Raspberry Pi Pico 2 W Quick-Start

By Michael Chandra (mlc27@sfu.ca), Howard Nguyen (hfn2@sfu.ca),
Hongrui Fan (hongruif@sfu.ca)

This guide works on Windows, macOS, and Linux (UNIX-based) systems, and supports both x86_64 (amd64) and ARM architectures.

Introduction

The **Raspberry Pi Pico 2 W** is a powerful and affordable dual-core microcontroller with built-in Wi-Fi, perfect for embedded projects. This guide walks you through:

- Setting up the Pico 2 W as a development board in VS Code.
- Writing and compiling code using the Pico SDK.
- Communicating with peripherals via **SPI** and **I2C**.

What You'll Need

Hardware

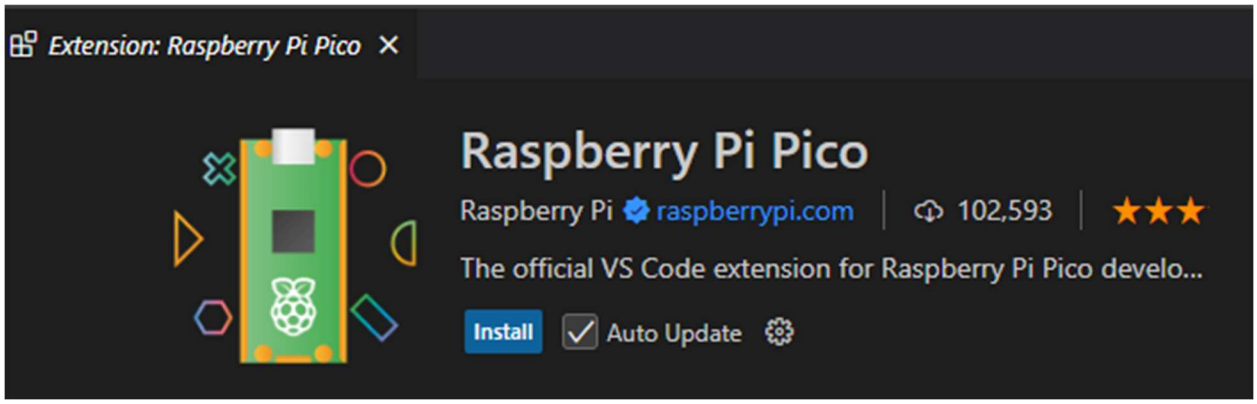
- Raspberry Pi Pico 2 W
- Micro-USB cable

