



Introduction to the Raspberry Pi

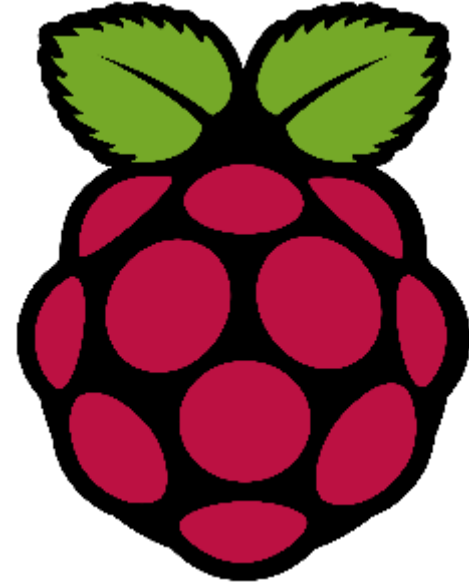
Jani Kalasniemi



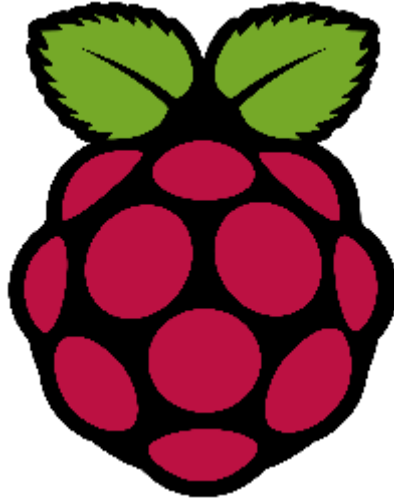
Idea^s

Agenda

- **FIRST STEPS**
 - What is a Raspberry Pi?
 - Setting up your Raspberry Pi
 - Using your Raspberry Pi as a Desktop Computer
- **PROGRAMMING WITH PYTHON**
 - Start Programming in Python
 - Write your own Function
 - Using GPIO to drive outputs and read inputs
- **MAKING A DESKTOP APPLICATION**
 - Learn about event-driven programming
 - Create a simple Graphical User Interface(GUI)



FIRST STEPS



What is a Raspberry Pi?

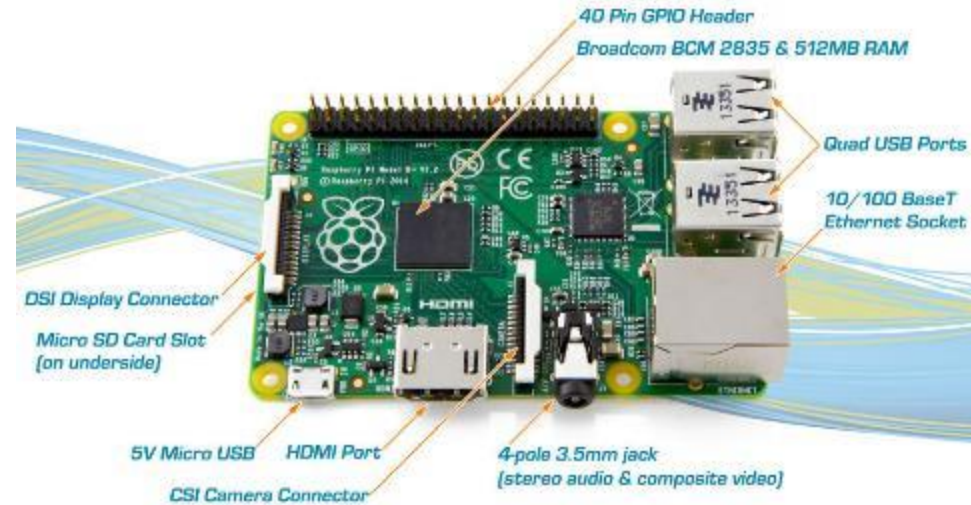
- A credit-card-sized computer
- Runs on several operating systems
 - Raspbian **WE WILL USE RASPBIAN**
 - Windows 10 IoT Core
 - RetroPie
 - OpenElec
- Possible to connect with a variety of sensors to interact with the physical world.
 - Make decisions based on processing of gathered sensor data



Getting started with your Raspberry Pi

Basic 1-2-3

1. Connect the Raspberry Pi to the keyboard, mouse and the screen.
2. Plug in the power and wait for it to boot.
3. **Username : pi**
Password : raspberry
(should not be needed)



Using your Raspberry Pi as a Desktop Computer

- Try to access the internet through your Raspberry Pi and find a picture that reflects your group. Hands up when you are done, for help if needed!
- Save this picture to the desktop.



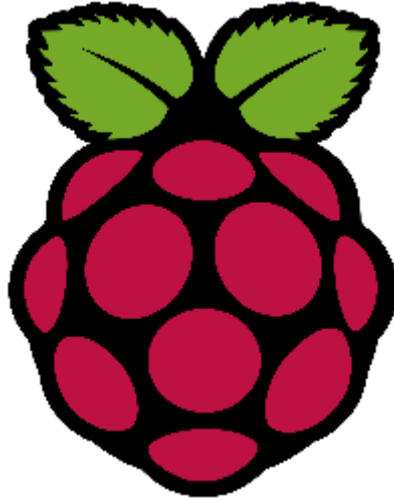
Communicating through the terminal

- Start the terminal by going to Accessories and then clicking on the Terminal program.
- Run through the commands and get a feel for how to change directories, and list its contents.

