13

HARMONIC MOTION WITH ACCELEROMETER

STEMJAM Teaching Guide

Developing make spaces to promote creativity
around STEM in schools
Acronym: STEMJAM
Project no. 2016-1-ES01-KA201-025470

www.stemjam.eu

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

HARMONIC MOTION WITH ACCELEROMETER

ABSTRACT

This activity extend the previous one on “Harmonic Motion” with the aim to offer a direct observation of kinematic variables of interest, like acceleration. Similar observation are useful to older students (15-16 years) for the understanding of the equations of motion ad of their solutions. For younger pupils (12-14 years), it may simply be funny to observe the harmonic motion of the mBot itself, as a complimentary measurement with respect to more standard experiments.

For the full exploitation of the gyroscope, we also developed a short guide about the measurements of pitch and roll angles. You may find it useful in physics laboratories, e.g. in order to measure the slope of an inclined plane and of course to help understanding the principle of navigation control.

DIDACTIC OBJECTIVES

While playing the activity you will…

- Physics: explore Harmonic Motion, identify its main parameters and equations.
- Physics: directly observe the oscillations as a function of time in the values of acceleration.
- Technology: learn about a gyroscope. The principle of navigation control: pitch, roll and heading.

While implementing the code you will learn about…

- Computer Science: algorithm development.
- Mathematics (advanced, for students 17-18 years old): numerical derivatives.

STEM Subject: Science☐ Technology ☒ Engineering☐ Mathematics☒

Education Level: 12-14 years☒ 14-16 years☒

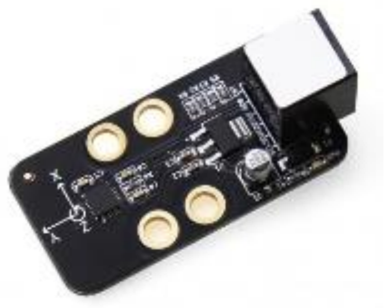
PROBLEM STATEMENT

Hang the mBot from a spring and measure the acceleration and the period of its harmonic motion, thanks to a 3-axis Accelerometer and Gyro sensor. Exploit the gyroscope for the purpose of navigation control, by learning about the pitch and roll -8and heading) angles.

BOM (Bill of Materials Needed)

- mBot **RANGER**:

- Me **3-Axis Accelerometer and Gyro Sensor**:



- Me **LED Matrix 8 × 16** or / and **TFT LCD screen:**

- Me **Potentiometer**:

ACTIVITY DESCRIPTION

The accelerometer-gyro sensor is exploited to study the harmonic motion of the mBot attached to a spring, as well as to perform measurements of pitch and roll angles.

First version

In the following we first deal with the acceleration measurements: we list the simple steps needed to play the activity, discuss the characteristics of the Gyro sensor, illustrate the algorithm and code, and finally show the main results. Then, in the second part of the document, we focus on the standard application of the gyroscope to navigation (pitch and roll). Examples and pictures will help to highlight why learning these topics with the mBot results attractive, simple and hence more effective.

Through the text, useful tips leading to better results are mentioned and highlighted in orange color.

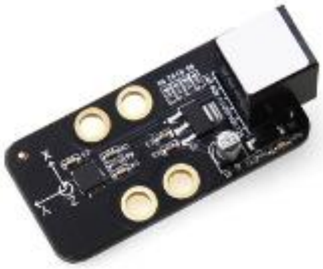
The mBot is made to hang from a spring and perform an harmonic motion. From the gyro-sensor outputs, proportional to acceleration, the oscillation period is evaluated. Both mBlock (Scratch) and Arduino versions of the code are made available. In Arduino, the sensor data are shown in serial output and can be plot in an external software in order to get a deeper understanding of the equations of motion.

Experimental Procedure

1. Prepare the mBot with the Accelerometer/Gyro sensor and the white screen. Despite the sensor is probably designed to be mounted below the chassis, similarly to the line follower sensor, here it is convenient to mount the sensor in vertical position on one side of the robot (in the proposed example the sensor was mounted on the right hand side of the mbot, connections up, with the x-axis horizontal and pointing backward, y-axis downward, z-axis horizontal and pointing far from the mBot). The reason for this choice will be detailed below.
2. Hang a spring from a proper support and attach the mBot to it (you can either insert the spring in the m-shaped hole on the back of the chassis or use twine. For safety reasons make sure the support is stable and heavy. Take into account the mBot is heavy and if the spring has a low elastic constant, as expected, the spring will stretch considerably, in our case 1.5 m).
3. Turn on the mbot, preloaded with the code, and start an oscillatory motion.
4. The oscillation period is shown on the led display and the spring constant can eventually be evaluated. If the program is run from Arduino standard interface with a usb cable, the three Gyro sensor output will be shown in standard output (serial monitor).
5. Discuss the results with students (guidance and suggestion below).

(Connection map: 1. Led Screen 3. Gyro Sensor . See electronics sheet for details)

The Me 3-axis accelerometer and gyro-sensor



This sensor, based on the MPU6050 chip, integrates measurements of acceleration and angular velocity and returns information on the orientation and motion of the robot. An introduction to the sensor is offered here [1]. To interface the sensor to Scratch and Arduino / C++, the Makeblock company offers a programming library based on the Gyro class documented here [2]. Actually this library limits the potential of the MPU6050, in the sense it give access to a few variables. Namely the three main outputs are (apparently) three angles, the X-axis, Y-axis and Z-axis angles. The relationship of this outputs with physical variables of interest is not clear, neither in the documentation, nor on the makeblock forum.

We summarize here all the information found:

1. In the forum, the technical support mention that the 3 variables are the Euler angles defining the orientation of the sensor with respect to its initial position (set when the function gyro.begin() is invoked). In many fields of application, including aeronautics, these angles are also known as pitch, roll and heading (or yaw): they are the standard angles given by a gyroscope and are used for navigation purposes. The sensor is indeed essentially designed for robot navigation.

2. The three output are effectively angles: the x- and y-axes angles vary from -90 to +90, while the z-axis angle range from 0 to +360. The value in the z-axis also drifts consistently and linearly with time (see fig.3). This is actually a common problem in electronic gyroscopes and it is why we suggested to mount the sensor vertically for this experiment, so that the z-axis remains horizontal during the harmonic motion and we do observe the oscillations along the x- and y-axes, without this distortion.

3. Nevertheless the information in 1 is incorrect. The 3 outputs are not simply Euler Angles determining the orientation, but each is proportional to the related component of the vector **a - g** where **a** is the acceleration of the robot and **g** is the gravity acceleration (e.g. The “x-axis angle” is proportional to the x component of the acceleration **a-g**). When the robot is at rest on a table, with the y axis downward, the y-axis angle is almost -90°, detecting gravity (**1 g** corresponds to 90 degree), and close to zero on the other axes.

Algorithm workflow and Comments to code

The algorithm is represented in the workflow below. Essentially the signal, proportional to acceleration, is measured along the 3 axes and maxima and minima of the oscillations are located by derivation. A five-point numerical derivative is used for the purpose [3]. If you are not familiar with this technique, consider it lies on the idea of taking into account the values of the function at a few points before and after the point of interest , like it is represented in the figure below.

The derivative at the point *n* is evaluated as a linear combination of the function values at neighbouring points, with proper coefficients, according to the formula.

$$f_n' = \frac{1}{12} f_{n-2} - \frac{8}{12} f_{n-1} + \frac{8}{12} f_{n+1} - \frac{1}{12} f_{n+2}$$

For further explanation and derivation of this formula, see reference [3].

When the sign of the derivative changes a maximum or minimum occurs, and the period can then be evaluated as twice the time between two of these critical points.