Software

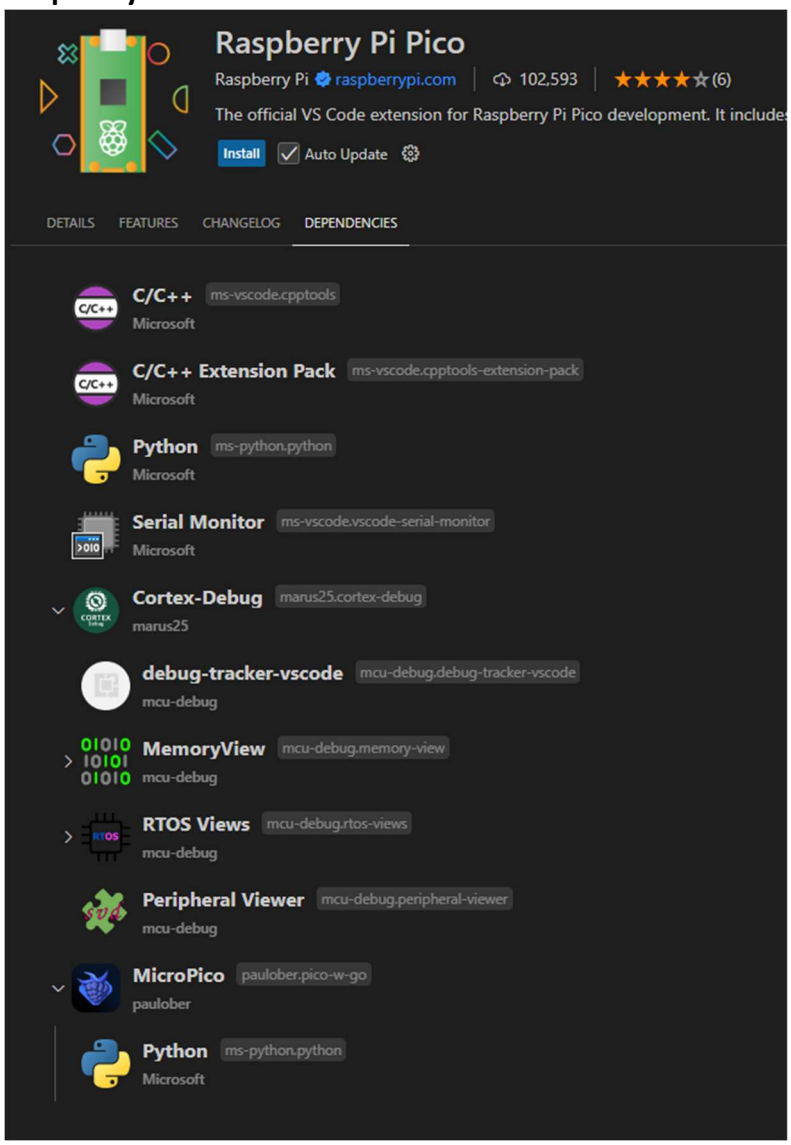
- Visual Studio Code

Installing the Pico SDK and Toolchain via VS Code

1. Search **Raspberry Pi Pico** in VS Code's Extensions Tab and install



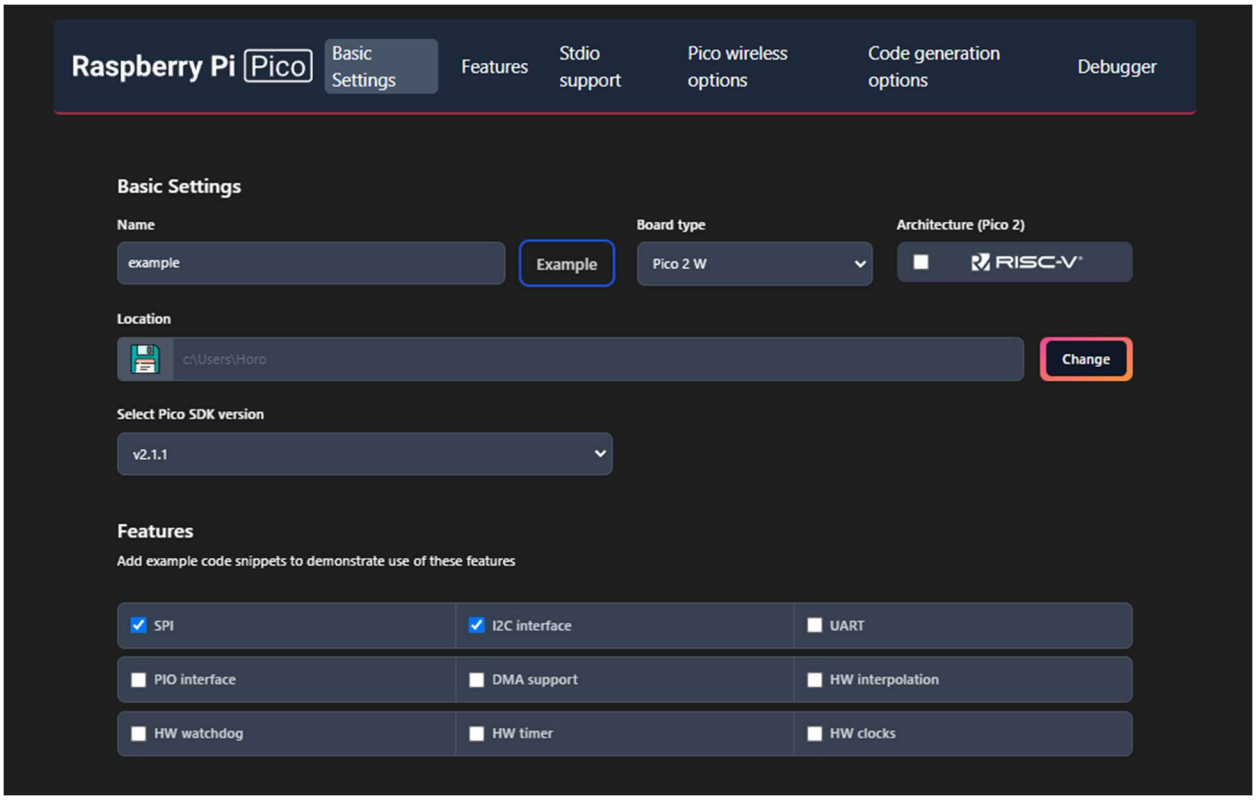
2. Check if all dependencies are installed, they should be automaticlly installed with **Raspberry Pi Pico**



