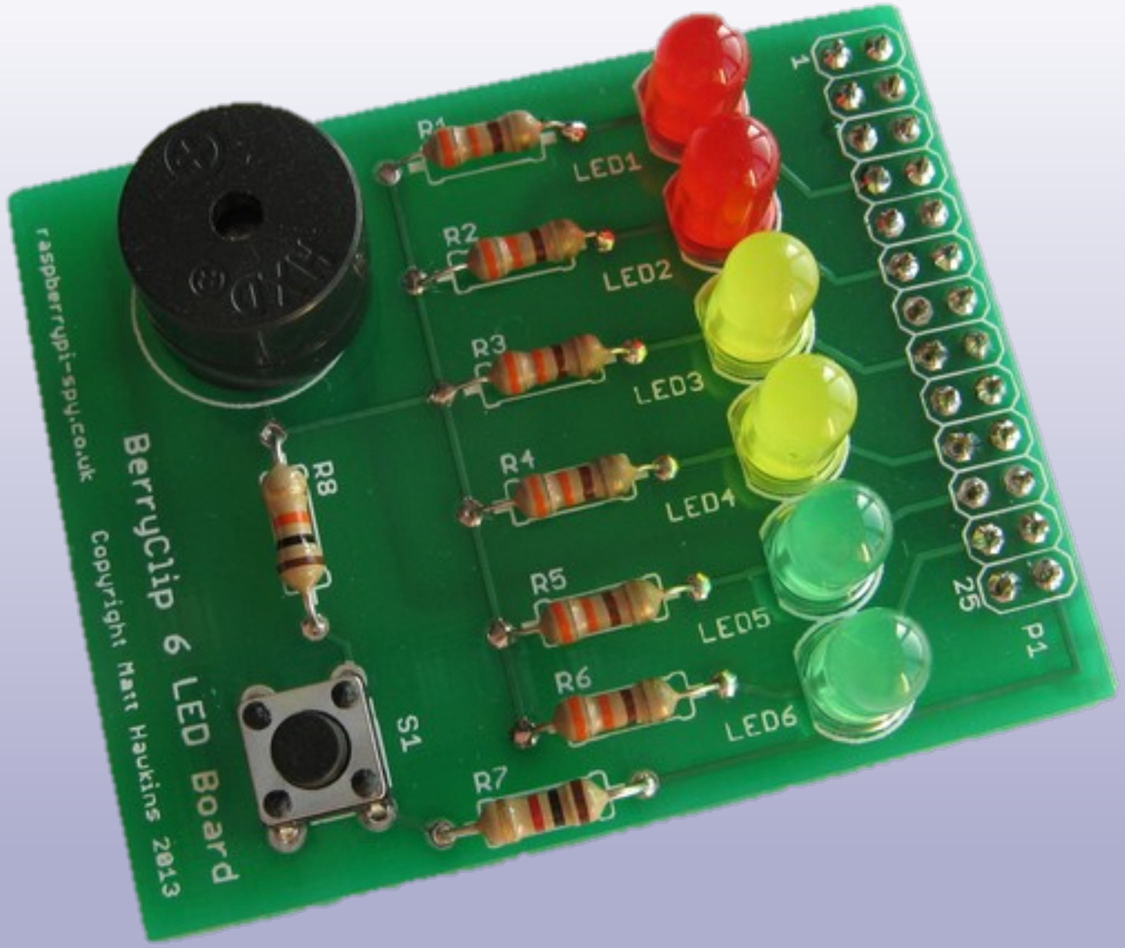
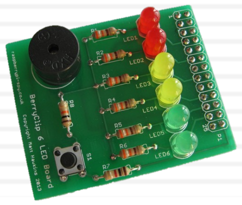


PROCAT

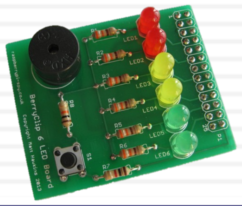
PROSPECTS COLLEGE OF ADVANCED TECHNOLOGY

