Introduction to Embedded Systems (IES)

Module 3
Interfacing to sensors

Version 2022-12-19, Kjeld Jensen kjen@sdu.dk

In this module we will interface sensors to the Raspberry Pi Pico microcontroller and explore how to read data from the sensors using the Arduino programming language. We will begin with examples of digital sensors and then continue with examples of analog sensors.

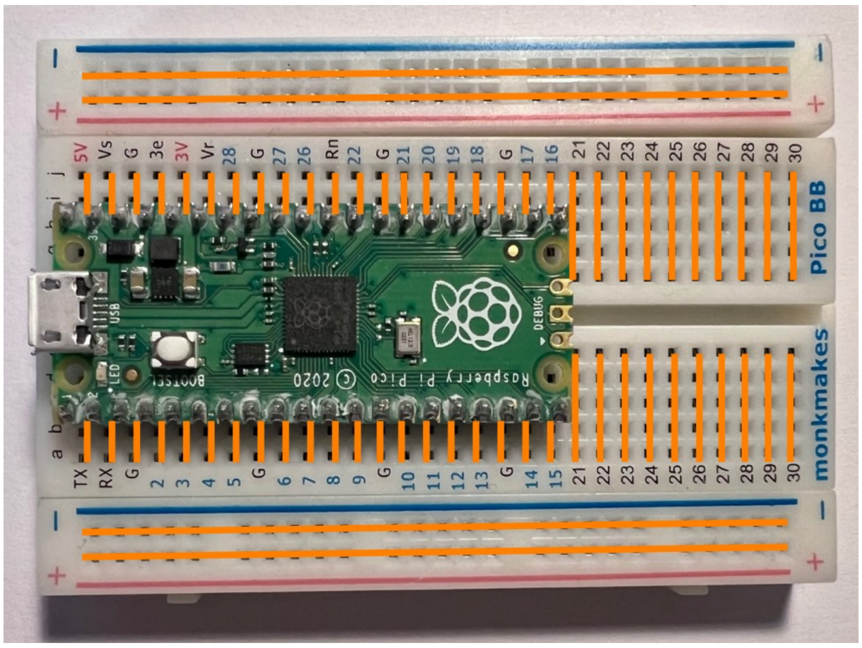
Agenda

1. Review of module 2
2. Feedback on submitted reports
3. Interfacing to sensors exercises
   A) Breadboard
   B) Push button (digital input)
   C) Water level sensor (digital input)
   D) Potentiometer (analog input)
   E) Computer mouse (analog input)

A) Breadboard

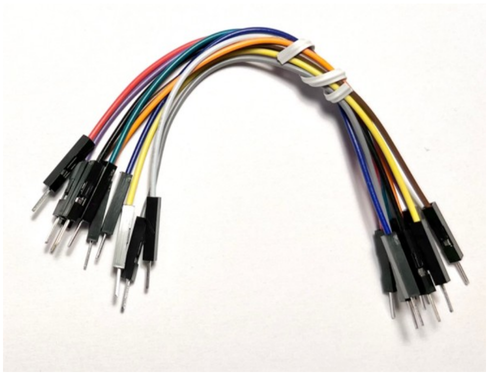
To connect sensors an other electronics to the Pico we use the breadboard that the Pico is mounted on.

The orange lines on the picture below shows how the pinholes are connected. These connections facilitate building up the experimental circuits using test wires.



5V Vs G 3e 3V Vr 28 G 27 26 Rn 22 G 21 20 19 18 G 17 16 21 22 23 24 25 26 27 28 29 30
TX RX G 2 3 4 5 G 6 7 8 9 G 10 11 12 13 G 14 15 21 22 23 24 25 26 27 28 29 30
Pico BB
monkmakes
Raspberry Pi Pico © 2020

Connections between breadboard pinholes

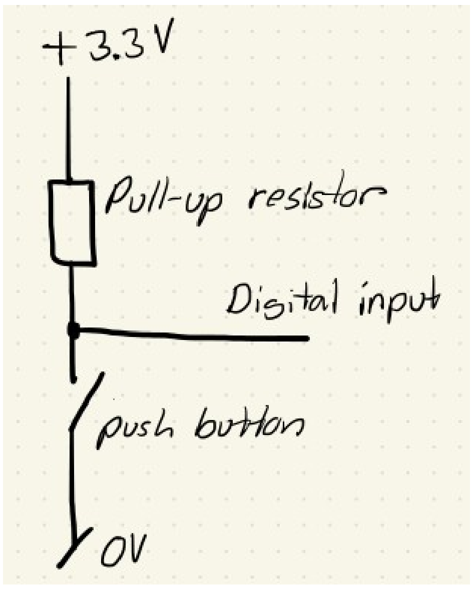


Test wires used to build circuits

Please notice that this information is also available in the file `ies_kit.pdf`

B) Push button (digital input)
In this exercise we will connect a push button to the Pico as a digital input sensor and then in the Arduino sketch read the status of the push button.