New Pico Project

1. Find your **Raspberry Pi Pico Project: Quick Access** which looks like this

2. Click **New C/C++ Project**
3. Settings for SPI and I2C



4. Then, **Create** (It may take some time downloading Pico SDK)
5. After creating **new project**, you will see **example.c** in project folder.
6. **example.c** contains a initialization for SPI and I2C, which should be same or similar to this. (see next page)

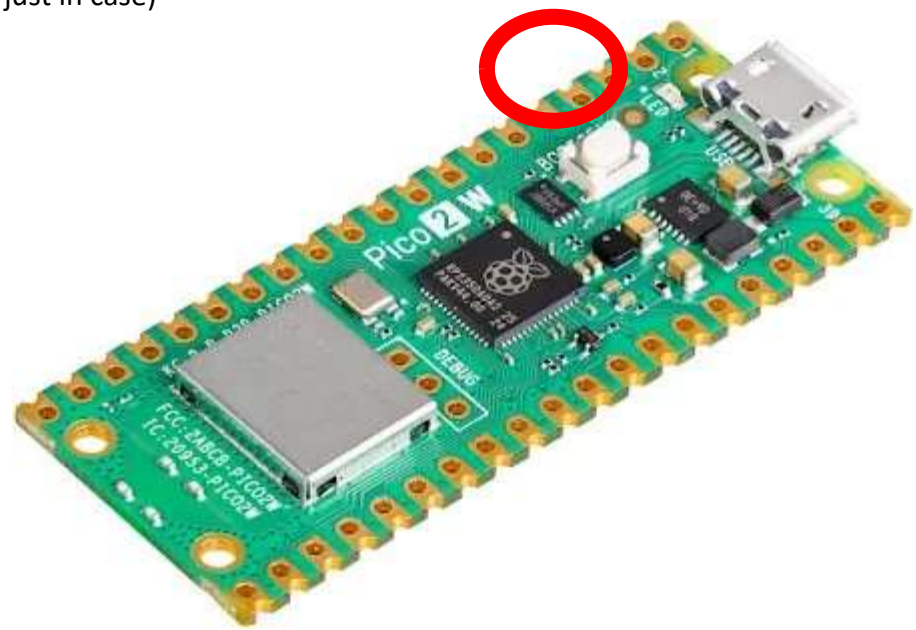
```c
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/spi.h"
#include "hardware/i2c.h"

// SPI Defines
// We are going to use SPI 0, and allocate it to the following GPIO pins
// Pins can be changed, see the GPIO function select table in the datasheet for information on GPIO assignments
#define SPI_PORT spi0
#define PIN_MISO 16
#define PIN_CS   17
#define PIN_SCK  18
#define PIN_MOSI 19

// I2C defines
// This example will use I2C0 on GPIO8 (SDA) and GPIO9 (SCL) running at 400KHz.
// Pins can be changed, see the GPIO function select table in the datasheet for information on GPIO assignments
#define I2C_PORT i2c0
#define I2C_SDA 8
#define I2C_SCL 9

int main()
{
    stdio_init_all();

    // SPI initialisation. This example will use SPI at 1MHz.
    spi_init(SPI_PORT, 1000*1000);
    gpio_set_function(PIN_MISO, GPIO_FUNC_SPI);
    gpio_set_function(PIN_CS,   GPIO_FUNC_SIO);
    gpio_set_function(PIN_SCK,  GPIO_FUNC_SPI);
    gpio_set_function(PIN_MOSI, GPIO_FUNC_SPI);

    // Chip select is active-low, so we'll initialise it to a driven-high state
    gpio_set_dir(PIN_CS, GPIO_OUT);
    gpio_put(PIN_CS, 1);
    // For more examples of SPI use see https://github.com/raspberrypi/pico-examples/tree/master/spi

    // I2C Initialisation. Using it at 400Khz.
    i2c_init(I2C_PORT, 400*1000);

    gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
    gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
    gpio_pull_up(I2C_SDA);
    gpio_pull_up(I2C_SCL);
    // For more examples of I2C use see https://github.com/raspberrypi/pico-examples/tree/master/i2c

    while (true) {
        printf("Hello, world!\n");
        sleep_ms(1000);
    }
}


