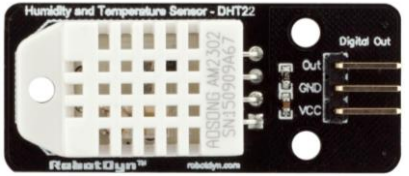
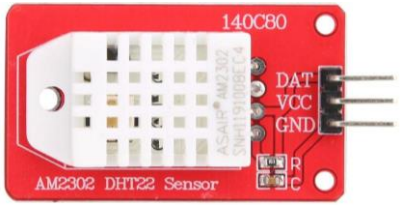
Learning outcomes

- Introduce DHT humidity and temperature sensor
- Introduce PIR movement sensor
- Introduce IR object sensor
- Introduce ultrasonic distance sensor

DHT humidity and temperature sensor

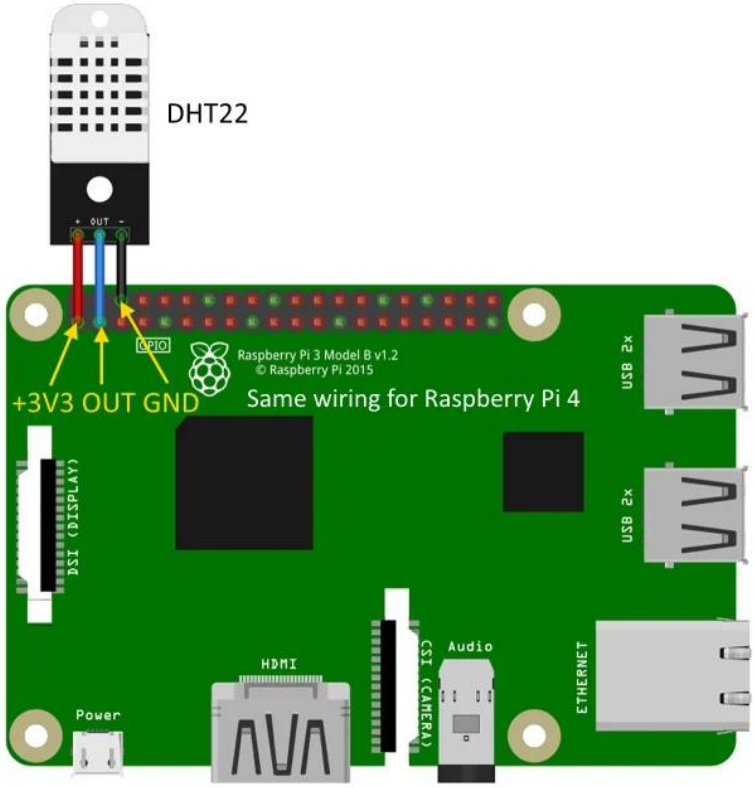
- A digital-output relative humidity and temperature sensor uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin.
- Typical models are DHT11 and DHT22.





DHT Circuit

- Connection to DHT sensor is simple.
- Connect VCC to the 3.3v power of Raspberry Pi
- Connect GND to the ground
- Connect DAT to one of the GPIO pin.



DHT program (DHT.py)

- It is complicated to use GPIO raw input to interpret the data from sensor.
- So DHT library is recommended to get the temperature
- It could directly get the number result

PIR movement sensor

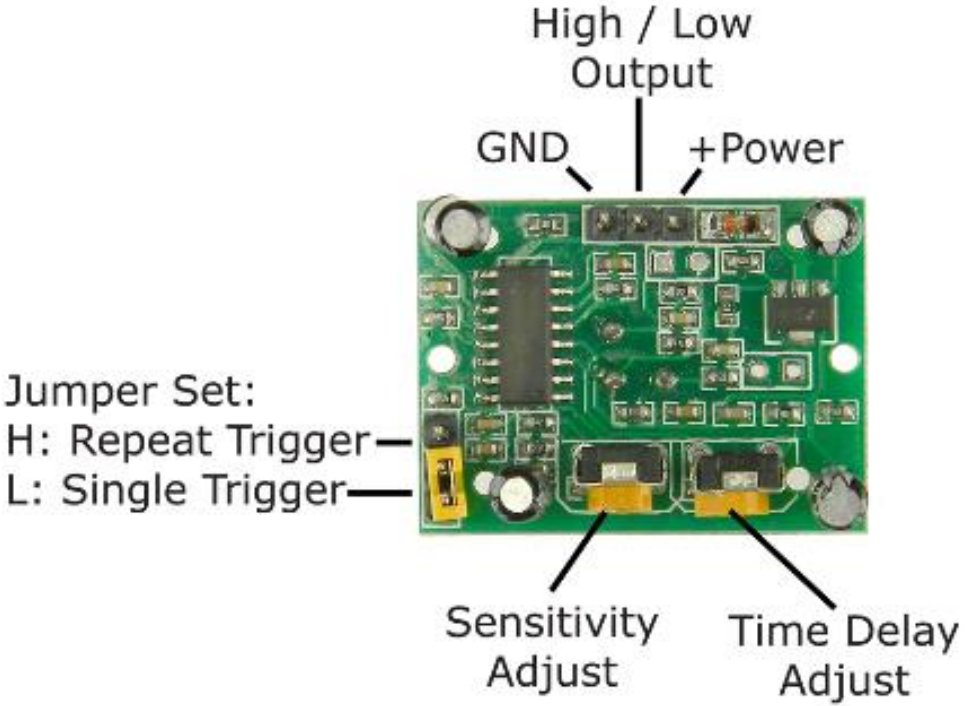
- We as humans are emitting some infrared light, which depends on the temperature of our body.
- The **passive infrared (PIR)** sensor measures infrared (IR) light radiating from objects in its field of view
- We can't see the infrared light with our eyes, but the sensor can detect it.
- The sensor will detect variations of that infrared light, which means, in other words, movement.

