```
  //REMEMBER: when doing operations with 'long' variables, always put 'L' after
    constants otherwise you will have bugs!

  if (dist > 500L || dist < 2L)
  {
    Serial.println("Invalid range!");
  }
  else
  {
    Serial.print(dist);
    Serial.println(" cm");
  }
  delay(1000);
}
```

sourcecodes/Ultrasound/ultrasound.ino

Task 2.3 *Create a car parking sensor by combining your existing circuit with the circuit from experiment 2 (piezo speaker). Your code must do the following: If the distance is under 100cm it should beep periodically, at a faster rate as the distance gets smaller. If the distance is under 20cm it should beep continuously without interruption.*

2.3 Get Started with Python

Python is a powerful tool for data science. It would be nature to use it to accept data from the sensors and send them to the database. We will use python3 in the lab.

2.3.1 Basic of Python

Download and install Python at `https://www.python.org/downloads/`. In this lab, we will use two modules : numpy and matplotlib. Matplotlib is a low level graph plotting library in python that serves as a visualization utility. NumPy provides an array object that is up to 50x faster than traditional Python lists, the array object in NumPy is called ndarray which is frequently used by Matplotlib.

Open the command windows and install the following modules with the following commands:
python -m pip install pyserial (for serial communciations)
python -m pip install numpy (for data analysis)
python -m pip install matplotlib (for graphical display)

Matplotlib plots your data on instances of class Figure. Each Figure could contain one or more Axes (Axes is also a class). The simplest way of creating them is using pyplot.subplots, a function which returns a instance of Figure and a instance or an array of Axes. We can then use Axes.plot to draw some data on the Axes:

```python
import matplotlib.pyplot as plt
import numpy as np

fig, ax = plt.subplots()  # Create a figure containing a single axes

# create two ndarray data type in numpy:
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

# call functions in matplotlib to create and draw graph:
ax.plot(xpoints, ypoints)
plt.show()
```

Listing 2.2: Plot a line by python)

Task 2.4 *Try Listing 2.3, 2.4 and understand them. Hint: You can find Basic Python Programming tutorial (numpy, etc) at website:* https://www.w3schools.com/python/default.asp *and graphical display tutorial at* https://matplotlib.org/tutorials/index.html*.*

```python
import matplotlib.pyplot as plt


fig = plt.figure()  # an empty figure with no Axes
fig,ax = plt.subplots()  # a figure with a single Axes
fig,axs = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
ax.plot([1,2,3,4],[4,3,2,1])
axs[0][0].plot([1,2,3,4],[1,2,3,4])
axs[0][1].plot([1,2,3,4],[1,6,4,2])
plt.show()
```

Listing 2.3: Plot three figures with different axes

```python
import matplotlib.pyplot as plt
import numpy as np

#x is a array with 100 evenly spaced numbers from 0 to 2
x = np.linspace(0, 2, 100)

# Note that even in the OO-style, we use `.pyplot.figure` to create the Figure.
fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-label to the axes.
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend()  # Add a legend.
plt.show()
```

Listing 2.4: Plot figures with different labels

2.3.2 Realtime Data Visualization by Matplotlib

In this lab, we would get to know how to plot data in realtime, here is an example:

```python
import matplotlib.pyplot as plt
import numpy as np

fig, ax = plt.subplots()

x = np.linspace(0, 100, 100)
y = np.zeros(100); z = np.zeros(100) # y and z are two arrays with 100 zero

for i in range(100):
    y[i] = np.random.rand(); z[i] = np.random.rand() # generate random number
    ax.plot(x[:i],y[:i],x[:i],z[:i]) # plot datas from 0 to i
    fig.show()
    plt.pause(0.001) # create a 0.001 second delay
```

Listing 2.5: Plot a dynamic data by python

We notice that the whole graph was being compressed when data were updated. This is not convenient to observe the graph. We can use ax.set_xlim and ax.set_ylim to set x and y-axis view limits:

```python
import matplotlib.pyplot as plt
import numpy as np
```

```
fig, ax = plt.subplots(figsize=(20, 5))

x = np.linspace(0, 100, 100)
y = np.zeros(100); z = np.zeros(100) # y and z are two arrays with 100 zero
ax.set_xlim(0,100); ax.set_ylim(0,1) # set x and y-axis view limits

for i in range(100):
    y[i] = np.random.rand(); z[i] = np.random.rand() # generate random number
    ax.plot(x[:i],y[:i],x[:i],z[:i]) # plot datas from 0 to i
    fig.show()
    plt.pause(0.001) # create a 0.001 second delay
```

