|  |  |
| --- | --- |
|  | **engduino®**Tutorial  Engduino Support Team - support@engduino.org |

Using Accelerometer to measure Distance and Height

# Overview:

Engduino has 3-axis xyz-accelerometer built in. This would allow us to apply trigonometry to calculate the angle between the axes and turn it into many interesting applications. This example will use the accelerometer on Engduino to calculate the angle. By applying trigonometry, we can measure the distance and height of an object.

# Aim:

This tutorial aims to guide you through on how does an accelerometer works and gives you the steps to create a MATLAB application that measures distance and height.

# Objectives:

* Understand the working principle of Accelerometer
* Use Accelerometer reading to calculate the angle
* Apply trigonometry to calculate distance and height of an object

# Learning Outcomes:

By the end of this tutorial, you should be able to

* Understand how does an accelerometer works
* Calculate angle using the xyz-axis on accelerometer
* Use tan, cos, sin function in MATLAB

# Pre-requisite

* Engduino MATLAB Support toolbox and MATLAB installed
* Engduino configured to make it discoverable in MATLAB

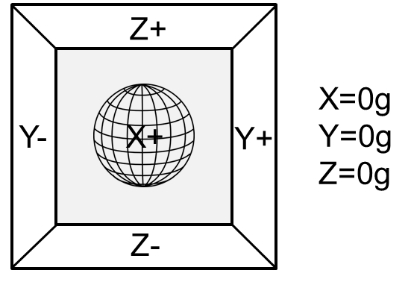
# Tutorial

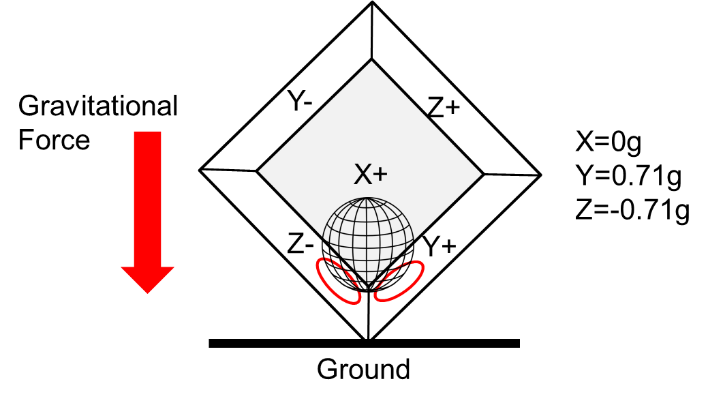
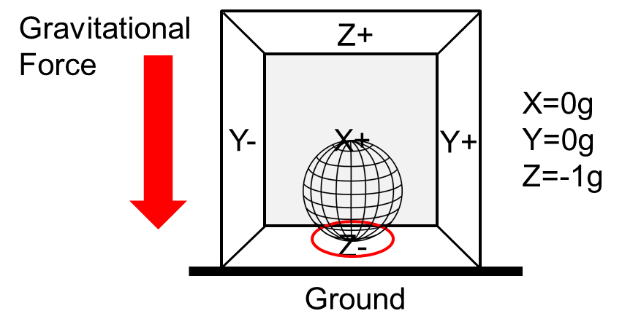
If you have gone through our previous example that uses the accelerometer to get acceleration, then double integrate it to get the distance, you should realise that the measurement is not very reliable. In fact, most of the time it does not work quite well.

As such, in this tutorial, we are going to use a much more reliable approach, which is using the angle given by accelerometer, apply the trigonometry to calculate the distance and height.

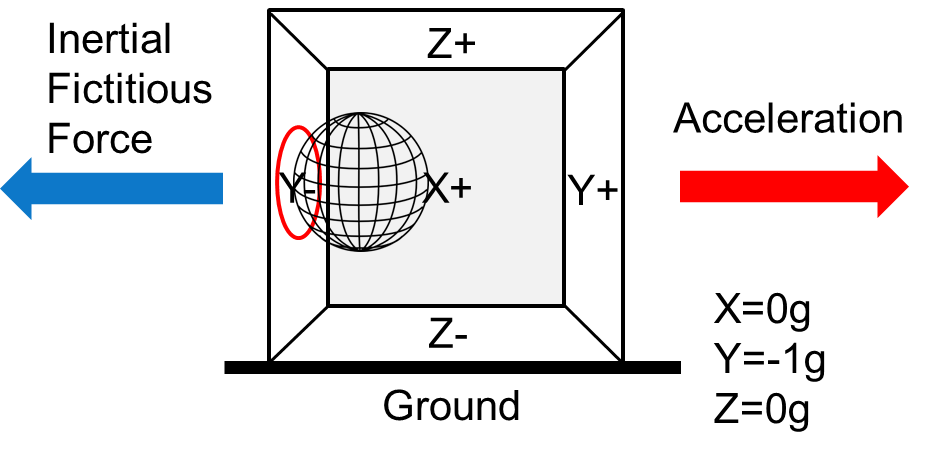
**1. Understand how an accelerometer works**

First, let us explain the working principle of the accelerometer to help you better understand why using the angle to calculate the distance is better than using acceleration. Imagine that the accelerometer is like a box with a ball in it.

