

Pico course

Andrew McDonnell

On behalf of the IEEE Computer Society SA Chapter
July 11, 2023

About the presenter

- Long time professional software developer, networking, DSP & HPC
- Nowadays in cybersecurity (appsec / testing)
- Some example contributions
 - <https://hackaday.io/project/4758-sentrifarm>
 - <https://github.com/pastcompute/pico-oregon-to-mqtt>
 - <https://www.youtube.com/watch?v=R3sGlzXfEkU>

Assumptions

About you

- you may have some knowledge of Python
- you may instead know about another computer language
- neither are essential, but will be helpful
- however you will need competency in common computer skills

And also you have

- A laptop capable of working with a USB drive-like device
- A laptop that you connected to your phones or the workshop wifi
- A serial terminal program (not essential, but very helpful)

About these slides

The intent of this course is to **do it yourself, ask questions, and try things!**

As such, the slides do not form a complete tutorial but rather act as a guide

So you may wish to take your own notes to augment them

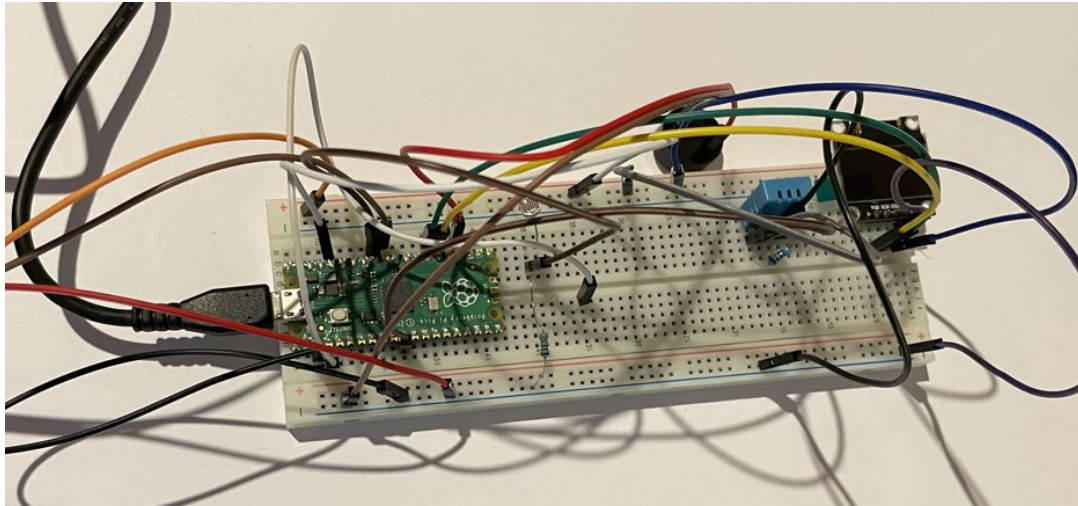
As we will see, you can even copy/paste the code from github if you like

Occasionally there may be some photos, to help you get the idea.

(You might notice they are not professionally laid out, this is by design...)

Don't forget to have fun!

Your kit might end up like this at the end. That's ok!



About your kit

Your kit includes a Raspberry Pi Pico, along with a number of components including LEDs, switches, resistors, sensors and an OLED display

The kit also includes a breadboard and jumper cables

We will learn about these as we go

A full is the github (next page)

Speaking of github

See <https://github.com/pastcompute/course-ieee-cs-pico>

You will have been informed by email in the lead up to today

Hopefully you downloaded everything!

Course practical overview

- Introduce the Raspberry Pi Pico, the wokwi simulator, and CircuitPython
- Loading circuit python and flashing the onboard LED
- Basic Digital I/O using LEDs and switches
- Using the serial port for debugging
- Displaying text and graphics with the OLED
- Measuring temperature and humidity
- Making sound
- Detecting light and sounds

Course flow

For each of the practical areas

- We will start with an introduction to the concepts
- Go through wiring up the board, either on the simulator or with your kit
- Go through entering the program and running it
- Take your time to experiment and learn, ask question or challenge yourself

WARNING

Please take care with your cables.

Don't force the micro end of the USB cable into the PICO it only goes one way

In particular, try and avoid shorting pins or jumpers to the wrong place!

