PhysicsKIT 4STEM

MOTION & FORCES

Lesson Plan 1 – Gravitational Acceleration

ATERMON

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission.

Project N°: 2020-1-FR01-KA201-080433

Table of Contents

1. Lesson Plan: Gravitational Acceleration 2
1.1 General information 2
1.2 Short description 2
1.3 Learning objectives 2
1.4 Links to curriculum 2
1.5 Materials required 3
1.6 Duration 3
2 Lesson Plan 3
2.1 Introduction 3
2.1.1 Theory for Gravitational Acceleration 3
2.1.1 Theory for Infrared (IR) Sensor 4
2.2 Preparation 6
2.3 Investigation 10
2.4 Conclusion 17
2.5 Follow-up exercise (optional) 17
References or Resources 18

1. Lesson Plan: Gravitational Acceleration

1.1 General information

This lesson plan is connected to Module 1: Motion & Forces of the PhysicsKIT Curriculum. It uses an infrared sensor module along with a Python program to collect data on the acceleration of an object.

1.2 Short description

In this lesson plan we will conduct a scientific experiment to measure the acceleration of an object caused by the gravitational force of the Earth. For this purpose, we will make an experimental apparatus using our PhysicsKIT and operate it by an appropriate program. Then we will collect data and analyse them to measure the acceleration of the object.

1.3 Learning objectives

The main learning objectives of this lesson plan are:

- concept and content understanding of what motion, acceleration, speed, and velocity is.
- designing and performing an experiment or scientific investigation with collection of data, analysis, and presentation of results.
- familiarizing with circuits and programs to interact with GPIO pins of Raspberry Pi.
- understanding basic structures of Python programming language (namely, use in program the syntax, purpose and function of while loop, if statements, definition of function, etc.
- understanding data visualization and processing basic statistics.

1.4 Links to curriculum

The domains, subdomains, subjects/topics that this lesson plan can be linked to are:

- Physics: motion, acceleration, speed, velocity, gravitational force and mass.
- Science: scientific method, observation, investigation, experimentation, measurements, data analysis and interpretation of results.
- Computer Science/Informatics: processing unit and peripherals, interfaces, programming language and main structures, coding.
- Technology: electronics, open-source hardware and software, sensors, digital signal, circuits, single board computers.
- Maths/Statistics: data visualisation and basic statistics.

1.5 Materials required

For this lesson plan (and for each student group) we will use the Raspberry Pi 3 Model B+ with the installed software, a keyboard and a mouse, the Python program named *acceleration.py* and the following materials from the PhysicsKIT:

- 1 x Infrared Sensor Module KY-032
- 3 x female-to-male jumper cables
- 3D-printed slotted target

1.6 Duration

The duration of this lesson plan is estimated to be about 45-60 mins, i.e., one classroom hour.

2 Lesson Plan

The lesson plan is divided in four phases, which are introduction, preparation, investigation and conclusion. As a follow-up there is also an optional exercise at the end.

2.1 Introduction

2.1.1 Theory for Gravitational Acceleration

Understanding gravitational acceleration requires to firstly explain what gravity is. Gravity is a natural force that we experience in everyday life. Gravity pulls objects towards each other, it keeps the Earth revolving around the sun, and it pulls objects and beings towards the ground and in particular the center of the planet.

Gravitational acceleration is the acceleration of an object in free fall within a vacuum. This is the steady gain in speed caused by the gravity force. In vacuum, all bodies accelerate at the same rate, regardless of the mass or composition of a body/object.

On Earth the gravitational acceleration depends on altitude, latitude and longitude, causing it to range from 9.764 m/s^2 to 9.834 m/s^2. As a convenience, the gravitational acceleration is defined as 9.80665 m/s^2.

Newton's law of gravitation states that a gravitational force between two masses is equal in magnitude for each mass, and is aligned to draw the two masses toward each other:

$$F = \mathrm{G}\frac{m_1 m_2}{\mathrm{r}^2},$$

where m_1 and m_2 are any two masses, G is the gravitational constant, and r is the distance between the two masses.

The attraction force *F* onto a mass *m* can be expressed as:

$$F = mg$$

The *g* is the frictionless, free-fall acceleration sustained by the mass *m* under the attraction of the gravitational source.

2.1.1 Theory for Infrared (IR) Sensor

An Infrared Sensor or IR Sensor is used in many electronics and DIY projects. It is often used as an Obstacle Detection Module or a Proximity Sensor. It can also be used in robotics projects or even as digital tachometers or in alarm systems.

An IR sensor emits and receives Infrared radiation. A real-life example of IR sensor implementation is smartphones. There is an IR sensor next to the earpiece and every time you talk on the phone, the sensor detects the distance between the screen and your ear and then turns off the screen until you finish with the call.

An IR sensor consists of three parts:

- An IR LED used as an IR Emitter.
- A Photo Diode used as IR Receiver.
- A Comparator IC unit with the necessary components to control the sensor.