Silvia Tinena Coris
Electronics & Automation Course Leader

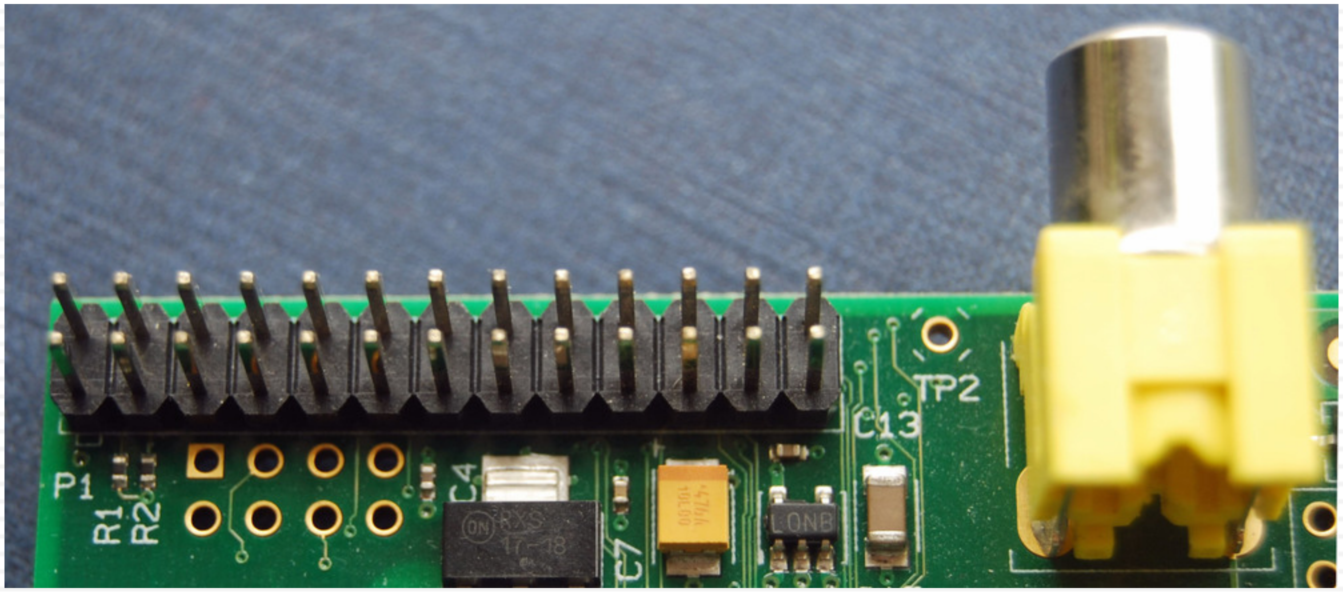
Table of contents



What does GPIO mean?

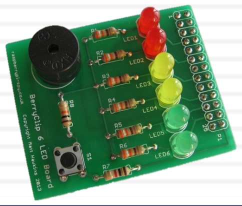


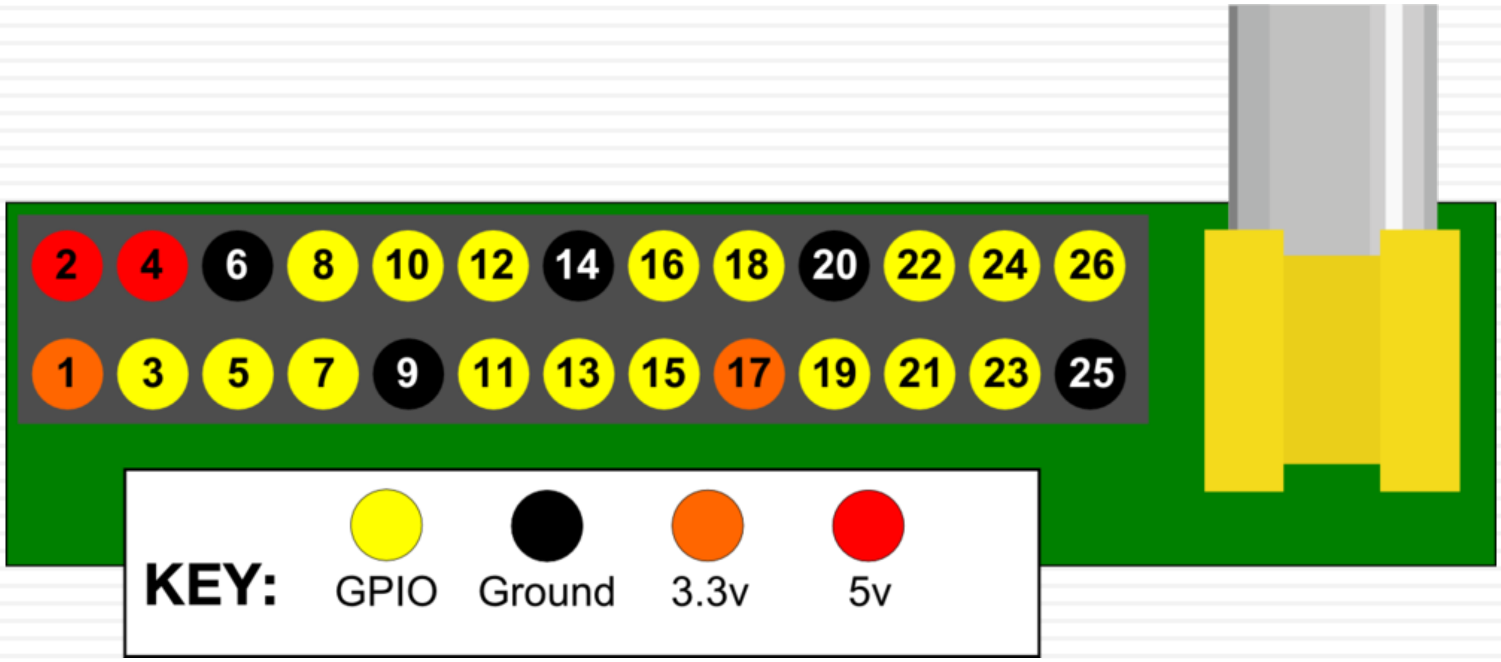
- GPIO = General Purpose Input/Output pins



