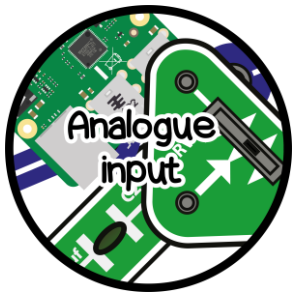
SNAP CIRCUIT – ANALOGUE INPUT, PWM OUTPUT



WARNING

Make sure you pay close attention to the details, as failure to do so could result in your Raspberry Pi suffering damage and invalidating the warranty. We cannot be held responsible for any damage resulted with incorrectly following the guide.

DESCRIPTION

The Raspberry Pi is very good at talking to many different electrical components with the help of digital input and outputs, serial and with Bluetooth and Wi-Fi, but the one thing that it does lack is analogue.

ANALOGUE? DIGITAL?

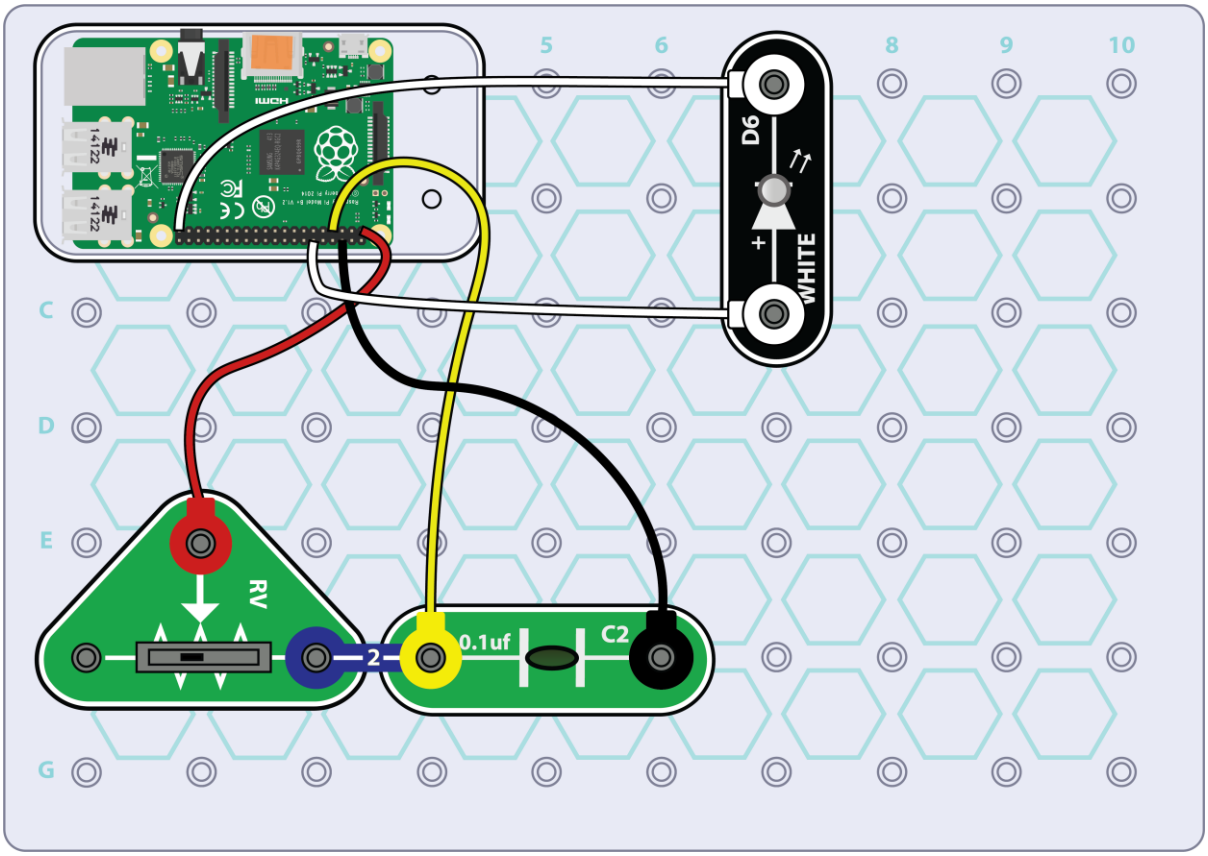
So what is the difference between analogue and digital? Which is better?

In short it comes down to the way it transfers its' information, digital is transmitted as either a 1 or a 0, nothing in-between and nothing else. 1 and 0's are normally seen as on and off – Like a standard light switch, where the light is either on or off.

