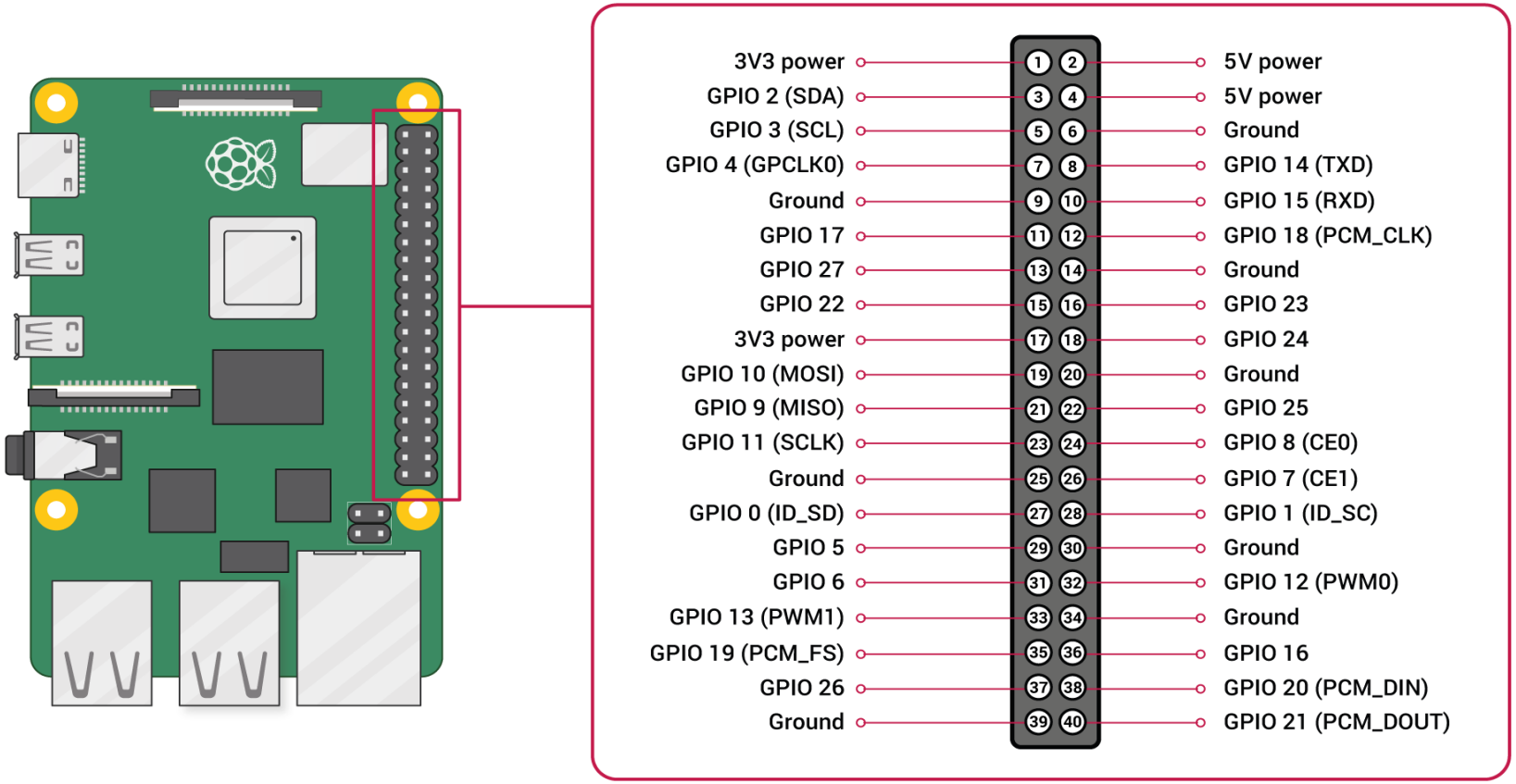
Fundamental of Cognitive Interaction with Robots

