PI PICO

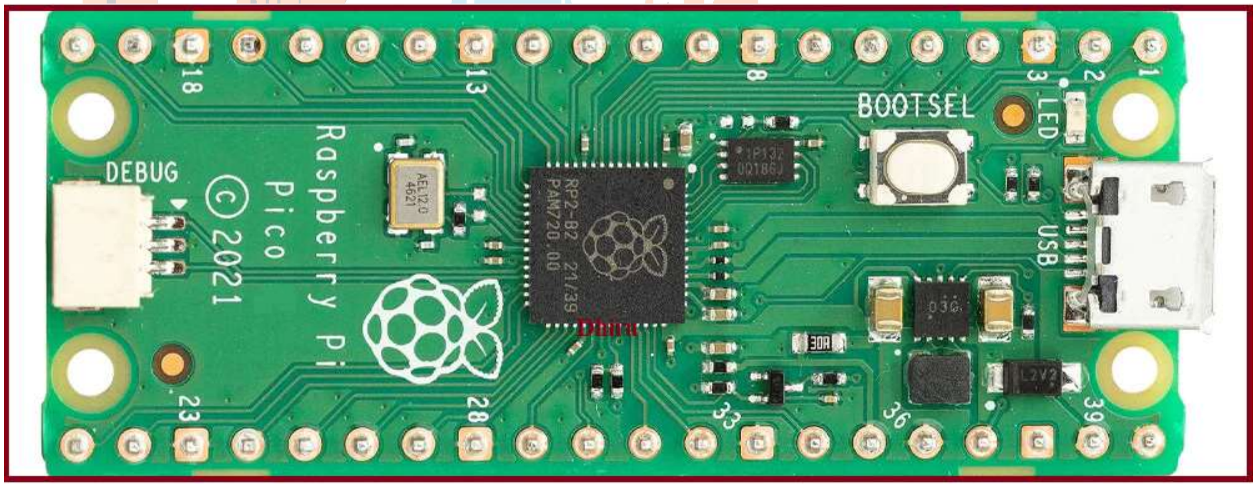
By-Dharmendra Kumar Yadav

Overview

Raspberry Pi Pico, a brand new exciting Microcontroller board based on RP2040 Microcontroller from the Raspberry Pi Foundation. The Raspberry Pi Pico is a low-cost Arm-based microcontroller that we can program using C/C++ and MicroPython.

Over the years Raspberry Pi boards have become a must tool for students, hobbyists,s or Industrialists. But when it comes to cost, the Raspberry Pi Board is overtaken by Arduino, ESP32, STM32, or other AVR, ARM, and PIC Microcontrollers. The Raspberry Pi computer costs around ₹ 3000-₹4000 whereas the other microcontrollers barely cost 300-400₹ only. This is the reason why Raspberry Pi Foundation released their low-cost powerful competitive Raspberry Pi Pico Board with RP2040, a Dual Core ARM Cortex-M0+ Microcontroller.

The tutorial covers the RP2040 Microcontroller, its features & specifications. We will also learn about the Raspberry Pi Pico Board, its layout, and specifications. The detailed guide of Raspberry Pi Pico Pins like ADC pins, I2C Pins, SPI Pins, UART, etc can help you to interface any sensors or module with this powerful board.



Since it's a Raspberry Pi Pico getting started tutorial, so we will only program the device using Micropython. For that, you can either use Thonny IDE or you can also go with uPyCraft IDE. In some other tutorials, we will learn how to program Raspberry Pi Pico with C/C++. Even the Arduino IDE will support Raspberry Pie Pico in the future as it is in the development phase now.

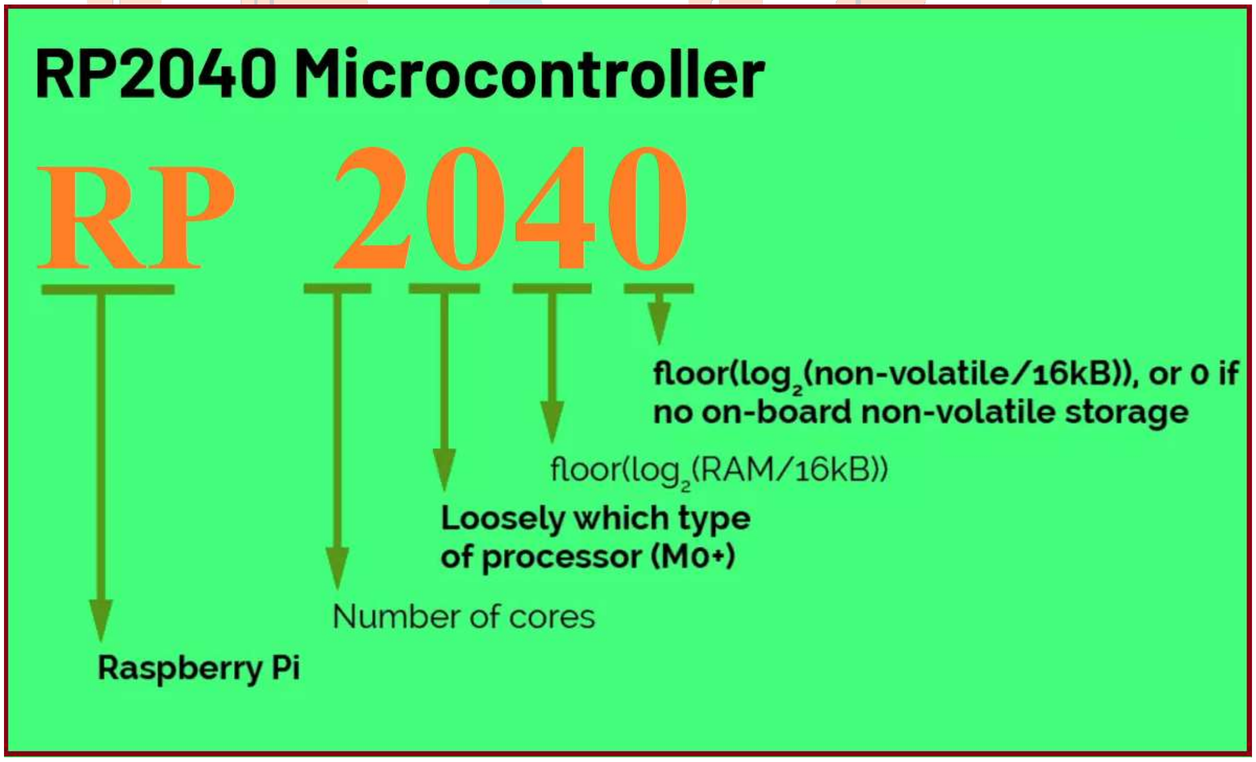
What is the RP2040 Microcontroller?

Earlier all the Raspberry Pi boards like **Raspberry Pi 3** or **4** or **Raspberry Pi Zero** featured Broadcom Processors like **BCM2835, BCM2836, BCM2711,** etc. The RP2040 chip was announced on 21st January 2021 and is the first processor designed by the Raspberry Pi Foundation.

The RP2040 is a **32-bit** dual **ARM Cortex-M0+ microcontroller** integrated circuit released simultaneously as part of the Raspberry Pi Pico board. The processor is a low-cost microcontroller and costs around **US$4**. The chip is **40nm** silicon in a **7×7 mm QFN-56** package.

The RP2040 contains two **ARM Cortex-M0+** cores clocked at **133 MHz** together with **264 KB** of RAM. The Program memory is external and supports up to **16 MB**. The device has everything you expect from a modern microcontroller like **UARTS, SPI,** and **I2C** ports, and there are timers, PWM, DMA, and a 12-bit analog-to-digital converter **(ADC)**.

Meaning of RP2040


RP2040 Microcontroller
RP 2040
floor(log2(non-volatile/16kB)), or 0 if no on-board non-volatile storage
floor(log2(RAM/16kB))
Loosely which type of processor (M0+)
Number of cores
Raspberry Pi

The name RP2040 has an exciting meaning explained below.
1. RP means: Raspberry Pi

2. Number 2 means: Processor Cores as a dual-core microcontroller. So, the value is 2.
3. Number 0 means: Type of Processor Core as it is ARM Cortex-M0+. So, the value is 0.
4. Number 4 means: Represents On-chip RAM. RP2040 has 264 KB of RAM. The formula to get 4 values is floor (log2 (ram / 16k)). So, the value is 4.
5. Number 0 means: Represents On-chip Flash. As there is no on-chip flash, the value is 0.

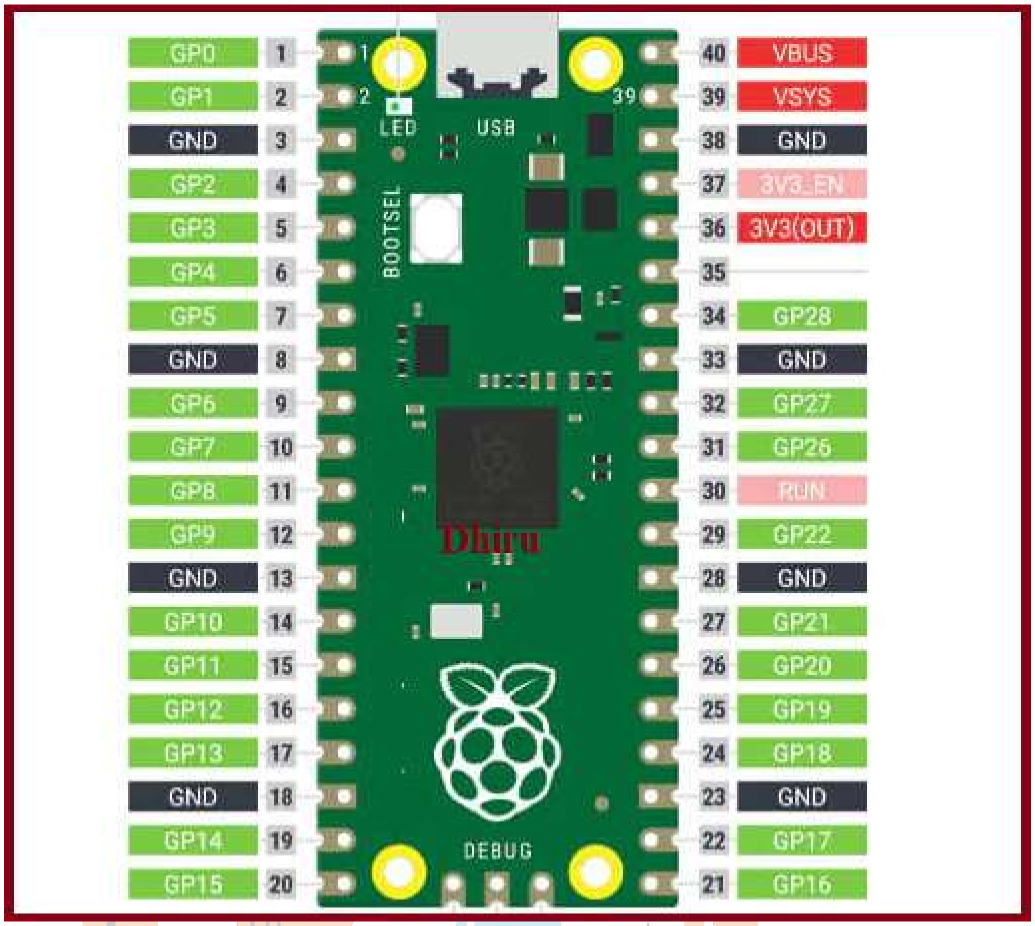
RP2040 Key features:

1. **133MHz** dual ARM Cortex-M0+ cores
2. 264kB SRAM in six independent banks
3. Support for up to **16MB** of off-chip Flash memory via a dedicated QSPI bus
4. **DMA** controller
5. Fully-connected AHB crossbar
6. Interpolator and integer divider peripherals
7. On-chip programmable **LDO(Low-dropout_regulator)** to generate core voltage
8. 2 on-chip **PLLs** to generate USB and core clocks
9. **30 GPIO** pins, 4 of which can be used as **analog inputs**

Introduction to Raspberry Pi Pico

The Raspberry Pi Pico is the first microcontroller board based on the RP2040. It looks a lot like other **microcontroller boards** with the MCU in the center, a micro-USB connector on one end, and a row of contacts along each side. A 3-pin **debug connector** is available at the other end of the board.

The Raspberry Pi Pico measures **51 by 21 mm**, which is the exact same size as an ESP32 Pico Kit & slightly larger than an Arduino Nano or Micro. The Pico comes with **2 MB of QSPI Flash memory** and 25 of the 30 GPIO pins of the RP2040 have been brought out on the extension connectors. The board is **breadboard friendly** and fits perfectly on a breadboard.

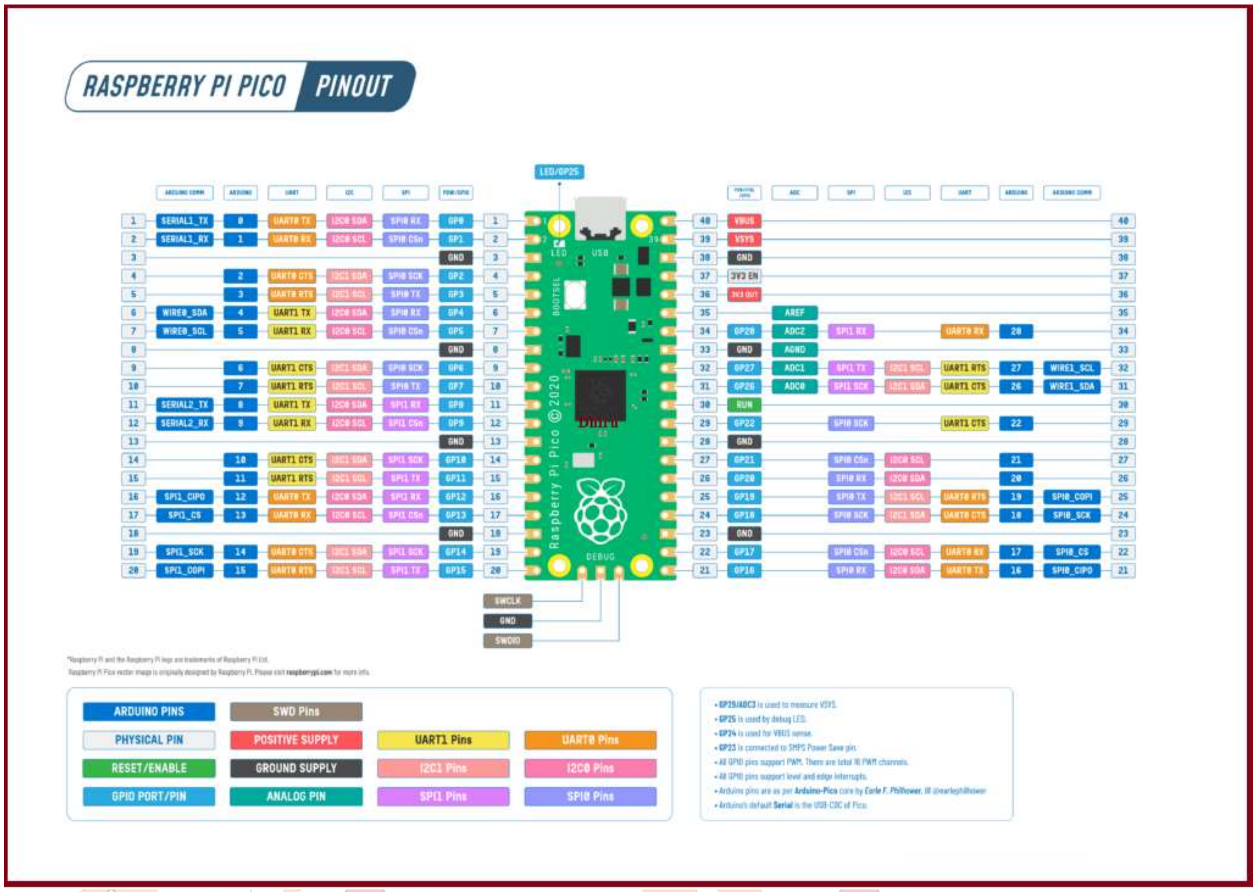


Features of Raspberry Pi Pico

Following are the features of the Raspberry Pi Pico Board.
1. Based on **RP2040** Microcontroller
2. **2 MB** of SPI Flash Memory
3. **Type B Micro-USB** port for power & programming
4. **40 DIP** style IO Pins
5. 3-pin ARM Serial Wire Debug (SWD) interface
6. **12 MHz Crystal** oscillator
7. Boot Selection Button
8. Programmable LED connected to GPIO 25
9. **3.3V** Fixed Output **Buck-Boost SMPS Converter**

Raspberry Pi Pico Pinout

There are **40 pins** on the Raspberry Pi Pico. Out of those **40 pins**, 26 pins are Input-Output (IO Pins). All those 14 pins are analog, digital, and other Serial Pins. There are **14 power** and system-related pins. The remaining 3 more pins are used for **SWD Debugging**.



- **GP29/ADC3** is used to measure VSYS.
- **GP25** is used by debug LED.
- **GP24** is used for VBUS sense.
- **GP23** is connected to SMPS Power Save pin.
- All GPIO pins support PWM. There are total 16 PWM channels.
- All GPIO pins support level and edge interrupts.
- Arduino pins are as per **Arduino-Pico** core by *Earle F. Philhower, III* @earlephilhower
- Arduino's default **Serial** is the USB-CDC of Pico.

