SASTA OLIPHANT SCIENCE AWARDS

SOUTH AUSTRALIAN SCIENCE TEACHERS ASSOCIATION

Prize Winner

Computer Programming, Apps & Robotics

Year 5-6

Sien Mitchell

Colonel Light Gardens Primary School

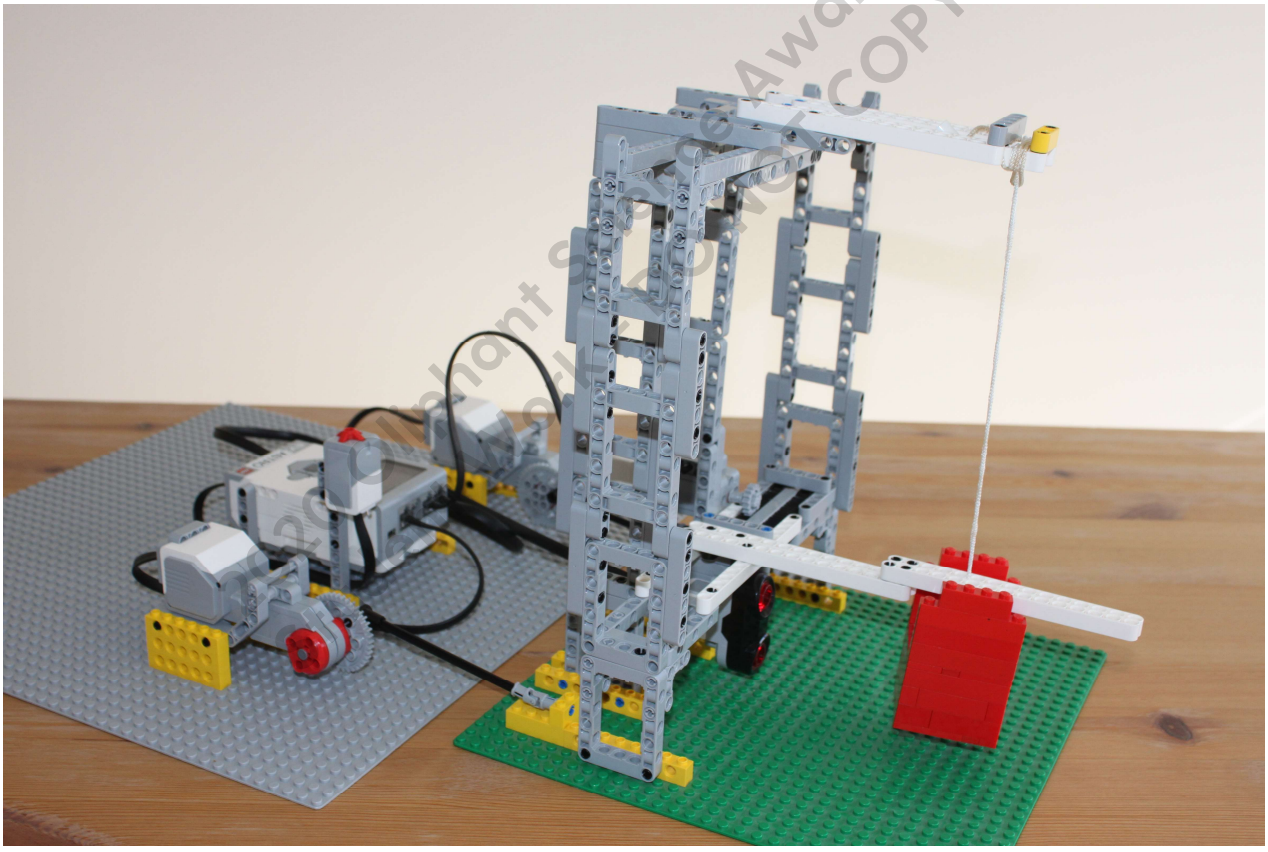
Government of South Australia
Department for Education

