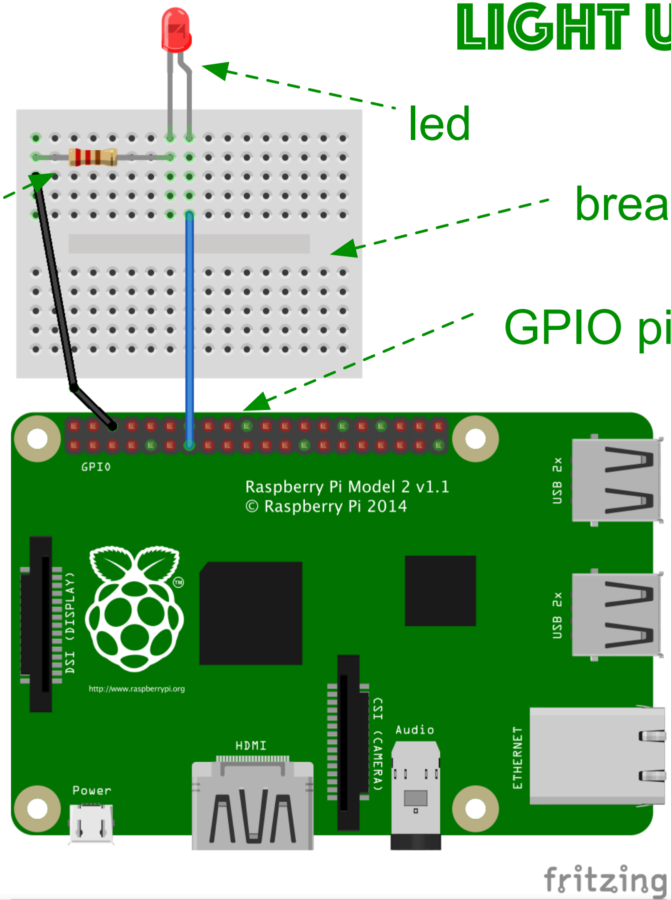
CoderDojo HAM

LIGHT UP AN LED

SCRATCH



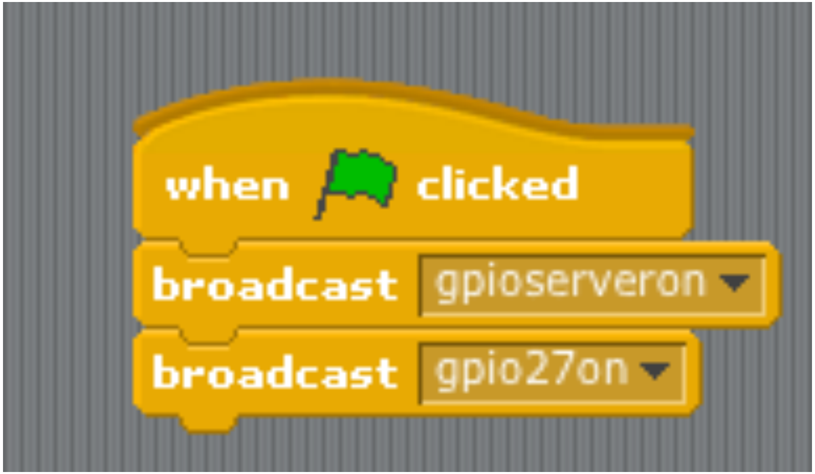
led

breadboard

resistor

GPIO pins

Jumper wire



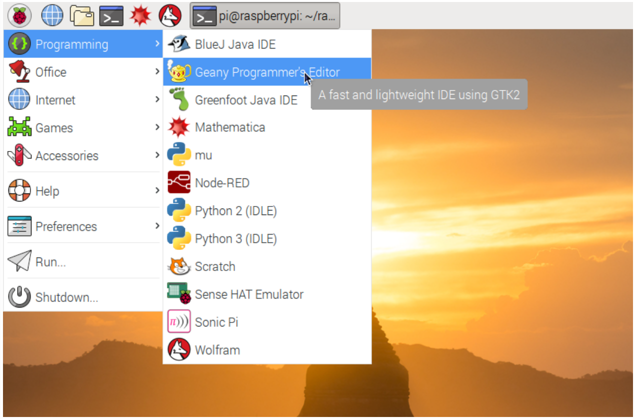
```
when [flag] clicked
broadcast gpioserveron
broadcast gpio27on
```

How do you think you can turn the led off?

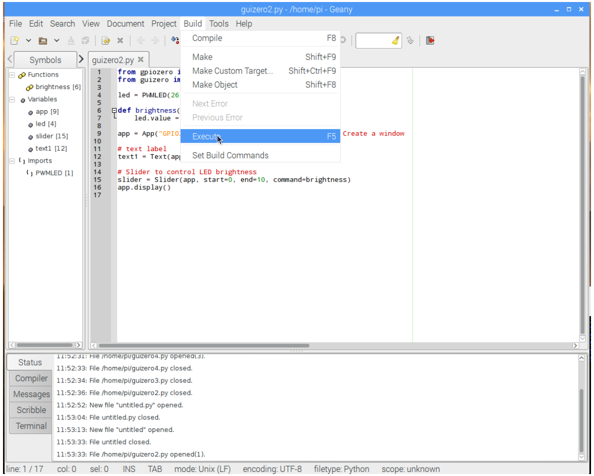
Can you make the led flash on and off?

RUNNING python™

1 Select the Geany app:



2 Type your code in the new window that opens (you don't have to include the #comments in red)



```
from gpiozero i
from guizero im

led = PWMLED(26

def brightness(
    led.value =

app = App("GPIO                              Create a window

# text label
text1 = Text(ap

# Slider to control LED brightness
slider = Slider(app, start=0, end=10, command=brightness)
app.display()
```

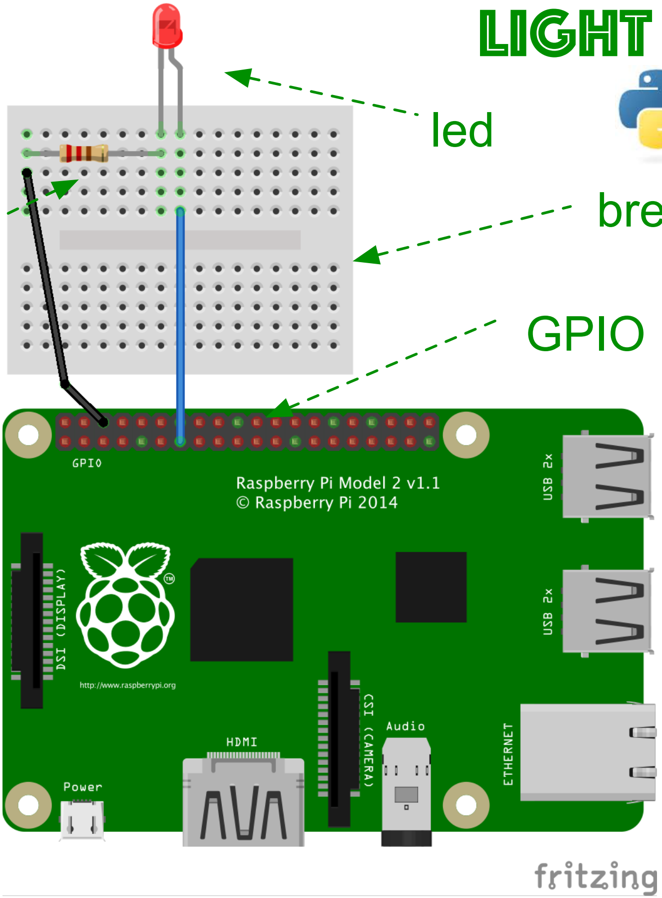
```
11:52:31: File /home/pi/guizero4.py opened(3).
11:52:33: File /home/pi/guizero4.py closed.
11:52:34: File /home/pi/guizero3.py closed.
11:52:36: File /home/pi/guizero2.py closed.
11:52:52: New file "untitled.py" opened.
11:53:04: File untitled.py closed.
11:53:13: New file "untitled" opened.
11:53:33: File untitled closed.
11:53:33: File /home/pi/guizero2.py opened(1).
```

line: 1 / 17 col: 0 sel: 0 INS TAB mode: Unix (LF) encoding: UTF-8 filetype: Python scope: unknown

3 Select 'Build' -> 'Execute' or press F5 to run the code. Don't forget you save your work as a .py file (e.g mycode.py)

2

LIGHT UP AN LED



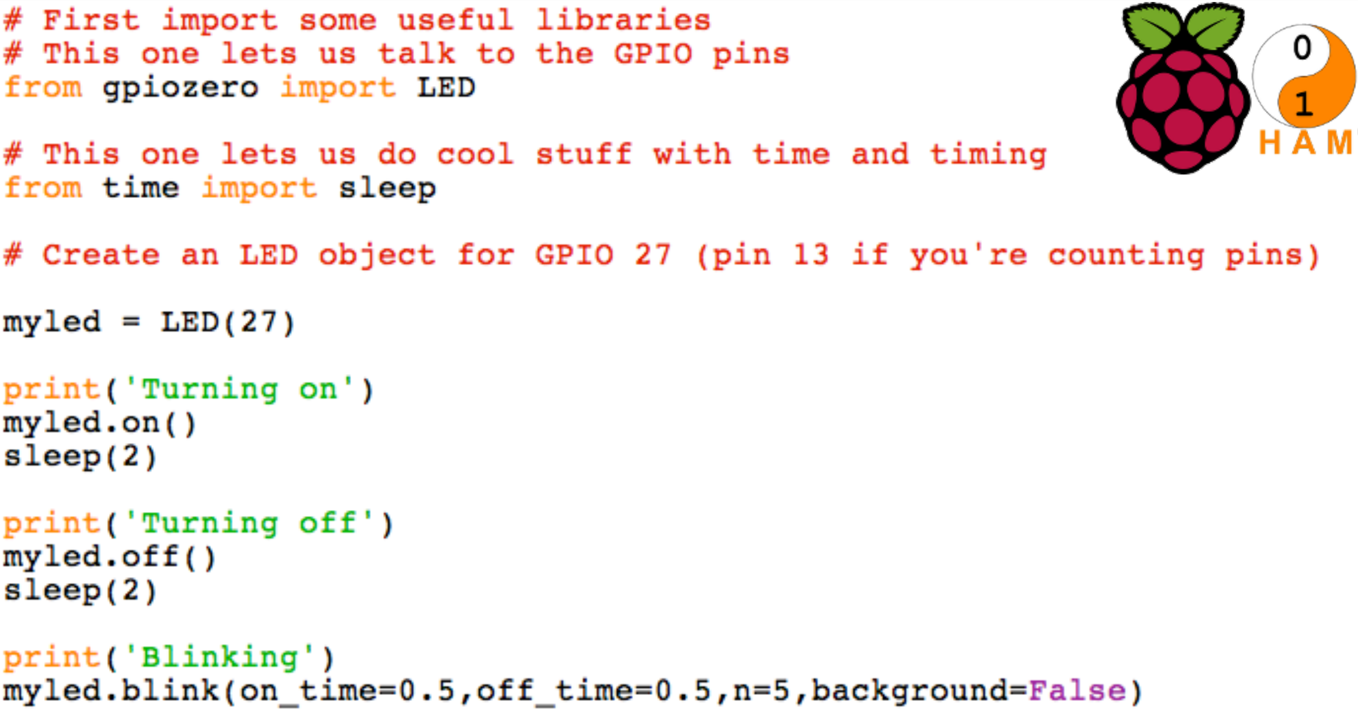
led

breadboard

resistor

GPIO pins

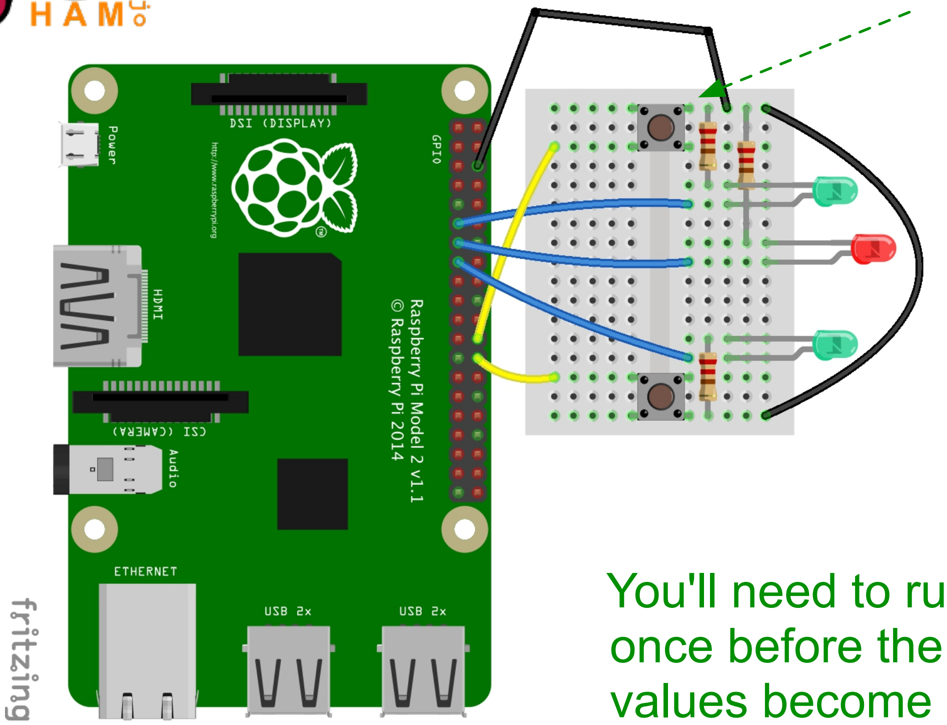
Jumper wire



```
# First import some useful libraries
# This one lets us talk to the GPIO pins
from gpiozero import LED

# This one lets us do cool stuff with time and timing
from time import sleep

# Create an LED object for GPIO 27 (pin 13 if you're counting pins)

myled = LED(27)

print('Turning on')
myled.on()
sleep(2)

print('Turning off')
myled.off()
sleep(2)

print('Blinking')
myled.blink(on_time=0.5,off_time=0.5,n=5,background=False)
```

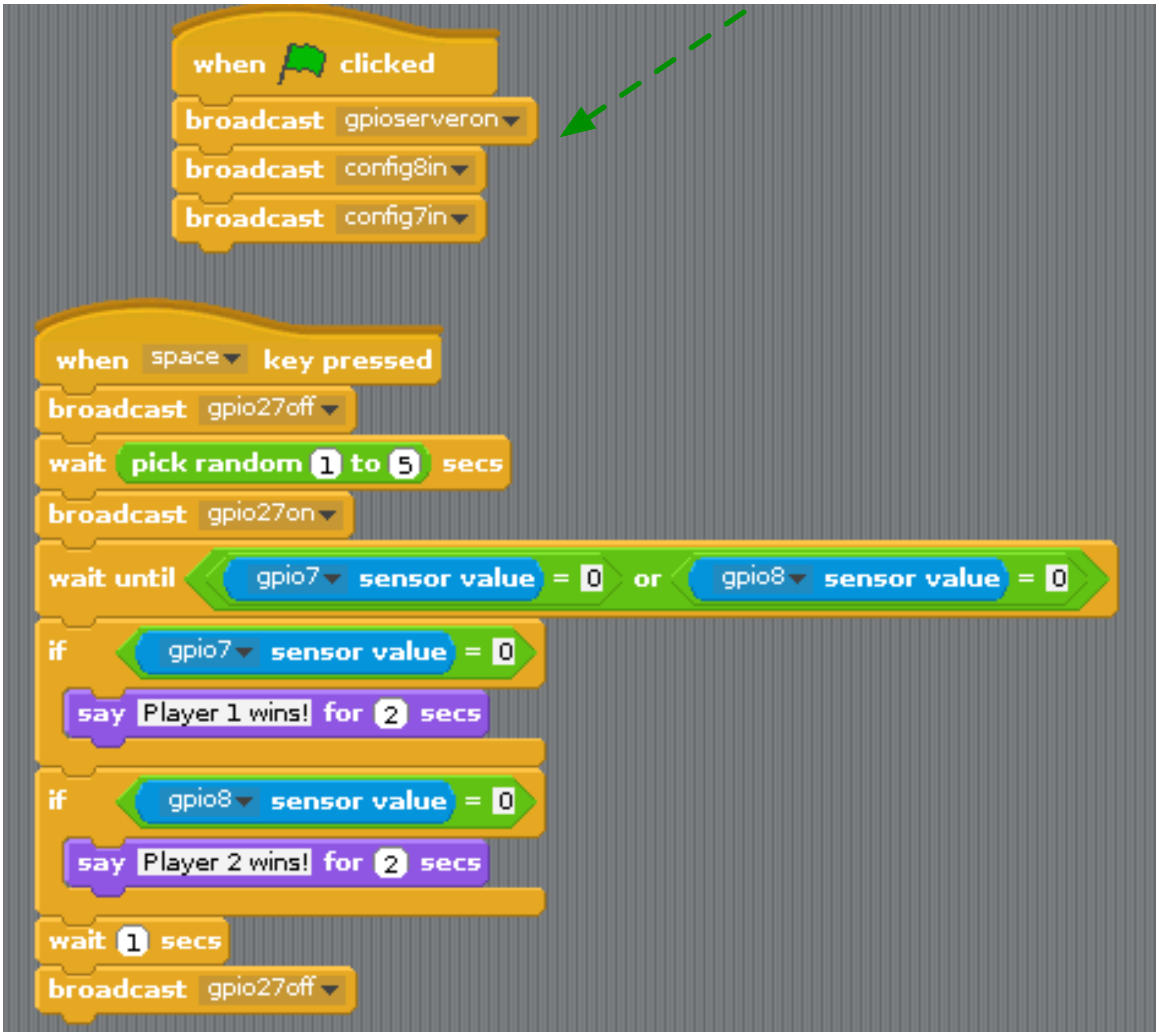
Can you change the speed and number of flashes?

