ECE-GY 6483- Real Time Embedded Systems

EMBEDDED CHALLENGE:

The Embedded Gyrometer "The Need for Speed"

Group 14

TEAM MEMBERS:

Name: Sriram Narayan Koushik Chitrapu (ID:sc9948)

Name: Weining Wu (ID:ww2644)

Name: Daichong Meng (ID:dm5449)

Name: Devashish Gawde (ID:dg4015)

TABLE OF CONTENTS

1 INTRODUCTION

In the past decade we have seen an explosion of wearable health devices ranging from heart rate sensors to step counters to distance trackers. These devices are designed to help us meet our fitness goals and help us keep in good physical shape.

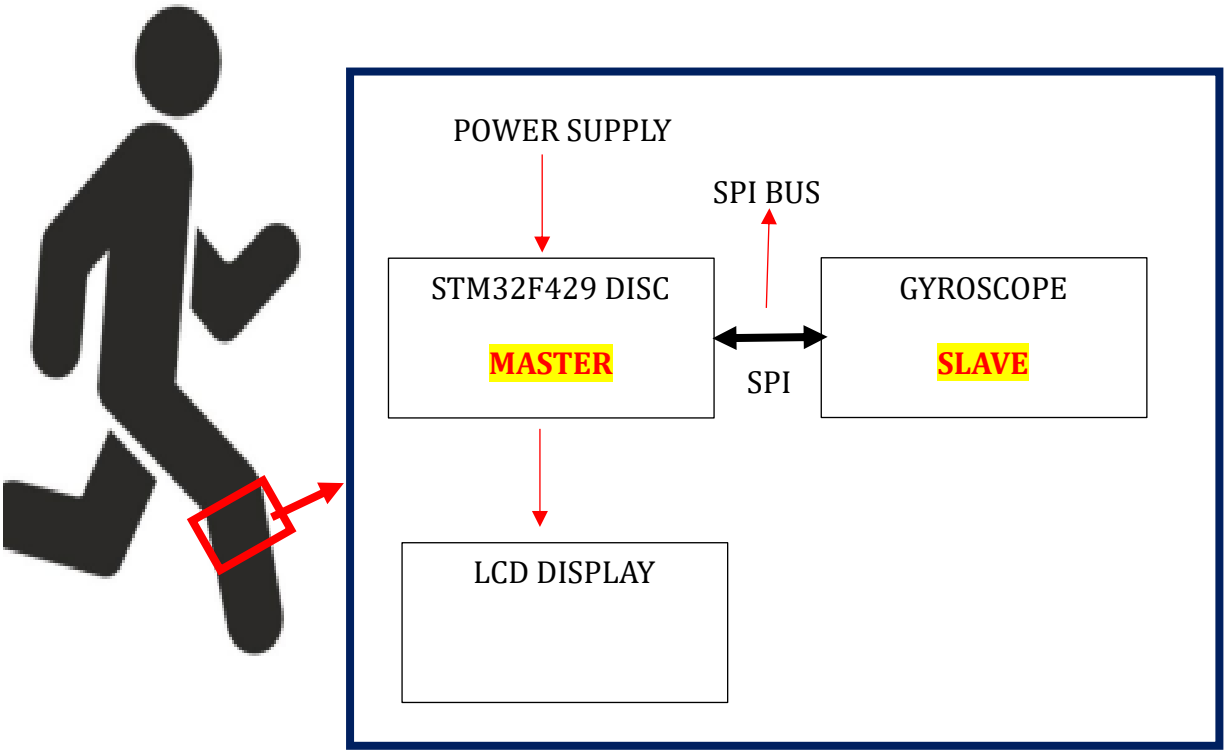
The objective of this semester's embedded challenge is to build an innovative **"Wearable Speedometer"** designed to measure the angular velocity and calculate the distance travelled using the onboard Gyroscope(L3GD20) instead of the conventional way of estimation via GPS or accelerometer. The 3-axis angular velocities are measured based on the sensitivity of the gyroscope when a person wears the board (STM32F429 DISCOVERY) around his knee to record the movement. This device is ideal for fitness enthusiasts, and anyone interested in tracking their movement efficiently and accurately.

2 DESIGN DETAILS:

2.1 REQUIREMENTS:

SR.NO	DESCRIPTION	DETAIL
1.	EDITOR	VS CODE WITH PLATFORM IO Extension installed
2.	LANGUAGE	EMBEDDED C/C++ (Along with Compiler Installed)
3.	BOARD	STM32F429 DISCOVERY with LCD Display (Drivers present in the src folder of submission)
4.	HAL	MBED
5.	GYROSCOPE	I3G4250D (Onboard)
5.	POWER SUPPLY	POWERBANK/ **LAPTOP(USED)**

