# DualMCU

## RP2040 + ESP32

**SKU: UE0002**

**Product Reference Manual**

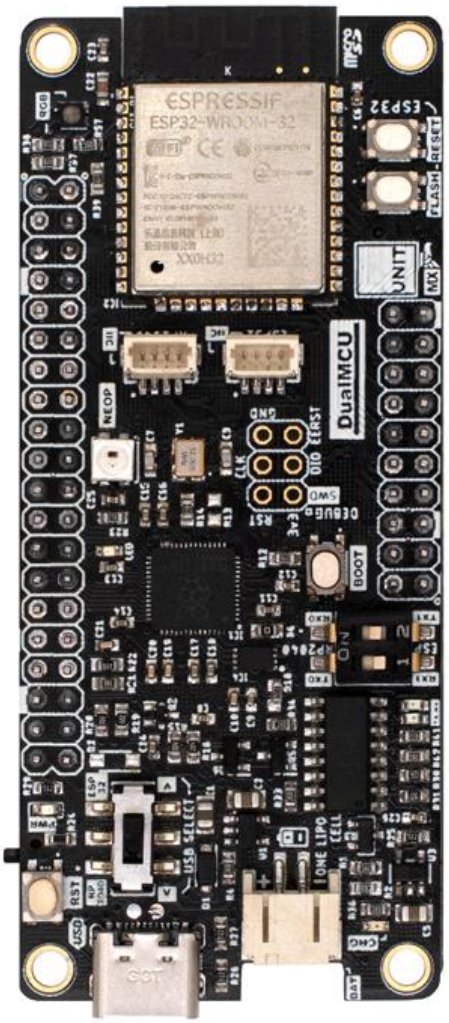
### Description

Unit DualMCU brings the Raspberry Pi RP2040 microcontroller plus an Espressif ESP32 WROOM chip in a simple one module, make the most of the dual core 32-bit Arm® Cortex®-M0+ to make Internet of Things projects with Bluetooth® and Wi-Fi connectivity.

DualMCU has been thought of as a mix between two 32-bit microprocessors: a Raspberry RP2040 cortex M0+ running at 133 Mhz and an Espresiff ESP32 running up to 240 MHz, in this way we implement all the power of both microcontrollers in a single development card. With a PCB size of 36mm x 84mm and using surface mount technology, four programmable cores with wireless functions and features are found with very low power consumption.

Both microcontrollers can communicate via serial protocol by simply activating a 2-way DIP SWITCH, for example, we can program the ESP32 as a wireless co-processor to provide the RP2040 microcontroller with WIFI or BLUETOOTH communication. For programming, there is a mechanical selector that allows us to select which MCU connects using a single USB type C connector.

Interconnect an unlimited number of I2C devices using its STEMMA, QWIIC and GROVE compatible JST-SH I2C connectors and dive into real-world projects with the onboard RGB 2020 and WS2812B LED.




With over 46 general purpose pins (GPIO), DualMCU exposes most output ports of both microcontrollers via pin headers and JST-SH connectors (STEMMA, QWIIC and GROVE I2C compatible). It has more common peripherals such as: UART, I2C, SPI, PWM, ADC, Touch and state machines. The multiple forms of connection to sensors, actuators or even between both microcontrollers will only be limited by your creativity.

Among its available connectors, it has a Micro SD socket for high-capacity memories (up to 64 GB tested) directly connected to the ESP32 microcontroller through SPI communication.

Optionally, the ATECC608A can be incorporated and used for authentication tasks, which can be accessed by both MCUs through selection pads for I2C communication.

The DualMCU also features a 3.7V LiPO battery charging system making it ideal for applications that require a portable and rechargeable wireless system. Battery charging is done through its USB Type-C connector or through an external source such as a solar cell connected by its dedicated VIN pin for external power sources.

**Applications.**

Internet of Things (IoT), prototyping, machine learning, evaluation & development.

**Support.**

**DualMCU is supported in the Arduino development environment and Python Integrated Development Environment (IDE)** such as Thonny via an interactive prompt (REPL) to execute commands immediately in mycropython and circuitpython programming language.

## Features

### Raspberry Pi RP2040 Microcontroller

- 133MHz 32bit Dual Core Arm® Cortex®-M0+
- 264kB on-chip SRAM
- Direct Memory Access (DMA) controller
- Support for up to 16MB of off-chip Flash memory via dedicated QSPI bus
- Programmable IO (PIO) for extended peripheral support
- 30 GPIO pins, 4 of which can be used as analog inputs
- 4 channel ADC with internal temperature sensor, 0.5 MSa/s, 12-bit conversion
- 16 PWM channels
- SWD Debugging
- 2 UARTs
- 2 SPI controllers
- 2 I2C controllers
- 8 PIO state machines
- USB 1.1 controller and PHY, with host and device support
- 2 on-chip PLLs to generate USB and core clock
- 40nm process node
- Multiple low power mode support
- Internal Voltage Regulator to supply the core voltage
- Advanced High-performance Bus (AHB)/Advanced Peripheral Bus (APB)

### Memory

- W25Q16JVUXIQ 2MB NOR Flash
- 532MHz Quad SPI
- 66MB/S continuous data transfer rate
- 100K program/erase cycles
- More than 20-year data retention

### Espressif ESP32 WROOM 32 Wi-Fi/Bluetooth® Module

- 240MHz 32bit Dual Core Xtensa LX6
- 520kB on-chip SRAM
- 448 Kbyte ROM for booting and core functions
- 8 MB Integrated SPI flash
- 1 kbit EFUSE (non- erasable memory) for MAC addresses, module configuration, Flash-Encryption, and Chip-ID
- IEEE 802.11b/g/n (802.11n up to 150 Mbps) single-band 2.4 GHz Wi-Fi operation, center frequency range of operating channel (2412 ~ 2484 MHz)
- Bluetooth® 4.2 BR/EDR and Bluetooth LE specification, NZIF receiver with –97 dBm sensitivity, Class-1, class-2 and class-3 transmitter, AFH
- +12 dBm transmitting power, the internal PCB antenna in the module eliminates the need for an external antenna
- ADC, I2C, SDIO, QSPI, UART, I2S, Two-Wire Automotive Interface (TWAI®), compatible with ISO11898-1 (CAN Specification 2.0)
- On-chip Hall Sensor

### WS2812B LED

- RGB NeoPixel for full color indication
- Connected to RP2040 GPIO

### RGB LED

- Common Anode
- Connected to ESP32 GPIO

### Builtin LED

- LED for general purpose blinking, connected to RP2040 GPIO25

**MicroSD CARD**

- Connected to ESP32 vía VSPI

**ATECC608A** (do not populate component)

- Authentication chip connected to ESP32 or RP2040 accessed by both MCUs through selection pads for I2C communication

**I/O**

44 x GPIO pins with following capabilities:

- 11 of which can be used as 12-bit ADC - Analog to Digital Converter inputs (4 from RP2040 and 7 in the ESP32 microcontroller)
- 42 of which can be used as Digital Pin (26 from RP2040 and 16 in the ESP32 microcontroller)
- 3 x UART peripherals (2 UARTs in the RP2040 and 1 in the ESP32)
- 2 x SPI (one from RP2040 and one in the ESP32)
- 3 x I2C (Two of which have a JST-1.0 mm pitch on-board connectors, compatible with STEMMA QT, QWIIC and GROVE devices with no soldering)
- All 26 digital pins of RP2040 can be driven by the PWM block with a maximum of 16 controllable PWM outputs
- All ESP32 pins digital outputs can be used as PWM pins (GPIOs 36 and 39 can't generate PWM)
- 3 x Capacitive Touch Sensor from ESP32 microcontroller
- ESP32 CAN 2.0

**Power**

- 3.3 v LDO 600 mA
- 3.3V Power/enable pin
- VUSB Output /VIN (3.2 - 6 V) Pin
- Built in 200mA+ lipoly charger with charging status indicator LED

**Connector**

- 2 x I2C JST-SH (1 mm pitch)
- 1 microSD Card Holder
- 1 USB Type C
- 1 JST-SH Battery Connector (2 mm pitch)

**Switch**

- Power Switch
- USB Communication Selector
- DIP Switch for UART communication
- Reset button and Bootloader select button for RP2040 quick restarts (no unplugging-replugging to relaunch code)
- ESP32 Reset button and Flash/Boot button for manually entering flash mode

**OSC**

- 12 MHz crystal for perfect timing

CONTENTS

## 1 The Board

1.1 Accessories

USB type C cable
40-pin 2.54mm male headers
20-pin 2.54mm male headers

## 2 Ratings

2.1 Recommended Operating Conditions

| Symbol | Description | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| **VIN** | Input voltage from VIN pin | 4 | 5 | 6 | V |
| **VBUS** | Input voltage from USB connector | 4.6 | 5 | 5.5 | V |
| **VUSB** | Output voltage from VUSB pin | 4.4 | 4.8 | 5.2 | V |
| **VBAT** | Output battery voltage | 3.2 | 3.7V | 4.25 | V |
| **V3v3** | 3.3V output to user application | 3.25 | 3.3 | 3.35 | V |
| **I3v3** | 3v3 output current (including onboard IC) | 600 | - | - | mA |
| **VIH (RP2040)** | Input high-level voltage | 2 | - | 3.6 | V |
| **VIL (RP2040)** | Input low-level voltage | -0.3 | - | 0.8 | V |
| **IOH Max (RP2040)** | Current at VDD-0.4v, output set high | | | 8 | mA |
| **IOL Max (RP2040)** | Current at VSS+0.4v, output set low | | | 8 | mA |
| **VOH (RP2040)** | Output high voltage, 8 mA | 2.62 | - | 3.3 | v |
| **VOL (RP2040)** | Output low voltage. 8 mA | 0 | - | 0.5 | V |
| **RPU (RP2040)** | Pull-Up Resistance | 50 | - | 80 | kΩ |
| **RPD (RP2040)** | Pull-Down Resistance | 50 | - | 80 | kΩ |
| **IIOVDD_MAX (RP2040)** | Maximum Total IOVDD current* | | | 50 | mA |

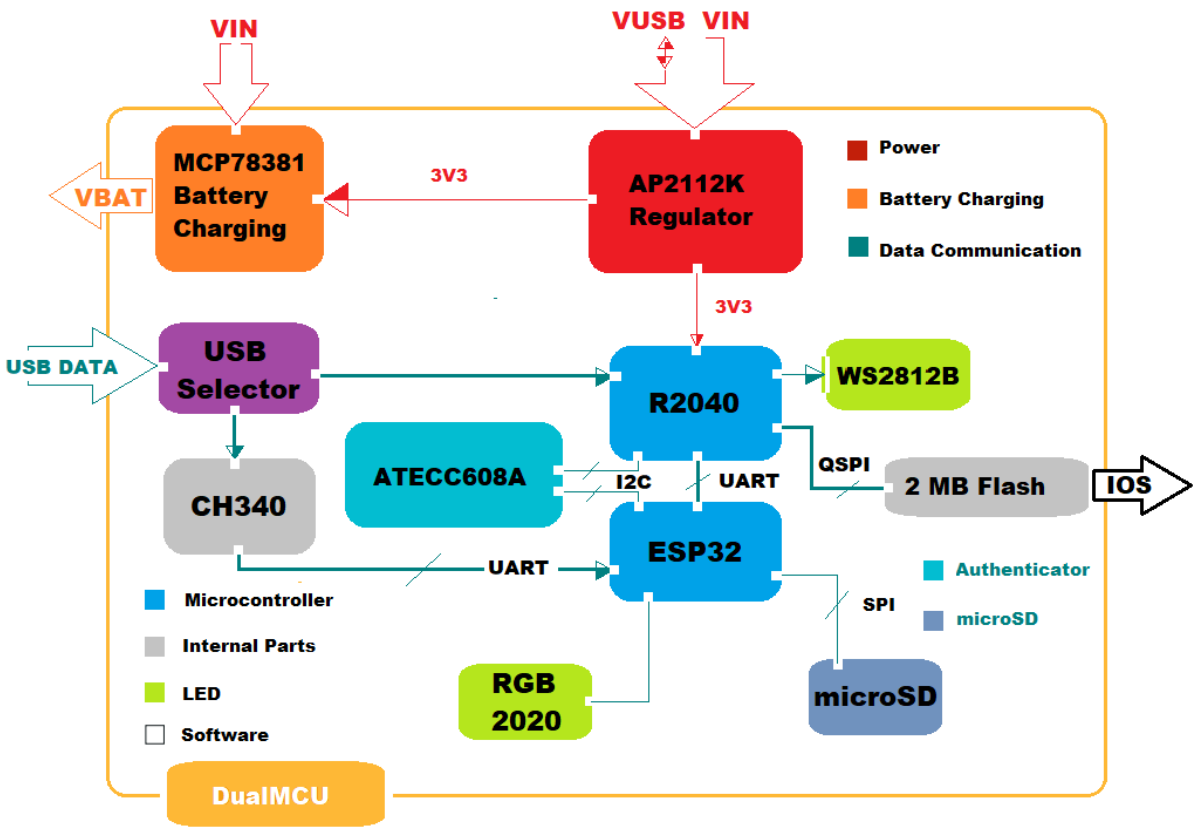
| | | | | | |
|---|---|---|---|---|---|
| **IIOVSS_MAX (RP2040)** | Maximum Total VSS current due to IO (IOVSS)** | | | 50 | mA |
| **TOP (RP2040)** | Operating temperature | -20 | - | 80 | °C |
| **VIH (ESP32)** | Input high-level voltage | 2.475 | - | 3.6 | V |
| **VIL (ESP32)** | Input low-level voltage | -0.3 | - | 0.825 | V |
| **IOH Max (ESP32)** | Current at VDD=3.3V, VOH >=2.64V, output set high*** | | | 40 | mA |
| **IOL Max (ESP32)** | Current at VDD = 3.3 V, VOL =0.495 V, output set low | | | 28 | mA |
| **VOH (ESP32)** | Output high voltage | 2.64 | - | 3.3 | V |
| **VOL (ESP32)** | Output low voltage | - | - | 0.33 | V |
| **RPU (ESP32)** | Pull-Up Resistance | - | 45 | - | kΩ |
| **RPD (ESP32)** | Pull-Down Resistance | - | 45 | - | kΩ |
| **TOP (ESP32)** | Operating temperature | -40 | - | 85 | °C |

* Sum of all current being sourced by GPIO and QSPI pins
**Sum of all current being sunk into GPIO and QSPI pins
***Per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, VOH>=2.64 V, as the number of current-source pins increases

## 3 Functional Overview

The RP2040 microcontroller alongside with the ESP32 MCU and the on-board leds and on-boards connectors, provides opportunities for low-power Internet of Things (IoT) development and whatever your microcontroller application, from motor control to machine learning, from digital audio to automotive, the DualMCU RP2040 & ESP32 has the performance, feature set, and support to make your product fly.

