CamJam EduKit Worksheet Five

Project Push button for physical input.

Description In this project, you will learn how to wire and code a push button to provide physical input.

Equipment Required

- The circuit built in CamJam EduKit Worksheet Two, plus the following:
  - The button from the EduKit.
  - 2 additional M/M Jumper wires.

Additional Parts

You will be adding a button to the LED circuit that you made in CamJam EduKit Worksheet Two. Here are the additional components. You may skip this section if you already know about these components.

Button

A button will complete a circuit when the button is pressed or 'closed'. What that means is that current will not flow across the button until it is pressed. When it is released, the circuit will be 'broken'.

Internal pull-up Resistor

The hidden internal pull-up resistor inside the Raspberry Pi Pico has a special purpose.

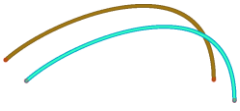
You are going to use one of the GPIO pins as an input pin, meaning that it will react to the outside world. If you tried to detect the state of one of the GPIO input pins without the pull-up resistor, it would randomly be either on or off, so you must have a way to ensure that it is on or off in a reliable way.

For this, you will use what is called a 'pull-up resistor'. When the button is not pressed, there will be an internal current flowing from the 3.3 volt power supply of the Pico to the input pin of the Pico. When read, the input pin will be seen as being 'on'.

When you press the button, another connection is made which will make the current 'flow to ground', which means that the input pin will be seen as being 'off'. This is how you will detect the button.

The MicroPython statement 'button = Button(12)' will activate the internal pull-up resistor on that pin.

Jumpers



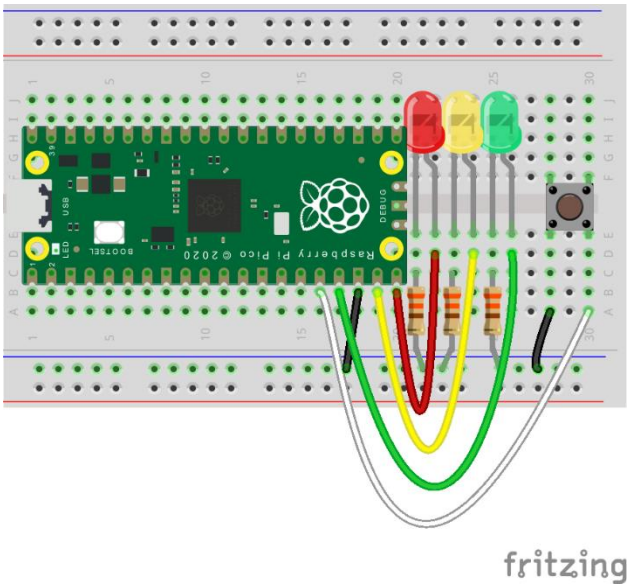
You will be adding two more Male/Male (M/M) jumpers, which have a pin on each end.

Jumper wires vary in colour and the ones that you have are unlikely to match the colours used in the diagrams. If you feel it would help, you could write on paper what 'actual' colour is used for a given colour in the diagram.

Building the Circuit

Exit Thonny and unplug the micro-USB cable from your computer. We shall add components to the breadboard to expand the circuit we build in Worksheet 2.

Add the button to the breadboard as per the diagram below. Push down hard to ensure that the pins are inserted fully into the breadboard. This may take a few tries.



Connect a new jumper wire (shown in white in the picture on the left) from the same column of the breadboard as GP12 to the right side of the button.

Connect the 'ground rail' to the other side of the button, shown here by the right-hand black wire.

How does the button circuit work?

When the button is not being pressed, a current will flow from the Pico's internal 3.3-volt power supply to the internal pull-up resistor, then to the GP12 pin of the Pico. All of this happens internally within the Pico.

When the button is pressed, the current takes another route. Flowing from the internal 3.3-volt power, it goes:

- through the internal pull-up resistor
- then out of pin GP12
- then through the white wire
- then through the button
- then through the black wire
- and back to the common ground (0v).

This "splits" enough of the current from the internal pull-up resistor to ground. Voltage on the white wire is much lower, and no longer high enough for the input GPIO pin to be seen as 'on'.

After adding the components, double check that the circuit is correct and matches the diagram above. Using the wrong wiring can damage the Pico and other components.

Code

Plug the small end of the micro-USB cable in to the Pico and connect the other end to your computer. Start Thonny and create a new file. Type in the following code:

```
# CamJam EduKit 1 - Basics
# Worksheet 5 - Button

# Import Libraries
import time  # Provides time related commands
from picozero import Button # The picozero Button function

# Set pin 12 as a button input
button = Button(12)

print("-------------")
print("Button + GPIO")
print("-------------")

# The commands indented after this 'while' will be repeated
# forever or until 'Ctrl-C' is pressed.
while True:
    # If the button is pressed, button.is_pressed will be 'true'
    if button.is_pressed:
        print("Button pressed")
        time.sleep(1) # Sleep for 1 second
    else:
        print("Waiting for you to press the button")

    time.sleep(0.5) # Sleep for 0.5 seconds
```

Save the file on to the Pico as `5-button.py`.

Explanation of the Code

```
# Set pin 12 as a button input
button = Button(12)
```

The code above tells picozero that there is a button attached to pin GP12. It also activates the Pico's internal pull-up resistor. When the pin detects a voltage nearing 3.3v, it is read as 'true', or 'on'. If the voltage nears 0v, then it is 'false', or 'off'.

```
while True:
```

The commands in the block indented after this 'while' will be repeated forever, or until 'CTRL-C' is pressed.

```
if button.is_pressed:
```

The 'if' statement gets the status of the button on pin 12. The value is either:

- 'True', meaning the button is being pressed, or
- 'False', meaning the button is not being pressed.

If the value is true, meaning the button is being pressed, the block of code indented below the '`if`' statement will be executed.

```
print("Button pressed")
```

```
    time.sleep(1)
else:
```

If the value of the input button is ‘false’, then different code will be run. The block of code that is indented after the ‘else’ statement will be executed.

```
print("Waiting for you to press the button")
time.sleep(0.5)  # Sleep for 0.5 seconds
```

There is a half-second wait between looking at the state of the button to allow any messages to appear on the screen for long enough to read.

Running the Code

Before running the code, double check that the new button is added correctly as a Button in the code. Mistakenly setting it to ‘Buzzer’ or ‘LED’ can damage the Pico.

Click the green Run icon on the top menu bar.

The program will wait for the button to be pressed and report the status to the screen. This will continue until you press ‘Ctrl-C’ to exit Python.

Note: Do not disassemble this circuit as it will be used in the following worksheets.

Concepts

Let us go through some of the important concepts before looking at the code:

- A ‘variable’ is a name that contains a value. That value can be changed within your code at any time. It is often easier to use a variable to contain a number because it is easier to remember the name.
- ‘Constants’ are like variables, except that they cannot be changed within your code. In Python, there are two special ‘constants’ called ‘True’ and ‘False’. They have the values ‘1’ and ‘0’.
- Indentation is very important in Python. It is used to group commands together after certain other commands, like ‘if’ and ‘while’. Everything at the same indentation distance after these commands will be grouped together as a ‘code block’.