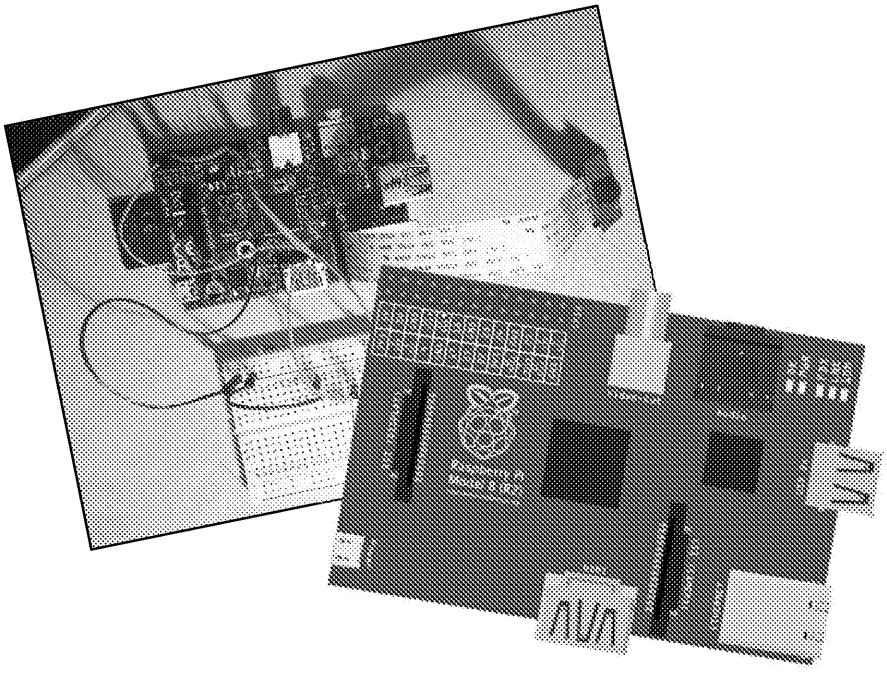
Raspberry Pi Projects

Using Python v3



POD 5826

AE10.5826

computing@zigzageducation.co.uk
zigzageducation.co.uk

Contents

Teacher's Introduction

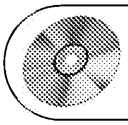
This resource contains a variety of projects aimed at a range of ability levels. The projects are designed to be interactive and fun. They start at an easy level such as *Benchmarking the Raspberry Pi*, and progress to much more complex projects such as creating a *Digital Camera*. Each project is marked with a time duration (short, medium or long) and a difficulty rating (easy, medium or hard).

Each project also includes an extension section and a 'stretch yourself' section. These are intended for students who wish to challenge themselves further after completing the main project tasks.

Some projects are dependent on (or at least refer to information in) other projects. If students are going to start using the add-ons to the Raspberry Pi, the project they need to start with, which has all the main information in it about this, is the *Binary Conversion*. This explains how to use breadboards, LEDs and buttons.

Important:
Extreme care should be taken when creating circuits with breadboards, jumper leads and resistors.

At the back of the resource are answers and solutions for every challenge. The full python code for each exercise is not printed but is included on the CD-ROM.



The accompanying CD-ROM includes full python scripts for each project, plus videos demonstrating three of the projects in action.

Prior Knowledge
The projects make use of the **Python (v3)** programming language, and students will require prior understanding of Python to complete the tasks. They will need to know how to:

- Create an input for the user
- Output any information the user needs
- Create a sequence
- Use selection including conditions
- Create both counting loops and condition loops

ZigZag Education have resources to support teaching and learning of Python – go to **zzed.co.uk** (search for 'Python') for more details.

February 2015

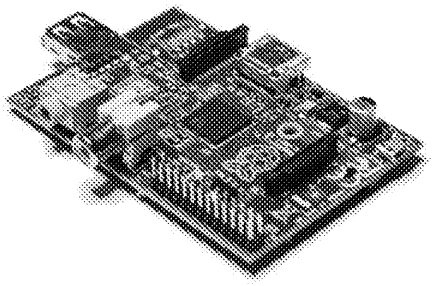
Introduction to the Ras

The Raspberry Pi is a credit-card-sized computer developed in
Raspberry Pi Foundation. It is a very capable little computer th
fun electronic projects, some of which have been included in t

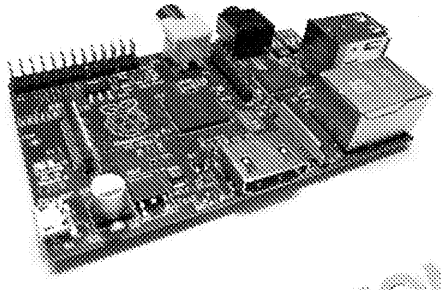
One of the great things about the Raspberry Pi is that there is
is a flexible platform that can be used for anything from creati
making a camera. The low price of the Ras, ry Pi has meant
experiment in creating a wealth of d ife t projects.

Th spberry Pi

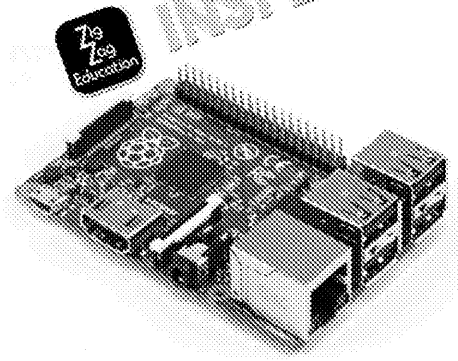
There are three main models of the Raspberry Pi:



Model A
This is the lower-spec mo
It is a lighter model that c
is often used in embedded
reasons. The model A do
chip.

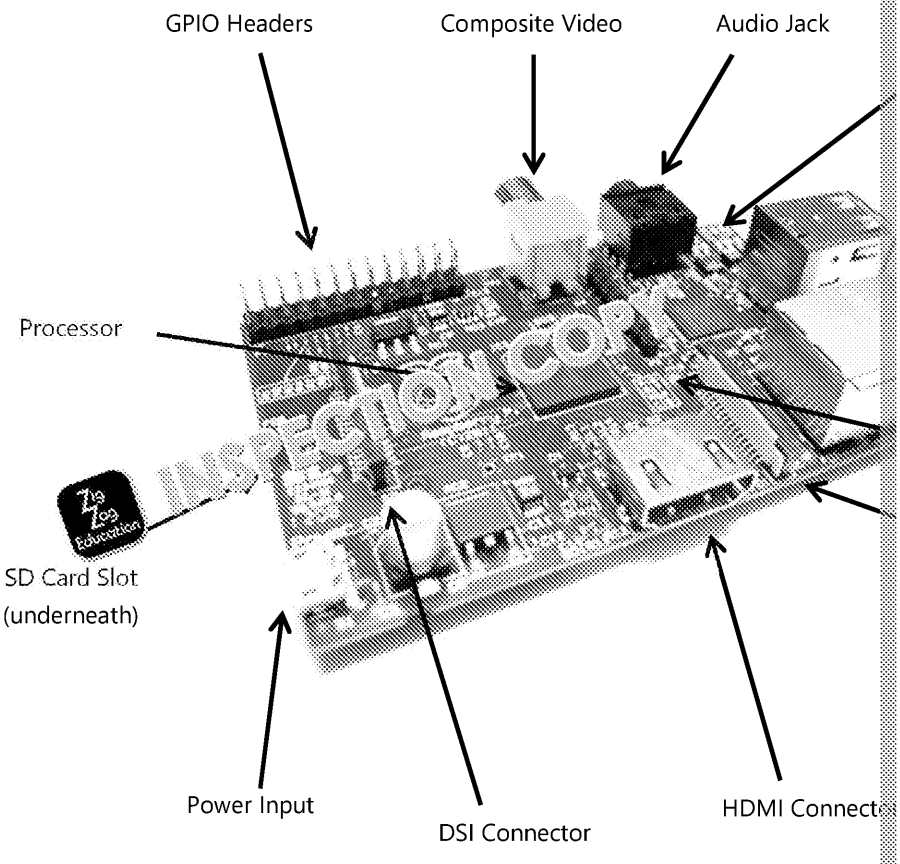


Model B
This is the original model
is higher in spec that the
does have an Ethernet chi



Model B+
This is the highest-spec m
Pi. It replaced the model
many additional extras to
more USB ports, more GPI
consumption and better a

A Closer Look at the Raspberry Pi (Model B)



PROCESSOR

This is an ARM chip that is 32-bit and 700 MHz system. It has

GENERAL PURPOSE INPUT AND OUTPUT (GPIO) HEADERS

The GPIO headers consist of two rows of 13 pins that can be c
and outputs. You can program the Raspberry Pi to say which t

COMPOSITE VIDEO

This is a standard RCA connector that provides video signals.
standard-definition video. For higher-definition video the HDM
used.

AUDIO JACK

This is a standard 3.5mm analogue audio jack. It can be u
head
speakers.

STATUS LEDS

There are five LEDs that provide feedback about the system:

- ACT – This is a green LED that lights up when the SD ca
- PWR – This is a red LED that lights up when power is co
- FDX – This is a green LED that lights up if the Ethernet co
- LNK – This is a green LED that lights up when Ethernet i
- 100 – This is a yellow LED that lights up when the conne

USB PORTS
There are two USB 2.0 ports on the model B (one on the mode
to connect a mouse and keyboard.

ETHERNET PORT
This can be used to connect the Raspberry Pi to the internet. A
also be used in a USB port to do this.

JOINT TEST ACTION GROUP (JTAG) HEADERS
These headers are not generally used when using the Raspberr
programming, testing and debugging the device in production

THE CAMERA SERIAL INTERFACE (CSI) CONNECTOR
This port allows a camera cable to be connected directly to t

HIGH-DEFINITION MULTIMEDIA INTERFACE (HDMI) CONNECTOR
This provides digital video and audio output and enables a mo
The HDMI signal can be converted to DVI, composite or SCART
adapters if needed.

DISPLAY SERIAL INTERFACE (DSI) CONNECTOR
This is used to connect a 15-pin flat ribbon cable that can be u
screen.

POWER INPUT
This allows power to be connected using a micro USB.

SECURE DIGITAL (SD) CARD SLOT
This is used to insert an SD card that will store the operating sy
Raspberry Pi.

The Basic Kit

This is the basic set of kit that you will need when using the Ra

USB KEYBOARD AND MOUSE
A USB connection is needed on a keyboard and mouse to be a
to the Raspberry Pi.

A MICRO USB POWER SUPPLY
This should supply 5 V and at least 700 mA of current. An und
Pi might still work but its reliability will decrease.

AN SD CARD
An SD card that is 4 GB or more will be needed to hold the ope
files and programs that are created.

AN HDMI CABLE
This is used to connect to a monitor. If the monitor does not h
connection then a converter will be needed to a connection th

ETHERNET CABLE / WIRELESS DONGLE

This is used to connect the Raspberry Pi to a network.

USB HUB

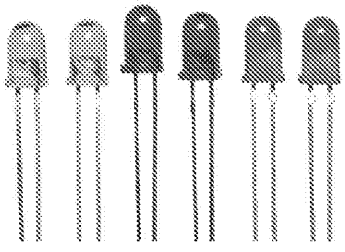
There may be a number of USB connections that need to be m
Pi, including a mouse and keyboard. It is advisable to have a U
Raspberry Pi only provides two USB ports.

MONITOR

A monitor with an HDMI connector is best; otherwise a conver

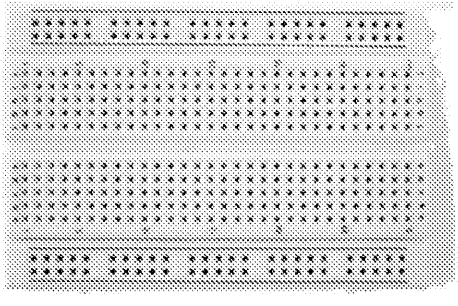
Raspberry Pi Add-ons

The[illegible] number of add-ons to the Raspberry Pi that ar
som[illegible]e projects. These include:



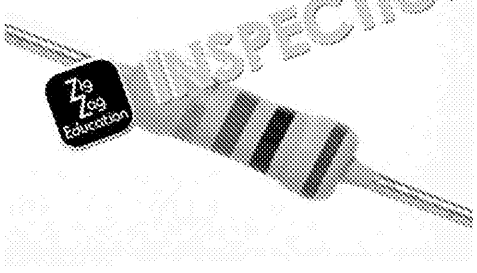
LIGHT-EMITTING DIODES (L

LEDs are used in a numb
a way of emphasising an
programming with the Ra



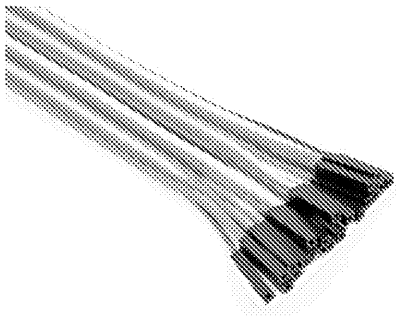
BREADBOARD

These are used to create
that they can be connecte
with the Raspberry Pi.



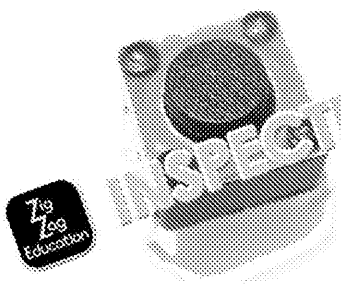
RESISTOR

These are used in breadb
circuits using LEDs. You
one.



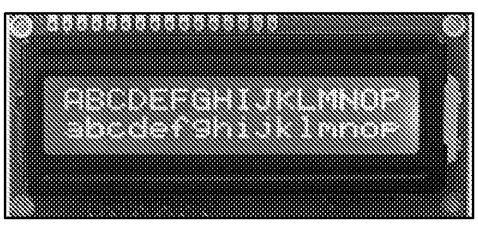
JUMPER WIRES

These are used for crea
on breadboards.



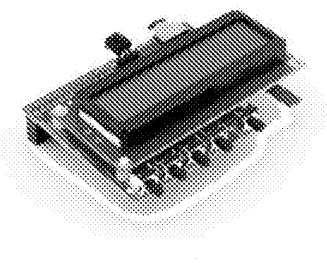
BUTTONS

These are buttons that p
to create inputs for some



LCD SCREEN

These can be purchased
the Raspberry Pi and are
projects as a fun output.



PI-FACE

This is an add-on that pl
Raspberry Pi. It has but
and LEDs built into it tha

PI CAMERA

This is a camera board t
the Raspberry Pi to crea

Important:
Extreme care should be taken when creating circuits with breadboards, j

Benchmarking the Raspe

DURATION: SHORT
DIFFICULTY: EASY

In this project we are going to compare the relative speed of t
using the Command Line versus the Graphical User Interface.

Project Aims

- To provide a benchmar
comparison of th
differen between using the
m d line and graphical user
face to run Python programs
on the Raspberry Pi
- To calculate the difference in speed between using the command line and the graphical user interface
- To understand how much processing power it takes to run a graphical user interface
- To learn how to use Python to create timestamps that can be used for benchmarking tests
- To provide a benchmark comparison between the processing power of the Raspberry Pi and a desktop computer

Project Outcom

- It will take a t
starting to co
- It will count fr
outputting ea
screen
- It will automa
time taken to
and output it
- It will allow th
benchmark vi
line and the g
interface
- It must be abl
desktop PC fo

Project Equipme

- Basic Raspber
- Desktop PC

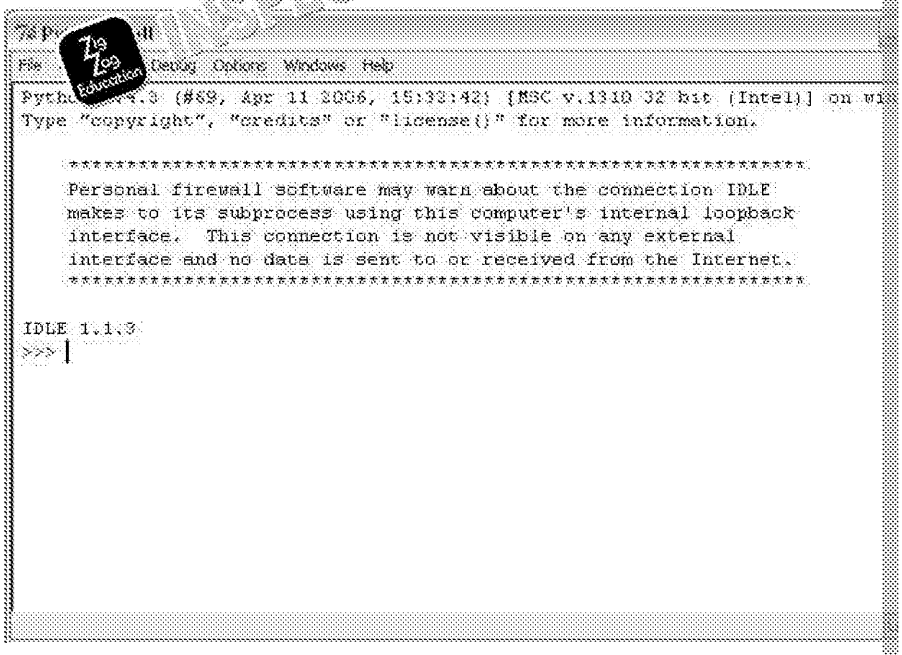
Project Tasks

Project Tasks

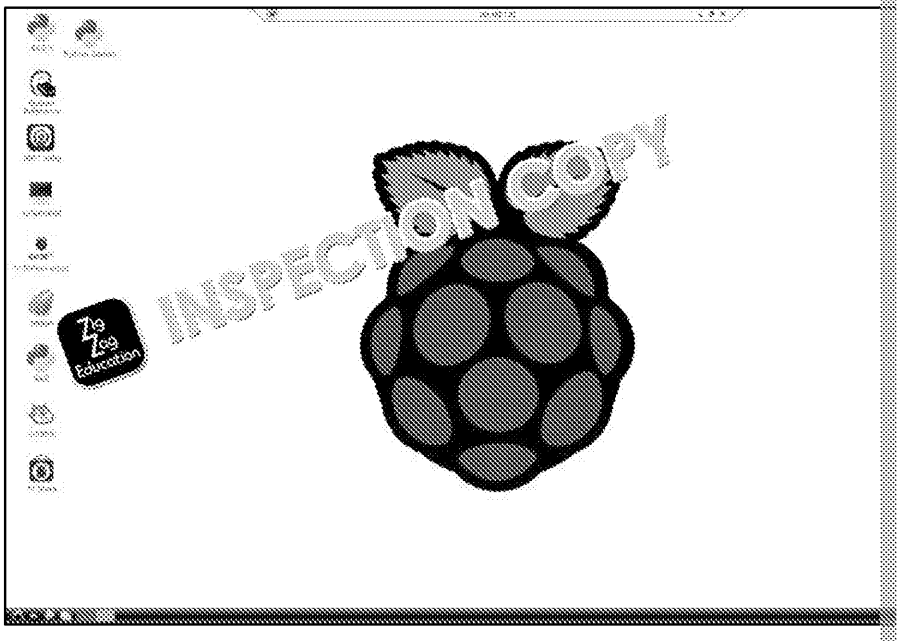
A) Research – Command Line Interface (CLI) and Graphical User Interface (GU

- What are they?
- What is the difference between the two?
- Name a situation when each might be used.
- What are the benefits and drawbacks of using each?

Command Line Interf



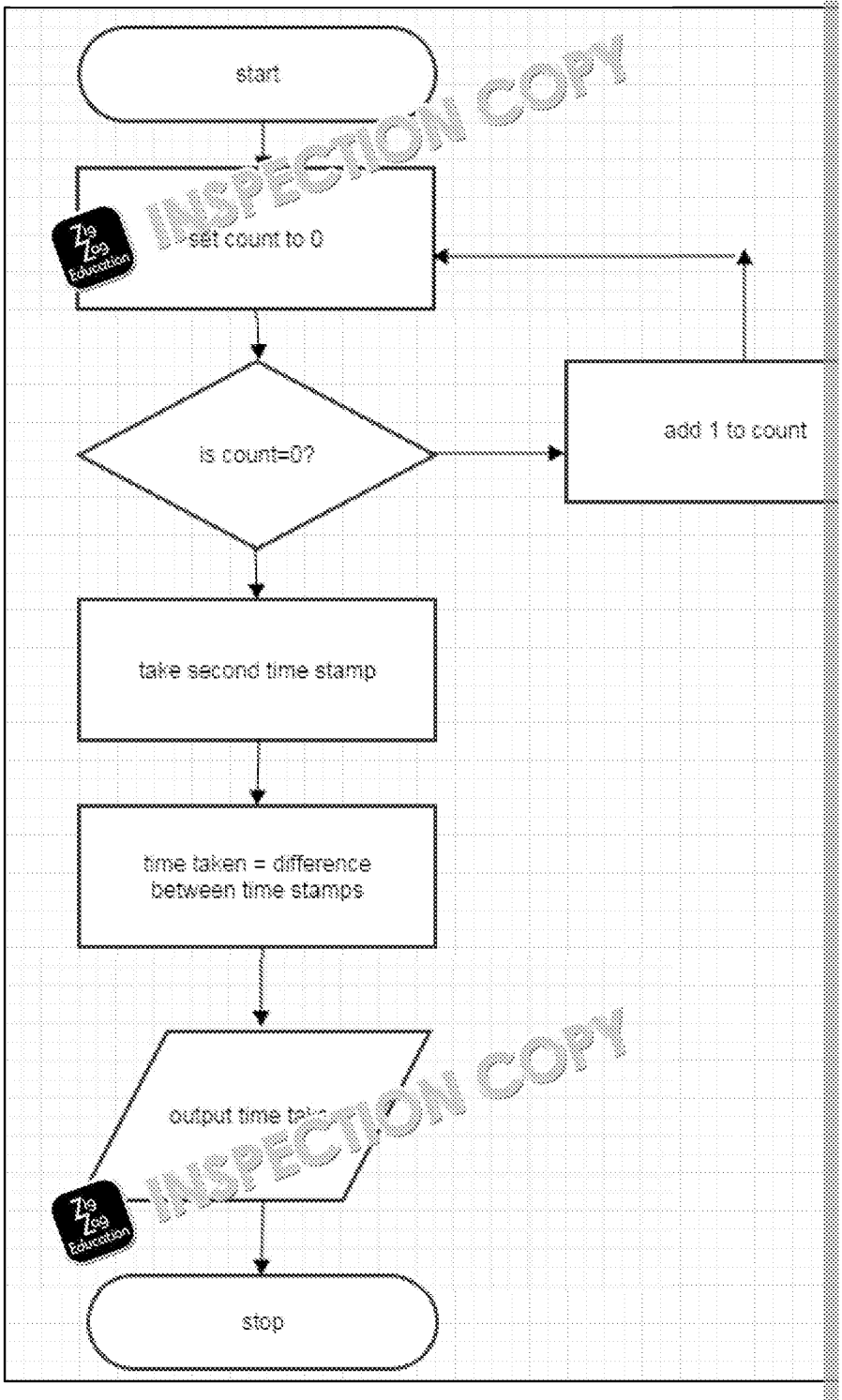
Graphical User Interface



B) Design the program

We are going to create a program to benchmark the time take
Raspberry Pi to run a task in the CLI, against the time taken for
GUI. The benchmark task will be a count from 1 to 10,000.