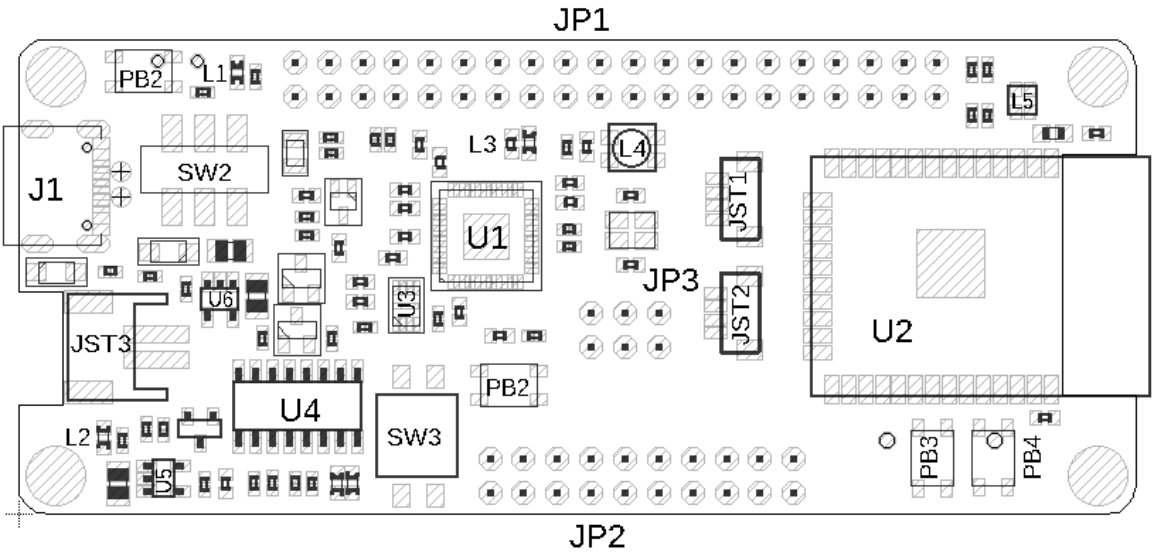
### 3.1 Block Diagram




Block Diagram of DualMCU RP2040 + ESP32

### 3.2 Board Topology

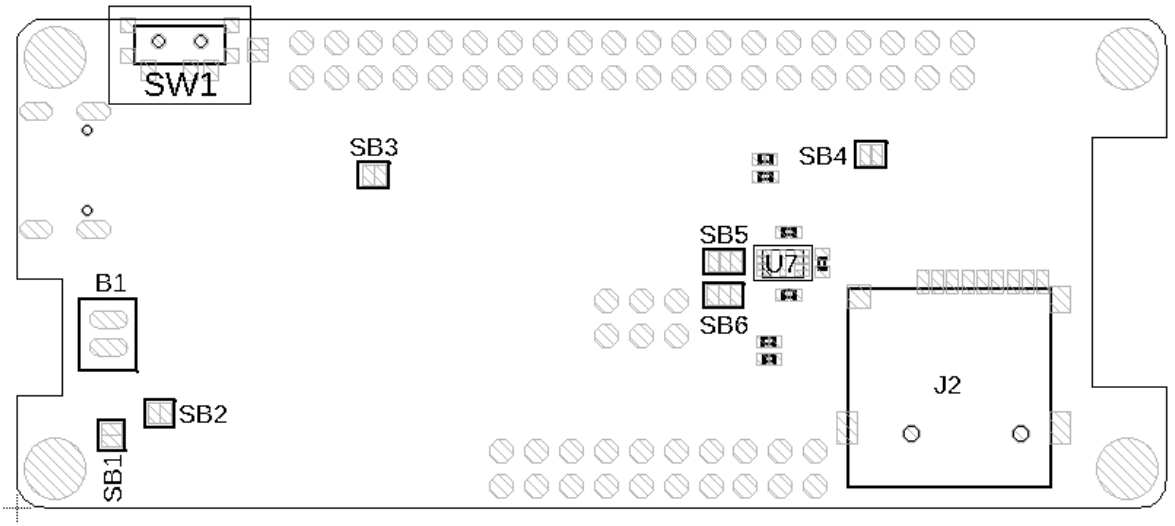
Front View




Front View of DualMCU RP2040 + ESP32 Topology

| Ref. | Description | Ref. | Description |
|---|---|---|---|
| U1 | Raspberry pi RP2040 Microcontroller | U4 | CH340C USB bus convert IC |
| U2 | Espresiff ESP32 WROOM Wi-Fi/Bluetooth® Module | U5 | MCP73831 Battery Charge Management IC |
| U3 | W25Q16JVUXIQ 2MB Flash IC | U6 | AP2112K 3v3 LDO Voltage Regulator |
| L1 | Power On LED | L2 | Charge LED |
| L3 | Builtin LED | L4 | WS2812B LED |
| L5 | RGB 2020 LED | J1 | Male USB Type C Connector |
| PB1 | RP2040 Reset Button | PB2 | RP2040 Boot Button |
| PB3 | ESP32 Flash Button | PB4 | ESP32 Reset Button |
| JP1 | RP2040 GPIO Header | JP2 | ESP32 GPIO Header |
| JP3 | RP2040 (SWD) Debug Header | JST1 | RP2040 I2C JST Connector |
| JST2 | ESP32 I2C JST Conector | JST3 | JST Connector for LiPo Battery |
| SW2 | USB Communication Selector | SW3 | UART DIP Switch |

Back view




Back View of DualMCU RP2040 + ESP32 Topology

| Ref. | Description | Ref. | Description |
|---|---|---|---|
| U7 | ATECC608A-MAHDA-T Crypto IC | J2 | Micro SD Card Connector |
| SW1 | Power Switch | SB1 | Charge LED Solder Bridge (disconnected) |
| SB2 | VBUS Sense Solder Bridge (disconnected) | SB3 | Steps ADC3 Leakage Solder Bridge (disconnected) |
| SB4 | ESP32 Reset Solder Bridge (disconnected) | SB5 | SCL Signal Selector Solder Bridge for ATECC608A-MAHDA-T (disconnected) |
| SB6 | SDA Signal Selector Solder Bridge for ATECC608A-MAHDA-T (disconnected) | B1 | Lipo Battery Solder Pads |

3.3 Processor

The DualMCU is powered by a revolution of the Raspberry Pi RP2040 silicon microcontroller (U1) manufactured on a modern 40nm process node is the debut microcontroller from Raspberry Pi and it brings signature values of high performance and ease of use to the microcontroller space.

With a large on-chip memory, six independent banks of 264 KB SRAM, direct memory access, two symmetric Arm® Cortex®-M0+ clocked at 133MHz, deterministic bus fabric and

rich peripheral set augmented with a unique Programmable I/O (PIO) subsystem, Serial wire debug (SWD) available from boot through the integrated JP3 header, a USB 1.1 device interface implemented for uploading code and 2MB of off-chip Flash memory via dedicated QSPI bus on the RP2040, all these outstanding features provide professional users with unrivalled power and flexibility.

The RP2040 controls the peripherals and digital pins, as well as analog pins (A0-A3). The I2C JST connections (SDA) and (SCL) are used for connecting to the onboard peripherals and are optionally pulled up with a 4.7 kΩ resistor.

### 3.4 Wi-Fi/Bluetooth® Connectivity

Wi-Fi and Bluetooth® connectivity is provided by the Espressif ESP32 WROOM-32 (U2) module, clocked up 240 MHz and 512 KB on-chip SRAM and 4 MB integrated SPI Flash memory. Multiple programmable GPIOs, ADC, I2C, UART, Capacitive Touch Sensor and Two-Wire Automotive Interface (TWAI®), compatible with ISO11898-1 (CAN Specification 2.0) are exposed via JP2 y JP1 pin headers.

The ESP32 module includes a dual core Xtensa LX6 CPU that can also be programmed independently of the RP2040 through the USB mechanical selector (SW2) via a CH340C USB bus convert IC (U4).

### 3.5 External Memory

The RP2040 (U1) has access to an external 2 MB of flash memory (U3) via a QSPI interface. All the application code and data must be stored in an external flash chip. Six dedicated pins are used to communicate with a separate QSPI flash, using execute-in-place (XIP) technology to run code directly from flash without needing to copy it to RAM first.

### 3.6 Cryptography IC

Optionally, the ATECC608A can be incorporated and used for authentication tasks, which can be accessed by both MCUs through selection pads for I2C communication. The ATECC608A Cryptographic IC (U7) provides secure boot capabilities alongside SHA and AES-128 encryption/decryption support for security in Smart Home and Industrial IoT (IIoT) applications.

### 3.7 RGB LED

The common anode RGB LED (L5) is also controlled by the ESP32 module such that the LED is on when the digital state is LOW and off when the digital state is HIGH.

3.8 WS2812B LED

The WS2812B (L4) led is an intelligent control RGB NeoPixel for full color indication integrated in a 3535 component package and is connected to RP2040 at pins GPIO16 for power enable and GPIO17 for intelligent digital port (DI).

3.9 MicroSD Card Socket

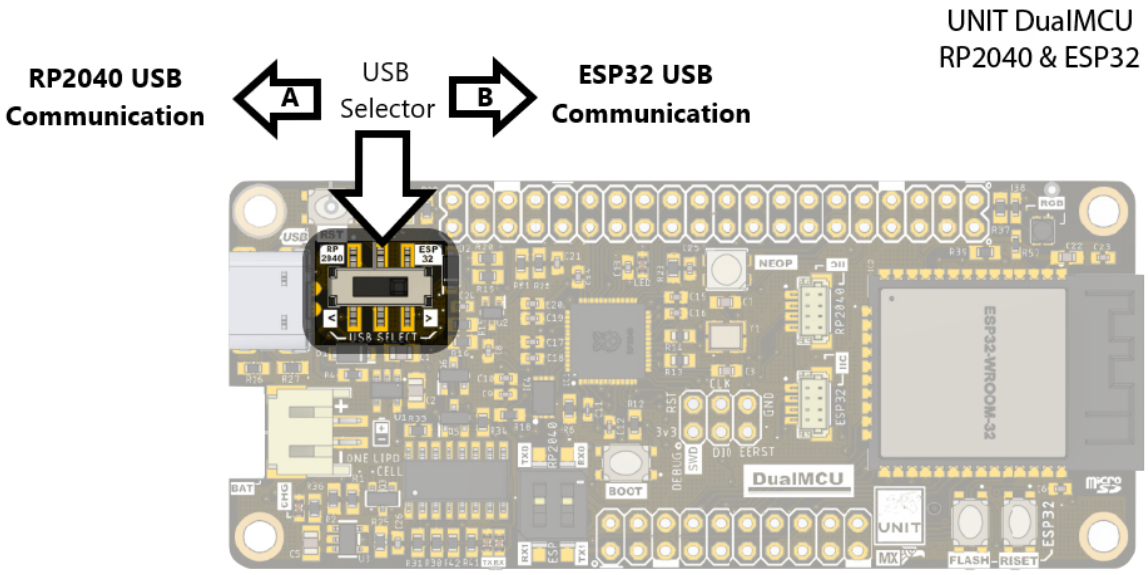
The DualMCU interfaces a microSD card with ESP32 using the microSD card socket connected via an SPI interface. Using a microSD card becomes very handy for applications where we need to store files that are larger than the size of SPIFF (flash file system) of ESP32.

3.10 Power Switch

The toggle power switch (SW1) provides the utility to turn the DualMCU ON or OFF and allows battery charging even when the power switch is in the off position.

3.11 Mechanical selector for the USB Communication

The mechanical USB selector (SW2) allows the programming and serial USB communication from one MCU at the time with only one USB Type-C connector.



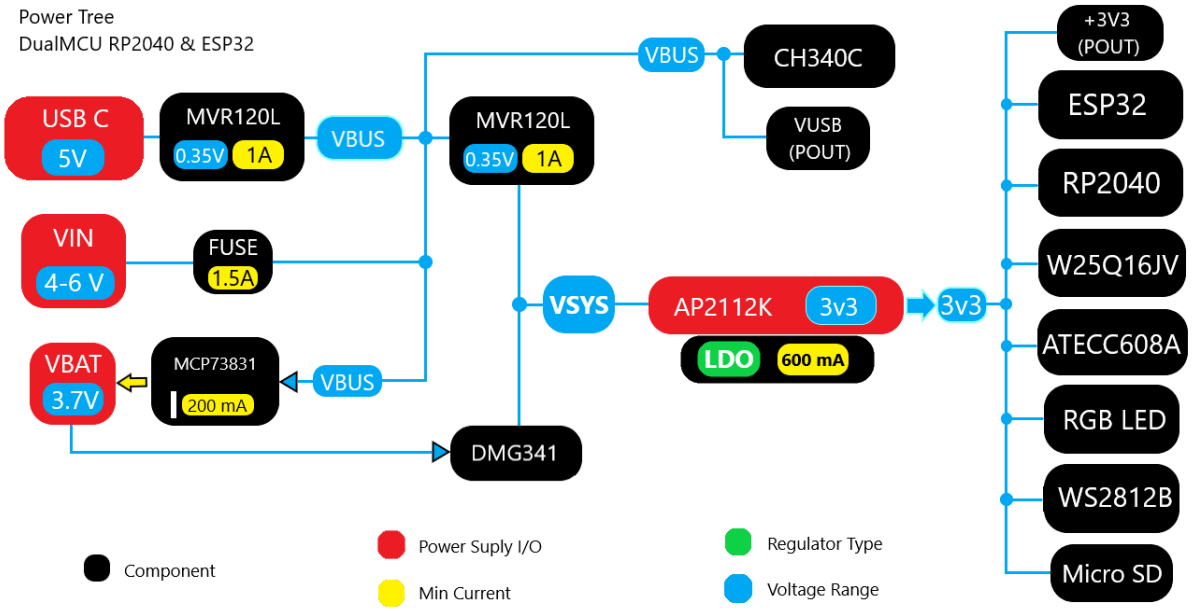

Positions for the mechanical USB selector

### 3.12 DIP Switch for UART Communication

The DIP switch (SW3) allows the UART communication between both MCUs, connecting RP2040-UART0 (Tx_GPIO0 & Rx_GPIO1) to ESP32-UART1 (RX_IO17 & TX_IO16) respectively.