SASTA South Australian Science Teachers Association

ROWE SCIENTIFIC PTY. LTD. www.rowe.com.au

Earthquake Alert

Sien Mitchell



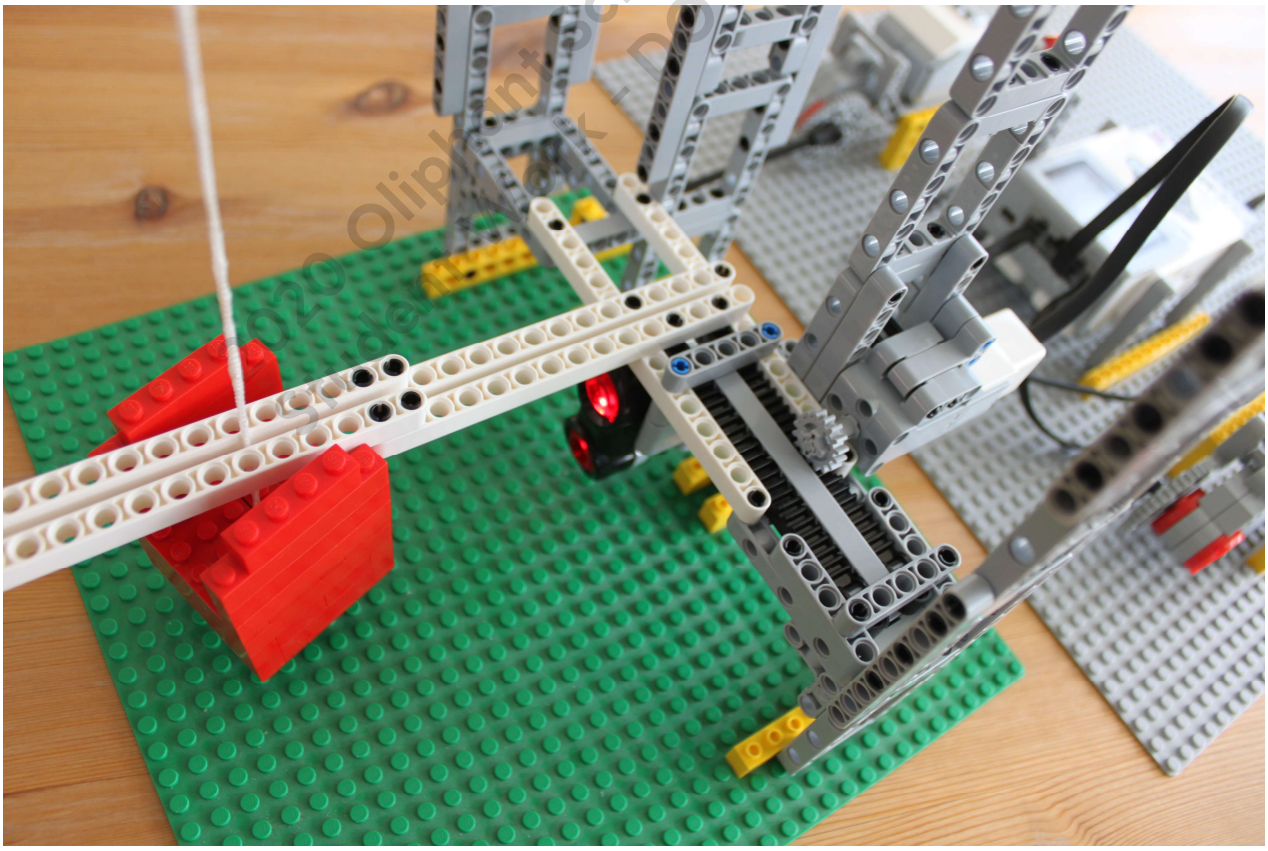
Aim

I built a seismometer to detect vibrations and simulate natural disasters like earthquakes, tsunamis, landslides and volcano eruptions. To simulate any earth movements, I programmed two motors that would shake the earth's surface forwards and backwards. It measures the distance of the weight and sends the movement data to the Earthquake Alert program. The Earthquake Alert program plots a line graph of the movement over time and rates the intensities from the alert warning colour scale.

How does the Seismometer work?