Basic Commands cheat-sheet	
Command	Description
<code>ls</code>	List contents of directory
<code>cd <dir></code>	Change to directory: <dir>
<code>mkdir <dir></code>	Make a new directory called <dir>
<code>rmdir <dir></code>	Delete the empty directory, <dir>
<code>rm <file></code>	Delete the file, <file>
<code>nano</code>	Run the <i>nano</i> text editor program



PROGRAMMING WITH PYTHON



Programming in Python3

- General-purpose programming language used for scientific and numerical applications as well as desktop and web applications.
- Open source language that has a lot of online resources for problems you might come across.
- We are using Thonny IDE for programming with Python3 in our Raspberry Pi 3 Model B
- **<https://stackoverflow.com> is your best friend for programming troubles**



The GPIO layout

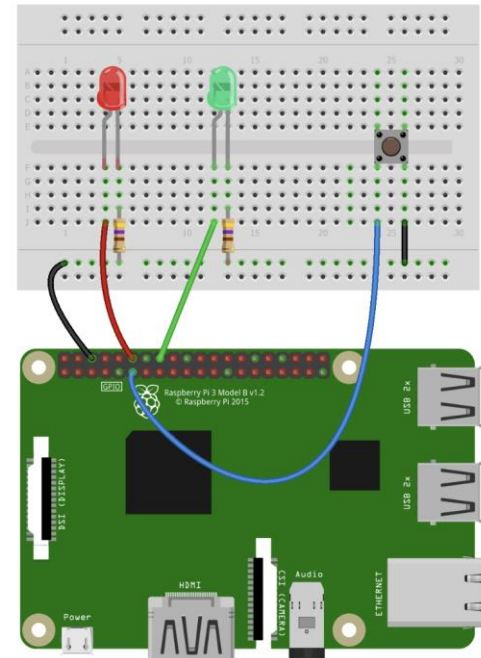
General Purpose Input / Output



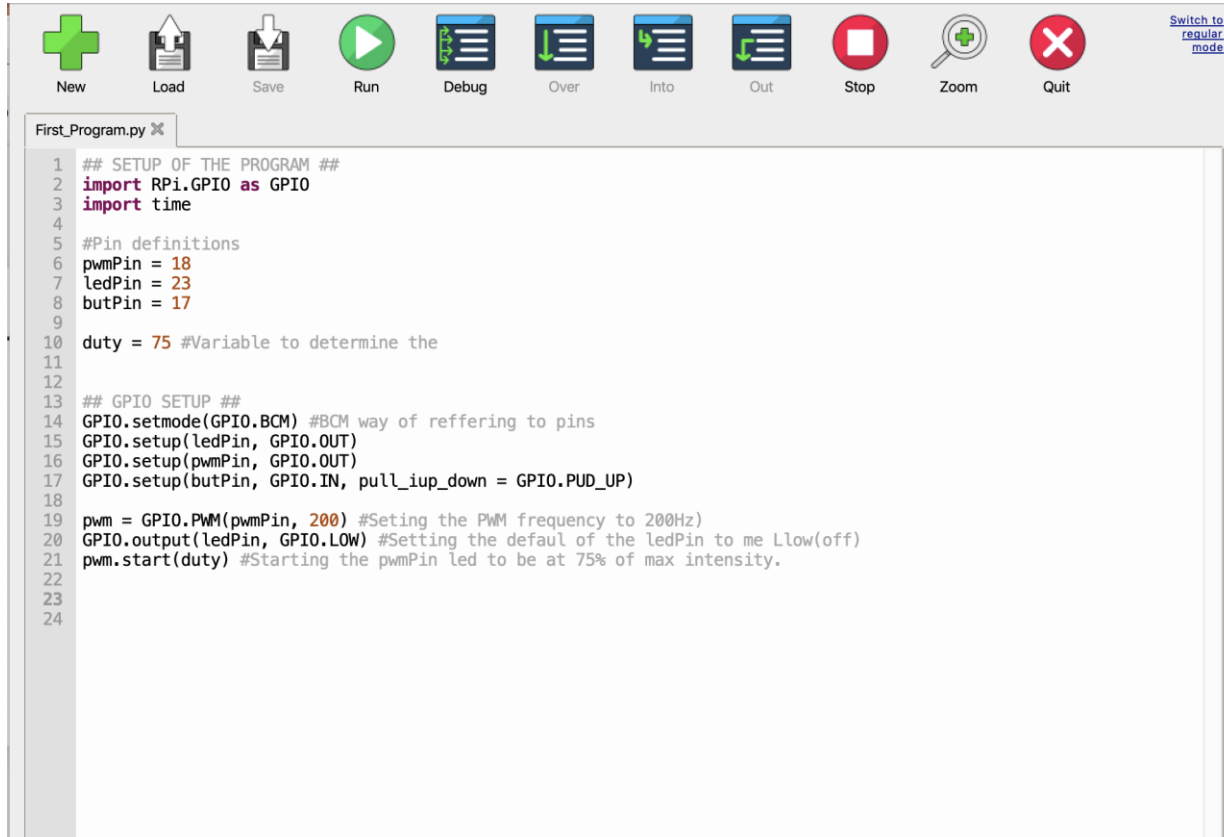
3V3 POWER	1	2	5V POWER
GPIO2 SDA1 (2C)	3	4	5V POWER
GPIO3 SCL1 (2C)	5	6	Ground
GPIO4 1-wire	7	8	GPIO14 UART0_TXD
Ground	9	10	GPIO15 UART0_RXD
GPIO17	11	12	GPIO18 PWM_CLK
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3 POWER	17	18	GPIO24
GPIO10 SPI_MOSI	19	20	Ground
GPIO9 SPI_MISO	21	22	GPIO25
GPIO11 SPI_CS0	23	24	GPIO8 SPI_CS1
Ground	25	26	GPIO7 SPI_CS2
ID_SD (GPI2 with PWR)	27	28	ID_SC (GPI5 with PWR)
GPIO5	29	30	Ground
GPIO6	31	32	GPIO12
GPIO13	33	34	Ground
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
Ground	39	40	GPIO21

Use the GPIO to drive outputs and read inputs

- Build the circuit in the figure.
Use 470 Ohm resistor
- Next we will make a program that makes one LED blink on and off, and the other dim up and down in intensity when the Button is pressed.



