Infrared Remote Control

What is Infrared?

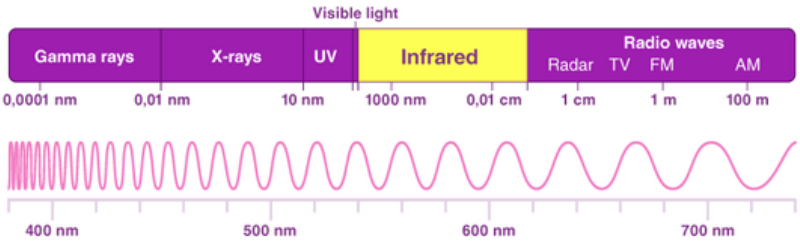
- Infrared is electromagnetic radiation (EMR) with wavelengths longer than those of visible light and shorter than radio waves.
- It is commonly used by older TV remote controls

Infrared Remote Control

- These remotes emit an infrared signal for as long as the button is held down
- Unlike radio or bluetooth. The remotes need direct line of sight to the receiver, anything in the way will block the signal.
- They have a limited range, up to about 8 Meters.

Infrared Receiver

- These are used to by our pi picos to accept the infrared signal that we will then decode.
- They have 3 pins **Signal**, **Ground** and **3.3/5V power**
- They are directional, the black side must face the remote.



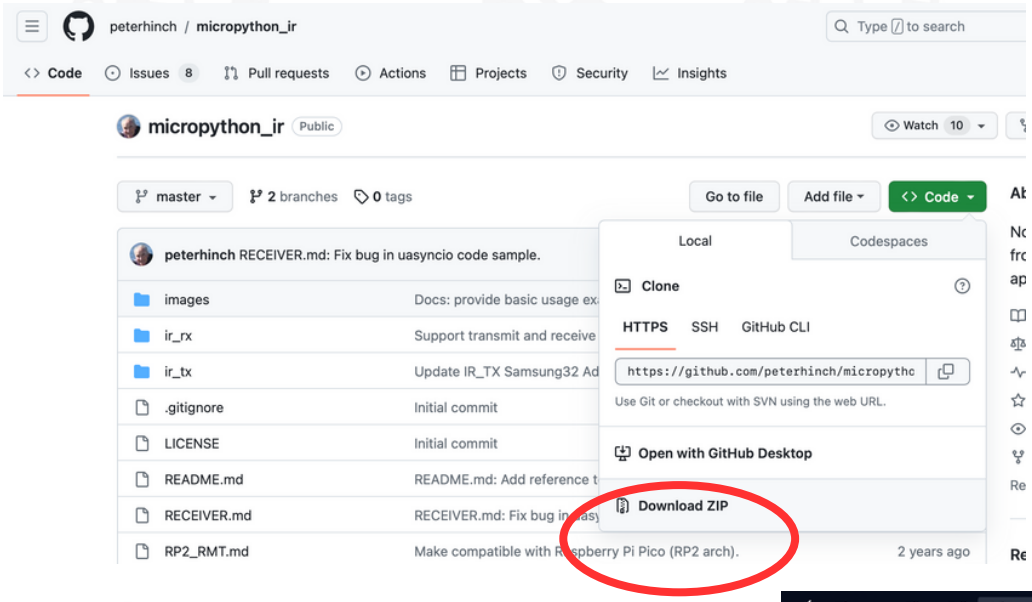
Install the micropython_ir module

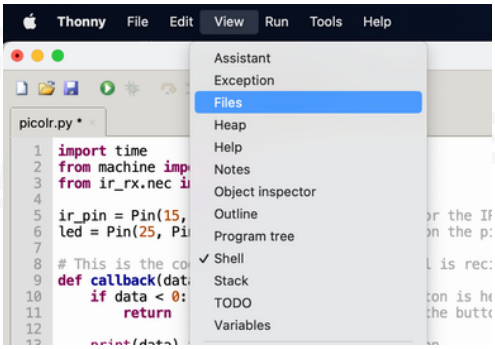
- To get our IR receivers working we need to install a new module to the pi pico. It can bee downloaded from this github page:

https://github.com/peterhinch/micropython_ir

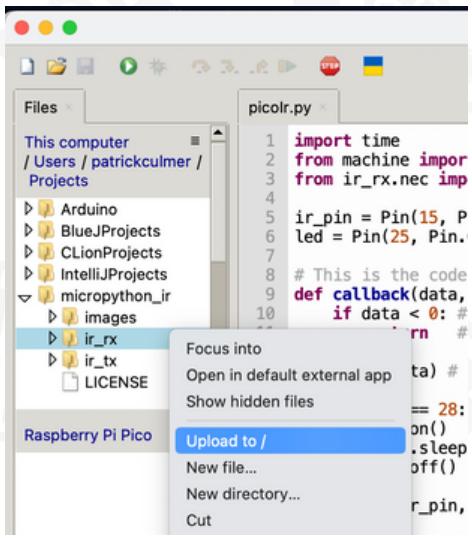
This link will be posted to #coding-general on the Hack Sussex discord.

(If you know how to clone the repo you don't need my help)

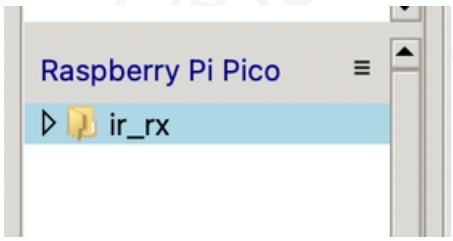




- Now we have the module downloaded to our computer we can upload it to the pi pico
- On Thonny, click on **View** in the toolbar and then **Files.**
- On the toolbar on the left, navigate to the **mictropython_ir** directory you just downloaded.
- Right click on the **ir_rx** folder and select **Upload to /**