REACTION GAME WITH SCRATCH



button

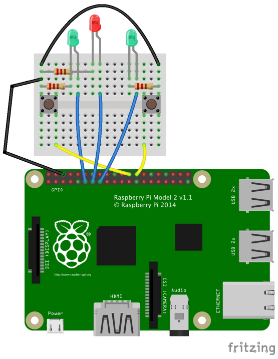
You'll need to run this block once before the gpis sensor values become available



```
when ⚑ clicked
broadcast gpioserveron
broadcast config8in
broadcast config7in

when space key pressed
broadcast gpio27off
wait pick random 1 to 5 secs
broadcast gpio27on
wait until < gpio7 sensor value = 0 > or < gpio8 sensor value = 0 >
if < gpio7 sensor value = 0 >
    say Player 1 wins! for 2 secs
if < gpio8 sensor value = 0 >
    say Player 2 wins! for 2 secs
wait 1 secs
broadcast gpio27off
```

Can you use the green LEDs to show who won?

REACTION GAME WITH python

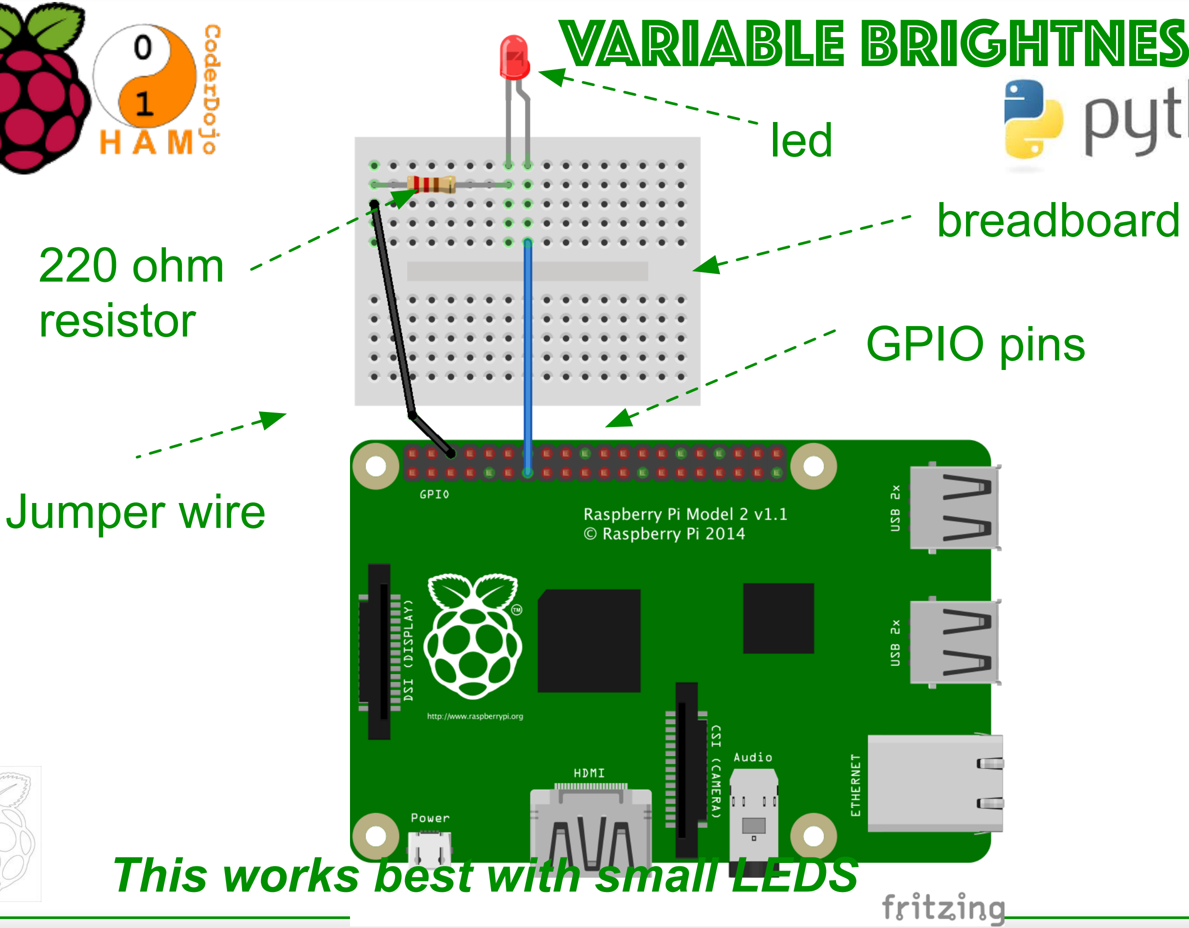


```
# First import some helpful libraries
# This one lets us use LEDs and button with the GPIO pins
from gpiozero import Button, LED
# This one has useful time functions
from time import sleep
# This library lets us do random stuff
import random

led = LED(27) # Our LED is on GPIO27 (pin13)

player_1 = Button(7) # Button connected to GPIO7 (pin26)
player_2 = Button(8) # Button connected to GPIO8 (pin24)

time = random.uniform(1, 8) # Wait between 1 and 8 seconds
sleep(time)
led.on() # Turn LED on

while True: # Forever
    if player_1.is_pressed:
        print("Player 1 wins!")
        break
    if player_2.is_pressed:
        print("Player 2 wins!")
        break

led.off() # Turn LED off
```

Can you use the green LEDs to show who won?

Can you make the game 'best of 3' ?

VARIABLE BRIGHTNESS LED



led

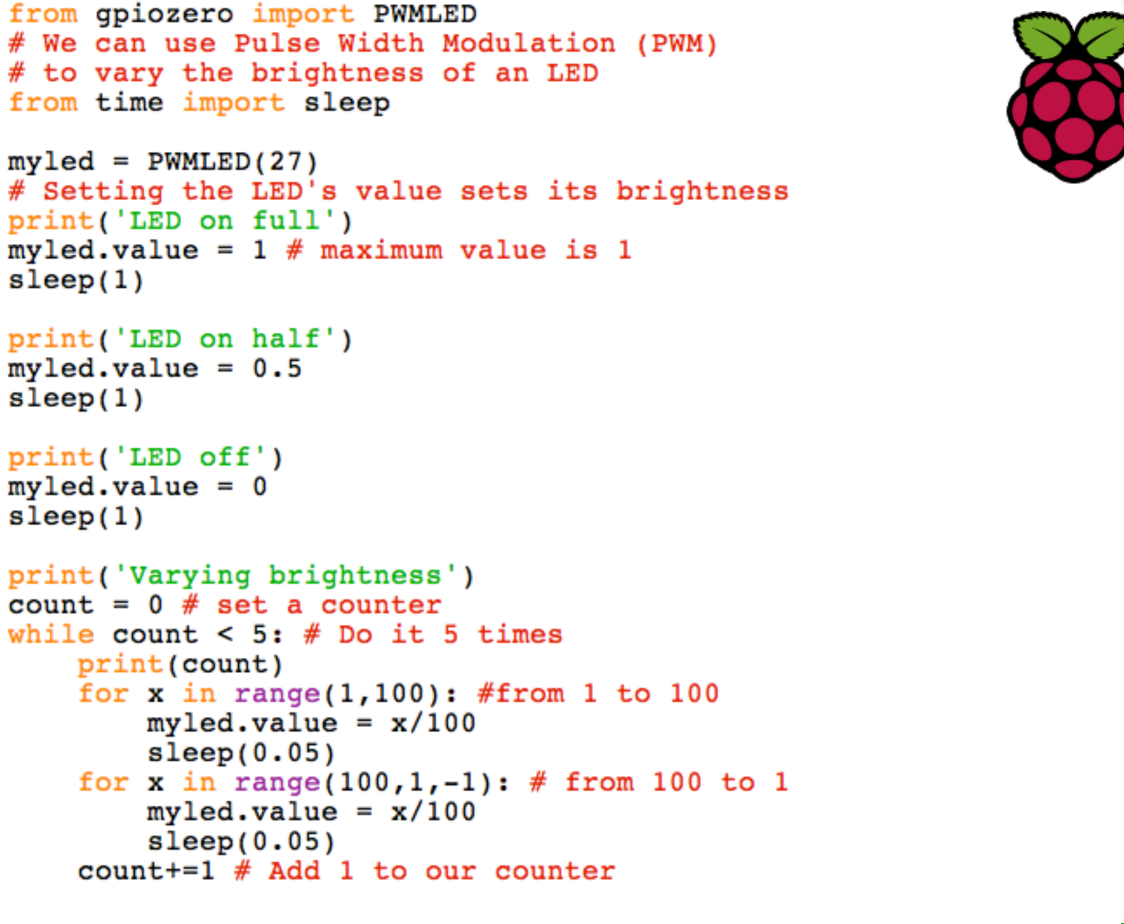
breadboard

220 ohm resistor

GPIO pins

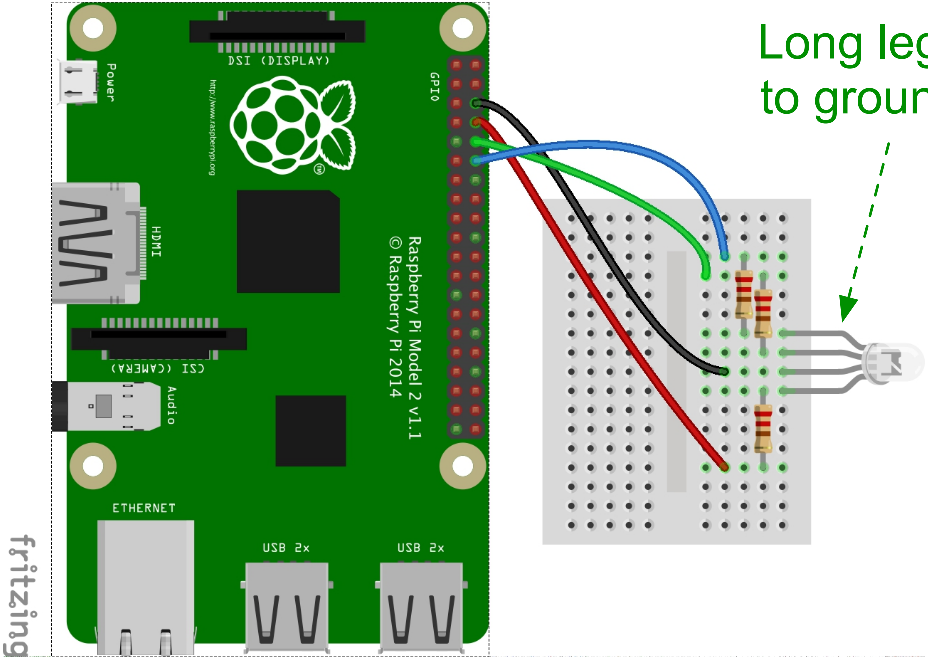
Jumper wire

This works best with small LEDS



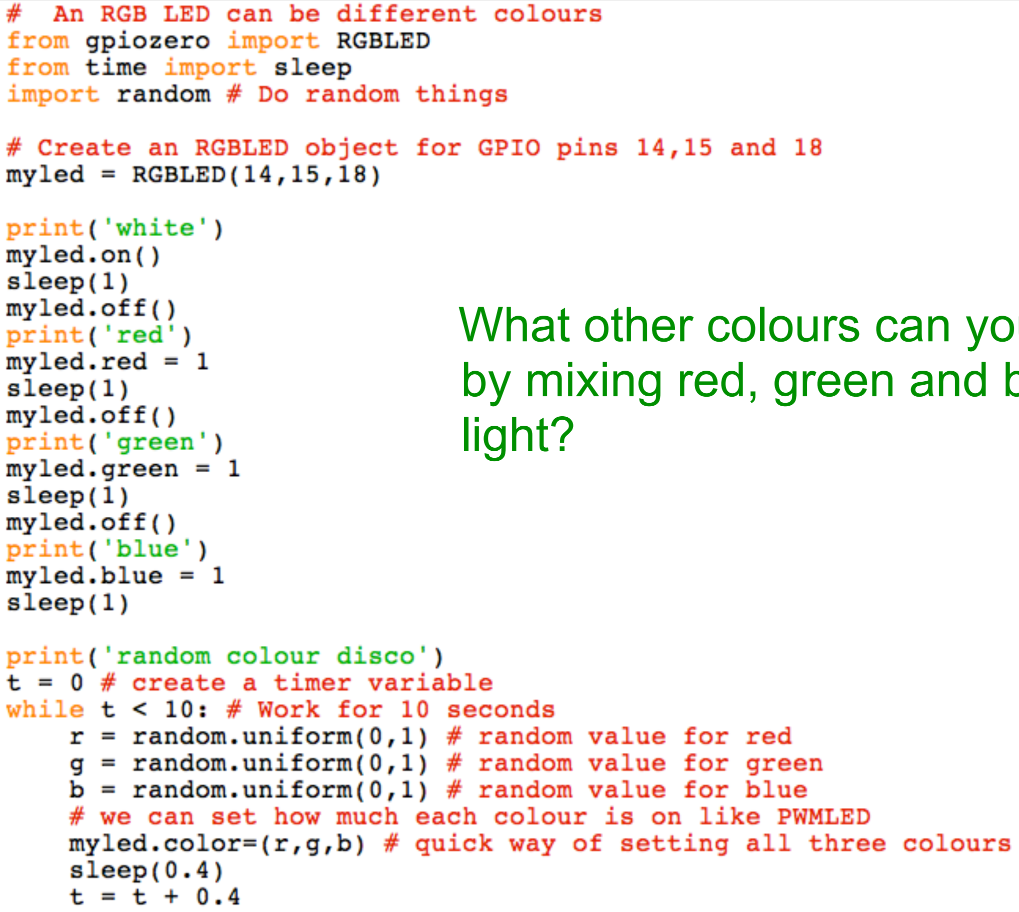
```python
from gpiozero import PWMLED
# We can use Pulse Width Modulation (PWM)
# to vary the brightness of an LED
from time import sleep

myled = PWMLED(27)
# Setting the LED's value sets its brightness
print('LED on full')
myled.value = 1 # maximum value is 1
sleep(1)

print('LED on half')
myled.value = 0.5
sleep(1)

print('LED off')
myled.value = 0
sleep(1)

print('Varying brightness')
count = 0 # set a counter
while count < 5: # Do it 5 times
    print(count)
    for x in range(1,100): #from 1 to 100
        myled.value = x/100
        sleep(0.05)
    for x in range(100,1,-1): # from 100 to 1
        myled.value = x/100
        sleep(0.05)
    count+=1 # Add 1 to our counter
```

RGB LED



Long leg connects to ground

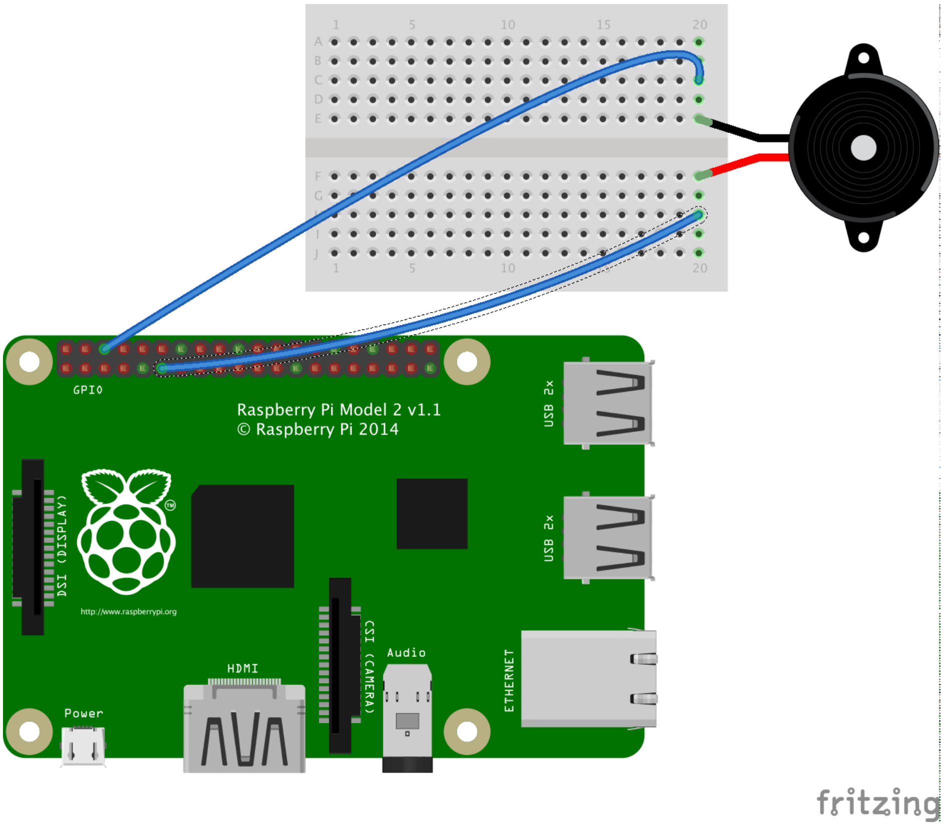
RGB Led



```
#  An RGB LED can be different colours
from gpiozero import RGBLED
from time import sleep
import random # Do random things

# Create an RGBLED object for GPIO pins 14,15 and 18
myled = RGBLED(14,15,18)

print('white')
myled.on()
sleep(1)
myled.off()
print('red')
myled.red = 1
sleep(1)
myled.off()
print('green')
myled.green = 1
sleep(1)
myled.off()
print('blue')
myled.blue = 1
sleep(1)

print('random colour disco')
t = 0 # create a timer variable
while t < 10: # Work for 10 seconds
    r = random.uniform(0,1) # random value for red
    g = random.uniform(0,1) # random value for green
    b = random.uniform(0,1) # random value for blue
    # we can set how much each colour is on like PWMLED
    myled.color=(r,g,b) # quick way of setting all three colours
    sleep(0.4)
    t = t + 0.4
```

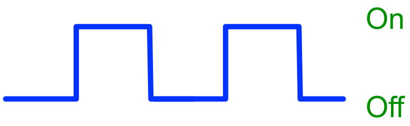
What other colours can you create by mixing red, green and blue light?

