



Hardware Hacking Evening

Hypatia Network



Agenda

Why?

Circuits

Raspberry Pi

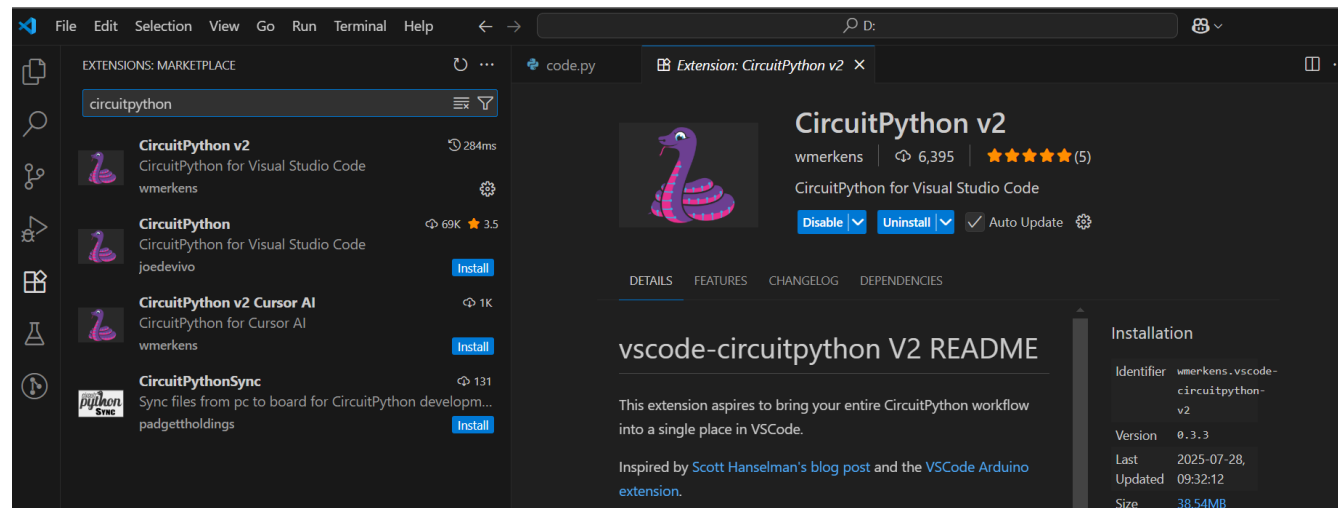
Coding

Human Interface Devices

Free time

While I Talk

- Grab a **Pi Pico**, a **breadboard**, and 3-4 **jumper wires**
- Download and install **Visual Studio Code**
- Install the **CircuitPython v2** extension








9/3/20XX



What?



Build your own Thing which
you can plug into your
computer and it acts as a
**keyboard, mouse or USB
gamepad**




Why?



If you can interface hardware
and software **you can do
anything**



So?



Hardware is fun and easier
than it looks!