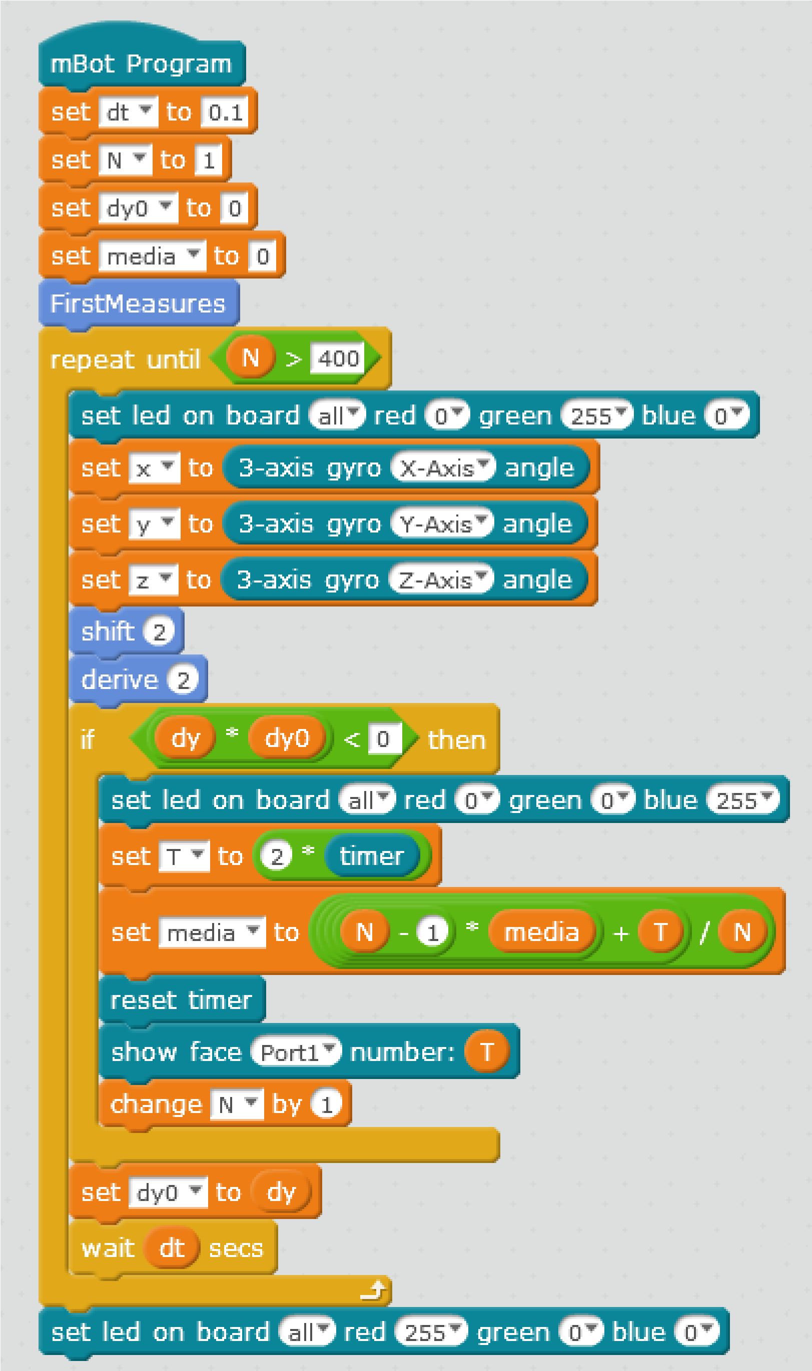
A technical remark. Actually this algorithm detects the period with an error which we should be aware of: because the derivative is made as a 5-points finite-difference, the peaks (either min or max) can be detected only two points after they occur, causing a delay in timer start and stop. This delay is always slightly different because of the finite sampling frequency, here chosen to be f=10 Hz. The maximum error that can be committed is lower than the sampling time dt=100 ms. Since the measured period is of the order of two seconds, this is within 5% and might be considered acceptable and comparable to other error sources.

Arduino Code

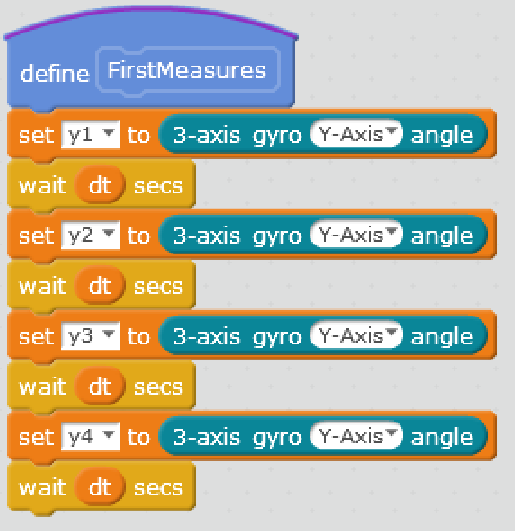
```
//Libraries.   MeMCore  is the main library to control  mBot and other Makeblock products

#include <Arduino.h>

#include <Wire.h>

#include <SoftwareSerial.h>

#include <MeMCore.h>


// Global variables declaration.  Special classes are made available for motor control and sensor and other mBot  components. In particular. MeRGBLed for the onboard led,  MeGyro for the gyro sensor, MeLEDMatrix for the led white screen

MeDCMotor motor_9(9);

MeDCMotor motor_10(10);

double x, y, z;  double A, B;

double T, t, t0;

double x1,x2,x3,x4,x5;      double y1,y2,y3,y4,y5;

double dt, dy, dy0;

double N, media;

MeRGBLed rgbled_7(7, 7==7?2:4);

MeGyro gyro;
```

```
MeLEDMatrix ledMtx_1(1);


void setup(){                          //Commands that are  run once at program start

   Serial.begin(9600);                 //Estabilish serial/usb connection

   gyro.begin();                       //Initialize the gyro sensor

   ledMtx_1.setColorIndex(1);          //Screen setup

   ledMtx_1.setBrightness(6);

   Serial.println("Data from mBot Gyro sensor");    //Header for standard output

   Serial.println(" xAngle  yAngle  zAngle ");

   N=1;

   dy0=0;

   FirstMeasures();                    // Get the first four points (without doing anything else)

                                        // Because we use a five point derivative,

                                       // we have to enter the loop with the first 4 points already available

}


void loop(){                           //Commands run repeatedly till program ends
  //On-board leds are hold green while running
   rgbled_7.setColor(0,0,255,0);
   rgbled_7.show();

   //Read gyro sensor outputs and print values on standard output
  // for data visualization and future use
   x = gyro.getAngle(1);    y = gyro.getAngle(2);    z = gyro.getAngle(3);
   Serial.print(x);   Serial.print("  ");    Serial.print(y);    Serial.print("  ");    Serial.println(z);

   //Rearrange last 4 previous data and the new one. See function definition below.
   shift(2);

   //Evaluate numerically the derivative at point n . See function definition below.
   dy=derive(2);

   // if a minimum or maximum is found, period is evaluated and timer reset.
   // On Boards Leds turns to blue for a while
   if (dy*dy0<0){
     rgbled_7.setColor(0,0,0,255);
     rgbled_7.show();
     t = millis()/1000.0 - t0;              //Retrieve timer value
     T=2*t;
     media=((N-1)*media+T)/N;          //average over the periods measured up to now
     t0 = millis()/1000.0;                  //reset timer;
     N=N+1;                              //number of half oscillations
     }

    dy0=dy;                              //store current derivative for comparison to the next one
    ledMtx_1.showNum(media,3);    //show the average period on the onboard screen
    delay(100);                         //SAMPLING TIME = 100 ms
    gyro.update();                      //retrieve updated data from the gyro sensor
}

//Of the five current points of the function, discard the older, shift the others and add the new one (=current sensor reading) in
position 5
void shift(int i){
  if (i==1) {
    x1=x2; x2=x3; x3=x4; x4=x5; x5=x;   }

 else if (i==2) {
    y1=y2; y2=y3; y3=y4; y4=y5; y5=y;  }

 else {Serial.println("Possible input for the shift function are 1 and 2 only, corresponding to the x and y axes");}
}

//Evaluate numerically the derivative at point n according to the formula given in the text
double derive(int i){
  double der;
  if (i==1) {
    der=(x1-8*x2+8*x4-x5)/12; }//five-points finite-differences

 else if (i==2) {
    der=(y1-8*y2+8*y4-y5)/12; }//five-points finite-differences

  else {Serial.println("Possible input for the shift function are 1 and 2 only, corresponding to the x and y axes");}
  return der;
}

//Read the first few points
void FirstMeasures(){
    gyro.update();    y1=gyro.getAngle(2);    delay(100);       //SAMPLING TIME = 100 ms
    gyro.update();    y2=gyro.getAngle(2);    delay(100);
    gyro.update();    y3=gyro.getAngle(2);    delay(100);
    gyro.update();    y4=gyro.getAngle(2);    delay(100);    gyro.update();
}
```

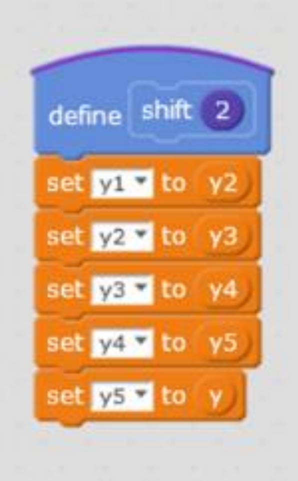
mBlock scratch Code



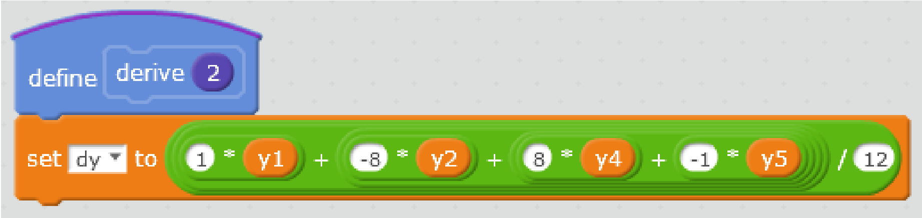
```
mBot Program
set dt to 0.1
set N to 1
set dy0 to 0
set media to 0
FirstMeasures
repeat until (N > 400)
    set led on board all red 0 green 255 blue 0
    set x to 3-axis gyro X-Axis angle
    set y to 3-axis gyro Y-Axis angle
    set z to 3-axis gyro Z-Axis angle
    shift 2
    derive 2
    if (dy * dy0 < 0) then
        set led on board all red 0 green 0 blue 255
        set T to 2 * timer
        set media to (((N - 1) * media) + T) / N
        reset timer
        show face Port1 number: T
        change N by 1
    set dy0 to dy
    wait dt secs
set led on board all red 255 green 0 blue 0
```



```
define FirstMeasures
set y1 to 3-axis gyro Y-Axis angle
wait dt secs
set y2 to 3-axis gyro Y-Axis angle
wait dt secs
set y3 to 3-axis gyro Y-Axis angle
wait dt secs
set y4 to 3-axis gyro Y-Axis angle
wait dt secs
```