Here is a flow chart for an algorithm to solve this task. There ar
three things wrong with it.



Challenge: Can you fix it?

C) How to take a timestamp in Python using the time module

To take a timestamp we need to use the **time.time** function in

```
import time                                  #imports the t
time_stamp_1 = time.time()                   #takes the 1st
                                             assigns it to
                                             time_stamp_1
```

D) Build the program

Below is the main program. The section shown will take two tim
before performing the count, one after.

```
import tim
time_stamp_1 = time.time()                   #takes the 1st
                                             assigns it to
                                             time_stamp_1

for count in range(1,10000):
    print (count)                            #prints all th
                                             10,000


time_stamp_2 = time.time()                   #takes the 2nd
                                             assigns it to
                                             time_stamp_2
```

The end of the program is missing. This should be a calculation
perform the count. The result of calculation should then be ou

Challenge: Can you work out how to do this?

E) Benchmarking

- Run your program in the GUI using IDLE. Make a note o
  taken.
- Run your program in the CLI. Make a note of the time ta
- Calculate the difference in speed both in time and as a p
- What do you notice?
- Can you explain your findings?

Extension

- When running the program via the GUI, move the mous
- What do you notice?
- Can you explain the difference?

STRETCH YOURSELF

- Run your Python prog[illegible] on a desktop PC.
- How [illegible] PC benchmark compared with the Raspberry Pi?
- [illegible] you explain the difference?

Remotely Connecting to a Raspberry Pi

DURATION: SHORT
DIFFICULTY: MEDIUM

For this project we are going to compare, connect to and control
our Windows PC. Being able to connect remotely allows us place
unobtrusively or in a place we cannot easily access while still bein

Project Aims

- Learn how set up a remote connection to the Raspberry Pi.
- To learn how to determine the Raspberry Pi's IP address on the network.
- To learn how to install and configure TightVNC software needed for the connection.
- To learn how to access and control the Raspberry Pi from a Windows PC.

Project Outcom

- The software T
  downloaded o
- A remote con
  to the Raspbe

Project Equipment

- Basic Raspberry Pi kit connected (wired or wirelessly) to y
- Desktop PC

Project Tasks

A) Research – why might we want to be able to access our Raspbe

B) D in the Raspberry Pi's network address .....

C) Ins TightVNC on the Raspberry Pi .....

D) Configure TightVNC to allow remote connections .....

E) Install TightVNC on the Windows PC .....

F) Connecting to and controlling the Raspberry Pi .....

Project Tasks

A) Research – why might we want to be able to access our Raspberry Pi

Using the Internet, research and list five possible scenarios whe
needed to be able to access their Raspberry Pi remotely. Think
do with a Pi if it is connected wirelessly to your network and pc
instead of the mains power pack.

B) Determine the Raspberry Pi's network add

*Every computer on a net[illegible] has a unique network address. Thi
known as the c[illegible] as **IP address** and is needed so that other
net[illegible] where to send messages for our computer. Think
our [illegible] address, or even our phone number. Without the addr
our friends wouldn't know how to contact us. In fact, on a netwo
needs a network connection has an IP address, even devices suc
consoles, smartphones and smart TVs.*

Before we can use another computer to control our Raspberry
our Pi's IP address. Finding it is easy:

Either via the command line, or via an LXTerminal Window, ent

```
ifconfig
```

Done correctly, this will generate a message along the lines of

```
Link encap:Ethernet  HWaddr b8:27:eb:5f:f6:bb
inet addr:192.168.1.89  Bcast:192.168.1.255  Mask 255.2
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:252 errors:0 dropped:0 overruns:0 frame:0
TX packets:100 errors:0 dropped:0 overruns:0 carrier:0
Collisions:0 txqueuelen:1000
RX bytes:37493 (36.6 KiB)  TX bytes:9[illegible] (9.3 KiB)
```

This information tells us lots [illegible] ful information about the Pi
However, the inform[illegible] on we need sits in the second line:

```
ine[illegible]:[illegible].168.1.89  Bcast:192.168.1.255  Mask 255.2
```

The IP address of our Raspberry Pi is the first collection of four
192.168.1.89. Make a note of whatever address is given here as
tell our Windows PC where to send messages to talk to our Ra

C) Install TightVNC on the Raspberry Pi

To control our Raspberry Pi from another computer we need a r ***application****. This will run on both our Raspberry Pi and on the V will use to remotely control the Pi. The application forms the cor remote control of the Pi.*

One such application that works extremely well with the Raspbe freeware application allows and controls remote connections, ke secure from unauthorised users.

Installing TightVNC is simple. Either from the command line, o window, enter:

```
sudo apt-get install tightvncserver
```

The software will take several minutes to install.

D) Configure TightVNC to allow remote connections

Once it is installed, we need to configure TightVNC with a pass allow us to connect remotely to our Raspberry Pi:

First, make sure startx is running, then open an LXTerminal win

Within the terminal window, enter:

```
tightvncserver
```

TightVNC will ask us to enter and confirm a password. Pick a p remember but difficult for someone else to guess! A mixture o characters is always a good method for a secure password.

Next, we will be asked whether we wish to enter a 'view-only' p password is used in case we wish someone to be able to view v our Pi, but not to control it. Enter 'y' if you want such a facility,

Once the password has been successfully entered, TightVNC w configuration message containing **a port number**:

```
New desktop at tcp:raspberrypi:1
```

In this case, the port number is '1'. Make a note of this port nu the information to remotely connect to the Pi.

NOTE: Every time we restart the Raspberry Pi we will need to again. Always make a note of the port number.

E) Install TightVNC on the Windows PC

TightVNCServer on the Raspberry Pi forms one end of the con
need. To form the other end of the connection we need to inst
Windows PC:

Using the Internet, visit the website *http://www.tightvnc.com*

Download and install the appropriate version of TightVNC on t
the 32-bit version if unsure which is the right version to use. W
'Typical' settings. TightVNC runs on all versions of Windows fro

When prompted, enter and confirm the same password you us
TightVNC on the Raspberry Pi. Use the same password again a
password, or choose something else if you wish. The installatio
com

F) Connecting to and controlling the Raspberry Pi

To connect to and remotely control our Raspberry Pi, we need
TightVNC Viewer on our Windows PC. We will need two pieces
gathered previously:

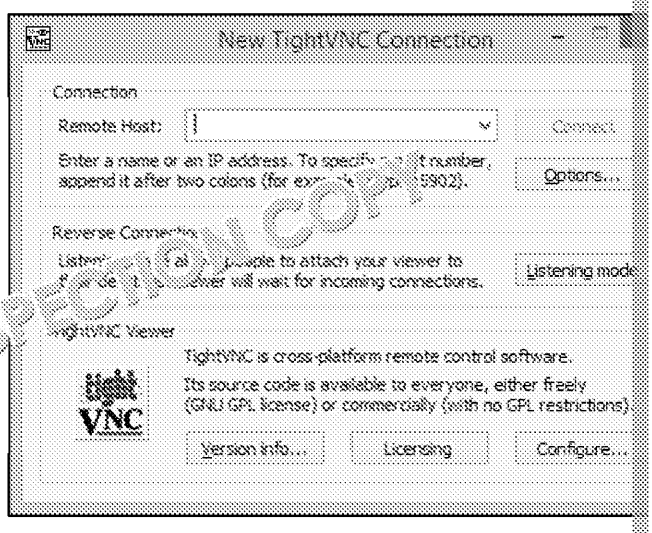
- The IP address of the Raspberry Pi:

```
inet addr:192.168.1.89  Bcast:192.168.1.255  Mask 255.2
```

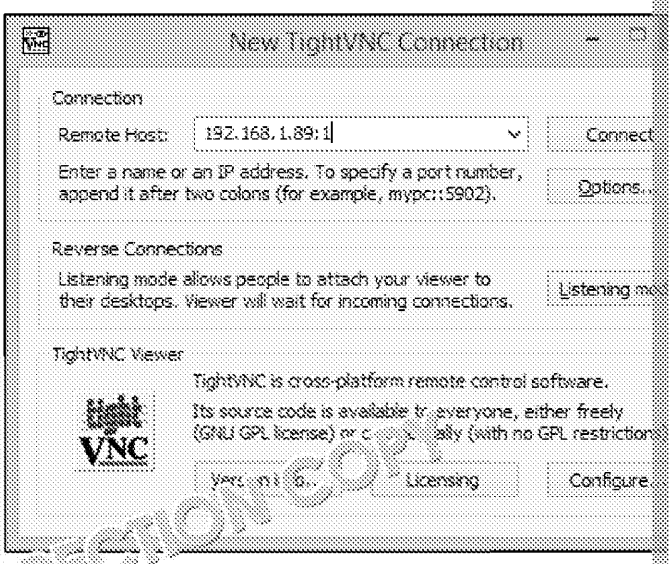
- The port number from TightVNC on the Raspberry Pi:

```
New 'X' desktop is raspberrypi:1
```

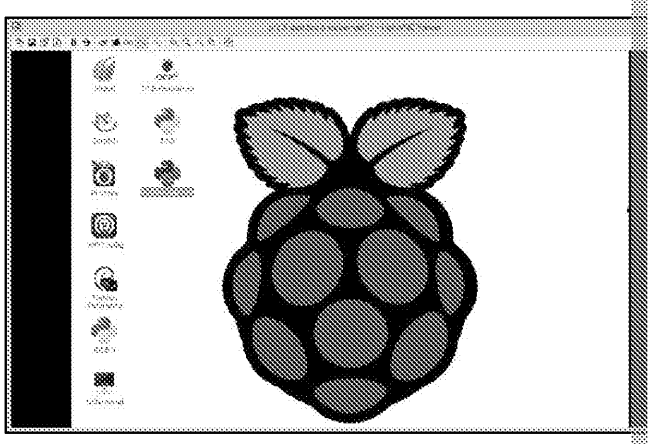
When run, TightVNC Viewer will show this menu:



TightVNC Viewer needs to know the **IP address and the port n
with the Raspberry Pi.** All we need to do is enter the address a
'**Remote Host**' box. The port number follows the IP address, an
separated by a colon. In the example shown here, the details a



Now 'C nect', enter the **TightVNC password for the Ras** have rol of our Pi!



Using OSX or Linux to connect

When using OSX or Linux, the process is the same except that available for these platforms. Try a different freeware applicati the VNC' instead.

Scratch Yourself

- t nect to the Raspberry Pi with an iPad, Android tablet -style app. Many are available as freeware.
- Combine this project with the Pi Camera project to create a wirel battery-powered CCTV camera that can be operated remotely.
- Find out how to change the Raspberry Pi TightVNC password.

Interactive Quiz

DURATION: SHORT
DIFFICULTY: EASY

In this project we are going to create an interactive quiz that w
input their answers and a score will be kept.

Project Aims

- To learn how to program an interactive quiz
- To learn how to create an input for the user to input an answer
- To use the GPIO interface to light up LEDs

Project Outcom

- To choose a t
- To create an i has five multi
- To keep a sco
- To repeatedly a correct ansv
- To output a fi
- To use LEDs t or incorrect a

Project Equipment

- Basic Raspberry Pi kit

Project Tasks

A) Choosing a theme .....

B) Writing the program .....

Extension .....

Project Tasks

A) Choosing a theme

We need to choose a theme for our quiz. Choose a theme tha
familiar with, something you enjoy. We need to think of five q
theme and three multiple-choice answers for each. Plan your
to your quiz.

B) Writing the program

Now we have planned our qu... ca...go ahead and program
Once we have learnt h...w ...ogram one question, the rest ar

Open P...th... ...s will open into the Python shell. Open a ne
you... This way you can program your quiz in the new win
questions appear in the shell. To run the program you just nee

To create the code for one question, we would enter the follow

```
_______ ("Welcome to my quiz")
score = 0
print ("Question 1")
print ("A. option 1\n"
       "B. option 2\n"
       "C. option 3\n")
answer1 = _________("answer: ")
if _________ == "A":
        print("Correct!")
        score = _________ + 1
_________:
        print("Incorrect!")
```

Challenge: Ca
each missing p

The score is set to 0 to begin with and each time we get a que
increases by 1.

We repeat this process for each qu... ...until we have comple

We can make our p... ran ...more reliable by taking the input a
convert... ...lower case. This means that if they typed 'A'
gen... the correct response. We can do this by using the lo

```
if lower.answer1 == "A":
```

Once all questions have been asked, output the users score us

```
print (" Your score is", str(score))
```

Why do we need to convert the score variable to a string to pr
we do not do this?

Extension

Try to put each question into a condition loop so that the ques
repeated until the answer is correct. You can do this by using

You may need to research and find out how you can do this.

STRETCH YOURSELF

Find out how to create a GPIO circuit w... ...EDs, one green and on
(Hint: look at the 'Input/Ou... ...' project if you have it in front

Exp... yo... ...program so that the green LED lights up if the answ
re... ...s incorrect.

Input/Output Basi

DURATION: SHORT
DIFFICULTY: MEDIUM

For this project we are going to learn about the Raspberry Pi's
Purpose Input Output) interface and how we can use it to acce
and to output signals to components such as LEDs.

Project Aims

- To learn about the R[illegible]y Pi's GPIO interfa[illegible]
- [illegible]ar[illegible]how to use the GPIO [illegible]face as an output to light LEDs
- To learn how to use the GPIO interface as an input with buttons.
- To learn how to program inputs and outputs in Python

Project Outcom

- It will light up
- It will make a
- It will turn an [illegible] a button

Project Equipment

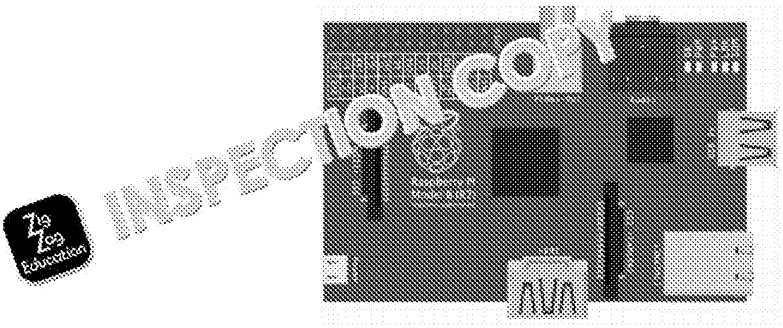
- Raspberry Pi kit
- Full-size breadboard
- 1 × LED
- 1 × button
- 3 × female-to-male ju
- 3 × male-to-male jum
- 1 × 470 ohm resistor

Project Tasks

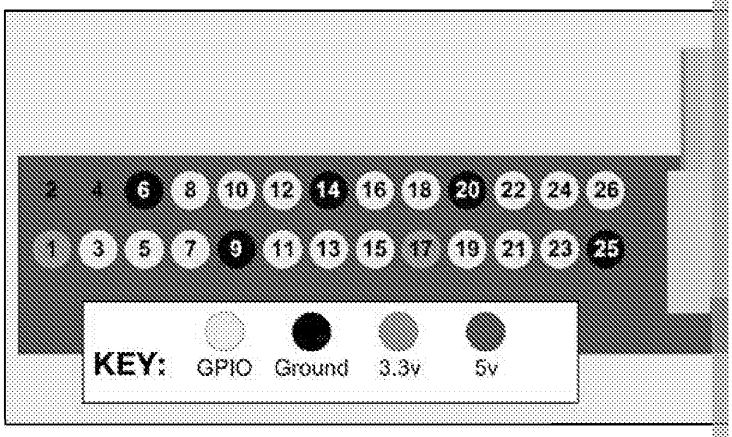
Project Tasks

A) Understanding the Raspberry Pi's GPIO interface

One of the many great things about the Raspberry Pi is the built (General Purpose Input Output) interface. This interface allows t with other devices and components. The GPIO interface is the se the edge of the board, along from the Video-Out port.



Each pin carries a signal in or out of the Pi. We can configure 17 inputs or outputs. The remaining pins are power or ground pins carefully. Power can be delivered at 3.3 V or 5 V, depending on t



NOTE: Be extremely careful wh u g the GPIO pins – conn wrong pins can dam

The ra ve a name/number, and there are two methods – 'BC d 'BOARD'. Board numbering assigns a number from and this is the method we will use.

REMEMBER: Pins 1, 2, 4, 6, 9, 14, 17, 20 and 25 are power/gro be used for inputs/outputs.

B) Using the GPIO to light an LED – set up the circuit

First we need to build the inputs and outputs.

We will need the following components:

- Raspberry Pi kit
- Full-size breadboard
- 1 × LED
- 1 × button
- 2 × female-to-male jumper leads
- 2 × male-to-male jumper leads
- 470 ohm resistor

Connect the components as follows:

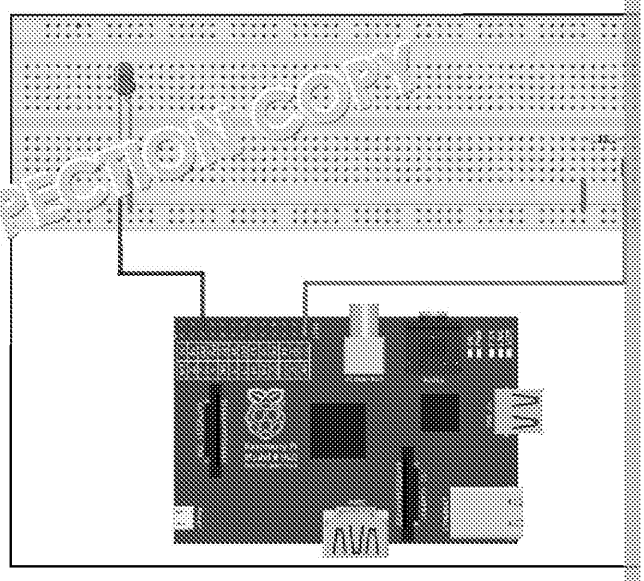
Place the resistor on the breadboard, one leg in **slot e55**, th

Using one male-to-female jumper lead, connect the female (ground), and the male end to **slot c59**. This connects the re

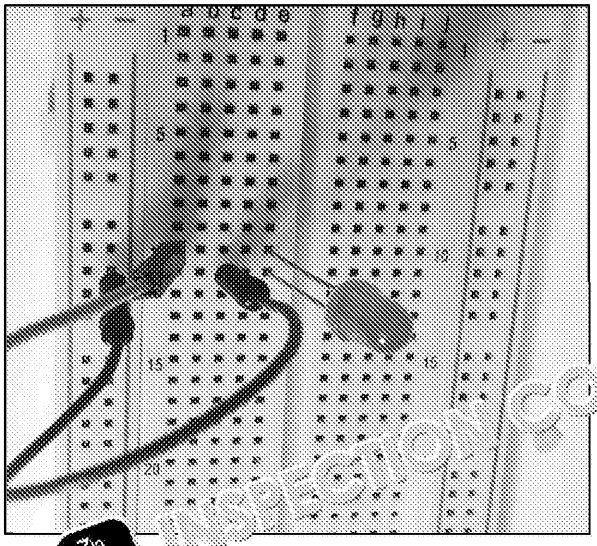
Using a male-to-male jumper lead, place one end in **slot a5 ground (blue) track at slot 55**. We now have a resistor-prot connection.