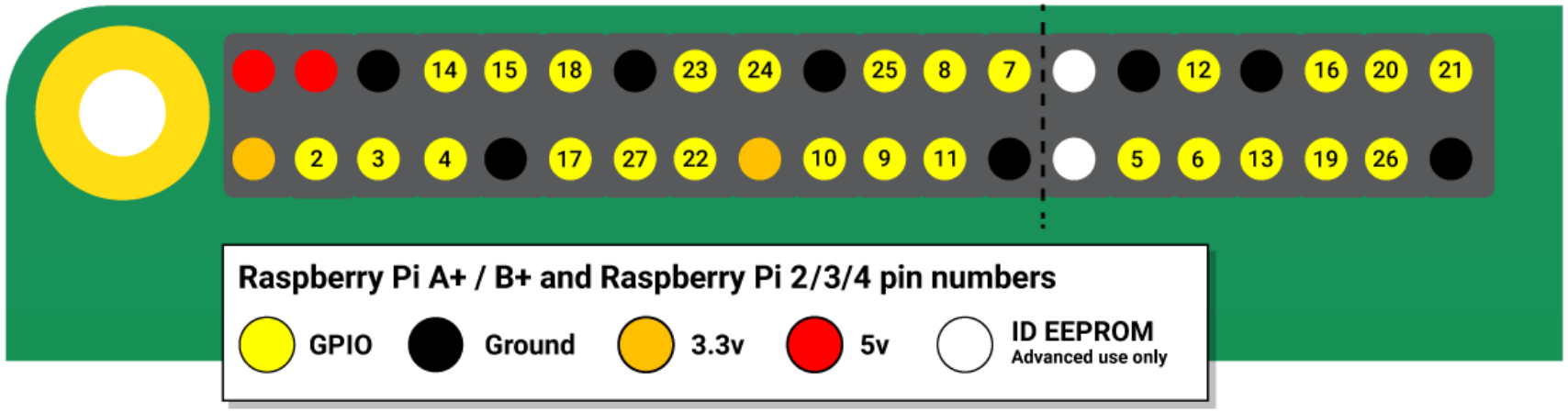
Lecture 4

GPIO pins in Raspberry Pi

- Raspberry Pi boards contain 40 GPIO (General Purpose Input/Output) pins.
- GPIO pins can be used to interface with the physical world whether by reading data from sensors, using components such as LEDs and displays, or controlling motors.



GPIO pins in Raspberry Pi



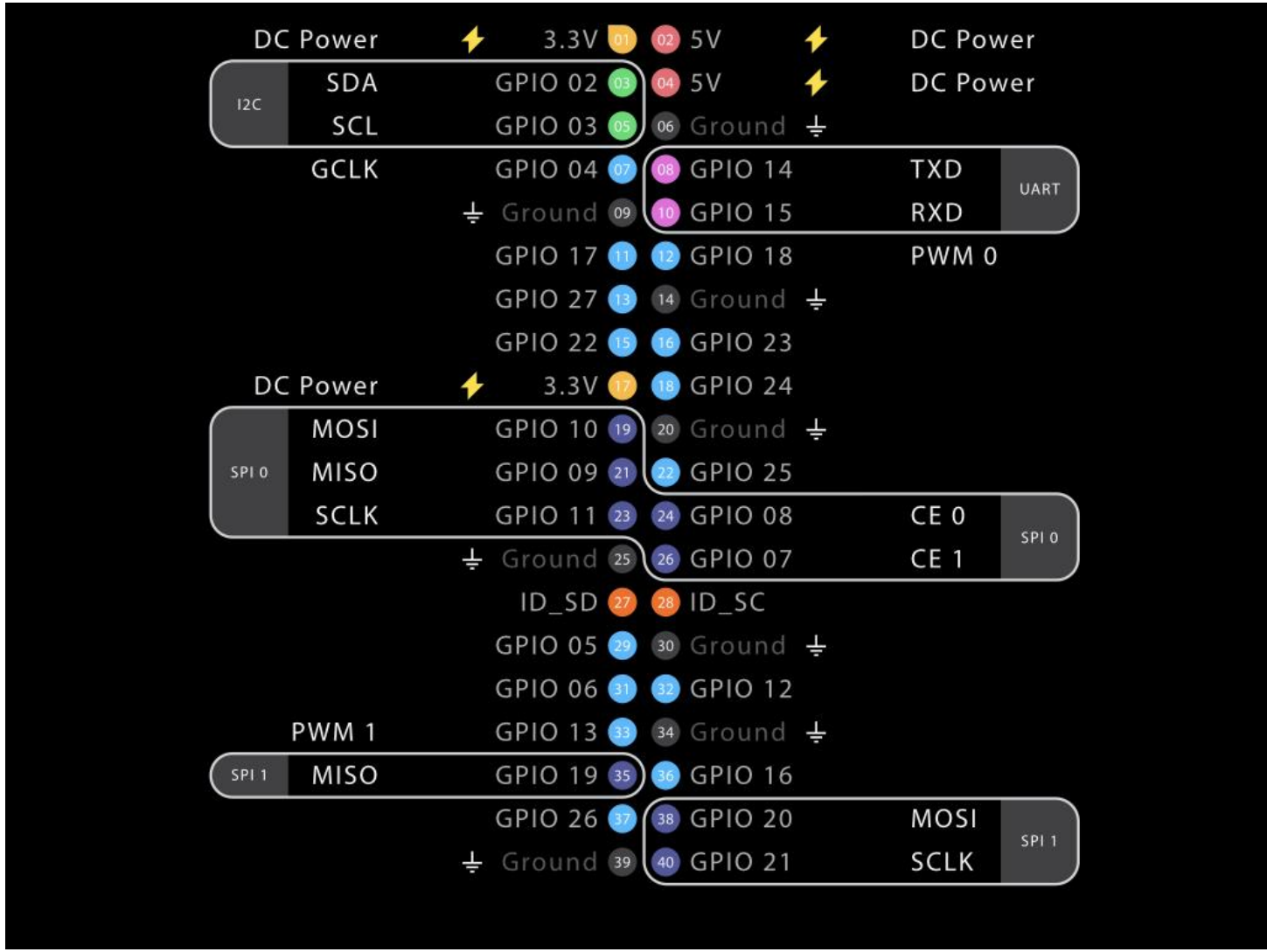
- The numbering of the GPIO pins is not in numerical order; GPIO pins 0 and 1 are present on the board (physical pins 27 and 28) but are reserved for advanced use.
- Any of the GPIO pins can be designated (in software) as an input or output pin.
- **Voltages:**

