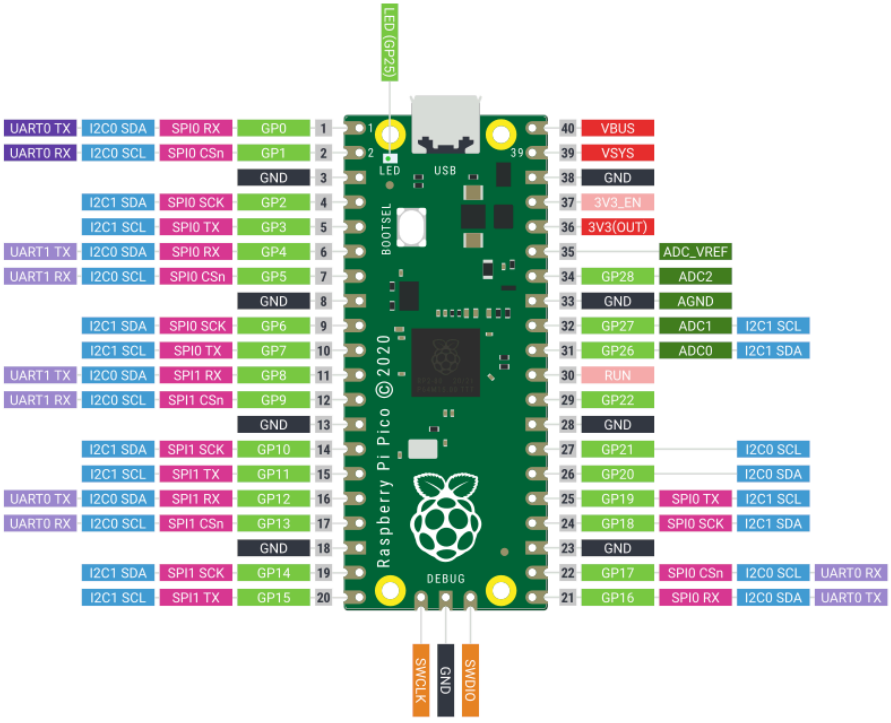
MicroPython is a full implementation of the Python 3 programming language that runs directly on embedded hardware like Raspberry Pi Pico and Raspberry Pi Pico W. To start working with a Raspberry Pi Pico board follow the *Drag-and-Drop MicroPython* manual:

www.raspberrypi.com/documentation/microcontrollers/micropython.html

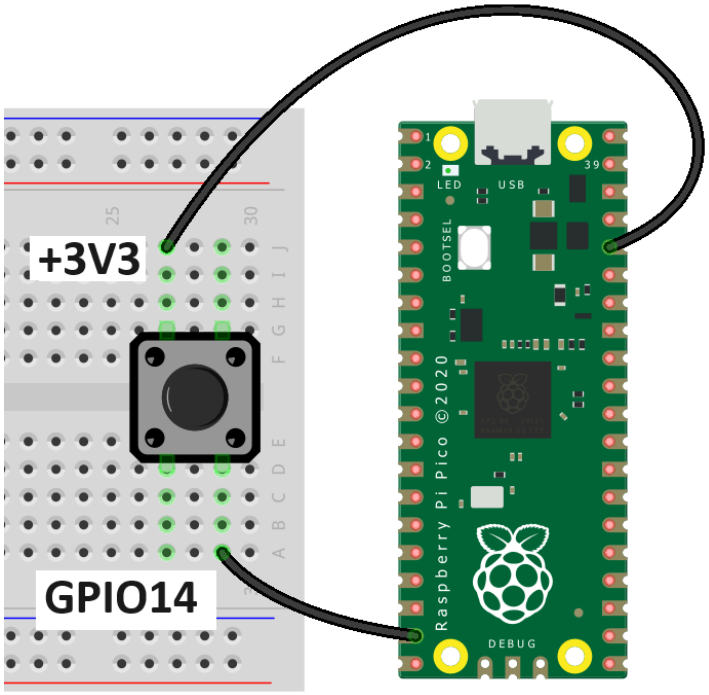


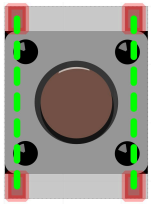
Task 1. Create a MicroPython script to blink the built-in LED every 500[ms]. The LED is connected to GPIO 25.

```
import machine
import utime as t

led_builtin = machine.Pin(25,machine.Pin.OUT)
i = 0

while True:
      led_builtin.toggle()
      t.sleep(0.5)
```

```
    i+=1
    print(f'Current interation is {i}')
```

Task 2. Connect a tact switch to the Raspberry Pi Pico board. Use GPIO 14 (GP14). The other switch terminal should be connected to the 3V3 pin. Create a MicroPython script to send a *"Button pressed"* message to the computer after the button has been pressed.