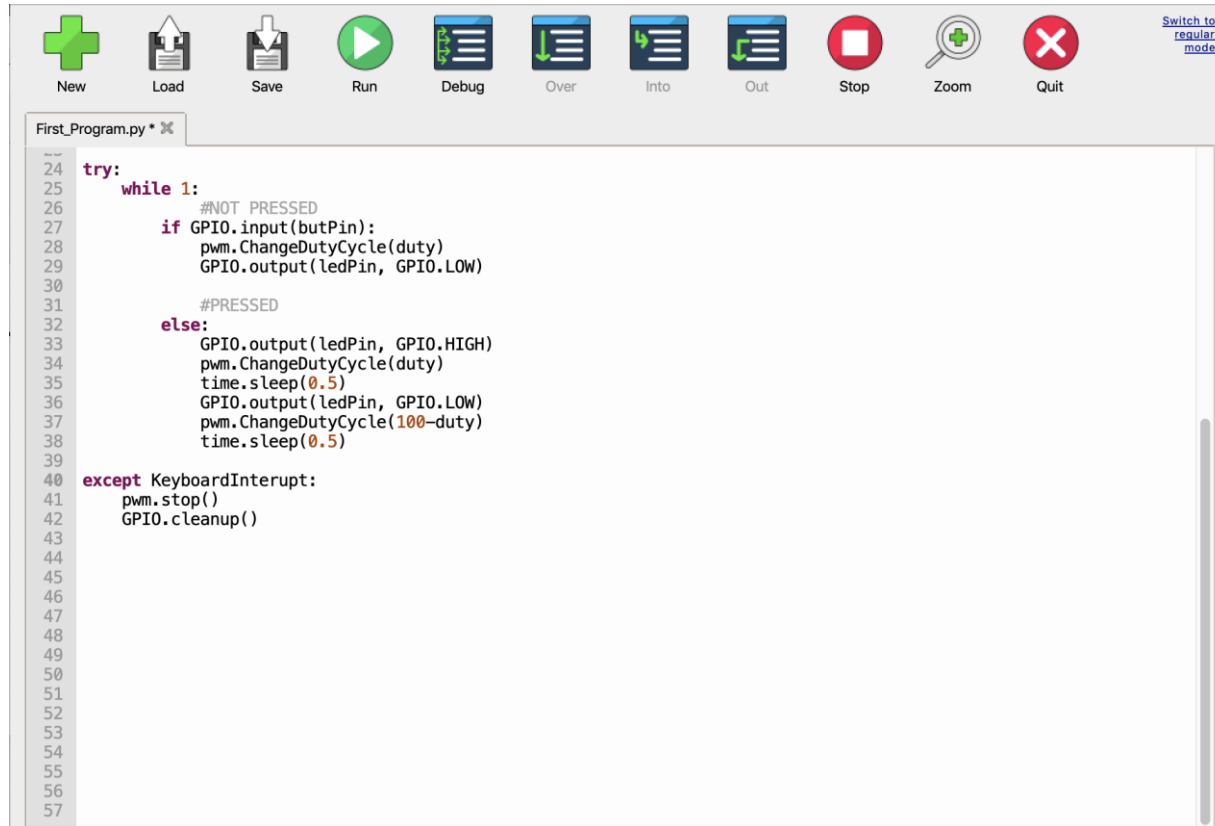
Use the GPIO to drive outputs and read inputs



The image shows a screenshot of a code editor interface. At the top, there is a toolbar with icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. Below the toolbar, the code editor displays a Python script named 'First_Program.py'. The script is a Python program that uses the RPi.GPIO library to control an LED and a PWM signal. The code includes comments and is numbered from 1 to 24.

```
1  ## SETUP OF THE PROGRAM ##
2  import RPi.GPIO as GPIO
3  import time
4
5  #Pin definitions
6  pwmPin = 18
7  ledPin = 23
8  butPin = 17
9
10 duty = 75 #Variable to determine the
11
12
13 ## GPIO SETUP ##
14 GPIO.setmode(GPIO.BCM) #BCM way of referring to pins
15 GPIO.setup(ledPin, GPIO.OUT)
16 GPIO.setup(pwmPin, GPIO.OUT)
17 GPIO.setup(butPin, GPIO.IN, pull_iup_down = GPIO.PUD_UP)
18
19 pwm = GPIO.PWM(pwmPin, 200) #Setting the PWM frequency to 200Hz)
20 GPIO.output(ledPin, GPIO.LOW) #Setting the default of the ledPin to me Llow(off)
21 pwm.start(duty) #Starting the pwmPin led to be at 75% of max intensity.
22
23
24
```

Use the GPIO to drive outputs and read inputs



The image shows a screenshot of an IDE window titled "First_Program.py * x". The window contains a Python script for controlling an LED using a GPIO pin. The script is as follows:

```
24 try:
25     while 1:
26         #NOT PRESSED
27         if GPIO.input(butPin):
28             pwm.ChangeDutyCycle(duty)
29             GPIO.output(ledPin, GPIO.LOW)
30
31         #PRESSED
32     else:
33         GPIO.output(ledPin, GPIO.HIGH)
34         pwm.ChangeDutyCycle(duty)
35         time.sleep(0.5)
36         GPIO.output(ledPin, GPIO.LOW)
37         pwm.ChangeDutyCycle(100-duty)
38         time.sleep(0.5)
39
40 except KeyboardInterrupt:
41     pwm.stop()
42     GPIO.cleanup()
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
```

The IDE interface includes a toolbar with icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. A "Switch to regular mode" link is visible in the top right corner.

Common errors and debugging

- Python is sensitive to indentation.
- Make sure that the variables and the functions have the same exact writing. Watch out for small and Capital letters!
- Remember that python starts counting from zero.
i.e when using loops.



