ECE-GY 6483 REAL TIME EMBEDDED SYSTEMS

Project Report

THE EMBEDDED GYROMETER

“THE NEED FOR SPEED”

Submission Date: December 22, 2021

Submitted by:

SL NO.	NAME	NET ID	NYU ID
1	JASMINE BATRA	JB7854	N18693326
2	NAVYA SRAVANI JAMMALAMADAKA	NSJ9072	N19774260

1.OVERVIEW OF THE PROJECT

The goal of embedded system challenge is to assemble a wearable speedometer which can calculate velocity by estimating angular velocity accessible from our built-in gyroscope(L3GD20) - without a GPS. Our gyroscope can quantify 3-axis angular velocity. Decisively setting the sensor on the legs or feet to calculate the distance travelled by the person and capturing the angular velocity and then converting the angular velocity into linear forward velocity.

2.INTRODUCTION

Gyroscope sensor is a device that can measure and maintain the orientation and angular velocity of an object. These can measure the tilt and lateral orientation of the object whereas accelerometer can only measure the linear motion. Gyroscope sensors are also called as Angular Rate Sensor or Angular Velocity Sensors. These sensors are installed in the applications where the orientation of the object is difficult to sense by humans.

Gyroscopes are built into compasses on ships and aircraft, the steering mechanism in torpedoes, and the guidance systems installed in ballistic missiles and orbiting satellites among other places.

The principle of Gyroscope sensor can be understood by observing the working of Vibration Gyroscope sensor.

Gyroscope Sensors are used for versatile applications. Ring laser Gyros are used in Aircraft and Source shuttles whereas Fiber optic Gyros are used in racecars and motorboats.

The main functions of the Gyroscope Sensor for all the applications are Angular velocity sensing, angle sensing, and control mechanisms. Image blurring in cameras can be compensated by using Gyroscope Sensor-based optical image stabilization system.

3.OVERVIEW OF PROGRAMMING LANGUAGE USED

1) EMBEDDED C: Embedded C programming plays a key role to make the microcontroller run & perform the preferred actions. At present, we normally utilize several electronic devices like mobile phones, washing machines, security systems, refrigerators, digital cameras, etc. The controlling of these embedded devices can be done with the help of an embedded C program.

4. FLOW OF THE CODE

Step1: In main function, Initialize values for chipselect

Step 2: Setup the spi for 8 bit data, high steady state clock, second edge capture, with a 1MHz clock rate.

Step 3: PART-1: READING VALUES FROM GYROSCOPE

- Select the device by setting chip select low.
- Send 0x20, the command to read the CTRL_REG1 register.
- Assign bits for PD, Zen, Yen, Xen.
- Read xl and xh values using for loop.
- Concatenate the values taken from xl and xh by calling xlxhconcatenate() function.
- Scale the data received : -255 to 255.

Step 4: PART 2: CONVERTING ANGULAR VELOCITY TO LINEAR VELOCITY

- Calculate values of Linear Velocity using the formula- Linear velocity= w*r
  [r= radius of leg, or LV= (AV*(3.14/180)*radius of leg)]

Step 5: PART 3: DISTANCE CALCULATED

- Calculate the Distance traveled using formula- Linear velocity*time.

Step 6: Printing Angular Velocity, Linear velocity and Distance travelled for each iteration.