  Two 5V pins and two 3.3V pins are present on the board, as well as a number of ground pins.

  The remaining pins are all general purpose 3.3V pins, meaning outputs are set to 3.3V and inputs are 3.3V-tolerant.

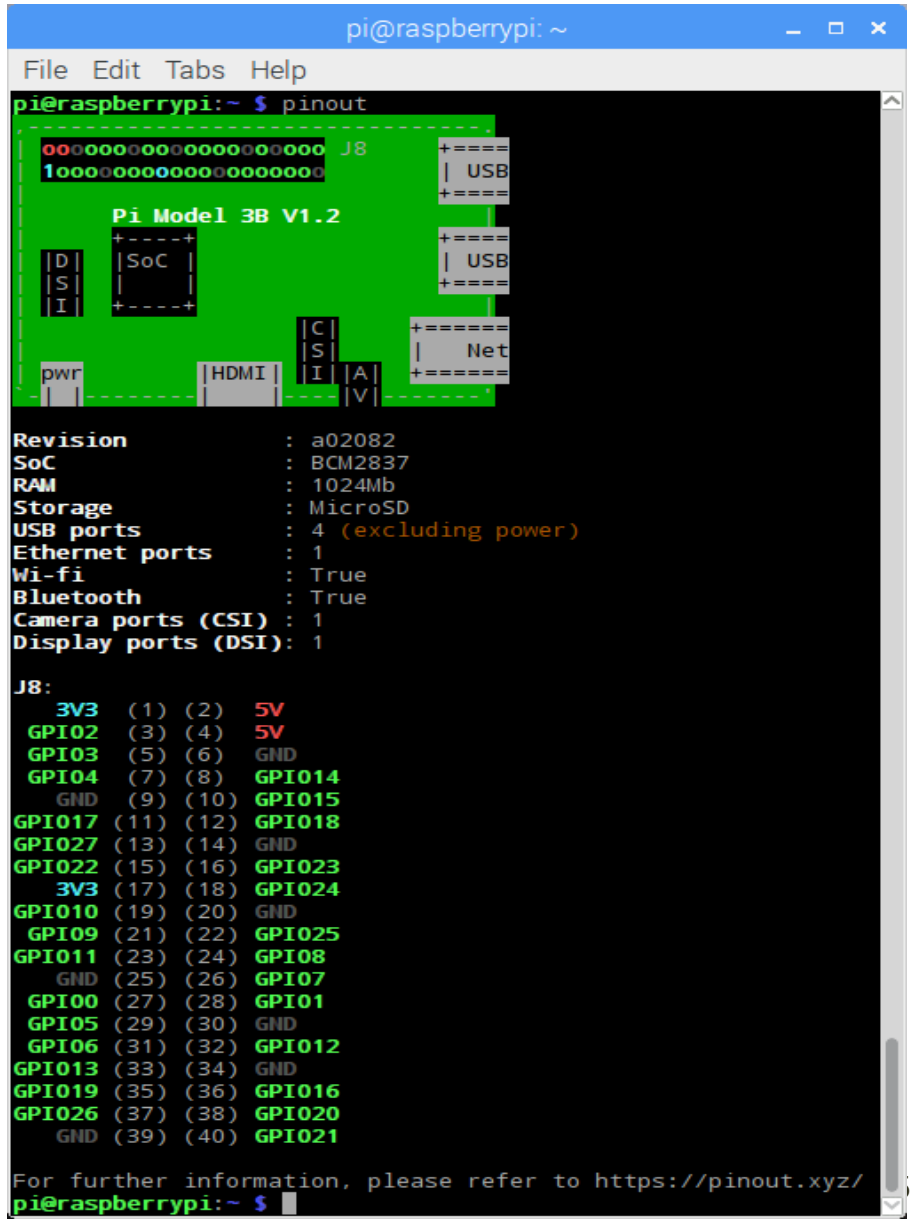
More !