BUZZ BUZZER

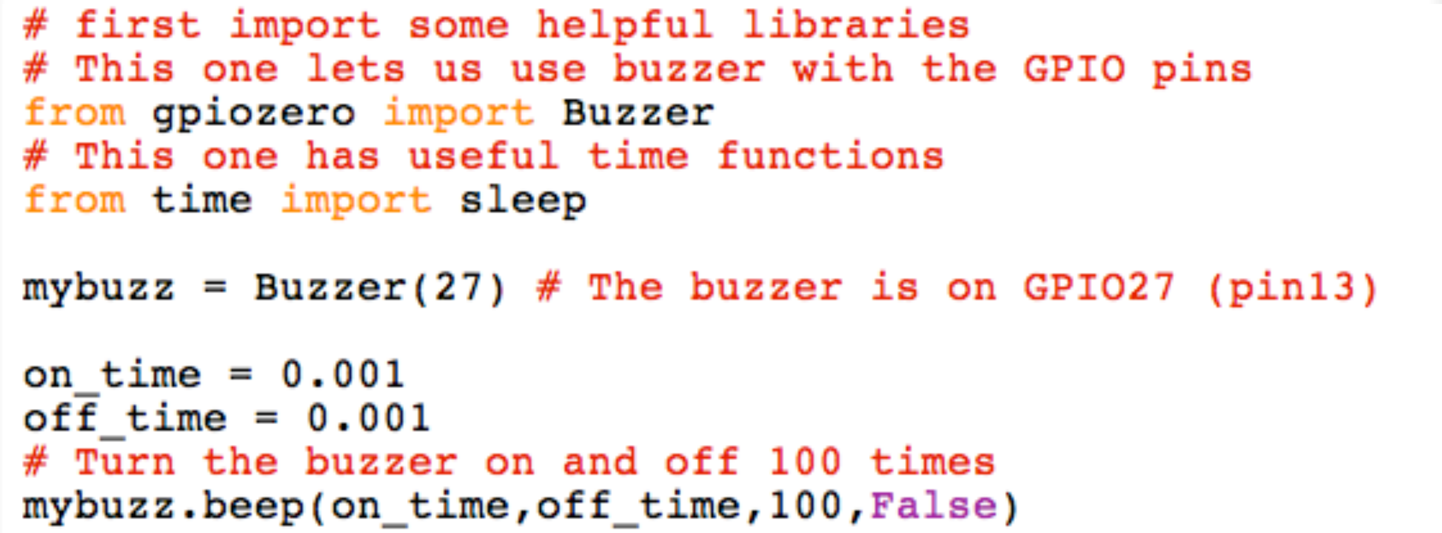


```
#First import the gpiozero library and load the buzzer functions
from gpiozero import Buzzer
# Import the time library's sleep function
from time import sleep

mybuzz = Buzzer(27) # Buzzer connected to GPIO27

mybuzz.on()
sleep(1)
mybuzz.off()
```

We can change the note that the buzzer makes by feeding it a square wave. In other words, turn it on and off quickly!



BUZZ BUZZER WITH python



```
# first import some helpful libraries
# This one lets us use buzzer with the GPIO pins
from gpiozero import Buzzer
# This one has useful time functions
from time import sleep

mybuzz = Buzzer(27) # The buzzer is on GPIO27 (pin13)

on_time = 0.001
off_time = 0.001
# Turn the buzzer on and off 100 times
mybuzz.beep(on_time,off_time,100,False)
```

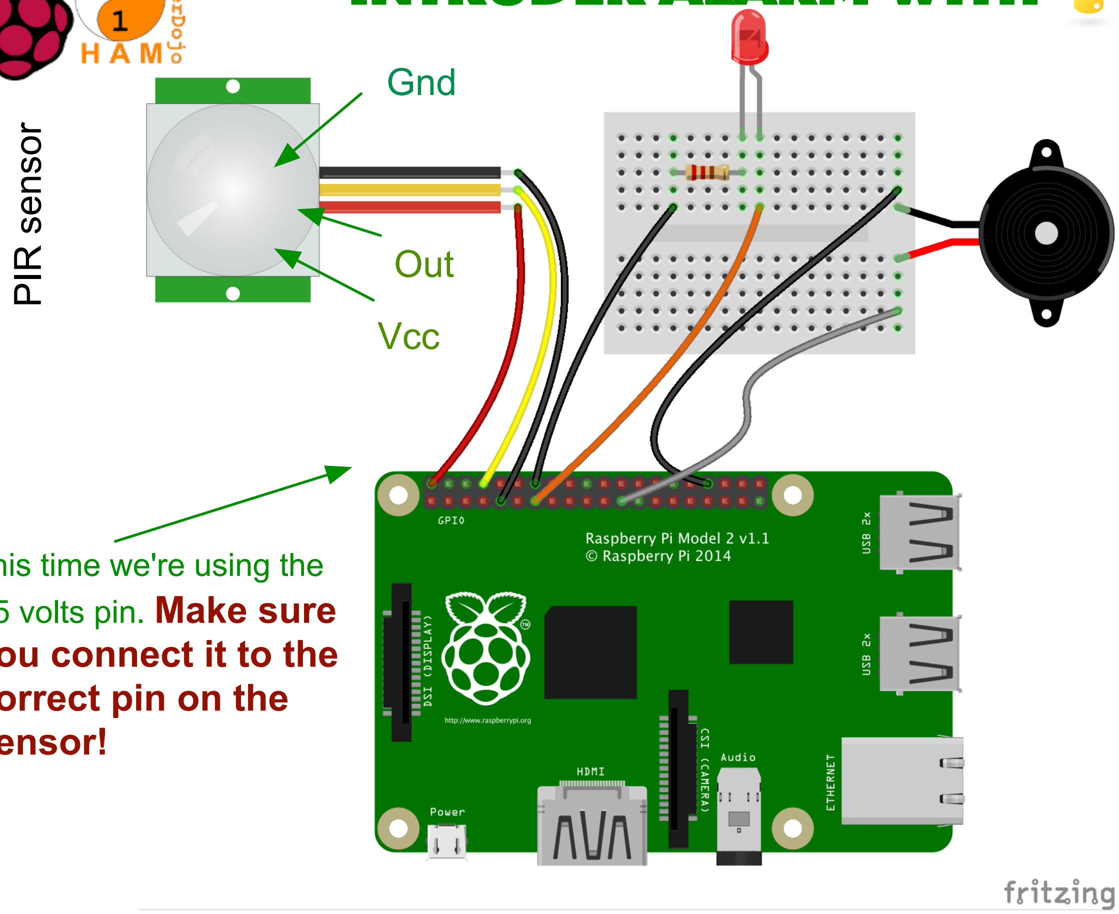
Try adjusting the values of off_time and on_time that you use to see how it affects the sound.

Can you modify the code so that it plays a series of notes of increasing frequency?

What happens if you change False to True?

Can you modify the 'Reaction Time' code to include a buzzer that sounds when a player presses their button? Make it play a different note for each player

INTRUDER ALARM WITH python

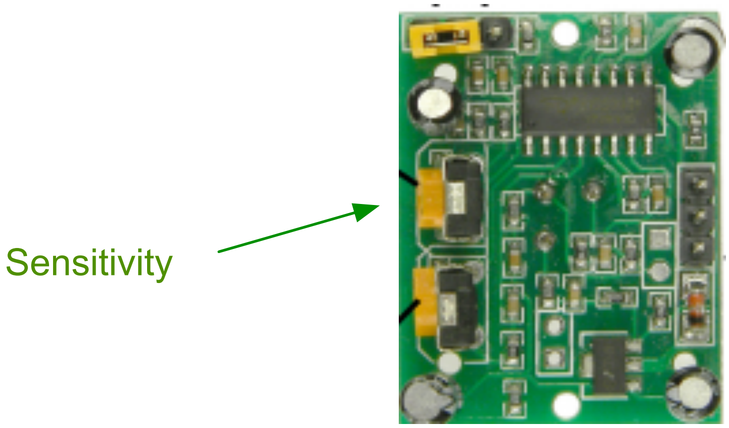


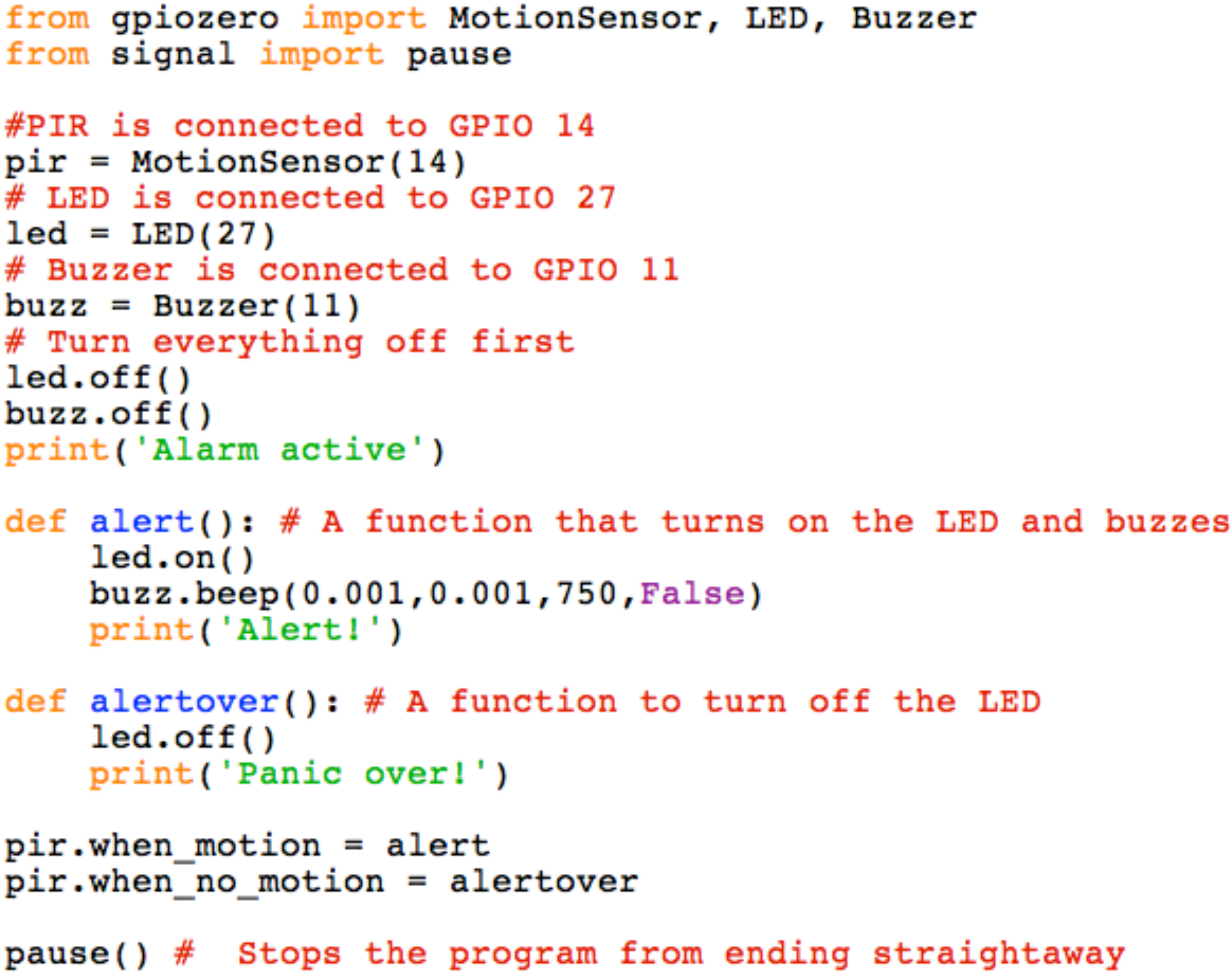
This time we're using the +5 volts pin. **Make sure you connect it to the correct pin on the sensor!**

The PIR sensor uses Infra-red light to detect movement. Can you see it?

The sensor's output is 0 when no motion is detected, then changes to 1 when something moves.

Too sensitive? You can adjust its range:



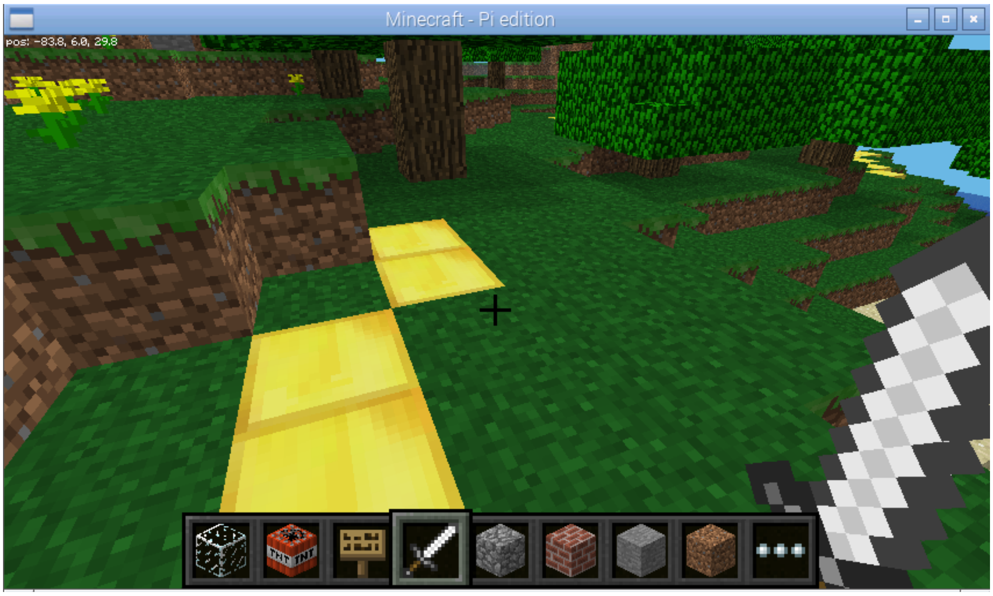


```
from gpiozero import MotionSensor, LED, Buzzer
from signal import pause

#PIR is connected to GPIO 14
pir = MotionSensor(14)
# LED is connected to GPIO 27
led = LED(27)
# Buzzer is connected to GPIO 11
buzz = Buzzer(11)
# Turn everything off first
led.off()
buzz.off()
print('Alarm active')

def alert(): # A function that turns on the LED and buzzes
    led.on()
    buzz.beep(0.001,0.001,750,False)
    print('Alert!')

def alertover(): # A function to turn off the LED
    led.off()
    print('Panic over!')

pir.when_motion = alert
pir.when_no_motion = alertover

pause() #  Stops the program from ending straightaway
```

Can you modify the code so that the LED flashes?

The buzzer only sounds for a few seconds. How about making it continue until a button is pressed to silence the alarm?

