Microcontroller

Introduction HackLab

`https://github.com/KABK-HackLab/microcontroller-intro`

What is a MicroController

MCU (MicroController Unit)

A small self-contained computer on a single chip. The microcontroller is the core of an embedded system.

Embedded systems are specialized, computer-based systems combining hardware and software with the purpose of interacting with the physical world using IO Ports

The Microcontroller contains:
- Processor (that executes our code)
- Flash Memory (for program storage)
- Ram memory (for processing, variables)
- IO (Input / Output Ports) via ‘pins’

There are various brands and development environments.

While a MCU is only a single chip, most of them come on a microcontroller-board that makes it easier to work with. For example it features a usb connector to upload code.

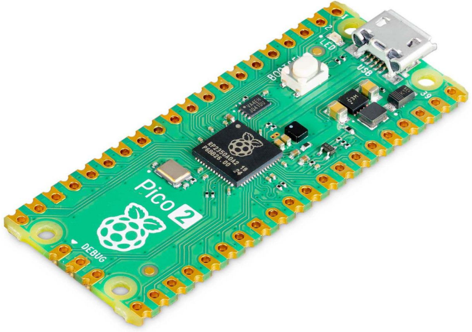
The microcontroller-board is often just called microcontroller.

What microcontroller is best depends on your project and personal preference.

Each one has there one advantages and disadvantages.

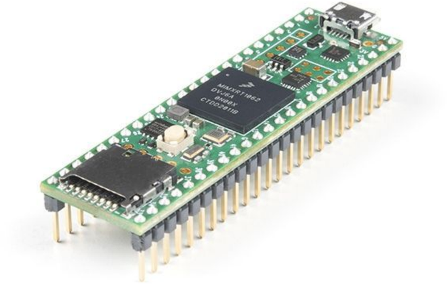
Arduino Well known, large community, open source software library. Various boards ranging in price and performance. Arduino is both a microcontroller board and a software development environment (IDE)

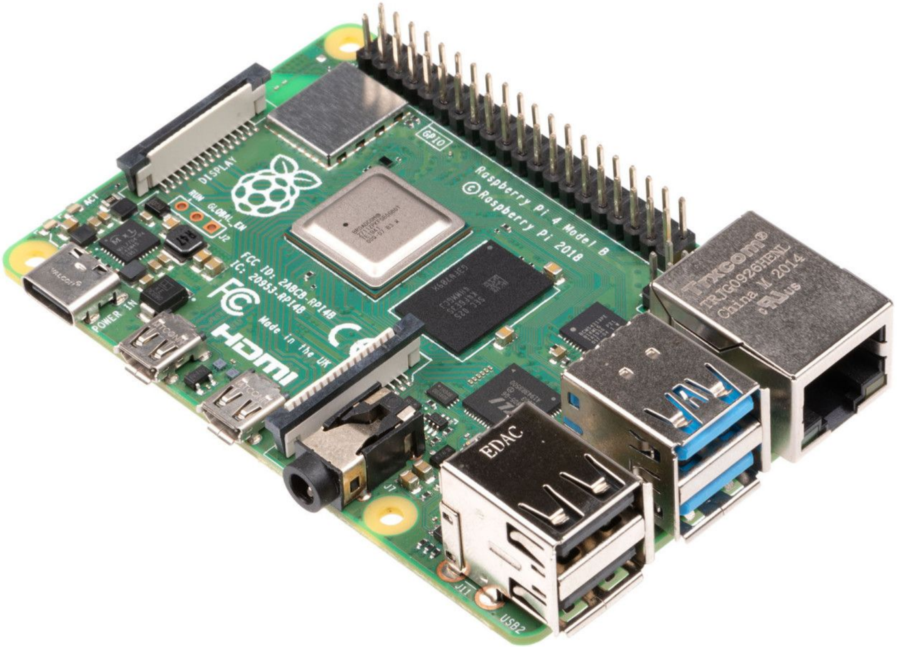
Raspberry Pico Cheap, Not the fastest board, buthas interesting set advanced features (e.g. dual core). Not to be confused with the Raspberry Pi.



ESP32 Includes build in Wifi and Bluetooth. Compatible with arduino IDE.

Teensy Powerful platform that extends on the arduino platform with advanced features, performance and software libraries. Expensive





Raspberry Pi

The Raspberry Pi is not a microcontroller!

