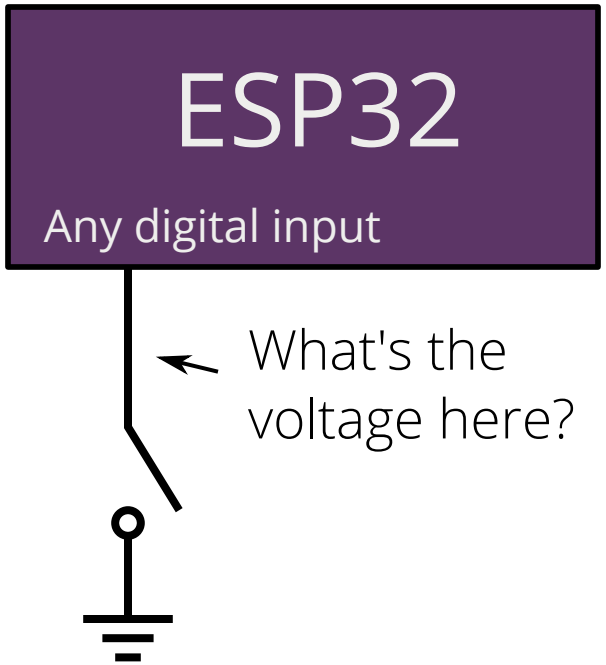
EN 1: Engineering in the Kitchen

Steven Bell

27 September 2023

Tufts UNIVERSITY

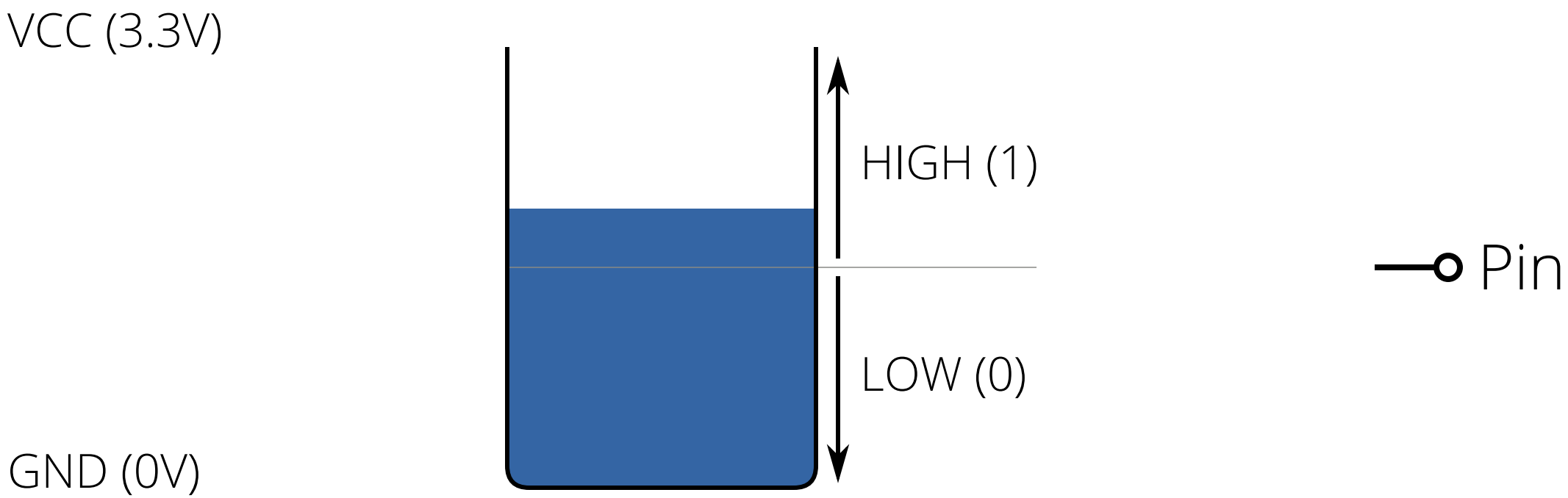
Wire up a switch



(Ground - "GND" pin in ESP32)