- As well as input and output devices, the GPIO pins can be used with a variety of alternative functions:
  - PWM (all pins)
  - I2C
  - SPI
  - Serial



GPIO pinout

- A pinout reference for your Raspberry Pi can be accessed by opening a terminal window and running the command `pinout`.
- This tool is provided by the GPIO Zero Python library, which is installed by default on the Raspberry Pi OS, but not on Raspberry Pi OS Lite.



GPIO Pins

Permissions

- In order to use the GPIO ports your user must be a member of the gpio group. The pi user is a member by default, other users need to be added manually.

```
sudo usermod -a -G gpio <username>
```

WARNING

- LEDs should have resistors to limit the current passing through them.
- Do not use 5V for 3.3V components.
- Do not connect motors directly to the GPIO pins, instead use an H-bridge circuit or a motor controller board.

GPIO in Python

GPIO Zero Library

- Controlling the GPIO ports requires many more lines of code, but this is already written for you and made easy to use with the GPIO Zero Library.
- The library is comprehensively documented at gpiozero.readthedocs.io.
- GPIO Zero library is installed by default on Raspberry Pi.

Importing GPIO Zero

- In Python, libraries used in a script must be imported by name at the top of the file.
- For example, to use the Button from GPIO Zero:

```
from gpiozero import Button
```

Now Button is available directly in your script:

```
button = Button(2)
```

- Alternatively, the whole GPIO Zero library can be imported:

```
import gpiozero
```

In this case, all references to items within GPIO Zero must be prefixed:

```
button = gpiozero.Button(2)
```

GPIO Zero Library

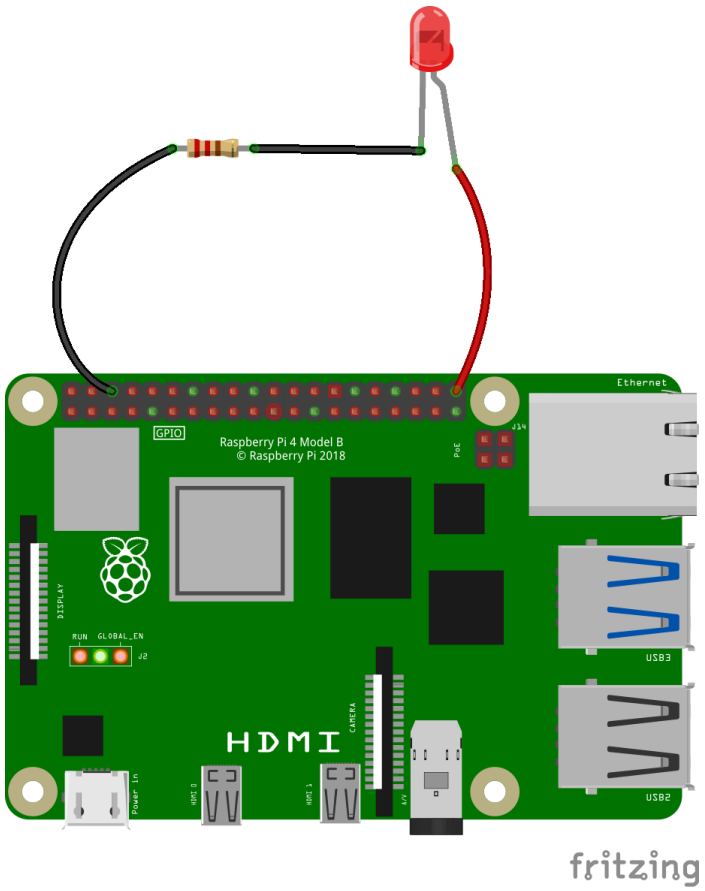
LED

To control an LED connected to GPIO21, you can use this code:

```
from gpiozero import LED
from time import sleep

red = LED(21)

while True:
    red.on()
    sleep(1)
    red.off()
    sleep(1)
```



Run this in an IDE like Thonny, and the LED will blink on and off repeatedly.