Each time connect and double check before connecting the USB to the laptop

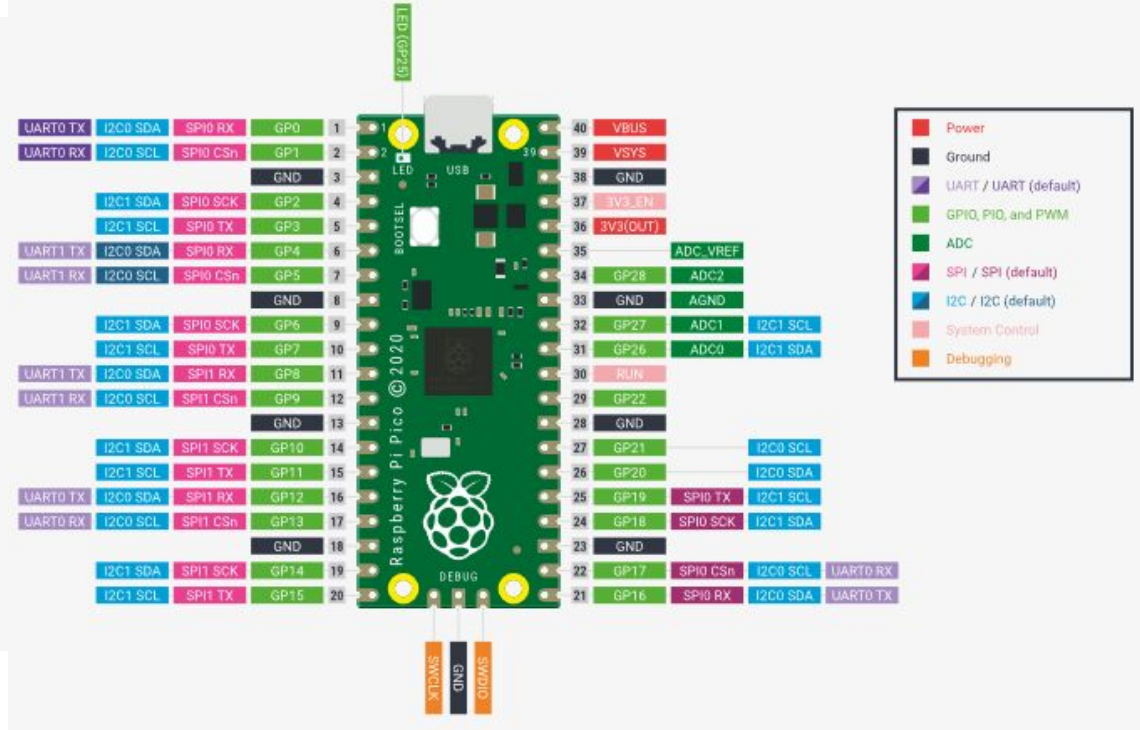
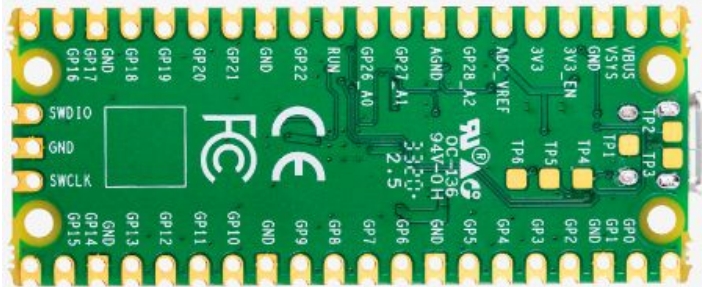
Introduction - Raspberry Pi Pico

- Microcontroller and official board was released in January 2021
- Targeted embedded systems, hobbyist and electronics applications
- ARM based MCU, with common functions such as A/D, i2c and SPI
- Set apart by dual core, the PIO module, and easy programming by default
- The main RP2040 ARM chip also available individually and in other boards
- On-board flash
 - (with circuit python, looks like a disk drive)

How the PICO compares to well known small computers

- Raspberry Pi Pico / RP2040 chip
 - Small, cheap, fast, low power use, won't support a complex operating system
 - variants - standard (in your kit), wifi/bluetooth, other manufacturers boards
- Raspberry Pi
 - runs Linux, does many things, is much more capable for tasks that require it
 - physically larger, uses more power, has multiple USB ports, etc.
- Arduino and clones
 - MCU like the Pico, more established in the market, simpler
 - Slower, one core, many different MCUs, no PIO, variable sizes and costs

Pico hardware overview. (bookmark this page!)



Ways to write software for the pico

Using C or C++ and the Pico SDK (not covered today)

Using C++ and the Arduino SDK (not covered today)

Using micro python, or circuit python

- we are using the latter today

Ways to program the pico

- Using UF2 files as a disk drive (we will do this today, to load circuit python)
- When using CircuitPython, we can save Python using standard tools
- Using a Raspberry Pi, another pico, or Arduino, etc (not today)

Please download the file ending in '.uf2' from the Github

(if you did not do this already)

Lets program the Pico with the Circuit Python firmware

- Find the file you downloaded earlier in your file explorer
- Hold down the button on the pico
- At the same time plug the USB cable into the computer
- It should be detected as a USB Disk Drive
- Copy the Circuit Python uf2 file to it (drag, etc.)
- It should take a few seconds to save, then it should disconnect and reset

We only need to do this once today!

Test that it worked...