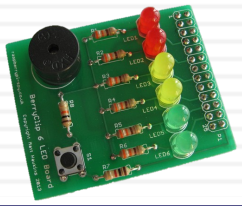
- Physical interface between the Pi and the outside world

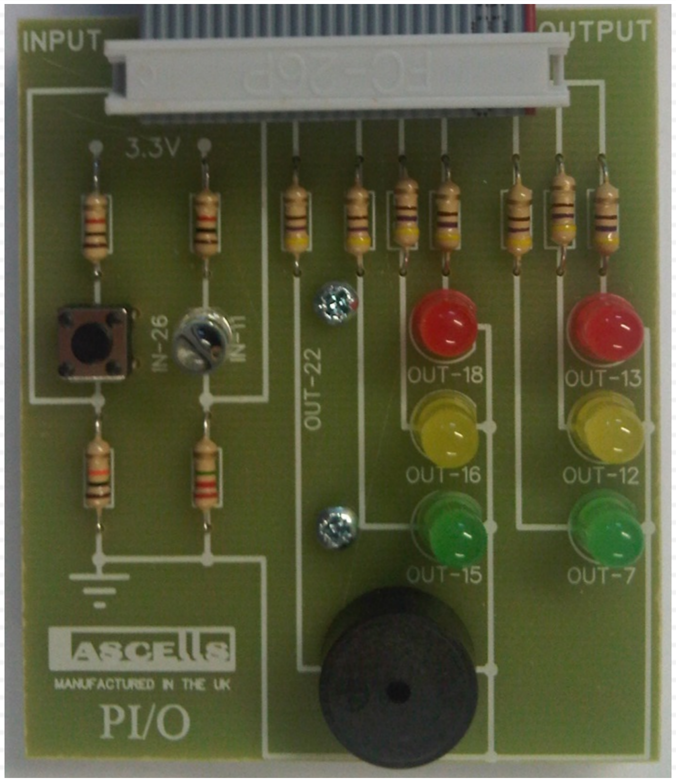
Connecting the GPIO





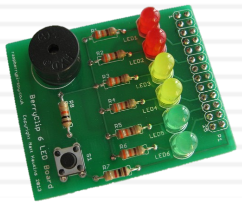
Our BerryClip





- Green LED:
  - Outputs 7 and 15
- Yellow LED:
  - Outputs 12 and 16
- Red LED:
  - Outputs 13 and 18
- Buzzer:
  - Output 22
- Switch:
  - Input 26

First steps



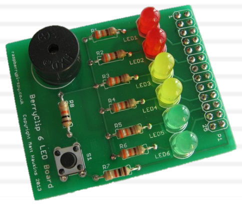
- Make sure that the Raspberry Pi is switched off before connecting the GPIO Board to the GPIO header pins

