Micropython

Using Visual Studio Code

Leo Kuipers – l.a.Kuipers@saxion.nl

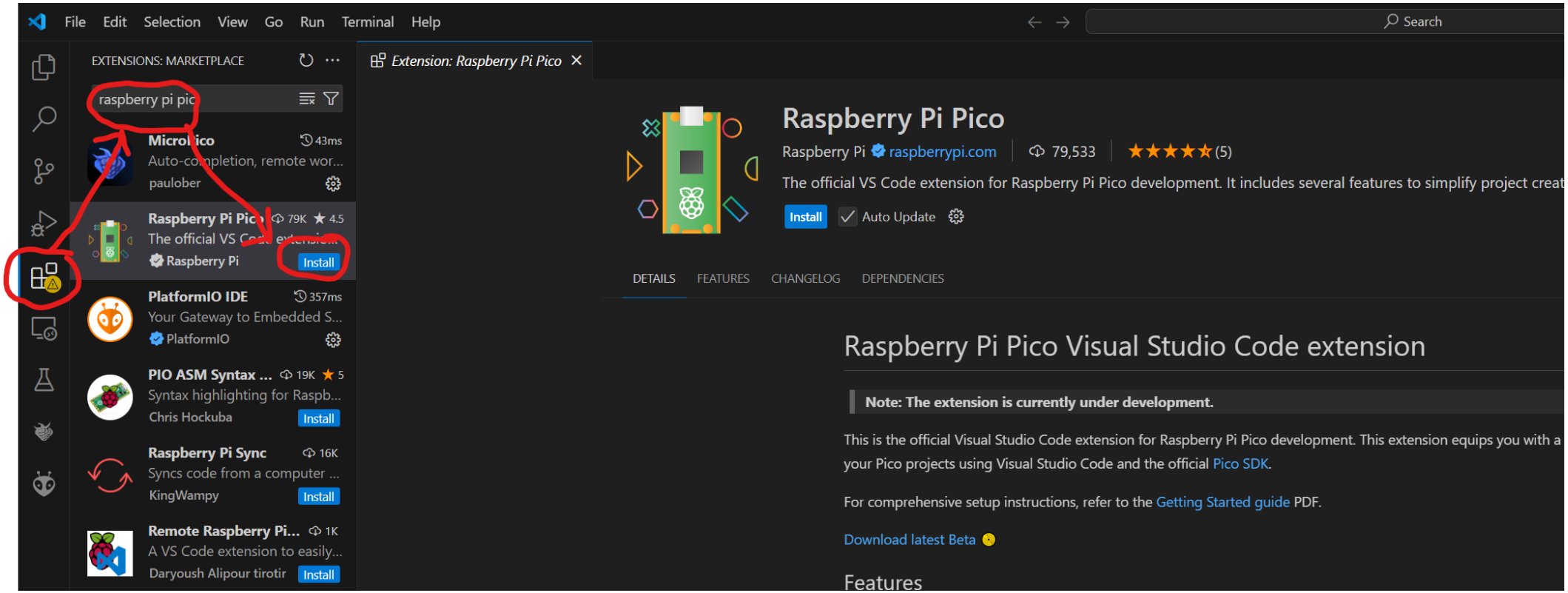
Why Visual Studio Code?

- It just works better.
- Thonny IDE is a meh editor.
- Visual Studio Code is a proper IDE.
- VS Code “Extensions” add a lot of functionality. E.g.:
  - Python Intellisense → = Autocomplete!
  - Git directly from within the editor
  - Github Copilot → Use Copilot AI to help you to program code