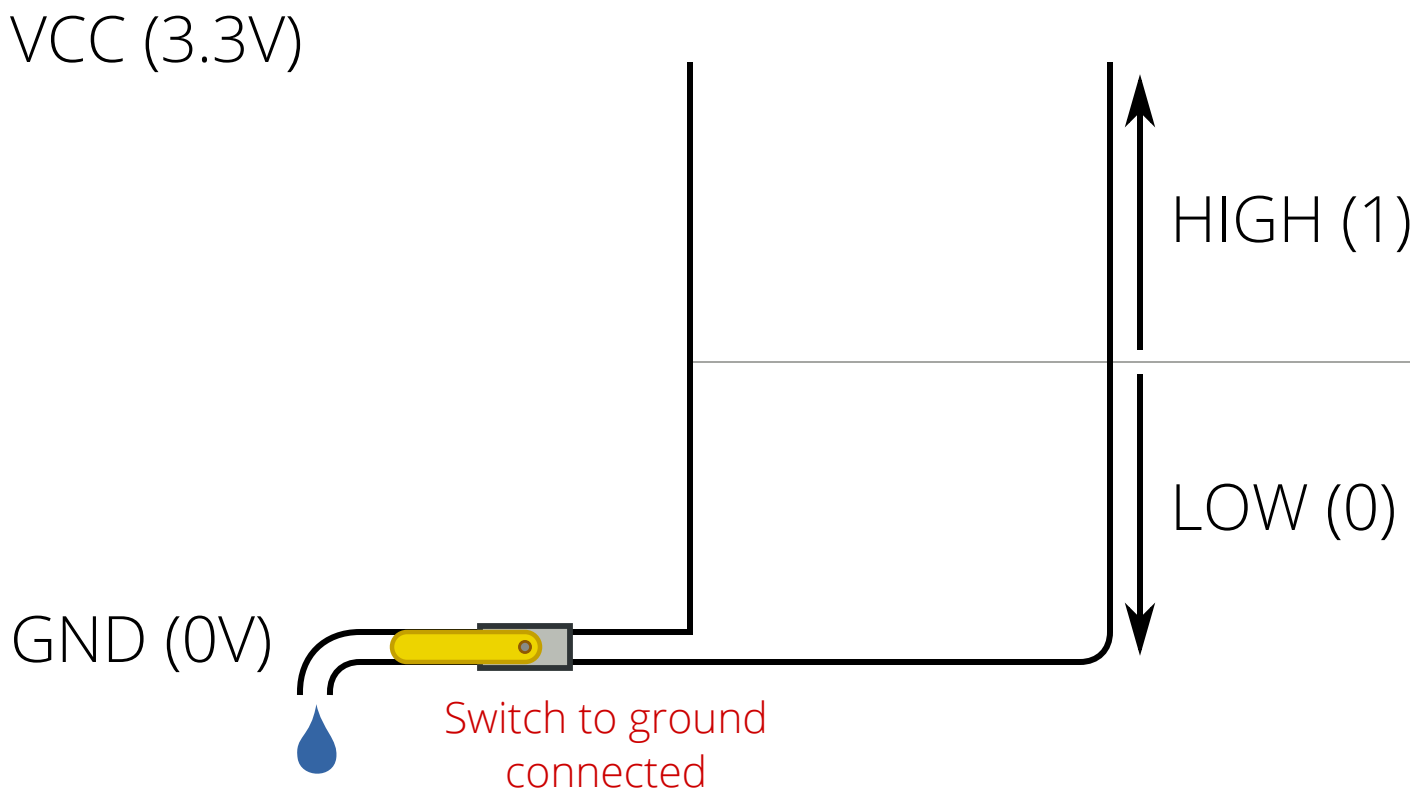
An analogy for an input pin

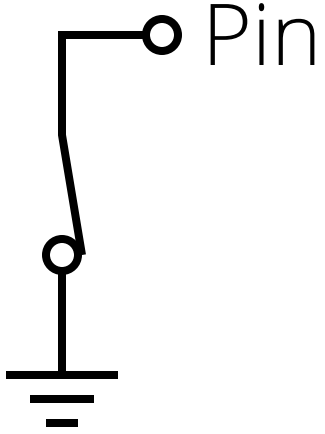
An input pin just reads whether the voltage (level) is above a threshold.



