

CamJam EduKit - Worksheet Two

Project Controlling LEDs with MicroPython

Description In this project, you will learn how to connect and control LEDs (Light Emitting Diodes) with the Raspberry Pi Pico.

Equipment Required

For this worksheet, you will require:

- From CamJam EduKit 1:
 - 400 Point Breadboard
 - 1 x Red LED
 - 1 x Yellow LED
 - 1 x Green LED
 - 3 x 330Ω Resistors
 - 4 x M/M jumper wires
- A Raspberry Pi Pico of any type.

On-board LED

The Raspberry Pi Pico has an on-board LED. You can control this LED without any external components.

Code

Follow the instructions in Worksheet One to open the Thonny editor. Create a new file by selecting the File menu item 'New'. Type in the following:

```
# CamJam EduKit 1 - Basic
# Worksheet 2 - Onboard LEDs for Raspberry Pi Pico

# Import Libraries
import time # A collection of time related commands
from picozero import pico_led # Onboard LED function from picozero

print("LED on")
pico_led.on()

print("Wait for one second")
time.sleep(1)

print("LED off")
pico_led.off()
```

Once you have typed all the code and checked it, save the file on the Raspberry Pi Pico and call it `2-led-onboard-pico.py`.

Running the Code

🟢 Click the green Run icon on the top menu bar. You should see the Raspberry Pi Pico's on-board LED light for one second, then turn off again.

Building your first circuit

In this first circuit, you will be connecting three LEDs to the Raspberry Pi Pico and using MicroPython to turn them on and off.

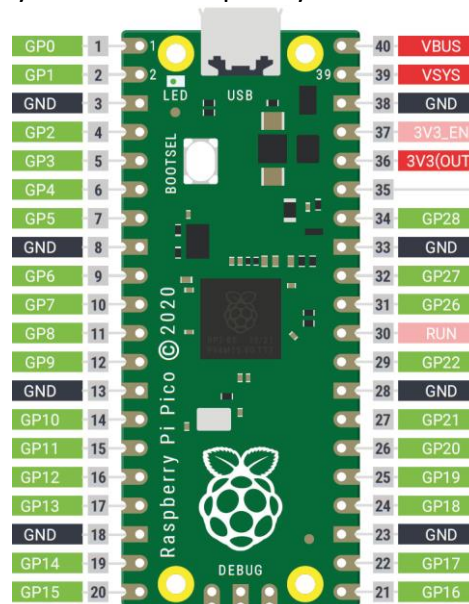
It is important that you read the following sections, as you need to understand the Raspberry Pi Pico GPIO pins, how the holes in the breadboard are connected together, and which leg of the LED is which.

Before you build the circuit, let's look at the parts you are going to use.

GPIO pins for the Pico

GPIO stands for **General Purpose Input Output**. With the GPIO pins, the Raspberry Pi Pico can control and monitor the outside world when connected to electronic circuits. The Raspberry Pi Pico can control LEDs, turning them on or off, or motors, or many other things. It is also able to detect whether a switch has been pressed, or what the temperature is, or whether there is light. With the CamJam EduKit, you will eventually learn to control LEDs and a buzzer and detect when a button has been pressed.

The diagram below shows the pin layout for the Raspberry Pi Pico.



You will notice that the bottom left pin is 'GP15' on the diagram – you will refer to this in MicroPython as GPIO pin '15'.

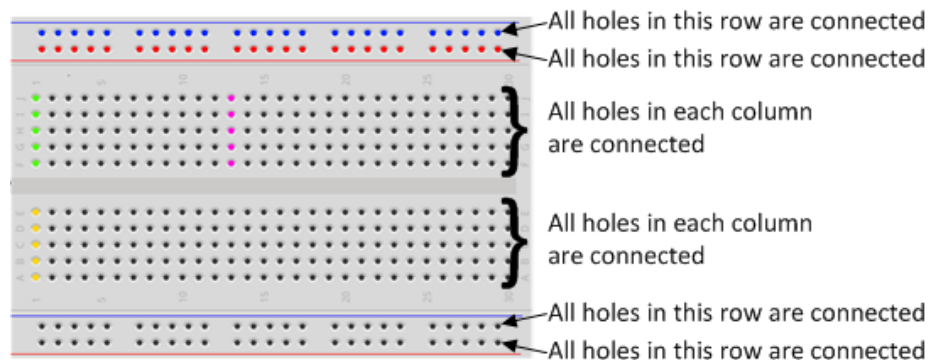
Some pins have different functions. There are pins that provide power at 5 volts and 3.3 volts, ground pins (0 volts), input/output pins and some pins that interface to external circuits in more complex ways. You are just going to use the basic GPIO and Ground (GND) pins in these worksheets.

