Embedded Systems Project

Documentation

Husnain Khan/ ITE104308

7/3/2022

Semester 5

Tale of Contents

1. Introduction 3

a. Project description 3

Hardware 3

Software 3

Basic Features 3

Extended Features 3

b. Project motivations 4

2. User manual 4

Playing Field 4

Starting the game 4

Avoiding the walls 4

Score counting 5

3. Project analysis 6

Modularization 6

Abstract / algorithmic description 7

Analyzation of alternative approaches 8

4. Implementation 9

Specific description 9

Memory Mapping approach 9

App module 9

Timer Module 10

MPU6050 Module 10

Dem128064B Module 10

Software structure 13

Schematic 13

5. Test 14

Specific test cases 14

Expected results 14

6. Conclusion 14

1. Introduction

The project is the application of STM32 microcontroller with external sensor (MPU) and an output module (LCD). A prototype game is created using STM32F401RE microcontroller in which Sensor interfacing (MPU 6050) and LCD interfacing (DEM 128064B) is performed. Motion of a ball (3x3) pixels on the LCD is controlled using the MPU Sensor.

a. Project description

Hardware

1. Sensors: MPU 6050 sensor.
2. Microcontroller: STM32 F401RE
3. Output: DEM 128064B
4. Additional: Jumper Wires

Software

1. STM32 Cube IDE
2. HTerm

Basic Features

Basic features of the project include two moving vertical lines on the LCD display module, one from the top and one from the bottom and a ball of 3x3 pixels on displace LCD module. The movement of the ball is controlled by the tilt of MPU 6050 Sensor. When the ball is collided with one of these lines the game is over. When project is started a welcome message is shown for a period of time. The top vertical line starts to move from right side in the left direction, at the same time lower line is introduced the speed of the lower line is less then upper line. There will be a point when user won't have space to pass the ball between both lines. User has to avoid such a situation otherwise the ball can hit the line. The ball is limited to stay between the field, which is 64x128 means the ball can't cross the field. When both of these lines reach one by one to the left most side and cross it, they will appear again from the right side.

Extended Features

The speed of the ball is controlled by the tilt angle, while the speed of the lines movement on the LCD depends upon the current level. There will be different levels in which this game can be played. The level increase after scoring certain points. Every single time the lines crosses the boundary from left side the score is counted. At the end when game is over and a message is shown to the user which includes score of the user.

b. Project motivations

The main motivation for this project was to expand my knowledge of Embedded systems especially creating a functional product for the user, building drive for the MPU sensor and LCD. This allowed me to gain more exposure working with complex code and allowed me to get hands-on-practice to work better on similar projects.

2. User manual

Playing Field

The playing field is 64 x 128. It has a ball of size 3 x 3 is shown on the screen. Two vertical lines appears from the right most side of the screen and disappear when these lines cross the left most side.

Starting the game

To start the program, connect the cable with a source. After that a welcome message will be shown to the user on the screen for certain time. User can start the game after the message is gone. A ball is shown on the left most side of the screen, user can control the movement and speed of the ball using the sensor. If the sensor is tilted to the right side the ball will start to move towards the right side, approaching the lines. The speed of the ball can be controlled by the tilt angle. If tilt angle is increased the ball will move faster.

Two lines from the right most side (one line from the top and one from the bottom) of the screen will start to move towards the left side where the ball is placed in the beginning. The speed of the top line is greater than the lower line.

Avoiding the walls

User has to control the movement of the ball using the sensor such that the collision of the ball with walls is avoided. If the ball is collided with any of the wall then game play will stop and game over message will be shown to the user for certain time.