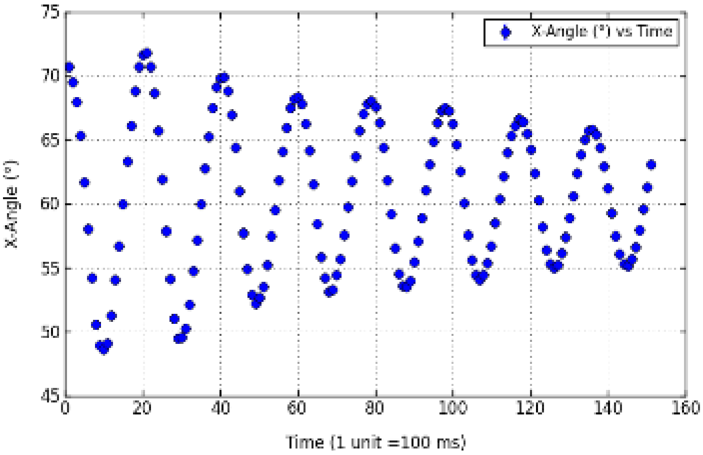
```
define shift 2
set y1 to y2
set y2 to y3
set y3 to y4
set y4 to y5
set y5 to y
```



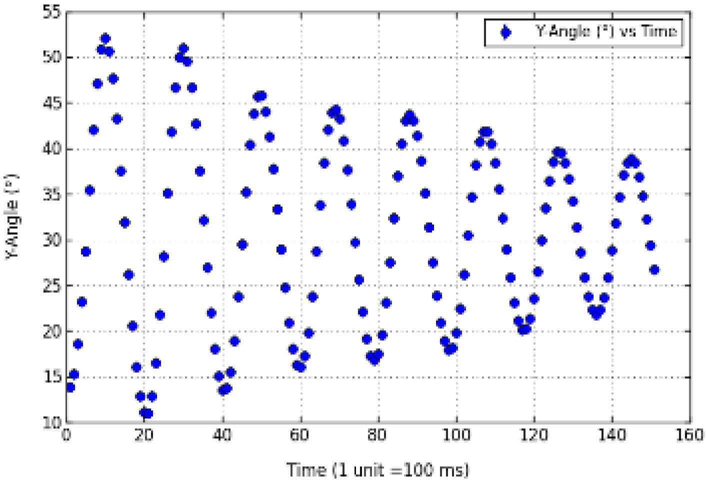
```
define derive 2
set dy to ((1 * y1) + (-8 * y2) + (8 * y4) + (-1 * y5)) / 12
```

Experimental Results

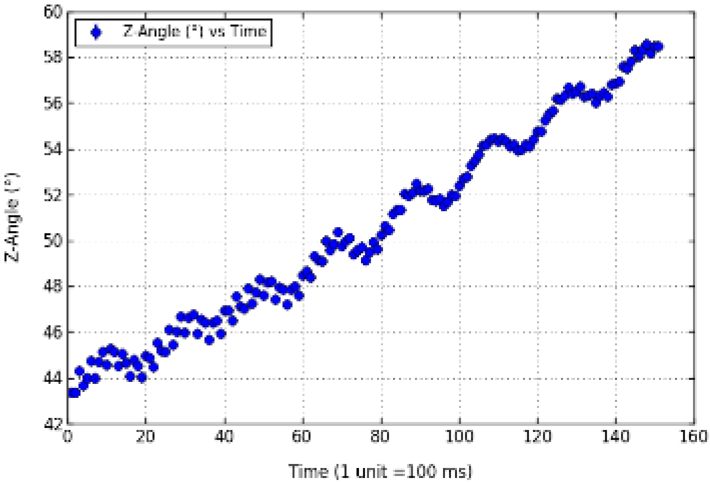
Data shown in figure 3 were retrieved from Arduino standard output and plot in an external software. The harmonic oscillation are clearly visible on both the x and y axes channel. As it is apparent, at this timescale (sampling time 100 ms) the sensor outputs good quality data with very low noise level.



In this experiment the estimated period is approximately 2 seconds, but slowly reduces with time to 1.75 seconds.



A similar analysis is of interest with students aged 15 or above, studying physics: the oscillations of acceleration with time can be easily appreciated, and offer useful insights into the equation of motion. As a further development, one could add to the program the direct measurement of mBot displacement through the ultrasonic sensor.



On the z-axis channel a sizeable and linear sensor drift is superimposed to the oscillations and therefore the channel is not the most convenient one for data analysis or further observations. This is why we suggested to mount the sensor in vertical position (or in any position that would let the z-axis perpendicular to the harmonic motion.

Overall the “Me 3-axis Acceleration and Gyro sensor” proved to be a good quality sensor even if not so easy to use because of the incomplete documentation.

Second versión

Roll Angle and Oscillating Motion

To carry out the activity, and to be able to calculate the oscillating movement to which the mBot can be subjected, we need to know what **the roll angle** is. It **is defined as the angle of lateral inclination that the perpendicular of the plane of the vehicle forms with forms the vertical.** When the mBot is in a horizontal plane, the angle of inclination will be approximately 0, since it has a small margin of error.

When the mBot is in an inclined plane, the angle of inclination will be equal to that of the surface. In addition, the Ranger will show graphically in the LED Matrix its lateral inclination, that is, the angle it presents with respect to the longitudinal axis (Y). If this angle exceeds the value set by the potentiometer, it will warn of this by means of an acoustic and luminous signal.

Since the Auriga board that the Ranger incorporates has an integrated inertial measurement module (IMU) with a gyro and a 3-axis accelerometer, so it is enough to use the accelerometer value corresponding to the longitudinal axis of the Ranger (Y axis) to obtain the lateral angle of inclination directly.

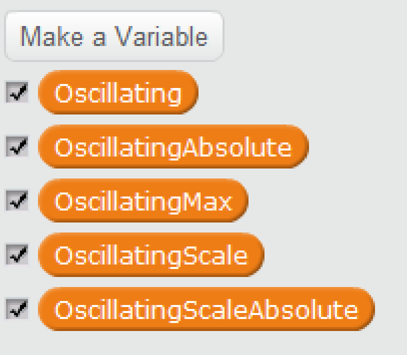
Also, in the TFT LCD Screen we will show the angle of the plane.

In the following we discuss the steps necessary to go through the full activity:

a. First, we connect the Led Matrix and TFT LCD Screen to mBot Ranger.

b. We connect the Potentiometer Sensor too.

c. And then, we start programming the mBot Ranger:

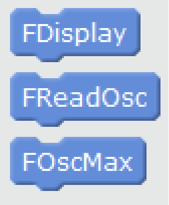


```
Make a Variable
☑ Oscillating
☑ OscillatingAbsolute
☑ OscillatingMax
☑ OscillatingScale
☑ OscillatingScaleAbsolute
```

The most significant variables are:

- Oscillating: Y axis IMU angle.
- OscillatingAbsolute: Absolute Value of Roll.
- OscillatingMax: Maximum Value of the Angle.
- OscillatingScale: Scaled Angle Value.
- OscillatingScaleAbsolute: Absolute Scale Angle Value.

We define three functions as follow:

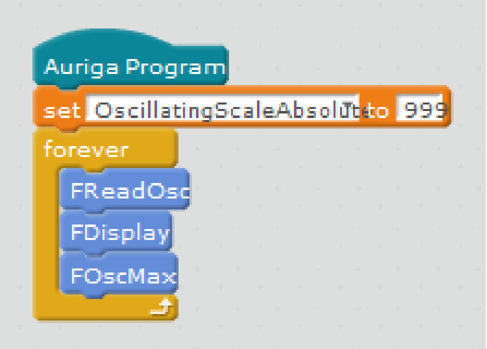


```
FDisplay
FReadOsc
FOscMax
```

- FDisplay: The LED Matrix and TFT LCD Screen will display all the data instantly.

- FReadOsc: This function is responsible for collecting the angle of the Auriga shield.

- FOscMax: Function that will keep the maximum angle established by the Potentiometer.

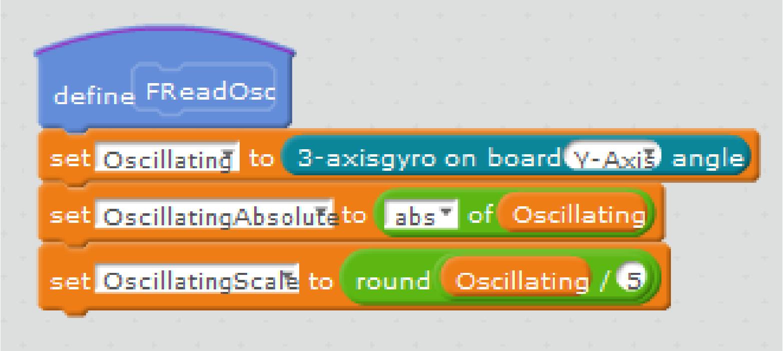


```
Auriga Program
set OscillatingScaleAbsolute to 999
forever
    FReadOsc
    FDisplay
    FOscMax
```

When we start the program, a default value will be set to the variable "OscillatingScaleAbsolute", which is the value of the Ranger angle indicator.

Then, the code will repeat the loop entering each function in search of changes in the longitudinal axis.