An analogy for an input pin

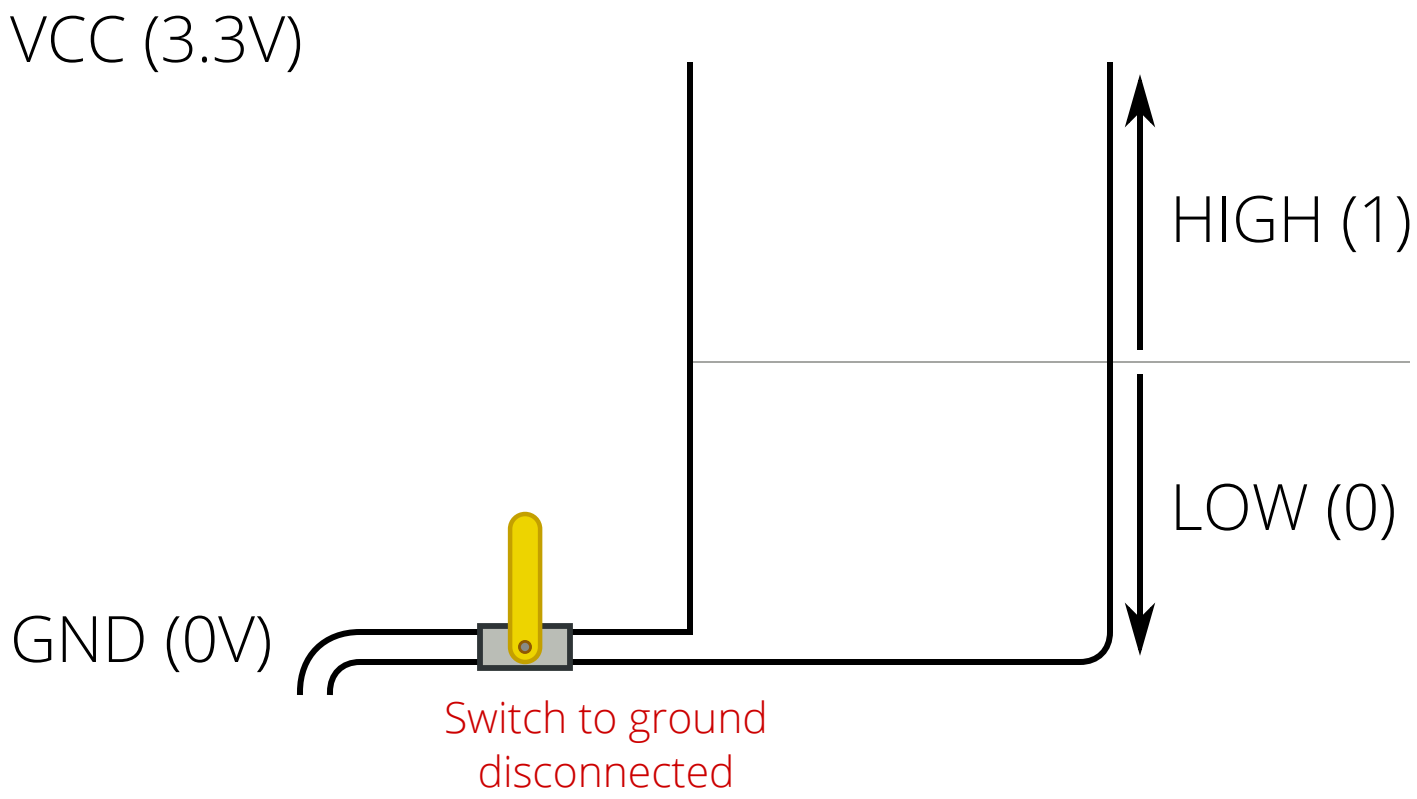
By reading the level, we can tell if the switch is making a connection to ground