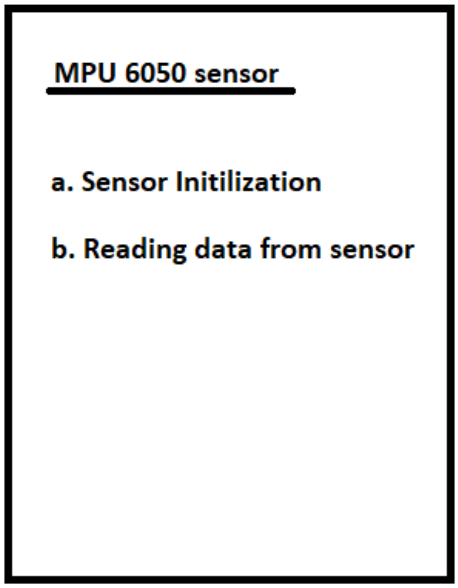
Score counting

After the game is ended, the points are shown to the user. Points are counted every single time when any of the line crosses the right side of the screen

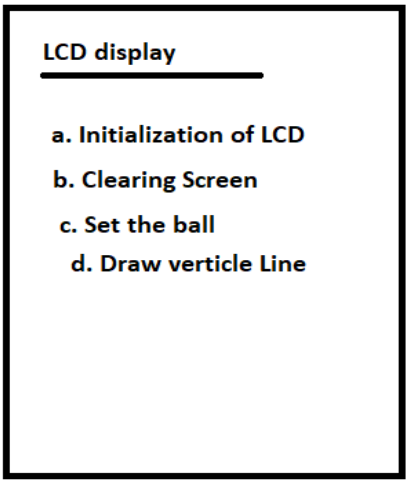
3. Project analysis

Modularization

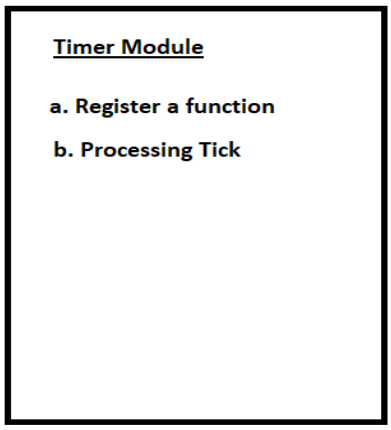
- **Sensor:** A separate driver module is required for the sensor. The driver must initialize the sensor so a function for the initialization is required. Driver should also be able to read the data from the sensor for that a separate function is created.



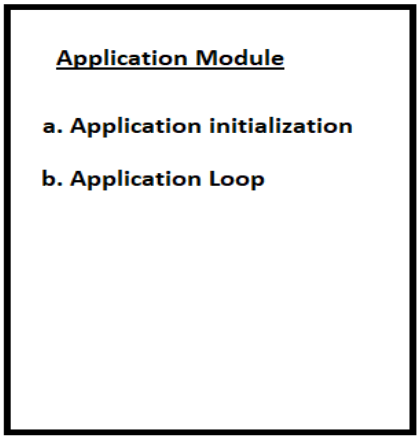
- **LCD display:** A module specifically for the LCD display is created in which initialization of the LCD (a driver) is done. It is also required to clear the screen at some point so a clearing function is helpful to clear the content from the screen. A virtual ball along with 2 vertical lines are displayed during the game play so a function to set the ball to a certain position, and set vertical line on a position is also required. When the ball is moved to new position previous position is to be cleared so there should be a function to remove the ball from given position. LCD module is more specific to the game.



- **Timer Module:** Timer module should have a timer register function which is to be used to register the function along with the frequency, it should also contain a function to process every single tick.



- **Application Module:** Which should contain two function, one is to be used for all initialization, for example initialization of the sensor, LCD and for registering the functions to the time. Another function is to be called in a loop.



Abstract / algorithmic description.

- **Memory mapping approach**

  In this approach, the content on the screen are stored locally in. The content to be displayed on the screen (Vertical lines, Ball position etc.) are stored in a local buffer (2d array of size same as LCD) is mapped. When something is changed on a local buffer it has to reflect on the LCD display. A refresh function is required to directly map the content store locally onto the LCD display.

- **Storing the content on the screen in a 2d Array**

  One option to store the content locally is to use a 2d array of int. The content on the screen to be displayed are locally stored in a 2d array and is directly mapped to the LCD.

- **Storing the position of the ball and position of the Line in a struct**

  Another option to store the content of the screen locally is to store the position of the ball and position of both moving lines separately and every single time the position of the ball or the position of one of the lines is changed it is reflected back to the screen by refreshing.

- **Bidirectional communication**

  In this approach, read and write operations are performed simultaneously, the content on the screen are read, analyzed and modified accordingly what is to be written on the screen. With this approach it is not required to store anything locally.