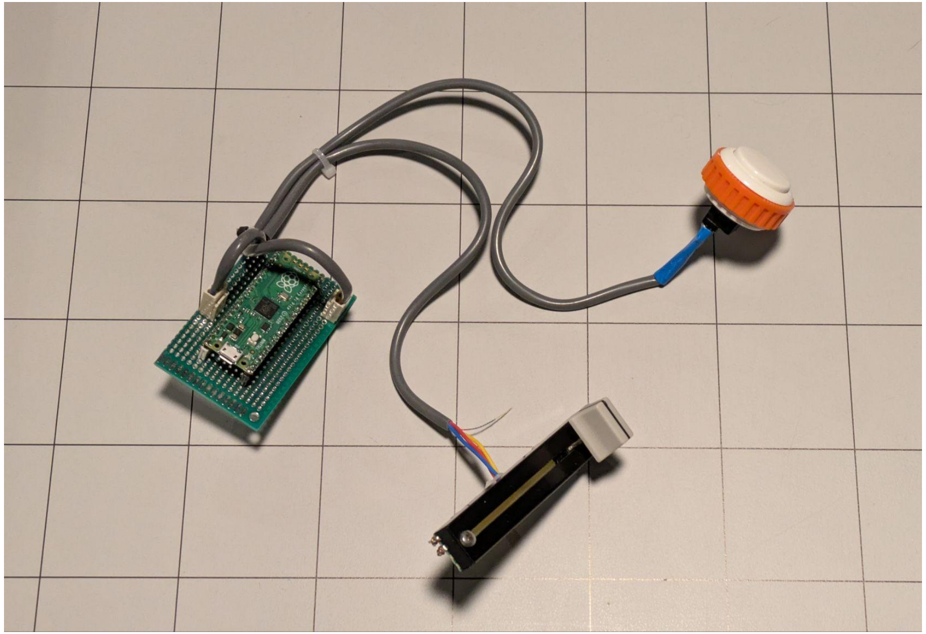
It is called a single-board computer and is actually a complete mini computer. In contrast to a microcontroller this can run a complete linux desktop system with graphics, audio, network, storage.

Similar to a microcontroller it features an IO port that can be used to communicate with the other hardware.

A microcontroller does not have an OS.

It only runs the code we put upload to the chip ourselves.

DEMO TIME



Create a basic music instrument

Microcontroller functions as the interface between

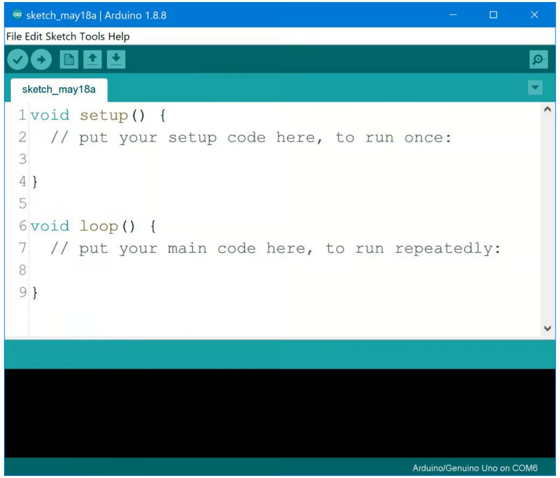
software: audio synthesis software, running laptop

and

hardware: buttons / faders

There are various programming development environments.

Similar to the microcontroller boards, each one has there one advantages and disadvantages.

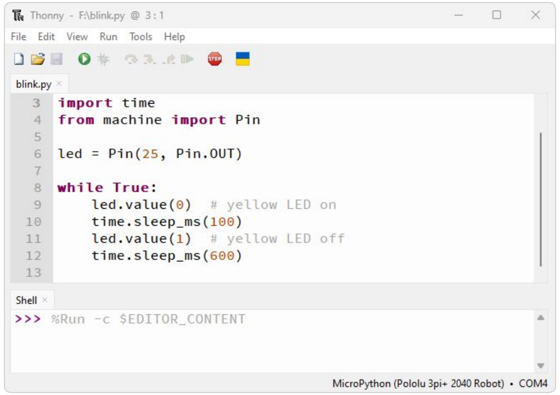

sketch_may18a | Arduino 1.8.8
File Edit Sketch Tools Help
sketch_may18a
Arduino/Genuino Uno on COM6

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Arduino IDE

For use with arduino and compatible boards. Used the the c/c++ programming language with specialised software libraries for writing code for microcontrollers.