- Install visual studio code from: https://code.visualstudio.com

RP2040 Micropython support

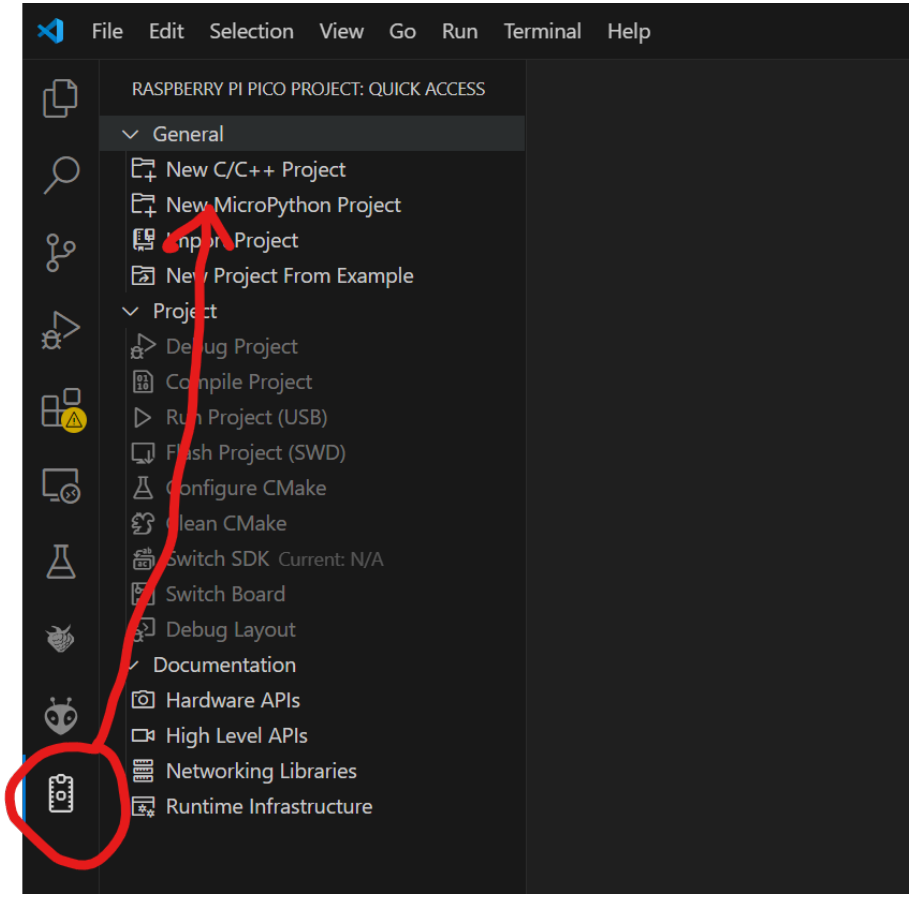
- Install the Raspberry Pi Pico extension