The seismometer has two large motors, a medium motor, an ultrasonic sensor and a touch sensor and is programmed using EV3 software. The power and rotation of the two large motors can be changed for the earthquakes shake by using the EV3 controls. Power starts at 20% and can also be increased or decreased by 10% in the EV3 controls (up or down button). Rotation of the large motors can be set to 90 or 180 degree (left or right button) to make the earthquake tremors weaker or stronger.

By holding down the touch sensor the motors rotate the arms to simulate the earthquake, once you let go of the touch sensor the earthquake stops. The seismometer has a string with a weight attached. Inside the red weight I added marbles to make it heavier. Because of the shakes the weight swings and the distance is measured every second by the ultrasonic sensor.



The medium motor is used like a brake to stop the weight from swinging around. By pressing the centre button, it will rotate the cog which moves the track to close the arms. Then after 5 seconds it opens the arms.

How does the Earthquake Alert work?

The Earthquake Alert program is written in Python. The computer and EV3 device connect by Bluetooth. Both devices message each other to check if the connection has happened. To open Earthquake Alert window, you run the Python program in Visual Studio Code. After that you run the EV3 program using the EV3 controls. To simulate the 5 second earthquake shake, you click on the Earthquake button in the Earthquake Alert window.

It will get the distance, power and rotation messages from EV3. The distance data is used to plot the line graph as it happens. The distance of the weight swing is what you can feel when a tremor happens. How big the swing is means it feels like a strong tremor. So really the distance of the swing is the actual movement. The bar graph plots the weight swing movement to the different alert warning scale. To work out the scale I experimented with different powers and rotations.

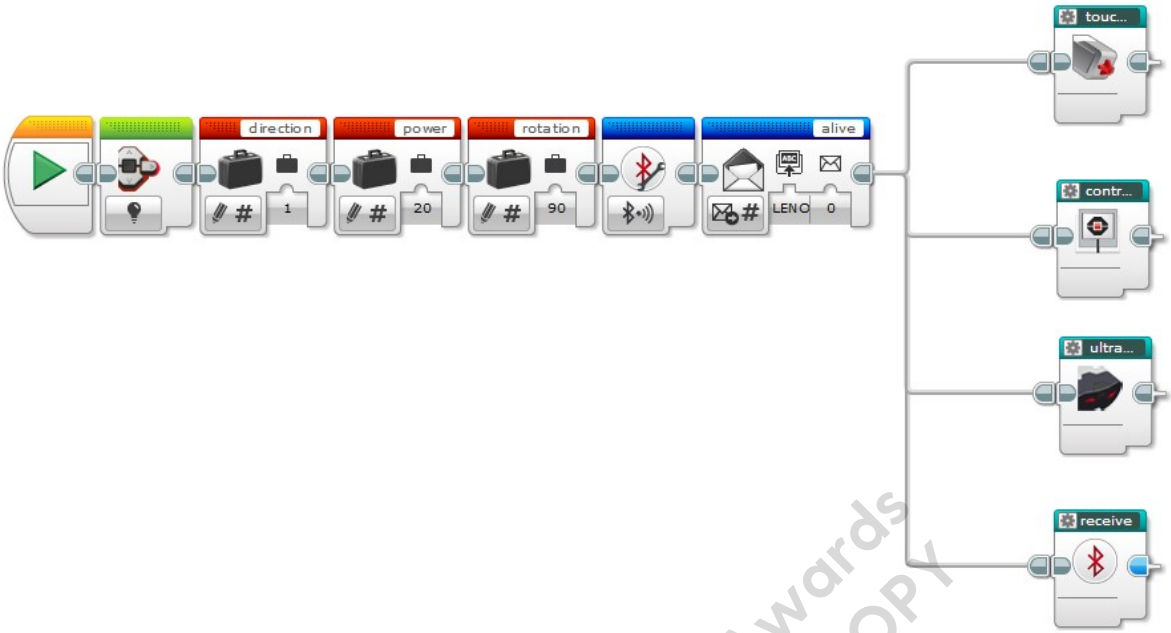
This is my Alert Warning scale.