LED methods include on(), off(), toggle(), blink(), and value()

GPIO Zero Library

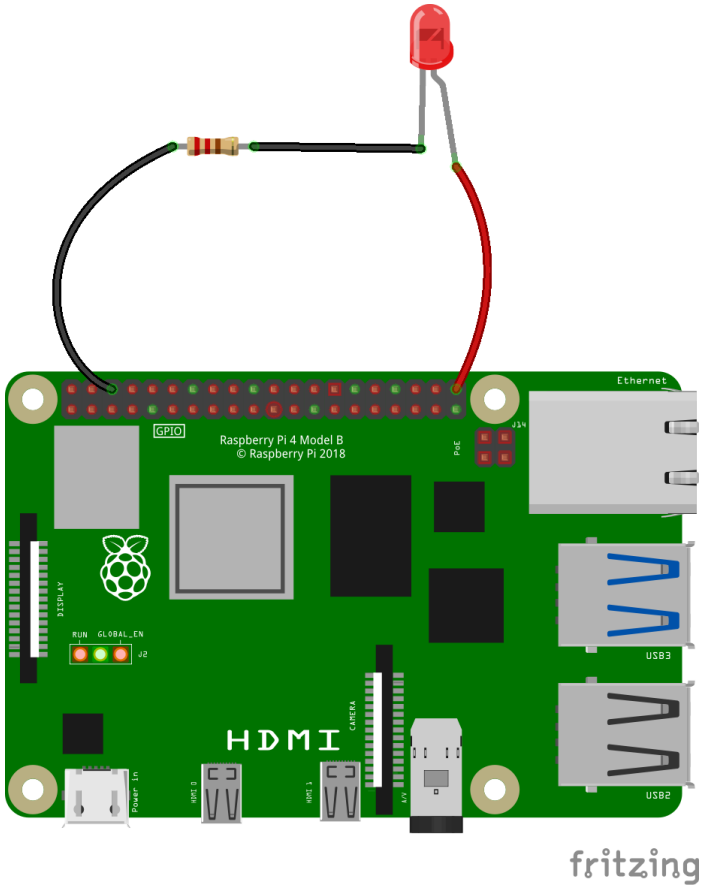
LED with variable brightness (PWM)

To change the brightness of an LED, PWMLED is used using values between 0 and 1:

```
from gpiozero import PWMLED
from time import sleep

led = PWMLED(21)

while True:
    led.value = 0  # off
    sleep(1)
    led.value = 0.5  # half brightness
    sleep(1)
    led.value = 1  # full brightness
    sleep(1)
```

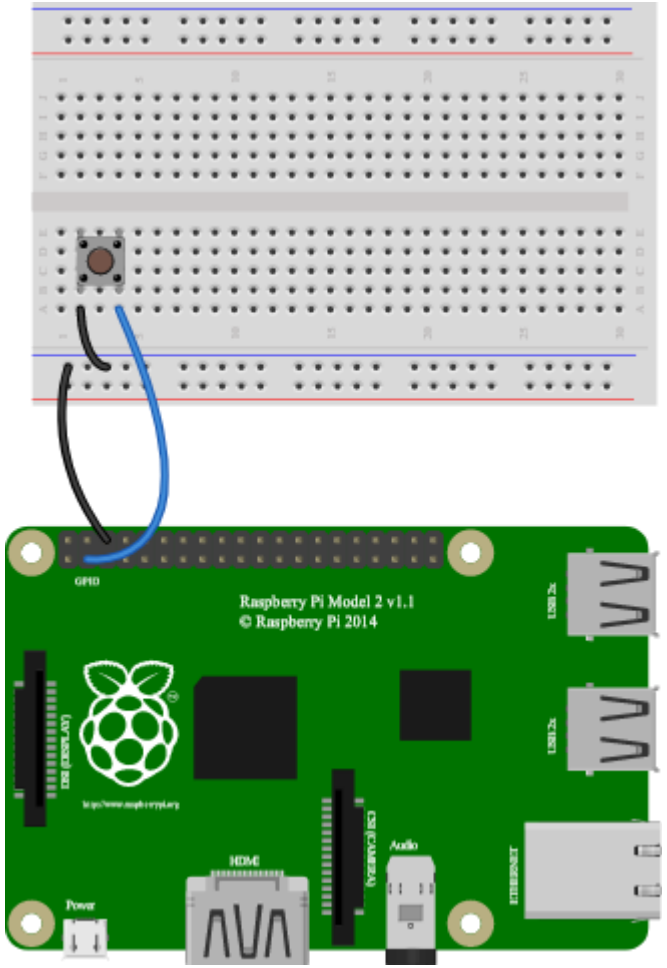


Button

Check if a Button is pressed:

```
from gpiozero import Button

button = Button(2)
while True:
    if button.is_pressed:
        print("Button is pressed")
    else:
        print("Button is not pressed")
```



Run a function every time the button is pressed:

```
from gpiozero import Button
from signal import pause

def say_hello():
    print("Hello!")

button = Button(2)
button.when_pressed = say_hello
pause()
```