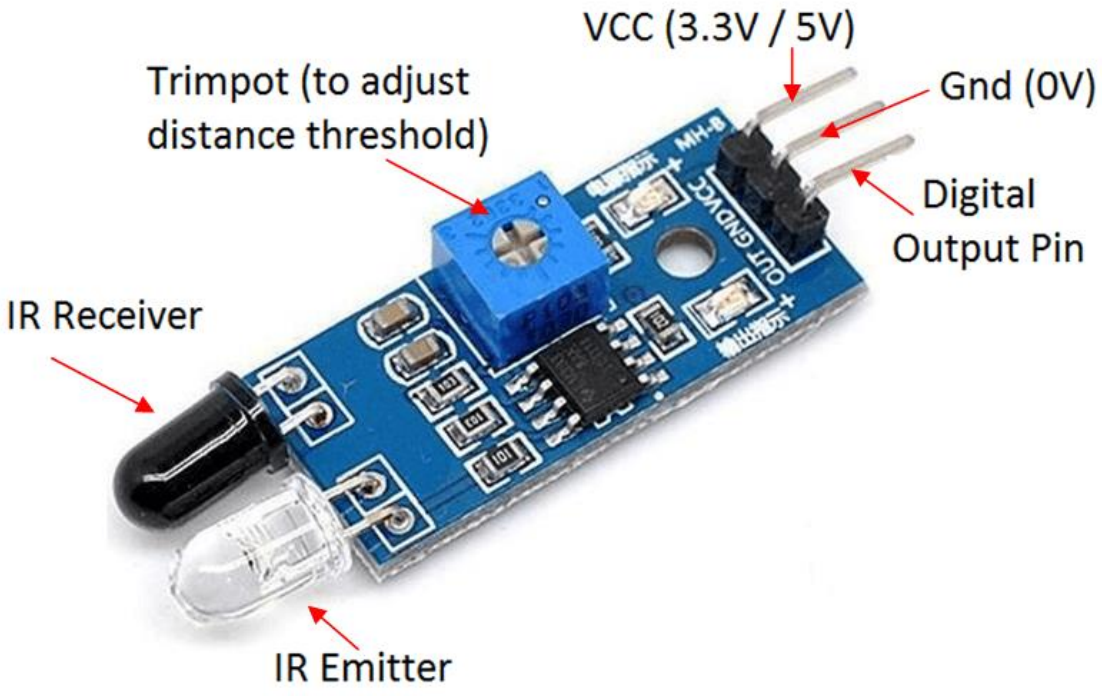

FIGURE 1 EXAMPLE OF AN IR SENSOR

Source: Solarduino (https://solarduino.com/infrared-ir-sensor-module-with-arduino/)

The IR sensor module comes with three pins for connectivity with other devices that support the I2C protocol, such as the Raspberry Pi 3. There is one pin for power (VCC 3.3/5 V), one for ground (GND) and one for data transmission (Digital Output).

It also has an attached potentiometer (trimpot) for adjusting the distance threshold.

There are two ways of implementing an IR sensor based on the application requirements.

1. IR Receiver and IR Emitter are placed side-by-side: the IR Emitter continuously emits infrared light and if there is any obstacle/object in front of the sensor, the infrared light hits the object and bounces back. The reflected signal is captured by the IR Receiver and the control circuit will reflect a Logic HIGH on its output.
2. IR Receiver and IR Emitter are placed facing each other: the infrared light from the IR Emitter always falls on the Receiver and if there is an object in between the Emitter and the Receiver, then there will be an obstruction to the infrared light and the control circuit will detect this and produces appropriate output.

The circuit board of an IR sensor includes the following electronic components (Figure 2):

- IR LED
- Photo Diode
- 150Ω Resistor
- 10 KΩ Resistor
- 10 KΩ Potentiometer
- LM358 control chip
- LED
- 1 KΩ Resistor

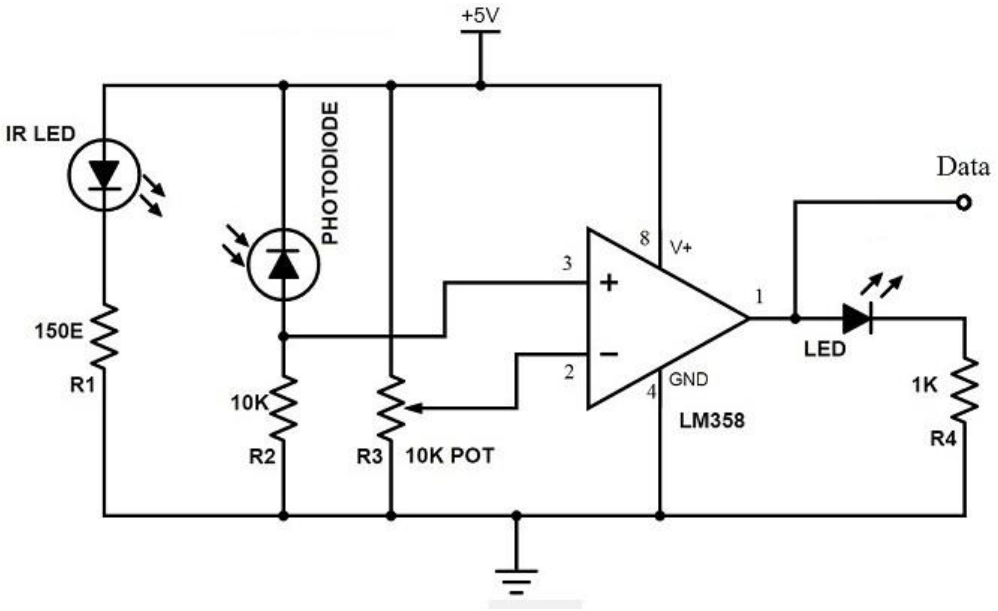

FIGURE 2 CIRCUIT SCHEMATIC OF AN IR SENSOR

Some applications using IR sensors are the following:

- Proximity sensor
- Obstacle detection module
- Contactless tachometer
- Line follower robot
- Car reverse sensor
- Mobile proximity sensor

2.2 Preparation

First thing we need to do is to make a circuit and connect our sensor to the GPIO pins of our Raspberry Pi. Before we proceed, we turn off our Raspberry Pi and unplug it. For our circuit, we will need the IR sensor and three female-to-male jumper wires. The complete circuit is in the schematic diagram (Figure 3) that follows:

The IR sensor KY-032 comes with three pins: power (VCC), ground (GND), and output (OUT). Connect the following pins using the jumper wires:

- VCC to Pin 2 (5V)
- GND to Pin 6 (GND)
- OUT to Pin 7 (GPIO4)

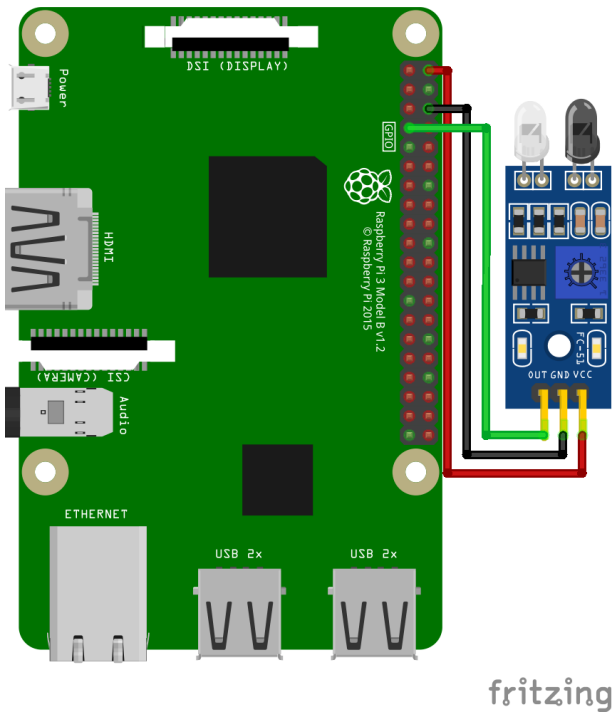

FIGURE 3 SCHEMATIC DIAGRAM OF CIRCUIT WITH IR SENSOR CONNECTED TO GPIO PINS

Source: PhysicsKIT4STEM project

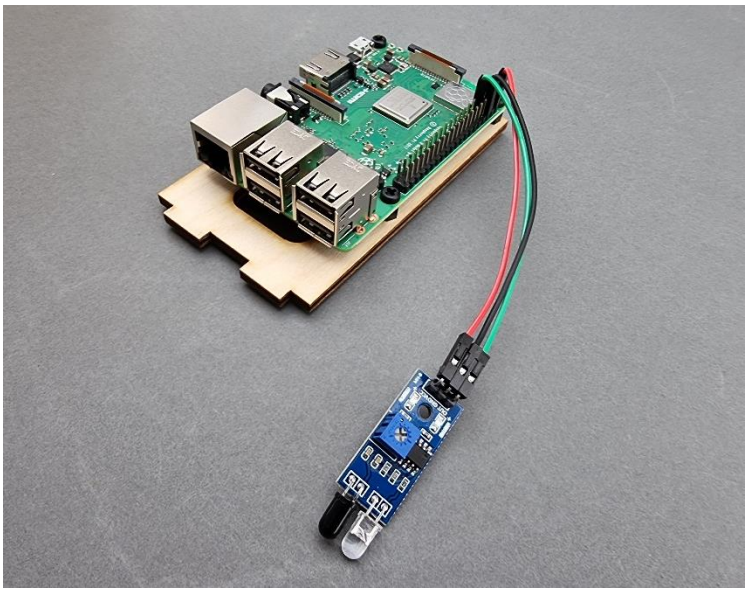

FIGURE 4 RASPBERRY PI CONNECTED WITH IR SENSOR

Source: PhysicsKIT4STEM project

Before we start our experiment, it is important to understand the program that we will run. Various comments have been inserted into the program for that purpose. The whole program is presented below.

```
### PROGRAM START ###

#IMPORT LIBRARIES
import RPi.GPIO as GPIO
import time
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as sc

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

#IR SENSOR SETUP
IR_PIN = 4 # GPIO number of to which the sensor is connected
GPIO.setup(IR_PIN, GPIO.IN)

#VARIABLES TO BE PLOTTED
y = [0.000] # Contains the values of the different heights starting from 0
yTemp = [] # Contains only the input height values
t = [] # Contains the recorded time for each height
y2 = [] # Contains the y-axis values for the 2y/t plot
t2 = [] # Contains the time values for the 2y/t plot
print("Acceleration Experiment")

while True:
    #TAKE INPUT FROM THE USER
    print("How many interruptions will the falling object have?")
    interruptionCount = int(input("Interruptions: "))
```

```
    print()
    print("Enter the height of each interruption (in meters), measured from bottom
to top. The height of interruption number 1 has been taken as 0.\n")
    print("INPUT HEIGHTS")
    print("h1 = 0.0")
    while interruptionCount > len(y):
        try:
            height = float(input("h" + str(len(y)+1) + " = "))
            y.append(height)
            yTemp.append(height)
        except:
            print("Your input wasn't a number. Enter value again. ")
    print()

    while True:
        print("Drop the object when you are ready.")
        #MEASURE THE TIME OF EACH INTERRUPTION
        trigger = 0 # Variable for triggering the sensor.
        sleepTime = (y[1] - y[0])/10 # Adjustable parameter. Time in seconds for
the sensor to "sleep" after a detection. Avoids continuous measurements. The
suggested value is (y1 - y0)/10.

        # Record values
        while len(t) < len(y):
            # Sensor detects an obstruction
            if GPIO.input(IR_PIN) == False and trigger == 0:
                t.append(time.time())
                trigger = 1
                time.sleep(sleepTime)
            # Sensor detects no obstruction
            elif GPIO.input(IR_PIN) == True and trigger == 1:
                trigger = 0
                time.sleep(sleepTime)

            # Rounds and makes a correction to the time values so that they start
from 0
        t0 = t[0]
        for i in range(len(t)):
            ti = t[i]
            t[i] = ti - t0
