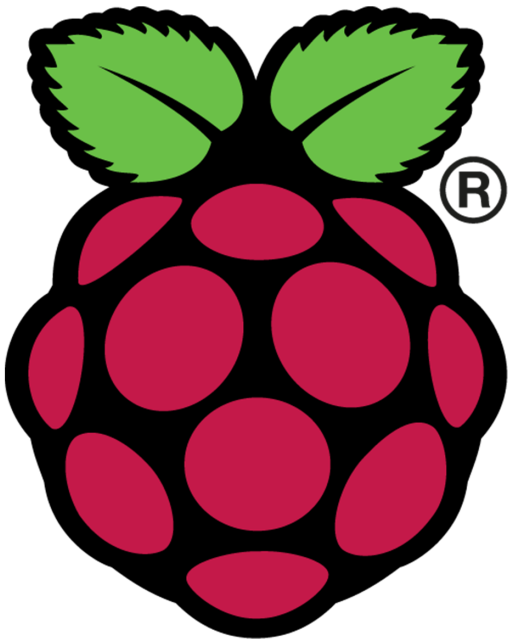
introduction to raspberry pi



Paul Baumgarten, 2018
pbaumgarten.com

Introduction

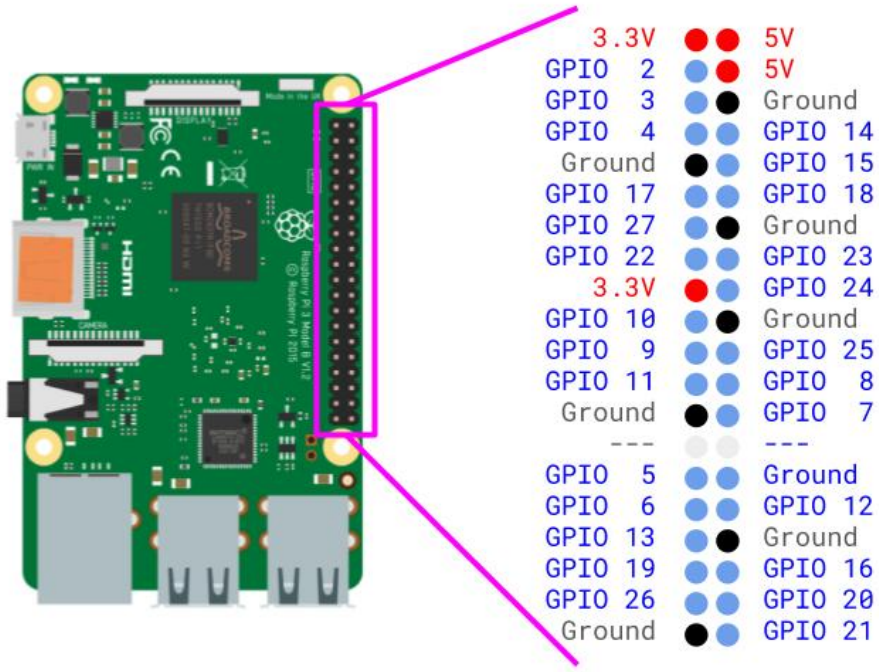
The Raspberry Pi 3b, which is what we are using today, comes with a range of built in ports and functions including:

- Quad core ARM Cortex-A53 1.2 GHz processor
- 1 GB of RAM (LPDDR2 900 Mhz)
- 4 x USB 2.0
- 1 x HDMI video out
- 1 x 3.5mm analog audio out
- 1 x Ethernet 10/100 networking
- 1 x Wifi 802.11n 2.4 GHz
- 1 x Bluetooth 4.1 classic
- Camera serial interface (CSI)
- Display serial interface (DSI)
- 40 GPIO (general purpose input/output) pins

Don't worry if none of the above means much to you (I really just mention it for any curious nerds who are into that kind of thing). The important bits for us today will be the camera interface and the GPIO.

