EGR 226 Microcontroller Programming and Applications

Winter 2021

Instructor: Prof. Trevor Ekin

Final Project Report

Gabriel Gasbarre

April 14th, 2021

Contents

[Introduction 4](#_Toc70278815)

[Objectives and Requirements: 4](#_Toc70278816)

[Project Objective 4](#_Toc70278817)

[Hardware constraints: 4](#_Toc70278818)

[Software Constraints: 4](#_Toc70278819)

[Additional Objectives: 4](#_Toc70278820)

[Equipment: 5](#_Toc70278821)

[Procedure 6](#_Toc70278822)

[Hardware configuration 6](#_Toc70278823)

[MSP Pin Connections 6](#_Toc70278824)

[Enclosure 7](#_Toc70278825)

[LCD Cover & Controls 8](#_Toc70278826)

[Cylindrical Robotic Arm 8](#_Toc70278827)

[Software configuration 9](#_Toc70278828)

[Software Flowchart: 9](#_Toc70278829)

[Interrupts: 10](#_Toc70278830)

[Main Function: 11](#_Toc70278831)

[Main menu: 11](#_Toc70278832)

[Arm Menu: 12](#_Toc70278833)

[Motor Menu: 12](#_Toc70278834)

[RGB LED Menu: 13](#_Toc70278835)

[Conclusion 13](#_Toc70278836)

[Project Objective Completion: 13](#_Toc70278837)

[Project Hardships: 14](#_Toc70278838)

[Future additions: 14](#_Toc70278839)

List of Figures:

[Figure 1: Robotic Arm Servos & Control Potentiometers Schematic 6](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278734)

[Figure 2: LCD & Control Potentiometer Schematic 6](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278735)

[Figure 3: RGB LED & Motor w/ Pushbuttons Schematic 6](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278736)

[Figure 4: Keypad Schematic 6](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278737)

[Figure 5: Enclosure & Setup Diagram 7](#_Toc70278738)

[Figure 6: LCD Cover 3D Model 8](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278739)

[Figure 7: EEzybotARM 3D Model 8](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278740)

[Figure 8: Software Configuration Flowchart 9](#_Toc70278741)

[Figure 9: Port3IRQHandler 10](#_Toc70278742)

[Figure 10: Timer32 Interrupt handler 10](#_Toc70278743)

[Figure 11: Main Menu Screen 11](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278744)

[Figure 12: Port1IRQHandler 11](#_Toc70278745)

[Figure 13: Arm Control Screen 12](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278746)

[Figure 14: UpdateArmPosition function 12](#_Toc70278747)

[Figure 15: Belt Control Menu Screen 12](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278748)

[Figure 16: Motor Speed Update Function 12](#_Toc70278749)

[Figure 17: LED Control Menu Screen 13](file:///C:\Users\Gasba\Documents\@EGR-226\EGR%20226%20Final%20Project%20Report.docx#_Toc70278750)

[Figure 18: Red LED Brightness Update Function 13](#_Toc70278751)

# Introduction

## Objectives and Requirements:

### Project Objective

* + - Design a control system on the TI MSP432 Launchpad microcontroller board using a numeric Keypad and 16x4 LCD. The microcontroller will control the LCD display menu functions using keypad entry.

### Hardware constraints:

* + - This control system must include the following hardware:
      * Servo Motor “Door” (PWM)
      * LED “Door Feedback” (GPIO)
        + A Red LED on any time the door is closed, otherwise, off
        + A Green LED on any time the door is open, otherwise, off
      * DC Motor (PWM Speed)
        + Push-Button “Emergency Stop” (GPIO Interrupt)
      * RGB LEDs (PWM Brightness)
        + Push-Button RGB LED Enable/Disable (GPIO Interrupt)

### Software Constraints:

* + - A Main Menu will be displayed on the LCD allowing the user to select between three options: (1) Door Menu (2) Motor Menu (3) Lights Menu
    - LCD Backlight must be driven by a PWM.

### Additional Objectives:

* This control system will include the following supplementary hardware
  + - * Servo motor 3 Degree of Freedom Cylindrical Robotic Arm
        + 1K Ohm potentiometer control
      * 3D Printed DC Motor Mount
        + This mount secures the motor and allows it to drive a belt.