- Unplug the cable and plug it back in
- Your PICO drive in your file explorer should have some additional files now
 - settings.toml
 - boot_out.txt
 - code.py
 - A folder caller lib
- The important one is code.py - open this in a text editor like Notepad

Simple LED test

Type the following into the notepad, then save the file as code.py

```
import board
import digitalio
import time

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

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

If everything went as expected, shortly the LED on the board should now flash!

Introduction to the wokwiki simulator

This lets you use the browser to write programs and run them in a simulator

Then you can download them into the Pico (UF2 for C, or copy/paste Python)

Within the limits of the software you can try things before you build them

Wikwi

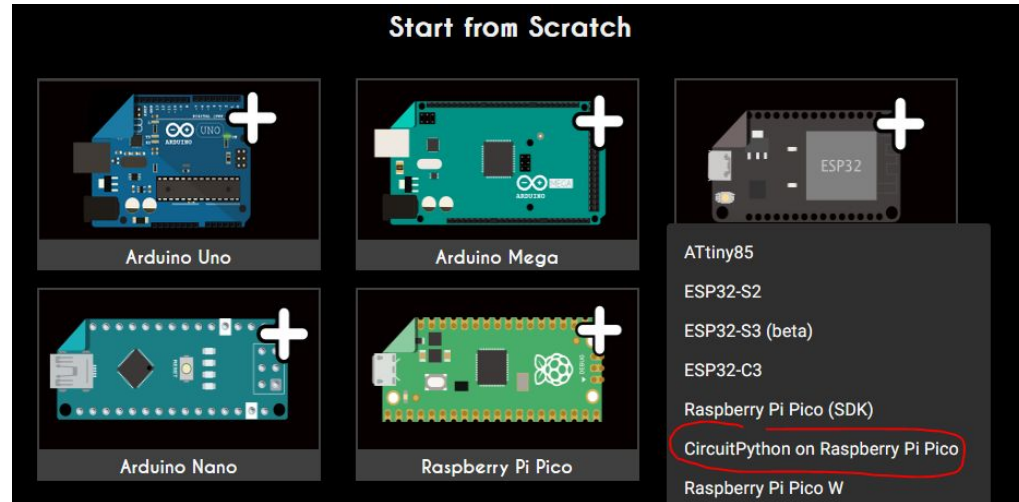
Create a new project by going to
<https://wokwi.com/projects/new/circuitpython-pi-pico>

Or, go to <https://wokwi.com/> (create an account so you can save your work)

Scroll down to “Start from Scratch”

Choose other options, and

Circuit Python on Raspberry Pi Pico



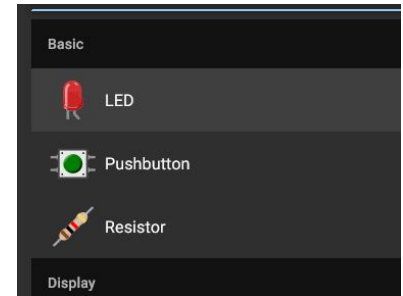
Simulation - circuit

To get started, we will first simulate flashing a LED

It may help to open the 'Docs' button to a new window and scroll to diagram editor

To get started:

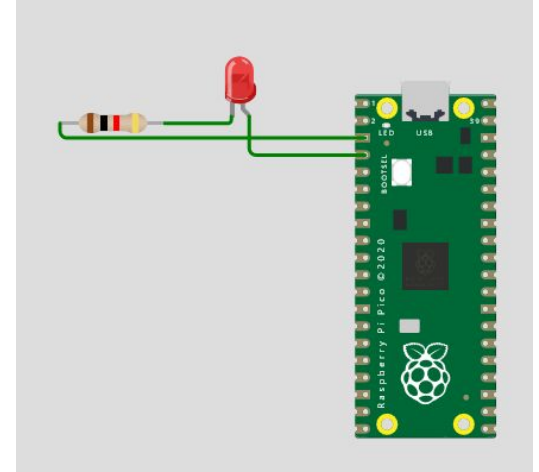
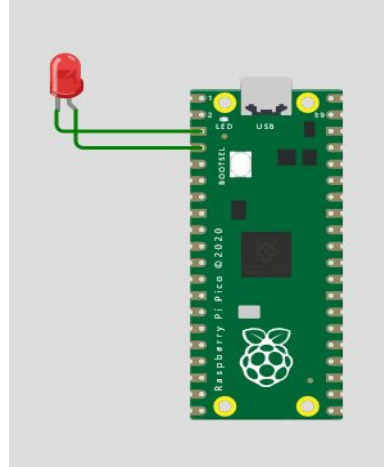
- Click the blue Plus icon to add a part
- Select the LED
- Click it and drag near the top left
- Click one end of the LED wires and drag to the fourth pin on the Pico
- Click on the other LED leads and then drag to the third pin



Simulation - LED

The rest of the slides, we will skip the detailed instructions, just follow along and ask for help if needed...

To add a resistor, delete the existing wires, add a resistor, and new wires



Simulation - updated led program

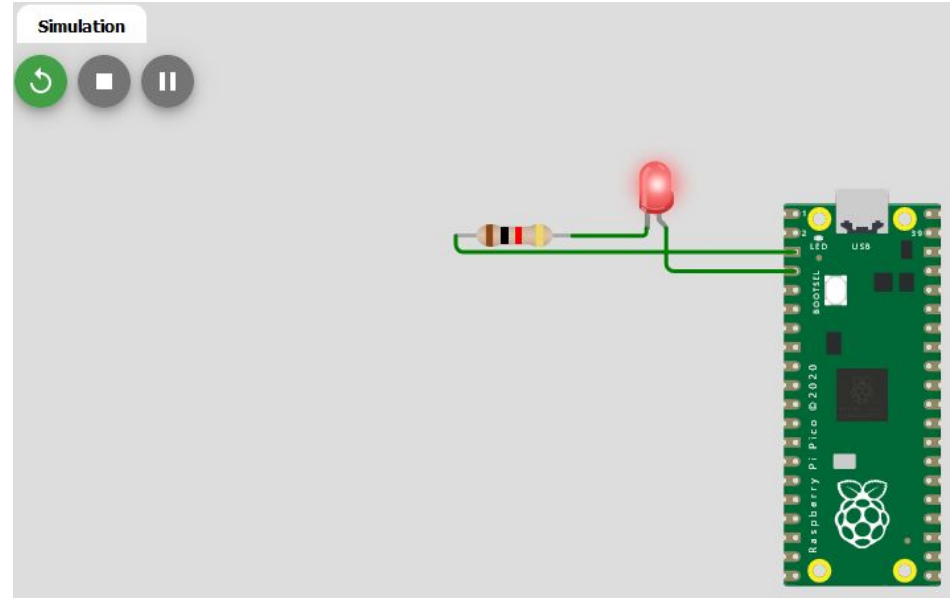
GIT EXAMPLE: 2