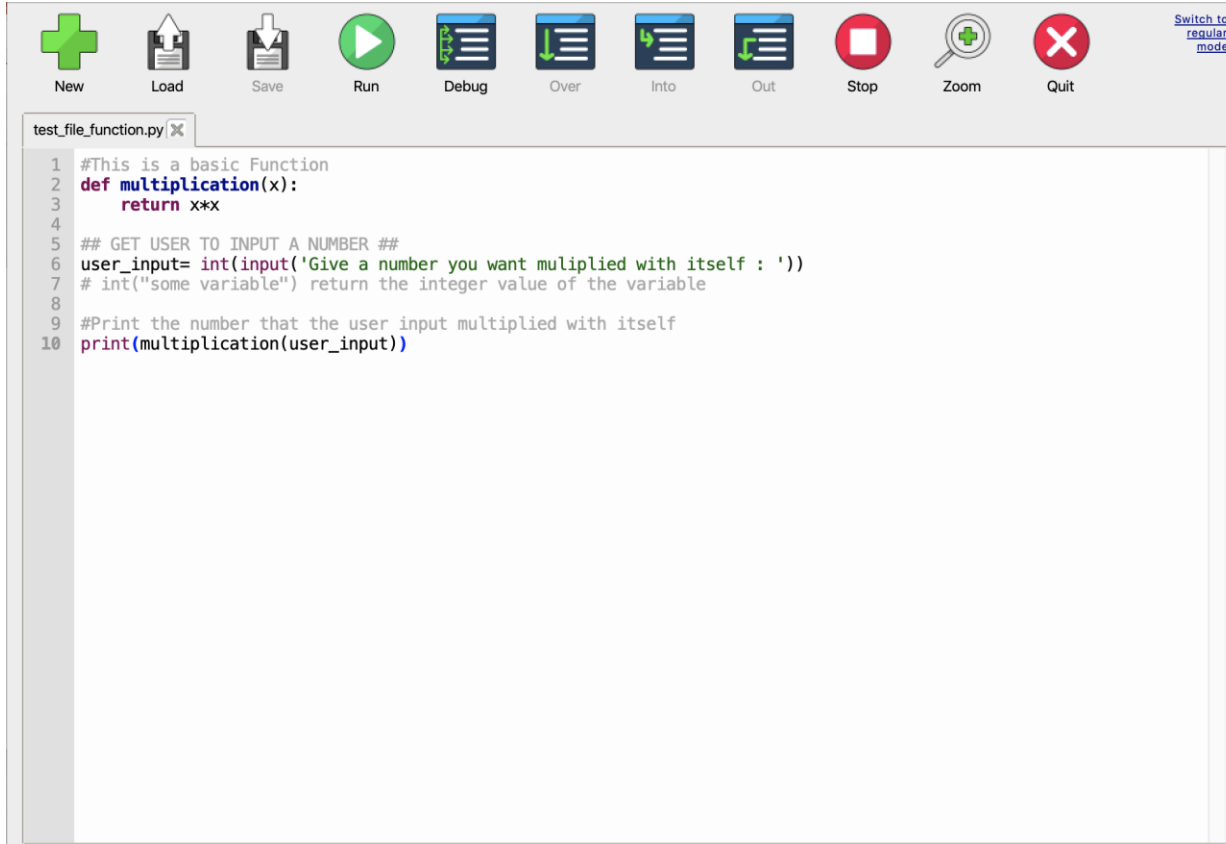
Test your program and what you have made!

Questions?



Idea^s

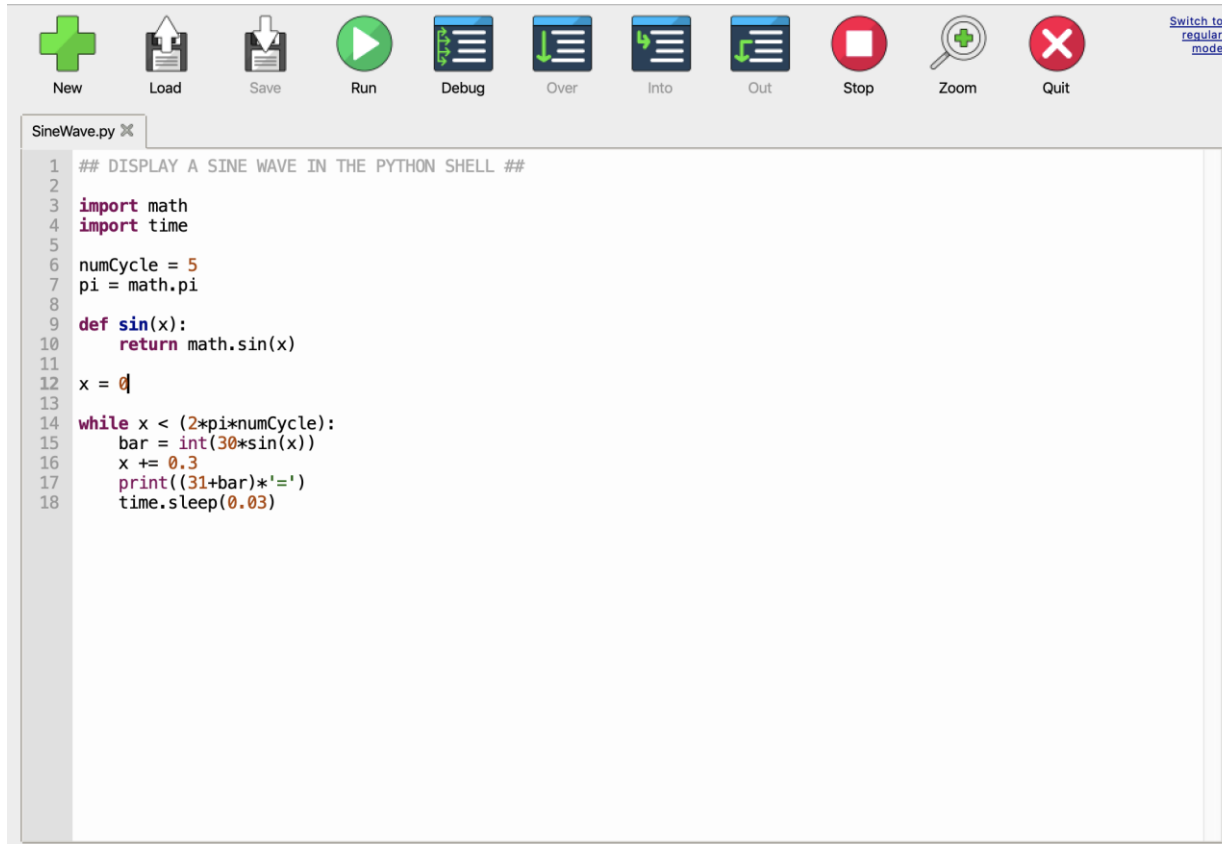
Write your own Functions



The screenshot shows the PyCharm IDE interface. At the top, there is a toolbar with icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. A link for "Switch to regular mode" is visible in the top right corner. The main editor window displays a Python script named "test_file_function.py" with the following code:

```
1 #This is a basic Function
2 def multiplication(x):
3     return x*x
4
5 ## GET USER TO INPUT A NUMBER ##
6 user_input= int(input('Give a number you want multiplied with itself : '))
7 # int("some variable") return the integer value of the variable
8
9 #Print the number that the user input multiplied with itself
10 print(multiplication(user_input))
```