## Equipment:

|  |  |  |
| --- | --- | --- |
| **Part** | **Description** | **Model** |
| CCS (Code Composer Studio) | Integrated development environment to develop applications for Texas Instruments embedded processors. | 10.0.00010 |
| MSP432 | Mixed-signal microcontroller from Texas Instruments. | MSP432P401x |
| EGR 226 Embedded Systems Design/Build Project Winter 2021 | Project proposal document | N/A |
| LCD | EGR 226 Lab Kit LCD display with HD44780 controller | HD44780 |
| Keypad | EGR 226 Lab Kit component | N/A |
| 75:1 Micro Metal Gearmotor | High-power, 6 V brushed DC motor with a 75.81:1 metal gearbox | Pololu item #: 2361 |
| MG90S Servo x4 | Micro servo motor with metal gear | MG90S |
| RGB LED | Common ground, three color (red, green, blue) LED | N/A |
| LED x2 | Red and green external LED | 560MR2D |
| PoiLee 1K Ohm Potentiometer | 1K ohm top adjustable linear taper rotary potentiometer | B07CDQQKPQ |
| MOSFET Transistor x3 | Insulated-gate field-effect transistor | 2N7000 |
| Pushbutton switch x3 | Omron brand external switch | B3F |
| Resistor x5 | Varying value current resistors | N/A |

# Procedure

## Hardware configuration

### MSP Pin Connections

LCD & Control Potentiometers: Robotic Arm Servos & Control Potentiometers:

Diagram

Description automatically generatedDiagram, schematic

Description automatically generated

Figure : Robotic Arm Servos & Control Potentiometers Schematic

Figure : LCD & Control Potentiometer Schematic

Diagram

Description automatically generated RGB LED & Motor w/ Pushbuttons: Keypad:

Figure : RGB LED & Motor w/ Pushbuttons Schematic

Table

Description automatically generated with low confidence

Figure : Keypad Schematic

### Enclosure

A picture containing control panel

Description automatically generated

Figure : Enclosure & Setup Diagram

Featured above in FIGURE\_\_\_\_, The components for this project have been surface mounted atop a 14L plastic storage container to allow for the containment of wires, along with protection for the MSP432 microcontroller.

Though the enclosure for this project creates a much more visually appealing result, the removal of a direct breadboard connection for nearly every component introduces many problems. Certain components, such as the potentiometers and pushbuttons, do not have connection pins suitable for the jumper wires included in the EGR 226 kit. This meant that wires needed to be stripped and directly tied onto leads to secure a lasting connection. This also meant that there were significantly more wires than usual, crowding the build area and creating chaos.

### LCD Cover & Controls

Cover:

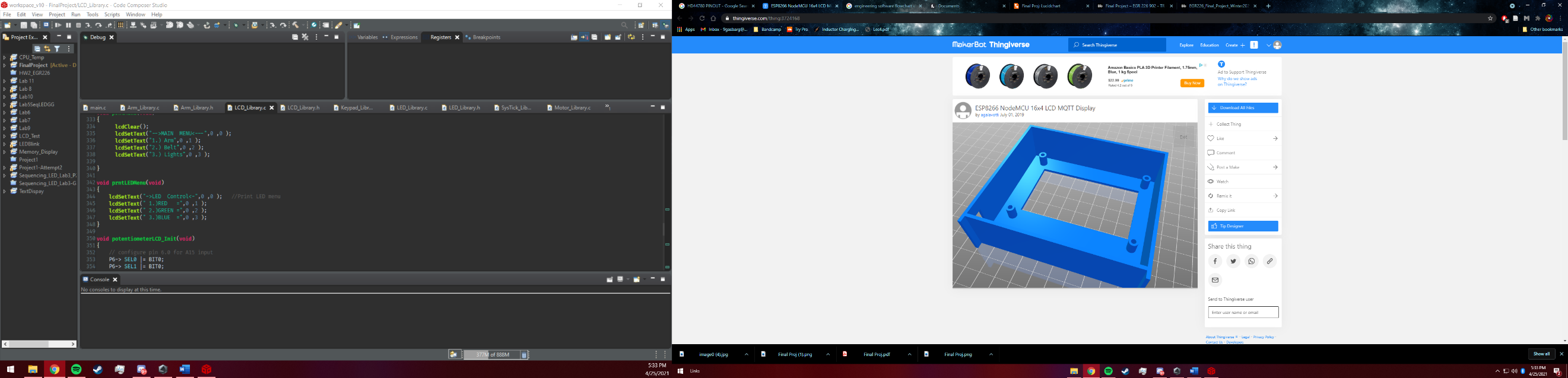
 Featured to the right in FIGURE\_\_\_ is a 3D model of the ESP8266 NodeMCU 16x4 LCD MQTT Display from Thingiverse. This design was used to house the LCD, as well as mount the contrast and brightness potentiometers for the display. Though there was not a lot of room for wires here, the cover certainly made the organization as well as the overall look of the project much sleeker.