```
import board
import digitalio
import time

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

led2 = digitalio.DigitalInOut(board.GP2)
led2.direction = digitalio.Direction.OUTPUT

while True:
    led.value = True
    led2.value = False
    time.sleep(0.5)
    led.value = False
    led2.value = True
    time.sleep(0.5)
```



Save (or CTRL+S) then Play (blue play button)
The LEDs should alternately flash

Aside about controlling LEDs and other digital I/O

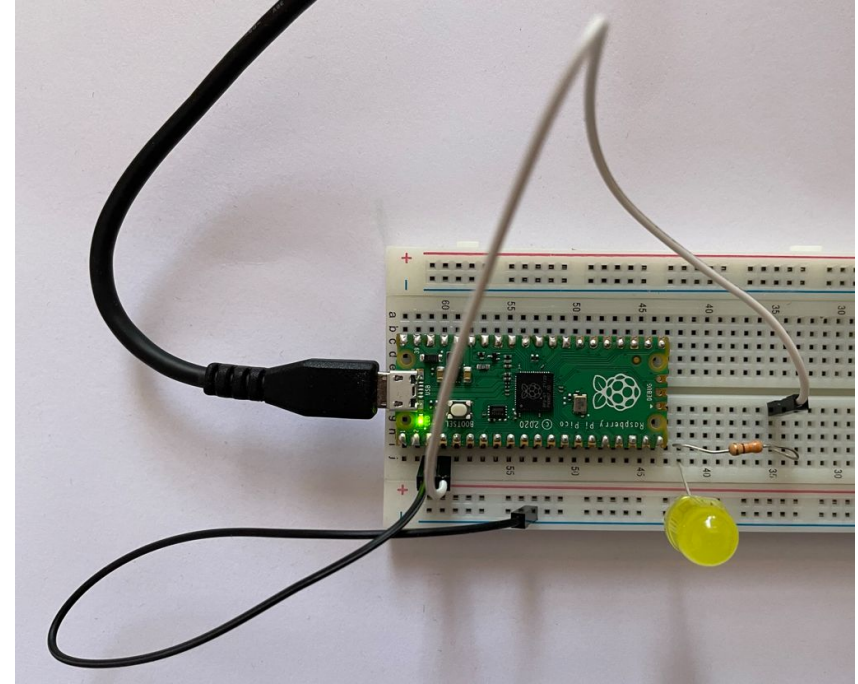
- This is not really a course about electronics in general, but this is important!
- there are two ways round the for LED to work
 - It has a Cathode and Anode
 - Anode connects to positive power, and Cathode to 0V (or negative power)
 - By using the Digital I/O for power, the LED turns on when we set it to 1 (True)
 - With a micro, it can also be the other way around
 - Instead, connect the Anode to power (3V) and the cathode to digital I/O
 - This will turn the LED on when we set it to 0 (False)
- The resistor is important because it stops the board “sending” or “sinking” too much current through the LED, and also changes the brightness

THIS IS NOT AN ELECTRONICS COURSE ... DESCRIPTIONS ARE APPROXIMATE!

Connecting your kit

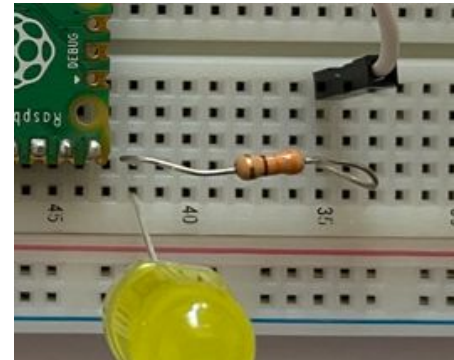
Now you have seen the simulation, we are going to do the same using the parts you have in your kit.

- Have handy your discovery kit manual pinout back page handy
- Do not connect USB yet



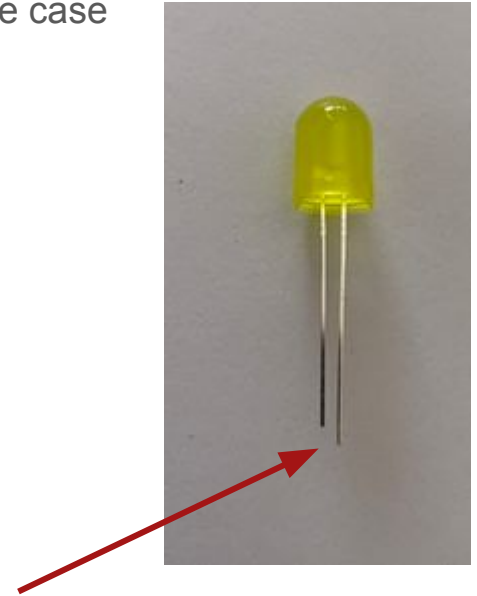
Connecting your kit

- Connect the pico with the USB socket to the edge of the breadboard
- Connect the jumper wires
 - Pico pin 4 to, for example, row 35 on the same side
 - Pico pin 3 to the ground bar (blue negative sign)
- Connect the resistor
 - The resistors in your LED kit are 470Ω , they have a yellow and purple colour bar
 - (this might be a tad low for many applications but is fine for our learning)
 - Connect one end to another hole on row 35
 - Connect other end to for example, row 42



Connecting your kit

- Connect the LED
 - The longer LED lead is the “Anode”, this is also flattened on the case
 - The shorter cathode goes to the ground bar
 - The longer anode joins the resistor, on row 42



Connecting your kit

- WARNING!
 - Connecting the wrong (power) pins incorrectly can damage your board!!!!
 - In particular, take care not to short pins together, especially to power or GND
 - Also take care to connect the correct pins, otherwise you have face frustrating bugs
 - Note, you won't hurt the LED if you get it backward, it just wont turn on
- Now, finally, connect the micro USB
 - Take care to get it the right way around
 - Lastly, connect it to the computer
 - From the test program before, the board LED should be flashing

Running our circuit python program

- Find the USB drive corresponding to the pico
- Open the file code.py in your text editor
- Delete the content
- Copy/paste the program from the wokwi editor
- When you save it, both LEDs should flash if everything is wired properly!


You can now experiment, by changing the time the LEDs are on, for example...

Preparing circuit python for additional tasks

We can also use libraries - these help us talk to do other interesting things

We have one for the temperature sensor, we'll need later

- Use the files from the git repository
- Copy the mpy files to D:\lib (the lib folder on the pico drive)
- And copy font5x8.bin to D: (the root folder of the pico drive)



- adafruit_dht.mpy
- adafruit_ssd1306.mpy
- adafruit_framebuf.mpy

Debugging with the serial port

- Hopefully you installed putty, picocom, or another terminal program
- The operation to find the serial port varies between computer
 - Windows - open device manager, find the COM port e.g. COM3 or whatever the USB COM is
 - Linux - it is probably /dev/ttyUSB0 but could be /dev/ttyACM0
 - Mac - I haven't had a chance to test, sorry!
- Connect the terminal program with speed 115200
- Edit and save code.py again, you should see