There are two I2C peripherals available, I2C0 and I2C1. Similarly, there are two SPI peripherals, SPI0 and SPI1The number of UART Pins is also two, UART0 and UART1. You can assign any of these to the pins on which they are available.

Before you start using Raspberry Pi Pico, you have to **solder 40-pin male headers**, 20 on each side of the board.

RP2040 is an **ARM Cortex-M0+** dual-core microcontroller. That means you have two separate processors inside the package that can execute your code parallelly. It is like having two microcontrollers for the price of one. Both cores run at 133 MHz and they could be even overclocked! Cortex-M0+ is a series of ARM microcontrollers that are optimized for power consumption. Like all ARM Cortex microcontrollers, RP2040 also has a 32-bit core, which means it can execute complex 32-bit instructions in every instance. It is also a **RISC** processor which can execute a single instruction in just 1 clock cycle.

Let's see a comparison between RP2040 and the popular **ATmega328P**, which is used in the Arduino Uno board. Only a few features are compared.

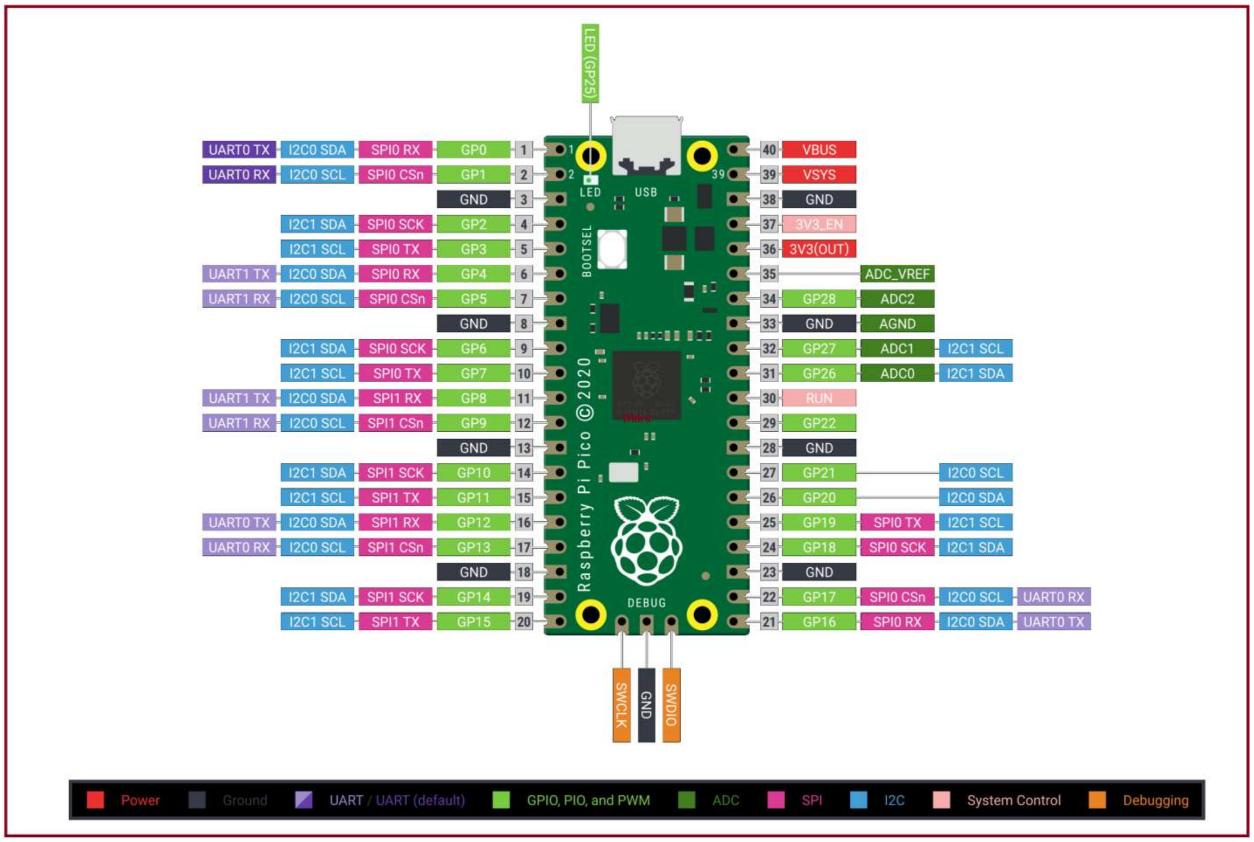
Property	RP2040	ATmega328P
Core Type	ARM Cortex-M0+	8-bit AVR
Instruction Type	RISC	RISC
Instruction Size	32-bit	8-bit
Core Count	2	1
Maximum Clockspeed	133 MHz	20 MHz
SRAM	264 KB	2 KB
Flash Size	16 MB	32 KB
GPIO Count	30	23
ADC Resolution	12 bit	10 bit
PWM	All GPIO pins	Limited GPIO pins
Interrupt	All GPIO pins	2 GPIO pins
UART	2	1
SPI	2	1
I2C	2	1
USB	Yes	No
Price	₹200	₹500
Firmware Lock Protection	No	Yes
Working Voltage	1.8 to 3.3V	1.8 to 5.5V

Comparison between RP2040 and ATmega328P

There are more differences between the two and features that make the RP2040 a really good choice for your projects and products. RP2040 was designed by Raspberry Pi’s own engineers and uses a **40 nm** node for fabrication. It is available in a **56-pin**

QFN package with a **7 x 7 mm** size. This is one of the caveats for using RP2040. It can be a little difficult to hand-solder it. The another one is having no firmware lock protection since the flash memory has to be off-chip. So you can not protect your firmware from being copied easily.

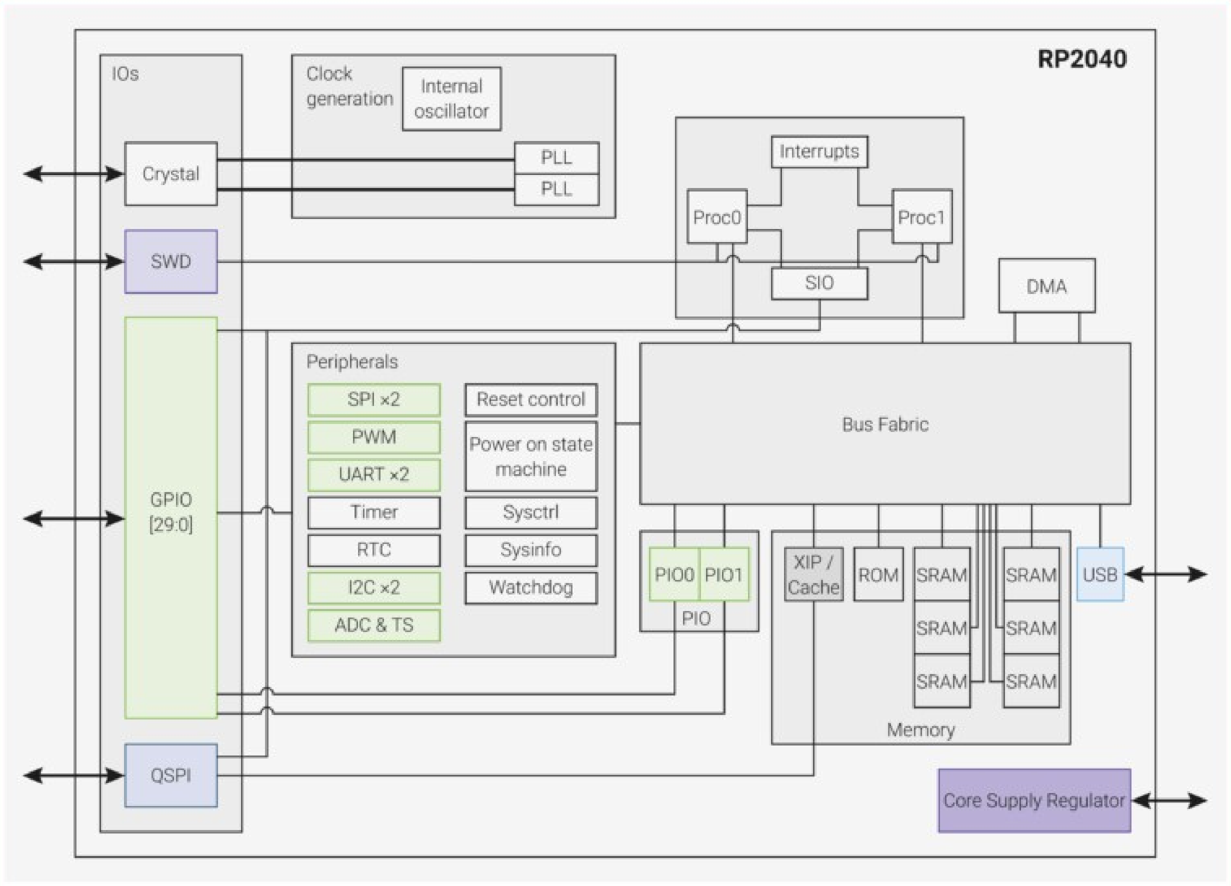
Specifications



- Dual ARM Cortex-M0+ @ 133MHz
    - On-chip PLL allows variable core frequency
- 264kByte high-performance SRAM in six independent banks
- Support for up to 16MB of off-chip Flash memory via dedicated QSPI bus with eXecute In Place (XIP)
- DMA controller
- Fully-connected AHB crossbar
- Interpolator and integer divider peripherals
- On-chip programmable LDO to generate the core voltage
- 2 on-chip PLLs to generate USB and core clocks
- 30 multi-function General Purpose IO (4 can be used for ADC)
    - 1.8-3.3V IO Voltage (NOTE: Pico IO voltage is fixed at 3.3V)
- 12-bit 500ksps Analogue to Digital Converter (ADC)
- Peripherals
    - 2 UARTs
    - 2 SPI controllers

    - 2 I2C controllers
    - 16 PWM channels
    - USB 1.1 controller and PHY, with host and device support
    - 8 PIO state machines
- 2 × Programmable IO (PIO) blocks, 8 state machines total
    - Flexible, user-programmable high-speed IO
    - Can emulate interfaces such as SD Card and VGA
- QFN-56 7x7mm package
-
-

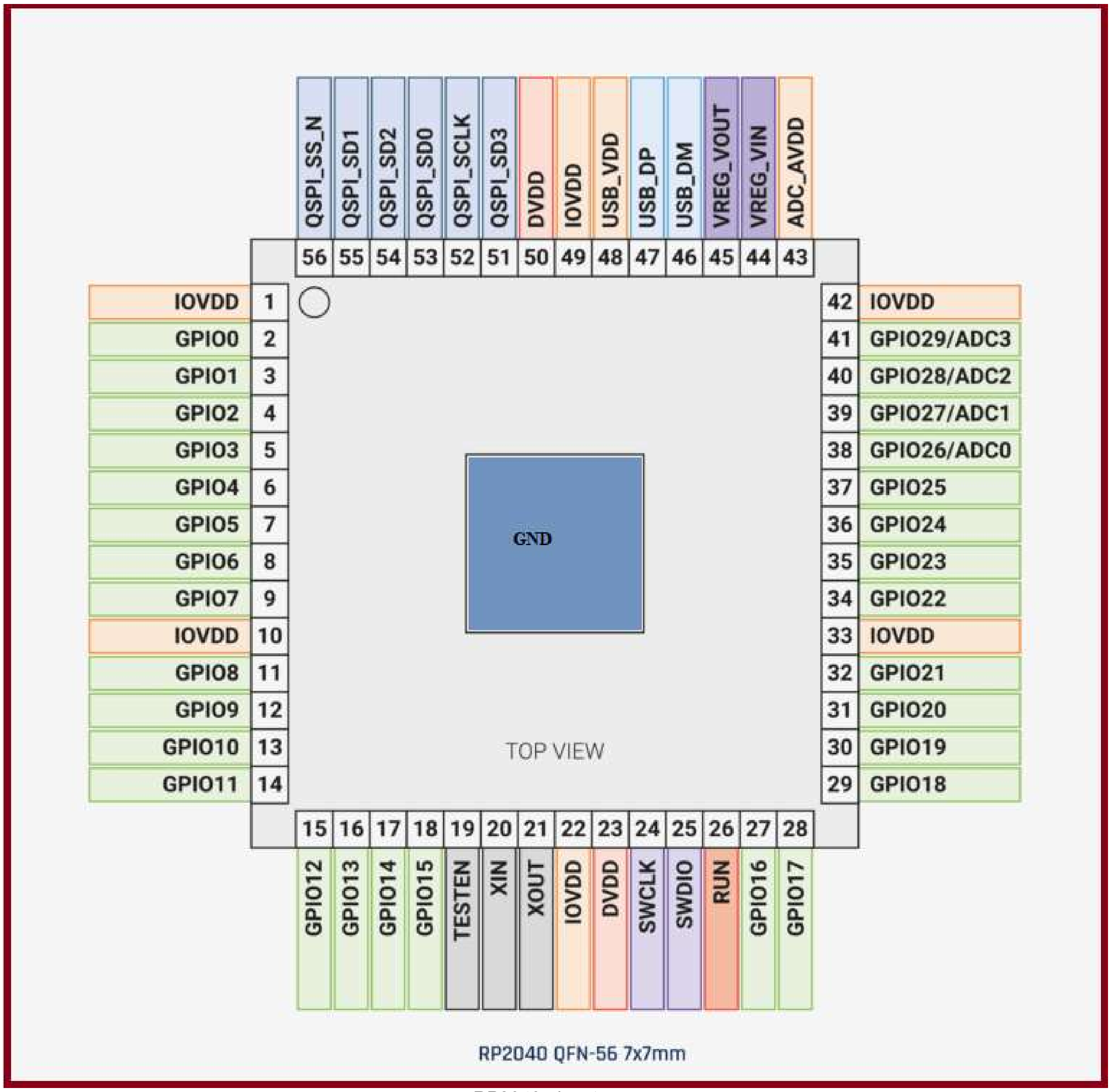
Block Diagram



RP2040 Block Diagram

As you can see, there is no flash memory embedded in the chip. We have to add an external Flash memory to store all the programs. A dedicated set of **QSPI** pins are provided for this so that you don't need to spare any GPIOs for the purpose. There are dedicated pins for SWD (Serial Wire Debug), USB, and crystal oscillator.

Pinout



RP2040 pinout

Programming Raspberry Pi Pico

The Pi Pico can be programmed using **C/C++** or **Python**, among other languages. Pico is adaptable to a vast range of applications and skill levels, and getting started is as easy as dragging and dropping a file. If you are working with C, then it is recommended to use a **Linux-based system** like a Raspberry Pi Computer as it is easy to download the SDK and write C Programs in Linux.

But I will recommend using **MicroPython** to program the Raspberry Pi Pico Board. MicroPython is a Python Language Interpreter that is developed for Microcontrollers and **embedded systems**. The **Syntax for MicroPython** is very similar to Python. So, if you worked with Python, then working with MicroPython will be very easy.

To program the Raspberry Pi Pico using Micropython, you can either use:
1. Thonny IDE
2. uPyCraft IDE

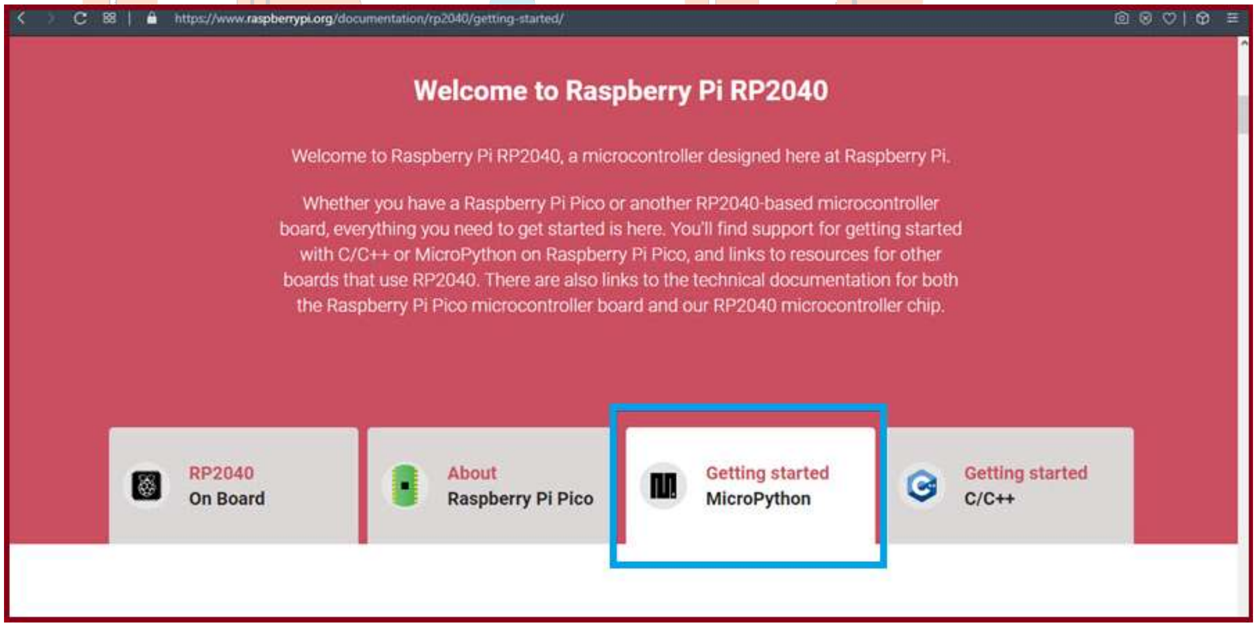
But before getting started with Raspberry Pi Pico, you have to install MicroPython on Raspberry Pi Pico Board.