Alert Warning	***Colour***	***What it feels like***
Not felt	light green	Barely felt
Weak	green	Felt by a few people indoors or in higher buildings
Moderate	light blue	Felt by some people. Some objects broken.
Strong	yellow	Felt by everyone and animals get frightened, big objects can be moved.
Severe	orange	Structures can be damaged, buildings can collapse.
Extreme	Indian red	Well built structures can be destroyed, bridges damaged, underground pipes and tunnels broken.
Disastrous	red	Damage everywhere, all building collapsed, people can be killed.

By running earthquake simulations for different power and rotation settings and earthquake shake time length. You can look for movement patterns in the Earthquake Alert window. The strength of the tremors is recorded in the bar graph and table which can be used to predict any damage to structures. Earthquake Alert can be used to warn people when a damaging earthquake is about to happen. This could save lives if the people are evacuated just before a disastrous earthquake.

EarthquakeShake EV3 project

This is the main earthquake program.



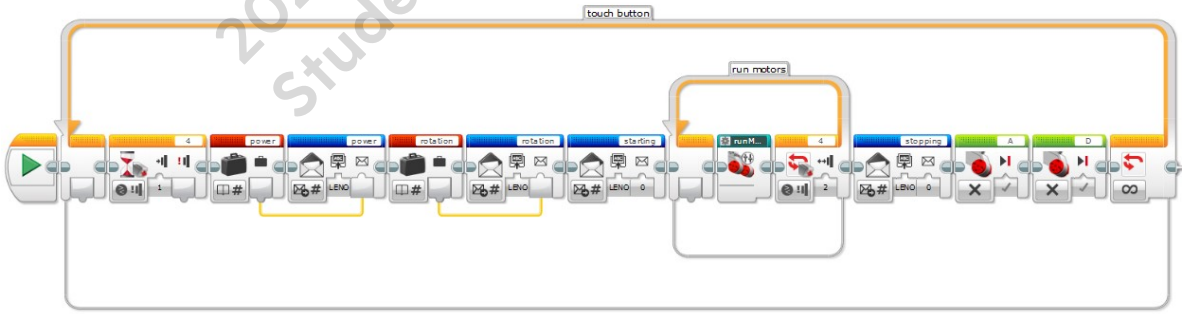
Each myBlock program runs independently at the same time. Each myBlock program use wait blocks which are waiting for messages or buttons to happen.

To start the earthquake hold the touch sensor button down. To stop the earthquake let go of the touch sensor button.

To control the power and rotation of the large motors. Press up and down button for power. Press left for 90 degree rotation and right for 180 degree rotation.

Press centre button to brake the swinging weight.

