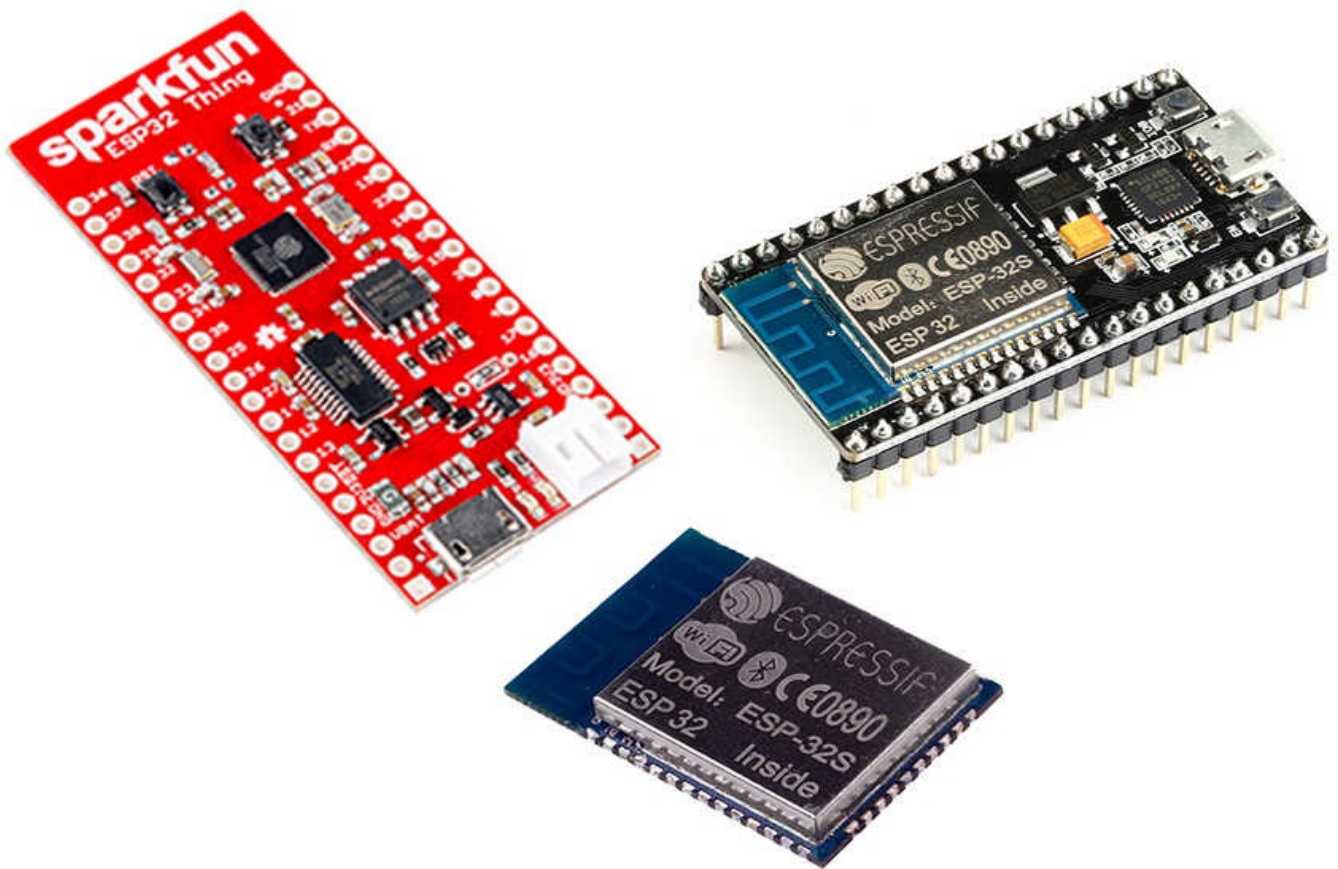


MicroPython for ESP32

Development Workshop



Agus Kurniawan

MicroPython for ESP32 Development Workshop

MicroPython for ESP32 Development Workshop

Agus Kurniawan

1st Edition, 2017

Copyright © 2017 Agus Kurniawan

Table of Contents

[MicroPython for ESP32 Development Workshop](#)

[Preface](#)

[1. Preparing Development Environment](#)

[1.1 MicroPython Boards for ESP32](#)

[1.2 Electronics Components](#)

[1.2.1 Arduino Starter Kit](#)

[1.2.2 Fritzing](#)

[1.2.3 Cooking-Hacks: Arduino Starter Kit](#)

[1.2.4 Arduino Sidekick Basic kit v2](#)

[1.2.5 Grove - Starter Kit for Arduino](#)

[1.2.6 DFRobot - Arduino Kit for Beginner v3](#)

[1.3 Development Tools](#)

[1.4 Testing](#)

[2. Setting Up MicroPython](#)

[2.1 Getting Started](#)

[2.2 Connecting MicroPython Boards to Computer](#)

[2.3 Flashing The Latest MicroPython Firmware](#)

[2.4 Development Tools](#)

[2.5 Python programming](#)

[2.6 Hello MicroPython: Blinking LED](#)

[2.6.1 Wiring](#)

[2.6.2 Writing Program Using Serial/UART Tool](#)

[2.7 Uploading Python Script File to MicroPython Board](#)

[3. GPIO Programming](#)

3.1 Getting Started

3.2 Wiring

3.3 Writing a Program

3.4 Testing

4. PWM and Analog Input

4.1 Getting Started

4.2 Demo Analog Output (PWM) : RGB LED

4.2.1 Wiring

4.2.2 Writing Program

4.2.3 Testing

4.3 Demo Analog Input: Working with Potentiometer

4.3.1 Wiring

4.3.2 Writing Program

4.3.3 Testing

5. Working with I2C

5.1 Getting Started

5.2 Writing Program

5.3 Writing Program

5.4 Testing

6. Working with UART

6.1 Getting Started

6.2 Wiring

6.3 Writing a Program

6.4 Testing

7. Working with SPI

7.1 Getting Started

7.2 Wiring

7.3 Writing a Program

[7.4 Testing](#)

[8. Working with DHT Module](#)

[8.1 Getting Started](#)

[8.2 Wiring](#)

[8.3 Writing MicroPython Program](#)

[8.4 Testing](#)

[9. Working with WiFi](#)

[9.1 Getting Started](#)

[9.2 Scanning WiFi Hotspot](#)

[9.3 Developing WiFi Application](#)

[Source Code](#)

[My Books for ESP8266 Development](#)

[Contact](#)

Preface

This book was written to help anyone want to get started with MicroPython development for ESP32 boards. It describes the basic elements of MicroPython development.

Agus Kurniawan

Depok, August 2017

1. Preparing Development Environment

1.1 MicroPython Boards for ESP32

MicroPython is a lean and efficient implementation of the Python programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments. This book will focus on MicroPython for ESP32 chip.

The following is a sample of ESP32 Module.



We can deploy MicroPython on ESP32 boards. For instance, you can deploy on these boards:

- Espressif ESP32 Development Board, <https://www.adafruit.com/product/3269>
- SparkFun ESP32 Thing, <https://www.sparkfun.com/products/13907>
- Watterott ESP-WROOM-32-Breakout, <http://www.watterott.com/de/ESP-WROOM32-Breakout>
- NodeMCU-32S Lua WiFi IOT Development board, <http://www.aliexpress.com>
- Adafruit HUZZAH32 – ESP32 Feather Board, <https://www.adafruit.com/product/3405>

You can find other development board based on ESP32.

NodeMCU-32S Lua WiFi IOT Development board:



SparkFun ESP32 Thing.



Watterott ESP-WROOM-32-Breakout.



1.2 Electronics Components

We need electronic components to build our testing, for instance, Resistor, LED, sensor devices and *etc.* I recommend you can buy electronic component kit. We can use electronics kit from Arduino to be developed on MicroPython board. The following is a list of electronics kit which can be used in our case.

1.2.1 Arduino Starter Kit

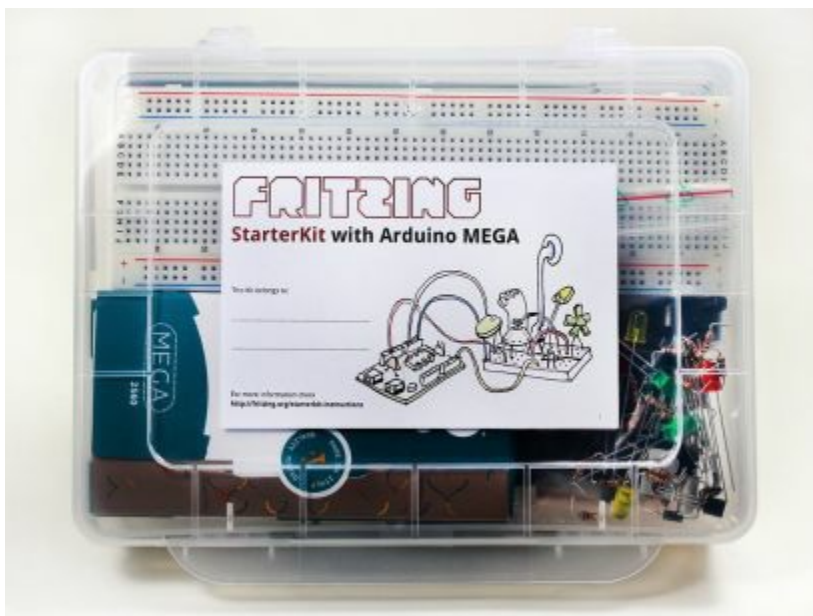
Store website: <http://arduino.cc/en/Main/ArduinoStarterKit>



1.2.2 Fritzing

Store website: <http://shop.fritzing.org/> .

You can buy Fritzing Starter Kit with Arduino UNO or Fritzing Starter Kit with Arduino Mega.



1.2.3 Cooking-Hacks: Arduino Starter Kit

Store website: <http://www.cooking-hacks.com/index.php/shop/arduino/starter-kits/arduino-starter-kit.html>

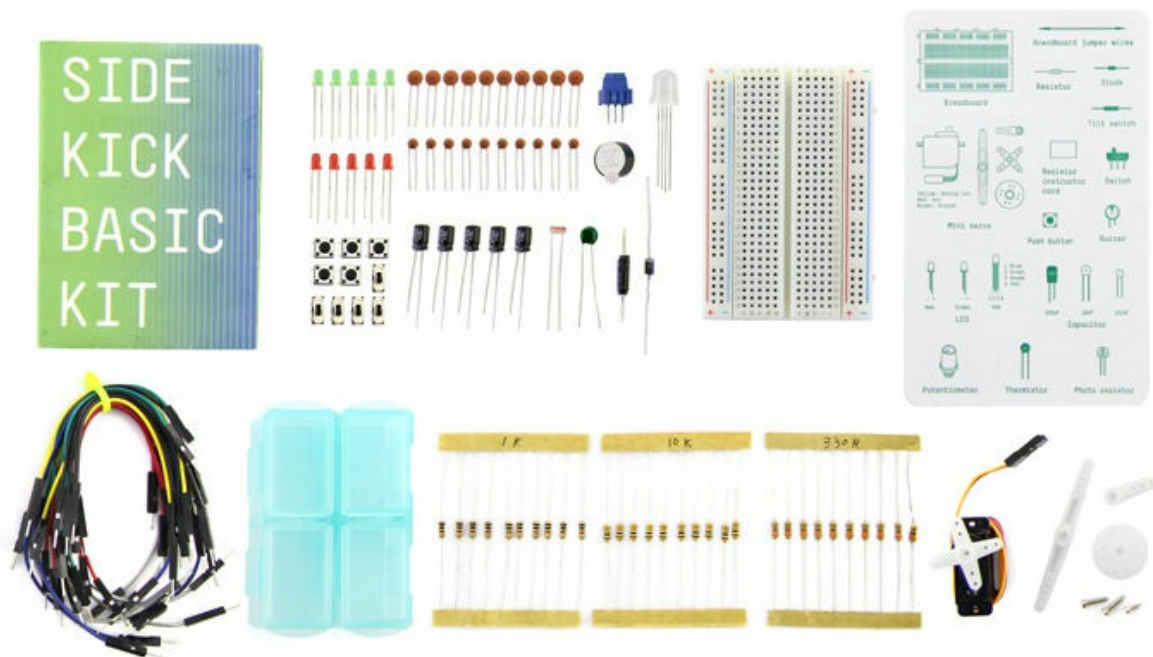


1.2.4 Arduino Sidekick Basic kit v2

Store website: <http://www.seeedstudio.com/depot/Sidekick-Basic-Kit-for-Arduino-V2-p-1858.html>

You also can find this kit on this online store.

<http://www.exp-tech.de/seeedstudio-sidekick-basic-kit-for-arduino-v2>



1.2.5 Grove - Starter Kit for Arduino

Another option, you can buy this kit on Seeedstudio,
<http://www.seeedstudio.com/depot/GroveStarter-Kit-for-Arduino-p-1855.html> .

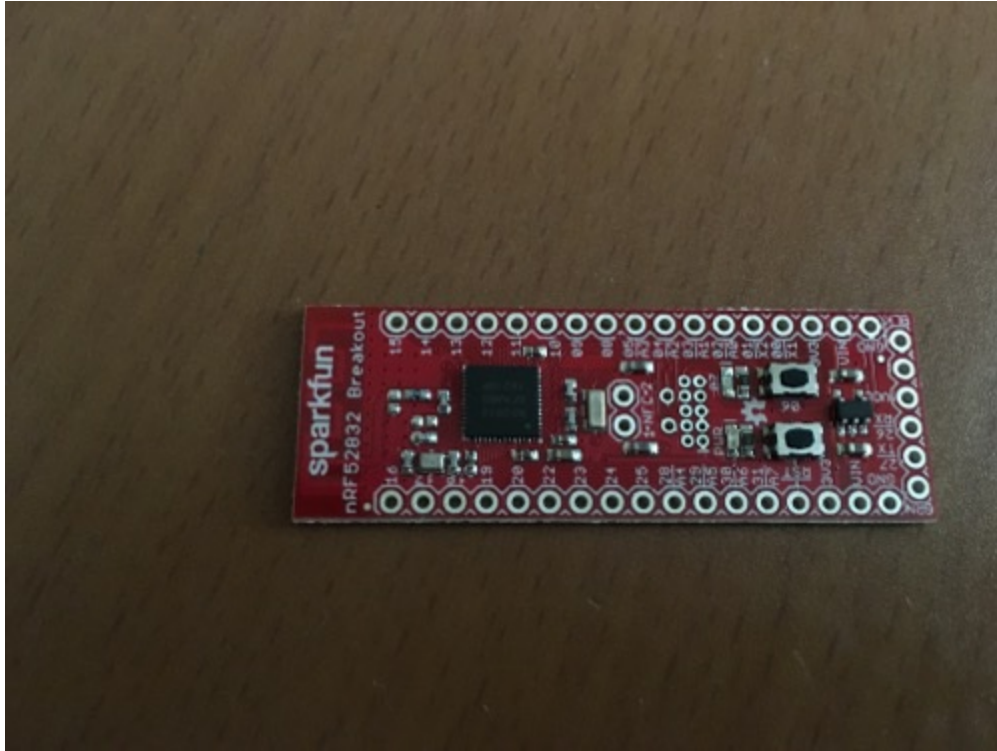


1.3 Development Tools

To develop app with MicroPython target, we can use any editor. You can learn how to install it on chapter 2.

1.4 Testing

For testing, I used SparkFun ESP32 Thing boards for MicroPython on Windows, Linux and Mac.



I also used Arduino Sidekick Basic kit for electronic components and some sensor and actuator devices.



2. Setting Up MicroPython

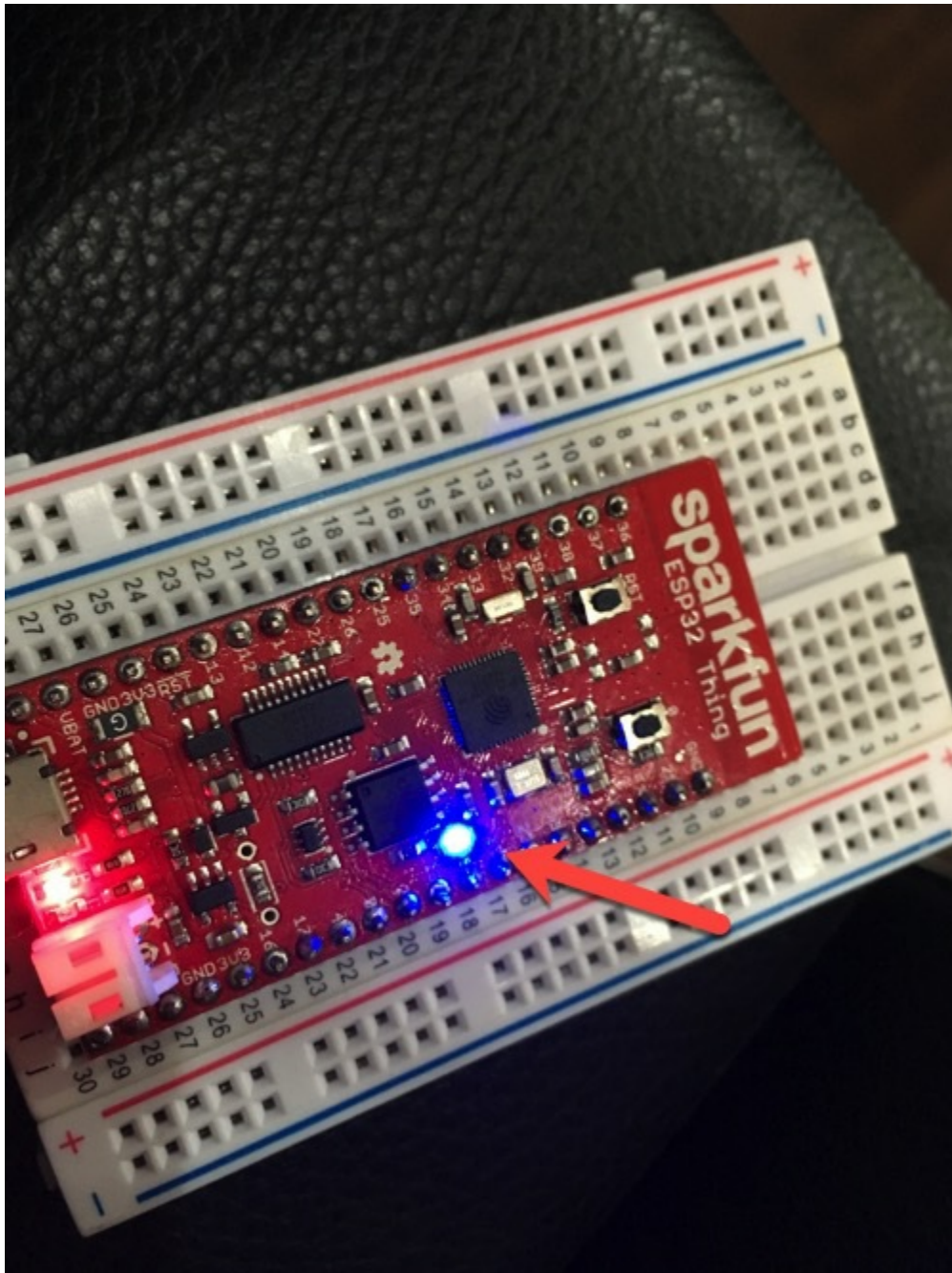
This chapter explains how to work on setting up MicroPython board.

2.1 Getting Started

In this chapter, we learn how to get started with MicroPython board. We try to reflash the latest MicroPython firmware and then test some basic scripts. For testing, I use SparkFun ESP32 Thing as sample for MicroPython board.

2.2 Connecting MicroPython Boards to Computer

Firstly, you connect MicroPython board to PC via USB/microUSB cable. After connected, you may get lighting on blue LED, for instance for, SparkFun ESsp32 Thing board.



If you are working on Windows platform, open Device Manager, you

should see ESP32 board detected on Ports (COM & LPT). If you don't see your ESP32 board on Windows, you should install a windows driver for your ESP32 board.

For Mac, you can check it on Terminal. Type this command.

```
ls devtty.usb*
```

For Linux, you can use `ls devtty*` .

Then, you see a list of driver.

For sample, you can see my board that is recognized as `devtty.usbserial-DN02MX9H` .

A screenshot of a macOS Terminal window. The title bar shows three colored window control buttons (red, yellow, green) on the left, and a blue folder icon followed by the text 'codes — -bash — 80x14' on the right. The terminal content shows a user prompt 'agusk\$' followed by the command 'ls /dev/tty.usb*'. The output of the command is '/dev/tty.usbserial-DN02MX9H'. Below the output, the prompt 'agusk\$' is shown again with a cursor, indicating the command has finished executing.

```
codes — -bash — 80x14
agusk$ ls /dev/tty.usb*
/dev/tty.usbserial-DN02MX9H
agusk$
```

2.3 Flashing The Latest MicroPython Firmware

In this section, we try to flash the latest MicroPython firmware. You can get the latest MicroPython firmware

<https://micropython.org/download#esp32>. Download *.bin file.

In this scenario, I try to flash MicroPython firmware on Windows, OSX and Linux.

We can use esptool.py tool from <https://github.com/espressif/esptool> to deploy MicroPython firmware to ESP32 boards. This tool also is recommended for Windows platform. You can install it via pip.

```
$ pip install esptool
```

You can download the latest MicroPython for ESP32 on <https://micropython.org/download#esp32>. It's daily build firmware. Firstly, we clear the existing firmware on ESP32 board. Then, we flash MicroPython.