To read the push button as a digital input on the Pico we must make it switch between *logic high* (3.3V) and *logic low* (0V).

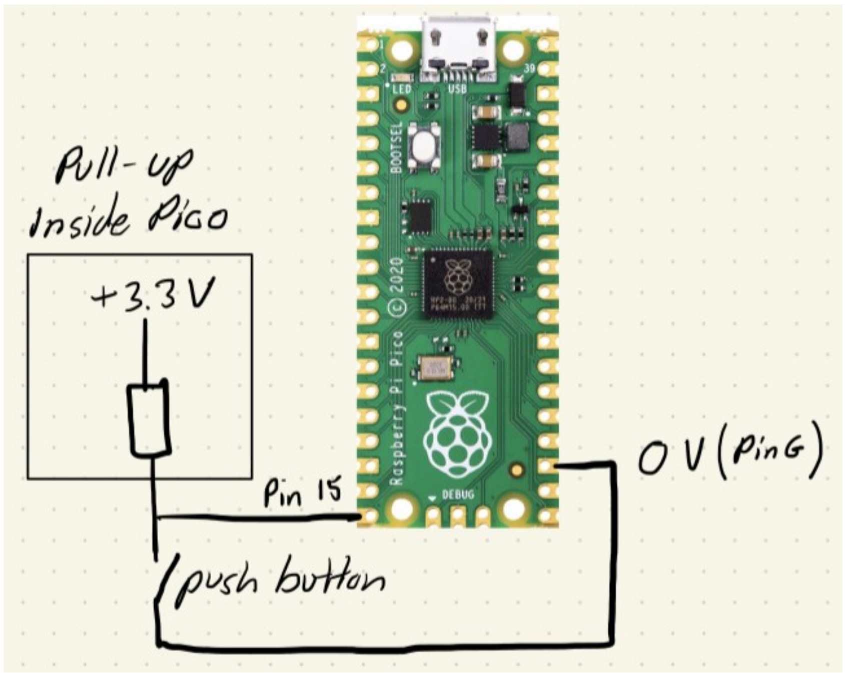


We do this by adding a pull-up resistor in series with the push button like the circuit above:

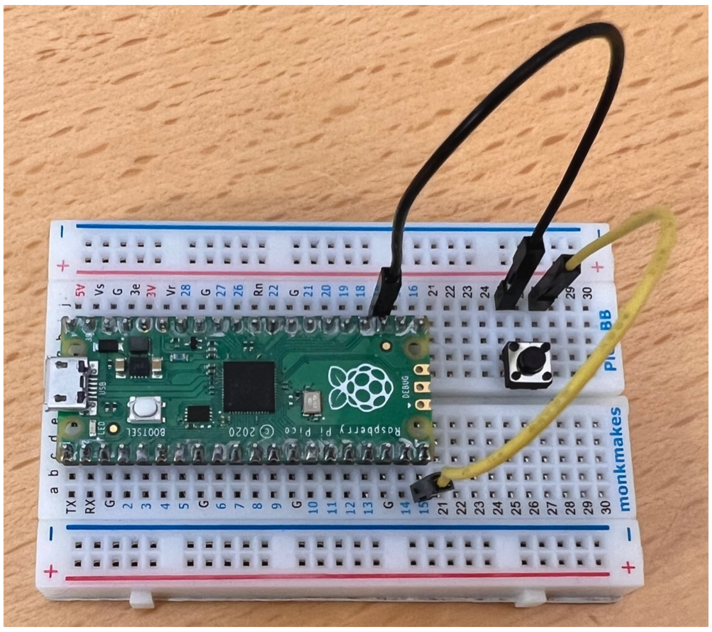
- When the button is open, *Digital input* will be logic high because the pull-up resistor is connected to 3.3V.
- When the button is pushed, the digital input will become logic low because of the direct connection between *Digital input* and 0V

Please notice that 0V is often called Ground, GND or G.