PAVED WITH GOLD

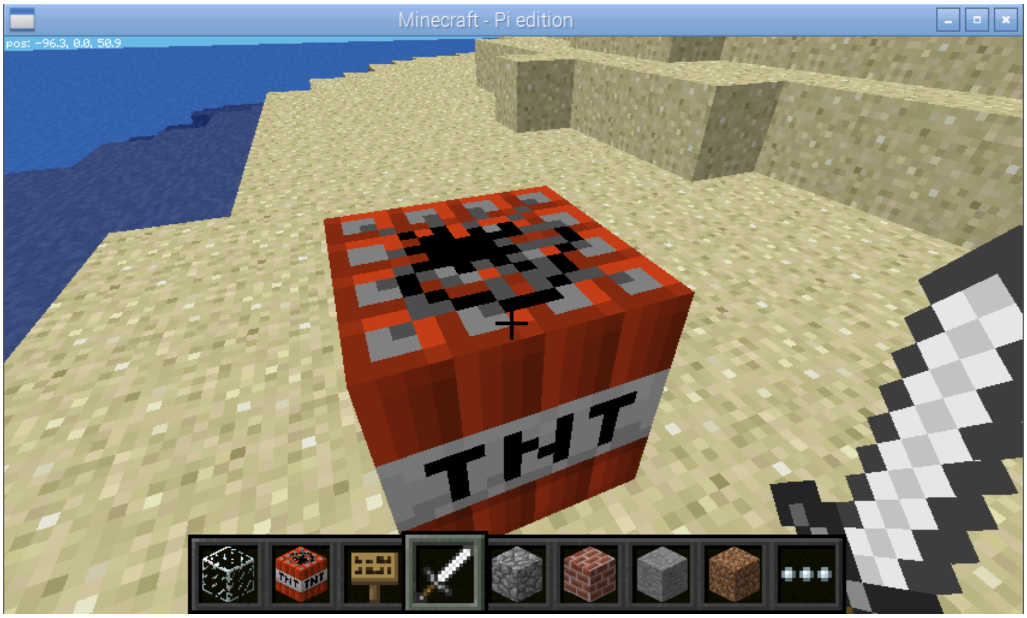


We're going to program Minecraft so that we leave a trail of gold behind us.

```
# load the Minecraft API - the functions to talk to Minecraft
import mcpi.minecraft as minecraft
import mcpi.block as block
import time

# Connect our program to Minecraft
mc = minecraft.Minecraft.create()

# Write our function to leave a trail of gold behind us

def goldsteps():
# Get the position of the player
        pos = mc.player.getTilePos()
# Get the type of block underneath (y-1) the player
        b = mc.getBlock(pos.x,pos.y-1,pos.z)
# if the blokc is grass...
        if b == block.GRASS.id:
# ...change it to gold
                mc.setBlock(pos.x,pos.y-1,pos.z,block.GOLD_BLOCK.id)

#Main code block

while True:
        time.sleep(0.25)
# run our function
        goldsteps()
```

Can you modify your code so that it leaves a different block behind?

BOOM!

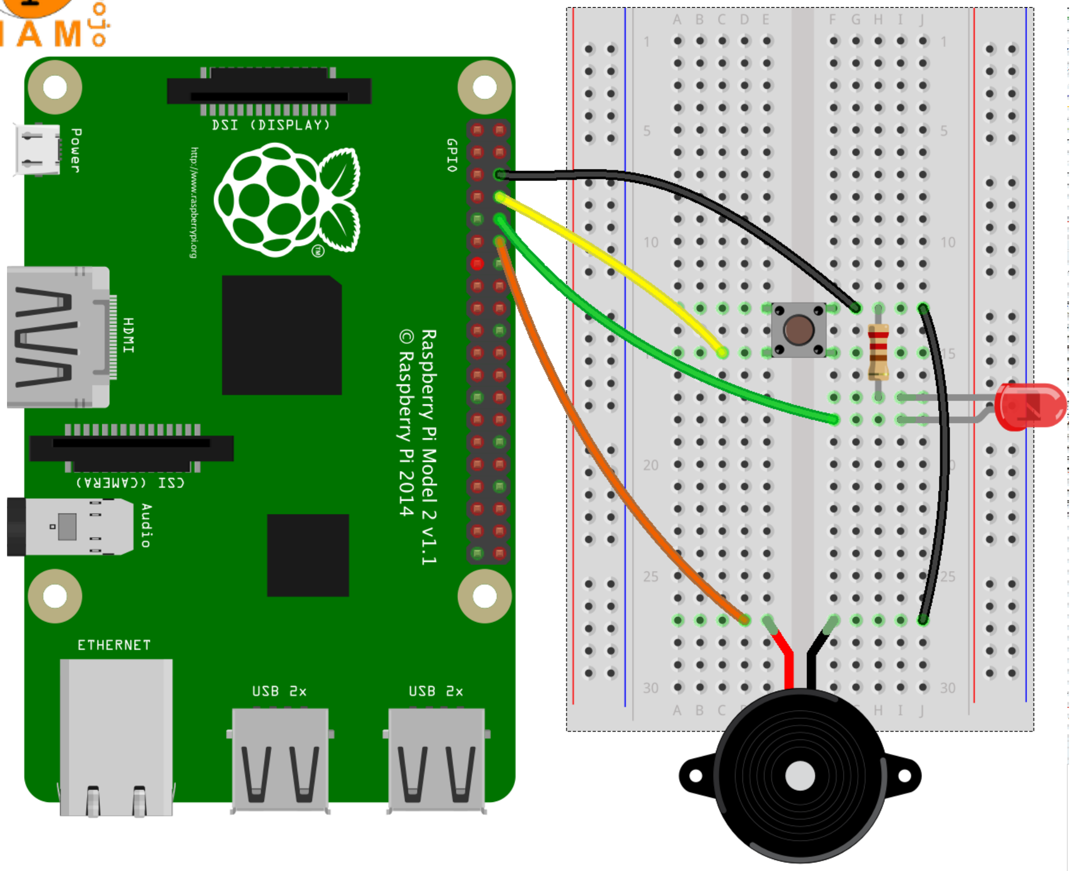


In Minecraft Pi Edition there is no way to detonate TNT... unless we use some Python!

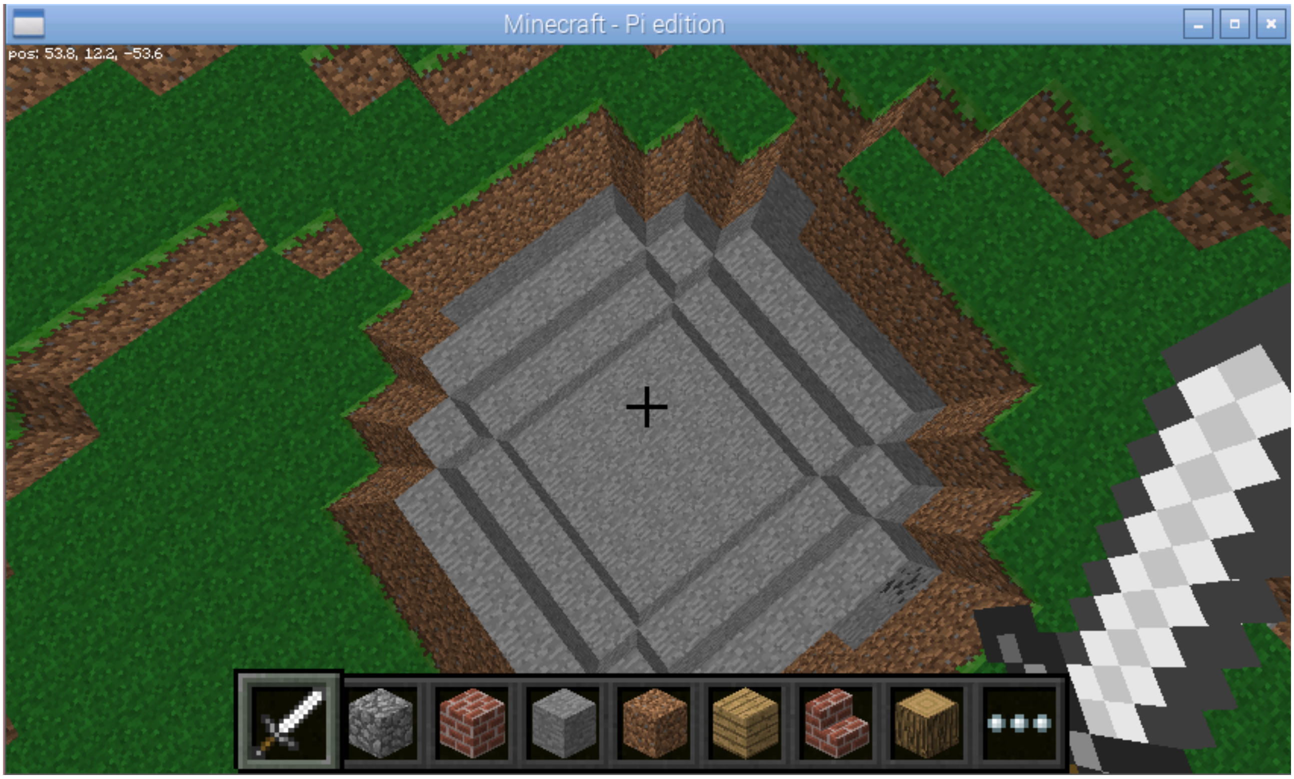
```
# load the Minecraft API - the functions to talk to Minecraft
import mcpi.minecraft as minecraft
import time
import random

#Connect our program to Minecraft
mc = minecraft.Minecraft.create()

while True:

# Read the list of events that have happened and look for things being hit
        hits = mc.events.pollBlockHits()
# check each event
        for hit in hits:
# get the block type that was hit
                block = mc.getBlockWithData(hit.pos.x, hit.pos.y, hit.pos.z)
# if its data value is 0...
                if block.data == 0:
# ... set it to 1
                        block.data = (block.data + 1)
                        mc.setBlock(hit.pos.x, hit.pos.y, hit.pos.z, block.id, block.data)
# send a chat message to confirm
                mc.postToChat("block is now" + str(block.data))
        time.sleep(0.1)
```

Run your code.
Place a TNT block
Switch to your sword.
Use it on the TNT (RIGHT click)
Hit the TNT a couple of times (LEFT click)

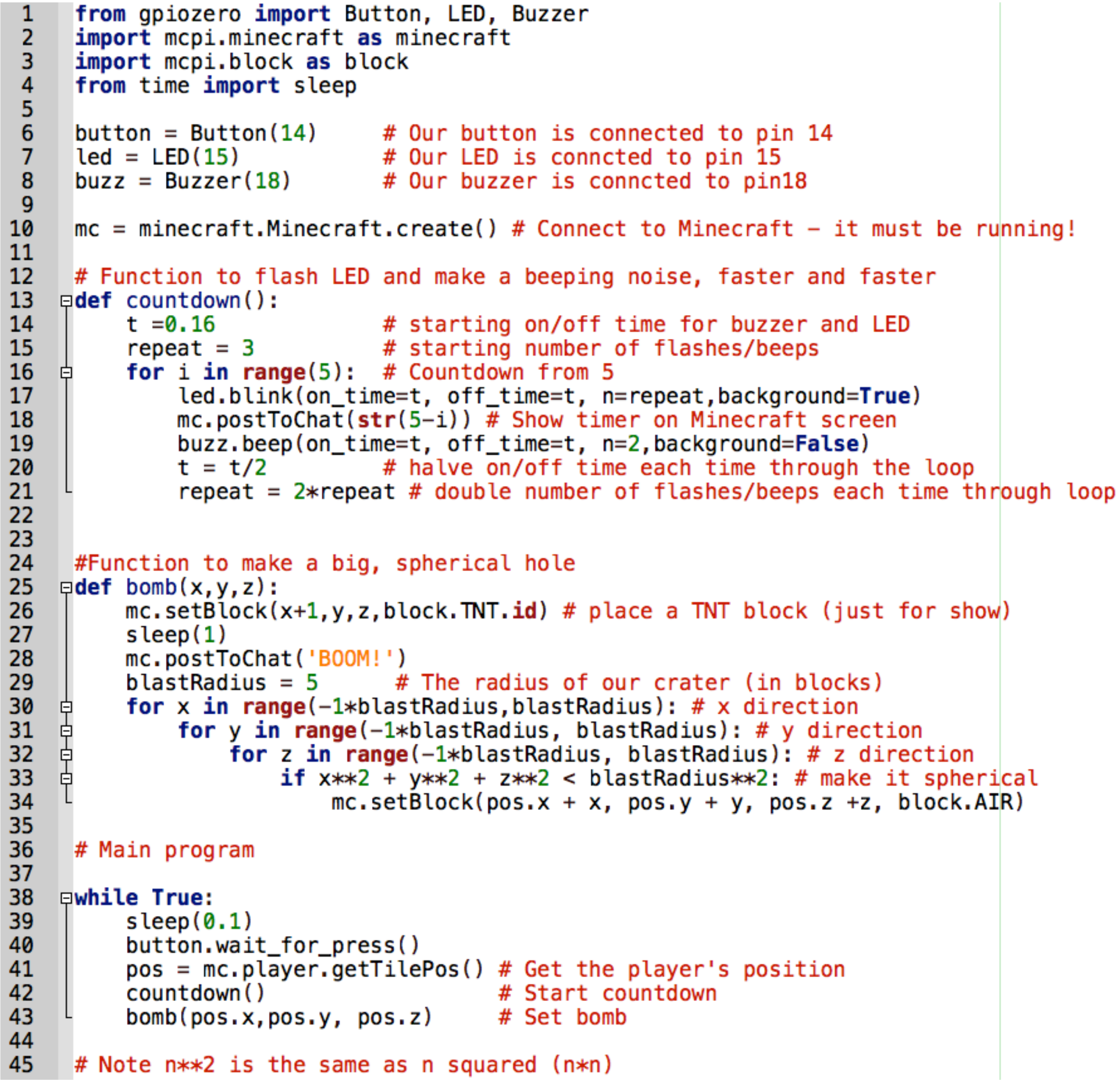
BIGGER BOOM!



Let's make a more impressive explosion crater!

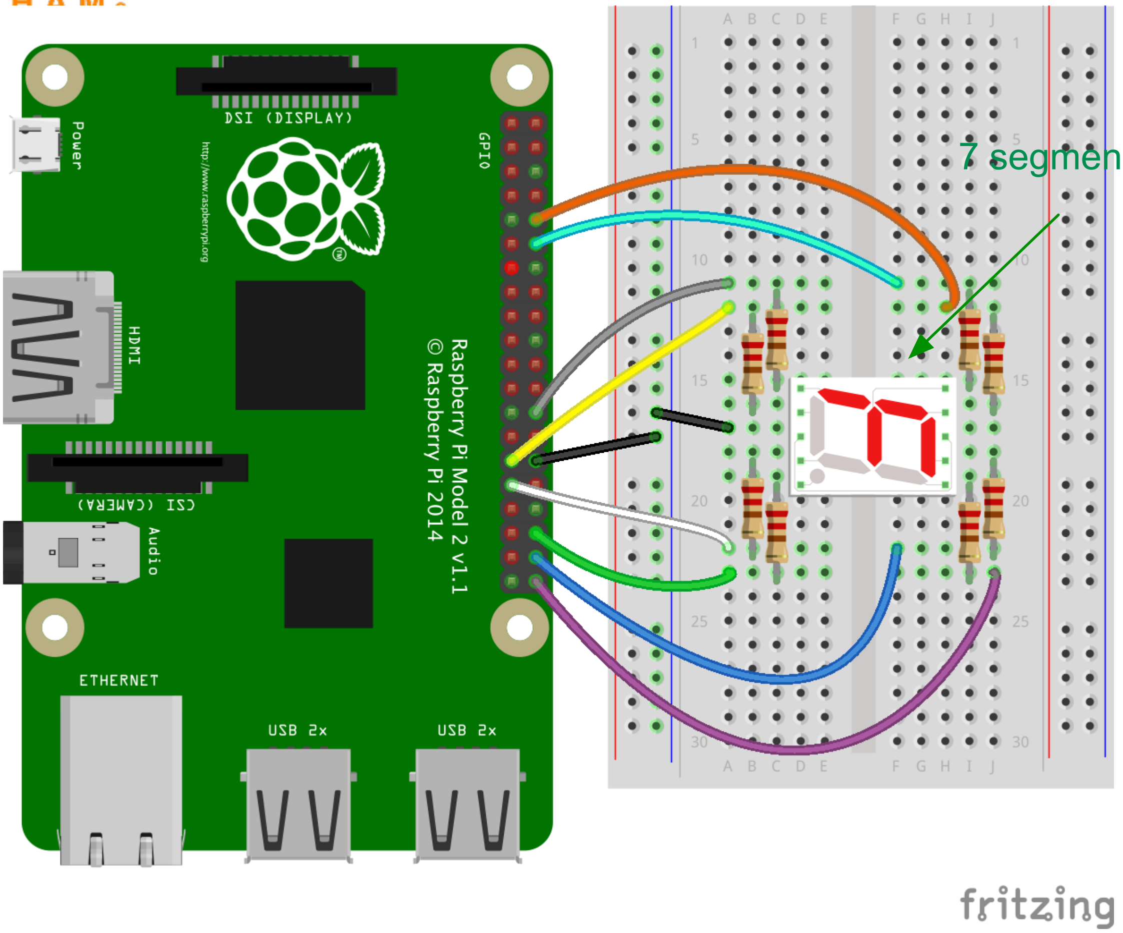


We'll combine lots of the Python code we've
already used before (led, buzzer, reaction game)