            if i != 0:
                 t2.append(ti - t0)
        print()

        y2 = 2*np.array(yTemp)/np.array(t2) # y2 = 2y/t

        #SUMMARY OF RESULTS
        # Print a formatted table of the results
        print("DATA SUMMARY")
        print("-----------------------")
        print("-Time (s)-|-Height (m)-")
        print("----------|------------")
        for i in range(len(t)):
            print('{:7.3f}'.format(t[i]) + " | " + '{:7.3f}'.format(y[i]))

        # Define a linear function for the linear fit
        def lin(x,m,b):
            return m*x+2*b
        fit = sc.curve_fit(lin,t2,y2)
```

```
        # Fit parameters
        a = round(fit[0][0],2)
        b = round(fit[0][1],2)

        # Ucertainties for fit parameters
        aUncertainty = round(np.sqrt(fit[1].diagonal()[0]),2)
        bUncertainty = round(np.sqrt(fit[1].diagonal()[1]),2)

        # Format the numbers for display
        A = str(a)
        B = str(b)
        Aun = str(aUncertainty)
        Bun = str(bUncertainty)

        # Create the arrays for the linear fit
        t2_fit = np.linspace(t2[0],t2[-1],100)
        y2_fit = lin(t2_fit,a,b)

        # Print results
        print()
        print("FIT RESULTS")
        print("g = "+str(A)+" ± "+str(Aun) )
        print("v0 = "+str(B)+" ± "+str(Bun))
        print()

        #PLOT RESULTS
        # Plot for measured data
        plt.subplot(1,2,1)
        plt.plot(t, y, 'ok', label="Measured heights")
        plt.title('Height vs. Time')
        plt.xlabel('t (s)')
        plt.ylabel('y (m)')
        plt.legend(loc="upper left")

        # Plot for the lineal fit
        plt.subplot(1,2,2)
        plt.plot(t2,y2,'ok',label="2y/t")
        plt.plot(t2_fit,y2_fit,'--b',label="Linear                                    fit:
$gt+2v_0$\n$g="+A+"\pm"+Aun+"$\n$v_0="+B+"\pm"+Bun+"$\n")
        plt.legend(loc='upper left')
        plt.title("Linear Fit")
        plt.xlabel("t (s)")
        plt.ylabel("2y/t (m/s)")

        plt.show()

        # Repeat experiment with same configuration?
        repeatWithSameConfiguration = input("Do you want to try again with the
same configuration? [y/n] ")
        if repeatWithSameConfiguration == 'n' or repeatWithSameConfiguration ==
'N':
            break
        elif repeatWithSameConfiguration == 'y' or repeatWithSameConfiguration ==
'Y':
            t = []
            t2 = []
            print()
            continue
```

```
        else:
            print("Enter \"y\" or \"n\".")

    # Repeat experiment with a different configuration?
    repeatAll = input("Do you to try again with a different configuration? [y/n]
")
    if repeatAll == 'n' or repeatAll == 'N':
        break
    elif repeatAll == 'y' or repeatAll == 'Y':
        errors = []
        y = [0.0]
        t = []
        yTemp = []
        y2 = []
        t2 = []
        print()
        continue
    else:
        print("Enter \"y\" or \"n\".")

### PROGRAM END ###
```

TROUBLESHOOTING

There might be two errors appearing when running the program:

1. **`matplotlib`** module error
   Open a terminal window and run the following command to install matplotlib:
   `sudo apt-get install python3-matplotlib`

2. **`scipy`** module error
   Open a terminal window and run the following command to install scipy:
   `sudo apt-get install python3-scipy`

In case other module errors appear, it is always helpful to search on the Python documentation for Raspberry Pi.

