EE447 Laboratory Project Final Report
Acceleration-Based 8x8 LED Dot Matrix Driver

Muhammet Ali Arslan – 2261543

Ramazan Cem Çıtak – 2231686

Submission Date: January 8, 2023

***Abstract*— This paper describes the theoretical and experimental results of the project of Acceleration-Based 8x8 LED Dot Matrix Driver.**

***Keywords*— I2C, SPI, C Programming, ARM Assembly, Tiva Launchpad.**

I. Introduction

This document is the final report for the EE447 Introduction to Microprocessor course. The aim of the project is to drive an 8x8 LED Dot Matrix using acceleration data that is measured by ADXL345 acceleration sensor. Also, the acceleration data is shown in Nokia 5110 LCD screen and onboard LEDs on the microcontroller unit, which is the Tiva Launchpad, are on and off according to three acceleration data.

II. Requirements for the Project and Background Information

A. Requirements of the Project

Firstly, the acceleration data should be converted to the angle of the gradient in three axes of Cartesian coordinates. Bu using the datasheet of ADX [1, p.3], we see that the resolution is 256 in g-range, so we divided the raw acceleration data into 256. This gives us the acceleration data that can be used in the formulas [2] given for calculating the angles in each axis as seen in Figure 1.

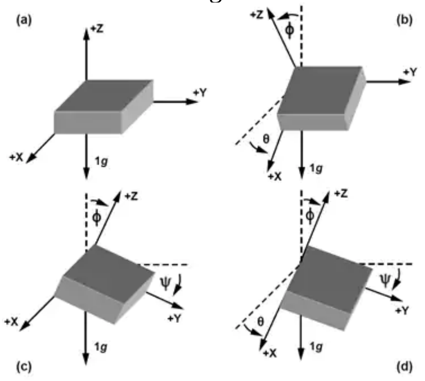


Figure 1. The acceleration angles in each axis

The formulas used are given in Figure 2.

$$\theta = \tan^{-1}\left(\frac{A_{X,OUT}}{\sqrt{A_{Y,OUT}^2 + A_{Z,OUT}^2}}\right)$$

$$\psi = \tan^{-1}\left(\frac{A_{Y,OUT}}{\sqrt{A_{X,OUT}^2 + A_{Z,OUT}^2}}\right)$$

$$\phi = \tan^{-1}\left(\frac{\sqrt{A_{X,OUT}^2 + A_{Y,OUT}^2}}{A_{Z,OUT}}\right)$$

Figure 2. The formulas used for calculating the angles

Secondly, by using the angles obtained from the calculations, each LED in 8x8 LED Dot Matrix should be on and off according to the angle ranges. Only X and Y axes angle values are used, that is, the 8x8 LED Dot Matrix represents the two-dimensional XY plane as in Figure 3.

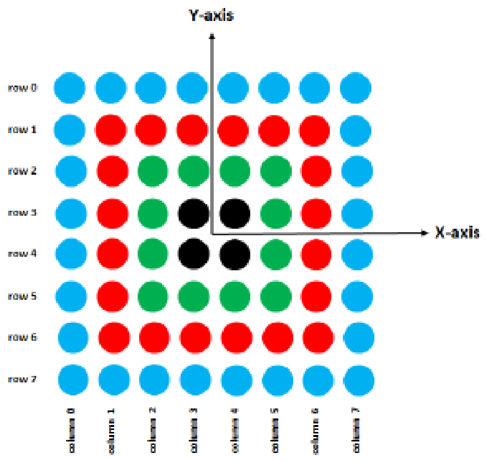


Figure 3. The 8x8 LED Dot Matrix

For example, the magnitude of the angles of X and Y axes are smaller than 10, no LEDs will be on. Between 10 and 30 degrees of magnitude of angles, the black area LEDs will be on and off. There are 30 to 50, green area, 50 to 70, red area and 70 to 90, blue area on 8x8 LED Dot Matrix, magnitude ranges for the LEDs to be turned on and off. Rows 0 to 3 represents the positive values of X angles, and column 4 to 87 represents the positive values of Y angles.