Exercise B.1) Please connect the push button to pin 15 on the Pico using the breadboard.



Please notice that the Pico has a built-in pull-up (50k Ohm) resistor that can be enabled, so you do not have to add this to the breadboard.



When completed it should look like this.

Exercise B.2) Please use the example program below to read data from the push button. A copy is available as `read_digital_input.ino` in the examples folder.

```
void setup() {
  // Setup Pins
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(15, INPUT_PULLUP);
}

void loop() {
  // Read Pin 15
  bool logic_high = digitalRead(15);
  if (logic_high == true) {
    // Turn off LED
    digitalWrite(LED_BUILTIN, LOW);
  } else {
    // Turn on LED
    digitalWrite(LED_BUILTIN, HIGH);
  }
}
```

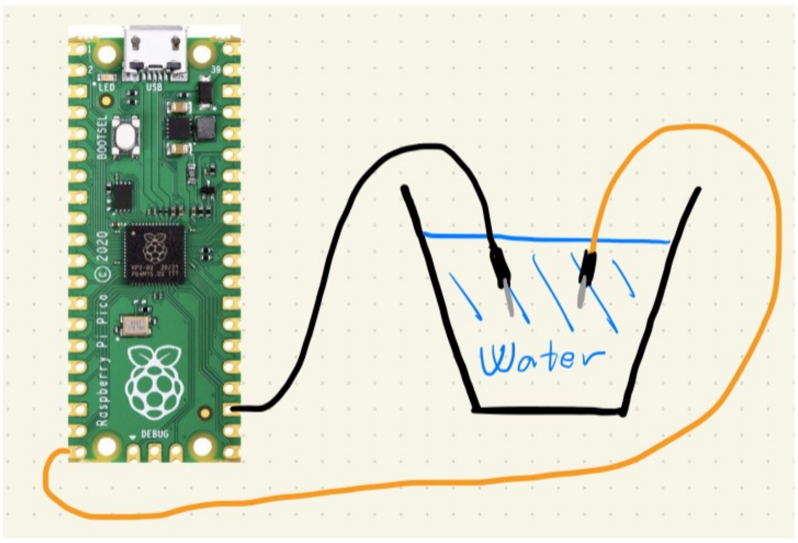
Question B.3) What happens when you push the button? Please explain in your own words, why this happens.

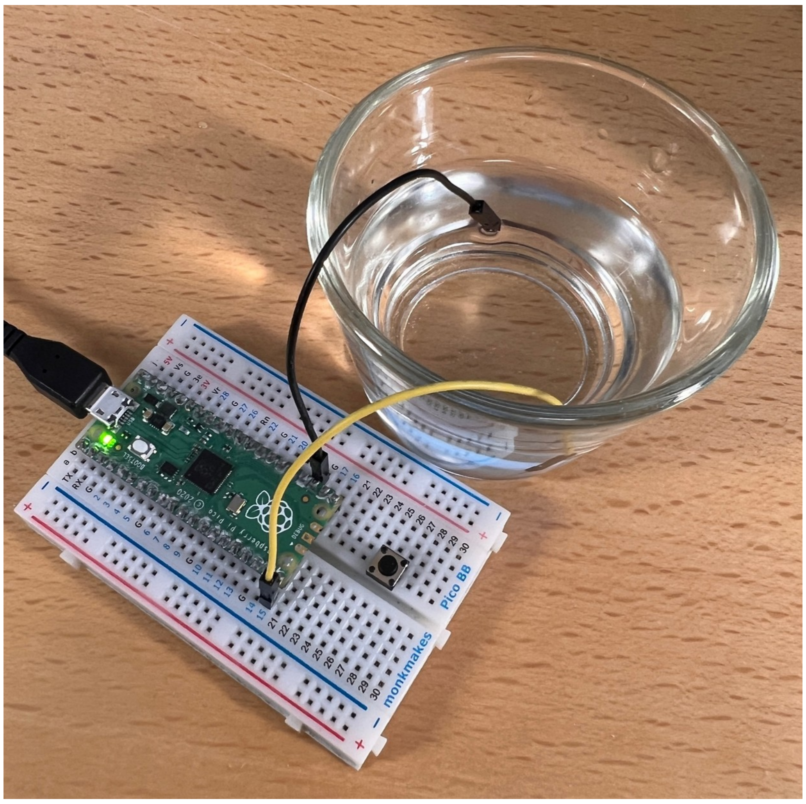
C) Water level sensor (digital input)

In this exercise we will remove the push button and instead make a water level sensor.

