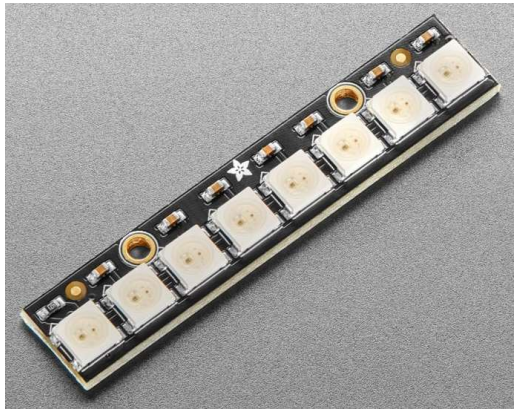
List of Componets

- Microcontroller – ESP32-E
- Sensors
- Motors & Driver
- Header Wires
- Battery & Regulators
- Addon Boards

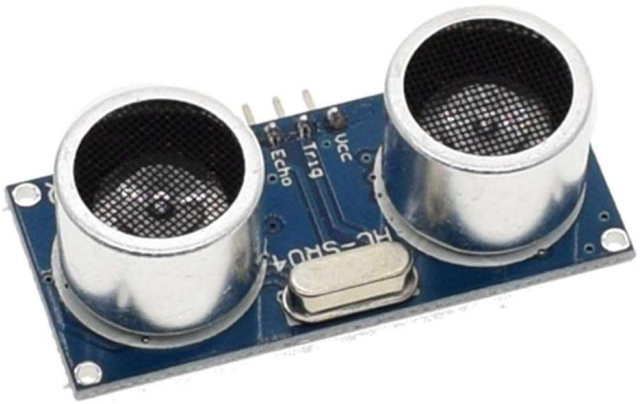
FireBeetle ESP32-E

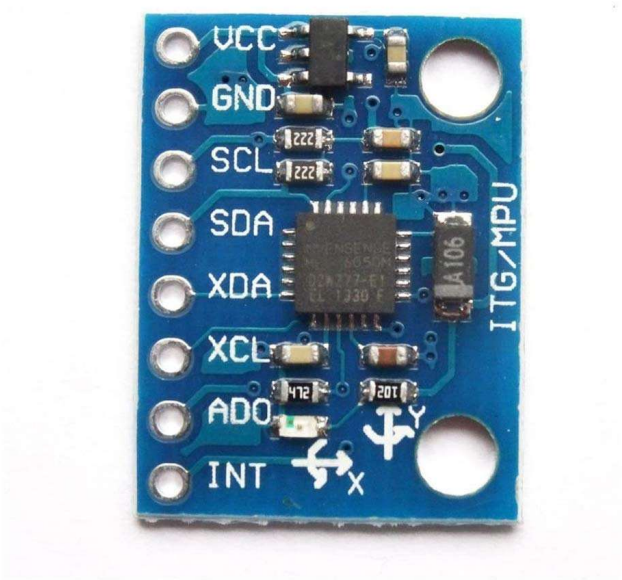


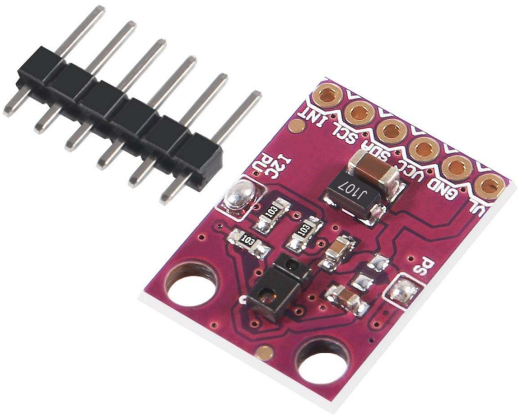
Lights/LEDs



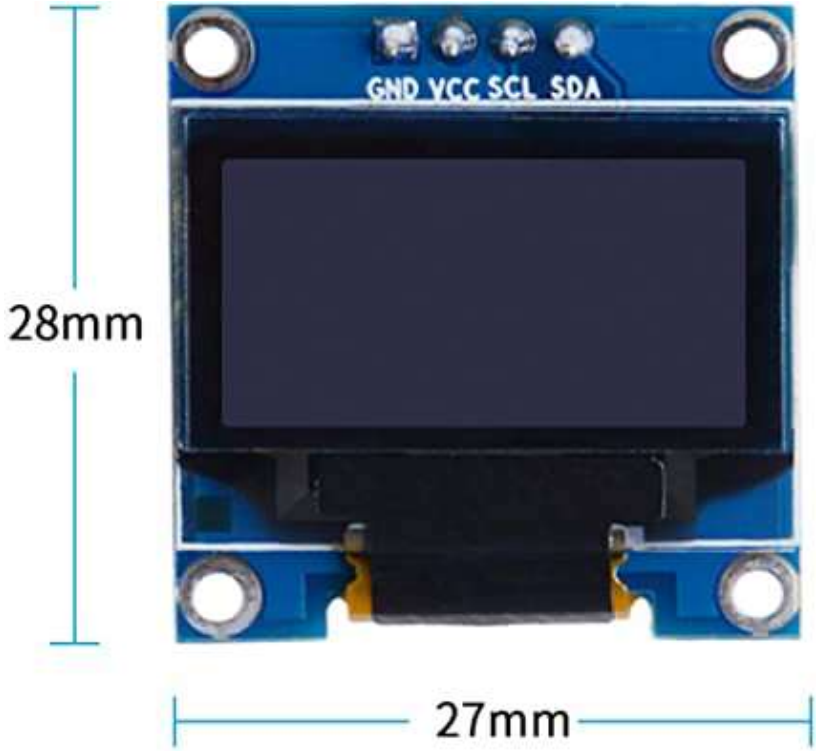
Sensors


Echo
Trig
Vcc
HC-SR04


VCC
GND
SCL
SDA
XDA
XCL
ADO
INT
ITG/MPU

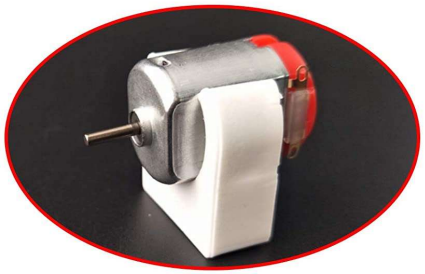

INT
SCL
SDA
VCC
GND

Display

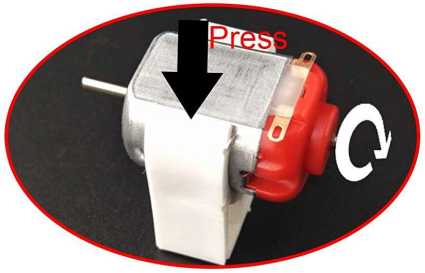


Motors

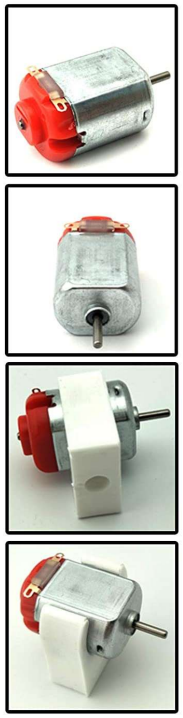
Two steps to complete the installation

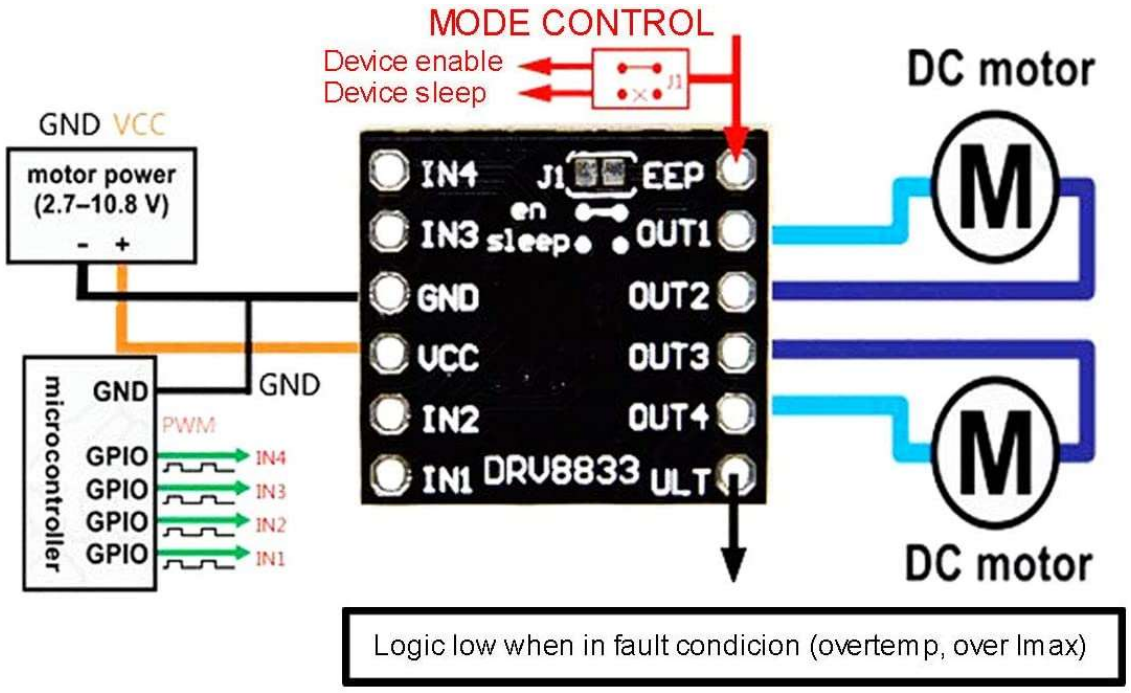


Step 1:Install the motor vertically



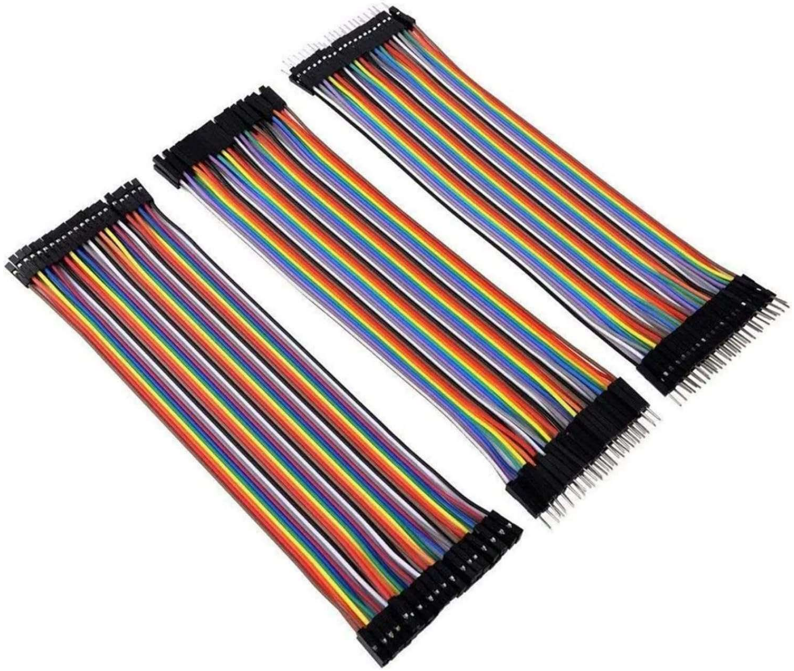
Step 2:Press the motor to rotate



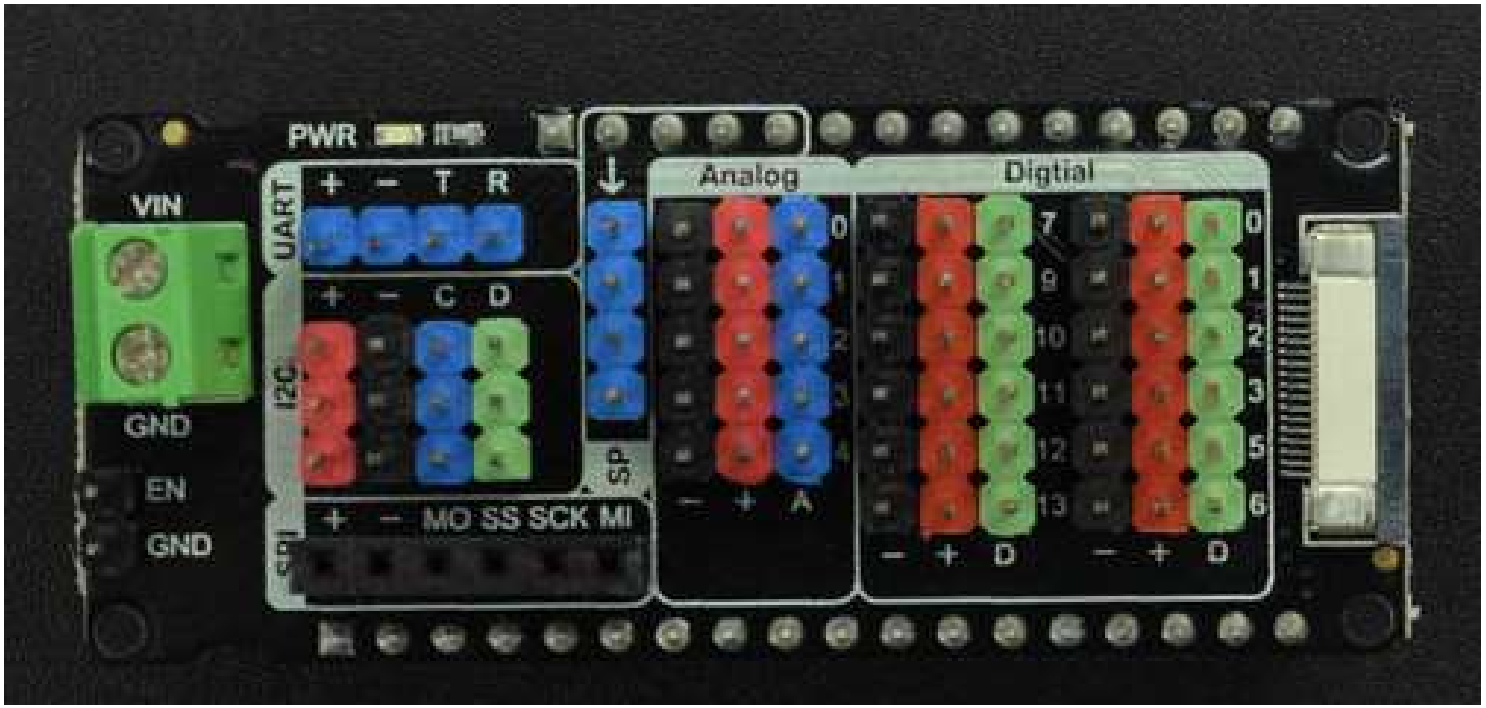


Logic low when in fault condicion (overtemp, over Imax)

Others

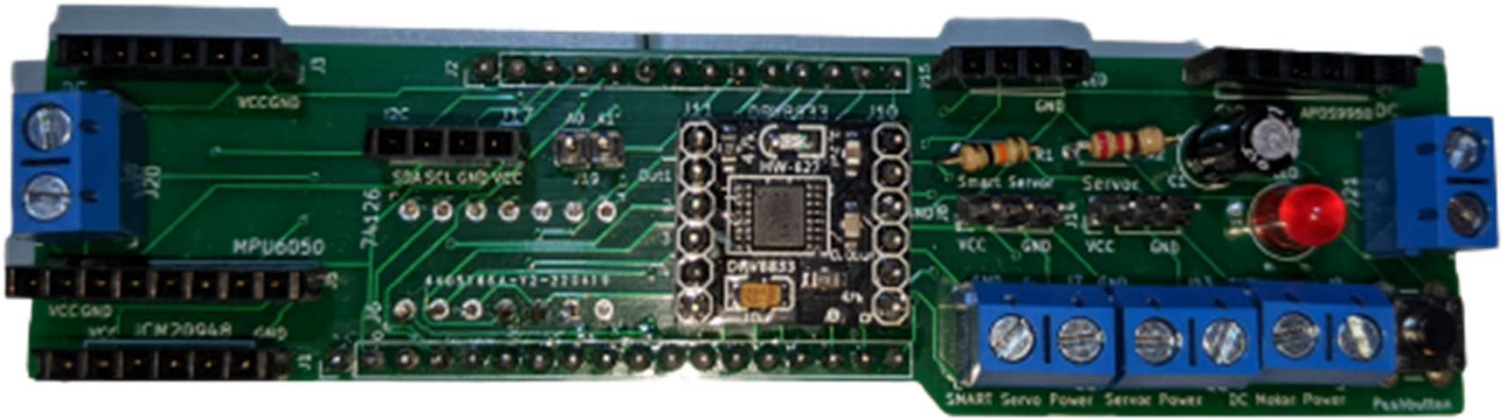


Making Connections



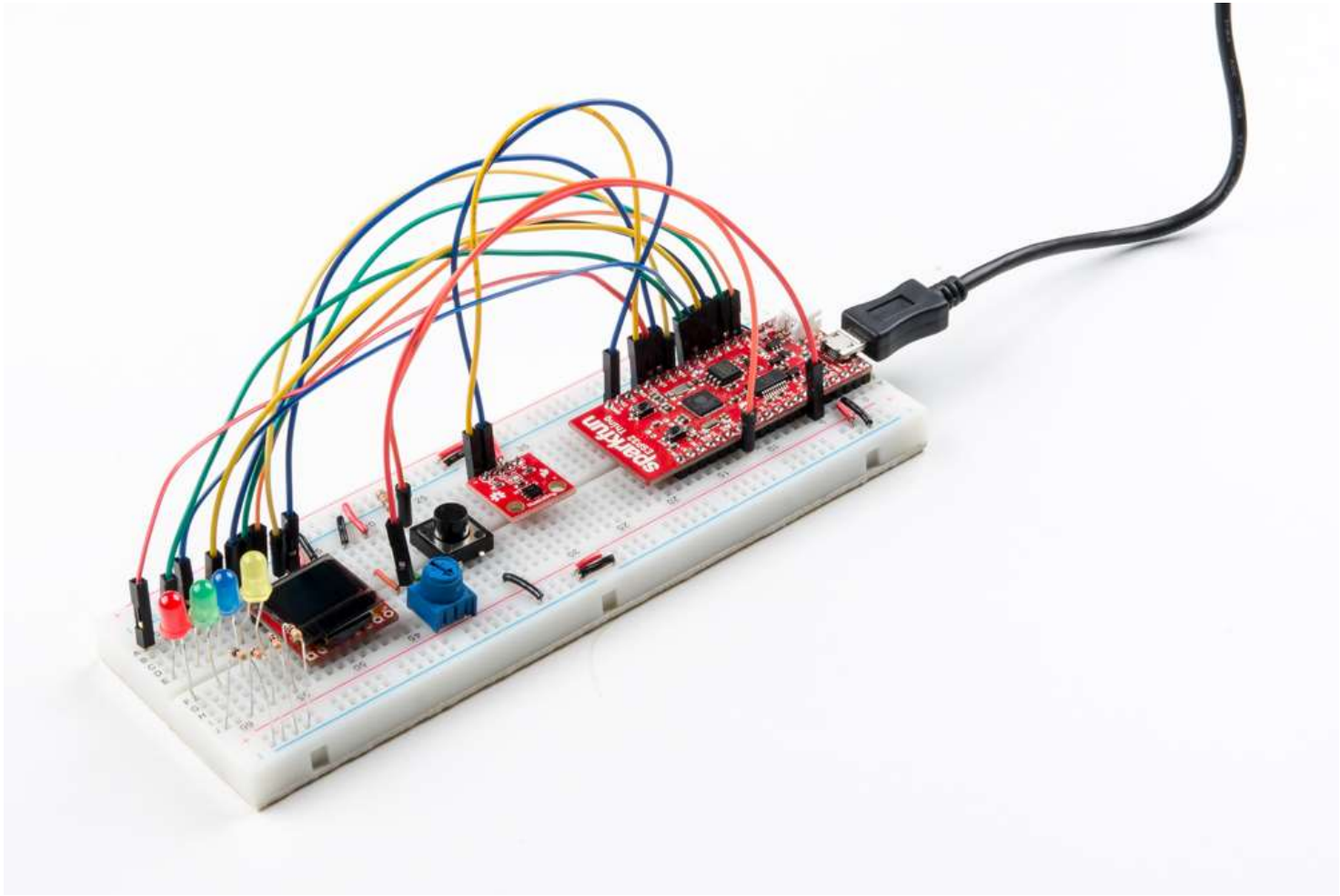
Gravity: IO Shield for FireBeetle 2 (ESP32-E/M0)

Making Connections

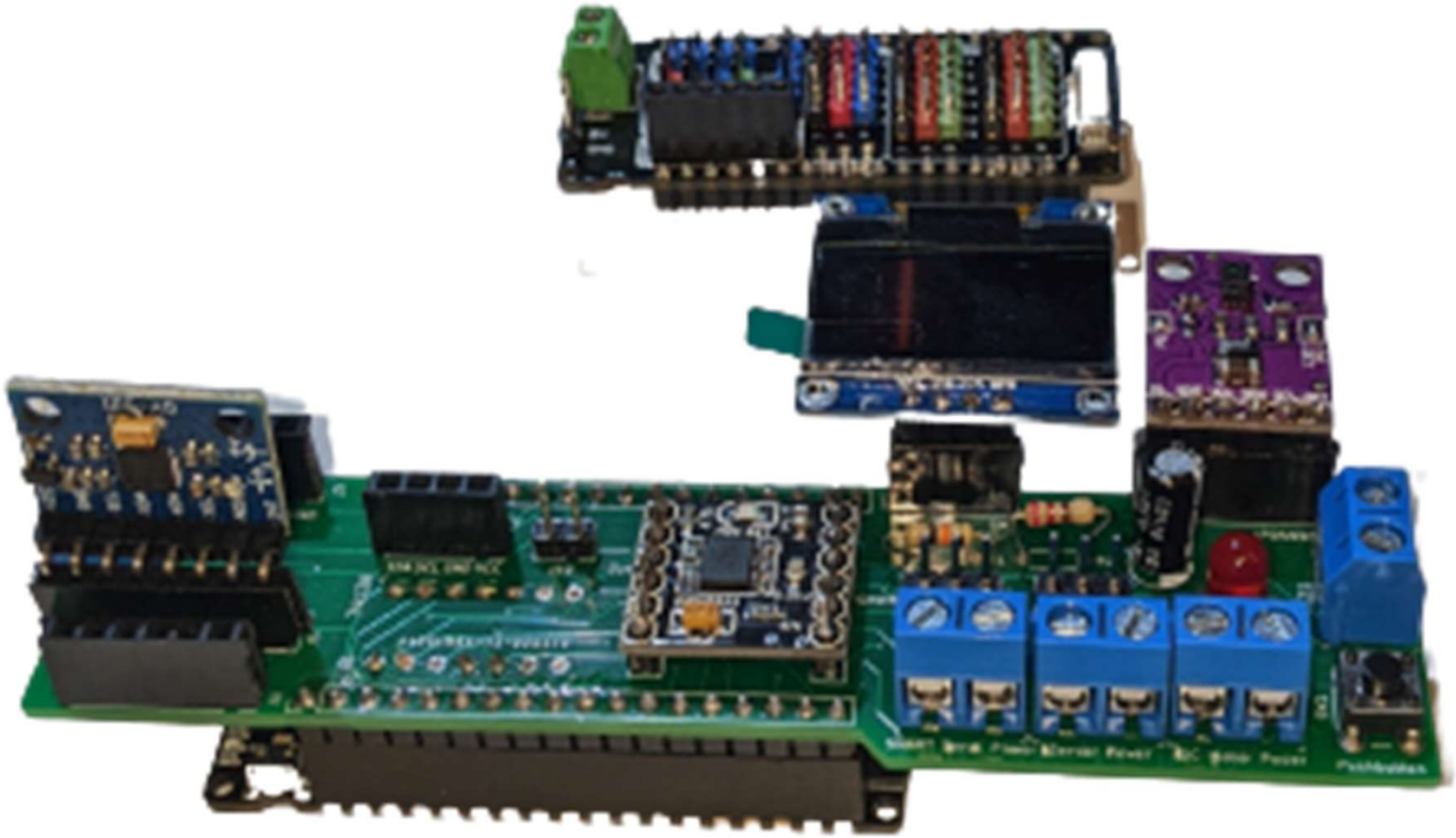


Our Custom Addon Board

A Typical Connection



Making your life easier?