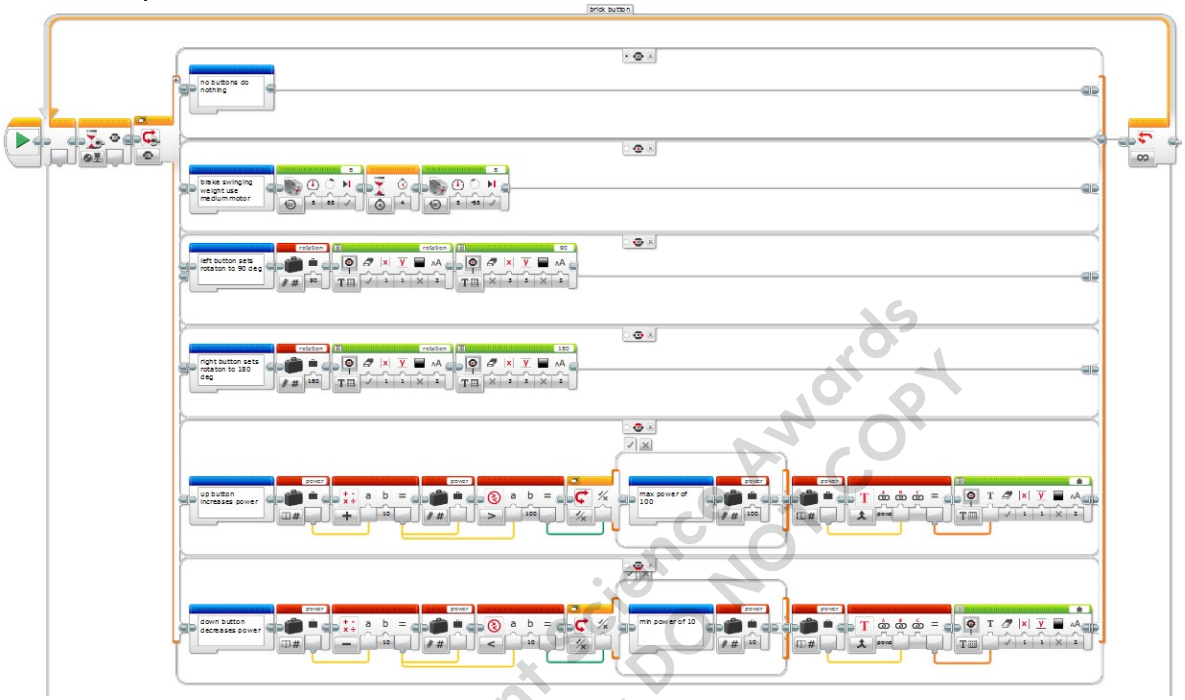
touchButton myBlock



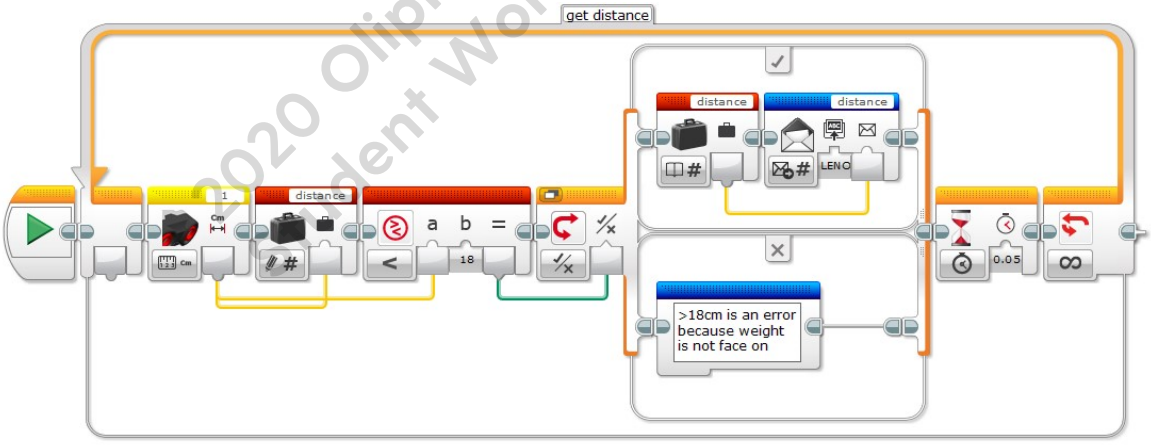
Waits until touch sensor detects button press. Send messages by EV3 bluetooth to computer. The messages are power percent and rotation degree. Run the large motors as set by controls.

After letting go of touch sensor the large motors are stopped.