This is very simple as the water is conductive like a pushed button, though with a much higher resistance.

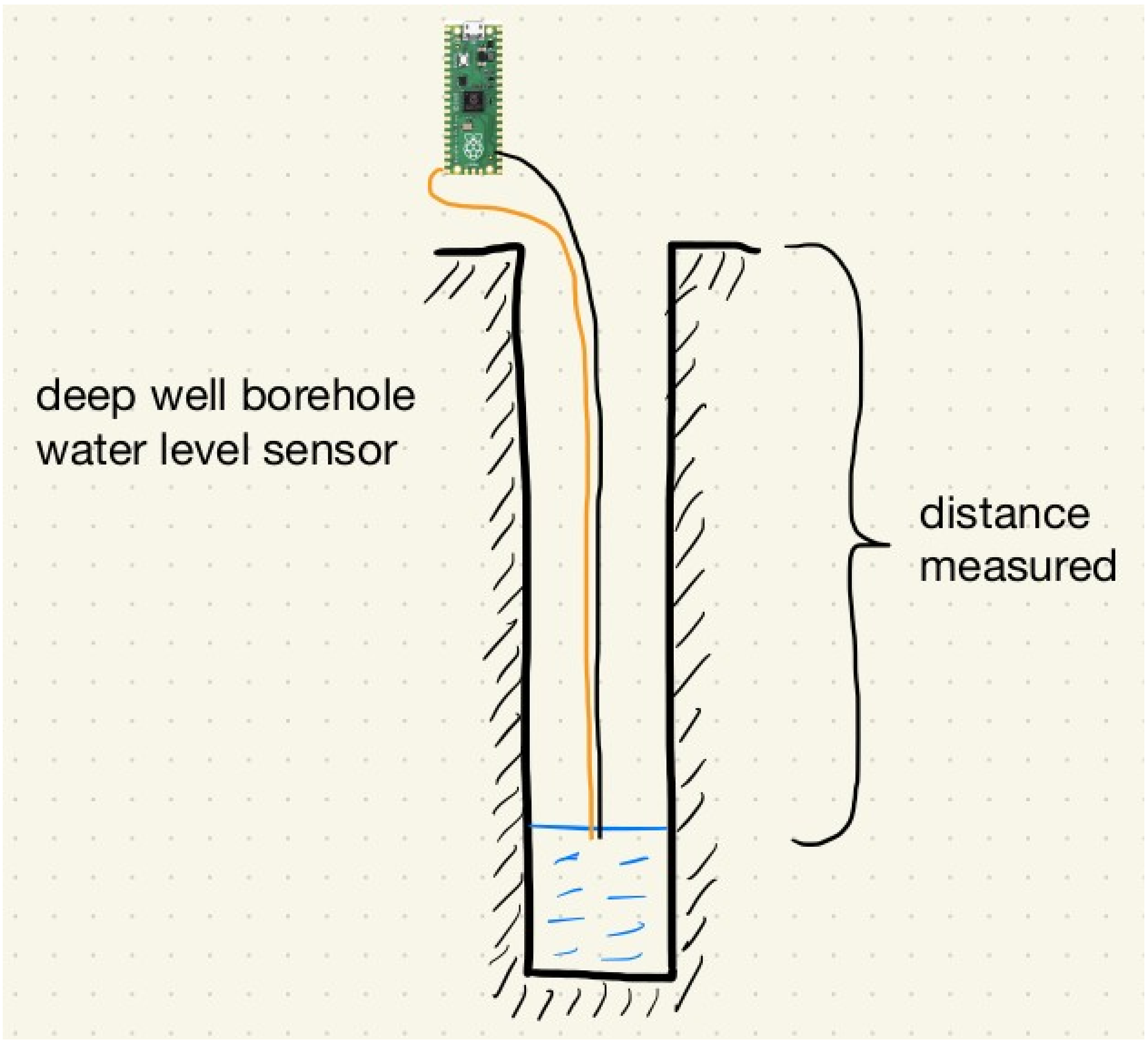
Exercise C.1) Please replace the push button by a glass of water like below and verify that the LED lights up if there is water between the wires or not.