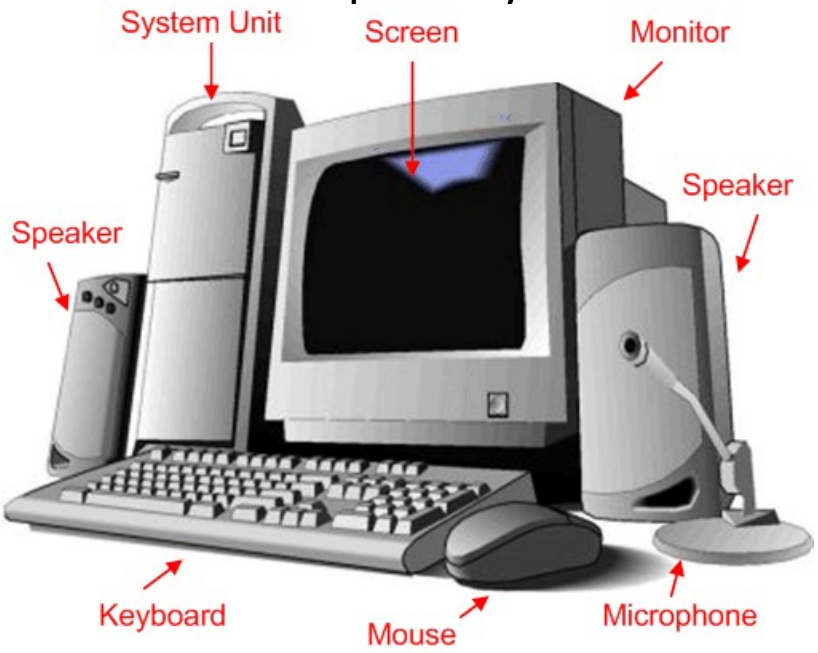


What is Embedded Systems?

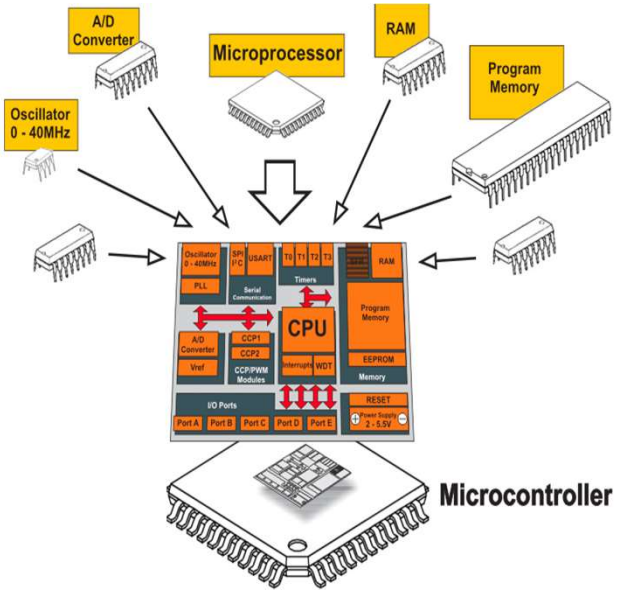
- An **embedded system** is a special kind of computer that is designed to do one specific job. Unlike your regular computers or embedded system can only perform one or few tasks. It's like a tiny brain (**microcontroller - uC**) that makes sure these devices can do their job correctly.

Computer System Vs Microcontroller

Computer System

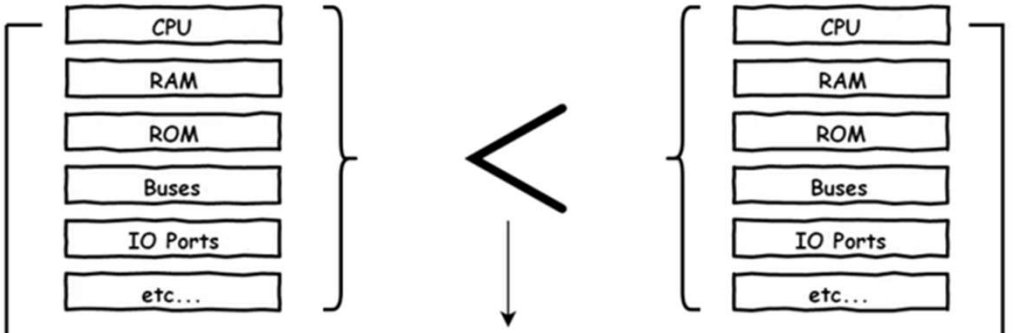


Microcontroller (uC)



Microcontroller

Computer



Computer is Larger Than ">" Microcontroller While Having The Same Components

	Typical Microcontroller	Typical Computer
CPU Speed	~16 DMIPS @ ~16MHz	~2000 DMIPS @ ~1GHz
RAM	~1KB	~8GB
Main Memory	~1KB	~1TB
Power Consupmtion	~1 mW	~150W
IO Ports	Parallel, serial RS-232, USB, etc	Prallel, Serial RS-232, USB, HDMI, etc

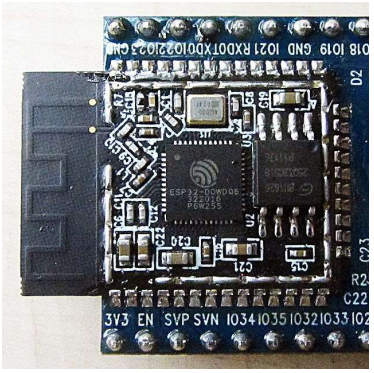
The CPU of a Microcontroller is called **Microprocessor**

The CPU of a Computer is called **Processor**

These Figures Are Not Exact By Any Means, I've Just Made Them Up For Demonstration Purposes Only!

ESP32-E

- Processer: Tensilica LX6 dual core processer (One for high speed connection; one for independent programing).
- Frequency: 240MHz
- SRAM: 520KB
- Flash: 16Mbit
- Operating voltage: 3.3V
- Input voltage: 3.3V~5.5V

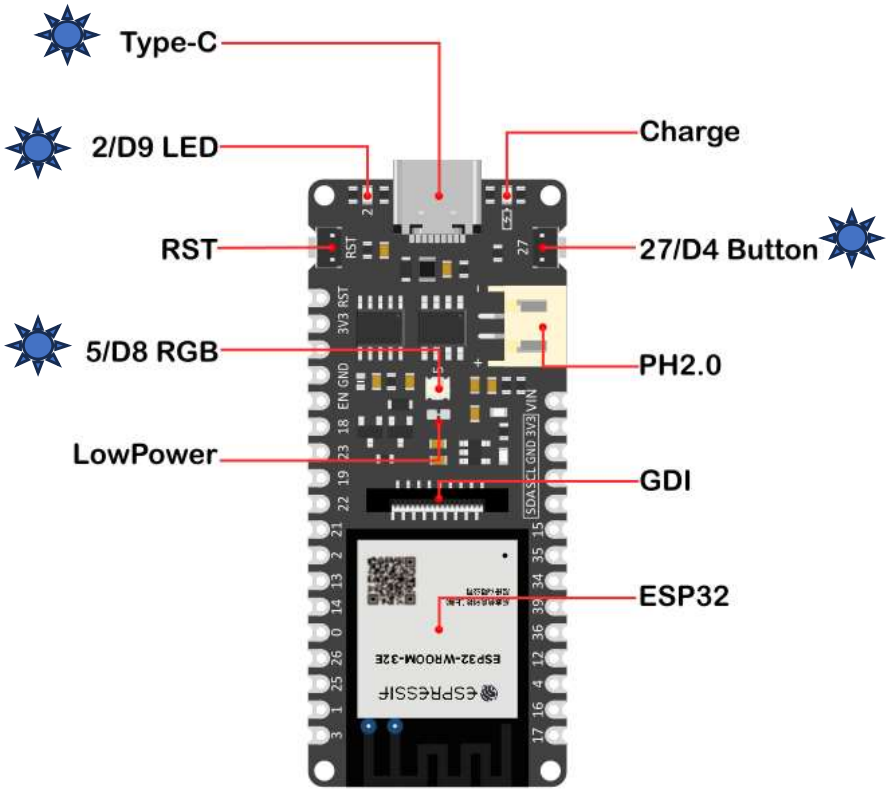


- Support electric current of low power consumption: 10 μA
- Support maximum discharge current: 600mA@3.3V LDO
- Support maximum charge current: 500mA
- Wi-Fi standard - Wi-Fi protocol: 802.11 b/g/n/d/e/I/k/r
- Frequency range: 2.4~2.5 GHz
- Bluetooth protocol: Comply with BR/EDR/BLE standard of Bluetooth v4.2.

ESP32-E (Inputs & Outputs)

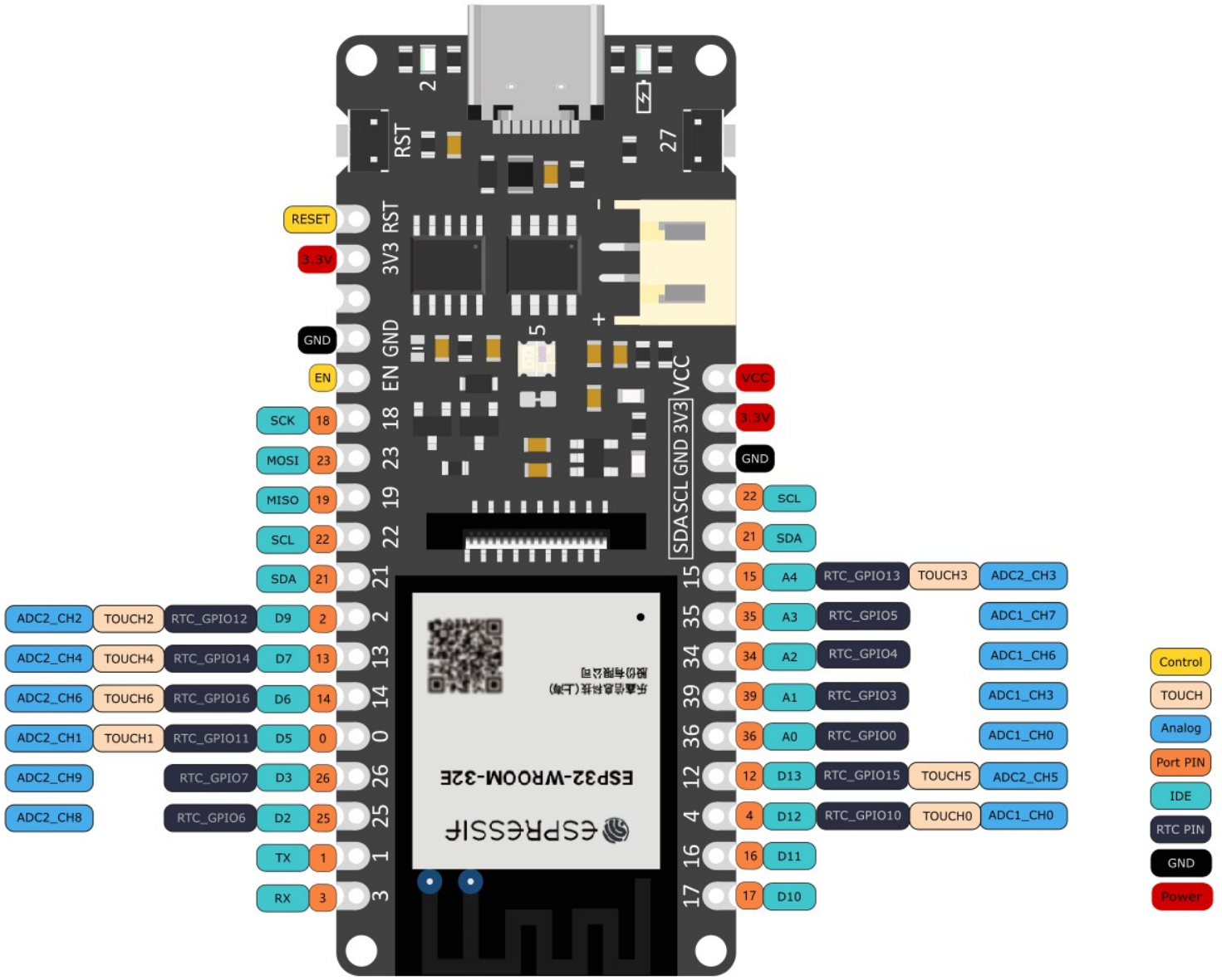
Why do we need inputs and outputs?

- On-chip clock: 40MHz crystal and 32.768 KHz crystal.
- Digital I/O: 10 (default setting of arduino)
- Analog input: 5(default setting of arduino)
- SPI: 1 (default setting of arduino)
- I2C: 1 (default setting of arduino)
- I2S: 1 (default setting of arduino)
- RGB_LED: 5/D8



What is Digital I/O and Analog I/P?

FireBeetle ESP32-E

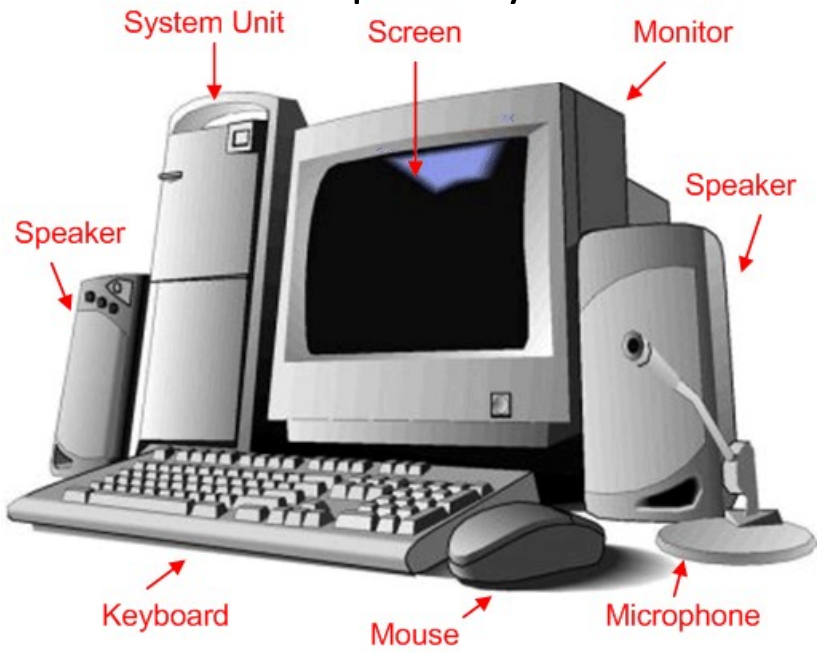


How do you program the uC?

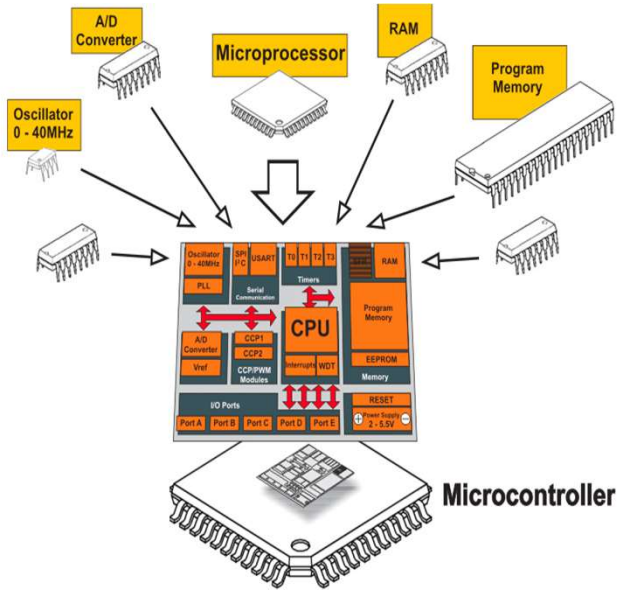
- You need a language to program the uC
  - Assembly language – very difficult and complex – efficiency is best
  - C/C++ language – difficult – good
  - Object oriented language – easy – okay
  - Python language – very easy – not so good
  - Scratch – super easy – bad.

Computer System Vs Microcontroller

Computer System



Microcontroller



Microcontroller

CPU
RAM
ROM
Buses
IO Ports
etc...

Computer

CPU
RAM
ROM
Buses
IO Ports
etc...

Computer is Larger Than ">" Microcontroller While Having The Same Components

	Typical Microcontroller	Typical Computer
CPU Speed	~16 DMIPS @ ~16MHz	~2000 DMIPS @ ~1GHz
RAM	~1KB	~8GB
Main Memory	~1KB	~1TB
Power Consupmtion	~1 mW	~150W
IO Ports	Parallel, serial RS-232, USB, etc	Prallel, Serial RS-232, USB, HDMI, etc

The CPU of a Microcontroller is called **Microprocessor**

The CPU of a Computer is called **Processor**

These Figures Are Not Exact By Any Means, I've Just Made Them Up For Demonstration Purposes Only!

Introduction to Arduino

- open-source electronics prototyping platform based on flexible, easy-to-use hardware and software (hw/sw platform for physical computing)
- Hardware
  - Arduino Board (ESP32-E (firebeetle 2 – ESP32-E))
- Programming Language
  - Arduino Programming Language (Object Oriented C)
- Programming Tool (IDE)
  - Arduino Programming Environment (Arduino IDE)

Arduino Programming Language

- core: functions for
  - digital/analog input/output
  - serial communication
  - Time
  - Interrupts
- libraries for
  - controlling stepper motors
  - reading and writing to EEPROM ("permanent" storage)
  - software implementation of serial protocol
  - controlling led ...
- See the supported Libraries @
  - https://www.arduinolibraries.info/architectures/esp32

Installing the ESP32 FireBeetle Board

- 1) Open the preferences window from the Arduino IDE. Go to **File**> **Preferences.**
- Enter **https://dl.espressif.com/dl/package_esp32_index.json** into the “Additional Board Manager URLs” field as shown in the figure below. Then, click the “OK” button.
- Open boards manager. Go to **Tools** > **Board** > **Boards Manager…**
- Search for ESP32 (version <2.0.16) and press install button
- Go to Select Other Board and Port
  - Check for “**FireBeetle 2 ESP32-E**“

Testing the Installation

- Open the Arduino IDE
- Select your Board in **Tools** > **Board** menu (**FireBeetle 2-ESP32-E**)
- Select the Port (if you don't see the COM Port in your Arduino IDE, you need to install the ESP32 CP210x USB to UART Bridge VCP Drivers – windows only – see prev slide)
- Open the following example under **File** > **Examples** > **Digital** > **BlinkwithoutDelay**
- Add this line on the top
  - #define LED_BUILTIN 2
- Press the **Upload** button in the Arduino IDE. Wait a few seconds while the code compiles and uploads to your board.
- If everything went as expected, you should see a "**Done uploading.**" message.
- The board LED should blink

Arduino Terminology

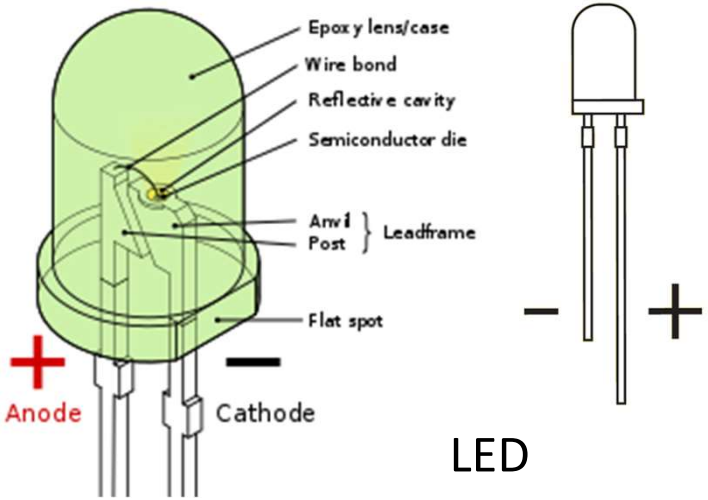
- "*sketch*" – a program you write to run on an Arduino board
- "*pin*" – an input or output connected to something.
  - e.g. output to an LED, input from a knob.
- "*digital*" – value is either HIGH or LOW.
  - (aka on/off, one/zero) e.g. switch state
- "*analog*" – value ranges, usually from 0-255.
  - e.g. LED brightness, motor speed, etc.

Arduino "Language"

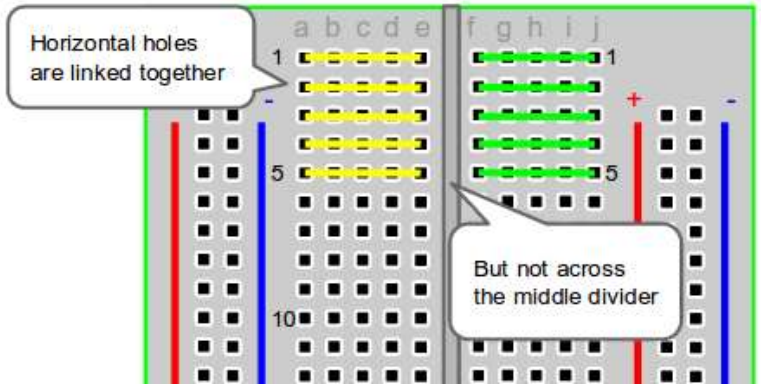
- Language is standard C (but made easy)
- Lots of useful functions
  - `pinMode()` – set a pin as input or output
  - `digitalWrite()` – set a digital pin high/low
  - `digitalRead()` – read a digital pin's state
  - `analogRead()` – read an analog pin
  - `analogWrite()` – write an "analog" value
  - `delay()` – wait an amount of time
  - `millis()` – get the current time
- And many others. And libraries add more.

Before We begin!

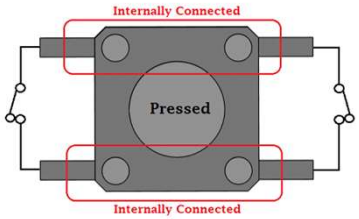
- Using the Breadboard

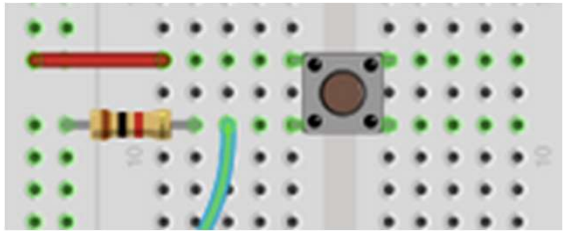


LED

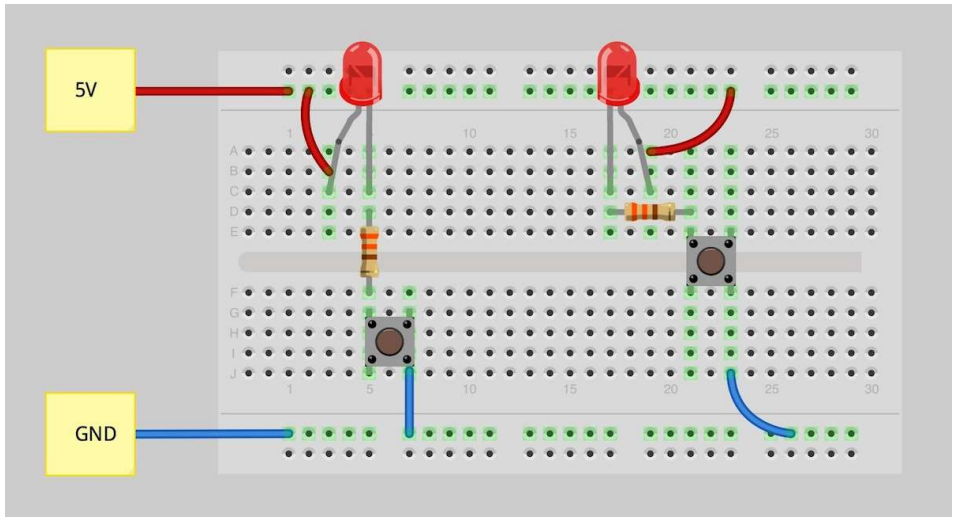


Breadboard

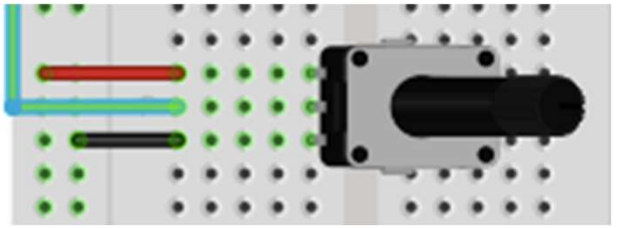




Push Button



Potentiometer



First Run – Hello Blink!