The arduino ecosystem features an extensive range of libraries to connect a wide range of hardware. Sensors, displays, shields (extension boards), etc


Thonny - F:\blink.py @ 3 : 1
File Edit View Run Tools Help
blink.py
Shell
MicroPython (Pololu 3pi+ 2040 Robot) • COM4

```
import time
from machine import Pin

led = Pin(25, Pin.OUT)

while True:
    led.value(0)   # yellow LED on
    time.sleep_ms(100)
    led.value(1)   # yellow LED off
    time.sleep_ms(600)

```

```
>>> %Run -c $EDITOR_CONTENT
```

Thonny

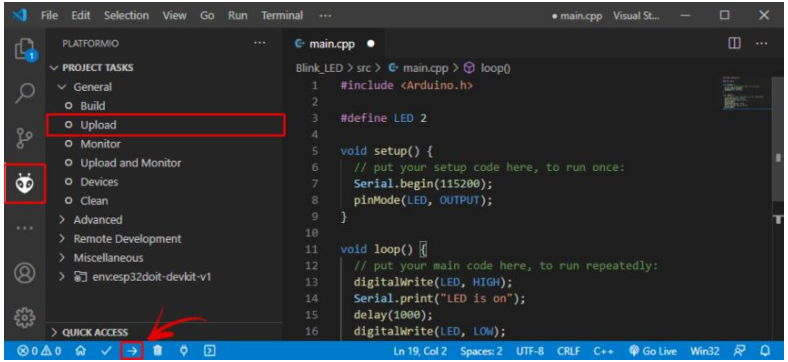
Focussed on the python programming language.

Can be used to write python code for PC's as well as microcontrollers.

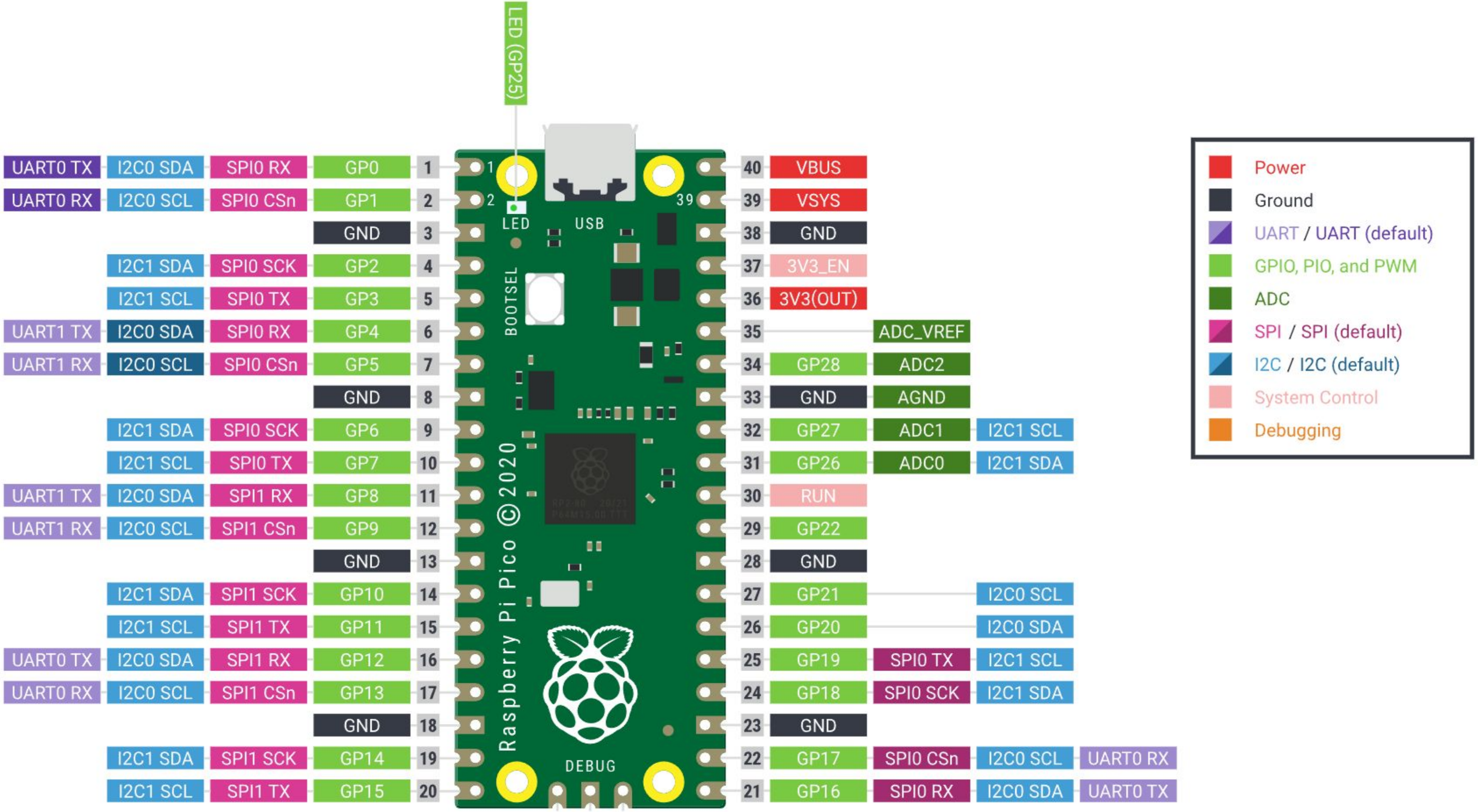
Python is arguably easier for beginners, but less performant than the C/C++ language