Install MicroPython on Raspberry Pi

Download MicroPython Binary

The easiest and fastest way to run MicroPython on Raspberry Pi Pico is to download the prebuilt binary from the official Raspberry Pi Pico's website.

Go to the documentation page of Raspberry Pi Pico and click on "Getting Started MicroPython" tab.

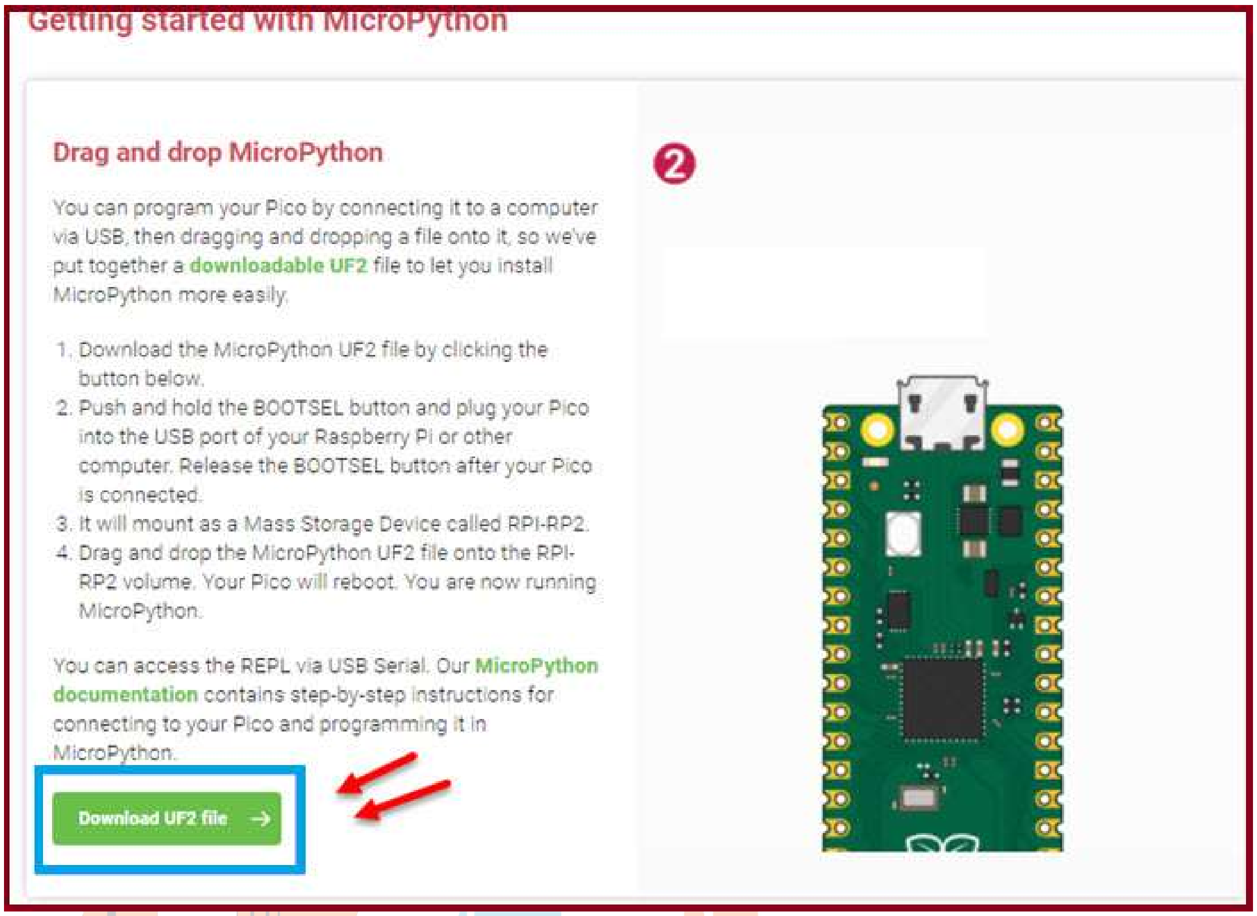

https://www.raspberrypi.org/documentation/rp2040/getting-started/
RP2040 On Board
About Raspberry Pi Pico
Getting started MicroPython
Getting started C/C++

Welcome to Raspberry Pi RP2040

Welcome to Raspberry Pi RP2040, a microcontroller designed here at Raspberry Pi.

Whether you have a Raspberry Pi Pico or another RP2040-based microcontroller board, everything you need to get started is here. You'll find support for getting started with C/C++ or MicroPython on Raspberry Pi Pico, and links to resources for other boards that use RP2040. There are also links to the technical documentation for both the Raspberry Pi Pico microcontroller board and our RP2040 microcontroller chip.

The content below the tab changes according to the selected tab and when you click on "Getting Started MicroPython", a text related to Getting started with MicroPython appears along with a small animation on how to install MicroPython on Raspberry Pi Pico.



Getting started with MicroPython

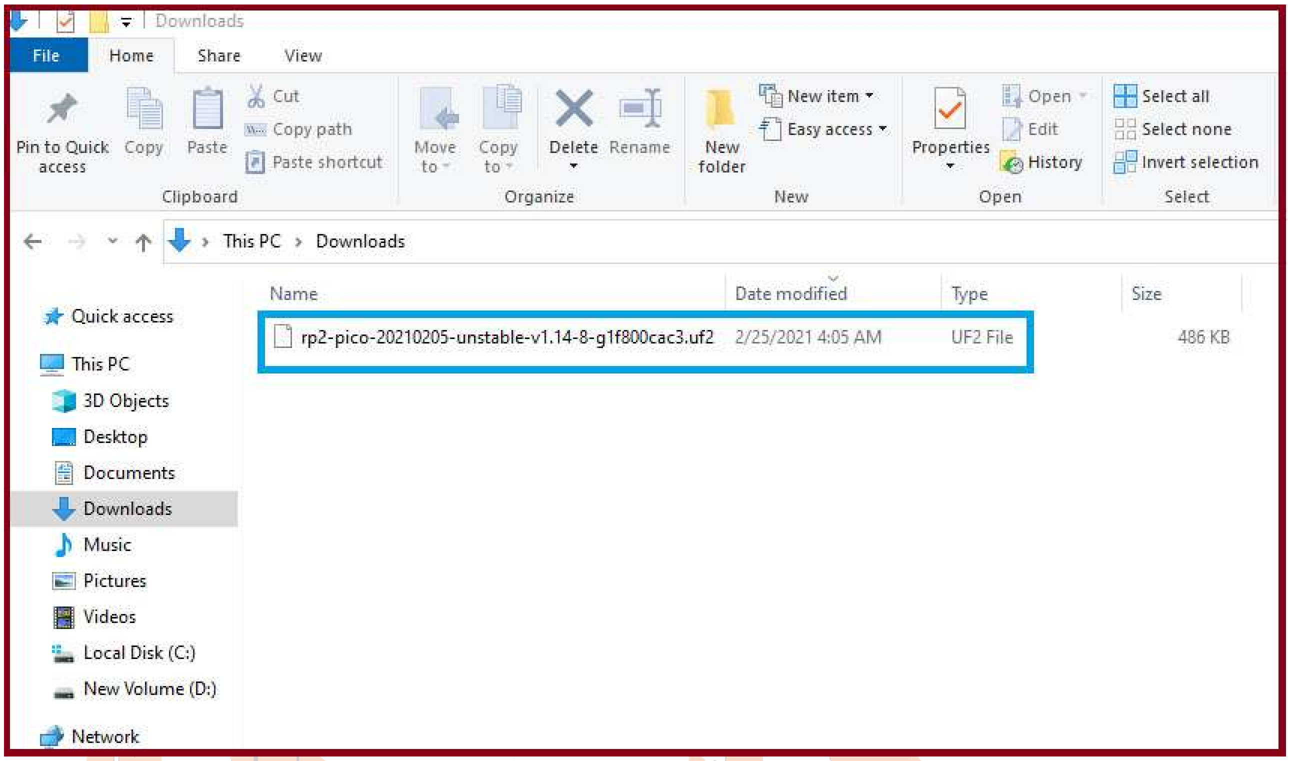
Drag and drop MicroPython

You can program your Pico by connecting it to a computer via USB, then dragging and dropping a file onto it, so we've put together a downloadable UF2 file to let you install MicroPython more easily.

1. Download the MicroPython UF2 file by clicking the button below.
2. Push and hold the BOOTSEL button and plug your Pico into the USB port of your Raspberry Pi or other computer. Release the BOOTSEL button after your Pico is connected.
3. It will mount as a Mass Storage Device called RPI-RP2.
4. Drag and drop the MicroPython UF2 file onto the RPI-RP2 volume. Your Pico will reboot. You are now running MicroPython.

You can access the REPL via USB Serial. Our MicroPython documentation contains step-by-step instructions for connecting to your Pico and programming it in MicroPython.

Read all the information and click on “Download UF2 file” option. A MicroPython Binary in the form of a .uf2 file will be downloaded.

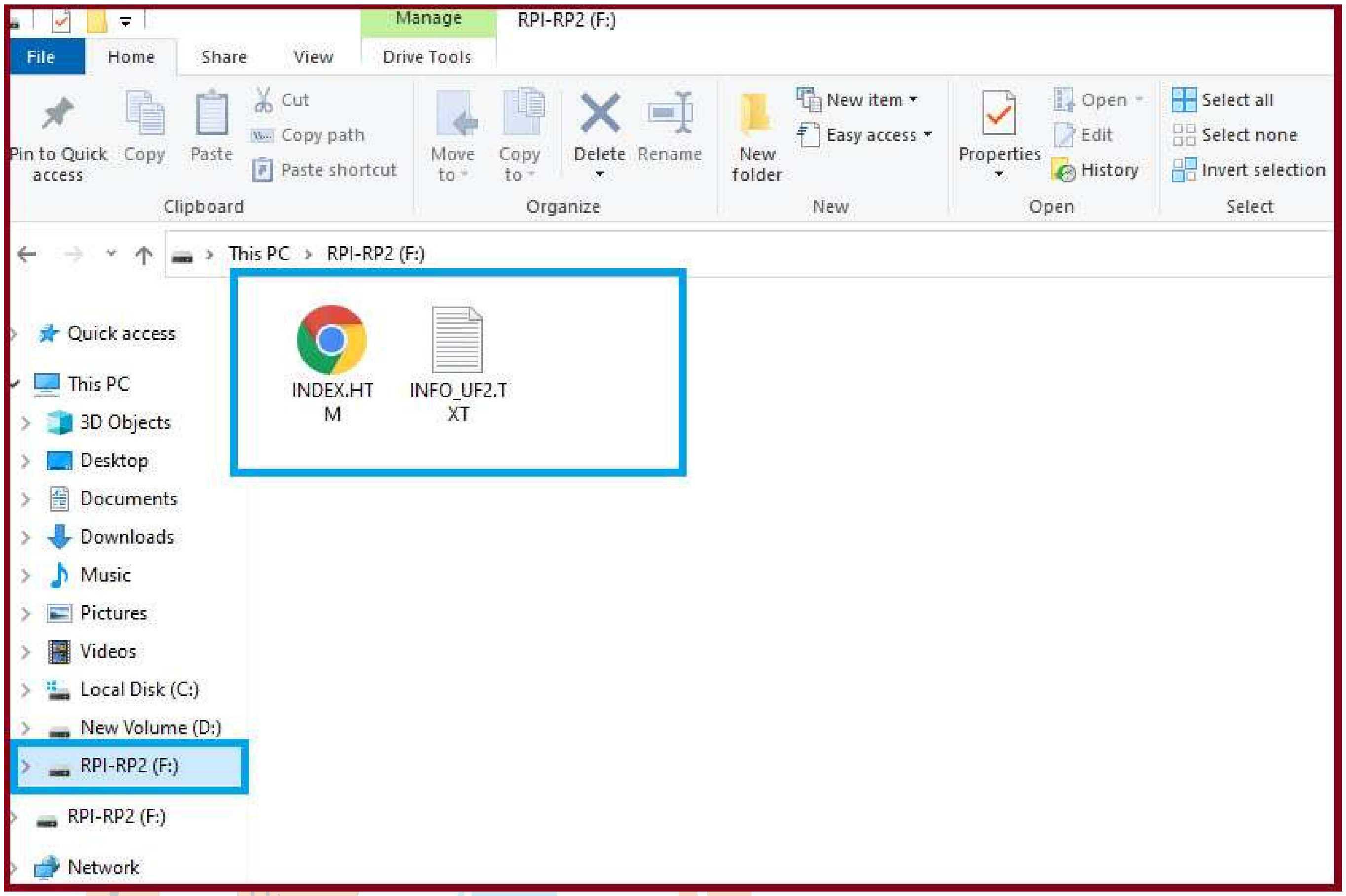


Name	Date modified	Type	Size
rp2-pico-20210205-unstable-v1.14-8-g1f800cac3.uf2	2/25/2021 4:05 AM	UF2 File	486 KB

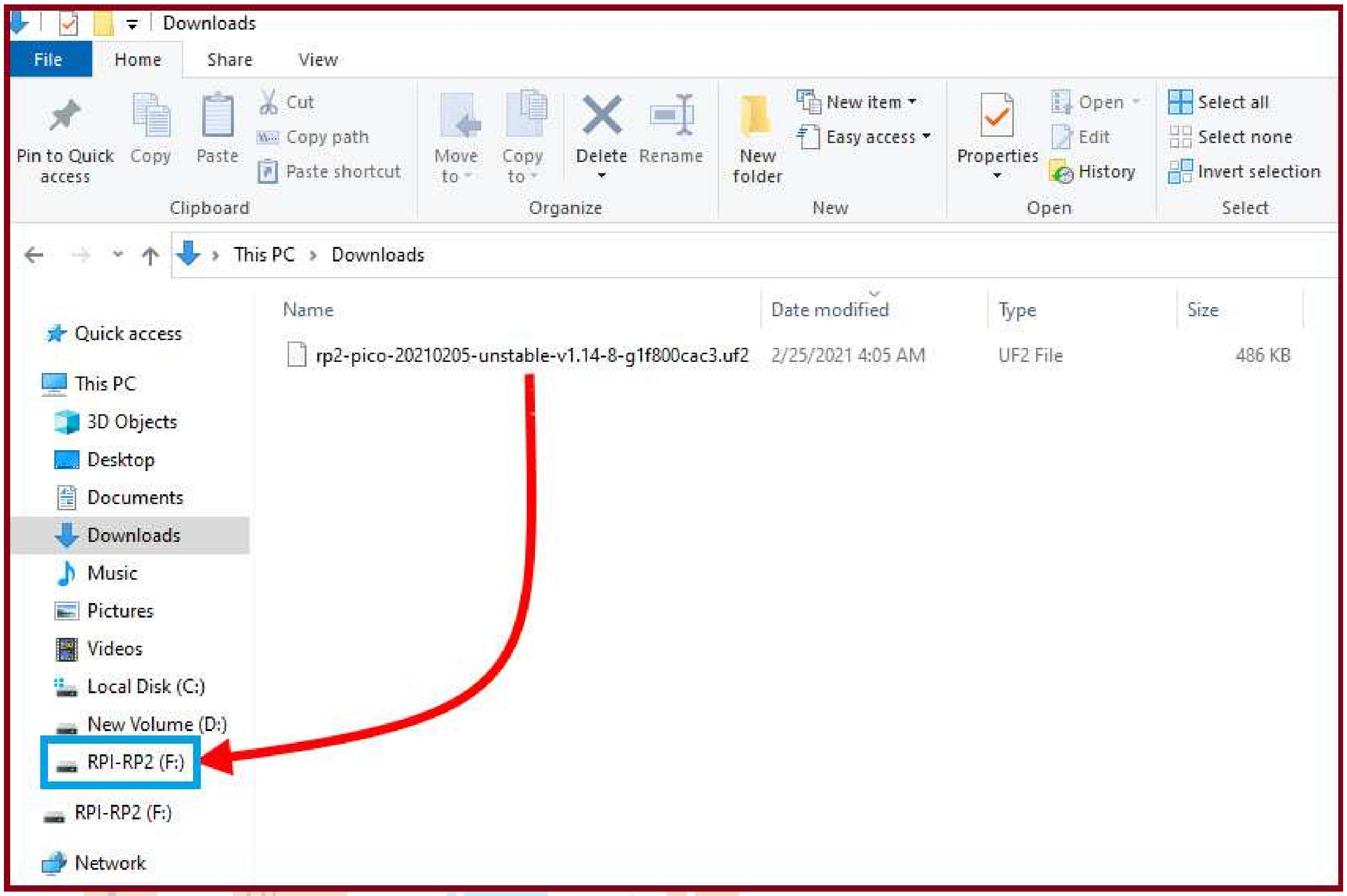
After downloading the MicroPython Binary, we have to upload this firmware in to the Raspberry Pi Pico. For that, first we have to put the Pico in bootloader mode.