controls myBlock



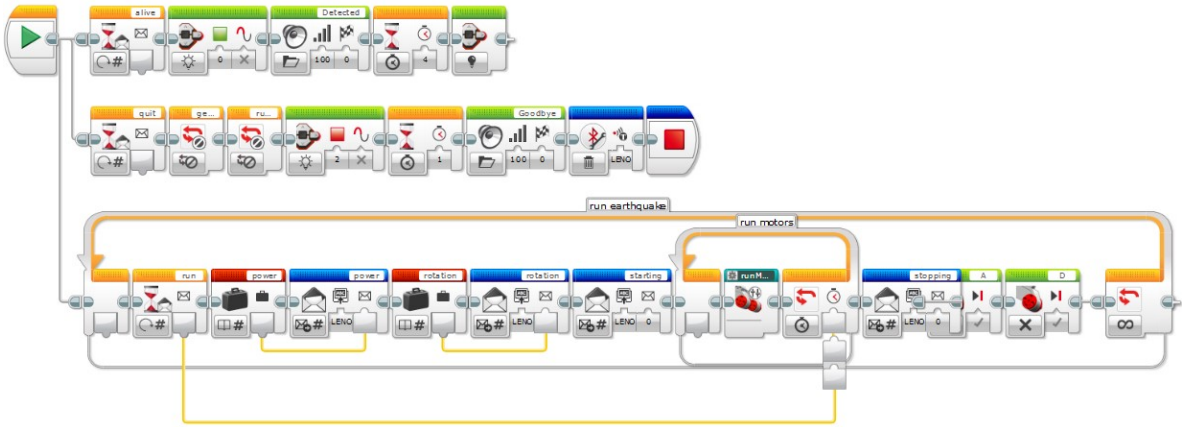
ultrasonic myBlock



Measures distance using an ultrasonic sensor.
If ultrasonic sensor detects distance under 18cm. It sends distance values to the computer via bluetooth.

Wait 0.05 seconds for computer to plot graph.

receive myBlock



Receives message from computer to connect EV3 program.

Receives message from computer to quit EV3 program. Stops get distance loop. Stops run earthquake loop. Disconnect bluetooth to computer.

Waits until message from the computer to run earthquake for 5 seconds.
Sends power and rotation values to the computer.
Runs the large motors for 5 seconds.

runMotors program

If negative direction the large motor rotates backwards.
If postive direction the large motor rotates forwards.

Power and rotation is set in controls myBlock program.
Both large motors use the same power and rotation.

resetBluetooth program

It turns on bluetooth on EV3 and resets connection.

EarthquakeAlert Python window

Power (%)	Rotation (deg)	Time Taken (sec)	Distance (cm)	Alert Warning
10	90	3.94	3.20	weak
20	90	3.20	4.40	moderate
30	90	7.96	6.30	strong
10	180	3.68	2.30	weak
20	180	3.06	7.50	strong
30	180	1.23	9.00	severe
40	180	12.06	12.60	disastrous