```
Code stopped by auto-reload. Reloading soon.  
soft reboot
```

```
Auto-reload is on. Simply save files over USB to run them or enter  
REPL to disable.  
code.py output:  
Hello, Pi Pico!
```

Basics of digital I/O

- As we saw with the LED, we have
 - Pins - GP0 through to GP25, see the discovery kit pinouts
 - By default these are digital I/O but can be change to perform other functions
 - GP25 is hard wired to the on board LED
 - GP0 and GP1 may be mapped to a serial port by default, so avoid these
- The pins can be configured to be output or input

Digital I/O

Outputs

- Writing True, corresponding to binary '1', (high), turns on a pin
- If you measure with a multimeter it will show 3.3Volts
- Writing False, corresponding to binary '0', turns off a pin
- If you measure with a multimeter it will show 0Volts
- As we saw this can turn on a LED
- With additional components, we can buzz a speaker, or turn on a relay

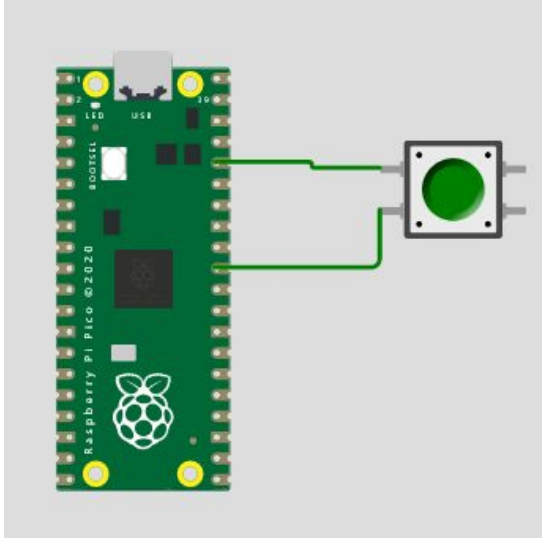
Digital I/O

Inputs

- We can read whether the pin is high or low
- Thus, a switch shorting a pin to 3V3 will read high, or to GND will read low
- We usually use other components, and always use resistors
- Without the resistors, we could read a random value if not connected
- Lets demonstrate with an example

Responding to a push button

Configure the following in wokwi, and run the following program



GIT EXAMPLE: 3