To do that, plug-in a ***micro-USB cable to micro-USB port of Raspberry Pi Pico. Now, hold the BOOTSEL button on the Pico and plug-in the other end of the USB cable to a USB port of the host computer (while holding the BOOTSEL button).***

You can release the button after a couple of seconds when the Raspberry Pi Pico appears as a Mass Storage Device with name “RPI-RP2”. If you open it, you will see a text file and an HTML file.



Now, go to the downloads folder and drag-and-drop the downloaded MicroPython UF2 file onto RPI-RP2 device. After copying, the Raspberry Pi Pico will restart and run MicroPython. The mass storage device will disappear after you copy the MicroPython UF2 file.

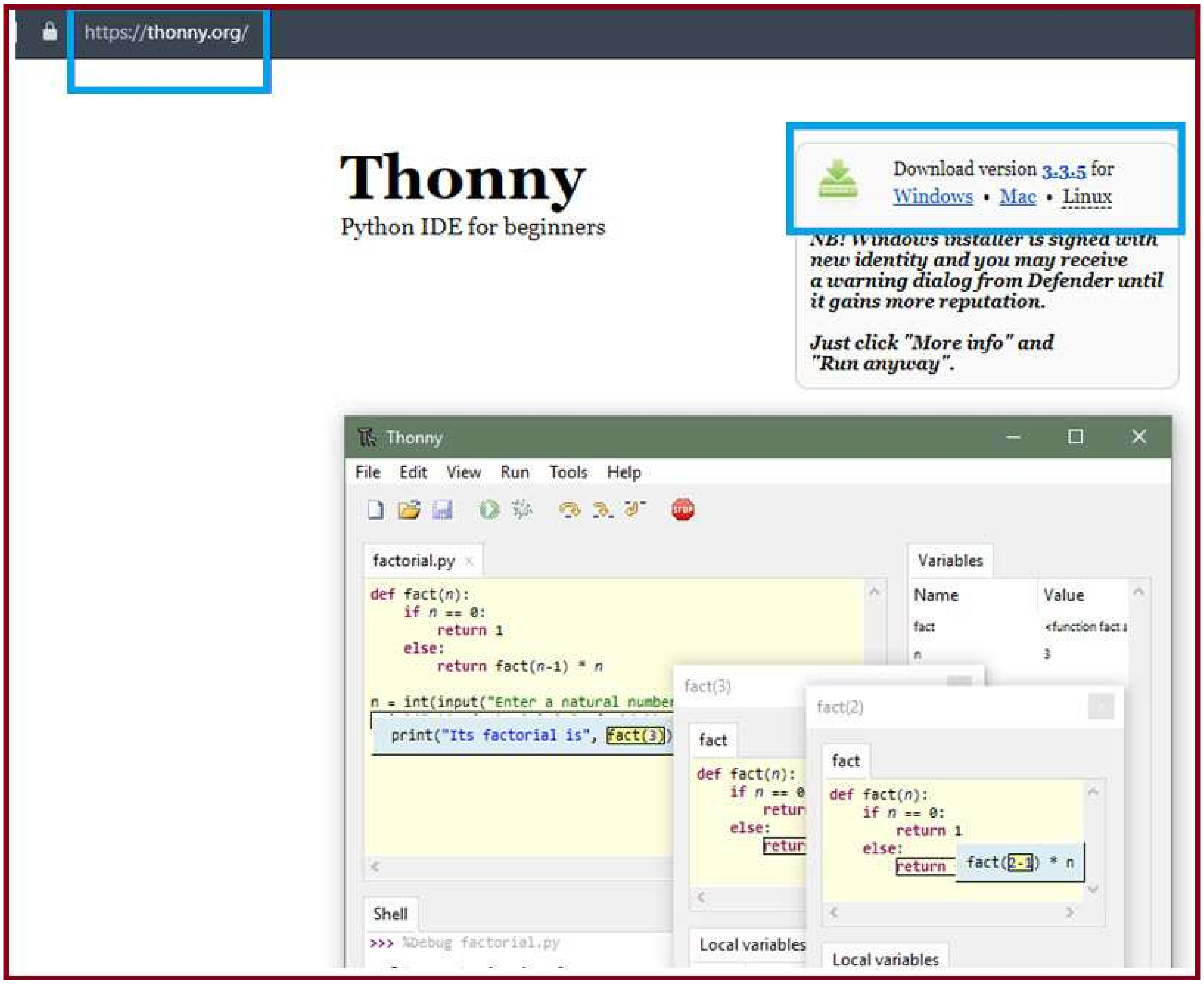


Name	Date modified	Type	Size
rp2-pico-20210205-unstable-v1.14-8-g1f800cac3.uf2	2/25/2021 4:05 AM	UF2 File	486 KB

Your Raspberry Pi Pico is now running MicroPython. You are now ready to program Raspberry Pi Pico with MicroPython.

1.Downloading Thonny

If your host computer is either Linux or Mac, then you can communicate with Raspberry Pi Pico using terminal and Minicom. But here, we will see how to program Raspberry Pi Pico using Thonny IDE.



Thonny

Python IDE for beginners

Download version 3.3.5 for Windows • Mac • Linux

NB! Windows installer is signed with new identity and you may receive a warning dialog from Defender until it gains more reputation.

Just click "More info" and "Run anyway".

```
def fact(n):
    if n == 0:
        return 1
    else:
        return fact(n-1) * n

n = int(input("Enter a natural number
print("Its factorial is", fact(3))
```

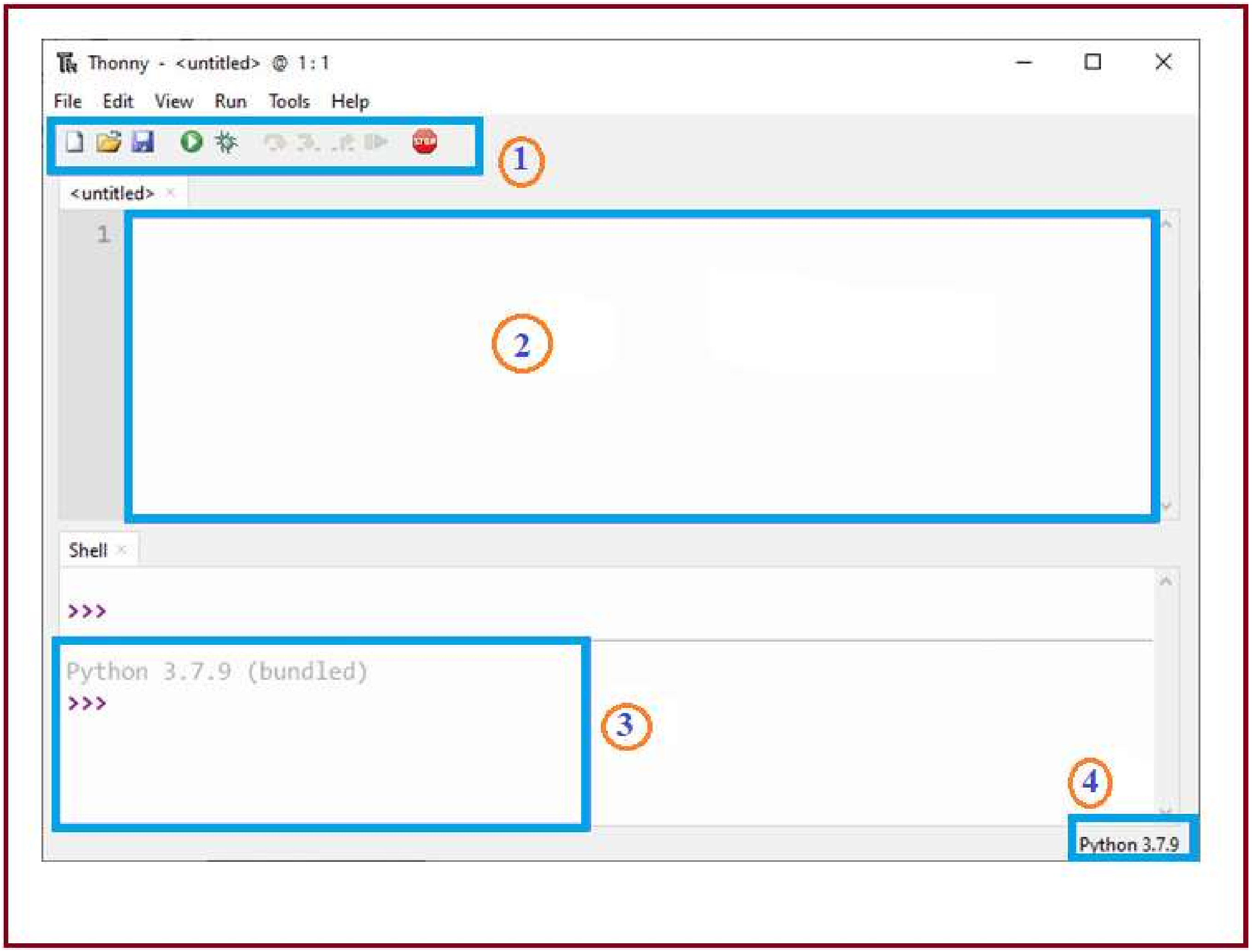
```
>>> %Debug factorial.py
```

Thonny is a simple Python IDE available for Windows, Mac and Linux. The Raspberry Pi OS comes with Thonny preinstalled. Since I am using a Windows system, I downloaded the Windows version of Thonny. An executable called “thonny-3.3.5.exe” is downloaded.

Double click on the downloaded executable and install Thonny. There is nothing special with this installation and it is very straight forward. Optionally, you can select to create a desktop shortcut.

Configuring Thonny

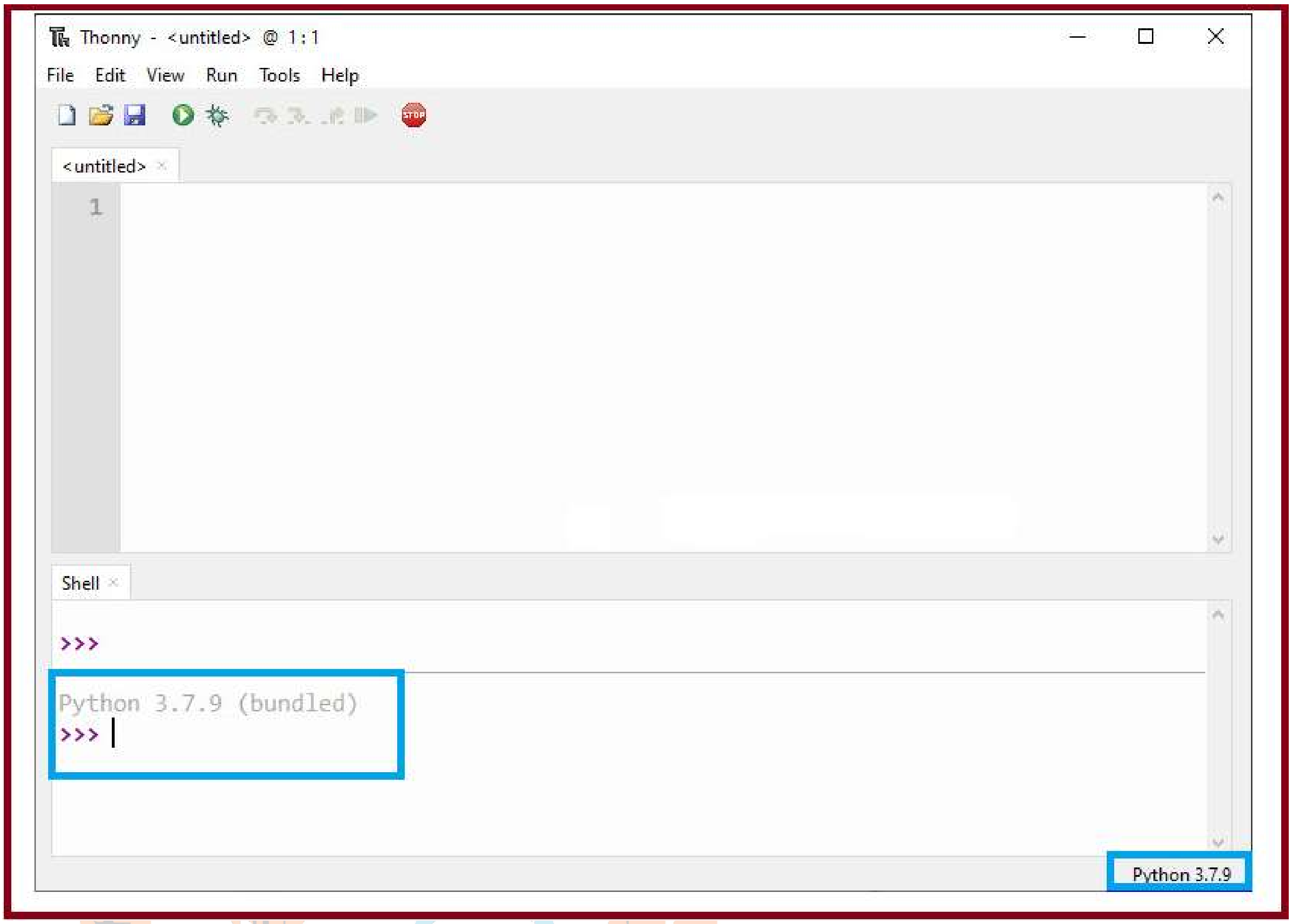
After downloading and installing Thonny IDE, open it. Make sure that Raspberry Pi Pico is already plugged into the host computer. Thonny IDE is very simple. Its layout can be divided into four parts: Toolbar, Script Area, Shell, Interpreter.



```
>>>
Python 3.7.9 (bundled)
>>>
```

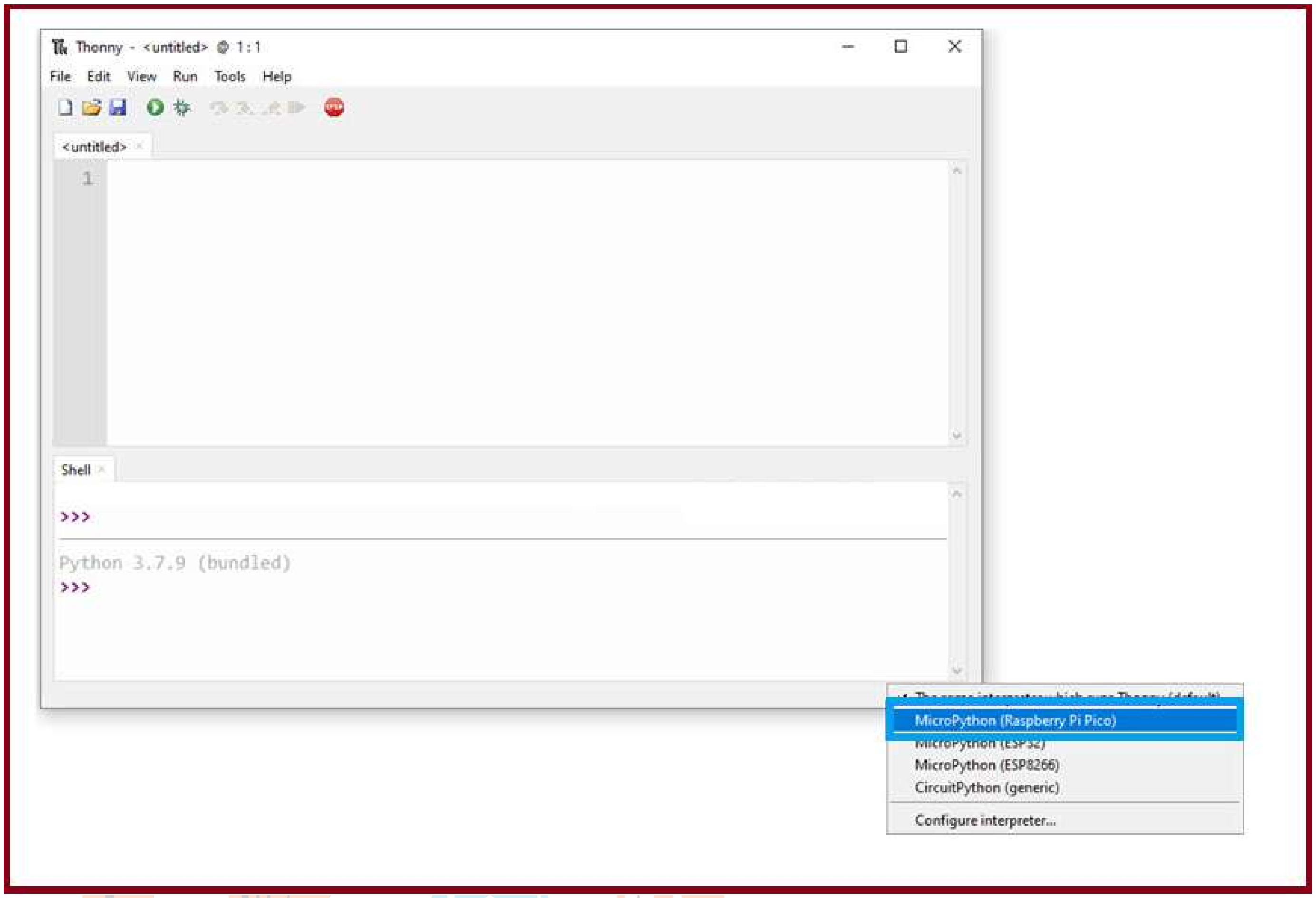
- The Toolbar: Contains icons for saving, running and stopping the programs.
- The Script Area: This is where you write the Python Programs.
- The Shell: The Python Shell is an interactive REPL (Read-Evaluate-Print-Loop) block where you can give individual commands to the interpreter and it will execute them.
- The Interpreter: Select the right interpreter from the bottom right of the IDE.