Shutdown button

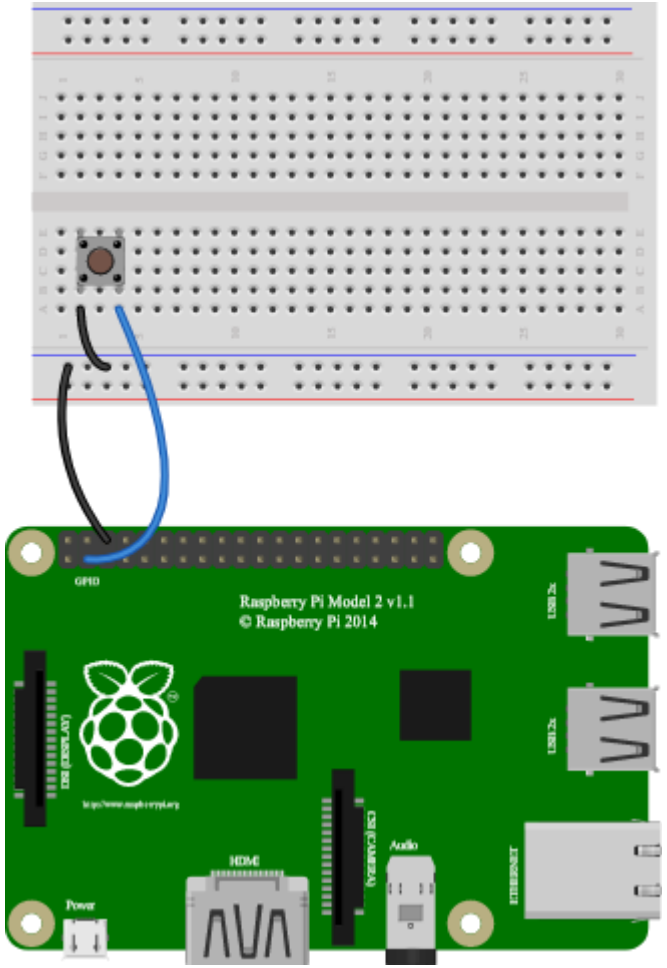
- The Button class also provides the ability to run a function when the button has been held for a given length of time.
- This example will shut down the Raspberry Pi when the button is held for 2 seconds:

```
from gpiozero import Button
from subprocess import check_call
from signal import pause

def shutdown():
    check_call(['sudo', 'poweroff'])

shutdown_btn = Button(2, hold_time=2)
shutdown_btn.when_held = shutdown

pause()
```



Keyboard controlled LED

```
import curses
from gpiozero import LED

led = LED(17)
actions = {
    curses.KEY_UP:  led.on,
    curses.KEY_DOWN:  led.off
}

def main(window):
    next_key = None
    while True:
          if next_key is None:
            key = window.getch()
        else:
            key = next_key
            next_key = None
        if key != -1: # KEY PRESSED
            action = actions.get(key)
            if action is not None:
                action()
            next_key = key
            while next_key == key:
                next_key = window.getch()

curses.wrapper(main)
```

- You can control an LED using a keyboard
- Up_arrow: led on
- Down_arrow: led off

Button controlled camera

- Using a button to take a picture from camera

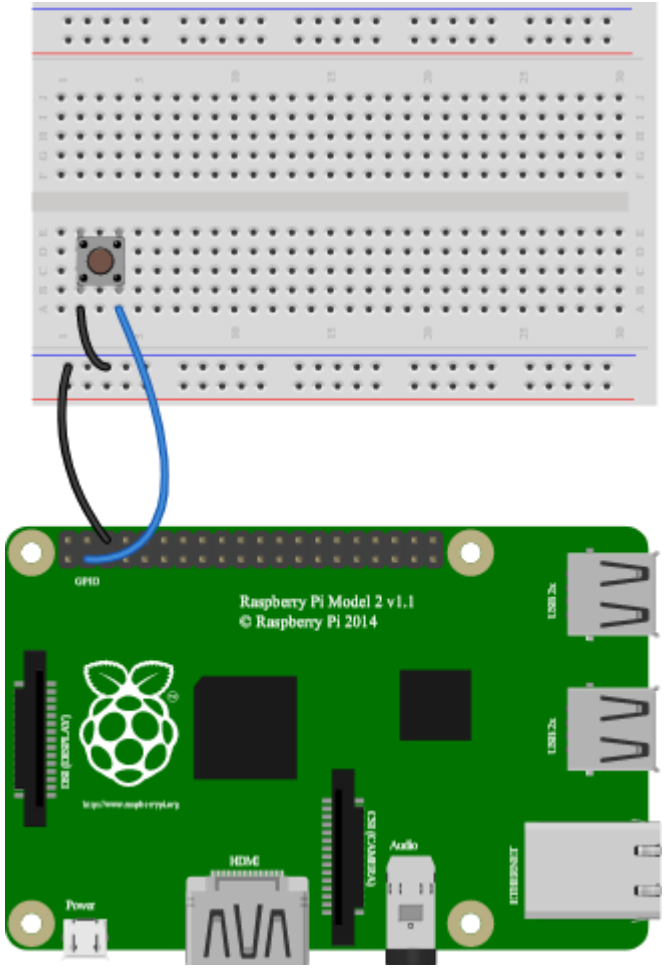
```
from gpiozero import Button
from picamera import PiCamera
from datetime import datetime
from signal import pause

button = Button(2)
camera = PiCamera()

def capture():
    timestamp = datetime.now().isoformat()
    camera.capture('/home/pi/%s.jpg' % timestamp)

button.when_pressed = capture

pause()
```