BlinkWithoutDelay.ino

```
40  unsigned long previousMillis = 0;  // will store last time LED was updated
41
42  // constants won't change:
43  const long interval = 1000;  // interval at which to blink (milliseconds)
44
45  void setup() {
46    // set the digital pin as output:
47    pinMode(ledPin, OUTPUT);
48  }
49
50  void loop() {
51    // here is where you'd put code that needs to be running all the time.
52
53    // check to see if it's time to blink the LED; that is, if the difference
54    // between the current time and last time you blinked the LED is bigger than
55    // the interval at which you want to blink the LED.
56    unsigned long currentMillis = millis();
57
58    if (currentMillis - previousMillis >= interval) {
59      // save the last time you blinked the LED
60      previousMillis = currentMillis;
61
62      // if the LED is off turn it on and vice-versa:
63      if (ledState == LOW) {
64        ledState = HIGH;
65      } else {
66        ledState = LOW;
67      }
68
69      // set the LED with the ledState of the variable:
70      digitalWrite(ledPin, ledState);
71    }
72  }
73
```

Success

BlinkWithoutDelay.ino

```
40  unsigned long previousMillis = 0;  // will store last time LED was updated
```

Output

```
Sketch uses 237185 bytes (18%) of program storage space. Maximum is 1310720 bytes.
Global variables use 21056 bytes (6%) of dynamic memory, leaving 306624 bytes for local variables. Maximum is 327680 bytes.
esptool.py v4.5.1
Serial port /dev/cu.wchusbserial1110
Connecting....
Chip is ESP32-D0WD-V3 (revision v3.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 08:3a:f2:39:25:90
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00005fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Flash will be erased from 0x0000e000 to 0x0000ffff...
Flash will be erased from 0x00010000 to 0x00049fff...
Compressed 18992 bytes to 13110...
Writing at 0x00001000... (100 %)
Wrote 18992 bytes (13110 compressed) at 0x00001000 in 0.4 seconds (effective 349.8 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 146...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (146 compressed) at 0x00008000 in 0.1 seconds (effective 338.6 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 592.2 kbit/s)...
Hash of data verified.
Compressed 237552 bytes to 130756...
Writing at 0x00010000... (12 %)
Writing at 0x0001d169... (25 %)
Writing at 0x00024374... (37 %)
Writing at 0x00029553... (50 %)
```

Understand the code

sketch_may22a.ino

```
1   // Define directives
2   //# define name value
3   #define X 1
4   // Define global variables
5   //typecast name [inital vaues];
6   int X = 3;
7
8   void setup() {
9     // put your setup code here, to run once:
10
11  }
12
13  void loop() {
14    // put your main code here, to run repeatedly:
15    // Call your function here
16   readme();
17  }
18
19  void readme()
20  {
21    // put your function here
22  }
```

setup()

Runs only ones

loop()

Runs forever

Download Codes

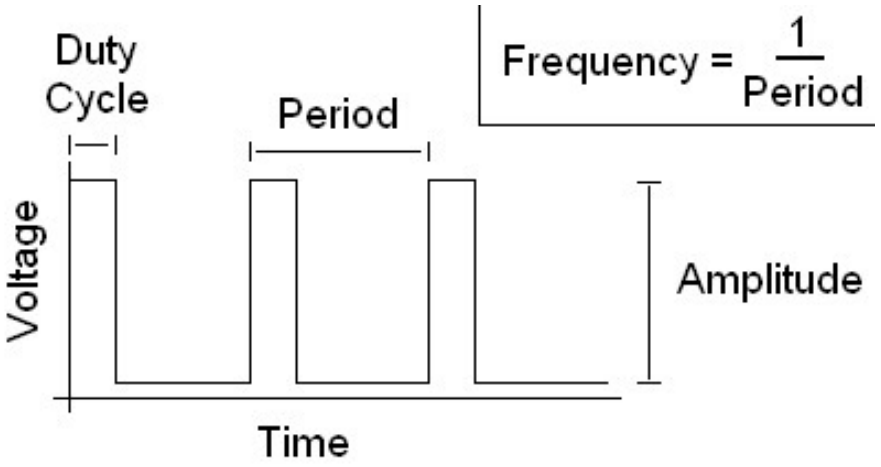
Another Blink Example

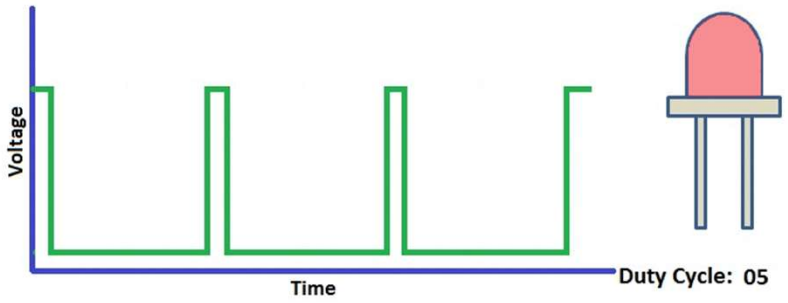
BuiltIn_LED.ino

```
1  #define LED_BUILTIN 2
2
3  // the setup function runs once when you press reset or power the board
4  void setup() {
5    // initialize digital pin LED_BUILTIN as an output.
6    pinMode(LED_BUILTIN, OUTPUT);
7  }
8
9  // the loop function runs over and over again forever
10 void loop() {
11   digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
12   delay(1000);                       // wait for a second
13   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
14   delay(1000);                       // wait for a second
15 }
```

Pulse Width Modulation (PWM)

- PWM signal properties.
  - Signal's frequency;
  - Duty cycle;
  - PWM channel;
  - GPIO where you want to output the signal.





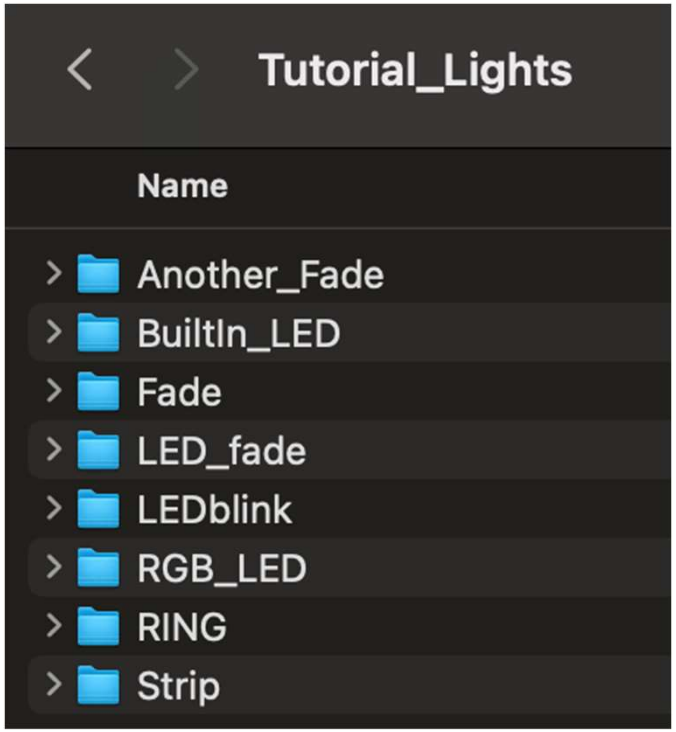
Built-in Fade

Fade.ino

```
15
16  int led = 2;           // the PWM pin the LED is attached to
17  int brightness = 0;    // how bright the LED is
18  int fadeAmount = 5;    // how many points to fade the LED by
19
20  // the setup routine runs once when you press reset:
21  void setup() {
22    // declare pin 9 to be an output:
23    pinMode(led, OUTPUT);
24  }
25
26  // the loop routine runs over and over again forever:
27  void loop() {
28    // set the brightness of pin 9:
29    analogWrite(led, brightness);
30
31    // change the brightness for next time through the loop:
32    brightness = brightness + fadeAmount;
33
34    // reverse the direction of the fading at the ends of the fade:
35    if (brightness <= 0 || brightness >= 255) {
36      fadeAmount = -fadeAmount;
37    }
38    // wait for 30 milliseconds to see the dimming effect
39    delay(30);
40  }
41
```

Run other examples on fade

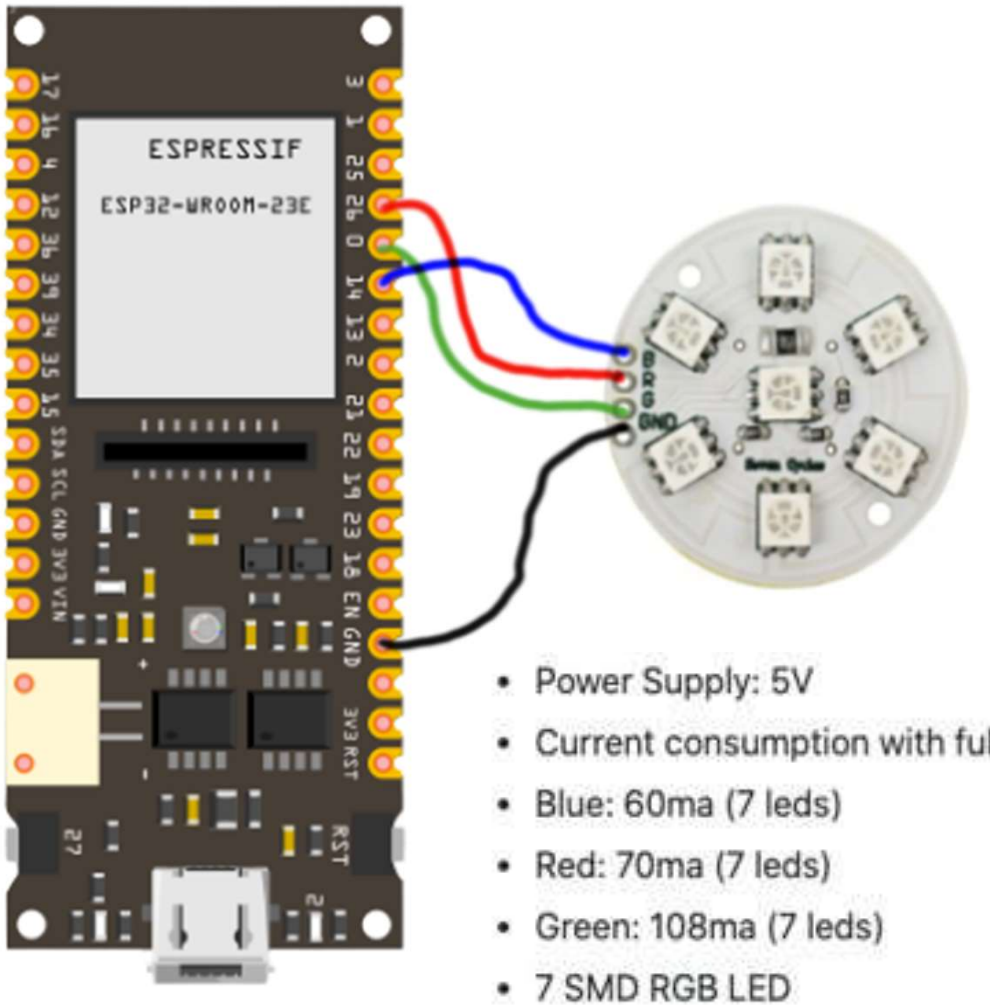
- In the unzipped folder -> go to Tutorial_Lights ->



- Run examples LED_Fade, Another_Fade

Working with a RGB LED disc

- Load the example - RGB_LED.ino
- Connect the RGB Led to the FB2 using four jumper cables provided.
- Understand the code and run the code.
- Modify the code to generate your own colors and color sequences.



- Power Supply: 5V
- Current consumption with full 5V:
- Blue: 60ma (7 leds)
- Red: 70ma (7 leds)
- Green: 108ma (7 leds)
- 7 SMD RGB LED
- 6000 mcd

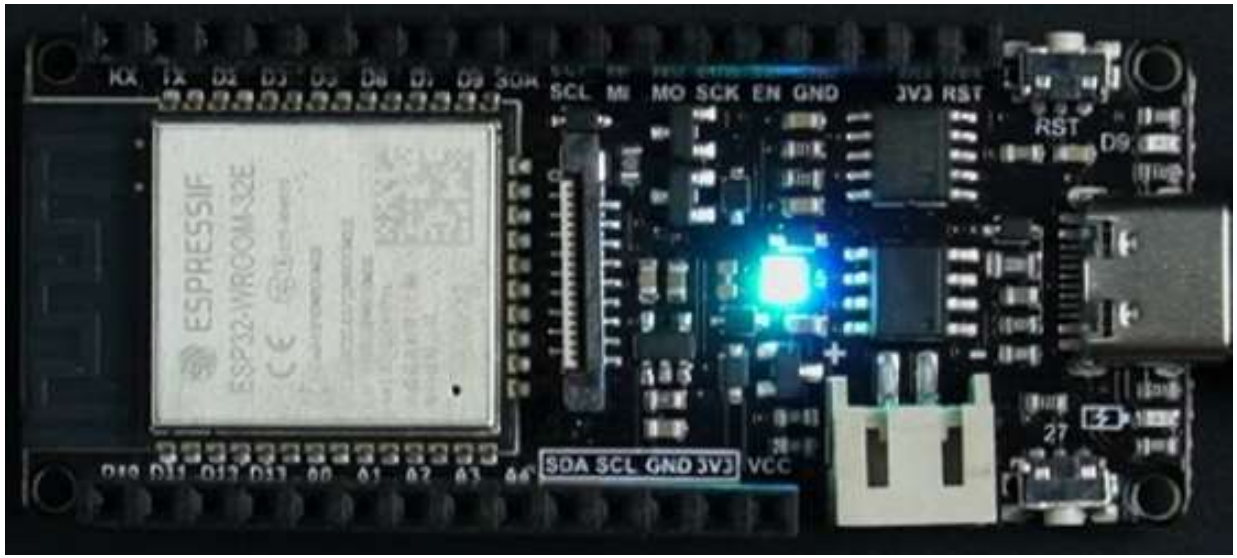
RGB LED Code (version2)

```
int B = 26; //Connect Blue led to Digital pin 3
int R = 0;//Connect Red led to Digital pin 5
int G = 14;//Connect Green led to Digital pin 6

//Connect the 5V pin of light disc to GND Pin of Arduino

void setup()
{
  pinMode(3,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
}

void loop()
{
  analogWrite(B,random(255));
  analogWrite(R,random(255));
  analogWrite(G,random(255));
  delay(80);
}
```

Working with a single Neopixel

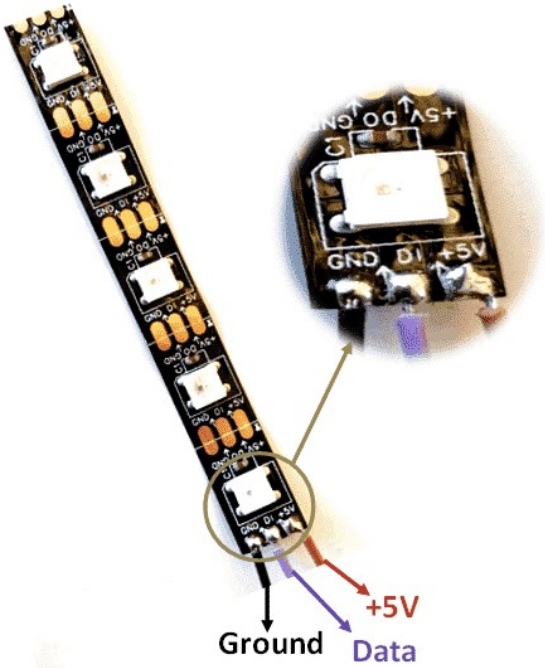
- A 5050 LED - the same type used by NeoPixel - is available on the FireBeetle 2 board.
- What is a neopixel?
  - Addressable RGB-LED
  - We have one connected to D8 (GPIO)
- Load the sketch Neopixel.ino, compile and run the code.
- Error ? – install FastLED Arduino library
- Modify the code to generate your own colors and color sequences.

No wiring needed!



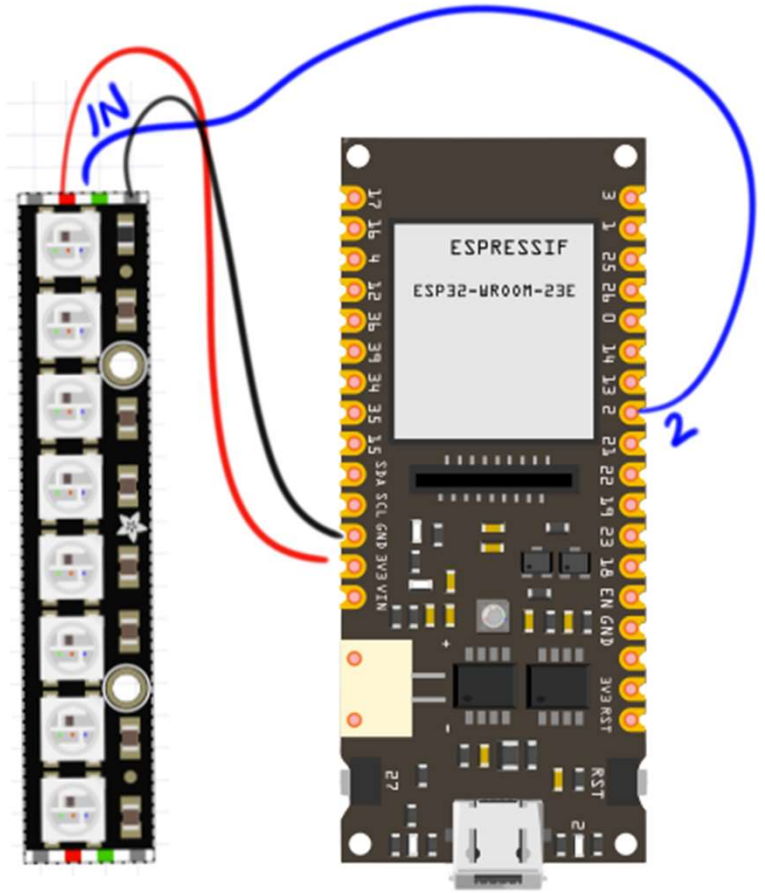
Neo Pixel Strip

- FireBeetle ESP32-E IoT Microcontroller with Header
- Adafruit - NeoPixel Strip 13 inches 10 LED (OR)
  - Adafruit NeoPixel Stick



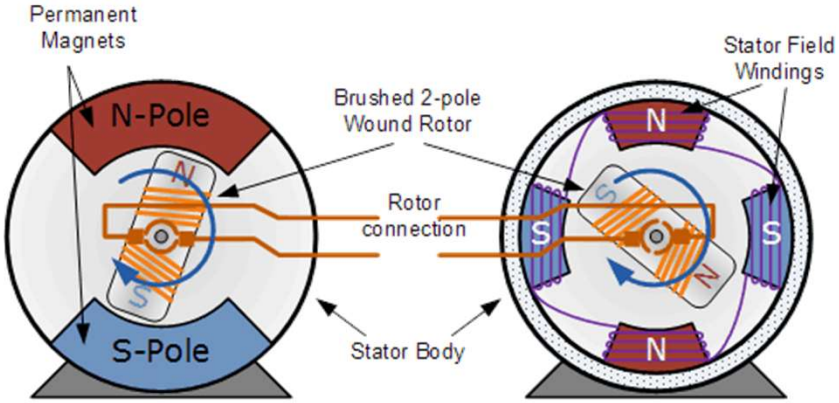
Install Arduino Neopixel Library

- Open the code in folder – Strip – open Strip.ino
- Open Library Manager under Tools -> Manage Libraries
- Search for an install “NeoPixel” Arduino library
- Modify the code to generate your own colors and color sequences.