Here it is what each function actually does:

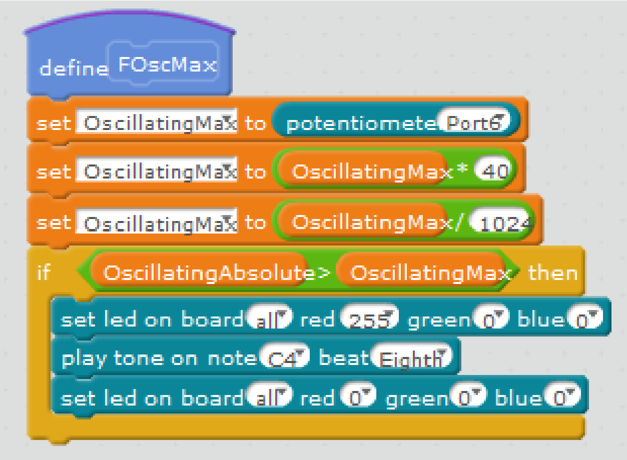


```
define FReadOsc
set Oscillating to 3-axisgyro on board Y-Axis angle
set OscillatingAbsolute to abs of Oscillating
set OscillatingScale to round Oscillating / 5
```

The value of “Oscillating” variable will be the value obtained by the longitudinal axis Y.

The value of “OscillatingAbsolute” variable is the absolute value of “Oscillating” variable.

And the value of “OscillatingScale” variable is the value of “Oscillating” variable rounded and divided by 5.



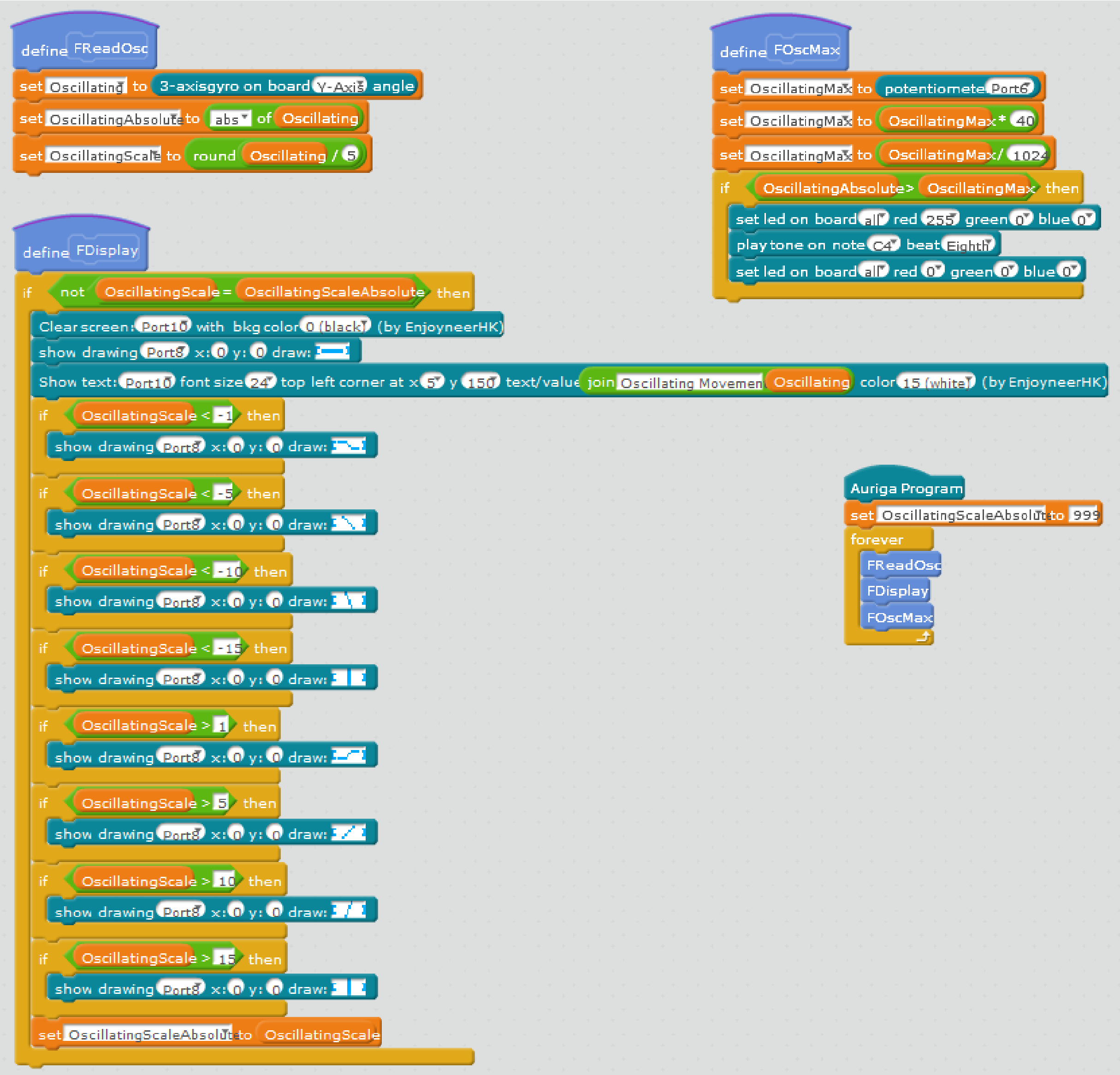
```
define FOscMax
set OscillatingMax to potentiomete Port6
set OscillatingMax to OscillatingMax * 40
set OscillatingMax to OscillatingMax / 1024
if OscillatingAbsolute > OscillatingMax then
    set led on board all red 255 green 0 blue 0
    play tone on note C4 beat Eighth
    set led on board all red 0 green 0 blue 0
```

Here, we can see how the “OscillatingMax” variable collects the value of the potentiometer and sets the maximum value at which the mBot Ranger can incline.

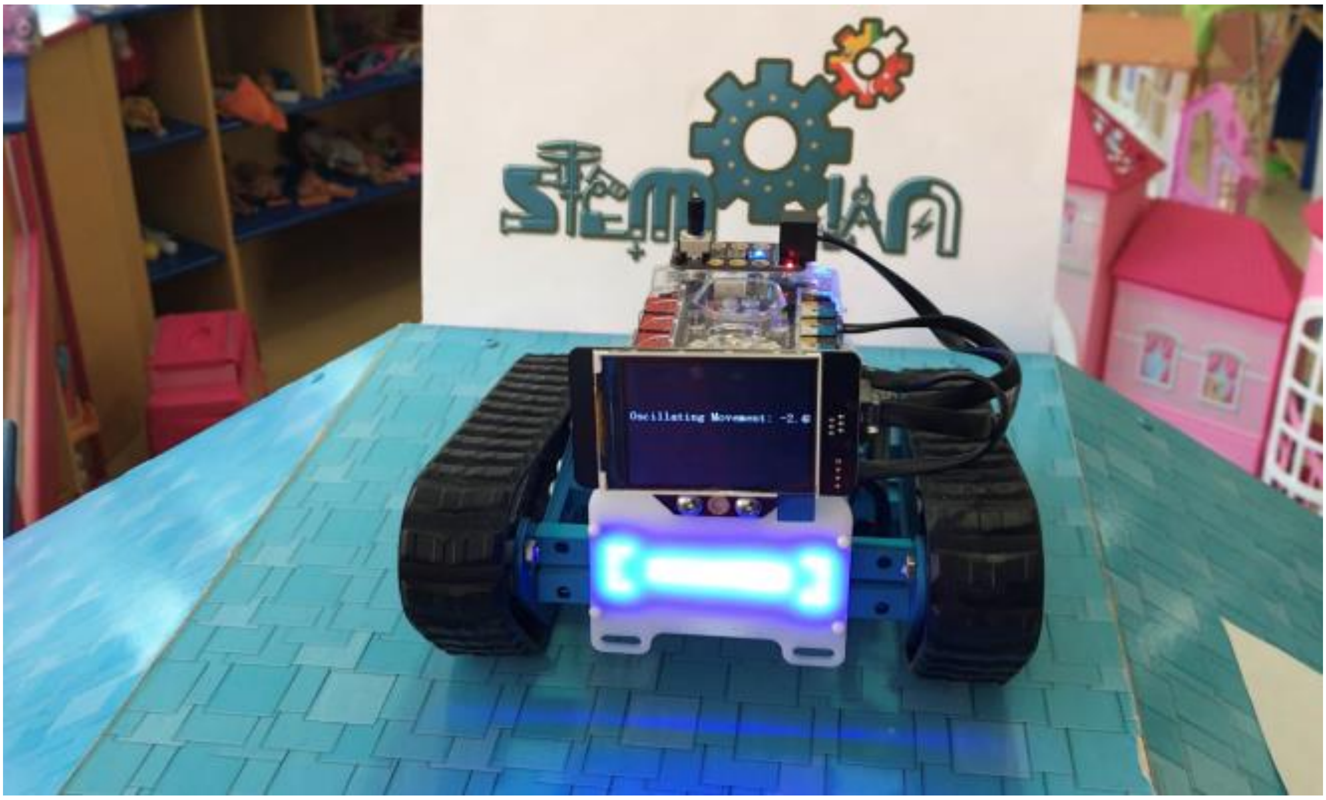
If the mBot exceeds that maximum angle, an acoustic and luminous signal will start to show.