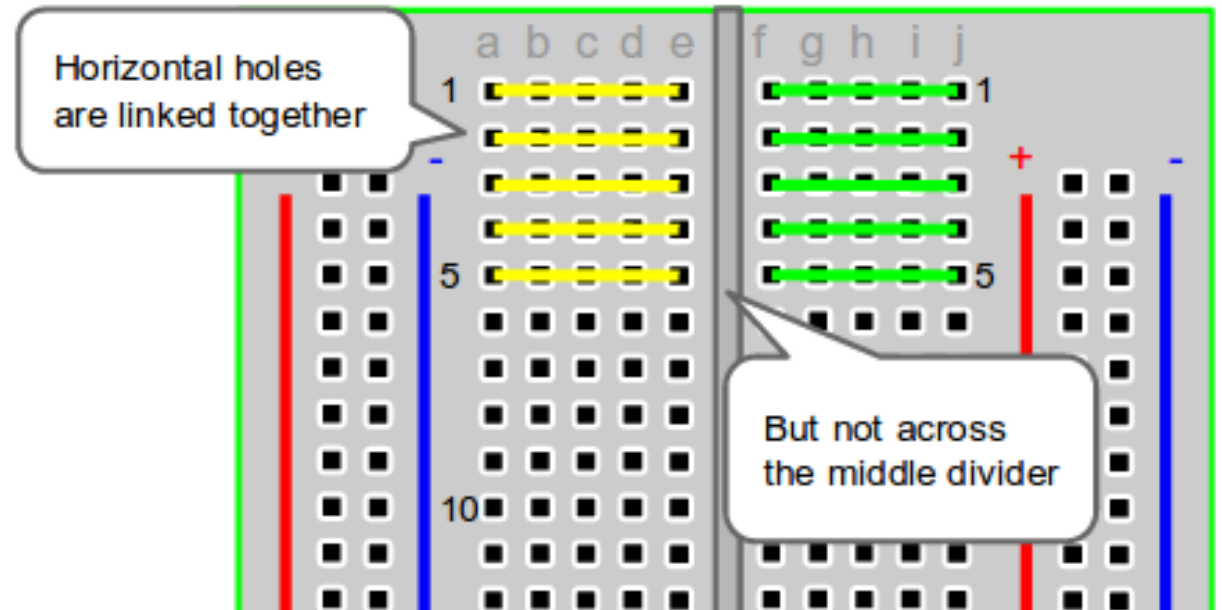
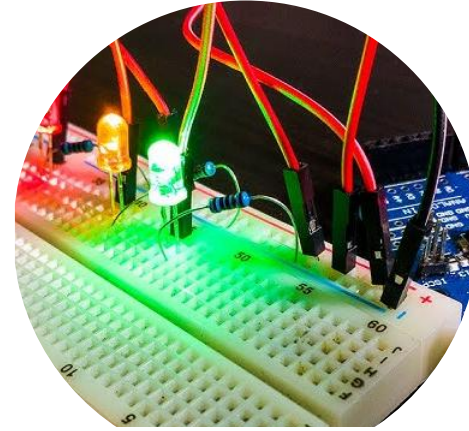
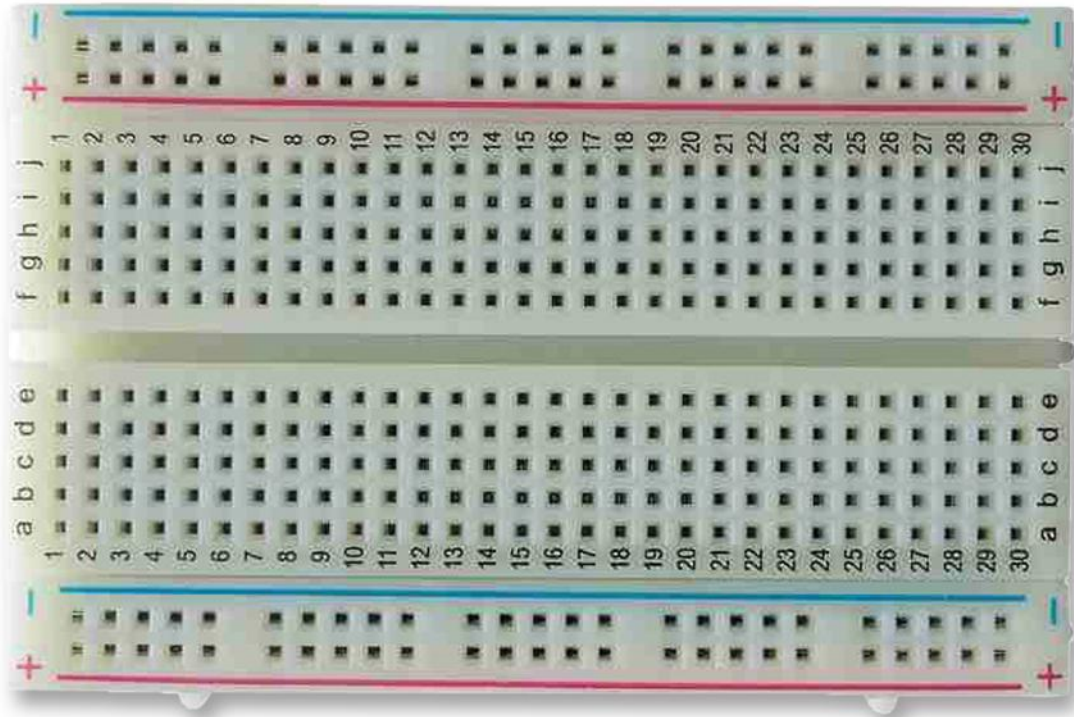