Types of Motors

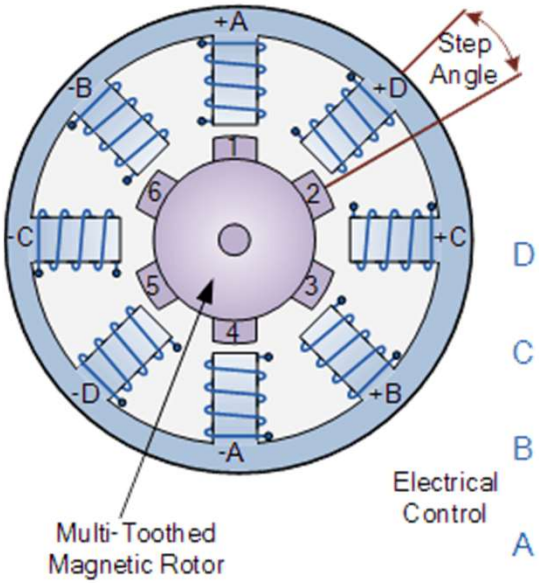
- DC Motors
- Stepper Motors
- Servo Motors

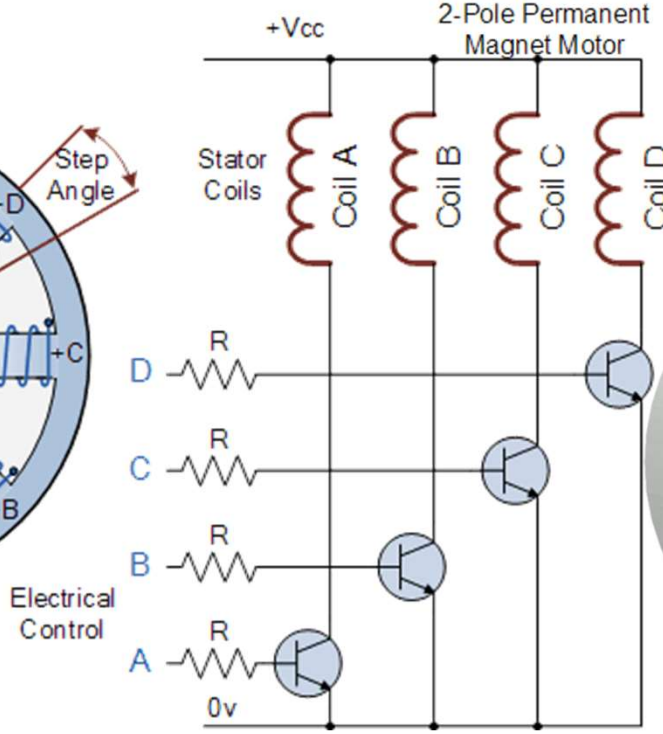


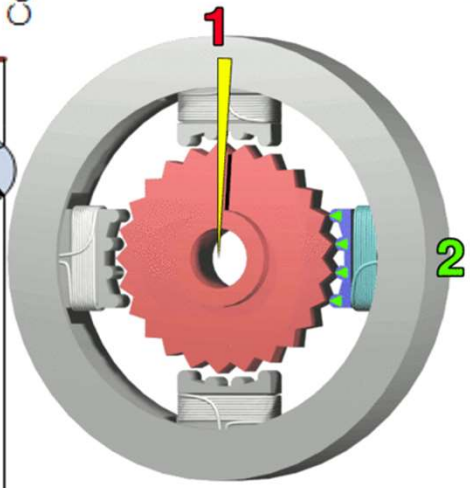
2-Pole Permanent Magnet Motor

4-Pole Wound Field Motor

DC motor





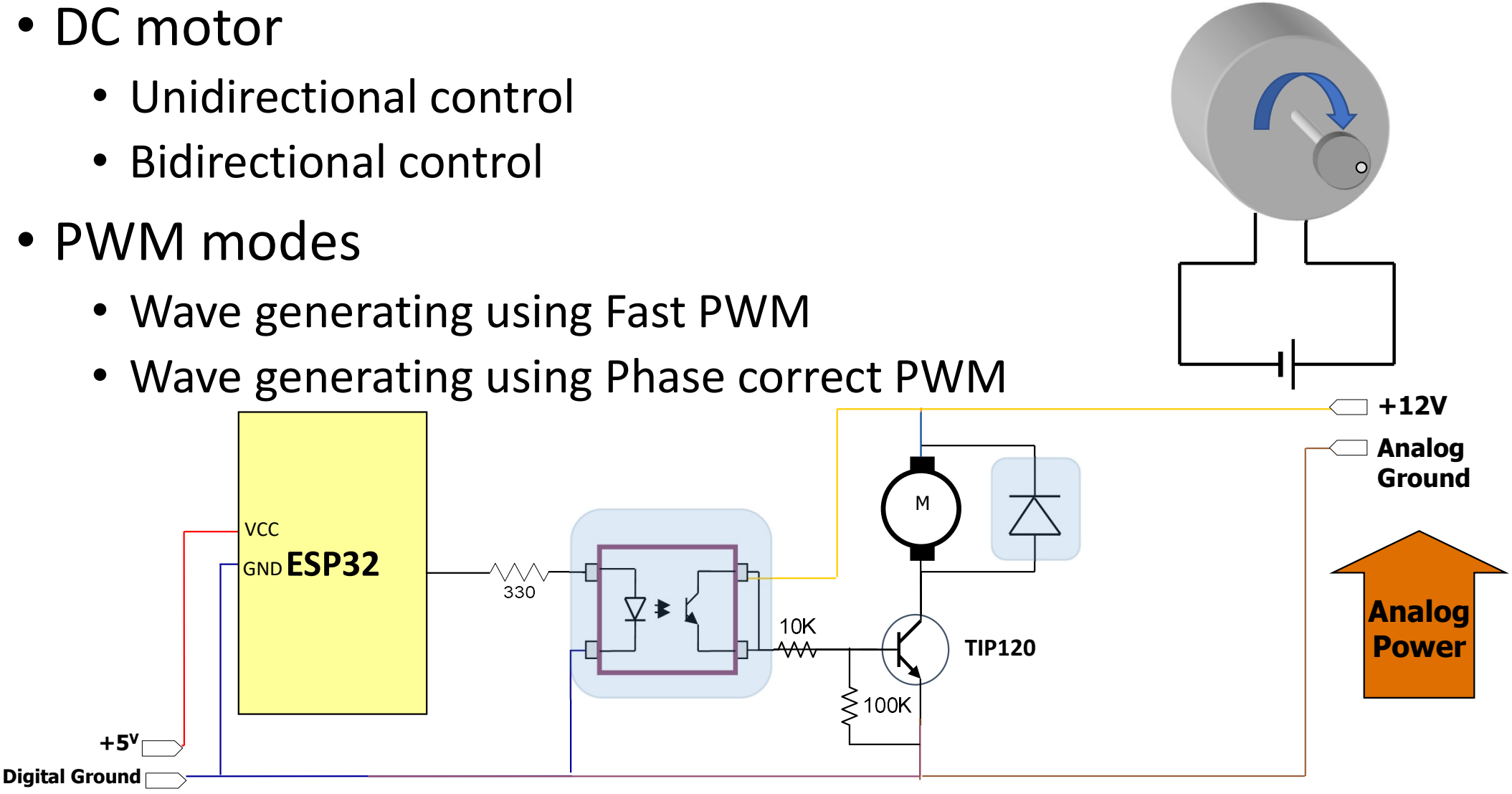


Stepper motor

Servo

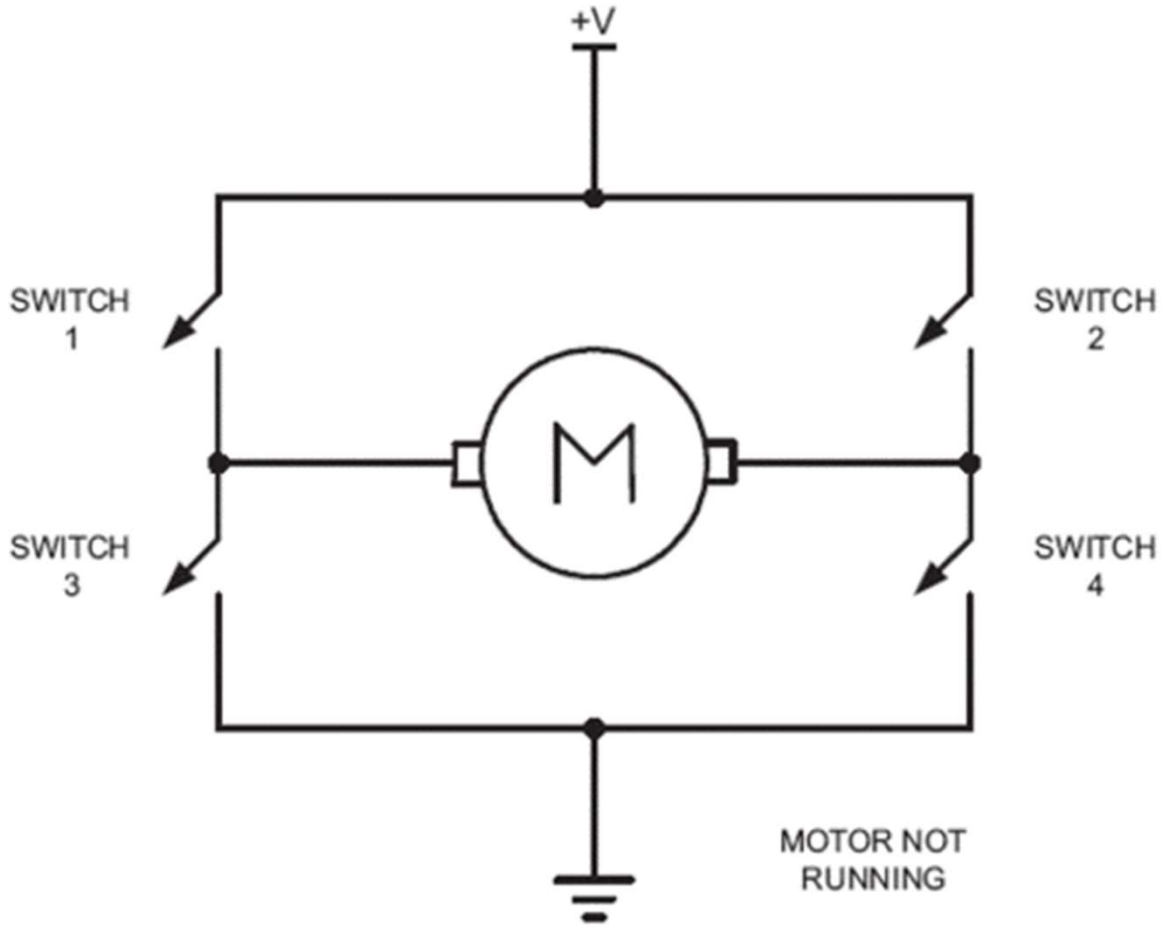
DC Motor Drive

- DC motor
  - Unidirectional control
  - Bidirectional control
- PWM modes
  - Wave generating using Fast PWM
  - Wave generating using Phase correct PWM

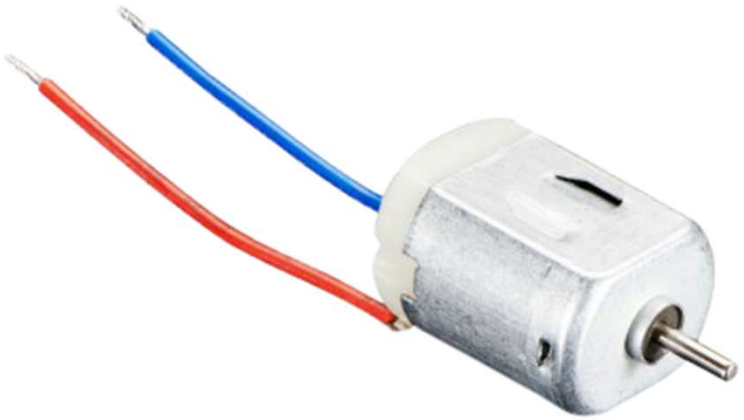


Bidirectional Control using H-Bridge

H-Bridge Configuration



DC Motor



- Operating Temperature: -10°C ~ +60°C
- Rated Voltage: 6.0VDC
- Rated Load: 10 g*cm
- No-load Current: 70 mA max
- No-load Speed: 9100 ±1800 rpm
- Loaded Current: 250 mA max
- Loaded Speed: 4500 ±1500 rpm
- Starting Torque: 20 g*cm
- Starting Voltage: 2.0
- Stall Current: 500mA max
- Body Size: 27.5mm x 20mm x 15mm
- Shaft Size: 8mm x 2mm diameter
- Weight: 17.5 grams

DRV8833 motor driver

ULT PIN:Low level is sleep mode (need to disconnect the J2 short solder joint behind the module)

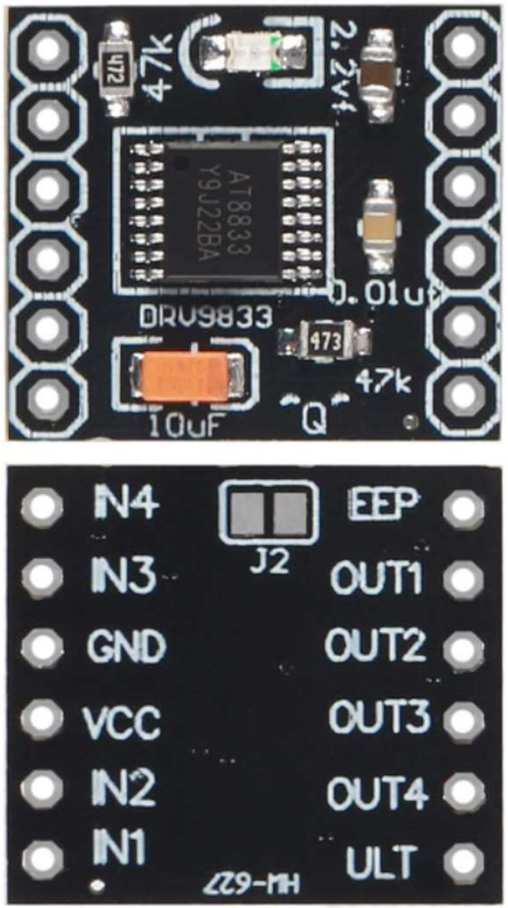
OUT1,OUT2:1-channel H-bridge controlled by IN1/I

OUT3,OUT4:2-channel H-bridge controlled by IN3/I

EEP PIN:Output protection.Default no need to conn

VCC:3-10V

GND is grounded



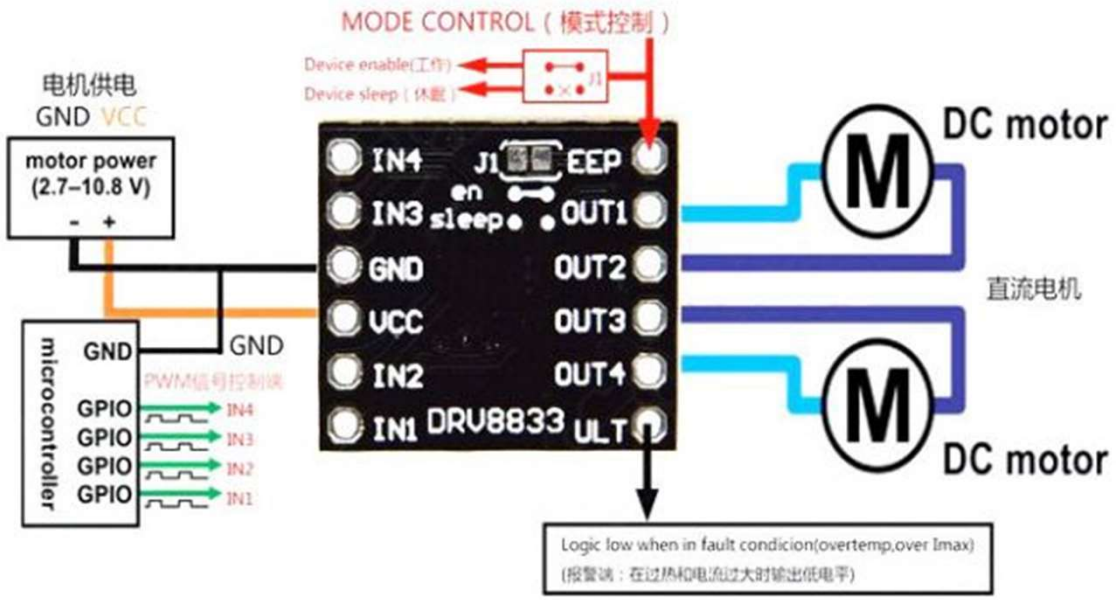
Working Principle

Connection shows below.

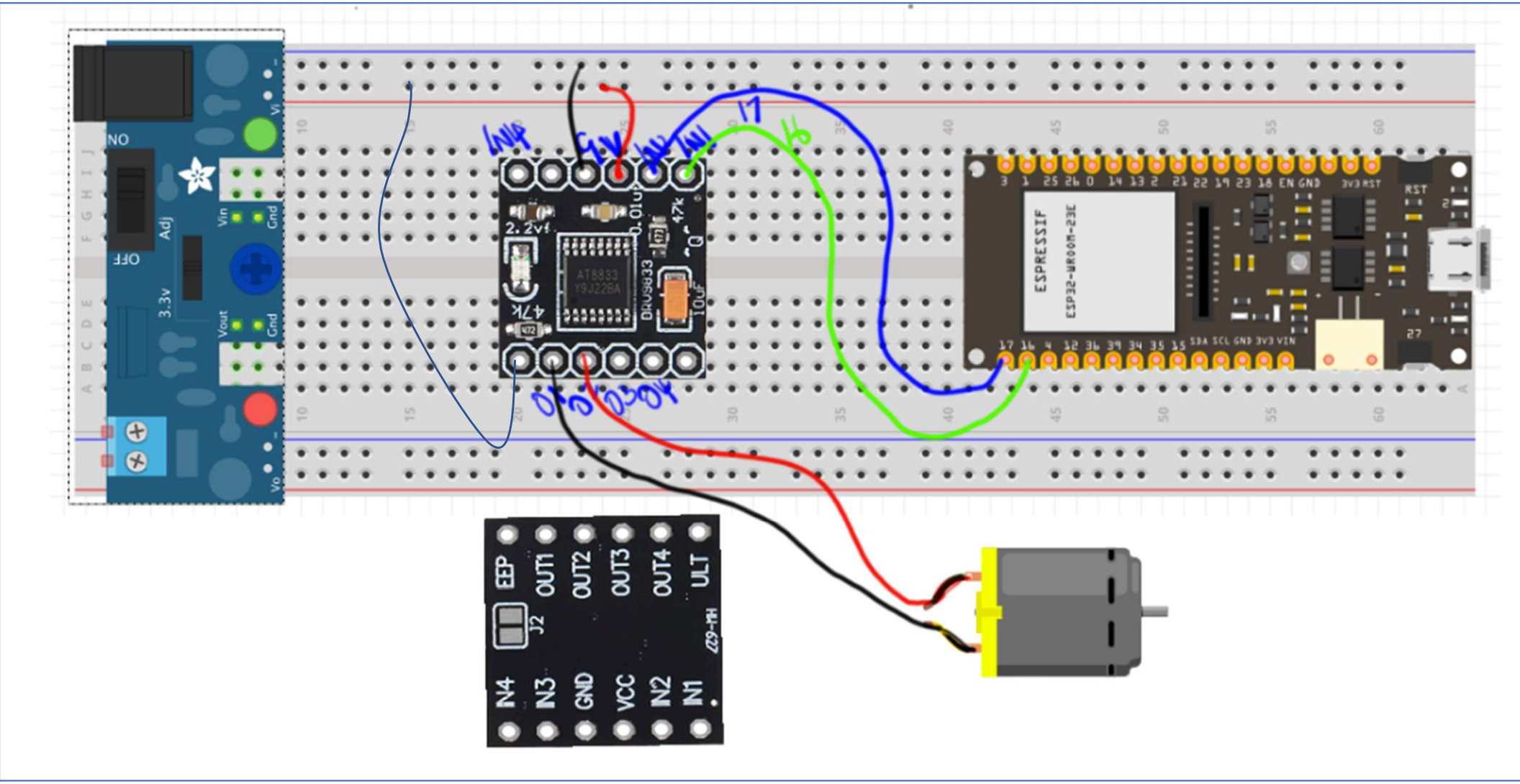
Using two PWM pins in the board and driver as bridge, you can control dc motor direction or speed. More detail will be in the code.

note: Each motor need two PWM pins.

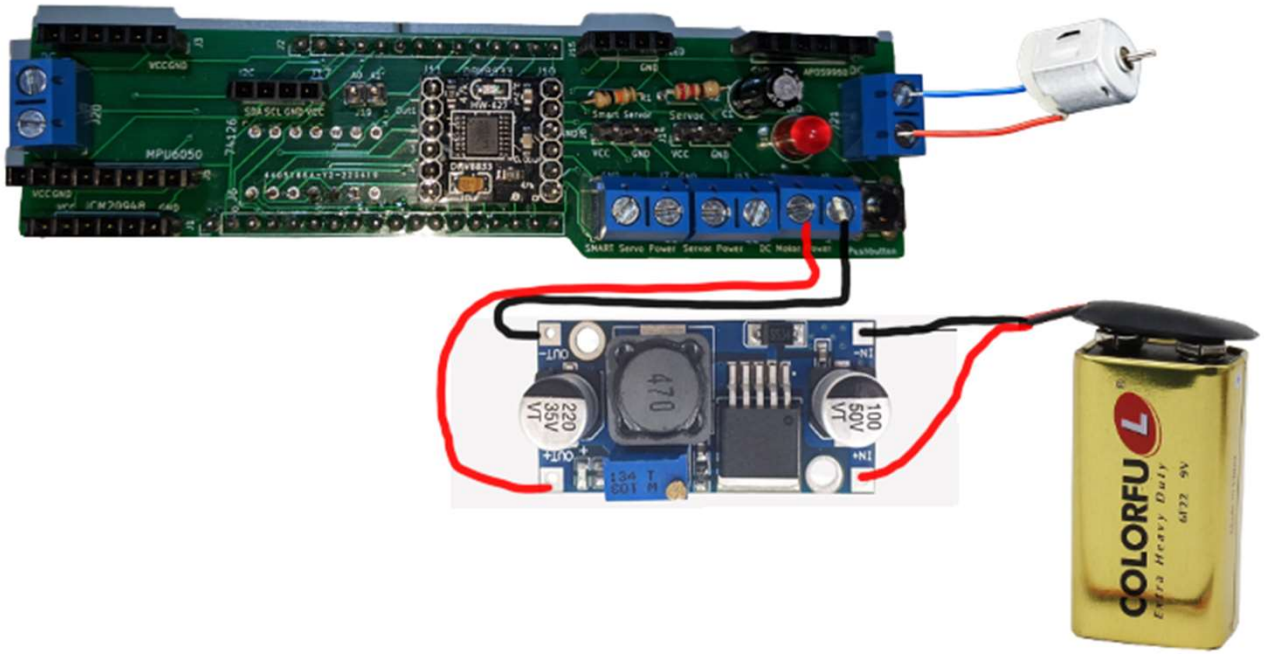
note: B



Connections using BreadBoard



Connections using Addon Board



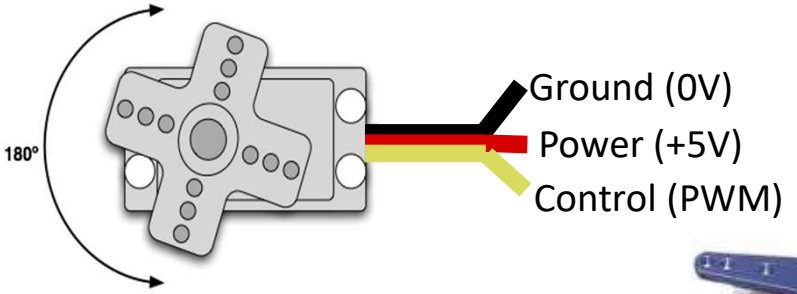
Running the DC Motor

- Get into the folder Tutorial_Motors
- Open the sketch
  - DC_MotorFullSpeed.ino
  - Explain the operation of the code

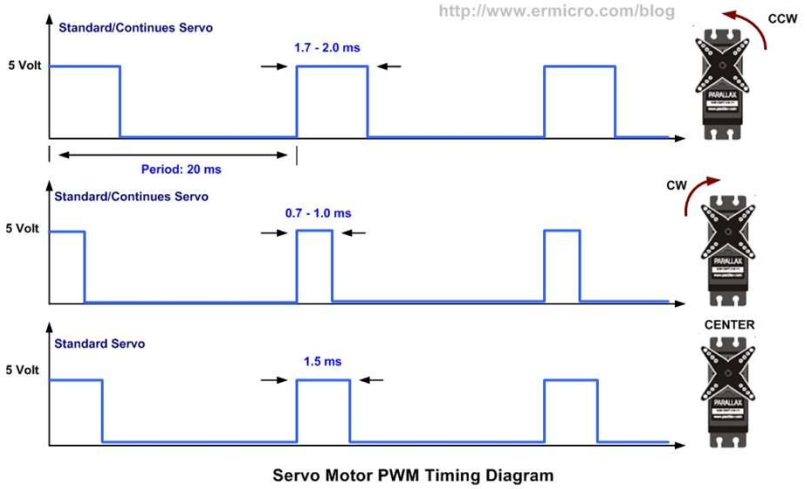
- Open the sketch
  - DC_MotorCntrl_AnalogWrite.ino
  - Explain the operation of the code
  - Modify the code to gradually increase and decrease the speed of the motor.