By default, Thonny IDE is configured to interpret Python 3.x.x.



```
>>>
Python 3.7.9 (bundled)
>>>
```

Click on Python 3.7.9 (or whatever the version is) and select MicroPython (Raspberry Pi Pico) interpreter. As soon as you select the MicroPython interpreter, the shell at the bottom changes to MicroPython.



```
>>>
Python 3.7.9 (bundled)
>>>
```

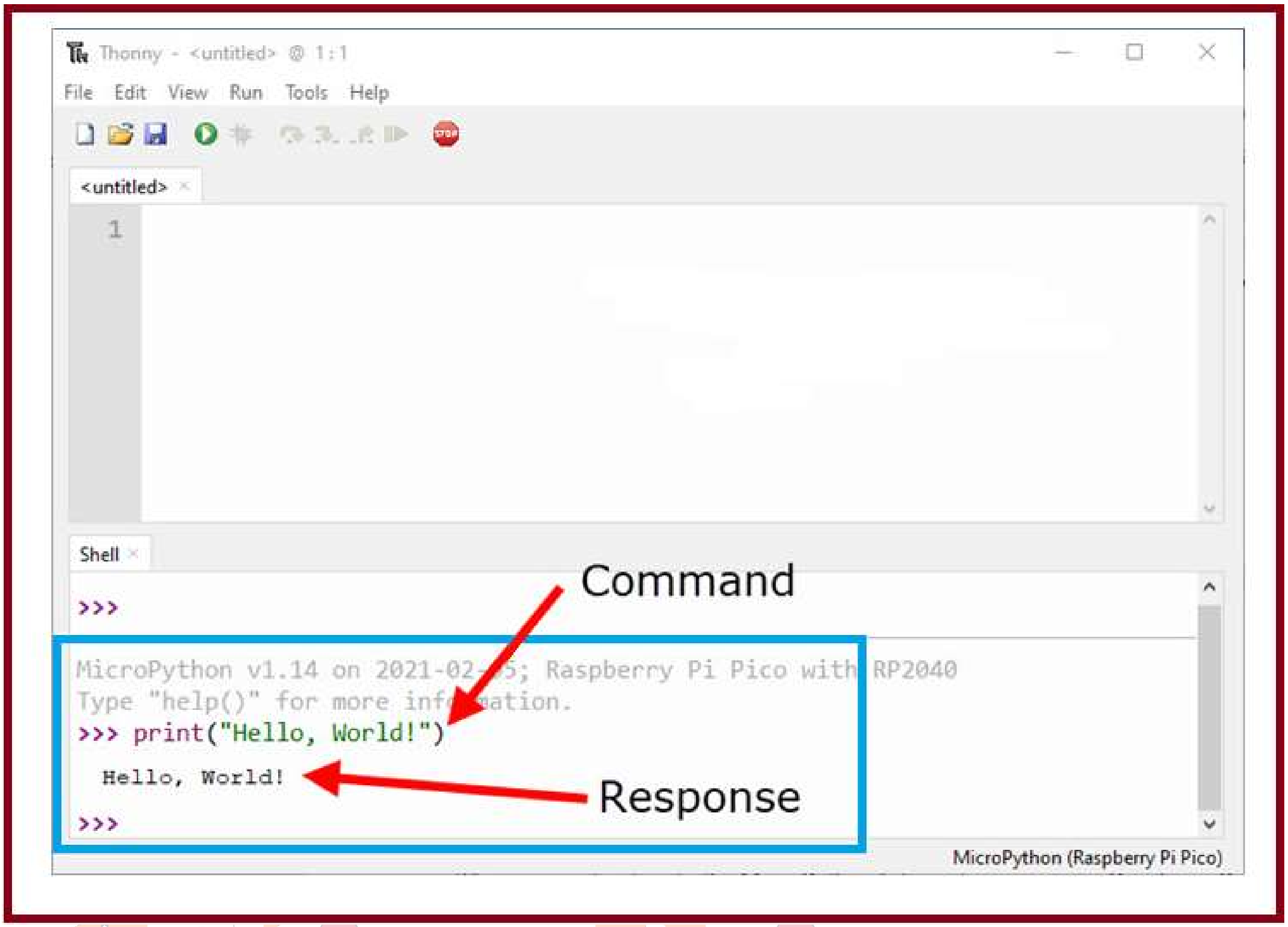
Since MicroPython supports interactive REPL, you can enter commands in the shell and Raspberry Pi Pico will execute them. Let us try this. We will start with Hello World of programs which is to print Hello World.

Programming

In the Shell, type the following next to “>>>” symbol and hit enter.

print(“Hello, World!)

This is an instruction to the MicroPython Interpreter running on Raspberry Pi Pico. Up on receiving this command, the MicroPython will respond with the message “Hello, World!” and prints it on the shell itself.


Thonny - <untitled> @ 1:1
File Edit View Run Tools Help
<untitled>
Shell
Command
Response
MicroPython (Raspberry Pi Pico)

```
>>>
MicroPython v1.14 on 2021-02-05; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print("Hello, World!")
 Hello, World!
>>>
```

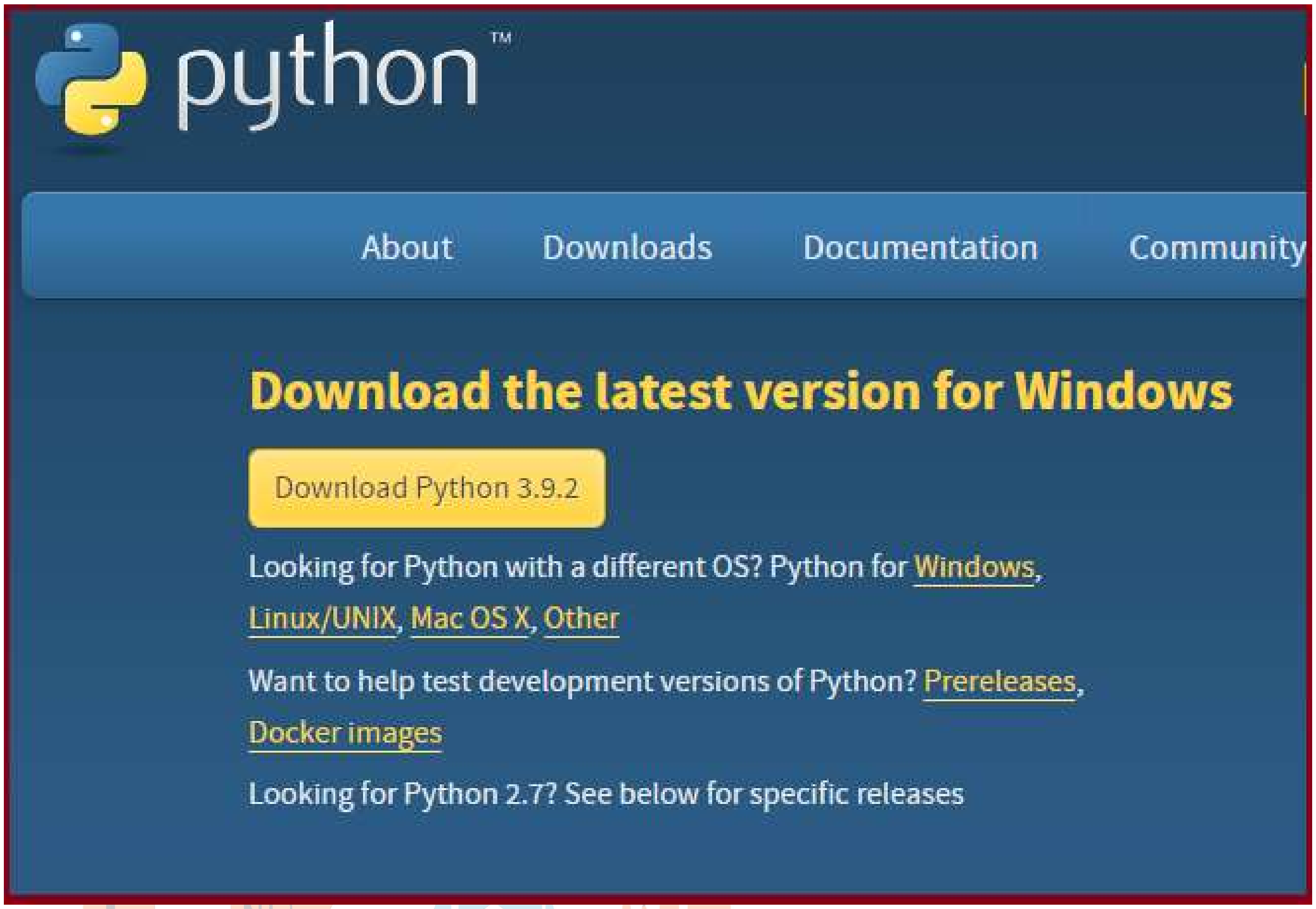
2.Installing uPyCraft IDE

Now we will learn how to use Raspberry Pi Pico in Micro-python firmware Using Thonny .nowwe are going to learn how to install uPyCraft IDE as it can be run in any major operating system and is always extremely interactive and easy. We will be installing uPyCraft IDE in Windows.

Installing Python3

Before we start the installation of the uPyCraft IDE IDE, make sure you have the latest version of Python 3.7 or the latest installed on your windows-based. If you don't have the python 3 package installed on your Windows, follow these steps to install one:

- First of all go this link here and download latest version of Python3.


About
Downloads
Documentation
Community

Download the latest version for Windows

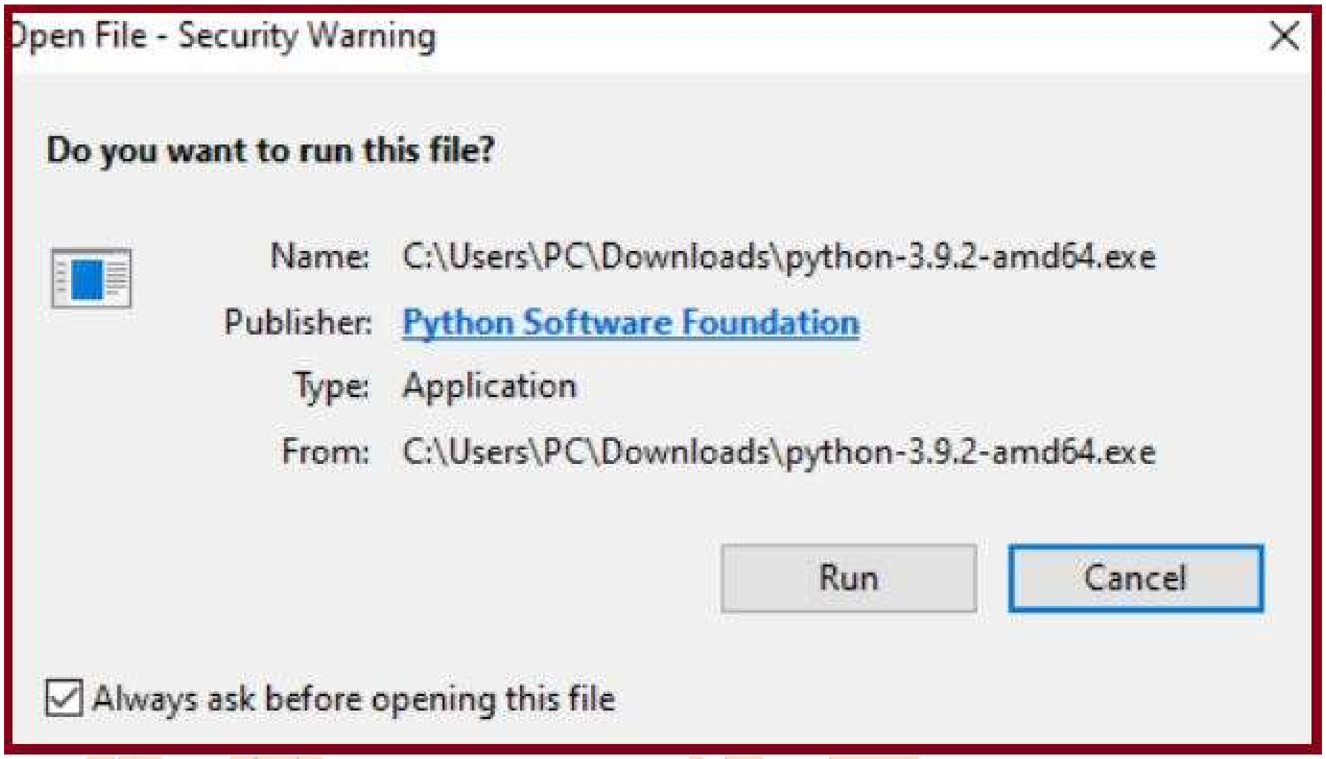
Download Python 3.9.2

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

After you click the Download Python 3.9.2 button an .exe file will start downloading.
After the download is completed click on it and the following appears.



Open File - Security Warning

Do you want to run this file?

Name: C:\Users\PC\Downloads\python-3.9.2-amd64.exe

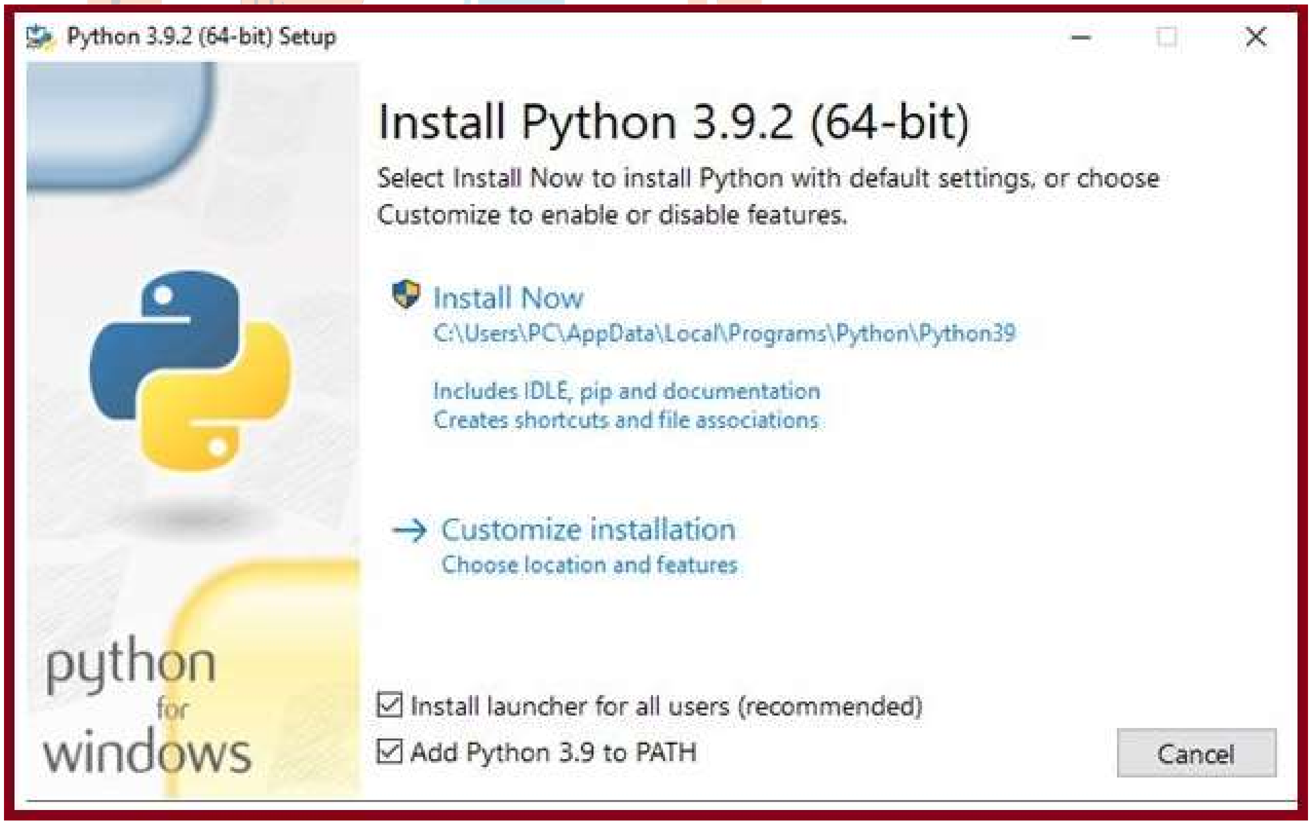
Publisher: Python Software Foundation

Type: Application

From: C:\Users\PC\Downloads\python-3.9.2-amd64.exe

☑ Always ask before opening this file

Press the run button. The following screen appears. Make sure to tick Add Python 3.9 to a path and then click Install Now.