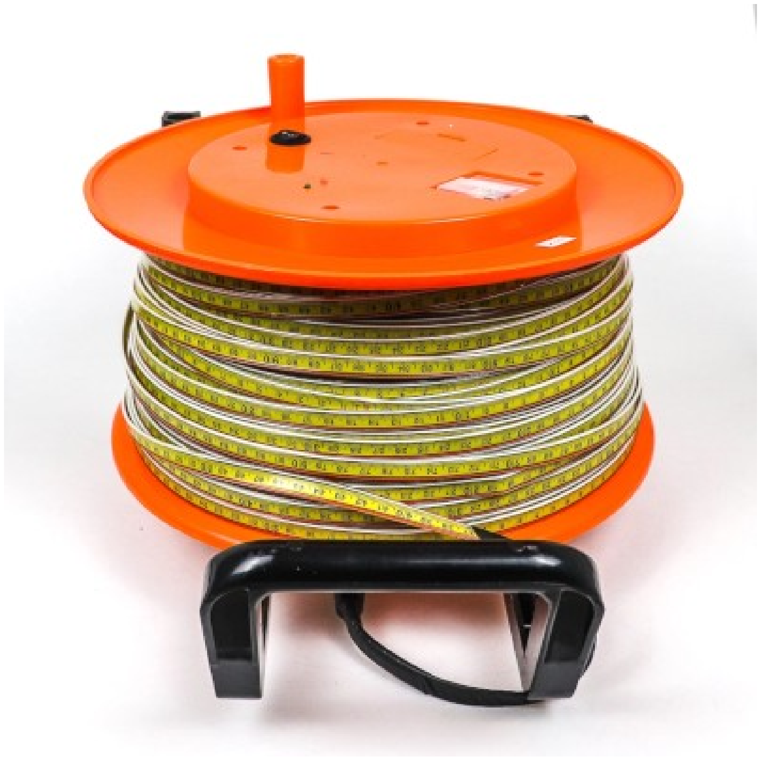


Question C.2) If the water is very clean you may have to submerge both wires about 2 cm, i.e. not only the tip for the LED to light up. Why is this?

An example of where to use a water level sensor is for measuring the water level in a borehole.



Some commercial sensors are based on this principle. Please look at the product below which you lower into the borehole and read the length from ground level to the tip down in the borehole using the scale.



Notice how the meter scale has a red and a white wire on the side. It uses the same principle to detect when the tip is below water level.

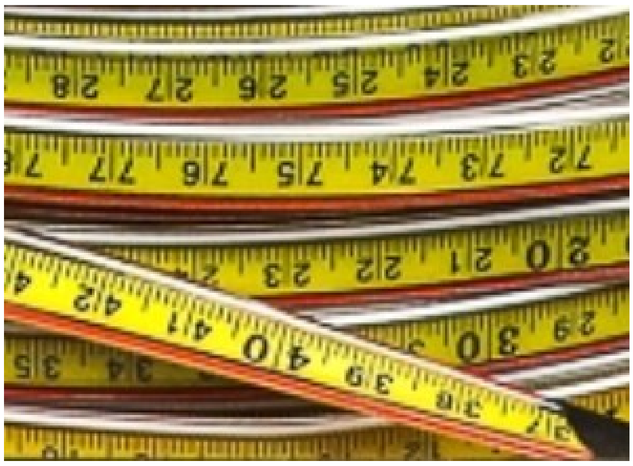
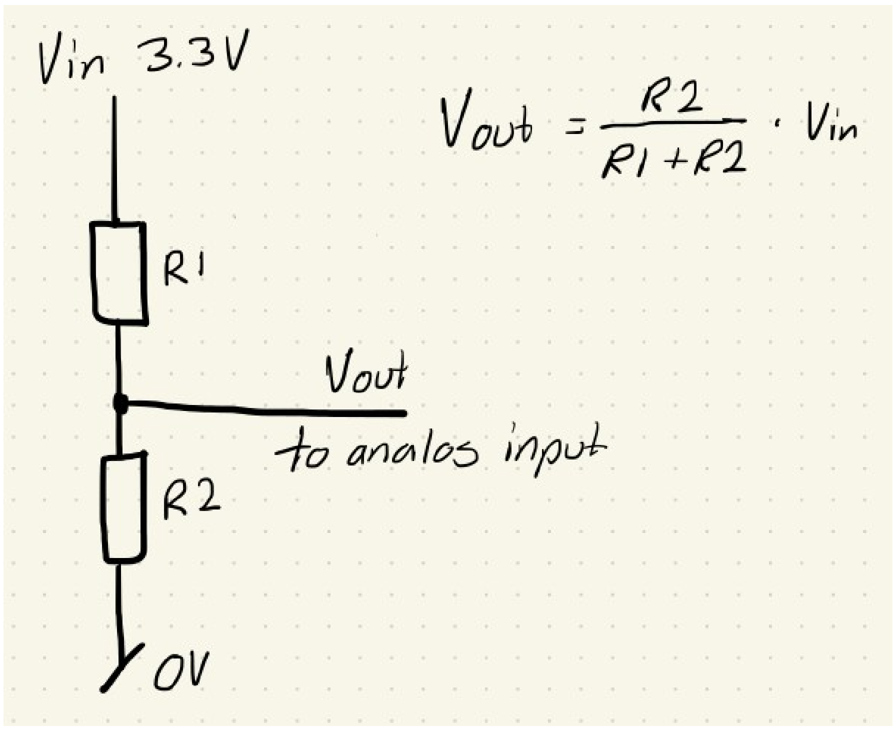