2.3 Investigation

Now that we have familiarized ourselves with our experimental apparatus, we can start experimenting. Once again make sure that the sensor is correctly connected to the GPIO pins of the Raspberry Pi and open the *acceleration.py* program using Thonny Python. Figure 5 shows how to setup the equipment.

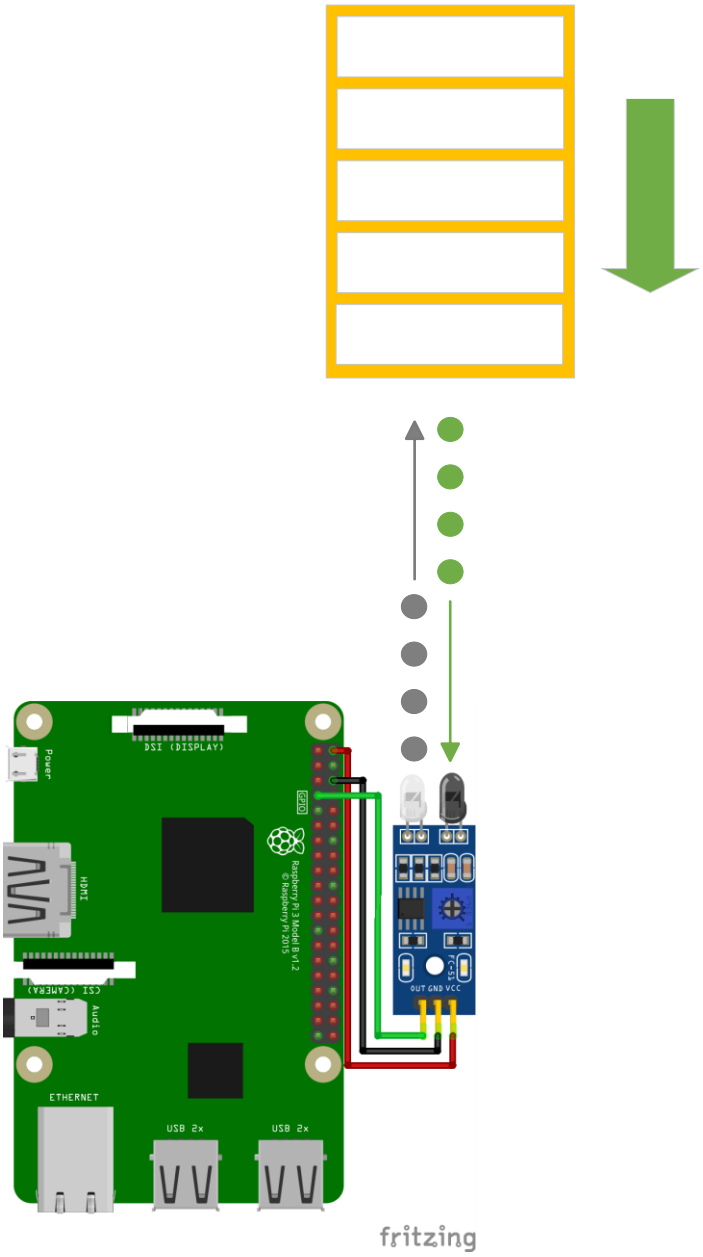

FIGURE 5 EXPERIMENT DIAGRAM

Place the 3D printed target in a safe, but close distance above the sensor, so when it falls, it does not hit the sensor.