Listing 2.6: Plot a dynamic data by python

Task 2.5 *Try to write a code to continuously generate random data sequence and to display them in realtime. hint: you could set view limits multiple times*

2.4 Connect Arduino to Python

2.4.1 Transmit Characters from Python to Arduino

Transmit a character to Arduino. Please upload the List 2.7 to Arduino. You need to connect an LED on Arduino pin 9.

```
short LED = 9;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}

void loop() {
  char state;
  // put your main code here, to run repeatedly:
  if (Serial.available()){
    state = Serial.read();

      if (state == '1'){
         digitalWrite(LED, HIGH);
          delay(1000);
      }
      else{
         digitalWrite(LED, LOW);
         delay(1000);
      }
      }
     Serial.println(state);
    }
```

Listing 2.7: Receive a Character by Arduino

Save the python code in the List 2.8 to a file named "serialtest.py".

```
import serial
import time

ser = serial.Serial('COM12', 9600)

while True:
  ser.write(b'1')
  time.sleep(1)
```

Listing 2.8: Transmit a Character from Python)

Task 2.6 *Understand the code and run the python code in the command window. Explain the observation. In order to run the python code, you need to type in the command "python serialtest.py" in the command window. Note that you should already be in the directory where you save the python code.*

Now we can try to transmit a string (a sequence of characters) to the Arduino. Try the List 2.7 and the List 2.8.

```
#include <string.h>
short LED = 9;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}

void loop() {
  String state;
  // put your main code here, to run repeatedly:
  if (Serial.available()){
    state = Serial.readStringUntil('\n');

      if (state == "on"){
         digitalWrite(LED, HIGH);
          delay(1000);
      }
      else{
         digitalWrite(LED, LOW);
         delay(1000);
      }
      }
     Serial.println(state);
    }
```

Listing 2.9: Receive a String by Arduino

```
import serial
import time

ser = serial.Serial('COM3', 9600)

while True:
  ser.write(b'on\n')
  time.sleep(1)
```

Listing 2.10: Transmit a String from Python

Task 2.7 *Understand the code and explain the observation.*

2.4.2 Receiving Multiple Data by Python

```
int a = 11;
int b = 12;
int c = 13;
```

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.print(a);
  Serial.print(',');
  Serial.print(b);
  Serial.print(',');
  Serial.println(c);
}
```

Listing 2.11: Transmit a String of Multiple Integers to Python

```
import serial
import time

ser = serial.Serial('COM18', 9600)
receivestring = ser.readline()   # ignore the first read in

while True:
  receivestring = ser.readline().decode("utf-8").strip()  # read one line and
    convert bytes to a string
  receivelists = receivestring.split(',')
  print(receivelists)
  time.sleep(1)
```

Listing 2.12: Receive a String of Multiple Integers from Python)

2.4.3 Receiving Realtime Data From Arduino

Here is an example of receiving data from Arduino and display them in realtime, please successfully run them.

```
/* ReadAnalogVoltage
Reads an analog input on pin 0 , converts it to voltage ,
and prints the result to the serial monitor .
*/
// the setup routine runs once when you press reset :
void setup () {
// initialize serial communication at 9600 bits per second :
Serial.begin (9600) ;
}
// the loop routine runs over and over again forever :
void loop () {
// read the input on analog pin 0:
int sensorValue = analogRead (A0) ;
// Convert the analog reading ( from 0 - 1023) to a voltage (0 - 5V):
float voltage = sensorValue * (5.0 / 1023.0) ;
// print out the value you read :
Serial . println ( voltage ) ;
delay(10);
}
```

Listing 2.13: Arduino: Transmit potentialmeter date to python