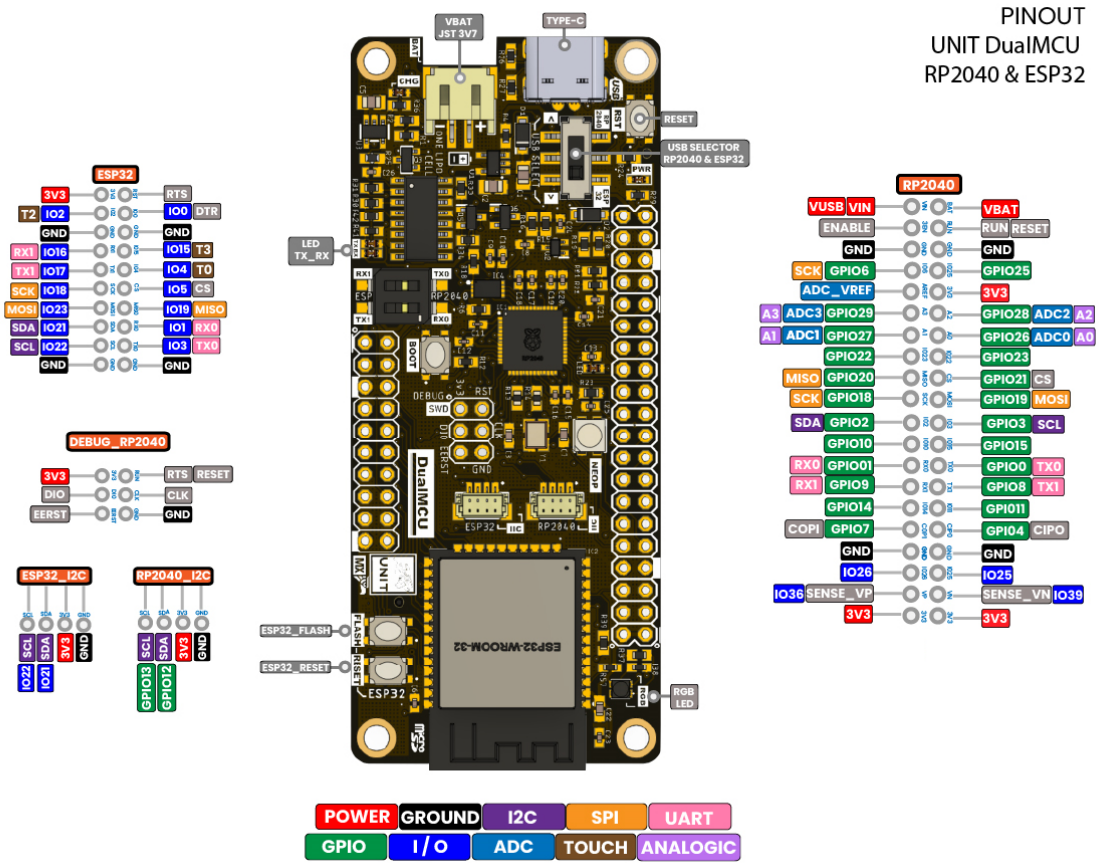
### 3.13 Power Tree

The DualMCU can be powered by either the USB-C port (J1) or alternatively via VIN on JP1. The VIN pin also delivers the USB voltage from J1 (approx 4.8 V) when the external power supply on the VIN does not exist. An AP2112K LDO Voltage Regulator provides 3V3 to the RP2040 and ESP32 microcontrollers and all other peripherals. The maximum voltage supply for VIN is 6 V and the nominal current of AP2112K is 600 mA.




DualMCU RP2040 & ESP32 Power Tree

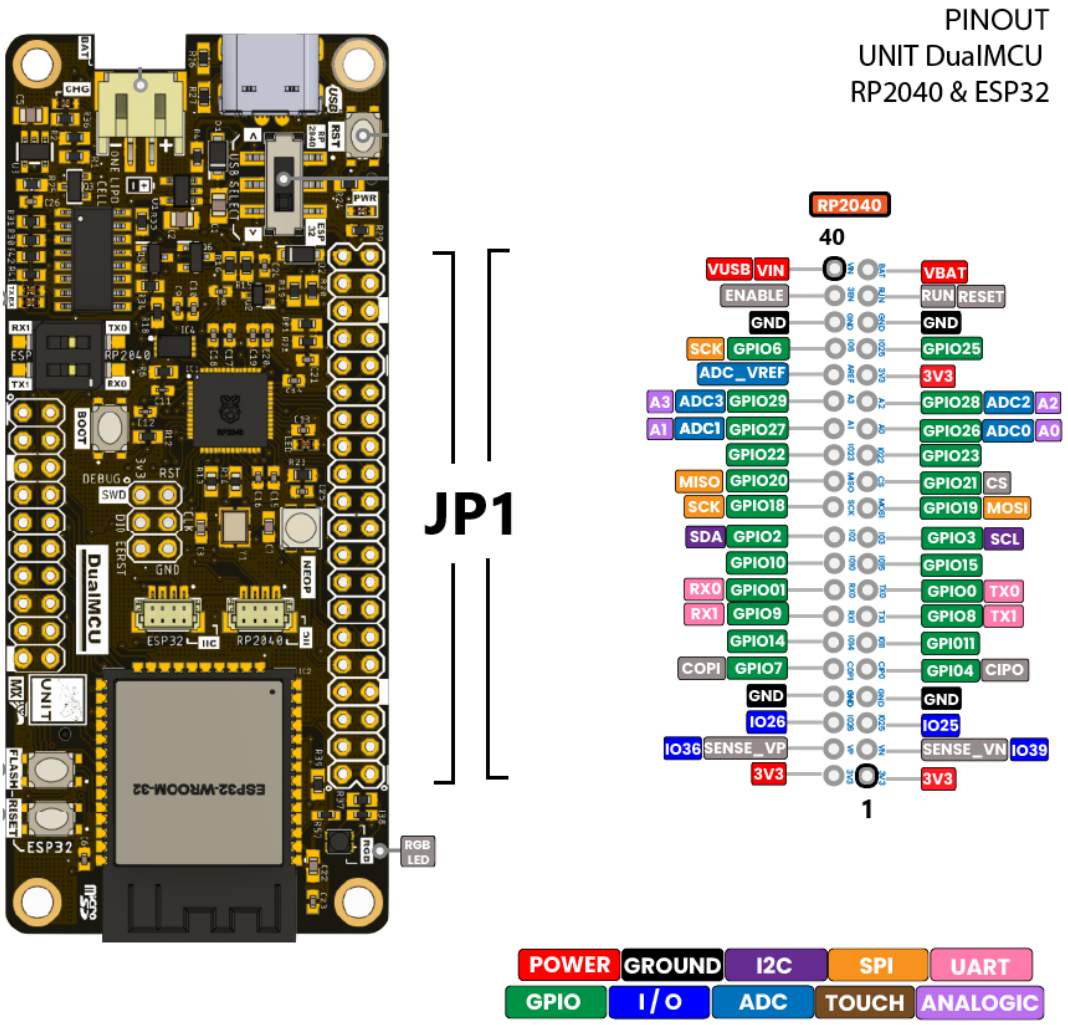
## 4 Connector Pinouts




DualMCU RP2040&ESP32 General Pinout

### 4.1 J1 USB-C

| Pin | Function | Type | Description |
|---|---|---|---|
| 2 , 11 | VBUS | Power | 5V USB Power |
| A7 , B7 | D- | Differential | USB differential data- |
| A6 , A6 | D+ | Differential | USB differential data+ |
| 1 , 12 | GND | Power | Ground |




DualMCU RP2040&ESP32 JP1 Pinout

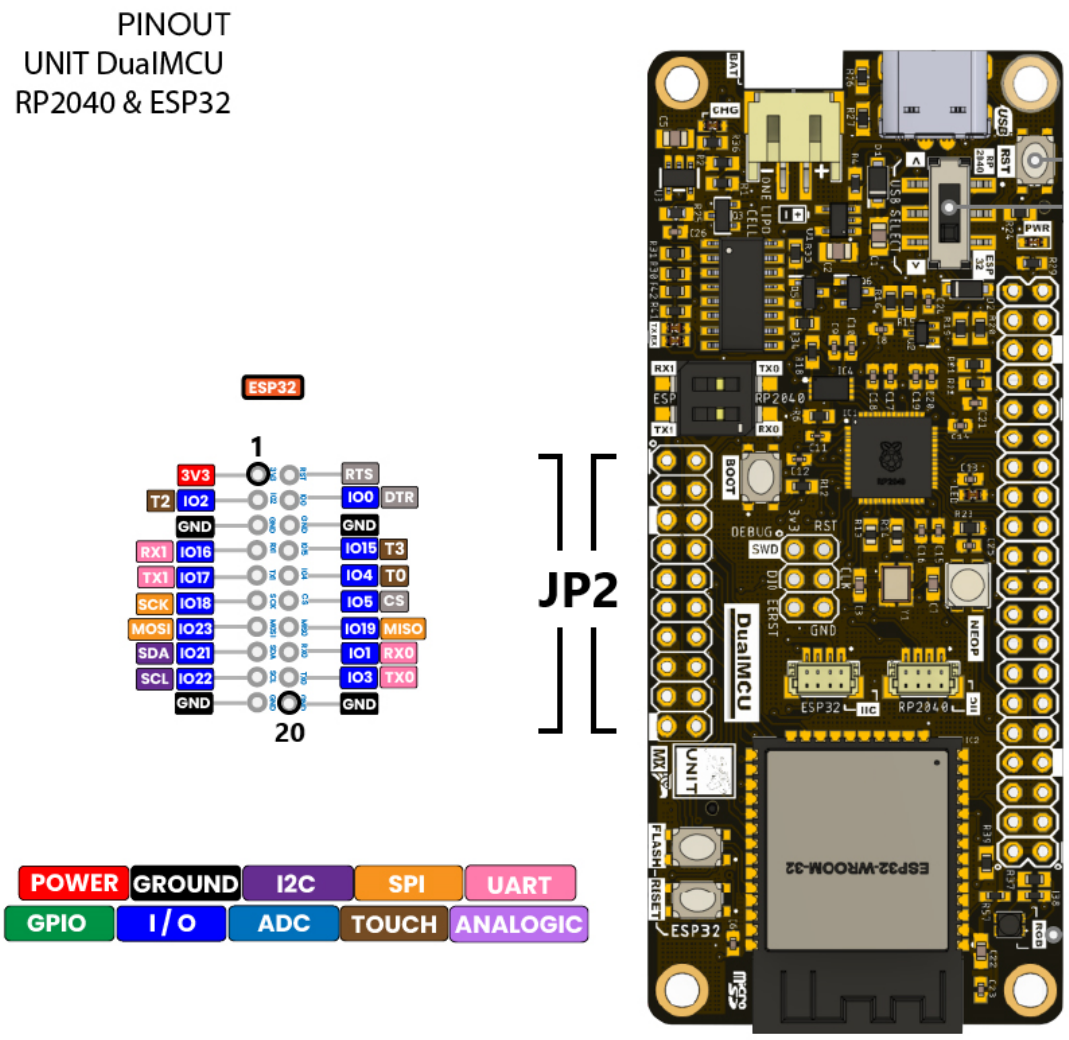
4.2 JP1

| Pin | Function | Type | Description |
|---|---|---|---|
| 1 | 3v3 | Power | 3.3V Power |
| 2 | 3v3 | Power | 3.3V Power |
| 3 | VN | Analog | ESP32 Sense Vp, ADC1-CH3, RTC_GPIO3, GPIO39 |
| 4 | VP | Analog | ESP32 Sense Vn, ADC1-CH0 RTC_GPIO0, GPIO36 |
| 5 | IO25 | Analog, RTC, Digital | ESP32: DAC1/ ADC2-CH8/ RTC_GPIO6/ GPIO25 |

| | | | |
|---|---|---|---|
| 6 | IO26 | Analog, RTC, Digital | ESP32: DAC2/ ADC2-CH9/ RTC_GPIO7/ GPIO26 |
| 7 | GND | Power | Ground |
| 8 | GND | Power | Ground |
| 9 | CIPO | Digital | RP2040: digital pin GPIO4/ Controller In, Peripheral Out |
| 10 | COPI | Digital | RP2040: digital pin GPIO7/ Controller Out, Peripheral In |
| 11 | IO11 | Digital | RP2040: digital pin GPIO11 |
| 12 | IO14 | Digital | RP2040: digital pin GPIO14 |
| 13 | TX1 | Digital | RP2040: digital pin GPIO8/ UART1_TX_PIN |
| 14 | RX1 | Digital | RP2040: digital pin GPIO9/ UART1_RX_PIN |
| 15 | TX0 | Digital | RP2040: digital pin GPIO0/ UART0_TX_PIN |
| 16 | RX0 | Digital | RP2040: digital pin GPIO1/ UART0_RX_PIN |
| 17 | IO15 | Digital | RP2040: digital pin GPIO15 |
| 18 | IO10 | Digital | RP2040: digital pin GPIO10 |
| 19 | IO3 | Digital | RP2040: digital pin GPIO03/SCL_1 |
| 20 | IO2 | Digital | RP2040: digital pin GPIO02/SDA_1 |
| 21 | MOSI | Digital | RP2040: digital pin GPIO19/SPI |
| 22 | SCK | Digital | RP2040: digital pin GPIO18/SPI |
| 23 | CS | Digital | RP2040: digital pin GPIO21/SPI |
| 24 | MISO | Digital | RP2040: digital pin GPIO20/SPI |
| 25 | IO23 | Digital | RP2040: digital pin GPIO23 |

| | | | |
|---|---|---|---|
| 26 | IO22 | Digital | RP2040: digital pin GPIO22 |
| 27 | A0 | Analog/Digital | RP2040: analog pin GPIO26 |
| 28 | A1 | Analog/Digital | RP2040: analog pin GPIO27 |
| 29 | A2 | Analog/Digital | RP2040: analog pin GPIO28 |
| 30 | A3 | Analog/Digital | RP2040: analog pin GPIO29 |
| 31 | 3v3 | Power | 3.3V Power |
| 32 | AREF | Analog | NC |
| 33 | IO25 | Digital | RP2040: digital pin GPIO25 |
| 34 | IO6 | Digital | RP2040: digital pin GPIO06 |
| 35 | GND | Power | Ground |
| 36 | GND | Power | Ground |
| 37 | RUN/RESET | Digital | RP2040: Reset |
| 38 | 3EN | Digital | AP2112K: Enable |
| 39 | VBAT | Power | Battery Voltage Output |
| 40 | VUSB | Power | VUSB: Output Pin / Input Pin |

Each UART can be connected to a number of GPIO pins as defined in the GPIO muxing table in Section 2.19.2 of RP2040 official datasheet: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf .

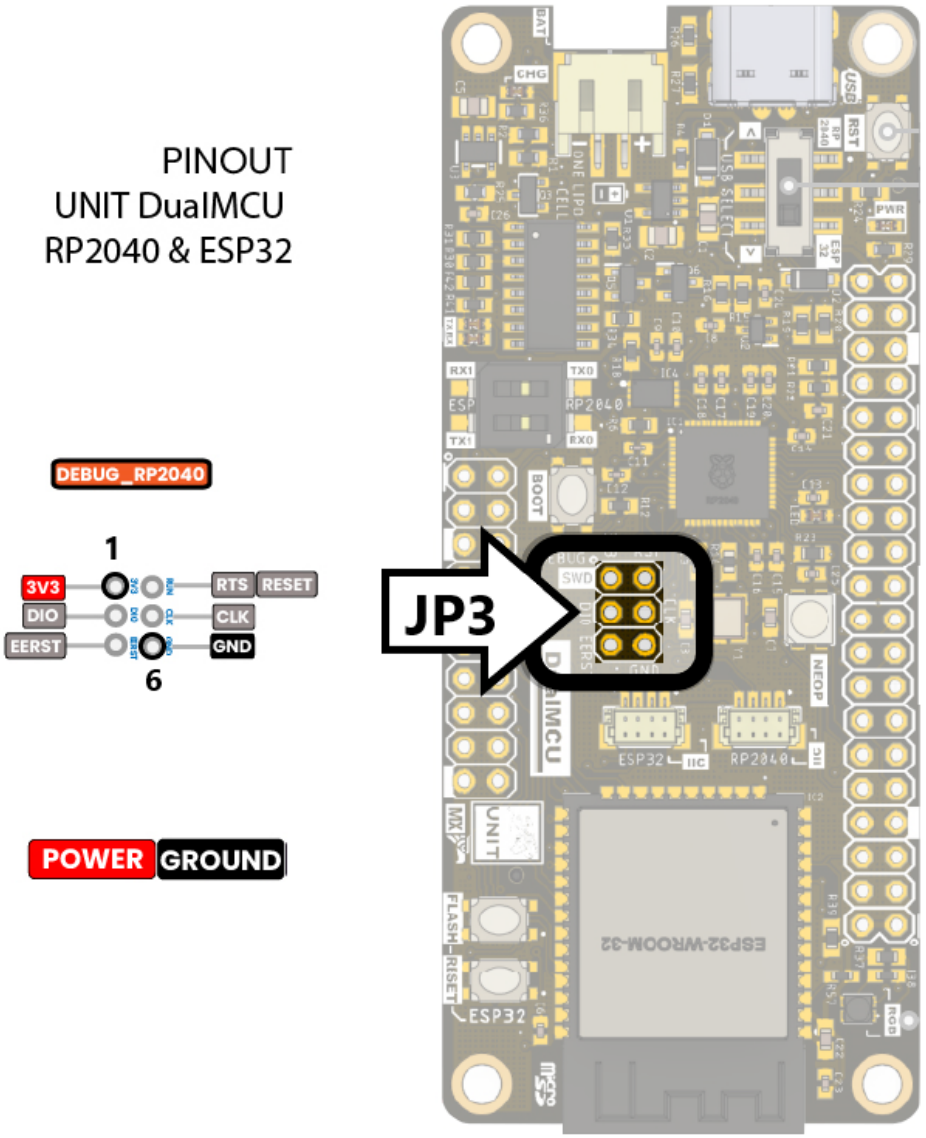



DualMCU RP2040&ESP32 JP2 Pinout

4.3 JP2

| Pin | Function | Type | Description |
|---|---|---|---|
| 1 | 3v3 | Power | 3.3V Power |
| 2 | RST | Digital | ESP32: Reset |
| 3 | IO2 | Analog, Digital | ESP32: ADC2-CH2, GPIO2, TOUCH T2, RTC_GPIO12 |
| 4 | IO0 | Digital | ESP32:DTR, GPIO0 |
| 5 | GND | Power | Ground |
| 6 | GND | Power | Ground |

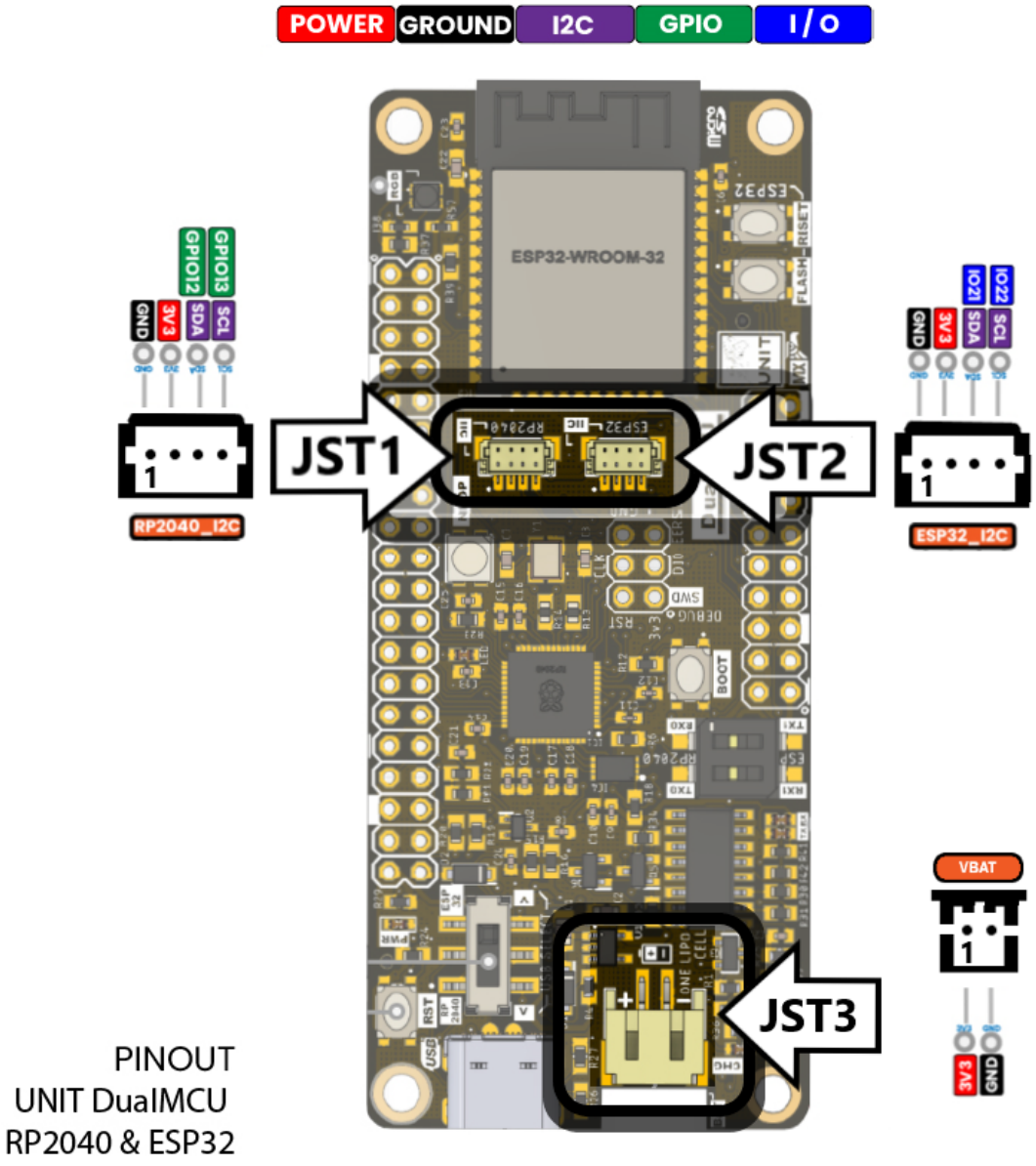
| | | | |
|---|---|---|---|
| 7 | RX1 | Digital | ESP32: UART2 - RXD, GPIO16 |
| 8 | IO15 | Analog, Digital | ESP32: ADC2-CH3, GPIO15, TOUCH T3, RTC_GPIO13 |
| 9 | TX1 | Digital | ESP32: UART2 - TXD, GPIO17 |
| 10 | IO4 | Analog, Digital | ESP32: ADC2-CH0, GPIO15, TOUCH T0, RTC_GPIO10 |
| 11 | SCK | Digital | ESP32: SPI-SCK, GPIO18 |
| 12 | CS | Digital | ESP32: SPI-CS, GPIO05 |
| 13 | MOSI | Digital | ESP32: SPI-MOSI, GPIO23 |
| 14 | MISO | Digital | ESP32: SPI-MISO, GPIO19 |
| 15 | SDA | Digital | ESP32: I2C-SDA, GPIO21 |
| 16 | RX0 | Digital | ESP32: UART0 - RXD, GPIO1 |
| 17 | SCL | Digital | ESP32: I2C-SCL, GPIO22 |
| 18 | TX0 | Digital | ESP32: UART0 - TXD, GPIO3 |
| 19 | GND | Power | Ground |
| 20 | GND | Power | Ground |




DualMCU RP2040&ESP32 JP3 Pinout

4.4 RP2040 SWD JP3

| Pin | Function | Type | Description |
|---|---|---|---|
| 1 | 3v3 | Power | 3.3V Power |
| 2 | RUN/RESET | Digital | RP2040: Reset |
| 3 | SWDIO | Digital | Serial Wire Debug |
| 4 | SWCLK | Digital | Serial Wire Debug |
| 5 | QSPI_SS | Digital | Bootsel |
| 6 | GND | Power | Ground |




DualMCU RP2040&ESP32 I2C and Battery Connectors Pinout

4.5 RP2040 I2C JST1

| Pin | Function | Type | Description |
|---|---|---|---|
| 1 | GND | Power | Ground |
| 2 | 3v3 | Power | 3.3V Power |
| 3 | SDA | Digital | RP2040: digital pin GPIO12/SDA_0 |
| 4 | SCL | Digital | RP2040: digital pin GPIO13/SCL_0 |

4.6 ESP32 I2C JST2

| Pin | Function | Type | Description |
|---|---|---|---|
| 1 | GND | Power | Ground |
| 2 | 3v3 | Power | 3.3V Power |
| 3 | SDA | Digital | ESP32: I2C-SDA, GPIO21 |
| 4 | SCL | Digital | ESP32: I2C-SCL, GPIO22 |

4.7 JST3 Battery Connector

| Pin | Function | Type | Description |
|---|---|---|---|
| 1 | VBAT | Power | Positive voltage for the optional lipoly battery |
| 2 | GND | Power | Ground |

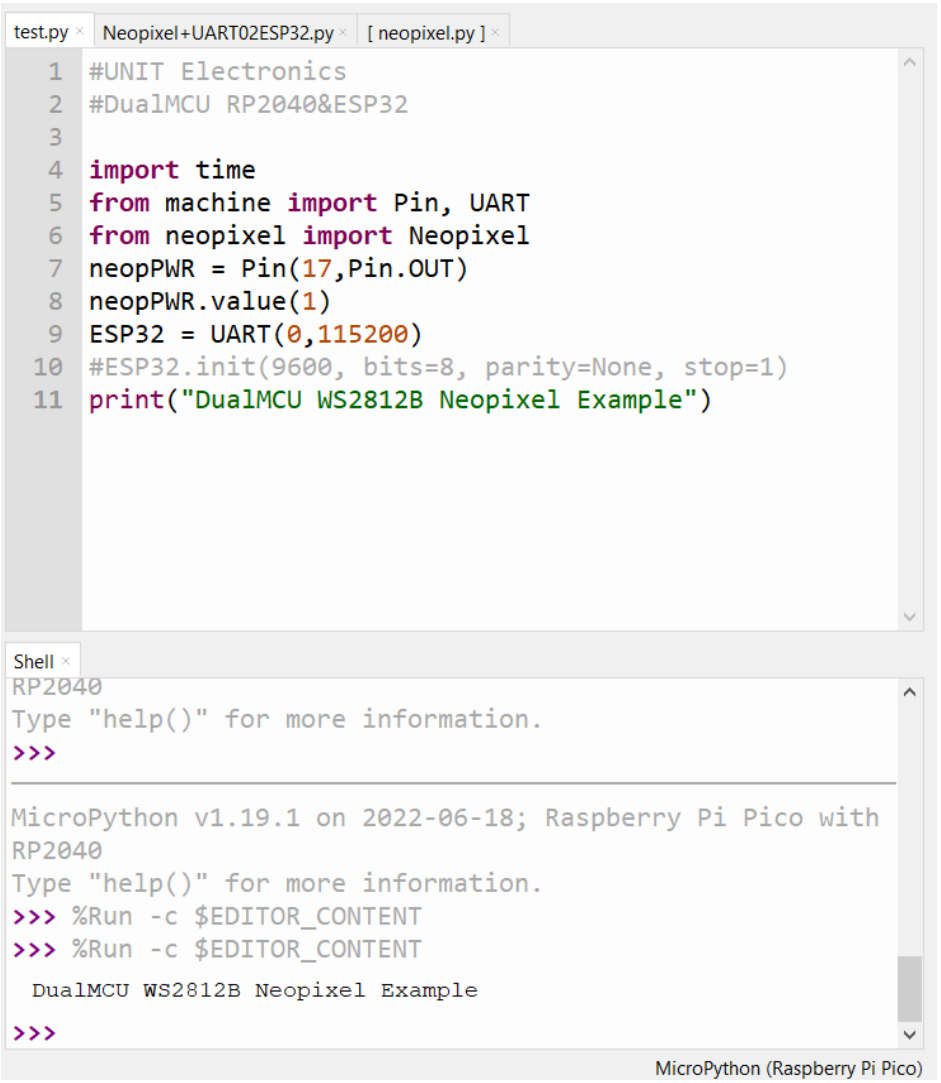
## 5 Board Operation

If you want to program your UNIT® DualMCU while offline you need to install the Thonny[1] or Arduino[2] IDE. To connect the DualMCU to your computer, you'll need a USB-C cable. This also provides power to the board, as indicated by the Power On LED.

### 5.1 Getting Started with Micropython - Thonny IDE

**What is Thonny used for?**

Thonny is a free Python Integrated Development Environment (IDE) that was especially designed with the beginner Pythonista in mind.



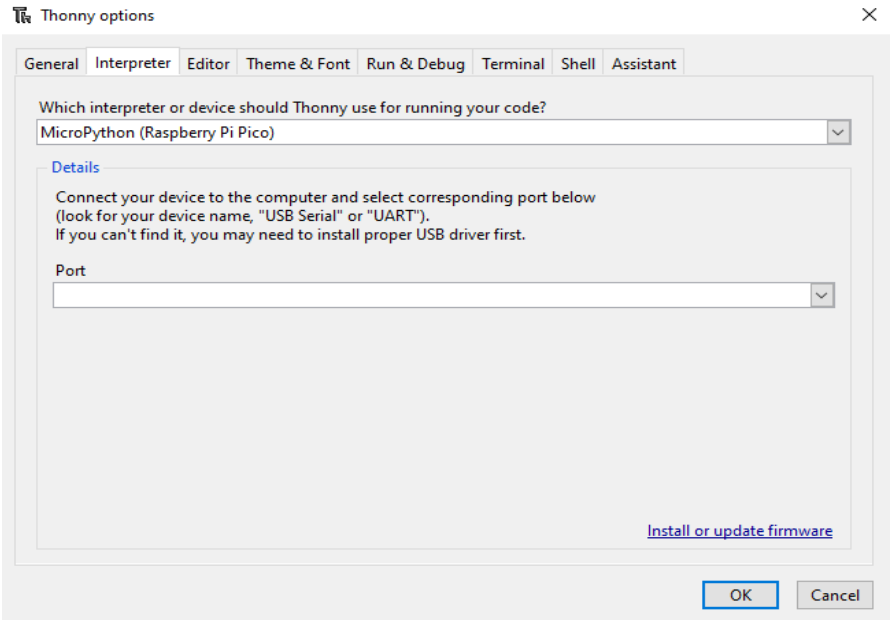

First you need to install the Thonny Environment IDE on your system.