VSCODE + Platform IO

Similar to Arduino IDE in that it uses C/C++ based language for microcontrollers. It has many advanced Features such as autocomplete, realtime debugging (stepping through code) and AI integration. Somewhat intimidating for beginners.

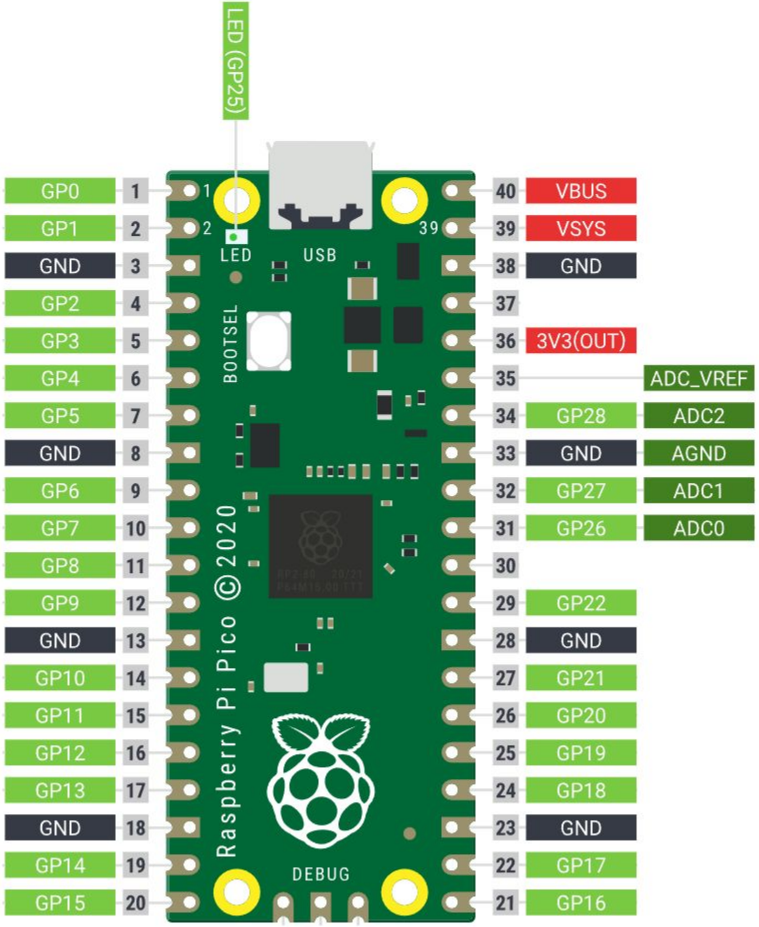


```
#include <Arduino.h>

#define LED 2

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED, HIGH);
  Serial.print("LED is on");
  delay(1000);
  digitalWrite(LED, LOW);
```