PIR movement sensor

- Real life applications
  - Automatic lighting
  - Security alarms
- Output is simple: it is binary output - Either there is a movement which has been detected or there is not.

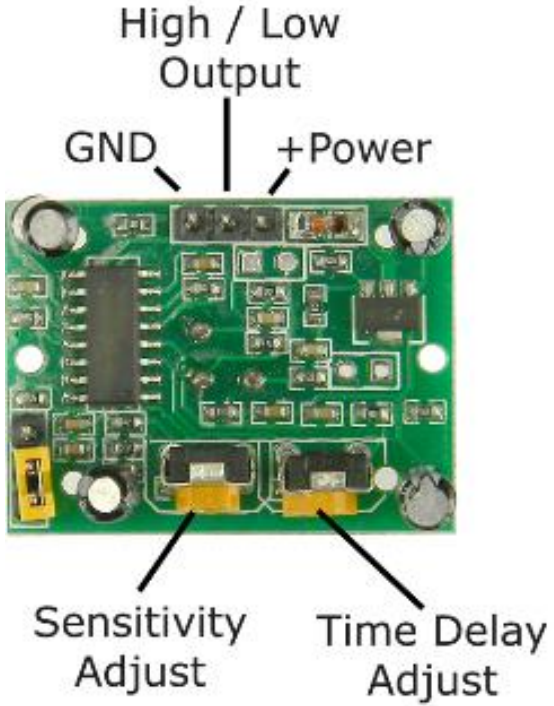
PIR movement sensor (Jumper Set)

- Jumper Set is used to set the mode of the sensor
  - If we set it to High (connect High pin and the middle pin with jumper), it is **Retrigger mode**. The output of sensor will remain HIGH as long as movement is being detected
  - If set to Low (**Normal mode**), every movement detected will result in a HIGH then LOW pulse output.
  - **Retrigger mode** is more useful most of the time.



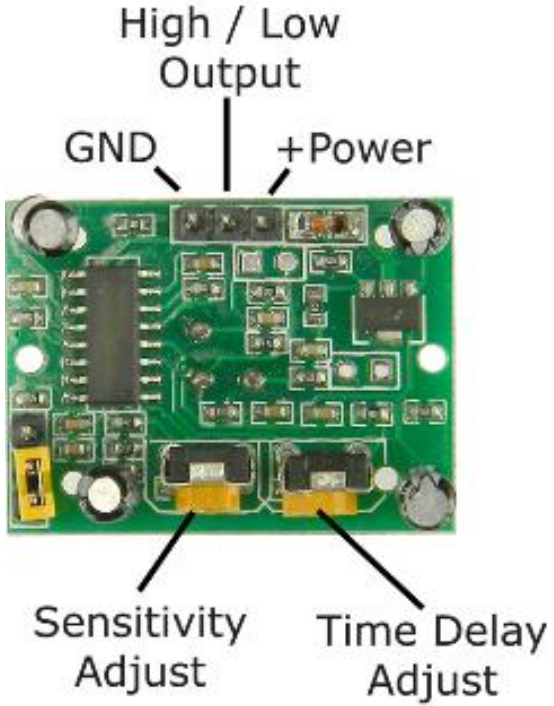
PIR movement sensor (Adjust)

- There are two adjustment on the sensor, we could use screwdriver to adjust.
- **Sensitivity Adjust** defines range of the sensor
  - if we turn it counterclockwise, range will drop to minimum of 3 meters
  - If we turn it clockwise, range will increase to maximum of 7 meters
- **Time Delay Adjust** defines how long output will remain HIGH when movement is detected
  - counterclockwise will make the time shorter down to 1 sec.
  - clockwise will make the time longer up to 5 min.