Circuits

Start with electronics

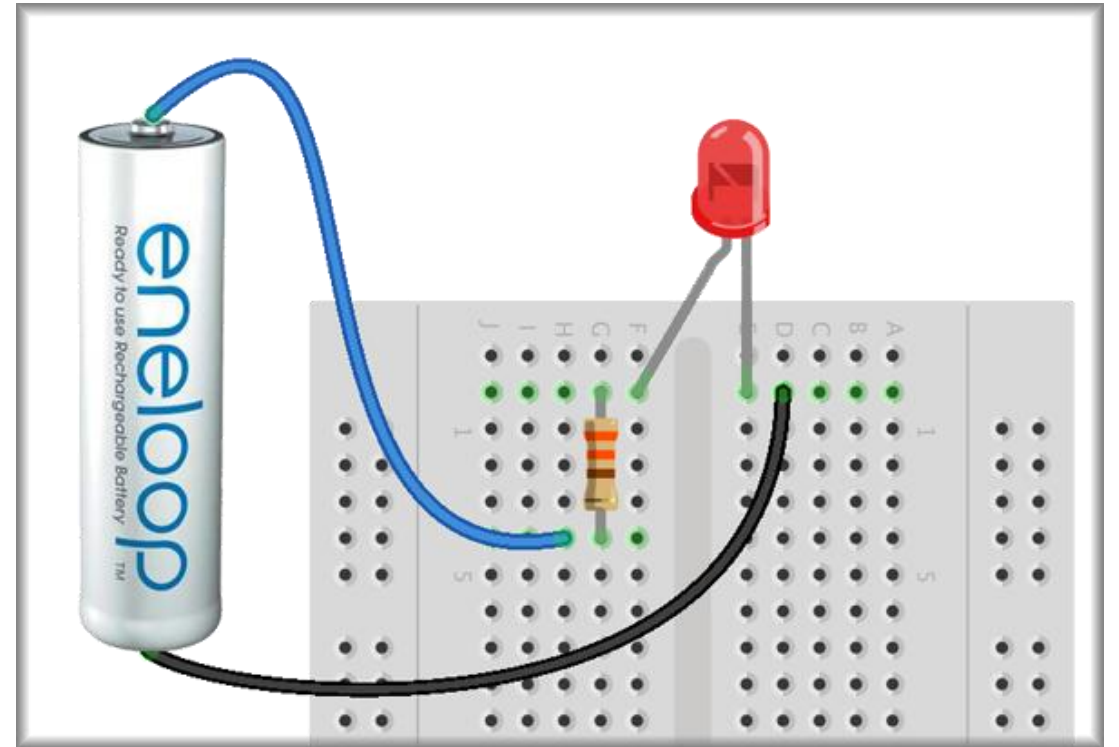
Breadboard

Plug stuff into it to prototype your Thing



Electronic Circuits 101

- Electricity flows through circuits
- LEDs have a right and a wrong way around
- Use black to mean ground
- Never plug in an LED without a resistor too



Wires and Connectors

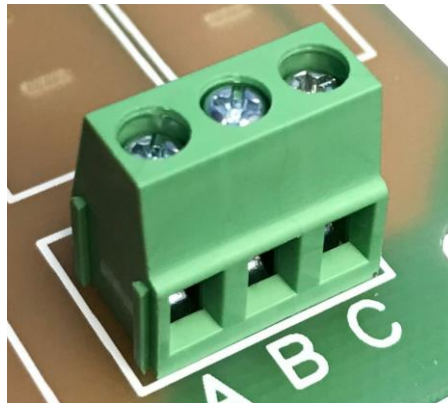
Dupont wire

JST

Screw terminals

Arcade (spade) connectors

Wagos



The logo features a large blue circle with the text "Raspberry Pi Pico" in white. To the left of the circle is a dashed teal line, and at the bottom right is a small purple circle.