.

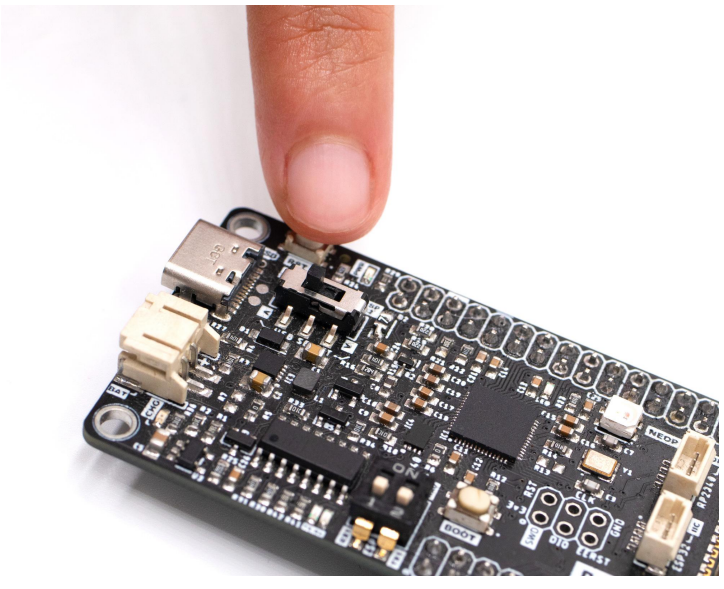
5.2 Steps to run micropython interpreter on the RP2040.

1) Once installed Thonny IDE on your operating system:

**Head to the menu and select: Run > Select Interpreter.... This will open a new window for the Thonny options. In the Interpreter tab, select MicroPython (Raspberry Pi Pico) as the interpreter**.

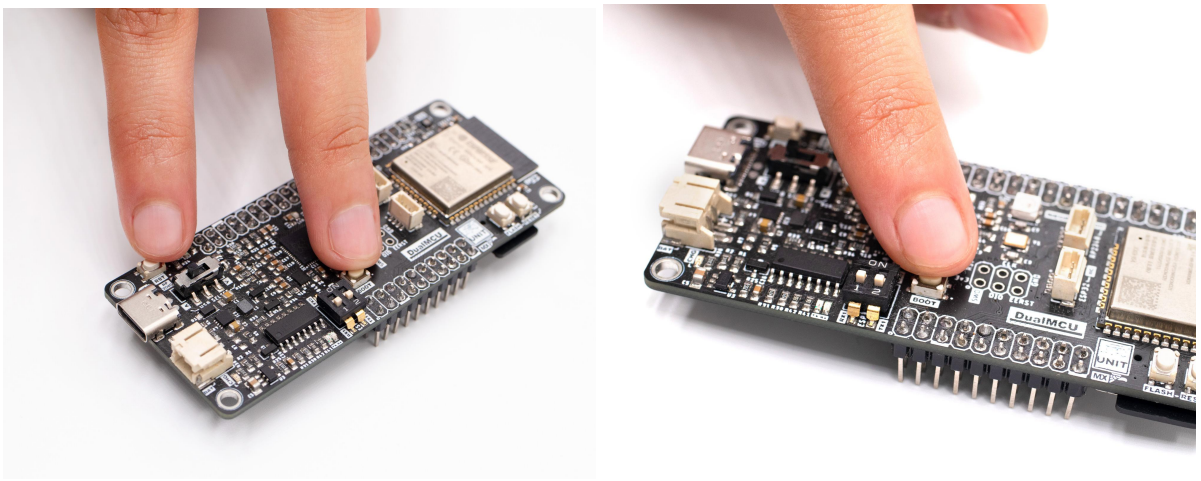



2) To download the Micropython firmware on the RP2040, plug the USB-C cable into the DualMCU, move the mechanical USB selector to the “A” position (see section: *3.11* Mechanical selector for the USB Communication) and press and hold the RP2040 reset button (PB1), you can find it onboard with the label “RST”.



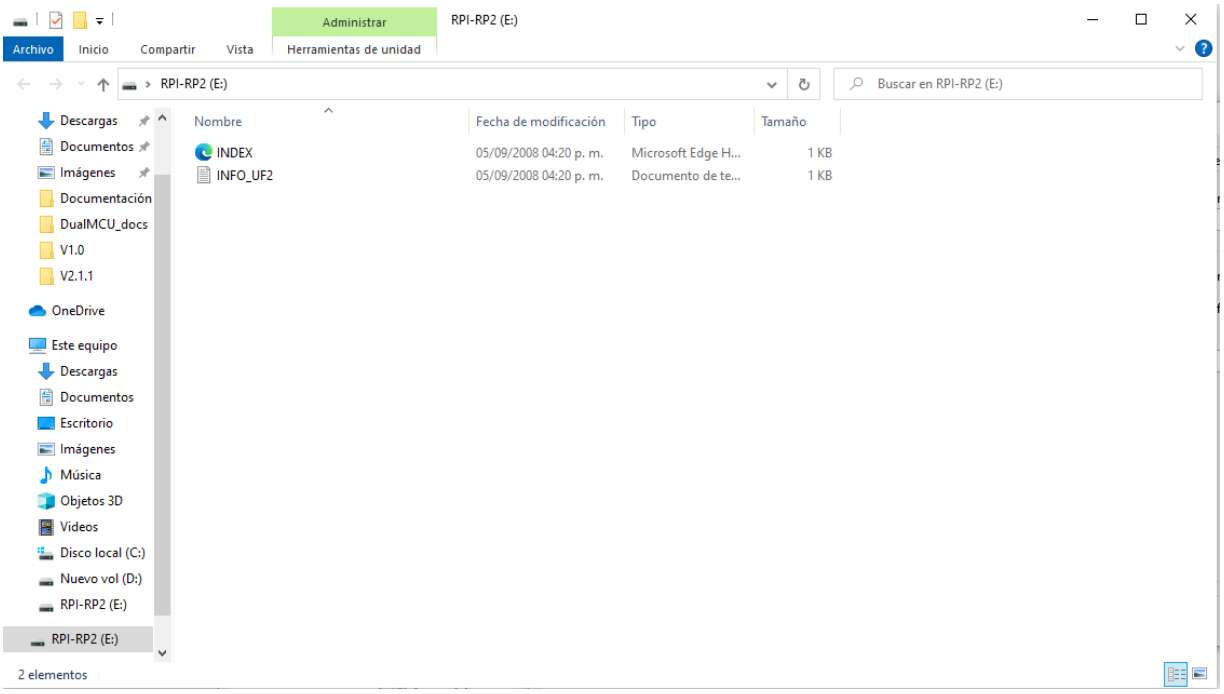
DualMCU “RP2040 Reset Button”

With the other hand, press and hold the RP2040 boot button (PB2) labeled “BOOT” and release the reset button (PB1) “RST”.



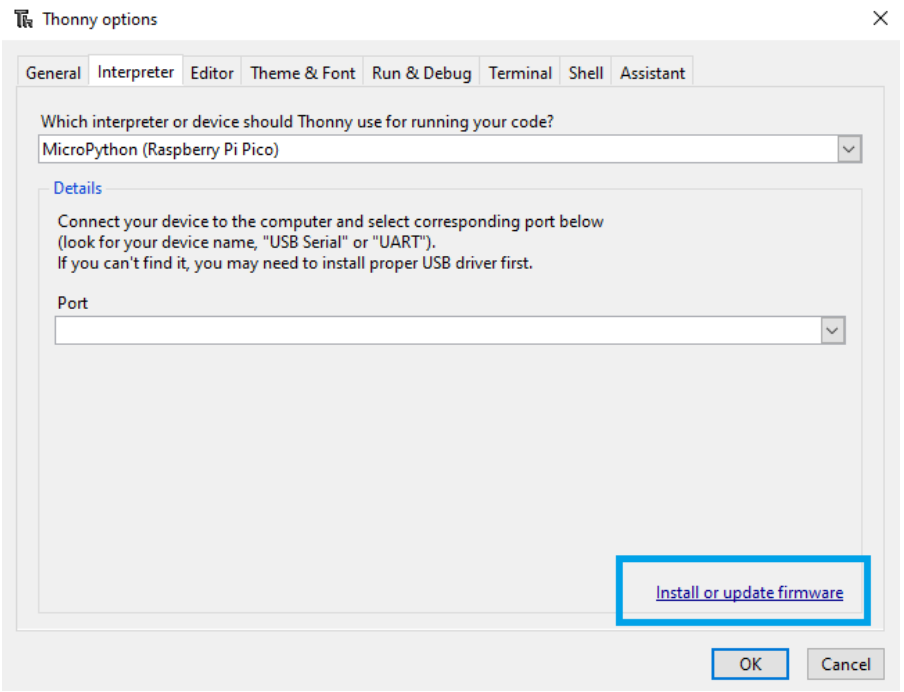
Press and hold the RP2040 “Boot and Reset buttons” to enter bootloader mode

The board should appear as a removable drive called **RPI-RP2**.

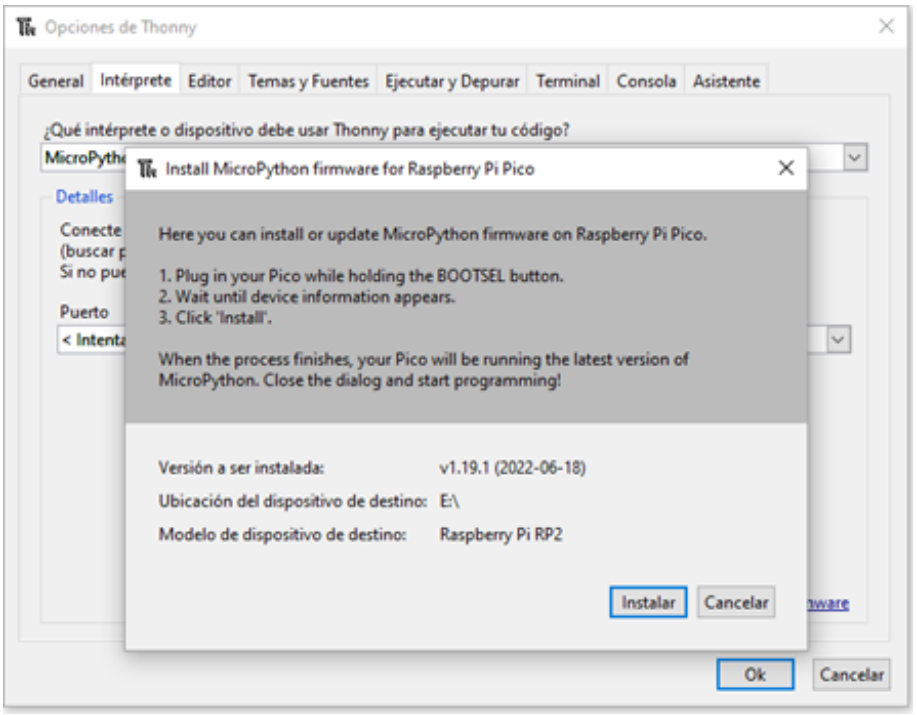



*Removable drive called **RPI-RP2***

3) Go to Thonny IDE and click on *"Install or Update firmware" to start* the firmware download from Raspberry Pi Pico.

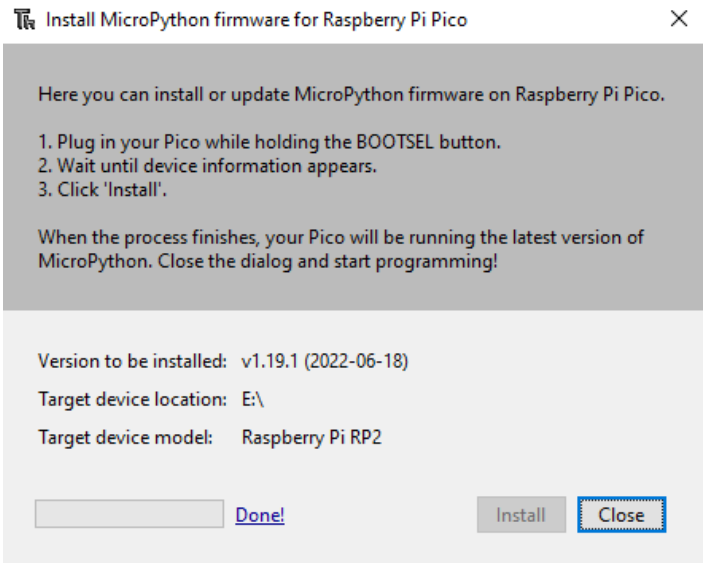



The following window should appear:




4) Then, click on "*install*".

Once the micropython firmware download is finished, the message “Done!” will appear, indicating that the firmware load was done correctly, click on "Close" and the DualMCU ready to start loading our programs on the RP2040 chip.




**Note:**

*You can also download the micropython firmware UF2 file from Raspberry pi source and then drag and drop into the removable drive “**RPI-RP2**”. The board will automatically reboot.*

Click to download the Micropython UF2 File

5.3 Examples (Micropython)

*The Raspberry Pi foundation makes available the necessary tools, documentation, and examples to get started with the RP2040. If you haven't already, look into the documentation on the Pico. We'll use this as a point of reference when using the RP2040 chip into the DualMCU board to fit your needs in this tutorial.*

Raspberry Pi: Pico Python SDK (.pdf)

We'll be using the MicroPython examples from this repo using Thonny IDE.