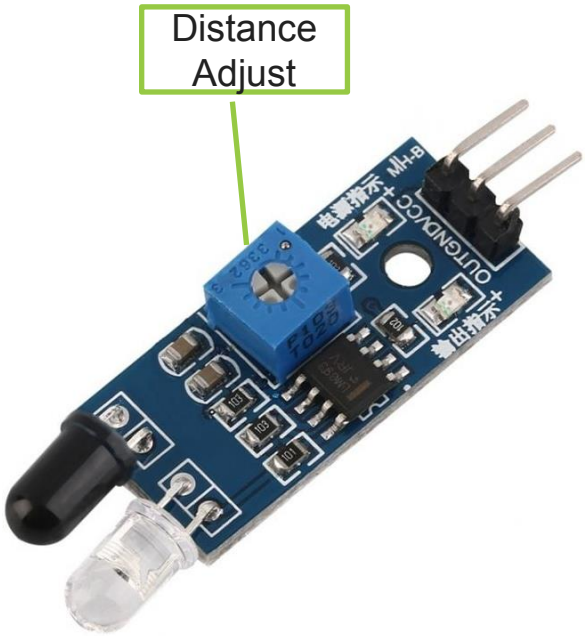
PIR movement sensor (sensor_move.py)

- We just connect the Power to 5V power, GND to ground, and Output to one GPIO pin (use a resistor between to protect GPIO).
- Now we are able to read PIR's data with Python.



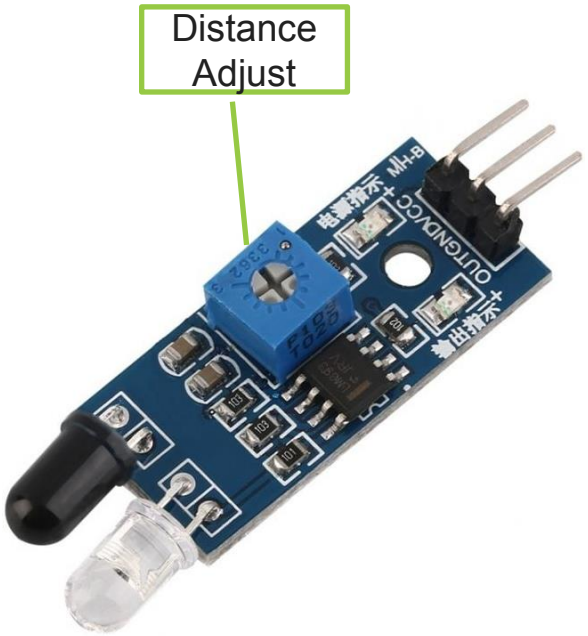
IR object sensor

- **Infrared (IR) sensor** has one IR LED which will emit Infrared light.
- It also has an IR photodiode which act as light detector. Photodiode is sensitive to IR light of the same wavelength which is emitted by the IR LED.
- So we could use IR sensor to detect object in front of it as the IR emitted will be reflected back by object and received by photodiode.
- It also has a potentiometer which is used to adjust the detection distance.
  - Turning it counterclockwise will make it only detect very close object (or can not detect at all)



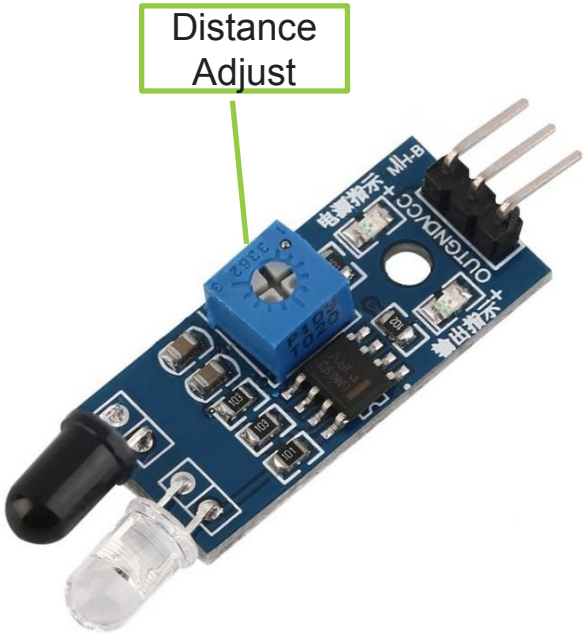
IR object sensor

- You will find two small LEDs on the sensor, one will turn on once connected to a circuit with power. Another will turn on once it detect something. We could use such feature for easy testing.
- As IR sensor is detecting emitted infrared light, if a surface could absorb most of the light (black color surface basically), then it could not detect anything (output will always be LOW).
- Following same logic, if we cast light upon the photodiode, then it considered item detected all the time (output will always be HIGH).