Thirdly, the angles of X, Y and Z axes are displayed on the Nokia 5110 LCD screen.
The angle ranges by the formula we used are from -90 to +90 degrees, which are the angles that arctanc function can calculate. In addition, the onboard RGB LED is on and red, green, and blue for X, Y and Z axes, respectively when the angle value of an axis is negative.

As a bonus feature of this project, we implemented an increasing brightness change when the acceleration data, or the angle of an axis, is higher. In other words, an LED is brighter close to the outer line of the 8x8 LED Dot Matrix.

B. I2C

When using I2C we configured TIVA board as master and ADXL345 sensor as a slave. After that we configured PORTD as our I2C port and configured alternate function 3 which is I2C. We used 400kHz frequency with I2C which means we wrote MTPR register 0x1 which was calculated. (Formula is 16 MHz (SystemFreq)/((2)*(6+4)/400000)=RegisterValue+1) The main concern with the I2C was the timings. The timing diagram was taken from the Texas Instruments website, and we used it extensively. After configuring and trying countless times to try reading data from the sensor we found out that our sensor was not working. After changing the sensor, we start taking data from the sensor. Reading and writing was done with the multiple read and multiple write fashion. After starting signal master should send slave address and should wait for acknowledgment from the sensor. After that master should send slave memory address which selects memory address register for the write or read operation, after sending slave memory address again master should wait for the acknowledgment. For reading operation master should send slave address plus one (Restart operation) after the first slave memory address, after that master should read the data and send acknowledgment for the data and if the reading is more than one byte master should not send not-acknowledgment until the end and took the data that is send. For writing operation master should send slave address normally and read the data afterwards.

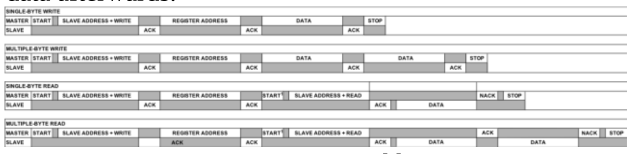

SINGLE-BYTE WRITE
MASTER START SLAVE ADDRESS + WRITE REGISTER ADDRESS DATA STOP
SLAVE ACK ACK ACK
MULTIPLE-BYTE WRITE
MASTER START SLAVE ADDRESS + WRITE REGISTER ADDRESS DATA DATA STOP
SLAVE ACK ACK ACK ACK
SINGLE-BYTE READ
MASTER START SLAVE ADDRESS + WRITE REGISTER ADDRESS START SLAVE ADDRESS + READ NACK STOP
SLAVE ACK ACK ACK DATA
MULTIPLE-BYTE READ
MASTER START SLAVE ADDRESS + WRITE REGISTER ADDRESS START SLAVE ADDRESS + READ ACK NACK STOP
SLAVE ACK ACK ACK DATA DATA

Figure 4. I2C Device Addressing

C. SPI

SPI is used for the Tiva Launchpad, which is the master device, to communicate with Nokia 5110 Screen and 8x8 LED Dot Matrix. For both screen and dot matrix, which are slave devices, we send 8-bit data and do not receive any data. Data is sent in the command of the clock signal for the synchronization. Data is hold in the FIFO in SPI, and the oldest data is sent first to the slave device.

D. About the Nokia 5110 LCD Screen

For Nokia 5110 configuration, we first made GPIO and SPI configuration, then followed the Nokia 5110 configuration on the manual. While writing our code, we used the table [4, p.16] given in the datasheet. As in Figure 5, the screen consists of horizontal banks from 0 to 5, and vertically there are 5 pixels, or columns in each unit. We used banks 0, 1 and 2 to display the X, Y and Z angles on the screen.