Button controlled camera

- Another example uses one button to start and stop the camera preview, and another to capture:

```
from gpiozero import Button
from picamera import PiCamera
from datetime import datetime
from signal import pause

left_button = Button(2)
right_button = Button(3)
camera = PiCamera()

def capture():
    timestamp = datetime.now().isoformat()
    camera.capture('/home/pi/%s.jpg' % timestamp)

left_button.when_pressed = camera.start_preview
left_button.when_released = camera.stop_preview
right_button.when_pressed = capture

pause()
```

- Note that the camera preview is not sent over VNC by default.
- To enable this option, go to VNC options on RP > troubleshooting and enable "experimental direct capture mode"

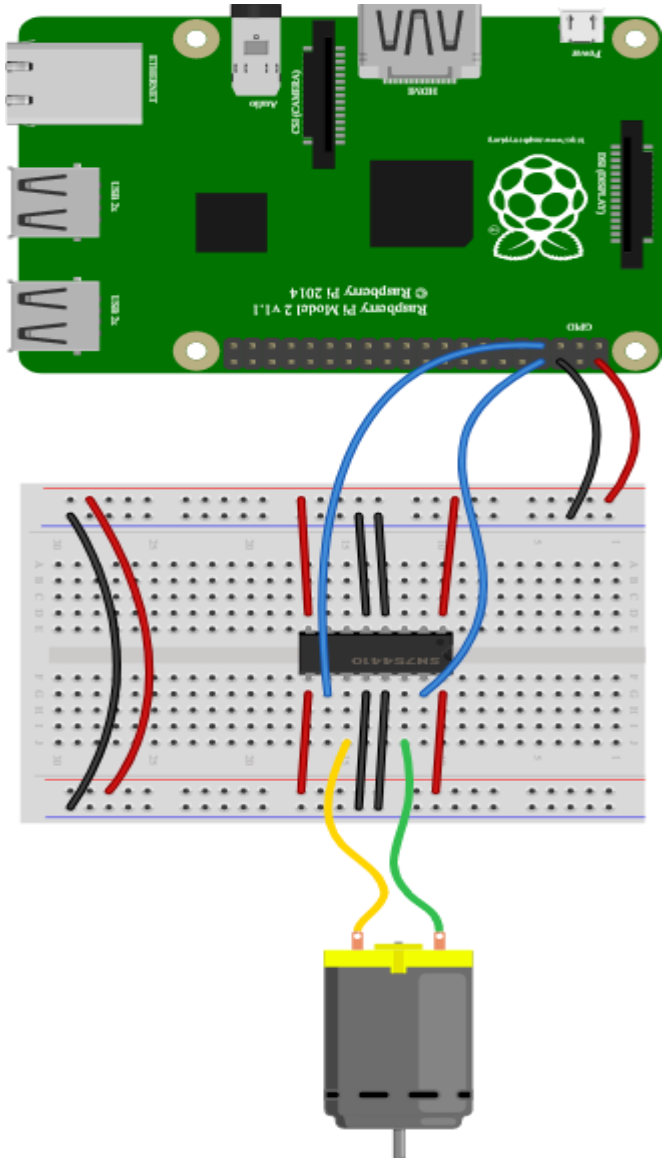
Motors

- Turn a Motor forwards and backwards:

```
from gpiozero import Motor
from time import sleep

motor = Motor(forward=4, backward=14)

while True:
    motor.forward()
    sleep(5)
    motor.backward()
    sleep(5)
```