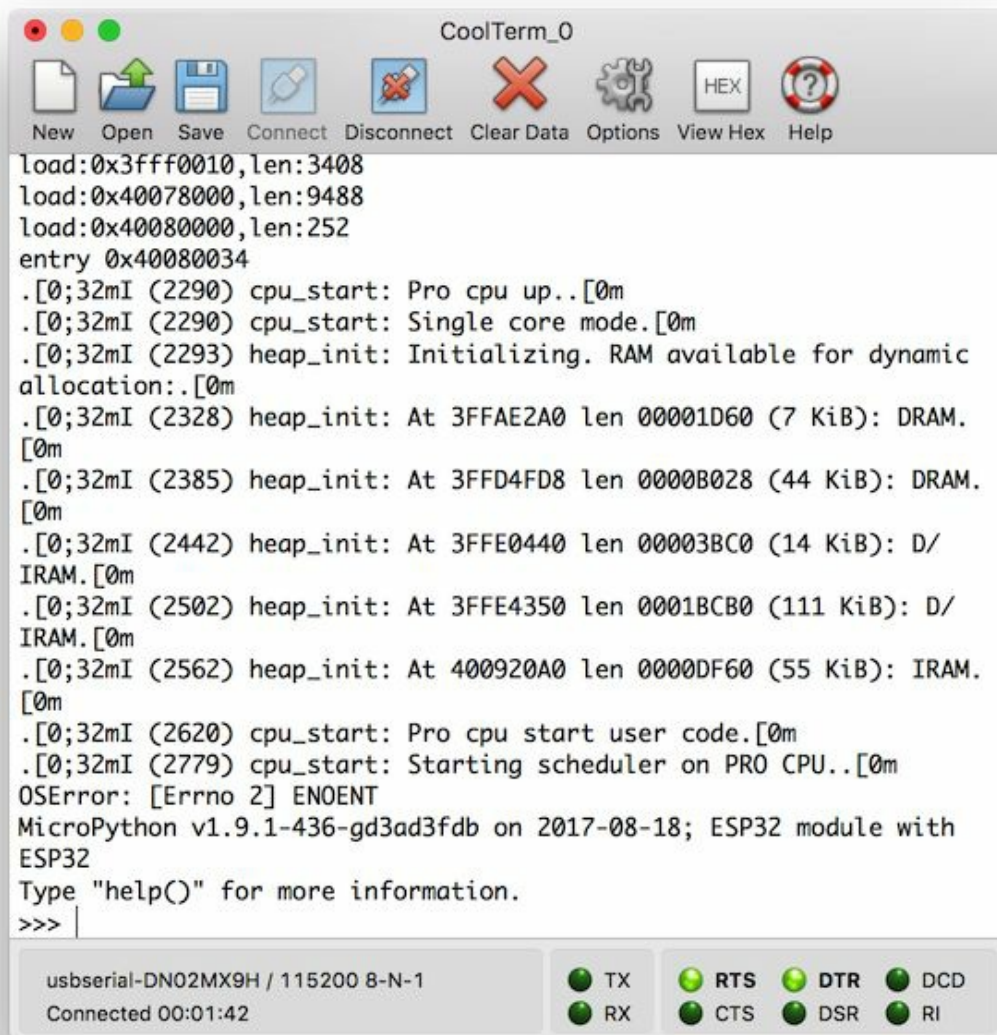
For instance, I flash my SparkFun ESP32 Thing on port *devcu.usbserial-DN02MX9H* and MicroPython firmware, *esp32-20170818-v1.9.1-436-gd3ad3fdb.bin*. The flash baudrate is 115200.

```
$ esptool.py --port devtty.usbserial-DN02MX9H erase_flash  
$ esptool.py --chip esp32 --port devtty.usbserial-DN02MX9H
```

```
codes — -bash — 80x34
[agusk$ ls /dev/tty.usb*
/dev/tty.usbserial-DN02MX9H
[agusk$ esptool.py --port /dev/tty.usbserial-DN02MX9H erase_flash
esptool.py v2.0.1
Connecting.....
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 0)
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 6.1s
Hard resetting...
[agusk$ esptool.py --port /dev/tty.usbserial-DN02MX9H --baud 460800 write_flash -
-flash_size=detect 0 esp32-20170818-v1.9.1-436-gd3ad3fdb.bin
esptool.py v2.0.1
Connecting.....
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 0)
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 1085904 bytes to 603722...
Wrote 1085904 bytes (603722 compressed) at 0x00000000 in 14.8 seconds (effective
585.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting...
agusk$ █
```

After completed flashing, you reset your board. Then, using a serial tool, such as CoolTerm (<http://freeware.the-meiers.org>), you should connect to ESP32 board. Don't forget to use baudrate 115200. After connected, you should see ">>>". If you don't see it, please press ENTER on your keyboard.

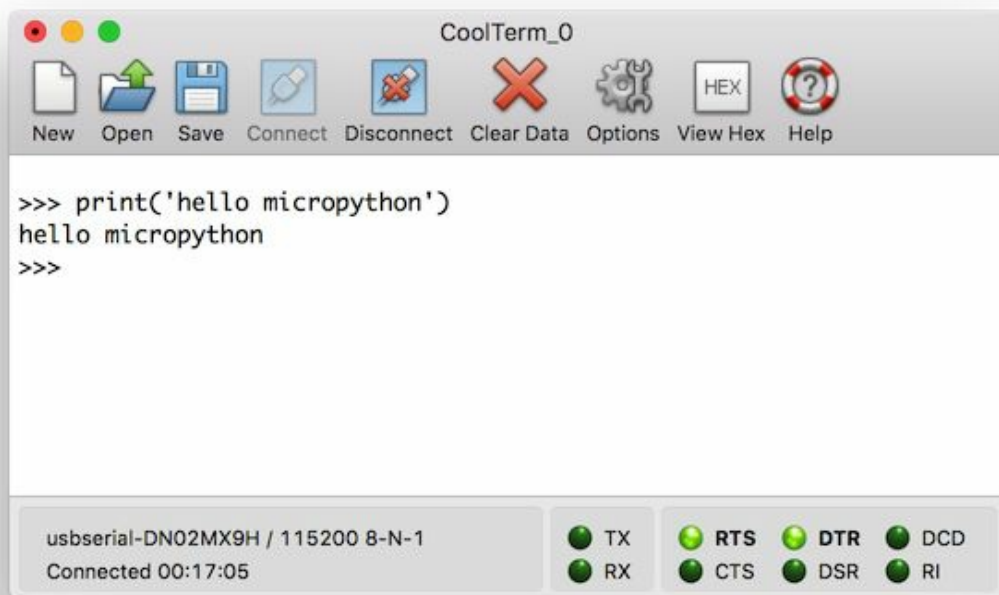


If you want to see booting messages on MicroPython, you can press RST (Reset) button your ESP32 board.

For testing, you can type this command.

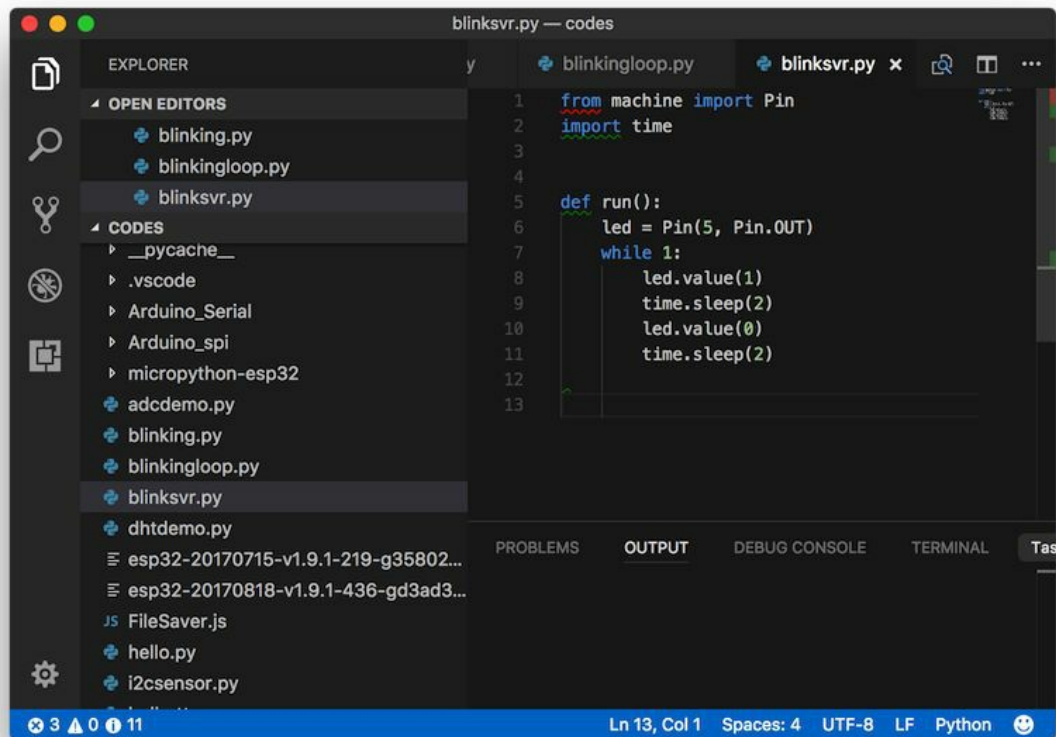
```
>>> print('hello micropython')
```

You should see "hello micropython" response from MicroPython board.



2.4 Development Tools

To write MicroPython codes, you can use any text editor, for instance, Visual Studio Code from Microsoft, <http://visualstudio.com>.

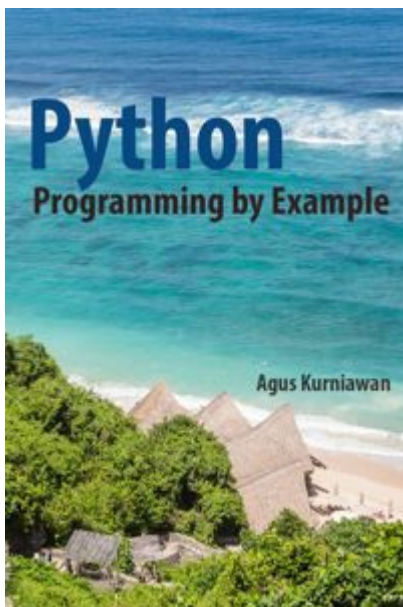


2.5 Python programming

You should have basic knowledge about Python programming to develop MicropPython program. I recommend you read Python tutorial on books or online tutorial.

I also write a book about Python, with titled **Python Programming by Example**. You can review it on

<http://blog.aguskurniawan.net/post/pythonbook01.aspx>.

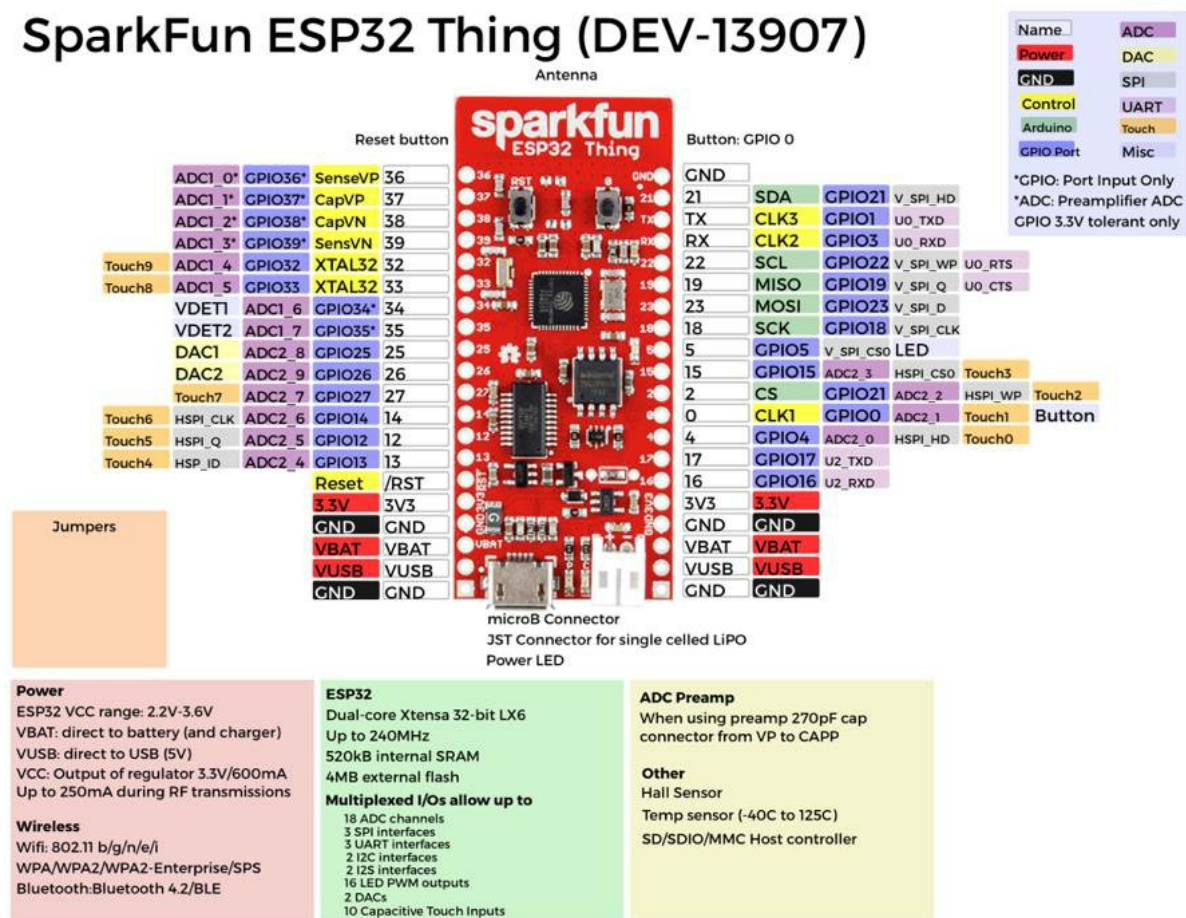


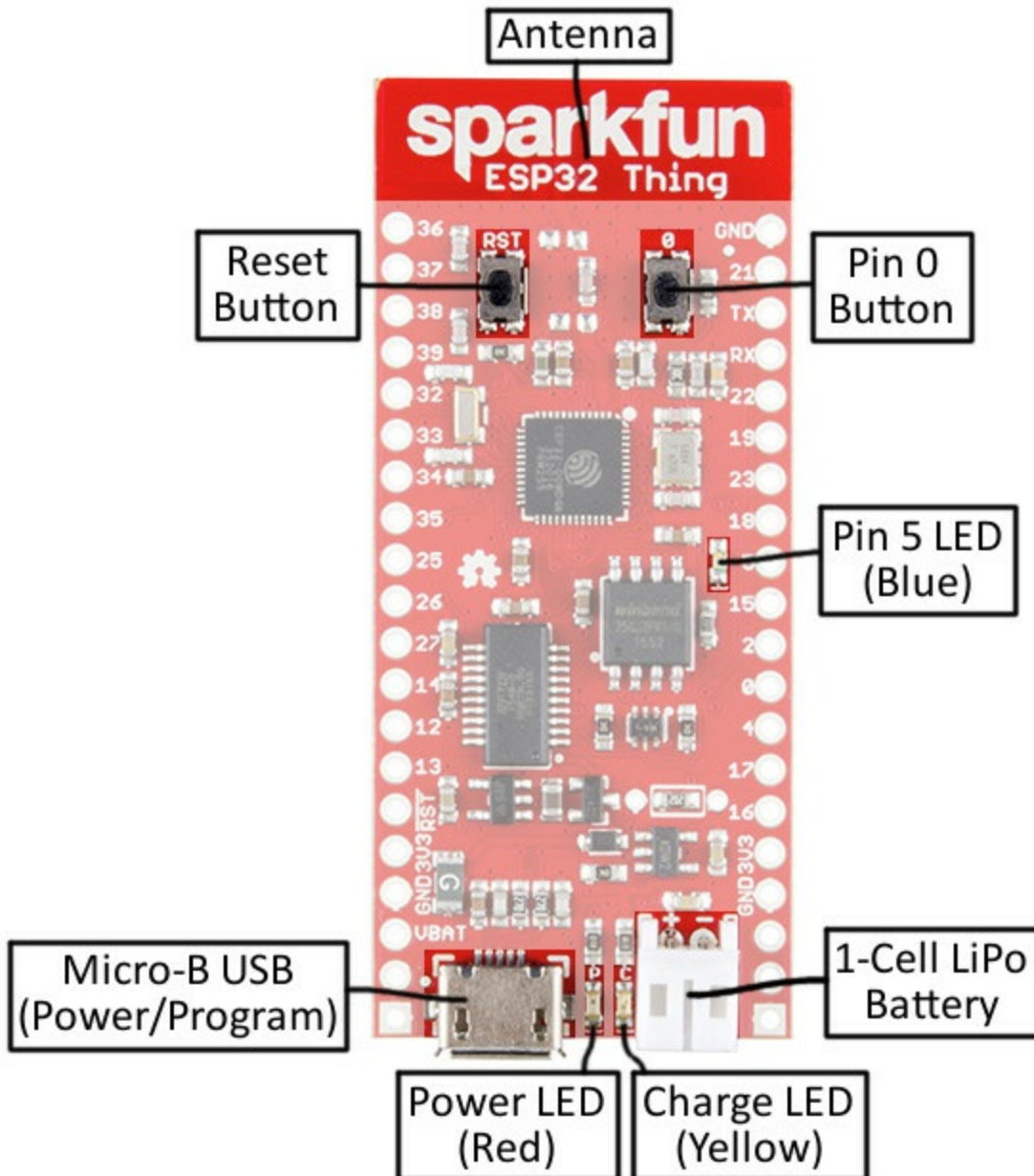
2.6 Hello MicroPython: Blinking LED

In this section, we build a blinking LED program via Python shell via MicroPython firmware. We use serial tool to write a program.

Firstly, you must know MicroPython board layout, for instance ESP32 on SparkFun ESP32 Thing board, you can see the layout as follows.

SparkFun ESP32 Thing (DEV-13907)





2.6.1 Wiring

In this case, we don't do anything. ESP32 board usually provides built-in LED. For my ESP32 board, SparkFun ESP32 Things, has already built-in LED that is connected to GPIO5.

Now you connect ESP32 board to Computer via USB cable.

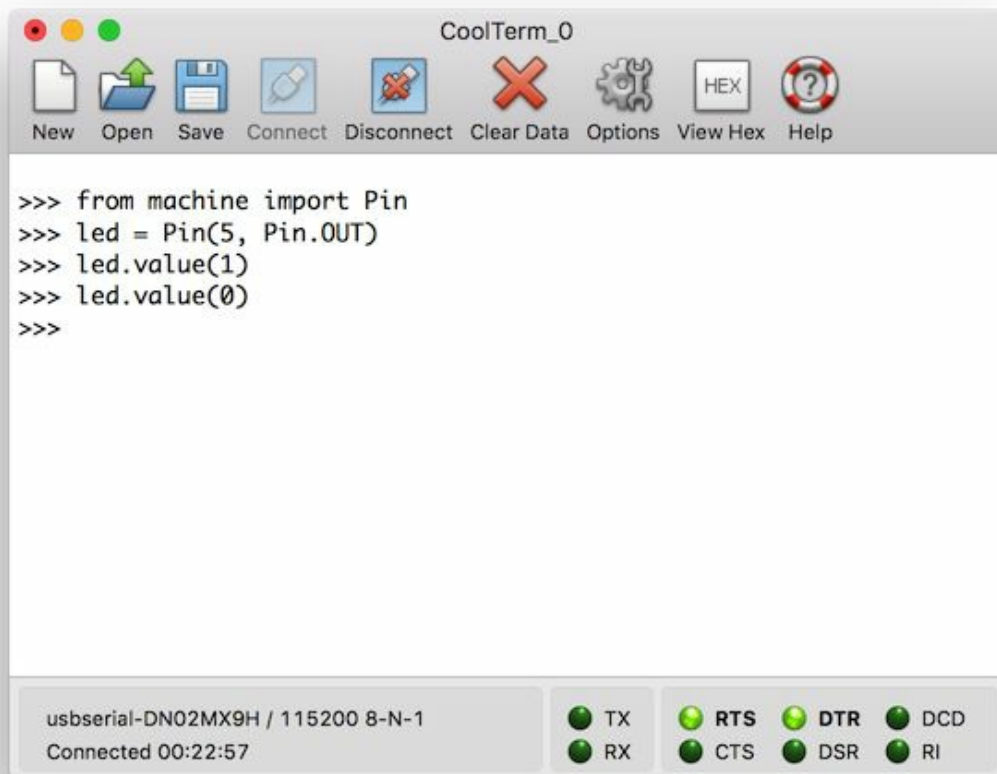


2.6.2 Writing Program Using Serial/UART Tool

The first demo is to blink a LED. Type this script per lin on Serial/UART tool.

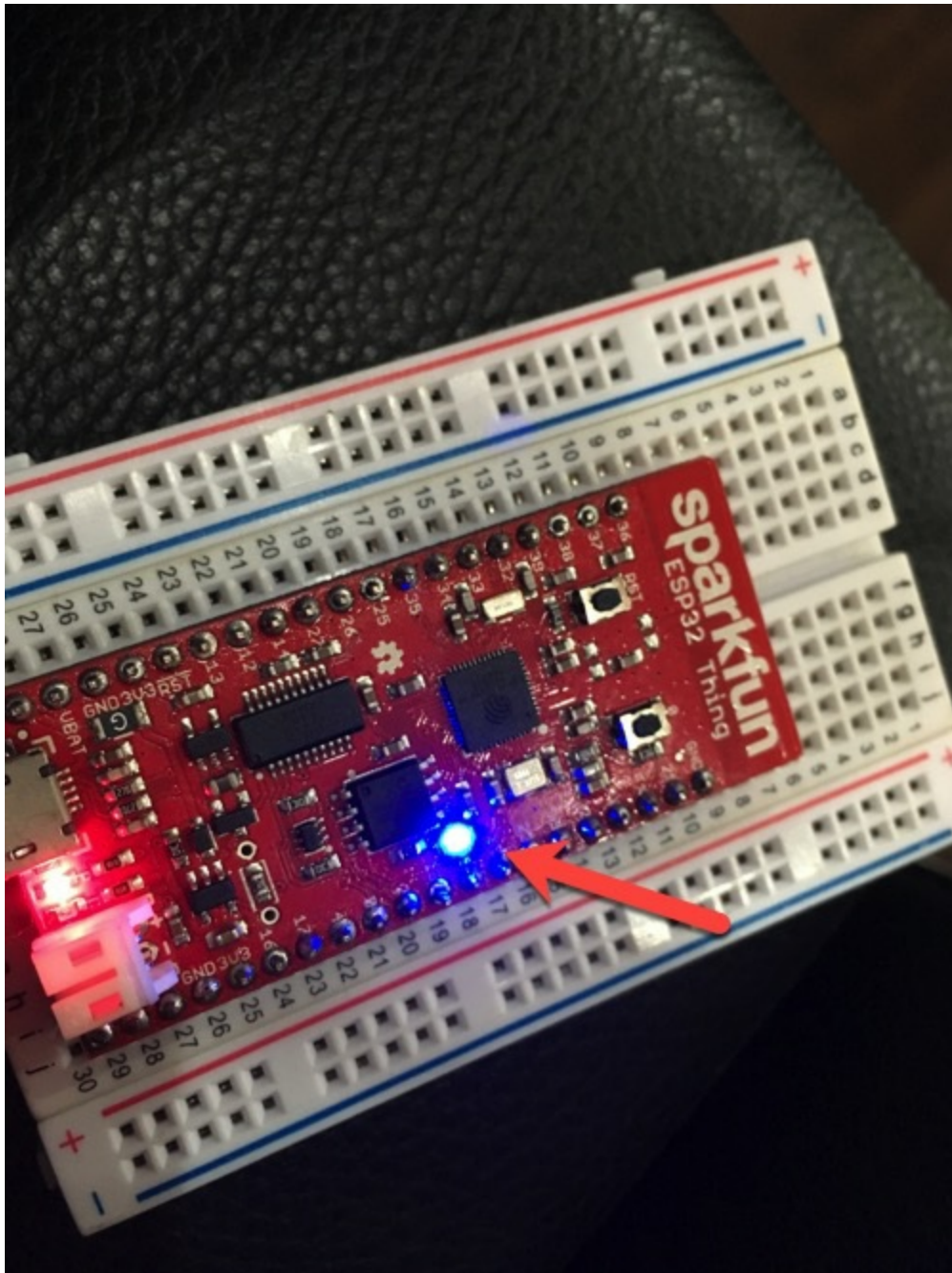
```
from machine import Pin

led = Pin(5, Pin.OUT)
led.value(1) # turn on
led.value(0) # turn off
```



This script will turn on a LED on GPIO5 and then turn off it. You can change GPIO value based on your ESP32 board.