```
import serial
import matplotlib.pyplot as plt
import numpy as np

ser = serial.Serial('COM3', 9600)
receivestring = ser.readline()
```

```
fig, ax = plt.subplots()
ax.set_ylim(0, 5.1)
ax.set_xlim(0, 1)
x = np.linspace(0,1,100)
y = np.zeros(100)
i = 0

while True :
    receivedata = float(ser.readline().decode("utf -8").strip())
    print(receivedata)
    if(i<100):
        y[int(i)] = receivedata
        ax.plot(x[:int(i)],y[:int(i)])
    else:
        x = np.linspace(i/100,i/100+1,100)
        for j in range(99):
            y[j] = y[j+1]
        y[99] = receivedata
        ax.set_xlim(i/100,i/100+1)
        ax.plot(x,y)
    fig.show()
    i += 1
    plt.pause(0.01)
```
Listing 2.14: Python: Receiving data from Arduino and display them in realtime)

Chapter 3

Lab 3

In this lab, we are going to become familiar with the Pitch, Yaw, Roll System and how to use them to rotate a object in 3D systems.

3.1 Pitch, Yaw, Roll System

Pitch, yaw and roll are three dimensional movement of an object. We can use the movement of an aeroplane to demonstrate it as shown in Fig. 3.1.

Let $\theta \in [-\pi/2, \pi/2], \psi \in (-\pi, \pi], \phi \in (-\pi, \pi]$ denote the angles which the object rotates along the axis corresponding to pitch, yaw and roll.

Note: Figure 3.1 is from wikipedia website https://commons.wikimedia.org/wiki/File:Yaw_Axis_Corrected.svg.

3.1.1 3D Rotation in Python

In python, try the following code to see how to rotate a 3d system by pitch, yaw and roll.

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(projection='3d')
```

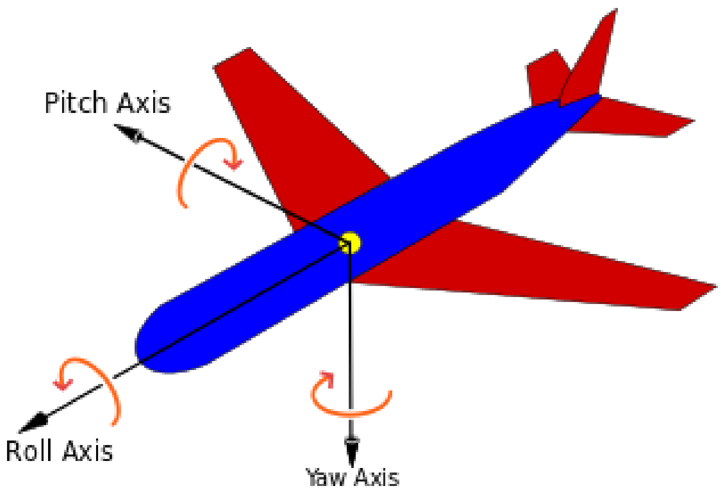


Figure 3.1: Pitch, Yaw and Roll of an aeroplane.

```
# Set the axis labels
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')

# Rotate the axes and update
for pitch in range(0, 90):
    yaw = roll = 0

    # Update the axis view and title
    ax.view_init(pitch, yaw, roll)
    plt.title('Pitch: %d , Yaw: %d , Roll: %d ' % (pitch, yaw, roll))

    plt.draw()
    plt.pause(.001)
```

Listing 3.1: Rotation in 3D

Run the code in Listing 3.1. You will see that the pitch, yaw and roll are supposed to be in the range of $(0, 360)$ instead of $(-\pi, \pi)$.

Task 3.1 *In the code, Be familiar with the commands and explain the code.*

3.2 Accelerometer for finding Orientation

3.2.1 Introduction of The Accelerometer

An accelerometer (Fig. 3.2(a)) measures the g-force acceleration, i.e. acceleration in relation to a state of free-fall. It is the acceleration which is felt by the body as weight, as opposed to a state of free-fall which is felt as "weightlessness." At rest on a table, the g-force on an accelerometer is caused by the normal force exerted on it by the surface of the table which keeps it from falling freely — hence, the g-force shown by the accelerometer is pointed in the opposite direction of gravity (up instead of down). If there is acceleration in direction orthogonal to the gravity direction, e.g. when a car or roller-coaster accelerates fast and you feel pressed against the seat, it is also an example of a g-force. In the example, it is pointed in the direction of the acceleration of the car.