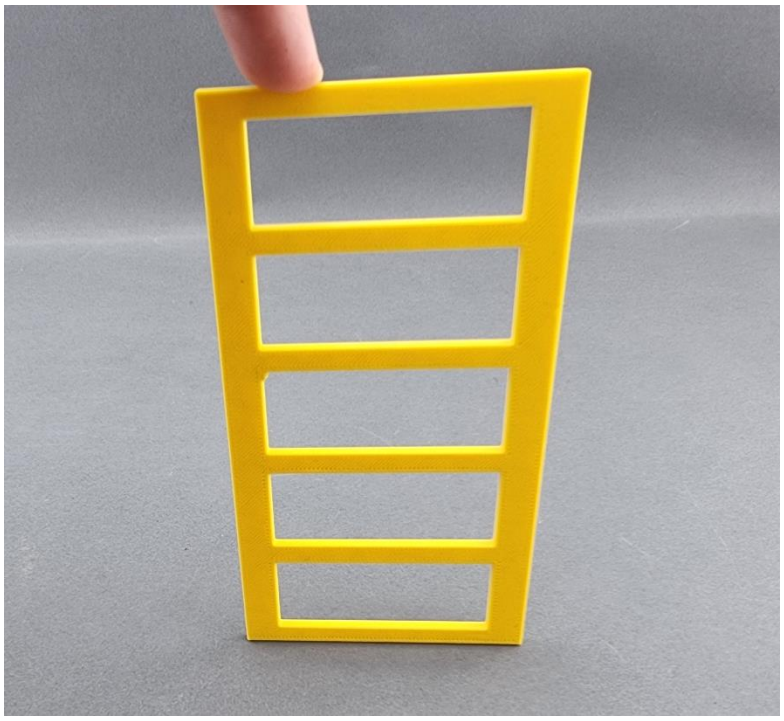

FIGURE 6 3D PRINTED SLOTTED TARGET

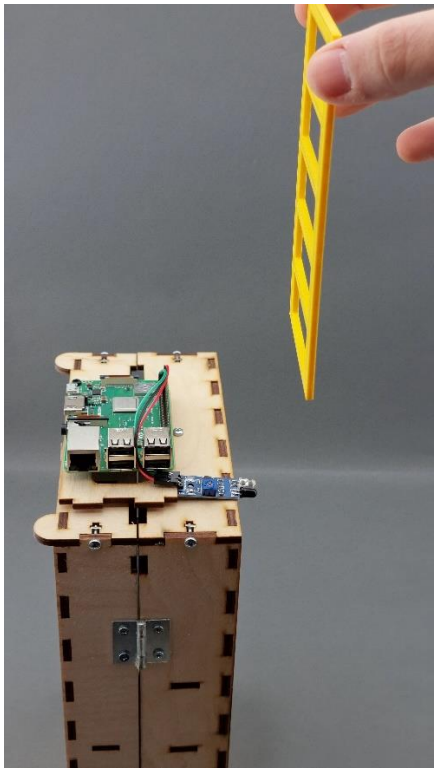

FIGURE 7 PLACING THE 3D PRINTED TARGET ABOVE THE SENSOR IN A CLOSE DISTANCE

Back to our program in Thonny, click the Play button or hit F5 on the keyboard to run it.

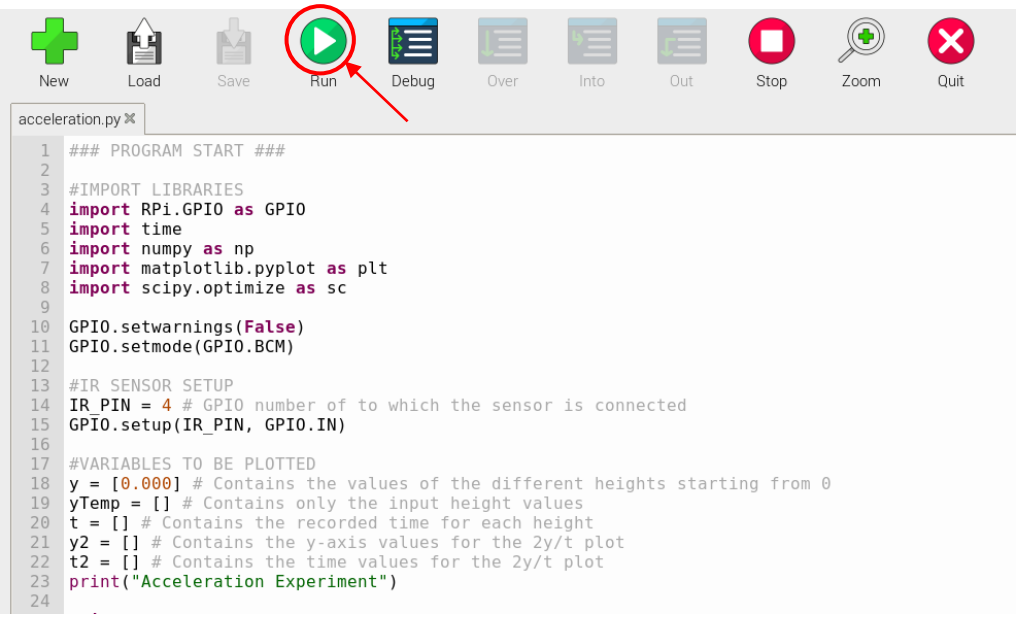

FIGURE 8 SCREENSHOT FROM THONNY PYTHON

The program will first ask input for the number of interruptions on the target. In our case we have **6 interruptions**. Write 6 and hit the Enter button. The program asks us to input the height of each interruption (the 1st is set to zero by default).

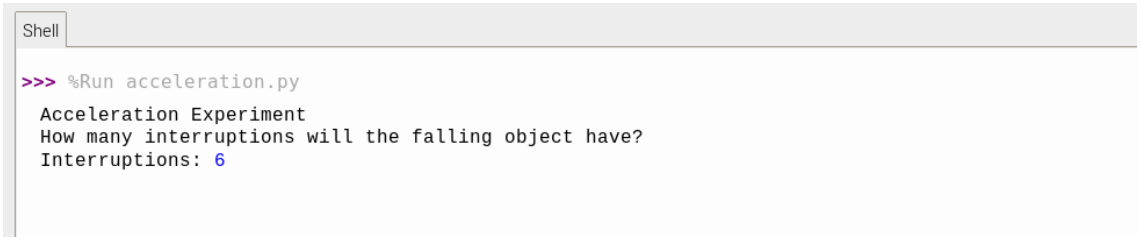

FIGURE 9 NUMBER OF INTERRUPTION IN THONNY

You can measure each interruption from the bottom of the target to the top of each one of the empty spaces on the target. The input numbers should be converted to meters.