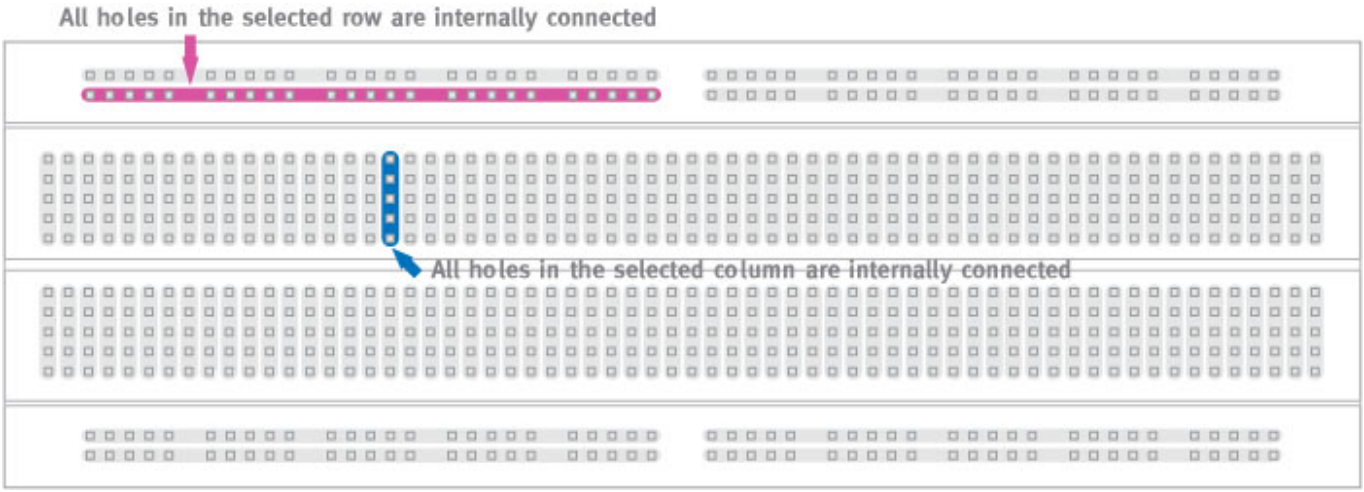
The GPIO, or general purpose input/output, are wires that are directly connected to the CPU of the Raspberry Pi, and you can control them through your Python program. You can instruct the pins to turn on or off electrical power (at 3.3 volts), or use Python to ask if there is a source of power being supplied to a particular pin (coming from a sensor of some kind). We will use the GPIOs to connect our various buttons, sensors and LEDs.

The following diagram will be important to reference as you work with the GPIOs. It shows the purpose and numbering of each pin.



To make wiring our sensors and accessories a bit easier, you have also got a breadboard. The holes on this device are all interconnected making it easier to wire different components together than if we were doing it with loose wire.

The broadboard is split into a few different sections, the long columns running down the long edges are all connected (as seen in purple in the diagram). In the middle section, each horizontal row of holes is interconnected, with a break at the midway point (as seen in blue in the diagram).



Power on and login

The Pi will automatically turn itself on when you plug the power in. The Pi draws power through its micro USB port. Note: You need a power supply of at least 2500 mA (be aware most phone chargers only provide 1500 to 2000 mA). The Pi will sometimes "appear" to work with a less capable power supply but will sometimes behave in weird unexplained ways if you aren't giving it enough power.

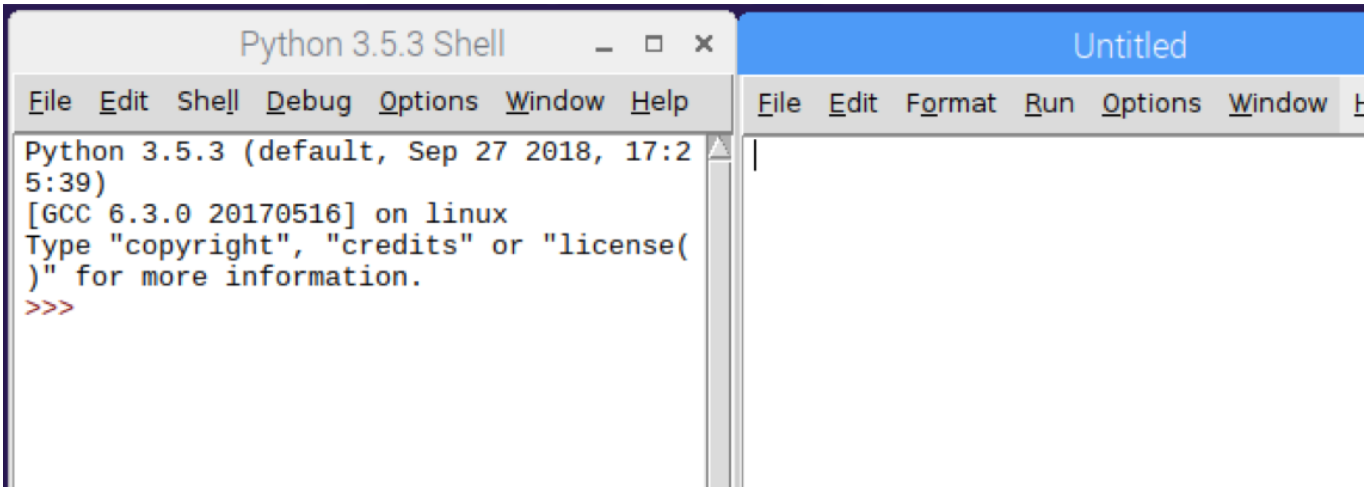
The Pi's for this weekend have been pre-installed ready for our workshop. They should automatically start up and connect to the wifi. If you are asked for a username and password at any time, they are as follows:

- Username: pi
- Password: lausanne

Using Python

Using the main menu (icon on the top-left of screen), open `Programming` and then start `Python 3 (IDLE)`.

By default, IDLE will only open the `shell` screen. You also need to open a second screen for your Python program code. Use the menu `file` / `new file` so it looks like this:



Test your Python works with a simple program. In the `Untitled` window, type

```
name = input("What is your name?")
print("Hello, "+name)
```

Use the menu `file` / `save` to save your Python program. Then menu `run` / `run module` to execute. If all works correctly you will be prompted with the question on the shell screen. Ensure you click the mouse into the shell screen before typing your response. Once tested, you can create any Python program you like this way. If you have time feel free to experiment with your own programs, in the meantime we'll move on with the workshop.

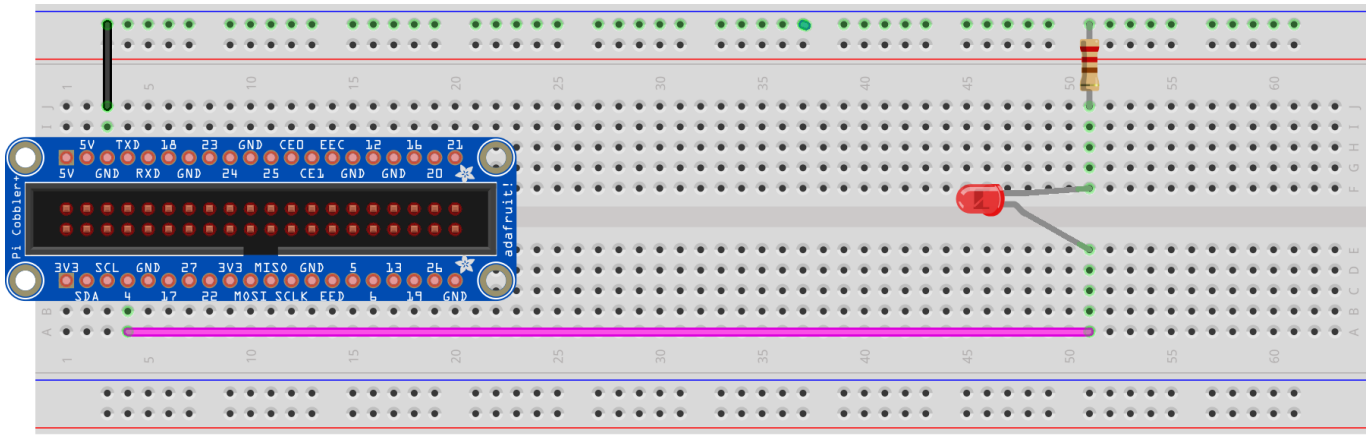
How to use an LED

You will need:

- An LED (obviously)
- A resistor around 200 ohms (these are the ones with the blue shell in your pack).
- Two connector wires

Wire your LED on the breadboard similar to the following diagram. Have a go and then if you are unsure, have an instructor check it. The resistor is important to moderate the flow of current so we don't overload the Raspberry Pi CPU which is quite delicate.

Notice the lengths of the pins on the LED are different. This is because the LED will only work one way. In this case, the longer pin should be the one that gets wired up to GPIO 4, and the shorter pin should go to Ground.