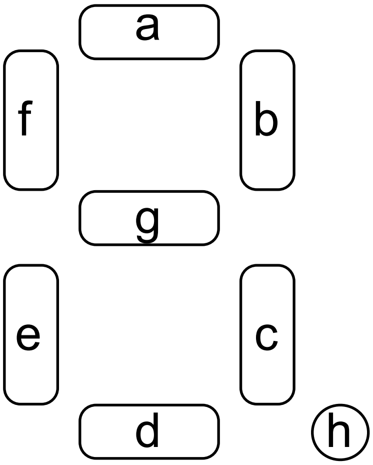


```python
from gpiozero import Button, LED, Buzzer
import mcpi.minecraft as minecraft
import mcpi.block as block
from time import sleep

button = Button(14)     # Our button is connected to pin 14
led = LED(15)           # Our LED is conncted to pin 15
buzz = Buzzer(18)       # Our buzzer is conncted to pin18

mc = minecraft.Minecraft.create() # Connect to Minecraft - it must be running!

# Function to flash LED and make a beeping noise, faster and faster
def countdown():
    t =0.16             # starting on/off time for buzzer and LED
    repeat = 3          # starting number of flashes/beeps
    for i in range(5):  # Countdown from 5
        led.blink(on_time=t, off_time=t, n=repeat,background=True)
        mc.postToChat(str(5-i)) # Show timer on Minecraft screen
        buzz.beep(on_time=t, off_time=t, n=2,background=False)
        t = t/2         # halve on/off time each time through the loop
        repeat = 2*repeat # double number of flashes/beeps each time through loop


#Function to make a big, spherical hole
def bomb(x,y,z):
    mc.setBlock(x+1,y,z,block.TNT.id) # place a TNT block (just for show)
    sleep(1)
    mc.postToChat('BOOM!')
    blastRadius = 5      # The radius of our crater (in blocks)
    for x in range(-1*blastRadius,blastRadius): # x direction
        for y in range(-1*blastRadius, blastRadius): # y direction
            for z in range(-1*blastRadius, blastRadius): # z direction
                if x**2 + y**2 + z**2 < blastRadius**2: # make it spherical
                    mc.setBlock(pos.x + x, pos.y + y, pos.z +z, block.AIR)

# Main program

while True:
    sleep(0.1)
    button.wait_for_press()
    pos = mc.player.getTilePos() # Get the player's position
    countdown()                  # Start countdown
    bomb(pos.x,pos.y, pos.z)     # Set bomb

# Note n**2 is the same as n squared (n*n)
```

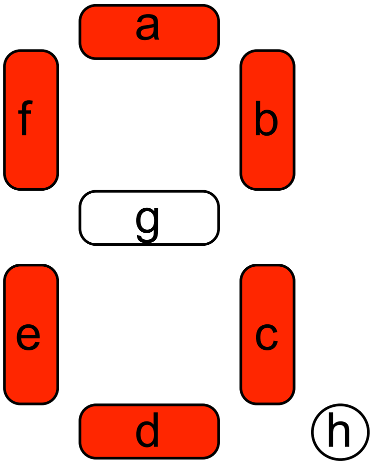
7 SEGMENT DISPLAY WITH python™



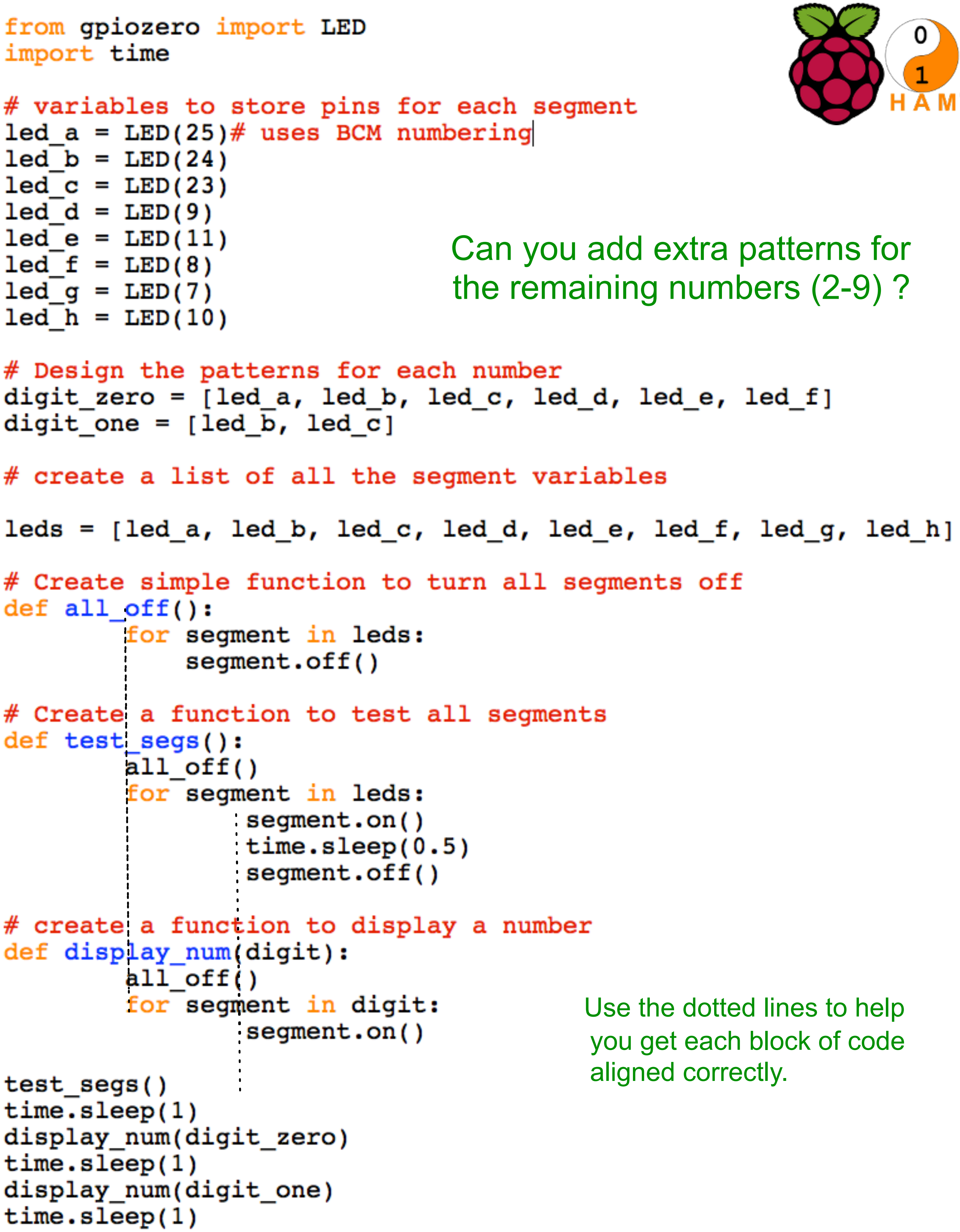
A seven segment display uses LEDs to show numbers. Each LED element is normally referred to by a letter, and can be switched on or off to make the correct shape.



So by turning on a, b, c, d, e, and f, we can make a zero



The Python code on the next page can be used to control the seven segment display.



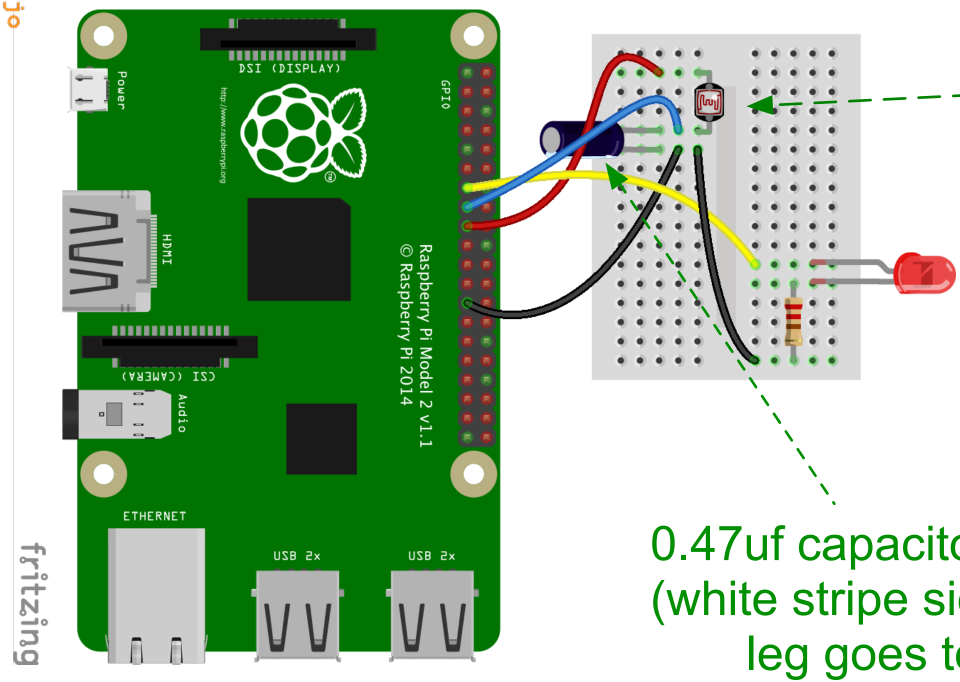
```
from gpiozero import LED
import time

# variables to store pins for each segment
led_a = LED(25)# uses BCM numbering
led_b = LED(24)
led_c = LED(23)
led_d = LED(9)
led_e = LED(11)
led_f = LED(8)
led_g = LED(7)
led_h = LED(10)

# Design the patterns for each number
digit_zero = [led_a, led_b, led_c, led_d, led_e, led_f]
digit_one = [led_b, led_c]

# create a list of all the segment variables

leds = [led_a, led_b, led_c, led_d, led_e, led_f, led_g, led_h]

# Create simple function to turn all segments off
def all_off():
        for segment in leds:
            segment.off()

# Create a function to test all segments
def test_segs():
        all_off()
        for segment in leds:
                segment.on()
                time.sleep(0.5)
                segment.off()

# create a function to display a number
def display_num(digit):
        all_off()
        for segment in digit:
                segment.on()

test_segs()
time.sleep(1)
display_num(digit_zero)
time.sleep(1)
display_num(digit_one)
time.sleep(1)
```

Can you add extra patterns for the remaining numbers (2-9) ?

Use the dotted lines to help you get each block of code aligned correctly.

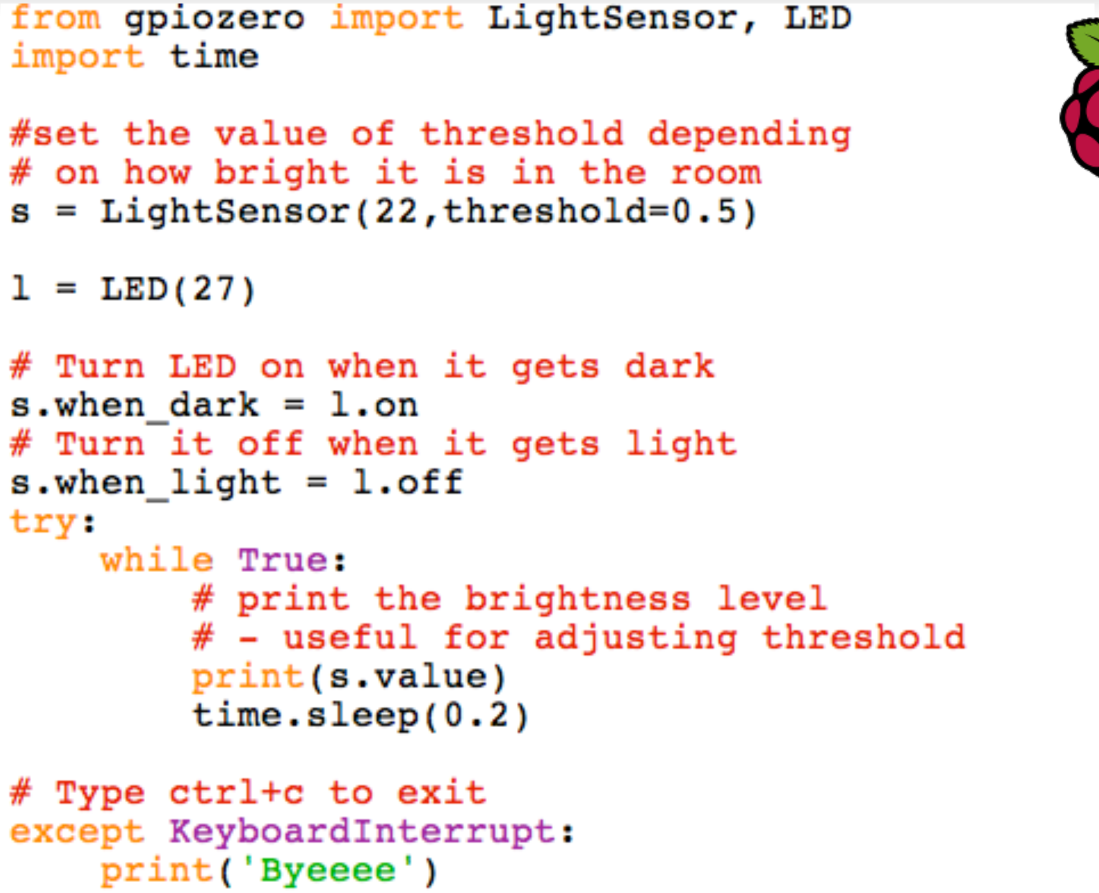
Extend and modify the code so that the leds count down from 9 to 0. Can you use a loop and another list?

LIGHT SENSOR



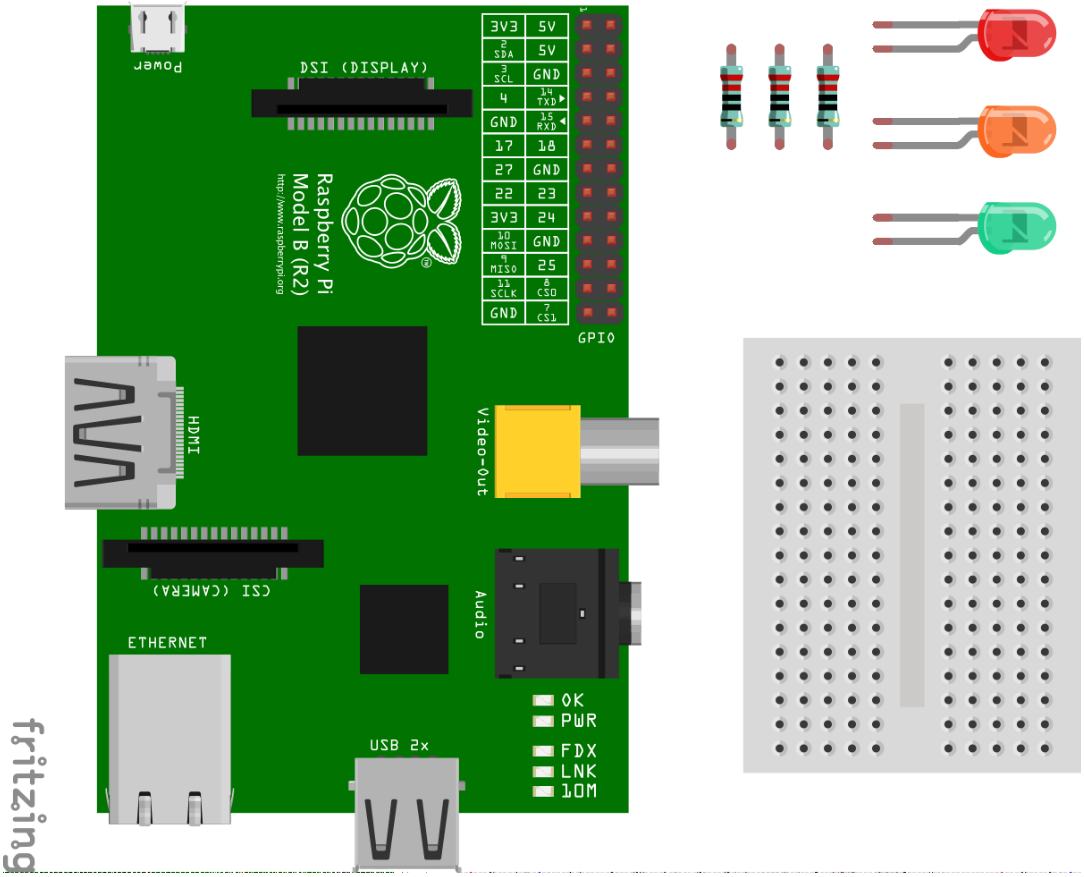
LDR

0.47uf capacitor
(white stripe side/short
leg goes to gnd)