You will also notice that the physical pin numbers do not match the GPIO pin numbers – we will always use the GPIO pin numbers.

The Breadboard

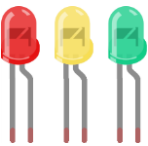
The breadboard is a way of connecting electronic components to each other without having to solder them together. They are often used to test a circuit design before creating a Printed Circuit Board (PCB).

The holes on the breadboard are connected in a pattern. On the breadboard in the CamJam EduKit, the holes in the top row are all connected together and are marked with blue dots. The second row of holes, marked with red dots, are the same – they are connected together. The same goes for the two rows of holes at the bottom of the breadboard.



In the middle, the columns of holes are connected to each other vertically, with a break in the middle. So, for example, all the green holes marked are connected, but they are not connected to the yellow holes, nor the purple ones. Therefore, any wire you poke into the green holes will be connected to other wires poked into other green holes.

The LEDs



Three LEDs are supplied in the EduKit – one red, one yellow, and one green. LED stands for Light Emitting Diode. An LED glows when electricity is passed through it.

When you pick up the LED, you will notice that one leg is longer than the other. The longer leg (known as the 'anode') should always be connected to the positive side of the power supply. The shorter leg (known as the 'cathode') should always be connected to the negative side of the power supply, known as 'ground'.

LEDs will only work if power is supplied the correct way round (i.e. if the 'polarity' is correct). You will not damage the LEDs if you connect them the wrong way round – they will just not light. If you find that they do not light up in your circuit, it may be because they have been connected the wrong way round.

The Resistors



Resistors are a way of limiting the amount of electricity going through a circuit; specifically, they limit the amount of 'current' that is allowed to flow. The measure of resistance is called the Ohm (Ω), and the larger the resistance, the more it limits the current. The value of a resistor is marked with coloured bands along the length of the resistor body.

The EduKit is supplied with two sets of resistors. There are three 330Ω resistors. There is also one $4.7k\Omega$ (or 4700Ω) resistor. The Pico has a $4.7k\Omega$ resistor built in, so you will not be using the one supplied in the EduKit.

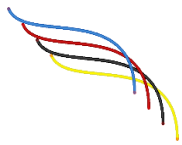
In the LED circuit, you will be using the three 330Ω resistors. You can identify the 330Ω resistors by the colour bands along the body. The colour coding will depend on how many bands are on the resistors supplied:

- If there are four colour bands, they will be Orange, Orange, Brown, and then Gold.
- If there are five bands, then the colours will be Orange, Orange, Black, Black, Brown.

You must use resistors to connect LEDs up to the GPIO pins of the Raspberry Pi Pico. The Raspberry Pi Pico can only supply a small current. The LEDs will want to draw more and, if allowed to, they will burn out the Raspberry Pi Pico. Therefore, putting the resistors in the circuit will ensure that only this small current will flow, and the Raspberry Pi Pico will not be damaged.

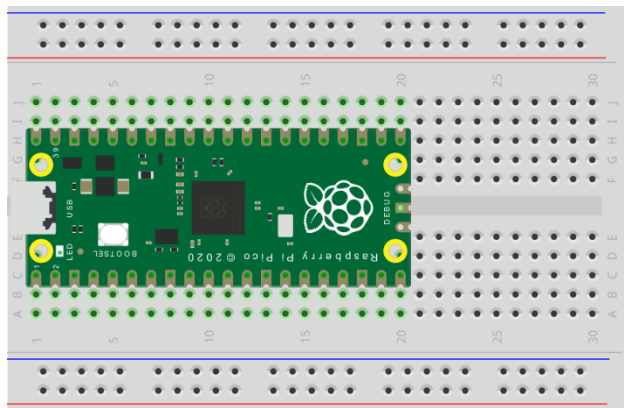
It does not matter which way round you use the resistors. Current flows in both ways through them.