If there is no gravitational force or any other fields that might affect the position of the ball, the ball will simply float in the middle of the box. In this case, without touching any sides of the box, the accelerometer will measures 0g on all three axis.

If we take this model and put it on earth, the gravitational force will act on the ball and cause it to touch the sides of the box. Imagine that the sides of the box are pressure sensitive, the accelerometer will detects this pressure force and gives the reading as shown in the above model. We can compare the readings to get the orientation.



In the model above, we move the accelerometer to the right, the ball will move in the opposite direction of the acceleration due to inertial fictitious force. The accelerometer will gives the reading of this force which is directed at the opposite direction of the acceleration. However, in this case we are assuming that this force is purely our acceleration, but in reality, shaking, rolling can also add forces to the acceleration and the accelerometer cannot differentiate these forces. As the result, there will always be error in the calculation. In comparison, you probably notice that accelerometer performs better at giving the angle and orientation than giving acceleration. Thus, using angle gives us a more accurate result.

**2. Work out the direction of the angle**

Try to work out the angle when you rotate the Engduino. Figure out the xyz-axis of the accelerometer first. Plug in the Engduino to the computer, launch MATLAB and create a new script. Use the code below to connect to Engduino.

if (~exist('e', 'var'))

e = engduino();

end

This code will make the connection to Engduino. Ensure that MATLAB is able to connect to Engduino by checking at the message in the command window before you continue.

Once you have connected the Engduino, stop the script and add the following code below the existing code.

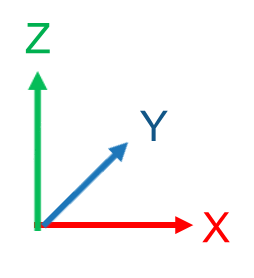
while(1)

newReading = e.getAccelerometer()

pause(0.5);

end

This code will read the accelerometer and returns the xyz axis readings respectively.

Work out the axis, you will get approximately either -1g or 1g on the z-axis if you lay the Engduino flat on a table.

Once, you have figure out the xyz axis of the accelerometer, proceed to find out the tilt angle when you rotate the Engduino. You can use cos, sin, tan to find out the angle in between two axis.

Modify the while loop to the below code.

while(1)

newReading = handles.board.getAccelerometer();

gx = newReading(1);

gy = newReading(2);

gz = newReading(3);

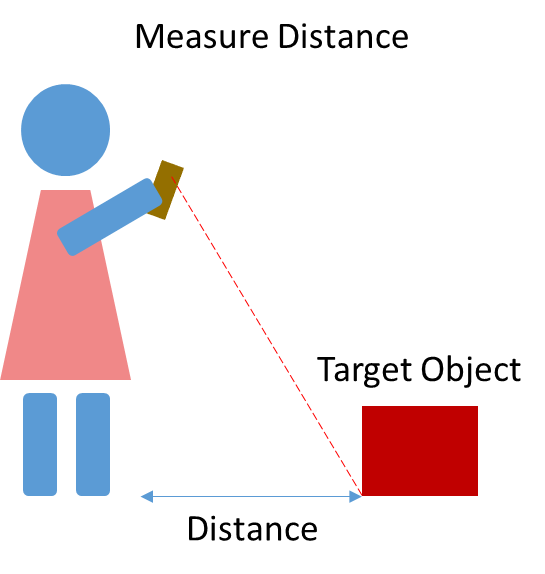
angle = atand(gz/gy);