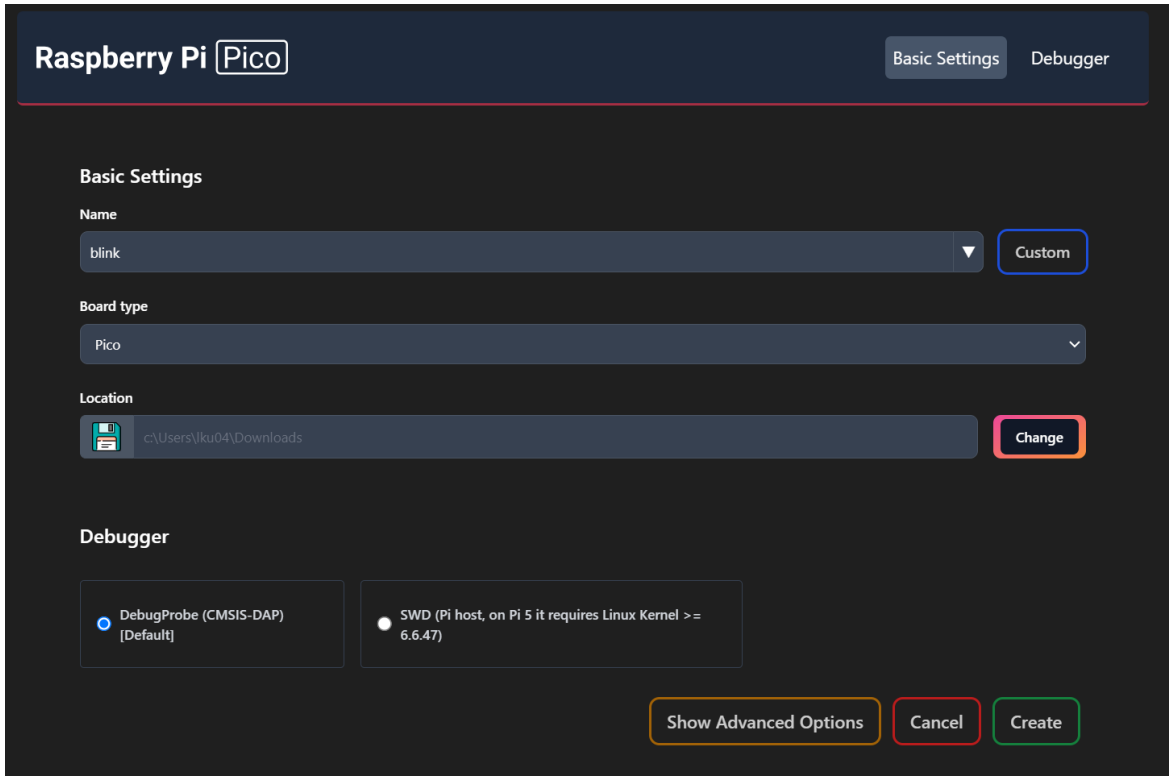
Raspberry Pi Pico extension

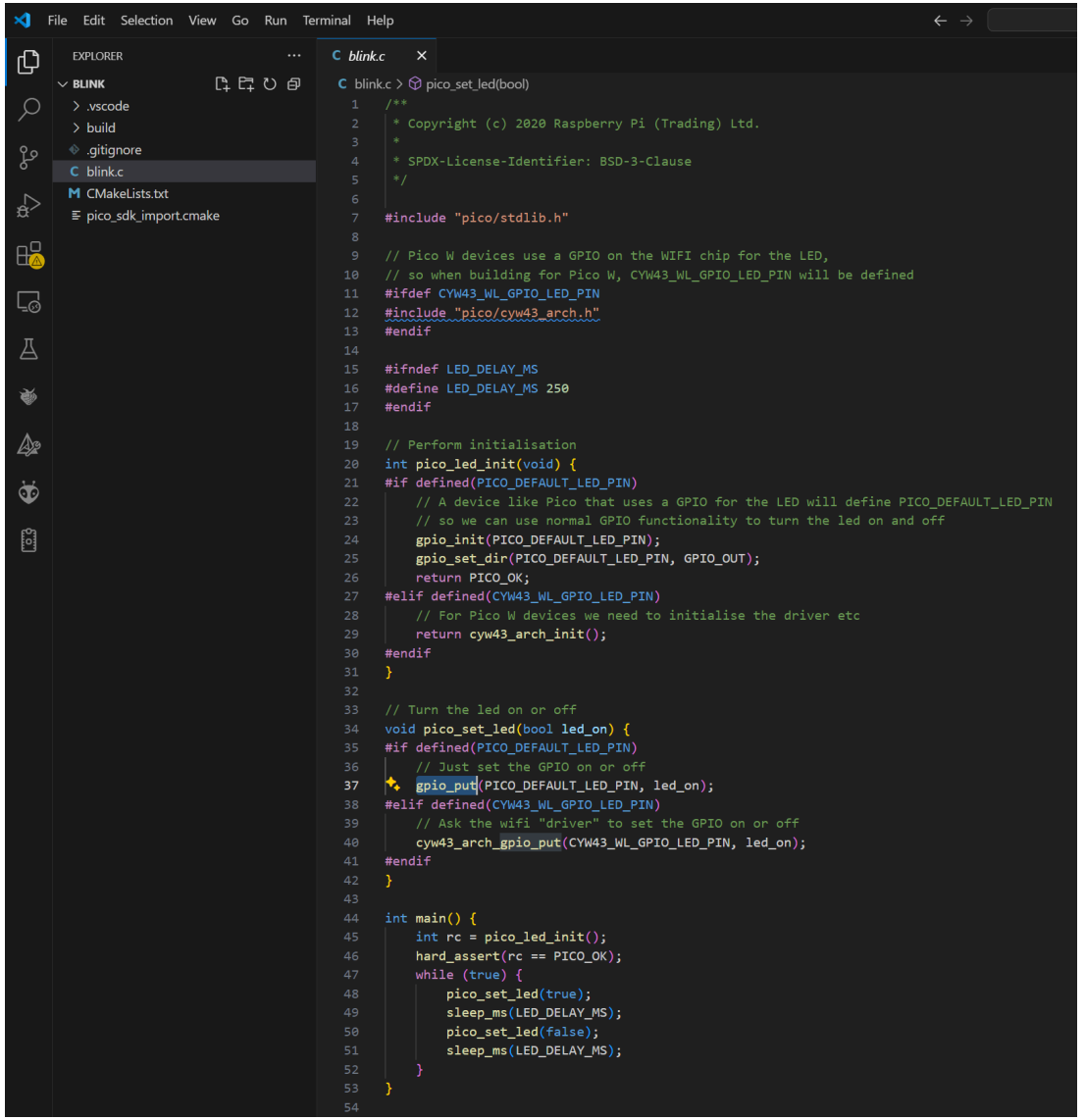
- The Raspberry Pi Pico extension allows you to code in 2 different languages
  1. C/C++ → This uses the Pico SDK as framework. Note that this is NOT Arduino framework! Arduino libraries will not work!
  2. Micropython → this is implemented using the MicroPico extension https://github.com/paulober/MicroPico