The Jumper Wires



Jumper wires are used on breadboards to 'jump' from one connection to another. The ones you will be using in this worksheet's circuit have the same connector on each end. The end with the 'pin' will go into the breadboard and is known as the 'male' end. Jumpers with 'male' connectors at both ends are M/M jumper wires. Wires with male and female connectors are M/F jumpers. The jumper wires vary in colour and are unlikely to match the colours used in the diagrams.

Pico breadboard installation



fritzing

Unplug the micro-USB cable from the Pico. Sit the Pico into the breadboard so that it straddles the middle gap. The bottom-left pin, GP0, should be in the breadboard row marked with a 1.

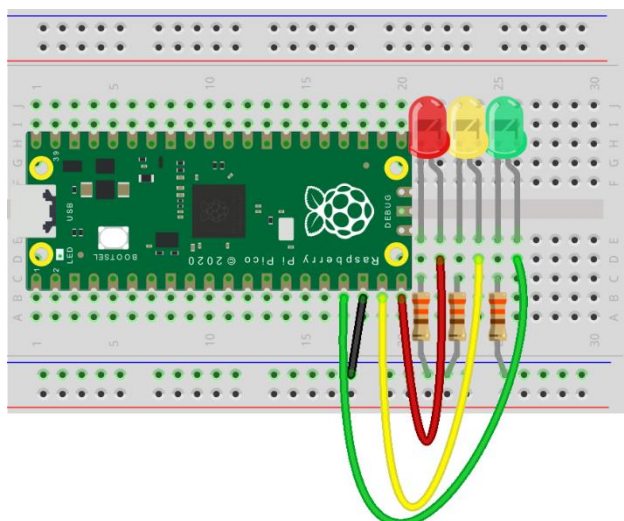
Before pushing the Raspberry Pi Pico down, make sure the header pins are all properly positioned – if you bend a pin, it can be difficult to straighten it again without it breaking.

Gently push the Pico a little bit on one side, then the other side. Keep the board horizontal during the process. Repeatedly press alternate sides until the Pico is in place, so that the plastic parts of the header pins are touching

the breadboard. If the pins are misaligned, remove the Pico, straighten any bent pins and try again.

Building the Circuit

Unplug the USB cable from your computer.



fritzing

You will be using one of the 'ground' (GND) pins to act like the 'negative' or 0 volt end of a battery.

The 'positive' ends of the power supply will be provided by three of the other GPIO pins, one for each of the three LEDs. You will be using the pins GP15, GP14 and GP13 for the Red, Yellow and Green LEDs respectively.

When the pins are 'taken high', which means that they output 3.3 volts, the LEDs will light.

Look at the circuit diagram on the left.

The power for each LED will be provided by the Pico, from pins GP15, GP14 and GP13. You can control them from MicroPython - you can make the GPIO pins supply either 0 volts (off) or 3.3 volts (on).

There are in fact three separate circuits in the diagram. Each one consists of the power supply (the Raspberry Pi Pico), an LED that lights when the power is applied, and a resistor to limit the current that can flow through the circuit.

Each circuit uses a 'common ground rail'. In other words, you will be connecting all the circuits to the same 'ground' pin (0 volts, marked as GND) of the Pico. You are going to use the second row up from the bottom of the breadboard to provide GND/0 volts. Remember that the holes on the two top and two bottom rows are connected together. Using a M-M jumper wire, connect the third pin from the right (GND) of the Pico to the second row up of the breadboard. This is shown in the diagram as a black wire.

Next, push the legs of the three LEDs into the breadboard, with the long leg on the right as shown in the circuit diagram.