Figure : LCD Cover 3D Model

Designer: ALEX GALAVOTTI

Link: <https://www.thingiverse.com/thing:3724168>

Controls:

As mentioned above, the backlight brightness and contrast for the LCD are controlled using 2 individual, top adjustable rotary potentiometers. To change the contrast of the LCD, a potentiometer was wired directly into V0 of the HD44780 LCD to vary the voltage across the contrast pin.

To alter the backlight of the LCD was significantly more difficult as it was to be changed using pulse width modulation through ADC14 and TimerA. More about how the brightness is changed can be found in the interrupt section of the software configuration.

### Cylindrical Robotic Arm

Figure : EEzybotARM 3D Model

Featured to the right is a 3D model example of the EEzybotARM from Carlo Franciscone via Thingiverse. This arm is controlled with 4 MG90S servos, which receive DAC signals from the 3 control potentiometers and the claw pushbutton. The DAC values are updated through the ADC14 interrupt routine. More about how the servo positions change through ADC input can be found in the interrupt section of the software configuration.

Link: <https://www.thingiverse.com/thing:1015238>

## Software configuration

### Software Flowchart:

Diagram

Description automatically generated

Figure : Software Configuration Flowchart

### Interrupts:

Claw Pushbutton input:

The Claw pushbutton interrupt updates the claw position by toggling a TimerA value to the claw servo of either 6000 (2ms duty cycle time, fully open) or 3000 (1ms duty cycle time, fully closed.) At the same time, this also updates the indicator LED’s, which sets the green LED on when the claw is open, and the red LED on when the claw is closed. Featured below is the Port3 interrupt routine, which performs all the aforementioned functions.



Figure : Port3IRQHandler

Timer32 Interrupts:

Timer32 has been configured such that at every 50ms interval ADC values from each of the 4 potentiometers (1 for LCD Backlight, 3 for robotic arm) are converted and saved to memory registers. This interrupt routine also updates the value of the backlight control variable, so that the LCD is constantly being updated with the most recent brightness value, no matter where the user is in the program.

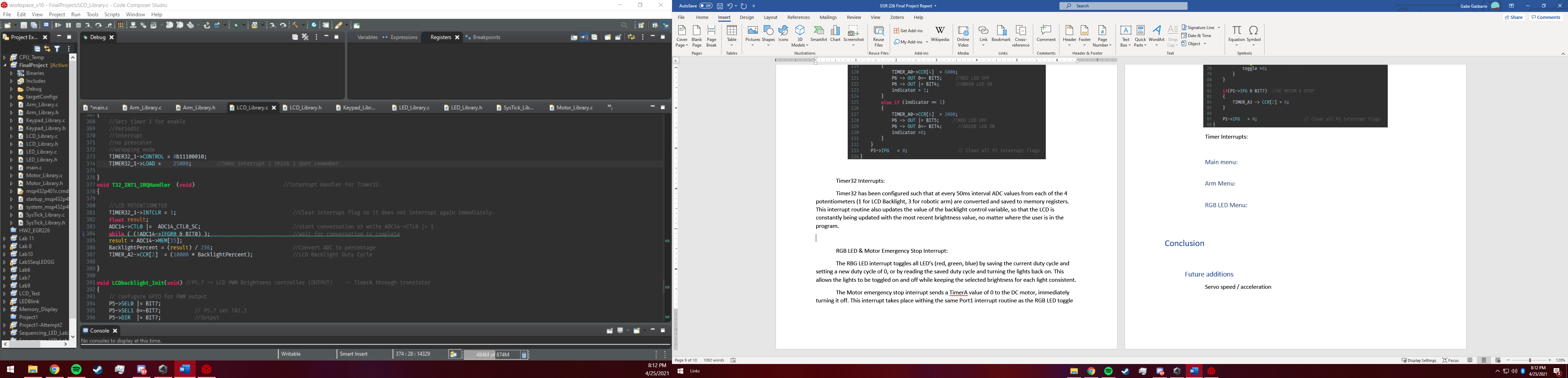


Figure : Timer32 Interrupt handler

RGB LED & Motor Emergency Stop Interrupt:

The RBG LED interrupt toggles all LED’s (red, green, blue) by saving the current duty cycle and setting a new duty cycle of 0, or by reading the saved duty cycle and turning the lights back on. This allows the lights to be toggled on and off while keeping the selected brightness for each light consistent.

The Motor emergency stop interrupt sends a TimerA value of 0 to the DC motor, immediately turning it off. This interrupt takes place withing the same Port1 interrupt routine as the RGB LED toggle switch but is differentiated by reading which interrupt flag has been risen. Featured below is the entire Port1 interrupt handler, with both RGB LED and motor emergency stop functions.

A picture containing text, scoreboard

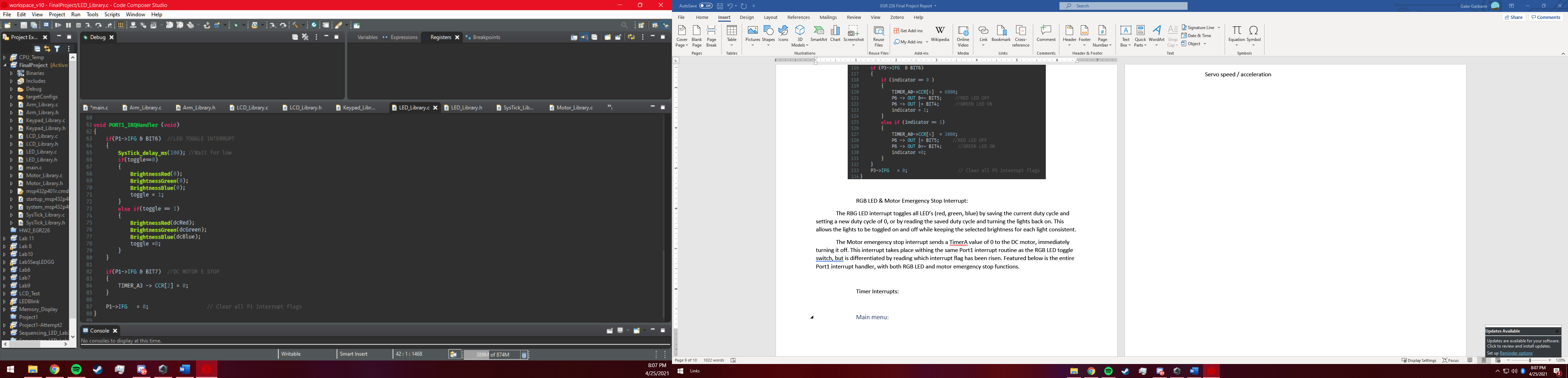
Description automatically generated

Figure : Main Menu Screen

Figure : Port1IRQHandler

## Main Function:

### Main menu:

Upon initial powerup of the program, the MSP will send the initial display menu to the LCD, featured to the right. From here, the program waits for a user input until one of the three options has been selected.

### A picture containing text, scoreboard, display Description automatically generated Arm Menu:

Figure : Arm Control Screen

By pressing “1” from the main menu, the user is brought to the arm menu, where arm control is initiated. To send new TimerA values to each of the 4 servos on the robotic arm, the function “UpdateArmPos” is continuously run until an exit function is read from the keypad. This function, featured below, reads each ADC memory value for its respective potentiometer, saves it, and converts it into a duty cycle of between 1ms (fully closed servo position) and 2ms (fully open servo position.)