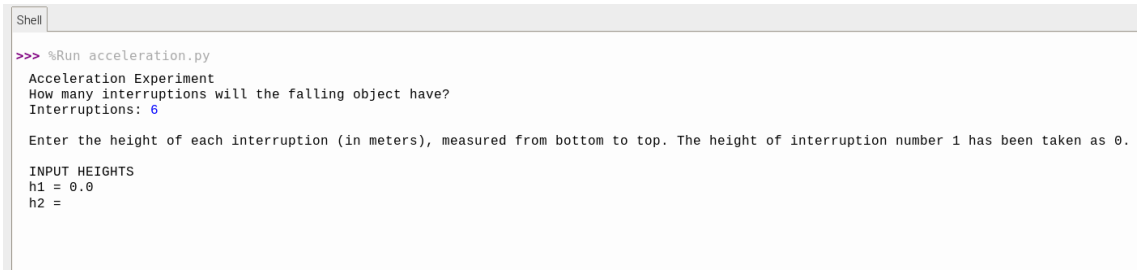

FIGURE 10 INPUT HEIGHTS

```
Shell
>>> %Run acceleration.py
  Acceleration Experiment
  How many interruptions will the falling object have?
  Interruptions: 6

  Enter the height of each interruption (in meters), measured from bottom to top. The height of interruption number 1 has been taken as 0.

  INPUT HEIGHTS
  h1 = 0.0
  h2 = 0.028
  h3 = 0.056
  h4 = 0.086
  h5 = 0.115
  h6 = 0.143

  Drop the object when you are ready.
```

FIGURE 11 MANUALLY TYPE EACH HEIGHT, THEN HIT ENTER, THEN DROP THE OBJECT IN FRONT OF SENSOR

For convenience, the following numbers should be used as input:

TABLE 1 INPUT VALUES FOR INTERRUPTION HEIGHT

h1 = 0.0
h2 = 0.028
h3 = 0.056
h4 = 0.086
h5 = 0.115
h6 = 0.143

After inputting the numbers, the program will proceed to measure the time for each interruption as the target falls and interrupts the IR sensor. To achieve best results, the target should fall as close to the sensor as possible. Follow the following steps:

Place the sensor on the side of your desk (Figure 12):

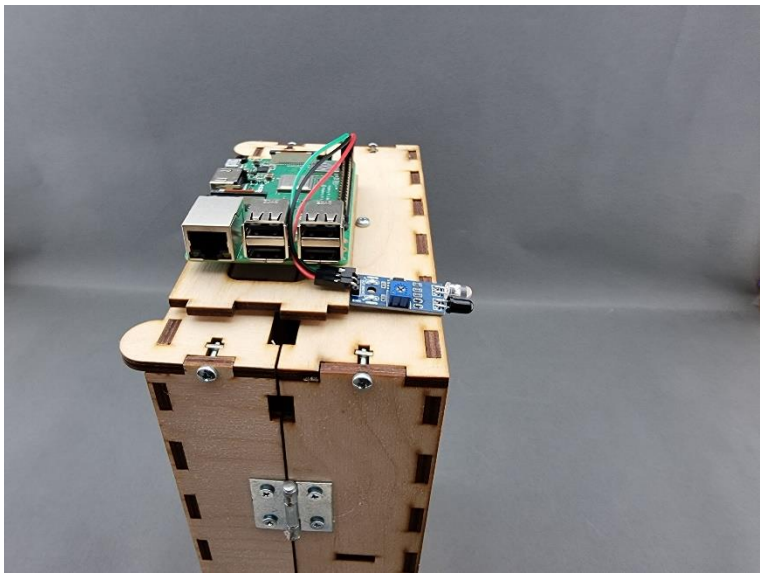

FIGURE 12 PLACING THE IR SENSOR AT THE EDGE OF A SURFACE

Hold the 3D printed slotted object above the sensor (Figure 13):

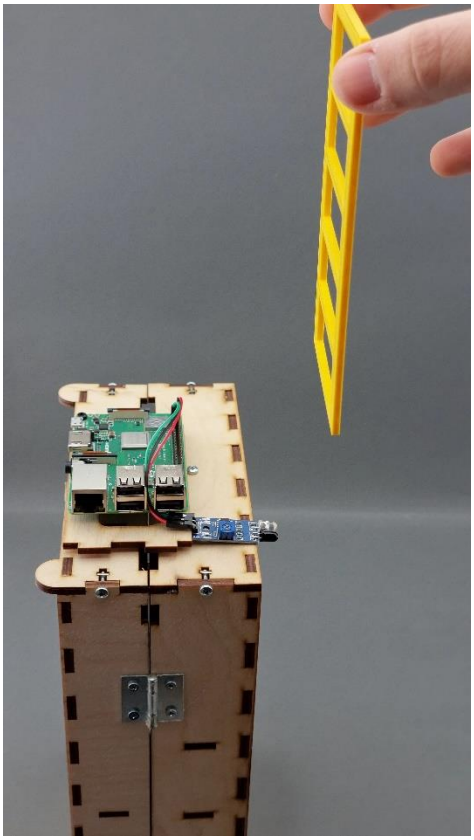

FIGURE 13 PLACING THE 3D PRINTED OBJECT ABOVE THE IR SENSOR IN A CLOSE DISTANCE

Drop the object as close to the sensor as possible (Figure 14):