Now, place the remaining components and leads as follows:

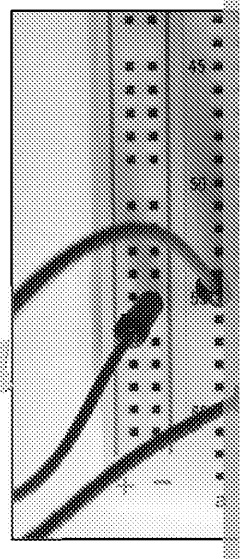
LED	LED slots	Ground lead slots	+ve
led1	e10, e11	c11, ground track 11	a10, 5



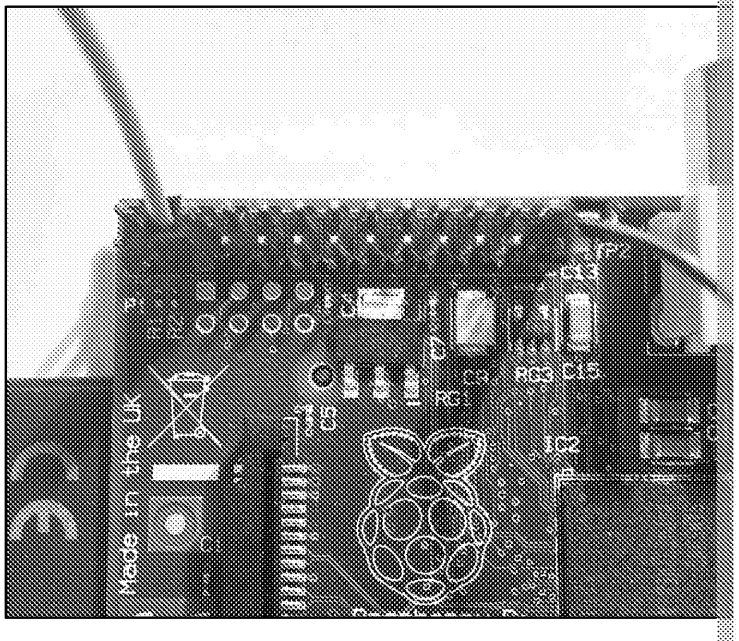
The LED should look like this:



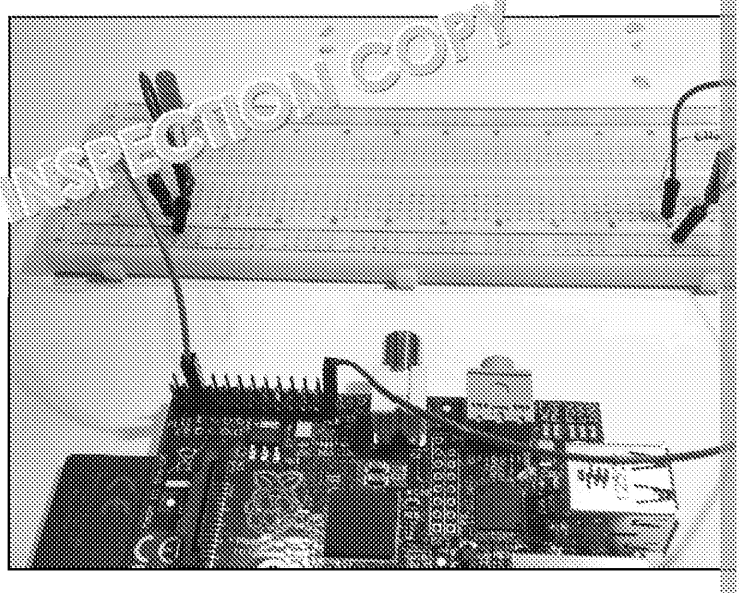
The resistor sh



The pins should look like this:



The whole circuit for an LED should look lik



C) Using the GPIO to light an LED – writing the code

We can use Python to control the GPIO pins. First we have to i
GPIO module, then state that we will be using a GPIO pin and
an input or output. The code below configures one pin, pin 5, a
with an LED).

```
import RPi.GPIO as GPIO              #imports GPIO module -
GPIO.setmode(GPIO.BOARD)             #set GPIO in 'board' m
GPIO.setwarnings(False)              #switches off warning

led1 = 5                             #assigns pin 5 to LED1
GPIO.setup(led1 = GPIO.OUT)          #configures pin 5 as a
```

To operate an LED we simply tell the pin to carry a current or no

```
GPIO.output(led1, 1)                 #turns LED1 on
```

Challenge: Can you work out how you would turn the L

We can extend this code to write a short program that flashes
by the user. Open an IDLE window, and enter the following co

```
import RPi.GPIO as GPIO              #imports GPIO module
GPIO.setmode(GPIO.BOARD)             #set GPIO in 'board'
GPIO.setwarnings(False)              #switches off warnin

from time import sleep               #import sleep functi

led1 = 5                             #assigns pin 5 to LE
GPIO.setup(led1 = GPIO.OUT)          #configures pin 5 as
delay = float(input("Enter the delay in seconds: "))  #
```

This code now needs a loop adding to it to make the LED flash

Challenge: Can you write the code for the loop to make

Save your program as LED_flash.py

Programs using the GPIO interface need to be run with root ac
LXTerminal window, and run the program with the command:

```
sudo python3 LED_flash.py
```

When run, we should have a flashing LED.

D) Using the GPIO to receive an input from a button

We can use the GPIO interface to receive inputs from componen buttons. To receive an input from a button, we have to tell the R 'listen' for a signal. The button is connected to the Pi via an elec press the button, the circuit is complete and current flows; when current flow stops. When the pin receives current, an input is reg

First we need to build the inputs and outputs.

We will need the following components:

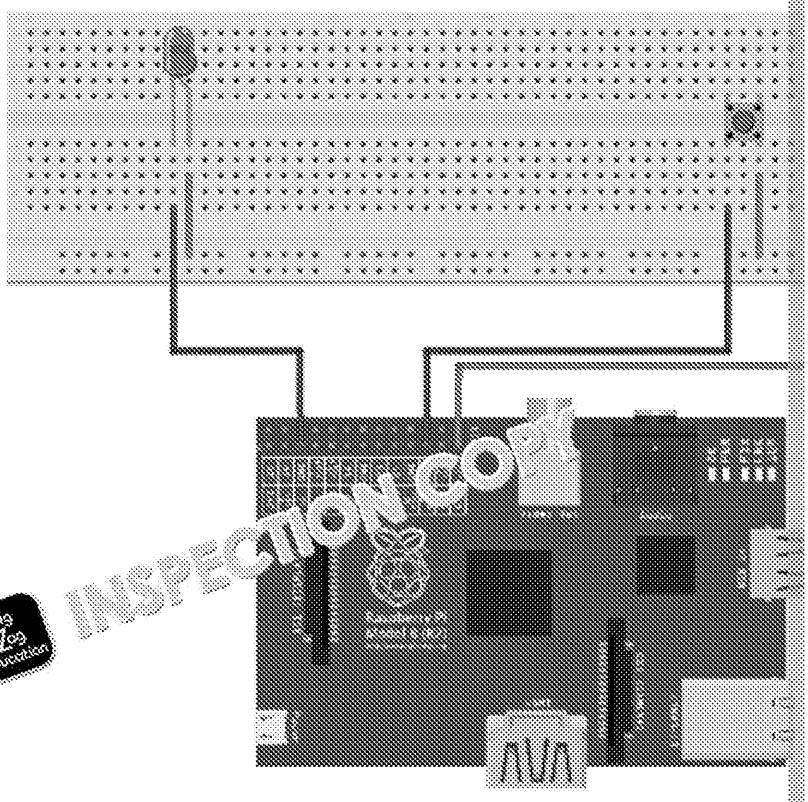
- Raspberry Pi kit
- Full-size breadboard
- 1 × LED
- 1 × button
- 3 × female-to-male jumper leads
- 3 × male-to-male jumper leads
- 1 × 470 ohm resistor

Connect the components as follows:

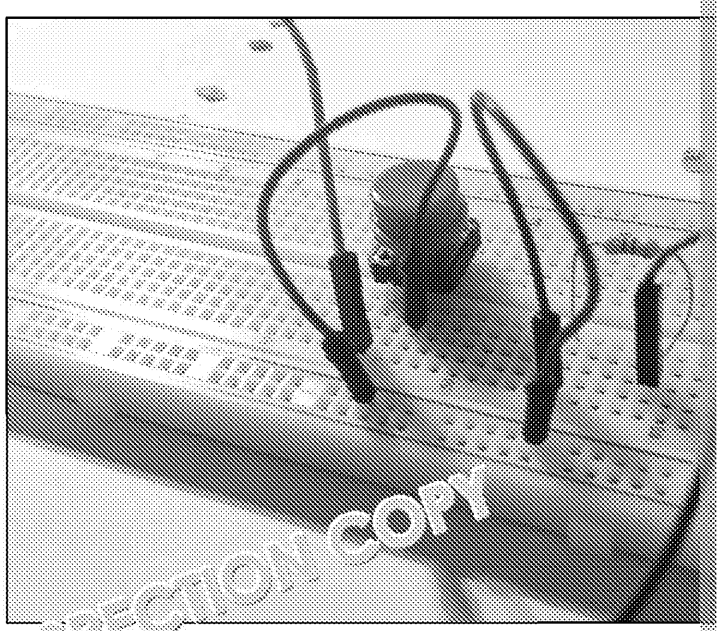
1. Place the resistor on the breadboard, one leg in **slot e55**
2. Using one male-to-female jumper lead, connect the fem **25** (ground), and the male end to **slot c59**. This connect
3. Using a male-to-male jumper lead, place one end in **slo** the **ground (blue) track at slot 55**. We now have a resis connection.

Now, place the remaining components and leads as follows:

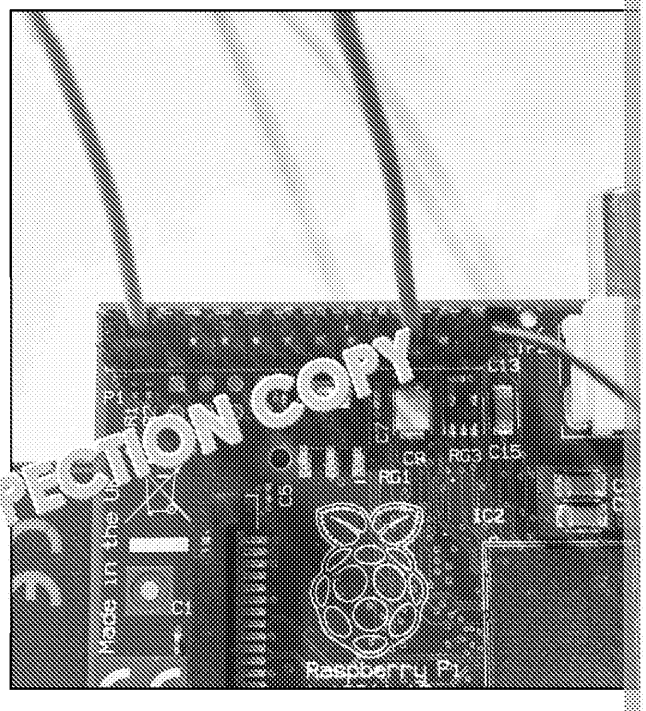
LED	led1
LED slots	a10, e10
Ground lead slots	c11, ground track 11
+ve lead slots / GPIO pin	a10, 5
BUTTON	button1
BUTTON slots	e45, e47
Ground lead slots	c47, ground track 47
+ve lead slots / GPIO pin	a45, 21



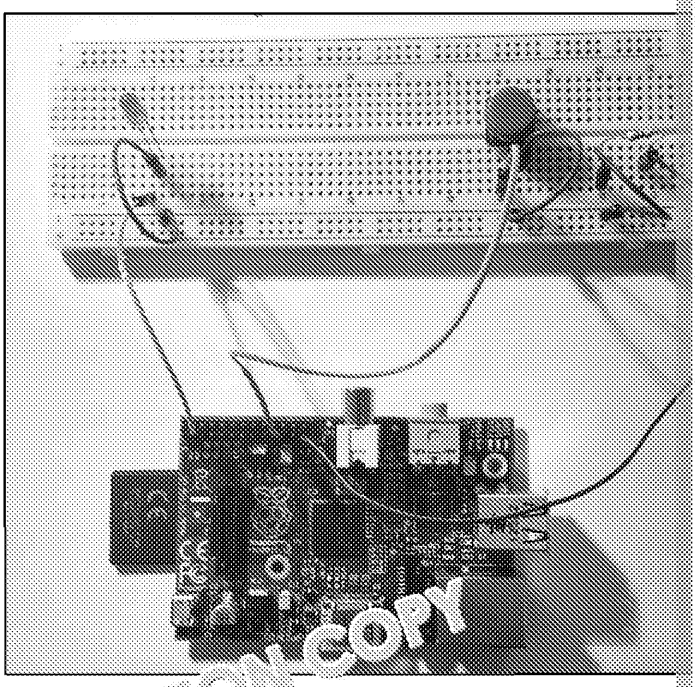
The button should look like this:



The pins should look like this:



The whole circuit for an LED and button should look like this:



Next we have to import and configure a GPIO pin as an input:

```
import RPi.GPIO as GPIO                 #imports GPIO module
GPIO.setmode(GPIO.BOARD)                #set GPIO in 'board'
GPIO.setwarnings(False)                 #switches off warnin

button1 = 21                            #assigns pin 21 to b
GPIO.setup(button1 = GPIO.IN)           #configures pin 21 a
```

The code to tell the Pi to listen for a signal is:

```
while True:                                    #repea
      while GPIO.input(button1):               #liste
            pass
      if GPIO.input(button1) == False:         # whe
                                               #do w
                                               print
```

We can control the LED with the button. When the button is pr
when released the LED is unlit. Open a new IDLE window and e
code:

```
import RPi.GPIO as GPIO                        #imports GPIO module
GPIO.setmode(GPIO.BOARD)                       #set GPIO in 'board'
GPIO.setwarnings(False)                        #switches off warnin
led1 = 5                                       #assigns pin 5 to LE
button1 = 21                                   #assigns pin 21 to b
GPIO.setup(led1 = GPIO.OUT)                    #configures pin 5 as
GPIO.setup(button1 = GPIO.IN)                  #configures pin 21 a
from time import sleep                         #import sleep functi
while True:                                    #repeat continuously
     while GPIO.input(button1):                #listen
            pass
     if GPIO.input(button1) == False:          # whe
               GPIO.output(led1, 1)            #turn
               sleep(1)                        #paus
               GPIO.output(led1, 0)            #turn
```

Save the program with the filename LED_button.py

Open an **LXTerminal** window, and run the program with the co

```
sudo python3 LED_button.py
```

When the button is pressed, the LED will light for 1 second.

Stretch Yourself

- Add more LEDs and buttons, perhaps even different coloured ones
- Work out how to have different patterns of LEDs lit depending on

Binary Conversio

DURATION: LONG
DIFFICULTY: HARD

For this project we are going to convert denary (base 10) numb
and learn how to get the Raspberry Pi to light LEDs to output t

Project Aims

- To understand how to convert from denary to binary
- To program a denary-to-binary converter in Python
- To learn about the Raspberry Pi's GPIO interface
- To learn how to use the GPIO interface as an output to light LEDs
- To use LEDs as a method of outputting binary numbers

Project Outcom

- It will represe binary
- It will convert integers from
- It will use LED binary numbe

Project Equipme

- Basic Raspbe
- Full-size brea
- 8 LEDs
- 9 × female-to
- 9 × male-to-
- 1 × 470 ohm

Project Tasks

A) Convert denary to binary .....

B) Write a program to convert denary t... in... .....

Extension: Understanding the GPIO interface .....

Project Tasks

A) Convert denary to binary

At its most basic level, a computer is a collection of circuits. Thes
either have current flowing through them (on) or not (off). Data
circuits on or off.

Binary notation uses two digits, 1 and 0. We can use a 1 to repre
on, and a 0 to represent a circuit that is off. Thus, we can represe
1s and 0s. Each individual digit is known as a 'bit'. 8-bit binary u
use an LED to represent each bit. The LEDs will either be lit (1) or

The denary system uses place values that increment in powers o

Place value	,000,000	1,000,000	100,000	10,000	1

Num are represented by putting digits under the place value
number 123 is represented as:

Place value	10,000,000	1,000,000	100,000	10,000	1
Digit					

This gives us (1 × 100) + (2 × 10) + (3 × 1) = 123.

In denary the biggest number we can represent using 8 digits is

Place value	10,000,000	1,000,000	100,000	10,000	1
Digit	9	9	9	9	

(9 × 10,000,000) + (9 × 1,000,000) + (9 × 100,000) + (9 × 10,0
100) + (9 × 10) + (9 × 1) = 99,999,999.

Binary place values increment in powers of 2:

Place value	128	64	32	16	8	

Remember, binary also uses only 2 digits, 1 and 0. Binary numb
placing a 1 or 0 under each place value. We place a 1 if we nee
0 if we do not. In binary, the denary number 123 would look lik

Place value	128	64	32	16	8	
Number	0	1	1	1	1	

(1 × 64) + (1 × 32) + (1 × 16) + (1 × 8) + (1 × 2) + (1 × 1) = 12
0111 1011.

In binary, the biggest denary number we can represent with 8 bi

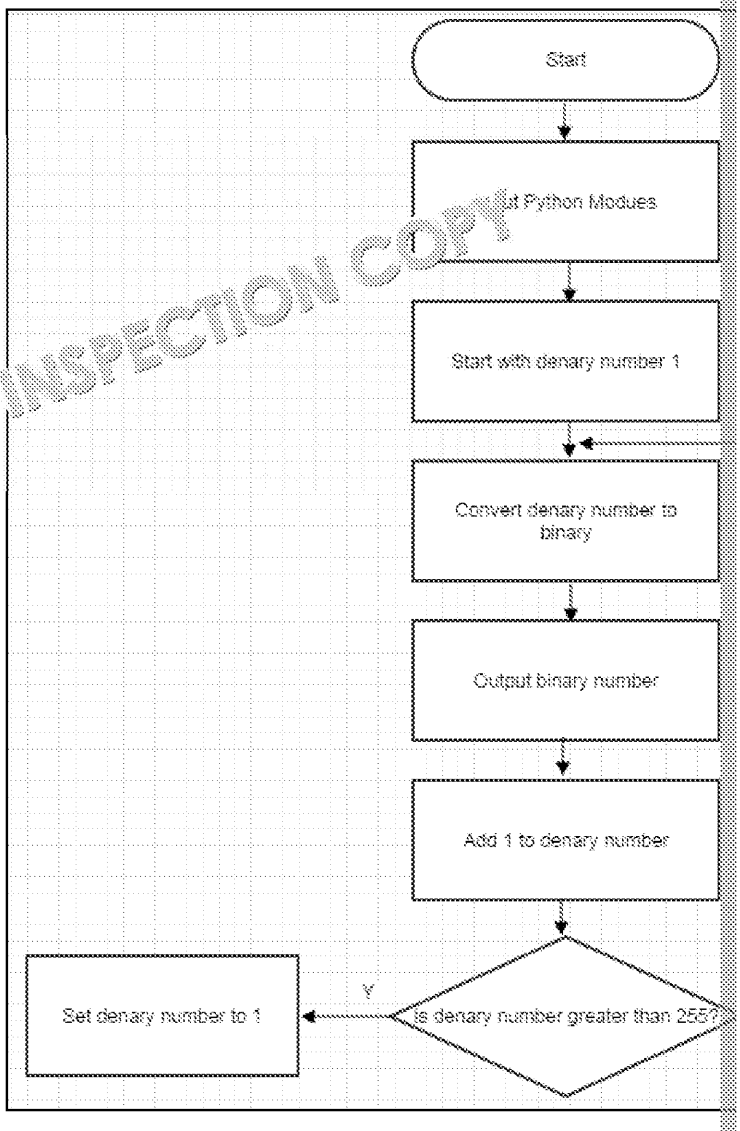
Pla e	128	64	32	16	8	
Nu	1	1	1	1	1	

(1 × 128) + (1 × 64) + (1 × 32) + (1 × 16) + (1 × 8) + (1 × 4) +
255, or in binary 1111 1111.

Challenge: Convert the following denary numbers
7 25 93 164

B) Write a program to convert denary to binary

We can write a Python program to convert denary to binary. A
this program would be:



First, open up a **new IDLE3 window**.

To aid us we will need to use a Pyth[illegible]dule called 'numpy'.
built-in binary conversion f[illegible]that will simplify what we h
use the 'time' mod[illegible]o implement a slight delay. Add code t
and 'tim[illegible]

```
#bin[illegible]onversion

#import modules
from numpy import binary_repr
from time import sleep
```

Now, add code to set the denary number, and convert the den

```
denary = 1                              #variable, holds den

print(denary)                           #ouput denary number
binary = binary_repr(denary, 8)         #convert denary numb
print(binary)                           #output binary
sleep(0.5)                              #pause for 0.5 secon
```

Next we need to add the code that increments the denary num
number exceeds 255, the count needs to be set to 1. This is bec
would require more than 8 bits to display it.

Challenge: Can you write the code to increment the den
when 255 is reached?

Lastly, we need to include a loop so that the program runs con

```
#binary conversion

#import modules
from numpy import binary_repr
from time import sleep

denary = 1                              #variable, holds

while True:
      print(denary)                     #ouput denary num
      binary = binary_repr(denary, 8)   #convert denary nu
      print(binary)                     #output binary
      sleep(0.5)                        #pause for 0.5 se

      denary += 1                       #increment denary
      if denary > 255:                  #if denary number
            de                          #reset it to 1
```

Save the program giving it the name 'denary_to_binary.py'.