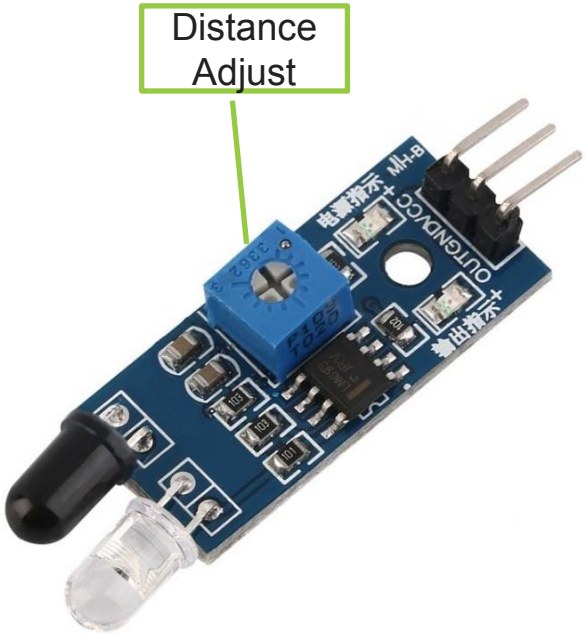
IR object sensor

- As a result, we could also use IR sensor to distinguish between a black and a white color object. (Useful in case like line tracing robot).
  - Mount the IR sensor about 1 to 2 inches above a black surface.
  - Turn the potentiometer fully clockwise.
  - Slowly turn the potentiometer counterclockwise until the signal LED just turns off.
  - Do not change the distance between the IR sensor and the black surface now.
  - The signal LED will turn on when the sensor is above a white surface and off when it is above a black surface.



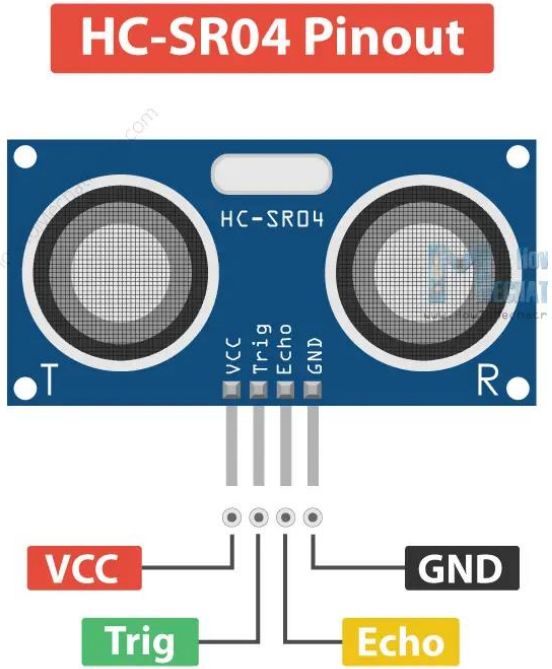
IR object sensor (sensor_object.py)

- Connect IR sensor to circuit is also very simple.
- We connect the Power to 5V power, GND to ground, and Output to one GPIO pin (use a resistor between to protect GPIO).
- Now we are able to read IR sensor's data with Python.



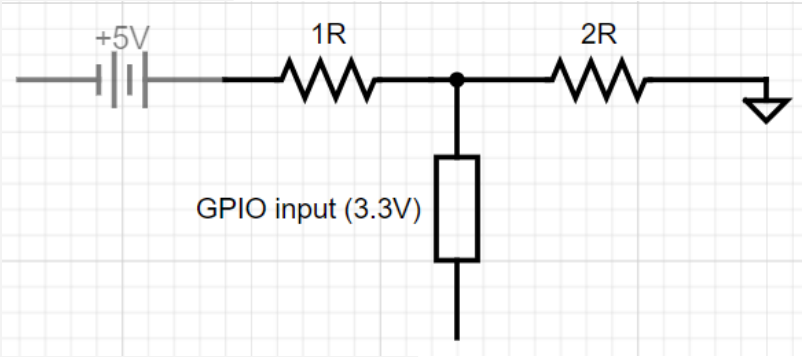
ultrasonic distance sensor (HC-SR04)

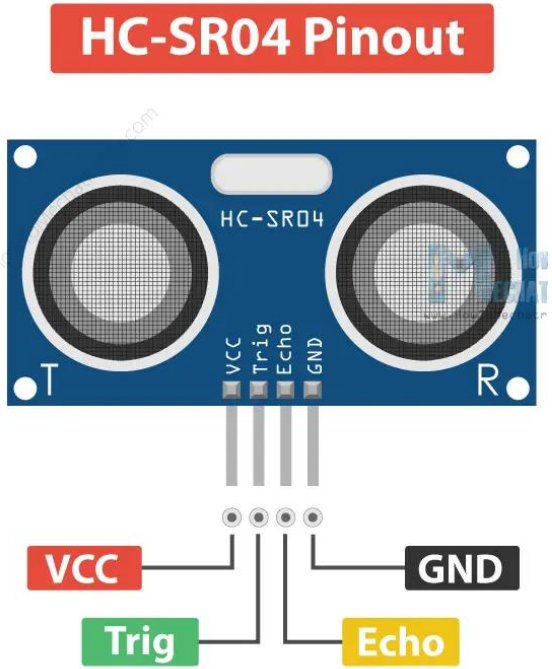
- An **Ultrasonic Sensor** is a sensor that send ultrasonic sound and hear it back. By calculating the sending time and hearing time, we could calculate the distance from the sensor to object in front up to 4 meters away.
- It could be used in robot car to avoid obstacles.
- It could also be used to simulate the parking alarm feature of car which beep more frequently or louder when car is closer to another object.