EarthquakeAlert Python program

```
#! /usr/bin/env python3

##############################################
#  Earthquake Alert program by Sien Mitchell
##############################################

import time                             # python libraries
import serial
import signal
import EV3BT
import matplotlib.pyplot as graph
```

```
from matplotlib.widgets import Button
import matplotlib.image as mpimg

try:                                    # if python crashes do this

  EV3 = serial.Serial('COM3')                   # python bluetooth connects to EV3 port

  def onclick(event):                           # close window it quits python
    global loopContinue, EV3
    print('closing window')
    print('')
    data = EV3BT.encodeNumeric('quit',2)
    EV3.write(data)
    loopContinue = False

  def onButtonclick(event):                     # click Earthquake button it messages EV3
    global loopContinue, EV3
    print('request earthquake')
    data = EV3BT.encodeNumeric('run',5)
    EV3.write(data)

  def keyboardInterruptHandler(signal, frame):          # Ctrl C it quits python
    global loopContinue
    print('keyboard quit')
    loopContinue = False

  signal.signal(signal.SIGINT, keyboardInterruptHandler)  # check for Ctrl C quit python


  ######################
  #     Variables
  ######################
  loopContinue = True
  power = 20
  rotation = 180
  stationaryDistance = 8.6                      # distance in cm to red weight when stationary
  maxDistance = 0
  minDistance = 100
  row = 1
  startTime = time.time()                       # reference time for x line graph
  startTable = 0                                # reference time for table
  currentTime = 0                               # current time used to draw line graph
  lengthTime = 0.0                              # length of time
```

```
#######################
#     Window
#######################
graph.rcParams['toolbar'] = 'None'                  # hides toolbar
window = graph.figure('Earthquake Alert', figsize=(10,6))
graph.connect('close_event', onclick)               # calls onclick function when closing window

lineGraph=graph.subplot2grid((2, 8), (0, 0), colspan=6) #2 rows 8 col on grid, start position
tableGraph=graph.subplot2grid((2,8), (1, 0), colspan=6,frame_on=False)
barGraph=graph.subplot2grid((2, 8), (0, 7), colspan=1)
imagePlot=graph.subplot2grid((2, 8), (1, 6), colspan=2 ,frame_on=False)

#######################
#     Image
#######################
image = mpimg.imread('earthquake.png')
imagePlot.imshow(image)
imagePlot.xaxis.set_visible(False)
imagePlot.yaxis.set_visible(False)

#######################
#     Line Graph
#######################
x = []
y = []

line, = lineGraph.plot(x,y)
lineGraph.set_xlabel('Time (sec)')                  # label x axis
lineGraph.set_ylabel('Distance (cm)')               # label y axis
lineGraph.set_xlim(0,60)                            # set x axis min and max time
lineGraph.set_ylim(stationaryDistance-5,stationaryDistance+5)   # set y axis min and max distance

#######################
#     Bar Graph
#######################
objects = ['Movement']
labels = ['Weak 2','Moderate 4','Strong 6','Severe 8','Extreme 10','Disastrous 12']

bar  = barGraph.bar([1],[0], width = .02)

barGraph.set_xticks([1])
barGraph.set_xticklabels(objects)

#barGraph.set_ylabel('Alert Warning')
barGraph.set_ylim(0,14)
barGraph.set_yticks([2,4,6,8,10,12])
barGraph.set_yticklabels(labels)
```

```
######################
#     Table
######################
tableLabels = ['Power (%)','Rotation (deg)','Time Taken (sec)', 'Distance (cm)','Alert Warning']
tableText = [ ['','','','',''] ]

tableGraph.xaxis.set_visible(False)
tableGraph.yaxis.set_visible(False)

table = tableGraph.table(cellText=tableText, colLabels=tableLabels,loc='upper center', cellLoc='center')
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1,1.5)
cellHeight = table[0,0].get_height()
cellWidth = table[0,0].get_width()

######################
#     Button
######################
buttonLocation = graph.axes([0.87, 0.045, 0.09, 0.075])
button = Button(buttonLocation, 'Earthquake',  hovercolor='grey')
button.on_clicked(onButtonclick)

######################
#     Program
######################

print('connecting to EV3')
data = EV3BT.encodeNumeric('alive',0)
EV3.write(data)

while loopContinue:                  # loopContinue until false
    n = EV3.inWaiting()              # checks if EV3 has messages
   if n > 0:

      dataBT = EV3.read(n)           # reads messages

      while dataBT is not None:      # while messages in bytes is not empty
         message,value,dataBT  = EV3BT.decodeNumeric(dataBT)
         if message == "power":
            power = int(value)
            print ("power",power,"%")
         elif message == "rotation":
            rotation = int(value)
            print ("rotation",rotation,"deg")
         elif message == 'distance':
            #print('distance = ', value)
```

```
    if value > maxDistance:
        maxDistance = value              # highest distance

    if value < minDistance:
        minDistance = value              # lowest distance

    height = maxDistance - minDistance     # bar height
    if height > 0:
        bar[0].set_height(height)

        if height < 2:
            alert = 'not felt'
            bar[0].set_color('lightgreen')
            alertColour = 'lightgreen'
        elif height < 4:
            alert = 'weak'
            bar[0].set_color('green')
            alertColour = 'green'
        elif height < 6:
            alert = 'moderate'
            bar[0].set_color('lightblue')
            alertColour = 'lightblue'
        elif height < 8:
            alert = 'strong'
            bar[0].set_color('yellow')
            alertColour = 'yellow'
        elif height < 10:
            alert = 'severe'
            bar[0].set_color('orange')
            alertColour = 'orange'
        elif height < 12:
            alert = 'extreme'
            bar[0].set_color('indianred')
            alertColour = 'indianred'
        else:
            alert = 'disastrous'
            bar[0].set_color('red')
            alertColour = 'red'

    currentTime = time.time()
    lengthTime = currentTime - startTime
    xMin = lengthTime - 55               # show 55sec of data
    if (xMin < 0):                     # time cannot be less than 0sec
        xMin = 0
    xMax = xMin + 60                   # set max time 5sec ahead

    lineGraph.set_xlim(xMin,xMax)        # change x label as time moves
```

```
            y.append(value)                  # add new y distance values
            x.append(lengthTime)
            line.set_data(x,y)               # add new data to line
         elif message == 'alive' :           # check if EV3 is turned on
            print('EV3 is connected')
            data = EV3BT.encodeNumeric('alive',0)
            EV3.write(data)
         elif message == 'starting':
            print('starting earthquake')
            startTable =  time.time()
            maxDistance = 0
            minDistance = 100
         elif message == 'stopping':
            print('stopping earthquake')

            if row >= 10:
               row = 1
               print('restarting table at row 1')

            timeTaken = '{:.2f}'.format(currentTime - startTable)
            height = '{:.2f}'.format(maxDistance - minDistance)

            #['Power ','Rotation', 'Time', 'Distance', 'Alert Warning']
            table.add_cell(row,0,text=str(power), width=cellWidth, height=cellHeight, loc='center')
            table.add_cell(row,1,text=str(rotation), width=cellWidth, height=cellHeight, loc='center')
            table.add_cell(row,2,text=timeTaken, width=cellWidth, height=cellHeight, loc='center')
            table.add_cell(row,3,text=height, width=cellWidth, height=cellHeight, loc='center')
            table.add_cell(row,4,text=alert, width=cellWidth, height=cellHeight, loc='center')
            table[(row,4)].set_facecolor(alertColour)
            row = row + 1
      else:
         time.sleep(0.01)

      graph.draw()
      graph.pause(0.001)

except serial.serialutil.SerialException:
   print('')
   print('Bluetooth ERROR cannot open port')
   print('Make sure EV3 is turned on and paired')
   print('')
   print('Reset Bluetooth connection by running "resetBluetooth" on EV3')
   exit()

graph.close('all')
EV3.close()                              # close EV3 program and connection
```