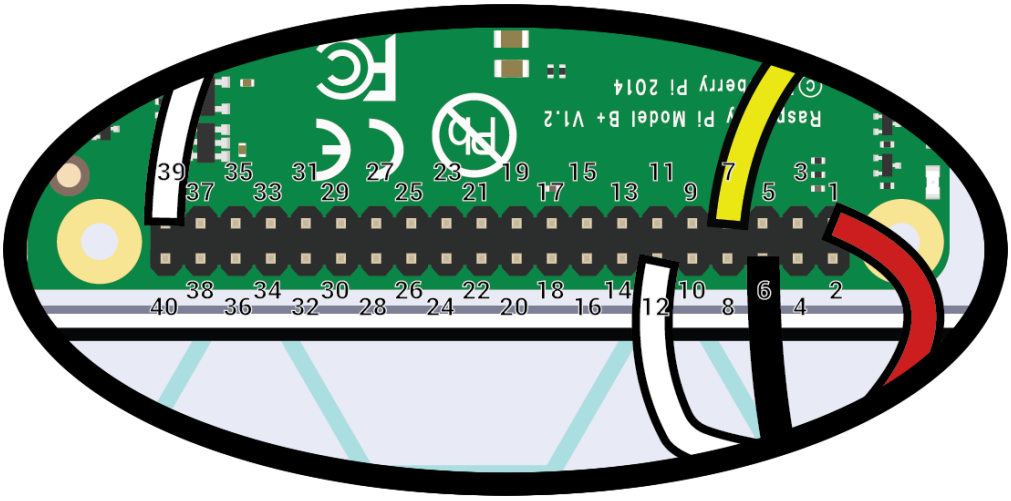
Analogue is replacing the simple on and off switch with a dimmer allowing you to control the brightness of the light between off and full brightness. As we are using an electrical voltage to work out the position of a variable resistor, this voltage can be anywhere between 0 volts and 3.3 volts.

As analogue can be a voltage in-between 0 and 3.3 volts, we need to convert this to digital, as the Raspberry Pi doesn't have analogue natively (without adding extra components), we can make using your Snap Circuits set a very rough guess circuit to do this.

THE CIRCUIT



Firstly place the C2 capacitor, with the RV variable resistor next to it and link these with the 2 snap bridge. Next place the White LED on the plate, pay close attention to the + marking on the LED, as placed the wrong way around can result in the LED not lighting up when it should.

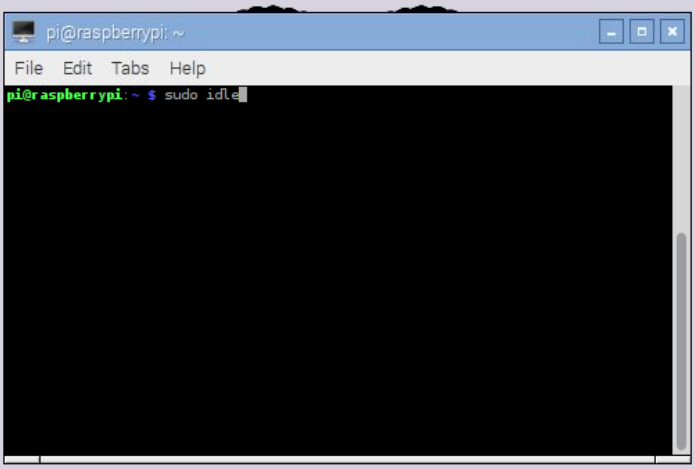


Now to connect the Snap to Pi wires between the Raspberry Pi and circuit.

Carefully count the number of pins on the Raspberry Pi before attaching any wires, be sure to double and triple check that the wire matches the correct pin!