Servo Motors

- Can be positioned from 0-180º (usually)*
- Internal feedback circuitry & gearing takes care of the hard stuff
- Easy three-wire PWM 5V interface
- PWM freq is 50 Hz (i.e. every 20 millisecs)
- Pulse width ranges from 1 to 2 millisecs
  - 1 millisec = full anti-clockwise position
  - 2 millisec = full clockwise position

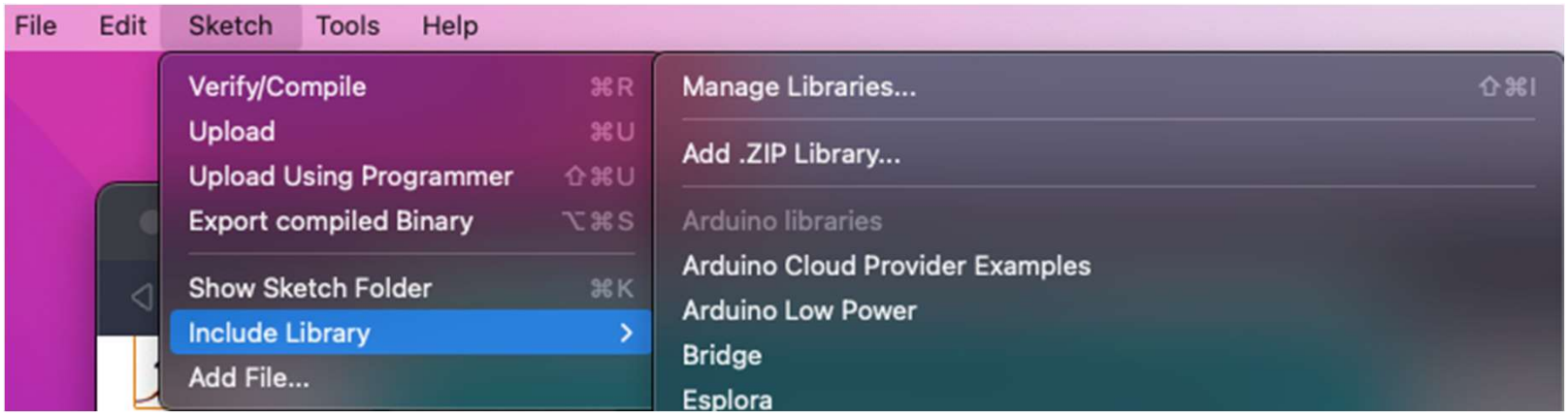


*You can also find 360º continuous servo, 270º servo, smart servo (chain)

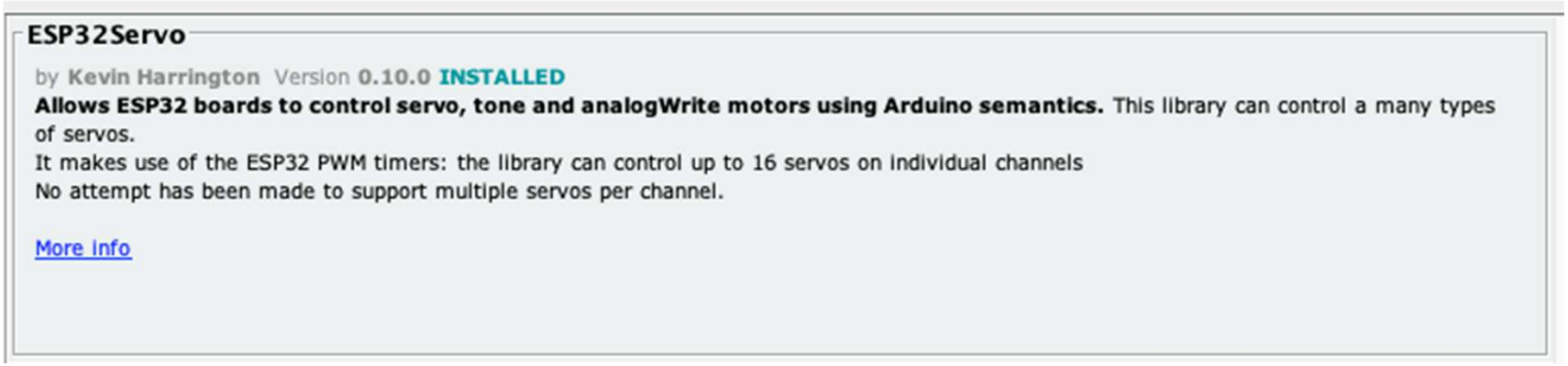


Servo Motor PWM Timing Diagram

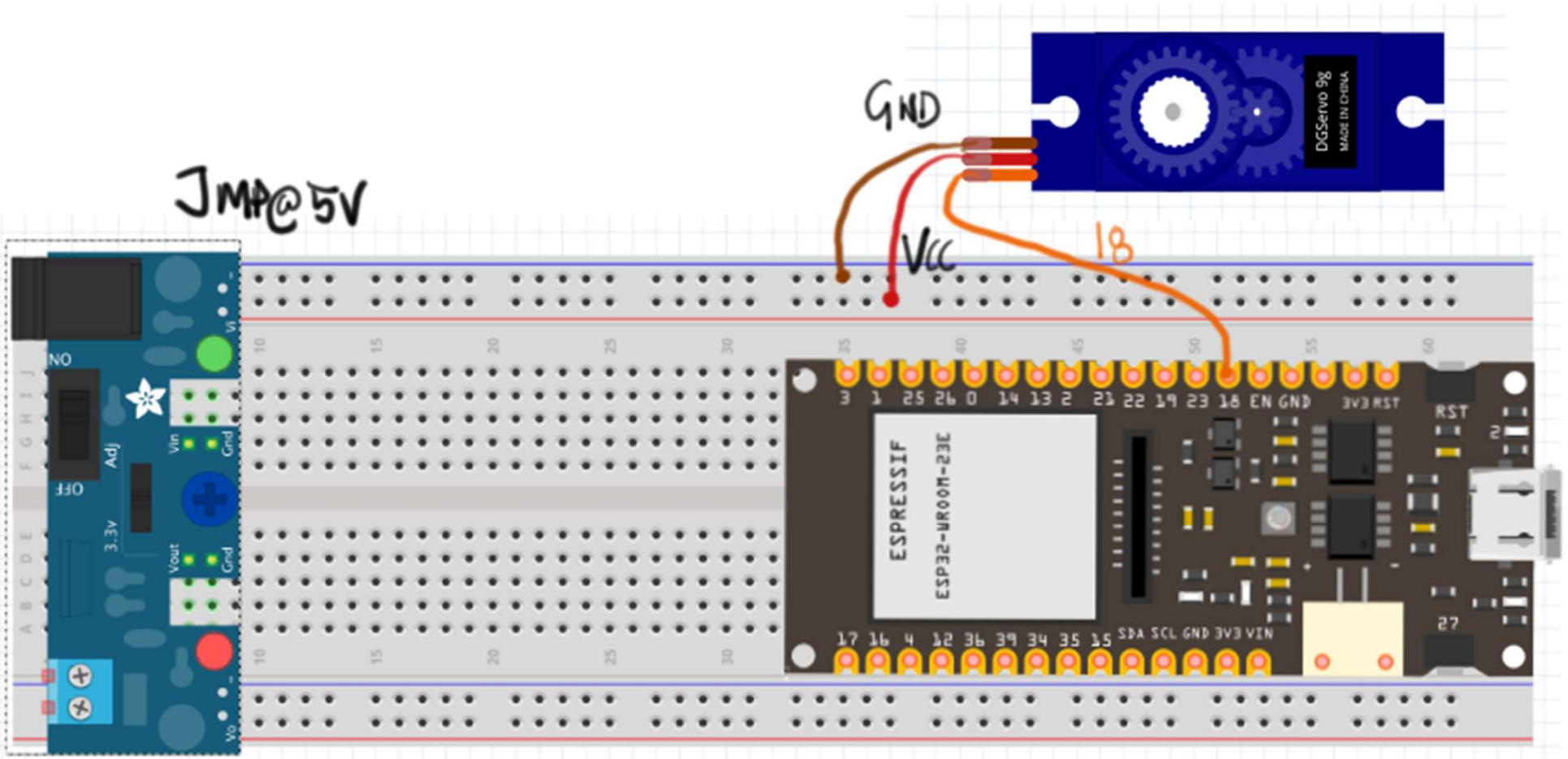
Install Library



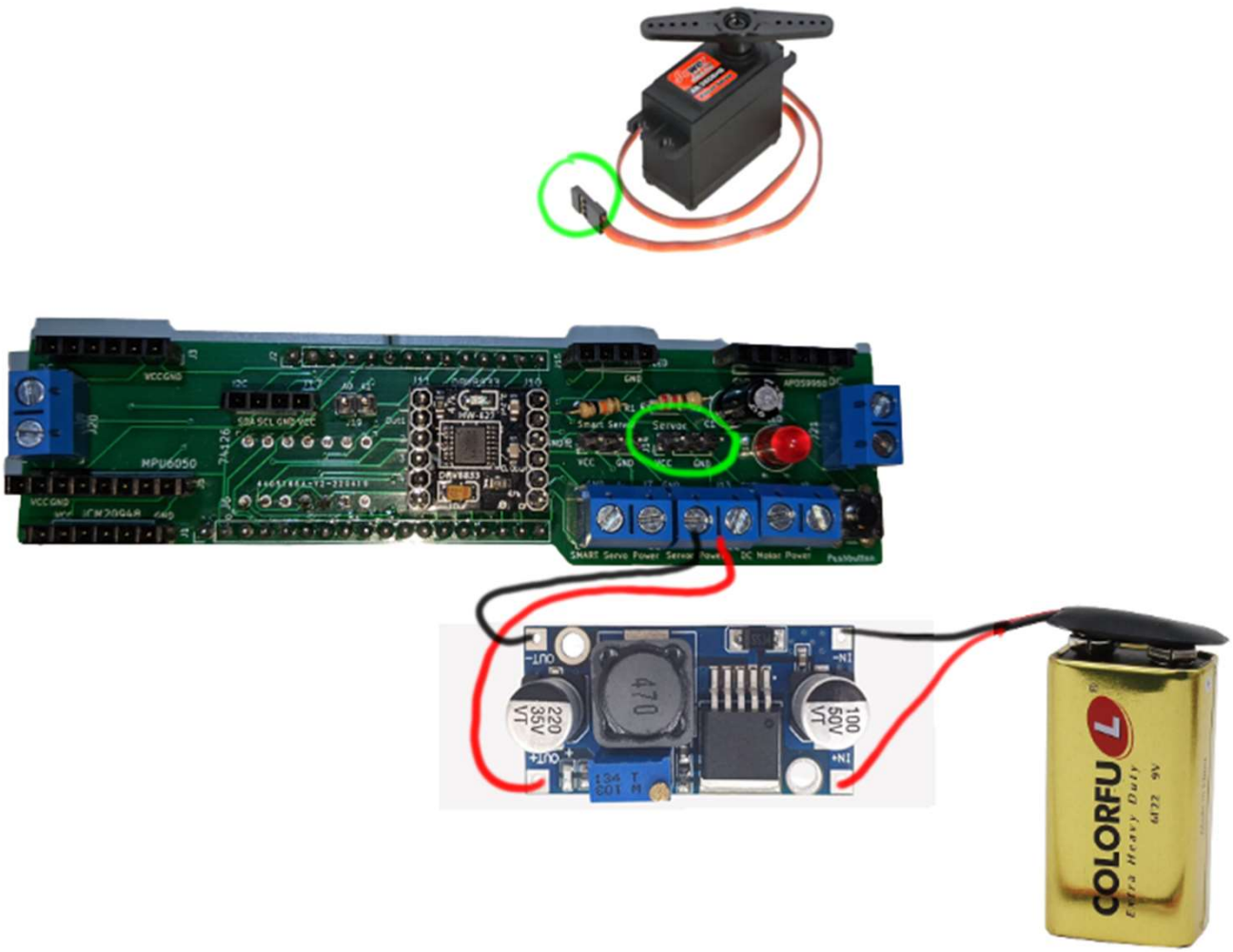
- Search and install the ESP32Servo library in the Library Manager



Connections using BreadBoard



Connections using Addon Board



Running the ServoMotor

- Get into the folder Tutorial_Motors
- Open the sketch
  - Servo_sweep.ino
  - Explain the operation of the code

- Open the sketch
  - Servo_sweep_button.ino
  - Explain the operation of the code

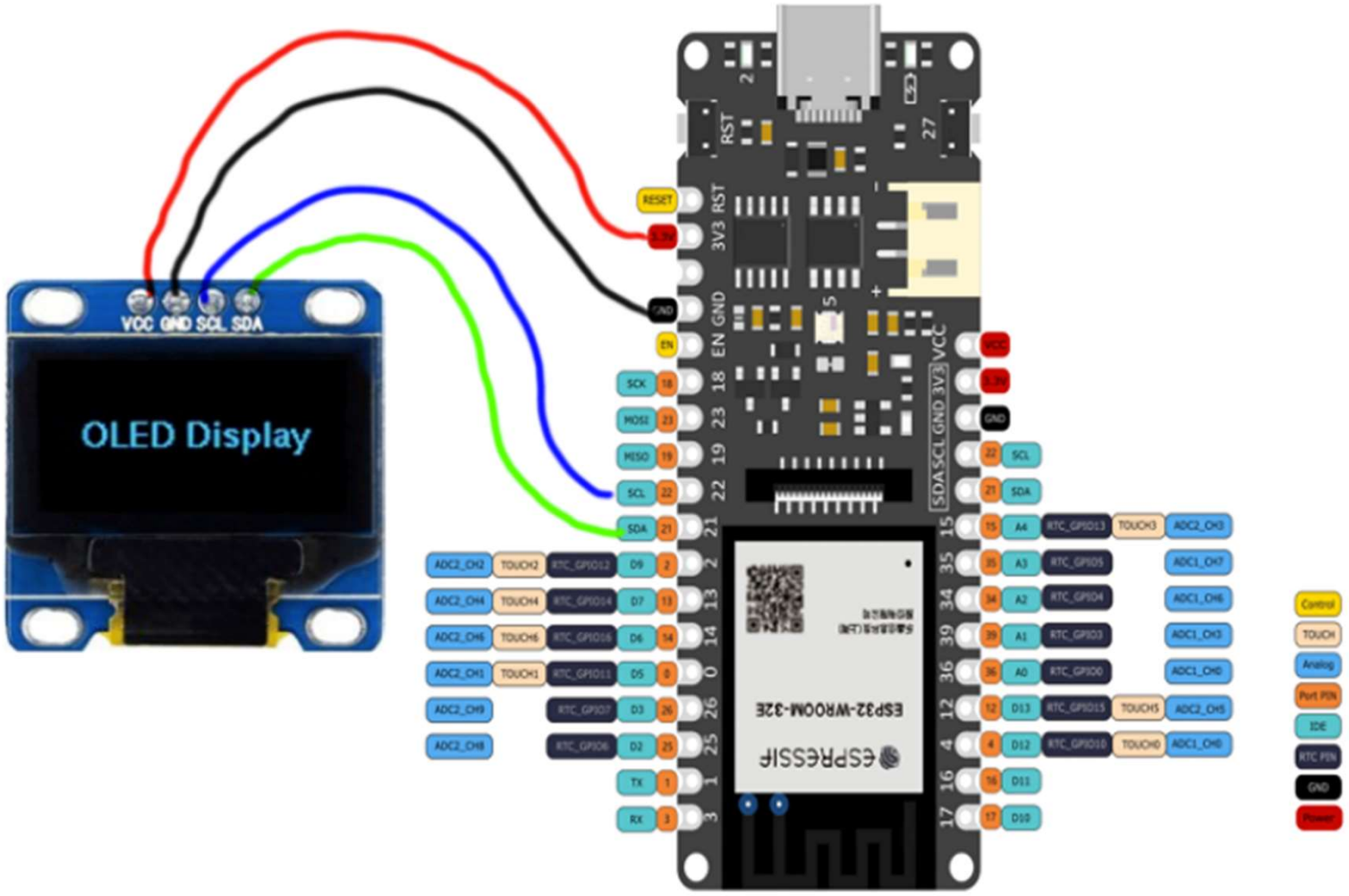
Displays

- Types of Displays
  - LCD
  - LED
  - OLED

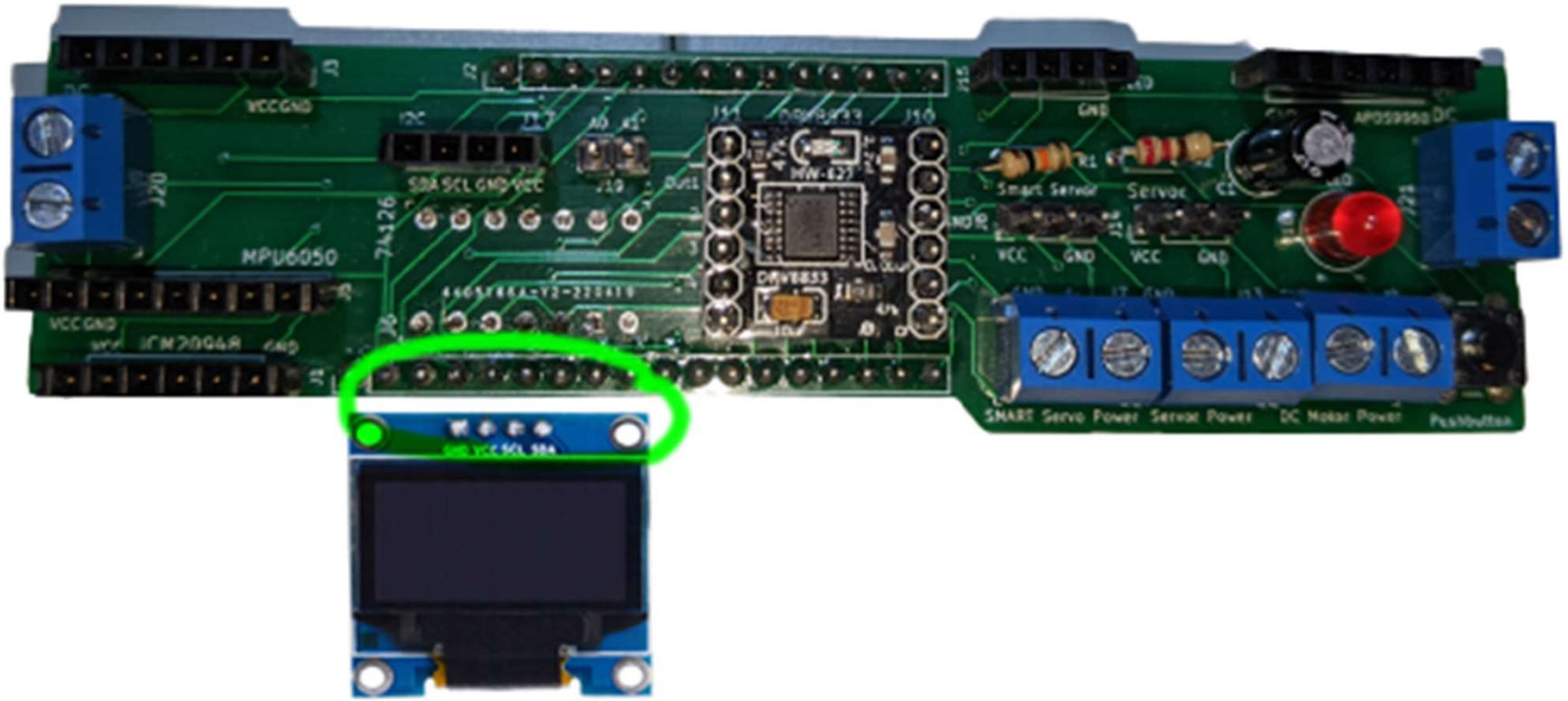
- Our Display
  - SSD1306 OLED LCD Display 128x64

- Communication
  - I2C Protocol
- Applications
- How to interface the display with ESP32?

Connections using BreadBoard



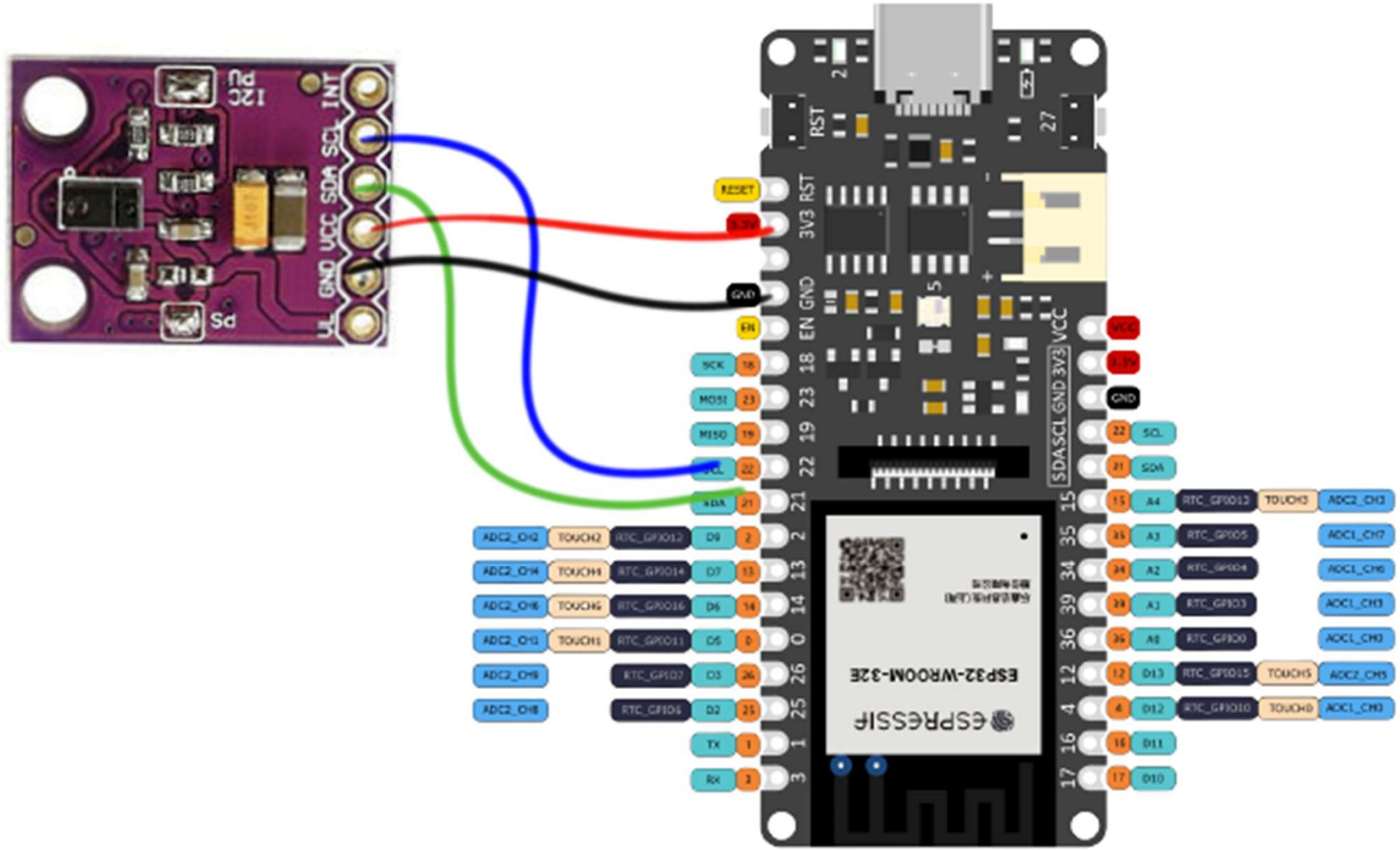
Connections using Addon Board



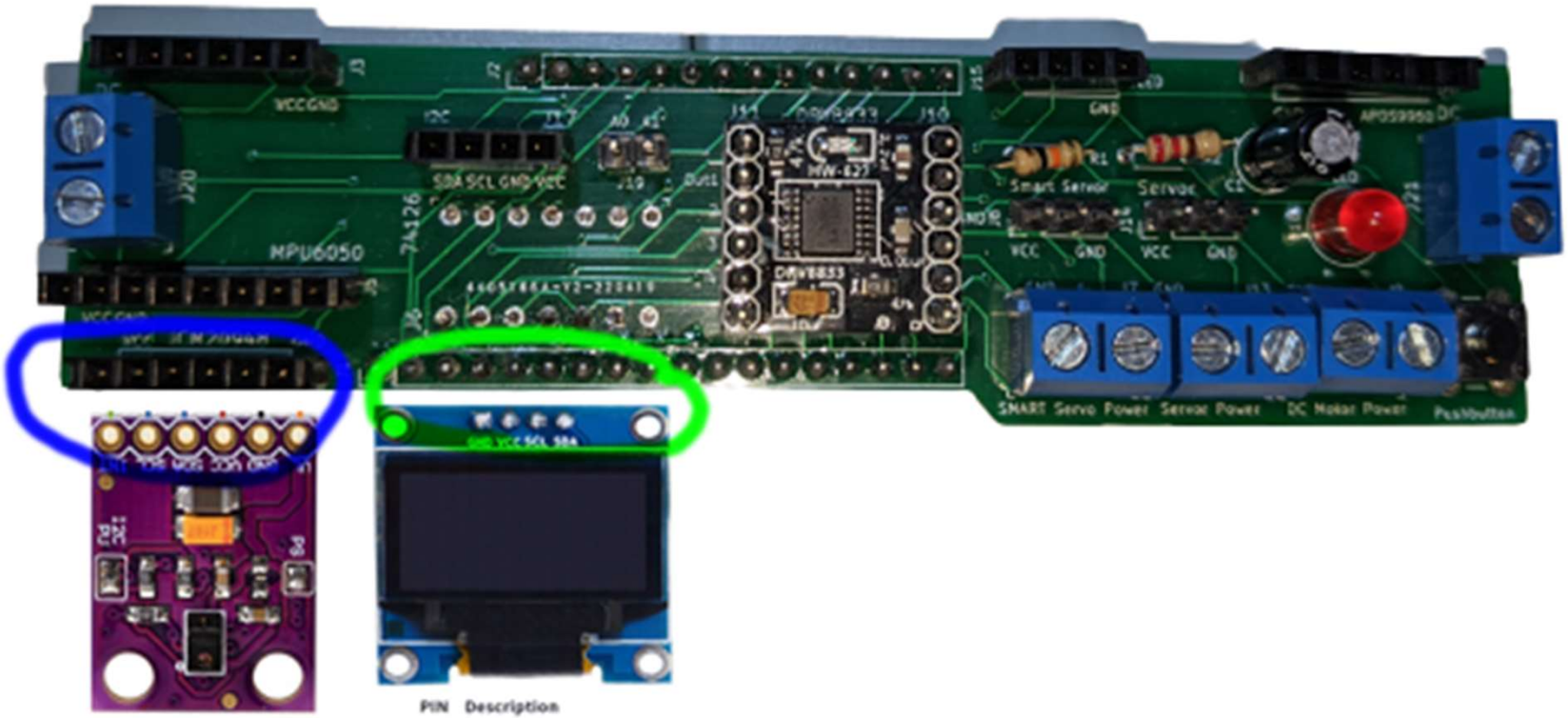
What is APDS9660?