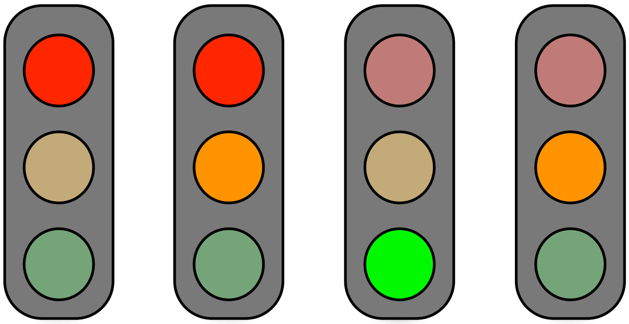
```
from gpiozero import LightSensor, LED
import time

#set the value of threshold depending
# on how bright it is in the room
s = LightSensor(22,threshold=0.5)

l = LED(27)

# Turn LED on when it gets dark
s.when_dark = l.on
# Turn it off when it gets light
s.when_light = l.off
try:
    while True:
        # print the brightness level
        # - useful for adjusting threshold
        print(s.value)
        time.sleep(0.2)

# Type ctrl+c to exit
except KeyboardInterrupt:
    print('Byeeee')
```

TRAFFIC LIGHTS CHALLENGE



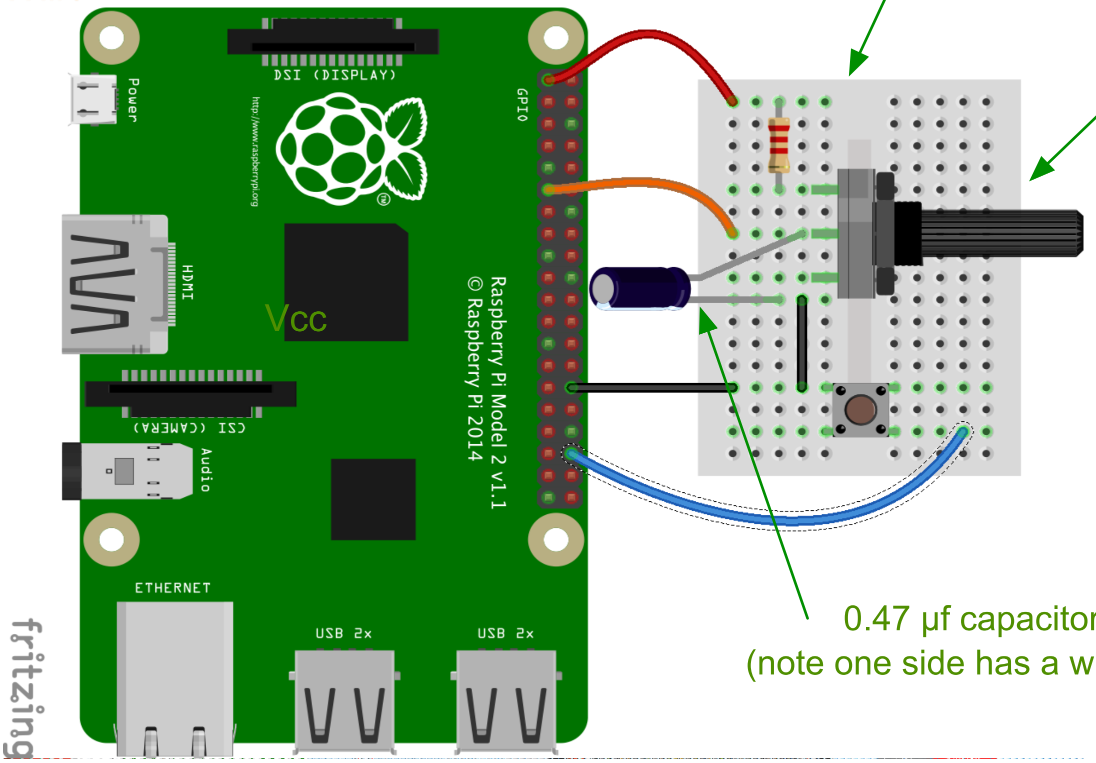
Can you build a traffic light simulator?

Connect the LEDs and resistors to the Pi and then write the code to make them display the correct sequence.



Remember the STOP (red) and GO (green) cycles should be longer than the GET READY (when amber is on).

FROZEN

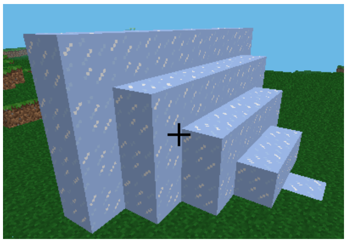


2.2K resistor

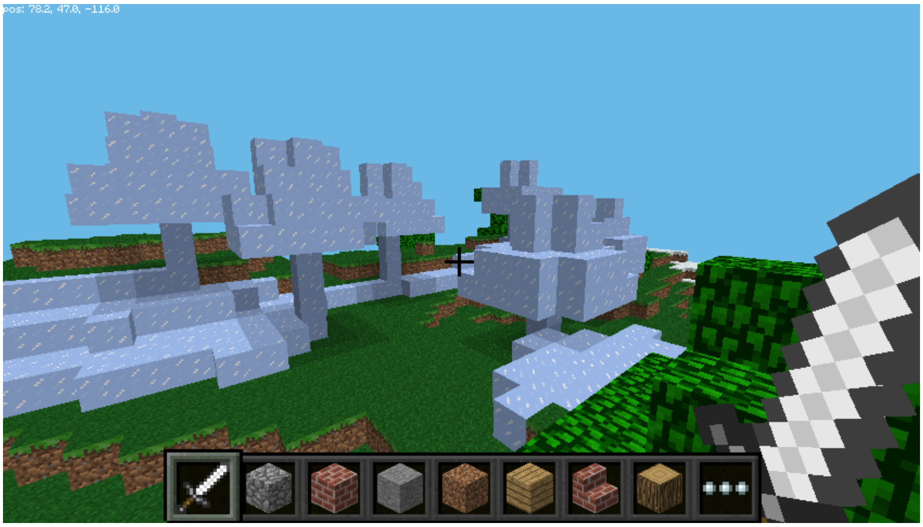
100K variable resistor

0.47 µf capacitor (note one side has a white stripe)

We want to be able to freeze Minecraft blocks, but have control over how far our magic powers can reach.

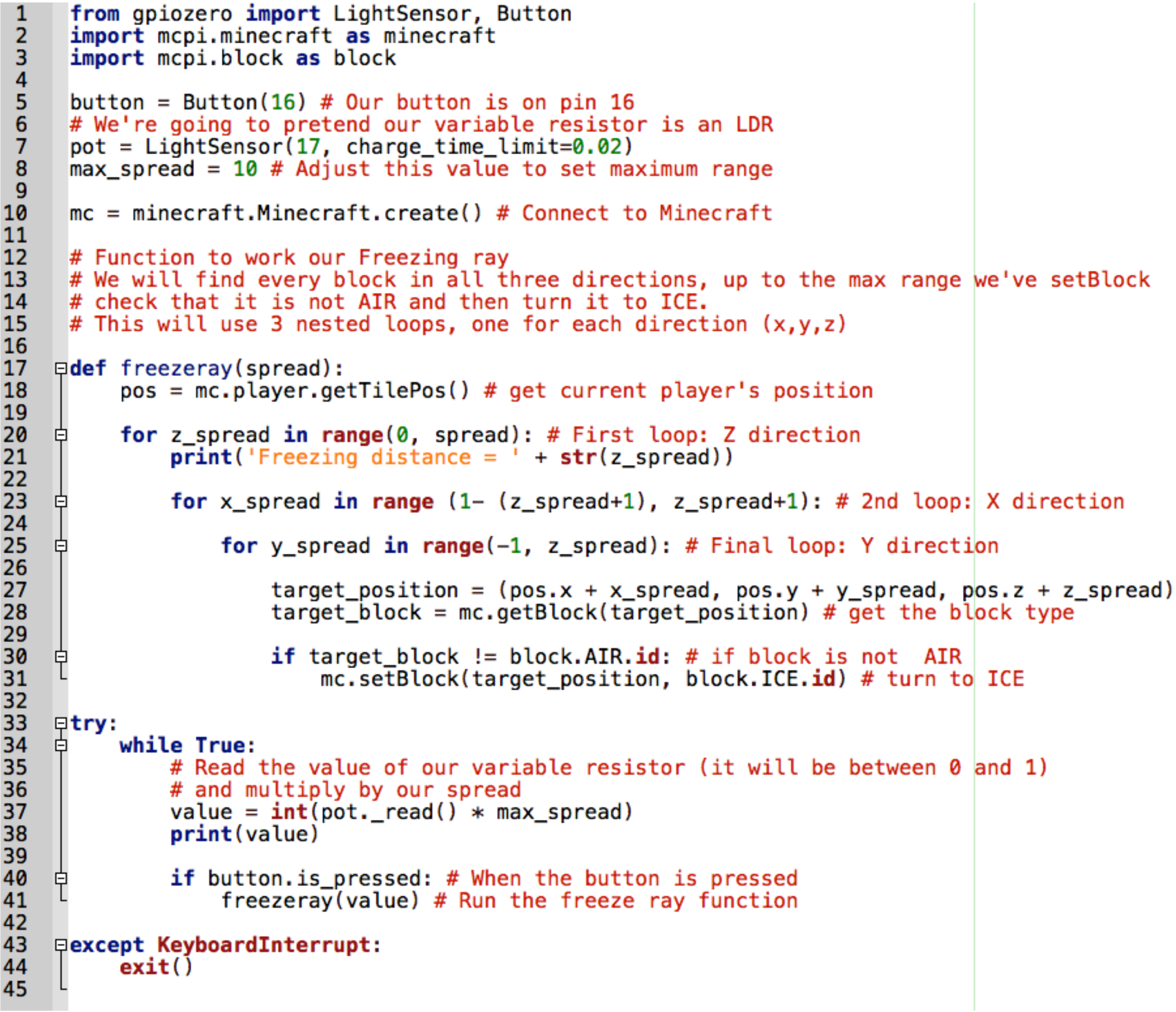


Let's have the coldness spreading out from us in this shape, but only freezing blocks that are NOT air.



We can adjust the variable resistor to set the range of our freezing. Just like an LDR, it is an analogue component so we use a capacitor to make a timing circuit.

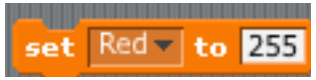
We're pretending the variable resistor is an LDR (LightSensor) and hacking the gpiozero class so that it does what we want.



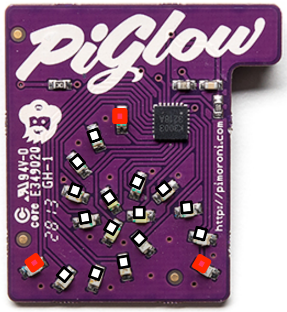
```python
from gpiozero import LightSensor, Button
import mcpi.minecraft as minecraft
import mcpi.block as block

button = Button(16) # Our button is on pin 16
# We're going to pretend our variable resistor is an LDR
pot = LightSensor(17, charge_time_limit=0.02)
max_spread = 10 # Adjust this value to set maximum range

mc = minecraft.Minecraft.create() # Connect to Minecraft

# Function to work our Freezing ray
# We will find every block in all three directions, up to the max range we've setBlock
# check that it is not AIR and then turn it to ICE.
# This will use 3 nested loops, one for each direction (x,y,z)

def freezeray(spread):
    pos = mc.player.getTilePos() # get current player's position

    for z_spread in range(0, spread): # First loop: Z direction
        print('Freezing distance = ' + str(z_spread))

        for x_spread in range (1- (z_spread+1), z_spread+1): # 2nd loop: X direction

            for y_spread in range(-1, z_spread): # Final loop: Y direction

                target_position = (pos.x + x_spread, pos.y + y_spread, pos.z + z_spread)
                target_block = mc.getBlock(target_position) # get the block type

                if target_block != block.AIR.id: # if block is not  AIR
                    mc.setBlock(target_position, block.ICE.id) # turn to ICE

try:
    while True:
        # Read the value of our variable resistor (it will be between 0 and 1)
        # and multiply by our spread
        value = int(pot._read() * max_spread)
        print(value)

        if button.is_pressed: # When the button is pressed
            freezeray(value) # Run the freeze ray function

except KeyboardInterrupt:
    exit()

```

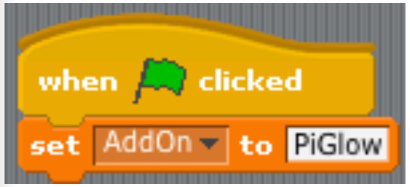
Can you modify the code to increase the maximum range?

Can you make a lava-ray?

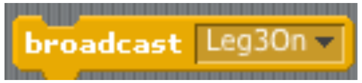


```
set [Red] to [255]
```

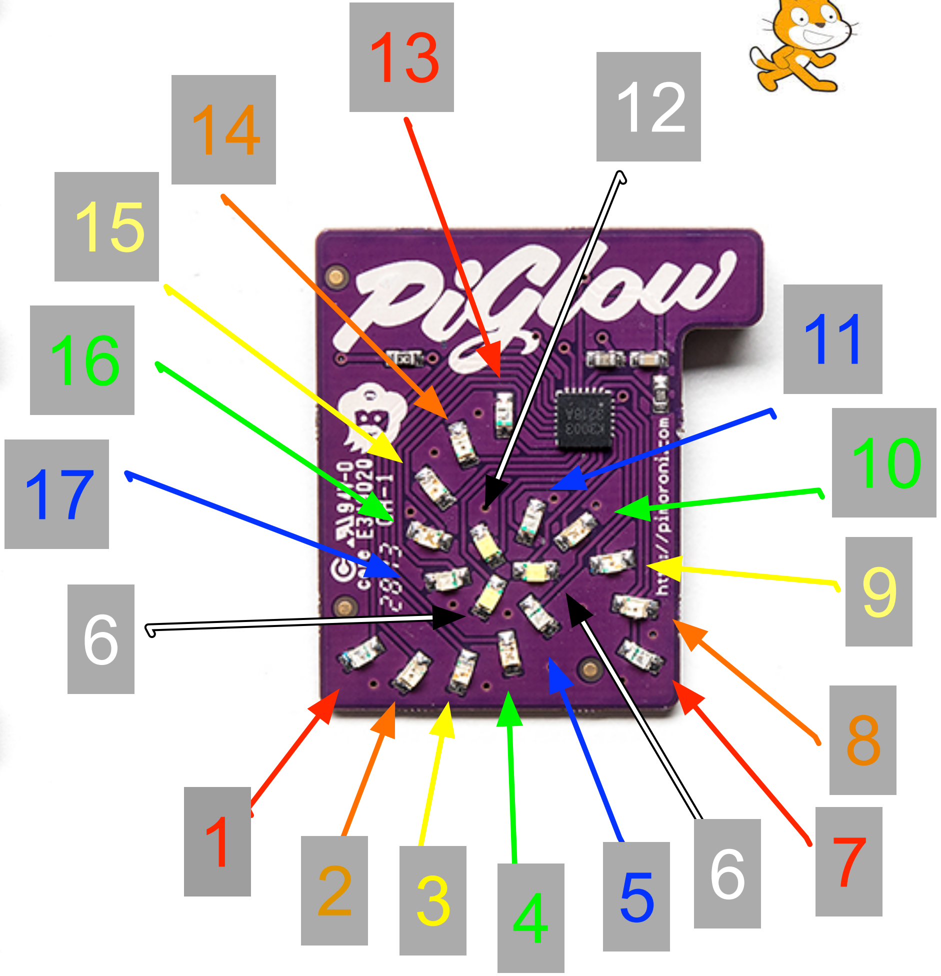


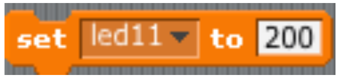


```
when green flag clicked
set [AddOn] to [PiGlow]
```



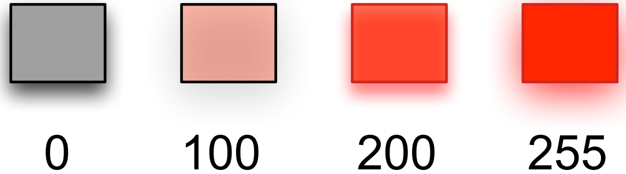
```
broadcast [Leg3On]
```





```
set [led11] to [200]
```

Brightness

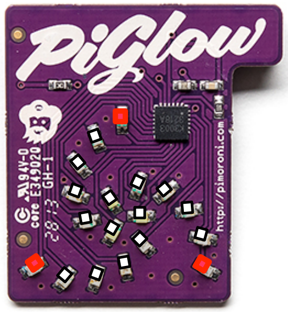


WARNING

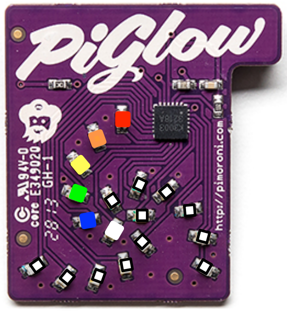
Brightness 255 is very bright!
Don't stare at the LEDs when set to 255 - It may hurt your eyes.
Especially the white ones!

```
import piglow
```

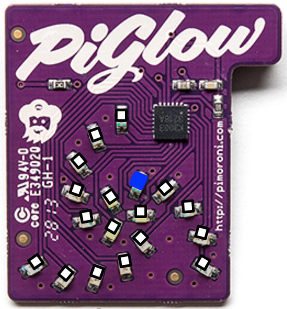
```
piglow.red(255)
piglow.show()
```



```
piglow.leg(3,255)
piglow.show()
```



```
piglow.set(11,255)
piglow.show()
```



[The numbers for the LEDs in Python are one less than in Scratch]

Fed up with typing piglow.show()*?*

```
piglow.auto_update = True
```

(but complex patterns will now run slower)

Turn everything off

```
piglow.all(0)
```

Turn everything off when the code ends.