Making a $\sin(x)$ function to get some output



The image shows a screenshot of an IDE interface. At the top, there is a toolbar with icons for New (green plus), Load (floppy disk), Save (floppy disk), Run (play button), Debug (bug), Over (bug with arrow), Into (bug with arrow), Out (bug with arrow), Stop (red square), Zoom (magnifying glass), and Quit (red X). A link "Switch to regular mode" is visible in the top right corner. Below the toolbar, a tab labeled "SineWave.py" is open. The code editor contains the following Python code:

```
1  ## DISPLAY A SINE WAVE IN THE PYTHON SHELL ##
2
3  import math
4  import time
5
6  numCycle = 5
7  pi = math.pi
8
9  def sin(x):
10     return math.sin(x)
11
12  x = 0
13
14  while x < (2*pi*numCycle):
15     bar = int(30*sin(x))
16     x += 0.3
17     print((31+bar)*'=')
18     time.sleep(0.03)
```

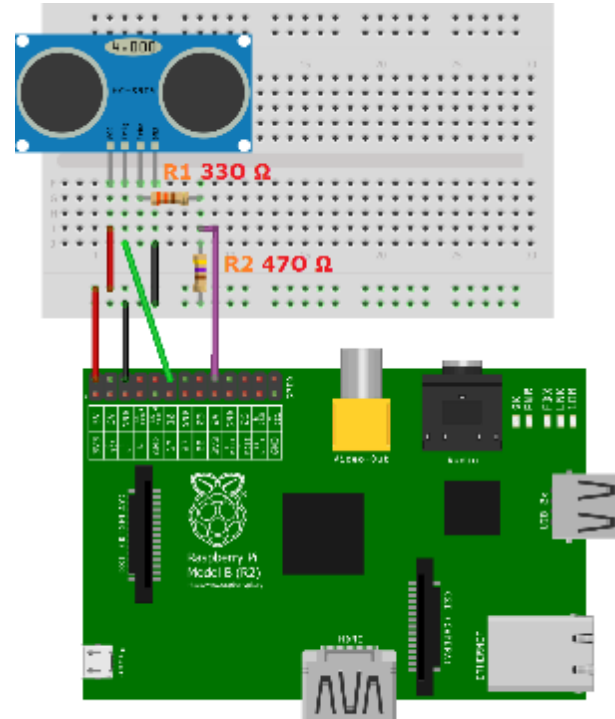

Using a sensor with the Raspberry Pi

- The GPIO on the Raspberry Pi can only handle a maximum of 3.3V, but many of the available sensors can return a signal of 5V to the GPIO. This will damage the Raspberry Pi.
- How do we deal with this?
 - Resistors connected as voltage dividers.
- Important to be aware of this.



Measuring Distance with an Ultrasonic Sensor: HC-SR04

- The Trigger sends a burst of sound that bounces and hits the Echo.
- If we measure the time it takes from sending to receiving we can calculate the distance.
- NOTE: Two different resistors here