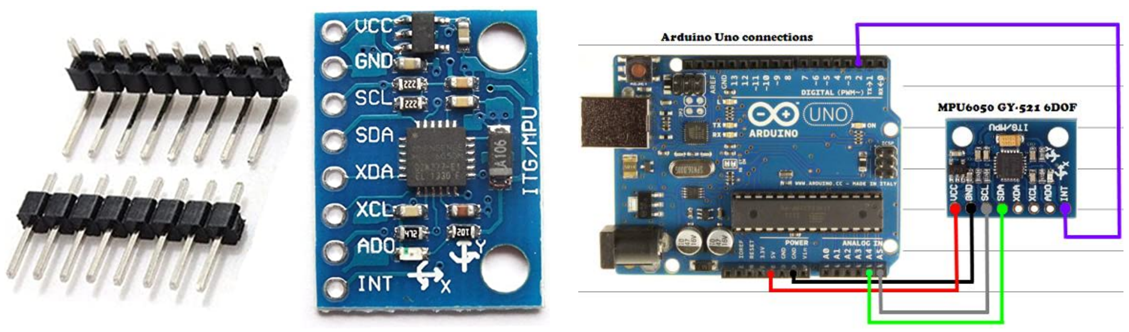
Most modern accelerometers are micro-electro-mechanical systems (MEMS). This is a miniature accelerometer from Freescale with a 14-bit ADC resolution. You can set the acceleration range to ±2g, ±4g, or ±8g. We are using a break-out board for this sensor (Fig. 3.2(a)) which facilitates its use in Arduino projects. To use the break-out board with a bread-board the supplied headers are soldered on the board – this has been done for you.

3.2.2 Download the Libraries

Check the libraries subfolder of your Arduino sketches folder: Under Windows, it will likely be called "My Documents \Arduino \libraries". For Mac users, it will likely be called "Documents/Arduino/libraries". On Linux, it will be the "libraries" folder in your sketchbook. If the libraries folder does not contain subfolders called Adafruit_Adafruit_MPU6050-2.2.4 and Adafruit_Sensor, proceed as follows:

Please download library for MPU6050 from the following link

- Download the file `Adafruit_MPU6050-2.2.0.zip` to a folder of your choice. In the Arduino IDE, go to the menu item Sketch $\longrightarrow$ Include Library $\longrightarrow$ Add .ZIP Library, and in the file-dialog, select the zip file.



(a) The break-out board for MPU6050 and the supplied headers.

(b) The circuit for running the demo code.

Figure 3.2: The MPU6050. Note that the XYZ axes directions are clearly marked.

- Download the file `Adafruit_Sensor-master.zip` to a folder of your choice. In the Arduino IDE, go to the menu item Sketch ⟶ Include Library ⟶ Add .ZIP Library, and in the file-dialog, select the zip file.
- Restart the IDE.

3.2.3 Make the Circuit

Now, insert the headers on which the sensor is mounted to the breadboard. Hold the sensor break-out board from the sides, align to the holes properly and press from the headers side: **Warning:** Do not use excessive force.
The break-out board has the following marked pins, which you should connect as instructed below.
Hint: Use longer jumpers so that you will be able to tilt the breadboard easily later.

Vin. The chip uses 3 VDC, hence, the breakout has a regulator on board which takes in 3-5VDC and safely converts it to 3 VDC. Connect this pin to the 5V pin of your Arduino.

GND. Connect it to the GND of the Arduino.

3Vo. This is the 3.3V output from the regulator. We will not use it.

I2, I1. These are for interrupts: We will not need them.

SCL. I2C clock pin, connect to the I2C clock line of the Arduino. On Uno, this is **A5**. On some variants, a separate notch marked "SCL" is also provided.