THE PYTHON CODE

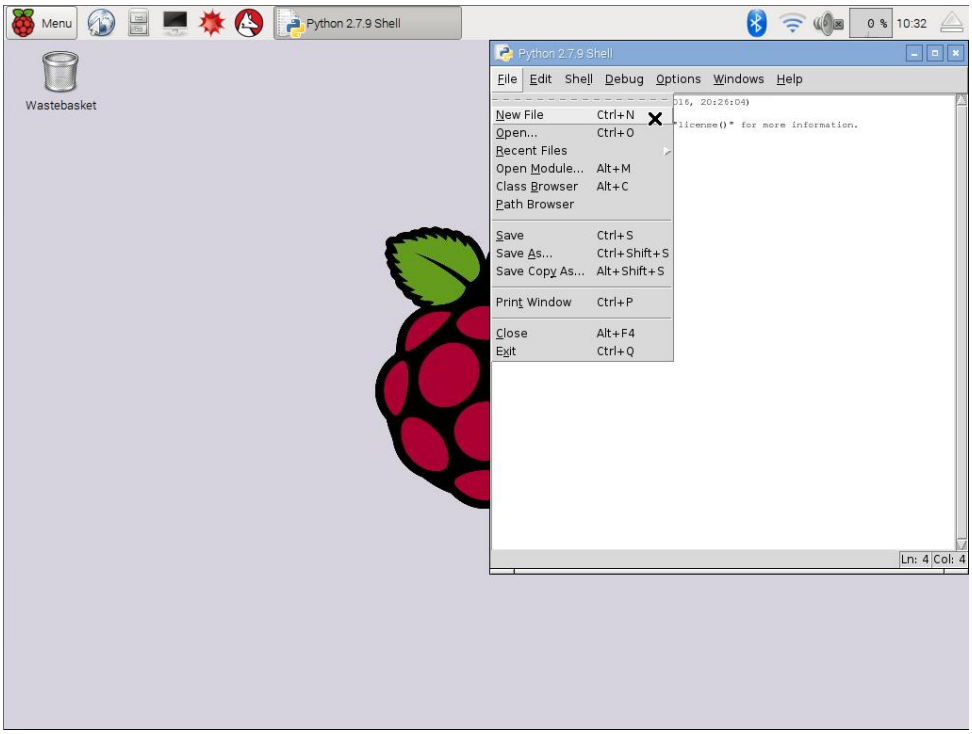
With the Raspberry Pi booted and showing the desktop, open Terminal by either clicking the Black computer screen icon on the task bar, or click Pi Menu (start button), followed by Accessories and then Terminal.

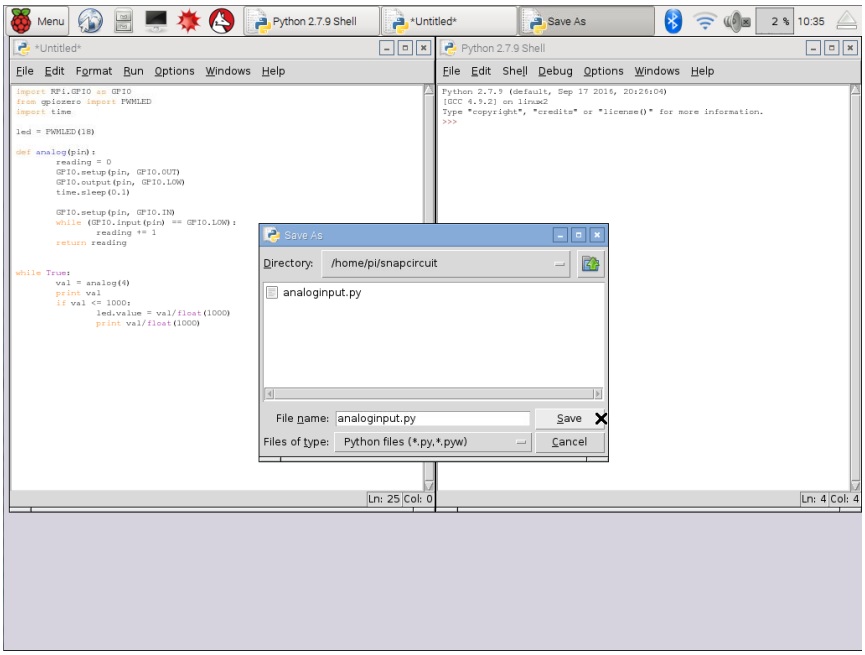


In this box type in sudo idle. This will open IDLE for typing our Python code with permission to control the GPIO pins.

Click File and then New File;

In the window that pops up, click *File*, *Save As* and give it a name to remember