Howto open the Pico Extension?



C/C++ project example: Blink





C/C++ project example: Blink

```c
#include "pico/stdlib.h"

// Pico W devices use a GPIO on the WIFI chip for the LED,
// so when building for Pico W, CYW43_WL_GPIO_LED_PIN will be defined
#ifdef CYW43_WL_GPIO_LED_PIN
#include "pico/cyw43_arch.h"
#endif

#ifndef LED_DELAY_MS
#define LED_DELAY_MS 250
#endif

// Perform initialisation
int pico_led_init(void) {
#if defined(PICO_DEFAULT_LED_PIN)
    // A device like Pico that uses a GPIO for the LED will define PICO_DEFAULT_LED_PIN
    // so we can use normal GPIO functionality to turn the led on and off
    gpio_init(PICO_DEFAULT_LED_PIN);
    gpio_set_dir(PICO_DEFAULT_LED_PIN, GPIO_OUT);
    return PICO_OK;
#elif defined(CYW43_WL_GPIO_LED_PIN)
    // For Pico W devices we need to initialise the driver etc
    return cyw43_arch_init();
#endif
}
```

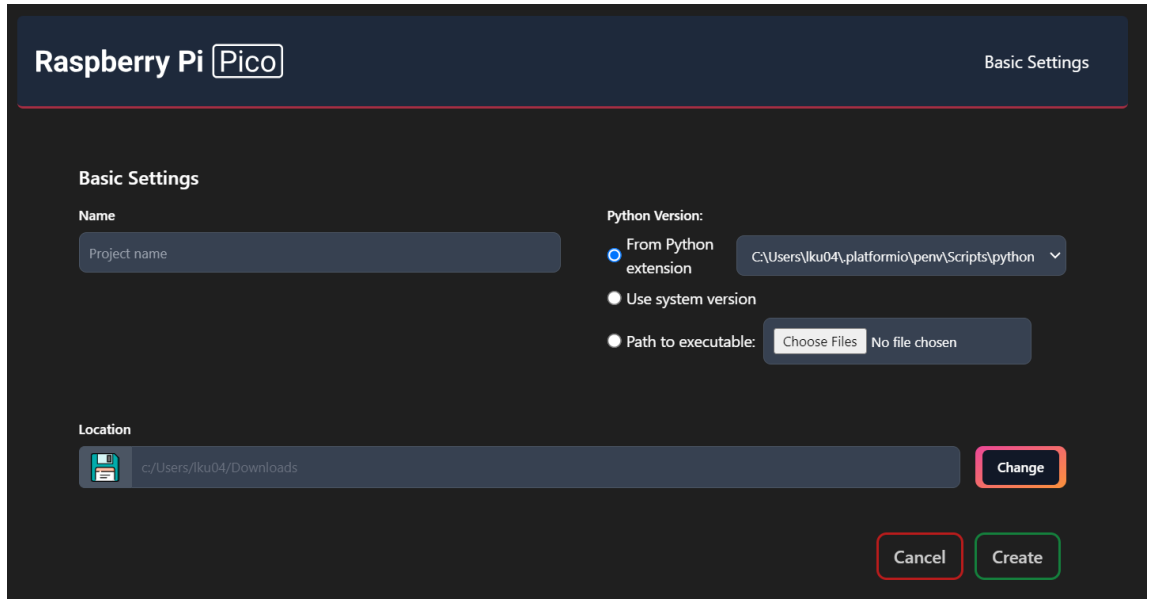
```c
// Turn the led on or off
void pico_set_led(bool led_on) {
#if defined(PICO_DEFAULT_LED_PIN)
    // Just set the GPIO on or off
    gpio_put(PICO_DEFAULT_LED_PIN, led_on);
#elif defined(CYW43_WL_GPIO_LED_PIN)
    // Ask the wifi "driver" to set the GPIO on or off
    cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, led_on);
#endif
}

int main() {
    int rc = pico_led_init();
    hard_assert(rc == PICO_OK);
    while (true) {
        pico_set_led(true);
        sleep_ms(LED_DELAY_MS);
        pico_set_led(false);
        sleep_ms(LED_DELAY_MS);
    }
}
```