First a really simple test, let's make the LED turn on for 2 seconds then turn off.

```
import time
import easyaspi

LED_PIN = 4                 # change this number to whatever GPIO pin you used
led = easyaspi.LED(LED_PIN) # Create our LED variable

led.set(True)               # Turn the LED on
time.sleep(2)               # Wait 2 seconds
led.set(False)              # Turn the LED off

print("All done!")
```

To up the ante, modify `main.py` to make it blink a given number of times.

```
import time
import easyaspi

LED_PIN = 4                 # change this number to whatever GPIO pin you used
led = easyaspi.LED(LED_PIN) # Create our LED variable

blinks = int(input("How many times do you want the LED to blink?"))
while blinks > 0:
    led.set(True)           # Turn the LED on
    time.sleep(0.5)         # Wait half a second
    led.set(False)          # Turn the LED off
    time.sleep(0.5)         # Wait half a second
    blinks = blinks - 1
print("All done!")
```

From now on, any time we want an LED to turn on or off, we only need to update our `main.py` code.

LED summary

The key parts to using an LED are:

- Create the LED variable

```
import time
import easyaspi

led = easyaspi.LED( pin_number )
```

- Turn an LED on

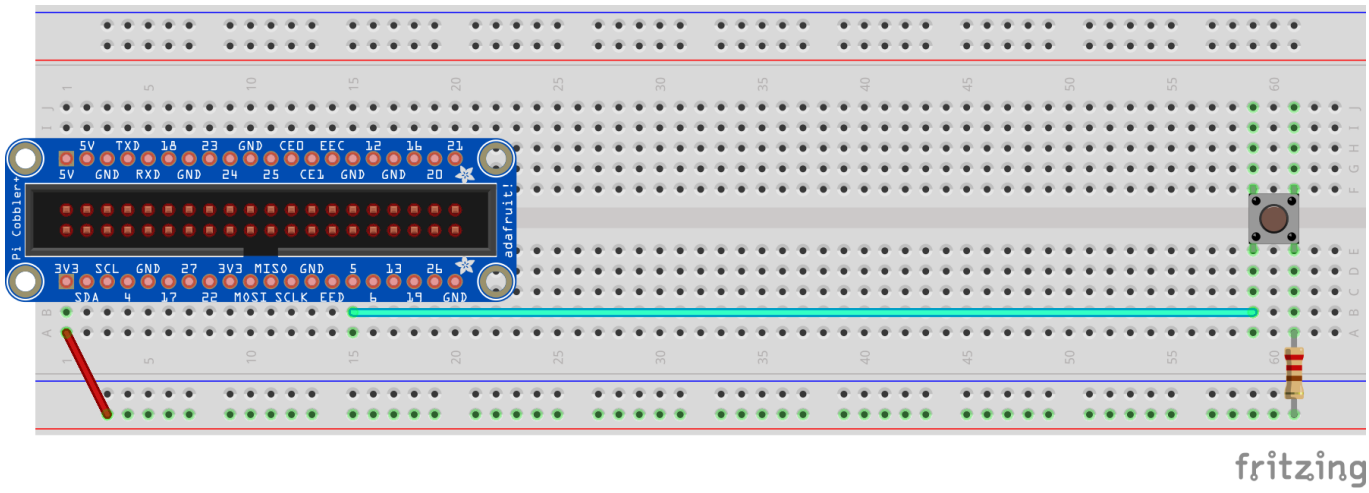
```
led.set(True)
```

- Turn an LED off

```
led.set(False)
```

How to use a button

This diagram shows the new wiring to add. **Do not disconnect the LED wiring, you will still use the LED!**



There's a couple of different ways a button can be used. We're just going to query the state of the button at a given moment in time. Using the Button class we have created, the `.get()` function will return `True` if the button is being pressed and `False` if it is not.