```
import board
import digitalio
import time

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

button1 = digitalio.DigitalInOut(board.GP26)
button1.direction = digitalio.Direction.INPUT
led.value = False

while True:
    time.sleep(0.1)
    if button1.value == True:
        print("on")
        led.value = True
    else:
        print("off")
        led.value = False
```

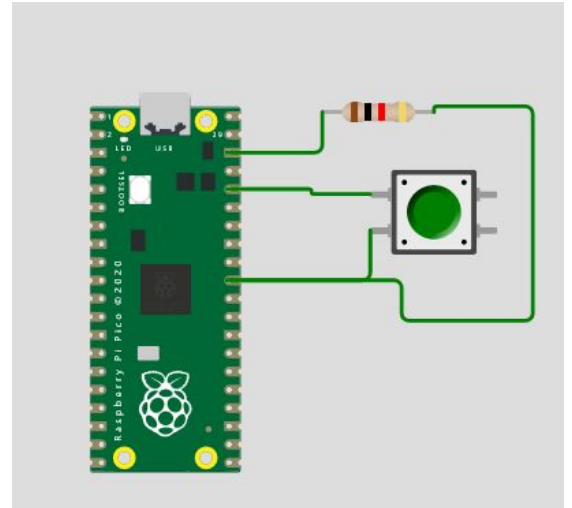
Fixing the switch

It is always on, or gets stuck due to noise... we need to add a “pulldown” resistor

This time, the switch works as we hope

This is because, when the switch is not pushed, GP26 sees 0V through the resistor. When the switch is pushed, the 3V3 is directly connected.

There is a reverse mechanism called pull-up as well, you need different at different times depending on the circuitry.



Wiring it up

Now that it works in the simulator, you can do the same for your real device!

Remember, you need to copy/paste the text into your notepad or editor

You could also try keeping your existing LED and using that instead of the on-board LED used for the example.

Also try changing the time delay for when the switch is measured

Drawing on the OLED

The display in your kit uses an SSD1306 controller and the i2c interface

We will connect the pins:

- SDA to GP20 (pin26)
- SCL to GP21 (pin27)
- Vdd to the rail connected to the Pico 3V3 pin
- Gnd to the rail connected to Pico GND pin like we did with the LED

Drawing on the OLED

GIT EXAMPLE: 4

Use the following program

```
import board
import digitalio
import time
import busio
import adafruit_ssd1306

i2c = busio.I2C(board.GP21, board.GP20)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
display.fill(0)
display.show()

dw = display.width
dh = display.height
display.line(1, 1, dw - 2, dh - 2, True)
display.line(dw-2, 1, 1, dh-2, True)
display.show()
```

(A very quick intro to i2c, used by the OLED)

- Uses digital I/O for communication between chips
- 2 wires - SCK (clock), SDA (data), the host requests a sensor to respond
- Good articles at
 - <https://learn.sparkfun.com/tutorials/i2c/all>
 - <https://learn.adafruit.com/working-with-i2c-devices/terminology>

Luckily for today, all this is hidden from us by Circuit Python!

Drawing text on the OLED

Once you learn this, you can use it for your remaining exercise if you like

```
import board
import digitalio
import time
import busio
import adafruit_ssd1306

i2c = busio.I2C(board.GP21, board.GP20)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)