Raspberry Pi Pico

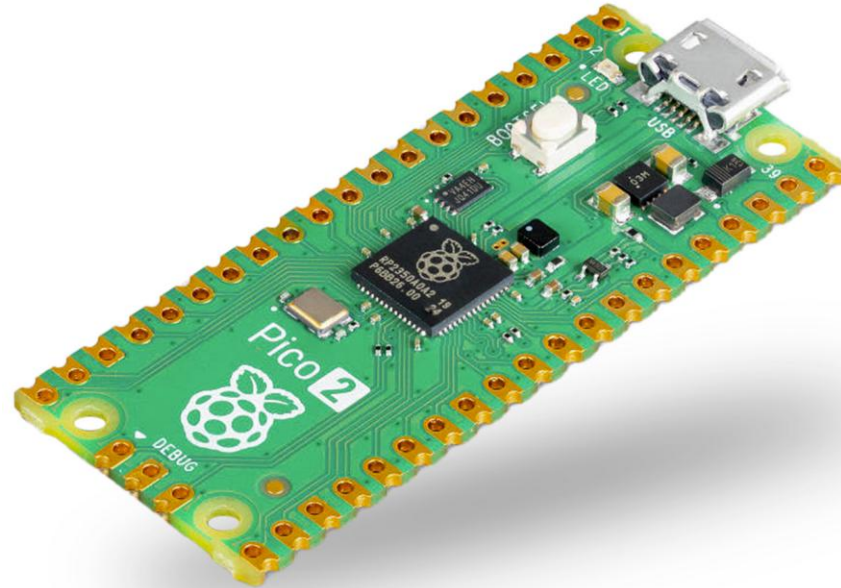
Raspberry Pi 5

- A proper computer
- Can run basically anything
- Install an OS via an SD card
- £48



Raspberry Pi Pico

- Microcontroller
- Speaks Python or C
- <£5
- May come with headers



A note on USB Micro cables

- They are not all alike
- Get one that does data
- Do not spend hours debugging a “dead pi pico” that turns out to be a rubbish cable
- Do not be like Sarah



Let's Go!

- Plug your Pico in to your computer
- It should pop up like a USB stick
- Like a USB stick, you should eject before disconnecting

UF2s

- C vs Python
- Micropython vs Circuitpython
- <https://circuitpython.org/downloads>
- Download for Pico (just pico)
- Copy to the drive