Replace the content of your `main.py` with the following to test a button:

```
import time
import easyaspi

BUTTON_PIN = 5                      # Change to whatever pin you connected to
button = easyaspi.Button(BUTTON_PIN)# Create our button vairable

end_at = time.time() + 30           # We'll run for 30 seconds
while time.time() < end_at:
    if button.get():
        print("You are pressing the button")
    time.sleep(0.3)                 # Wait part of a second and check again
```

Do you still have your LED wired up? We could use our button to turn it on and off! Press once to turn the LED on, press again to turn it off.

```
import time
import easyaspi

LED_PIN = 4
BUTTON_PIN = 5
led = easyaspi.LED(LED_PIN)
button = easyaspi.Button(BUTTON_PIN)

end_at = time.time() + 30           # We'll run for 30 seconds
while time.time() < end_at:
    if button.get():
        print("You are pressing the button")
        led_status = led.get()      # What is the status of the LED?
        led.set( not led_status )   # Flip the status of the LED
    time.sleep(0.3)                 # Wait part of a second and check again
```

Button summary

The key parts to using a Button are:

- Create the Button variable

```
import easyaspi
button = easyaspi.Button( pin_number )
```

- Retrieve if the button is being pressed

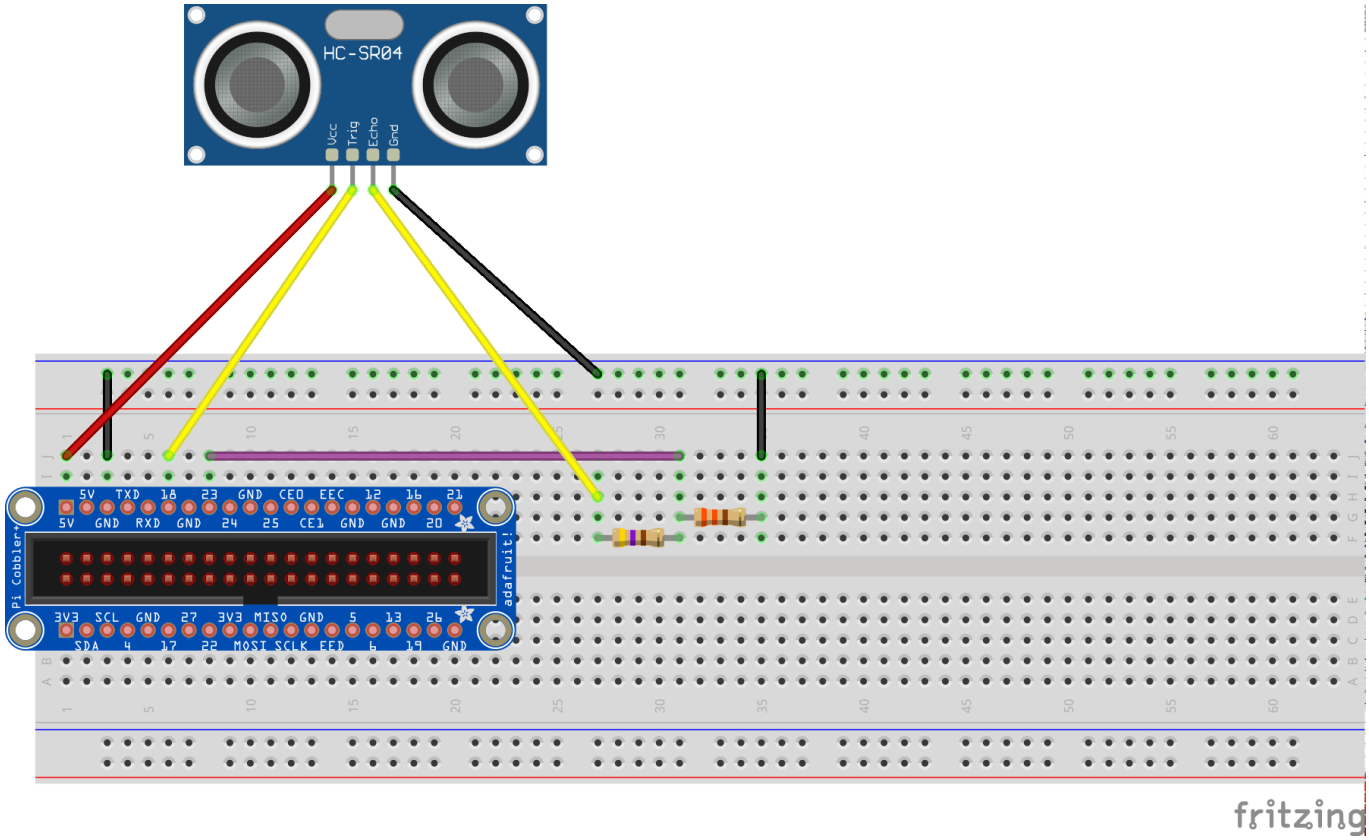
```
button_state = button.get()
```

How to use an ultrasonic sensor

An ultrasonic sensor works by sending a high freqency sound pulse, and using the speed of sound (we assume sea-level), it can calculate how far an object is by measuring how long it took for it to hear an echo of that pulse. Pretty cool eh?! Who would have thought you'd be using Physics today!