CODE-

```
#include <mbed.h>

SPI spi(PF_9, PF_8, PF_7); // mosi, miso, sclk
DigitalOut chipselect(PC_1);

//The function xlxhconcatenates the xl and xl values and returns the concatenated value
int16_t xlxhconcatenate(int16_t xl, int16_t xh)
{
  int16_t temp;
  temp = 0;
  int16_t z = xl;
  while (xh > 0)
  {

    temp = (temp * 16) + (xh % 16);
    xh = xh / 16;
  }
  while (temp > 0)
  {

    z = (z * 16) + (temp % 16);
    temp = temp / 16;
  }
  return z;
}

//Main function
int main()
{

  // Chip must be deselected
  chipselect = 1;
  int i;
  double linearvelocity[40], Distancetravelled[40];

  // Setup the spi for 8 bit data, high steady state clock,
```

```
// second edge capture, with a 1MHz clock rate
spi.format(8, 3);
spi.frequency(1000000);

double Angularvelocity[40];
int16_t xl[40], xh[40];

while (true)
{

  //PART-1: READING VALUES FROM GYROSCOPE

  // Select the device by seting chip select low
  chipselect = 0;

  // Send 0x8f, the command to read the CTRL_REG1 register
  spi.write(0x20);

  //Assigning bits for PD, Zen, Yen, Xen
  spi.write(0b00001111);
  chipselect = 1;
  chipselect = 0;
  spi.write(0xA8);
  int16_t reg = spi.write(0x00);
  //printf("register = 0x%X\n", reg);

  // read the value to X_h and x_l
  // Send a dummy byte to receive the contents of the CTRL_REG1 register
  // chipselect = 1;
  // chipselect = 0;
  // spi.write(0b10101000);

  for (int i = 0; i < 40; i++)
  {
    chipselect = 1;
    chipselect = 0;
    spi.write(0b10101000);
    xl[i] = spi.write(0x00);
    //printf("xl[i] = %d \n",xl[i]);
    chipselect = 1;
```

```
  chipselect = 0;
  spi.write(0b10101001);
  xh[i] = spi.write(0x00);

  //PART 2: CONVERTING ANGULAR VELOCITY TO LINEAR VELOCITY

  Angularvelocity[i] = xlxhconcatenate(xl[i], xh[i]);
  Angularvelocity[i] = ((double)Angularvelocity[i] * 0.00747680664);
  printf("Angularvelocity values= %lf\n ", Angularvelocity[i]);

  //PART 3: CALCULATING THE LINEAR VELOCITY

  linearvelocity[i] = (((Angularvelocity[i] * 3.14) / 180) * 83);
  printf("Linear velocity values= %lf \n ", linearvelocity[i]);

  //PART 4: CALCULATING THE DISTANCE TRAVELLED

  Distancetravelled[i] = (((double)linearvelocity[i]) * 0.5);

  float sum = 0;
  for (int k = 0; k < 40; k++)
  {
    sum += Distancetravelled[k];
  }

  printf("Distance= %lf \n ", sum);

  wait_us(500000);
  printf("%d\n\n", i);
}
```

OUTPUT :