Analyzation of alternative approaches

- **Comparison of Bidirectional and Memory mapping approach**

  In Bidirectional communication read and write operation is performed at the same time. The content on the screen (LCD) is read. Every single time the ball is moved to next position. The previous position of the ball is to be cleared (means whole filed of the ball 3 x 3) from the screen, same is for the vertical lines, previous line is cleared from the screen. It gets tricky when the ball is moved backward and vertical line is moving towards the wall, it is possible when the previous ball position is cleared some pixels of the line get effected to handle this problem additional checks are required.

  With memory mapping approach every single pixel which is to be displayed on the screen is stored locally and is reflected back when ball position or vertical line position is updated. With this approach a lot of RAM is required as everything needs to be stored locally so this is not a problem in our case the microcontroller which is used has enough RAM to deal with this problem. The advantage of using memory mapping approach is it only requires one function to set the content which is locally stored on the screen. It's much easier to write the text on the LCD by simply modifying the locally stored buffer and refreshing the content on the screen.

- **Comparison of Storing the content in a 2d Array with Storing the Position of the Ball and Position of the Line in a Struct**

  The contents can be stored in 2d array of int and it is also possible to store the position of the ball and only the position of the line. The 2d array is a better option in our case because it is much easier to write content on the 2d array and that array will reflect the content on the screen, especially when writing some message to the user.

4. Implementation

Specific description

Memory Mapping approach

Memory mapping approach is implemented in which a 2d array of int is created inside Dem128064B module which is used to store the content locally in RAM.

Using this approach, collision detection, moving the ball to new position and moving vertical lines is much easier because it is simple to remove the ball and moving line from previous position and simply setting it to new position, only display array needs to be modified to change.

App module

App module has two functions. One of the functions (app_init) is used for the initialization of all the modules used in this project like MPU6050 sensor, Dem128064B LCD Module at first then the registration certain functions take place inside using Timer module. Only those functions are registered inside the timer module those are called after a certain period. A welcome message is also a part of initialization which is shown to the user before game is started.

App module has another function(app_loop) which take no parameters and is called in the main loop. It has to check if the game is over(gameEnd). This function checks the gameEnd flag continuously, the gameEnd flag is only set when collision is detected. When gameEnd flag is set the whole game is reset, the screen is cleared, ball is set to the starting position, both of the lines are set to the right most side of the screen so these lines can start from the starting position, a game over message is shown to the user for certain time period after that gameEnd flag is reset and user can start playing the game again.

App module has a function for collision detection which is used to detect the collision for upper and lower line with the ball.

- **Calculating the ball position using the tilt angle**

  The movement of the ball on the screen and rate at which ball moves in all four direction is to be dependent upon the tile angle of the sensor. For that new position of the ball is calculated which is dependent upon the tilt angle. The new position is calculated by adding the tilt angle divided by 90 which is maximum angle to the previous position. This calculated new position will be multiple of 10 because the tilt angle is multiple of 10. So if the previous position is 10 and the new calculated position is 13 or less then 20 then the ball will stay at the previous position but for the next iteration new position is calculated by the previously stored position which is 13 and if the new position value reaches to 20 then the ball will be moved to next position and is cleared from the previous position, new position will be copied to the previous position to be used for next iteration. For example let's say previous position is 10 which is stored and the tilt angle at this point is 300 which is actually 30 degree, so to calculate the new position of the ball previous position is added with the tilt angle divided by 90 which will be 10 + 300/90 the new position will be 13, so this value is 1 if it is divided by 10 that means ball will not move to the next position, now considering the previously stored position which is 13 and till angle is 600 then the new position will be 13 + 600/90 which is

equal to 19 approximately, which is less then 20 but for the next iteration if the tilt angle is same as previously 600 then the new position will be more than 20. Then the ball will be moved to the next position and is to be cleared from the previous position. Same process is done for row and columns of the ball.

Timer Module

Timer Module can be used to multiplex a single hardware timer to multiple software timers I.e. periodic function calls. The number of software timers is limited to TIMER_MAX_TIMERS. Software timers (callbacks) can be registered using a timer_register function which takes two parameters one is for the function to be registered and other is for the clock divider, timer_tick is called inside the hardware timer (SysTick_Handler) to process every single tick which is of 1ms.