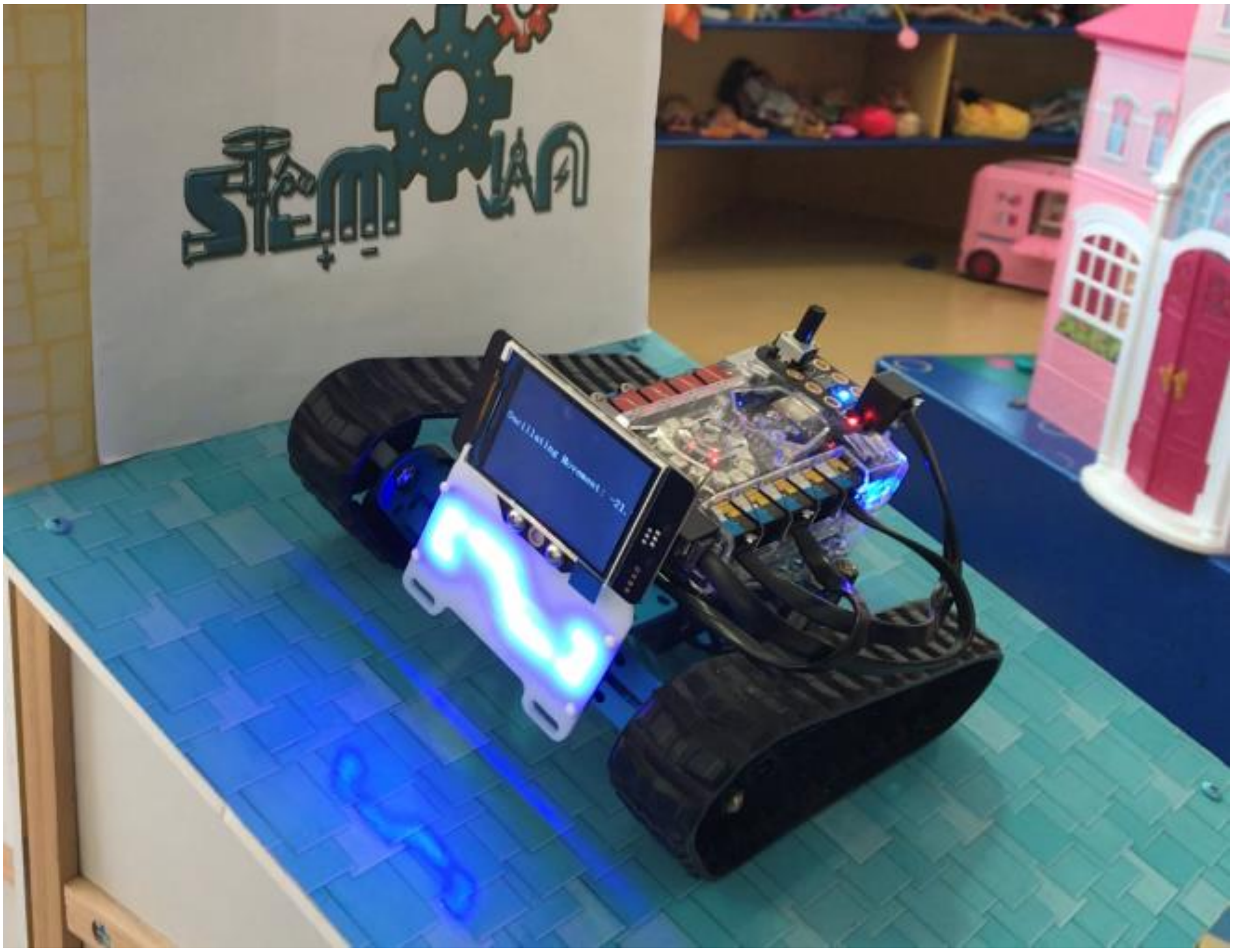
Finally the "FDisplay" function shows in real-time the inclination on the LED Matrix and the Roll Angle Value in the TFT LCD Screen.

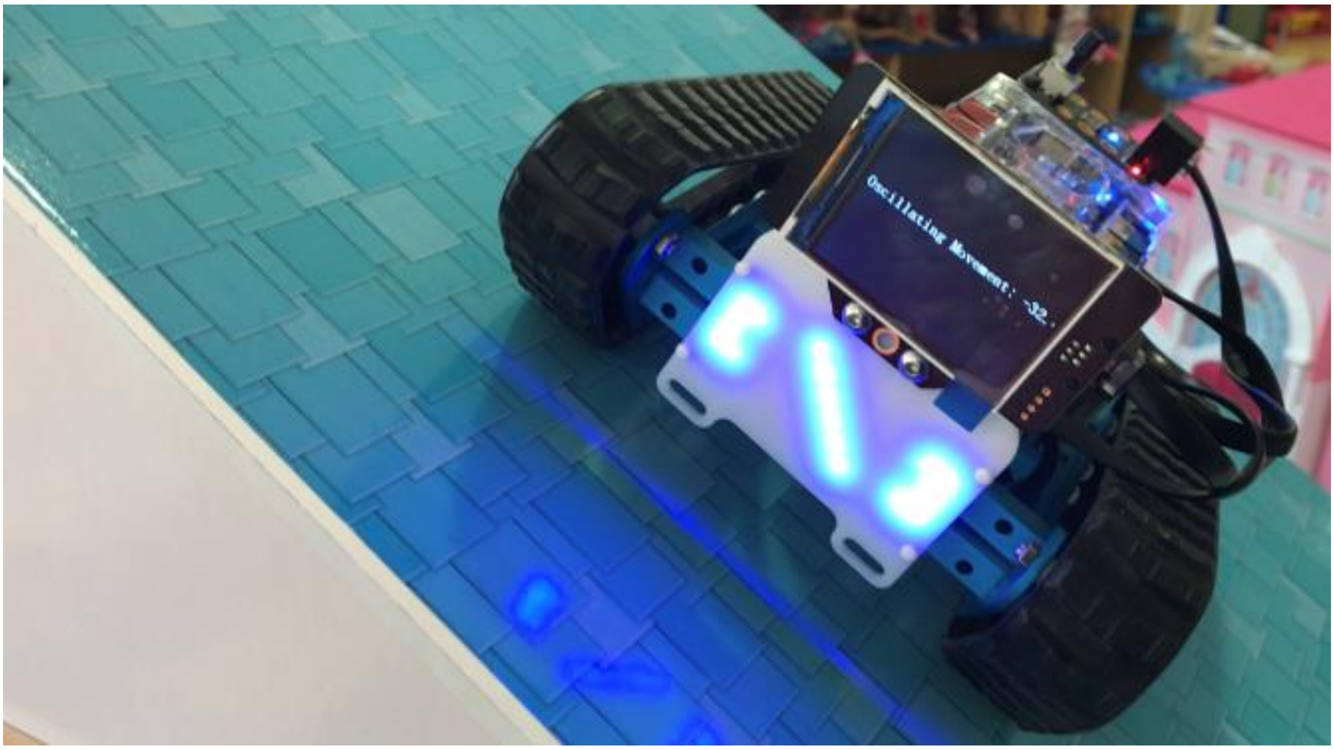
The full version of the program can be found below and in the following pages some photos from the activity are shared.