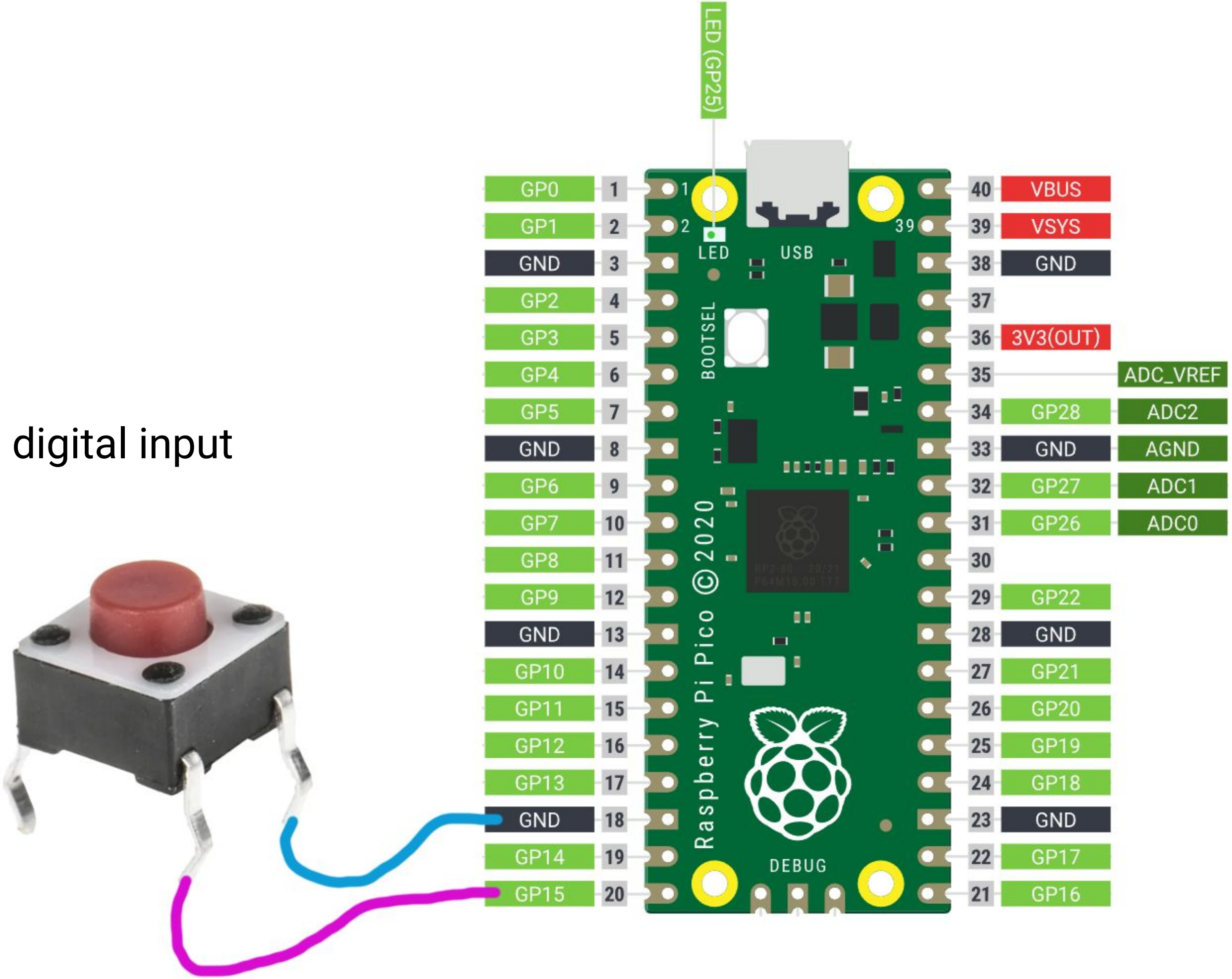


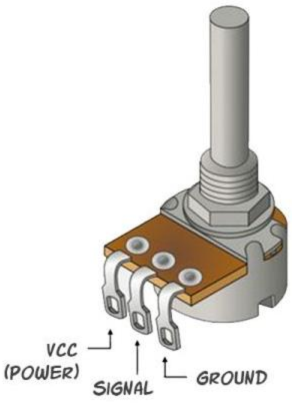
UART0 TX	I2C0 SDA	SPI0 RX	GP0	1	40	VBUS			
UART0 RX	I2C0 SCL	SPI0 CSn	GP1	2	39	VSYS			
			GND	3	38	GND			
	I2C1 SDA	SPI0 SCK	GP2	4	37	3V3_EN			
	I2C1 SCL	SPI0 TX	GP3	5	36	3V3(OUT)			
UART1 TX	I2C0 SDA	SPI0 RX	GP4	6	35		ADC_VREF		
UART1 RX	I2C0 SCL	SPI0 CSn	GP5	7	34	GP28	ADC2		
			GND	8	33	GND	AGND		
	I2C1 SDA	SPI0 SCK	GP6	9	32	GP27	ADC1	I2C1 SCL	
	I2C1 SCL	SPI0 TX	GP7	10	31	GP26	ADC0	I2C1 SDA	
UART1 TX	I2C0 SDA	SPI1 RX	GP8	11	30	RUN			
UART1 RX	I2C0 SCL	SPI1 CSn	GP9	12	29	GP22			
			GND	13	28	GND			
	I2C1 SDA	SPI1 SCK	GP10	14	27	GP21		I2C0 SCL	
	I2C1 SCL	SPI1 TX	GP11	15	26	GP20		I2C0 SDA	
UART0 TX	I2C0 SDA	SPI1 RX	GP12	16	25	GP19	SPI0 TX	I2C1 SCL	
UART0 RX	I2C0 SCL	SPI1 CSn	GP13	17	24	GP18	SPI0 SCK	I2C1 SDA	
			GND	18	23	GND			
	I2C1 SDA	SPI1 SCK	GP14	19	22	GP17	SPI0 CSn	I2C0 SCL	UART0 RX