Python 3.9.2 (64-bit) Setup

Install Python 3.9.2 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\PC\AppData\Local\Programs\Python\Python39

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.9 to PATH

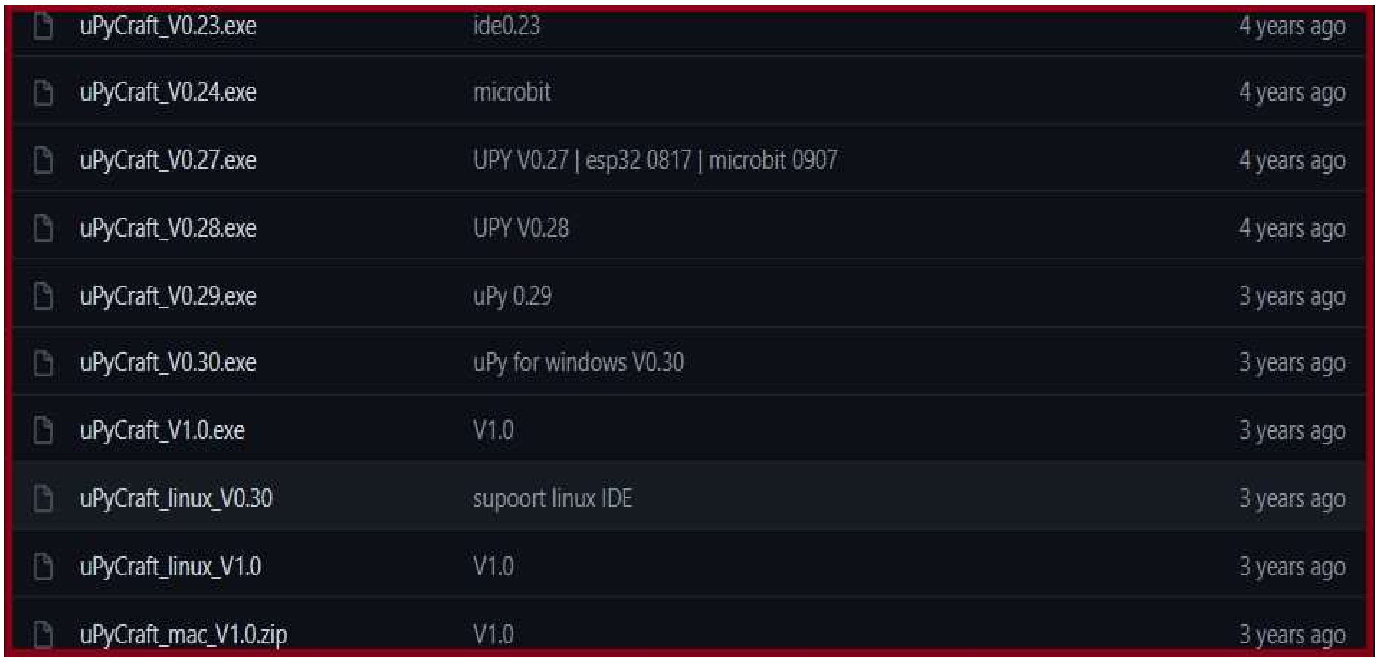
After the installation is complete you will receive a successful setup message.

Download and Install uPyCraft IDE

As we mentioned earlier, we will use uPyCraft IDE to program our Raspberry Pi Pico board. The reason we are using uPyCraft IDE is that it is simple and easy to use IDE among other MicroPython based IDEs available in the market.

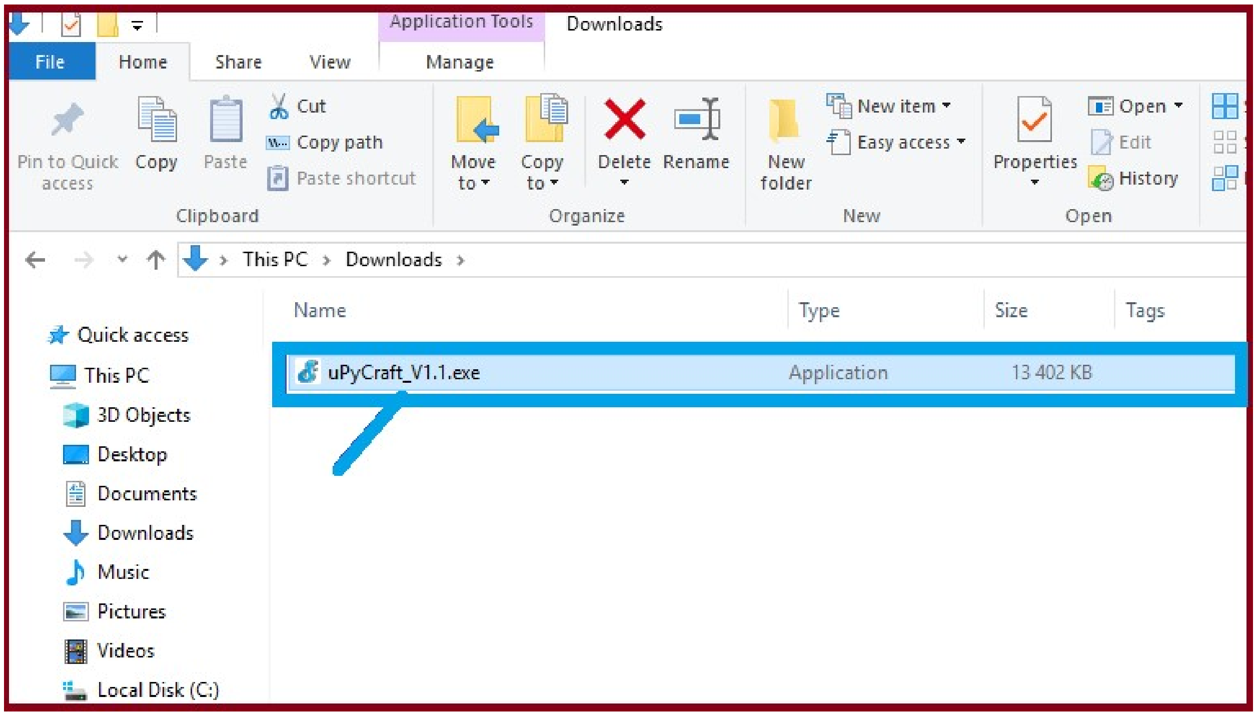
Now follow these steps to download and install uPyCraft IDE:

1. In order to download the uPyCraft IDE, go to official link and download the .exe file for windows as shown below. You can also download for Linux and Mac if you are using Linux or Mac operating system.



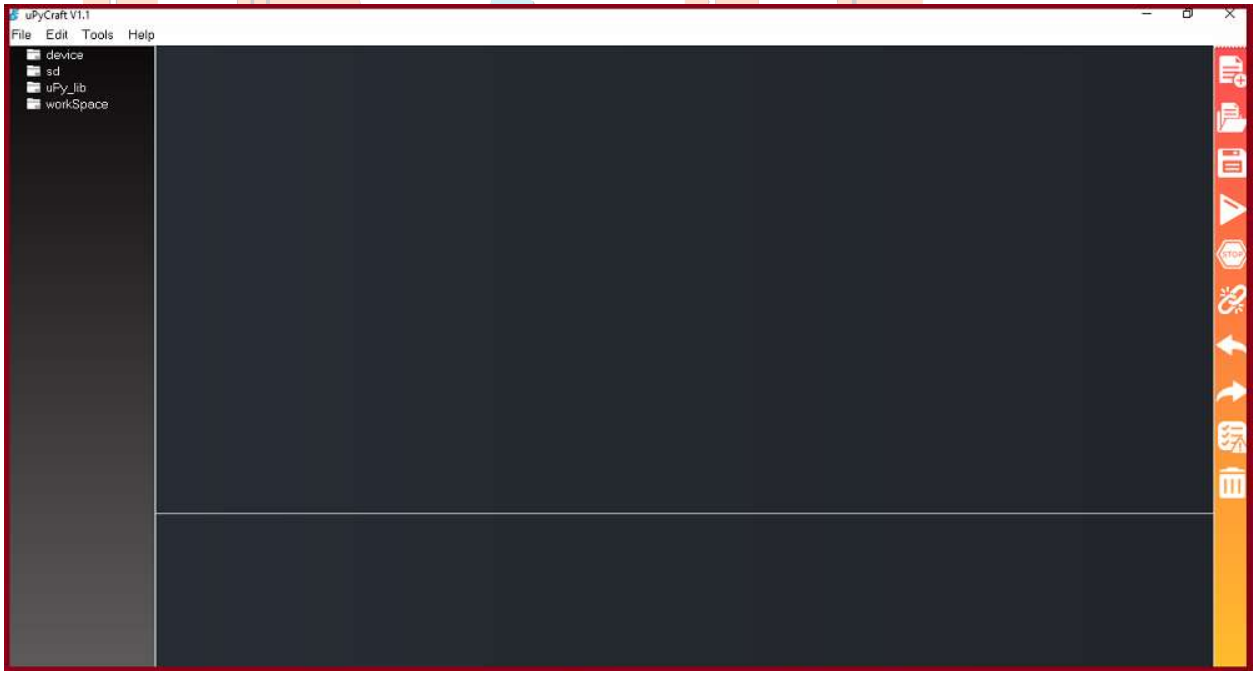
uPyCraft_V0.23.exe	ide0.23	4 years ago
uPyCraft_V0.24.exe	microbit	4 years ago
uPyCraft_V0.27.exe	UPY V0.27 | esp32 0817 | microbit 0907	4 years ago
uPyCraft_V0.28.exe	UPY V0.28	4 years ago
uPyCraft_V0.29.exe	uPy 0.29	3 years ago
uPyCraft_V0.30.exe	uPy for windows V0.30	3 years ago
uPyCraft_V1.0.exe	V1.0	3 years ago
uPyCraft_linux_V0.30	supoort linux IDE	3 years ago
uPyCraft_linux_V1.0	V1.0	3 years ago
uPyCraft_mac_V1.0.zip	V1.0	3 years ago

2. After that click on the *uPyCraft_VX.exe* which you download in the previous step:



Name	Type	Size	Tags
uPyCraft_V1.1.exe	Application	13 402 KB	

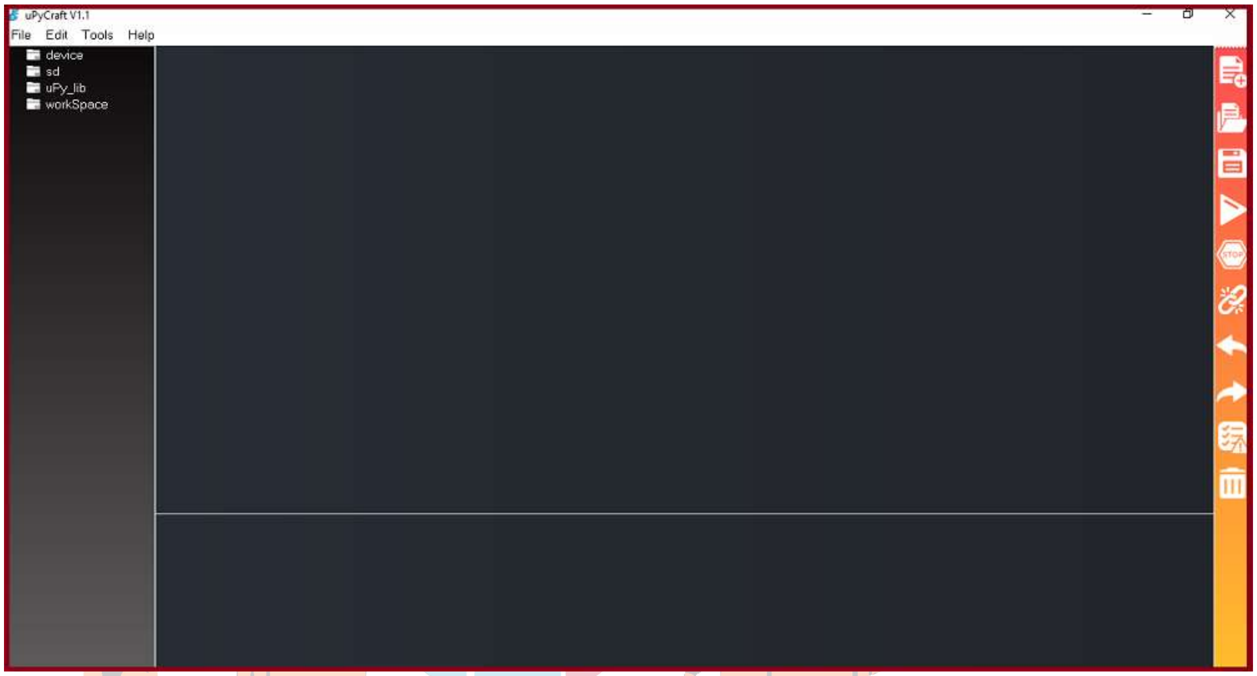
3. After installing the IDE, click on its icon and the following screen will appear.



Till now we have downloaded and installed uPyCraft IDE. We will use this IDE to write firmware and upload to Raspberry Pi Pico.

uPyCraft IDE Introduction

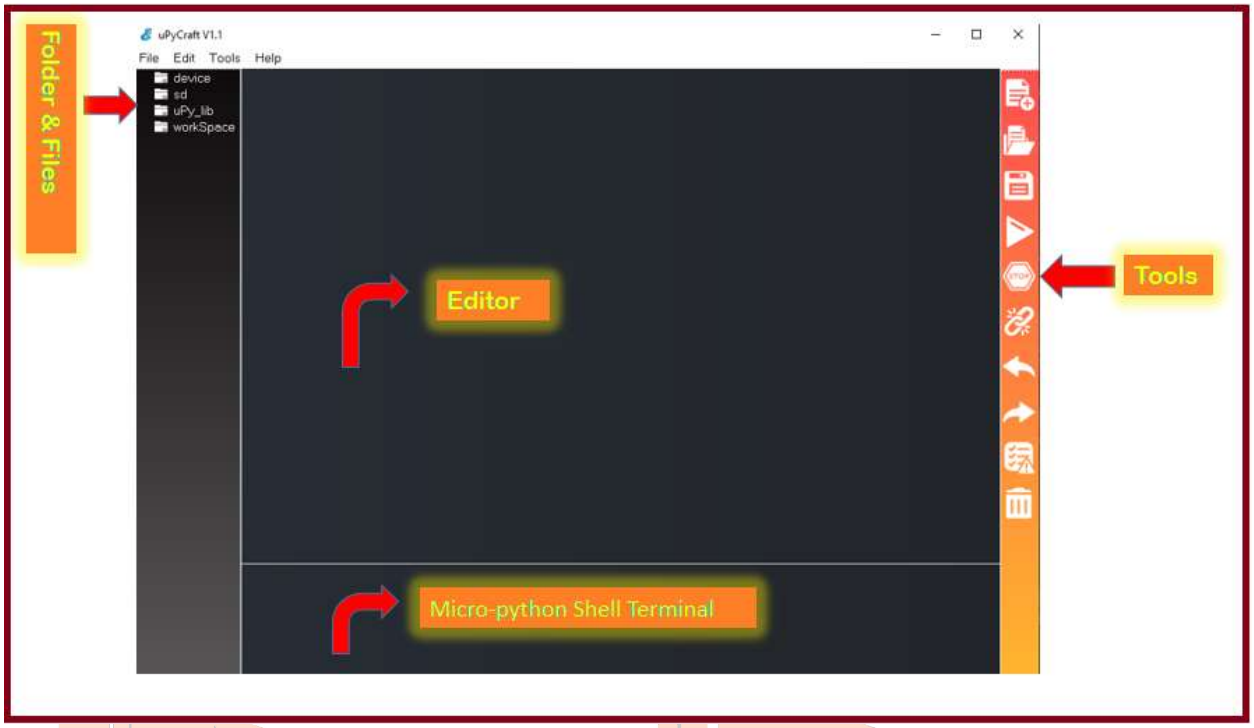
Now lets explore different windows and components of uPyCraft IDE. First open uPyCraft IDE, you will see a window like this:



uPyCraft IDE is a integrated development environment which is used to program development boards in MicroPython language. It makes the steps for firmware development, code debugging and uploading code to the Pi Pico board very easier under a single package.

Now we will learn about the different sections which are found on the IDE. The picture below shows the four main sections found in the IDE.