If you should see lighting LED after executed `led.value(1)`.



We want to run blinking continuously. Now we try to build a blinking LED. Type this script.

```
from machine import Pin
import time

led = Pin(5, Pin.OUT)

while 1:
```

```
led.value(1)
time.sleep(2)
led.value(0)
time.sleep(2)
```

The last script, press backspace/delete key keyboard to exit from while looping. Then, press ENTER to start the program.



You should see a blinking LED. If you want to stop, you can press RST button on ESP32 board.

2.7 Uploading Python Script File to MicroPython Board

In this section, I show you how to upload our Python file to MicroPython ESP32 board such as SparkFun ESP32 Thing board and then execute it. Firstly, we use the same wiring on previous section. Then, we create a file, called `blinksrv.py`, and write these scripts.

```
from machine import Pin
import time

def run():
    led = Pin(5, Pin.OUT)
    while 1:
        led.value(1)
        time.sleep(2)
        led.value(0)
        time.sleep(2)
```

To upload this file, we can use `ampy` from Adafruit. You can install it using `pip`.

```
$ pip install adafruit-ampy
```

If you are interested this tool, you can visit the official project on <https://github.com/adafruit/ampy>.

You can upload Python file using `ampy`.

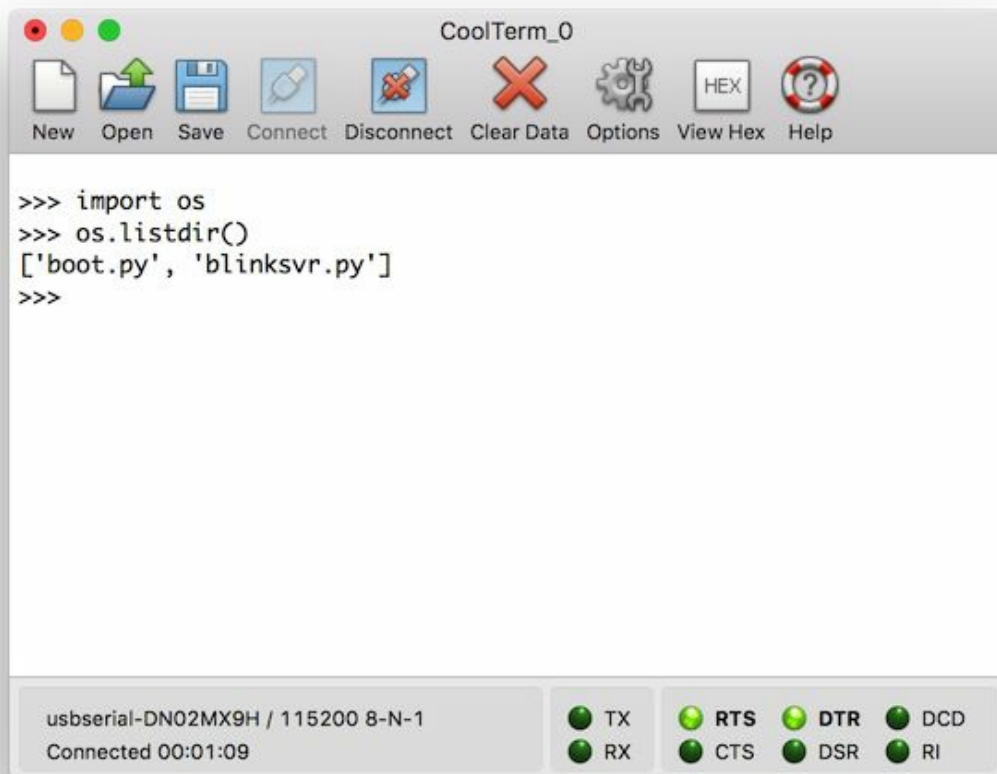
```
$ ampy --port devtty.usbserial-DN02MX9H put blinksrv.py
```

A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, followed by a blue folder icon and the text "codes — -bash — 80x18". The terminal content shows a user prompt "agusk\$" followed by the command "ampy --port /dev/tty.usbserial-DN02MX9H put blinksvr.py". The command has been executed, and the prompt "agusk\$" is shown again on the next line with a cursor. The terminal window has a light gray border and a white background.

```
codes — -bash — 80x18
agusk$ ampy --port /dev/tty.usbserial-DN02MX9H put blinksvr.py
agusk$
```

You change values on port and file name,

Now you can check this file on MicroPython terminal.

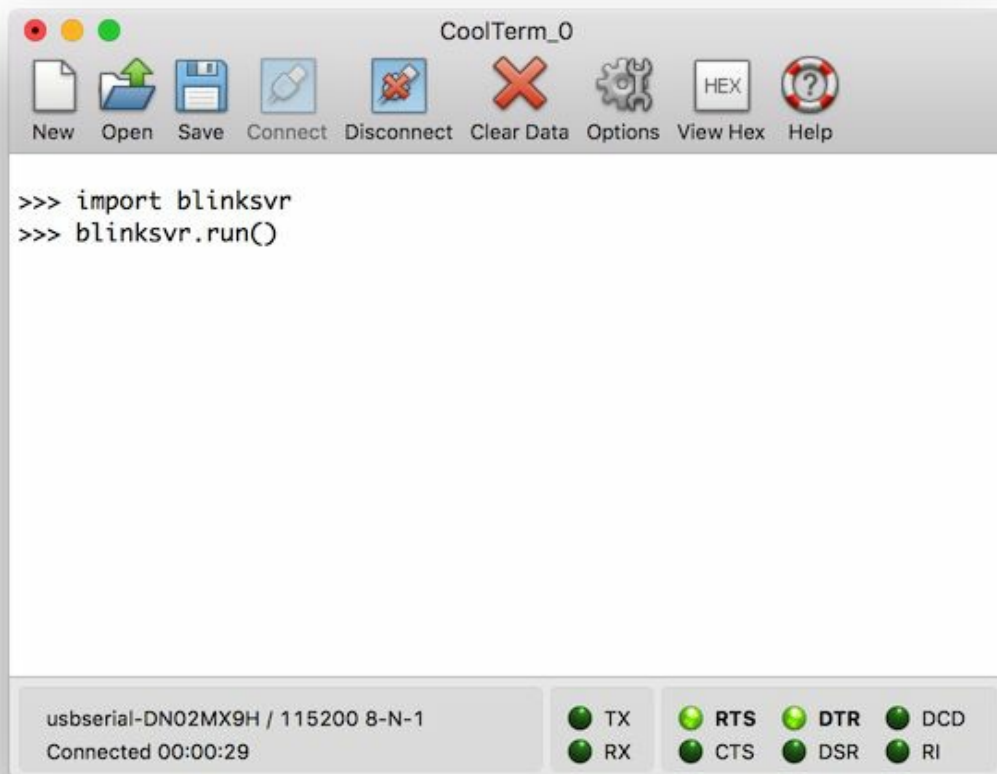


If you have problems while upload the file, you should reset your board. Make sure there is no application that uses board serial port.

Then, you can run by typing these command on serial terminal.

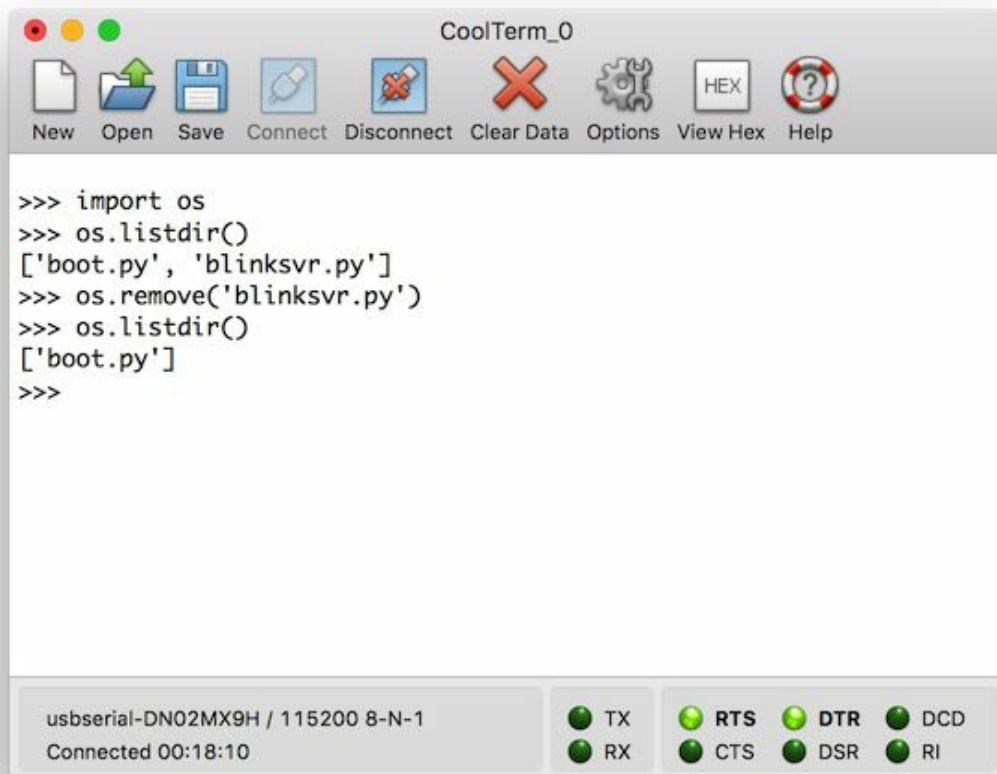
```
>>> import blinksvr
>>> blinksvr.run()
```

You should blinking LED.



As we know, your MicroPython board may have limited storage. If you want to delete the file, you can type this command, for instance, deleting `blinksvr.py`.

```
>>> import os
>>> os.remove('blinksvr.py')
```



If your program is still running, please reset your board and then delete the file.

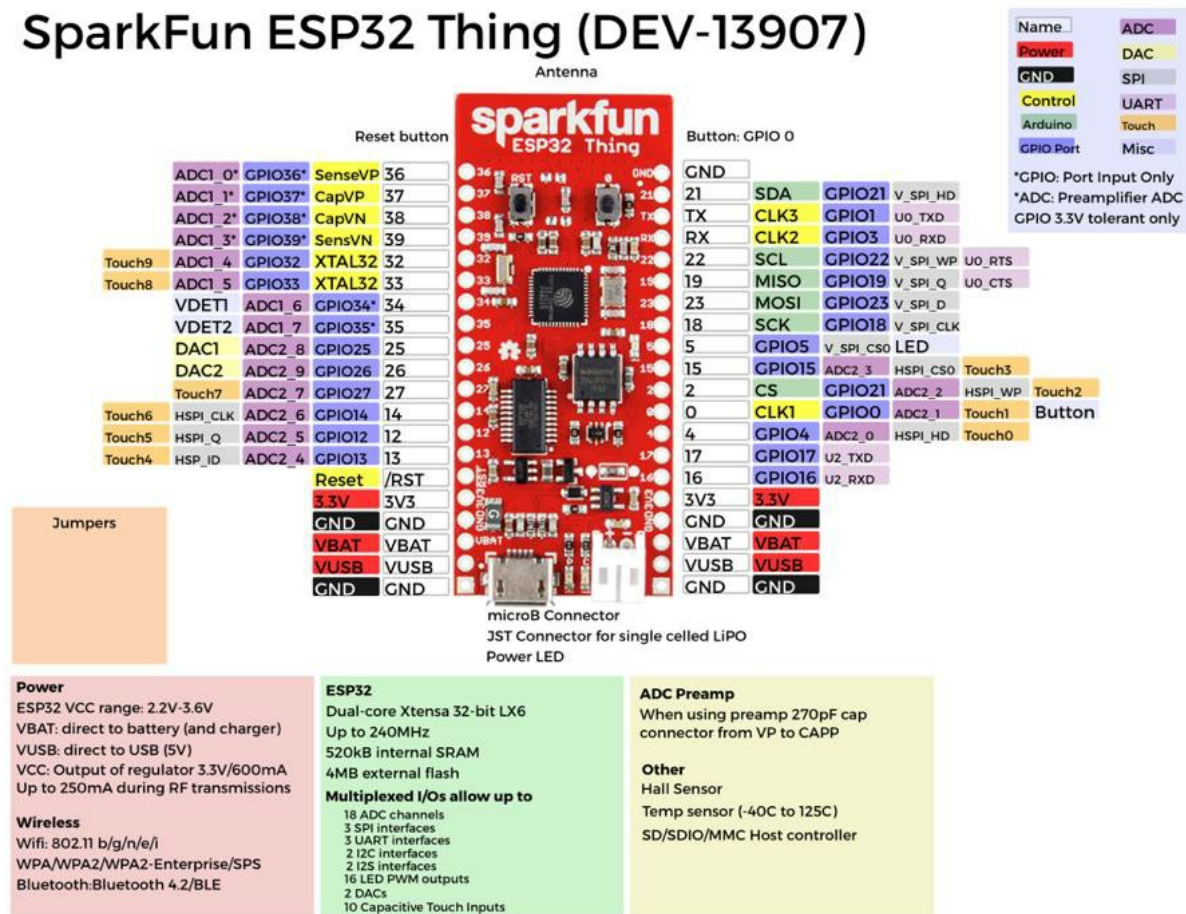
3. GPIO Programming

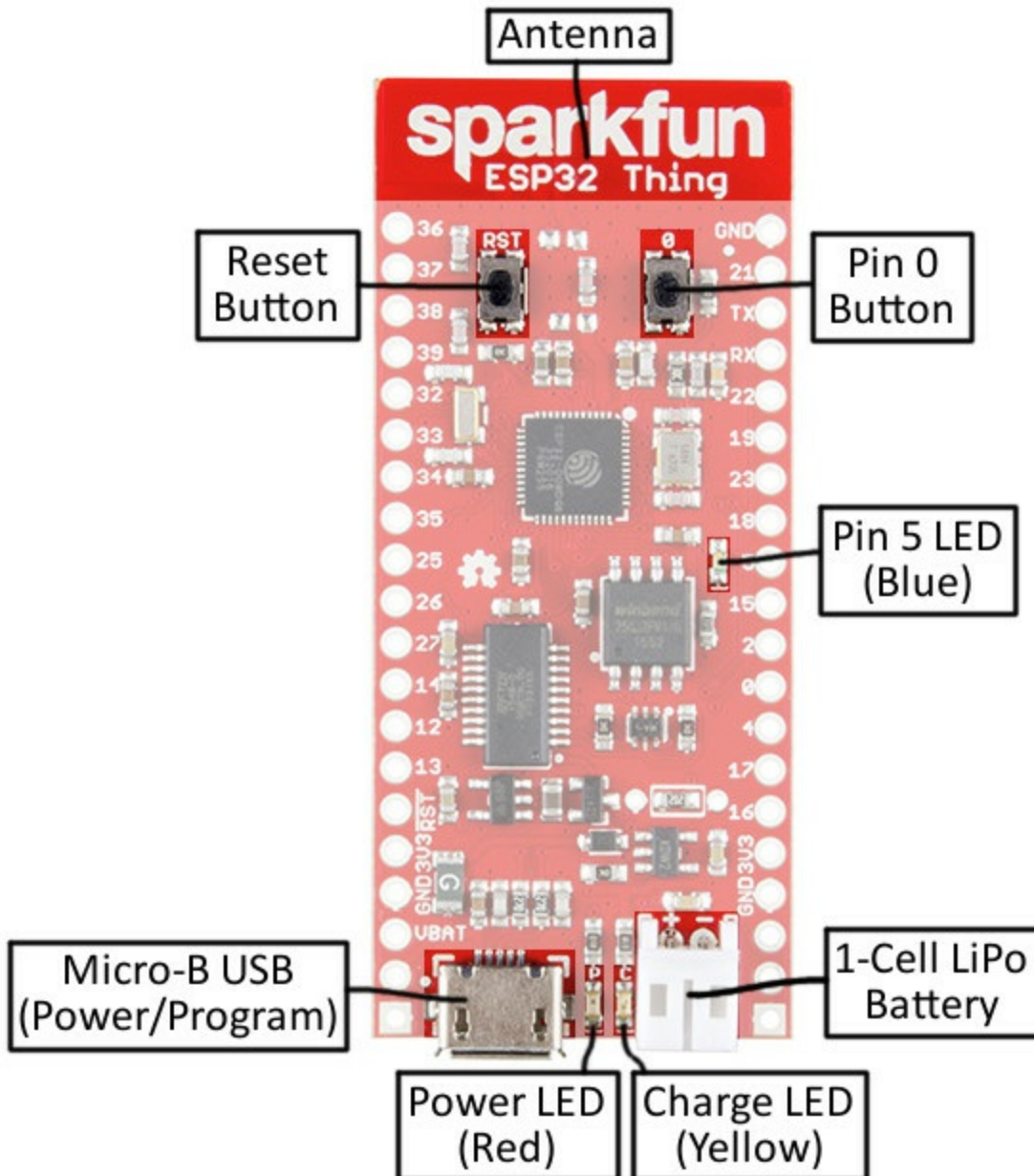
In this chapter I'm going to explain how to work with GPIO on MicroPython.

3.1 Getting Started

In general, GPIO can be used to control digital I/O on MicroPython boards. For MicroPython board-based ESP32, you should GPIO pins that are exposed by ESP32 board. Please check your board layout. For instance, the following is SparkFun ESP32 Thing layout.

SparkFun ESP32 Thing (DEV-13907)





In this chapter, we build a program to illustrate how MicroPython GPIO work. We need a LED and a pushbutton. For testing, I used SparkFun ESP32 Thing board as MicroPython board.

Let's start!.

3.2 Wiring

Since I use SparkFun ESP32 Thing board, a LED and push button are already available. A LED is connected to GPIO5 and a push button is connected to GPIO0.

3.3 Writing a Program

To create a program, we just create a new Python file, called **ledbutton.py**. Then, write these scripts.

```
from machine import Pin

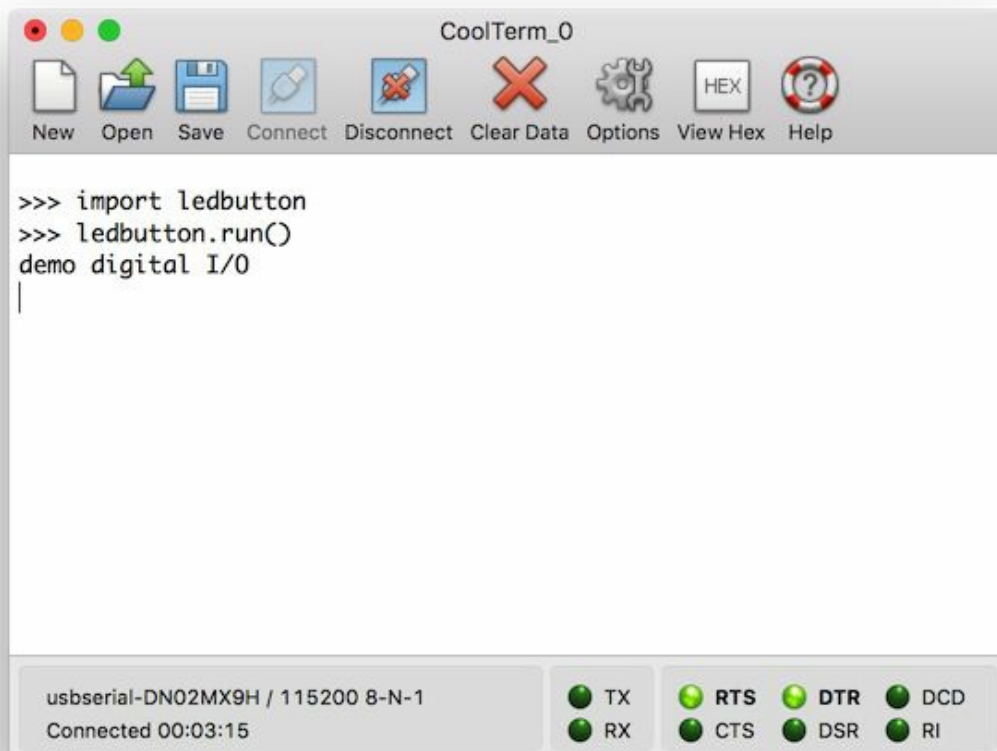
def run():
    print('demo digital I/O')
    led = Pin(5, Pin.OUT)    # create output pin on GPIO5
    button = Pin(0, Pin.IN)  # create output pin on GPIO0
    while 1:
        state = button.value()
        if state > 0:
            led.value(0)
        else:
            led.value(1)
```

Save these scripts.