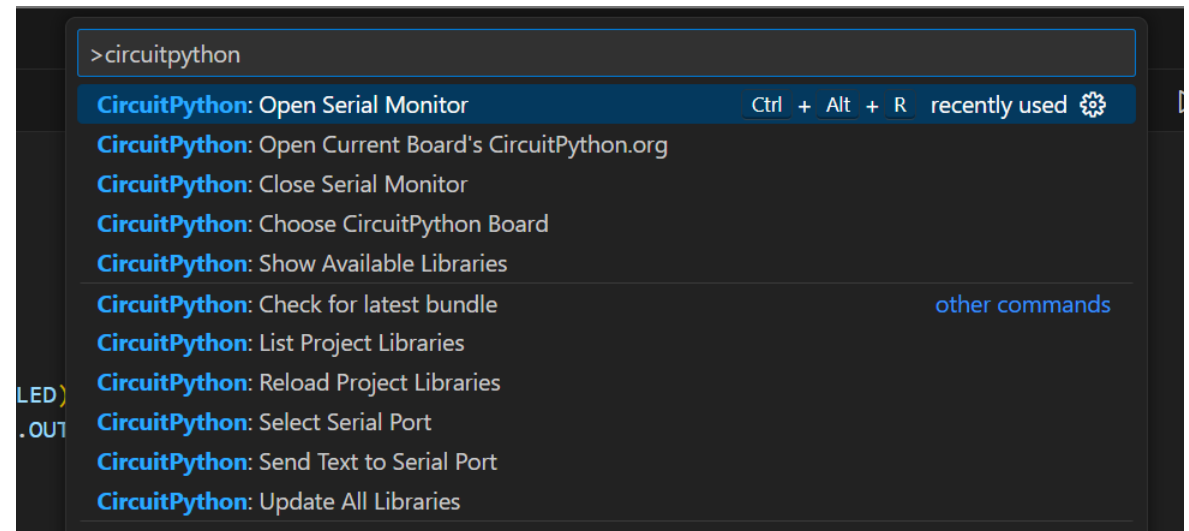


Code Editing with VS Code

- Open Visual Studio Code
- Install the CircuitPython V2 extension
- Open Folder -> Select the Pico drive -> Open

Code Editing with VS Code

- CircuitPython: Choose CircuitPython Board -> find Raspberry Pi:Pico
- CircuitPython: Select Serial Port -> select whatever's there (mine said COM8)
- CircuitPython: Open Serial Monitor





GPIOs

The bridge between hardware and
software

Blinky

```
import board
import digitalio
from time import sleep

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

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

Python programs on microcontrollers:

- Loop based
- Run the main loop lots of times as fast as you can
- Use digitalio to interface with the real world

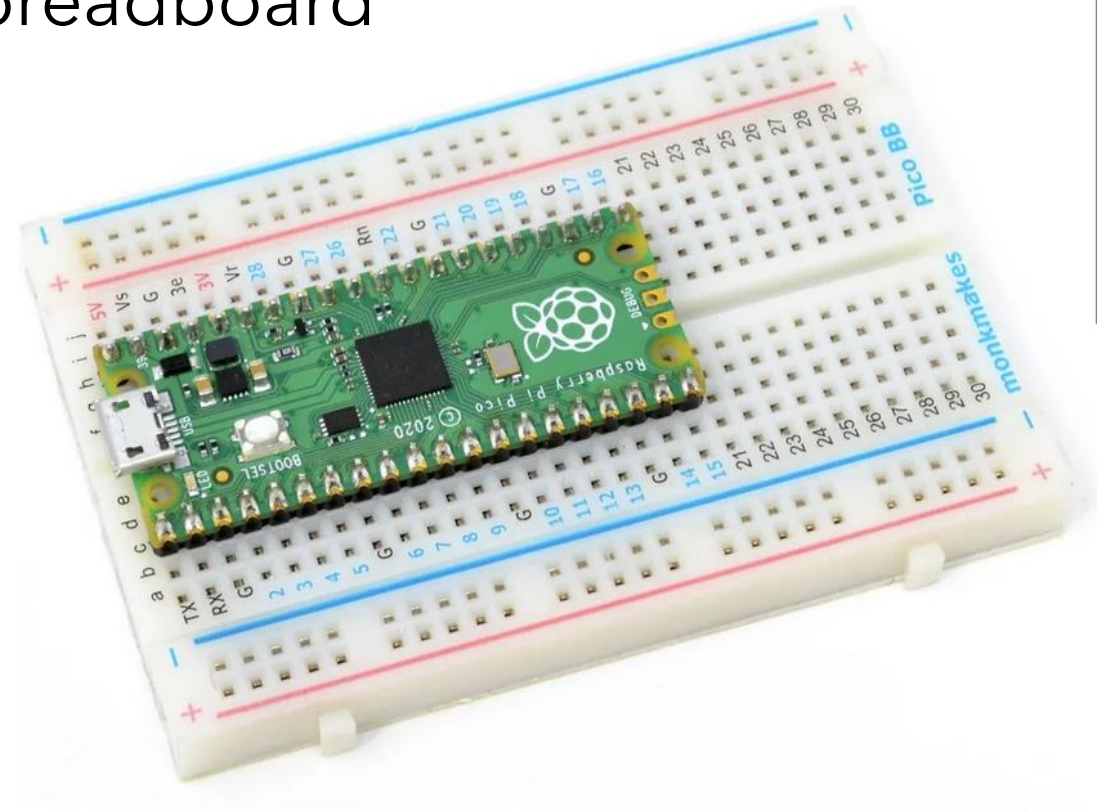
Breadboard time

A word of warning

- **Do not short circuit the power pins to ground**
- If you do this enough times you will damage your Pico
- If Mu suddenly beeps and your pico disconnects, it's probably because it's restarted, which is probably because something has short circuited