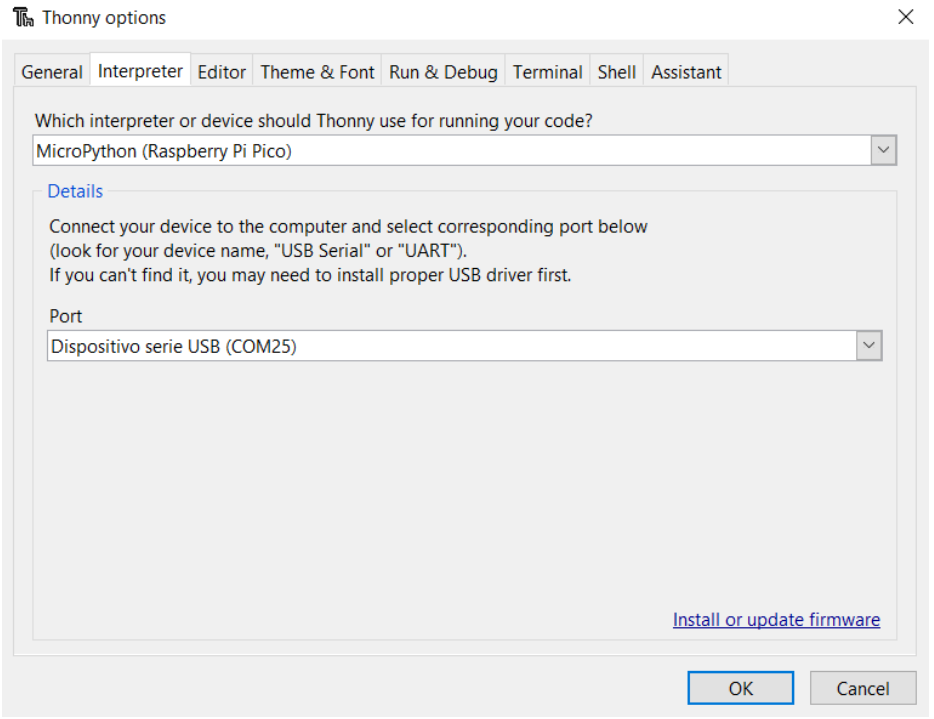
Github: Raspberry Pi - Micropython “Examples”

5.4 Micropython hello world on the RP2040

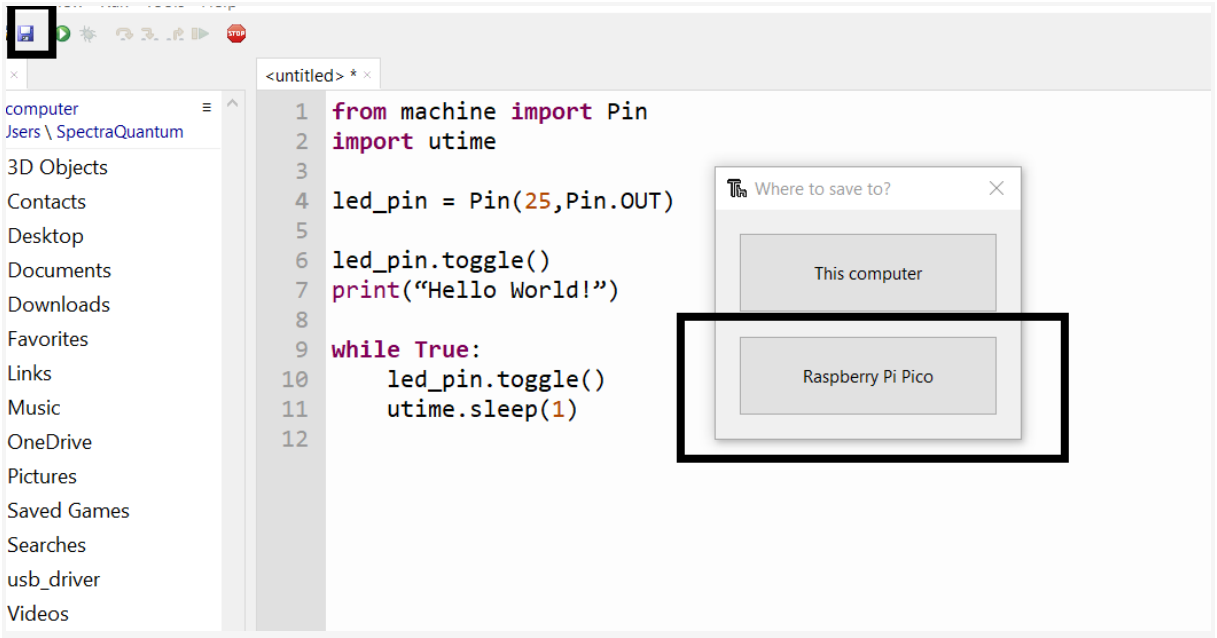
1) In Thonny editor click File->New to open new python file and start writing the code in the editor:

```
from machine import Pin
import utime

led_pin = Pin(25,Pin.OUT)

led_pin.toggle()
print("Hello World!")

while True:
    led_pin.toggle()
    utime.sleep(1)
```

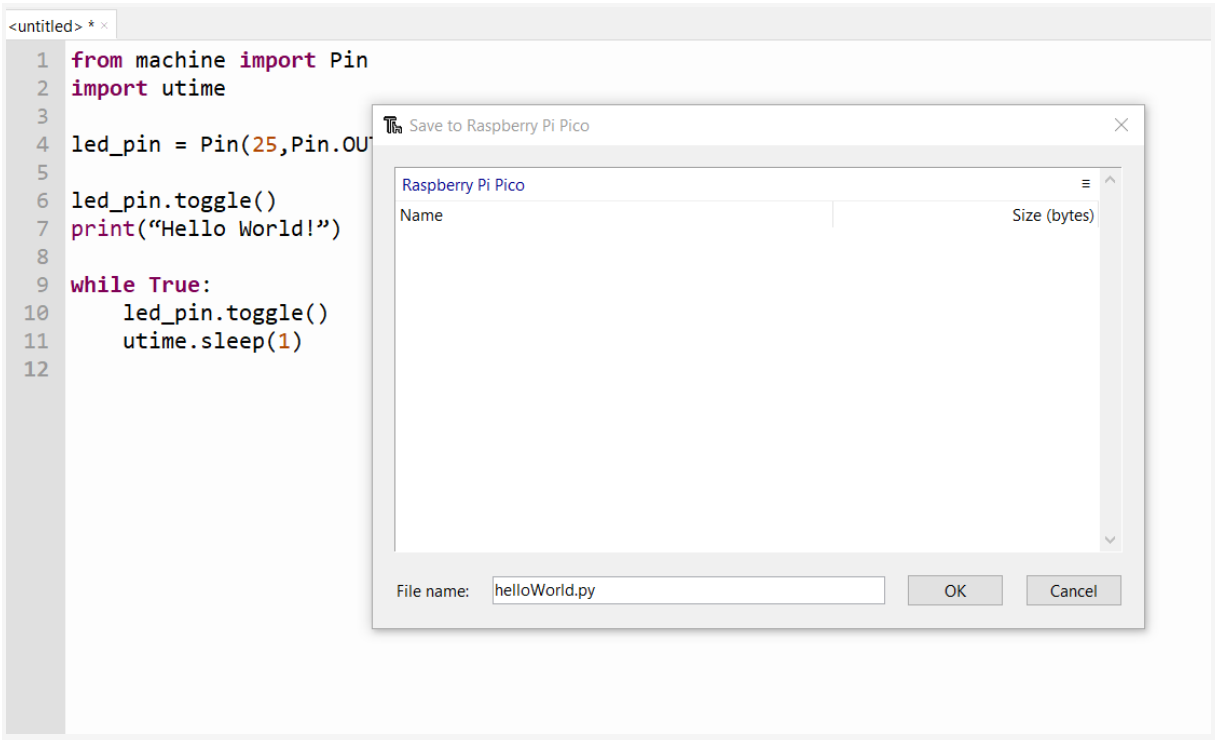
2) In the menu select: Run > Select Interpreter.... This will open a new window for the Thonny options. In the Interpreter tab, select MicroPython (Raspberry Pi Pico) as the interpreter and select the COM port with which your system recognizes the DualMCU for RP2040 (please make sure that the USB selector switch is in position "A", see section 3.11).



3) Head to the menu and select: File -> Save and choose “Raspberry Pi Pico”




4) Save the script with the name: “HelloWorld.py”



5) Hit the "Run current script" button or press (F5) key on the keyboard. In the Shell, you will see the following output:

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040

Type "help()" for more information.