```

Example compile and run

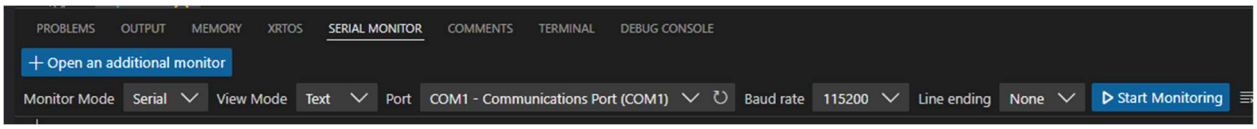
1. In **Raspberry Pi Pico Project: Quick Access**, find **Compile Project** and click it. Output as:

```
[63/63] Linking CXX executable example.elf
 *  Terminal will be reused by tasks, press any key to close it.
```

2. Now plug the Pico to Your PC use micro-USB while hold this button(BOOTSEL), release after connect. (You do not need to hold this button each time when connecting to PC, just in case)



3. Once Pico are connected you can either **Run Project(USB)** (Compile and flash) or **Flash Project(SWD)** you previous compiled to the Pico.
4. Then you will see print out from using **Serial Monitor** in **Panel** Once you press **Start Monitoring.** (Don't forget **SELECT** your Port)



5. You can see your Hello world. Something like these

```
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
```

SPI and I2C Read/Write from pin Examples

SPI Initialization

```
 1.  // SPI initialization
 2. void DPS310Barometer::initialize_spi() {
 3.   spi_init(spi1, 10 * 1000 * 1000);
 4.   gpio_set_function(PICO_SPI_1_SCK_PIN, GPIO_FUNC_SPI);
 5.   gpio_set_function(PICO_SPI_1_TX_PIN, GPIO_FUNC_SPI);
 6.   gpio_set_function(PICO_SPI_1_RX_PIN, GPIO_FUNC_SPI);
 7.
 8.   // Chip select is active-low, so we'll initialise it to a driven-high state
 9.   gpio_init(PICO_SPI_1_CSN_PIN);
10.   gpio_set_dir(PICO_SPI_1_CSN_PIN, GPIO_OUT);
11.   gpio_put(PICO_SPI_1_CSN_PIN, true);
12.
13.   // Make the CS pin available to picotool
14.   bi_decl(bi_1pin_with_name(PICO_SPI_1_CSN_PIN, "SPI CS"));
15. }
16.
```

SPI Read(Write-Read)

```
1. uint8_t DPS310Barometer::read_spi(uint8_t reg) {
2.   std::array<uint8_t, 2> tx_buf = {static_cast<uint8_t>(reg | 0x80U), 0};
3.   std::array<uint8_t, 2> rx_buf = {0, 0};
4.   gpio_put(PICO_SPI_1_CSN_PIN, false); // CS low
5.   spi_write_read_blocking(spi1, tx_buf.data(), rx_buf.data(), tx_buf.size());
6.   gpio_put(PICO_SPI_1_CSN_PIN, true); // CS high
7.   return rx_buf[1];
8. }
9.
```

SPI Write

```
1. void DPS310Barometer::write_spi(uint8_t reg, uint8_t value) {
2.   uint8_t tx_buf[2] = {reg, value};
3.   gpio_put(PICO_SPI_1_CSN_PIN, false); // CS low
4.   spi_write_blocking(spi1, tx_buf, sizeof(tx_buf));
5.   gpio_put(PICO_SPI_1_CSN_PIN, true); // CS high
6. }
7.
```