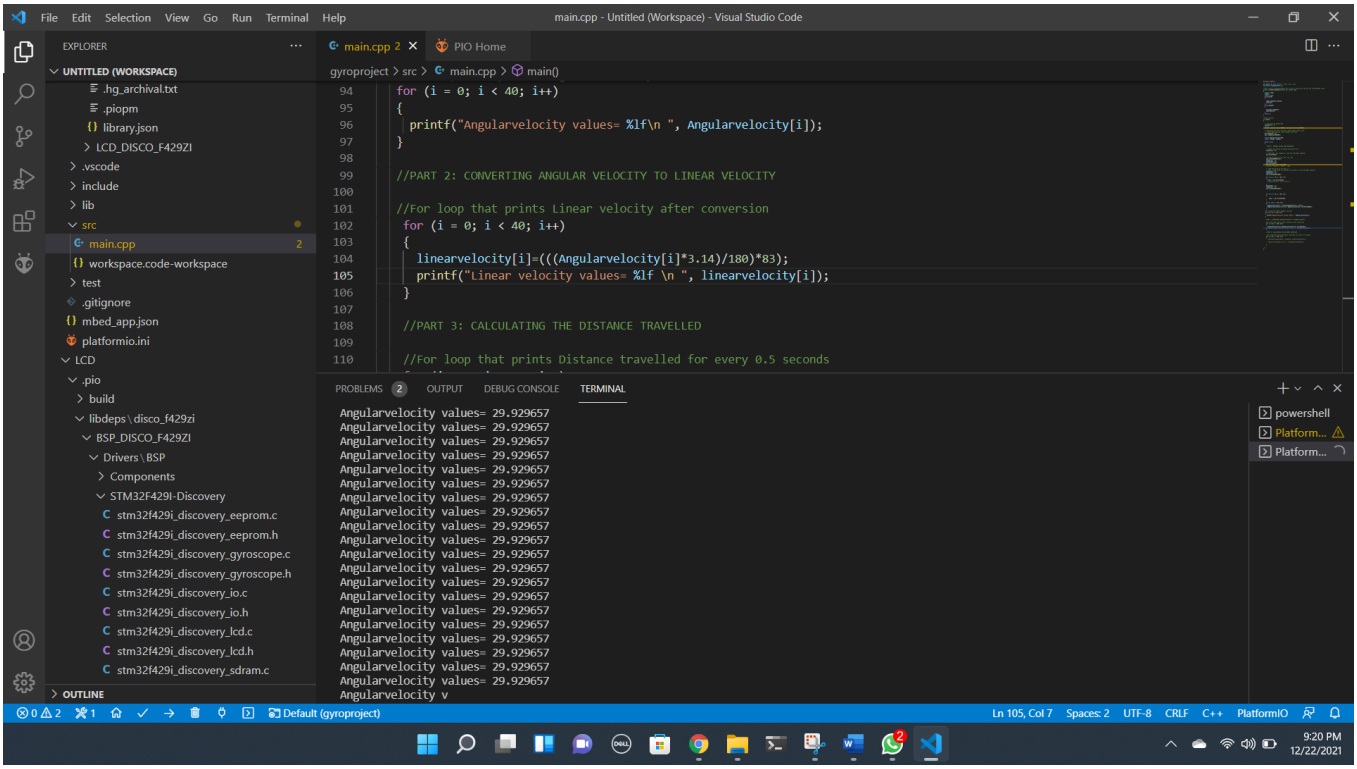


Fig 1: Values of Angular Velocity

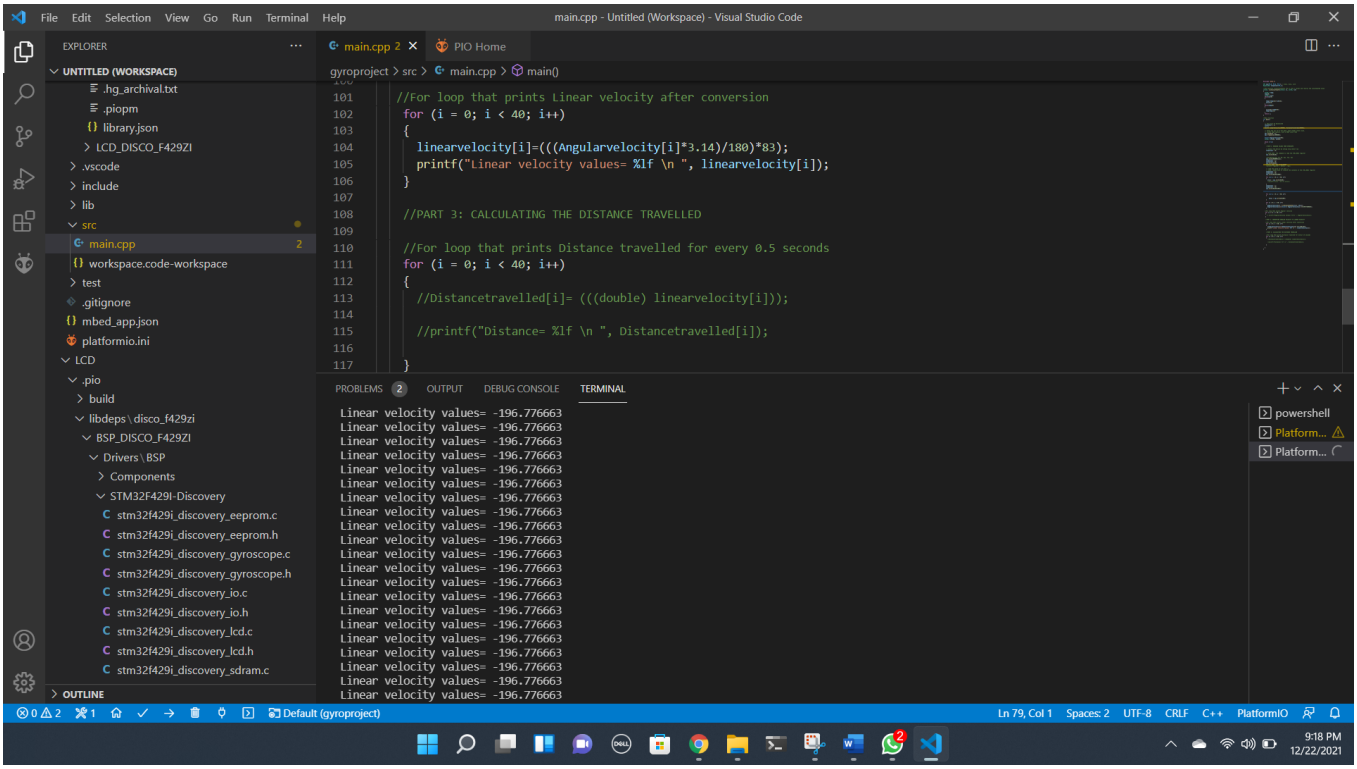


Fig 2: Values of Linear Velocity

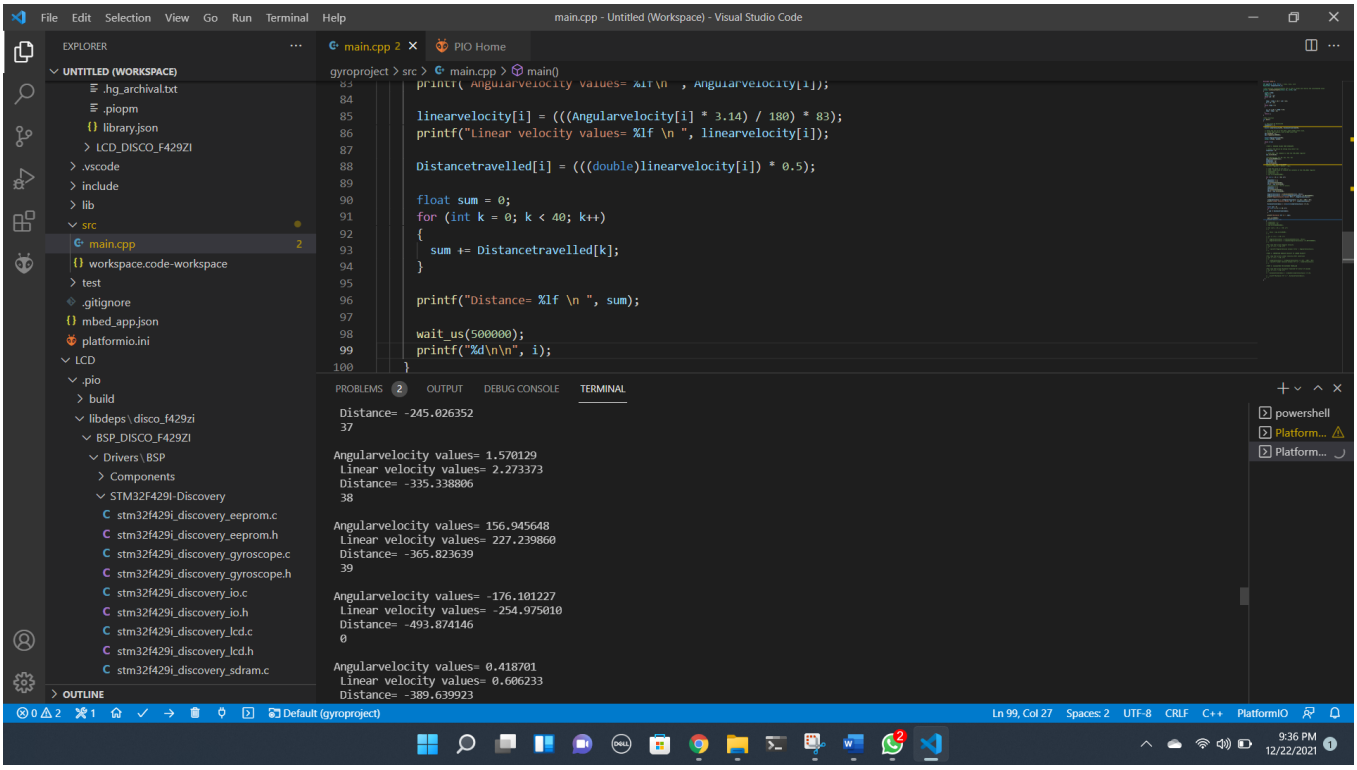


Fig 3: Values of Angular velocity, Linear Velocity and Distance Travelled for each iteration

CODE USING LCD INTERFACING

```
//Header Files
#include <mbed.h>
#include <LCD_DISCO_F429ZI.h>
#include "stm32f429i_discovery_lcd.h"
**********************************************************************************
***
//Interfacing LCD
LCD_DISCO_F429ZI lcd;
**********************************************************************************
***
SPI spi(PF_9, PF_8, PF_7); // mosi, miso, sclk
DigitalOut chipselect(PC_1);
**********************************************************************************
***
//The function xlxhconcatenates the xl and xl values and returns the concatenated value
int16_t xlxhconcatenate(int16_t xl, int16_t xh)
{
  int16_t temp;
  temp=0;
```

```
  int16_t z=xl;
  while(xh>0)
  {

    temp=(temp*16)+(xh%16);
    xh=xh/16;
  }
  while(temp>0)
  {

    z=(z*16)+(temp%16);
    temp=temp/16;
  }
  return z;
}
*****************************************************************************
***
//Main function
int main()
{

  // Chip must be deselected
  chipselect = 1;
  int i;
  uint8_t n[30];
  double linearvelocity[100000], Distancetravelled[100000];

  // Setup the spi for 8 bit data, high steady state clock,
  // second edge capture, with a 1MHz clock rate
  spi.format(8, 3);
  spi.frequency(1000000);

  double Angularvelocity[40];
  int16_t xl[40], xh[40];
  while (true)
  {
*****************************************************************************
***
```

