CS/EE 120B Custom Project Report

Full Demo link: https://youtu.be/efcC8ClQNcM

- Complexity 1 Joystick:
  - https://youtu.be/efcC8ClQNcM?t=40
- Complexity 2.1 EEPROM:
  - https://youtu.be/efcC8ClQNcM?t=56
- Complexity 2.2 Custom LCD characters:
  - https://youtu.be/efcC8ClQNcM?t=40
- Complexity 3 Nokia 5110 screen:
  - https://youtu.be/efcC8ClQNcM?t=40

Project Description:

Introduction

The player must select the correct translation of the displayed and/or audio morse code by moving the joystick and pressing a button to confirm. TThe player can press a button to play morse code through a speaker. The player can press the button again to replay the morse code as many times as desired. If the selected letter is correct, the player scores a point. If it is incorrect the player will lose a heart. Players start out with 3 hearts and the game continues so long as you stay alive. If the player loses all 3 hearts the game ends and the player score is displayed with the top score. The player must then press a button to restart the game.

Components

The possible characters the player can choose from are displayed on a Nokia 5110 screen. The player can interact with this project using a joystick to move the highlighted character. There are also 4 buttons. The button closest to the joystick is the select button used to select the highlighter character. The button second closest to the joystick plays the morse code sound. The third closest button resets the game and can work at any time. The last button clears memory of the highest score. There is also a LCD display that shows you how many lives you have left using heart custom characters. Below the hearts is where the written morse code is displayed using custom dot and dash characters. Once the game ends, the top score is shown as well as the player's score.

Complexities/Build-upons

1. Using the Nokia 5110 LCD screen to display the possible answers.
2. Joystick to move
3. Using the EEPROM to save the high score of the players (½)

4. Using custom characters/symbols for LCD screen (½)

User Guide:

Rules:

As soon as there is power, the game begins. Whenever the player presses the select button on the wrong character, a heart is lost and no points are gained nor are points lost. The player can try again until all lives are lost. Whenever the player presses the select button on the correct character, the player earns a point and the next morse code is shown. The player can play forever if they can continue to correctly select the right characters but the game immediately ends once the player loses all 3 lives. The game over screen is simply a message showing and labeling the top score as well as the player's score.

Controls:

On/Off

The power button on the power supply toggles the power on and off.

Joystick

Using the joystick the player can move around the on Nokia 5110 screen where all the options are. There is no continuous movement so after a single action, the player must allow the joystick to return back to the center before moving again. The only valid moves are up, down, left, and right. Diagonal movements are not recognized. Further, if the player hits the edge and tries to move past it, the player cannot continue and is forced to choose a different direction to move. Movement input from the joystick during the score page is ignored.

Button 1: Select Button

The select button is the button closest to the joystick. Pressing this button selects the character highlighted on the Nokia 5110 display. The game only recognizes the selection once the player has released the button to allow for the player to change answers midpress. Releasing the button on an incorrect answer takes a heart. On the last heart, immediately after releasing the button on an incorrect answer, the LCD will display the scores. The selection button does not do anything during the game over screen.

Button 2: Audio Play Button

The audio play button is the button 2nd closest to the joystick. The audio morse code plays as soon as the button is depressed. If the button is pressed and held, the audio will play continuously. If it is released while playing, it will finish playing before stopping. This button does nothing when displaying scores. It can also be used simultaneously with the joystick and select button.

Button 3: Reset Button

The reset button is the button 3rd closest to the joystick. This button will restart the game from the beginning anytime. The reset occurs once the button is released. If pressed and held during the game, the joystick and audio play button will work but the select button will not. The only way to reset the game from the score page is to press the reset button. If pressed and held during the score page, the memory clear button will not work.

Button 4: Memory Clear Button

The memory clear button is the farthest button from the joystick. It does not work during the game. It is only functional when the score page is shown. The button clears memory as soon as it is depressed. While it is depressed, the other buttons do not work.

Special Considerations:

The high score is initialized to the 255 because that is the maximum value an unsigned char can have. Before playing the game, the high score must be reset to 0. The user must lose the game to get to the score page, where the memory clear button should be pressed to reset the top score to 0. This only needs to be done once because the value is stored using EEPROM so that even if you turn the system off, it will remember the top score. Now, whenever a player is able to score anything above 0 points, their score will be saved as the new high score.

Source File Description:

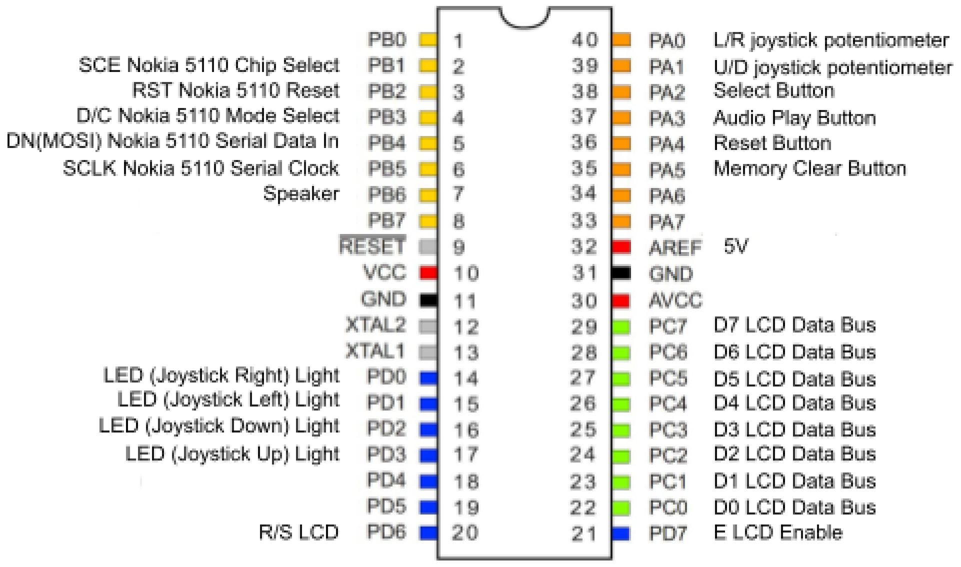
- main.c - The core of the project. All core tasks are implemented here. Below lists the parts that were from outside sources.
    - Timer.h code - Timer.h allows the system to implement timing. Timer.h was provided in EECS120b Lab 06 - Timer synchSMs (Reflex game)
    - PWM functions - The functions set_PWM() , PWM_on(), and PWM_off() give the system PWM functionalities to output onto the speaker. These functions were provided in EECS120b Lab 09 - PWM (Music player)
    - Task Scheduler structure - The task scheduler structure is a way to implement concurrent synchSMs. This template was provided in Zybooks Chapter 5.7 and EECS120b Lab 11 - scheduler (Scrolling game)
    - ADC_init() - Analog to Digital Conversion allows the system to get input from the joystick. ADC_init() was provide in EECS120B Lab 08 - A2D (light meter) but I made getADC() using only the Atmega1284 Datasheet
    - EEPROM - EEPROM allows for data to be stored and read from despite turning the power off and on again. This was used to store the highest score. The Atmega1284 Datasheet was essential for understanding and had useful examples. EEPROM is used using eeprom_write_byte() and eeprom_read_byte from #include <avr/eeprom.h>. The use of this library was approved by the TA.

The following source was used to help use this feature : https://www.nongnu.org/avr-libc/user-manual/group__avr__eeprom.html

  - LCD custom characters - The following source was useful in implementing custom LCD characters. https://www.electronicwings.com/avr-atmega/lcd-custom-character-display-using-atmega-16-32-

- io.c and io.h - Io.c and io.h implement functions that allow for data to be easily written to the LCD screen. Both files were provided in EECS120b Lab 07 - LCD
- nokia5110.c, nokia5110.h, and nokia5110_char.h - Nokia5110.c and nokia5110.h implement functions that allow for data to be easily written to the Nokia 5110 screen. Nokia5110_char.h contains the pixel data for ascii characters. Below are a few points to take note of.
  - All three files are from https://github.com/LittleBuster/avr-nokia5110
  - I changed nokia_lcd_write_char() to extend the area of pixels written to everytime a char is output.
  - I created 2 new functions nokia_lcd_inverse() and nokia_lcd_inverse_string(). These functions accomplish the same as their counterparts nokia_lcd_write_char() and nokia_lcd_write_string() except the output is inverted.

Component Visualization



Technologies Learned

In this project, I learned how to design and implement a multi-task synchronous state machine. More specifically, I learned to decompose a complex design into smaller scale tasks that can be run synchronously and how to convert them to code. Furthermore, I learned how to read datasheets, which made implementing the joystick much easier. I learned that by changing ADMUX, the port the ADC reads from changes, allowing us to read from multiple input ports. In addition, I learned how to create custom LCD characters by storing them. Similarly, I learned how Nokia 5110 screens convert ascii characters to pixels and how to manipulate this process to an extent. EEPROM is another technology that I learned to use. Lastly using PWM functionalities taught me how to use it change the frequency the speaker output and how long it would do it for.

In summary, I learned how to :

- use ADMUX to read from multiple potentiometers on different ports
- design and use custom LCD characters
- manipulate pixel output to translate ascii characters to pixels on the Nokia 5110
- use EEPROM to store data
- use PWM to manipulate frequency and duration of speaker output

All Files and Documents

https://drive.google.com/drive/folders/19VKn0H7n4cDavoql_fRwUQD7CTyKowdB?usp=sharing

Sources

- Timer.h
  - EECS120b Lab 06 - Timer synchSMs (Reflex game)
- PWM
  - EECS120b Lab 09 - PWM (Music player)
- Task Scheduler
  - Zybooks Chapter 5.7
  - EECS120b Lab 11 - scheduler (Scrolling game)
- Joystick (ADC)
  - ADC_init()
    - EECS120B Lab 08 - A2D (light meter)
  - Atmega1284 Datasheet
- EEPROM
  - #include <avr/eeprom.h>

    - https://www.nongnu.org/avr-libc/user-manual/group__avr__eeprom.html
    - Atmega1284 Datasheet

- Custom LCD Characters
    - https://www.electronicwings.com/avr-atmega/lcd-custom-character-display-using-atmega-16-32-
    - io.c and io.h
        - EECS120b Lab 07 - LCD
    - https://maxpromer.github.io/LCD-Character-Creator/

- Nokia 5110
    - nokia5110.c, nokia5110.h, and nokia5110_char.h
        - https://github.com/LittleBuster/avr-nokia5110
    - https://www.electronicwings.com/avr-atmega/nokia5110-graphical-display-interfacing-with-atmega16-32
    - https://learn.sparkfun.com/tutorials/graphic-lcd-hookup-guide