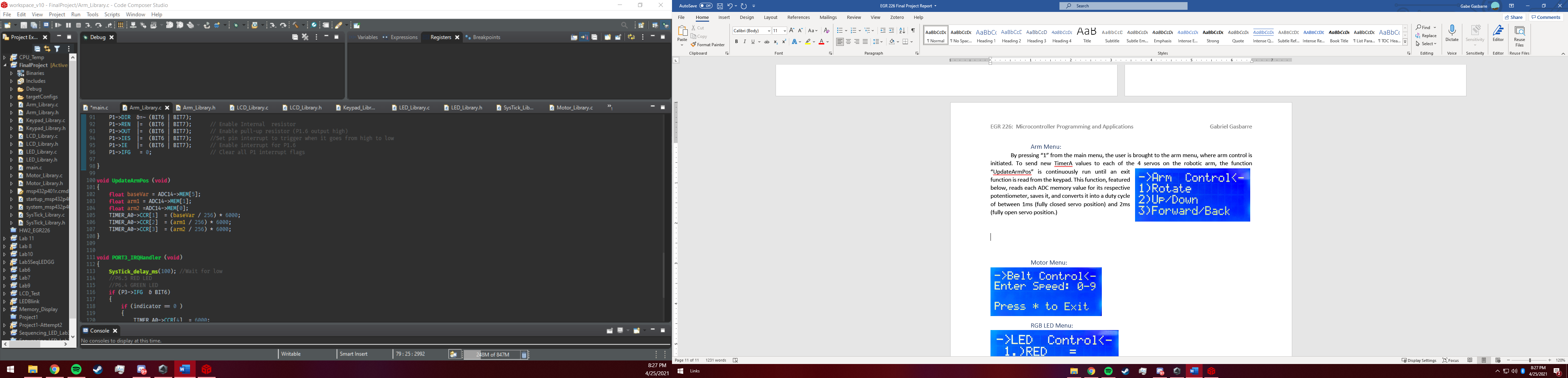


Figure : UpdateArmPosition function

A picture containing text, scoreboard, first-aid kit

Description automatically generated

Figure : Belt Control Menu Screen

### Motor Menu:

By pressing “2” from the main menu, the user is brough to the motor menu, featured to the right, which is aptly renamed “Belt control” as the motor shaft runs a pseudo conveyor belt. Once accessed, the motor menu waits for user input, which once received updates the TimerA duty cycle controlling the motor according to the value received from the user. Any input 0-9 is accepted and converted into the TimerA duty cycle through the “Motor\_TA” function. Featured below is the motor control function that updates the TimerA value depending on the user input.