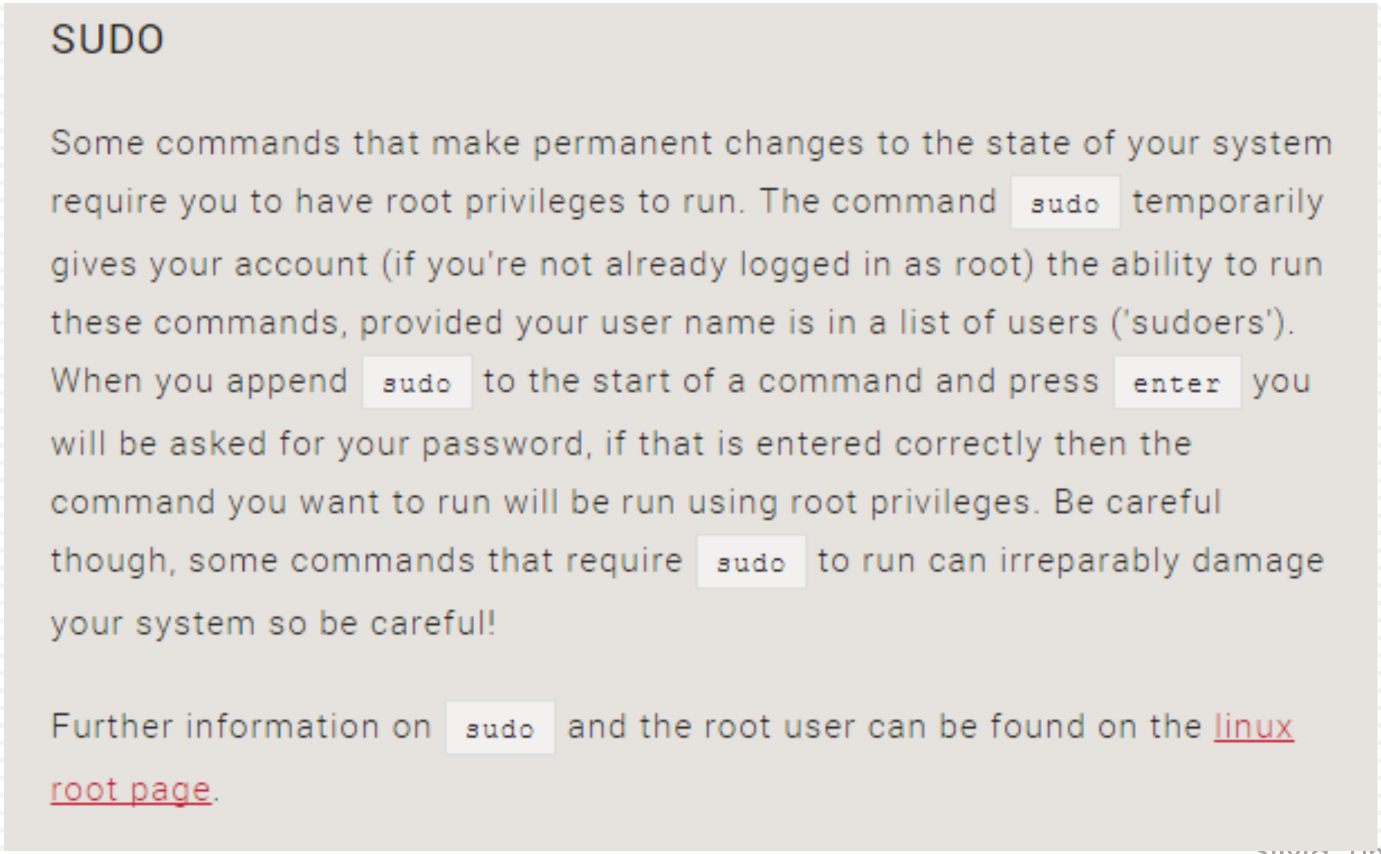
- After that, switch the Raspberry Pi on

- Using the Terminal (Menu/Accessories), type in: **sudo idle**
  - This means that you are a super user and thus you can access the GPIO
  - This will open the IDLE (the Python Shell)

- Write the code as usual

Super user (sudo)



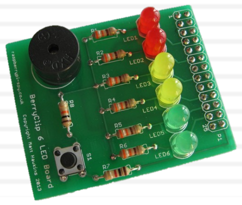


SUDO

Some commands that make permanent changes to the state of your system require you to have root privileges to run. The command `sudo` temporarily gives your account (if you're not already logged in as root) the ability to run these commands, provided your user name is in a list of users ('sudoers'). When you append `sudo` to the start of a command and press `enter` you will be asked for your password, if that is entered correctly then the command you want to run will be run using root privileges. Be careful though, some commands that require `sudo` to run can irreparably damage your system so be careful!

Further information on `sudo` and the root user can be found on the linux root page.

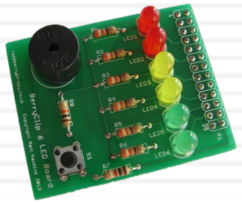
Important things to know



- There are some command lines that you should ALWAYS type before starting programming the GPIO board.
  - `import RPi.GPIO as GPIO`
    - This imports the GPIO library
  - `GPIO.setmode(GPIO.BOARD)`
    - This uses the board pin numbering

3. Program 1: Turning an LED on

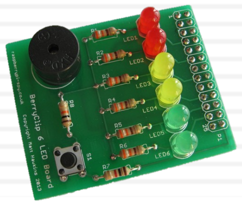
Turning an LED on



- Start with the essential commands
- Choose an LED to turn on and pay attention to the pin that it is connected to
- Decide whether it must be:
  - An input or an output
  - True (on) or False (off)
- Configure it and write this down on your program!

- `GPIO.setup(pin number, GPIO.IN or OUT)`
  - This sets the GPIO **pin** to **OUT**put
- `GPIO.output(pin number,True or False)`
  - This turns the output **pin** on (**True**) or off (**False**)

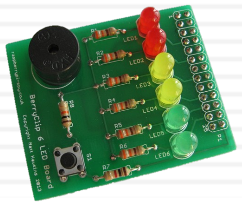
Turning an LED on – solution



- `import RPi.GPIO as GPIO`
  - This imports the GPIO library
- `GPIO.setmode(GPIO.BOARD)`
  - This uses the board pin numbering
- `GPIO.setup(7, GPIO.OUT)`
  - This sets the GPIO pin 7 to **OUT**put
- `GPIO.output(7,True)`
  - This gives a ‘1’ (**True**) to the output pin 7

4. Program 2: Flashing an LED

Flashing an LED



- Level 1:
  - Extend the last program so that:
    - The LED is ON for 1 second…
    - …and OFF for another second

To work with time we need to import the time library:

- `import time`

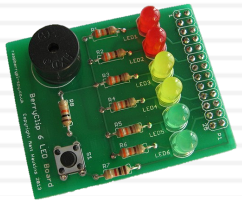
To make a never-ending loop we need to create a loop with a condition that is always true:

- `while(True):`

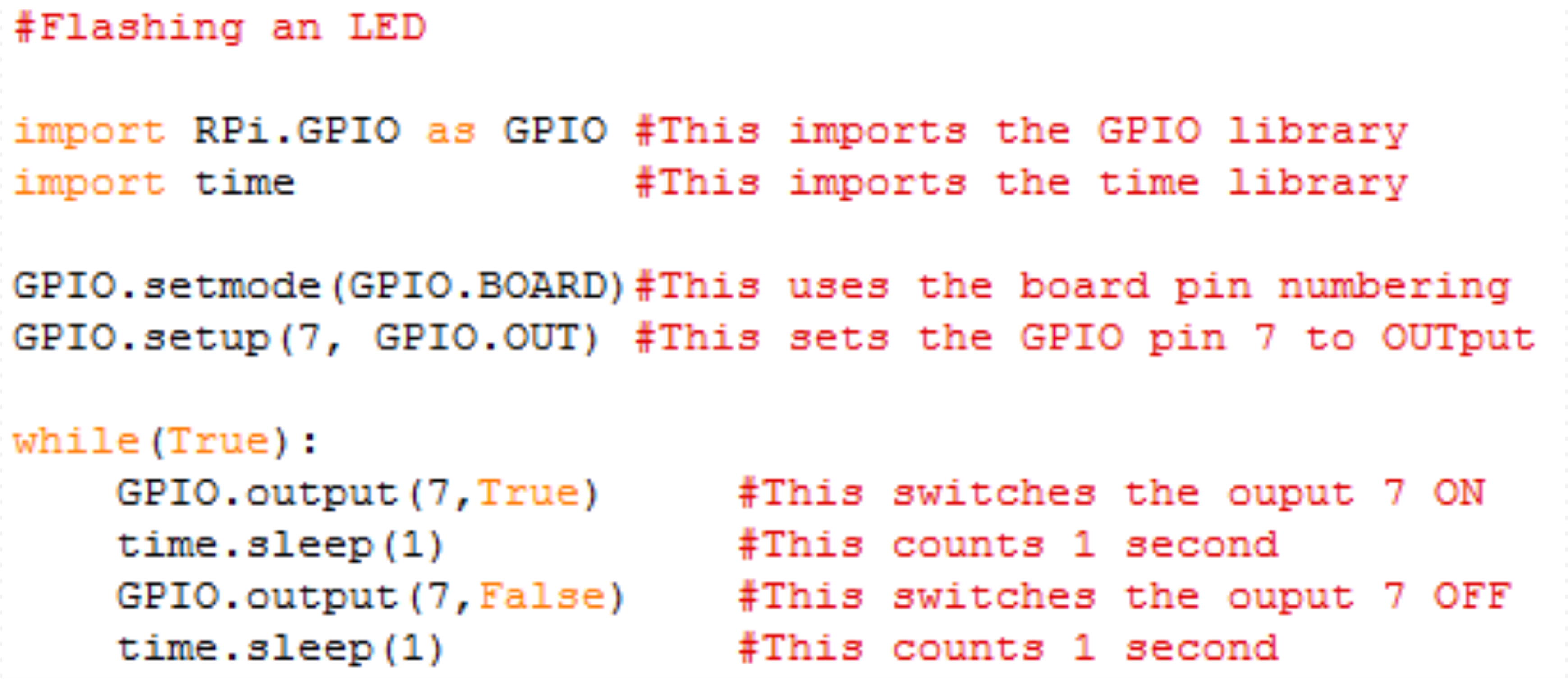
To time an action:

- `time.sleep(number of seconds)`

Flashing an LED – solution

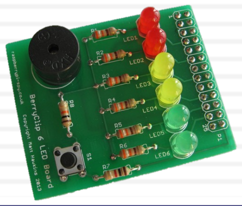


- Level 1:



```
#Flashing an LED

import RPi.GPIO as GPIO #This imports the GPIO library
import time             #This imports the time library

GPIO.setmode(GPIO.BOARD)#This uses the board pin numbering
GPIO.setup(7, GPIO.OUT) #This sets the GPIO pin 7 to OUTput

while(True):
    GPIO.output(7,True)     #This switches the ouput 7 ON
    time.sleep(1)           #This counts 1 second
    GPIO.output(7,False)    #This switches the ouput 7 OFF
    time.sleep(1)           #This counts 1 second
```