- APDS-9960 is a digital RGB, ambient light, proximity and gesture sensor device
- providing red, green, blue, clear (RGBC), proximity and gesture sensing with IR LED.
- Applications
  - Display Backlight Control
  - Correlated Color Temperature Sensing
  - Cell Phone Touch-screen Disable
  - Digital Camera Touch-screen Disable
  - Mechanical Switch Replacement
  - Gesture Detection

Connections using BreadBoard

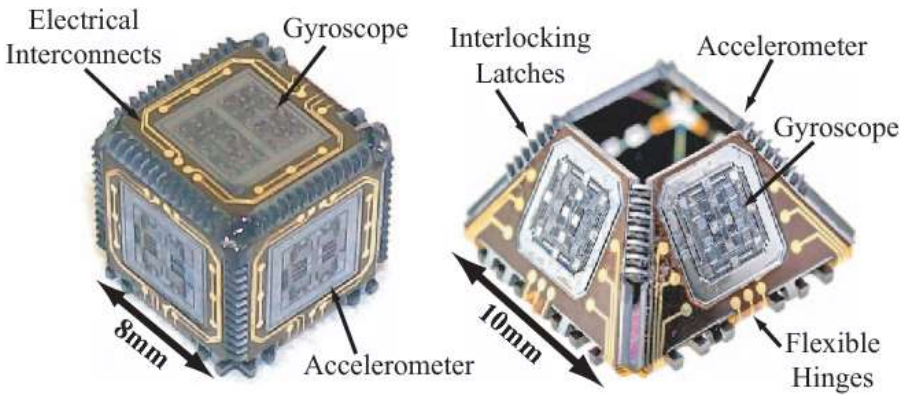


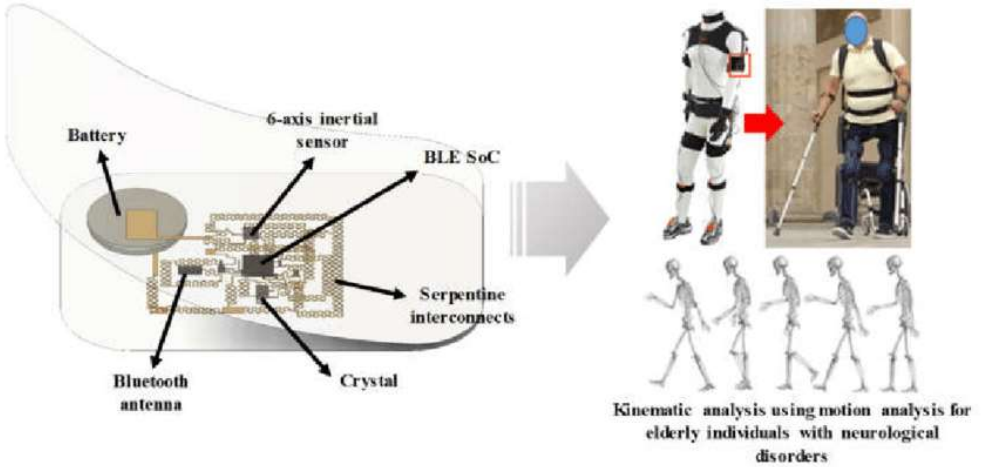
Connections using Addon Board



IMU

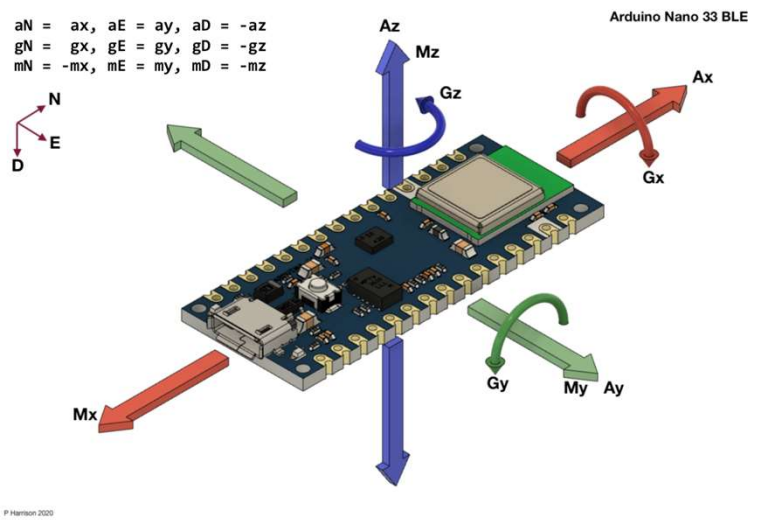
- What is Inertia Measuring Unit (IMU)?
  - What is inertia?
- Degrees of Freedom (DOF)
- Applications

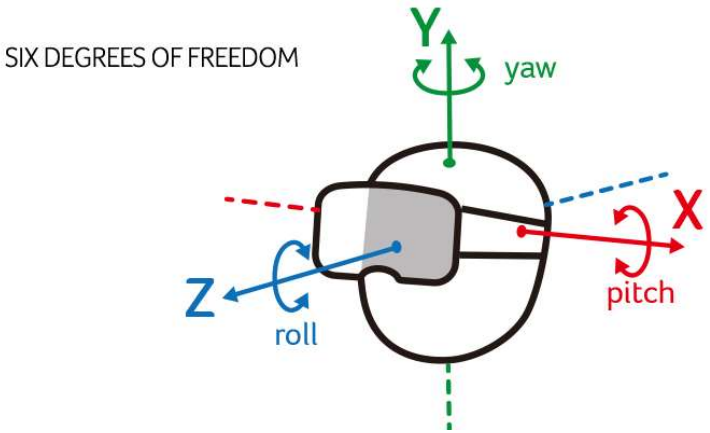




IMU Measures …

- Acceleration
  - Change of speed in all three axis
- Gyro
  - Change of angular momentum i
- Orientation
  - Yaw, Pitch, and Roll

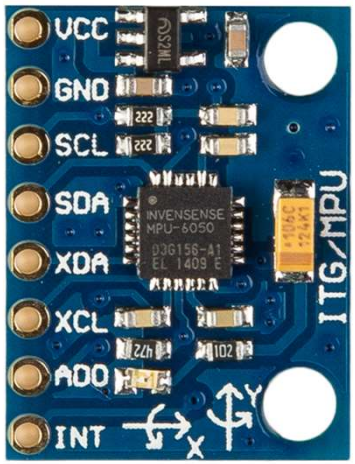




MPU6050

The MPU-6050 devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die, together with an onboard Digital Motion Processor™ (DMP™), which processes comple
MotionFusion algorithms.

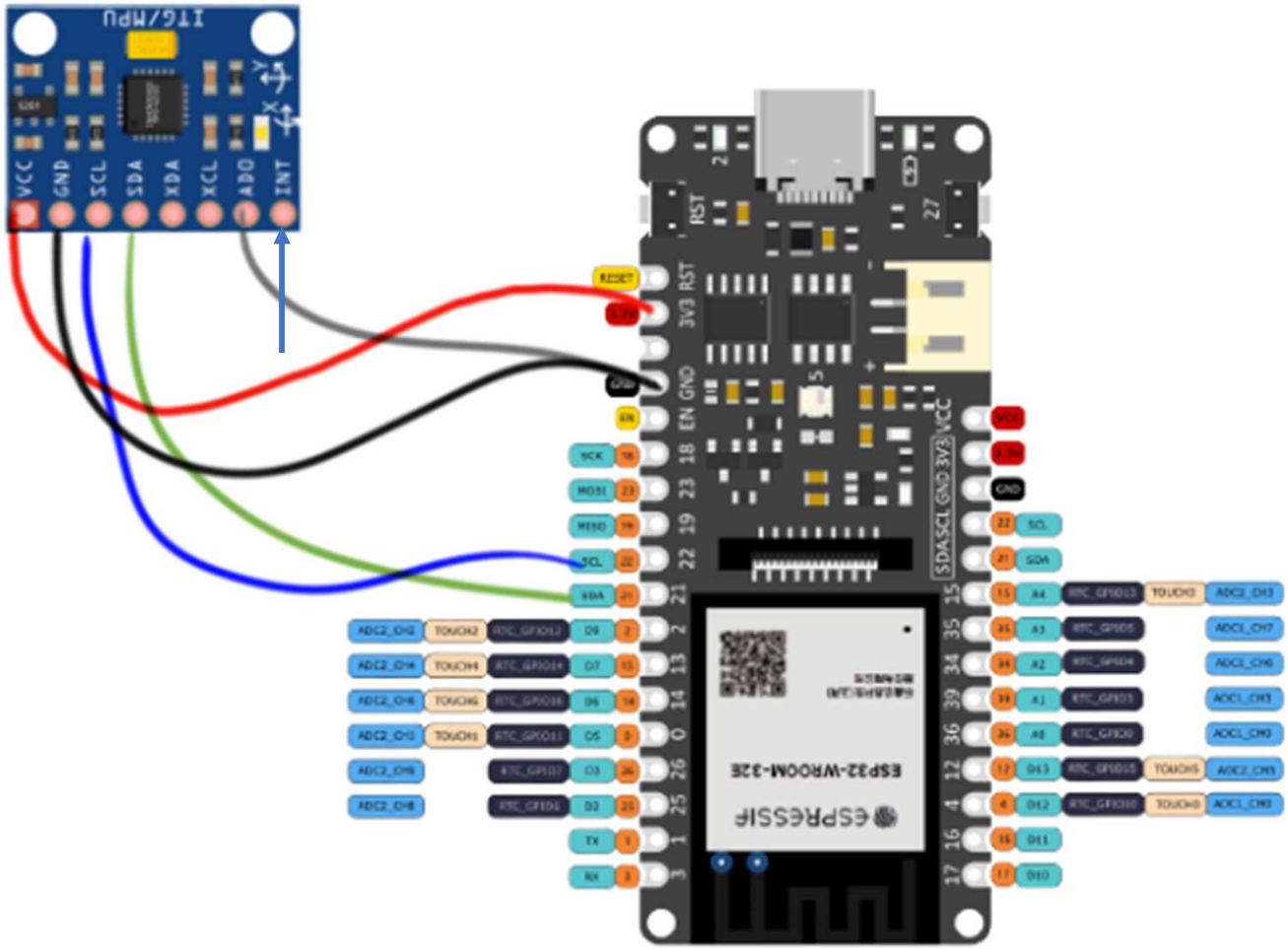
Uses I2C connection (SDA & SCL)



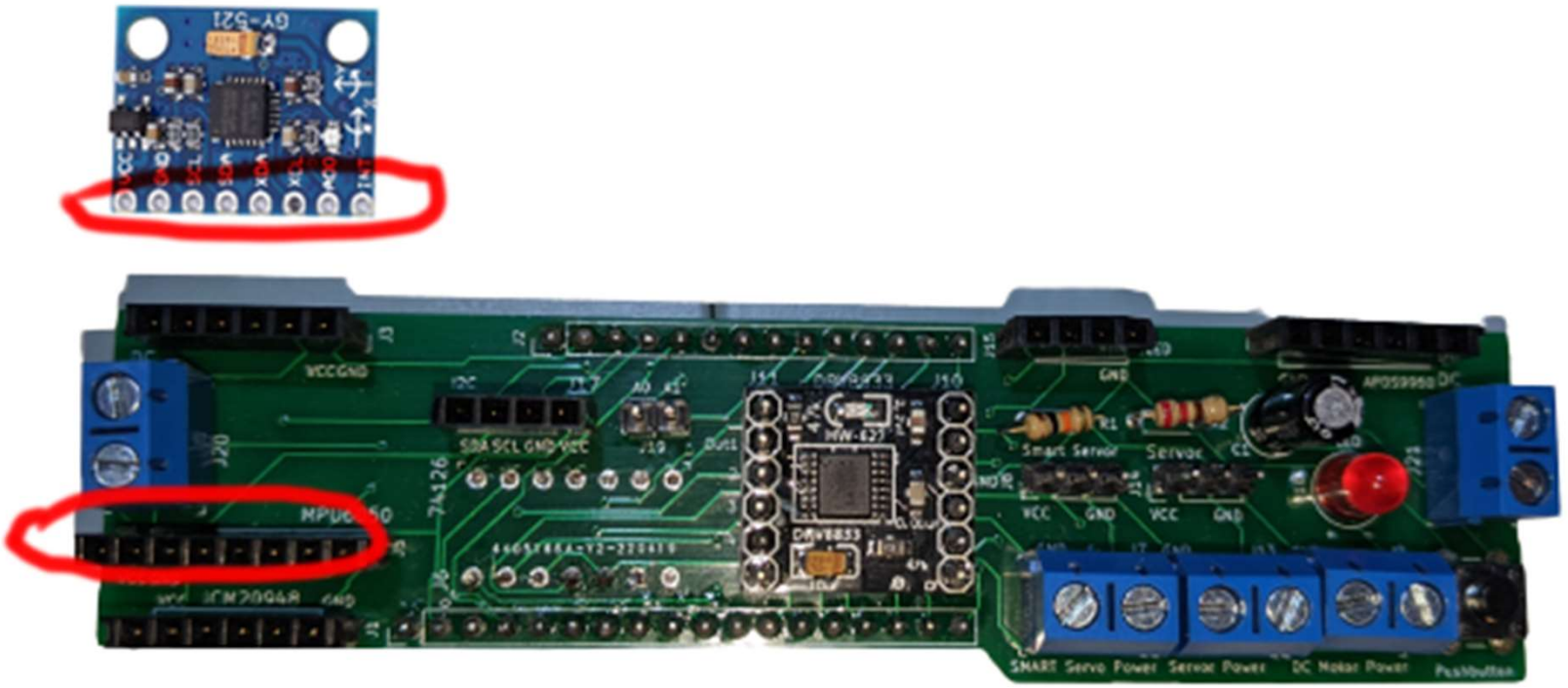
MPU6050

Connections using BreadBoard

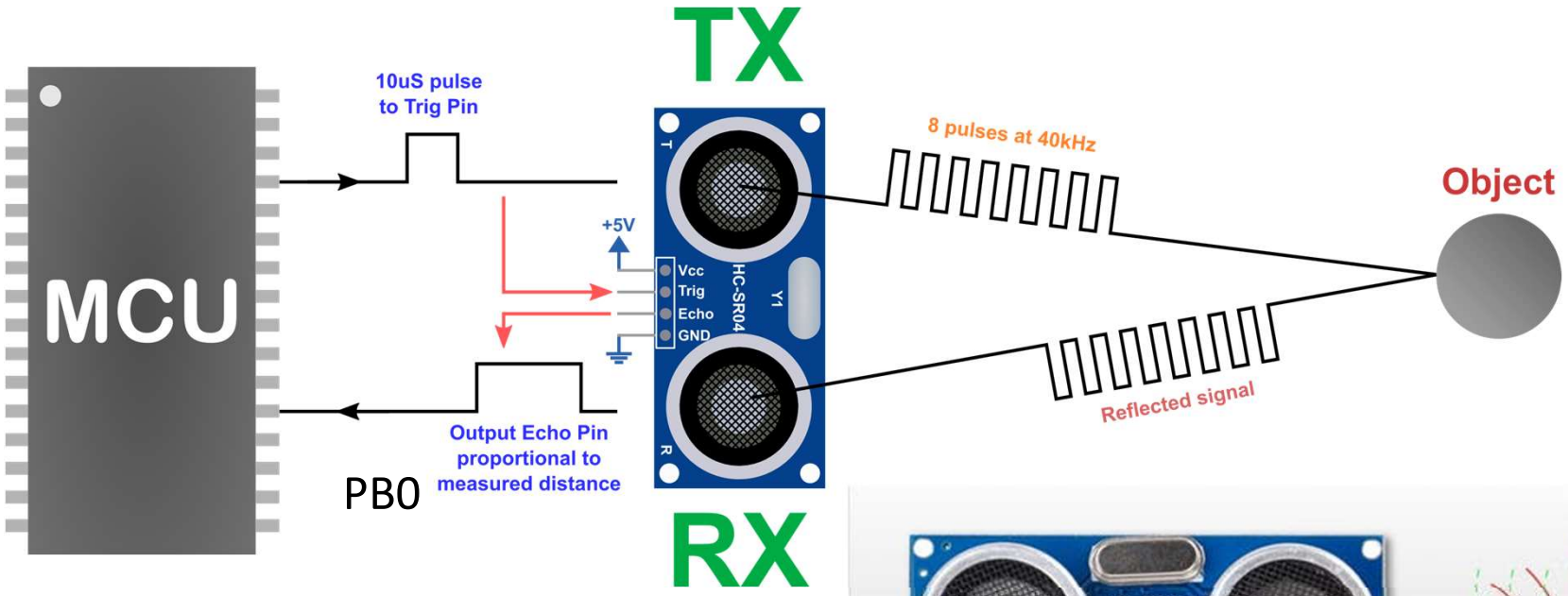
Uses I2C connection (SDA & SCL)