How to load and run the programs

1) Press the centre button on the EV3 controls to turn it on.
2) Open Bluetooth settings on the computer to check if EV3 is paired to the computer. Pairing two devices is when the devices send permissions to connect.
3) Make sure that all the cables are in the correct ports. Put the USB cable on the computer.

Port ID	***Sensor***	***Motor***	***Hardware***
A & D		large	
B		medium	
1	ultrasonic		
4	touch		
PC			USB cable from computer to EV3 brick

4) Click on Lego Mindstorms EV3 software on the computer.
5) Open project "EarthquakeShake.ev3".
6) Click on Available Bricks icon on hardware page in the EV3 software. Make sure EV3 is connected as USB not Bluetooth.
7) Click Download and Run icon on hardware page in the EV3 software.
8) Click on Visual Studio Code software on the computer.
9) Open file "EarthquakeAlert.py".
10) Click on the green Play icon in the Visual Studio Code software.
11) Click on the "Earthquake" button in Earthquake Alert window. The line graph displays the distance moved over time. The bar graph shows the movement as it happens in the Alert Warning colours. The table shows the biggest distance, alert warning scale, power, motor rotation and the time taken to run the motors.
12) Press the up button to increase the power or down button to decrease the power on the EV3 controls. Press the left button for rotation 90 degree or right button for 180 degree. Hold down the touch sensor button to run the earthquake shake however long you want.
13) Press the centre button to brake the swinging weight after each earthquake shake.
14) Close Earthquake Alert window.
15) Press the back button to shut down EV3.

Bibliography reference

https://www.ducksters.com/science/earthquakes.php
https://www.weatherwizkids.com/weather-earthquake.htm
https://en.wikipedia.org/wiki/Modified_Mercalli_intensity_scale
https://matplotlib.org/3.1.1/api/
http://www.geekdroppings.com/2018/01/21/raspberry-pi-and-the-lego-ev3-connected-by-bluetooth/

Acknowledgement

I learnt Lego Mindstorms EV3 all last year when I did the Robocup Junior Rescue competition. But this time I wanted to try something different, so Dad taught me how to code in Python.