Flashing an LED

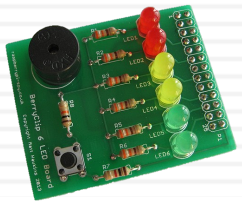


- Level 2:
  - Now the LED won't keep blinking forever, but the user will be able to choose how many times it has to blink
  - Print 'done' the sequence is completed

- Level 3:
  - The user will also be able to choose the length of the blink

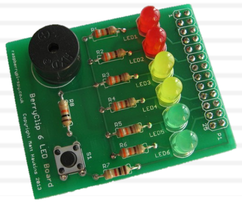
- Hint: use a routine

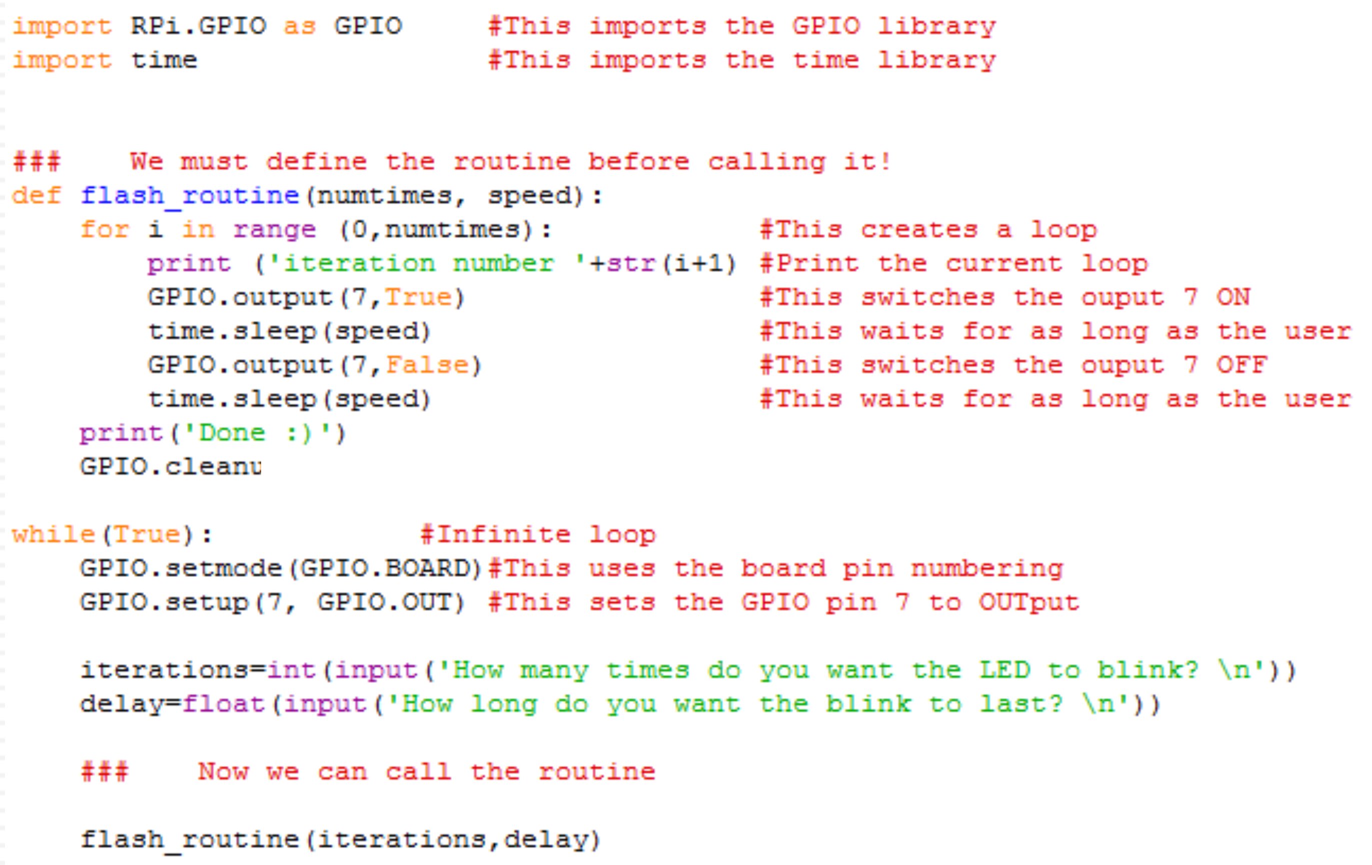
Flashing an LED – a bit of help



- The routine will have two input variables:
  - The number of times that we want the LED to flash
  - The length of each blink, in seconds
- Remember that the names of the variables INSIDE the routine don’t have to be the same as those OUTSIDE it
  - In fact, it is better if they are different to avoid confusion