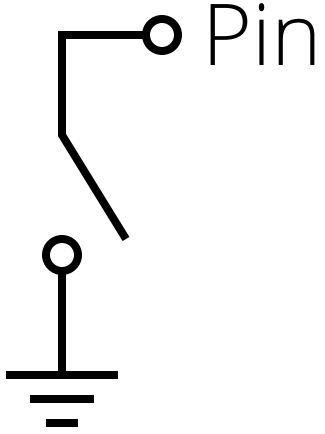




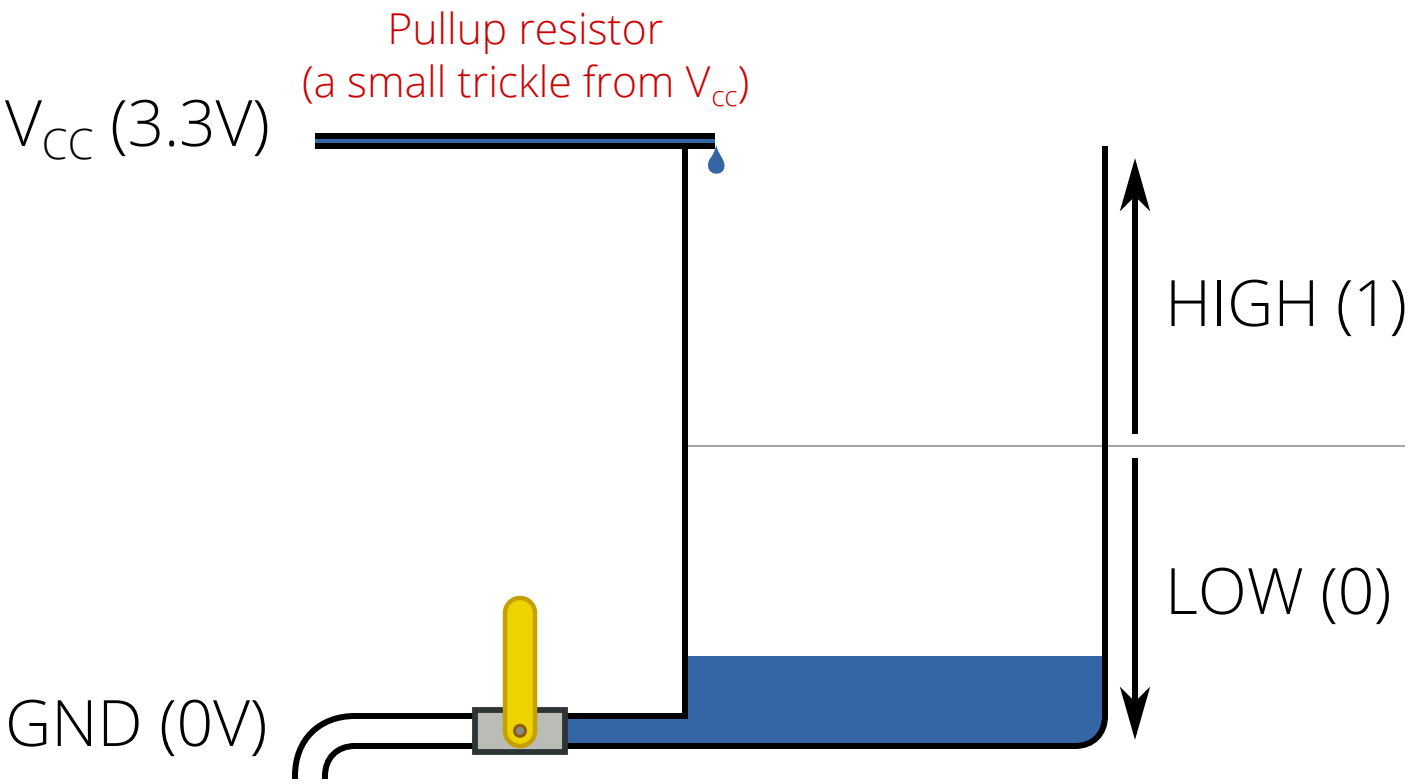
An analogy for an input pin

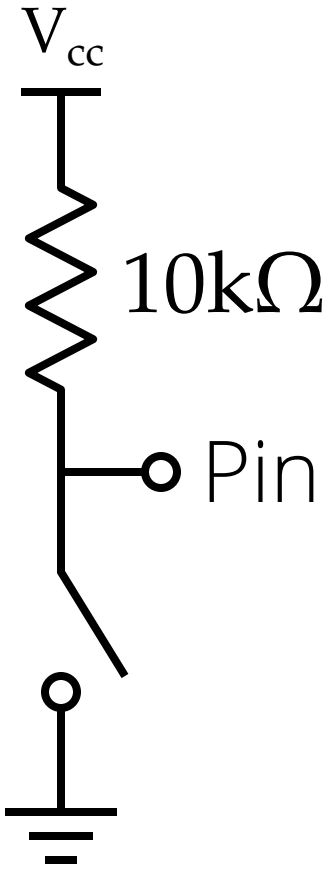
Except that when the switch is disconnected, we have no control over the level! We're at the mercy of the environment.