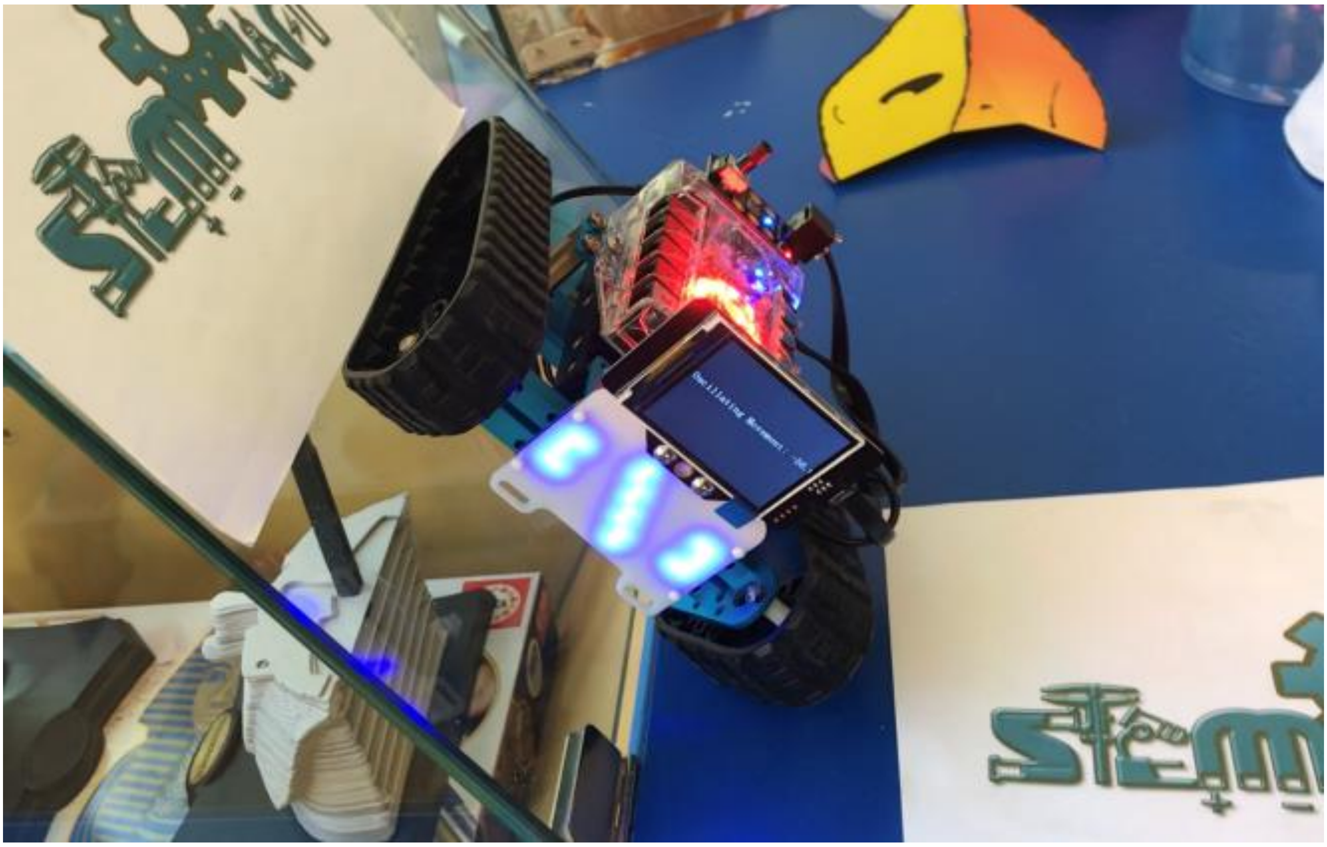


```
define FReadOsc
set Oscillating to 3-axisgyro on board Y-Axis angle
set OscillatingAbsolute to abs of Oscillating
set OscillatingScale to round Oscillating / 5

define FDisplay
if not OscillatingScale = OscillatingScaleAbsolute then
    Clear screen: Port10 with bkg color 0 (black) (by EnjoyneerHK)
    show drawing Port8 x: 0 y: 0 draw:
    Show text: Port10 font size 24 top left corner at x 5 y 150 text/value join Oscillating Movemen Oscillating color 15 (white) (by EnjoyneerHK)
    if OscillatingScale < -1 then
        show drawing Port8 x: 0 y: 0 draw:
    if OscillatingScale < -5 then
        show drawing Port8 x: 0 y: 0 draw:
    if OscillatingScale < -10 then
        show drawing Port8 x: 0 y: 0 draw:
    if OscillatingScale < -15 then
        show drawing Port8 x: 0 y: 0 draw:
    if OscillatingScale > 1 then
        show drawing Port8 x: 0 y: 0 draw:
    if OscillatingScale > 5 then
        show drawing Port8 x: 0 y: 0 draw:
    if OscillatingScale > 10 then
        show drawing Port8 x: 0 y: 0 draw:
    if OscillatingScale > 15 then
        show drawing Port8 x: 0 y: 0 draw:
    set OscillatingScaleAbsolute to OscillatingScale

define FOscMax
set OscillatingMax to potentiomete Port6
set OscillatingMax to OscillatingMax * 40
set OscillatingMax to OscillatingMax / 1024
if OscillatingAbsolute > OscillatingMax then
    set led on board all red 255 green 0 blue 0
    play tone on note C4 beat Eighth
    set led on board all red 0 green 0 blue 0

Auriga Program
set OscillatingScaleAbsolute to 999
forever
    FReadOsc
    FDisplay
    FOscMax
```


Oscillating Movement: -2.4


Oscillating Movement: -21.


Oscillating Movement: -32.


Oscillating Movement: -32.

Pitch angle

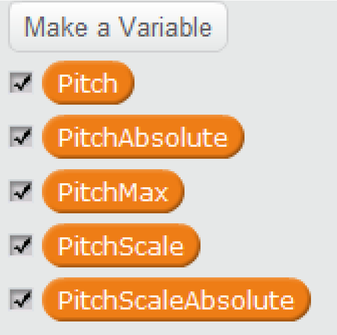
The Pitch Movement, whose name is of nautical origin, is used in the automotive world to indicate the longitudinal inclination of a vehicle with respect to the horizontal plane. In particular **the pitch angle is the one that the longitudinal axis of the vehicle forms with the horizontal plane.**

When traveling on a horizontal road the pitch angle will be approximately 0, since there is a small deviation of the sensor. When going up a slope, we will have a positive pitch angle. During the activity, the Ranger will graphically show in the LED Matrix its longitudinal inclination, that is, the angle of pitch or turn it presents with respect to the transverse axis (X). If this angle exceeds the value set by the potentiometer, it will warn of this with an acoustic and luminous signal. Also, in the TFT LCD Screen we will show the angle of the plane.

Since the Auriga board that incorporates the Ranger has an integrated inertial measurement module (IMU) with a gyro and a 3-axis accelerometer, so it is enough to use the accelerometer value corresponding to the transverse axis of the Ranger (X axis) to obtain the pitch angle directly.

a. In the following we discuss the steps necessary to go through the full activity. Similarly to the previous case, first we connect to the mBot Ranger both the Led Matrix and TFT LCD Screen as well as the Potentiometer Sensor.

b. And then, we start programming the mBot Ranger:

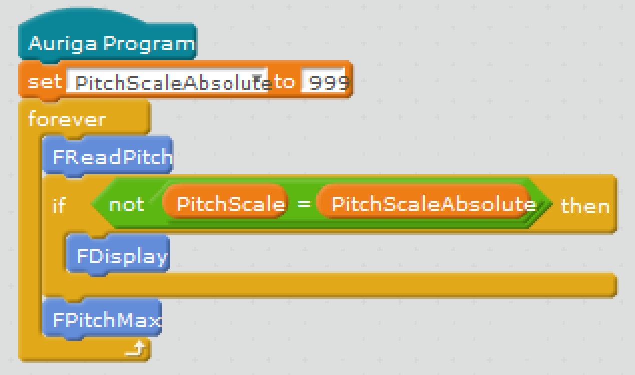


The most significant variables for:

- Pitch: X axis IMU angle.
- PitchAbsolute: Absolute Value of Pitch.
- PitchMax: Maximum Value of the Angle.
- PitchScale: Scaled Angle Value.
- PitchScaleAbsolute: Absolute Scale Angle Value.

We define three function as follow:

- FPitchMax: Function that will keep the maximum angle established by the Potentiom.
- FDisplay: The LED Matrix and TFT LCD Screen will display all the data instantly.
- FReadPitch: This function is responsible for collecting the angle of the Auriga shield.



```
Auriga Program
set PitchScaleAbsolute to 999
forever
    FReadPitch
    if not PitchScale = PitchScaleAbsolute then
        FDisplay
    FPitchMax
```

When we start the program, a default value will be set to the variable "PitchScaleAbsolute", which is the value of the Ranger angle indicator.

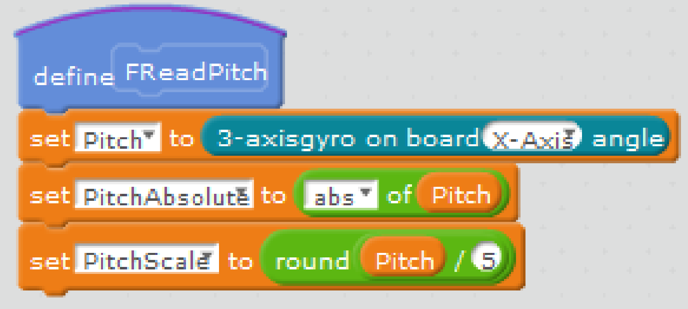
Then, the code will repeat the loop entering each function in search of changes in the transverse axis.

Here it is what each function actually does:

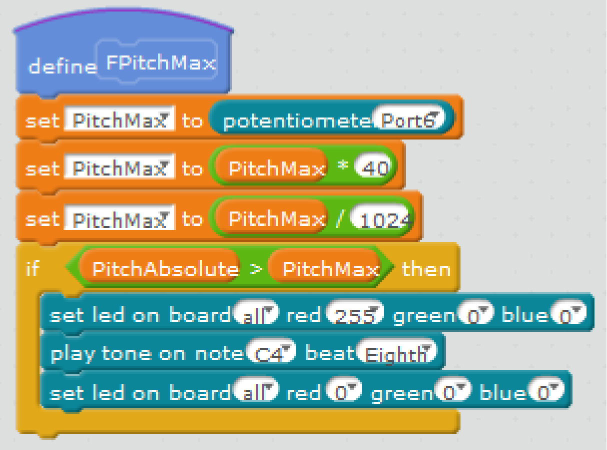
The value of "Pitch" variable will be the value obtained by the transverse axis X.

The value of "Pitch Absolute" variable is the absolute value of "Pitch" variable.