Image source: weqiequipment.com

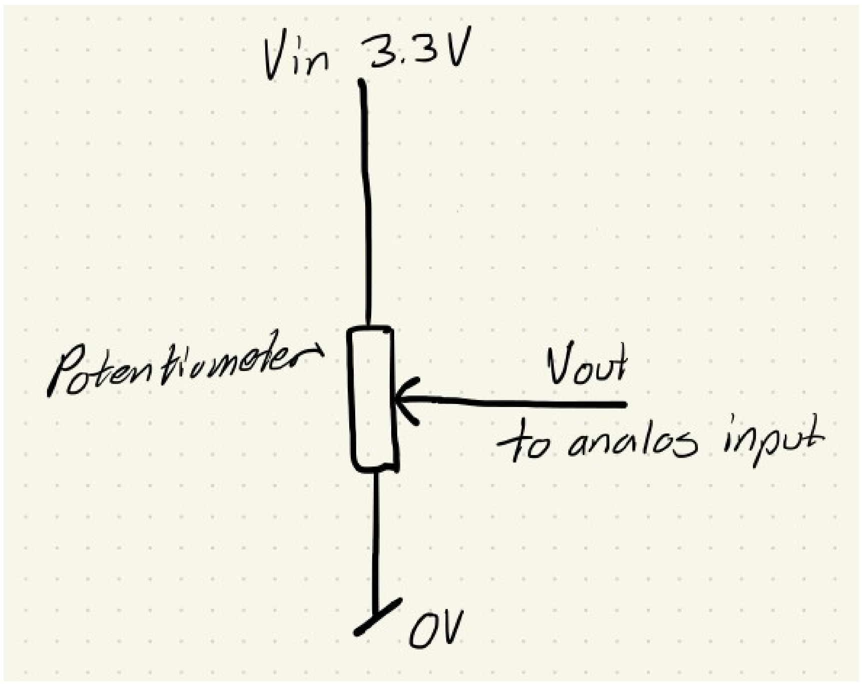
D) Potentiometer (analog input)
In this exercise we will connect a potentiomenter to the Pico as an analog input sensor and then in the Arduino sketch read the potentiometer position.

Please notice that for the Pico only the pins 26,27,28 can be used as analog inputs. In the course kit documentation you can find an overview of the Pico pins.