	I2C1 SCL	SPI1 TX	GP15	20	21	GP16	SPI0 RX	I2C0 SDA	UART0 TX

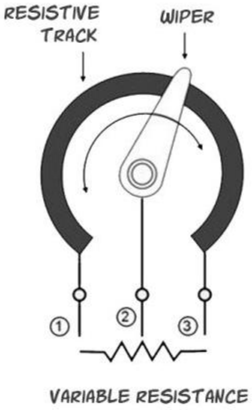
- 🟥 Power
- ⬛ Ground
- ◪ UART / UART (default)
- 🟩 GPIO, PIO, and PWM
- 🟩 ADC
- ◪ SPI / SPI (default)
- ◪ I2C / I2C (default)
- ■ System Control
- 🟧 Debugging


LED (GP25)
GP0
1
1
40
VBUS
GP1
2
2
LED
USB
39
39
VSYS
GND
3
38
GND
GP2
4
37
GP3
5
BOOTSEL
36
3V3(OUT)
GP4
6
35
ADC_VREF
GP5
7
34
GP28
ADC2
GND
8
33
GND
AGND
GP6
9
32
GP27
ADC1
GP7
10
31
GP26
ADC0
GP8
11
30
GP9
12
29
GP22
GND
13
28
GND
GP10
14
27
GP21
GP11
15
26
GP20
GP12
16
25
GP19
GP13
17
24
GP18
GND
18
23
GND
GP14
19
22
GP17
GP15
20
21
GP16
Raspberry Pi Pico © 2020
DEBUG

digital input







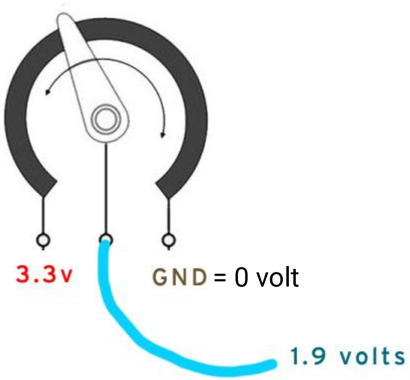
Potmeter overview

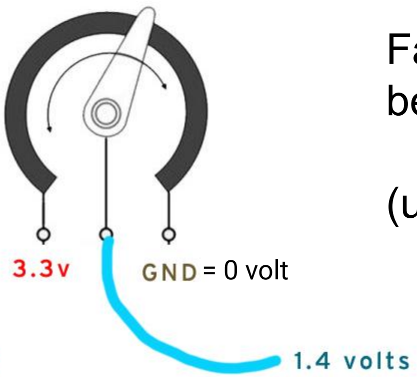
The Fader works exactly the same, different style / housing.

Potmeter: output is the middle pin (pin 2)

Fader: output position can vary between designs.