- **Tools-** On the furthest right side, you can see the different icons which can perform multiple tasks.
- **Editor-** In Editor we write our program code which is then executed and run onto the Raspberry Pi Pico module serially
- **Folder and files-** This section is found at the extreme left hand side of the screen you can view all the files which are saved on your module.
- **Micro-python shell terminal-**All messages are displayed in this section including errors that are found at the bottom of the screen



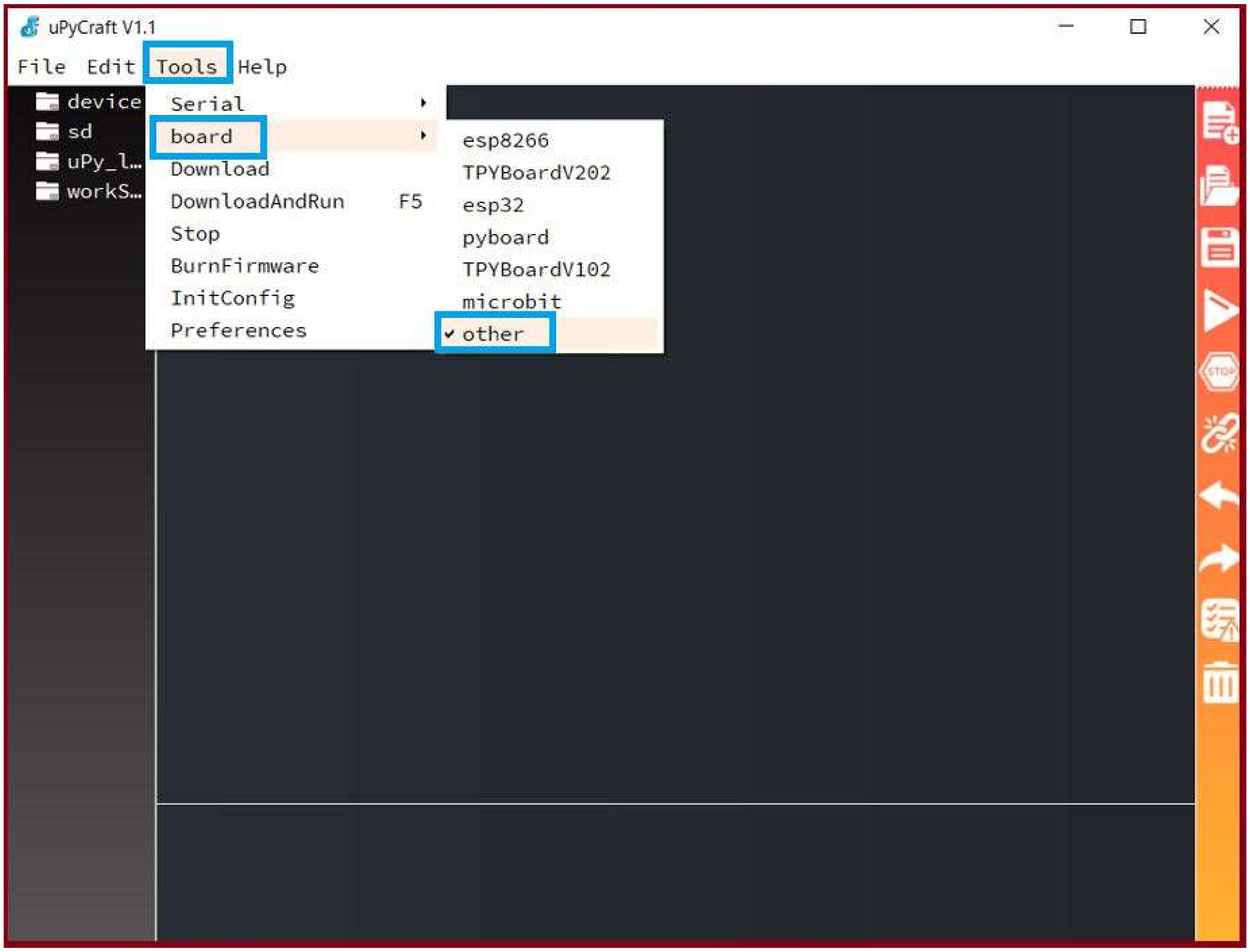
Writing Your First MicroPython Script

So far in this getting started tutorial on MicroPython for Raspberry Pi Pico, we learned to install software that is required to process and execute code on Pi Pico. Now let's write a simple script in uPyCraft IDE and execute it on our board. We will write a simple LED blinking script for Raspberry Pi Pico in Python programming language and will upload it to the board and see how it works.

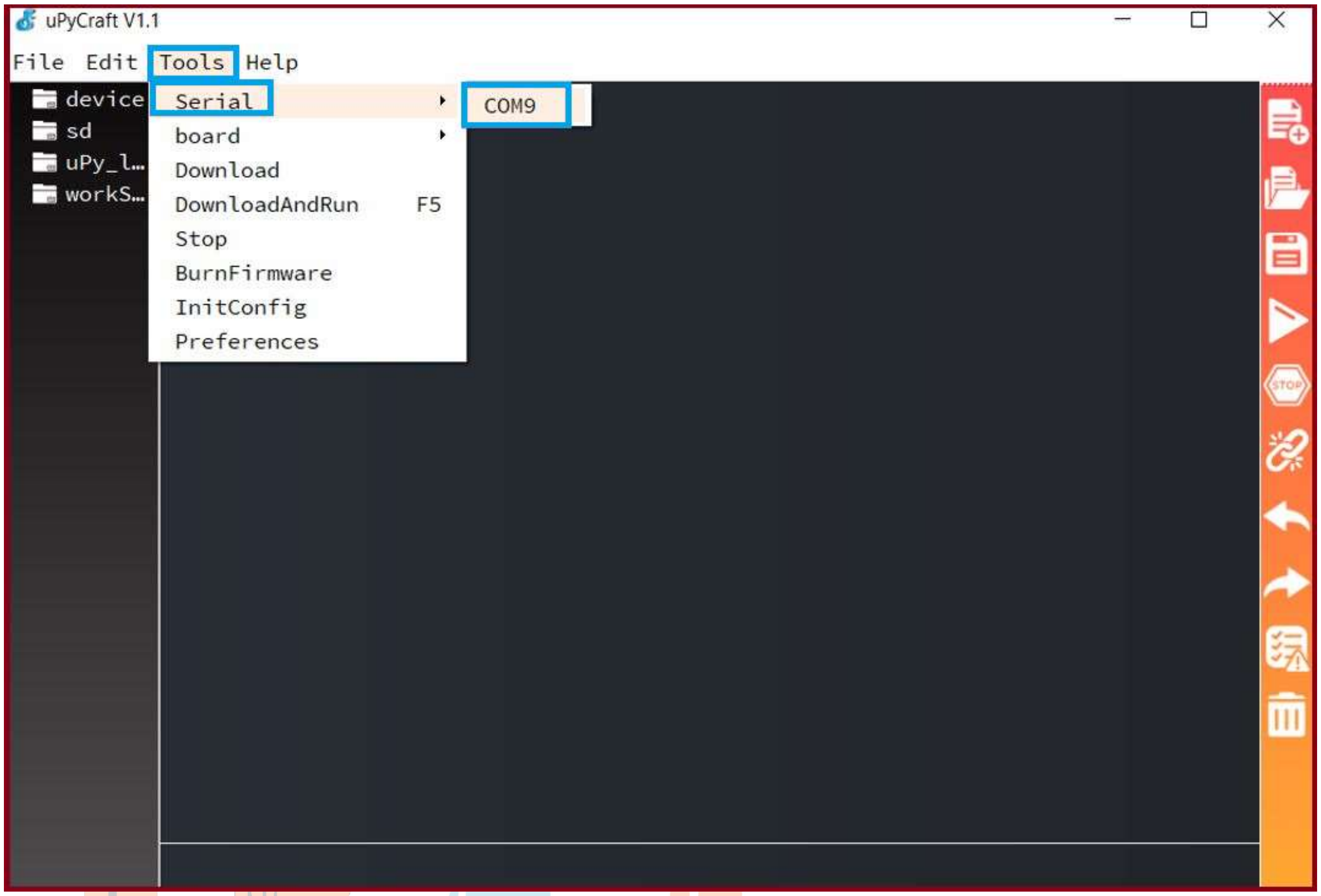
Serial Connection with Raspberry Pi Pico in uPyCraft IDE

Now let's see how to establish a communication between uPyCraft IDE and Raspberry Pi Pico.

- Connect your board to your computer using a USB cable.
- After that go to **tools > boards** and select 'other.'

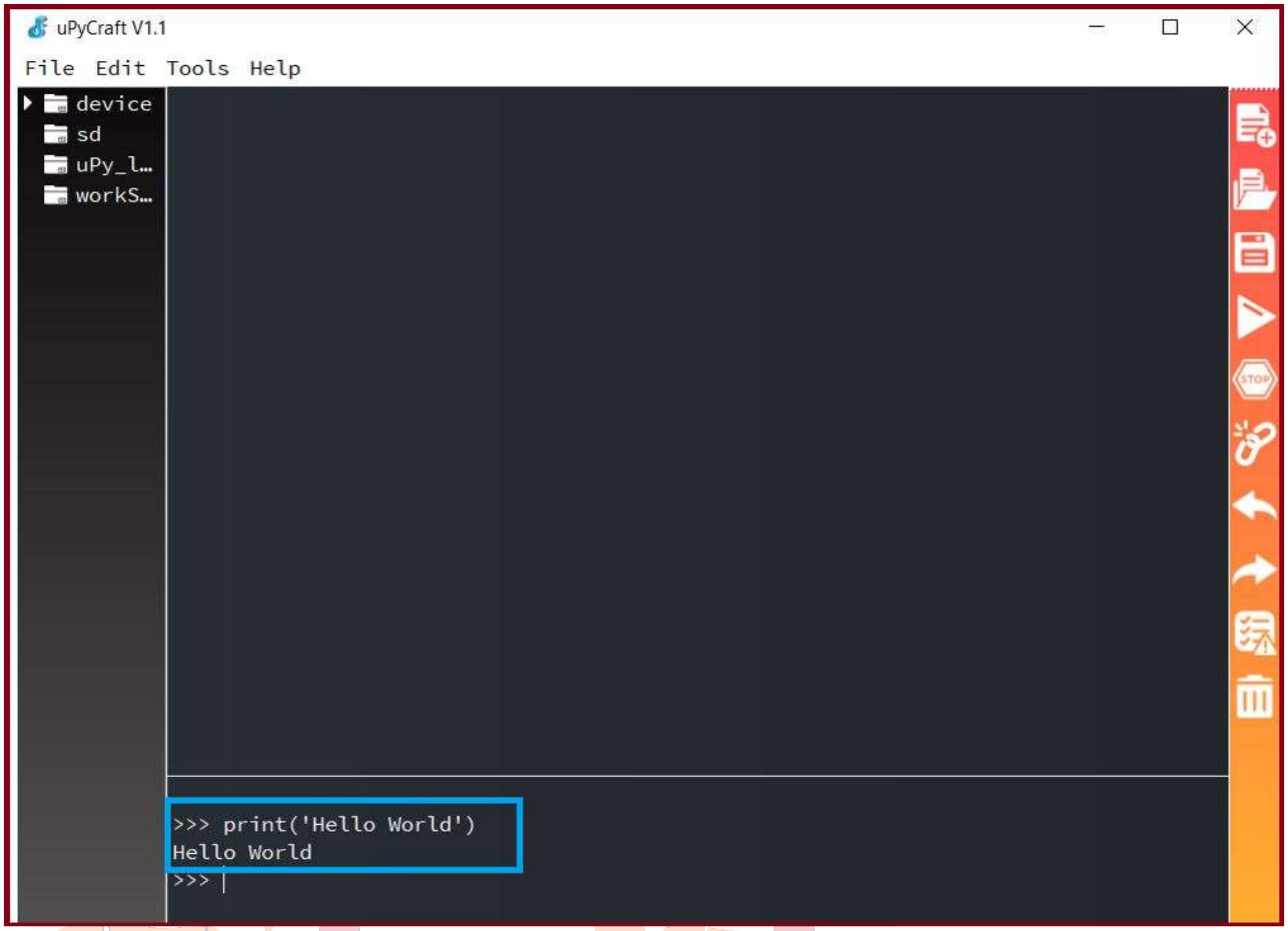


- Next, go to **tools > Serial** and select the serial port through which your board is connected.



The **>>>** will show up in the Shell window of uPyCratf IDE and it shows that a successful connection has been developed with your board. Now to see its working, you can type this command on the shell window and it will print “Hello World” on console in response.

```
>>> print('Hello World')
Hello World
>>>
```



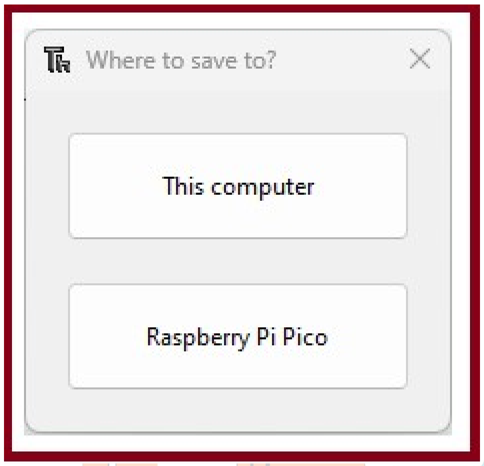
```
>>> print('Hello World')
Hello World
>>> 
```

Note: If you don’t see the output of print() command on the shell console of uPyCraft IDE that means your board has not made a serial connection with your computer

Saving the Script to the Raspberry Pi Pico Board

Stop the execution of the previous program by clicking on the **Stop** button if you haven't already.

With the code copied to the file, click on the **Save** icon. Then, select **Raspberry Pi Pico**.



Save the file with the following name: main.py.

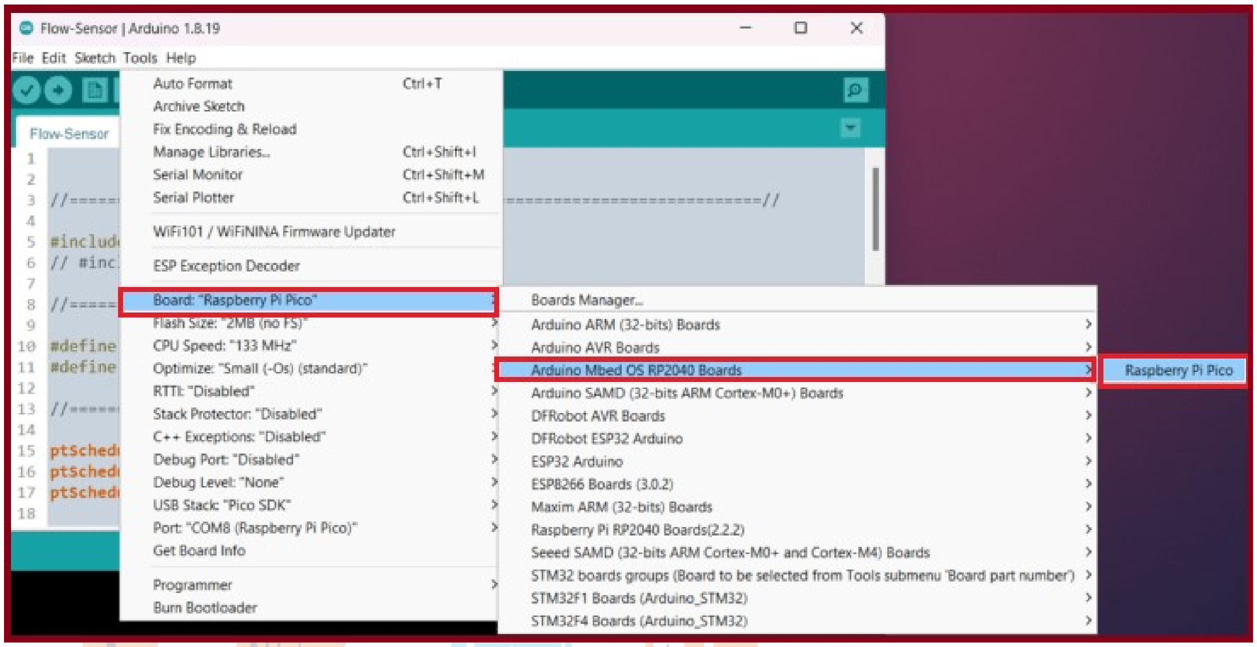
Note: *When you name a file main.py, the Raspberry Pi Pico will run that file automatically on boot. If you call it a different name, it will still be saved on the board file system, but it will not run automatically on boot.*

Finally, click **OK** to proceed.

Now, you can remove and apply power again to the board, or you can even power it using a different power supply that is not your computer. You'll notice that the board will automatically start blinking the LED when it starts

Setting up Arduino IDE

The first time you connect your Pico board to the computer, you will not see a COM port associated with it. This is because the Pico is still not ready to enumerate its USB port as a serial port. You need to modify the bootloader section of the RP2040 and enable the USB-CDC port.