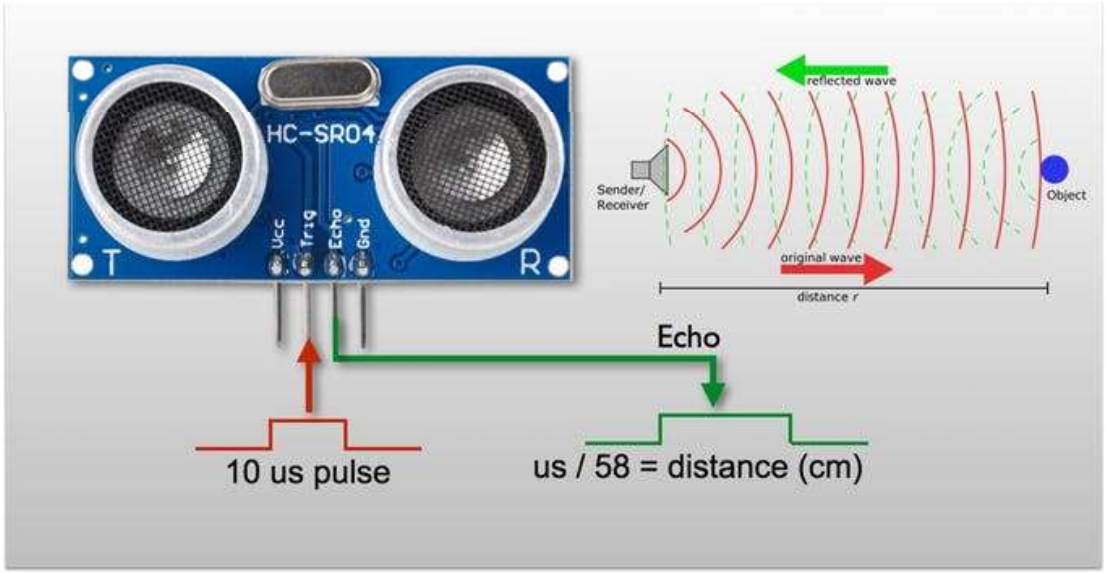


Connections using Addon Board



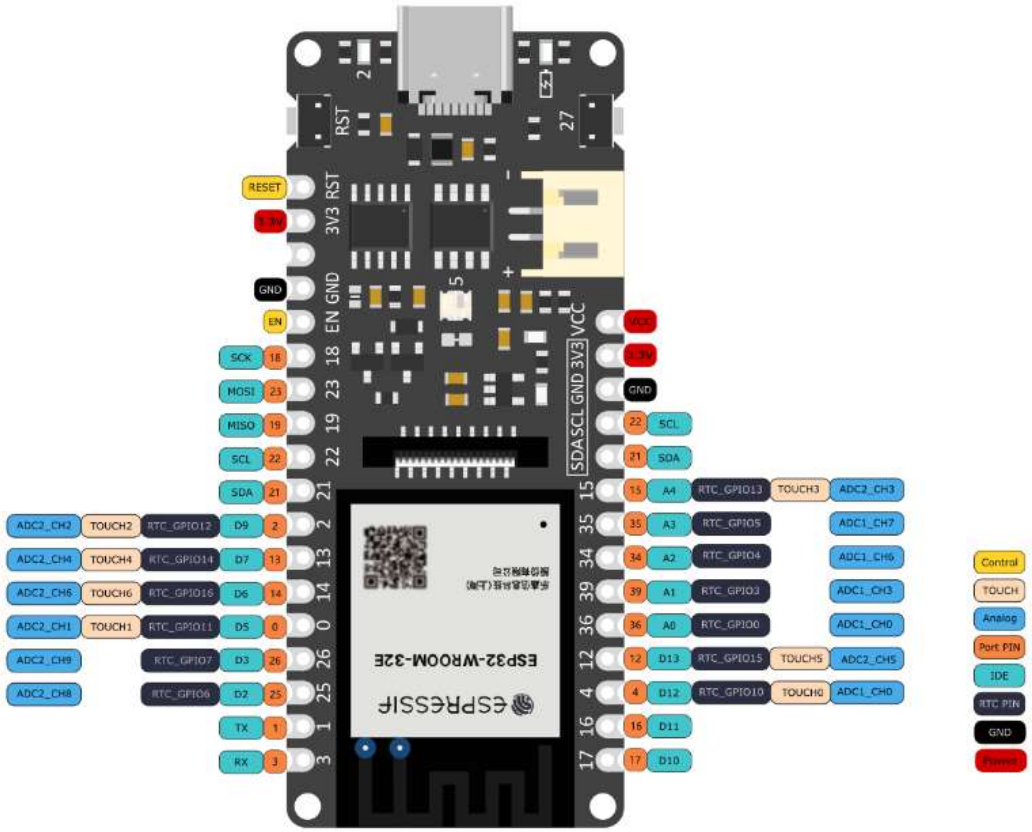
HC-SR04 Ultrasonic Sensor



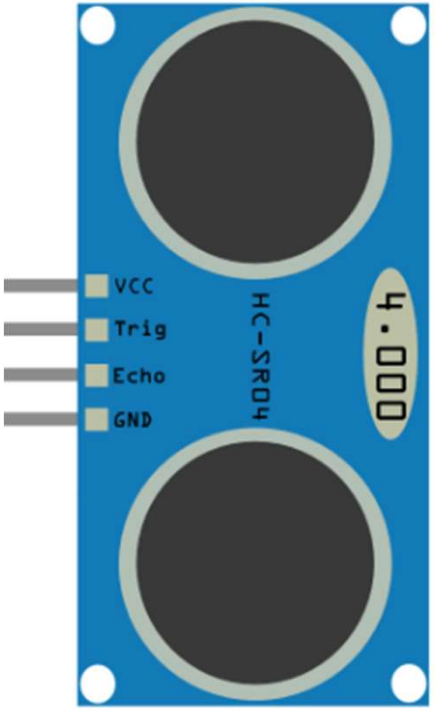


Range: 2cm – 400cm

US Sensor connection



VCC/3.3V
D5
D3
GND

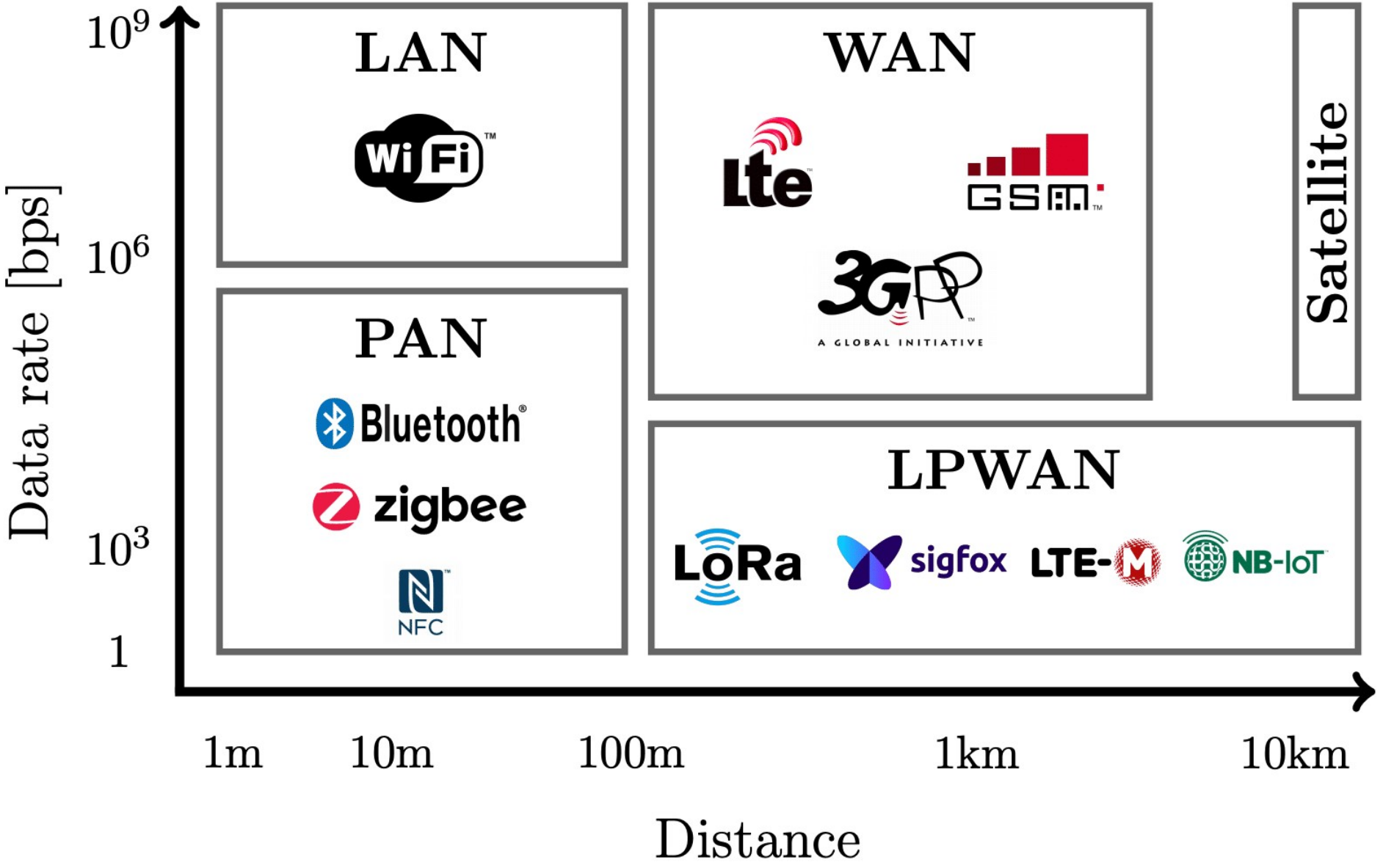


Executing the codes

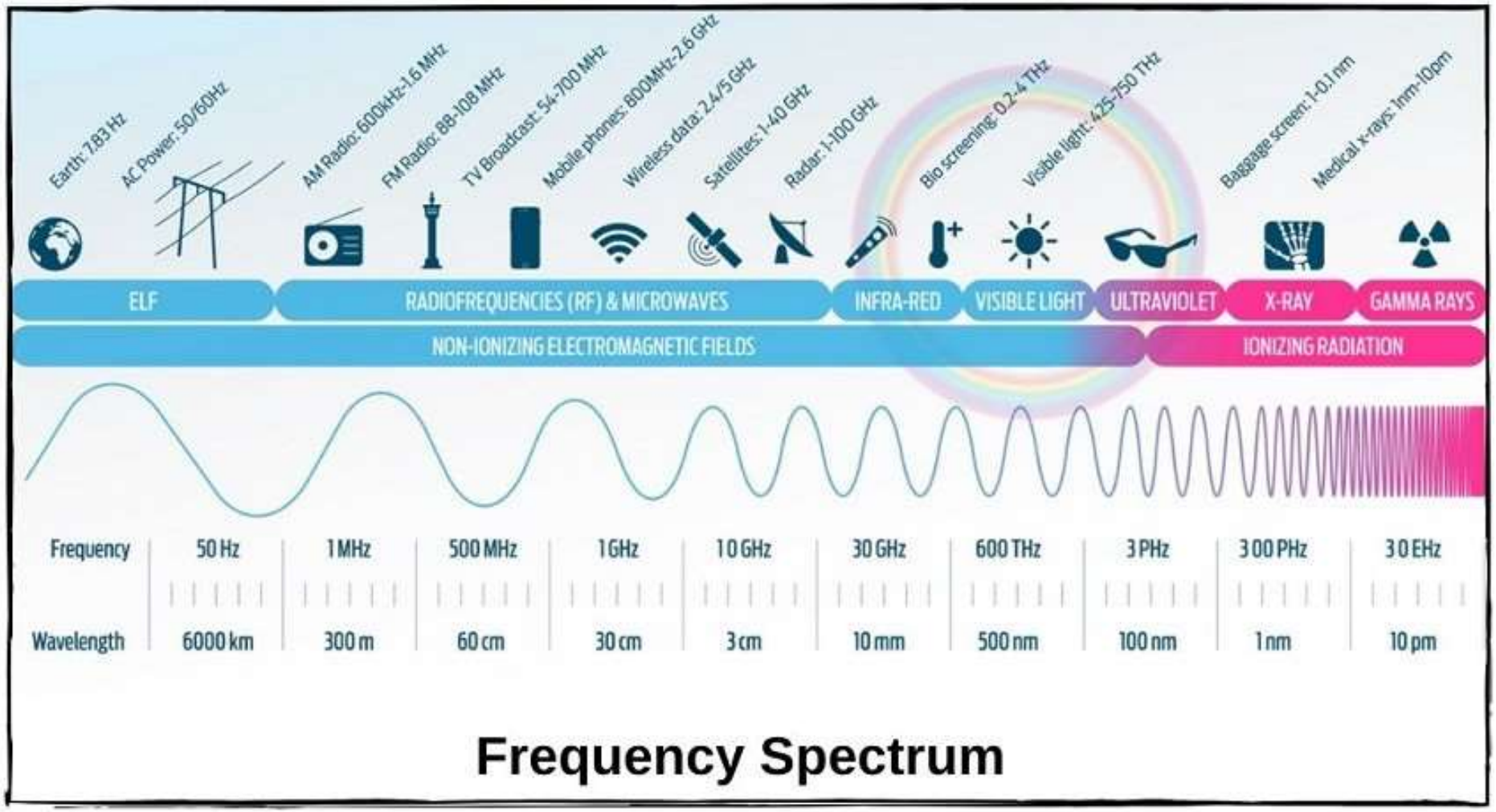
- Install Libraries:
  - Search and install Adafruit SSD1306 library.
  - Search and install Adafruit APDS9660 library.
  - Search and install Adafruit MPU6050

- Load and run the following program in sequence
  - OLED_Hworld
  - OLED_Animate
  - APDS9660_color_sensor
  - APDS9660_gesture_sensor
  - MPU6050_rawdata
  - MPU6050_CompFilter
  - MPU6050_RPY_Display
  - USSensor

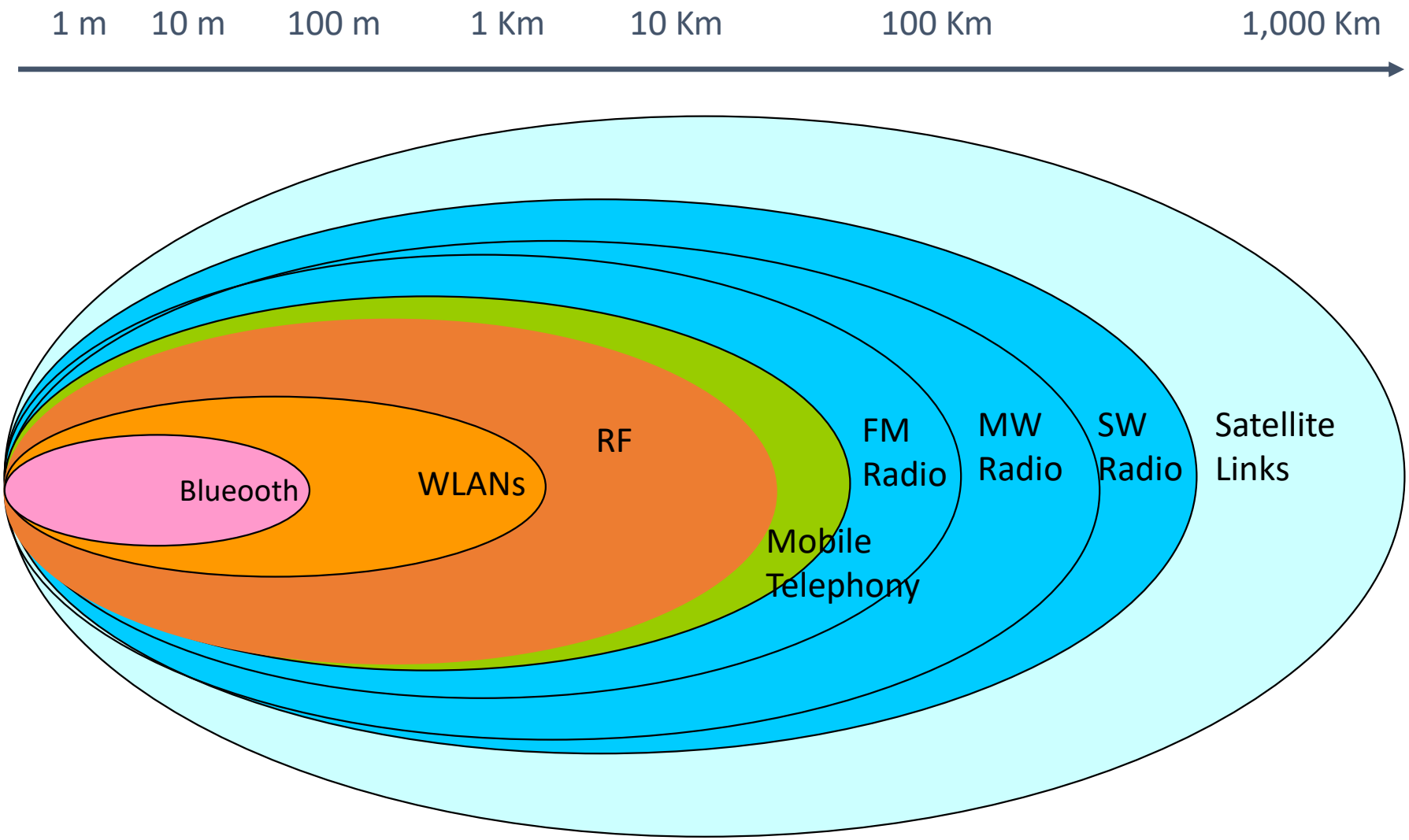
Wireless Communications



Electromagnetic spectrum

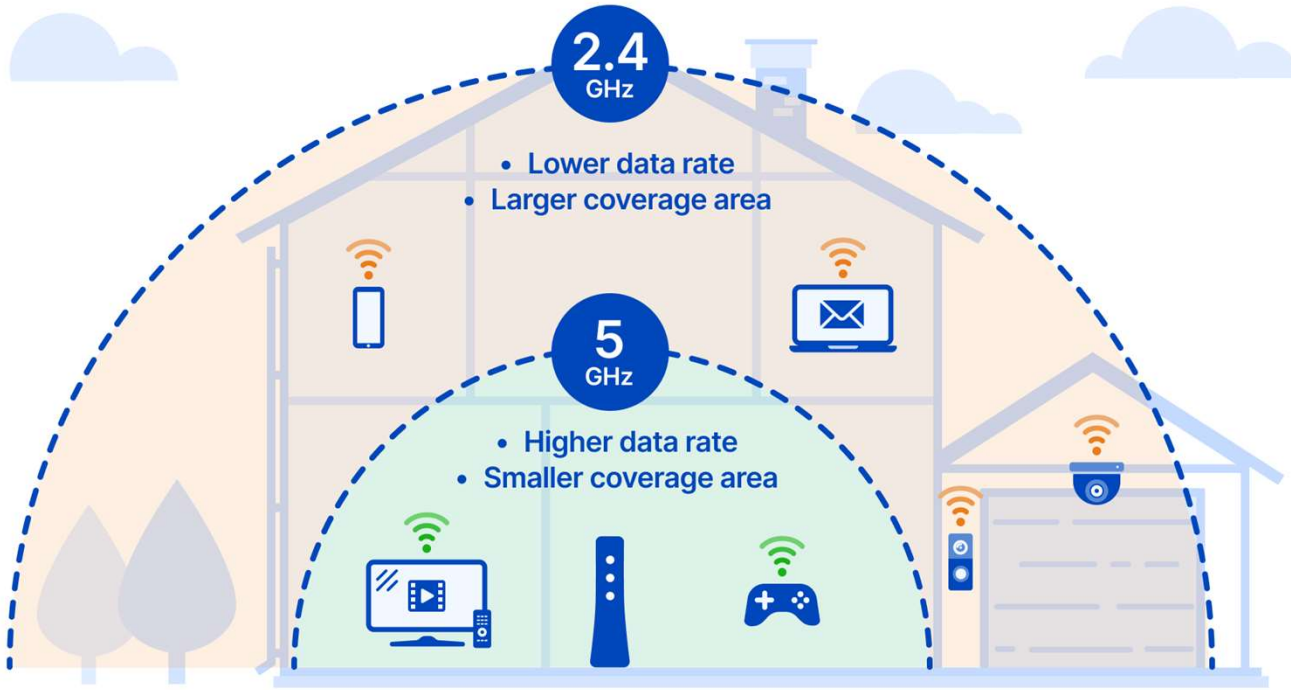


Wireless Systems: Range Comparison



WiFi

- Wi-Fi is a wireless networking technology that allows devices to communicate.



Web terminologies

- **WebSocket**: A WebSocket is a technology that allows a web page and a server to talk to each other instantly.

- **Webserver**: A webserver is a computer that stores websites and sends them to your device when you want to look at them.

- **Access Point**: An access point is a device that lets other devices connect to a WiFi network.

Codes on WiFi

- You'll first run the sketch WiFi_Connect
  - Note your ip address of your device/ESP32

- Load, understand and run the following sketches
  - esp32_weather
  - NTP_Server_time
  - esp32_websocket
  - esp32_accesspoint
  - esp32_webserver_servo

What is Bluetooth Low Energy (BLE)?

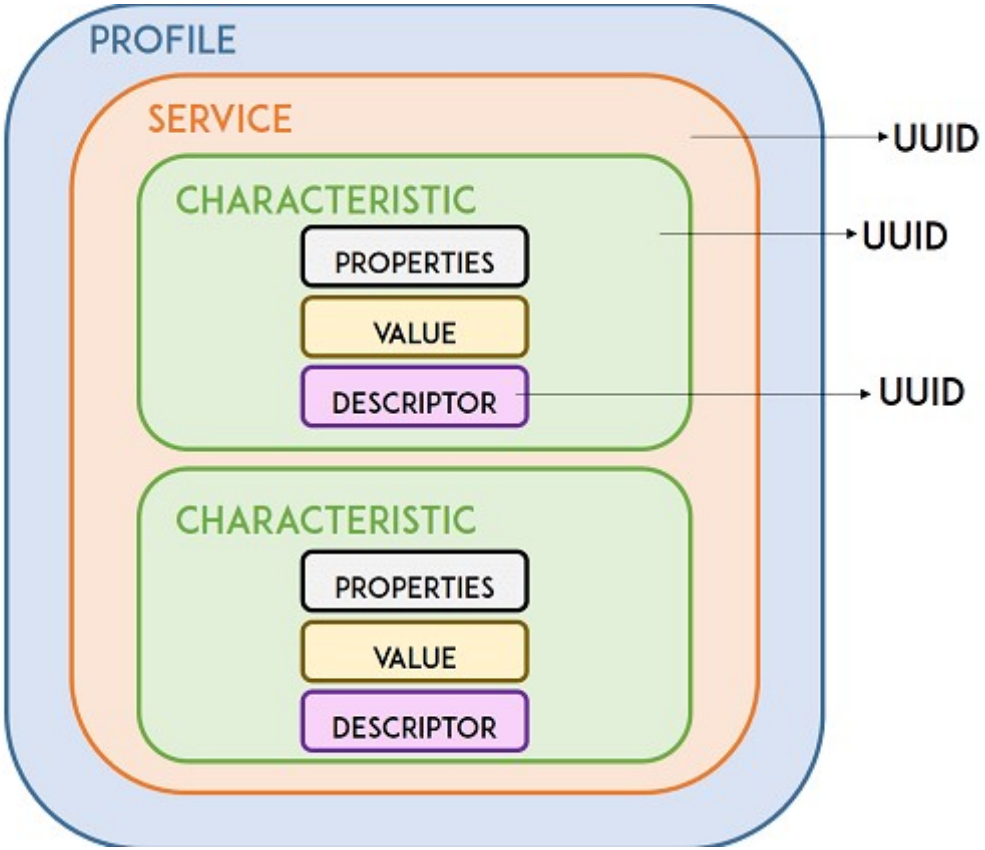
- BLE's primary application is short distance transmission of small amounts of data (low bandwidth)
- consume very low power (< 100x BT)
- point-to-point communication, broadcast mode, and mesh network
- Applications: healthcare, fitness, tracking, beacons, security, and home automation

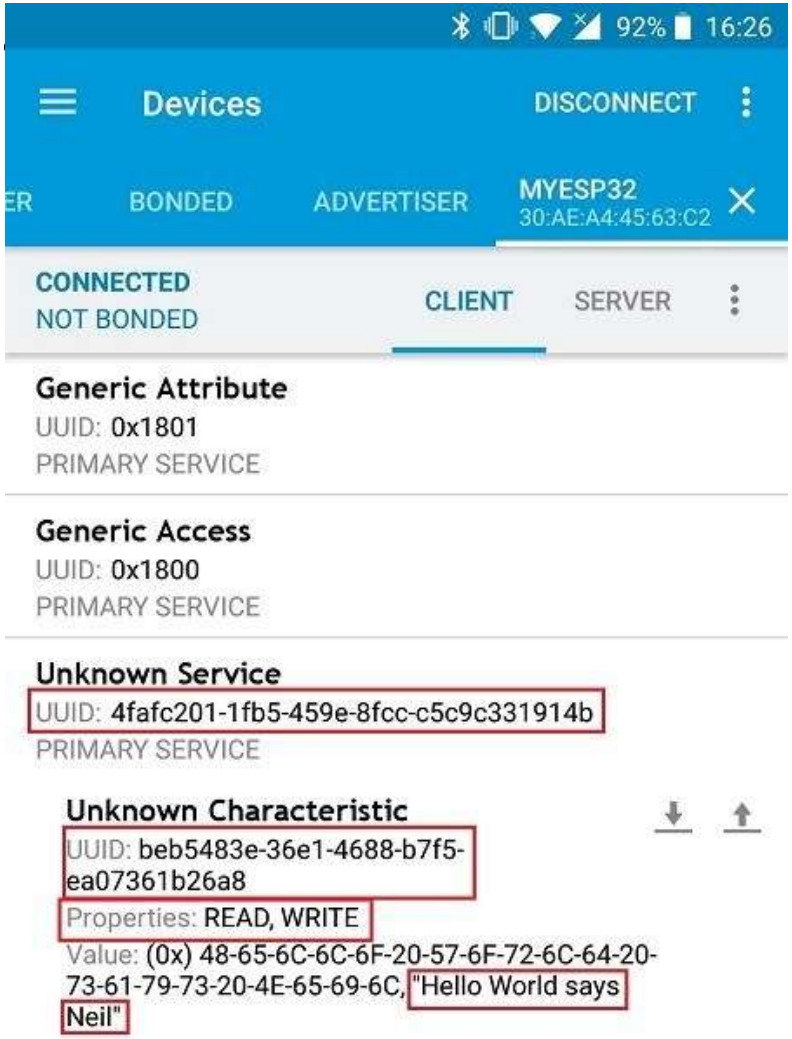
BLE Server and Client

- **Point-to-Point Communication:**
  - server advertises its existence, so it can be found by other devices, and contains the data that the client can read.
  - The client scans the nearby devices, and when it finds the server it is looking for, it establishes a connection and listens for incoming data.
- **Broadcast mode**:
  - the server transmits data to many clients that are connected;
- **Mesh network**:
  - all the devices are connected, this is a many to many connection.