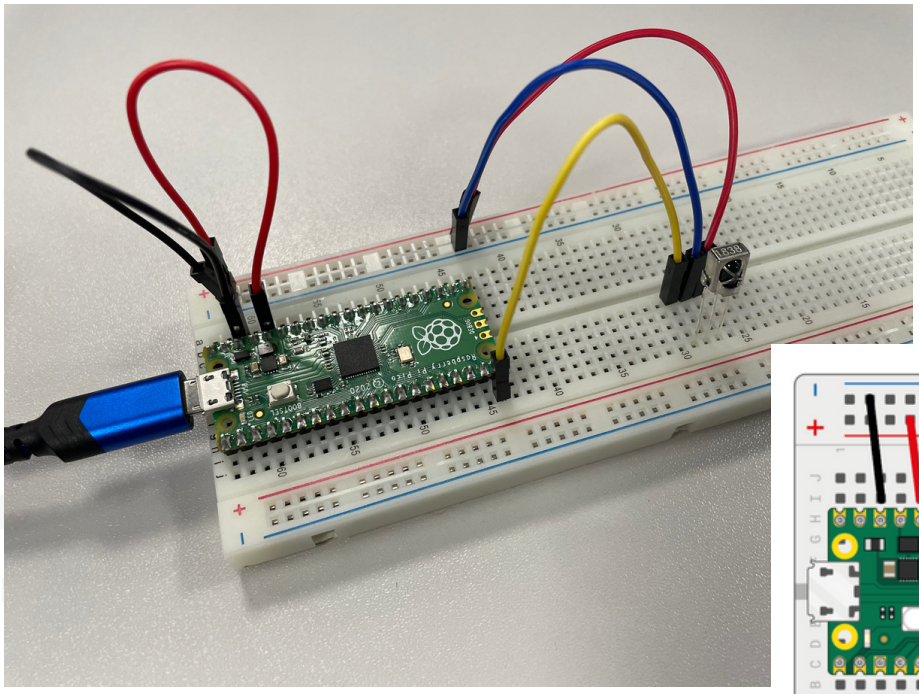


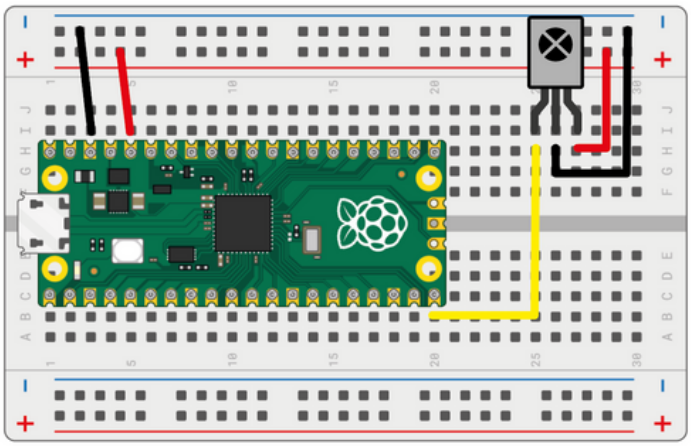
Your pico folder should look like this



Wire up the receiver

- Connect the left **Signal** pin, to **pin 15** on the pico.
- Connect the middle **Ground** pin to **GND** on the pico.
- Connect the right **Power** pin to **3V3(out)** on the pico





Finally, we can code!

- Type this code into Thonny and run
- The “callback” function runs whenever a button press is received.
- Whenever you press a button on the remote, you should see a number print out. This is the data your receiver received, think of it like an ID for each button.

```
import time
from machine import Pin
from ir_rx.nec import NEC_8

ir_pin = Pin(15, Pin.IN) # Set an input pin for the IR receiver

# This is the code that will run when a signal is recieved
def callback(data, addr, ctrl):
    if data < 0: # Data will be -1 if the button is held down
        return   # This stops the fuction is the button is held down

    print(data) # Print the 'ID' of your button


ir = NEC_8(ir_pin, callback) # This links our IR input pin to the code we want to run

while True:
    pass # Loop forever so the program doesn't end
```

Toggle an LED

- Now lets make the remote do something.
- Add this line to the begining of our program to setup an LED. Pin 25 is the LED built into the pico, so no extra wires necessary.

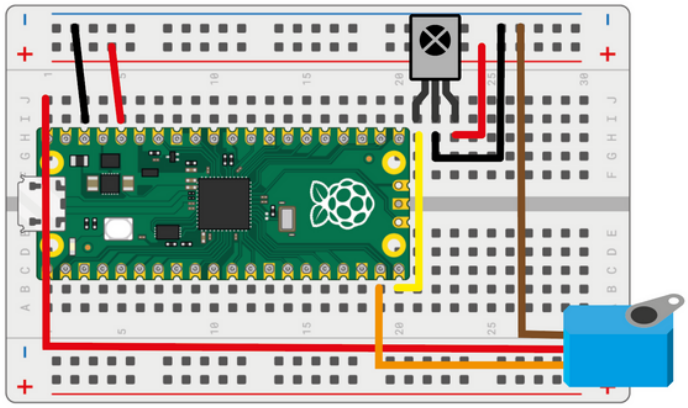
```
led = Pin(25, Pin.OUT) # Pin 25 is the LED on the pico itself
```

- Add statement to the callback function to toggle out LED when the specified button is pressed.
- 28 is the ID of the red OK button on the remote

```
    if data == 28:
        led.toggle()
```

Control a Servo Motor

- Wire up a servo motor to pin 14. Remember that servos need 5V power. The week 2 worksheet may help you.
- Add some code at the top of our program to setup the servo. Remember to import PWM!



```
import time
from machine import Pin, PWM
from ir_rx.nec import NEC_8

ir_pin = Pin(15, Pin.IN) # Set an input pin for the IR receiver
led = Pin(25, Pin.OUT) # Pin 25 is the LED on the pico itself

servo = PWM(Pin(14, Pin.OUT))
servo.freq(50)
duty_cycle = 1500
```

- We want to be able to hold down the button, so we will need to save the code of the last button press. Add a variable to save this at the top too.