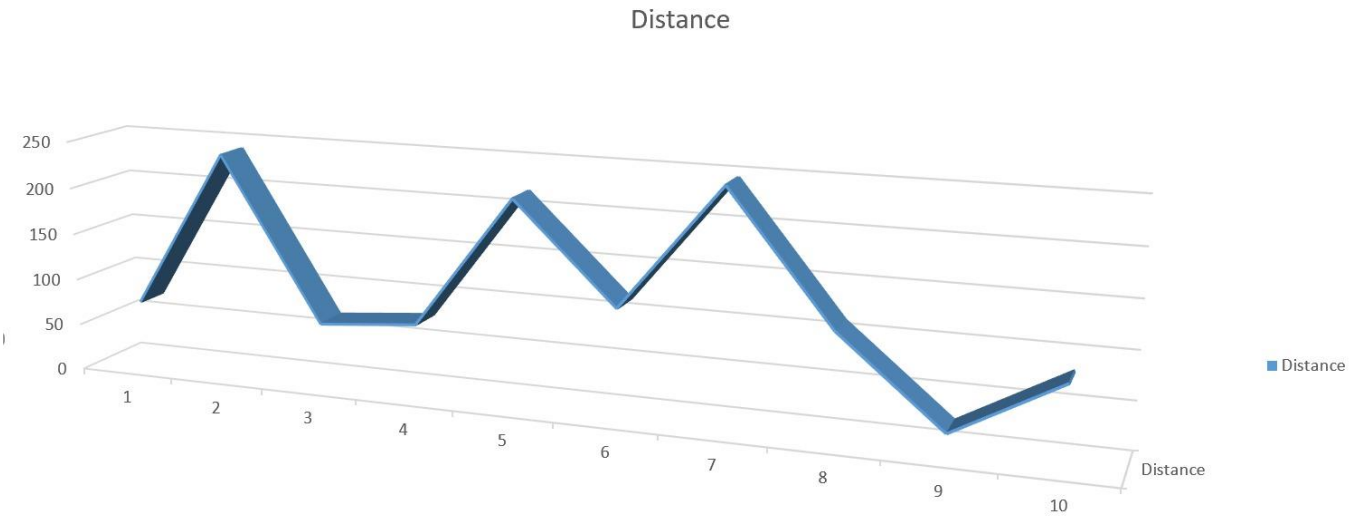
```
//PART-1: READING VALUES FROM GYROSCOPE

  // Select the device by seting chip select low
  chipselect = 0;

  // Send 0x20, the command to read the CTRL_REG1 register
  spi.write(0x20);

  //Assigning bits for PD, Zen, Yen, Xen
  spi.write(0b00001111);
  chipselect = 1;
  chipselect = 0;
  spi.write(0xA8);
  int16_t reg = spi.write(0x00);
  //printf("register = 0x%X\n", reg);


  for (int j = 0; j < 40; j++)
  {

  chipselect = 1;
  chipselect = 0;
  spi.write(0b10101001);
  xh[j] = spi.write(0x00);
    //printf("xh[j] = %d \n",xh[j]);
  chipselect = 1;
  chipselect = 0;
  spi.write(0b10101000);
  xl[j] = spi.write(0x00);
    wait_us(500000);
  }

  for (i = 0; i < 40; i++)
  {
    Angularvelocity[i] = xlxhconcatenate(xl[i], xh[i]);
    Angularvelocity[i]=((double) Angularvelocity[i] *0.00747680664);
    //printf("Angularvelocity values= %lf\n ", Angularvelocity[i]);
  }
```

```
*********************************************************************************
***

 //PART 2: CONVERTING ANGULAR VELOCITY TO LINEAR VELOCITY

  //For loop that prints Linear velocity after conversion
   for (i = 0; i < 40; i++)
   {
    linearvelocity[i]=(((Angularvelocity[i]*3.14)/180)*83);
   // printf("Linear velocity values= %lf \n ", linearvelocity[i]);
   }
*********************************************************************************
***

   //PART 3: CALCULATING THE DISTANCE TRAVELLED

   //For loop that prints Distance travelled for every 0.5 seconds
   for (i = 0; i < 40; i++)
   {
    Distancetravelled[i]=((double)(linearvelocity[i])*0.5);
    //printf("Distance= %lf \n ", Distancetravelled[i]);
   }
*********************************************************************************
***
   // PRINTING DISTANCE TRAVLLED VALUES ON THE LCD
   for(i=0;i<40;i++)
   {
    sprintf((char*)n, "Linearvelocity = %lf \n  ", linearvelocity[i] );
    lcd.DisplayStringAt(0, LINE(0), (uint8_t *)&n, LEFT_MODE);
   }

  }
}
```

5. LESSONS LEARNED

- This project can be used in real life scenarios where it can deal with problems like calculating velocity of a cricket ball or football to understand and evaluate player's performance.
- It can be used in smart gadgets like Fitbit to calculate the distance traveled by a person on a single day.

PLOT: The plot explains the distance(in metres) travelled basing upon different walking speed.



	1	2	3	4	5	6	7	8	9	10
Distance	71.8	240.8	67.4	77.08	217.27	114.37	245.81	112.122	22.037	83.3