Breadboard time

- Stick your Pi carefully in your breadboard
- USB facing outwards



Getting input into pico

- GPIO = General Purpose Input/Output
- Pins can be output (e.g. LED) or input (e.g. button)

☐ Advanced ☐ Rear View ☐ Upside-down

☒ SPI ☒ I2C ☒ UART

UART0 TX	I2C0 SDA	SPI0 RX	GP0	1	40	VBUS 5V
UART0 RX	I2C0 SCL	SPI0 CSn	GP1	2	39	VSYS 5V*
			Ground	3	38	Ground
	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 A2
			Ground	8	33	ADC Gnd
	I2C1 SDA	SPI0 SCK	GP6	9	32	GP27 A1
	I2C1 SCL	SPI0 TX	GP7	10	31	GP26 A0
UART1 TX	I2C0 SDA	SPI1 RX	GP8	11	30	RUN
UART1 RX	I2C0 SCL	SPI1 CSn	GP9	12	29	GP22
			Ground	13	28	Ground
	I2C1 SDA	SPI1 SCK	GP10	14	27	GP21
	I2C1 SCL	SPI1 TX	GP11	15	26	GP20
UART0 TX	I2C0 SDA	SPI1 RX	GP12	16	25	GP19
UART0 RX	I2C0 SCL	SPI1 CSn	GP13	17	24	GP18
			Ground	18	23	Ground
	I2C1 SDA	SPI1 SCK	GP14	19	22	GP17
	I2C1 SCL	SPI1 TX	GP15	20	21	GP16
						SPI0 CSn
						SPI0 RX
						SPI0 TX
						SPI0 SCK
						I2C0 SCL
						I2C0 SDA
						I2C1 SCL
						I2C1 SDA
						UART1 RX
						UART1 TX
						UART0 RX
						UART0 TX

<https://pico.pinout.xyz/>

Pressy Buttony