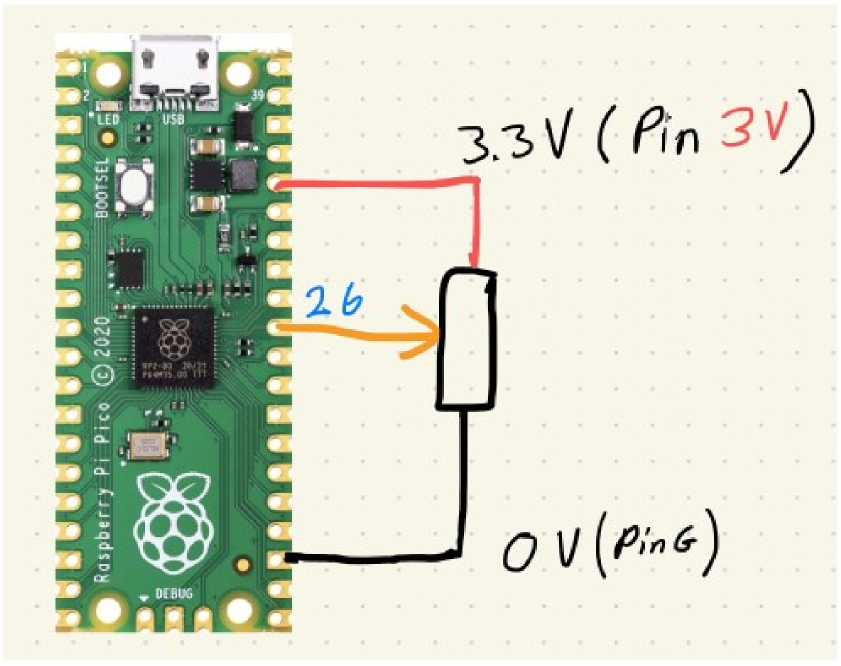
To read the potentiometer using an analog input on the Pico we must make a *voltage divider* between 3.3V and 0V as shown below.

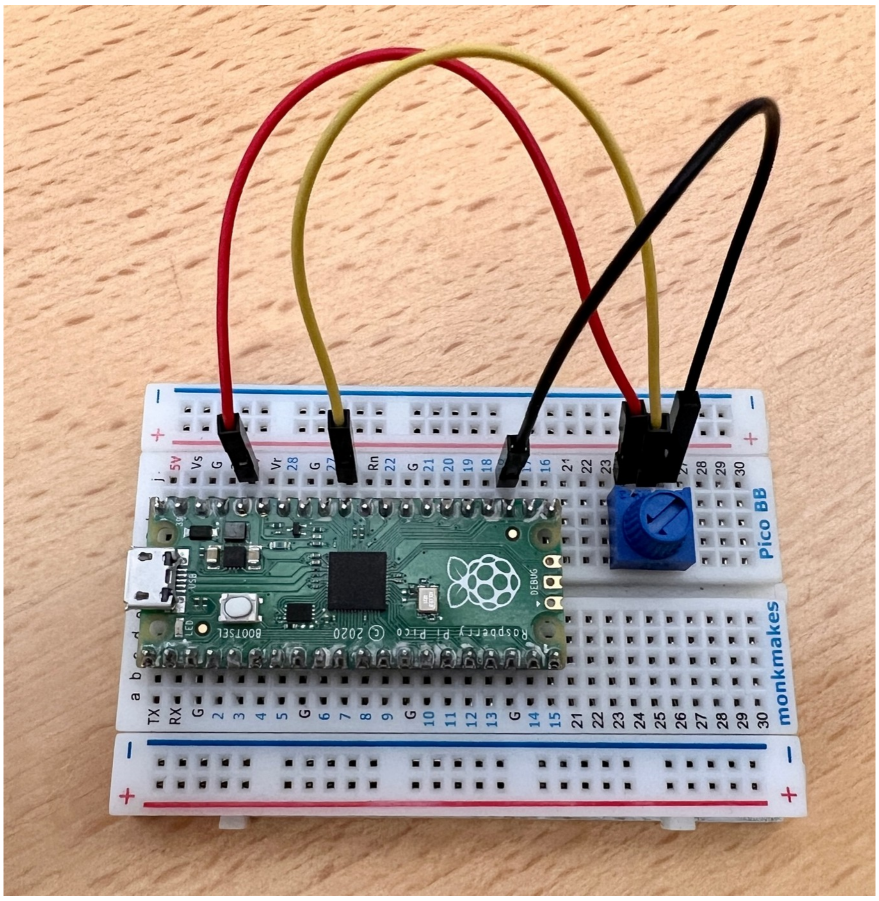


Instead of using the two resistors R1 and R2 we use a potentiometer where we by turning the potentiometer can vary R1 and R2.



Exercise D.1) Please connect the potentiometer to pin 26 on the Pico using the breadboard.