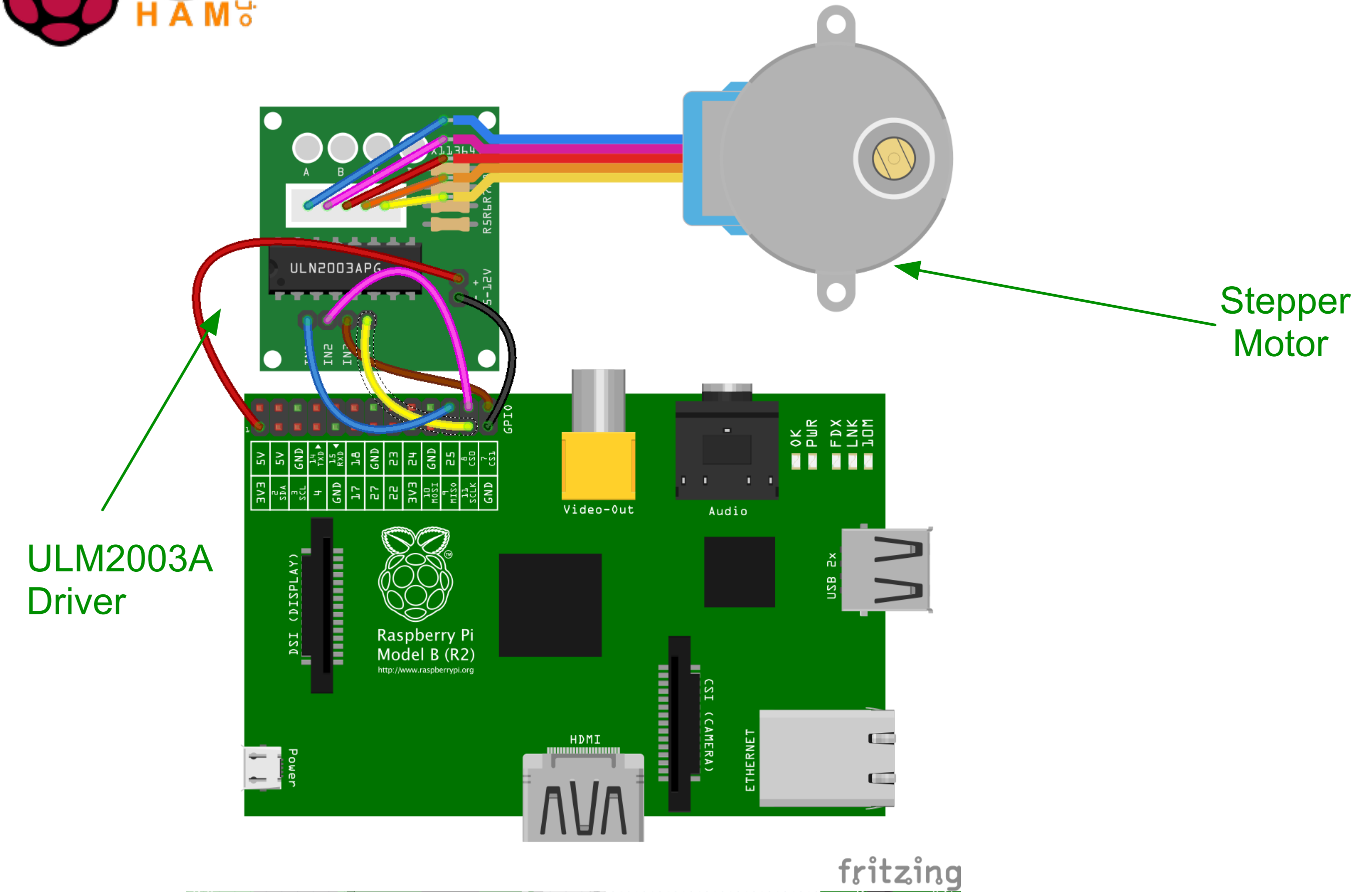
```
piglow.clear_on_exit = True
```

Challenge: CPU monitor

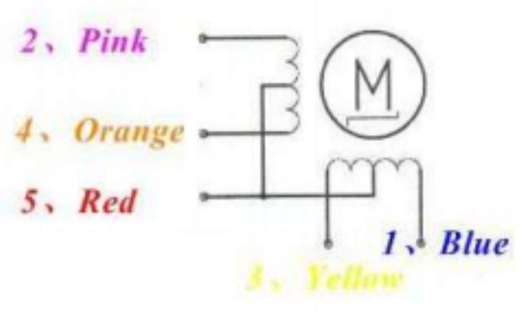
Can you write code that makes the PiGlow light up depending on how hard the CPU is working ?

```
import psutil
cpu = psutil.cpu_percent()
```

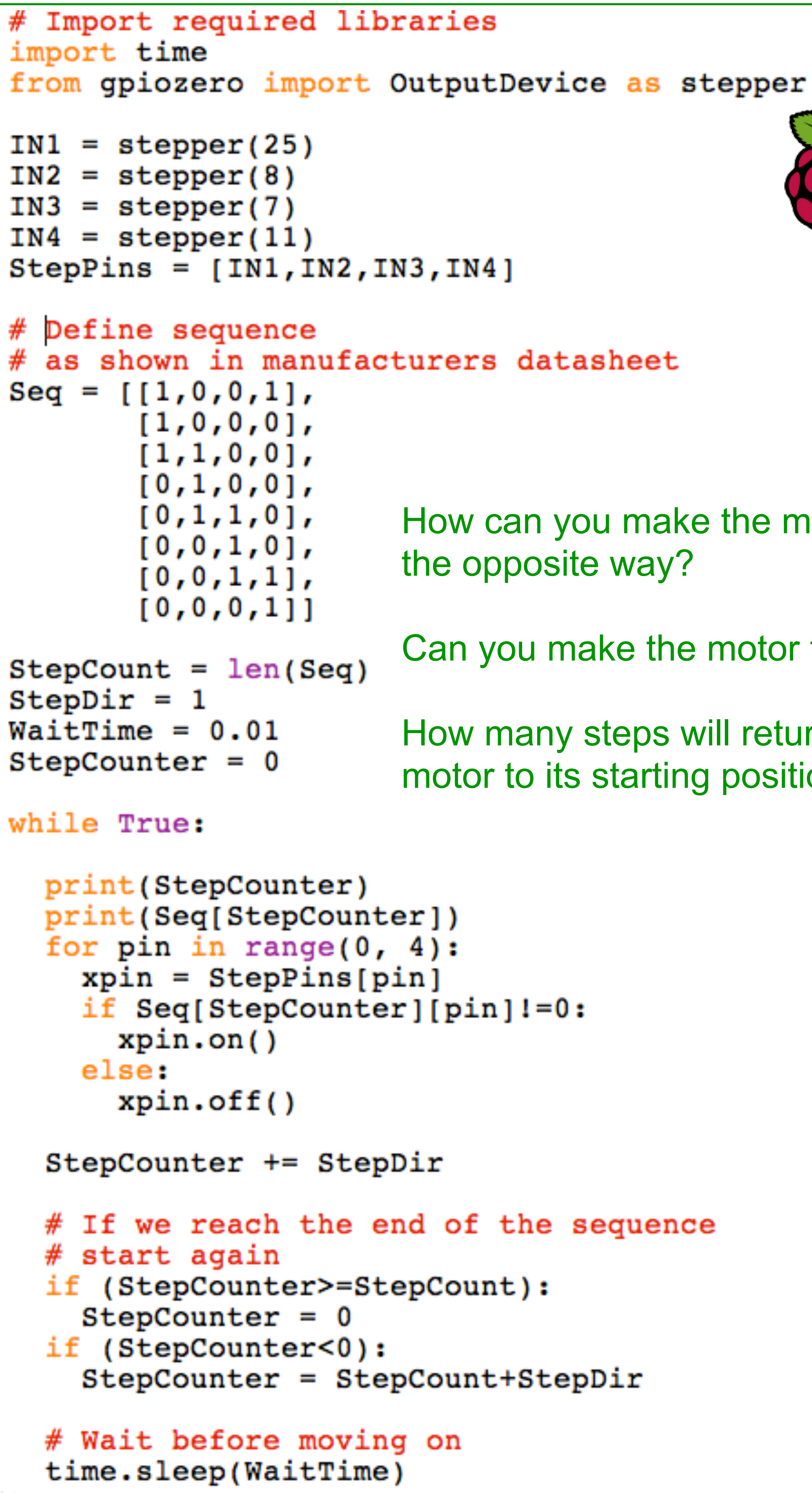
STEPPER MOTOR



Stepper motors are DC motors that move in steps. They have multiple coils that are organised in groups called "phases". By energising each phase in sequence, the motor will rotate, one step at a time.



We use a driver board which we can control from the Pi. This way we can reverse the current and turn the motor in both directions.



```
# Import required libraries
import time
from gpiozero import OutputDevice as stepper

IN1 = stepper(25)
IN2 = stepper(8)
IN3 = stepper(7)
IN4 = stepper(11)
StepPins = [IN1,IN2,IN3,IN4]

# Define sequence
# as shown in manufacturers datasheet
Seq = [[1,0,0,1],
       [1,0,0,0],
       [1,1,0,0],
       [0,1,0,0],
       [0,1,1,0],
       [0,0,1,0],
       [0,0,1,1],
       [0,0,0,1]]

StepCount = len(Seq)
StepDir = 1
WaitTime = 0.01
StepCounter = 0

while True:

  print(StepCounter)
  print(Seq[StepCounter])
  for pin in range(0, 4):
    xpin = StepPins[pin]
    if Seq[StepCounter][pin]!=0:
      xpin.on()
    else:
      xpin.off()

  StepCounter += StepDir

  # If we reach the end of the sequence
  # start again
  if (StepCounter>=StepCount):
    StepCounter = 0
  if (StepCounter<0):
    StepCounter = StepCount+StepDir

  # Wait before moving on
  time.sleep(WaitTime)
```

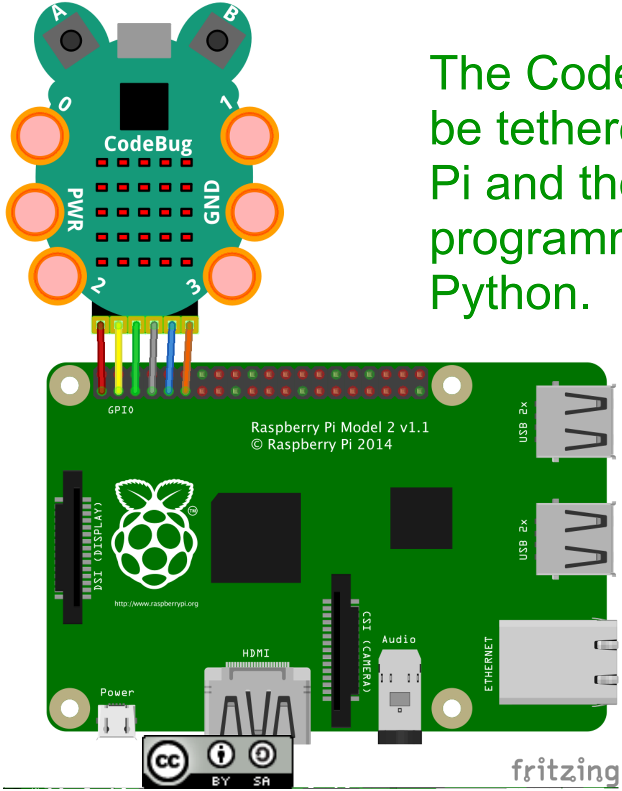
How can you make the motor turn the opposite way?

Can you make the motor turn faster?

How many steps will return the motor to its starting position?

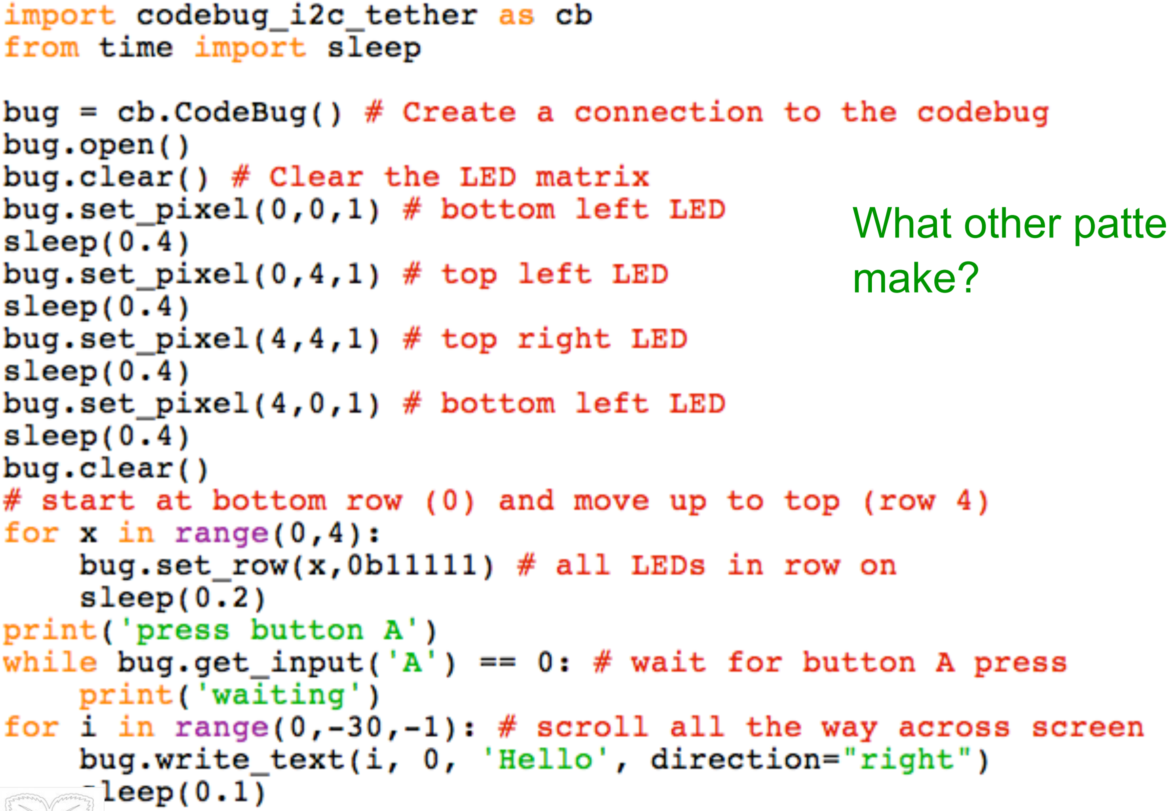
CODEBUG python™

It has 2 buttons (the eyes), 6 input/outputs (legs) and a 5x5 LED matrix



The CodeBug can be tethered to the Pi and then programmed using Python.