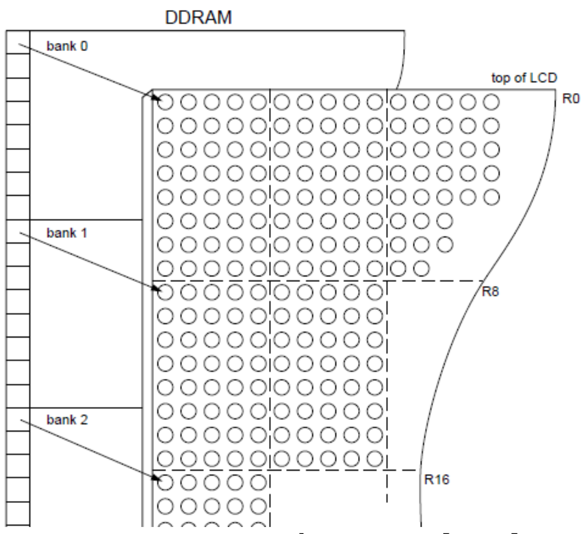


Figure 5. DDRAM Display Mapping [4, p.8]

The addresses in each unit can be seen in Figure 6.

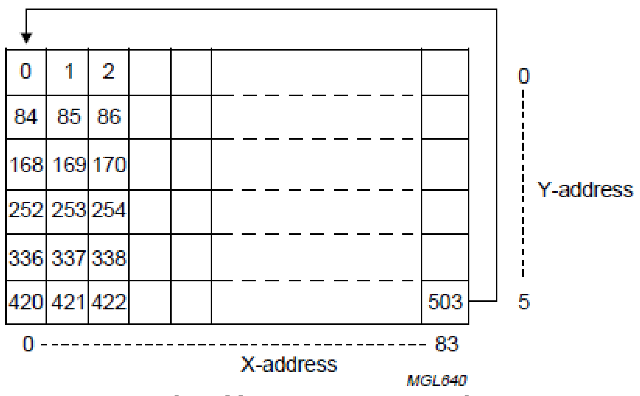


Figure 6. The addresses to write in Nokia 5110 Screen [4, p.10]

We send 8-bit data to the screen. From Figure X, we see that we have to send five 8-bit data to fill a unit (8x5) of the screen for each vertical line that consists of 8 rows, representing 8 bits that we send. In addition, before sending the data, we have to adjust the cursor of the screen by sending the correct data through SPI.

III. Implementation of the Project and Test Results

The project has four main parts: ADXL345 Acceleration Sensor, 8x8 LED Dot Matrix, Nokia 5110 LCD Screen and On Board LEDs on Tiva Launchpad.

A. *The Pseudocode for the Algorithm Used in the Project*

```
Initialize the I2C module:
- Enable the I2C module and set the
  clock frequency to 100mHZ
- Configure the I2C pins
- Enable the I2C module and set it to
  slave mode
Initialize the ADXL345 accelerometer:
- Put the ADXL345 into measurement mode.
- Disable Interrupts for the ADXL345
  pins.
Configure the ADXL345 interrupt pin:
- Set the interrupt pin Port B as an
  input
- Digital enable the interrupt pin
- Configure the interrupt pin as edge-
  sensitive
- Configure the interrupt pin as a
  single edge
- Configure the interrupt pin as
  falling edge
- Clear any pending interrupts on the
  interrupt pin 2

- Enable interrupts on the interrupt
  pin
Potentiometer for threshold setting:
- Initialize the ADC module and the
  potentiometer input pin
- Initialize the ports for interrupts
- Read the potentiometer input through
  reading interrupt
- Normalize the data
- Set offset on angle of gradient
  angles according to pot data
```

Nokia 5110 LCD:

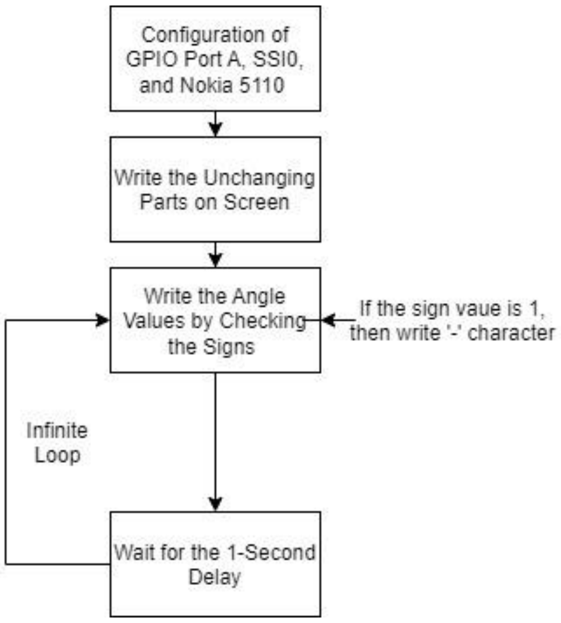

Configuration of GPIO Port A, SSI0, and Nokia 5110
Write the Unchanging Parts on Screen
Write the Angle Values by Checking the Signs
If the sign vaue is 1, then write '-' character
Infinite Loop
Wait for the 1-Second Delay

Figure 7. Nokia 5110 Flowchart

```
8x8 LED Dot Matrix:
- Initialize the dot matrix display:
- Disable interrupt
- Dot matrix column and row pins
- Enable the clock and configure the
  pin mux for the dot matrix pins
- Set the dot matrix pins as outputs
- Clear all the columns and rows by
  clearing data registers.
- Configure Interrupt
- Wait for interrupt:
- Convert the acceleration data to dot
  matrix column and row indices
- Convert the x acceleration to a dot
  matrix column index
- Return the column index
- Convert the y acceleration to a dot
  matrix row index
- Return the row index
- Update the dot matrix display
Now define Accelerator interrupt handler:
- Check if the interrupt was caused by
  the ADXL345
- Read the acceleration data from the
  ADXL345
- Calculate angles according to our
  angle formulas

while (1){ /* Infinite loop */
    Wait for an interrupt to occur
}
```

B. *Implementing ADXL345 Acceleration Sensor*

We used I2C communication for this sensor. The pins from board to screen are as follows:
PD0-SCL
PD1-SDA

Sensor configuration was simple. We just put the sensor into normal mode and afterwards we opened the FIFO register to take data from FIFO buffers. We read all 6 data registers in one read and wrote to the respective register.

C. *Implementing 8x8 LED Dot Matrix*

We used SSI2 for the 8x8 LED Dot Matrix.
PB4-CLK
PB5-CS
PB6-Rx (Not used)
PB7-Tx

Our main logic drove 8x8 LED's, LCD, and on-board LEDs with this data with some logic operations given in the codes.
Simple if-else loops were used for this, and we used single interrupt as form of SysTick timer. SysTick timer was configured with on-board LEDs in the code and the time for the timer is 100ms. In our main loop we perform LCD refresh every 1 second and do a count for it. Every 100ms in interrupt we took data from sensor, do logic operations for 8x8, LCD and on-board LED's and write respective memory address for the x,y,z magnitude and sign, as well as 8x8 column and row data, also LED light data.

Our SPI configuration for the 8x8 was free scale and 8MHz clocked. We used PORTB alternate function 2 which is for SPI, we opened digital enable pins for every pin and selected alternate function with AFSEL, we also selected pins as outputs also to make our writing easier we make pin select high as suggested in the TIVA board datasheet. PORTB pins 4-5-6-7 were used but port 6 was not connected since we did not read from the 8x8 LED's. Our writing was simple enough since the TIVA board automatically drives pin select pin which is pin 5, for writing operation. For writing operation, we waited for the transmission FIFO to be not full and afterwards we send our data. After that we wait for the operation to be done and we finish writing data. SPI registers SR was polled for the FIFO and busy bits and writing was done to DR register in SSI2. For both SSI and I2C configurations we polled the ready register after opening clocks and we configured the SSI when the clock was off. Our 8x8 LED's change brightness with increasing acceleration which was done by taking acceleration data and sending brightness register this data (0xA00) in 8x8.