Parts list:

- HC-SR04 or similar ultrasonic sensor
- 330 ohm resister (approximately)
- 470 ohm resister (approximately)
- Connector wires



Here is some demo code for using the sensor.

```
import time
import easyaspi

ULTRASONIC_TRIGGER = 18             # Change to whatever pin you connected to
ULTRASONIC_ECHO = 23                # Change to whatever pin you connected to

ultra = easyaspi.Ultrasonic(ULTRASONIC_TRIGGER, ULTRASONIC_ECHO)

end_at = time.time() + 30           # We'll run for 30 seconds
while time.time() < end_at:
    distance = ultra.get_distance()
    if distance > 0:
        print("Nearest object is",distance,"centimeters")
    else:
        print("Object out of range")
    time.sleep(0.2)                 # Wait part of a second and check again
```

Still got an LED connected? We could make the LED turn on if someone is getting too close!

```
import time
import easyaspi

LED_PIN = 4                         # Change to whatever pin you connected to
ULTRASONIC_TRIGGER = 18             # Change to whatever pin you connected to
ULTRASONIC_ECHO = 23                # Change to whatever pin you connected to
ultra = easyaspi.Ultrasonic(ULTRASONIC_TRIGGER, ULTRASONIC_ECHO)
led = easyaspi.LED(LED_PIN)

end_at = time.time() + 30           # We'll run for 30 seconds
while time.time() < end_at:
    distance = ultra.get_distance()
    if distance > 0 and distance < 20:
        led.set(True)
        print("TOO CLOSE! Only",distance,"centimeters")
    elif distance > 0:
        led.set(False)
        print("Nearest object is",distance,"centimeters")
    else:
        led.set(False)
        print("Object out of range")
    time.sleep(0.2)                 # Wait part of a second and check again
```

Ultrasonic summary

The key parts to using an Ultrasonic are:

- Create the Ultrasonic variable

```
import easyaspi
ultra = easyaspi.Ultrasonic( trigger_pin_number, echo_pin_number )
```

- Retrieve the distance in centimeters

```
distance = ultra.get_distance()
```

How to use the Picamera to take a photo

The camera is actually quite easy to use in your project. Go ahead and take a selfie or two.

```
import time
import easyaspi

camera = easyaspi.Camera()
print("I'm going to take a photo! Ready?")
print("3...")
time.sleep(1)
print("2...")
time.sleep(1)
print("1...")
time.sleep(1)
print("Smile!")
camera.photo("selfie.png")
print("All done")
camera.preview(False)               # Close the preview window when done
```

Notice that all we had to do is create the `camera` variable and then use the `.photo()` function? Nice and easy!

There's a couple of other things we can also do with it, like include a message in the image. For example, to add the time and date the photo was taken, add the additional two lines and modify the `camera.photo` line as shown below.

```
now = datetime.datetime.now()
message = now.strftime("%H:%M:%S %d/%m/%Y")
camera.photo("selfie.png", message)
camera.preview(False)               # Close the preview window when done
```

Ready to take it to the next level? Still got your button connected? How about we take a photo when the button is pressed....

```
import time
import easyaspi

BUTTON_PIN = 5
camera = easyaspi.Camera()
button = easyaspi.Button(BUTTON_PIN)

print("Press the button to take a photo")
while not button.get():
    time.sleep(0.1)                 # Wait part of a second and check again
print("SMILE!")
now = datetime.datetime.now()
message = now.strftime("%H:%M:%S %d/%m/%Y")
camera.photo("selfie.png", message)
camera.preview(False)               # Close the preview window when done
```

Can you make the LED blink while we are waiting for the button to be pressed? Check back when we made the button turn on/off when the button was being pressed. How did we make the LED flip, so that when it was off it turned on, and when it was on it turned off. You want to add something similar inside the while loop of this.

How to use the Picamera to record video

We're not just limited to still photos with the Picamera, we can also record video! (Be aware these video files get very large very quickly. When I was testing it, a few seconds of video were several hundred megabytes each).