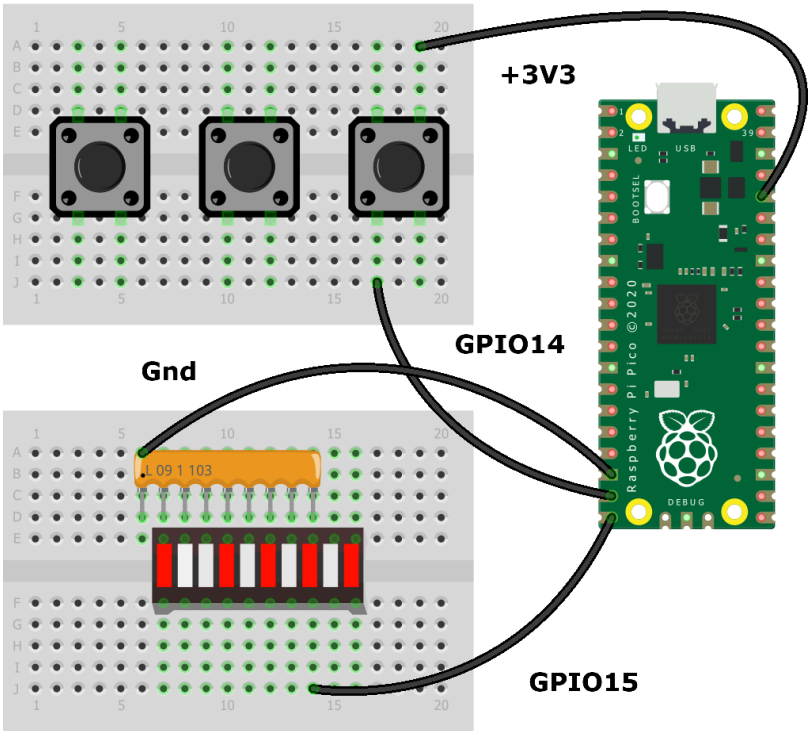


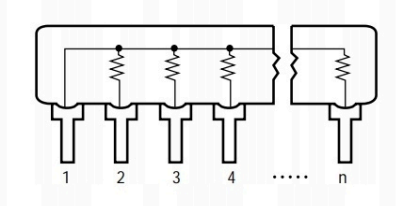
```
from machine import Pin
import utime

button = Pin(14,Pin.IN,Pin.PULL_DOWN)

while True:
    if button.value() == 1:
        print("Button pressed")
        utime.sleep(0.5)
```

Allow the user to decide about the message.

```
from machine import Pin
from utime import sleep

button = Pin(14,Pin.IN,Pin.PULL_DOWN)
message = input("Enter the message: ")
print('This is the message: ' + message)

while True:
    if button.value() == 1:
        print(message)
        sleep(0.5)
```

Task 3. Connect the following circuit. A button connected to the GPIO14 controls the built-in LED. If this button is pressed, the LED should emit light. Otherwise, it should be turned off. Use a 120R current-limiting resistor.






```
from machine import Pin
import utime

button = Pin(14,Pin.IN,Pin.PULL_DOWN)
led = Pin(15,Pin.OUT)
led.value(1)

while True:
    if button.value() == 1:
        led.value(0)
    else:
        led.value(1)
    utime.sleep(0.1)
```

Task 4. Use the Task 3 circuit. Create a MicroPython script to produce 4 LED blinks after the button has been pressed.

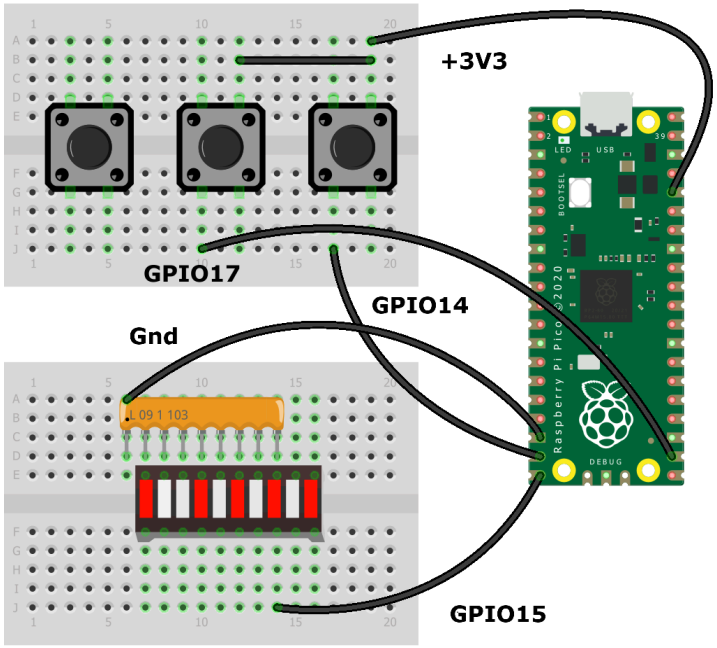
```
from machine import Pin
from utime import sleep

button = Pin(14,Pin.IN,Pin.PULL_DOWN)
led = Pin(15,Pin.OUT)

def led_blink(pin,times):
    for i in range(times):
        pin.toggle()
        sleep(0.5)
        pin.toggle()
        sleep(0.5)

led.value(0)
```

```
while True:
    if button.value() == 1:
        led_blink(led,5)
    sleep(0.1)
```

Task 5. Use the Task 3 circuit. Create a MicroPython script to implement the functionality of a monostable light switch.

```
from machine import Pin
from utime import sleep

button = Pin(14,Pin.IN,Pin.PULL_DOWN)
light = Pin(15,Pin.OUT)

def button_clicked(pin):
    if pin.value() == 1:
        sleep(0.1)
        while pin.value() == 1:
            pass
        return True
    else:
        return False

light.value(0)

while True:
    if button_clicked(button):
        light.toggle()
```

Task 6. Connect the following circuit.