counter = 0
display.fill(0)
while True:
    display.show()
    time.sleep(1.0)
    display.fill(0)
    display.text("{}".format(counter), 4, 8, True, size=2)
    counter = counter + 1
```

Reading temperature using a simple digital sensor

We will use the DHT-11 - this uses a custom single wire protocol

We also add a pull-up resistor on the data line, just to make sure

- Look at the pins left to right from the grid side of the sensor
- Connect (1) to the rail connected to the Pico 3V3 pin
- Connect (2) to a GPIO, e.g. GP7 (pin 10)
- Connect (4) to the rail connected to Pico GND pin like we did with the LED
- From your resistors, find a 4k7 and connect it between (1) and (2)



Reading temperature using a simple digital sensor

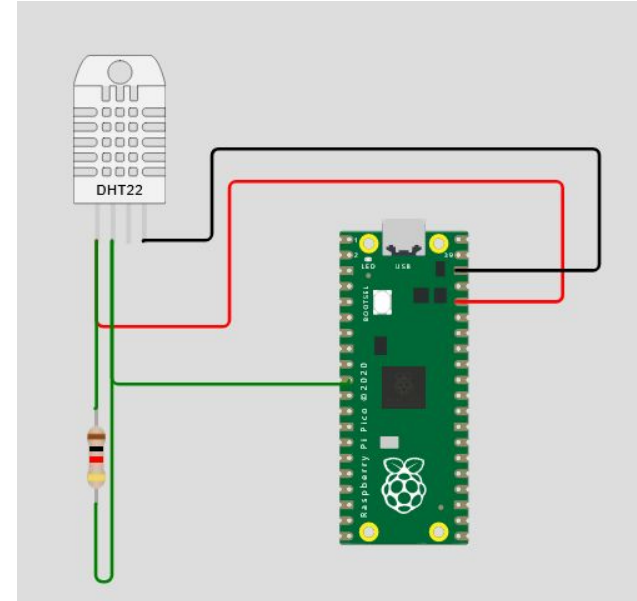
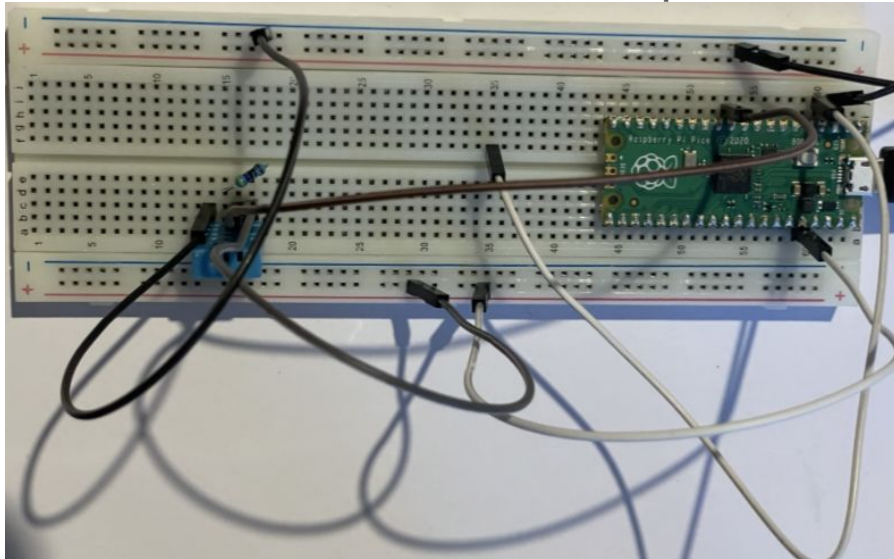
- What about that 1-wire protocol?
- Think back to i2c...
- it is very similar, except there is no clock wire
- This means it might not be able to send data as fast
- It also means you can usually only have one sensor per wire, unlike i2c
- If you are interested, there will be good tutorials on adafruit website, etc.



Reading temperature (continued)

We can also do it in the simulator as well

- except the simulator has a different model, a DHT-22
- the simulator needs a requirements.txt file



Reading temperature using a simple digital sensor

You need to have previously setup
the adafruit mpy module as described earlier

Use the code as shown
(or copy it from the github)

Open the terminal to see the result!

If it is working, the LED should also flash

```
import board
import digitalio
import time
import adafruit_dht
led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT
led.value = True
dht = adafruit_dht.DHT11(board.GP8)
time.sleep(3.0)
while True:
    try:
        led.value = True
        t = dht.temperature
        h = dht.humidity
        time.sleep(0.2)
        print(dht.temperature, dht.humidity)
        led.value = False
        time.sleep(0.8)
    except RuntimeError as e:
        print("DHT error", e.args)
        time.sleep(3.0)
```

Altogether - a simple weather display...

```
import board
import digitalio
import time
import adafruit_dht
import busio
import adafruit_ssd1306

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

i2c = busio.I2C(board.GP21, board.GP20)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
display.fill(0)
display.show()

dht = adafruit_dht.DHT11(board.GP8)
time.sleep(3.0)
```

```
while True:
    try:
        led.value = True
        t = dht.temperature
        h = dht.humidity
        time.sleep(0.2)
        print(t, h)
        display.fill(0)
        display.text("{:2d}%".format(h), 4, 8, True, size=2)
        display.text("{:4.1f}".format(t), 50, 8, True, size=2)
        display.text("o".format(t), 98, 6, True, size=1)
        display.show()
        led.value = False
        time.sleep(0.8)
    except RuntimeError as e:
        print("DHT error", e.args)
        time.sleep(3.0)
```

Basics of PWM

Pulse Width Modulation

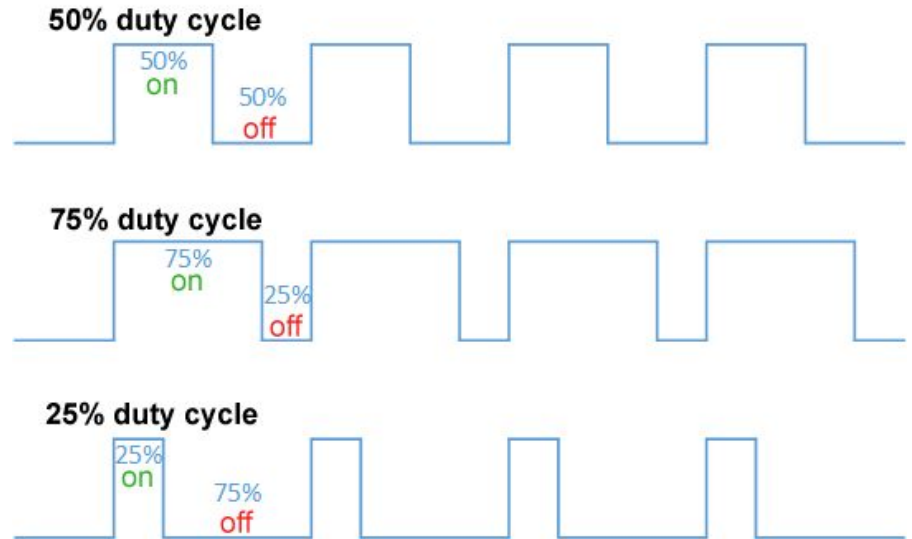
- vary how often digital I/O is on/off
- also vary the frequency

Useful for controlling motor speed

Useful for making buzzing audio

Useful for dimming a LED

The Pico can easily do this



Making a siren using PWM

- Connect one side of your buzzer to the GND link
- Connect the other side to the GP18 (pin 24) of the Pico

Making a siren using PWM

Experiment with the frequency or range...