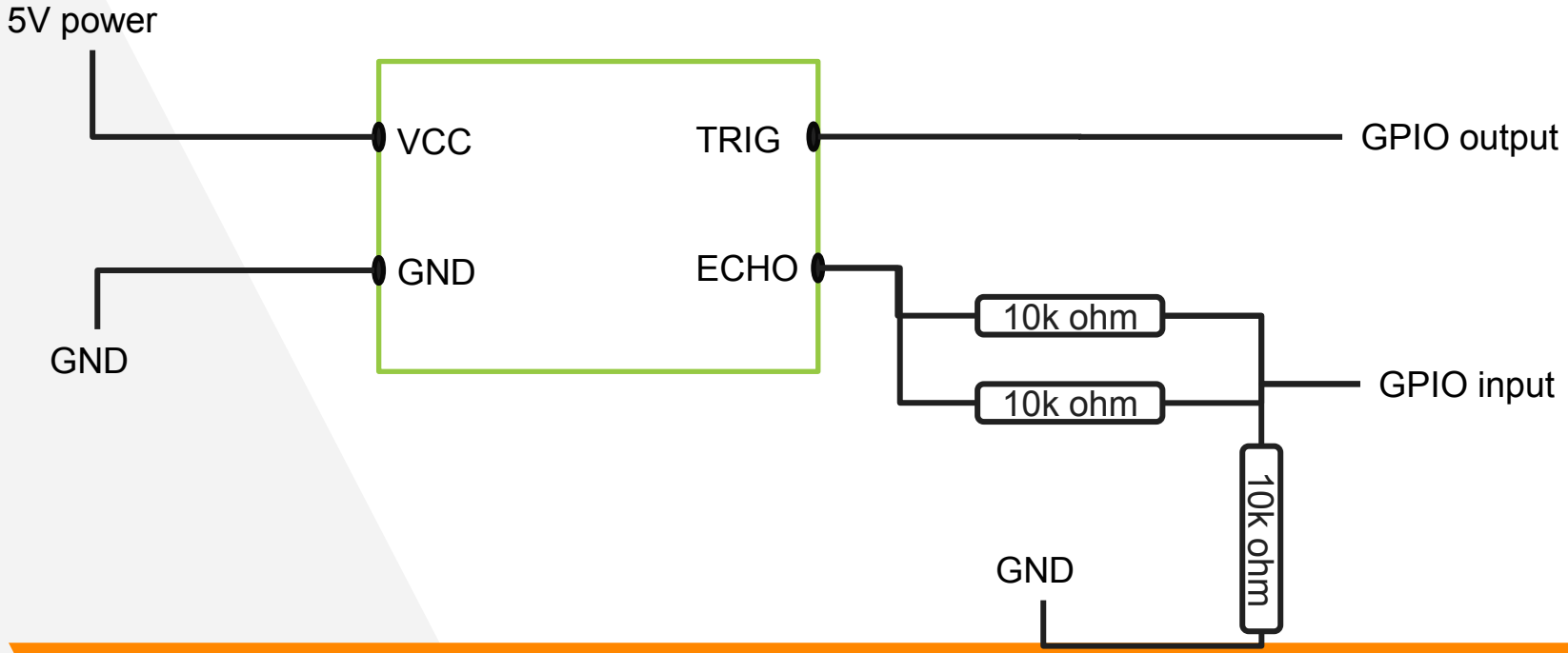
ultrasonic distance sensor (HC-SR04)

- VCC of ultrasonic sensor is 5V
- Output from Echo is also 5V.
  - So we can not directly connect it to GPIO pin (it could only accept no more than 3.3V).
  - But we could divide the voltage to make it work.



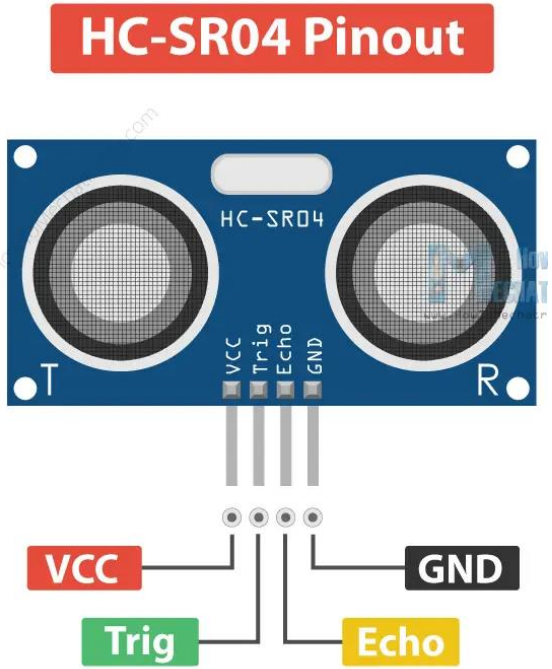


Circuit Diagram



ultrasonic distance sensor (sensor_distance.py)

- Now you are able to run the Python program.
- Read the comments carefully.