```
import time
import board
from digitalio import DigitalInOut, Direction, Pull

led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT

switch = DigitalInOut(board.GP15)
switch.direction = Direction.INPUT
switch.pull = Pull.UP

while True:
    if switch.value:
        led.value = False
    else:
        led.value = True

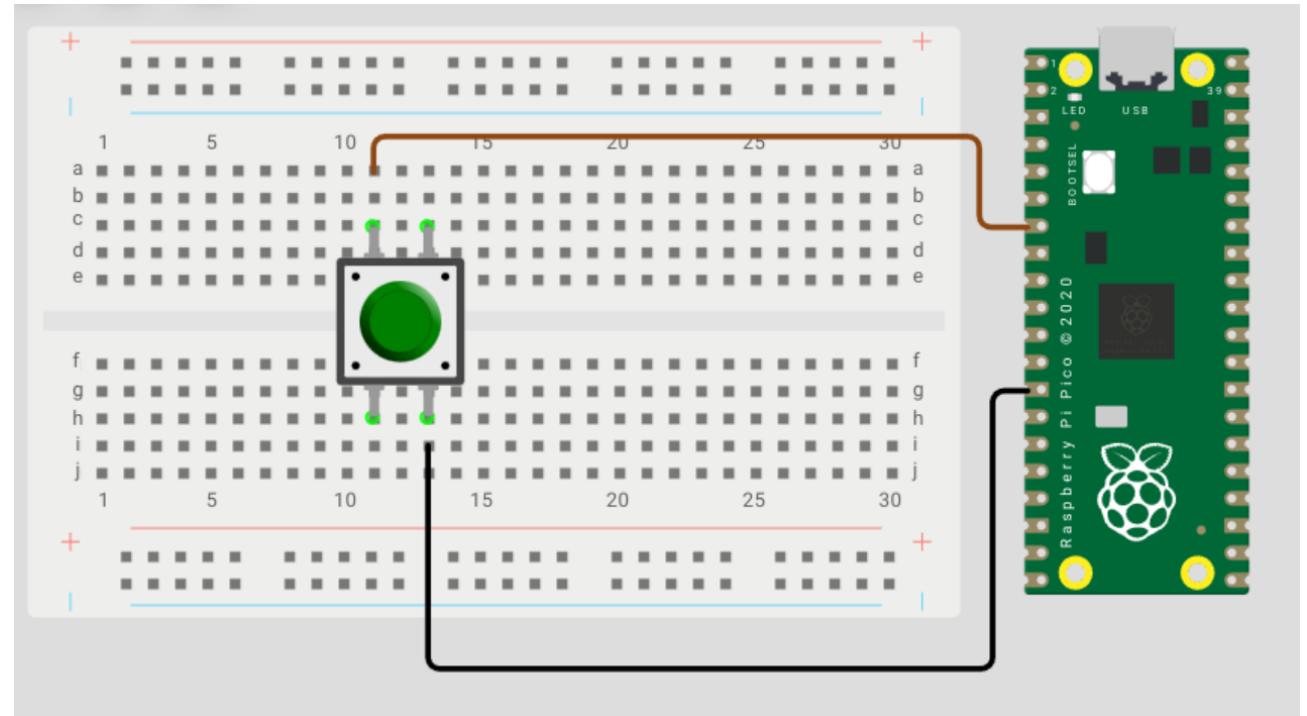
    time.sleep(0.01)
```

Applies a Pull Up resistor
between this pin and 3V3

Grab a jumper wire and touch
it between GP15 and GND

Wiring up a proper button

- Button = disconnected when not pressed, connected when pressed
- Use jumper cables to connect the button up to GP15 and GND



Now let's try detecting changes

```
from time import sleep
import board
from digitalio import DigitalInOut, Direction, Pull

switch = DigitalInOut(board.GP15)
switch.direction = Direction.INPUT
switch.pull = Pull.UP