Flow-Sensor | Arduino 1.8.19
File Edit Sketch Tools Help
Flow-Sensor
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP Exception Decoder
Get Board Info
Programmer
Burn Bootloader
Boards Manager...
Arduino ARM (32-bits) Boards
Arduino AVR Boards
Arduino Mbed OS RP2040 Boards
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
DFRobot AVR Boards
DFRobot ESP32 Arduino
ESP32 Arduino
ESP8266 Boards (3.0.2)
Maxim ARM (32-bits) Boards
Raspberry Pi RP2040 Boards(2.2.2)
Seeed SAMD (32-bits ARM Cortex-M0+ and Cortex-M4) Boards
STM32 boards groups (Board to be selected from Tools submenu 'Board part number')
STM32F1 Boards (Arduino_STM32)
STM32F4 Boards (Arduino_STM32)
Raspberry Pi Pico

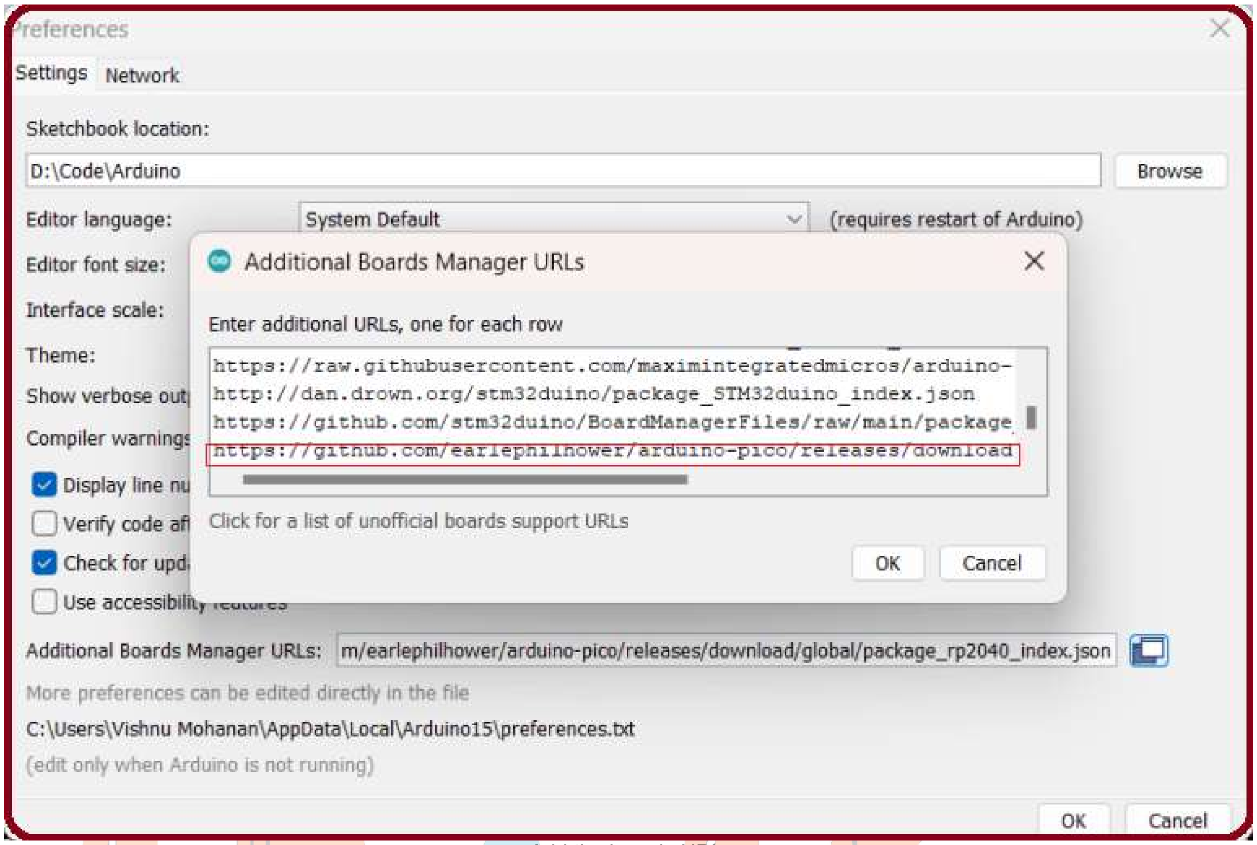
```
Board: "Raspberry Pi Pico"
Flash Size: "2MB (no FS)"
CPU Speed: "133 MHz"
Optimize: "Small (-Os) (standard)"
RTTI: "Disabled"
Stack Protector: "Disabled"
C++ Exceptions: "Disabled"
Debug Port: "Disabled"
Debug Level: "None"
USB Stack: "Pico SDK"
Port: "COM8 (Raspberry Pi Pico)"
```

Select Raspberry Pico from Arduino Mbed OS RP2040 Boards

To use the Arduino-Pico core, you should first add the boards URL to the Arduino IDE. Open the Preferences and add the following URL to the list.

https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json

Then open the Boards Manager and search for “pico”. From the list, install the **Raspberry Pi Pico/RP2040** item.


Preferences

Sketchbook location: D:\Code\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size:

Interface scale:

Theme:

Show verbose out

Compiler warnings

- [x] Display line nu
- [ ] Verify code af
- [x] Check for upd
- [ ] Use accessibility features

Additional Boards Manager URLs

Enter additional URLs, one for each row

```
https://raw.githubusercontent.com/maximintegratedmicros/arduino-
http://dan.drown.org/stm32duino/package_STM32duino_index.json
https://github.com/stm32duino/BoardManagerFiles/raw/main/package
https://github.com/earlephilhower/arduino-pico/releases/download
```

Click for a list of unofficial boards support URLs

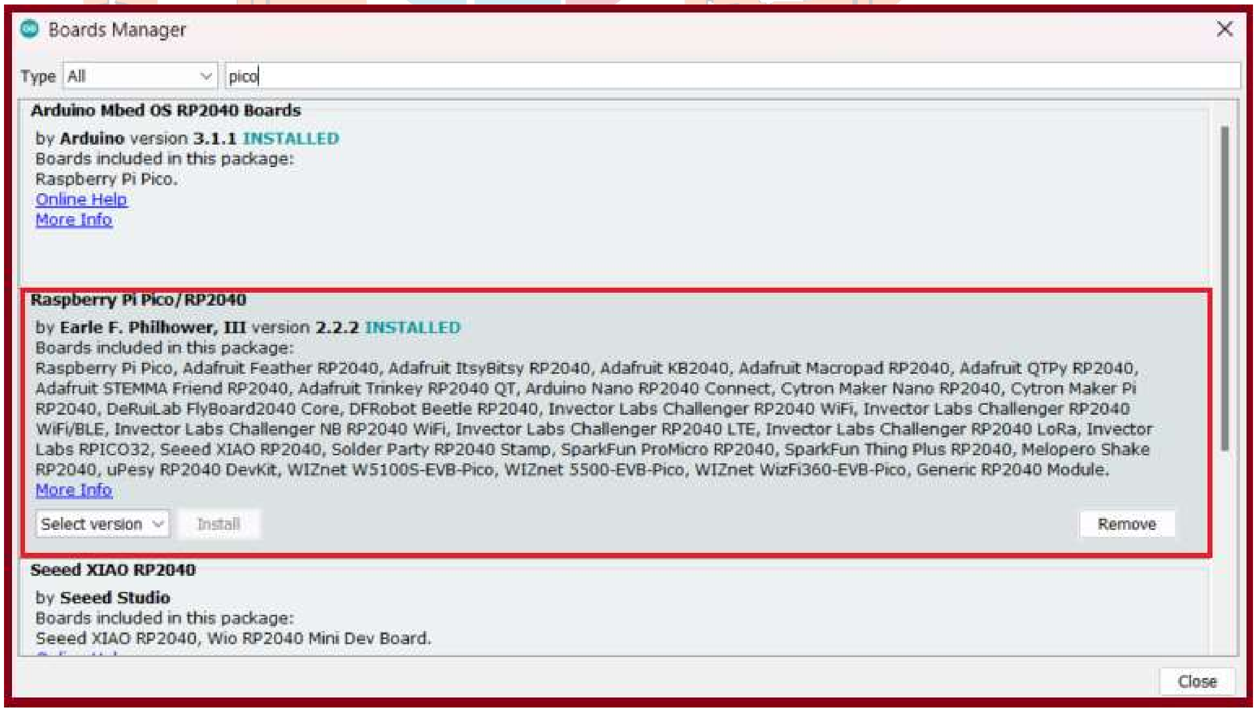
Additional Boards Manager URLs: m/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json

More preferences can be edited directly in the file

C:\Users\Vishnu Mohanan\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

Add the boards URL


Boards Manager

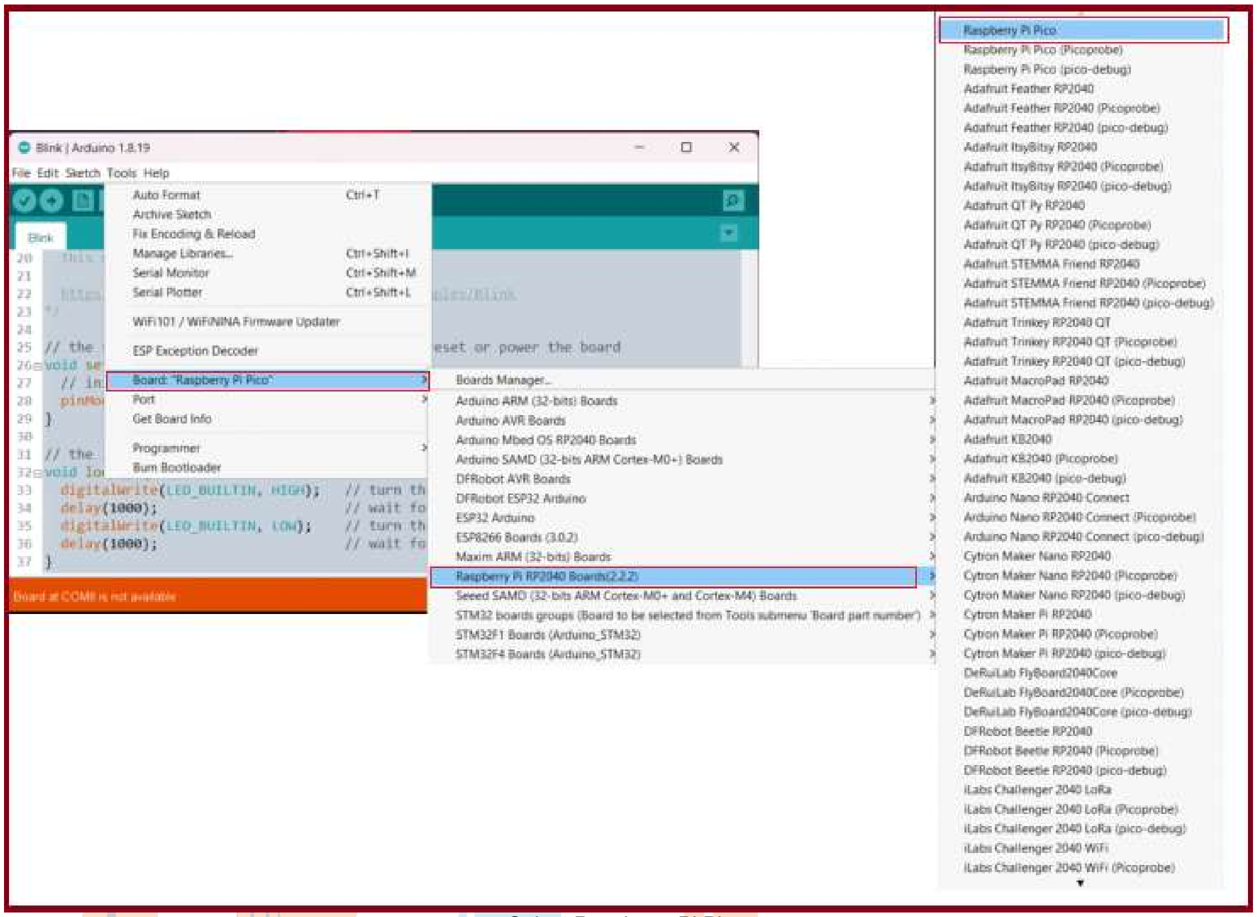
Type All pico

Arduino Mbed OS RP2040 Boards
by **Arduino** version **3.1.1** **INSTALLED**
Boards included in this package:
Raspberry Pi Pico.
Online Help
More Info

Raspberry Pi Pico/RP2040
by **Earle F. Philhower, III** version **2.2.2** **INSTALLED**
Boards included in this package:
Raspberry Pi Pico, Adafruit Feather RP2040, Adafruit ItsyBitsy RP2040, Adafruit KB2040, Adafruit Macropad RP2040, Adafruit QTPy RP2040, Adafruit STEMMA Friend RP2040, Adafruit Trinkey RP2040 QT, Arduino Nano RP2040 Connect, Cytron Maker Nano RP2040, Cytron Maker Pi RP2040, DeRuiLab FlyBoard2040 Core, DFRobot Beetle RP2040, Invector Labs Challenger RP2040 WiFi, Invector Labs Challenger RP2040 WiFi/BLE, Invector Labs Challenger NB RP2040 WiFi, Invector Labs Challenger RP2040 LTE, Invector Labs Challenger RP2040 LoRa, Invector Labs RPICO32, Seeed XIAO RP2040, Solder Party RP2040 Stamp, SparkFun ProMicro RP2040, SparkFun Thing Plus RP2040, Melopero Shake RP2040, uPesy RP2040 DevKit, WIZnet W5100S-EVB-Pico, WIZnet 5500-EVB-Pico, WIZnet WizFi360-EVB-Pico, Generic RP2040 Module.
More Info

Seeed XIAO RP2040
by **Seeed Studio**
Boards included in this package:
Seeed XIAO RP2040, Wio RP2040 Mini Dev Board.

Install the Raspberry Pi Pico/RP2040


Blink | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP Exception Decoder
Board: "Raspberry Pi Pico"
Port
Get Board Info
Programmer
Burn Bootloader
Board at COM9 is not available
Boards Manager...
Arduino ARM (32-bits) Boards
Arduino AVR Boards
Arduino Mbed OS RP2040 Boards
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
DFRobot AVR Boards
DFRobot ESP32 Arduino
ESP32 Arduino
ESP8266 Boards (3.0.2)
Maxim ARM (32-bits) Boards
Raspberry Pi RP2040 Boards(2.2.2)
Seeed SAMD (32-bits ARM Cortex-M0+ and Cortex-M4) Boards
STM32 boards groups (Board to be selected from Tools submenu 'Board part number')
STM32F1 Boards (Arduino_STM32)
STM32F4 Boards (Arduino_STM32)
Raspberry Pi Pico
Raspberry Pi Pico (Picoprobe)
Raspberry Pi Pico (pico-debug)
Adafruit Feather RP2040
Adafruit Feather RP2040 (Picoprobe)
Adafruit Feather RP2040 (pico-debug)
Adafruit ItsyBitsy RP2040
Adafruit ItsyBitsy RP2040 (Picoprobe)
Adafruit ItsyBitsy RP2040 (pico-debug)
Adafruit QT Py RP2040
Adafruit QT Py RP2040 (Picoprobe)
Adafruit QT Py RP2040 (pico-debug)
Adafruit STEMMA Friend RP2040
Adafruit STEMMA Friend RP2040 (Picoprobe)
Adafruit STEMMA Friend RP2040 (pico-debug)
Adafruit Trinkey RP2040 QT
Adafruit Trinkey RP2040 QT (Picoprobe)
Adafruit Trinkey RP2040 QT (pico-debug)
Adafruit MacroPad RP2040
Adafruit MacroPad RP2040 (Picoprobe)
Adafruit MacroPad RP2040 (pico-debug)
Adafruit KB2040
Adafruit KB2040 (Picoprobe)
Adafruit KB2040 (pico-debug)
Arduino Nano RP2040 Connect
Arduino Nano RP2040 Connect (Picoprobe)
Arduino Nano RP2040 Connect (pico-debug)
Cytron Maker Nano RP2040
Cytron Maker Nano RP2040 (Picoprobe)
Cytron Maker Nano RP2040 (pico-debug)
Cytron Maker Pi RP2040
Cytron Maker Pi RP2040 (Picoprobe)
Cytron Maker Pi RP2040 (pico-debug)
DeRuiLab FlyBoard2040Core
DeRuiLab FlyBoard2040Core (Picoprobe)
DeRuiLab FlyBoard2040Core (pico-debug)
DFRobot Beetle RP2040
DFRobot Beetle RP2040 (Picoprobe)
DFRobot Beetle RP2040 (pico-debug)
iLabs Challenger 2040 LoRa
iLabs Challenger 2040 LoRa (Picoprobe)
iLabs Challenger 2040 LoRa (pico-debug)
iLabs Challenger 2040 WiFi
iLabs Challenger 2040 WiFi (Picoprobe)

```
  digitalWrite(LED_BUILTIN, HIGH);   // turn th
  delay(1000);                       // wait fo
  digitalWrite(LED_BUILTIN, LOW);    // turn th
  delay(1000);                       // wait fo
}
```

Select Raspberry Pi Pico

After the installation is complete, go to **Tools → Board → Raspberry Pi RP2040 Boards (2.2.2)** and choose Raspberry Pi Pico from the list. If you have connected a fresh Pico board, the COM would not have been generated yet. But that's okay. The Arduino-Pico core can automatically detect the Pico board and modify the bootloader. You don't need to press any buttons. Simply open an example sketch and click the **Upload** button.

The Arduino-Pico core flashes the binary file by making the Pico board a mass storage device for a short duration, and prints the following lines to the console.

```
Sketch uses 50308 bytes (2%) of program storage space. Maximum is 2093056 bytes.
Global variables use 7236 bytes (2%) of dynamic memory, leaving 254908 bytes for local variables. Maximum is 262144
bytes.
C:\Users\Dhiru Yadav \AppData\Local\Arduino15\packages\rp2040\tools\pqt-python3\1.0.1-base-3a57aed/python3 -I
C:\Users\Dhiru Yadav\AppData\Local\Arduino15\packages\rp2040\hardware\rp2040\2.2.2/tools/uf2conv.py --serial COM9 --
family RP2040 --deploy C:\Users\DHIRU~1\AppData\Local\Temp\arduino_build_273735/Blink.ino.uf2
Resetting COM9
Converting to uf2, output size: 107008, start address: 0x2000
Flashing G: (RPI-RP2)
Wrote 107008 bytes to G:/NEW.UF2
```

The COM port will be disabled for a short duration but it will be back as soon as the flashing process is complete. Compared to the RP2040 Mbed OS core, the Arduino-

Pico core has much more customization options available since it is using the official C/C++ SDK directly. For this reason, we suggest you use the Arduino-Pico core whenever possible. The code size will be also smaller when using the Arduino-Pico core.

In the upcoming posts, we will explore the features of the RP2040 microcontroller in detail. We will also show how you can create a standalone C/C++ project that uses the official SDK.