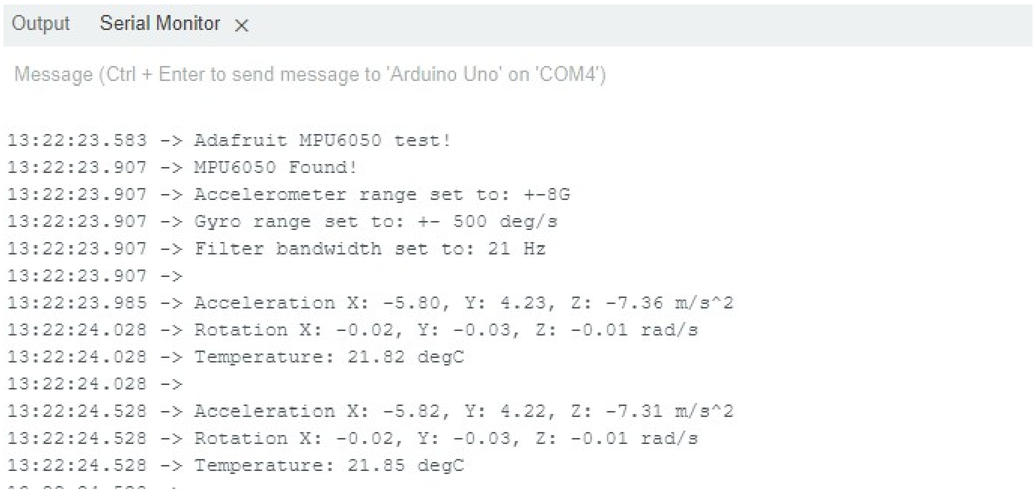
SDA. I2C data pin, connect to the I2C data line of the Arduino. On Uno, this is **A4**. On some variants, a separate notch marked "SDA" is also provided.

3.2.4 Run the Demo

In the Arduino IDE, open File ⟶ Example ⟶ adafruit MPU6050 ⟶ basic readings for MPU6050, compile, and upload to Arduino. Now open the serial monitor, you should see an output similar to that shown in Fig. 3.3.

There's three lines of output from the sensor in every sample:

- `Acceleration X: -5.60, Y: 4.30, Z: -7.44 m/s^2`
  This is the Adafruit_Sensor'ified nice output which is in m/s*s, the SI units for measuring acceleration. No matter what the range is set to, it will give you the same units, so this output is preferable to the raw one. The range can be set to 2g, 4g or 8g.


Output
Serial Monitor
Message (Ctrl + Enter to send message to 'Arduino Uno' on 'COM4')

```
13:22:23.583 -> Adafruit MPU6050 test!
13:22:23.907 -> MPU6050 Found!
13:22:23.907 -> Accelerometer range set to: +-8G
13:22:23.907 -> Gyro range set to: +- 500 deg/s
13:22:23.907 -> Filter bandwidth set to: 21 Hz
13:22:23.907 ->
13:22:23.985 -> Acceleration X: -5.80, Y: 4.23, Z: -7.36 m/s^2
13:22:24.028 -> Rotation X: -0.02, Y: -0.03, Z: -0.01 rad/s
13:22:24.028 -> Temperature: 21.82 degC
13:22:24.028 ->
13:22:24.528 -> Acceleration X: -5.82, Y: 4.22, Z: -7.31 m/s^2
13:22:24.528 -> Rotation X: -0.02, Y: -0.03, Z: -0.01 rad/s
13:22:24.528 -> Temperature: 21.85 degC
```

Figure 3.3: The output of the MMA8451 demo.

- `Rotation X: -0.01, Y: -0.03, Z: 0.01 rad/s`
  This give the rotation angle of the sensor in each direction
- `Temperature: 23.78 degC` This gives the temperature

Task 3.2 *Proceed as follows:*

1. *Tilt the breadboard different ways: are the signs of the XYZ accelerations as you expected from the directions marked on the breakout board?*

3.2.5 Show the Output on Python

The following Python code tries to read three float numbers from Arduino every 0.001 second.

```
import serial
import time

ser = serial.Serial('COM18', 9600)
receivestring = ser.readline()   # ignore the first read in

while True:
  receivestring = ser.readline().decode("utf-8").strip()  # read one line and
    convert bytes to a string
  receivelists = receivestring.split(',')
  print(receivelists)
  time.sleep(1)
```

Listing 3.2: Receiving Pitch, Yaw and Roll from Arduino.

Task 3.3 *Proceed as follows:*

1. *On the Arduino side, modify the code of the demo program, so that the Python program in Listing 3.2 is able to read what it writes on the serial port. Hint: print rotation XYZ in one line and separate then with comma.*
2. *Run the code and Explain your code to your supervisor.*

3.2.6 Rotation 3D Object

Task 3.4 *Try to combine code Listing 3.1 and Listing 3.2 such that the 3D object in Listing 3.1 will rotate according to the Pitch, Yaw and Roll received from Arduino.*