Learning outcomes

- Introduce Pulse-Width Modulation (PWM)
- Introduce how we use Raspberry Pi to send PWM signal
- Introduce DC motor and motor driver

Control the brightness of an LED

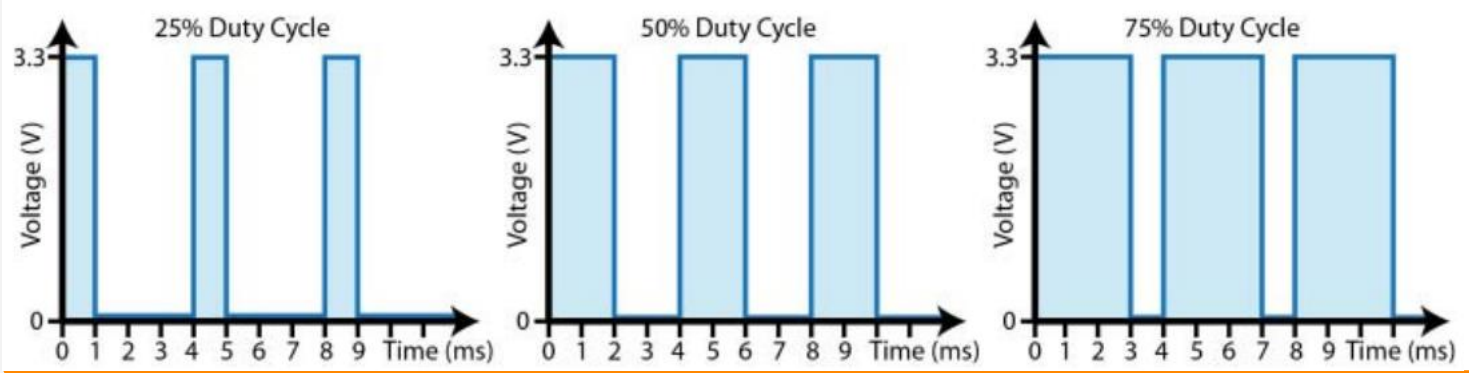
- When we were controlling LED with raspberry Pi, we never mentioned how to control its brightness.
- The first idea that we might come up with would be controlling it by changing the voltage going across the LED.
- However, as we mentioned, the voltage going cross LED would be considered as consistent. An LED is considered as a current-controlled device, where driving a current through the LED causes the forward voltage drop with a constant value. Therefore, trying to control an LED with a variable voltage will not work as you might expect.
- We could control the brightness by changing current, that is true. But the risk of damaging it is high, so we need to limit the current to a relative low value.
- Another idea is using pulsed current.

Control the brightness of an LED **using pulse signal**

- A pulse-width modulated (PWM) signal, is a current signal that rapidly switch between the state of High and Low.
- This could essentially rapidly switch the LED on and off.
- For example, if a rapid PWM signal is applied to the LED that is off for half of the time and on for half of the time, then the LED will appear to be only emitting about half of its regular operating condition light level.
- Our eyes don't see the individual changes if they are fast enough; they average over the light and dark interval to see a constant, but dimmer illumination.
- Also, our eyes would consider pulse-controlled LED to be brighter, even though the current stays the same.

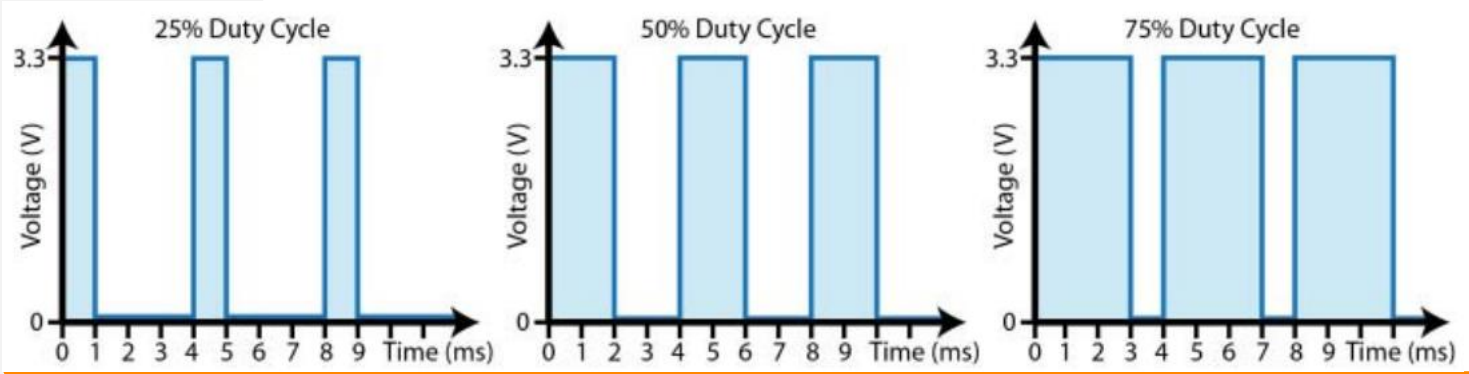
Pulse-Width Modulation (PWM)

- Below illustrates a PWM square wave signal at different duty cycles.
- The **duty cycle** is the percentage of time that the signal is high versus the time that the signal is low.
- In this example, a high is represented by a voltage of 3.3 V and a low by a voltage of 0 V. A duty cycle of 0% means that the signal is constantly low, and a duty cycle of 100% means that the signal is constantly high.