You can sit the CodeBug onto the GPIO pins rather than use wires



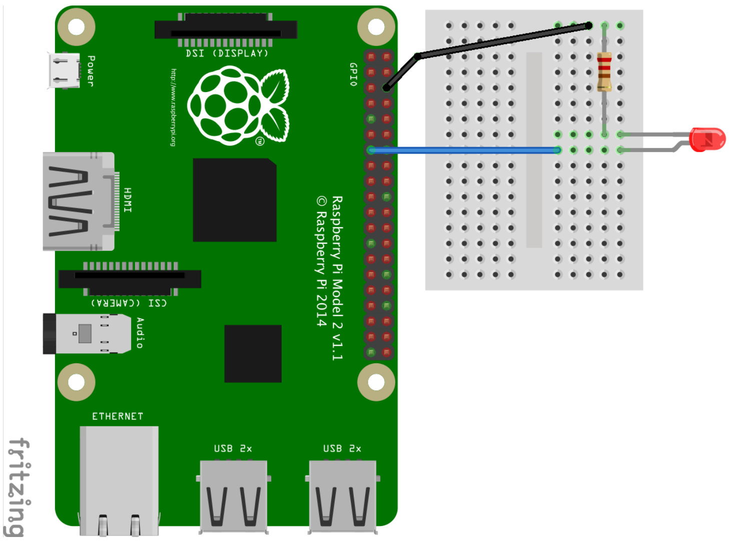
```python
import codebug_i2c_tether as cb
from time import sleep

bug = cb.CodeBug() # Create a connection to the codebug
bug.open()
bug.clear() # Clear the LED matrix
bug.set_pixel(0,0,1) # bottom left LED
sleep(0.4)
bug.set_pixel(0,4,1) # top left LED
sleep(0.4)
bug.set_pixel(4,4,1) # top right LED
sleep(0.4)
bug.set_pixel(4,0,1) # bottom left LED
sleep(0.4)
bug.clear()
# start at bottom row (0) and move up to top (row 4)
for x in range(0,4):
    bug.set_row(x,0b11111) # all LEDs in row on
    sleep(0.2)
print('press button A')
while bug.get_input('A') == 0: # wait for button A press
    print('waiting')
for i in range(0,-30,-1): # scroll all the way across screen
    bug.write_text(i, 0, 'Hello', direction="right")
    sleep(0.1)
```

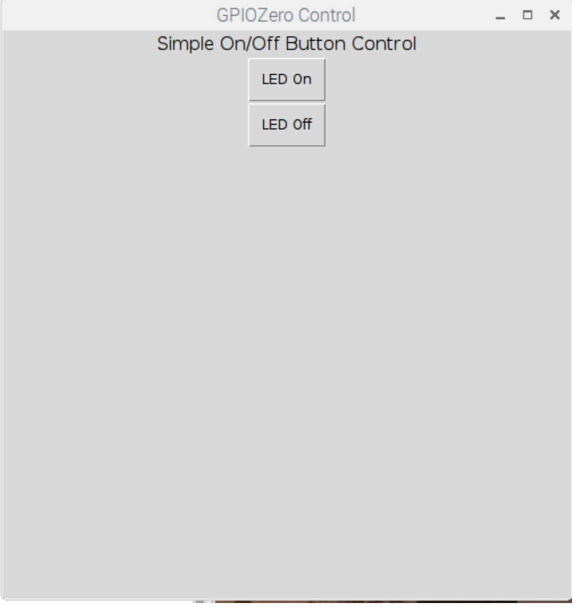
What other patterns can you make?

Can you make the text move faster?

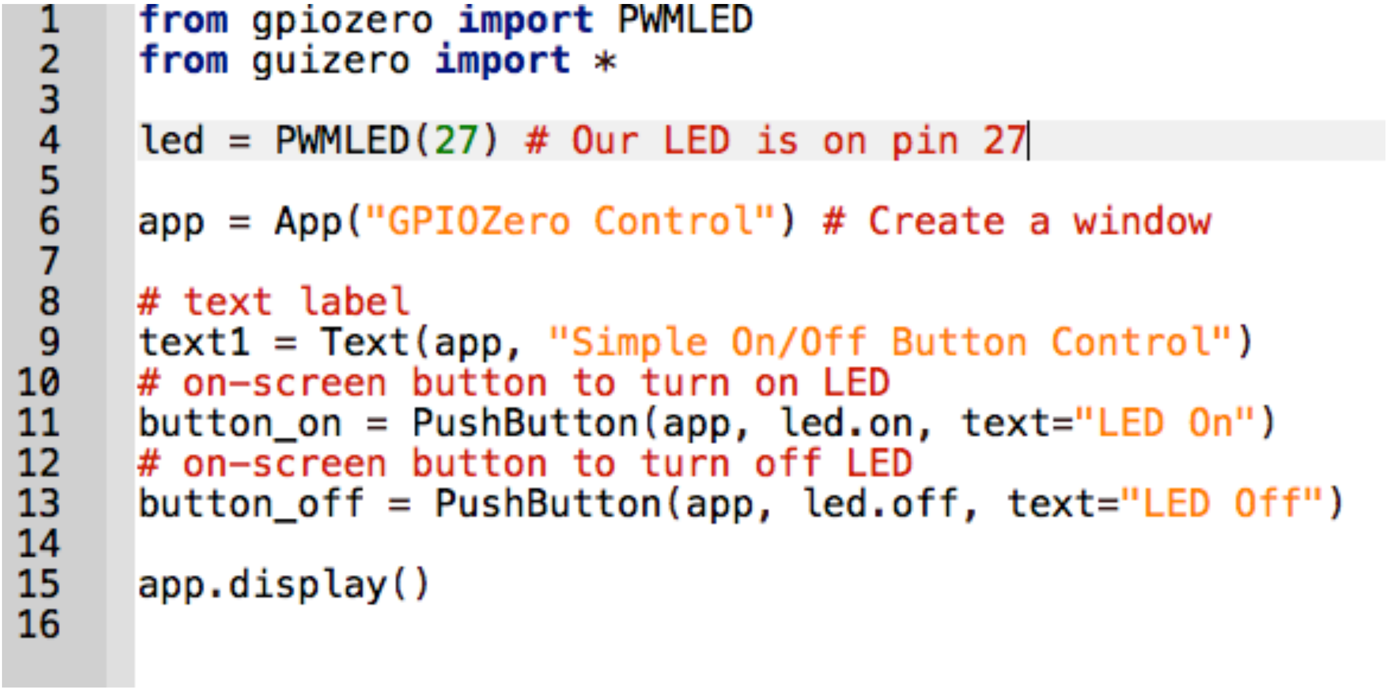
V 1.0

A SIMPLE GUI

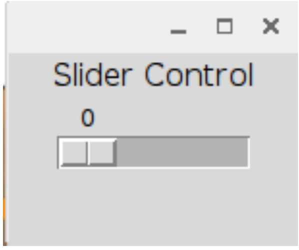




Let's use the guider library to make a simple GUI to control our LED circuit



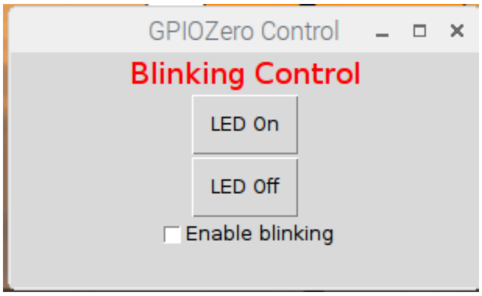
```
from gpiozero import PWMLED
from guizero import *

led = PWMLED(27) # Our LED is on pin 27

app = App("GPIOZero Control") # Create a window

# text label
text1 = Text(app, "Simple On/Off Button Control")
# on-screen button to turn on LED
button_on = PushButton(app, led.on, text="LED On")
# on-screen button to turn off LED
button_off = PushButton(app, led.off, text="LED Off")

app.display()

```



Now try a slider for LED brightness. This time we'll make the window smaller too.

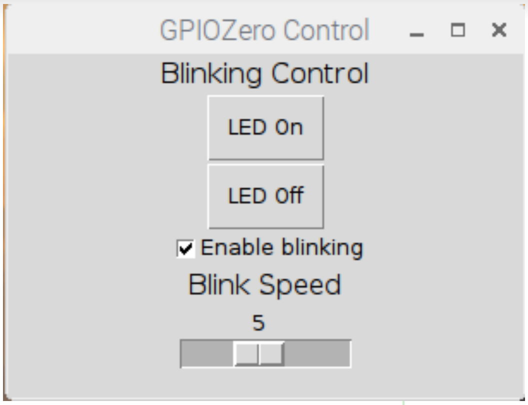
```
from gpiozero import PWMLED
from guizero import *

led = PWMLED(27) # Our LED is on pin 27

def brightness(value):
    led.value = float(value)/10.0

# Create a window 100 x 150 pixels
app = App("GPIOZero Control", height=100, width=150)

# text label
text1 = Text(app, "Slider Control")

# Slider to control LED brightness
slider = Slider(app, start=0, end=10, command=brightness)
app.display()

```

How about a checkbox and some coloured text?



```
from gpiozero import PWMLED
from guizero import *

led = PWMLED(27) # Our LED is on pin 27

blink_mode = False

def ledcontrol():
    global blink_mode
    if blink_mode:
        led.blink(on_time=0.5, off_time=0.5, background=True)
    else:
        led.on()

def set_mode():
    global blink_mode
    if blink_mode:
        blink_mode = False
    else:
        blink_mode = True

app = App("GPIOZero Control", height=150, width=300) # Create a window
# on-screen button to turn on LED
text1 = Text(app, "Blinking Control", color='red', size=14)
button_on = PushButton(app, ledcontrol, text="LED On")
# on-screen button to turn off LED
button_off = PushButton(app, led.off, text="LED Off")
# check-box to enable blinking of LED when switched on
checkbox = CheckBox(app, "Enable blinking", command=set_mode)


app.display()
```

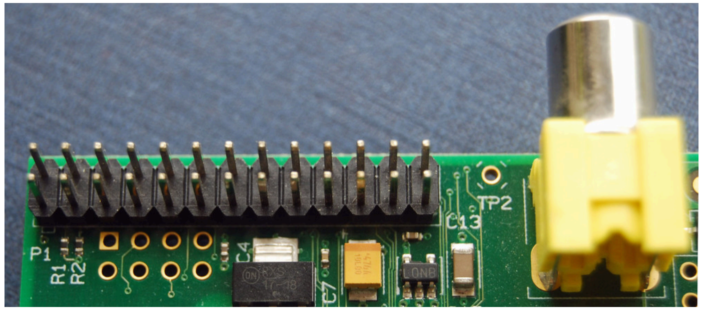
Put it all together!



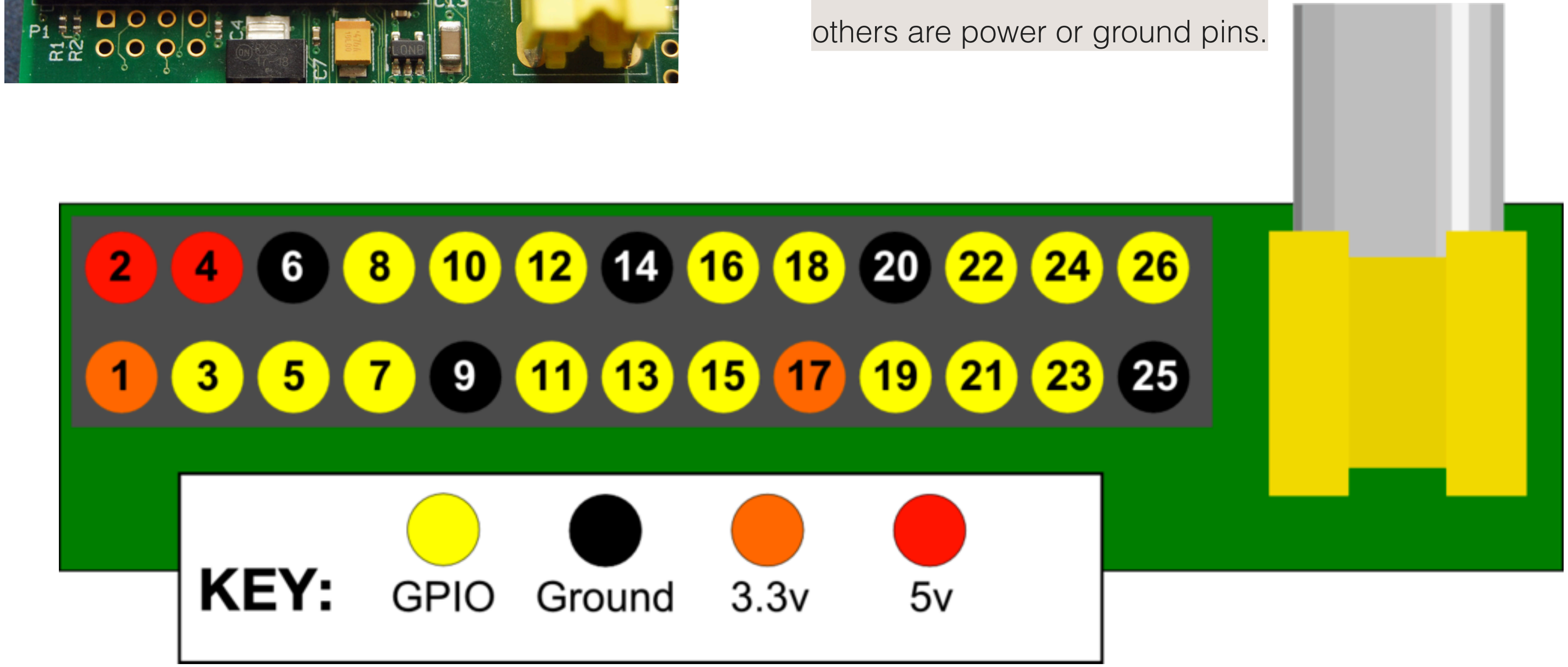
```python
from gpiozero import PWMLED
from guizero import *
from time import sleep

led = PWMLED(27) # Our LED is on pin 27

# variables to keep track of what's going on:
blink_mode = False # is blink mode enabled?
led_active = False # is the led turned on (in any mode)?
blink_freq = 0.5 # frequency of blinking
# these will be set as global variables by functions that use them

def ledcontrol(): # turns the led on or starts it blinking
    global blink_mode
    global blink_freq
    global led_active
    led_active = True
    if blink_mode:
         led.blink(on_time=blink_freq, off_time=blink_freq, background=True)
    else:
        led.on()

def led_turn_off(): # turns led off and stops blinking
    global led_active
    led_active = False
    led.off()

def set_mode(): # sets whether blink mode is on
    # this function is run whenever the box is checked or un-checked
    global blink_mode
    if blink_mode:
        blink_mode = False
    else:
        blink_mode = True

def speed(speed): # set frequenecy of blinking
    # this function is run whenever the slider is moved
    global blink_freq
    global led_active
    blink_freq = 1/float(speed) #freq between 0.1 and 1 second
    print(blink_freq)
    if (blink_mode and led_active):
        led_turn_off()
        led_active = True
        led.blink(on_time=blink_freq, off_time=blink_freq, background=True)


app = App("GPIOZero Control", height=200, width=300) # Create a window
# on-screen button to turn on LED
text1 = Text(app, "Blinking Control") # text label
button_on = PushButton(app, ledcontrol, text="LED On")
# on-screen button to turn off LED
button_off = PushButton(app, led_turn_off, text="LED Off")
# check-box to enable blinking of LED when switched on
checkbox = CheckBox(app, "Enable blinking", command=set_mode)
text2 = Text(app, "Blink Speed") # text label
# Slider to control LED brightness
slider = Slider(app, start=1, end=10, command=speed)
app.display()
```

Challenge: Create a GUI to control the colour of an RGB LED!

GPIO (GENERAL PURPOSE INPUT OUTPUT)



These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (**input**) or that the Pi can turn on or off (**output**). Seventeen of the 26 pins are GPIO pins; the others are power or ground pins.



Randomly plugging wires onto your GPIO will kill your PI!

MINECRAFT PI EDITION BLOCKS

AIR	0
STONE	1
GRASS	2
DIRT	3
COBBLESTONE	4
WOOD_PLANKS	5
SAPLING	6
BEDROCK	7
WATER_FLOWING	8
WATER	8
WATER_STATIONARY	9
LAVA_FLOWING	10
LAVA	10
LAVA_STATIONARY	11
SAND	12
GRAVEL	13
GOLD_ORE	14
IRON_ORE	15
COAL_ORE	16
WOOD	17
LEAVES	18
GLASS	20
LAPIS_LAZULI_ORE	21
LAPIS_LAZULI_BLOCK	22
SANDSTONE	24
BED	26
COBWEB	30
GRASS_TALL	31
WOOL	35
FLOWER_YELLOW	37
FLOWER_CYAN	38
MUSHROOM_BROWN	39
MUSHROOM_RED	40
GOLD_BLOCK	41
IRON_BLOCK	42
STONE_SLAB_DOUBLE	43
STONE_SLAB	44
BRICK_BLOCK	45
TNT	46
BOOKSHELF	47
MOSS_STONE	48
OBSIDIAN	49
TORCH	50
FIRE	51
STAIRS_WOOD	53
CHEST	54
DIAMOND_ORE	56
DIAMOND_BLOCK	57
CRAFTING_TABLE	58
FARMLAND	60
FURNACE_INACTIVE	61
FURNACE_ACTIVE	62
DOOR_WOOD	64
LADDER	65
STAIRS_COBBLESTONE	67
DOOR_IRON	71
REDSTONE_ORE	73
SNOW	78
ICE	79
SNOW_BLOCK	80
CACTUS	81
CLAY	82
SUGAR_CANE	83
FENCE	85
GLOWSTONE_BLOCK	89
BEDROCK_INVISIBLE	95
STONE_BRICK	98
GLASS_PANE	102
MELON	103
FENCE_GATE	107
GLOWING_OBSIDIAN	246
NETHER_REACTOR_CORE	247