>>> %Run -c $EDITOR_CONTENT

Hello World!
```

And the on-board builtin led (L3) should be blinking!

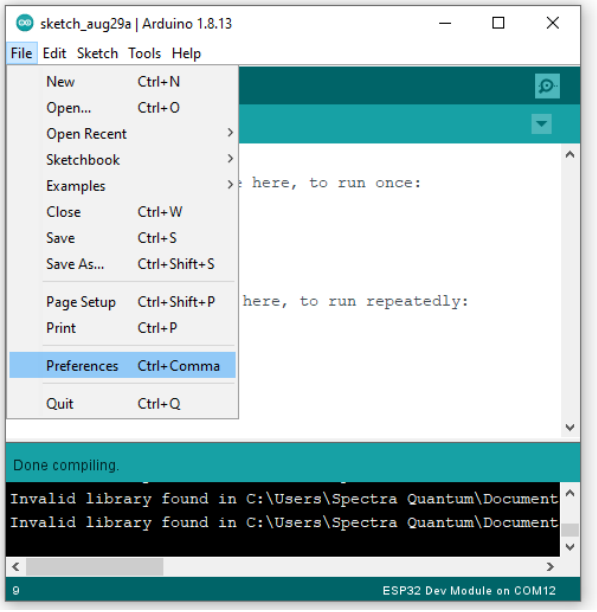
**Note:**

*The ESP32 MCU* can also be programmed from Thonny IDE and micropython, if you want to know how to do it, please visit the following repository:
https://github.com/UNIT-Electronics/DualMCU-RP2040&ESP32-Micropython-Guide

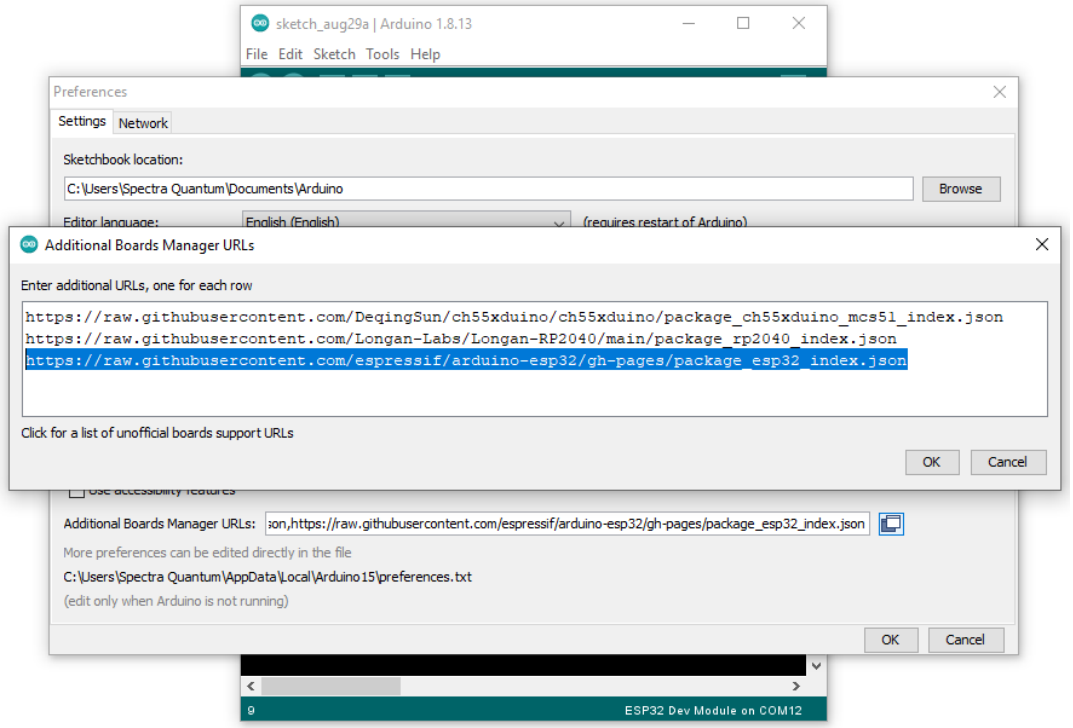
5.5 Getting Started with arduino IDE for ESP32 MCU

The Dual MCU could be programmed with the popular Arduino IDE, the language used is based on C and C++. The open-source Arduino Software makes it easy to write code and upload it to the board. The first thing is to have the Arduino environment installed.

1) Open the IDE and go to File -> Preferences:

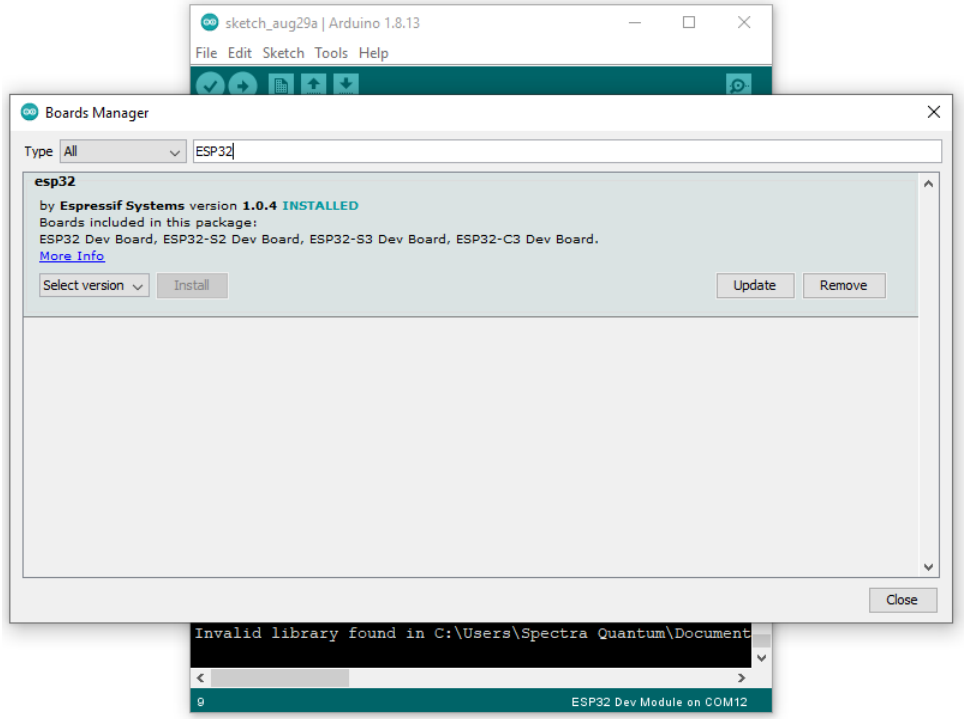



Into “Additional Board Manager URLs” field, add a coma and write the next URL:
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json




Click the "**Ok**" button.

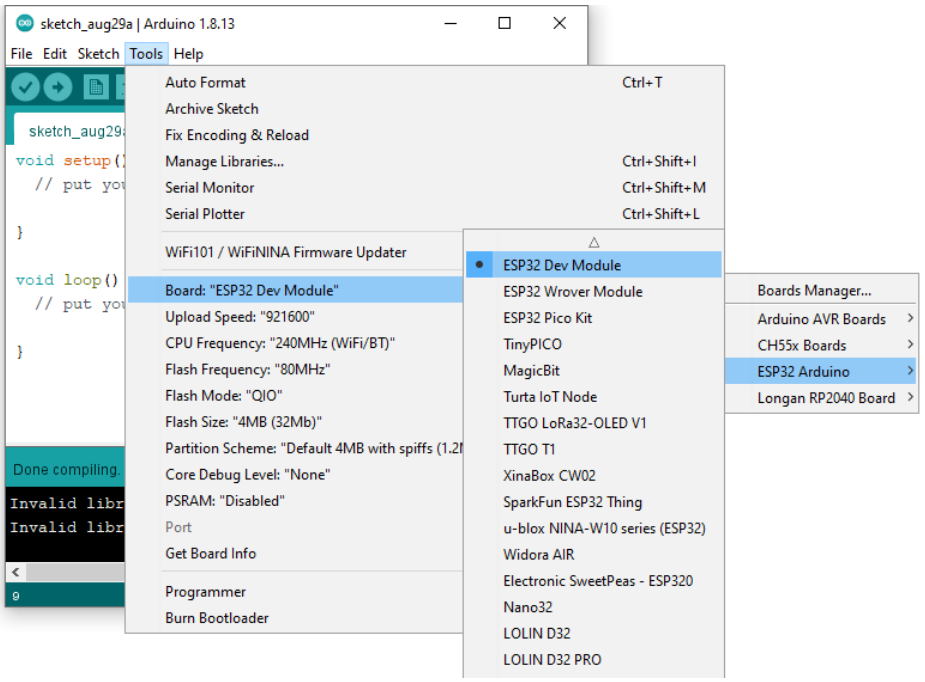
2) Head to the menu and select: ***Tools -> Board -> Boards Manager*** and type "ESP32" in the blank field, select the option "esp32 by Espressif Systems" and press "Install": Wait a moment while the package should be installed and press "Close"



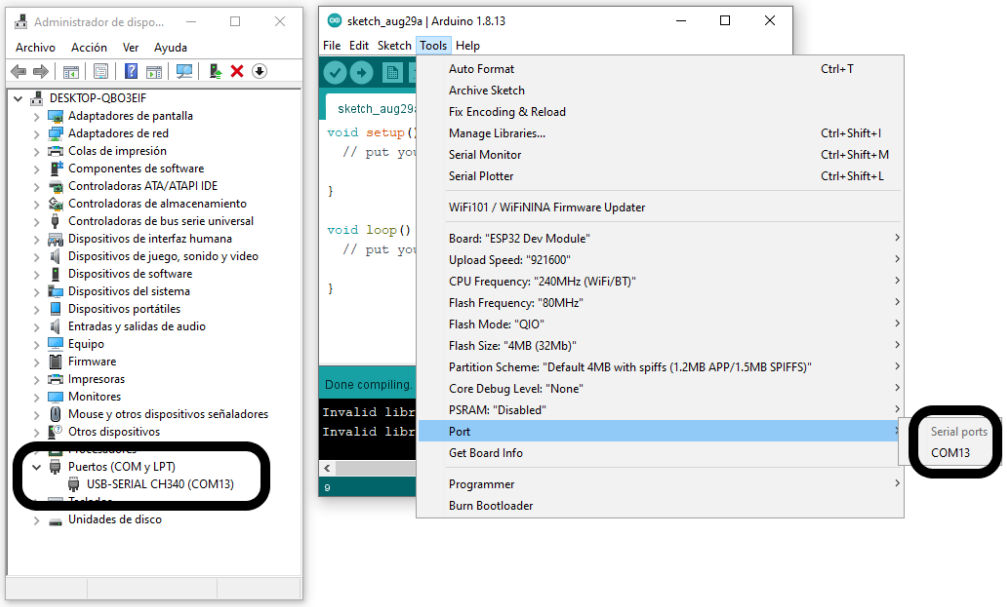
**Now you can plug the Dual MCU into your computer (remember to move the USB selector to position "B" when programming ESP32 MCU, see section: 3.11).**

5.6 Hello world on the ESP32 from Arduino IDE

1) In the "Tools" menu, select: ***Board -> ESP32 Arduino -> ESP32 Dev Module***



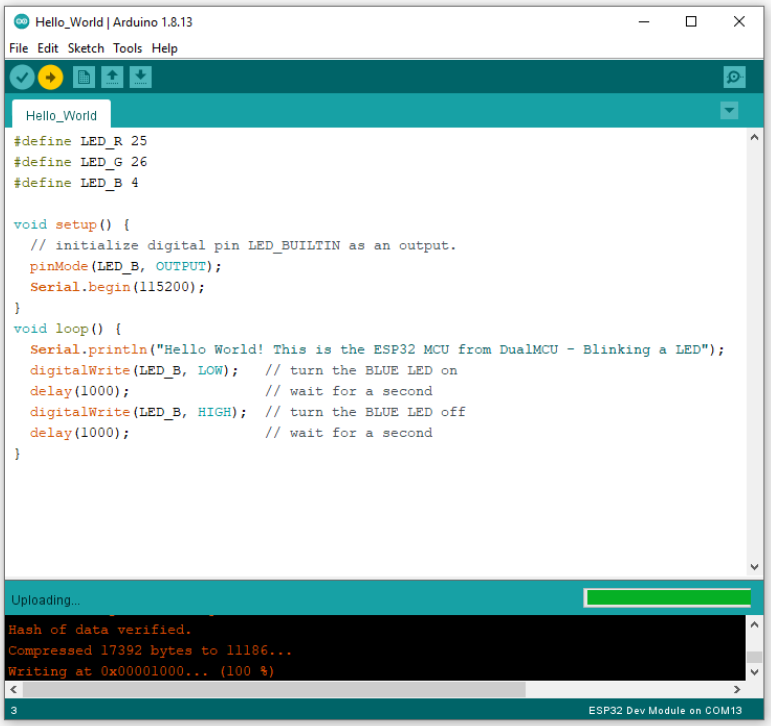

2) Connect the Dual MCU to your computer and go to the "***Tools***" menu and select the COM port for the DualMCU: (if you do not see the COM Port in your Arduino IDE, you need to install the [CH340C Driver])



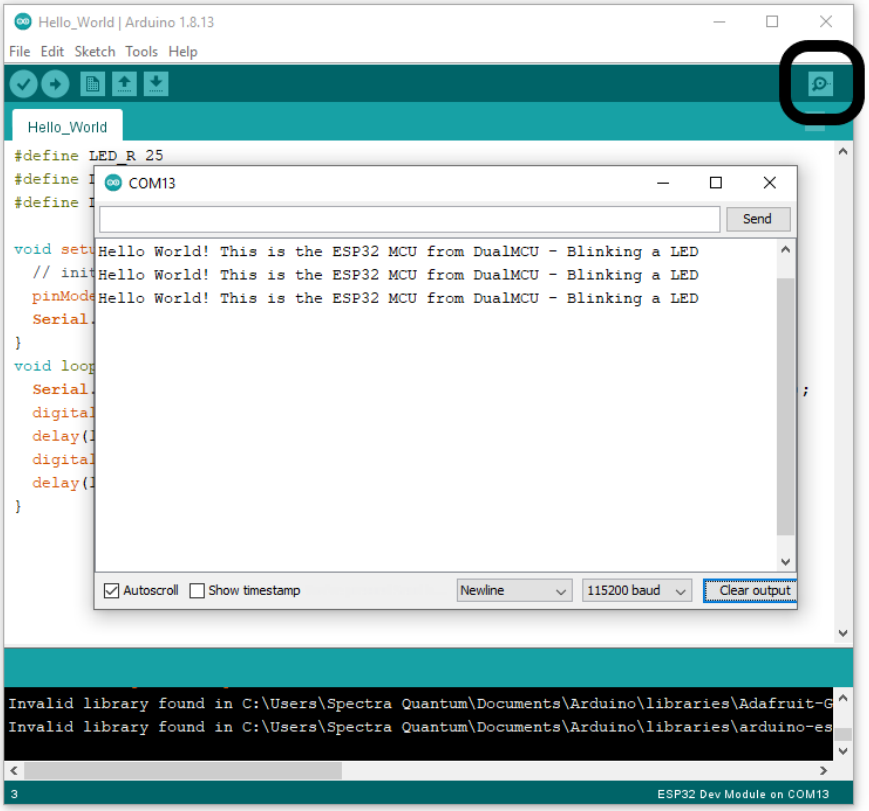
3) In the text editor space start writing the following code:

```
#define LED_R 2

#define LED_G 26

#define LED_B 25

void setup() {

  // initialize digital pin LED_BUILTIN as an output.

  pinMode(LED_G, OUTPUT);

  Serial.begin(115200);

}

void loop() {

  Serial.println("Hello World! This is the ESP32 MCU from DualMCU -
Blinking a LED");