Flashing an LED – solution

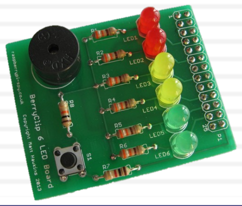




```
import RPi.GPIO as GPIO      #This imports the GPIO library
import time                  #This imports the time library


###    We must define the routine before calling it!
def flash_routine(numtimes, speed):
    for i in range (0,numtimes):            #This creates a loop
        print ('iteration number '+str(i+1) #Print the current loop
        GPIO.output(7,True)                 #This switches the ouput 7 ON
        time.sleep(speed)                   #This waits for as long as the user
        GPIO.output(7,False)                #This switches the ouput 7 OFF
        time.sleep(speed)                   #This waits for as long as the user
    print('Done :)')
    GPIO.cleanu

while(True):                 #Infinite loop
    GPIO.setmode(GPIO.BOARD)#This uses the board pin numbering
    GPIO.setup(7, GPIO.OUT) #This sets the GPIO pin 7 to OUTput

    iterations=int(input('How many times do you want the LED to blink? \n'))
    delay=float(input('How long do you want the blink to last? \n'))

    ###    Now we can call the routine

    flash_routine(iterations,delay)
```

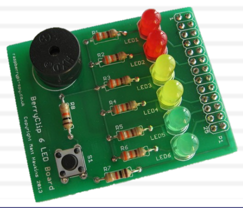
5. Program 3: Using a switch as an input

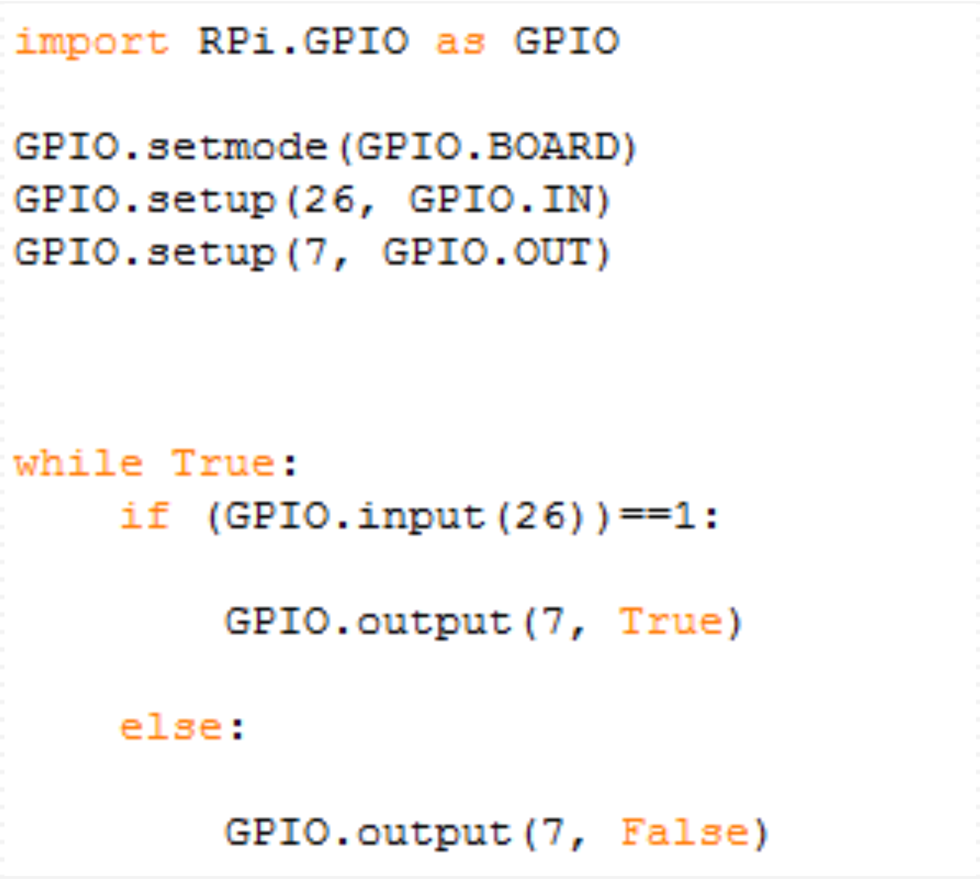
Using a switch as an input



- <u>Level 1:</u>
  - An LED should be ON as long as the push button is pressed.
- <u>Level 2:</u>
  - The buzzer should be ON as long as the LED is ON.
- <u>Level 3:</u>
  - The LED and the buzzer should ‘flash’ 5 times after the push button is pressed once.