Use 2 buttons (GPIO17, GPIO14) to control the LED connected to GPIO15 (GP15). The built-in LED (GPIO25) should blink every second.

```
from machine import Pin
from utime import sleep
import _thread

button_L = Pin(17,Pin.IN,Pin.PULL_DOWN)
button_R = Pin(14,Pin.IN,Pin.PULL_DOWN)
buildin_led = Pin(25,Pin.OUT)
external_led = Pin(15,Pin.OUT)
def led_blink():
    while True:
        buildin_led.toggle()
        sleep(0.5)
_thread.start_new_thread(led_blink,())

def button_clicked(pin):
```

```
    if pin.value() == 1:
        sleep(0.1)
        while pin.value() == 1:
            pass
        return True
    else:
        return False

while True:
    if button_clicked(button_R):
        external_led.value(1)
        print("R click")
    if button_clicked(button_L):
        external_led.value(0)
        print("L click")
```

Save this project in the memory of a Raspberry Pi Pico board as `main.py`. Disconnect and then reconnect the USB cable. Observe results.

Task 7. Communication Between Threads.

```
from utime import sleep
import _thread

def core0():
    global core1_takes_control
    counter = 0
    while True:
        for _ in range(3):
            print(f' Counter value = {counter}')
            counter += 1
            sleep(1)

        core1_takes_control = True

        print("Core 0 is waiting")
        while core1_takes_control:
```

```
            pass

def core1():
    global core1_takes_control
    counter = 100
    while True:
        print("Core 1 is waiting")
        while not core1_takes_control:
            pass

        for _ in range(3):
            print(f'Counter value = {counter}')
            counter += 10
            sleep(2)

        core1_takes_control = False


core1_takes_control = False

second_thread = _thread.start_new_thread(core1,())
core0()
```

Task 8. Resource sharing.

Comment `lock.acquire()` and `lock.release()`.

```
from utime import sleep
import _thread

def core0():
    global lock
    while True:
        lock.acquire()
        print('C')
        sleep(0.5)
        print('O')
        sleep(0.5)
```

```
            print('R')
            sleep(0.5)
            print('E')
            sleep(0.5)
            print('0')
            sleep(0.5)
            lock.release()

    def core1():
        global lock
        while True:
            lock.acquire()
            print('c')
            sleep(0.5)
            print('o')
            sleep(0.5)
            print('r')
            sleep(0.5)
            print('e')
            sleep(0.5)
            print('1')
            sleep(0.5)
            lock.release()

    lock = _thread.allocate_lock()

    _thread.start_new_thread(core1, ())
    core0()
```

Task 9. Polling the lock.

`lock.acquire(True,1)`

Parameters:

a. *blocking* (optional):
   - `True` (default) - thread waits until the lock becomes available.
   - `False` - thread does not wait; it acquires the lock if it's free, otherwise it moves on.

b. *timeout* (optional) - maximum time (in seconds) to wait for the lock.
    - -1 (default) - waits indefinitely.

Returns:

- `True` - lock was successfully acquired.
- `False` - lock was not acquired (when blocking=`False` or timeout occurs).

```
from utime import sleep
import _thread

def core0():
    global lock
    while True:
        wait_counter = 0
        while not lock.acquire(False):
#         while not lock.acquire(True,1):
            wait_counter += 1
             print(f'Core0 counted to {wait_counter} while
waiting')

        print('C')
        sleep(0.5)
        print('O')
        sleep(0.5)
        print('R')
        sleep(0.5)
        print('E')
        sleep(0.5)
        print('0')
        sleep(0.5)
        lock.release()

def core1():
    global lock
    while True:
```

```
        lock.acquire()
        print('c')
        sleep(0.5)
        print('o')
        sleep(0.5)
        print('r')
        sleep(0.5)
        print('e')
        sleep(0.5)
        print('1')
        sleep(0.5)
        lock.release()

lock = _thread.allocate_lock()

_thread.start_new_thread(core1, ())
core0()
```

Task 10. (*own work*) Create a MicroPython script to send how many times a selected button has been clicked. Use another switch to reset the counter. Present the counter value on the Node-RED-based interface.

Task 11. (*own work*) Built a device prototype to measure the user's reaction time. Present the current result and the best score on the Node-RED-based interface.

Presentation and submission (email) of the solution afterwards is mandatory.

For those interested:

1. MicroPython web page:
   micropython.org/download/rp2-pico/

2. Python programming tutorial:
   www.programiz.com/python-programming/first-program

3. Getting started with Raspberry Pi Pico:
   projects.raspberrypi.org/en/projects/getting-started-with-the-pico/0