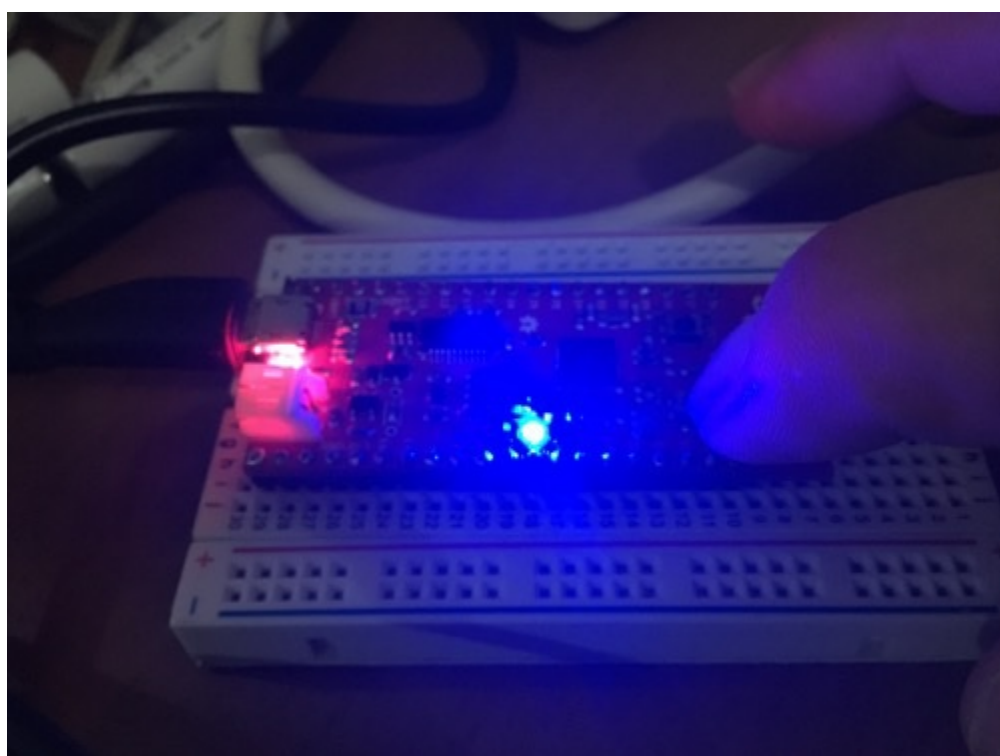
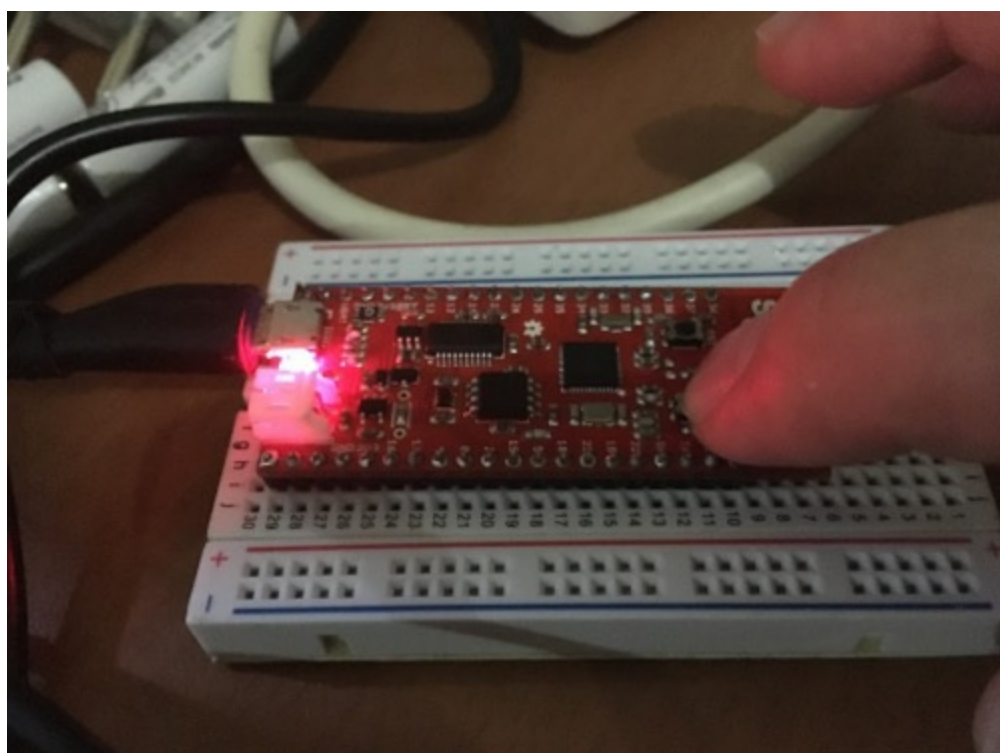
3.4 Testing

Now you can upload and run this program to MicroPython board via ampy. Now you can run it using MicroPython terminal. Type these command

```
>>> import ledbutton
>>> ledbutton.run()
```



For testing, try to press push button. You should see a lighting LED.



4. PWM and Analog Input

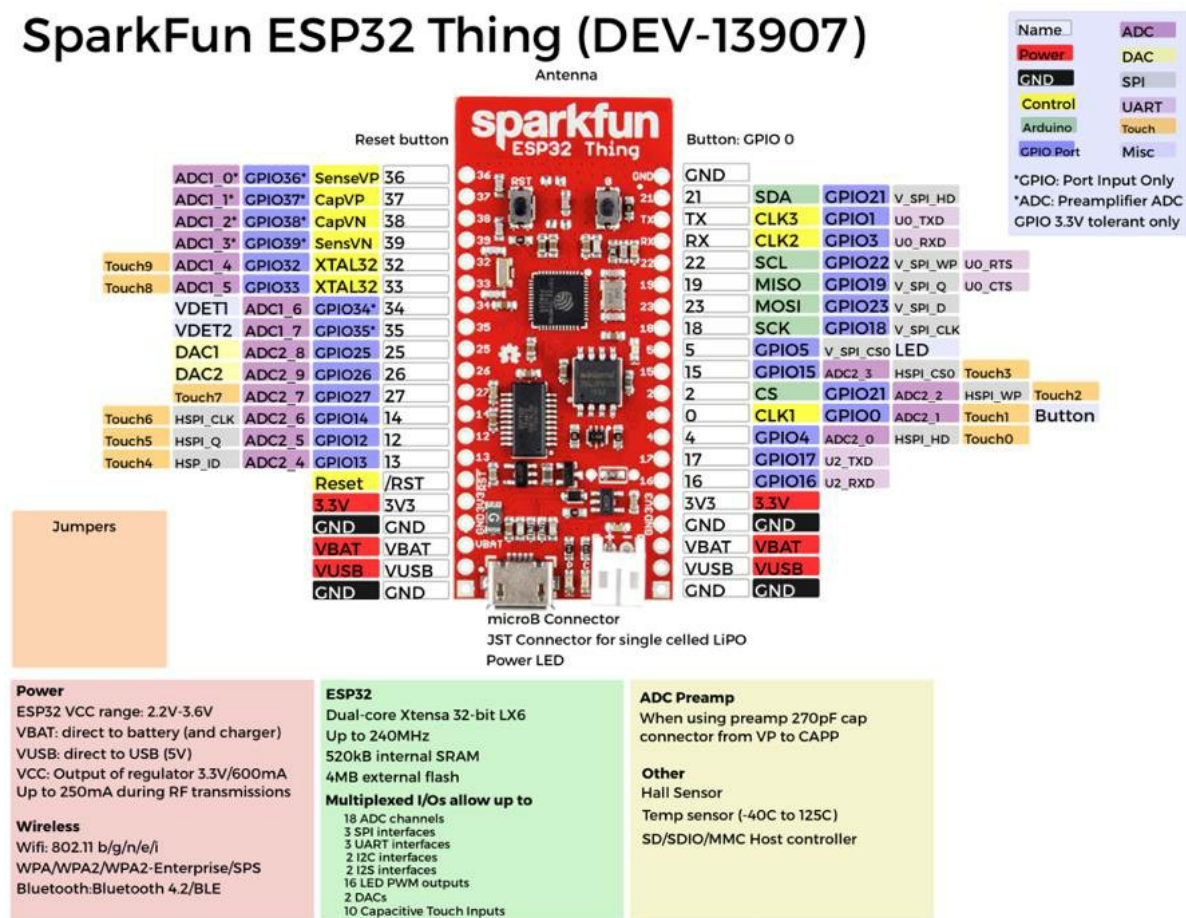
This chapter explains how to work with MicroPython board based ESP32 Analog I/O.

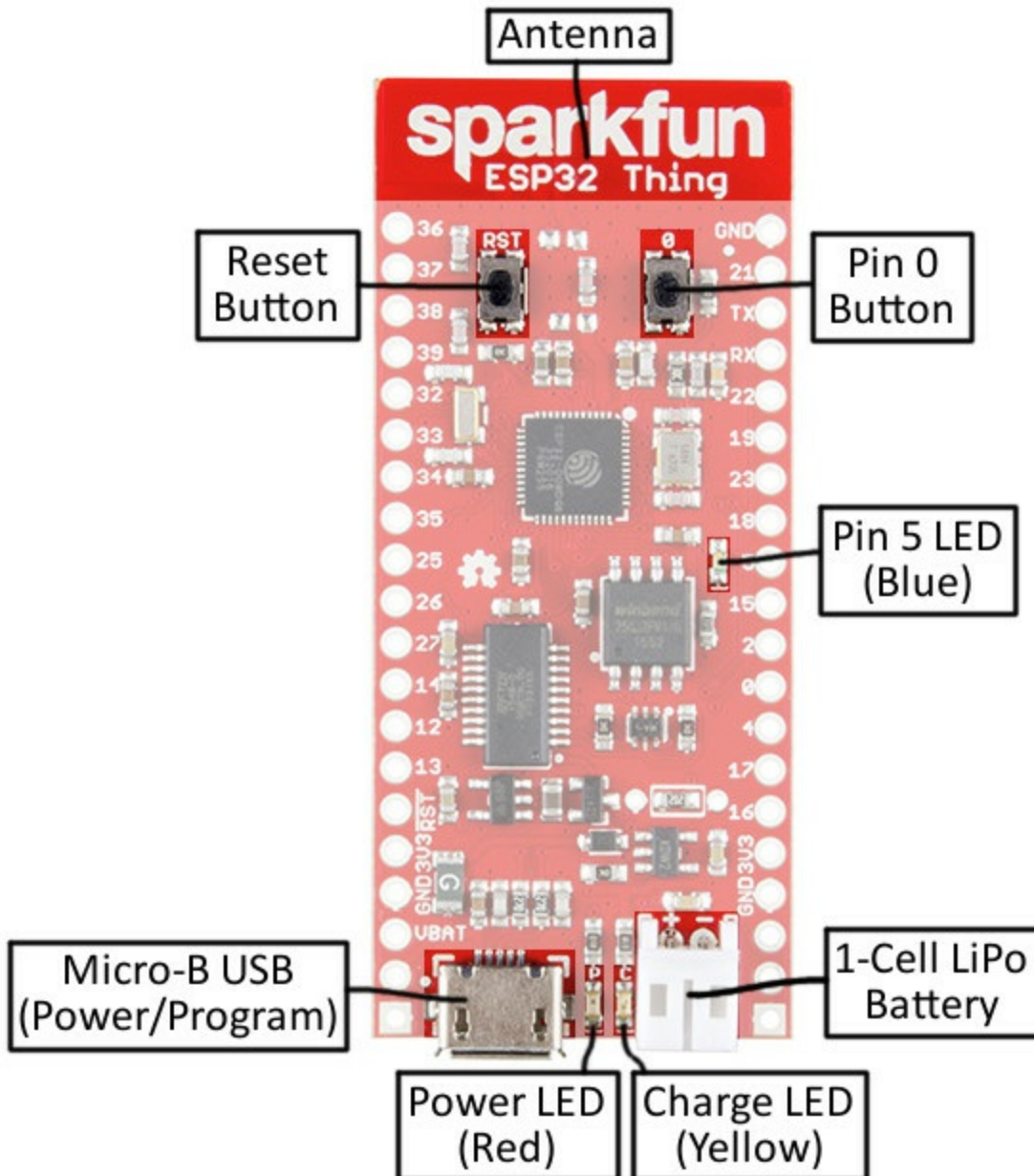
4.1 Getting Started

In this chapter, we learn how to work with PWM and Analog Input. For testing, I use NodeMCU board as MicroPython board. On the ESP32 board, we can use all GPIO pins for PWM.

You can see a sample of ESP32 board from SparkFun.

SparkFun ESP32 Thing (DEV-13907)





In this chapter, we try to access MicroPython Analog I/O using MicroPython program. There are two scenarios for our cases:

- Controlling RGB LED
- Reading Analog input using Potentiometer

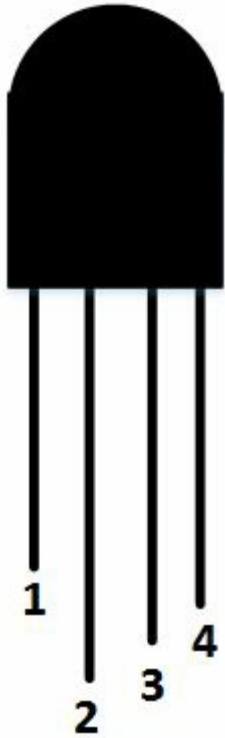
Let's start.

4.2 Demo Analog Output (PWM) : RGB LED

In this scenario we build a MicroPython program to control RGB LED color using MicroPython Analog output (PWM). RGB LED has 4 pins that you can see it on Figure below.



To understand these pins, you can see the following Figure.



Note:

- Pin 1: Red
- Pin 2: Common pin
- Pin 3: Green
- Pin 4: Blue

Now we can start to build a MicroPython application and hardware implementation.

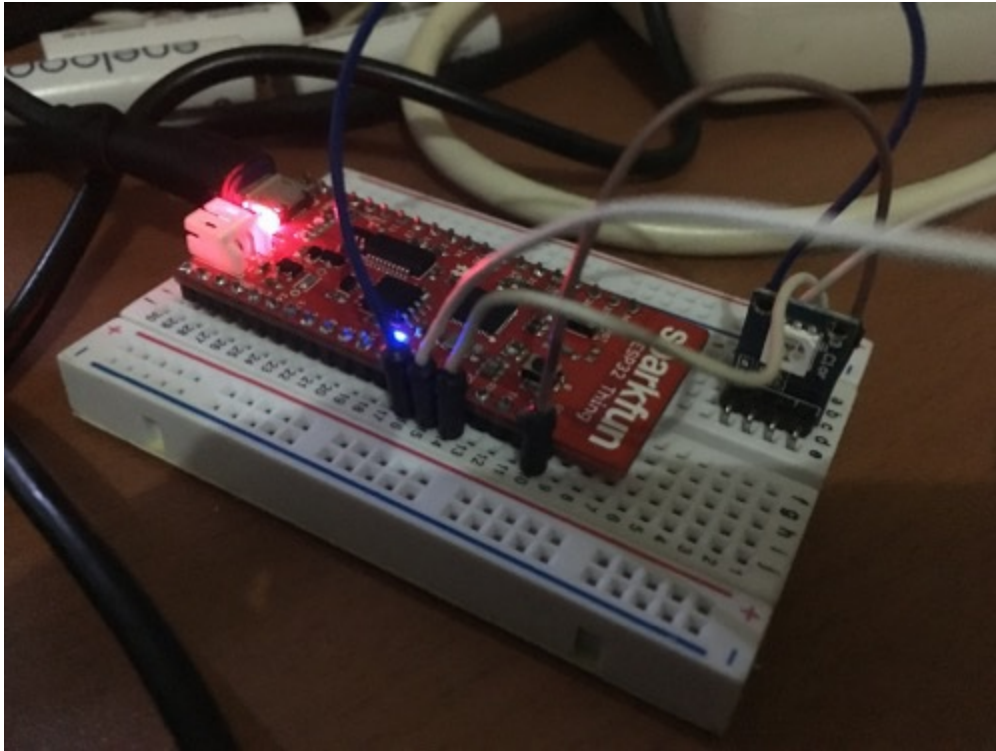
4.2.1 Wiring

For our testing, we configure the following PWM pins.

- RGB LED pin 1 (red) is connected to ESP32 GPIO22
- RGB LED pin 2 is connected to ESP32 3V3 (VCC +3.3V)
- RGB LED pin 3 (green) is connected to ESP32 GPIO19

- RGB LED pin 4 (blue) is connected to ESP32 GPIO23

Here is a sample implementation with SparkFun ESP32 Thing board and RGB Led.



4.2.2 Writing Program

To display a certain color, we must combine colors from red, green, blue. NodeMCU provides API for PWM which can set a value from 0 to 1023 using PWM library.

Let's start to build a program. Firstly, create a file, called **pwmdemo.py**. Then, write these scripts.

```
from machine import Pin, PWM
import time

gpio_red = 22
```

```

gpio_green = 19
gpio_blue = 23

def set_rgb(red, green, blue):
    pwm_red = PWM(Pin(gpio_red), freq=1000, duty=red)
    pwm_green = PWM(Pin(gpio_green), freq=1000, duty=green)
    pwm_blue = PWM(Pin(gpio_blue), freq=1000, duty=blue)

    time.sleep(2)
    pwm_red.deinit()
    pwm_green.deinit()
    pwm_blue.deinit()

def run():
    print('print PWM with RGB led')

    while 1:
        print('red')
        set_rgb(1023, 0, 0)
        print('green')
        set_rgb(0, 1023, 0)
        print('blue')
        set_rgb(0, 0, 1023)
        print('yellow')
        set_rgb(1023, 1023, 0)
        print('purple')
        set_rgb(323, 0, 323)
        print('aqua')
        set_rgb(0, 1023, 1023)

```

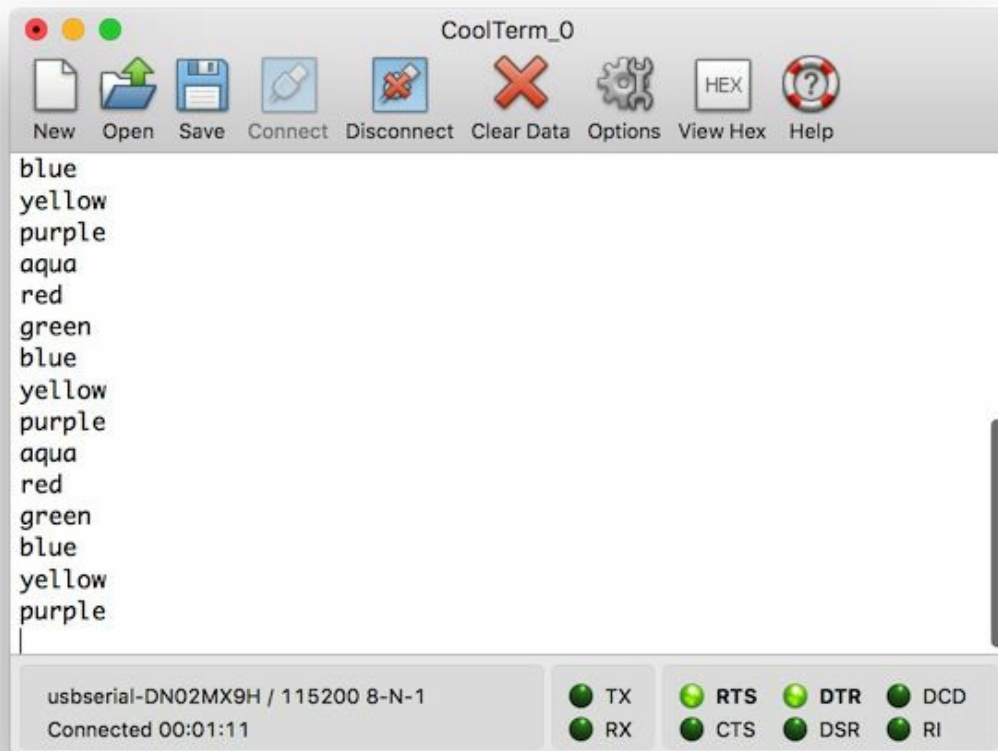
This program will generate six colors: red, green, blue, yellow, purple, and aqua.

Save this file.

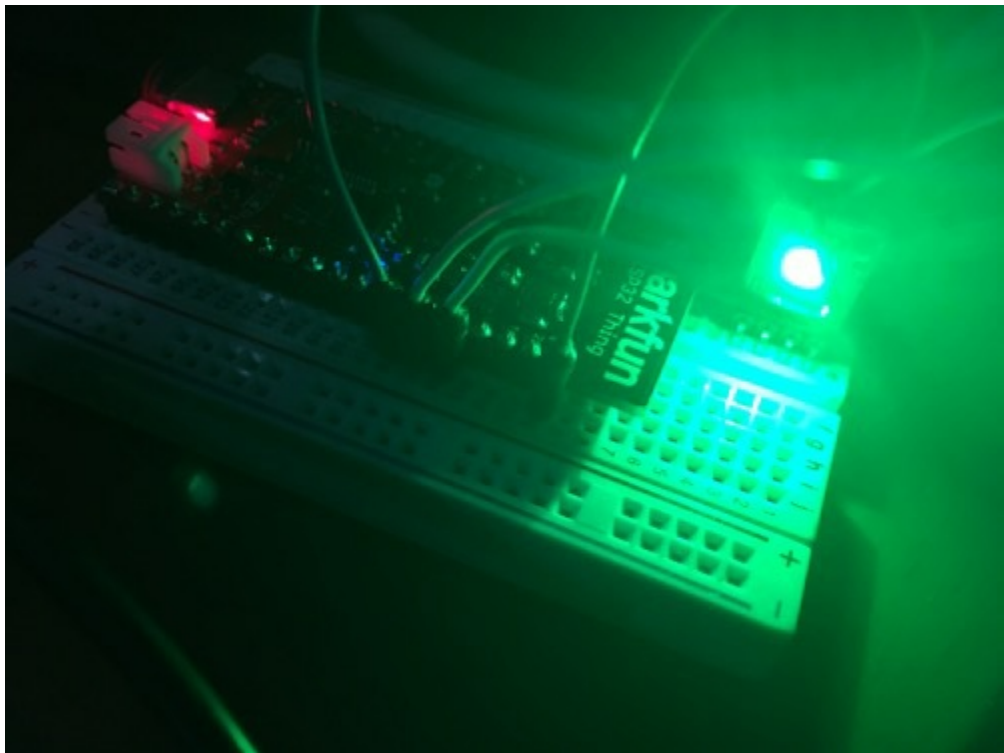
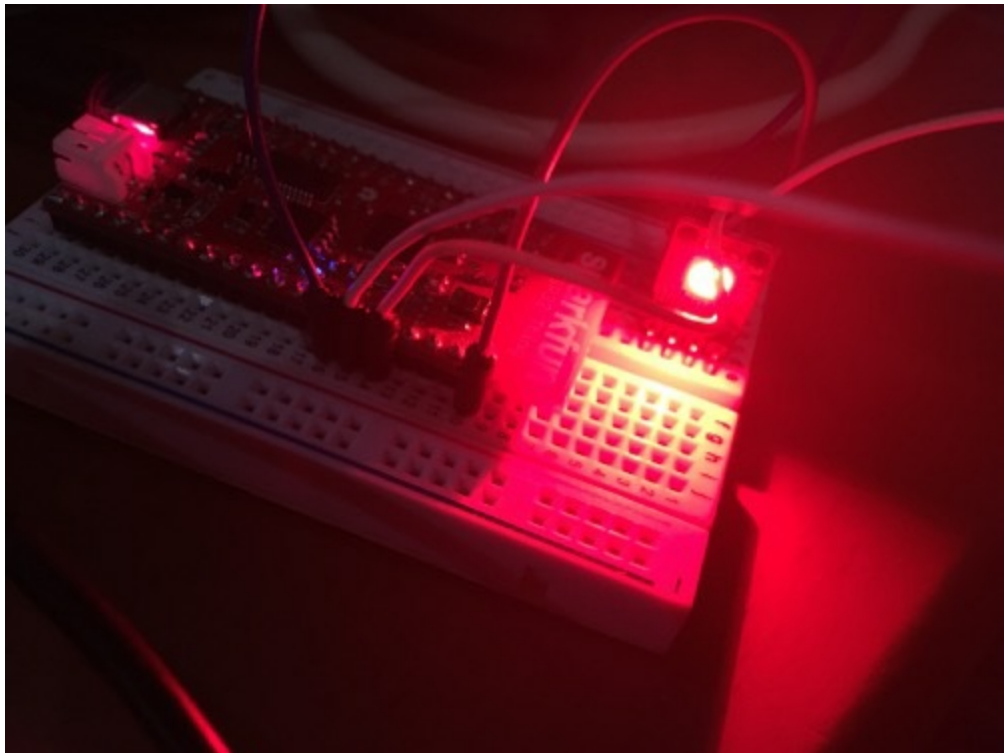
4.2.3 Testing