MPU6050 Module

MPU6050 module is used for getting tilt angles from the sensor. The data which is read from the sensor is in the form of acceleration in three axis x, y and z this acceleration is then converted to tilt angle and is stored in a structure of angles having two angles thetaX and thetaY. But before reading data from the MPU6050 the initialization is required which is done separately in a function called mpu6050_init which is used to initialize mpu6050 to read the acceleration in 3 axis. For initialization HAL_I2C_Mem_Write function is used which takes device address (7 bits address value in datasheet must be shifted to the left before calling the interface) MemAddress Internal memory address in our case it is defined PWR_MAGT_1_REG, memory address size and data to be written for the configuration. The return type is also checked if the return is not HAL_OK then the function will return 0. After that same function is used to configure Accelerometer for 2g value, the register address is also defined (ACCEL_CONFIG_REG) in the header file.

Dem128064B Module

Dem128064B module is used for the LCD display Dem128064B used in this project. The initialization of the LCD is done inside the function setUp. The implementation of the LCD module is more specific to this project in which memory mapping approach is implemented, for the position of the ball on the screen a struct type is used (previousPos) in which two int types store the exact position of the ball(prev_Row is for the row position of ball on LCD and prev_Col is for the col position of the ball on LCD). The LCD is connected to the PORTC of the microcontroller. All the pins used for this module are initialized in the header file of this module.

- **Division of the LCD display**

  The size of the LCD display is 64 x 128. Rows are further divided into pages so there are total 8 pages every page has 8 pixels. Columns are divided into two parts one is the left part of the screen and one is right part of the screen both of these parts have 64 columns. Data is written on the whole page. For that page and column is selected first and then data is written on the selected page and column.

- **Write on Screen**

  A function inside the module which is used to write the data on the screen. This function takes the page number, column address and data as an input parameter. To write data on the screen there is a protocol which is to be followed. First the screen should be turned on for

that a function(turn_on_Screen) is defined inside the module is to be called, then display on off register is set by calling(display_ON_OFF_Reg) function in which D/I and R/W values to be set to 0. After that it is to be check if the col address which is passed to this function is less than the maximum columns if that's the case then left side of the screen is used by calling set_Left_Side_Screen function else right side of the screen for that set_Right_Side_Screen function is defined. After setting the side of the screen the enable should be toggled by calling function toggle_Enable_Lcd which fist set the enable wait for 5 microseconds by calling getDelay function and the reset the enable. A function is called to set the page on which data is to be written, toggle enable is called again after selecting the page number, then column address is set by calling set_ColAddress which takes column address as a parameter, toggle enable is called again. After following these steps now data is set on the screen by calling the function send_Data which takes data as a parameter, control bus is to be set to write and toggle enable to execute.

- **Store the content on the LCD locally in a 2d Array:**

  A 2d array(display) is used to store the pixels of the screen in a RAM. This array is the exact copy of the LCD. When something is to written on the LCD display is first written into this display and is reflected back to the screen. For example, if the ball position is changed then the ball is first cleared from the previous position and ball is set to new position on this array, then refresh function is called which is used to write the content of the 2d array to LCD.

- **Collision detection:**

  Collision detection is done for both lines (upper line and lower line) with the ball, for that two separate functions (ballIsOnLine, ballIsOnLowerLine) are implemented for lower and upper line, collision detection is done every single time when line is moved or the ball is moved to the next position. The line column position is passed for collision detection then it is to be compared with the column of the ball if the column of the ball and line column position is equal then the row of the ball is compared with the length of the moving line (LEN_OF_MOVING_LINE) if both of these conditions are satisfied then ball is on line, 0 is returned for that else 1 is returned.

- **Refresh the screen:**

  Refresh screen is a function inside this module which is used to set pixels state which is stored inside 2d array named as display onto the screen. Write on screen function is used inside this function to write the data.

  Three loops are used to iterate over the display array, the outer most loop is for the column, the second inner loop is for the page, and the inner most and final the loop is for the row inside the page.

