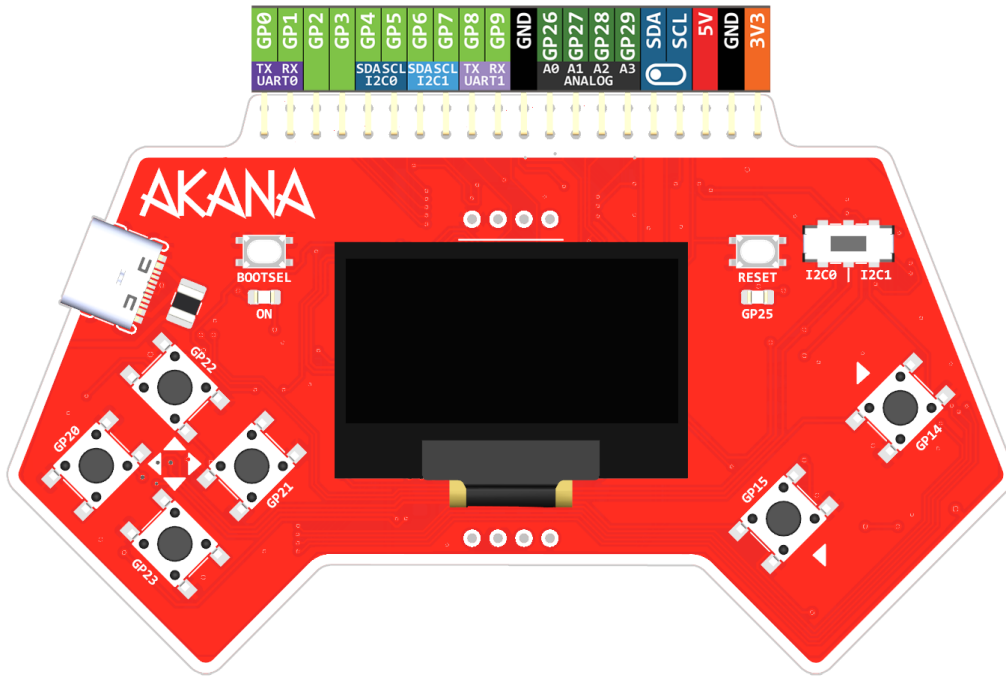


AKANA R1 CODING MANUAL

ARDUINO IDE (MBED OS / PICO SDK) | MICROPYTHON | CIRCUITPYTHON



CONTENTS

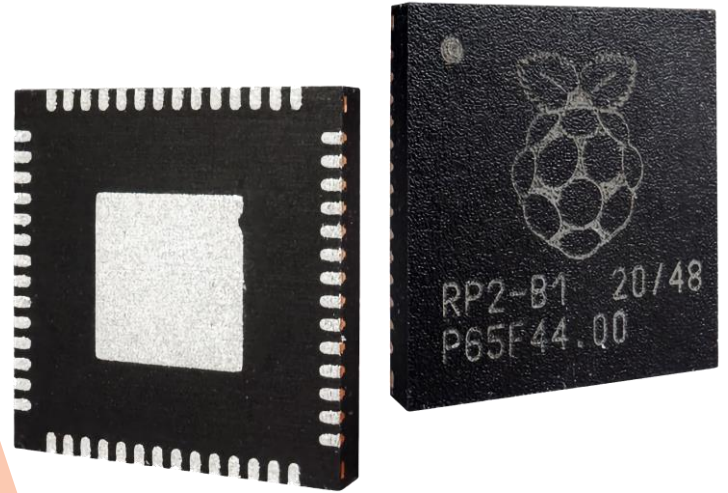
Why RP2040?	2
UF2 Bootloader	3
Installing Arduino Mbed OS on Akana R1	4
Coding Akana R1 with Arduino Mbed OS	5
Installing Arduino-Pico on Akana R1	6
Coding Akana R1 with Arduino-Pico	7
Installing MicroPython on Akana R1	8
Coding Akana R1 with MicroPython	9
Installing CircuitPython on Akana R1	10
Coding Akana R1 with CircuitPython	11
Versions	12

WHY RP2040?

METE HOCA Akana R1 is built on the **RP2040** microcontroller developed by the Raspberry Pi team, so its usage and software support are the same as that of the **Raspberry Pi Pico**, which also uses the RP2040.

Unlike the limited **32 KB** flash memory of the **Arduino Uno R3**, the Raspberry Pi Pico based on the RP2040 has **2 MB**, and the Akana R1 comes with 2 MB, optionally 8 MB or 16 MB of flash memory.

Moreover, while the Uno R3 has a single-core processor running at 16 MHz, the RP2040 features a **dual-core** processor running at 133 MHz. These features make the RP2040 quite versatile.



The Arduino team also recognized the capabilities of this powerful microcontroller and produced a model called the **Arduino Nano RP2040 Connect**, which uses the RP2040. The Arduino team enabled the RP2040 to be programmed with **Mbed OS** in the **Arduino IDE** and prepared software that supports both this board and the Raspberry Pi Pico directly.

Moreover, there's another method to program the RP2040 using the **Arduino IDE**. Software prepared by **Earle F. Philhower**, utilizing the RP2040's **Pico SDK** software kit, allows for much more efficient use of boards based on the RP2040.

MicroPython, a lightweight version of the powerful **Python** programming language tailored for microcontrollers, also supports the RP2040. There are numerous libraries and support available for using this microcontroller in MicroPython. The **PicoBricks** educational kit developed by the **Robotistan** team is also based on MicroPython, alongside block coding.

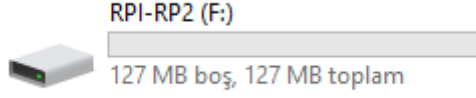


One of the biggest supporters of the RP2040 is **Adafruit**. This giant maker manufacturer has developed its own coding framework called **CircuitPython**, built on MicroPython but brought to a very different level with additional features, and it also provides excellent support in terms of libraries.

In summary, it's possible to code the Akana R1 in the Arduino IDE using 2 different methods and in the Python language using 2 different methods as well.

UF2 BOOTLOADER

Akana R1 can be easily programmed thanks to the **UF2 Bootloader** of the RP2040 it is built upon. With the UF2 Bootloader, the Akana R1 appears as a USB storage device on the computer. Furthermore, it does not require any driver installation on Windows 10, Mac, or even Linux.



So, what is UF2?

UF2 stands for **USB Flashing Format**. Developed by Microsoft, this structure allows codes to be easily loaded onto microcontrollers by copying them to removable USB drives. Thus, codes converted into UF2 files can be loaded easily with a simple drag-and-drop action.

To put the Akana R1 into UF2 Bootloader mode, the **BOOTSEL** button must be held down while connecting the USB cable. The board remains in this mode until the power is cut off, a UF2 file is copied, or the RESET button is pressed. You can tell that the Akana R1 has entered UF2 Bootloader mode when it appears as a USB storage device named **RPI-RP2** on the computer.



Another method to put the Akana R1 into UF2 mode is to hold the BOOTSEL button while pressing the **RESET** button once, with the USB cable connected. This method may require a bit of dexterity.

WARNING: Long USB cables or USB extension cables may impair signal integrity, preventing the Akana R1 from entering UF2 Bootloader mode.

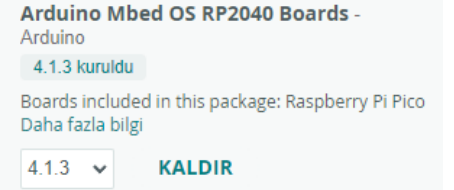
When switching between Mbed OS, Pico SDK, MicroPython, or CircuitPython programming methods, the Akana R1 must be put into UF2 Bootloader mode to ensure it is detected correctly.

INSTALLING ARDUINO MBED OS ON AKANA R1

The Arduino team, which has been increasingly using more powerful microcontrollers in their new products, has developed its own software based on **Mbed OS**, designed for next-generation microcontrollers with **ARM** architecture. Mbed OS is a lightweight yet powerful **RTOS** (Real-Time Operating System) framework that can be used in relatively weak ARM processors like microcontrollers.

This enables new-generation Arduino boards equipped with **32-bit ARM Cortex** series processors to perform tasks concurrently, providing capabilities that were previously unimaginable. The shared infrastructure of Mbed OS allows different models of ARM processor boards to run the same code easily.

The Arduino Mbed OS software has also been adapted for the RP2040, with a specific version released for the Raspberry Pi Pico. This version can be installed by selecting **Arduino Mbed OS RP2040 Boards** in the Arduino IDE's board manager.



Once the Raspberry Pi Pico board is selected, the Akana R1 will be ready for coding.

First, Mbed OS needs to be installed on the Akana R1. The Arduino IDE does this automatically. The only steps are to put the Akana R1 into **UF2 Bootloader** mode using the **BOOTSEL** button and load a simple sketch. For example, Arduino's famous **Blink** sketch can be a good starting point. If the GP25 LED starts blinking, the upload has been successfully completed.

After the first sketch uploaded via UF2 Bootloader, the Akana R1 will also have a **serial (COM) port** provided by the Mbed OS software. To upload subsequent sketches, it's essential to have selected the correct serial port. For this, it is sufficient to choose the relevant serial port under **Tools > Port** in the Arduino IDE menu.

Now, before each sketch upload, the Arduino IDE automatically puts the Akana R1 into UF2 Bootloader mode using this serial port and uploads the new sketch.

CODING AKANA R1 WITH ARDUINO MBED OS

We mentioned that **Mbed OS** enhances the capabilities of the already quite powerful RP2040 microcontroller. However, the Arduino team chose not to include some features of the RP2040 that are absent in most other Arduino models in the software built on Mbed OS.



arm MBED OS

One of the key features of the Akana R1 is the I2C channel switch, which allows selection between **I2C0** and **I2C1** channels. This enables all connected components to be switched to the desired I2C channel at once. However, the Arduino team has not included the second I2C channel, I2C1, in the Mbed OS software. As a result, this second channel **cannot be used** with this software.

The I2C0 channel provided by Arduino Mbed OS can be used on pins GP4 and GP5 in the design of the Akana R1. If there is no need to use the I2C1 communication channel, there is no downside to using the Akana R1 with the **Arduino Mbed OS** software.

On the other hand, the **Adafruit SSD1306** library, which is the most commonly used Arduino library for the **SSD1306** chip-based I2C OLED screen on the Akana R1, also has issues with Mbed OS software.

Both of these problems can be overcome by making some adjustments to the software and library files; however, these changes are erased with each update, requiring them to be reapplied each time.



Another screen library that can be preferred instead of the Adafruit SSD1306 library is **u8g2**. Developed by the community, this library can be used not only to drive the SSD1306 chip screens on the Akana R1 but also many other **monochrome OLED screens**.

The u8g2 library can be accessed through Arduino's built-in **library manager**.

INSTALLING ARDUINO-PICO ON AKANA R1

Earle Philhower's Arduino-Pico software, prepared using the **Pico SDK** published by the Raspberry Pi team, can be added externally to the Arduino IDE. This software is fundamentally a community project without any manufacturer backing. As a result, it receives significant support and continues to be developed rapidly.

The maker giant **Adafruit** also recommends this software for the boards it developed based on the RP2040.

This software, built on the Pico SDK based on the C++ programming language we use in the Arduino IDE, allows for generating very lightweight code and is often faster than the **Mbed OS** preferred by Arduino.

Since the Arduino-Pico software is not integrated into the Arduino IDE, it needs to be installed using a different method. From the main menu of the Arduino IDE, go to **File > Preferences**, and in the Preferences window, click the button to the right of the **Additional Board Manager URLs** section. Then, go to a **new line** in the opened window, paste the source link of the software, and click **OK** to add Arduino-Pico to the list of board software in the Arduino IDE.

Source link;

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

After that, from the Arduino IDE main menu, select **Tools > Board > Board Manager**, search for RP2040 in the opened window, and install the **Raspberry Pi Pico/RP2040** software prepared by Earle Philhower from the list. Support for the **Akana R1** has been added with version **3.9.3** of the software.

Finally, select **METE HOCA Akana R1** from the Raspberry Pi Pico/RP2040 section under **Tools > Board** to complete the board selection.

CODING AKANA R1 WITH ARDUINO-PICO

The **Arduino-Pico** software allows for the seamless use of all the features of the RP2040 and does not have any issues with commonly used hardware and libraries.

After installing Arduino-Pico, it is necessary to put the Akana R1 into **UF2 Bootloader** mode to upload the first sketch. This way, the RP2040 can respond to the new software's requests. After the first sketch is uploaded, all that remains is to select the newly created **serial port** in the **Tools** menu of the Arduino IDE. Now, the Akana R1 is ready to be programmed with Arduino-Pico.

The Arduino-Pico software enables the easy use of both I2C channels on the RP2040. By default, the I2C0 channel is set to GP4 (SDA) and GP5 (SCL), while the I2C1 channel is set to GP26 (SDA) and GP27 (SCL). However, since it is not logical to use the analog input pins GP26 and GP27 for the I2C1 channel, **GP6 (SDA)** and **GP7 (SCL)** pins are used for this channel on the Akana R1.

Fortunately, changing the I2C channel pins in Arduino-Pico is very easy. The **setSDA** and **setSCL** commands can be used to change the pins for the desired channel. To do this, it is sufficient to place the following lines at the top of the **void setup()** function in the sketch:

```
Wire.setSDA(4);
Wire.setSCL(5);
Wire1.setSDA(6);
Wire1.setSCL(7);
```

In this way, the I2C0 channel can be used with **Wire**, and the I2C1 channel can be used with **Wire1**. With the support for Akana R1 that came with version 3.9.3 of the software, there is **no longer a need** to add these codes. The new software version allows the buttons on the Akana R1 to be used with the constants **BTN_UP**, **BTN_DOWN**, **BTN_LEFT**, **BTN_RIGHT**, **BTN_ENTER**, and **BTN_BACK** without dealing with pin numbers.

The Arduino-Pico software has no issues with using the **Adafruit SSD1306** library to drive the I2C OLED screen on the Akana R1. Below is the setup line for using the I2C1 channel (**Wire1**). For the I2C0 channel, just using **Wire** is sufficient.

```
Adafruit_SSD1306 MyDisplay(128, 64, &Wire1, -1);
```

For the other screen library, **u8g2**, different setup lines are required for both I2C channels. Below is the setup line for the I2C0 channel in the first line, and for the I2C1 channel in the second line. The **2ND** code seen in the second line indicates the I2C1 channel.

```
U8G2_SSD1306_128X64_NONAME_F_HW_I2C MyDisplay(U8G2_R0, U8X8_PIN_NONE);
U8G2_SSD1306_128X64_NONAME_F_2ND_HW_I2C MyDisplay(U8G2_R0, U8X8_PIN_NONE);
```

The Arduino-Pico software has a number of interesting features. For example, after placing the following line of code inside the **void setup()** function, it is possible to put the Akana R1 into **UF2 Bootloader** mode by **double-clicking the RESET button**:

```
rp2040.enableDoubleResetBootloader();
```

It is also possible to reset the Akana R1 or put it into UF2 Bootloader mode via code. To do this, it is sufficient to place the commands from the following line in the desired location.

```
rp2040.reboot(); // RESET
rp2040.rebootToBootloader(); // UF2 Bootloader
```

INSTALLING MICROPYTHON ON AKANA R1

MicroPython programming language operates on a very different logic compared to Arduino's coding methods. First, it's essential to understand this logic.

The code loading and execution system used in the Arduino IDE is fundamentally based on converting the code into a language that the RP2040 can understand, namely machine code, packaging it into a UF2 file, and then automatically putting the Akana R1 into UF2 Bootloader mode via a message sent over the serial port to copy the UF2 file to the USB disk.

MicroPython, on the other hand, has a very different structure. Languages like this rely on a software component called an **interpreter**, which is embedded in the device, to interpret the transferred code step by step rather than compiling the code. This means that whenever the written code is changed, there is no need for lengthy compilation and uploading processes. As soon as the code is sent to the Akana R1, the MicroPython interpreter starts executing the code from the first line.

Before starting to code with MicroPython, the interpreter software needs to be loaded onto the Akana R1. There are two methods for this. The first is to switch the Akana R1 into UF2 Bootloader mode, download the UF2 file for **Raspberry Pi Pico** from the **MicroPython website**, and upload it.

MicroPython download link;

https://micropython.org/download/RPI_PICO/

The second method is to use an open-source software called **Thonny**, which can also be preferred for later coding. Thonny allows downloading and installing the required version from its settings screen. The Thonny application can be downloaded for different operating systems from the link below. There is also a **portable** version of the software that does not require installation.

Thonny download link;

<https://github.com/thonny/thonny/releases/>

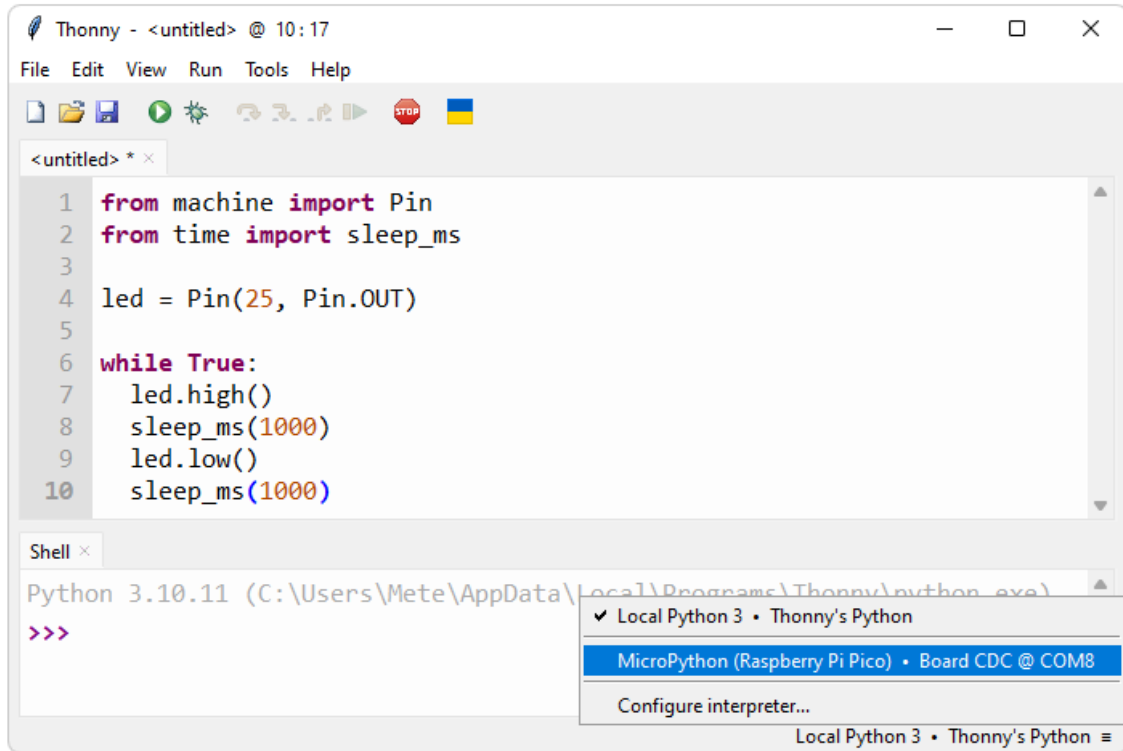
To load the MicroPython interpreter via Thonny, switch the Akana R1 into **UF2 Bootloader** mode, then select **Tools > Options** from the Thonny main menu to open the relevant window. In the **Interpreter** tab, select **MicroPython (RP2040)** at the top and try to automatically detect the port at the bottom to choose the microcontroller and the coding language.

Then, click on the **Install or Update MicroPython** link in the bottom right corner of the window, and in the opened installation window, select the **RPI-RP2** folder created by the Akana R1 in UF2 Bootloader mode from the **Target volume** section. Ensure that the **RP2** option is selected in the **MicroPython family** section.

After selecting **Raspberry Pi Pico** as the **Variant**, select the highest version that does not have a preview extension in the lower section and click the Install button to start the process. Thonny will download the selected MicroPython interpreter and load it onto the Akana R1.

CODING AKANA R1 WITH MICROPYTHON

There are several alternative software options for programming the Akana R1 with MicroPython. The most user-friendly ones are **Thonny**, **Mu Editor**, and Arduino's ongoing development project, **Arduino Lab for MicroPython**.



For beginners, Thonny and Mu Editor are recommended. Arduino Lab for MicroPython is still in its early stages and may become very useful in the future. It can be said that **Thonny** is the most suitable option for both beginners and advanced users.

Once you've ensured that MicroPython is installed on the Akana R1 and opened Thonny for the first time, clicking on the text in the lower right section to automatically select the hardware is sufficient to complete the setup.

In the photo above, you can see the MicroPython code that blinks the GP25 LED on the Akana R1. By writing the code and clicking the green run button in the menu, the code is sent to the Akana R1, and the LED starts to blink. It's important to remember that since there are no curly braces for code separation in Python, you need to indent the lines of code below using the **Tab** key.

```
from machine import Pin
from time import sleep_ms
```

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

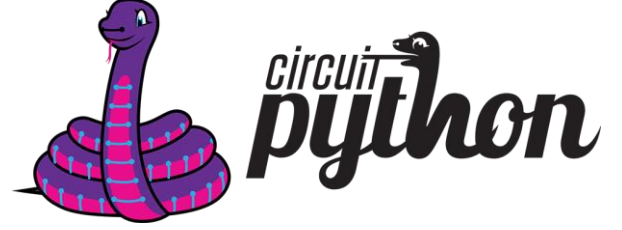
```
while True:
    led.high()
    sleep_ms(1000)
    led.low()
    sleep_ms(1000)
```

INSTALLING CIRCUITPYTHON ON AKANA R1

CircuitPython is a software developed by the maker giant **Adafruit** and is largely based on MicroPython. However, it differs significantly from MicroPython in terms of code structure, operation logic, and library compatibility.

In MicroPython, coding, file transfer, and execution processes are done through the serial port provided by the UF2 software. In contrast, development boards with CircuitPython installed appear continuously as a **USB storage disk** on the computer.

The codes and libraries loaded onto the boards are also stored within this USB disk, making them easily accessible through the computer's file manager.

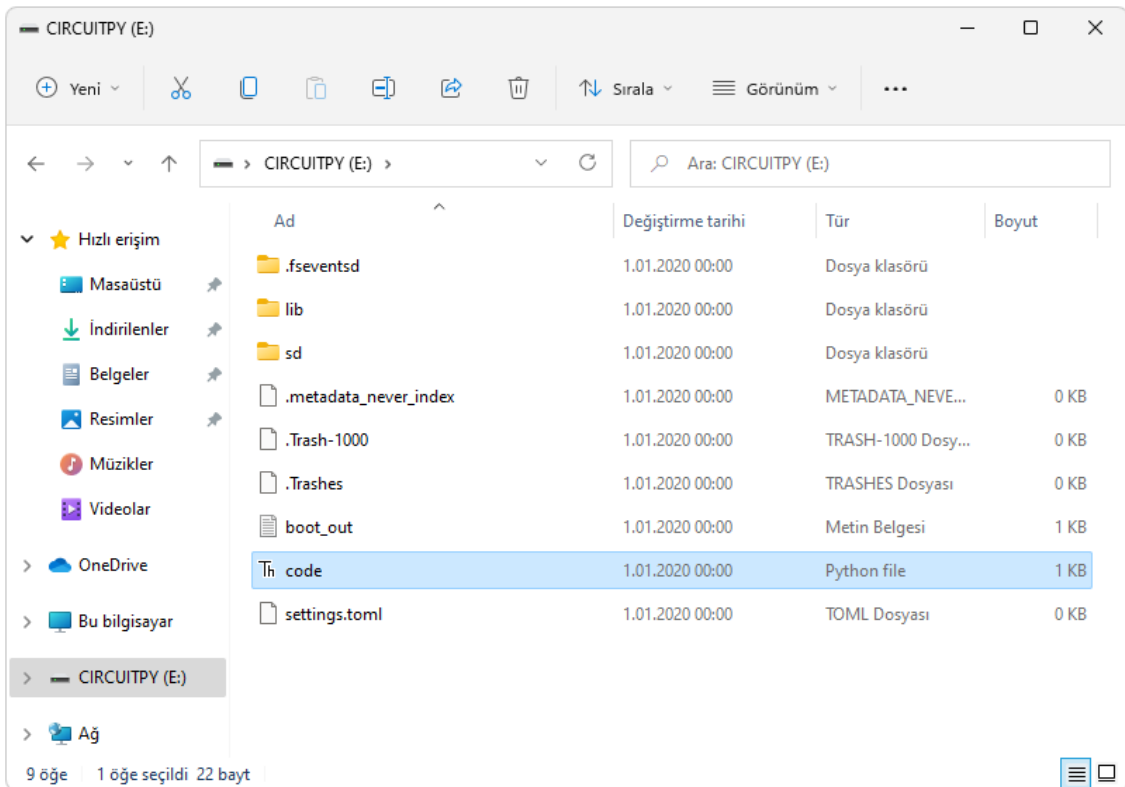


Before starting to code with CircuitPython, you need to install the interpreter software on the Akana R1. After switching the Akana R1 to UF2 Bootloader mode, simply download and install the UF2 file for **Raspberry Pi Pico** from the **CircuitPython website**.

CircuitPython download link;

https://circuitpython.org/board/raspberry_pi_pico/

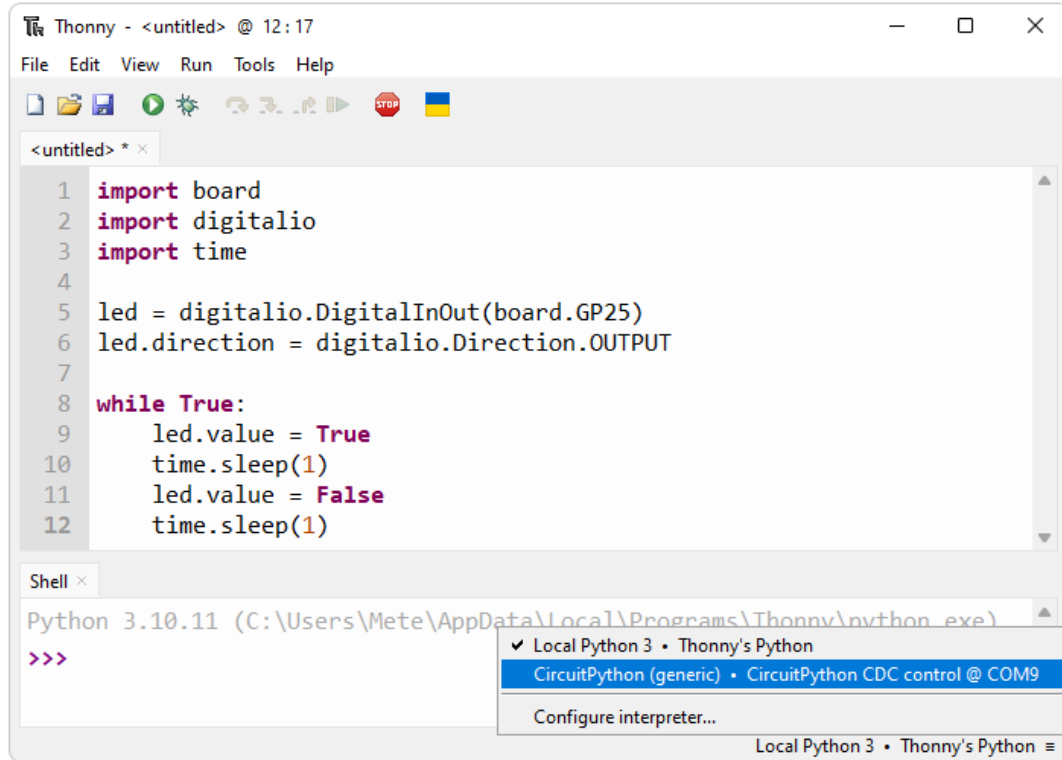
After the software is installed, a USB disk drive named **CIRCUITPY** will appear on the computer. Everything is contained within this drive. The **code.py** file inside the drive is the code file that will automatically run when the Akana R1 is powered on. This file can be opened and edited with any text editor or IDE. When the file is saved, the Akana R1 automatically restarts and runs the saved code.



CODING AKANA R1 WITH CIRCUITPYTHON

Coding the Akana R1 with the **CircuitPython** interpreter loaded is as easy as editing and saving the **code.py** file on the created disk drive. Of course, using an IDE software would be beneficial.

Adafruit recommends Mu Editor for beginners using CircuitPython. However, this software may become insufficient as one progresses in coding. Therefore, using **Thonny**, which is also recommended for MicroPython, could be a better choice.



To set Thonny to CircuitPython mode, simply click on the section seen in the bottom right of the image above and select the automatically detected device.

As seen, the code structure of CircuitPython differs slightly from MicroPython. Instead of **machine**, **board** and **digitalio** are used, and the pin configuration commands are also different.

```
import board
import digitalio
import time
```

```
led = digitalio.DigitalInOut(board.GP25)
led.direction = digitalio.Direction.OUTPUT
```

```
while True:
    led.value = 1
    time.sleep(1)
    led.value = 0
    time.sleep(1)
```



VERSIONS

Document Versions

Date	Changes
20 June 2024	Updates have been made regarding the support for Akana R1 in the Arduino-Pico package.
16 May 2024	Initial release