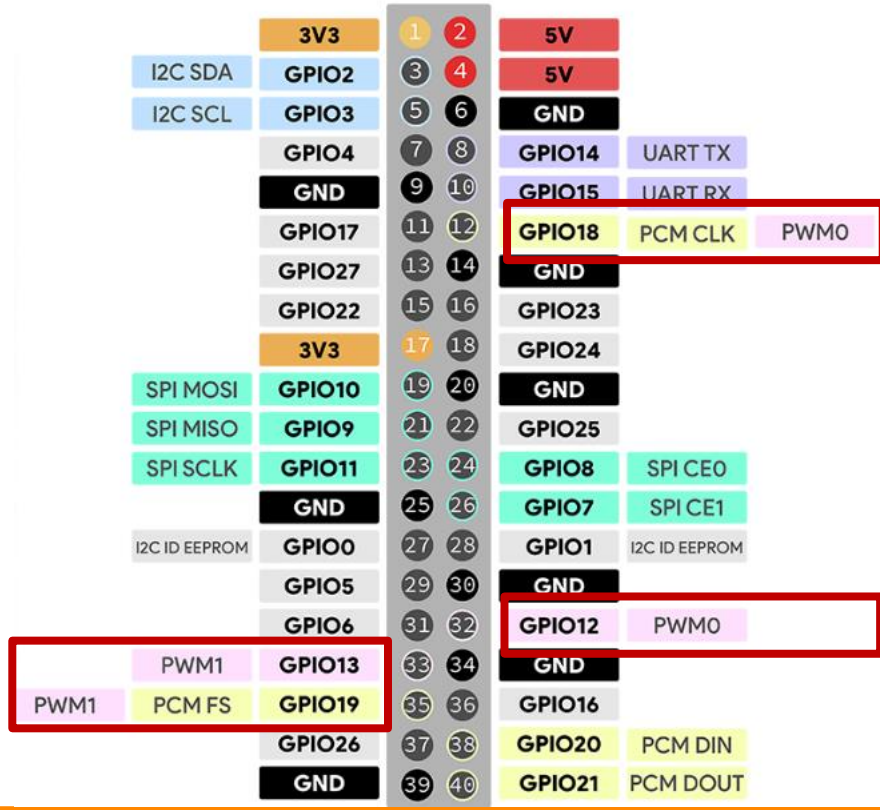
Pulse-Width Modulation (PWM)

- The **period (T)** of a repeating signal is the time it takes to complete a full cycle. In the example, the period of the signal in all three cases is 4 ms.
- The **frequency (f)** of a periodic signal describes how often a signal goes through a full cycle in a given time period.
- Therefore, for a signal with a period of 4 ms, it will cycle 250 times per second (1/0.004), which is 250 hertz (Hz).



software PWM and hardware PWM

- There are two ways of creating PWM signal in Raspberry Pi GPIO, Hardware and Software.
- **Hardware PWM** is produced by specific GPIOs as highlighted on the right.
  - The signal will be more stable, but it require other library or language to drive it (language C or PiGPIO library)
- **Software PWM** could be produced by any GPIOs, simulating pulse signal by turning it on or off. It will not be very stable.
  - This is what we will use and could be achieved by RPi.GPIO.



PWM in RPi.GPIO

- We first create a PWM instance using selected GPIO pin:
```
pwm = GPIO.PWM(PIN_NUM, frequency) # set initial frequency in Hz
```
- We could then start PWM by calling:
```
pwm.start(dc) # dc is the duty cycle with range [0,100]
```
- To change the frequency, we could call:
```
pwm.ChangeFrequency(f)   # f is the new frequency
```
- To change the frequency, we could call:
```
pwm.ChangeDutyCycle(d) # d is the new duty cycle
```
- To stop PWM, we just call:
```
pwm.stop()
```

Experience PWM with sound (buzzer_tune.py)

- Let's set up a very simple circuit.
- We are going to use sound to understand frequency and duty cycle.
- We connect 1 GPIO to 1 resistors, and another leg of resistor is connected to one Buzzer (+ side).
- Then we use wire to connect another side of Buzzer to the ground.
- Power on the Raspberry Pi, and run the program. You should enter the frequency and duty cycle in the command line.

use PWM to control LED (pwm_led.py)

- We have connected to LED many times, it is just the same this time.
- This program will continually change the duty cycle of signal, which simulates a breathing effect of LED light.
- Try modify the frequency and observe the differences.