If the angle for both x and y is between -10 and 10, there is no led is on. If an angle for x or y is higher than the threshold, respective row and column data is taken, and the respective led is on.

We added up angles/10 and wrote it to 8x8 LED's brightness register (0xA00), which is the bonus part of the project.

D. *Implementation of Nokia 5110 LCD Screen*

SSI0 is used for the SPI communication. The pins from board to screen are as follows:
PA7 – RST
PA3 – CE
PA6 – DC
PA5 – DIN
PA2 – CLK

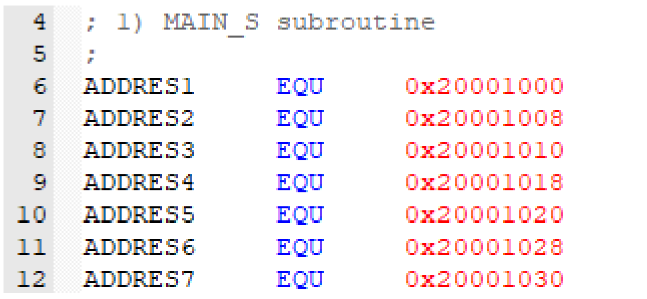
Also, the VCC and GND pins should be connected in common with other components.

From the datasheet [4, p.20], the clock cycle SCLK period should be at least 250 ns. Thus, SSI0 frequency should be at most 4 Mhz = 1/(250*10^(-9)), and using a prescaler of 2, we could use a lower frequency such as 16 Mhz/ (2*(1+2)) = 2.66 Mhz.

To print the characters to screen, we will use a large ASCII array [3], where each character consists of five 8-bit in the array. To use this array in ARM Assembly and refreshing the angle values on the screen, we used the `CONVRT` subroutine for the conversion from integer to ASCII [5].

The refreshing time for the screen is adjusted by using a delay subroutine that makes 1-second delay.

There are 6 addresses that are changed through pointers in the main C code for the screen refreshing. To display a number on the screen, we need sign data and magnitude data. Thus for three axes, we used 6 addresses as in Figure 8. We used 6 different saved registers in ARM Assembly for these magnitude and sign data. 0 represents a positive value, and 1 represents a negative value.



```
; 1) MAIN_S subroutine
;
ADDRES1     EQU     0x20001000
ADDRES2     EQU     0x20001008
ADDRES3     EQU     0x20001010
ADDRES4     EQU     0x20001018
ADDRES5     EQU     0x20001020
ADDRES6     EQU     0x20001028
ADDRES7     EQU     0x20001030
```

Figure 8. The 6 addresses for sign and magnitude values

E. *Using On Board LEDs on Tiva Launchpad*

GPIO PF1 for Red, PF3 for Green, and PF2 for Blue LED will be ON when the angle of the X, Y, or Z-axis is below zero, respectively. They are checked by using the sign data of the angles in the `Systick_Handler` Interrupt every one second.

F. *The Overview of the Project and Experimental Results*

Figures 9 and 10 show the overview of the project when all the connections are done while the system is working. The acceleration sensor is connected to the orange mini breadboard for the easiness. The 8x8 LED Dot Matrix can be seen with values of the angles on the screen.