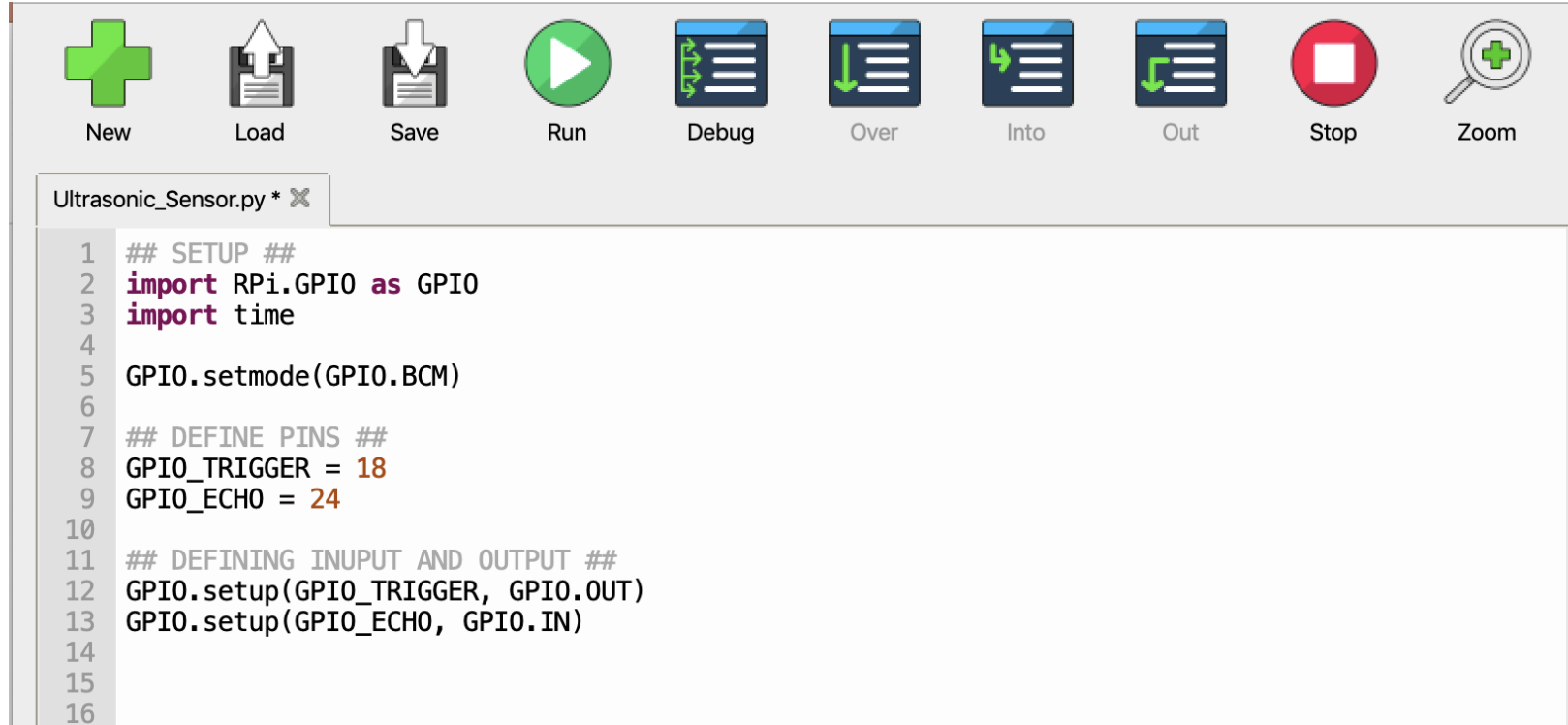


fritzing



Idea^s

Setting up the Raspberry Pi



The screenshot displays the IDE interface with a toolbar at the top and a code editor window below. The toolbar contains the following icons and labels: New (green plus), Load (floppy disk), Save (floppy disk with arrow), Run (green play button), Debug (blue bug), Over (blue document with green arrow), Into (blue document with green arrow), Out (blue document with green arrow), Stop (red square), and Zoom (magnifying glass with plus). The code editor window is titled "Ultrasonic_Sensor.py *" and contains the following Python code:

```
1  ## SETUP ##
2  import RPi.GPIO as GPIO
3  import time
4
5  GPIO.setmode(GPIO.BCM)
6
7  ## DEFINE PINS ##
8  GPIO_TRIGGER = 18
9  GPIO_ECHO = 24
10
11 ## DEFINING INUPUT AND OUTPUT ##
12 GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
13 GPIO.setup(GPIO_ECHO, GPIO.IN)
14
15
16
```

Now we need a function to calculate the distance

- With the sensor we can measure the time it takes for a signal to be sent out and reflected back.
- We need to convert the elapsed time to distance by using the speed of sound in air and multiplying it with the time divided by two (Because it travels to the object, and back).



The distance function

```
17 def distance():
18     # set Trigger to HIGH,
19     GPIO.output(GPIO_TRIGGER, True)
20
21     # set Trigger after 0.01ms to LOW
22     # this sends a signal that the ECHO can read in
23     time.sleep(0.00001)
24     GPIO.output(GPIO_TRIGGER, False)
25
26     StartTime = time.time()
27     StopTime = time.time()
28
29     # save StartTime
30     while GPIO.input(GPIO_ECHO) == 0:
31         StartTime = time.time()
32
33     # save time of arrival
34     while GPIO.input(GPIO_ECHO) == 1:
35         StopTime = time.time()
36
37     # time difference between start and arrival
38     TimeElapsed = StopTime - StartTime
39     # multiply with the sonic speed (34300 cm/s)
40     # and divide by 2, because: there and back
41     distance = (TimeElapsed * 34300) / 2
42
43     #Return the Distance that the sensor measured
44     return distance
45
```



Making it loop until we tell it to stop

```
47 try:
48     while True:
49         #Find the measured distance with distance()
50         dist = distance()
51         print ("Measured Distance = %.1f cm" % dist)
52
53         #Wait one second before a new measurement is made
54         time.sleep(1)
55
56     # Reset by pressing CTRL + C
57 except KeyboardInterrupt:
58     print("Measurement stopped by User")
59     GPIO.cleanup()
60
```



If the program is not giving any measurements

- Check the wiring!
- Check you code one more time.
- Check the resistors
- Check the wiring!

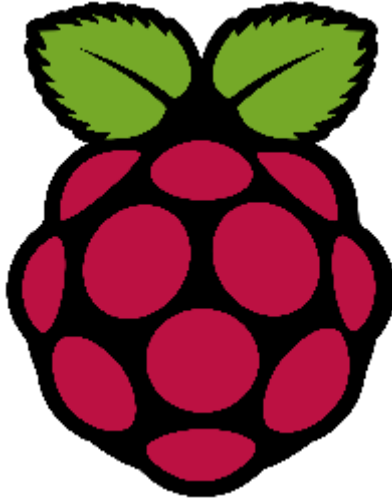


Questions?



Idea^s

MAKING A DESKTOP APPLICATION



Introducing Tkinter : A GUI Toolkit for Python

- De facto standard GUI for Python
- Toolkit for GUI programming in Python. Not the only one, but the most commonly used one.
- Enables you to build GUI's with buttons, sliders, drop downs, and other interactions for the user of the program/device you build.



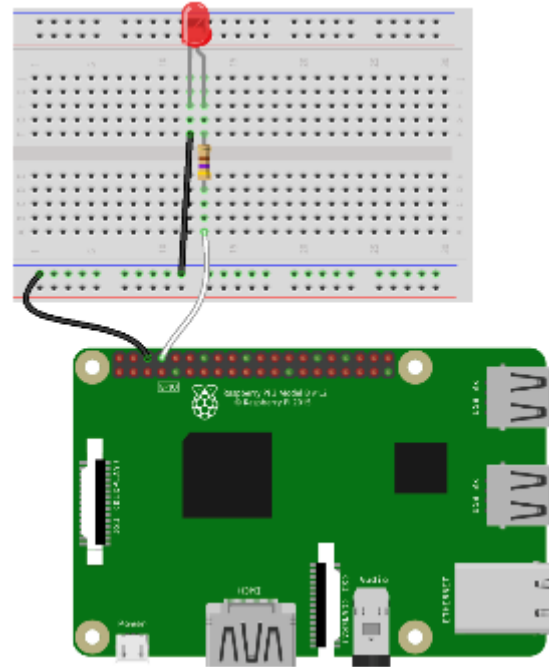
Event driven programming.