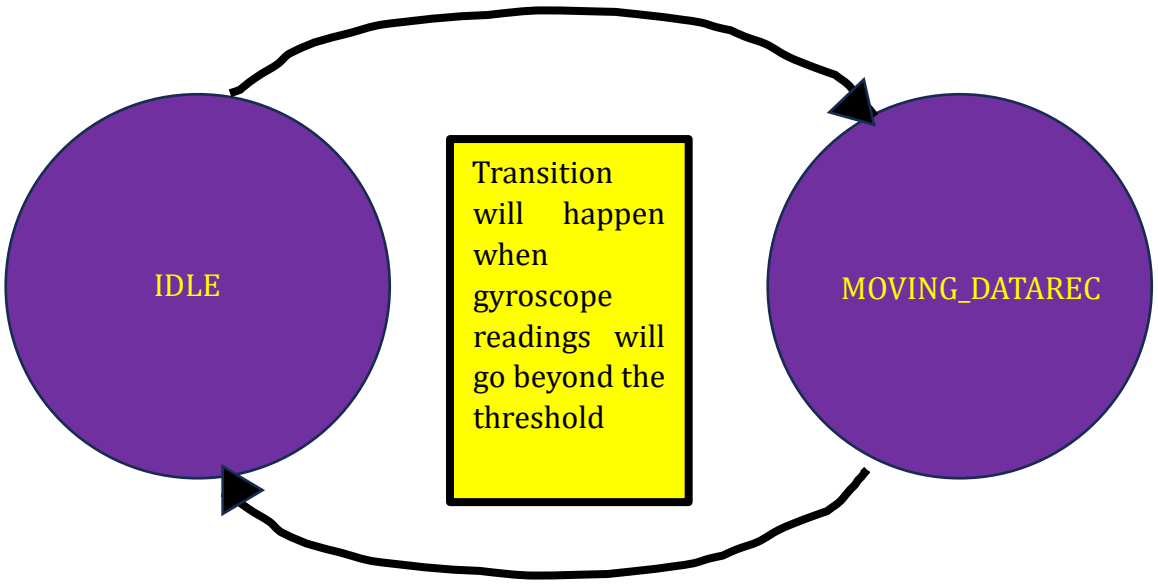
2.2 BLOCK DIAGRAM:



2.3 FINITE STATE MACHINE:

Implemented in the main() Function:

On Interrupt, Gyroscope Reading> Threshold



Post calculation, return back to IDLE state (if less than 20 sec)

2.4 DESIGN STEPS & IMPLEMENTATION

- Include all relevant libraries. Some of them are as follows:

```
#include <mbed.h>
#include <stdio.h>
#include <fstream>
#include <iostream>
#include <math.h>
#include <chrono>
#include "stm32f4xx_hal.h"
#include "drivers/LCD_DISCO_F429ZI.h"
#include <stdlib.h>
#include <float.h>
```

- Define the relevant registers, variables for the gyroscope, spi , filter and lcd.
- Implement the required functions related to the LCD, Gyroscope to get the data, calculate the distance and display the results on the terminal.

Main():

- Interface the gyroscope using the SPI Protocol to capture the velocities across the X, Y , Z Dimensions.
- Call the functions like getGyroData() to input the velocity readings from the gyroscope.
- With the help of the state machine implemented, the state will transition from IDLE to MOVING_DATAREC in order to perform the calculations at calculateDist3Dim() function, controlled with the help of the threshold(GYRO_THRESHOLD) value selected.
- Display the calculated results(Linear Distance and the StepCnt) across the LCD Screen at respective state transitions controlled with the help of the threshold value selected.

We use LCD_DISCO_F429ZI.h library to implement the LCD output.

- Initial_ScreenDisp function: show the welcome screen
- CALC_ScreenDisp function: show the calculating screen

- CALC_Final_ScreenDisp function: show the final and fixed screen to display the results and hints to restart.

For GetGyroData( ):

- Record those values for a period of 20 second duration (Stored in an array Variable) by the help of a resetTimer Object in the while loop.
- Sample the velocity values at every 0.5 s in synchronization to an interrupt event with the help of a Ticker Object.
- The data is filtered with the help of the moving average filter having a WINDOW_SIZE of 6.
- With the help of the scaling factor declared in the beginning of the code, convert the filtered velocity, and multiply it with the radius of the leg (assumed it to be around 0.5) and time to get the linear length.

$$\omega = \frac{\pi}{180} \cdot \frac{\omega_g}{1000}$$

$$Linear_{Length}\ (in\ m) = Velocity * (Radius = 0.5) * (Time = 0.5\ sec)$$

calculateDist3Dim( ):

- Calculate the overall distance travelled during the 20 seconds of measurements with the help the distance formula

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

- We can consider a single dimension measurement (in our case Z, as the values were changing in case of a forward movement, observed across the serial monitor while printing) in order to find the distance travelled but as the objective stated and also to have a good accuracy, we have used the above formula for calculation.
- Stepcnt>Threshold, Then increment the stepcnt by 1

Note:

- We have used the onboard black reset button to reset back the current execution and hence we have not implemented separate logic to start/reset the calculations.
- A zip file along with a readMe file has been added in our submission, Kindly refer it for the execution procedure.

3 CHALLENGES:

Main Challenges observed are:

- Gyroscope Sensitivity Calibration for the physical changes in minor movements or operating conditions like vibrations, temperatures etc.
- Filtering the data captured by the gyroscope.
- Conversion calculations and calibrations.
- Setting up of the threshold values to have a control on the state movement/transition.

Gyroscopes can produce noisy data due to factors like temperature changes or mechanical vibrations. Hence, when developing the product, we spent a lot of time testing and adjusting the filters parameters, gyroscope sensitivities to make sure that the result of our measurements are quite consistent with that of the real distance. We overcame this by implementing filters including the moving average filter and thresholding.

4 FUTURE IMPROVEMENTS

There are more features that can be developed with this product. Here are some possible ways forward:

- **Power supply:** We can equip this device with its own power supply, as well as a strap for it to be attached to the user's body, so it can be a portable device.
- **Button Interfacing:** Adding a separate button either to reset the execution completely or start the execution (which we believed might not be a good design choice as the user might think it's extra task for him to do it every time).
- **Wireless connectivity:** We can add integrated wireless communication modules such as Bluetooth or Wi-Fi. This will allow remote data monitoring and control and enable IoT capabilities.
- **Data logging:** We can have the ability to log data over time. This may include writing the data to an SD card or sending it to a connected cloud server for further analysis.