```
import pwmio
import time
import board

buzzer = pwmio.PWMOut(board.GP18,
frequency=750, variable_frequency=True,
duty_cycle=0)

while True:
    for i in range(10):
        buzzer.frequency = 750 + i * 40
        buzzer.duty_cycle = int(65536 / 2)
        time.sleep(0.1)
```

Basics of Analogue I/O

The Pico will read a voltage and convert it to a digital number

Example - $0V \rightarrow 0$, $3v3 \rightarrow 65535$, $1.65 \rightarrow 32768$

We can use this to read various sensors, for example sound level, light level

This is also very useful with a voltage divider for reading higher voltages

Such as a car battery - although some care is required!

Reacting to changes in ambient light

We will use a light dependent resistor in a voltage divider

And with the voltage, the Pico can detect daylight

Connect one side of the LDR to the 3v3 rail, the other to a spare grid

Connect a 3k3 resistor to the LDR grid, and the other to the GND rail

Jumper the connection to A1 (pin 32) of the Pico

Reacting to changes in ambient light

When it is dark, the LDR resistance is high (maybe 50kΩ).

This makes the voltage to GP17 closer to zero.

When it is low, the LDR resistance is low (maybe 3kΩ)

This makes the voltage to GP17 move toward the 3V3 supply

Thus our program can measure this and act accordingly...

Let us see if our math is accurate!

$$V_{measured} = V_{in} \frac{R2}{R1 + R2}$$

$$V_{measured} = 3.3 \frac{3300}{R1 + 3300}$$

$$V_{light} = 3.3 \frac{3300}{3000 + 3300} = 1.65V$$

$$V_{dark} = 3.3 \frac{3300}{50000 + 3300} = 0.20V$$

Reacting to changes in ambient light

Use the following program

We will measure the LDR voltage

This uses the OLED from earlier...

```
import board
import digitalio
import time
import busio
import adafruit_ssd1306
import analogio

ldr = analogio.AnalogIn(board.A1)
i2c = busio.I2C(board.GP21, board.GP20)
display = adafruit_ssd1306.SSD1306_I2C(128, 32, i2c)
display.fill(0)
display.show()
while True:
    v = ldr.value
    volts = v * 3.3 / 65535
    print(v, volts)
    display.fill(0)
    display.text("{:.1f}V".format(volts), 4, 8, True, size=2)
    display.show()
    time.sleep(1.0)
```

Making a buzz when it gets dark!

Experiment with the ldr threshold...

```
import pwmio
import time
import board
import analogio

buzzer = pwmio.PWMOut(board.GP18, frequency=750,
variable_frequency=True, duty_cycle=0)
ldr = analogio.AnalogIn(board.GP16)

while True:
    if ldr.value < 33:
        buzzer.frequency = 750
        buzzer.duty_cycle = int(65536 / 2)
    else:
        buzzer.duty_cycle = 0
    time.sleep(0.1)
```

Detecting sound

Connect Vcc on the sound module in your kit to the 3V3 rail

Connect GND to the GND rail

Connect Out to A0 (pin 31)

You can test by clapping your hands, the second LED on the sensor should light

Detecting sound

Try the following program

Notice it a little similar to the LDR

Can you tell how it is different?

Hint: the sensor is normally 'high' voltage

And, we latch due to the transient pulse

```
import pwmio
import time
import board
import analogio

buzzer = pwmio.PWMOut(board.GP18, frequency=750,
variable_frequency=True, duty_cycle=0)
ldr = analogio.AnalogIn(board.A0)

threshold = 27000
while True:
    v = ldr.value
    if v < 64000: print(v)
    if v < threshold:
        buzzer.frequency = 750
        buzzer.duty_cycle = int(65536 / 2)
        time.sleep(2)
    else:
        buzzer.duty_cycle = 0
        time.sleep(0.01)
```


Review

Some References

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

<https://circuitpython.org/libraries>

<https://docs.circuitpython.org/projects/bundle/en/latest/drivers.html>

<https://github.com/adafruit/circuitpython>

https://github.com/adafruit/Adafruit_CircuitPython_Bundle