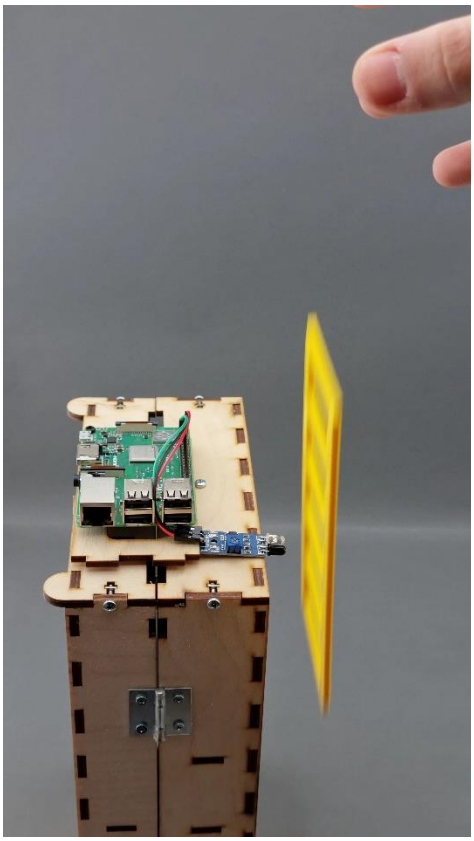

FIGURE 14 DROPPING THE OBJECT VERTICALLY IN FRONT OF THE IR SENSOR

See the program collecting the data and displaying a plot of each data point (position vs time graph). A second plot appears which shows the liner fit of the following equation:

$$\frac{2y}{t} = gt + 2v_0$$

Which derives from this equation:

$$y(t) = y_0 + v_{0y}t + \frac{1}{2}gt^2$$

The constant of gravitational acceleration can be determined from the slope in the plot, as well an estimation of the initial velocity of the object.

The program will show the calculated acceleration with a percentage difference of the expected value. After that, the program will ask the user to repeat the experiment with the same object or a different one.

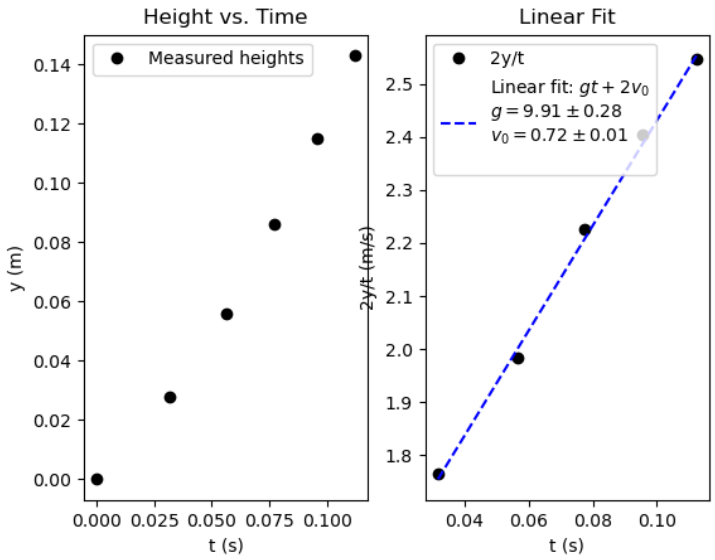

FIGURE 15 EXAMPLE OF CALCULATED ACCELERATION AS SHOWN IN THONNY

What are the results after 100 consecutive runs with the same object? Is the acceleration figure from the program close to the expected one? What happens if we use a different object?

It is suggested to setup a table and collect all the acceleration variations after 100 runs and then analyse the average calculated acceleration. Is it close to the real gravitational acceleration 9.80665 m/s^2?

2.4 Conclusion

We have succeeded measuring the acceleration of an object due to the gravitational force of the Earth. In this phase we recapitulate what we did and how, which were the main steps, discuss any difficulties experienced.

2.5 Follow-up exercise (optional)

This procedure can be performed for target of various shapes and different separations between interruptions. I can also serve as a good exercise to introduce students to probability distribution and statistical analysis. The Python program included in this lesson plan, contains comments for each part of the code, so the teacher or the student can make any modifications to it to serve their needs and experiments.

References or Resources

Here are some useful references and additional resources related to this lesson plan.

- Phys. Teach. 59, 134 (2021); https://doi.org/10.1119/10.0003472. Published Online: 11 February 2021
- Physical Computing with Python, Raspberry Pi Foundation. Retrieved online: https://projects.raspberrypi.org/en/projects/physical-computing/1
- Raspberry Pi Pinout. Retrieved online: https://pinout.xyz/
- Interfacing IR Sensor with Raspberry Pi (Proximity Sensor – Obstacle Detector) (2018), Electronics Hub. Retrieved online: https://www.electronicshub.org/interfacing-ir-sensor-with-raspberry-pi/
- Infrared (IR) Sensor Module with Arduino (2020), Solarduino. Retrieved online: https://solarduino.com/infrared-ir-sensor-module-with-arduino/
- Gravitational acceleration (2022), Wikipedia. Retrieved online: https://en.wikipedia.org/wiki/Gravitational_acceleration
- The acceleration of Gravity, The Physics Classroom, 1-D Kinematics - Lesson 5 - Free Fall and the Acceleration of Gravity. Retrieved online: https://www.physicsclassroom.com/class/1DKin/Lesson-5/Acceleration-of-Gravity