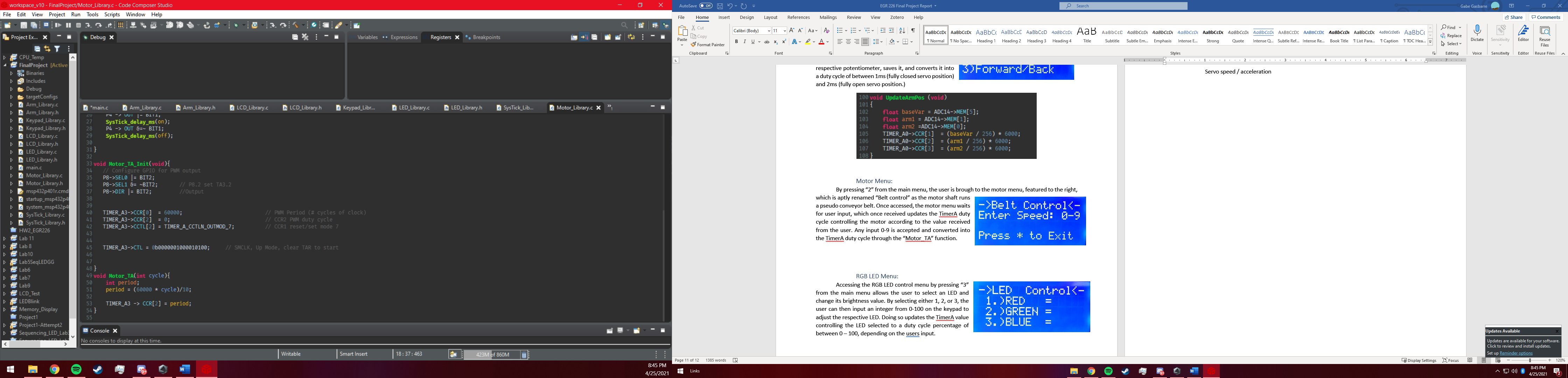


Figure : Motor Speed Update Function

A picture containing text, scoreboard

Description automatically generated

Figure : LED Control Menu Screen

### RGB LED Menu:

Accessing the RGB LED control menu by pressing “3” from the main menu allows the user to select an LED and change its brightness value. By selecting either 1, 2, or 3, the user can then input an integer from 0-100 on the keypad to adjust the respective LED. Doing so updates the TimerA value controlling the LED selected to a duty cycle percentage of between 0 – 100, depending on the users input. Featured below is the Red LED menu and duty cycle update function, which displays the entered duty cycle and updates the LED brightness.

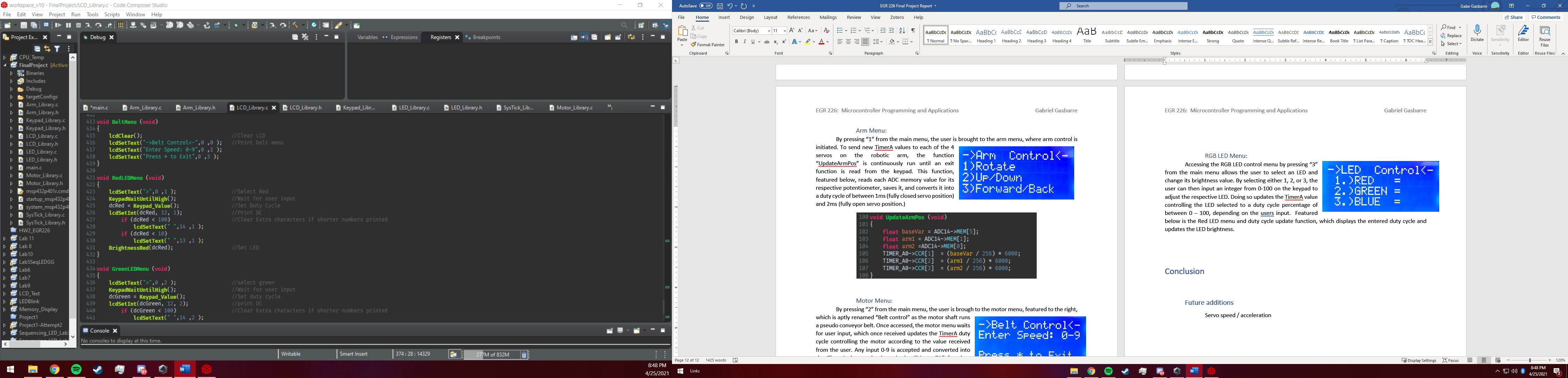


Figure : Red LED Brightness Update Function

### 

# Conclusion

## Project Objective Completion:

It was the main goal of this project to design an LCD and Keypad integrated control system that could accept user input to control multiple electronic components, such as LED’s, a servo, and a DC motor. It was an added goal that this system also controls a Robotic arm using 3 extra servos. With this in mind, and looking at the project demonstration, this project was undoubtedly a success. Repeatability, efficiency, and ease of use of the program all show the success in implementing multiple electronic components for inputting and outputting information. The program has also surpassed some of the initial goals in the sense that LDC readability offers much more information displayed to the user than initially intended, such as current LED brightness values and motor speed values.

## Project Hardships:

Though many components of this project have been completed in previous labs, there were specific portions of this project that prompted new issues that have not been experienced before. The most difficult of which to deal with was the issue of updating multiple ADC values simultaneously. In the original ADC lab, students only had to use a single ADC with a single conversion, however in this project multiple ADC’s had to be configured such that their values are updated simultaneously, along with their values being saved to individual memory locations. To do this, each ADC input pin was configured to store its resulting value in a memory location corresponding to the number ADC it is in the register document. From here, the ADC14 was configured such that it sweeps through each input, converting each value and saving them in the designated storage location.

## Future additions:

If this project were to be updated in the future, one of the primary goals would be to adjust the UpdateArmPos function such that the potentiometer input is unable to generate large servo value differences in a short amount of time. Doing so would mean the arm position does not change too rapidly, resulting in shaking / jerking motions, such as it currently does.