GATT - Generic Attributes Profile

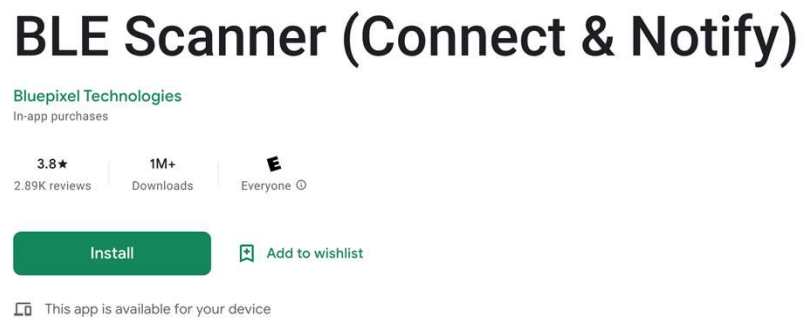
- GATT defines the way that two BLE devices send and receive standard messages

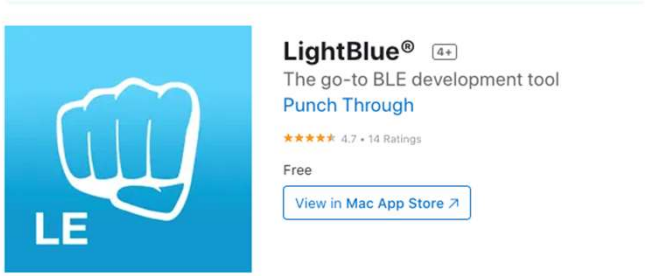




Parts Used

- FireBeetle 2 ESP32-E IoT Microcontroller with Header
- Dabble APP
- Built-in LEDs





Dabble/Stempedia

Dabble: One App for Sensing & Control

- Simplified project-making, wireless hardware control, sensing, IoT & much more.
- https://thestempedia.com/product/dabble/

Dabble Arduino Library

- https://github.com/STEMpedia/DabbleESP32
-

Codes on BLE

- Dabble App Control
  - Install the Dabble app

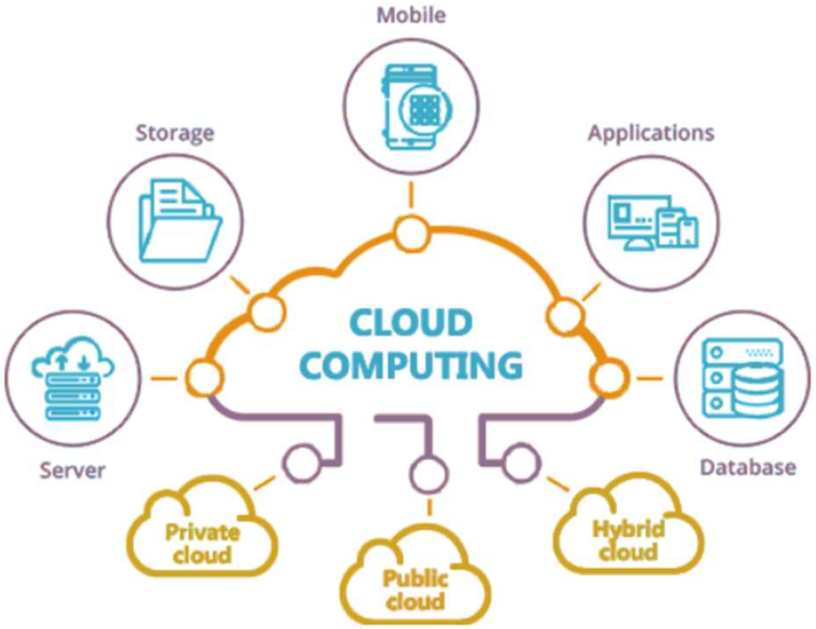
- Load, understand and run the following sketches

- Dabble to ESP32 LED state and brightness control
  - Dabble_LedBrightnessControl

- Dabble to ESP32 Neopixel Control
  - Dabble_NeopixelControl

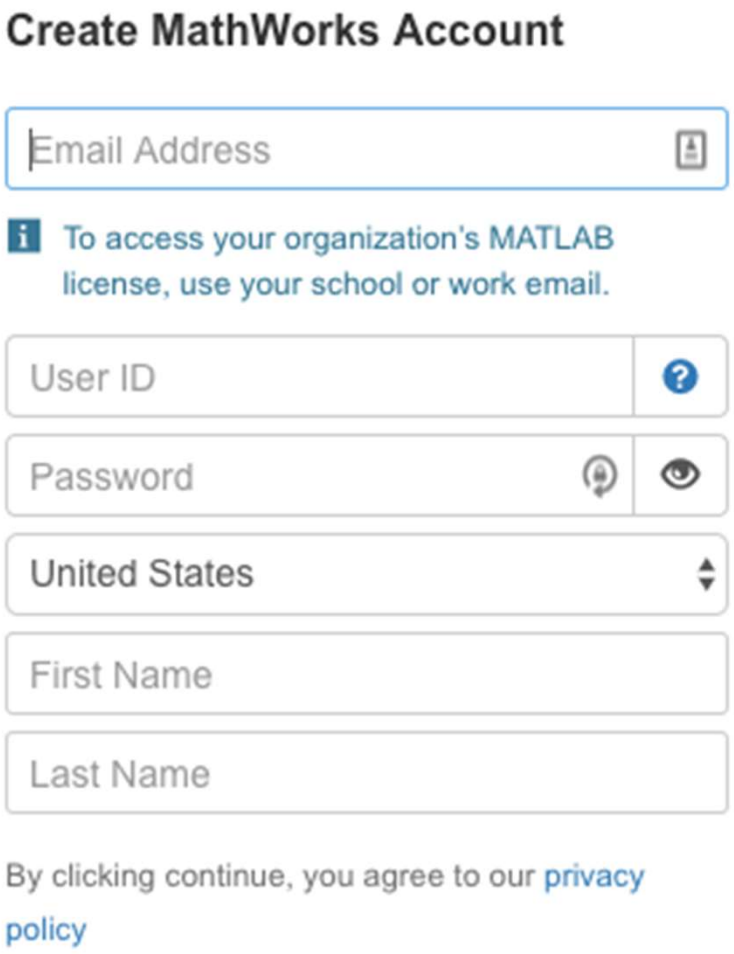
“The Cloud”

- **The Cloud -** refers to servers that are accessed over the internet, and the software and databases that run on those servers. Cloud servers are located in data centers all over the world.
- **Cloud Computing**: This involves using a network of remote servers hosted on the internet to store, manage, and process data, rather than using a local server or personal computer.

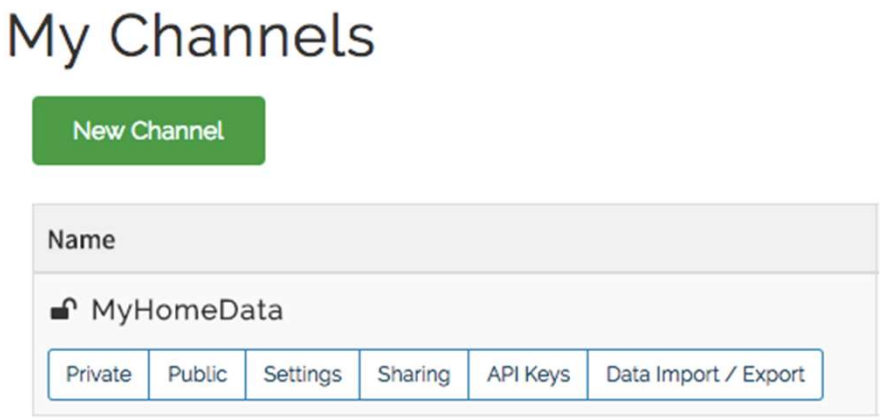


Our cloud - Thingspeak

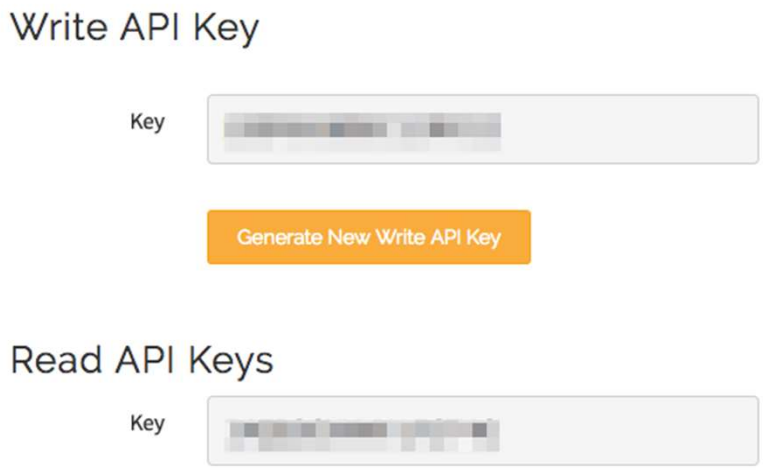
Sign up for ThingSpeak



Add new channel



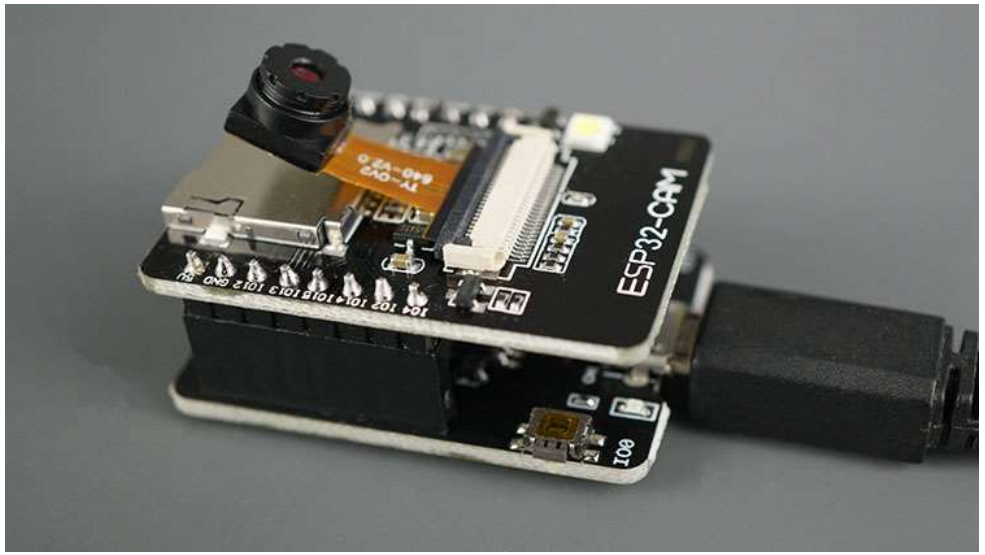
Get API keys

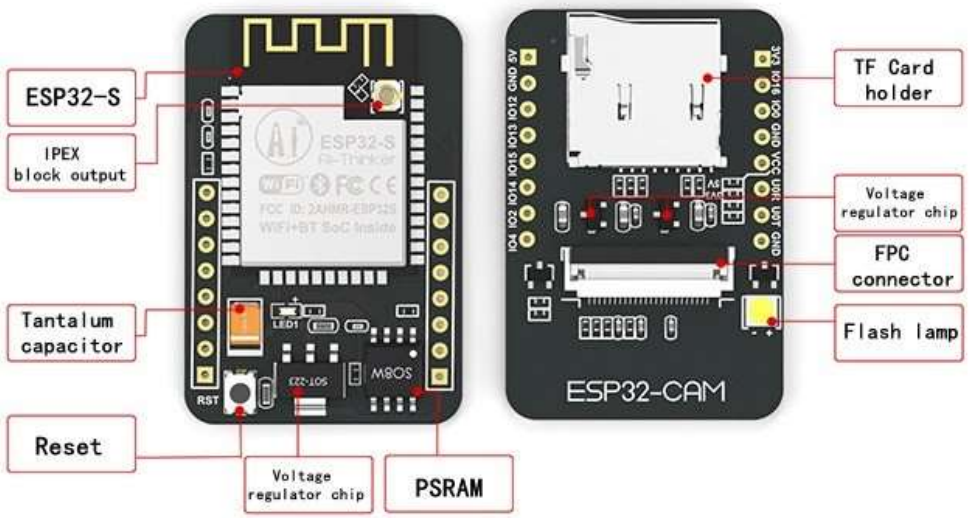


Codes on Cloud

- Sign up for a Thingspeak account using a valid email

- Load, understand and run the following sketches
  - APDS9960_Thingspeak

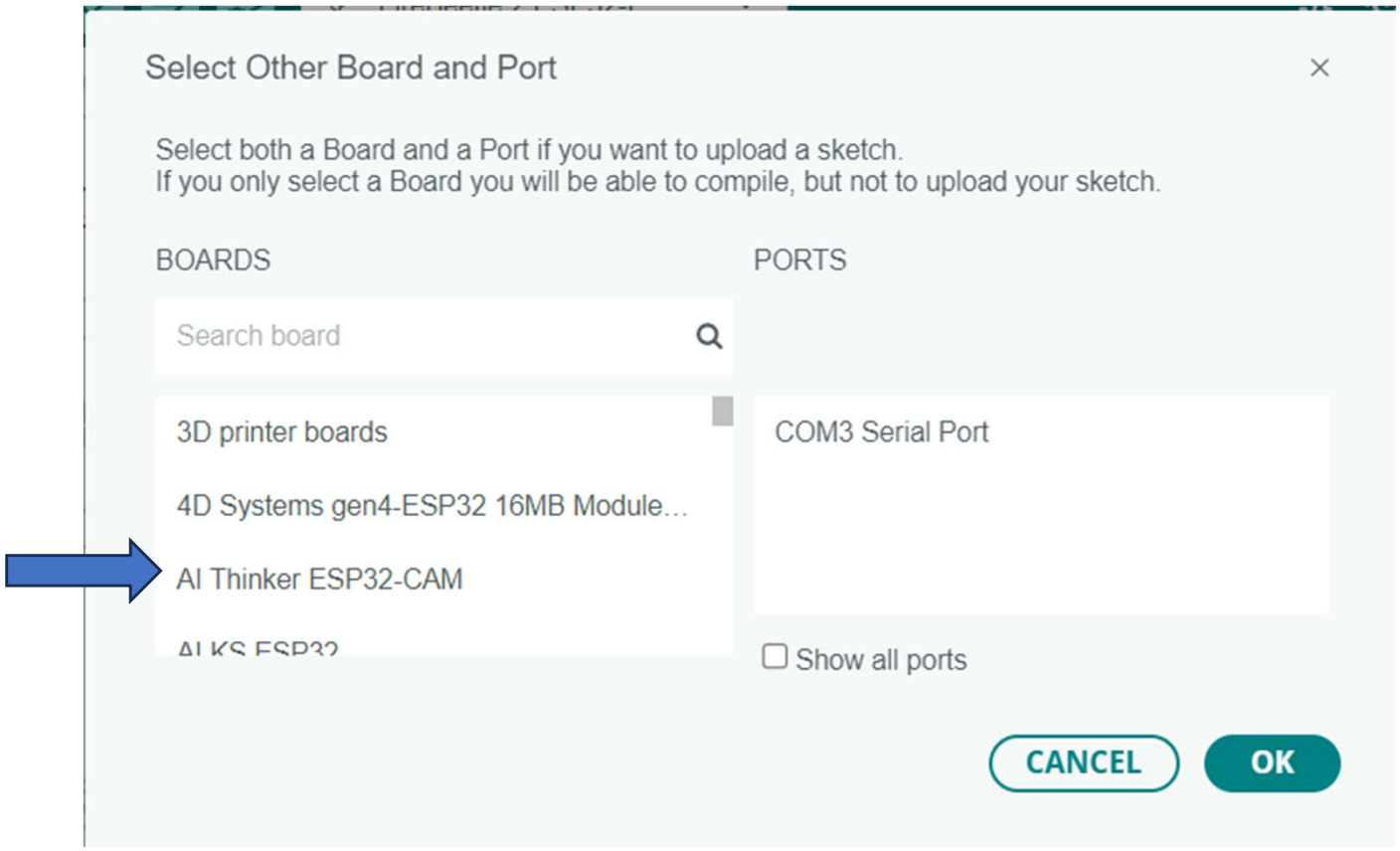




ESP32-CAM Board

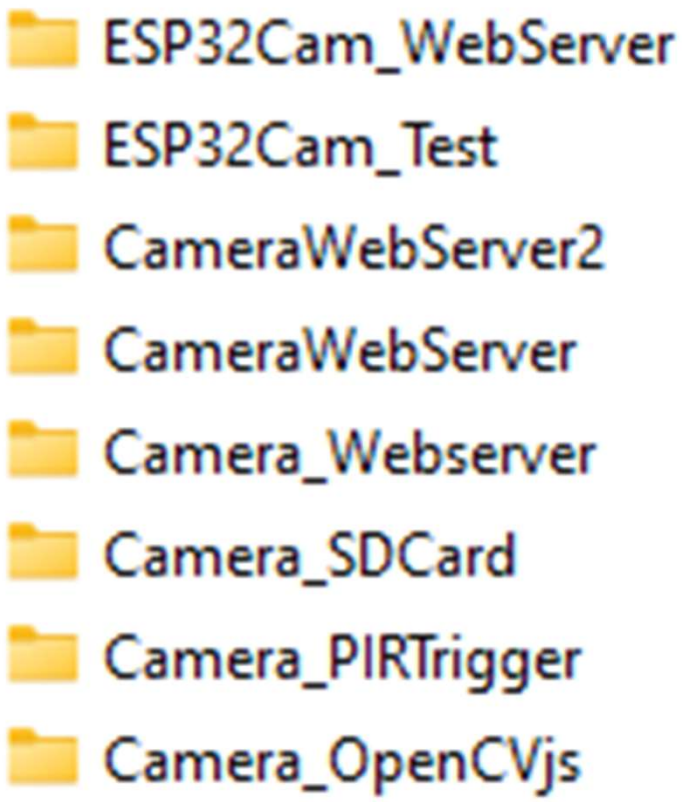
ESP32-CAM AI-Thinker

Select the Board



Codes for ESP32-CAM

- Click "ESP32 Arduino" and select “AI THINKER ESP32-CAM" motherboard from the pull-down menu;



Our Robot - Bolt

Small body, powerful function

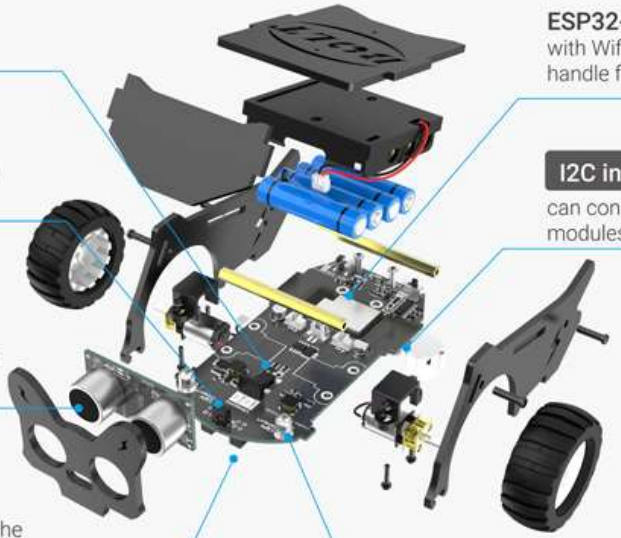
Bolt has a wealth of sensors onboard, with built-in Wifi and Bluetooth functions to meet a variety of programming environments. It can quickly realize functions such as light tracking, line tracking, obstacle avoidance, remote control, and light shows. Two expansion ports are reserved at the bottom, supporting 150+ Expansion Connections of Crowtail electronic modules and unlimited creativity.
Wait for you to play!

Buzzer
Play beautiful music

Infrared receiver
Realizes infrared remote control function

Ultrasonic ranging sensor
realize light show and automatic obstacle avoidance

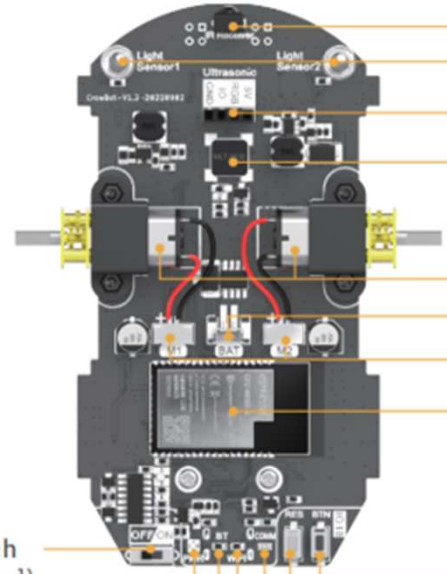
Dual line tracking sensors
allow Bolt to walk flexibly along the black line



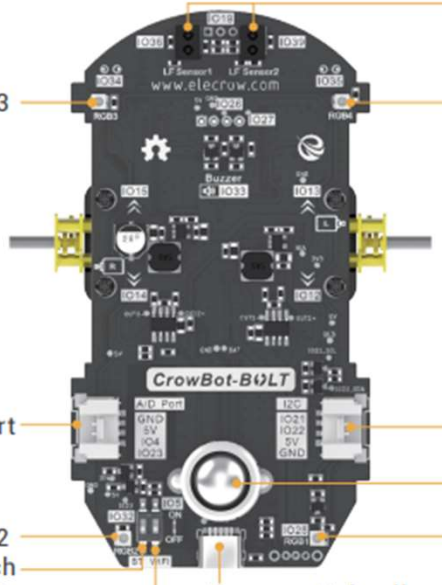
ESP32-Wrover-B Module
with Wifi and BLE, can use Bluetooth handle for remote control

I2C interface
can connect to 150+ kinds of Crowtail modules

Light diode
Detects ambient brightness and enables Bolt to automatically follow light



Infrared Receiver
Photodiode(Light Sensor)
Ultrasonic Module Interface
Buzzer
N20 DC Geared Motor
Battery Box Port
Motor Port
ESP32-C3-WROOM
Power Switch
Power Indicator[1)]
Bluetooth Indicator[2)]
WiFi Indicator[3)]
Program Start Button
Reset Button
Data Transmission Indicator[4)]

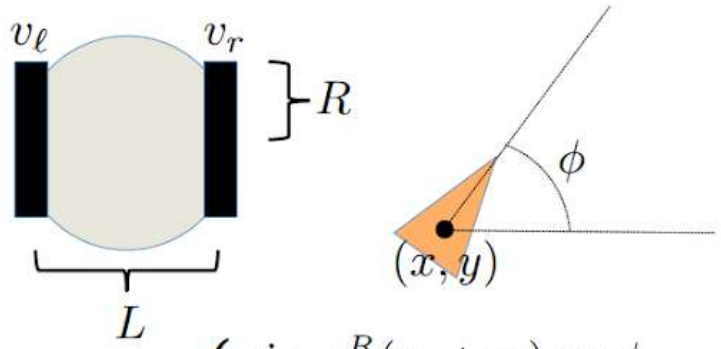


Line Tracking Sensor
RGB_LED-3
RGB_LED-4
A/D Port
I2C Port
Back Universal Wheel
RGB_LED-2
RGB_LED-1
Bluetooth Switch
WiFi Switch
Program Uploading Interface (Type-c)

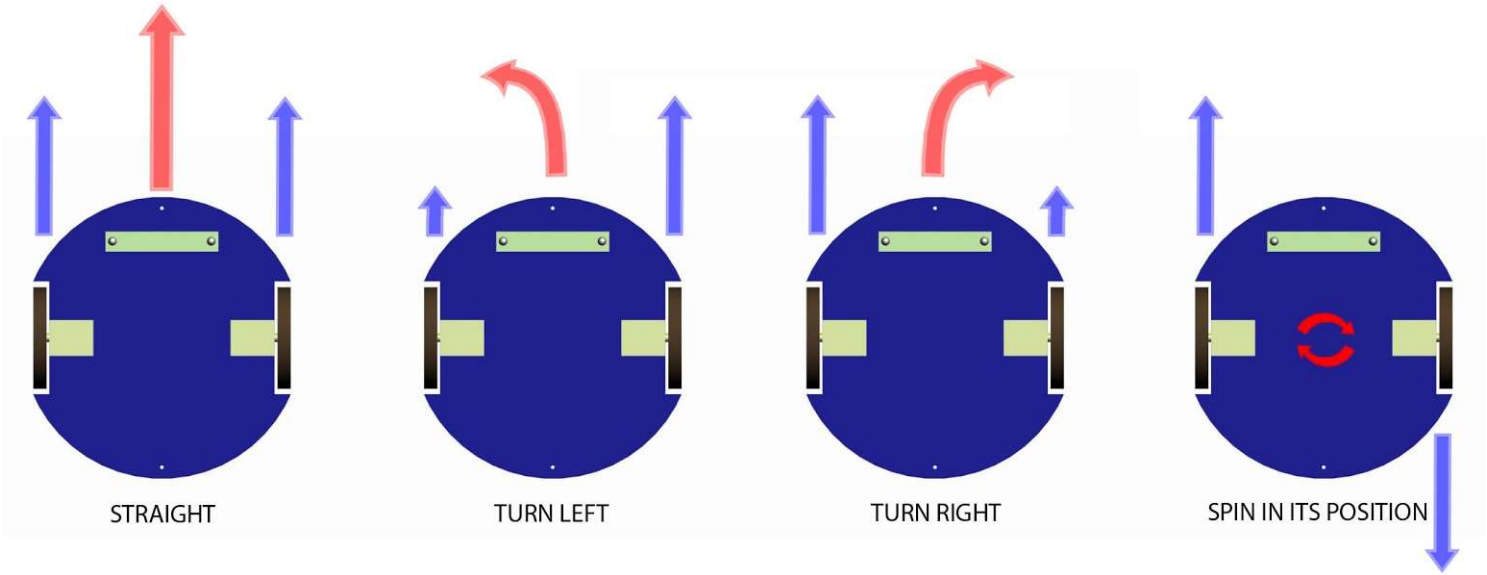
Note:

1) When the blue light is on, the power is sufficient; When the red light is on, it means that the voltage is lower than 3.3V and the battery needs to be replaced, otherwise the performance of program will be affected.
2) When the blue light is on, it means that the Bluetooth function is on.
3) When the blue light is on, the WiFi function is on.
4) When the blue light is on, it indicates that there is data transmission.

Differential Drive Robot



$$\begin{cases} \dot{x} = \frac{R}{2}(v_r + v_\ell)\cos\phi \\ \dot{y} = \frac{R}{2}(v_r + v_\ell)\sin\phi \\ \dot{\phi} = \frac{R}{L}(v_r - v_\ell) \end{cases}$$



Programming our robot

- Click "ESP32 Arduino" and select "ESP32-WROOM-DA Module" motherboard from the pull-down menu;