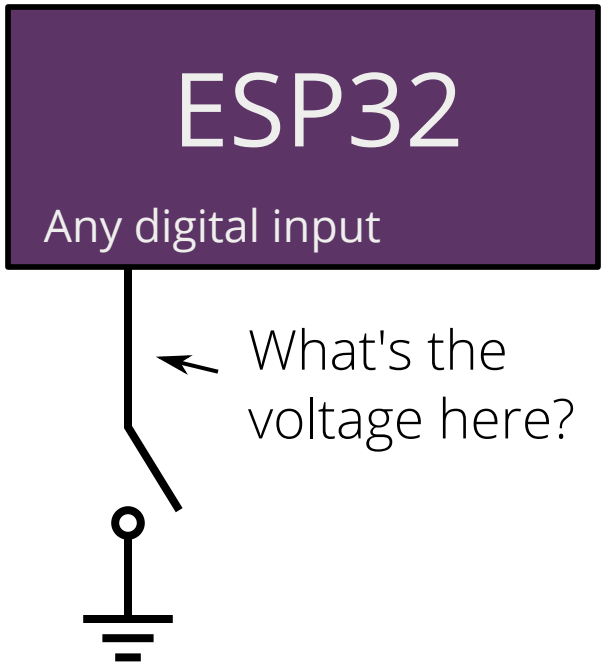


An analogy for an input pin



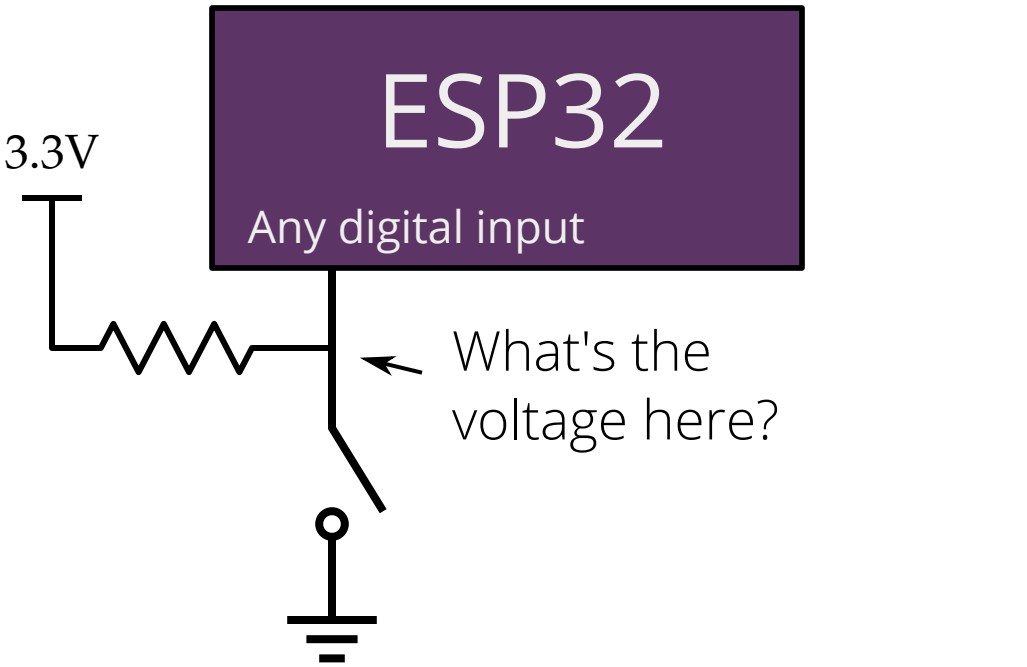


Wire up a switch



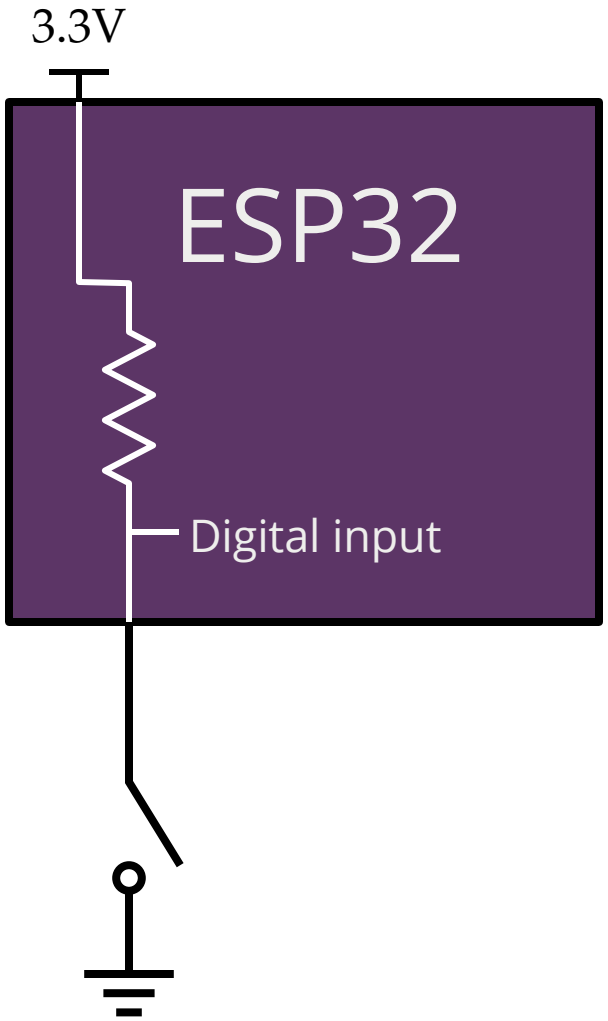
(Ground - "GND" pin in ESP32)