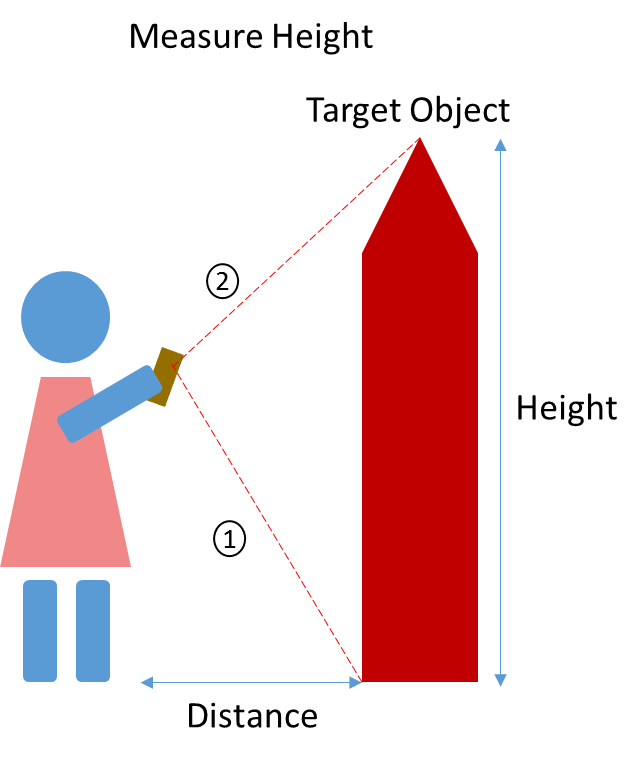
end

This code stores the xyz reading of the accelerometer to gx, gy, gz respectively. The function “atand()” perform operation. Try this using a different axis and find whether this makes sense to you.

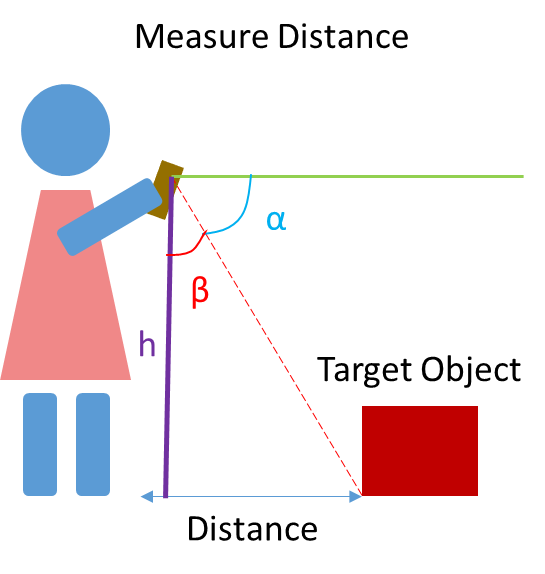
**3. How to get distance and height?**

Now that you can get the angle between the rotations, think about how to get the distance.

You want to measure the distance between the target object and you. Given that the accelerometer on the Engduino is able to give you the angle, how can you calculate the distance by pointing the Engduino at the object that was on the floor?

Similarly, think about how you can measure the height of the object that is placed on the floor.

Try to work out the equation before you turn into the next page for answer.

Let us assume h as your height - 0.3m. You should be able to calculate the angle α and β by pointing the Engduino on the target object.

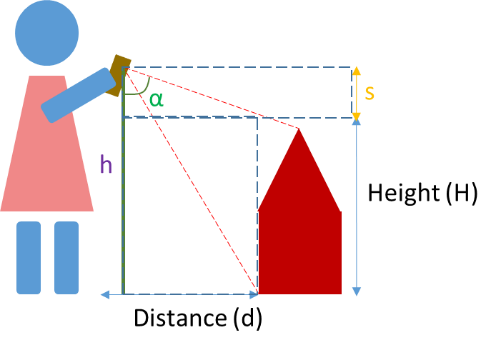
In our example, we use the Z and Y axis on the accelerometer to calculate the angle.

Now, we can calculate β.

The last part to calculate the distance,

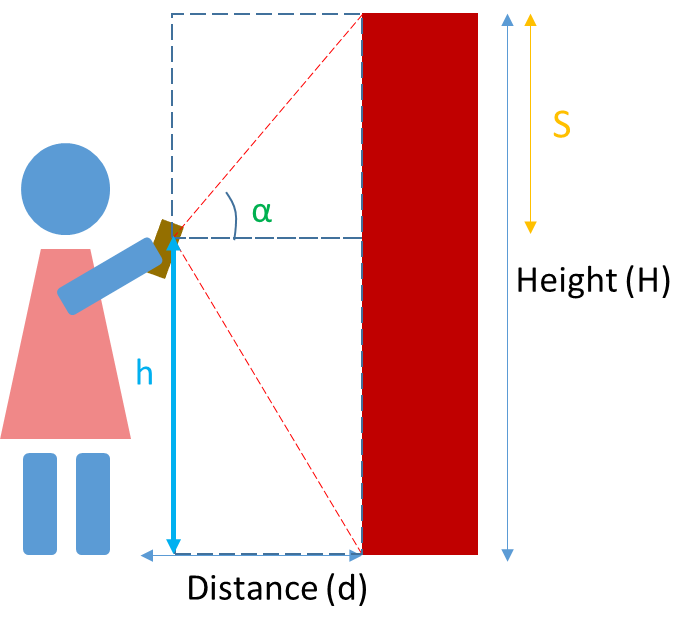
β)

Using the above equation, the distance d is known. The accelerometer can continue to give us the angle α. So to calculate H, we have two cases, the object is lower than us or the object is higher than us.