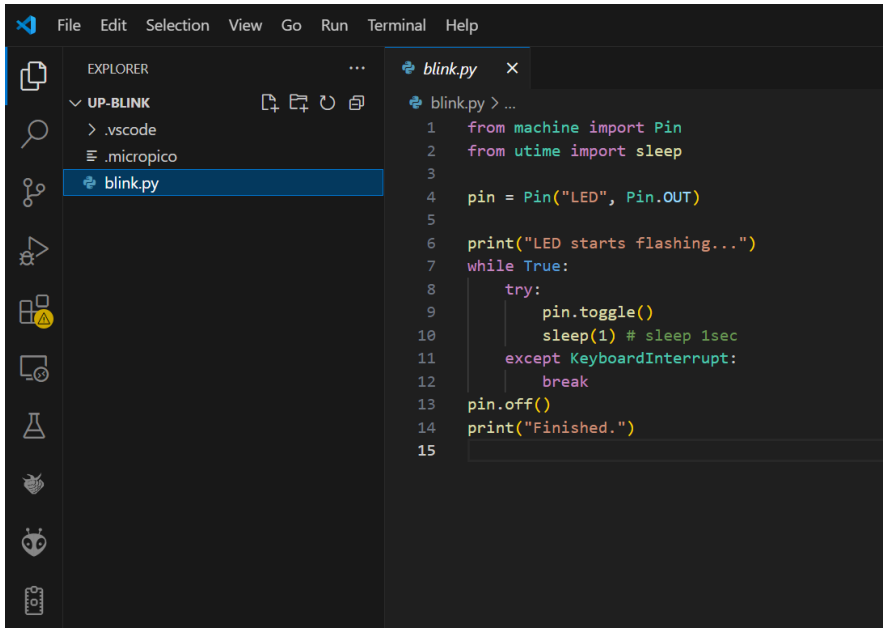
And then you try to compile the example…

.pico-sdk/toolchain/13_3_Rel1/bin/../lib/gcc/arm-none-eabi/13.3.1/../../../../arm-none-eabi/bin/ld.exe: cannot find -lpico_cyw43_arch_none: No such file or directory
collect2.exe: error: ld returned 1 exit status
ninja: build stopped: subcommand failed.