Wire up a switch



(Ground - "GND" pin in ESP32)

Internal pullups



(Ground - "GND" pin in ESP32)

Reading a switch

```
from machine import Pin # import necessary library
                        , make it an input
switch = Pin(12, Pin.IN, Pin.PULL_UP)
               ' pin to use       ' enable the pullup
print(switch.value())
```

Challenge: What happens if you leave out the `Pin.PULL_UP` part?

Challenge: Figure out how you can read the state of the switch if it is connected to 3.3V instead of GND.

A "normal" program:

Executes in sequence,
top to bottom.

Do thing 1.
If X, then
    Do thing 2.
Repeat 10 times:
    Do thing 3.
Do thing 4.

(all done!)

An embedded program:

Runs forever in a loop!

Repeat forever:
    Check some inputs
    if X, then
        Do thing 1
    Do thing 2
    if repetitions < 10:
        Do thing 4

(go back to top!)

An event loop in Python

```
from time import sleep
# Stuff that should happen once
from machine import Pin
switch = Pin(12, Pin.INPUT, Pin.PULL_UP)

while True:
    # Stuff that should happen repeatedly
    print(switch.value())
    sleep(1)
```

Constants in Python

We often have "special numbers" in our code

It helps to give them descriptive names instead of reusing the number everywhere.

```
PRESSED = 1

DELAY_TIME = 3000 # milliseconds

HELLO_MESSAGE = "Hello, microPython!"
```

`if` statements

```
if CONDITION:
  # Stuff to do if CONDITION is true
  # This is specified using indentation
else:
  # Stuff to do if CONDITION is false

# Stuff that is outside of the if-statement (un-indented)
```

CONDITION can be lots of things:

```
a == 3 # Check if a is equal to 3
x > y  # Check if x is greater than y
button.value() == 1
```

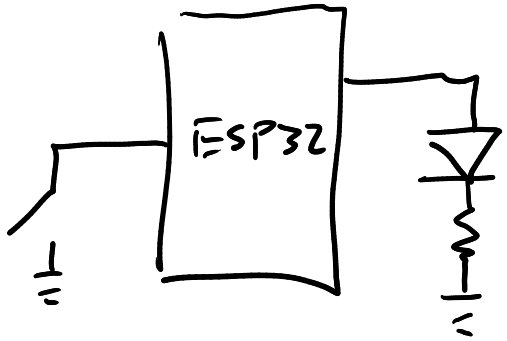
`if/elif` statements

"elif" is a contraction of "else if"

```
if FIRST_CONDITION:
  # Stuff to do if FIRST_CONDITION is true
elif SECOND_CONDITION:
  # Stuff to do if SECOND_CONDITION is true
else:
  # Stuff to do if neither condition is true
```

Write a program that turns on an LED when a switch is pressed.

(The ESP32 should read the switch and control the LED, don't just put the switch in series with the LED)



Challenge: Make the LED blink while the switch is pressed.

How do we *toggle* the LED when the switch is pressed?

```
last time = not pushed

while true:
    now = switch.value()
    if last time not pushed and now pushed:
        do stuff

    last time = now
```

How do we *toggle* the LED when the switch is pressed?

```
set LED state off
set switch state not pushed
while True:
    read the current state of the switch
    if the switch is pressed (and wasn't pressed last time):
        toggle the state of the LED
        set the LED to the new state
    save the current state as the last state
```

Challenge: Make the LED turn on for 2 seconds when the switch is pressed. It should go off after 2 seconds, even if the button is held down.

Challenge: The above, but you can't sleep() for more than 0.1 second.

OLED display

```
from s2pico_oled import OLED
from machine import Pin, I2C

i2c = I2C(0, sda=Pin(8), scl=Pin(9))
oled = OLED(i2c, Pin(18))
oled.test()
```

Draw some text at (0, 0):

```
oled.text("Yay MicroPython!", 0, 0, 1)
oled.show()
```

Course website has link for more functions!