The object is lower than us. We can get S simply by

This will give us a positive value. To get the object height, we just perform the following.

The object is higher than us. We can still get S with the same equation. But now S will be a negative value. As such we can still apply the same equation as above to get the height.

**4. Steps to create the program**

The pseudo code for the program is shown below.

Initialise the user’s height

Connect the engduino

Repeat

Get accelerometer reading

Calculate the angle using z and y axis

Calculate distance with the angle

if user want to calculate distance

print distance

else user want to calculate height

calculate height

print height

end

**5. Create the actual program**

Let us start coding the actual program. Create a new script on MALTAB. The first thing we do is initialise the variables needed. Variables are seen as a temporary memory space in the computer to store values that we will be using in our program.

We will set the frequency of our program as 100Hz, this will define how fast the program runs at its time interval. T = 1/frequency.

% Set reading frequency [Hz] - readings per second.

frequency = 10;

Initialise the user’s height which is roughly your height - 0.3m. We initialise measure\_distance to true when we are calculating the distance, and false when we are calculating the height.

% Height

height = 1.5;

measure\_distance = true;

The following lines check whether the object ‘e’ is available in MATLAB workspace. If it does not exist, it calls the function “engduino()” which will connect the Engduino hardware and store it as an object.

if (~exist('e', 'var'))

e = engduino();

end

Create the main while loop to do the calculation and keep the program running until we stop MATLAB.

while (1)

Read the accelerometer. The accelerometer reading will return the xyz value from the accelerometer respectively.

% Read acceleration vector from Engduino's accelerometer sensor.

newReading = e.getAccelerometer();

gx = newReading(1);

gy = newReading(2);

gz = newReading(3);

Next we will calculate the angle α in (degree) from the accelerometer, which is the outer angle α, to get the inner angle β, we simply use 90 – outer angle. The function atand() performs operation.

outer\_angle = atand(gz/gy);

inner\_angle = 90-outer\_angle;

Next, we want to convert the angle to radian because the MATLAB function tan() performs the calculation in radian.

angleInRadians = deg2rad(inner\_angle);

The below is the part of the code that perform the calculation. First, it checks whether the program is measure distance or height, then it performs the calculation and output the result accordingly. When measuring distance, we also set a limitation that the angle cannot be larger than 90 or smaller than 0. This is to ensure that the user is pointing to the object on the ground to measure the distance. We also round up our result to 2 decimal places.

if(measure\_distance)

% measuring distance and update the distance

if(inner\_angle>=90||inner\_angle<=0)

disp('Max');

else

distance = height\*tan(angleInRadians);

round\_up\_distance = round(distance\*100)/100;

textLabel = strcat('Distance ',(num2str(round\_up\_distance)));

disp(textLabel);

end

else

% measuring height and update the height

temp\_height = distance/(tan(angleInRadians));

object\_height = height-temp\_height;

round\_up\_height = round(object\_height\*100)/100;

textLabel2 = strcat('Height ',(num2str(round\_up\_height)));

disp(textLabel2);

end

Here, we use the push button on the Engduino to toggle between calculating distance and height.

if(e.getButton())

if(measure\_distance)

measure\_distance=false;

else

measure\_distance=true;

end

end

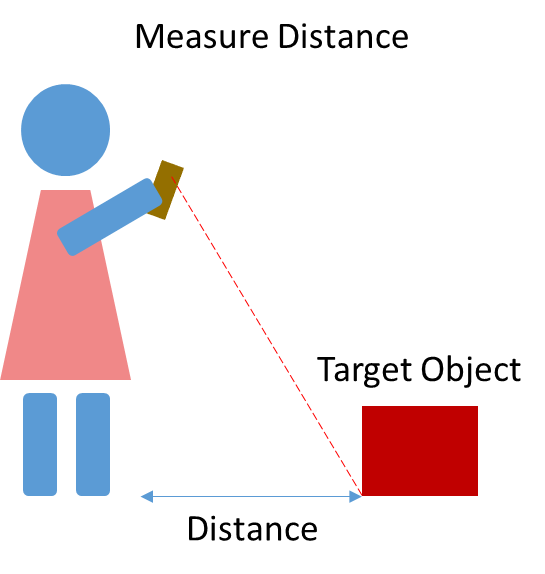
The last part is simply create a delay in our program to specify how fast the program is running.

pause(1/frequency);

That is all we need in the main while loop. We close the while loop with an “end”.

end

This completes our program. You may connect the Engduino and run the program. Look at the MATLAB command window for the result. Modify the height value to your height – 0.3m.

To measure the distance, point the Engduino to the object on the ground. 

To measure height, measure the distance between you and the object first, then press the push button, and rotate Engduino until it points to the top of the object.