DC motor

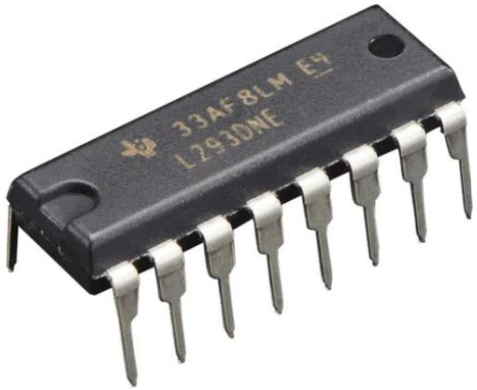
- DC motors are used in many applications, from toys to advanced robotics.
- They are ideal motors to use when continuous rotation is required, such as in the wheels of an electric vehicle. Typically, they have only two electrical terminals to which a voltage is applied. The speed of rotation and the direction of rota-tion can be controlled by varying this voltage.
- Most DC motors require more current than the RPi can supply, so we could use help of a motor driver and external power.

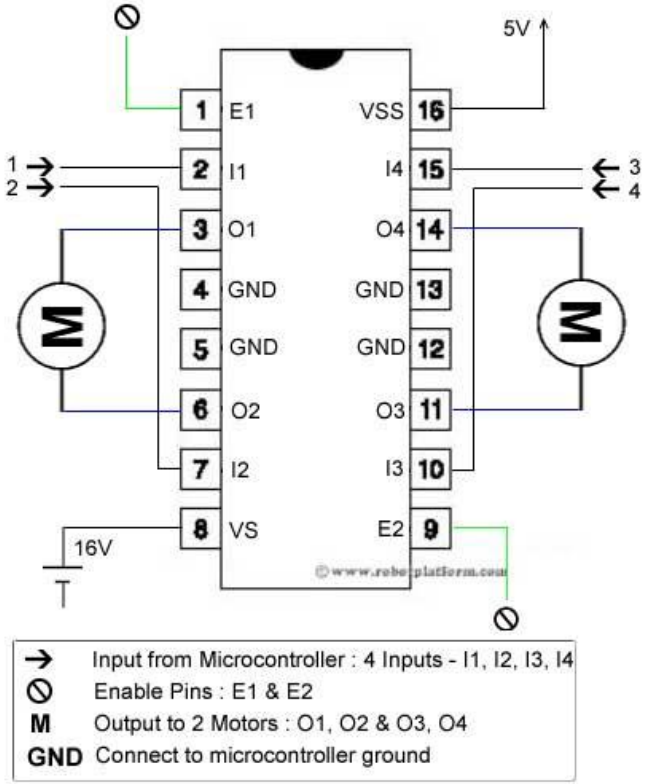
DC motor driver

- A **motor driver** transfer electrical energy from a source that could be of a given voltage, current at a certain frequency to an electrical output of desired voltage, current, frequency to the motor.
- It protects GPIO, it provides voltage and current higher than GPIO could produce, and it provides more control possibilities (like direction of the motor spinning).
- Usually, **we would need external power connected to driver.**

DC motor driver and diagram

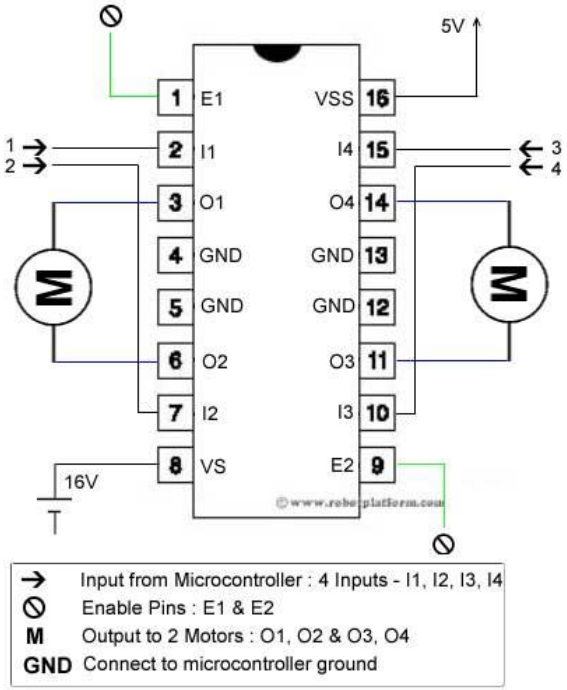
- A **motor driver or motor controller** is a microcontroller or processor we used to control a motor.
- We introduce **L293D** today (also because we have many of these).
- Right is the circuit diagram for L293D motor driver.





DC motor driver

- We could control 2 motors with one driver.
- Below is the table of PIN signals and corresponding behavior.
- Use GPIO to control these pins.



Pin 1	Pin 2	Pin 7	Function
High	High	Low	Turn counter-clockwise (Reverse)
High	Low	High	Turn clockwise (Forward)
High	High	High	Stop
High	Low	Low	Stop
Low	X	X	Stop

Motor and PWM

- PWM can be used to control the light level of LEDs, but it can also be used to control the speed of DC motors
- Typically, the frequency is in the kHz range for motor control. (I would say 5kHz to 20kHz is a safe range).
- Then we could just change the duty cycle to control its speed.