Now run the program, and binary numbers will appear on the

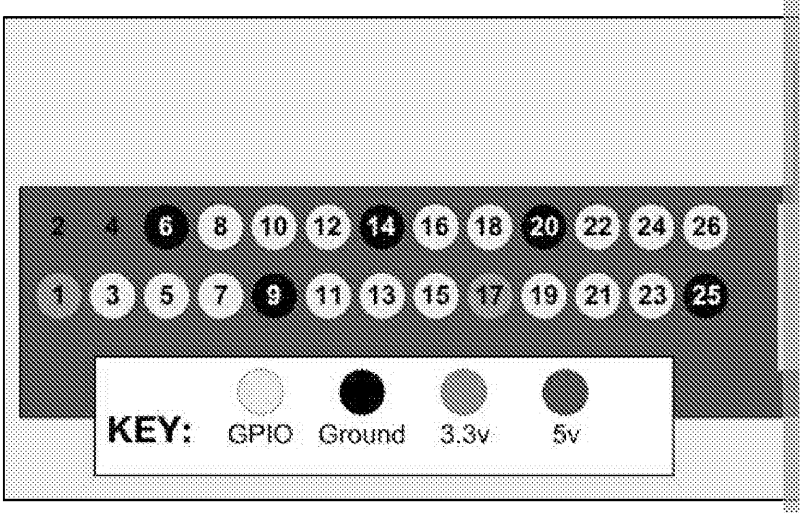
Extension: Understanding the GPIO interface

One of the many great things about the Raspberry Pi is the built (General Purpose Input Output) interface. This interface allows t with other devices and components. The GPIO interface is the se the edge of the board, along from the Video-Out port.

Each pin carries a signal in or out of the Pi. We can configure 17 inputs or outputs. The remaining pins are power or ground pins carefully. Power can be delivered at 3.3 V or 5 V, depending on t

NOTE: Be extremely careful wh u ir the GPIO pins – conn wrong pins can damac

The pi ea a name/number, and there are two methods – 'BC d 'BOARD'. Board numbering assigns a number from and this is the method we will use.



Remember: Pins 1, 2, 4, 6, 9, 14, 17, 20 and 25 are power/grou be used for inputs/outputs.

Connecting a single LED

Connecting a single LED is straightforward. First we will set up short Python program to turn the LE on and off.

We will need the follo mponents:

- ea board
- 70 ohm resistor – this protects the circuit
- 2 × female-to-male jumper leads – connects the compo
- 2 × male-to-male jumper leads – completes the circuits
- 1 × LED

Connect the components as follows:

Place the resistor on the breadboard, one leg in **slot e55**, the c

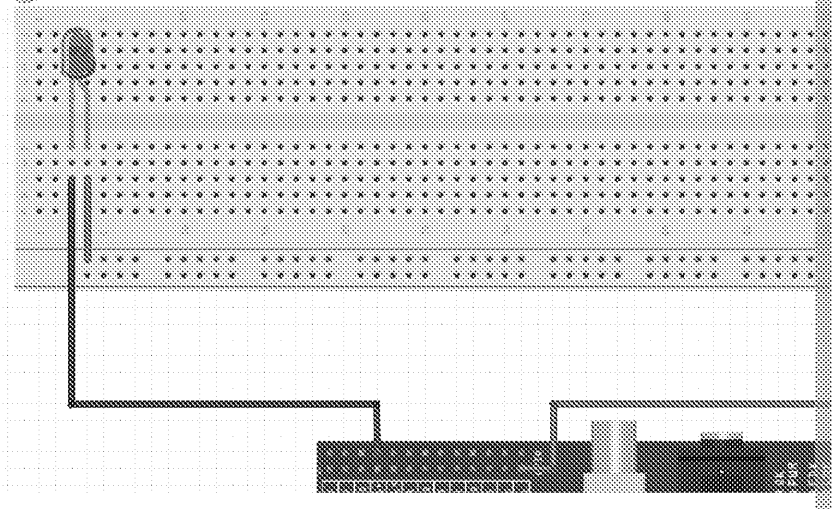
Using one male-to-female jumper lead, connect the female en
(ground), and the male end to **slot c55**. This connects the resis

Using a male-to-male jumper lead, place one end in **slot c55**, t
ground (blue) track at slot 60. We now have a resistor-protect

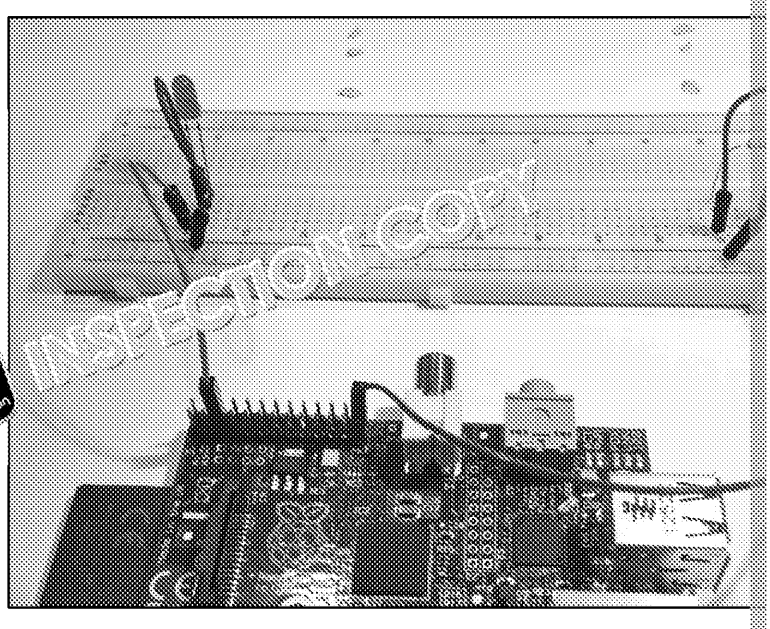
Place the LED on the breadboard, putting the **longer leg (posi
shorter leg (ground) in **slot e4**.

Connect one male-to-male jumper lead to **slot c4** and to the g

Complete the circuit using one female-to-male jumper lead, th
and the male end in **GPIO pin 8**.



The LED circuit should look like this:



We have now completed our initial hardware circuit. Next we n
switch the LED on and off.

Write a program to light the LED

We can use Python to write a simple program to light the LED. P
built-in modules that make programming much easier. To use m
import them into our program. To switch an LED on and off, we

- *RPi.GPIO* ***(note the lowercase 'i')***
- *time*

Open up a new IDLE3 window, and enter the following Python

```
#import modules
import RPi.GPIO as GPIO                    #required to a
from time import sleep                     #required to i
```

Be c to make sure you use upper-case and lower-case let
above.

Next, we have to tell the Raspberry Pi that we will be referring
numbers (known as BOARD mode):

```
GPIO.setmode(GPIO.BOARD)                   #set GPIO in
GPIO.setwarnings(False)                    #switches off
```

Now we have to tell the Raspberry Pi that we want to use a pin
how we are going to use it (input or output):

```
led = 8                                    #assign pin to


GPIO.setup(led, GPIO.OUT)                  #configure GPI
```

All we need to do now is tell the Pi to switch the LED on and o

```
While True:
      GPIO.output(led, 1)                  #LED on
      time.sleep(1)                        #pause
      GPIO.output(led, 0)                  #LED off
      time.sleep(1)                        #pause
```

The full program looks like this:

```
#binary conversion
#import modules
import RPi.GPIO as GPIO                    #required to a
from time import sleep                     #required to i

GPIO.setmode(GPIO.BOARD)                   #set GPIO in '
GPIO.setwarnings(False)                    #switches off

#assign pin to LED
led = 8

#co   e   o pin as output
GPIO  p(led, GPIO.OUT)

While True:
      GPIO.output(led, 1)                  #LED on
      time.sleep(1)                        #pause
      GPIO.output(led, 0)                  #LED off
      time.sleep(1)                        #pause
```

Now, save the program, giving it the name 'binary_LED.py'.

To run the program we need access to the command line. Ope
window and enter:

```
sudo python3 binary.py
```

The LED should flash on and off. If it doesn't, check the following:

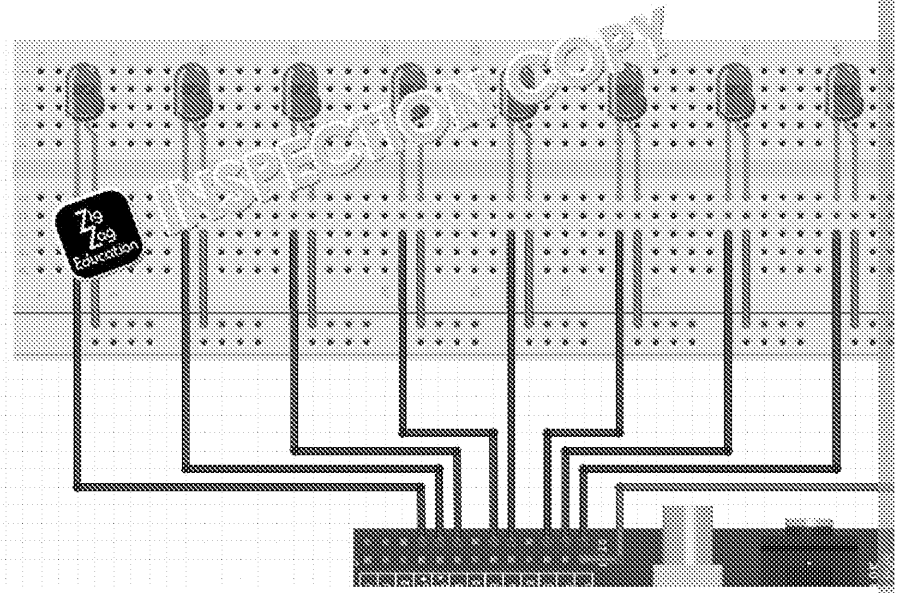
- That all leads are properly connected
- That connections are in the correct slots
- That the correct pins have be se
- That the LED is the correct way around
- at ython program is exactly as above

Bringing it all together

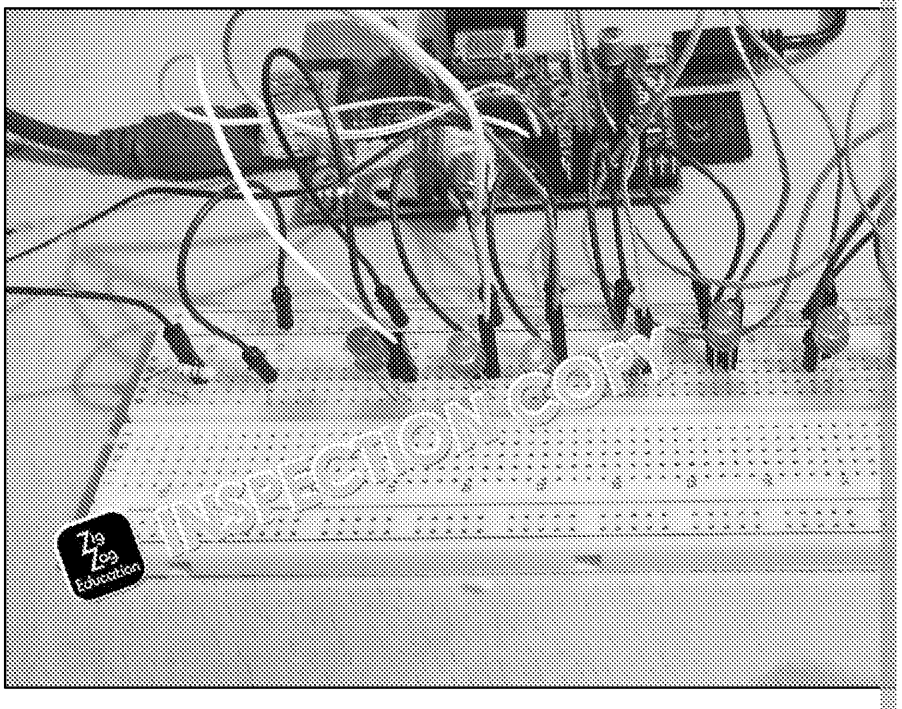
The next task is to extend our Python program to include the c
convert to binary and to use LEDs to output the binary numbe

First, set up the rest of the LEDs on the breadboard, using the fo
Remember, we've already set up the first LED (led), **but now ren**

LED	LED slots	Ground lead slo
led128	e3, e4	c4, ground track
led64	e9, e10	c10, ground track 10
led32	e15, e16	c16, ground track 16
led16	e22, e23	c23, ground track 23
led8	e28, e29	c29, ground track 29
led4	e34, e35	c35, ground track 35
led2	e39, e40	c40, ground track 40
led1 e45, e46	c46, ground track 46	c45, 26



You should find it looks like this:



Now we need to set up the rest of the GPIO pins:

```
#import modules
import RPi.GPIO as GPIO                    #required to a
from time import sleep                     #required to i
GPIO.setmode(GPIO.BOARD)                   #set GPIO in '
GPIO.setwarnings(False)                    #switches off

#assign pin to LED
led128 = 8
led64 = 10
led32 = 12
led16 = 16
led8 = 18
led4 = 22
led2 = 24
led1 = 26

#configure GPIO pi as output
GPIO.setup(led128, GPIO.OUT)
GPIO.setup(led64, GPIO.OUT)
GPIO.setup(led32, GPIO.OUT)
GPIO.setup(led16, GPIO.OUT)
GPIO.setup(led8, GPIO.OUT)
GPIO.setup(led4, GPIO.OUT)
GPIO.setup(led2, GPIO.OUT)
GPIO.setup(led1, GPIO.OUT)
```

The next step is to integrate the conversion code we wrote pre
the code, making sure the program matches what's printed be
import the '**numpy**' module:

```
#binary conversion

#import modules
import RPi.GPIO as GPIO                    #required to a
from numpy import binary_repr
from time import sleep                     #required to i
```

Now we have to write the code to o[illegible] nary with our LEDs
this, we need to convert o[illegible] number into a list containir
digits. From this list we can examine each individual digit – if it
LED, if [illegible] it off.

This [illegible] code we use:

```
binary = str(binary)
binary_list = []  for x in range(0,8):
      binary_list.append(binary[x])
```

Challen
what th

Now we have to examine each digit in turn and see whether it
digit in the list corresponds to place value 128, the second to p
on. The last digit in the list corresponds to place value 1. Becau
place value the digit corresponds to, we can tell the Raspberry
Remember, computers start at 0, so the first digit in our list is e

```
if binary_list[0] == "1":              #if the first elemen
    GPIO.output(led128, 1)             #switch on LED128
if binary_list[1] == "1":              #if the second eleme
    GPIO.output(led64, 1)              #switch on LED64
if binary_list[2] == "1":              #and so on…
    GPIO.output(led32, 1)
if binary_list[3] == "1":
    GPIO.output(led16, 1)
if binary_list[4] == "1":
    GPIO.output(led8, 1)
if binary_list[5] == "1":
    GPIO.output(led4, 1)
if binary_list[6] == "1":
    GPIO.output(led2, 1)
if binary_list[7] == "1":
    GPIO.output(led1, 1)
```

We also need to switch off all the LEDs before we increment to

```
GPIO.output(led128, 0)              #switches off all LE
GPIO.output(led64, 0)
GPIO.output(led32, 0)
GPIO.output(led16, 0)
GPIO.output(led8, 0)
GPIO.output(led4, 0)
GPIO.output(led2, 0)
GPIO.output(led1, 0)

#binary conversion

#im[illegible]no [illegible]
impo[illegible]i.GPIO as GPIO               #required to acces
from numpy import binary_repr
from time import sleep                      #required to imple
GPIO.setmode(GPIO.BOARD)                    #set GPIO in 'boar
GPIO.setwarnings(False)                     #switches off warn

#assign pin to LED
led128 = 8
led64 = 10
led32 = 12
led16 = 16
led8 = 18
led4 = 22
led2 = 24
led1 = 26

#configure GPIO pin as output
GPIO.setup(led128, GPIO.OUT)
GPIO.setup(led64, GPIO.OUT)
GPIO.setup(led32, GPIO.OUT)
GPIO.setup(led16, GPIO.OUT)
GPIO.setup(led8, GPIO.OUT)
GPIO.setup(led4, GPIO.OUT)
GPIO.setup(led2, GPIO.OUT)
GPIO.setup(led1, GPIO.OUT)
```

```
denary = 1                              #variable, holds de
while True:
    GPIO.output(led128, 0)              #switches off all
    GPIO.output(led64, 0)
    GPIO.output(led32, 0)
    GPIO.output(led16, 0)
    GPIO.output(led8, 0)
    GPIO.output(led4, 0)
    GPIO.output(led2, 0)
    GPIO.output(led1, 0)
    print(denary)                       #ouput denary numbe
    binary = binary_spi(denary, 8)      #convert denary num
    print(binary)                       #output binary

    binary = str(binary)                #converts the bina
                                        string
    binary_list = []                    #creates an empty
    for x in range(0,8):                #take each digit i
        binary_list.append(binary[x])       #and add it t

    if binary_list[0] == "1":           #if the first eleme
        GPIO.output(led128, 1)          #switch on LED128
    if binary_list[1] == "1":           #if the second ele
        GPIO.output(led64, 1)           #switch on LED64
    if binary_list[2] == "1":           #and so on…
        GPIO.output(led32, 1)
    if binary_list[3] == "1":
        GPIO.output(led16, 1)
    if binary_list[4] == "1":
        GPIO.output(led8, 1)
    if binary_list[5] == "1":
        GPIO.output(led4, 1)
    if binary_list[6] == "1":
        GPIO.output(led2, 1)
    if binary_list[7] == "1":
        GPIO.output(led1, 1)

    sleep(0.5)                          #pause for 0.5 sec

    denary += 1                         #increment denary
    if denary > 255:                    #if denary number
        denary = 1                      #reset it to 1
```

```
    if binary_list[6] == "1":
        GPIO.output(led2, 1)
    if binary_list[7] == "1":
        GPIO.output(led1, 1)

    sleep(0.5)                          #pause for 0.5 sec

    denary += 1                         #increment denary
    if denary > 255:                    #if denary number
        denary = 1                      #set it to 1
```

Finally, save the program using the name 'binary_LED.py'. C
window and run the program using:

```
sud  on3 binary_LED.py
```

Stretch Yourself

- Find out how negative numbers are represented in binary.
- Add to your program a means of outputting negative binary num

Up, Down, Left, R

DURATION: MEDIUM
DIFFICULTY: MEDIUM

In this project we are going to create a game. The game will b
computer is asking us in terms of the arrow keys up, down, left
computer says up, we need to hit the up arrow, etc.

Project Aims

- To learn to create a game in which you follow the computer says
- a keys
- Learn to use an LCD screen to output the computer's instructions

Project Outcom

- To output the down, left and
- To be able to to input a use
- To recognise response mat instruction
- To keep score matches
- To output the instructions o

Project Equipment

- Basic Raspberry Pi kit
- Breadboard
- Jumper leads
- LCD screen

Project Tasks

A) Planning the program .....

B) Connecting the LCD screen .....

C) Import the necessary libraries and initialise LCD screen .....

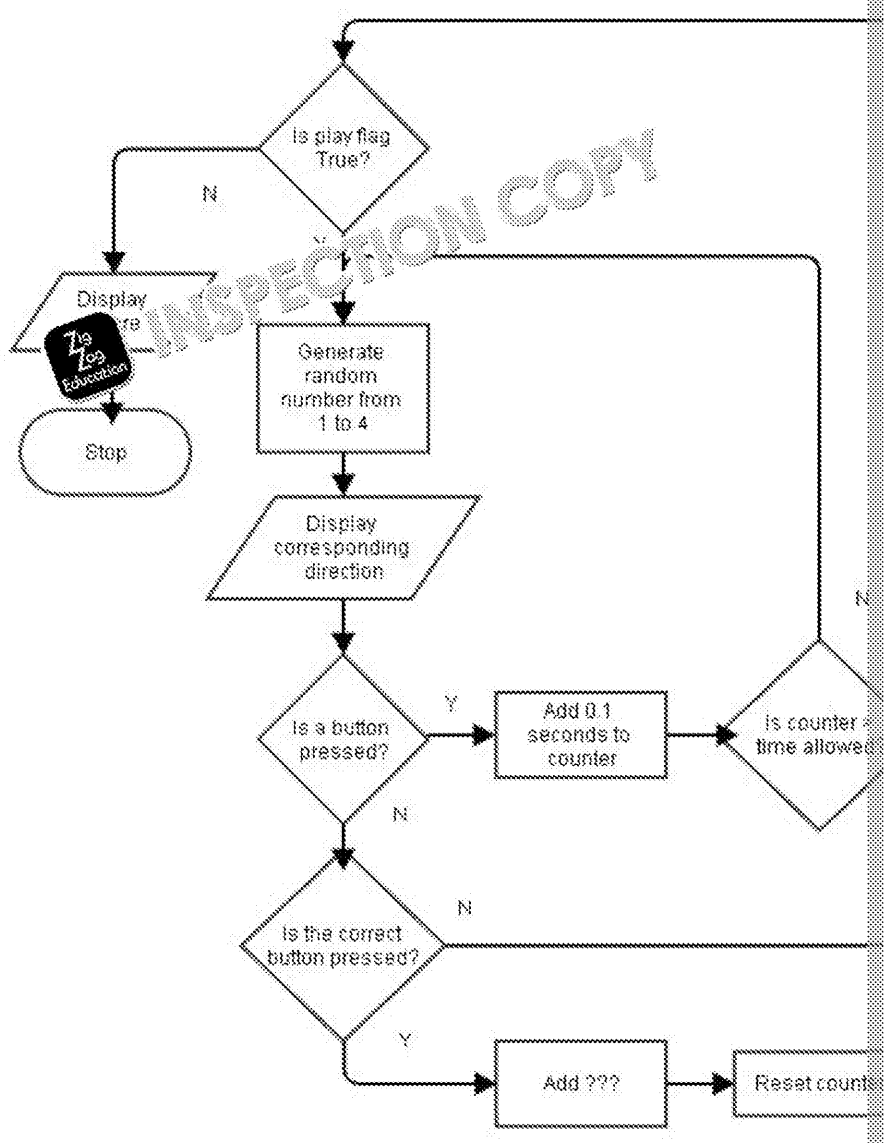
D) Define the variables .....