  digitalWrite(LED_G, LOW);    // turn the BLUE LED on

  delay(1000);                 // wait for a second

  digitalWrite(LED_G, HIGH);   // turn the BLUE LED off

  delay(1000);                 // wait for a second

}
```

4) Save the sketch as “Hello World” in the “***File***” menu and select: “***Save***”. Then click the upload button or press “CTRL + U” from keyboard to compile and upload the code to the ESP32 MCU:

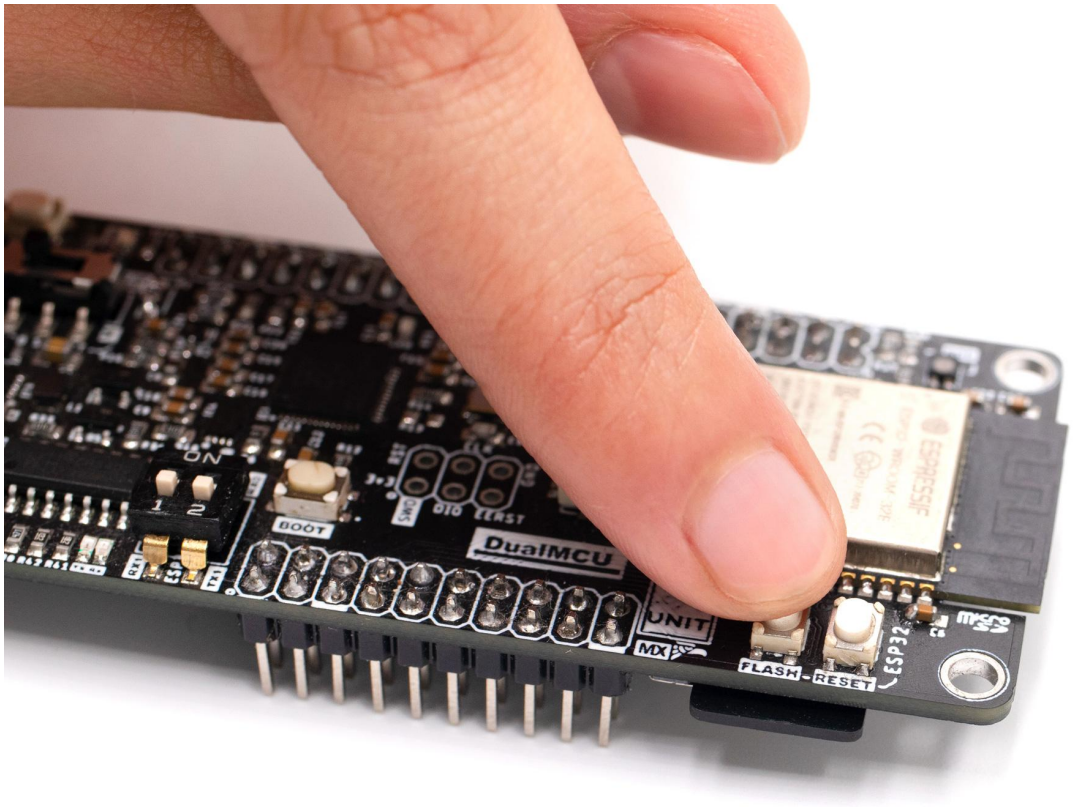



5) Finally, open the arduino serial monitor, you will see a message printing every second and the BLUE LED on the DualMCU blinking!



**Important!:**

Sometimes it is necessary to press the “Flash Button (PB3)” on the DualMCU during code upload.

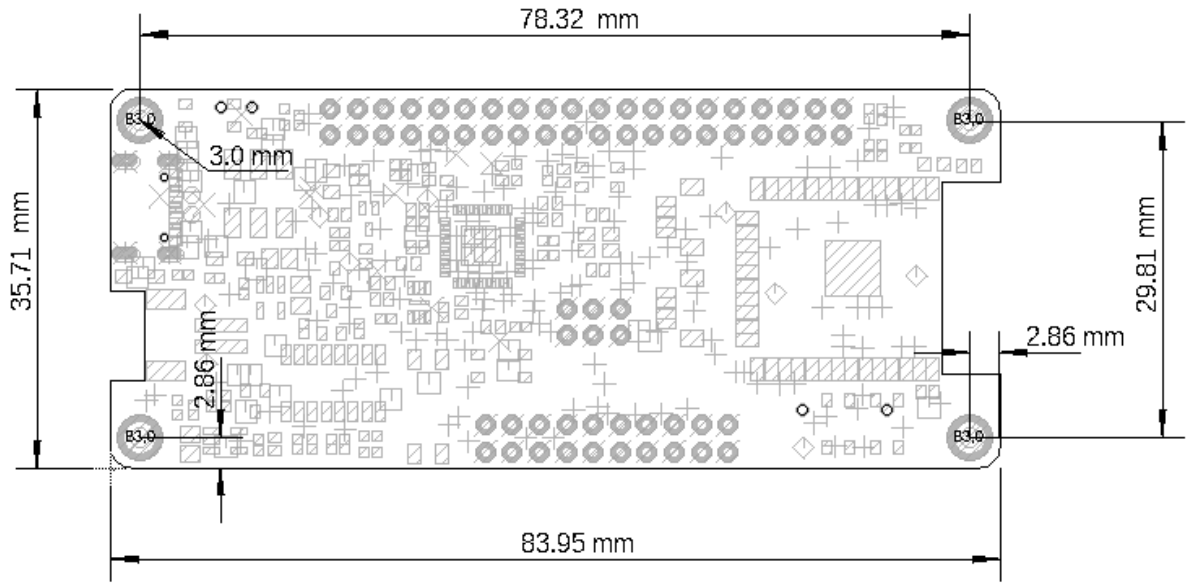



Press the ESP32 “Flash button” to start flashing the Arduino sketch

**Note:**

The RP2040 MCU can also be programmed from Arduino IDE, if you want to know how to do it, please visit the following repository:
https://github.com/UNIT-Electronics/DualMCU-RP2040&ESP32-Arduino Programming

## 6 Mechanical Information




Mechanical dimensions of DualMCU RP2040 + ESP32

## 7 Company Information

| | |
|---|---|
| **Company name** | UNIT Electronics |
| **Company website** | https://uelectronics.com/ |
| **Company Address** | Salvador 19, Cuauhtémoc, 06000 Mexico City, CDMX |

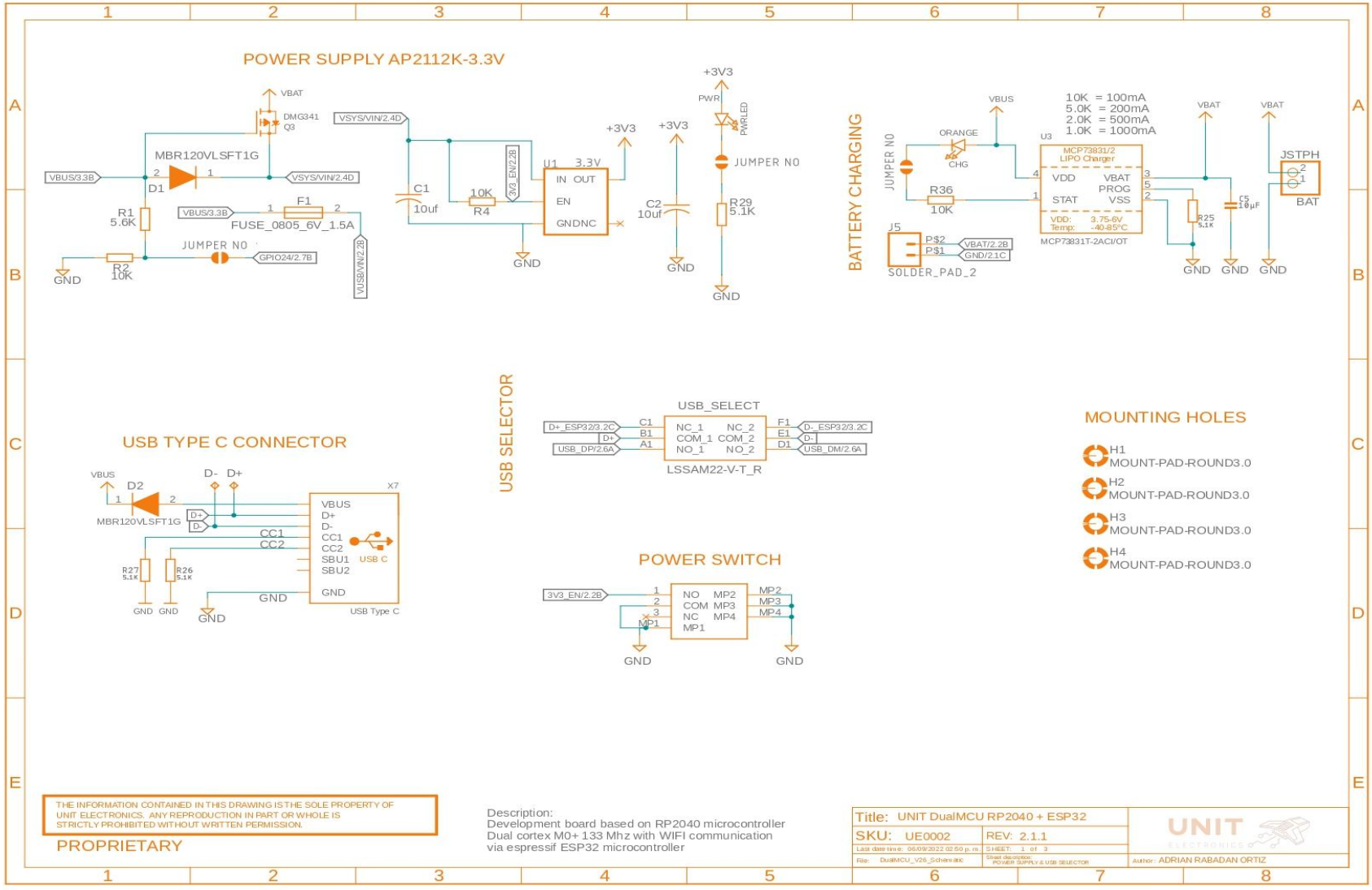
## 8 Reference Documentation

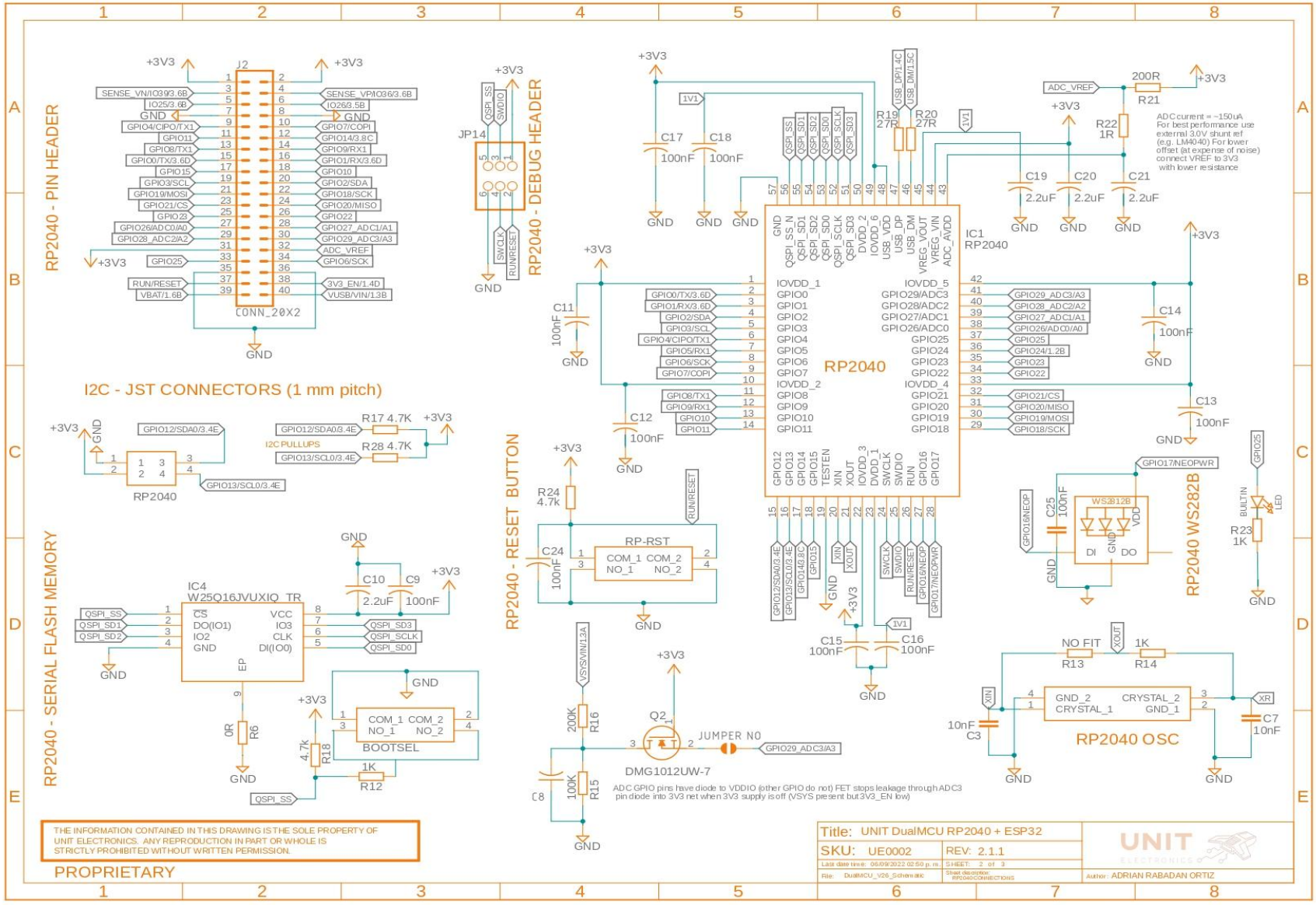
| **Ref** | **Link** |
|---|---|
| UNIT DualMCU Documentation | https://github.com/UNIT-Electronics/DualMCU |
| UNIT DualMCU Examples | https://github.com/UNIT-Electronics/DualMCU/tree/master/examples |
| Getting started with DualMCU RP2040 & ESP32 Micropython | https://github.com/UNIT-Electronics/DualMCU-RP2040&ESP32-Micropython-Guide |

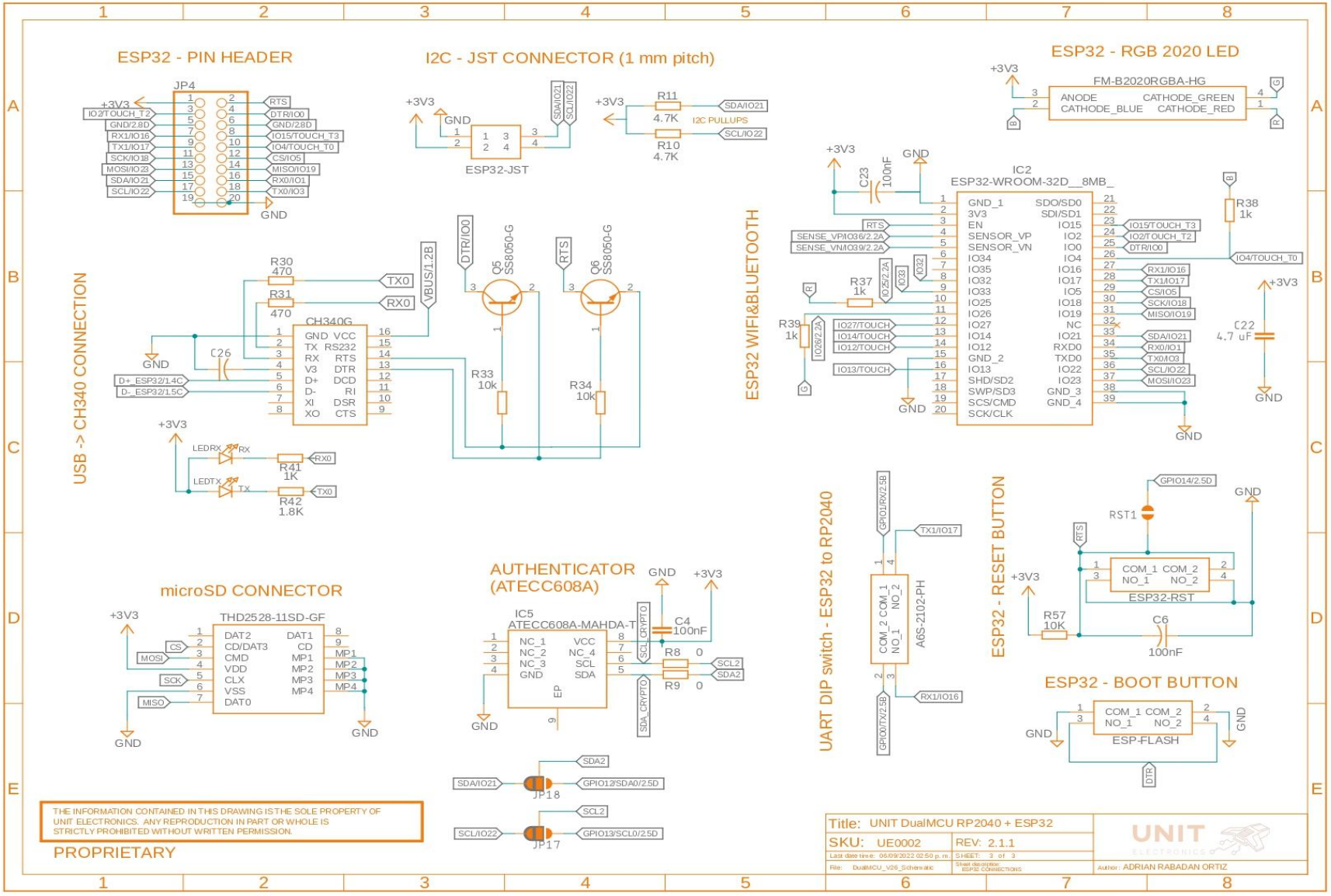
| | |
|---|---|
| Getting started with DualMCU RP2040 & ESP32 Arduino IDE | https://github.com/UNIT-Electronics/DualMCU-RP2040&ESP32-Arduino Programming |
| Getting started with DualMCU RP2040 C/C+ SDK | https://github.com/UNIT-Electronics/DualMCU-RP2040-Getting started with C/C++ SDK |
| Thonny IDE | https://thonny.org/ |
| Arduino IDE | https://www.arduino.cc/en/software |
| CH340 Driver | http://www.wch-ic.com/downloads/CH341SER_ZIP.html |
| Visual Studio Code | https://code.visualstudio.com/download |
| Raspberry Pi Pico RP2040 Documentation | https://www.raspberrypi.com/documentation/microcontrollers/ |
| Raspberry Pi Pico Python SDK | https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-python-sdk.pdf |
| raspberrypi/pico-micropython-examples | https://github.com/raspberrypi/pico-micropython-examples |
| Raspberry Pi Pico C/C++ SDK | https://www.raspberrypi.com/documentation/microcontrollers/c_sdk.html |
| raspberrypi/pico-C/C+-examples | https://github.com/raspberrypi/pico-examples |
| RP2040 Datasheet | https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf |
| ESP32 WROOM 8MB | https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf |

## 9 Appendix

9.1 Schematic




RP2040 - PIN HEADER
RP2040 - DEBUG HEADER
RP2040 - RESET BUTTON
RP2040 - SERIAL FLASH MEMORY
I2C - JST CONNECTORS (1 mm pitch)
RP2040 WS282B
RP2040 OSC
IC1 RP2040
IC4 W25Q16JVUXIQ_TR
DMG1012UW-7
BOOTSEL
JUMPER NO
CONN_20X2
ADC current = ~150uA For best performance use external 3.0V shunt ref (e.g. LM4040) For lower offset (at expense of noise) connect VREF to 3V3 with lower resistance
ADC GPIO pins have diode to VDDIO (other GPIO do not) FET stops leakage through ADC3 pin diode into 3V3 net when 3V3 supply is off (VSYS present but 3V3_EN low)
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF UNIT ELECTRONICS. ANY REPRODUCTION IN PART OR WHOLE IS STRICTLY PROHIBITED WITHOUT WRITTEN PERMISSION.
PROPRIETARY
Title: UNIT DualMCU RP2040 + ESP32
SKU: UE0002
REV: 2.1.1
Author: ADRIAN RABADAN ORTIZ


ESP32 - PIN HEADER
I2C - JST CONNECTOR (1 mm pitch)
ESP32 - RGB 2020 LED
USB -> CH340 CONNECTION
ESP32 WIFI&BLUETOOTH
UART DIP switch - ESP32 to RP2040
ESP32 - RESET BUTTON
ESP32 - BOOT BUTTON
microSD CONNECTOR
AUTHENTICATOR (ATECC608A)
ESP32-JST
ESP32-RST
ESP-FLASH
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF UNIT ELECTRONICS. ANY REPRODUCTION IN PART OR WHOLE IS STRICTLY PROHIBITED WITHOUT WRITTEN PERMISSION.
PROPRIETARY
Title: UNIT DualMCU RP2040 + ESP32
SKU: UE0002
REV: 2.1.1
Author: ADRIAN RABADAN ORTIZ