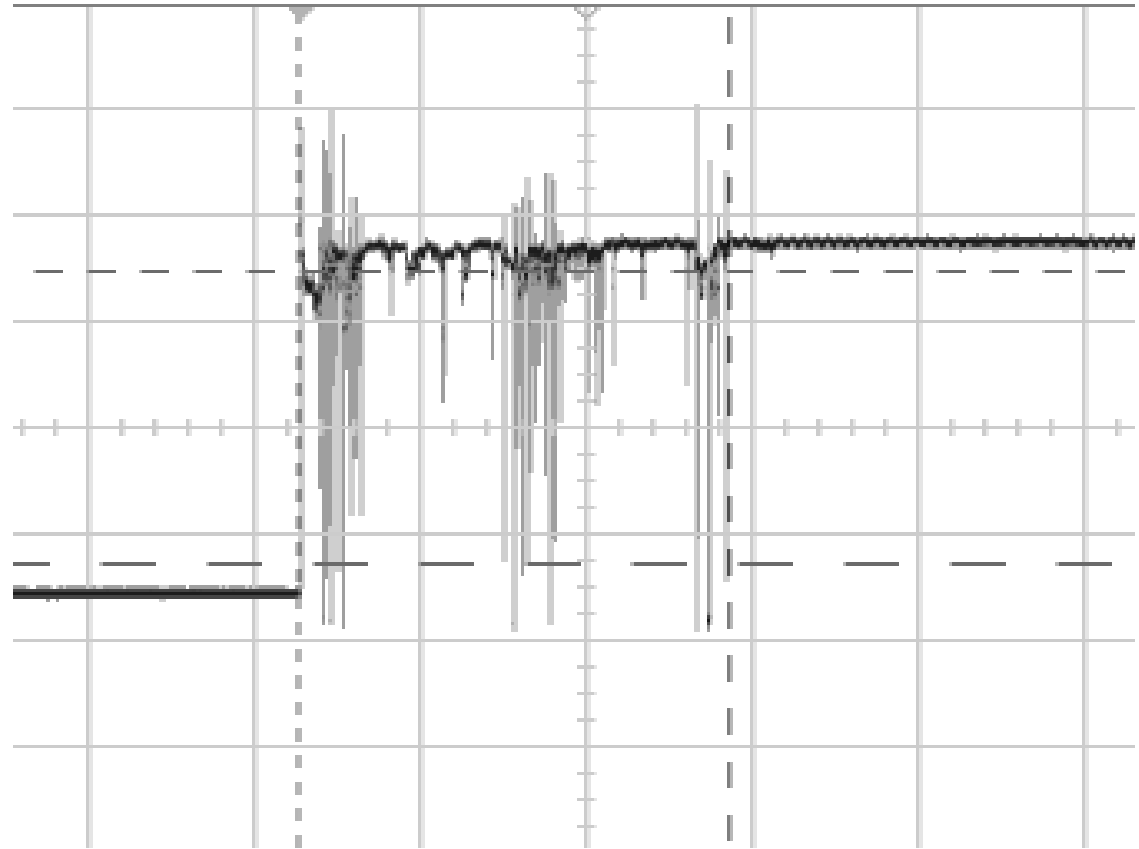
last_switch_state = switch.value

while True:
    if switch.value != last_switch_state:
        last_switch_state = switch.value
        if switch.value:
            print("Switch is ON")
        else:
            print("Switch is OFF")

    sleep(0.01)
```

Debouncing

- Physical inputs are noisy and analogue
- We need to turn them into nice clean digital reads
- Basic approach is to read multiple times and smooth it out



Debouncing

- Time to install a dependency
- Adafruit has a big ol' pile of these in the CircuitPython Bundle
- Command Palette -> CircuitPython: Show available libraries -> find the one you want and press Enter
- It should appear in the `lib/` folder on the Pico
- You can also download a big ol' zip of all the libraries at <https://circuitpython.org/libraries> and manage them manually on the Pico
- Also do this with 3rd party code

Debouncing

- Command Palette -> CircuitPython: Show available libraries
- Choose `adafruit_debouncer` and `adafruit_ticks`
- They should appear in the `lib/` folder on the Pico

Using Debouncer

```
import board
import digitalio
from adafruit_debouncer import Debouncer

pin = digitalio.DigitalInOut(board.GP15)
pin.direction = digitalio.Direction.INPUT
pin.pull = digitalio.Pull.UP
switch = Debouncer(pin)

while True:
    switch.update()
    if switch.fell:
        print('Just pressed')
    if switch.rose:
        print('Just released')
```



HID devices

Now we're on to the fun bit

HID devices



Install the library

- `adafruit_debouncer`
- `adafruit_ticks`
- `adafruit_hid`

Keyboard time

```
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
from time import sleep

keyboard = Keyboard(usb_hid.devices)

while True:
    keyboard.send(Keycode.CAPS_LOCK)
    sleep(1)
```

Some words of warning

- If you write the wrong code it will just type loads of letters into your laptop and it's gonna be really annoying
- Test as you go and put in `time.sleep(1)` if in doubt
- If you end up unable to edit your own code you can reset it by plugging in the pico while pressing the BOOTSEL button. Then start again from uploading the uf2

Now we're on to the fun bit

- You can get button presses from the real world
- You can press keyboard buttons in software
- <https://binney.github.io/electronics/> for some useful links and code snippets
- Some ideas:
 - Mouse left and right click, mouse move, scroll. Tab and space. Caps lock. Page up/down. Home and end. Ctrl, alt, Win and command. Multi key shortcuts e.g. take a screenshot. Media controls: volume up/down, play, pause, mute and unmute speaker, mute and unmute microphone. Launch calculator?! Gamepad controls: joysticks, dpad, square, triangle, x, circle, l1/2/3 r1/2/3.
 - Advanced ideas: pass data back from the computer to the Pico via **reports**. LEDs, vibrate.
- Go forth