E) Create the program .....

Project Tasks

A) Planning the program

Below is a simple flow chart for the game.

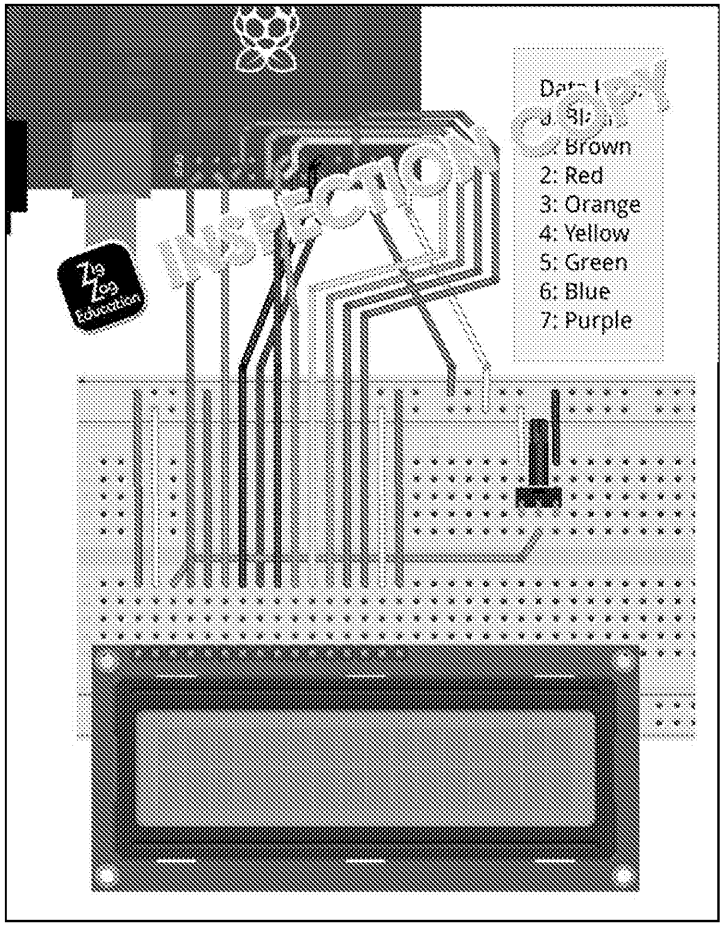


Challenge: Look at the flow chart and try to understa
showing ... there are three errors in the flow chart; can

B) Connecting the LCD screen

We need to connect the LCD screen to the Raspberry Pi. We d
breadboard and jumper leads (see the binary converter project
information on connecting add-ons to the Raspberry Pi).

The setup needs to be wired as follows:



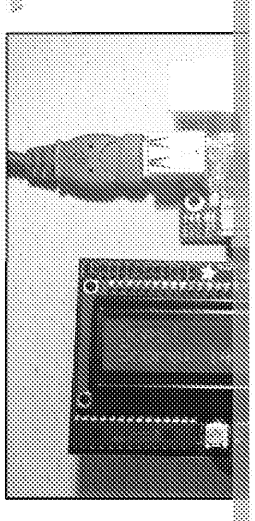
The LCD display that you can use can be a 5 V or a 3.3 V displa

If a 5 V display is used, we need to make sure that the display c
to the Raspberry Pi.

If it does, it will present 5 V h[illegible] the
GPIO pins and you [illegible] to destroy the
Raspberry Pi [illegible]sult.

When using the 5V LCD display you must connect the R/W pin on the display to ground.

You can also use a ribbon cable to connect the LCD screen to the Raspberry Pi (as shown on the right).



C) Import the necessary libraries and initialise LCD screen

There are three libraries that we will need for this project. Thes
and 'random', and from each of these libraries we need two sp
and 'randint'. These will be used to create a delay between ea
computer will give and to create random instructions. Input th

```
from Adafruit_CharLCDPlate import Adafruit_CharLCDPlate
from time import sleep
from random import randint
```

We now need to initialise the LCD screen using the following:

```
lcd = Adafruit_CharLCDPlate()        #identifier for LCD
lcd.begin(16,2)                      #initialise LCD
```

D) Define the variables

We then need to define four variables to store the data we will
program:

```
delay=2                  #how long the player has to p
counter = 0              #runs down the clock
score = 0                #keeps track of how many corr
play = True              #game continues while True
```

E) Create the program

Input the following code. There are some gaps that have been l

Challenge: Can you work out what each missing piece c

```
from Adafruit_CharLCDPlate import Adafruit_CharLCDPlate
from time import sleep
from random import randint
lcd = Adafruit_CharLCDPlate()                    #identif
lcd.begin(16,2)                                  #initial

#declare and initialise variables
delay=2                                          #how lon
                                                 press a
_____ = 0                                        #runs do
_____ = 0                                        #keeps t
                                                 correct
play = True                                      #game co
```

```
lcd.clear()                                  #clear t
lcd.message('Press the\n')                   #display
                                             instruct
lcd.message('correct button!')
sleep(2)                                     #pause
lcd.clear()
lcd.message('Ready...\n')
sleep(1)
lcd.message('Steady...')
sleep(1)
lcd.clear()
lcd.message(
sle
while          ay:                           #keep pl
    lcd.clear()
    number =randint(1,4)                     #generat
                                             from 1 t
    if _____________ == 1:                   #if ____
        lcd.message('Left\n')                #display
    ____ number == 2:                        #if 2
        _____________('Right\n')             #display
    elif number == 3:                        #if 3
        lcd.message('Up\n')                  #display
    _____:                                   #otherwi
        lcd.message('________')              #display
    while __________ < delay:                #as long a
        if lcd.buttonPressed(lcd.LEFT):      #if left b
            if number == 1:                  #and rando
                lcd.message('Correct!')      #output 'C
                _________ +=1                #add 1 to
            else:                            #otherwise
                play = _________             #play stop
            break                            #exit loop

        if lcd.                 ressed(lcd.):    #as above
            if _________ == 2:
                lcd.message('Correct')
                score +=1
            else:
                play = False
            break
```

```
        if lcd.buttonPressed(lcd.UP):
            if number ==3:
                _____('Correct!')
                score +=1
            else:
                _____ = False
            break

        if lcd.buttonPressed(lcd.DOWN):
            if number == 4:
                lcd.messag('Co  !')
                sc  =
                play = False
            break
        counter +=0.1                          #run down t
        sleep(0.1)                             #pause to a
    if counter > delay:                        #if the pla
        play = False                           #play stops

    counter = 0                                #reset coun
    sleep(delay)                               #pause

lcd.clear()                                    #clear LCD
lcd.message('Oops!\n')                         #game over
lcd.message('Score: ' + str(score))            #output sco
```

Run the program and follow the instructions on the LCD screen buttons for the computer's instructions.

STRETCH YOURSELF

- Find out how to make the ...ter's instructions speed up as the ... This way the ins... ...ll appear more quickly and the user w... more quickly ... scoring.
- ... out a way to light up a green LED if the instruction is match... and a red if the user response is incorrect.

Countdown

DURATION: MEDIUM
DIFFICULTY: EASY

For this project we are going to create a countdown timer, usin
outputs and inputs for display and control.

Project Aims

- To program a countdown timer in Python
- To learn how to use the Python time module to implement delays
- To learn how to validate an input to accept only certain numbers
- To learn how to use the OS module to clear the screen
- To use the Raspberry Pi's hardware facilities to create a matching LED output and a button control

Project Outcom

- It will allow th number of mi count down
- The inputs wi only allow a r minutes and
- As the timer c time remainin on the screen
- The display m before the ne displayed

Project Equipment

- Basic Raspberry Pi kit

Extension:

- Buttons
- LEDs
- Jumper wires
- Breadboard

Project Tasks

A) Flow charts.....

B) Des th am .....

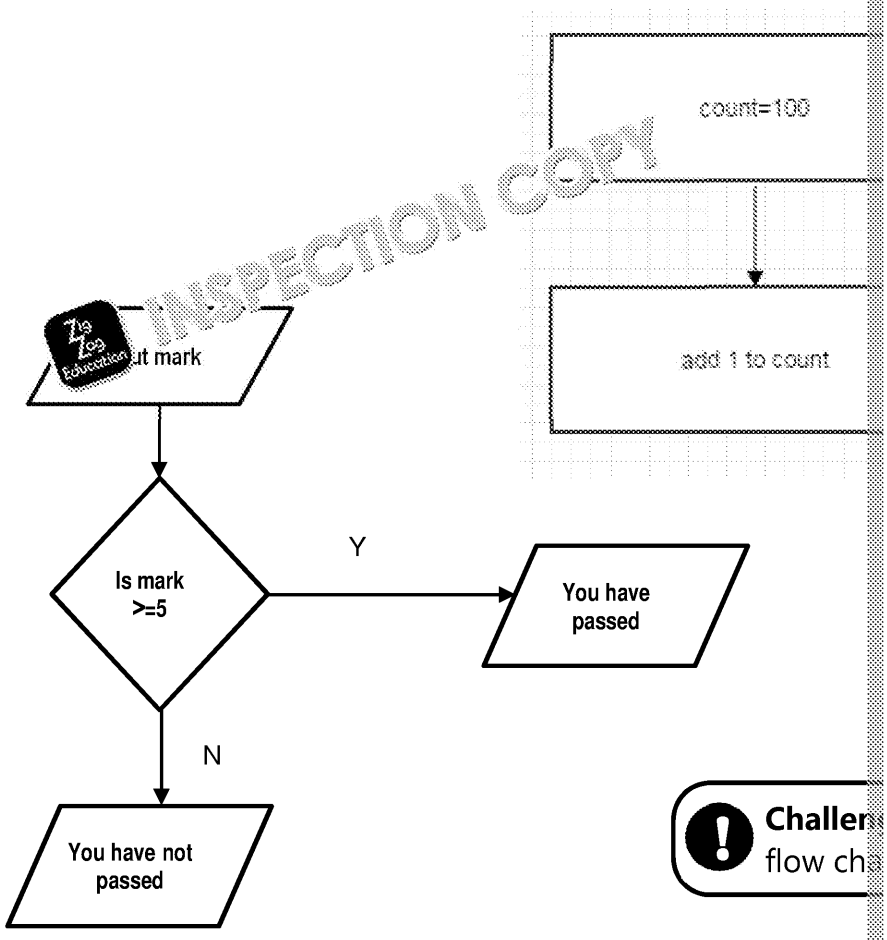
C) Cr he program .....

Extension .....

Project Tasks

A) Flow charts

Below are two flowcharts:



Before we can code our countdown timer, we will need to desi

B) Design the program

Create a flow chart using sequence, selection and iteration to r
the countdown prog m will work.

Con

- What inputs will be required?
- What outputs will be required?
- What decisions need to be made?
- What checks need to be made?

C) Create the program

You will need to create the following program in Python to cre
countdown timer. There are some gaps in the code.

Challenge: Can you guess what the missing pieces of code are?

```
#import time function from sleep module
#time will be used to implement the se   ds' delay
from time import sleep

#import os mod   s will be used to clear the displ
imp
#create a function to validate the inputs for minutes a
def get_number():
      #loop until a valid number is input
      while __________:
            try:
                  number=_____(input(""))
                  #if valid number is input, exit the l
                  if __________ >=0 and number <=59:
                        break
                  else:
                  #if a number outside the valid range
                  #display a message informing the user
                        __________("Enter a number from
            except:
                  #if a letter etc. is input
                  #display a message informing the user
                  print("Not a number. Enter a number f
      #return the input number
      __________ number

#main
#cle    e display
os.system("clear")

#get minutes and seconds for countdown
print("How many minutes? (0 to 59) ")
minutes = get_number()
print("How many seconds? (0 to 59) ")
seconds = get_number()
```

```
#check that there is still time to count down
if minutes >=0:
      #create countdown loop
      __________ True:
            #clear the display
            os.system("clear")
            #display time remaining
            display = "Time remaining: " + ___(minutes)
            print(__________)
            #pause for one second
            sleep(1)
            #reduce time remaining by one second
            seconds -=1
            #check to see if seconds remaining is now l
            #if yes, reduce minutes left by 1 and reset
            if __________ ==-1:
                  minutes -=1
                  seconds =59
            #check to see if minutes remaining is now l
            #if yes then exit the loop
            if __________ ==-1:
                  break

#display finish message
print("Time's up!")
```

Extension

The display needs formatting properly. Both minutes and seco
displayed with two digits. If less than 10 seconds remains, a '0'
front of the seconds to maintain the two digits. For example, in
minutes and 2 seconds as 10:2 the displ uld read 10:02. I
requirement for minutes.

Design three possible tests that you could use to prove that yo
meets the requirements.

Stretch Yourself

- Try to create an addition to the timer so that when the time rem
  light five LEDs, one for each second to count down. As the seconds
  the corresponding LED until none remain when the countdown ha
- Add a start/stop feature using a button.
- Add a stopwatch feature to the program, such as: once the butto
  starts and counts upwards when the button is pressed again. The
  displayed.
- Add an option so that the user may choose to run either the countd
  stopwatch.

Network Web Serv

DURATION: SHORT
DIFFICULTY: EASY

For this project we are going to turn our Raspberry Pi into a lo
server.

In this project we are going to compare the relative speed of t
using the Command Line versus the Gra h al User Interface.

Project Aims

- T n l t d what a webserver is
- arn how to install and configure a web server on a Raspberry Pi using Apache
- To learn how to access the web server's pages from another device on the network
- To understand how to add our own pages to the web server

Project Outcom

- It will host we accessed on c network
- It will use Apa software for h
- It will allow ea adding of pag

Project Equipme

- Basic Raspber
- Network conn

Project Tasks

A) Installing Apache on the Raspberry Pi .....

B) Access the Raspberry Pi web server from another device.....

C) Edit the default web page .....

D) Add a new page .....

Introduction to Web Servers

The Internet is a huge network of computers spread across the w these computers access information and some provide informati often presented via web pages. Computers that host websites ar known as web servers. When we visit a page or site on the Inter one or more web servers.

It's not just the Internet that has web servers. Many organisation and schools have web servers that can only be accessed from the area network (LAN). This type of netwo[illegible]ng with the web se intranet. Intranets help organisations share data and informatio information and data[illegible] from unauthorised users on the Inter

We [illegible] how to turn our Raspberry Pi into an intranet web serv[illegible] host web pages that can be accessed on our LAN.

Project Tasks

A) Installing Apache on the Raspberry Pi

The first step is to install Apache on our Raspberry Pi. Apache i source web server that is used by organisations over the world web pages. It is extremely simple to set up, but has many facili users can implement. However, our web server will be quite sim

To install Apache, make sure the Pi has an Internet connection, **LXTerminal** window (or from the command prompt) and type:

```
sudo apt-get install apache2 -y
```

The software will take a few moments to install.

During the installing process, Apache installs a sample web pag check that the web server is correctly activated. To view the we to know the **IP address** of our Raspberry Pi.

Every computer on a network has a unique network address. Thi the computer's IP address and is needed so that other computer whe[illegible] messages for our computer. Think of an IP addres addr[illegible] or even our phone number. Without the address or pho wouldn't know how to contact us. In fact, on a network any devi network connection has an IP address, even devices such as prin smartphones and smart TVs.

Before we can use another computer to control our Raspberry P our Pi's IP address. Finding it is easy:

Either via an **LXTerminal** window, or the command prompt, en

```
ifconfig
```

Done correctly, this will generate a message along the lines of

```
Link encap:Ethernet  HWaddr b8:27:eb:5f:f6:bb
inet addr:192.168.1.89  Bcast:192.168.1.255  Mask 255.2
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:252 errors:0 dropped:0 overruns:0 frame:0
TX packets:100 errors:0 dropped:0 ove  s:0 carrier:0
Collisions:0 txqueuelen:1000
RX bytes:37493 (36.6  bytes:9528 (9.3 KiB)
```

This i m ells us lots of useful information about the Pi
How the information we need sits in the second line:

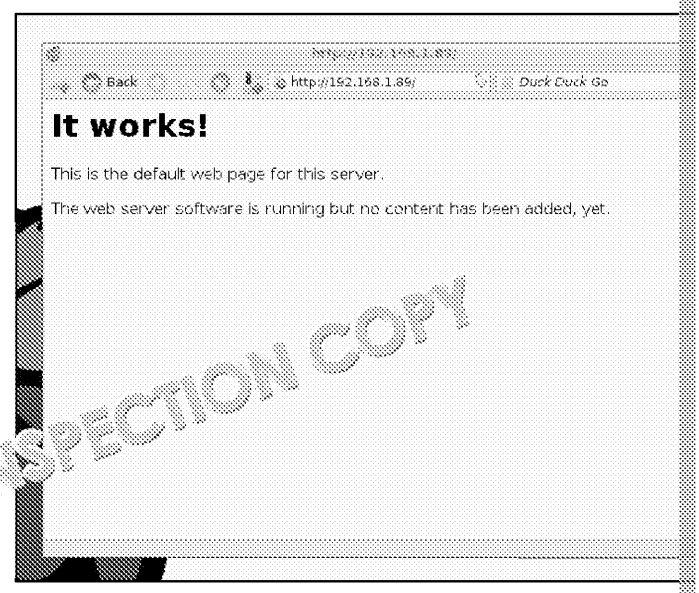
```
inet addr:192.168.1.89  Bcast:192.168.1.255  Mask 255.2
```

The IP address of our Raspberry Pi is the first collection of four
192.168.1.89 Make a note of whatever address is given here as
which we can access our web server.

Now, using Midori, enter the Pi's IP address into the address b
the beginning:

```
http://192.168.1.89
```

We should now see the default web page:



The web server is installed and configured.

**NOTE: Now that Apache is installed, it will run automatically
Raspberry Pi boots up.**

B) Access the Raspberry Pi web server from another device

Next, try accessing the same web page from another device. A
device's web browser, type the web server, as before. You shou
page:

C) Edit the default web page

The Apache web server we have installed has a specified direct
the server's web pages are hosted. We need to access this dire
remove or edit any web pages.

The web server directory is '/var/www'. To access this directory
LXTerminal Window. Next, type in:

```
cd /var/www
```

This will change the current directory to the web server's root d
this point should list contents of the directory:

```
pi@raspberrypi /var/www $ ls
index.hmtl
```

The home page of a website is usually given the name 'index.h
see, our web server's home page is n... tion.

To edit this page we will ... use the 'sudo' command to a
also have to use the nano text editor. Type:

```
sud... ...dex.html
```

The editor will display the web page's code. It is written in a lan
Hypertext Markup Language, or HTML for short.

```
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has
</body></html>
```

Notice that some of the text in the document sits with brackets

```
<html>
<body>
<h1>
<p>
```

Do not edit this text in any way! These are special keywords kn
a web browser how and where to display content on a web pa

Each tag is opened and must be closed. An opening tag uses a
a tag and a closing tag uses a '/' with in the angular brackets </
sits between the open and close tags is affected by them. For e
specifies a heading. Present the <h1> tag in the document

```
>. works!</h1>
```

Explanation:

<h1>	–	opens the tag
It works!	–	The content within the tag
</h1>	–	closes the tag

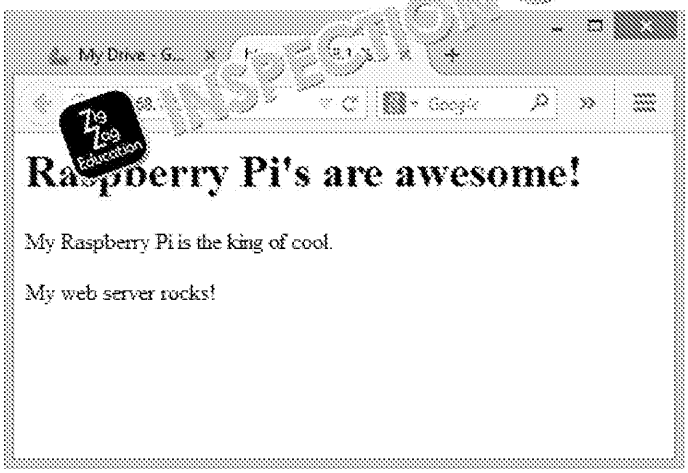
**Before we go any further, save a copy of the index.html file i
wrong.**

Now, try changing the content within the tags, but don't move
them unless you are comfortable using HTML.

Try this:

```
<html><body><h1>Raspberry Pi's are awesome!</h1>
<p>My Raspberry Pi is the king of cool.</p>
<p>My web server rocks!</p>
</body></html>
```

Save the index file using **ctrl** + **x**. Use **ctrl** + c to exit the editor
a browser on another device, open up the web server's page:



Try adding another paragraph:

```
<html><body><h1>Raspberry Pi's are awesome!h1>
<p>My Raspberry Pi is the king of cool.</p>
<p>My web server rocks!</p>
<p>ZigZag rocks too!</p>
</body></html>
```

D) Add a new page

Adding a new page is straightforward. First, using 'nano', open 'index.html' file and, using ctrl + o, save it under the file name change the text to read:

```
<html><body><h1>This is page 2!</h1>
<p>I made this page.</p>
<p>This means that I rock too!</p>
</body></html>
```

Save the file, then view it in your browser using the address:

http://192.168.1.89/page2

Now we need to add a **hyperlink** to our index page, which will
2 by clicking on a link.