To test the video recording function, we could start recording when the program starts, and then stop when the button is pressed.

```
import time
import easyaspi

BUTTON_PIN = 5
camera = easyaspi.Camera()
button = easyaspi.Button(BUTTON_PIN)

print("Press the button to stop the recording")
camera.record("myvideo.h264")
while not button.get():
    time.sleep(0.1)                # Wait part of a second and check again
camera.stop()
camera.preview(False)
```

Can you use the LED to indicate that video recording is in progress? It should be quite similar to what you did for the photo exercise.

Picamera summary

The key parts to using the Picamera are:

- Create the Camera variable

```
import easyaspi
camera = easyaspi.Camera()
```

- Take a photo (without a message)

```
camera.photo("myphoto.png")
```

- Take a photo (with a message)

```
camera.photo("myphoto.png", "my message")
```

- Start video recording

```
camera.record("myvideo.h264", "my message")
```

- Check to see if camera is recording video

```
recording_state = camera.recording      # Returns True or False
```

- Stop video recording

```
camera.stop()
```

- Close the camera preview window when finished

```
camera.preview(False)
```

Challenges

Now you've learnt the core bits of the Raspberry Pi, it's time to put them to use. Your `sensors.py` file is now complete. You only need to modify your main project file now.

Here are some ideas for potential mini-projects you might like to try:

Challenge 1: Motion sensor alarm that takes a photo of the intruder

You will need your Ultrasonic sensor and Camera for this to work. Use the camera to take a photo anytime someone comes within a meter of your Raspberry Pi! For this to be useful, you'll want to change the name of the PNG file each time so you don't just save over the top of the same file.

As a suggestion, I'd use a filename based on the current time and date. Perhaps using something like this? Make sure you include a time/date stamp annotation in the PNG file as well.

```
now = datetime.datetime.now()
filename = now.strftime("%Y-%m-%d-%H-%M-%S.png")
```

Challenge 2: Motion sensor alarm that records video of the intruder

Modify the previous challenge so that it records the intruder on film!

- Start recording when the intruder comes within a meter of the Raspbery Pi
- Stop recording once the intruder has gone away
- Keep looping, so if they return, you record them again

Challenge 3: Add some LEDs and Buttons to your intruder alarm

Modify either of the previous challenges (ie: use photos or video, whichever you prefer) to incorporate a few extra features:

- One LED that indicates the program is running... watching... guarding
- A second LED that indicates the alarm has been triggerd
- A button to end the program loop

Challenge 4: Make a selfie-studio

Moving away from the intruder detection system, here's a different idea using 2 buttons, 2 LEDs and the camera.

- One button used to take a photo when pressed
- A second button used to start/stop a video when pressed
- Keep looping so you can take as many photos and videos as you like
- One LED to indicate video recording in progress
- Another LED to indicate photo is being taken

More?

What other ideas can you come up with?

Want your own?

Want your own Raspberry Pi to tinker with at home? Why not?!

The basic Pi is less than 50 CHF, though you'd probably want a few extra bits initially but you can still get a good starter kit for less than 100 CHF.

I have links to various stores in Switzerland and online suppliers on my website, as well as some setup instructions to help you get started. I'll also post a digital copy of these notes.

- https://pbaumgarten.com/raspberrypi

If you do get your own, just be aware that the `easyaspi` package isn't installed by default on the Raspberry Pi. You'll need to open a terminal and install it with the following command:

```
sudo pip3 install easyaspi
```

Acknowledgements

Raspberry Pi is a trademark of the Raspberry Pi Foundation

Raspberry Pi logo is used in accordance with the published permissions of the Raspberry Pi Foundation
https://www.raspberrypi.org/trademark-rules/

Croston, B (undated) RPi.GPIO Python Module
https://sourceforge.net/p/raspberry-gpio-python/wiki/Home

Jones, D (0214) picamera
https://picamera.readthedocs.io/en/release-1.12/index.html

"Butix" (2016) Raspberry Pi layout illustration
https://commons.wikimedia.org/wiki/File:Raspberry_Pi_3_illustration.svg

Raspberry Pi Foundataion (undated) GPIO pin layout
https://www.raspberrypi.org/documentation/usage/gpio/

Hussain, A (2017) Distance Calculation with Ultrasonic Sensor
https://www.hackster.io/arbazhussain/distance-calculation-with-ultrasonic-sensor-26d63e