- When creating a GUI you use event driven programming.
- In the GUI all buttons, sliders, etc, are known as **Widgets**.
 - When a widget is used it make an event in you code start.

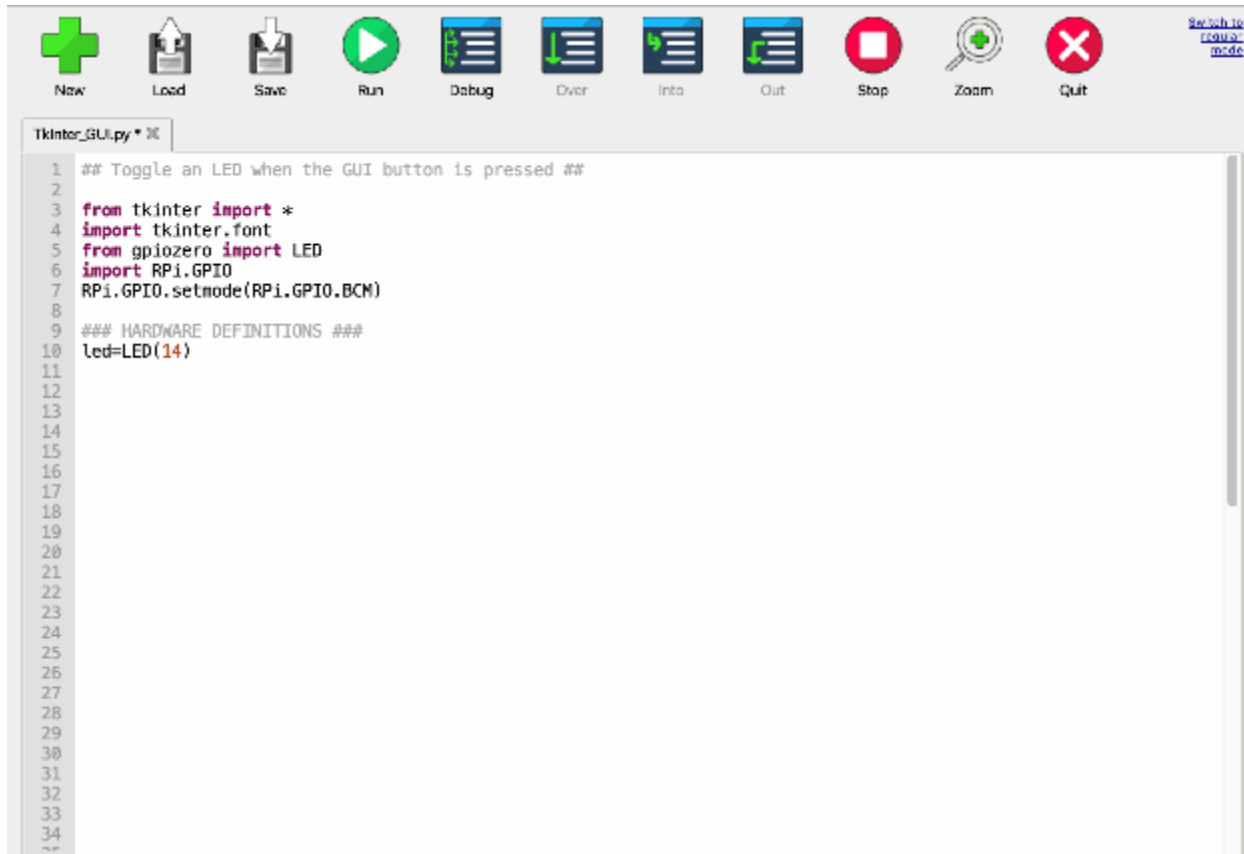


Create a simple Graphical User Interface(GUI)

- We will create a GUI that enables us to turn the LED on and off with a button in a GUI.
- Use the 470 Ohm resistor in the circuit.