To do this, open up the index using the nano editor and add the

```
<html><body><h1>Raspberry Pi's are awesome!h1>
<p>My Raspberry Pi is the king of cool.</p>
<p>My web server rocks!</p>
<p>ZigZag rocks too!</p>
<p>To visit page 2 click <a href="/page2.html">here.</a>
</body></html>
```

Save the file, and view it th[illegible] browser:

We now have a Raspberry Pi web server, two web pages, and a
from one to another.

STRETCH YOURSELF

- [illegible] code above, add a hyperlink from the second page back
- Find out how to add a hyperlink to an external website.
- Find out how to format text in colour, or in a different font.
- Insert a picture or two, or even a YouTube video.

Follow Me

DURATION: LONG
DIFFICULTY: HARD

For this project we are going to turn our Raspberry Pi into a chall
using buttons and LEDs. The LEDs will light in an increasingly leng
player must memorise the sequence and press the correct buttor
sequence to stay in the game. The sequence will also speed up as

Project Aims

- To learn ab[illegible] Raspberry Pi's GPIO interface
- To learn how to use the GPIO interface as an output to light LEDs
- To learn how to use the GPIO interface as an input with buttons
- To learn about procedures and how to use them in Python

Project Outcom

- It will use the interface
- It will use two sequence of l
- It will use a bu for the player sequence
- It will keep tra rounds the pl
- At the end of sequence will game will spe
- It will automa the player los

Project Equipment

- Basic Raspberry Pi kit
- Full-size breadboard
- 2 LEDs
- 5 × female-to-male jumper leads
- 5 × male-to-male jumper leads
- 1 × 470 ohm resistor

Project Tasks

A) Re[illegible]b[illegible]ow to use the Raspberry Pi's GPIO interface .....

B) Bu[illegible]e circuit .....

C) Learn about procedures .....

D) Learn how our game works .....

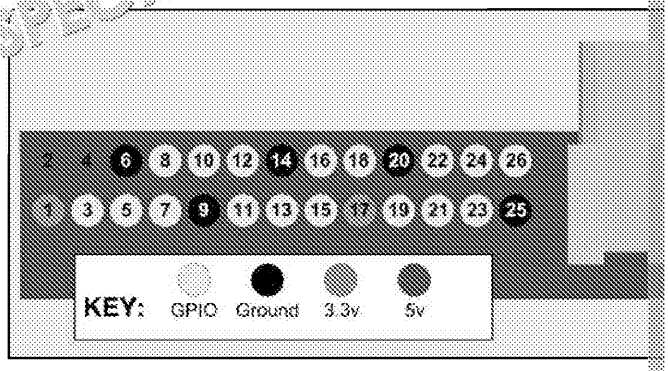
E) Write the program .....

Project Tasks

A) Remembering how to use the Raspberry Pi's GPIO interface

One of the many great things about the Raspberry Pi is the built (General Purpose Input Output) interface. This interface allows t with other devices and components. The GPIO interface is the se the edge of the board, along from the Video-Out port.

Each pin carries a signal in or out of the Pi. We can configure 17 inputs or outputs. The remaining pins are power or ground pins carefully. Power can be delivered at 3.3 V or 5 V, depending on t



NOTE: Be extremely careful when using the GPIO pins – conn wrong pins can damage the Pi.

The pins each have a name/number, and there are two methods – 'BCM' and 'BOARD'. Board numbering assigns a number from and this is the method we will use.

REMEMBER: Pins 1, 2, 4, 6, 9, 14, 17, 20 and 25 are power/grou be used for inputs/outputs.

We can use Python to control the GPIO pins. First, we have to i module, then state that we will be using the GPIO pins and con inputs or outputs. The code below configures two pins:

Pin 3 as a input (for use with a button)

Pin 19 as an output (for use with an LED)

```
import RPi.GPIO as GPIO              #imports GPIO module -
GPIO.setmode(GPIO.BOARD)             #set GPIO in 'board' m
GPIO.setwarnings(False)              #switches off warning

button1 = 3                          #assigns pin 3 to butt
led1 = 19                            #assigns pin 19 to LED
GPIO.setup(button1, GPIO.IN)         #configures pin 3 as a
GPIO.setup(led1 = GPIO.OUT)          #configures pin 19 as
```

To receive an input from a button, we have to tell the Raspberr
for a signal. The button is connected to the Pi via an electrical
the button, the circuit is complete and current flows; when we
current flow stops. When the pin receives current, an input is r
tell the Pi to listen for a signal is:

```
while True:                                  #repeat continu
    while GPIO.input(button1):               #listen
        pass
    if GPIO.input(button1) == False:         #when button is
                                             #do whatever, e
                                             pressed")
```

To operate an LED, simply tell the pin to carry a current or n

```
GPIO.output(led1, 1)                         #turns LED1 on
```

To turn an LED off:

```
GPIO.output(led1, 0)                         #turns LED1 off
```

B) Build the circuit

Before we program the game, we need to build the inputs and
game will use two LEDs to generate the sequence to be memo
to allow the user to replay the sequence.

We will need the following components:

- Basic Raspberry Pi kit
- Full-size breadboard
- 2 × LEDs
- 5 × female-to-male jumper leads
- 5 × male-to-male jumper leads
- 1 × 470 ohm resistor

Connect the components as follows:

Place the resistor on the breadboard, one leg in slot e55, the o

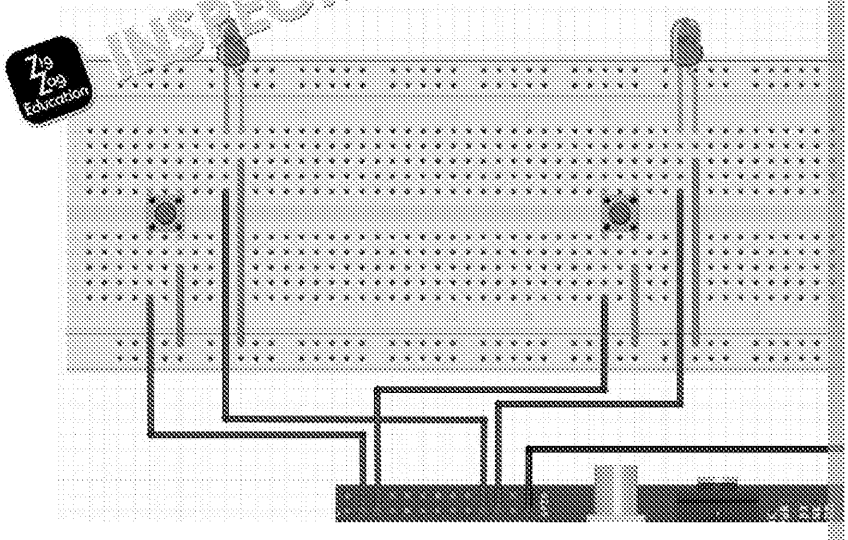
Using one male-to-female jumper lead, connect the female en
(ground), and the male end to slot c60. This connects the resist

Using a male-to-male jumper lead, place one end in slot c55, a
ground (blue) track at slot 55. We now have a resistor-protecte

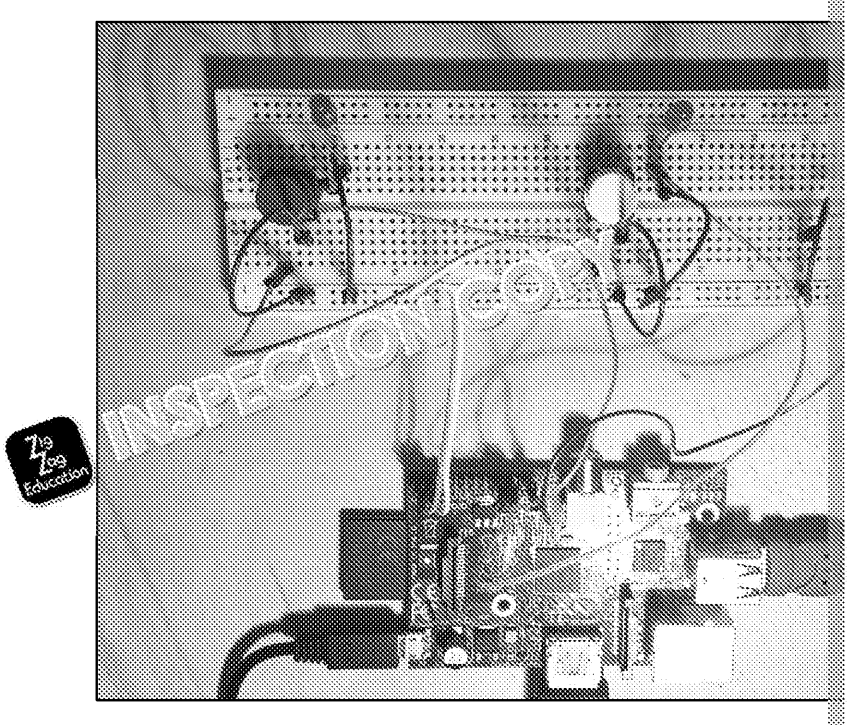
Now, place the remaining components and leads as follows:

LED	LED slots	Ground lead slots	+v
led1	j10, j11	h11, ground track 11	f10
led2	j40, j41	h41, ground track 41	f49

Button	Button slots	Ground lead slots	+v
button1	e5, e7	a7, ground track 7	a5,
button2	e35, e37	a37, ground track 37	a35



You circuit should look like this:



We have now completed our hardware building. Next we nee

C) Learn about procedures

When we program, we can find that we end up using the same
and again in different parts of our program. As a result, our co
longer than needed. Lengthy code is harder to debug when th
planned as there is more code to search through for mistakes.

One solution is to use what are known as 'procedures'. A proce
mini-program, if you like, that we run when we need it. Runnin
known as 'calling it'.

For example, suppose we had a program that lit eight LEDs. Ev
switch all the LEDs off, we would have to use code such as this

```
GPIO.output(led1, 0)                #turns LED1 off
GPIO.output(led2, 0)                #turns LED2 off
GPIO.output(led3, 0)                #turns LED3 off
GPIO.output(led4, 0)                #turns LED4 off
GPIO.output(led5, 0)                #turns LED5 off
GPIO.output(led6, 0)                #turns LED6 off
GPIO.output(led7, 0)                #turns LED7 off
GPIO.output(led8, 0)                #turns LED8 off
```

Doing it this way, we would need to use eight lines of code at
program where we needed to switch off the LEDs off. However
procedure instead:

```
def LEDs_off():
    GPIO.output(led1, 0)            #turns LED1 off
    GPIO.output(led2, 0)            #turns LED2 off
    GPIO.output(led3, 0)            #turns LED3 off
    GPIO.output(led4, 0)            #turns LED4 off
    GPIO.output(led5, 0)            #turns LED5 off
    GPIO.output(led6, 0)            #turns LED6 off
    GPIO.output(led7, 0)            #turns LED7 off
    GPIO.output(led8, 0)            #turns LED8 off
```

In the above code we have created (defined) a procedure which
LEDs. The procedure is created in the first line, using the statem
our procedure a name.

Now the procedure has been created, we can call it at any time
name:

```
LEDs_off()
```

Supposing there are four places in our program where we wou
all the LEDs. If we didn't use a procedure, we would need 32 lin
each of 4 times). By coding a procedure, and calling it each of t
only need 13 lines of code (9 to define it, one to call it each of
considerable saving in time and effort.

Additionally, if we needed to amend our code, say by adding a
have to add the line of code once, in the procedure. This also m
program much simpler, as we only have to debug the lines of c

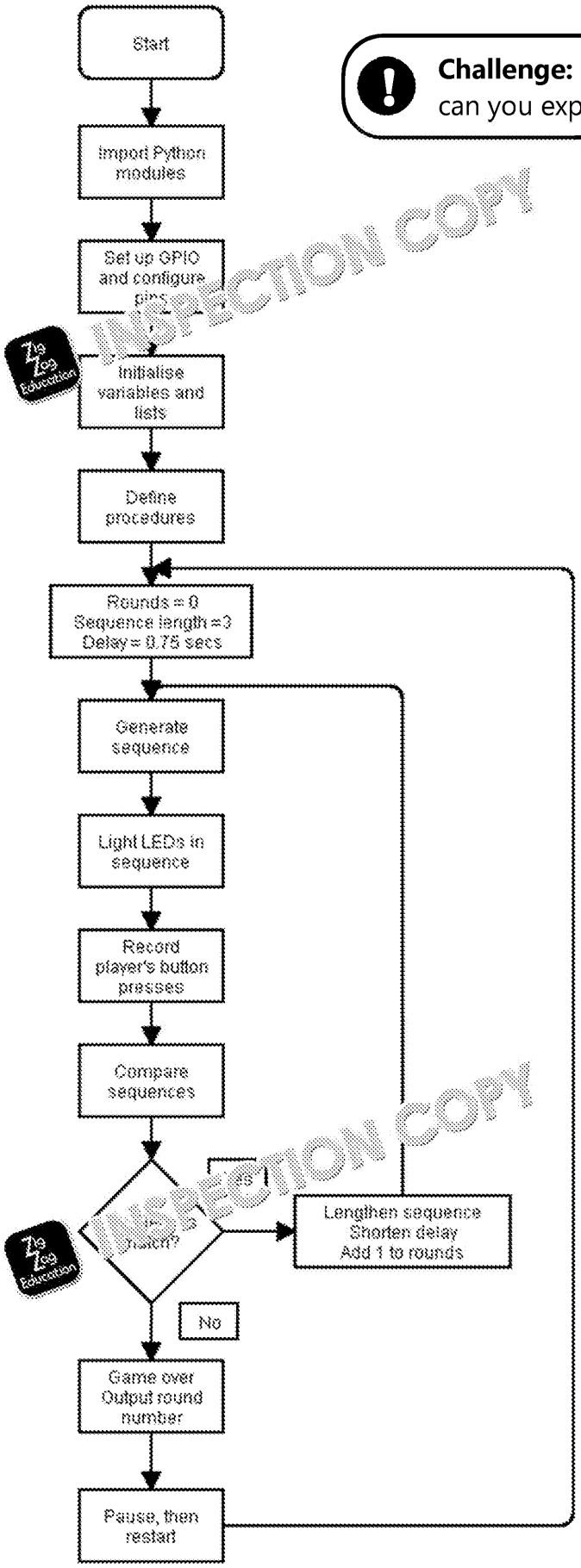
Overall, using procedures is a neater, simpler way to code a pro

Our memory game program uses several procedures to help si

- LEDs_off() – switches off all LEDs.
- make_sequence() – generates the LED sequence to be
- play_sequence() – lights the LEDs in the correct sequen
- player_turn() – records the player's inputs
- right_or_wrong() – determines whether the sequence h
  followed

D) Learn how our game works

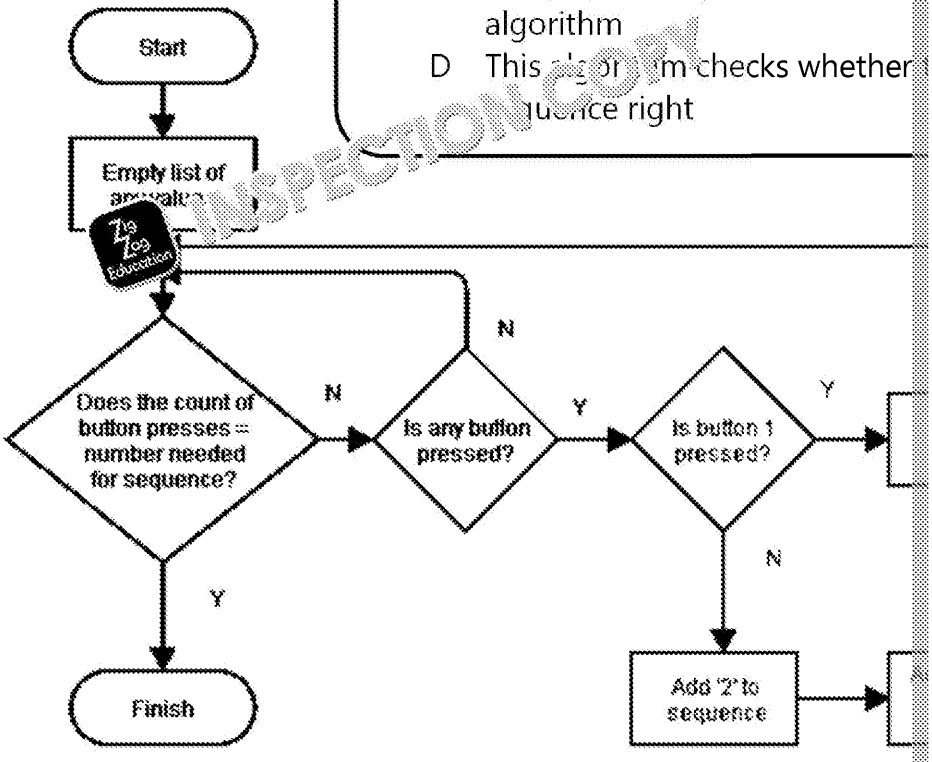
Before we code our game, we need to understand how it will w
The following flow chart describes the game:

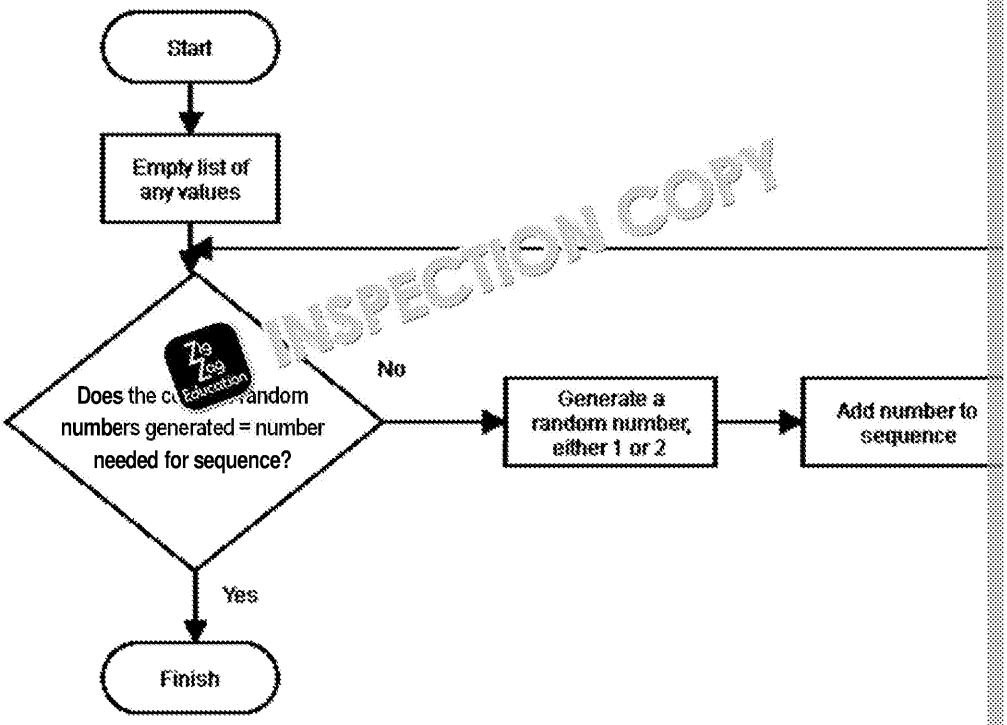


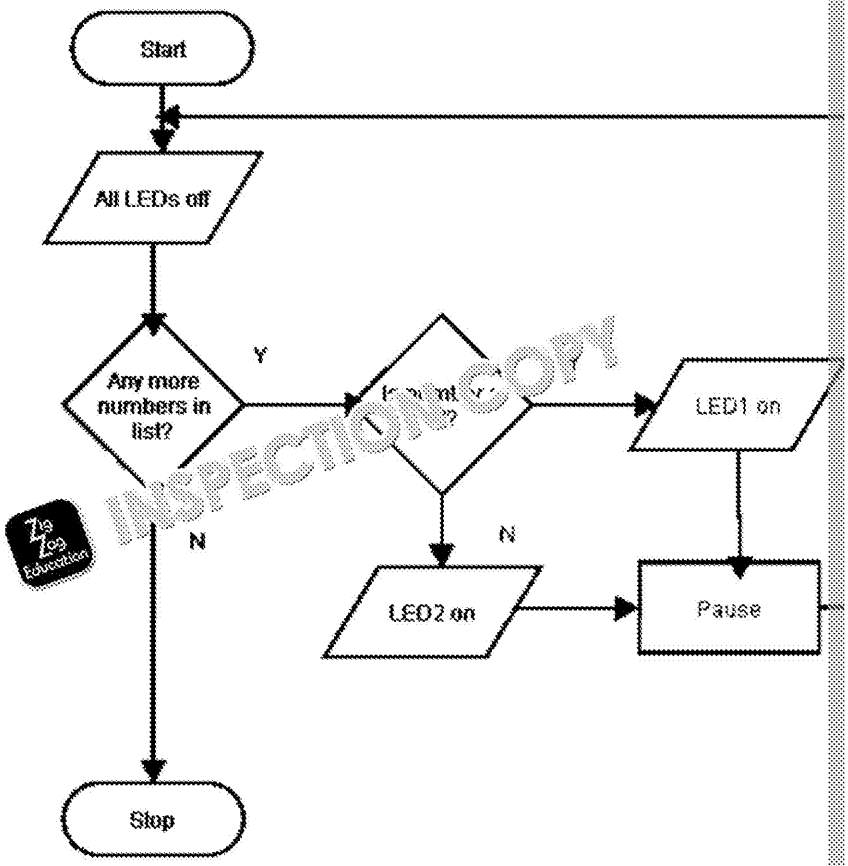
Challenge: Look at the
can you explain what it