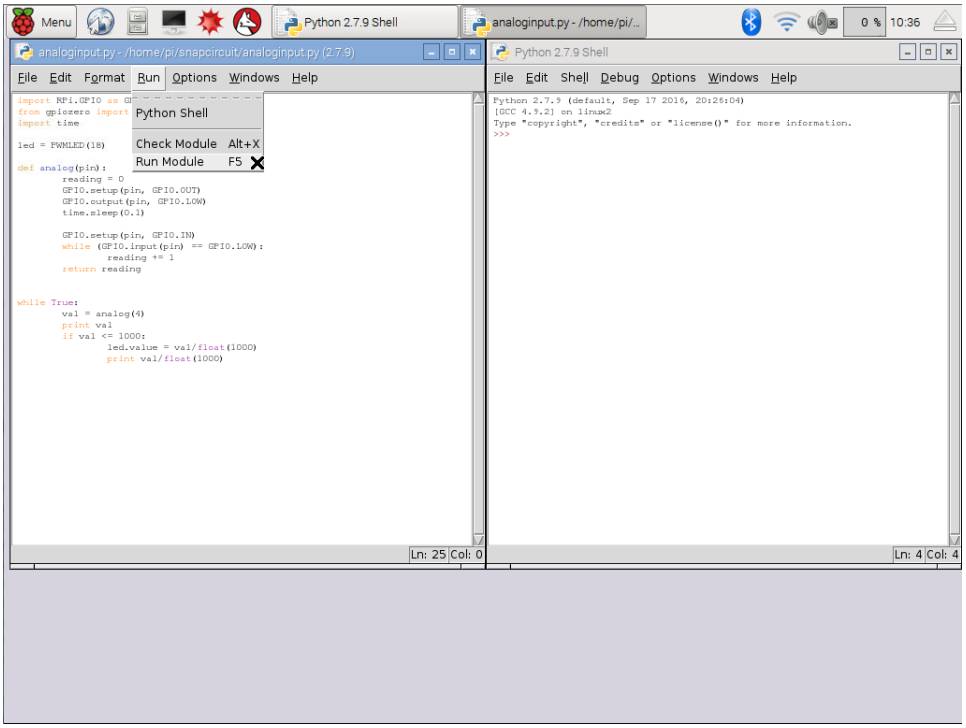


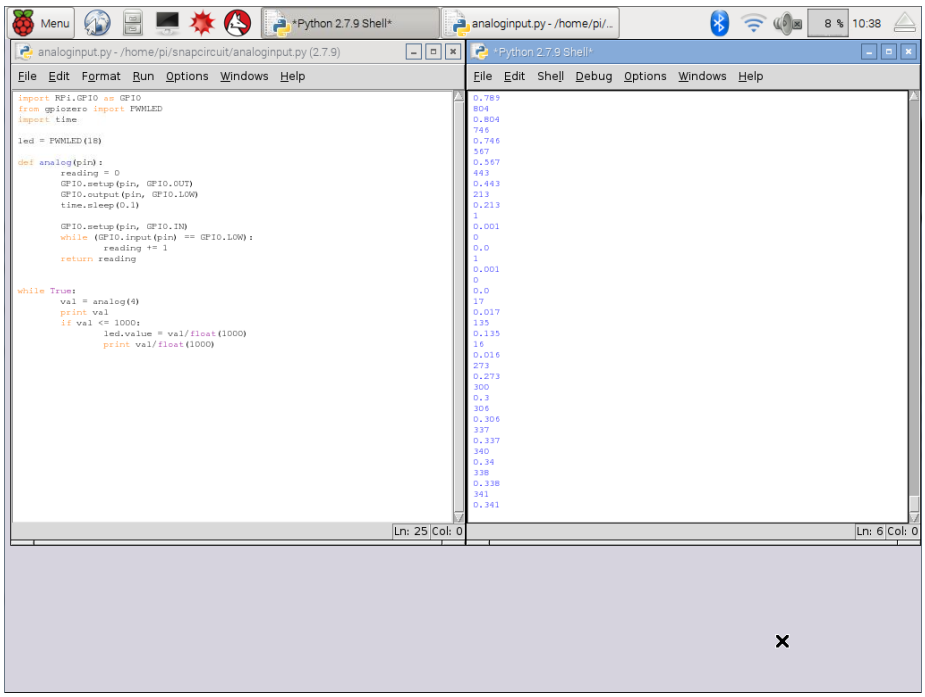
Type in the following code in to this window, pay attention to capitals and spacing;

```
import RPi.GPIO as GPIO
from gpiozero import PWMLED
import time

led = PWMLED(18)

def analog(pin):
        reading = 0
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, GPIO.LOW)
        time.sleep(0.1)

        GPIO.setup(pin, GPIO.IN)
        while (GPIO.input(pin) == GPIO.LOW):
                reading += 1
        return reading


while True:
        val = analog(4)
        print val
        if val <= 1000:
                led.value = val/float(1000)
                print val/float(1000)
```

Now double check the code to make sure no mistakes were made, otherwise the code might run and not give the expected results, or present you with an error.

When you are happy, click Run, and then *Run Module*.





Immediately the right hand screen will start to scroll with numbers, and hopefully the LED will light up. You should notice the numbers change in relation to the position of the variable resistor RV, lower numbers when the position is closest to the capacitor C2 and higher when the position is further from the capacitor.

WHAT IS HAPPENING

So what exactly is happening? As mentioned earlier, the Raspberry Pi doesn't have any analogue inputs, yet we managed to make one. This is down to exploiting the way the capacitor work;

At the start of def analog(pin):, it forces the capacitor to discharge by setting both pins to 0v to get it ready to start counting;

```
        reading = 0
        GPIO.setup(pin, GPIO.OUT)
        GPIO.output(pin, GPIO.LOW)
```

This is followed by a short delay of 0.1 seconds to get ready;

```
        time.sleep(0.1)
```

The output is then changed to an input so it can read what the capacitor is doing, and begin counting to give a value when it is nearly full (about 66% charged) from the variable resistor.