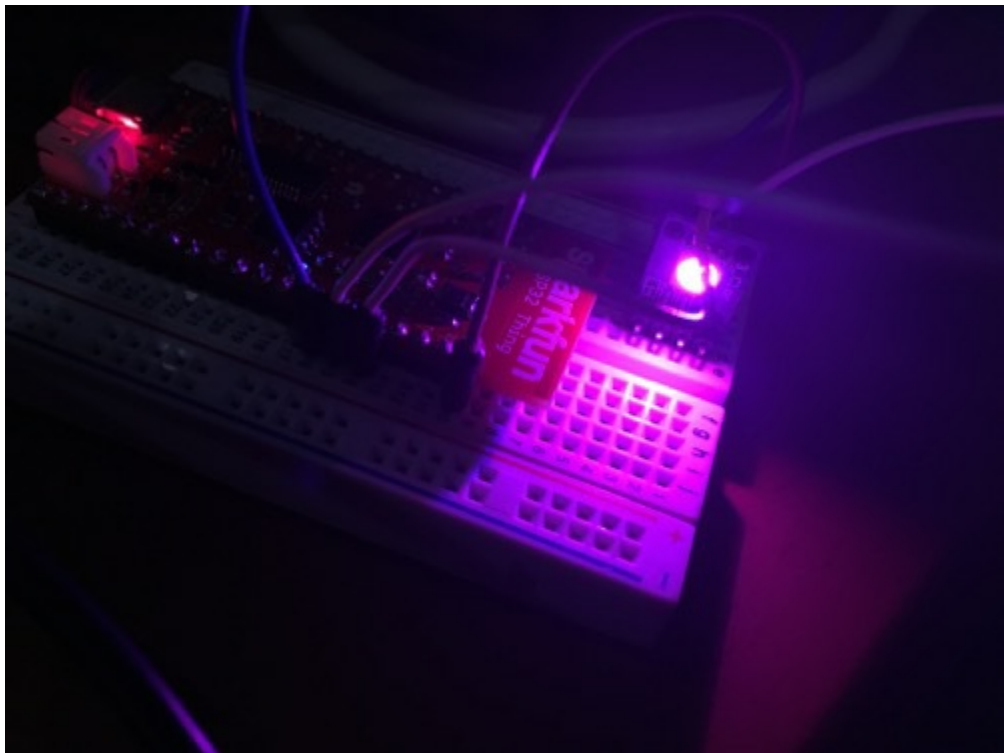
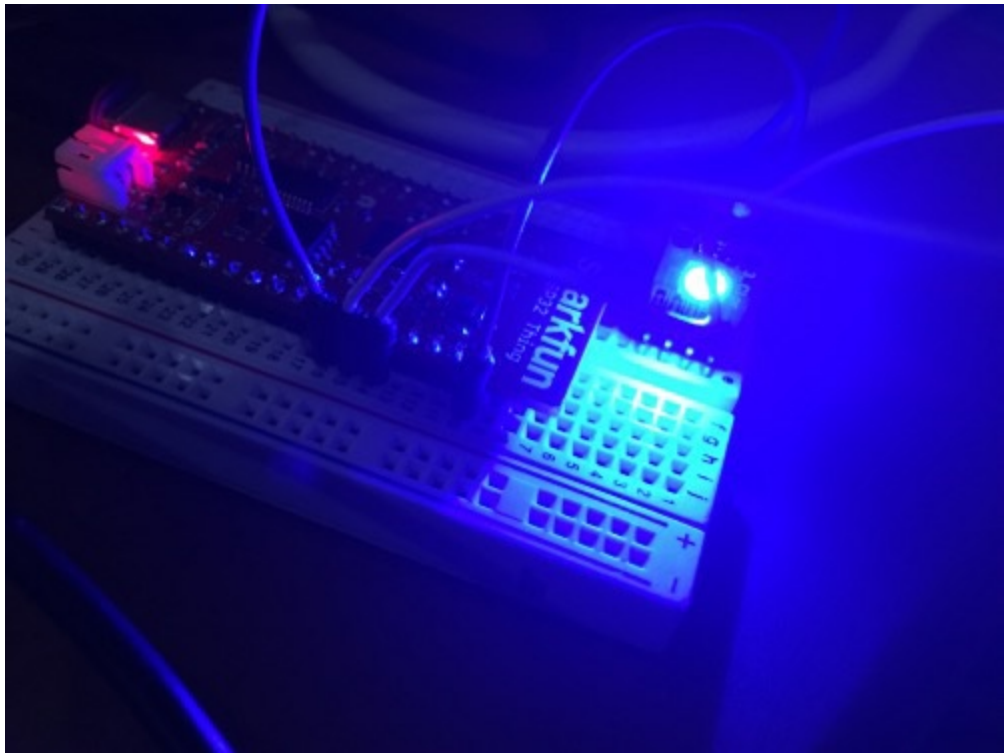
Upload and run the program. Then, run the program as follows.

```
>>> import pwmdemo  
>>> pwmdemo.run()
```



You should see several color on RGB LED. The following is a sample demo on RGB LED.



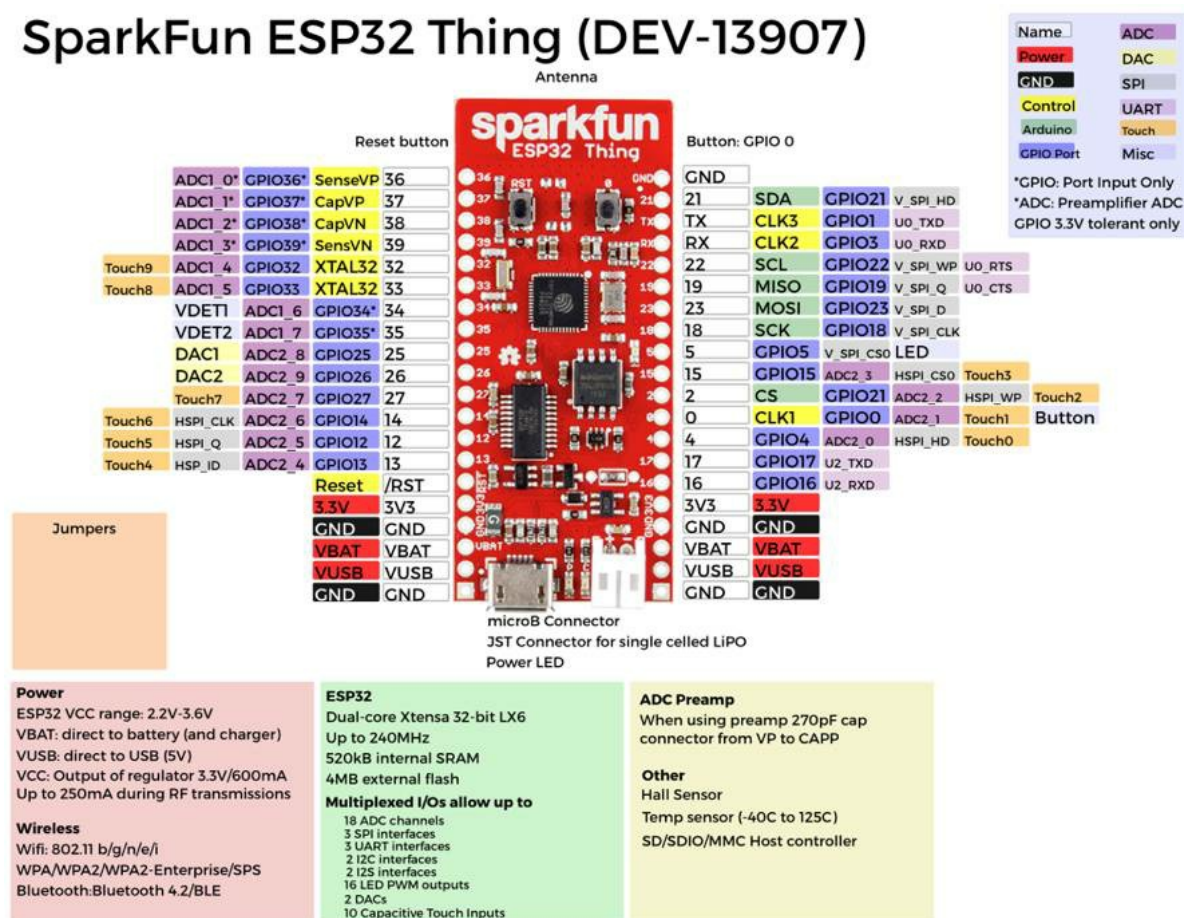


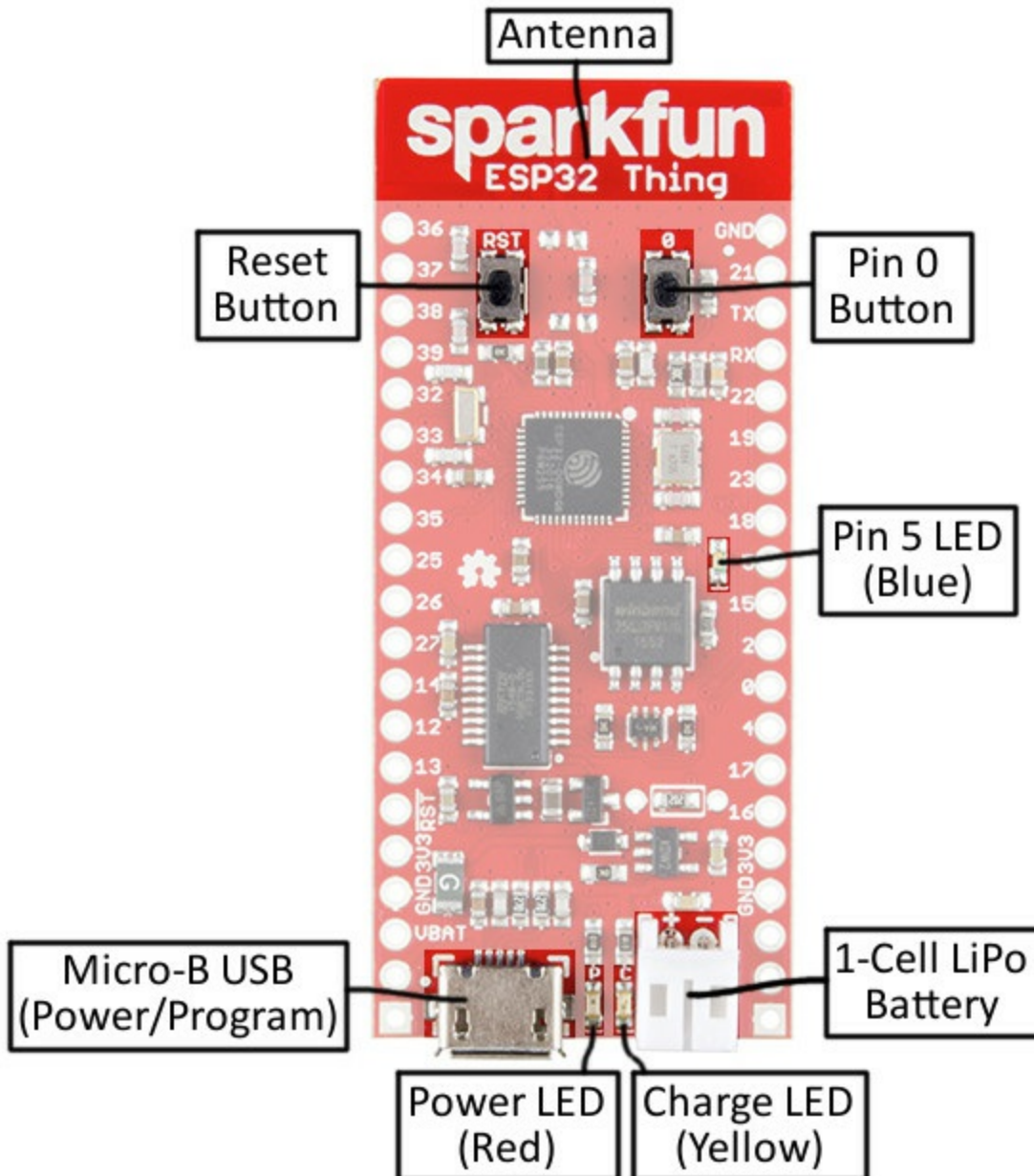
4.3 Demo Analog Input: Working with Potentiometer

In this section, we learn how to read analog input on MicroPython board. For illustration, I use Potentiometer as analog input source. Our scenario is to read analog value from Potentiometer. Then, display it on Lua shell.

ESP32 only has several ADC pins. For instance, you can see ADC pins on SparkFun ESP32 Thing board as below.

SparkFun ESP32 Thing (DEV-13907)

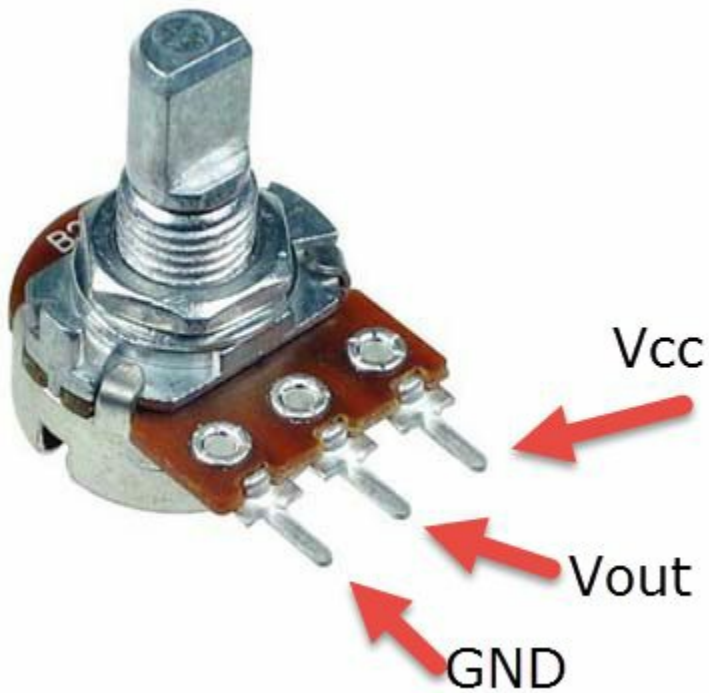




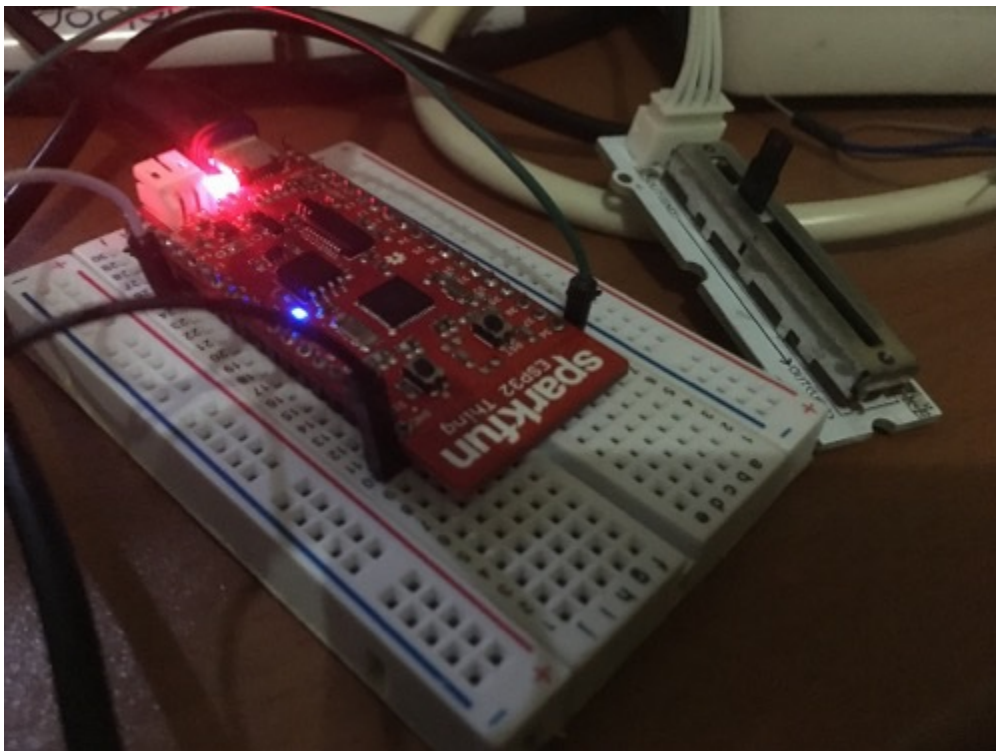
Let's start!.

4.3.1 Wiring

To understand Potentiometer, you see its scheme in Figure below.



You can connect VCC to ESP32 board on 3V3 pin (VCC +3.3V). Vout to ESP32 board Analog input on GPIO36 (ADC1_0). In addition, GND to ESP32 board GND. The following is hardware implementation. I use slide potentiometer.



4.3.2 Writing Program

Firstly, create a file, called **adcdemo.py**. To read analog input, we can use `adc.read()` function. Ok, Let's write these scripts.

```
from machine import Pin, ADC
import time

gpio_adc = 36

def run():
    print('ADC demo')

    while 1:
        adc = ADC(Pin(gpio_adc))
        print('ADC: ' + str(adc.read()))
        time.sleep(2)
```

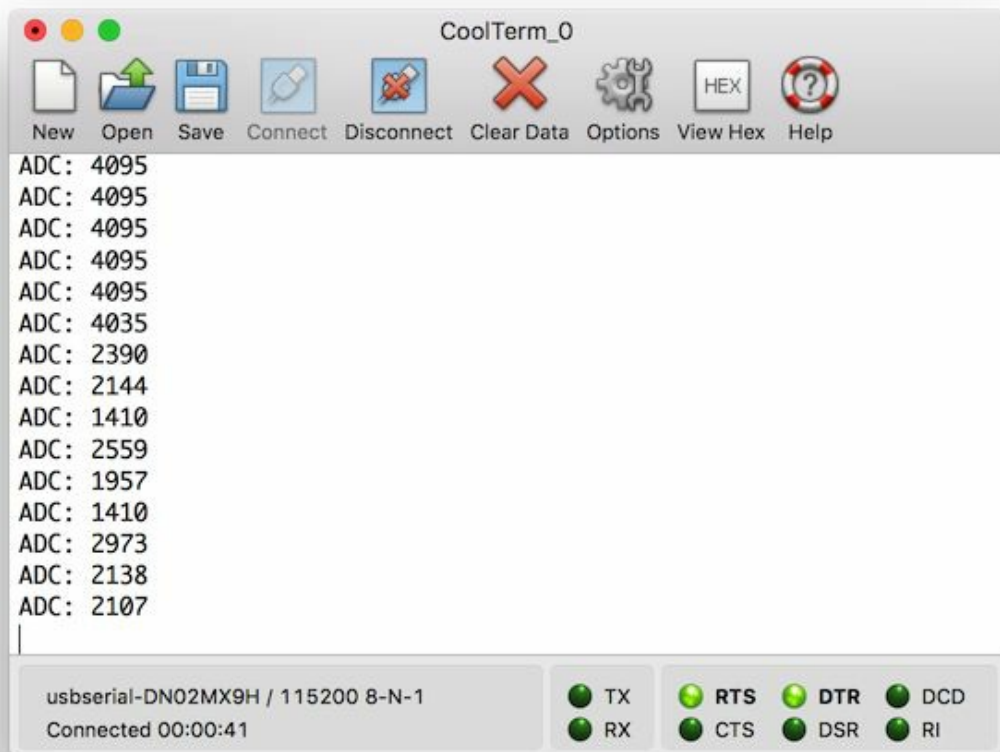
Save this code.

4.3.3 Testing

Upload and run this program. If succeed, you can run the program.

```
>>> import adcdemo
>>> adcdemo.run()
```

You should see the output on MicroPython terminal. Please change values on potentiometer



5. Working with I2C

In this chapter we learn how to work with I2C on MicroPython board.

5.1 Getting Started

The I2C (Inter-Integrated Circuit) bus was designed by Philips in the early '80s to allow easy communication between components which reside on the same circuit board. TWI stands for Two Wire Interface and for most parts this bus is identical to I²C. The name TWI was introduced by Atmel and other companies to avoid conflicts with trademark issues related to I²C.

I2C bus consists of two wires, SDA (Serial Data Line) and SCL (Serial Clock Line). MicroPython supports all pins for I2C software.

SparkFun ESP32 Thing (DEV-13907)

Antenna

Reset button

Button: GPIO 0

microB Connector

JST Connector for single celled LiPO

Power LED

Jumpers

Power
ESP32 VCC range: 2.2V-3.6V
VBAT: direct to battery (and charger)
VUSB: direct to USB (5V)
VCC: Output of regulator 3.3V/600mA
Up to 250mA during RF transmissions

Wireless
Wifi: 802.11 b/g/n/e/i
WPA/WPA2-Enterprise/SPS
Bluetooth:Bluetooth 4.2/BLE

ESP32
Dual-core Xtensa 32-bit LX6
Up to 240MHz
520kB internal SRAM
4MB external flash
Multiplexed I/Os allow up to
18 ADC channels
3 SPI interfaces
3 UART interfaces
2 I2C interfaces
2 I2S interfaces
16 LED PWM outputs
2 DACs
10 Capacitive Touch Inputs

ADC Preamp
When using preamp 270pF cap
connector from VP to CAPP

Other
Hall Sensor
Temp sensor (-40C to 125C)
SD/SDIO/MMC Host controller

Name

ADC

Power

GND

Control

Arduino

GPIO Port

DAC

SPI

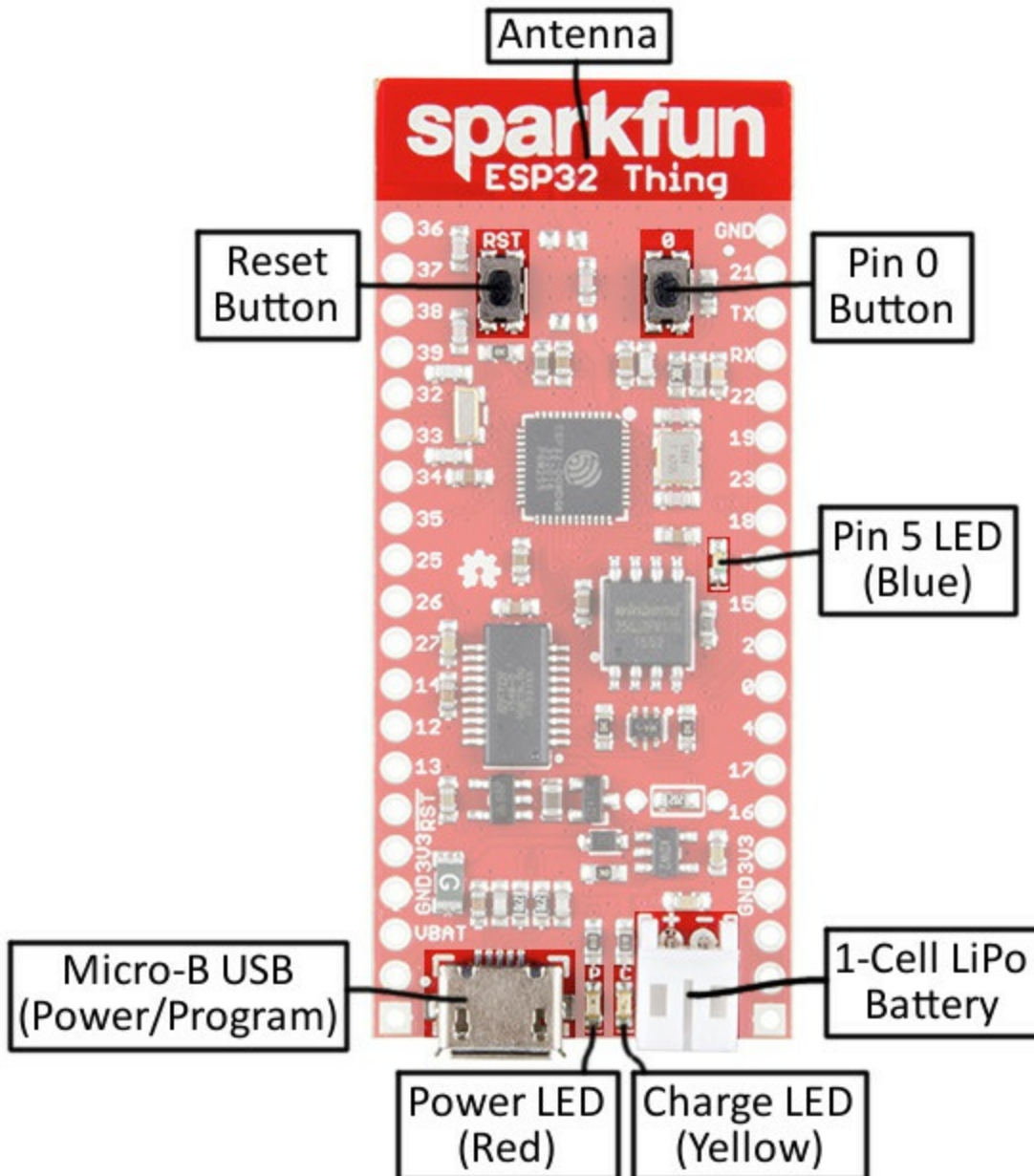
UART

Touch

Misc

*GPIO: Port Input Only
*ADC: Preamplifier ADC
GPIO 3.3V tolerant only

Pin	Function	Pin	Function
1	GND	21	SDA
2	CS	22	SCL
3	CLK1	23	MOSI
4	GPIO4	24	GND
5	GPIO5	25	3.3V
6	GPIO6	26	GND
7	GPIO7	27	VBAT
8	GPIO8	28	VUSB
9	GPIO9	29	GND
10	GPIO10	30	TX
11	GPIO11	31	RX
12	GPIO12	32	GND
13	GPIO13	33	TX
14	GPIO14	34	GND
15	GPIO15	35	TX
16	GPIO16	36	SenseVP
17	GPIO17	37	CapVP
18	GPIO18	38	CapVN
19	GPIO19	39	SensVN
20	GPIO20	40	XTAL32
21	GPIO21	41	XTAL32
22	GPIO22	42	XTAL32
23	GPIO23	43	XTAL32
24	GPIO24	44	XTAL32
25	GPIO25	45	XTAL32
26	GPIO26	46	XTAL32
27	GPIO27	47	XTAL32
28	GPIO28	48	XTAL32
29	GPIO29	49	XTAL32
30	GPIO30	50	XTAL32
31	GPIO31	51	XTAL32
32	GPIO32	52	XTAL32
33	GPIO33	53	XTAL32
34	GPIO34	54	XTAL32
35	GPIO35	55	XTAL32
36	GPIO36	56	XTAL32
37	GPIO37	57	XTAL32
38	GPIO38	58	XTAL32
39	GPIO39	59	XTAL32
40	GPIO40	60	XTAL32
41	GPIO41	61	XTAL32
42	GPIO42	62	XTAL32
43	GPIO43	63	XTAL32
44	GPIO44	64	XTAL32
45	GPIO45	65	XTAL32
46	GPIO46	66	XTAL32
47	GPIO47	67	XTAL32
48	GPIO48	68	XTAL32
49	GPIO49	69	XTAL32
50	GPIO50	70	XTAL32
51	GPIO51	71	XTAL32
52	GPIO52	72	XTAL32
53	GPIO53	73	XTAL32
54	GPIO54	74	XTAL32
55	GPIO55	75	XTAL32
56	GPIO56	76	XTAL32
57	GPIO57	77	XTAL32
58	GPIO58	78	XTAL32
59	GPIO59	79	XTAL32
60	GPIO60	80	XTAL32
61	GPIO61	81	XTAL32
62	GPIO62	82	XTAL32
63	GPIO63	83	XTAL32
64	GPIO64	84	XTAL32
65	GPIO65	85	XTAL32
66	GPIO66	86	XTAL32
67	GPIO67	87	XTAL32
68	GPIO68	88	XTAL32
69	GPIO69	89	XTAL32
70	GPIO70	90	XTAL32
71	GPIO71	91	XTAL32
72	GPIO72	92	XTAL32
73	GPIO73	93	XTAL32
74	GPIO74	94	XTAL32
75	GPIO75	95	XTAL32
76	GPIO76	96	XTAL32
77	GPIO77	97	XTAL32
78	GPIO78	98	XTAL32
79	GPIO79	99	XTAL32
80	GPIO80	100	XTAL32
81	GPIO81	101	XTAL32
82	GPIO82	102	XTAL32
83	GPIO83	103	XTAL32
84	GPIO84	104	XTAL32
85	GPIO85	105	XTAL32
86	GPIO86	106	XTAL32
87	GPIO87	107	XTAL32
88	GPIO88	108	XTAL32
89	GPIO89	109	XTAL32
90	GPIO90	110	XTAL32
91	GPIO91	111	XTAL32
92	GPIO92	112	XTAL32
93	GPIO93	113	XTAL32
94	GPIO94	114	XTAL32
95	GPIO95	115	XTAL32
96	GPIO96	116	XTAL32
97	GPIO97	117	XTAL32
98	GPIO98	118	XTAL32
99	GPIO99	119	XTAL32
100	GPIO100	120	XTAL32



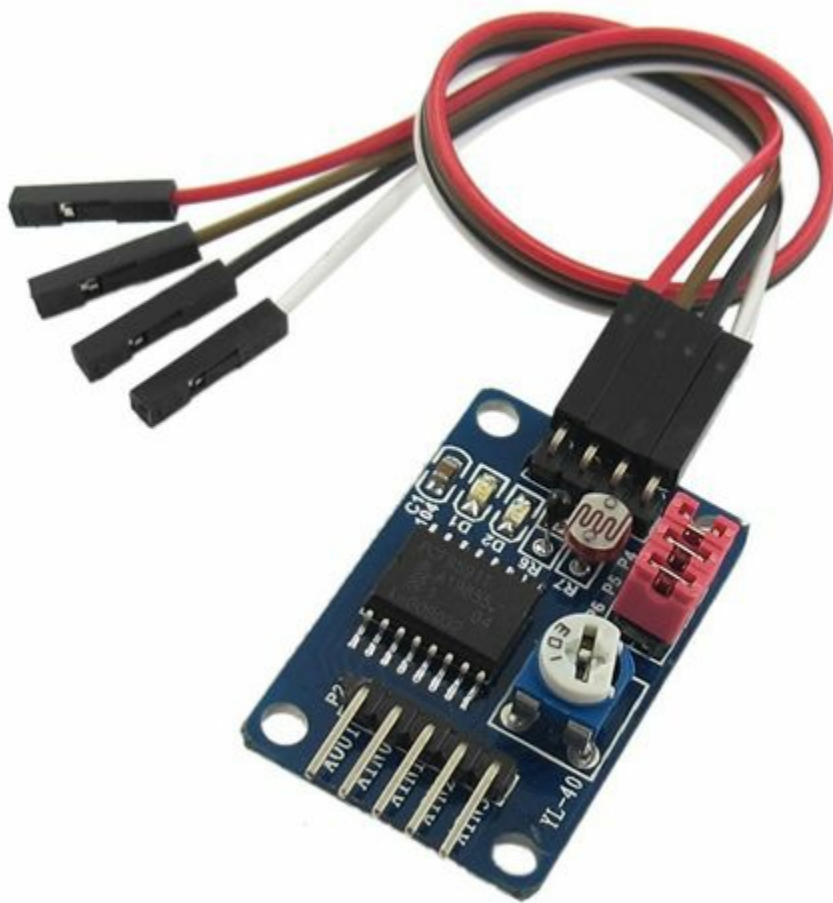
For testing, I used PCF8591 AD/DA Converter module with sensor and actuator devices. You can find it on the following online store:

- Amazon, <http://www.amazon.com/PCF8591-Converter-Module-Digital-Conversion/dp/B00BXX4UWC/>
- eBay, <http://www.ebay.com>
- Dealextreme, <http://www.dx.com/p/pcf8591-ad-da-analog-to-digital->

[digital-to-analog-converter-module-w-dupont-cable-deep-blue-336384](#)

- Aliexpress, <http://www.aliexpress.com/>

In addition, you can find this device on your local electronics store/online store.



This module has mini form model too, for instance, you can find it on Amazon, <http://www.amazon.com/WaveShare-PCF8591T-Converter-Evaluation-Development/dp/B00KM6X2OI/> .



This module use PCF8591 IC and you can read the datasheet on the following URLs.

- <http://www.electrodragon.com/w/images/e/ed/PCF8591.pdf>
- http://www.nxp.com/documents/data_sheet/PCF8591.pdf

For testing I2C on MicroPython, I use PCF8591 AD/DA Converter module and ESP32 module.

Let's start.

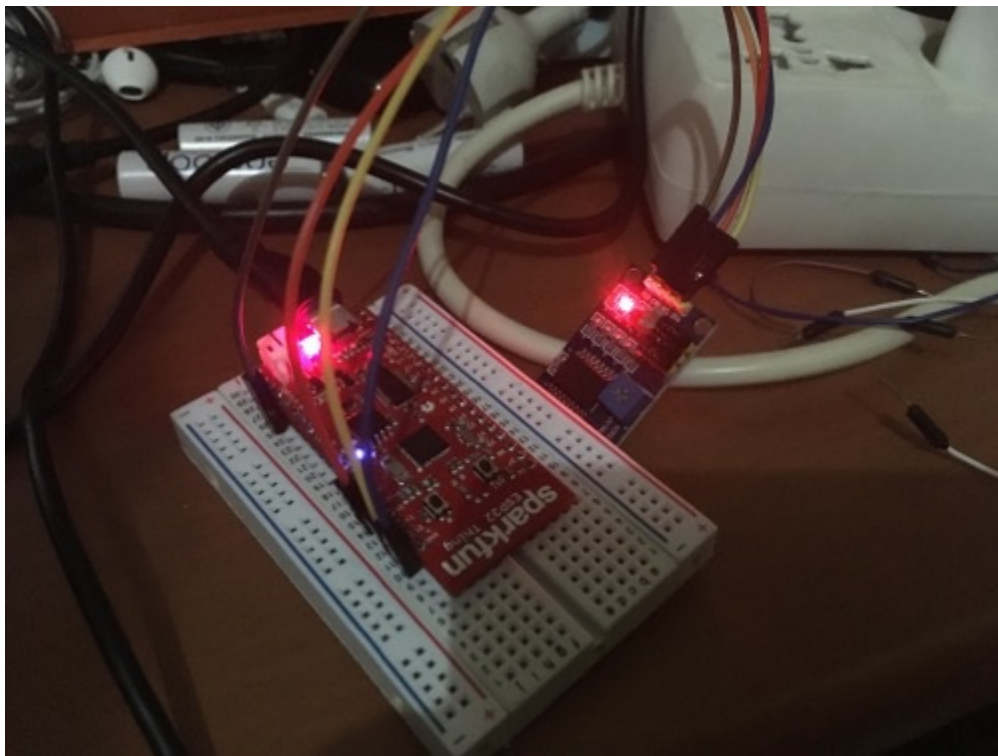
5.2 Writing Program

We connect PCF8591 AD/DA Converter module to ESP32 board directly.

The following is our wiring lab:

- PCF8591 AD/DA Converter module SDA --> ESP32 SDA (GPIO21)
- PCF8591 AD/DA Converter module SCL--> ESP32 SCL (GPIO22)
- PCF8591 AD/DA Converter module VCC --> ESP32 VCC (+3.3V)
- PCF8591 AD/DA Converter module GND --> ESP32 GND

Hardware implementation can be shown in Figure below.



5.3 Writing Program

Now you can start to write a MicroPython program for ESP32 board. Create a file, called **i2cdemo.py** and write these scripts.

```
from machine import Pin, I2C
import time

def run():
    print('read sensor from i2c protocol')
    PCF8591 = 0x48          # I2C bus address
    PCF8591_ADC_CH0 = '\x00' # thermistor
    PCF8591_ADC_CH1 = '\x01' # photo-voltaic cell
    PCF8591_ADC_CH3 = '\x03' # potentiometer

    # construct an I2C bus
    gpio_scl = Pin(22)
    gpio_sda = Pin(21)
    i2c = I2C(scl=gpio_scl, sda=gpio_sda, freq=100000)

    while 1:
        # read thermistor
        i2c.writeto(PCF8591, PCF8591_ADC_CH0)
        i2c.readfrom(PCF8591, 1)
        data = i2c.readfrom(PCF8591, 1)
        print('Thermistor: ' + str(ord(chr(data[0]))))

        # photo-voltaic cell
        i2c.writeto(PCF8591, PCF8591_ADC_CH1)
        i2c.readfrom(PCF8591, 1)
        data = i2c.readfrom(PCF8591, 1)
        print('photo-voltaic: ' + str(ord(chr(data[0]))))

        # potentiometer
        i2c.writeto(PCF8591, PCF8591_ADC_CH3)
        i2c.readfrom(PCF8591, 1)
        data = i2c.readfrom(PCF8591, 1)
        print('potentiometer: ' + str(ord(chr(data[0]))))

        time.sleep(2)
```

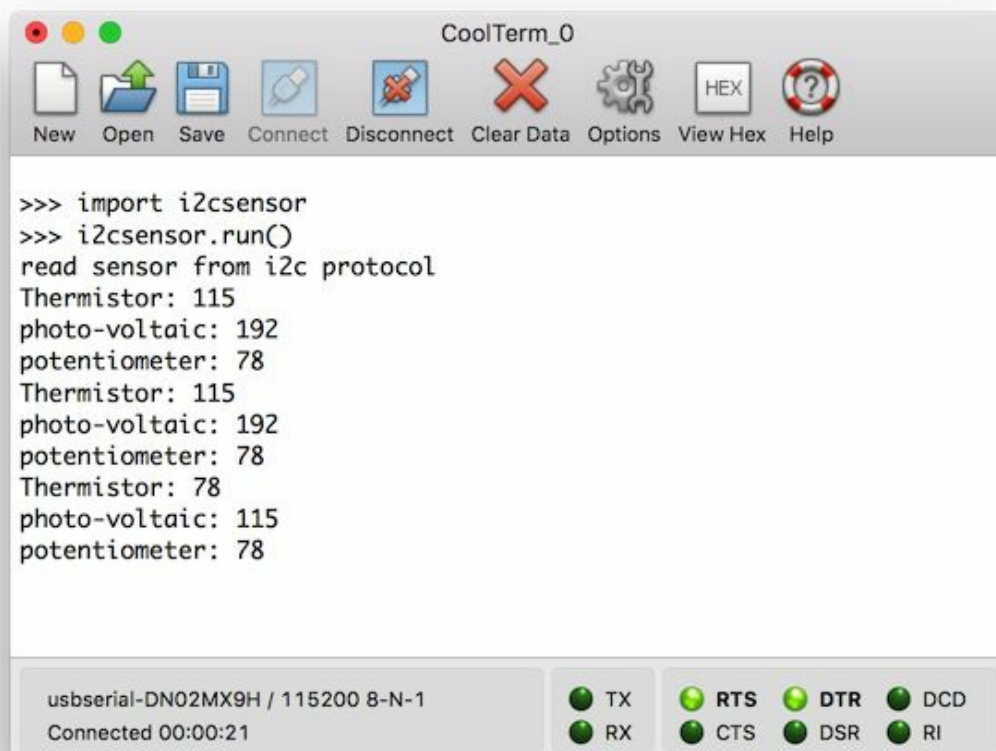

Save this code.

5.4 Testing

Now you can upload and run the MicroPython program to ESP32 board. You can run this command in MicroPython terminal.

```
>>> import i2csensor
>>> i2csensor.run()
```

If success, you should see the program output on MicroPython terminal. The following is a sample output.



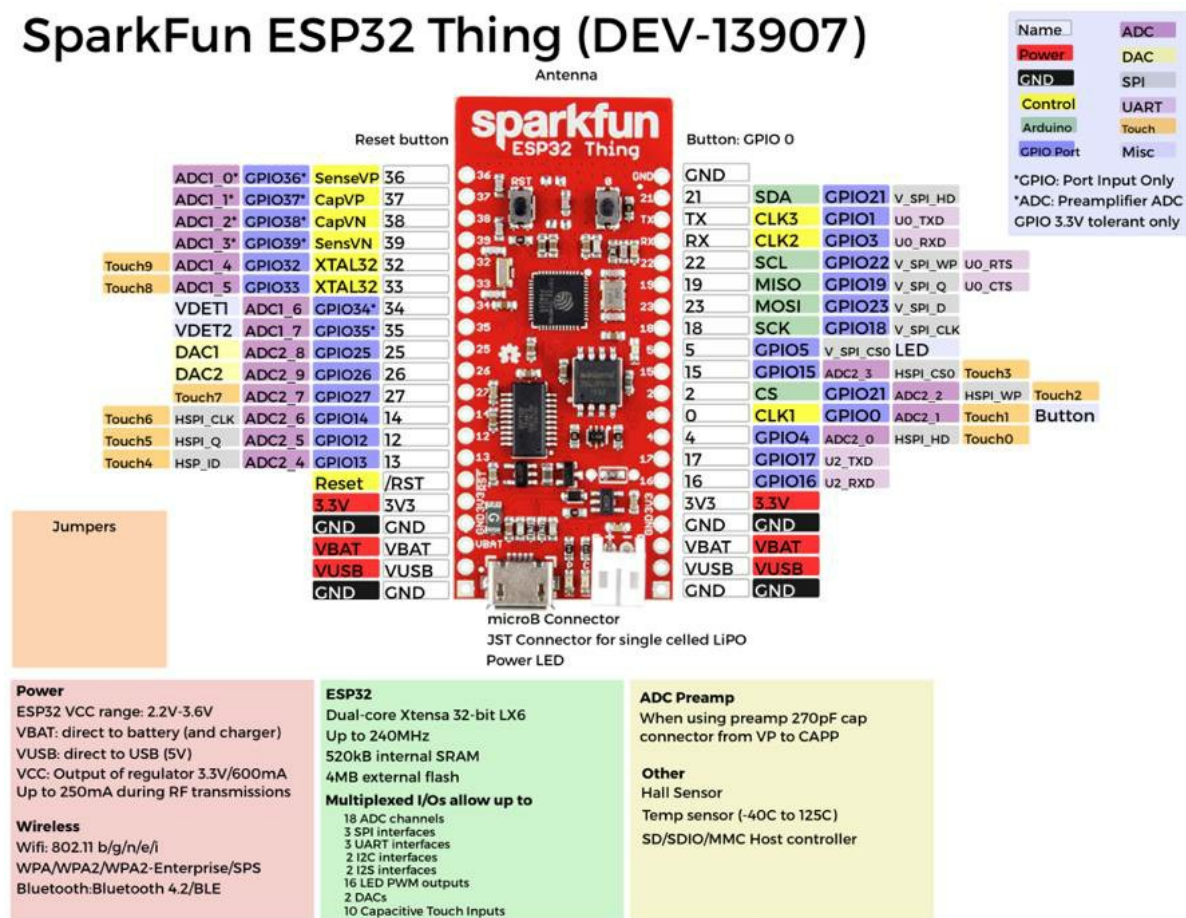
6. Working with UART

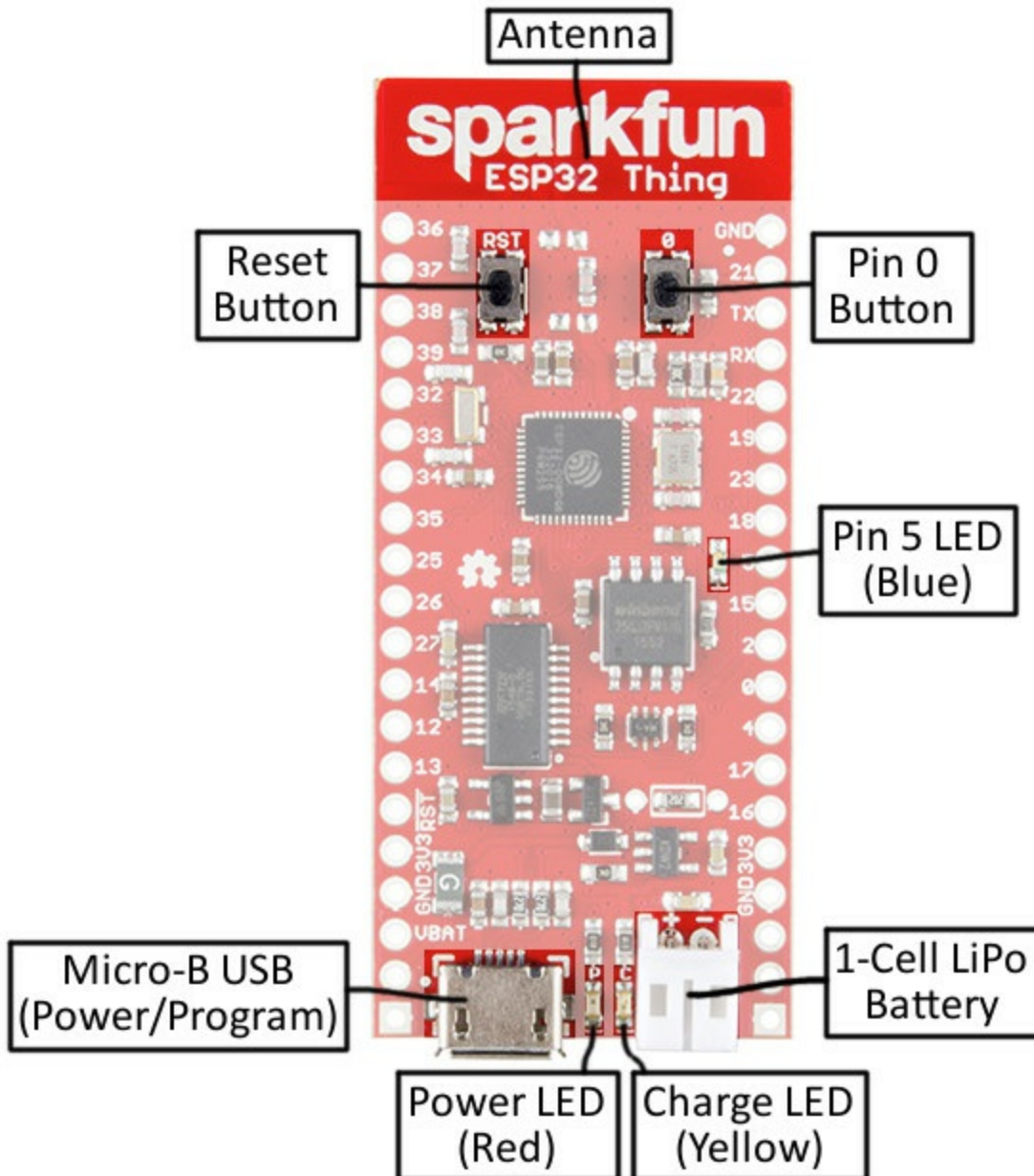
In this chapter I'm going to explain how to access UART on MicroPython board.

6.1 Getting Started

ESP32 chip provides three UARTs. You can see them on your ESP32 board. For instance, you can see UART pins on SparkFun ESP32 Thing board.

SparkFun ESP32 Thing (DEV-13907)





We can access UART using UART library.

In this chapter, I use Arduino board as UART source. We read incoming message from UART.

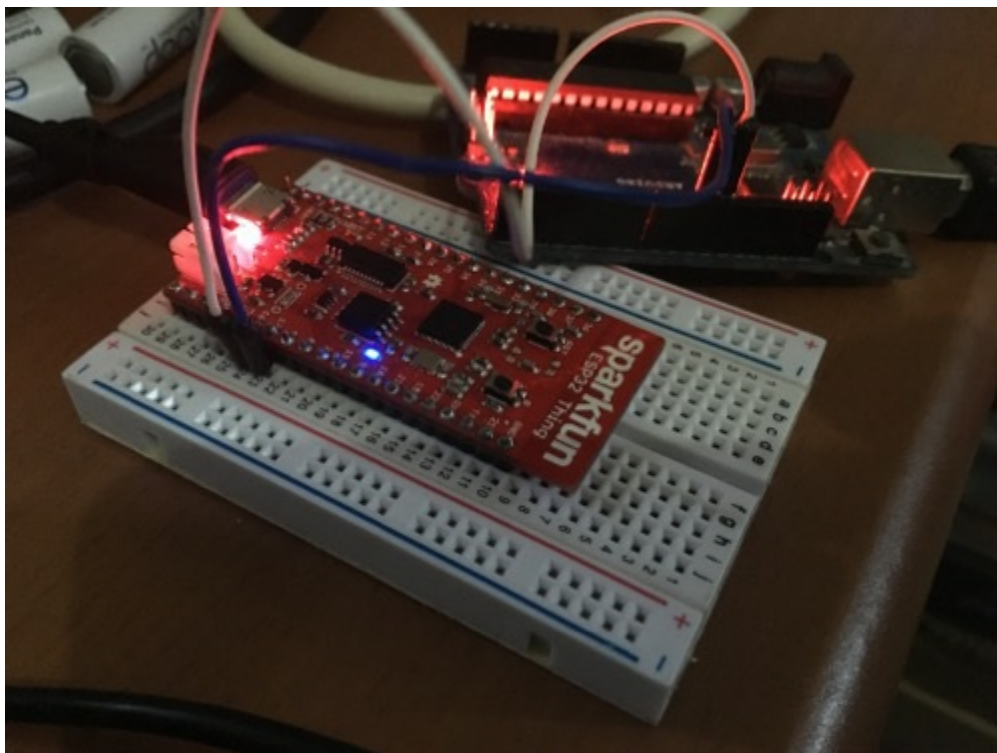
Let's start!.

6.2 Wiring

In this scenario, I use Arduino Uno which is connected to ESP32 board. Since my ESP32 board, SparkFun ESP32 Thing uses UART0 for USB so we connect Arduino to ESP32 board via UART2. We should connect RX pin to TX pin and TX pin to RX pin. The following is our wiring.

- ESP32 GPIO17 (U2_TXD) is connected to Arduino Digital 10 (RX)
- ESP32 GPIO16 (U2_RXD) is connected to Arduino Digital 11 (TX)
- ESP32 GND is connected to Arduino GND (optional)

My wiring implementation can be seen in Figure below.



6.3 Writing a Program

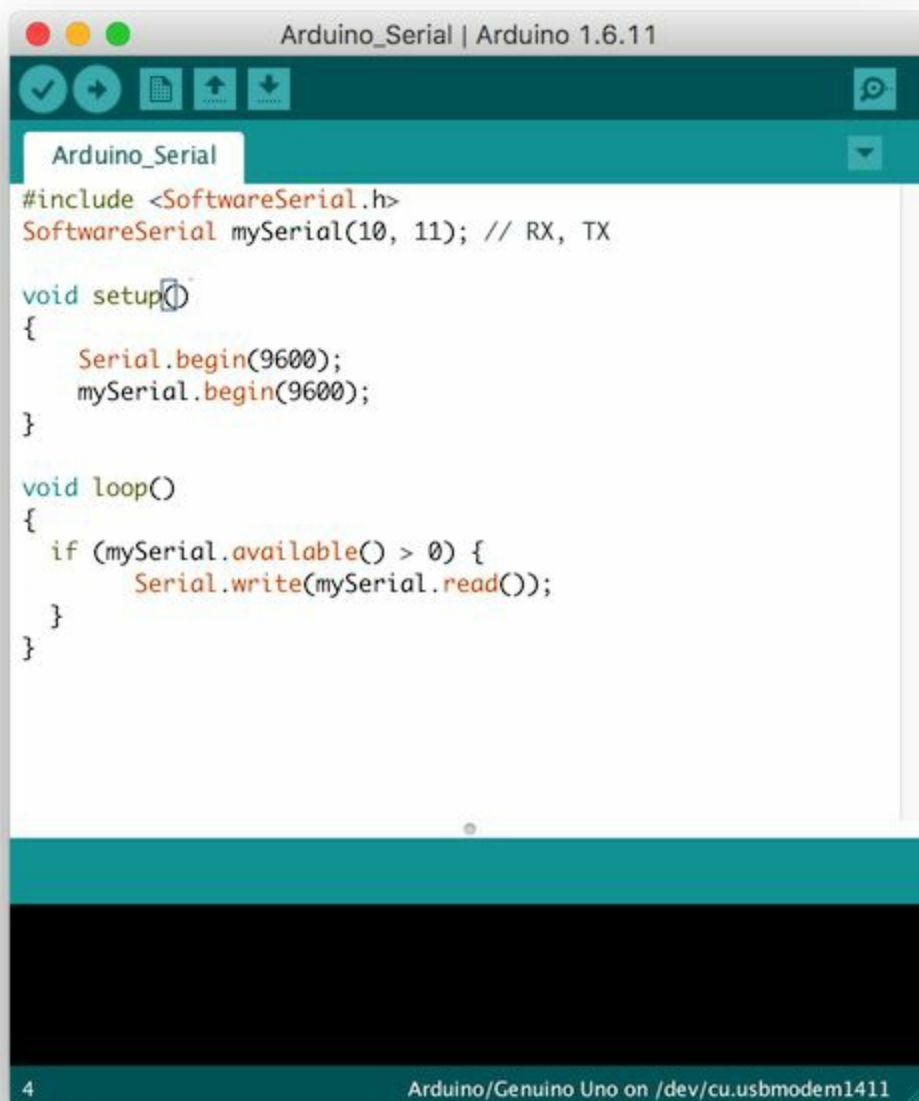
Firstly, we write a program for Arduino using Arduino IDE. We use SoftwareSerial to access Serial on Digital 10 and 11. This program will wait incoming UART data and then send to Arduino UART on 0 and 1 pins.

Write this program.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX

void setup()
{
    Serial.begin(9600);
    mySerial.begin(9600);
}

void loop()
{
    if (mySerial.available() > 0) {
        Serial.write(mySerial.read());
    }
}
```



Save this program. Then, upload it to Arduino board. Before uploading, please make sure Arduino UART (digital 0, 1, 10, and 11 pins) doesn't connect to any board.

The next step is to write a program for ESP32 board. Create a file, called **uartdemo.py**. Write these scripts.

```
from machine import UART
import time
```

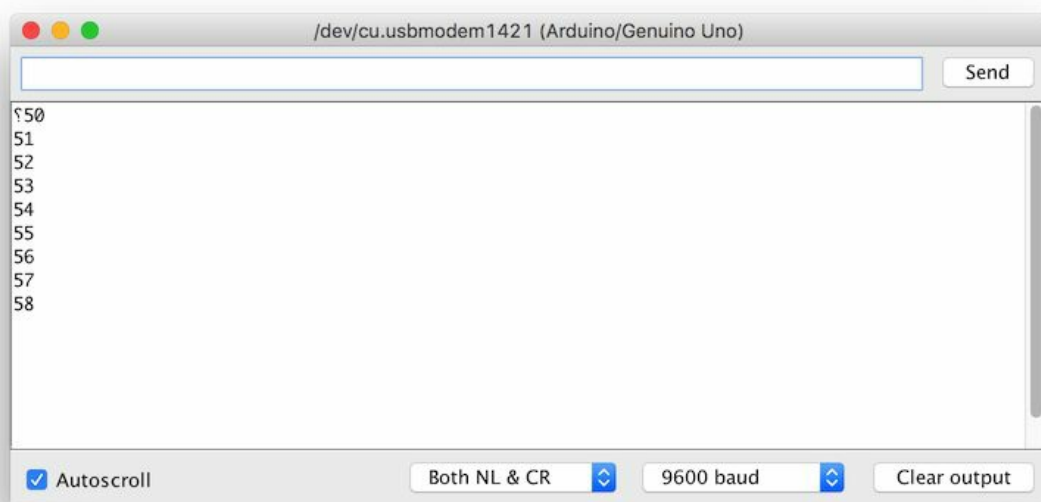
```
def run():  
    print('demo UART')  
  
    uart = UART(2, baudrate=9600)  
    counter = 50  
    while 1:  
        uart.write(str(counter) + '\r\n')  
        time.sleep(2)  
        counter += 1  
        if counter > 70:  
            counter = 50
```

Save this file.

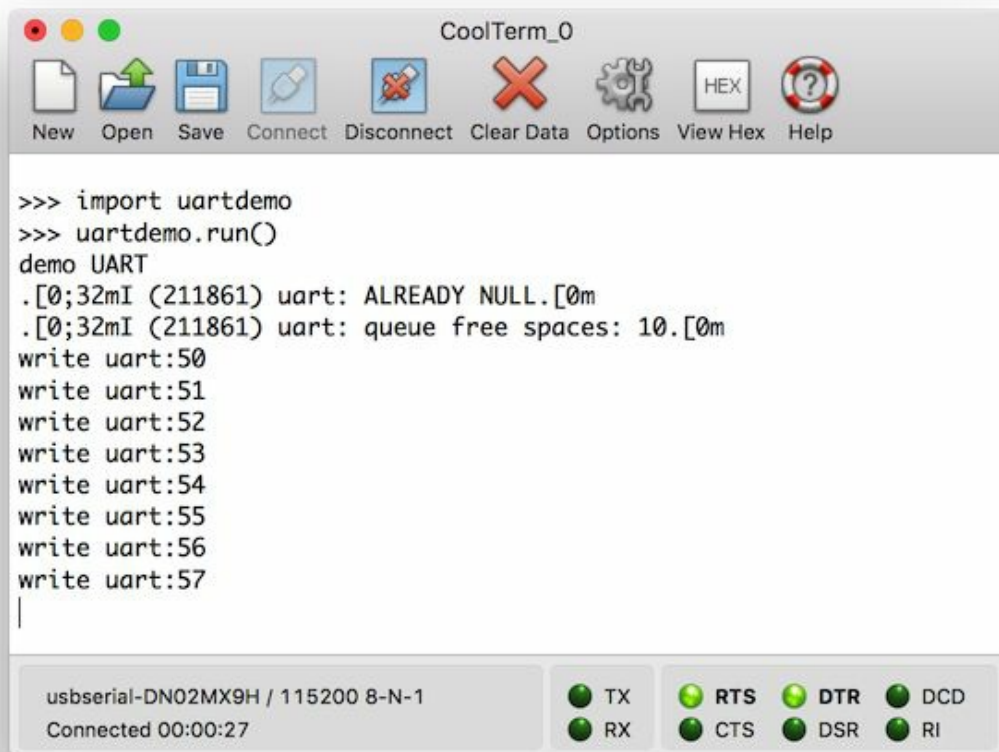
6.4 Testing

Now you can upload and run MicroPython program via ampy tool. If done, connect ESP32 UART to Arduino UART (Digital pins: 10 and 11). Now you can run Python program on MicroPython terminal from ESP32 board.

To see the UART output, open Serial Monitor tool from Arduino IDE. Set baud 9600. You should see the UART output.



The following is program output on MicroPython terminal.



7. Working with SPI

In this chapter I'm going to explain how to work with SPI on MicroPython board.

7.1 Getting Started

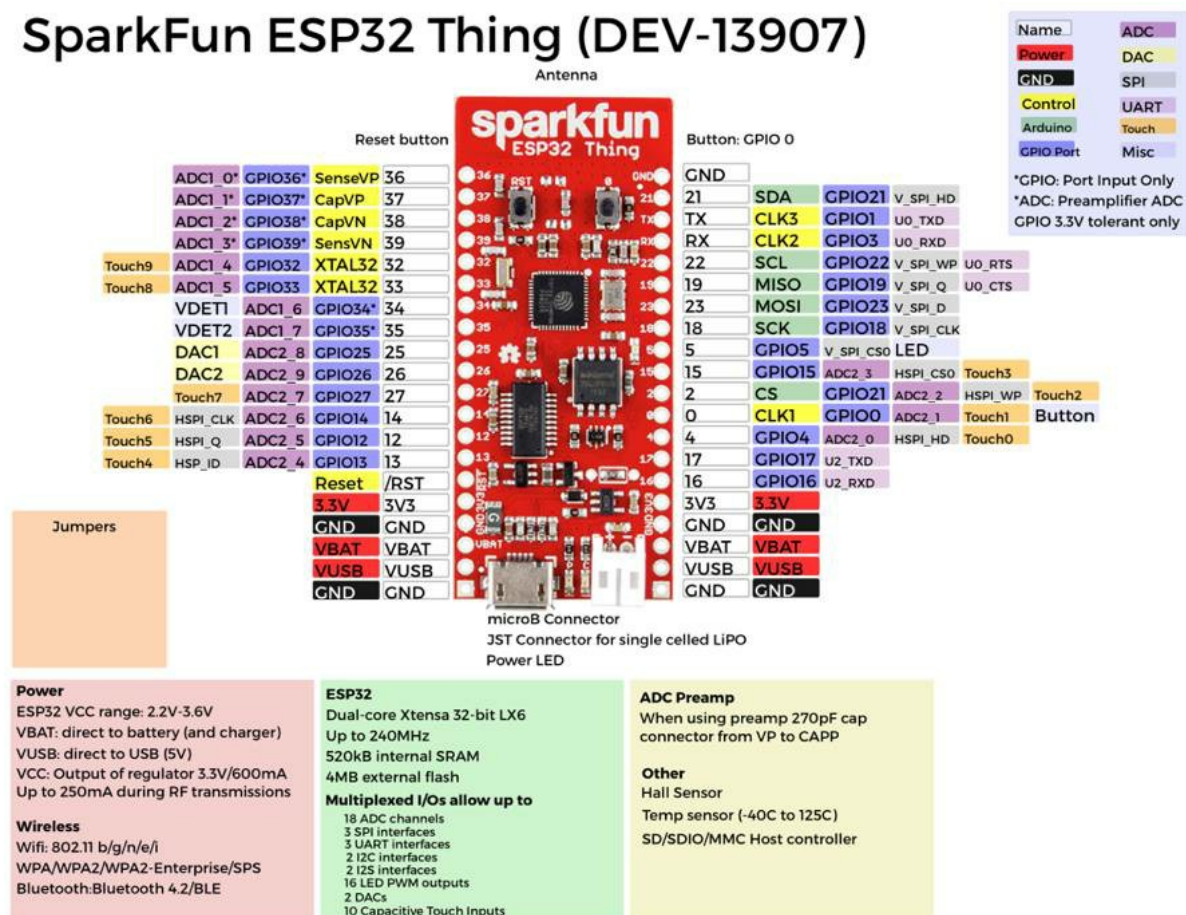
The Serial Peripheral Interface (SPI) is a communication bus that is used to interface one or more slave peripheral integrated circuits (ICs) to a single master SPI device; usually a microcontroller or microprocessor of some sort.

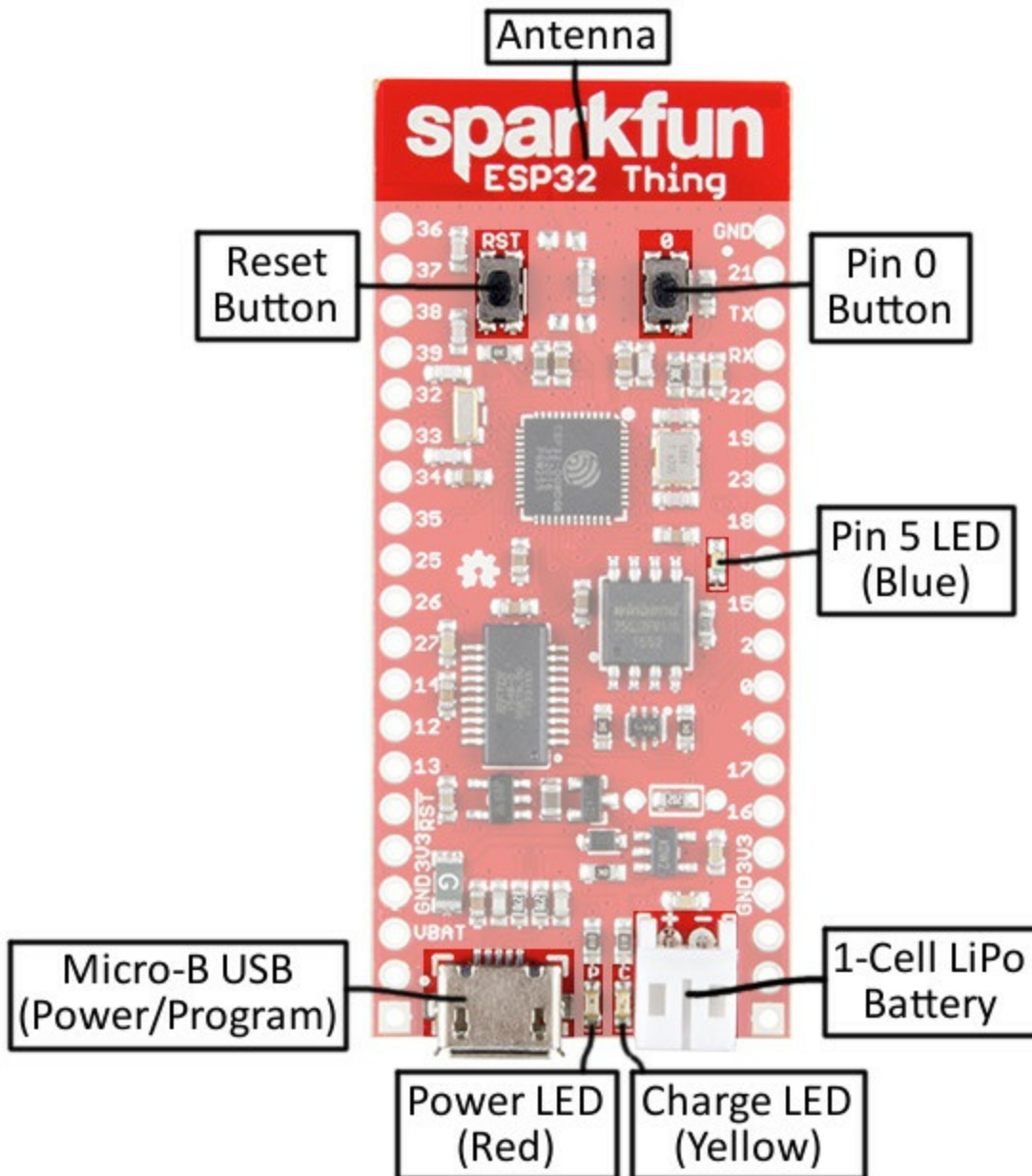
SPI in ESP32 board can be defined on the following pins:

- MOSI
- MISO
- SCK

You can see these pins on SPI SparkFun ESP32 Thing board, shown in Figure below.

SparkFun ESP32 Thing (DEV-13907)





We can use all pins for SPI. To access SPI, we can use SPI library.

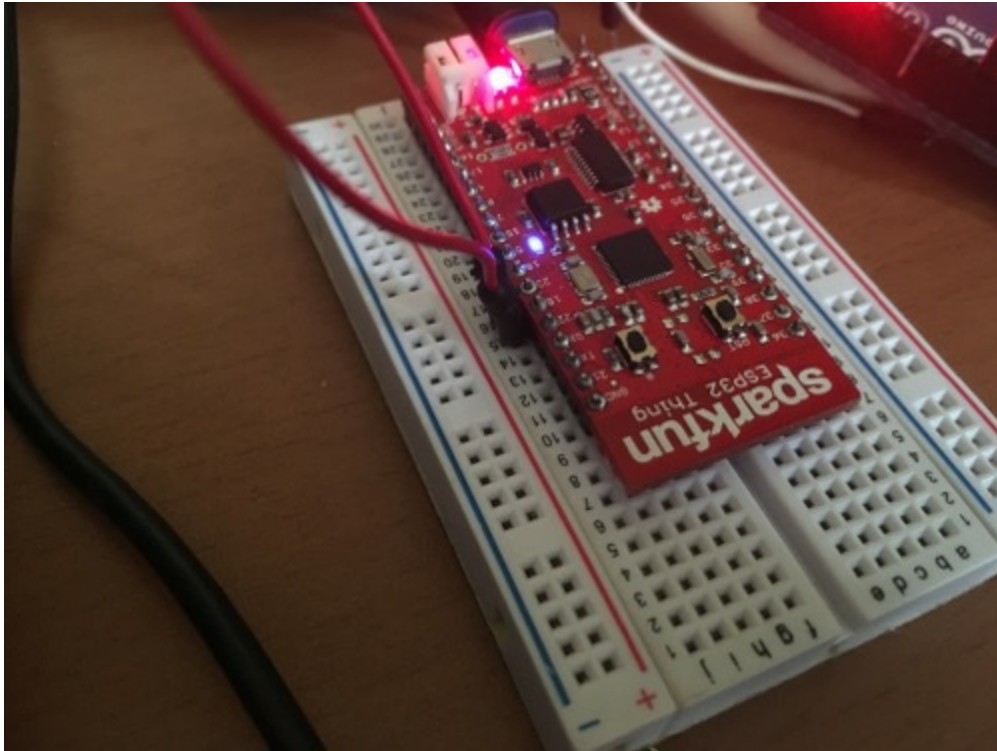
In this chapter, I develop SPI Loopback with MicroPython board.

Let's start!.

7.2 Wiring

For testing, we connect ESP32 MOSI (GPIO23) to MISO (GPIO19) using a jumper.

The following is a sample of wiring.



7.3 Writing a Program

The next step is to write a program for MicroPython board. Create a file, called **spidemo.py**, and write these scripts.

```
from machine import Pin, SPI
import random
import time

def run():
    print('demo spi')
    gpio_sck = Pin(18)
    gpio_mosi = Pin(23)
    gpio_miso = Pin(19)

    spi = SPI(2, sck=gpio_sck, mosi=gpio_mosi, miso=gpio_miso)
    while 1:
        tx = ''.join(chr(random.randint(50,85)) for _ in range(4))
        rx = bytearray(4)

        spi.write_readinto(tx,rx)

        print('tx: ' + str(tx))
        print('rx: ' + str(rx))

        time.sleep(2)
```

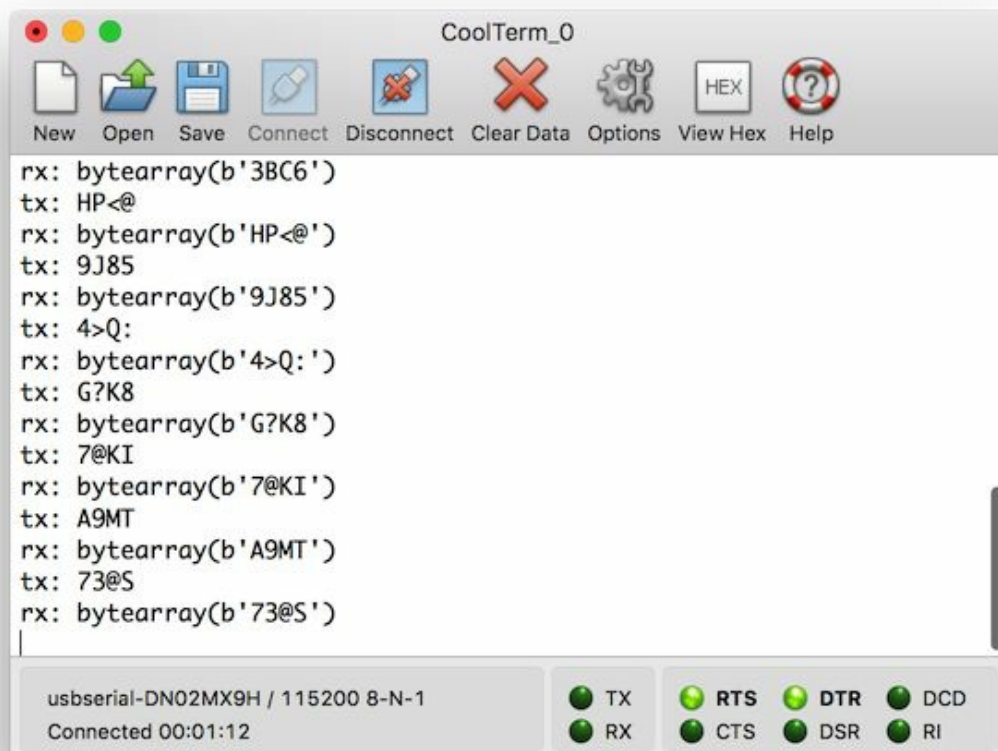
Save this code.

7.4 Testing

Now you can upload MicroPython program to MicroPython board on MicroPython terminal. If done, you can run the program.

```
>>> import spidemo  
>>> spidemo.run()
```

You should see received data from SPI.



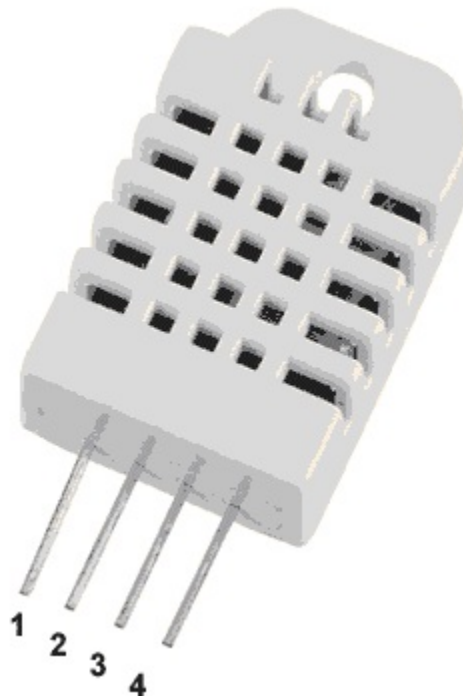
8. Working with DHT Module

In this chapter I'm going to explain how to work with DHT module on MicroPython boards.

8.1 Getting Started

In this chapter, we try to develop a simple application to access DHT module. This module can sense temperature and humidity. It's easy to find in electronic stores. You can see DHT22 layout in Figure below.

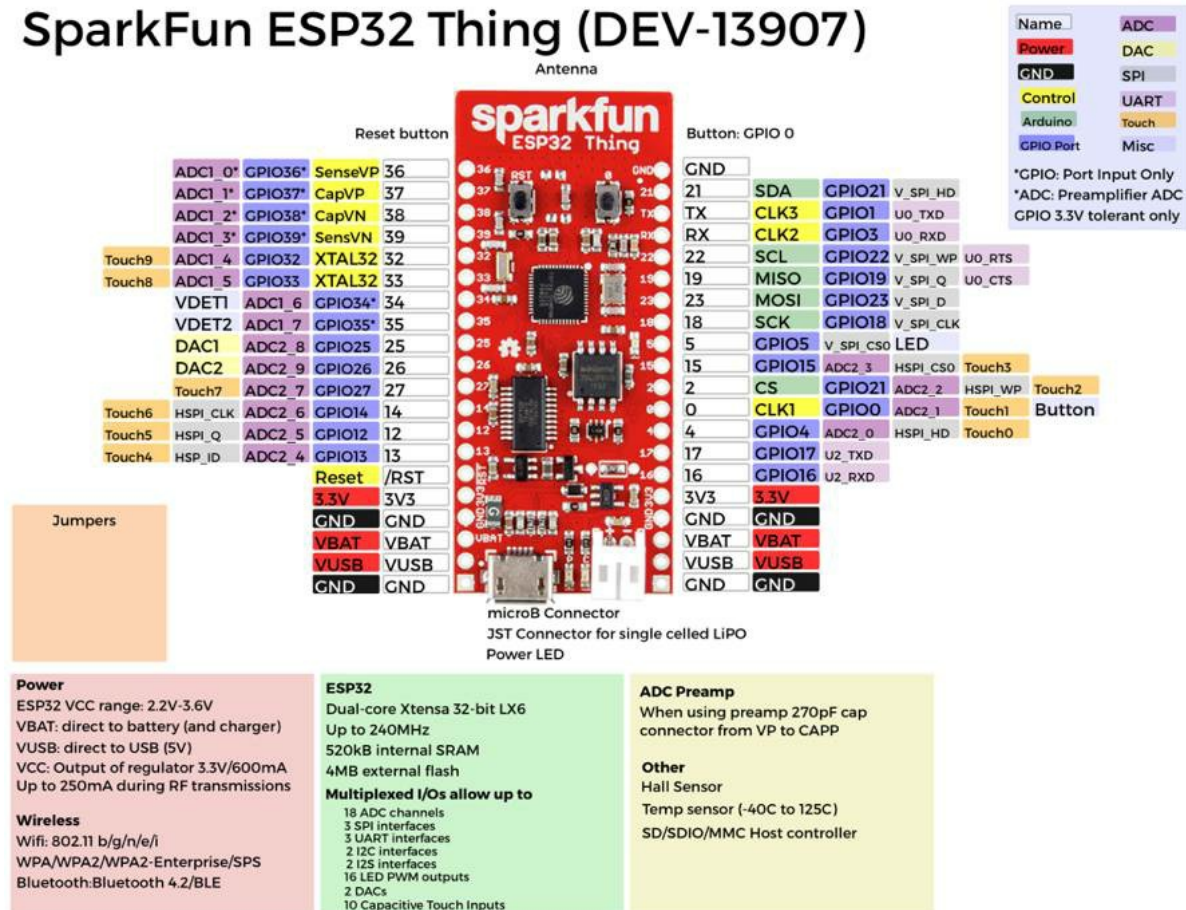
DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

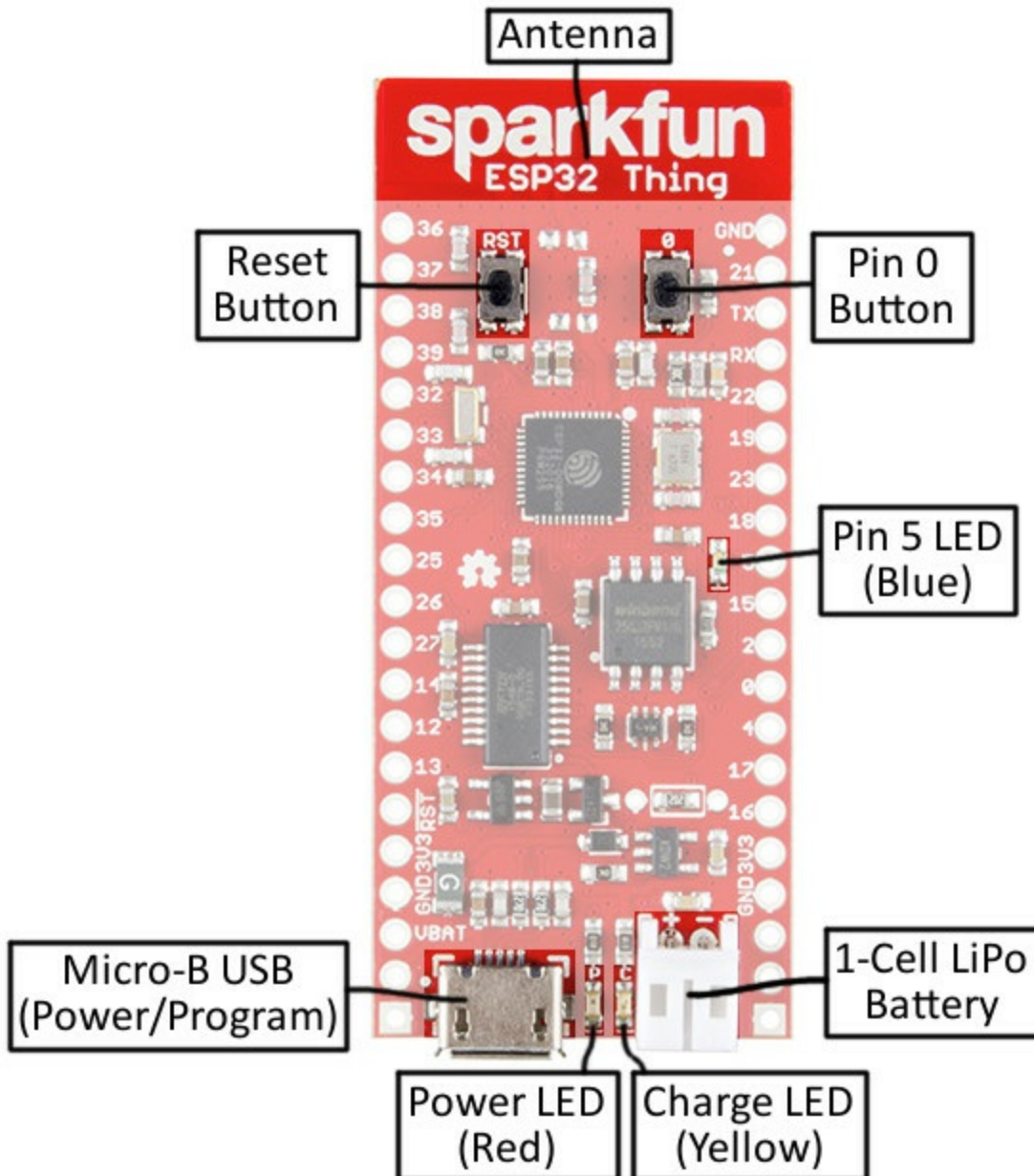


8.2 Wiring

I use SparkFun ESP32 Thing for MicroPython board. The following is SparkFun ESP32 Thing layout.

SparkFun ESP32 Thing (DEV-13907)



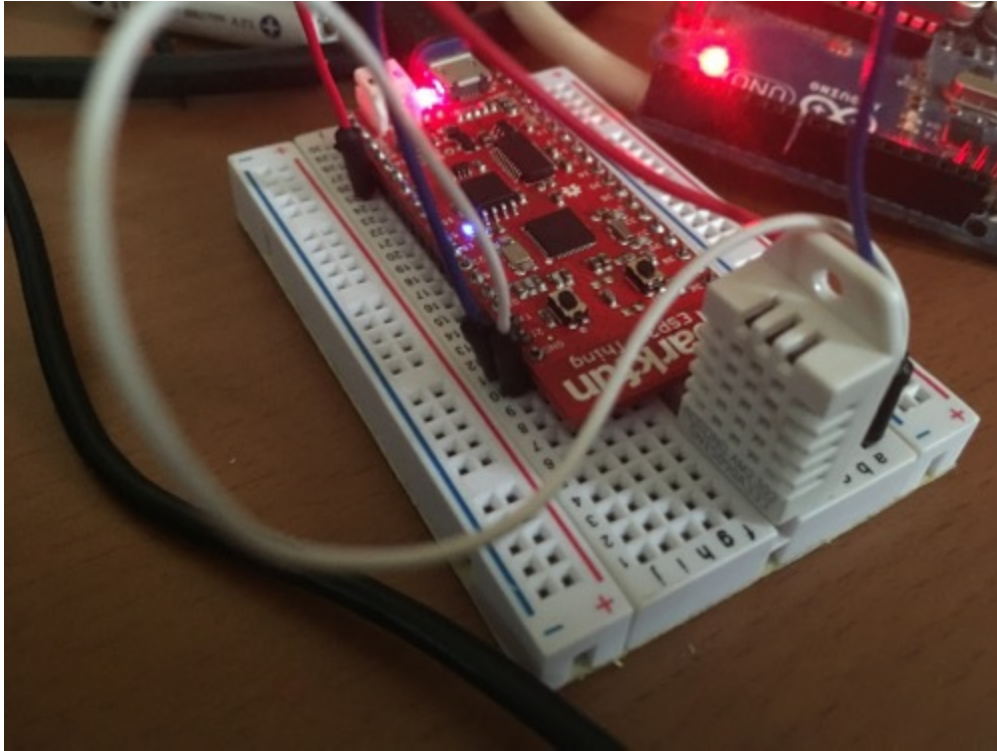


Our wiring for DHT22 and ESP32 as follows:

- DHT VCC is connected to ESP32 3.3V

- DHT GND is connected to ESP32 GND
- DHT Data is connected to ESP32 GPIO5 (D1)

The following is our implementation.



8.3 Writing MicroPython Program

Now we can access DHT using dht module from MicroPython. Open editor and write these scripts.

```
from machine import Pin
import dht
import time

def run():
    print('dht module demo')

    gpio_dht = Pin(21)
    d = dht.DHT22(gpio_dht)

    led = Pin(5, Pin.OUT)
    while 1:
        led.value(1)
        d.measure()
        temperature = d.temperature()
        humidity = d.humidity()

        print('Temperature: ' + str(temperature) + ' Celsius')
        print('Humidity: ' + str(humidity) + ' % RH')
        led.value(0)
        time.sleep(2)
```

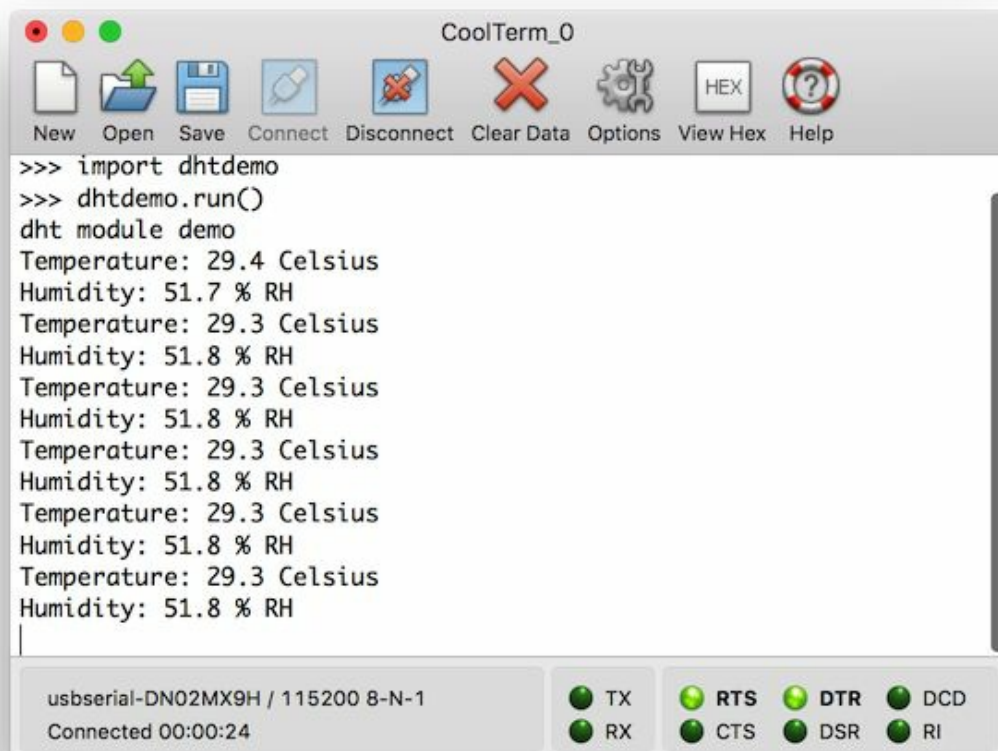
Save this program as dhtdemo.py.

8.4 Testing

Now you can upload `dhtdemo.py` to MicroPython board via ampy terminal. Then, run the program on MicroPython terminal.

```
>>> import dhtdemo  
>>> dhtdemo.run()
```

You should see temperature and humidity on Terminal.



9. Working with WiFi

In this chapter I'm going to explain how to work with WiFi on MicroPython boards.

9.1 Getting Started

In this chapter, we try to develop a simple application to access WiFi module.

9.2 Scanning WiFi Hotspot

I use SparkFun ESP32 Thing for MicroPython board. We try to scan existing WiFi. Write these scripts.

```
from machine import Pin
import network
import time

led = Pin(5, Pin.OUT)
wlan = network.WLAN(network.STA_IF)
wlan.active(True)

def run():
    print('Demo wifi scanning')
    while 1:
        led.value(1)
        print('scanning wifi...')

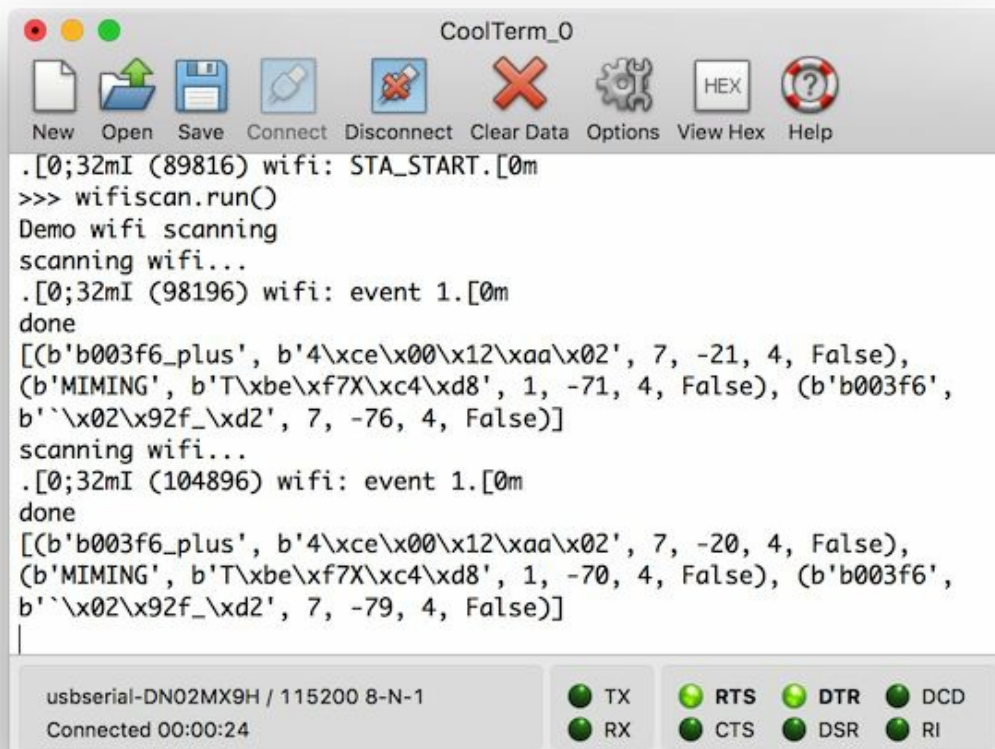
        result = wlan.scan()
        print('done')
        print(result)
        led.value(0)

        time.sleep(5)
```

Save this program as wifiscan.py.

Then, upload the program and run it.

```
>>> import wifiscan
>>> wifiscan.run()
```



9.3 Developing WiFi Application

In this section, we develop a simple application to access a website. Firstly, we connect to existing WiFi and then try to send a HTTP request to a webserver. Open editor and write these scripts.

```
from machine import Pin
import network
import usocket as socket
import time

ssid = '<ssid>'
ssid_key = '<ssid_key>'

wlan = network.WLAN(network.STA_IF)
wlan.active(True)


def run():
    print('Demo wifi scanning')

    print('Connecting to wifi')
    while not wlan.isconnected():
        wlan.connect(ssid, ssid_key)
        time.sleep(2)

    print('connected')

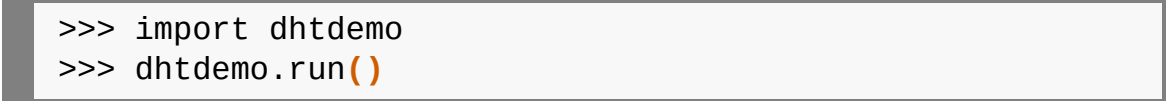
    # print mac and ip address
    mac = wlan.config('mac')
    ip = wlan.ifconfig()
    print(mac)
    print(ip)

    # test socket
    print('access a website')
    addr = socket.getaddrinfo('micropython.org', 80)[0][-1]
    s = socket.socket()
    s.connect(addr)
    s.send(b'GET HTTP1.1\r\nHost: micropython.org\r\n\r\n')
    data = s.recv(500)
    print(data)
    s.close()
```



Save this program as wifidemo.py.

Upload this program and run the program on MicroPython terminal.



```
>>> import dhtdemo  
>>> dhtdemo.run()
```

Source Code

You can download source code on

<http://www.aguskurniawan.net/book/esp32a234.zip> .

My Books for ESP8266 Development

I wrote two books related to ESP8266 development.

NodeMCU Development

Workshop, <http://blog.aguskurniawan.net/post/nodemcu.aspx>.



SparkFun ESP8266 Thing Development

Workshop, <http://blog.aguskurniawan.net/post/sparkfun8266.aspx>.



SparkFun ESP8266 Thing Development Workshop



Agus Kurniawan

Contact

If you have question related to this book, please contact me at aguskur@hotmail.com . My blog: <http://blog.aguskurniawan.net>