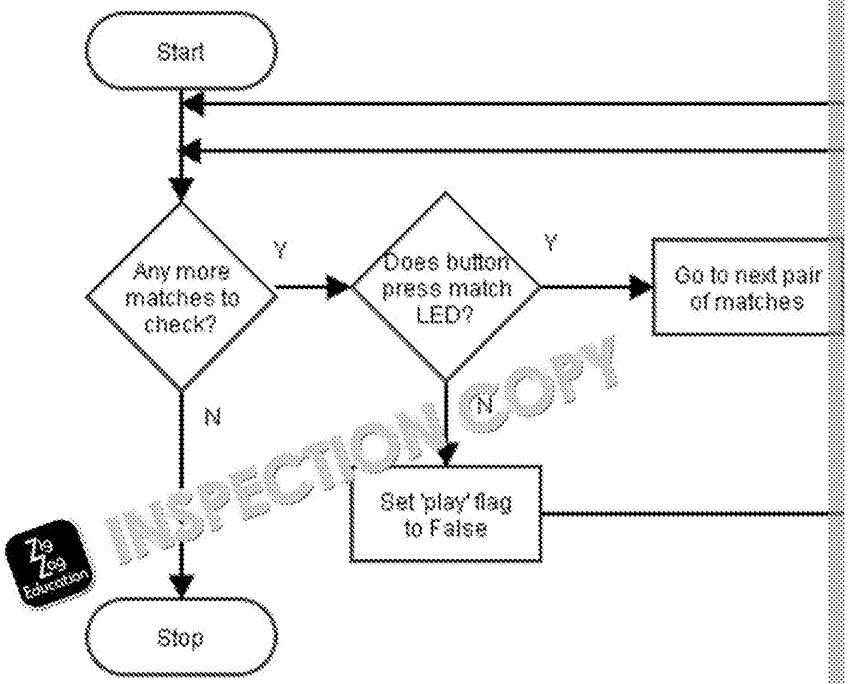
Challenge: There are four flow charts
needed to create the program.
Can you guess what each flow chart i
letter from for the correct statement
the correct flow chart:

A The sequence is generated usi
B The LED's are lit in sequence u
C The player's sequence is recor
algorithm
D This algorithm checks whether
sequence right









E) Write the program

First, open up a new IDLE3 window, then enter the following co

```
#watch the sequence of LEDs
#match the pattern

#import modules
from time import sleep                          #require
                                                delays
import random                                   #require
                                                random n
import RPi.GPIO as GPI                          #require

#se  PI
GPIO .mode(GPIO.BOARD)                          #sets GP
led1 = 19                                       #assigns
led2 = 21                                       #assigns
GPIO.setup(led1, GPIO.OUT)                      #sets pi
GPIO.setup(led2, GPIO.OUT)                      #sets pi
button1 = 3                                     #assigns
                                                button
button2 = 5                                     #assigns
                                                button
GPIO.setup(lbut, GPIO.IN)                       #sets pi
GPIO.setup(button2, GPIO.IN)                    #sets pi

#initialise variables and lists
global count                                    #variabl
                                                LEDs wil
global seq                                      #list st
                                                sequence
global player                                   #list st
global rounds                                   #variabl
                                                number
global delay                                    #variabl
                                                between

#dei  rocedures
def LEDs_off():                                 #turns a
    GPIO.output(led1, 0)                        #turns o
    GPIO.output(led2, 0)                        #turns o

def make_sequence():                            #procedu
                                                sequence
```

```
    for x in range(0,count):                    #iterate
                                                times

       number = random.randint(1,2)             #generat
                                                either 1

        seq.append(number)


def play_sequence():                            #procedu
                                                sequence

    for item in seq:                            #for eve
                                                sequence

        LEDs_off()                              #first s
        sleep(delay)                            #then pa


        if ite                                  #light L
                                                sequence
            GPIO.output(led1, 1)
        else:                                   #light L
                                                sequence

            GPIO.output(led2, 1)
        sleep(delay)                            #pause
        LEDs_off()                              #switch


def player_turn():                              #procedu
                                                player's

    counter = 0                                 #variabl
                                                moves th


    while counter != count:                     #loop
        LEDs_off()                              #switch
        while GPIO.input(lbut) and GPIO.input(button2):
            pass                                #while n
        if GPIO.input(lbut) == False:           #if left
            GPIO.output(led1, 1)                #light L
            player.append(1)                    #record
            sleep(0.5)                          #pause t
                                                reset

            counter +=1                         #increme
                                                button p
        if                nput(button2) == False:   #same bu
            GPIO.output(led2, 1)
            player.append(2)
            sleep(0.5)
            counter +=1
        LEDs_off()                              #turn al


def right_or_wrong():                           #procedu
                                                with pla
```

```
    global play                                    #flag. i
                                                   keep goi

    play = True
    for x in range(0,count):                       #for eve
        print (player[x],seq[x])                   #print p
                                                   sequence

        if player[x] != seq[x]:                    #if they
            play= False                            #flag se


#main program
count = 3                                          #sets in
                                                   LEDs lit

delay = 0.75                                       #sets in
                                                   seconds

rou  0                                             #sets ro


while True:                                        #game lo
    seq=[]                                         #empties
    player=[]                                      #empties

    LEDs_off()                                     #turn of
    make_sequence()                                #call ma
                                                   procedur

    print('watch the sequence...')
    sleep(2)
    play_sequence()
    print ('ready...')
    sleep(0.5)
    print ('steady...')
    sleep(0.5)
    print ('go!')
    player_turn()                                  #call pl
    right_or_wrong()                               #call se
                                                   procedur

    if play:                                       #if all
        print ('Yay, yo  right!')
        print ("  et's harder!")
        co  = 1                                    #increme
        ounds +=1                                  #increme
                                                   by 1

        delay = delay -0.1                         #decreas
                                                   by 0.1 s

        sleep(2)                                   #pause
    else:                                          #game ov
        print ('Oh dear, you got it wrong')
        print ('never mind')
```

```
        print ('try again!')
        print ('rounds completed: ' + str(rounds))
        count = 3                                   #reset c
        rounds = 0
        delay = 0.75
        print('Restarting shortly...')
        sleep(5)
```

Save the program with the file name '**follow_me.py**'. Open an
and run the program with the command:

```
sudo python3 follow_me.py
```


STRETCH YOURSELF

- Add more LEDs and buttons, perhaps even different coloured ones, each sequence.
- Work out how to record the number of correct button presses.
- Improve the on-screen display.
- Combine this project with the 'Camera' project to automatically take a photo of the player if he lasts for the most rounds.

Digital Camera

DURATION: MEDIUM
DIFFICULTY: HARD

One of the many great things about the Raspberry Pi is the var
can be attached to it, such as a camera. For this project we are
Raspberry Pi into a portable digital camera. We will also learn h
stamp a file.

Project Aims

- To learn how to use a Raspberry Pi
- To learn how to turn our Raspberry Pi into a portable digital camera
- To learn how to automatically give files a unique name

Project Outcom

- It will be port
- It will be oper
- It will use an L
  the camera is
  picture
- It will store ea
  unique time-
  file name

Project Equipment

- Basic Raspberry Pi kit
- Raspberry Pi Standard Camera
- Half-size breadboard
- 1 × LED
- 1 × button
- 1 × 470 ohm resistor
- 3 × female-to-mal
- Mobile phone por
- A couple of strong
- 2 × male-to-male
- Raspberry Pi case
- Blu-Tack (or simila

Project Tasks

Project Tasks

A) Taking a photo

There are several ways in which a Raspberry Pi Standard Camer
instructed to take a photo, but we will learn how to use Python t
will also allow us to use a button and LED to control the camera

NOTE: This guide assumes you have already installed the can
before we continue, following the instruct... supplied with i

Initially, our program will be ... very simple. Open up a new
will be ready to beg...

Firs... to import three Python modules, 'picamera', 'da
'pica... gives us access to the camera functions, 'datetime' a
the current time and date, and 'time' allows us implement dela
current time and date in a format suitable for a file name.

```
import picamera
from datetime import datetime
from time import sleep, strftime
```

Once a photo has been taken, we need to save it as a file. How
every photo a different and unique file name or the Pi will over
photo every time we take a new one. A great way to do this is
every photo, so that the file name is the date- and time that th
This gives each photo a unique file name.

To create a unique date- and timestamped filename, add this c

```
now = datetime.now().strftime('%d%m%Y%H%M%S')
print(now)
file_name = now + '.jpg'
```

The code instructs the Pi to check the curre... ...te and time and
file name the current date and time. '.jpg' gives the file name th

Now we need to ad... ...e code that takes the photo:

```
with ...am ...PiCamera() as camera:
came...esolution = (640,480)              #sets the reso
    camera.start_preview()                #starts the ca
    sleep(1)                              #records for 1
    camera.capture(file_name)             #saves the pho
    camera.stop_preview()
    camera.close()
```

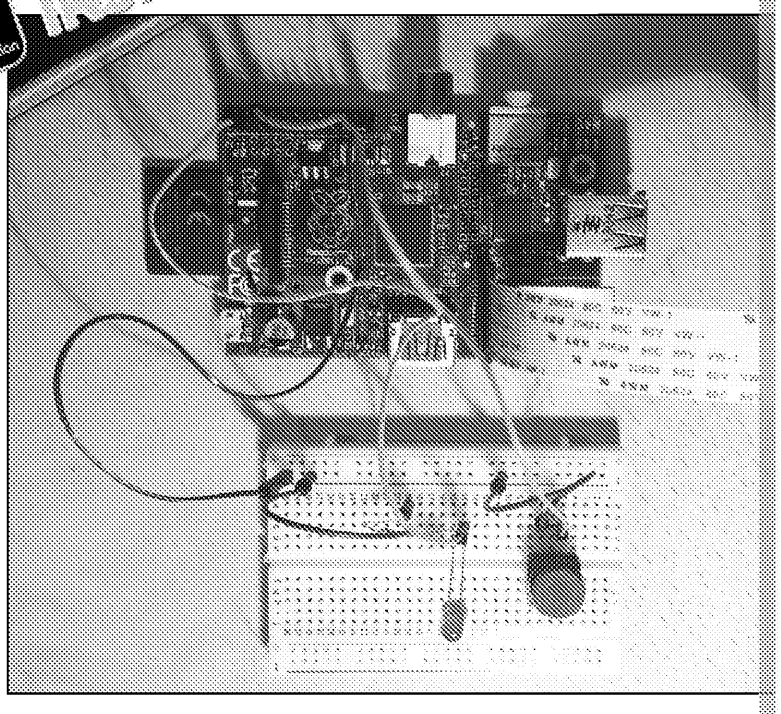
Save the file will the file name 'camera.py'. Open up an **LXTerm** the program:

```
sudo python3 camera.py
```

Now open up 'File Manager' and you should see a photo store with the current date and time as its file name.

B) Adding a button and LED

For our Raspberry Pi portable camera to be practical, we need to control the Pi without a keyboard or mouse. A simple way is to ad – the button to take a photo, and the LED to tell us when the Pi is



NOTE: For information on how the Raspberry Pi's GPIO interf 'Binary Conversion' project.

First we need to import the Raspberry Pi GPIO Python module. A code to the program. Make sure you use the correct upper-case

```
import picamera
from datetime import datetime
from time import sleep, strftime
import RPi.GPIO as GPIO
```

Next we need to set up two GPIO pins, one as an input (for the
output (for the LED).

```
led = 8                          #assign pin 8 to the L
but = 3                          #assign pin 3 to the b
GPIO.setup(led, GPIO.OUT)        #configure pin 8 as an
GPIO.setup(but, GPIO.IN)         #configure pin 3 as an
```

We will use the LED to tell us when the Pi is ready to take a pic
when the Pi is ready and unlit when it is busy:

```
GPIO.output(led, 1)              #Pi is ready to take p
```

The button is used to take a picture. Until the button is presse
patiently. Once the button is pressed, the LED is unlit, and the

```
while GPIO.input(but):
      pass

GPIO.output(led, 0)              #Pi is taking pho

with picamera.PiCamera() as camera:
camera.resolution = (640,480)    #sets the resolut
      camera.start_preview()     #starts the camer
      sleep(1)                   #records for 1 se
      camera.capture(file_name)  #saves the photo
      camera.stop_preview()
      camera.close()
```

All that's left to do is to add a loop so that the camera can take
completed code looks like this:

```
import picamera
from time import sleep, strftime
import RPi.GPIO as GPIO
from datetime import datetime

GPIO.setmode(GPIO.BOARD)
led=8
but=3
GPIO.setup(led, GPIO.OUT)
GPIO.setup(but, GPIO.IN)

while True:
    GPIO.output(led,1)
    while GPIO.input(but):
        pass
```

```
    GPIO.output(led,0)
    now = datetime.now().strftime('%d%m%Y%H%M%S')
    file_name = now + '.jpg'
    with picamera.PiCamera() as camera:
            camera.resolution = (640,480)
            camera.start_preview()
            sleep(1)
            camera.capture(file_name)
            camera.stop_preview()
            camera.close()
```

Save the program, but don't run it yet.

C) Build the camera

Now it's time to assemble the hardware. Making sure the Rasp
case, complete the following:

Place the resistor on the breadboard, one leg in slot c15, the o

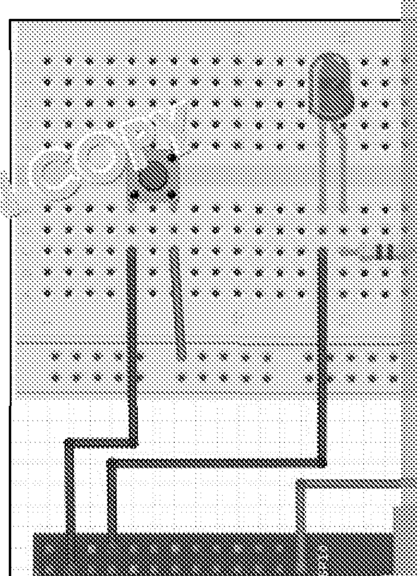
Using one male-to-female jumper lead, connect the female en
(ground), and the male end to the blue (ground) track on the b

Using a male-to-male jumper lead, place one end in slot a19, a
ground (blue) track at slot 19. We now have a resistor-protecte

Place the LED on the breadboard, putting the longer leg (posit
shorter leg (ground) in slot e15.

Complete the circuit using one female-to-male jumper lead, th
the female end in GPIO pin 8.

Connect one male-to-male jumper lead to slot c7 and to the ground track slot 7.



Place the button with a connecting leg in e5 and e7.

Complete the circuit using one female-to-male jumper lead, the male end in slot c5, the female end in GPIO pin 3.

Use a little Blu-Tack on the breadboard to hold any loose items in place. Do not use it on the GPIO pins.

We can now test the program to make sure it is working corre
LXTerminal window and type:

```
sudo python3 camera.py
```

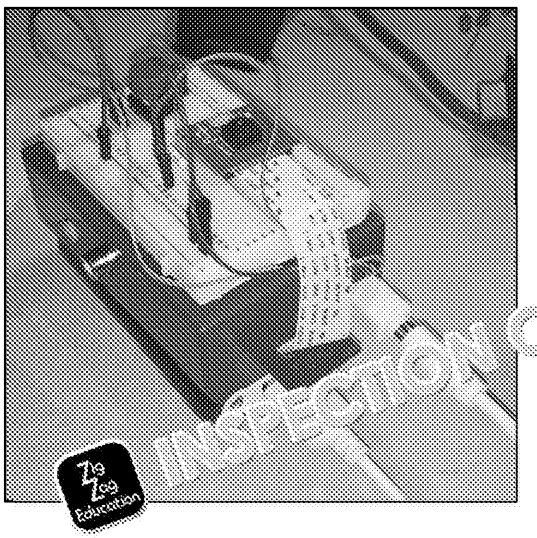
The LED should light. When the button is pressed, the Pi shoul
and take a photo. When the photo is taken, the LED should sw

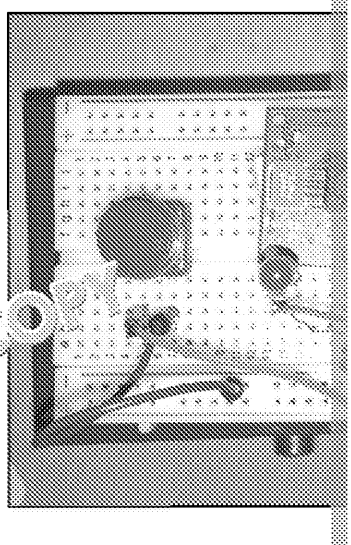
If the camera doesn't operate correctly, check the following:

- That all leads are properly connected
- That that connections are in the correct slots
- That the correct pins have been used
- That the LED is inserted the correct way round
- That the Python program is exactly as described

D) Making the camera portable

Finally, we need to make our camera portable. Restart the Rasp
time using the mobile phone external battery as a power supp
bands and/or Blu-Tack to stick the breadboard and battery to
and use a little Blu-Tack to stick the camera to a suitable positi
sure that you can easily see the LED and press the button whe
Set the 'camera.py' program running, and disconnect the moni
mouse. You now have a working, running, portable Raspberry





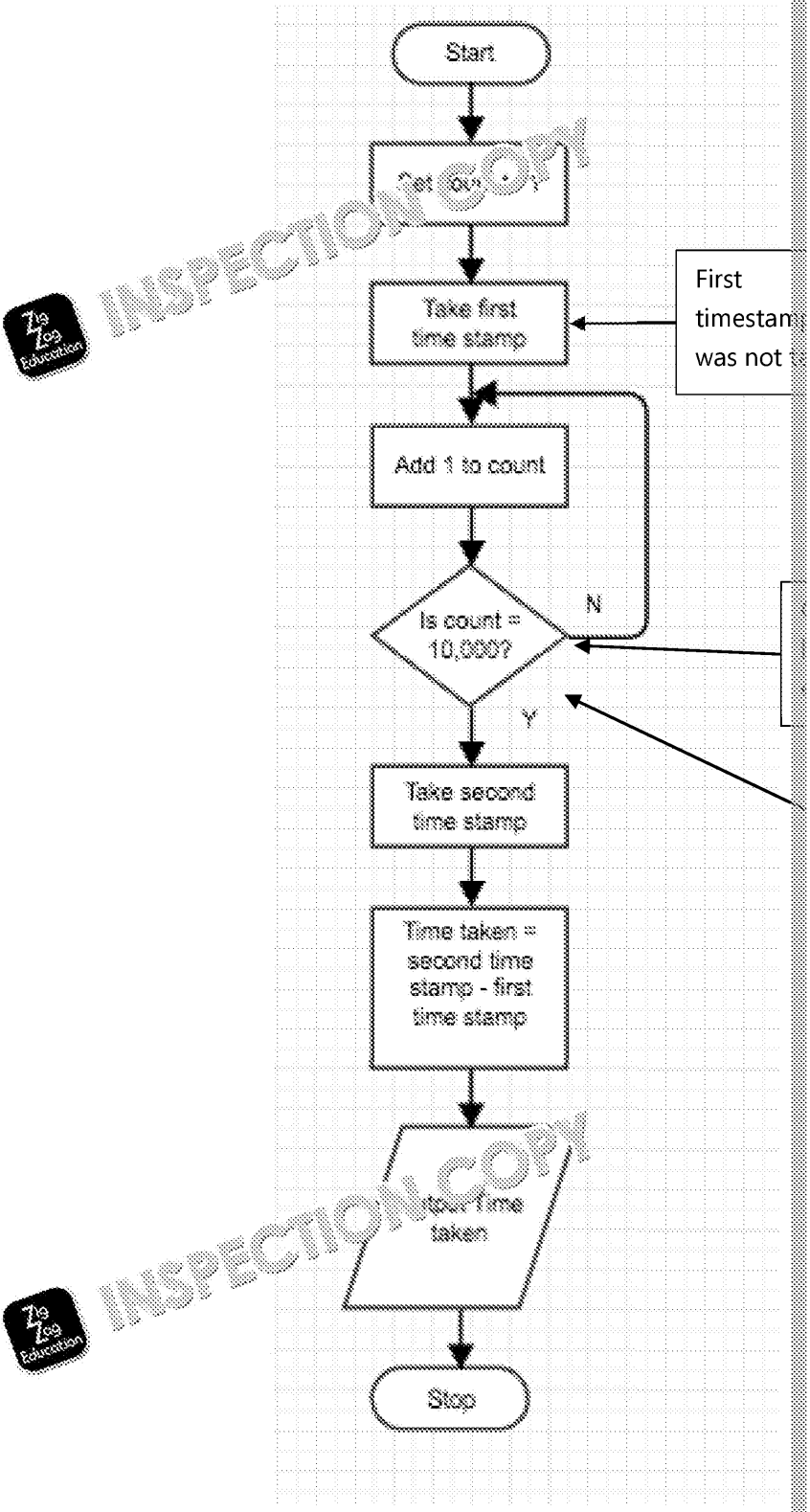
STRETCH YOURSELF

- Find out how to change the resolution of the image. HINT – at pre… record photos at 640 × 480 pixels.
- Change the program so that it records video instead of taking a …
- Find out how to reverse an image, and h… to change elements s…
- Using a wireless adap… to use this project with the 'Remoting… make a time-l… security camera whose images can be viewed … Note… …t need LEDs or buttons for this.