Exercise D.2) Please use the example program below to read data from the push button. A copy is available as `read_analog_input.ino` in the examples folder.

```
void setup() {
  // Setup serial port
  Serial.begin(115200);
}

void loop() {
  // Read Pin 26 (analog input A0)
  int value = analogRead(26);

  // Print the value to the serial port
  Serial.println(value);

  // Wait one second
  delay(1000);
}
```

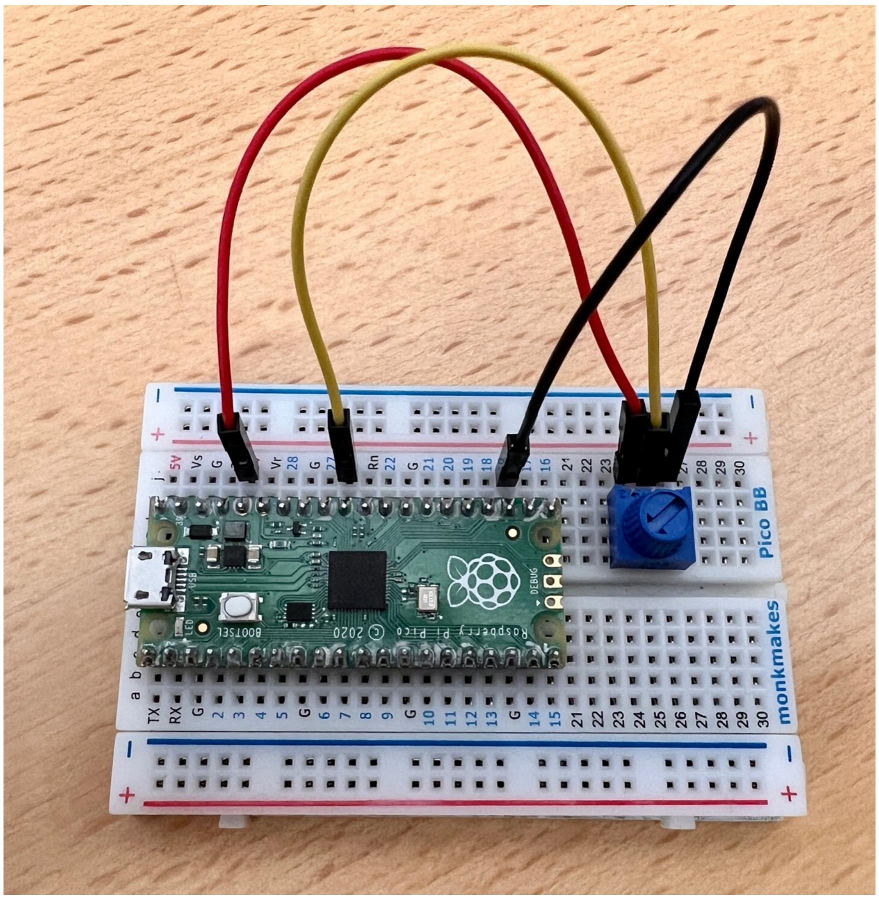
When you have uploaded the example program to the Pico, you have to open the Serial Montior (magnifying glass icon to the upper right of the Arduino Programming Software).

Question D.3) What are the minimum and maximum values that you see in the Serial Monitor when you turn the potentiometer knob?

Please explain in your own words what happens when you turn the potentiometer knob?

E) Computer mouse (analog input)
In this exercise we will use the potentiometer to create a computer mouse simulator.

We will use the circuit from exercise D with no changes.



Exercise E.1) Please use the example program below to simulate the movement of a computer mouse. A copy is available as `mouse_move.ino` in the examples folder.

```
// Import mouse driver
#include <Mouse.h>

void setup() {
  // Start mouse driver
  Mouse.begin();
}

void loop() {
  // If BOOTSEL button is pressed
  if (BOOTSEL) {
    // Read Pin 26 (analog input A0)
    int dx = analogRead(26);

    // Subtract half max value 1023
    dx = dx - 512;

    // Keep value within char limits
    dx = dx/4;

    // Move the mouse dx pixels
    Mouse.move(dx, 0, 0);

    // Wait until BOOTSEL button is released
    while (BOOTSEL) {
      delay(1);
    }
  }
}
```

Question E.2) Please explain in your own words what happens when you turn the potentiometer knob and press the BOOTSEL button?