Then, connect the three 330Ω resistors between the 'common ground rail' and the left legs of the three LEDs. You will need to bend the legs of each of the resistors to fit, but make sure that the wires of each leg do not touch one another.

Complete the circuit by connecting pins GP15, GP14, and GP13 to the right-hand legs of the red, yellow, and green LEDs. These are shown in the diagram with red, yellow, and green wires, but you may be using jumper wires of different colours.

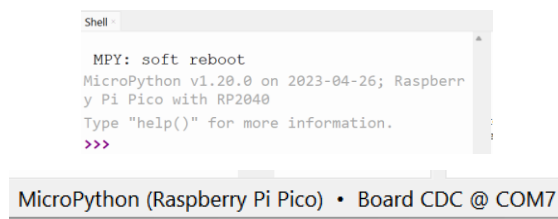
When you have finished adding the components, always double check that your circuit is correct. Wrong wiring can damage the Pico and other components.

You are now ready to write some code to switch the LEDs on.

Code

Plug the small end of the micro-USB cable into the Pico and connect the other end to the computer.

Start Thonny and check that there is a MicroPython startup message in the Shell area. Check the bottom right corner to see if the correct Python interpreter (MicroPython (Raspberry Pi Pico)) and the correct COM port are displayed.



```
Shell
MPY: soft reboot
MicroPython v1.20.0 on 2023-04-26; Raspber
y Pi Pico with RP2040
Type "help()" for more information.
>>>
```

MicroPython (Raspberry Pi Pico) • Board CDC @ COM7

Create a new file by going to the File menu item and clicking 'New'. Type in the following code:

```
# CamJam EduKit 1 - Basic
# Worksheet 2 - LEDs

# Import Libraries
import time # A collection of time related commands
from picozero import LED # The LED function from picozero

# Set pins 15, 14 and 13 to be LEDs
red = LED(15)
yellow = LED(14)
green = LED(13)

print("LEDs on")
red.on()
```

```
yellow.on()
green.on()

print("Wait for one second")
time.sleep(1)

print("LEDs off")
red.off()
yellow.off()
green.off()
```

Once you have typed in all the code and checked it, save the file to the Pico and call it `2-LED.py`

How does the code work?

Let us go through the code a section at a time and explain what is happening:

```
import time # A collection of time related commands
from picozero import LED # The LED function from picozero
```

The first and second lines above tell the Python interpreter (the thing that runs the Python code) that it will be using 'libraries' that will tell it how to work with time-related commands, like sleep, and the Pico's GPIO pins. Adding a 'library' to your code is a bit like adding a new channel to your TV so that you can watch something different. In this case, we are telling Python to import a library that gives you access to various time-related functions. On the second line, we are telling Python to only import the part of the picozero library that handles LEDs.

```
# Set pins 15, 14 and 13 to be LEDs
red = LED(15)
yellow = LED(14)
green = LED(13)
```

These three lines are telling the Python interpreter that pins 15, 14 and 13 are going to be used for red, yellow and green LEDs respectively. The red LED is attached to pin 15, the yellow to pin 14 and the green to pin 13.

```
print("LEDs on")
```

This line prints some information to the Shell window, so you can see what the program is trying to do.

```
red.on()
yellow.on()
green.on()
```

These three lines turn the three LED pins on; the three pins are made to provide power of 3.3 volts to the three GPIO pins.

```
print("Wait for one second")
time.sleep(1)
```

These two lines print a message to the Shell window, followed by instructions to 'sleep' for one second. The 'sleep' function is part of the 'time' library loaded on line 5 of the program. The number in brackets is the number of seconds that Python should 'sleep' and do nothing. You do not have to use a whole number – you can use decimal numbers (like 1.5).

```
print("LEDs off")  
red.off()  
yellow.off()  
green.off()
```

This prints out a message once again and then turns all three LEDs off.

Running the Code

You are now ready to run the code. Click the green Run icon. You should see the LEDs light for one second, then turn off again.

If you find that the code does not run correctly, there may be an error in what you have typed. You should check and re-edit the code, save the file again, and re-run it.

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