And the value of "PitchScale" variable is the value of "Pitch" variable rounded and divided by 5.



```
define FReadPitch
set Pitch to 3-axisgyro on board X-Axis angle
set PitchAbsolute to abs of Pitch
set PitchScale to round Pitch / 5
```



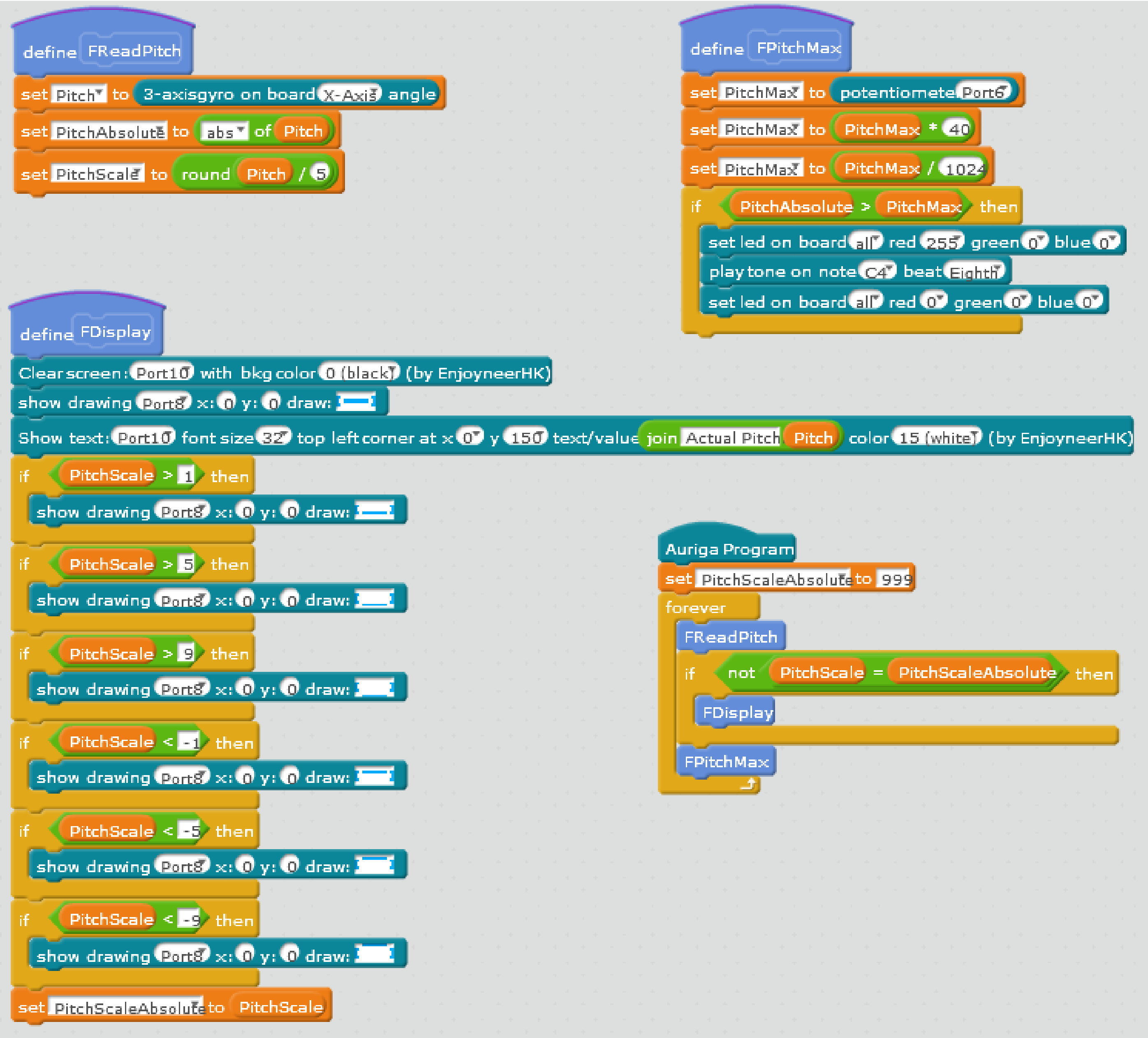
```
define FPitchMax
set PitchMax to potentiomete Port6
set PitchMax to PitchMax * 40
set PitchMax to PitchMax / 1024
if PitchAbsolute > PitchMax then
    set led on board all red 255 green 0 blue 0
    play tone on note C4 beat Eighth
    set led on board all red 0 green 0 blue 0
```

Here, we can see how the "PitchMax" variable collects the value of the potentiometer and sets the maximum value at which the mBot Ranger can incline.

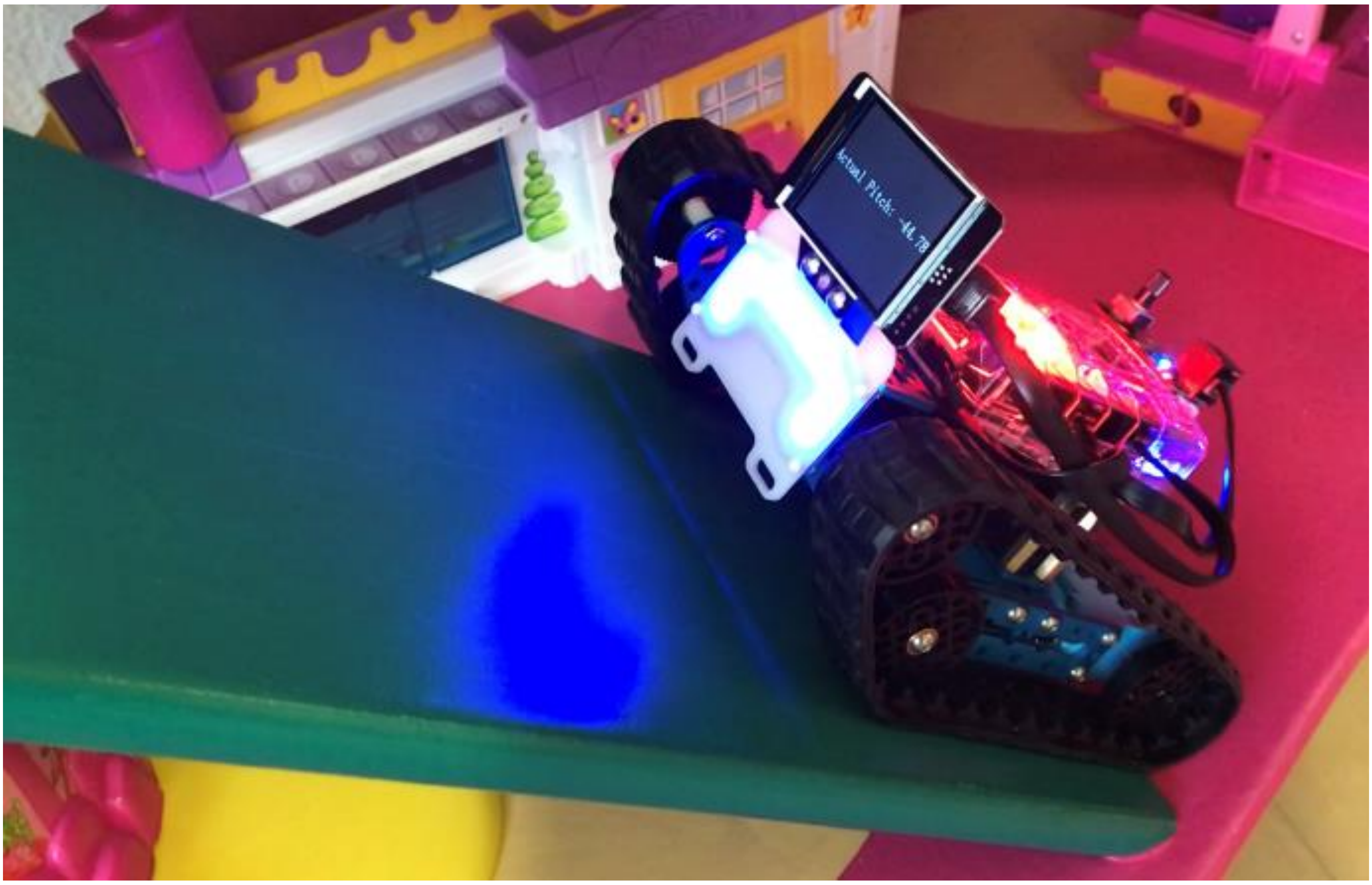
If the mBot exceeds that maximum angle, an acoustic and luminous signal will start to show.

Finally the "FDisplay" function shows in real-time the inclination on the LED Matrix and the Pitch Angle Value in the TFT LCD Screen.

The full version of the program can be found below and in the following pages some photos from the activity are shared.