Benchmarking the Raspberry Pi

Task B Challenge

This is the correct flow chart.



Task D Challenge

This is the calculation for the time taken and how to output it. This needs ad

```
time_taken = time_stamp_2 - time_stamp_1        #calculates the
print ('Time taken: ' + str(time_taken))        #outputs the d
```

Interactive Quiz

Task B Challenge

The full code is shown below (with missing parts underlined)

```
print("Welcome to my quiz")
score = 0
print ("Question 1")
print ("A. option 1\n"
       "B. option 2\n"
       "C. option 3\n")
answer1 = input("answer: ")
if answer1 == "A":
        print("Correct!")
        score = score + 1
else:
        print("Incorrect!")
```

Input/Output Basics

Task C Challenge (1)

This is the code to turn off an LED.

```
GPIO.output(led1, 0)        #turns LED1 off
```

Task C Challenge (2)

This is the loop to make the LED flash.

```
while True:   #create loop
     GPIO.output(led1, 1)          #turn on LED1
     sleep(delay)    #pause
     GPIO.output(led1, 0)          #turn off LED1
     sleep(delay)    #pause
```

Binary Conversion

Task A Challenge

- 7 = 0000 0111 binary
- 25 = 0001 1001 binary
- 93 = 0101 1101 binary
- 164 = 1010 0100 binary

Task B Challenge

This is the code that increments the denary number and resets it back to 0

```
denary += 1                          #increment denary number by 1
if denary > 255:                     #if denary number is greater t
     denary = 1                      #reset it to 1
```

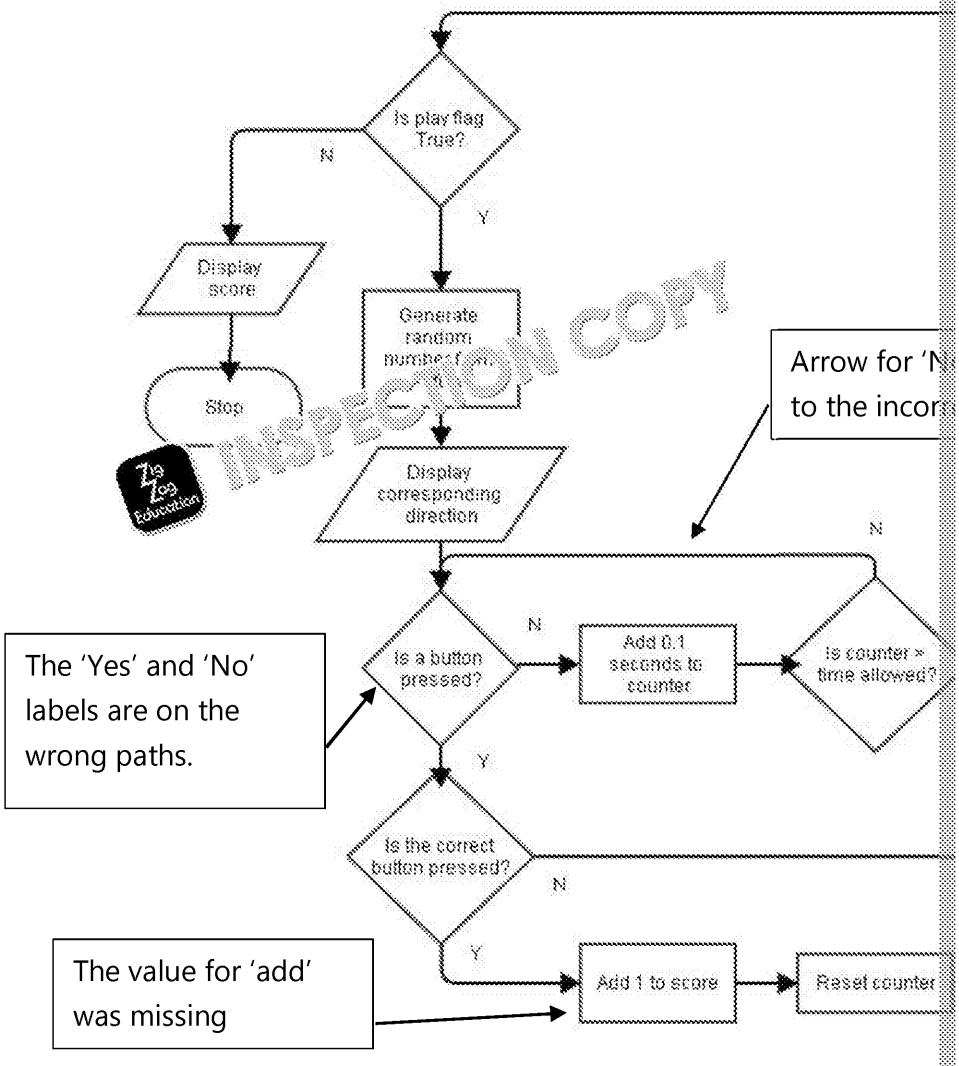
Extension Challenge

This code creates an empty list. Next it converts the binary number into a s
examine each individual digit. It then takes each individual digit in the bina
an individual element in the list.

Up, Down, Left, Right

Task A Challenge

This is the correct flow chart.



Task E Challenge

The full code is shown below (with missing parts underlined)

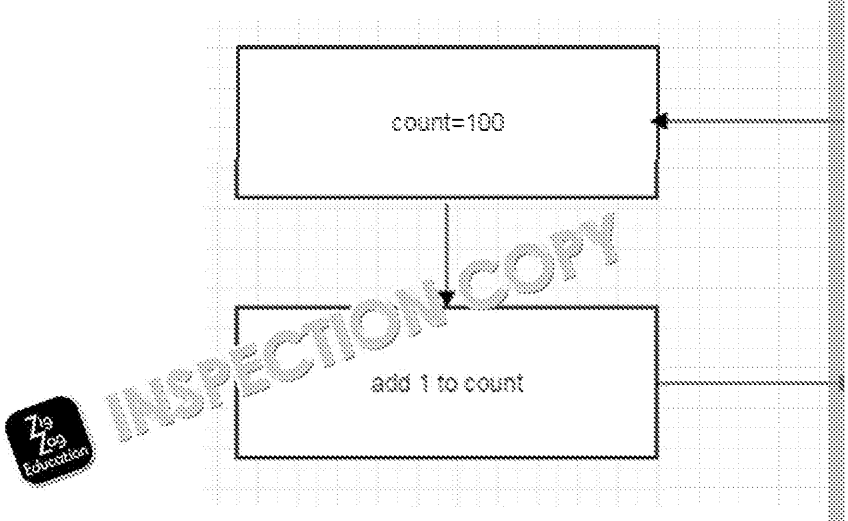
```
from Adafruit_CharLCDPlate import Adafruit_CharLCDPlate
from time import sleep
from random import randint
lcd = Adafruit_CharLCDPlate()                    #identif
lcd.begin(16,1)                                  #initial

#declare and initialise variables
delay=2                                          #how long
                                                 press a b
cour          0                                  #runs dow
scor                                             #keeps tr
                                                 correct b
play = True                                      #game con

while play:                                      #keep pla
     lcd.clear()
     number =randint(1,4)                        #generate
                                                 to 4
     if number == 1:                             #if randc
          lcd.message('Left\n')                  #display
     elif number == 2:     #if 2
          lcd.message('Right\n')                 #display
     elif number == 3:     #if 3
```

```
            lcd.message('Up\n')                    #display
      else:                                        #otherwis
            lcd.message('Down\n')                  #display

      while counter < delay:                       #as long
            if lcd.buttonPressed(lcd.LEFT):        #if left
                  if number == 1:                  #and rand
                        lcd.message('Correct!')    #output
                        score +=1                  #add 1 to
                  else:                            #otherwis
                        play = False               #play sto
                  break                            #exit loo

if lcd.buttonPressed(lcd.RIGHT):                   #as above

                  if number == 2:
                        lcd.message('Correct!')
                        score +=1
                  else:
                        play = False
            if lcd.buttonPressed(lcd.UP):
                  if number ==3:
                        lcd.message('Correct!')
                        score +=1
                  else:
                        play = False
                  break

            if lcd.buttonPressed(lcd.DOWN):
                  if number == 4:
                        lcd.message('Correct!')
                        score +=1
                  else:
                        play = False
                  break
            counter +=0.1                          #run down
            sleep(0.1)                             #pause to
      if counter > delay:                          #if the p
            play = False                           #play sto

      counter = 0                                  #reset co
      sleep(delay)                                 #pause

lcd.clear()                                        #clear LC
lcd.message('Oops!\n')                             #game ove
lcd.message('Score: ' + str(score))                #output s
```

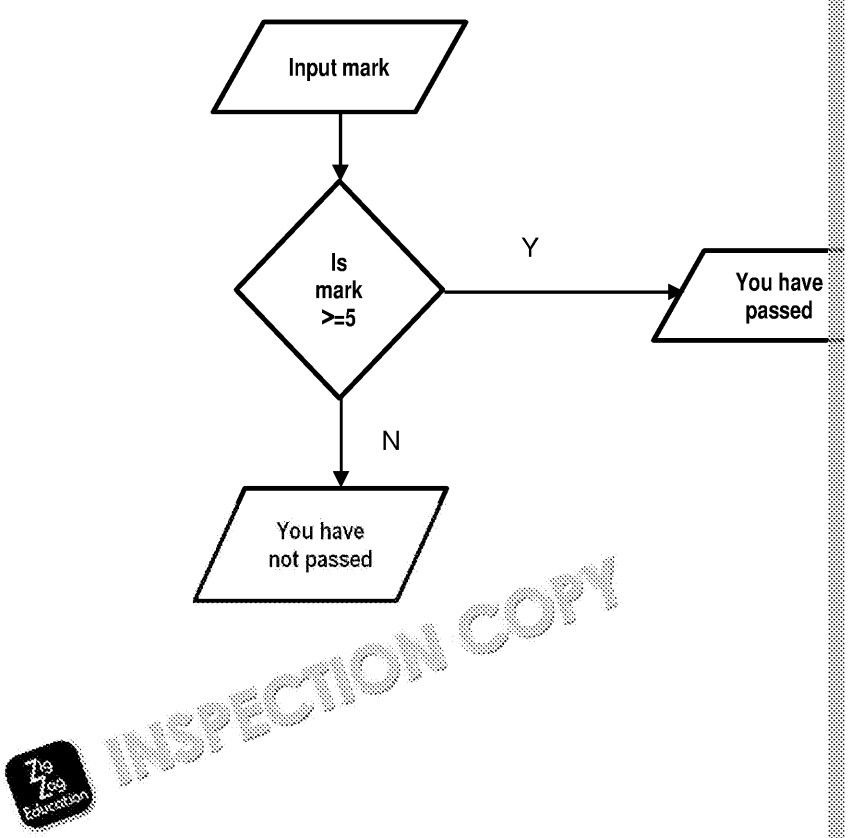

Countdown

Task A Challenge

This is showing a variable called **count** that is currently set to 100. 1 is then variable. This is all in a loop.

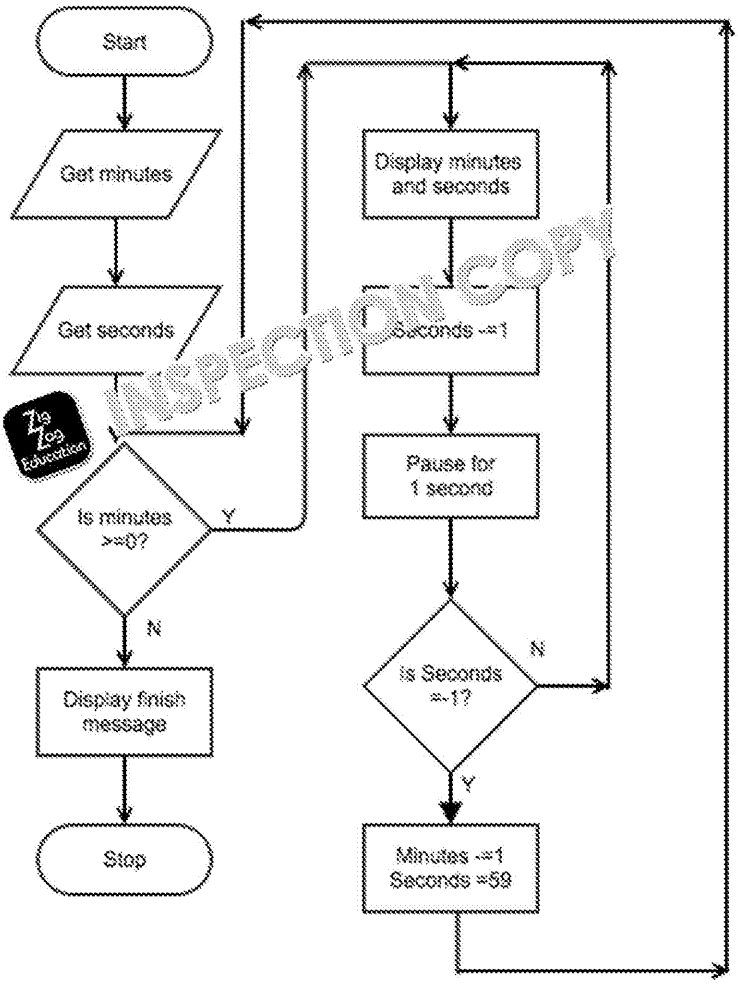


This shows a selection/decision. A mark is input, then the selection looks to greater than or equal to 5. If it is, the message 'You have passed' is displayed not passed' is displayed.



Task B Challenge
Simple Flow chart for countdown timer.

Note: does not include validation for inputs.



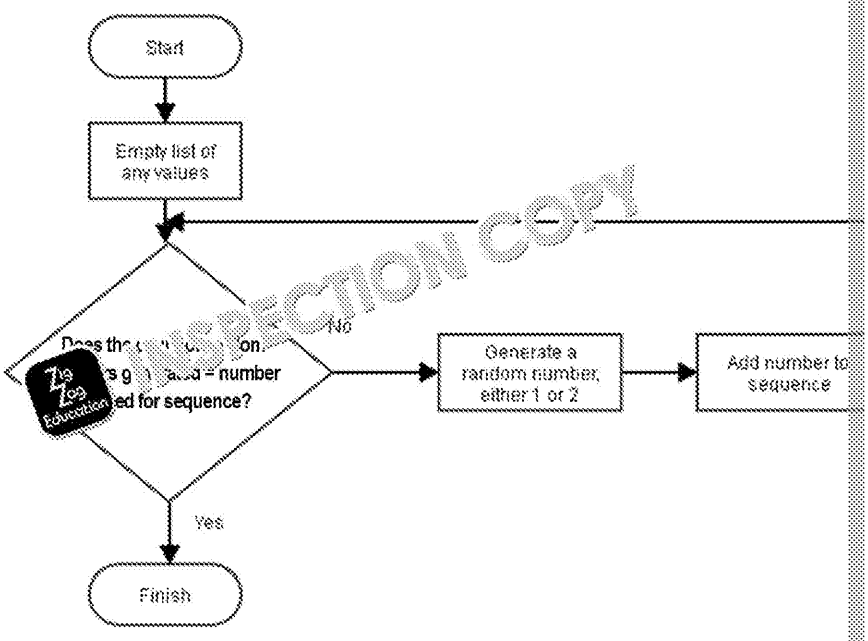
Task C Challenge
The full code is shown below (with missing parts underlined)

```
#import time function from sleep module
#time will be used to implement the seconds' delay
from time import sleep

#import os module. This will be used to clear the display
import os

#create a function to validate the inputs for minutes and seconds
def getNumber():
    #loop until a valid number is input
    while True:
        try:
            number=int(input(""))
            #if valid number is input, exit the loop
            if number >=0 and number <=59:
                break
```

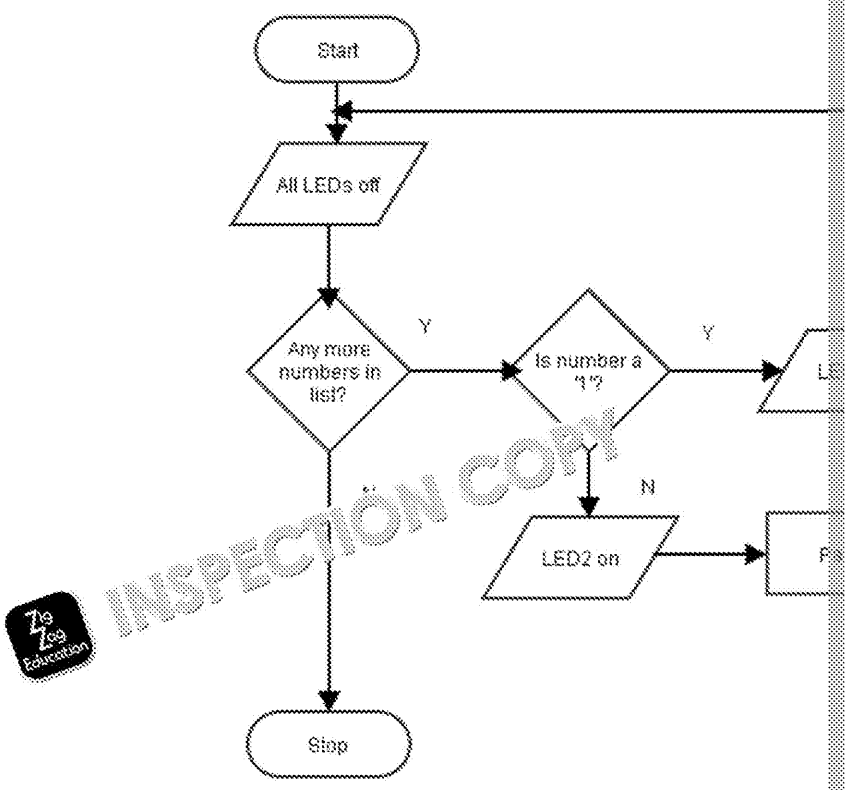
```
                        else:
                        #if a number outside the valid range is i
                        #display a message informing the user of
                                print("Enter a number from 0 to 59
                except:
                        #if a letter etc. is input
                        #display a message informing the user of
                        print("Not a number. Enter a number from
        #return the input number
        return number

#main

#clear the display
os.system("clear")

#get minutes and seconds for countdown
print("How many minutes? (0 to 59) ")
minutes = get_number()
print("How many seconds? (0 to 59) ")
seconds = get_number()

#check that there is still time to count down
if minutes >=0:
        #create countdown loop
        while True:
                #clear the display
                os.system("clear")
                #display time remaining
                display = "Time remaining: " + str(minutes)+":"+
                print(display)
                #pause for one second
                sleep(1)
                #reduce time remaining by one second
                seconds -=1
                #check to see if seconds remaining is now less t
                #if yes, take 1 minutes left by 1 and reset seco
                if seconds ==-1:
                        minutes -=1
                        seconds =59
                #check to see if minutes remaining is now less t
                #if yes then exit the loop
                if minutes ==-1:
                        break

#display finish message
print("Time's up!")
```

Task D Challenge (2)
These are the correct flow charts with the correct statements.

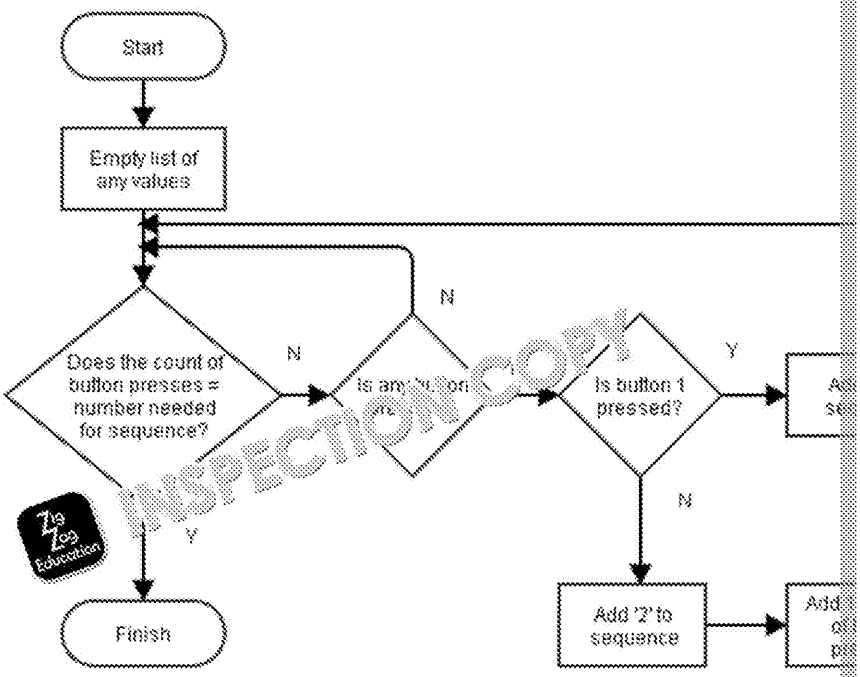
The sequence is generated using this algorithm – **FLOWCHART 2:**



The LED's are lit in sequence using this algorithm – **FLOWCHART 3:**



The player's sequence is recorded using this algorithm – **FLOWCHART 1:**



This algorithm checks whether the player got the sequence right – **FLOWCHA**