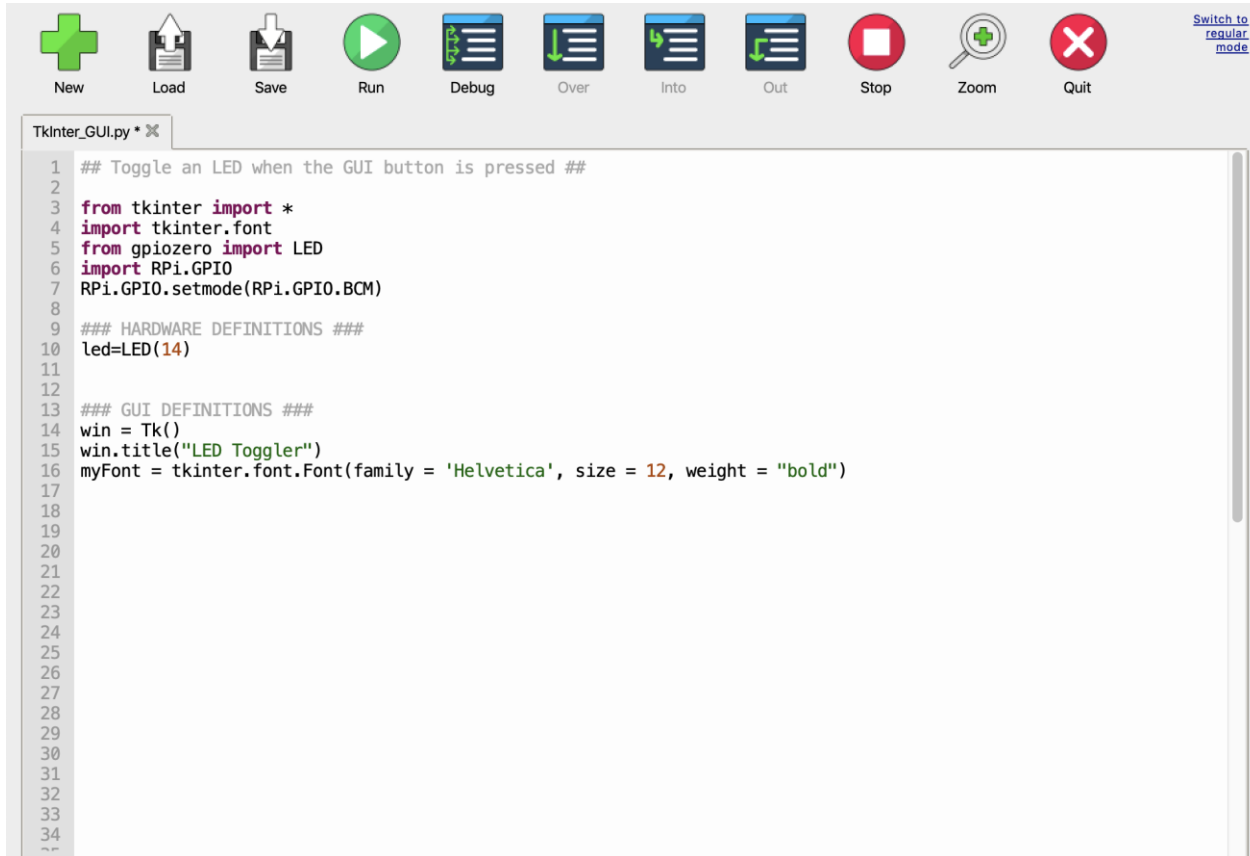
Building the code for this GUI



The screenshot shows an IDE window titled "Tkinter_GUI.py" with a toolbar at the top containing icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. The code in the editor is as follows:

```
1  ## Toggle an LED when the GUI button is pressed ##
2
3  from tkinter import *
4  import tkinter.font
5  from gpiozero import LED
6  import RPi.GPIO
7  RPi.GPIO.setmode(RPi.GPIO.BCM)
8
9  ### HARDWARE DEFINITIONS ###
10 led=LED(14)
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
```

Building the GUI



The screenshot shows an IDE window titled "TkInter_GUI.py *". The code is as follows:

```
1  ## Toggle an LED when the GUI button is pressed ##
2
3  from tkinter import *
4  import tkinter.font
5  from gpiozero import LED
6  import RPi.GPIO
7  RPi.GPIO.setmode(RPi.GPIO.BCM)
8
9  ### HARDWARE DEFINITIONS ###
10 led=LED(14)
11
12
13 ### GUI DEFINITIONS ###
14 win = Tk()
15 win.title("LED Toggler")
16 myFont = tkinter.font.Font(family = 'Helvetica', size = 12, weight = "bold")
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
~
```


Building the GUI

- Now we need to decide what the GUI will actually do.
 1. We will create a button that will switch the LED on and off
 2. We also have to make an exit button so that when we exit the program the LED turns off, and the GPIO port is reset to default as input ports.



Creating the widget triggered event : ledToggle()

```
19 ### Event Functions ###
20 def ledToggle():
21     if led.is_lit:
22         led.off()
23         ledButton["text"]="Turn LED on" # Change only the button text property
24     else:
25         led.on()
26         ledButton["text"]="Turn LED off"
27
```



Creating the widget triggered event : close()

```
29 def close():  
30     RPi.GPIO.cleanup()  
31     win.destroy()  
32
```



Creting the toggle button, aka a Widget

```
35 ### WIDGETS ###  
36  
37 # Button, triggers the connected command when it is pressed  
38 ledButton = Button(win, text='Turn LED on', font=myFont, command=ledToggle, bg='bisque2', height=1, width=24)  
39 ledButton.grid(row=0, column=1)  
40
```

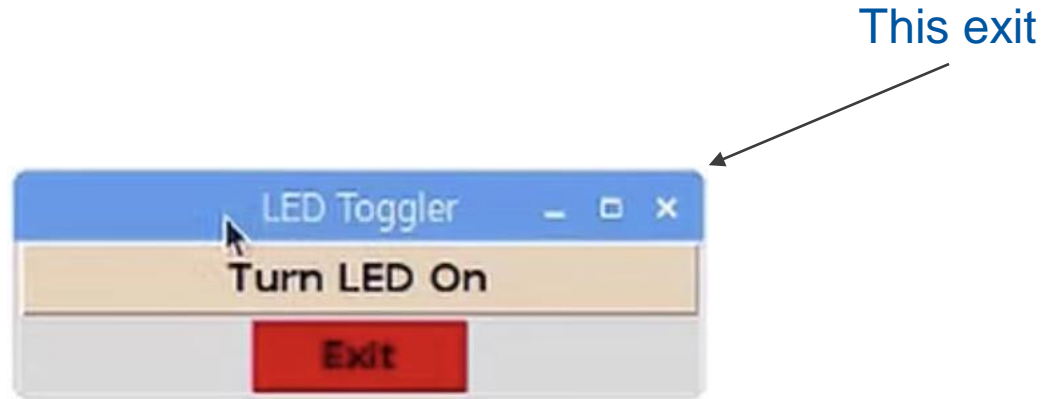


Creating the exit button

```
41 exitButton = Button(win, text='Exit', font=myFont, command=close, bg='red', height=1, width=6)  
42 exitButton.grid(row=2, column=1)  
43
```



What happens if we exit the program through the window exit?



Making sure that we have a clean exit

```
44 win.protocol("WM_DELETE_WINDOW", close) # cleanup GPIO when user closes window  
45
```



Idea^s

What happens if you run your program and press a GUI button?



Idea^s

Making the program run after we exit it on purpose

```
46 win.mainloop() # Loops forever
47
48 print('will this be printed?')
49
```

Will the last line be printed? If so, when?



Idea^s

Test your program and what you have made!

Questions?



Idea^s

Summary – What have we done?

- Introduced the Raspberry Pi and used it as a computer
- Learned simple commands to use in terminal
- Introduced the GPIO and how to get inputs from the physical world, interpret them, and create output from input
- Used a **XX** sensor to read and evaluate data from your environment
- Introduced the Tkinter package and created a simple, event-driven, GUI



Resources

- **Raspberry Pi GPIO examples:**
<https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/>
- **Python Documentation:**
<https://www.python.org/doc/>
- **Tkinter Documentation:**
https://www.tutorialspoint.com/python/python_gui_programming.htm





Idea^s