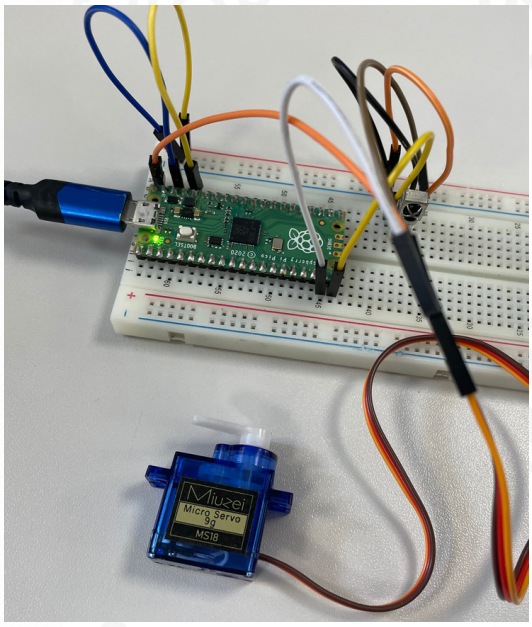
```
duty_cycle = 1500

last_button = -1
```

Modify the callback function

- If the data variable is -1 that means the button is held down. **Remove the return and instead set data to the last press.** We also need to save the button data for next time
- We are going to use **global variables** in the callback function, so we need to declare this with the **global** keyword
- We can now add some elif statements to rotate the serve. On the remote, the left arrow button has code 8 and the right code 90

```
14
15  # This is the code that will run when a signal is recieved
16  def callback(data, addr, ctrl):
17
18      global duty_cycle
19      global last_button
20
21      if data < 0: # Data will be -1 if the button is held down
22          data = last_button
23      else:
24          last_button = data
25
26      print(data) # Print the 'ID' of your button
27
28      if data == 28:
29          led.toggle()
30      elif data == 8:
31          if(duty_cycle >= 1000):
32              duty_cycle -= 1000
33          servo.duty_u16(duty_cycle)
34      elif data == 90:
35          if(duty_cycle + 1000 <= 65025):
36              duty_cycle += 1000
37          servo.duty_u16(duty_cycle)
38
39      time.sleep(0.1)
40
```

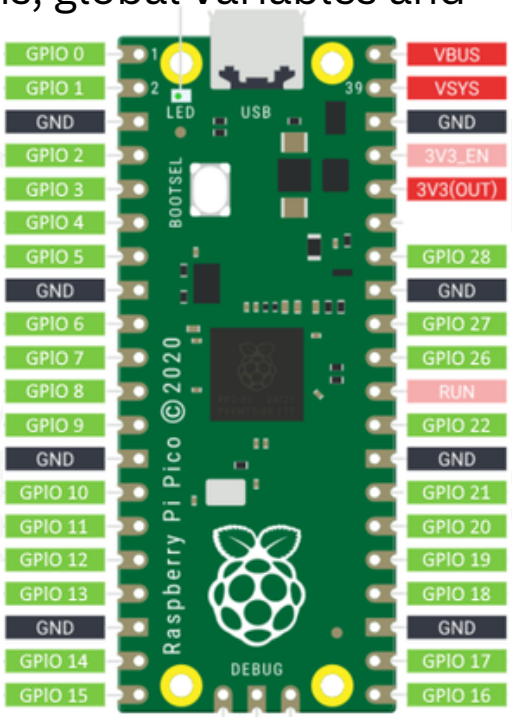


Review and extra challenges:

- We covered the concepts of how IR signals work and how we can implement them
- We should be more familiar with using functions, global variables and installing external modules

Challenge: What happens now if you hold down the OK button? Can you stop the LED blinking?

Pi Pico Code Docs



Worksheet produced by Patrick Culmer for use by HackSussex
https://github.com/supersand21/Robotics-Workshop