(usually labeled: pin 2)

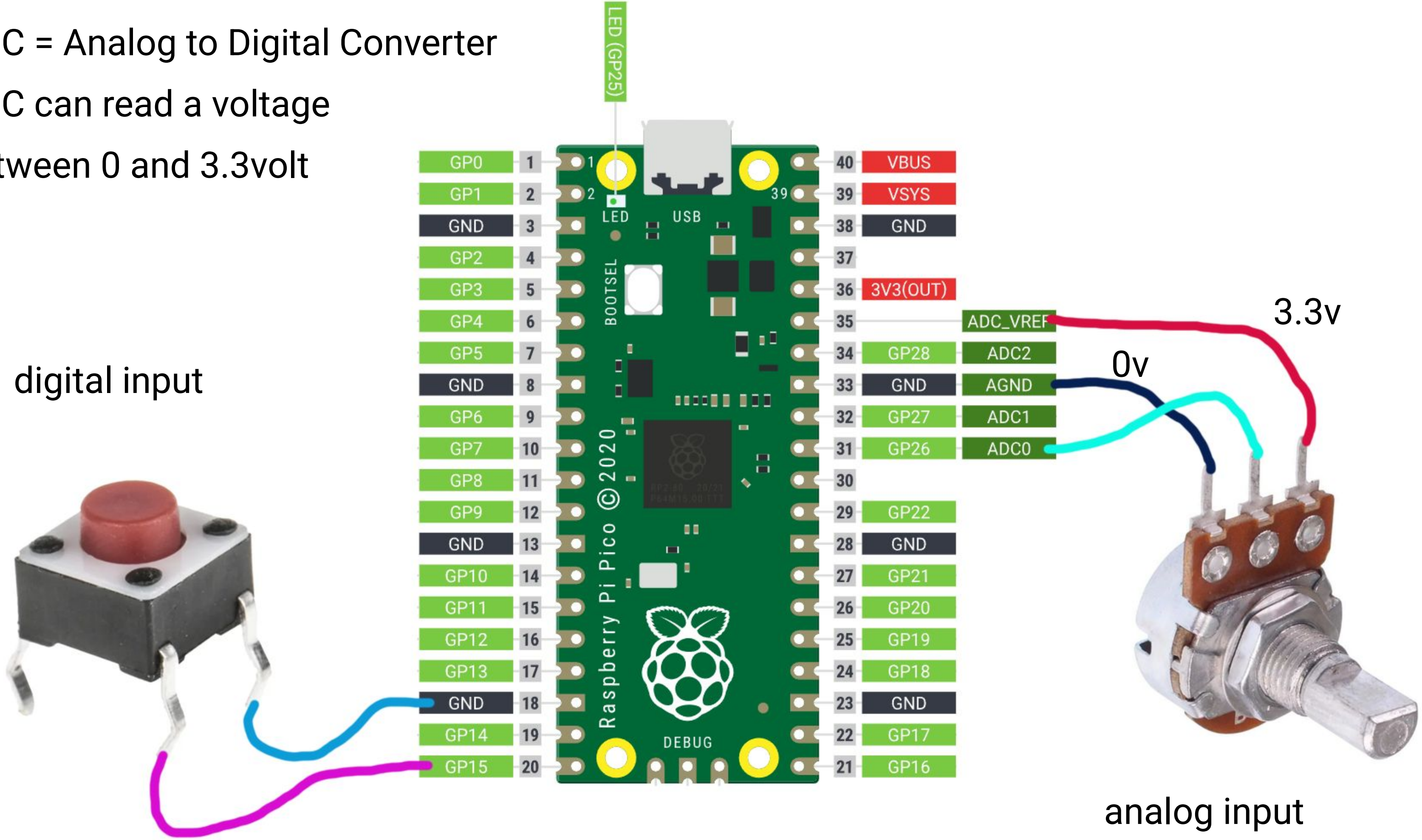




ADC = Analog to Digital Converter

ADC can read a voltage

between 0 and 3.3volt



[Coding Time]

- Download or clone the Hacklab microcontroller-intro repository

https://github.com/KABK-HackLab/microcontroller-intro

- Dwnload circuitpython for the raspberry pico:

https://circuitpython.org/downloads

- Install/upload circuitpython to the board:

1) Hold the bootsel button and connect the usb cable, the pico should show up as a drive.

2) Copy the utf file to the pico, The pico will restart

https://learn.adafruit.com/getting-started-with-raspberry-pi-pico-circuitpython/circuitpython

- Install Thonny:

https://thonny.org/

In Thonny go to Tools - Options - Interpreter and select circuitpython.