- Nevermind. Let’s continue to micropython ☺

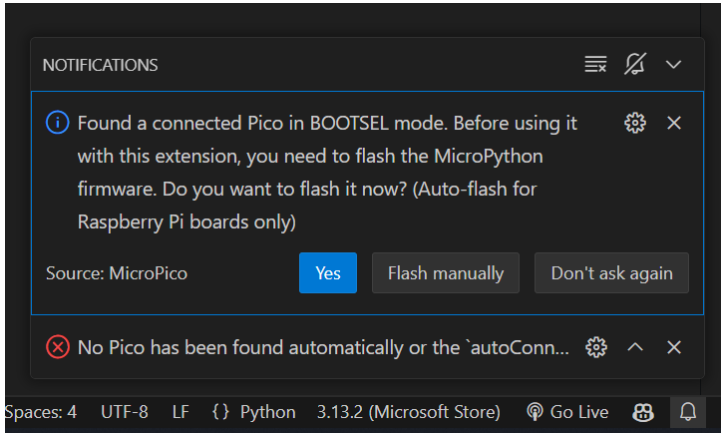
Micropython example

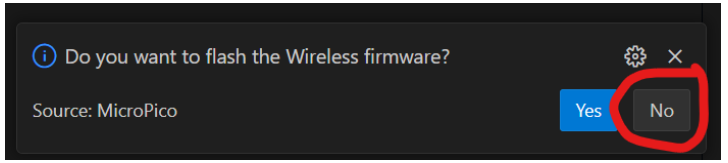


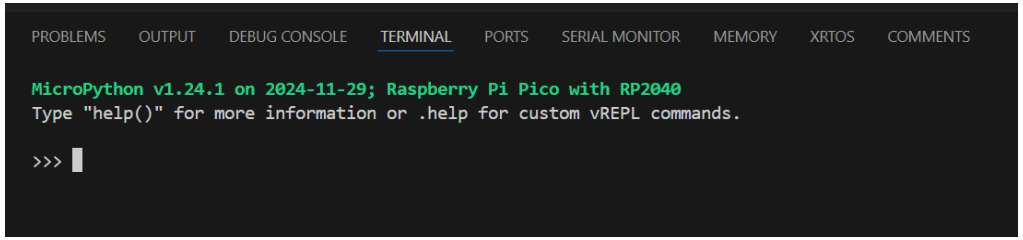


Micropython on Xiao RP2040

- Connect the Xiao RP2040
- Put it in bootsel mode
  - hold button B, press & release button R, release button B
- Notice the notification in VSCode
  - Click yes to flash micropython firmware
  - Do NOT flash Wireless firmware
    - Xiao RP2040 is not a Pico-W
    - Press no will flash non-wireless firmware
- You should now see the REPL in the terminal





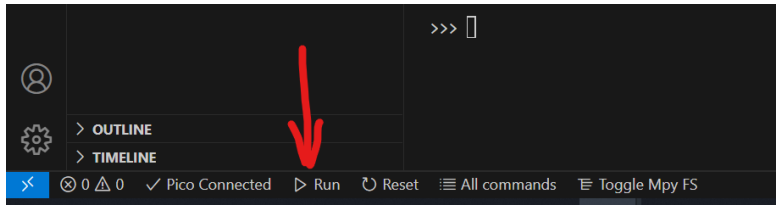


Micropython on Xiao RP2040

```
from machine import Pin
from utime import sleep

pinG = Pin(16, Pin.OUT)
pinR = Pin(17, Pin.OUT)
pinB = Pin(25, Pin.OUT)

print("LED starts flashing...")
pinR.on()
pinG.on()
pinB.on()
while True:
    try:
        pinG.toggle()
        sleep(1) # sleep 1sec
    except KeyboardInterrupt:
        break
pinG.on()
print("Finished.")
```

```
# XIAO RP2040 pins:

# pin 4 (D4) = I2C SDA
# pin 5 (D5) = I2C SCL
# pin 6 (D6) = UART TX
# pin 7 (D7) = UART RX / SPI Chip select
# pin 8 (D8) = SPI SCK
# pin 9 (D9) = SPI MISO
# pin 10 (D10) = SPI MOSI
# pin 11 = enable pin of RGB LED
#          (high = enabled)
# pin 12 = WS2812 data pin
# pin 16 = onboard green LED (use PIN_LED_G)
# pin 17 = onboard red LED (use PIN_LED_R)
# pin 25 = onboard blue LED (use PIN_LED_B)

# for ONBOARD LEDs: HIGH = OFF, LOW = ON
# so pin.on() = off ☺
```

- Click Run to run



Further reading

- https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf