UNIT ELECTRONICS

DualMCU ONE

Release 0.0.1

Cesar Bautista

Jun 03, 2024

CONTENTS

CHAPTER

ONE

MICROPYTHON GETTING STARTED

Note: You are reading the most recent version available.

CHAPTER

TWO

BLINK

2.1 Hello World Program

This chapter introduces the classic "Hello World" program and a simple counter in MicroPython. It is an excellent starting point to understand the basic syntax and structure of a Python program.

2.1.1 Program Code

The following code prints "Hello World!" to the console and then counts from 1 to 10:

```
print('Hello world! I can count to 10:')
for i in range(1,11):
    print(i)
```

2.1.2 Expected Output

When you run the above code, you should see the following output in the console:

```
Hello world! I can count to 10:
1
2
3
4
5
6
7
8
9
10
```

2.1.3 RGB LED

In this section, we will learn how to work an RGB LED using a microcontroller. We will learn how to use the LED to different GPIO pins and control its on and off states with a simple Python program.

2.2 Hardware Description

An RGB LED has three LEDs in one: red, green, and blue. You can control the color of the LED by varying the intensity of each of the LEDs. Here is an image of the RGB LED:

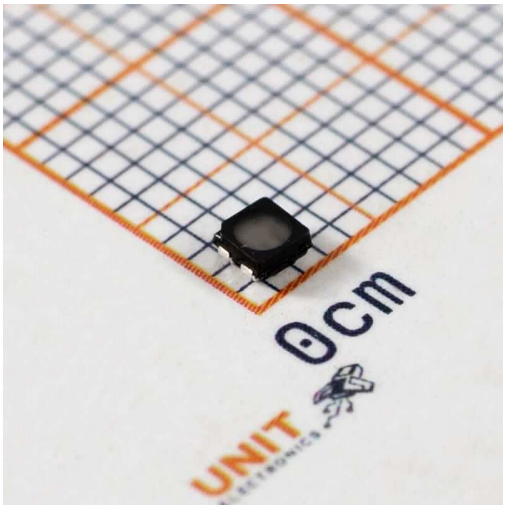

Fig. 2.1: RGB LED

2.2.1 Pin Connections

Table 2.1: RGB LED Connections

PIN	GPIO ESP32	GPIO RP2040
BLUE	27	
RED	25	25
GREEN	26	

2.2.2 RED LED Code

```
import machine
import time


led = machine.Pin(25, machine.Pin.OUT)

def loop():
    while True:
```

(continues on next page)

(continued from previous page)

```
        led.on()
        time.sleep(1)

        led.off()
        time.sleep(1)

loop()
```

2.2.3 RGB LED Code

Note: This code block is designed to work exclusively with the RGB LED on the DualMCU development board when using the ESP32 microcontroller.

```
import machine
import time

led_pin = machine.Pin(27, machine.Pin.OUT)
led_pin2 = machine.Pin(26, machine.Pin.OUT)
led_pin3 = machine.Pin(25, machine.Pin.OUT)

def loop():
     while True:
        led_pin.on()
        led_pin2.on()
        led_pin3.on()
        time.sleep(1)
        led_pin.off()
        led_pin2.off()
        led_pin3.off()
        time.sleep(1)

loop()
```

CHAPTER

THREE

ANALOG TO DIGITAL CONVERSION

The DualMCU development board, equipped with the RP2040 microcontroller, includes five analog pins. These pins are capable of reading analog voltages and converting them into digital values. Below you will find the details on how to utilize these pins for ADC operations.

3.1 Distribution of Pins

Below is a table showing the distribution of analog pins on the DualMCU board along with their corresponding GPIO pins on the RP2040.

Table 3.1: Pin Mapping

PIN	GPIO RP2040
A0	26
A1	27
A2	28
A3	29

3.2 Class ADC

The `machine.ADC` class is used to create ADC objects that can interact with the analog pins.

class machine.**ADC**(*pin*)

The constructor for the ADC class takes a single argument: the pin number.

3.3 Example Definition

To define and use an ADC object, follow this example:

```
import machine
adc = machine.ADC(0)  # Initialize ADC on pin A0
```

3.4 Reading Values

To read the analog value converted to a digital format:

```
adc_value = adc.read()
print(adc_value)
```

3.5 Example Code

Below is an example that continuously reads from an ADC pin and prints the results:

```
import machine
import time

# Setup
A0 = machine.Pin(26, machine.Pin.IN)  # Initialize pin A0 for input
adc = machine.ADC(A0)                 # Create ADC object

# Continuous reading
while True:
    adc_value = adc.read_u16()        # Read the ADC value
    print(f"ADC Reading: {adc_value:.2f}")  # Print the ADC value
    time.sleep(1)                     # Delay for 1 second
```

CHAPTER

FOUR

I2C (INTER-INTEGRATED CIRCUIT)

4.1 Pinout Details

Below is the pinout table for the I2C connections on the DualMCU ONE, detailing the corresponding GPIO connections for both the ESP32 and RP2040 microcontrollers.

Table 4.1: I2C Pinout

PIN	GPIO ESP32	GPIO RP2040
SDA	21	4
SCLK	22	5

4.1.1 SSD1306 Display

The SSD1306 display is connected using a JST 1.25mm 4-pin connector. The following table provides the pinout details for the display connection.

Table 4.2: SSD1306 Display Pinout

Pin	Connection
1	GND
2	VCC
3	SDA
4	SCL

The display is a 128x64 pixel monochrome OLED display equipped with an SSD1306 controller.

Library

The *ocks.py* library for MicroPython on ESP32 & RP2040 is compatible with the SSD1306 display controller.

4.2 Installation

1. Open Thonny.
2. Navigate to **Tools** -> **Manage Packages**.
3. Search for `ocks` and click **Install**.

Alternatively, download the library from ocks.py.

4.3 Microcontroller Configuration

class SoftI2C(*scl*, *sda*, ***, *freq=400000*, *timeout=50000*)

Change the following line depending on your microcontroller:

For ESP32:

```
>>> i2c = machine.SoftI2C(freq=400000, timeout=50000, sda=machine.Pin(21), scl=machine.
↪Pin(22))
```

For RP2040:

```
>>> i2c = machine.SoftI2C(freq=400000, timeout=50000, sda=machine.Pin(4), scl=machine.
↪Pin(5))
```

4.4 Example Code

Code:

```
import machine
from ocks import SSD1306_I2C

i2c = machine.SoftI2C(freq=400000, timeout=50000, sda=machine.Pin(*), scl=machine.Pin(*))

oled = SSD1306_I2C(128, 64, i2c)

# Fill the screen with white and display
oled.fill(1)
oled.show()

# Clear the screen (fill with black)
oled.fill(0)
oled.show()

# Display text
oled.text('UNIT', 50, 10)
oled.text('ELECTRONICS', 25, 20)
oled.show()
```

This code initializes the display, fills the screen with different colors, and displays text.

`sda=machine.Pin(*)` and `scl=machine.Pin(*)` should be replaced with the appropriate GPIO pins for your setup.

This version clarifies the structure, pin configurations, library usage, installation instructions, and example code for your project. Adjust the placeholders marked with * in your actual code based on your specific GPIO pin assignments.

4.4.1 Module Functions

CHAPTER

FIVE

SPI (SERIAL PERIPHERAL INTERFACE)

Explore SPI communication protocol and learn how to interface with SPI devices using the DualMCU ONE board. Well walk you through setting up SPI communication and communicating with SPI devices.

5.1 SDCard SPI

Warning: Ensure that the Micro SD contain data. We recommend saving multiple files beforehand to facilitate the use.

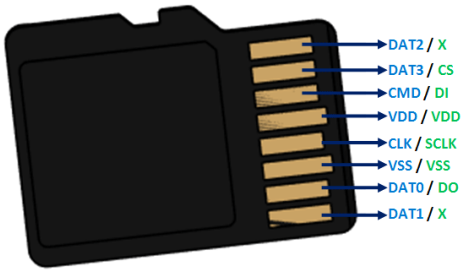

Fig. 5.1: Micro SD Card Pinout

5.1.1 Library

The sdcard_lib library for MicroPython on ESP32 and RP2040 is compatible with SD card memory storage capacities of 8GB, 16GB, and 32GB.

5.1.2 Installation

1. Open Thonny.
2. Navigate to **Tools** -> **Manage Packages**.
3. Search for `sdcard_lib` and click **Install**.

5.1.3 VSPI Interfacing.

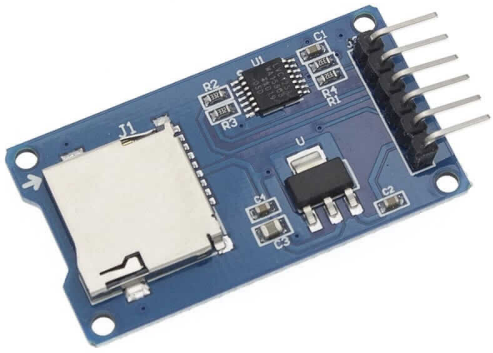

Fig. 5.2: Micro SD Card external reader

The conections are as follows:

This table illustrates the connections between the SD card and the GPIO pins on the ESP32 and RP2040 microcontrollers.

Table 5.1: VSPI Connections

SD Card	Pin Name	ESP32	RP2040
D3	SS	5	17
CMD	MOSI	23	19
VSS	GND		
VDD	3.3V		
CLK	SCK	18	18
D0	MISO	19	16

Descriptions

- SCK (Serial Clock)
- SS (Slave Select)

```
import machine, sdcard_lib, os

SCK_PIN = 18
MOSI_PIN = 23
MISO_PIN = 19
CS_PIN = 5
spi = machine.SPI(1, baudrate=100000, polarity=0, phase=0, sck=machine.Pin(SCK_PIN),␣
↪mosi=machine.Pin(MOSI_PIN), miso=machine.Pin(MISO_PIN))
spi.init()
sd = sdcard_lib.SDCard(spi, machine.Pin(CS_PIN))
os.mount(sd, '/sd')
os.listdir('/')

print("files ..."")
print(os.listdir("/sd"))
```

5.1.4 HSPI Interfacing.

This table details the connections between the SD card and the ESP32 microcontroller.

Table 5.2: HSPI Connections

SD Card	ESP32
D2	
D3	SS (Slave Select)
CMD	MOSI
VSS	GND
VDD	3.3V
CLK	SCK (Serial Clock)
VSS	GND
D0	MISO
D1	

For the test , we will utilize an ESP32 WROM-32E and a SanDisk Micros Ultra card with a capacity of 32 GB.

```
import machine
import os
import sdcard_lib as sdcard

spi = machine.SPI(2, baudrate=1000000, polarity=0, phase=0, sck=machine.Pin(14),␣
↪mosi=machine.Pin(15), miso=machine.Pin(2))

sd = sdcard.SDCard(spi, machine.Pin(13))

vfs = os.VfsFat(sd)
os.mount(vfs, "/sd")
```

(continues on next page)

(continued from previous page)

```
print("Files in the root of the SD card:")
print(os.listdir("/sd"))

try:
    with open("/sd/sample.txt", "r") as f:
        print(f.read())
except OSError:
    print("File not found or unable to read the file.")

os.umount("/sd")
```

CHAPTER

SIX

WS2812 CONTROL

This section describes how to control WS2812 LED strips using the DualMCU ONE board. The DualMCU ONE board has a GPIO pin embebbed connected to the single WS2812 LED.

Table 6.1: Pin Mapping for WS2812

PIN	GPIO RP2040
DIN	24

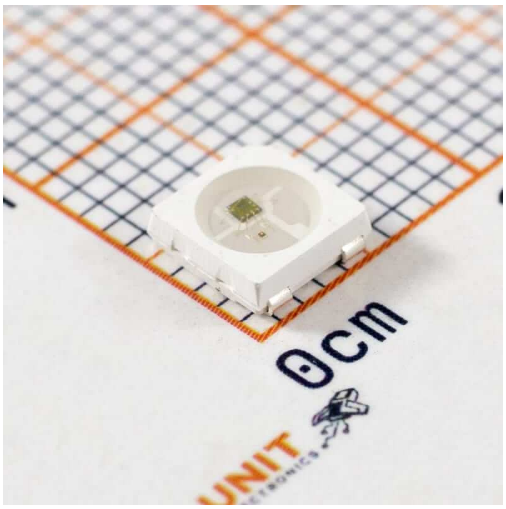

Fig. 6.1: WS2812 LED Strip

6.1 Code Example

Below is an example that demonstrates how to control WS1280 LED strips using the DualMCU ONE board:

```
from machine import Pin
from neopixel import NeoPixel
np = NeoPixel(Pin(24), 1)
np[0] = (255, 128, 0) # set to red, full brightness

np.write()
```

Tip: for more information on the NeoPixel library, refer to the NeoPixel Library Documentation.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

M

S