  Every page has 8 rows in it, so to find the current position of the row in the page a variable is created to save the product of current page number and page length(total number of pages), which is the current row, then all of rows are iterated inside the page and data to be send to the screen is calculated for the entire page. An 8 bit variable is used to store the data to be send to write for this page. After calculating the data which is to be written for the entire page

write on screen function is called with the page number, column address and data which is just calculated. Same process is done for the entire display array.

- **Set and clear the ball from given position:**

  The position of the ball is saved in a structure(previousPos) which has the row and column of the ball. The saved position is of 1 pixel but as whole the ball is 3 x 3, so to calculate the entire ball position a function(set_Ball_To_Position) which gets the row and column (position of the ball) as a parameter and calculate the entire ball field of the ball which is 3 x 3 and set it to the display array. This function is also used to clear the ball from the given position for that additional parameter(setOrClear) is passed to this function which is to be checked when clearing or setting the ball to the given position.

- **Draw Vertical line on the screen:**

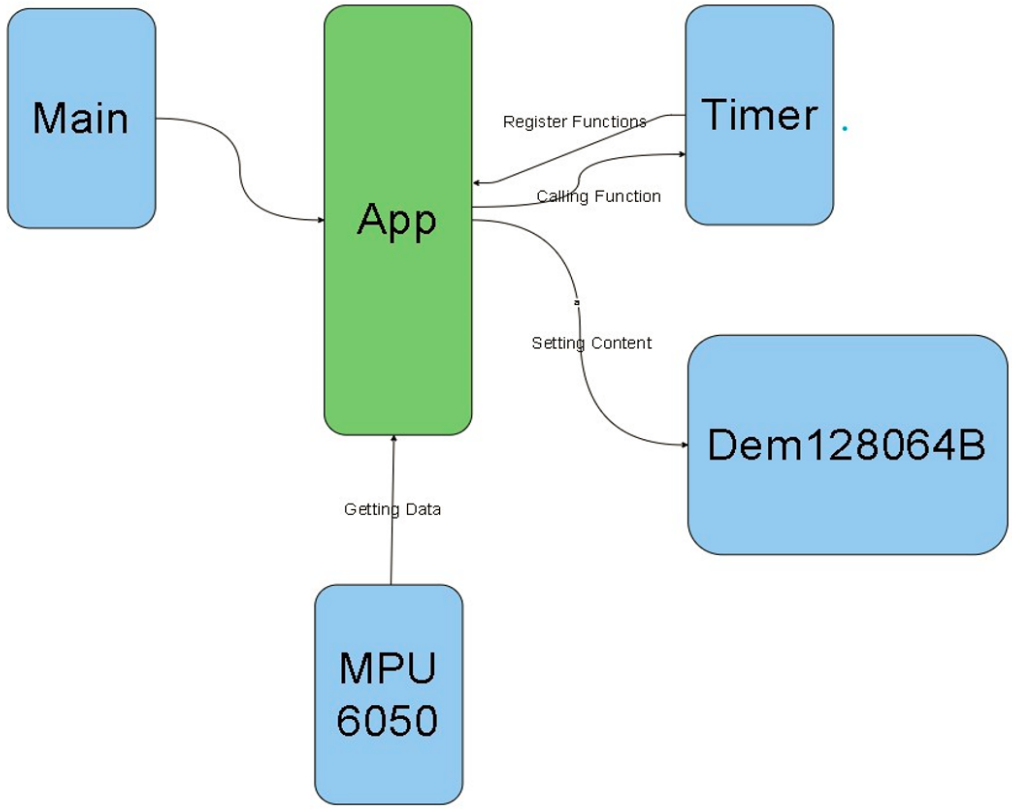
  As it is mentioned in the project description that two vertical lines will move towards the ball during the game play. A function is required to draw a vertical line. For that purpose, a function is created which gets the starting row of the vertical line starting column and the length of vertical line and draws a line. This line is drawn inside the 2d array of pixels and then reflected back when screen is refreshed. Same function is also used to clear the vertical line from the given position. Additional parameter is passed to check if the line is to be cleared from the given position or it is to be set.

- **Get Delay**

  A specific delay of 5 microseconds is required to toggle enable LCD, which is created using timer10 of STM32 F401RE microcontroller.