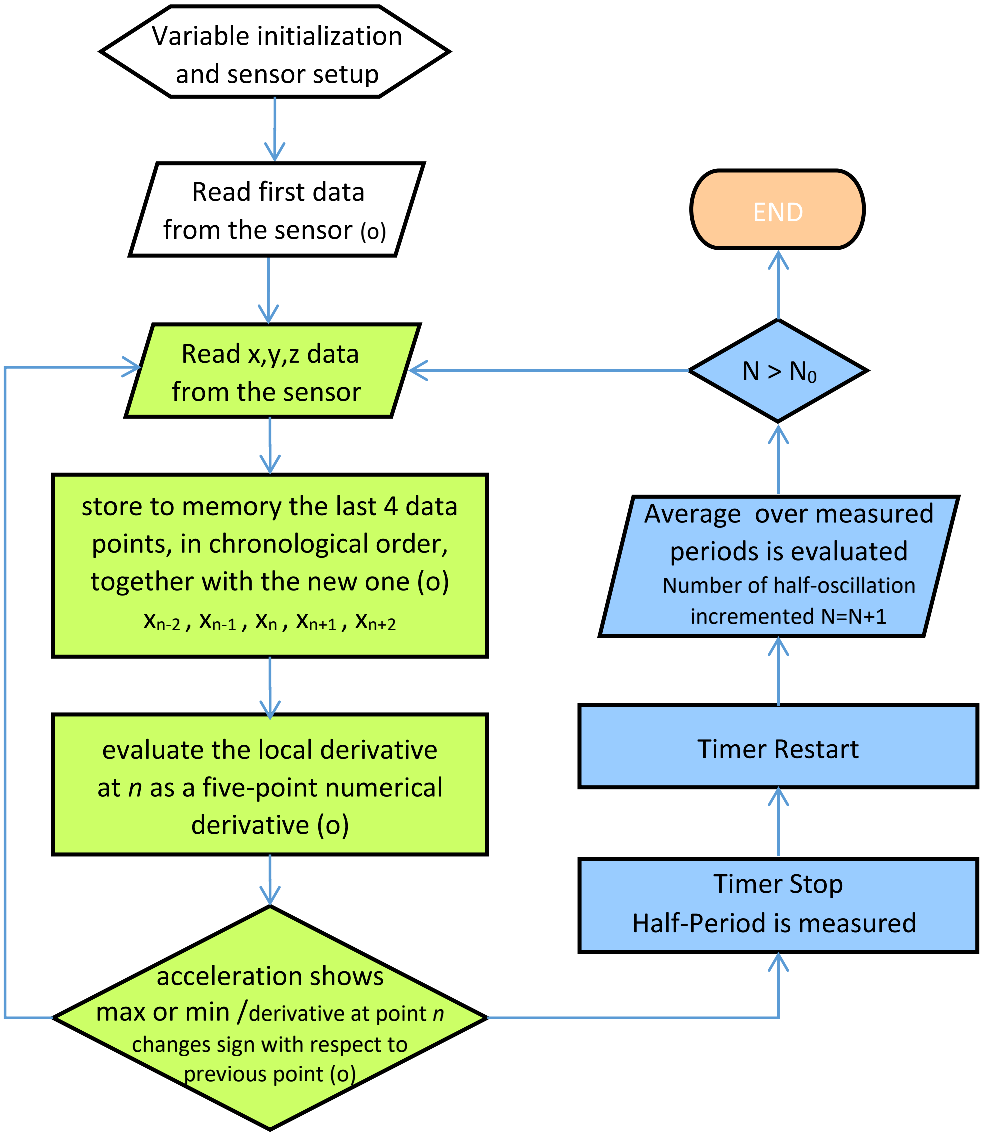
```
define FReadPitch
set Pitch to 3-axisgyro on board X-Axis angle
set PitchAbsolute to abs of Pitch
set PitchScale to round Pitch / 5

define FPitchMax
set PitchMax to potentiomete Port6
set PitchMax to PitchMax * 40
set PitchMax to PitchMax / 1024
if PitchAbsolute > PitchMax then
    set led on board all red 255 green 0 blue 0
    play tone on note C4 beat Eighth
    set led on board all red 0 green 0 blue 0

define FDisplay
Clear screen: Port10 with bkg color 0 (black) (by EnjoyneerHK)
show drawing Port8 x: 0 y: 0 draw:
Show text: Port10 font size 32 top left corner at x 0 y 150 text/value join Actual Pitch Pitch color 15 (white) (by EnjoyneerHK)
if PitchScale > 1 then
    show drawing Port8 x: 0 y: 0 draw:
if PitchScale > 5 then
    show drawing Port8 x: 0 y: 0 draw:
if PitchScale > 9 then
    show drawing Port8 x: 0 y: 0 draw:
if PitchScale < -1 then
    show drawing Port8 x: 0 y: 0 draw:
if PitchScale < -5 then
    show drawing Port8 x: 0 y: 0 draw:
if PitchScale < -9 then
    show drawing Port8 x: 0 y: 0 draw:
set PitchScaleAbsolute to PitchScale

Auriga Program
set PitchScaleAbsolute to 999
forever
    FReadPitch
    if not PitchScale = PitchScaleAbsolute then
        FDisplay
    FPitchMax
```


Actual Pitch: -44.78

Actual Pitch:

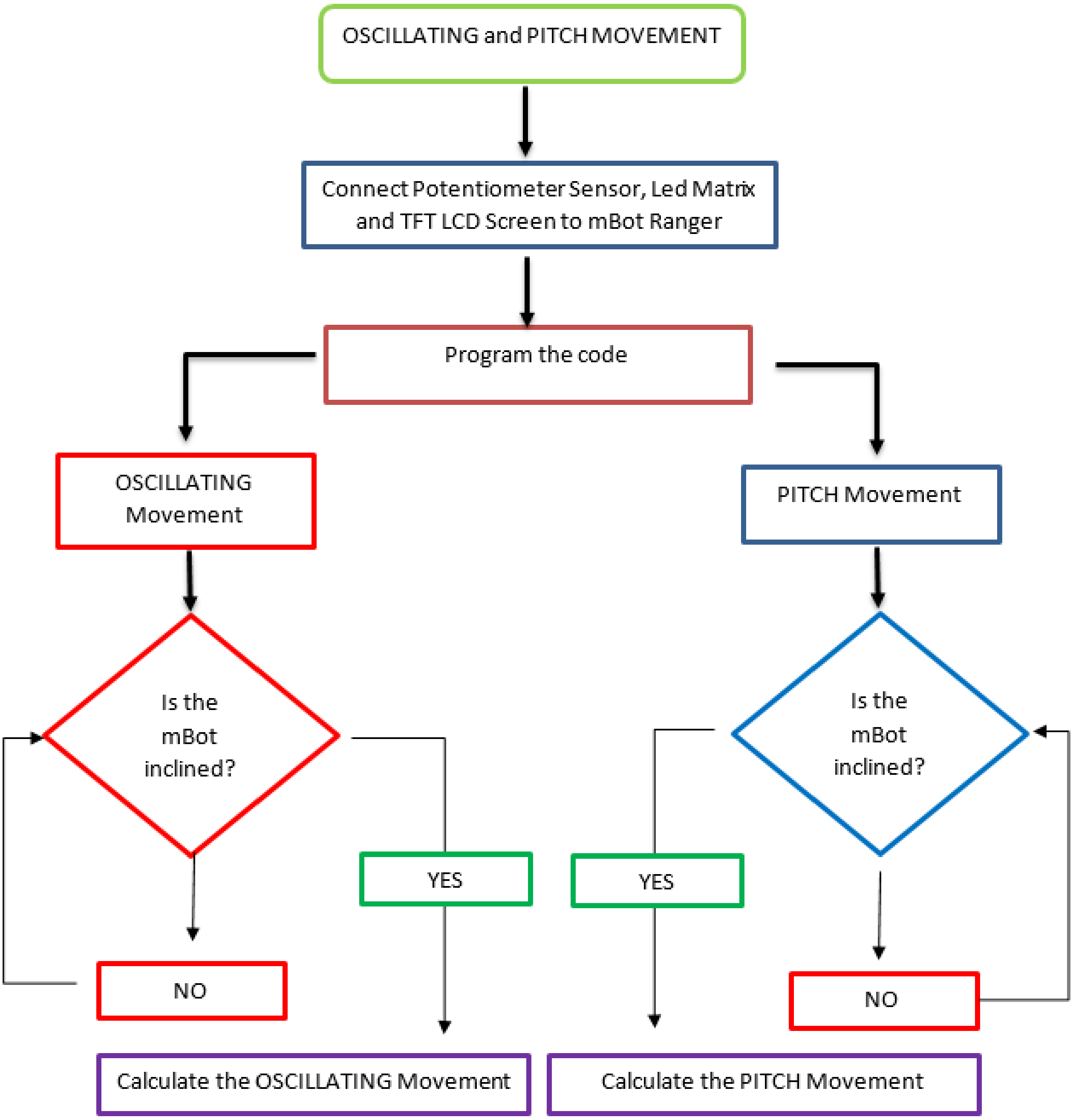
Actual Pitch: 22.33

First version


Variable initialization and sensor setup
Read first data from the sensor (o)
Read x,y,z data from the sensor
store to memory the last 4 data points, in chronological order, together with the new one (o)
x_{n-2} , x_{n-1} , x_n , x_{n+1} , x_{n+2}
evaluate the local derivative at *n* as a five-point numerical derivative (o)
acceleration shows max or min /derivative at point *n* changes sign with respect to previous point (o)
Timer Stop
Half-Period is measured
Timer Restart
Average over measured periods is evaluated
Number of half-oscillation incremented N=N+1
$N > N_0$
END

(o) only for the variable with the largest oscillation

Second version


OSCILLATING and PITCH MOVEMENT
Connect Potentiometer Sensor, Led Matrix and TFT LCD Screen to mBot Ranger
Program the code
OSCILLATING Movement
PITCH Movement
Is the mBot inclined?
Is the mBot inclined?
YES
YES
NO
NO
Calculate the OSCILLATING Movement
Calculate the PITCH Movement

STUDENT'S EVALUATION

Indicators for student evaluation may include:

- Physics: She/He performs laboratory measurements with care and accuracy.
- Physics: She/He is able to compare experimental results to theoretical models.
- Physics: She/He properly applies the relationships among the different parameters in simple exercises about springs.
- Physics: She/He recognizes the physics and maths of harmonic oscillations in other context beside mechanics .

BIBLIOGRAPHY

[1] Sensor Description http://learn.makeblock.com/en/me-3-axis-accelerometer-and-gyro-sensor/

[2] Gyro Sensor class definition (for Arduino/C++)
http://learn.makeblock.com/en/Makeblock-library-for-Arduino/class_me_gyro.html

[3] On five-point numerical derivatives
L.Demeio, Course in numerical analysis, Università Politecnica delle Marche
https://dipmat.univpm.it/~demeio/public/Analisi_Numerica/Lezioni/Derivate.pdf
https://en.wikipedia.org/wiki/Finite_difference_coefficient

SCALABILITY

The activity is suitable for students aged 12 or higher.

With older students (14-15) increasing details can be included and the simulation may be run from Arduino in order to record and plot outputs.

In future developments it could be interesting to perform more “movements of physics” with the mBot.