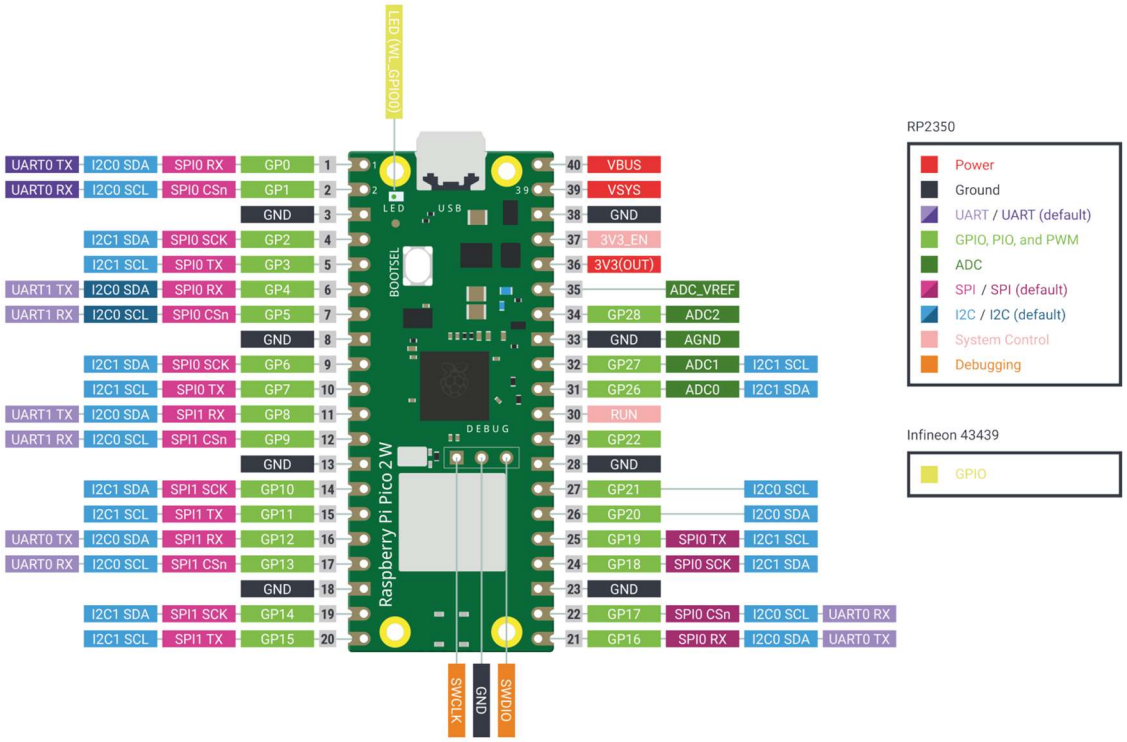
I2C

```
// I2C initialization
void DPS310Barometer::initialize_i2c() {
  i2c_init(i2c0, 100 * 1000); // Initialize I2C with 100kHz clock speed
  gpio_set_function(PICO_I2C_SDA_PIN, GPIO_FUNC_I2C);
  gpio_set_function(PICO_I2C_SCL_PIN, GPIO_FUNC_I2C);
  gpio_pull_up(PICO_I2C_SDA_PIN); // Enable pull-up resistors
  gpio_pull_up(PICO_I2C_SCL_PIN);
}

uint8_t DPS310Barometer::read_i2c(uint8_t reg) {
  uint8_t value = 0;
  i2c_write_blocking(
      i2c0, PICO_I2C_ADDR, ®, 1, true
  ); // Send register address
  i2c_read_blocking(
      i2c0, PICO_I2C_ADDR, &value, 1, false
  ); // Read value from register
  return value;
}

void DPS310Barometer::write_i2c(uint8_t reg, uint8_t value) {
  std::array<uint8_t, 2> data{reg, value};
  i2c_write_blocking(
      i2c0, PICO_I2C_ADDR, data.data(), data.size(), false
  ); // Write value to register
}

```

Reference

Pico 2 W Pinout

https://datasheets.raspberrypi.com/picow/pico-2-w-pinout.pdf



Pico SDK

For SPI

https://www.raspberrypi.com/documentation/pico-sdk/hardware.html#group_hardware_spi

For I2C

https://www.raspberrypi.com/documentation/pico-sdk/hardware.html#group_hardware_i2c