Using a switch as an input – solutions

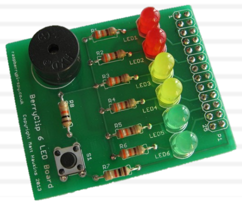




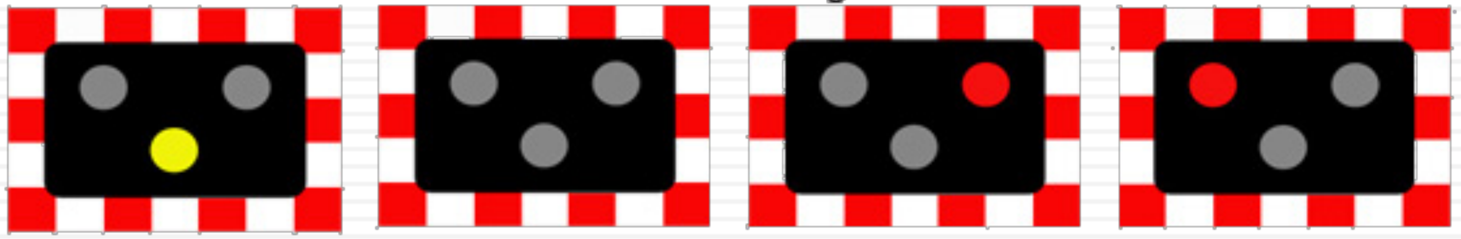
```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(26, GPIO.IN)
GPIO.setup(7, GPIO.OUT)



while True:
    if (GPIO.input(26))==1:

        GPIO.output(7, True)

    else:

        GPIO.output(7, False)
```

6. Program 4: Level Crossing

Level crossing



- Now, create a level crossing which follows the sequence below:



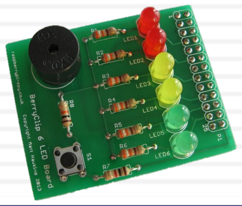
Amber LED on	All off	Right LED's on	Left LED's on
Amber LED turns on	All LED's off	Right LED's on	Left LED's on
On for 3 seconds	Off for 50ms	On for 50ms	On for 50ms

Left and right LED's continue to flash on and of until the cancel reed switch is triggered.

Instead of the reed switch… use the push button! When it is pressed, the flashing should stop

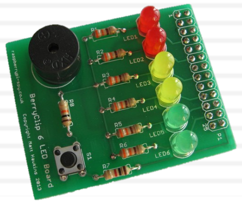
7. Program 5: Counter with LEDs

Counter with LEDs – easy version



- You should create a counter with LEDs
- Once the push button is pressed, it should count up to 3 (red, yellow, green, with the three right-hand sided LEDs)
- There must be a 1 second delay between the counts
- After it has reached 3, it should wait 1 second and turn everything off, waiting again for the push button to be pressed

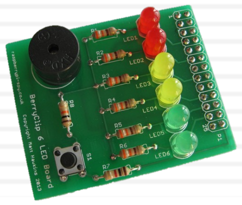
Counter with LEDs – medium version



- Now:
  - Use both rows of LEDs at the same time
  - Add the buzzer after the green LEDs are on
  - Display the following messages:

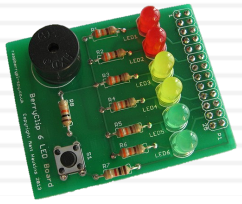
When red:	ready
When yellow:	steady
When green:	GO!

Counter with LEDs – difficult version



- Now, the counter will not be timed, but will depend on the pushes of the button
- The counter will be up to 6 (so use both rows of LEDs independently)
- The sequence must be:
  - Green, green, yellow, yellow, red, red, everything off
- Remember that, initially, all the LEDs are off

Counter with LEDs – solution



```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)

##We must define ALL the I/O
GPIO.setup(26, GPIO.IN)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)

counter=0
```

```
while True:
    if (GPIO.input(26))==1:
        counter = counter +1
        time.sleep(0.5)             #To prevent bouncing
        if counter>=1:
            GPIO.output(7, True)     #Illuminate one green LED at the first switch
        if counter >=2:
            GPIO.output(15, True)    #Illuminate the second green LED
        if counter >=3:
            GPIO.output(12, True)    #yellow
        if counter >=4:
            GPIO.output(16, True)    #yellow
        if counter >=5:
            GPIO.output(13, True)    #red
        if counter >=6:
            GPIO.output(18, True)    #red
        if counter >=7:
            counter=0                #re-initialisation of the counter
            GPIO.output(7, False)
            GPIO.output(15, False)
            GPIO.output(12, False)
            GPIO.output(16, False)
            GPIO.output(13, False)
            GPIO.output(18, False)
```

8. Programs 6 & 7

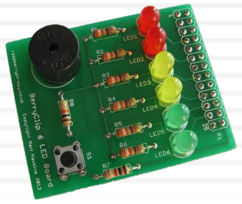
Assignment 2 workshop

Choose two out of the three applications that you have below:

- UK traffic light system
- 3-bit binary counter, UP or DOWN (or UP AND DOWN!)
- Roulette

9. Interesting Links

Interesting Links



- http://pi.gadgetoid.com/pinout
- http://www.raspberrypi.org/documentation/usage/gpio/
- http://www.raspberrypi-spy.co.uk/berryclip-6-led-add-on-board/berryclip-6-led-add-on-board-instructions/