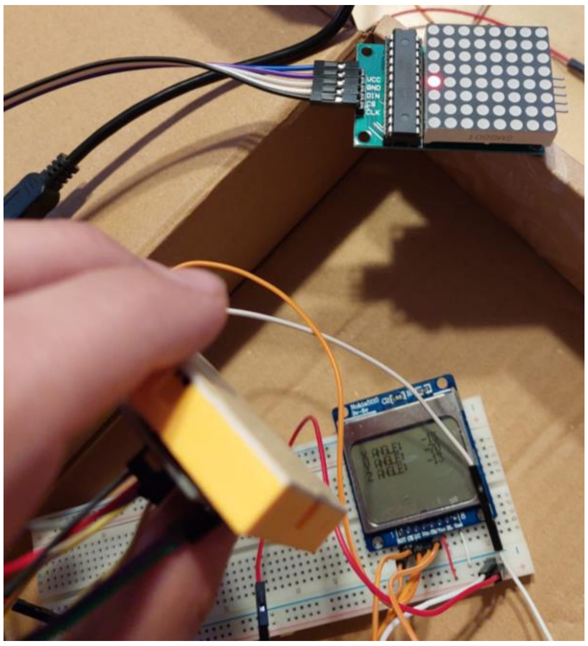


Figure 9. The entire project with connection while working

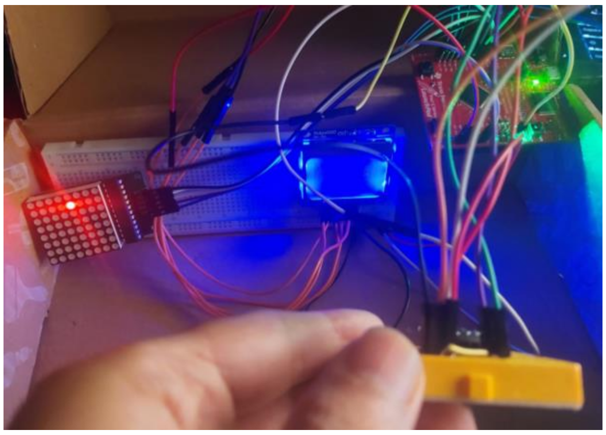


Figure 10. The entire project with connection with backlight (BL) of the screen

IV. Conclusion

In this project, we made implementations of our theoretical knowledge of I2C and SPI serial communication protocols as well as using our ARM Assembly and C programming in Tiva Launchpad MCU. Implementing I2C and SPI in such a project, we developed our skills in datasheet reading, especially for Tiva Launchpad as well as for the other components. Also, we have used our theoretical knowledge of logic design and creating algorithms.

V. The Sharing of the Tasks in the Project

Table 1: The Sharing of the Tasks

Tasks	Muhammet Ali Arslan	Ramazan Cem Çıtak
Main Logic for 8x8 LEDs	% 80	% 20
On board LEDs, `Systick_Handler` Interrupt, Screen Loop	% 20	% 80
ADXL345 I2C and Data Conversion to Angles	% 80	% 20
Nokia 5110 LCD Screen SPI	% 20	% 80
8x8 LED Dot Matrix SPI	% 70	% 30
Bonus: 8x8 LED Brightness Change	% 90	% 10

References

[1] *ADXL345 Datasheet*, Accessed: Jan. 8, 2023. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf

[2] C. J. Fisher, *Using An Accelerometer for Inclination Sensing*, May 6, 2011, Convergence Promotions LLC, Accessed: Jan. 8, 2023. Available: https://www.digikey.com/en/articles/using-an-accelerometer-for-inclination-sensing

[3] D. Valvano, Nokia 5110_4C123, Sep. 16, 2013. Accessed: Jan. 8, 2023. Available: http://users.ece.utexas.edu/~valvano/arm/Nokia5110_4C123.zip

[4] *PCD8544 48 × 84 pixels matrix LCD controller/driver Datasheet*, Accessed: Jan. 8, 2023. Available: https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf

[5] three-bee, *freq-based-motor*, Feb. 8, 2022. Accessed: Jan. 8, 2023. Available: https://github.com/three-bee/freq-based-motor

[6] *I2C Communication TM4C123G Tiva C Launchpad*, Accessed: Jan.8, 2023. Available: https://microcontrollerslab.com/i2c-communication-tm4c123g-tiva-c-launchpad/

[7] *SPI Communication TM4C123 – Communication Between Tiva Launchpad and Arduino*, Accessed: Jan. 8, 2023. Available: https://microcontrollerslab.com/spi-tm4c123-communication-between-tiva-launchpad-arduino/