Software structure

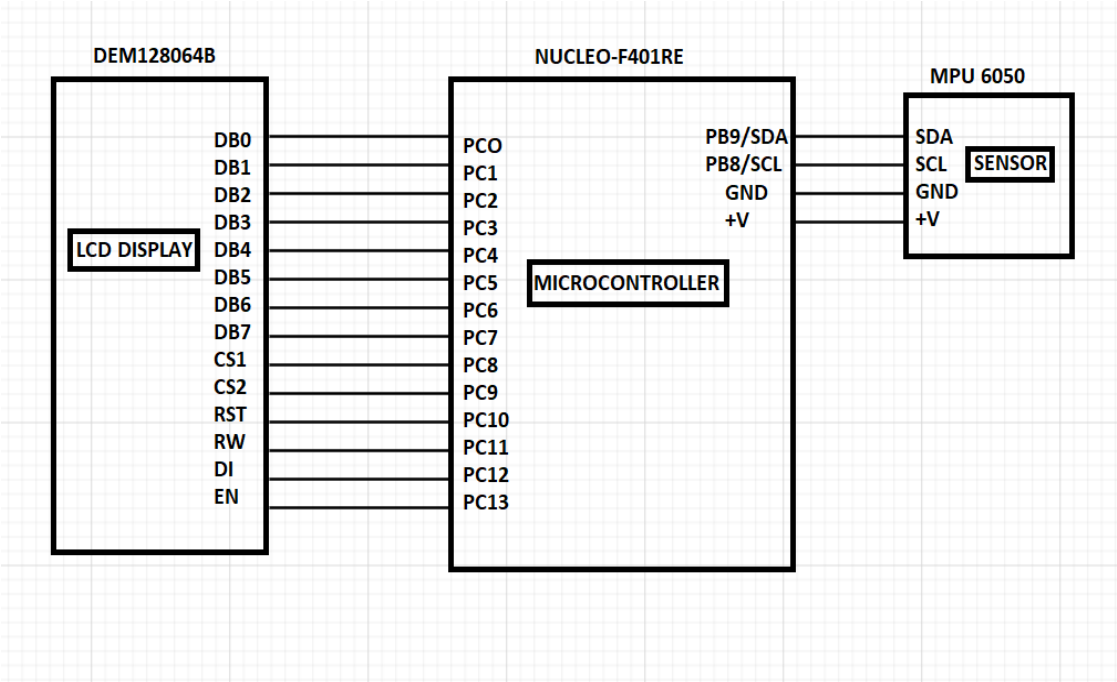
Software structure diagram is given below.



Application module(app) is the main module which is used for the initialization of all the modules used in the project like mpu6050, Dem128064B. After initialization the timer module is used to register the functions those are to be called periodically along with the frequency. Timer module also has a function which checks which function needs to be called when the time has reached for that function. Dem128064B module is used to draw content on the display, new position of the ball is calculated inside the app module and is set on the display by calling a function inside the Dem128064B module. Mpu6050 module is specifically used for the sensor used in this project. A function inside the app module is used to update the data from the mpu sensor (tilt angles), this function is called periodically so it is registered inside the timer.

Schematic

The Schematic diagram for the project is given below. MPU6050 sensor is connected to the microcontroller via I2C communication protocol and the LCD display DEM128064B is connected to the Portc of the microcontroller.



5. Test

Specific test cases

- Moving the ball to the left side.
- Moving the ball to the right side.
- Moving the ball to the top.
- Moving the ball to the bottom.
- Checking the corners if the ball can go out of the screen.
- Ball collision to the upper line.
- Ball collision to the lower line.
- Ball collision from the back side.
- Moving the ball with different speed.
- Counting points.

Expected results

All of the results of the test cases are same as expected. All test cases passed.

6. Conclusion

To conclude, the MPU6050 sensor is used as a sensor in this project, the acceleration in three axis is calculated by the sensor, from those values tilt angle is calculated, Interfacing is done with the STM32 microcontroller. LCD is (DEM128064B) is used to display content to the user, on the LCD a 3x3 ball is shown. The position and speed of the ball is controlled using MPU sensor. User has to control the ball movement and speed using mpu sensor. Two vertical lines moves from the right side towards the ball, user has to avoid the collision of the ball with the moving lines. If the ball is collided with any of the lines the game will stop and the points scored by the user are shown with a game over message.