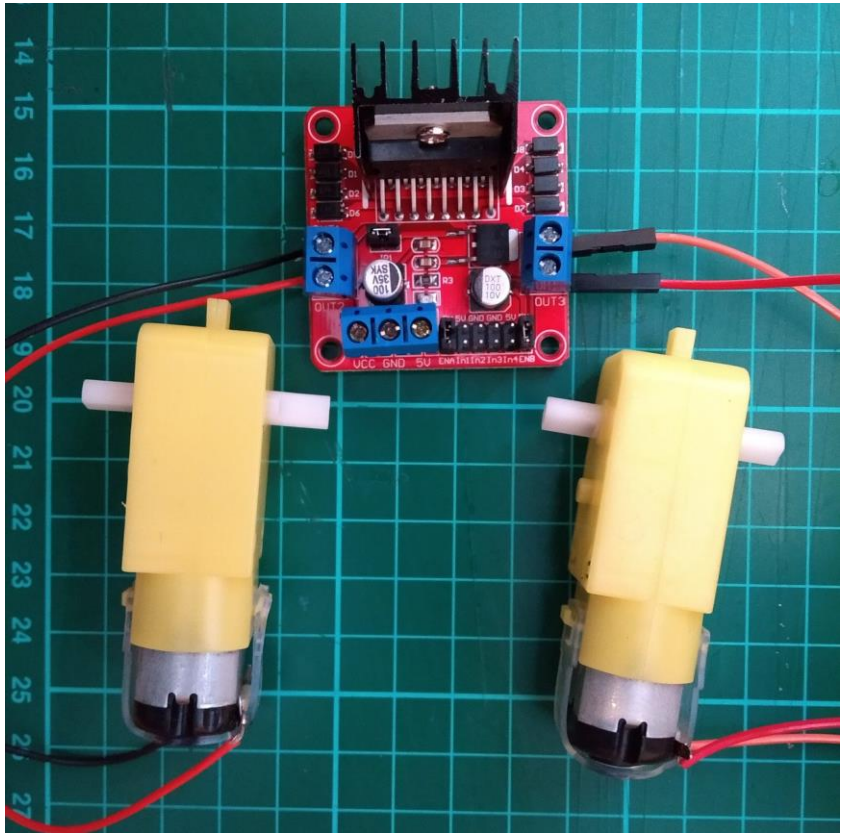
Robot

- Turn a Motor forwards and backwards:

```
from gpiozero import Robot
from time import sleep

robot = Robot(left=(4, 14), right=(17, 18))

for i in range(4):
    robot.forward()
    sleep(10)
    robot.right()
    sleep(1)
```

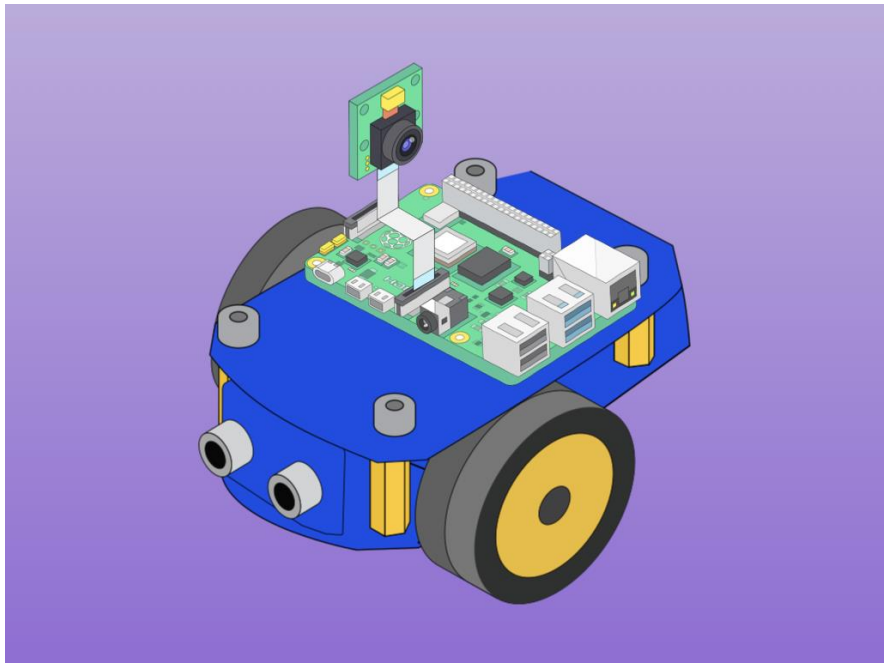


Keyboard Controlled Robot

- Build a keyboard-controlled robot add a camera to it so you can see where it is heading, and use a Wi-Fi device to view it remotely through VNC!

Adobe Acrobat Document

Code



Run a Raspberry Pi Program on Startup

Using rc.local file

- You will need root-level access to modify rc.local, so do so with sudo:

```
sudo nano /etc/rc.local
```

- Scroll down, and just before the exit 0 line, enter the following:

```
python /home/pi/program_name.py &
```

  where `program_name.py` is the program that you want to run at startup.

  Don't forget '&' at the end of the line.

- Save and exit with ctrl + x, followed by y when prompted to save, and then enter.
- Test it by restarting your Pi with `sudo reboot`.

Run a Raspberry Pi Program on Startup

The rc.local file will look as:

```
pi@raspberrypi: ~
  GNU nano 2.7.4                    File: /etc/rc.local

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

python /home/pi/blink.py &

exit 0

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Linter  ^_ Go To Line
```


Any Questions