Course Title: CEE-345 Microprocessor System Design

Project Title: Countdown Timer and Stopwatch

Student Name: Jacob Hillebrand and Alison Zimemrman

Instructor Name: Dr. Cheng Liu

University of Wisconsin-Stout

Menomonie, WI 54751

May 09, 2019

# TABLE OF CONTENTS

TABLE OF CONTENTS 2

1.0. PROJECT DESCRIPTION 3

2.0. BILL OF MATERIALS 4

3.0. DEIGN PROCESS 5

4.0. STUBLING BLOCKS AND CONCLUSION 13

# 

# PROJECT DESCRIPTION

This project was a timer and stopwatch. The brains of the system is the KL25z Freedom Development Board. The 2x16 character LCD screen displays the options to press to do certain commands given by the 16 key keypad. After you select what mode you want you can enter a time if timer selected or start and pause the stopwatch. If in the timer mode you can pause the timer with a key and when paused you are given the option to resume or quit to the beginning menu.

**1.1. *Introduction***

# BILL OF MATERIALS

|  |  |
| --- | --- |
| Part Name | Cost |
| 2x16 LCD screen | $6 |
| Diligent Keypad | $10 |
| KL25z Freedom Board | $31.11 |
| Potentiometer | $1 |
| Total | $48.11 |

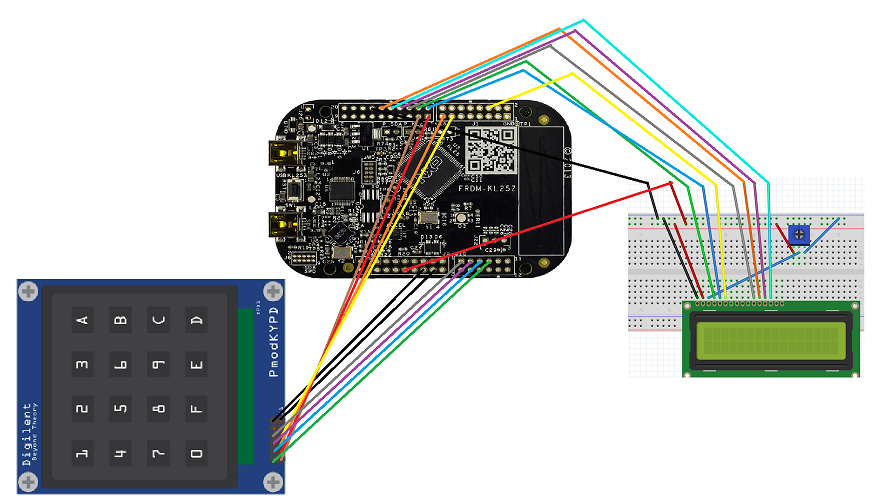
# 3.0. DEIGN PROCESS

# 3.1. HARDWARE DEVELOPMENT

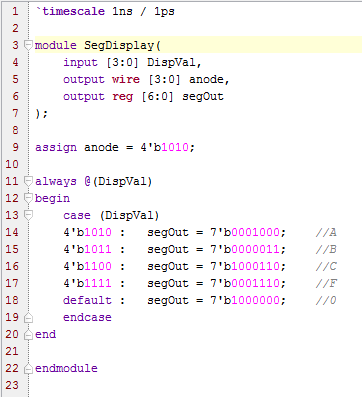
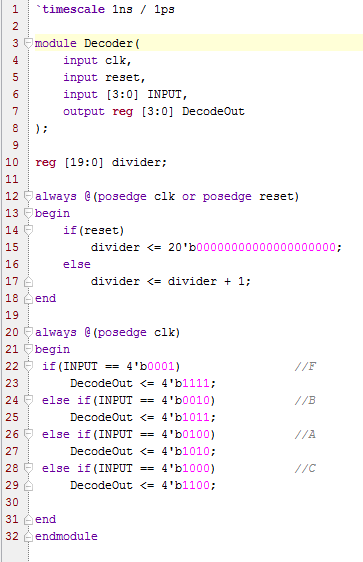
# RF Receiver Input

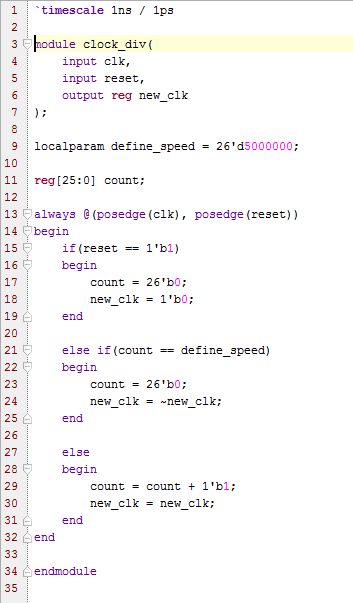
The receiver used in this project was a simple RF receiver that is triggered by a 315MHz Key-fob. The wiring of this receiver has the 4 inputs connected to the Basys3 board and the receiver being powered by a 9V battery supply. When programming this for testing, I set up that it would output on the 7-segment display to what directional input was received. This way, I was double checking that I would know from what input I would have later on to set the motors to work with. I set them up in a way that from a distance, one would be able to see what input is being given. It is configured that forward displays “F”, backwards displays “b”, left displays “A”, and right displays “C”. The directions also correlate with the buttons on the key fob such that forward is the “A” button, backward is the “B” button, left is the “C” button, and right is the “D” button. The following are images of the code that I used on this project:

## 3.2. WIRING DISGRAM



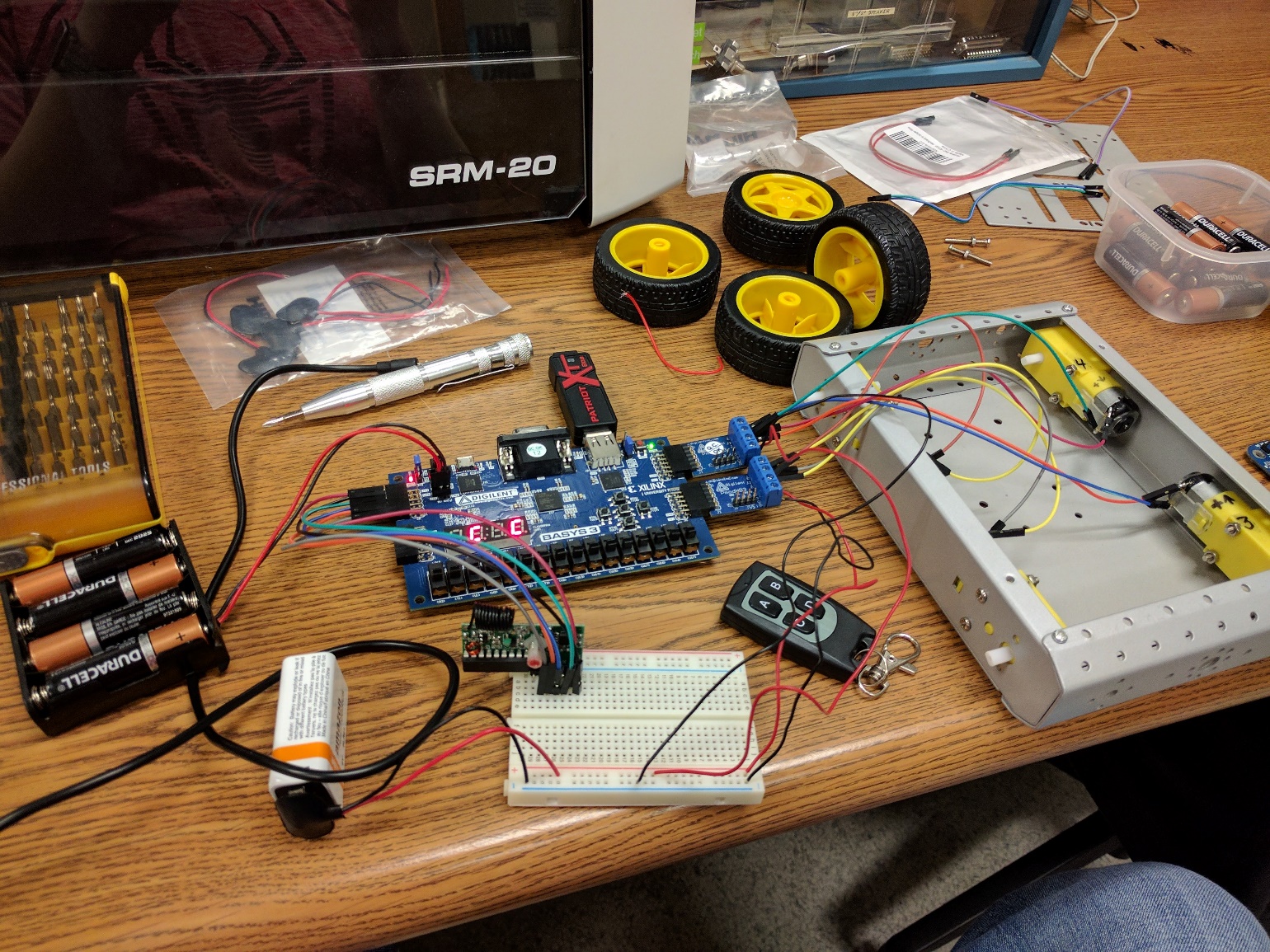
## 3.3. SOFTWARE DEVELOPMENT



The Decoder takes the input and changes it to have the output that we want to have for forward, backward, left, and right. The SegDisplay uses a case statement to output on the 7-segment display the letter that correlates with what direction it is going to be moving.

# 4.0. PROJECT ILLUSTRATION

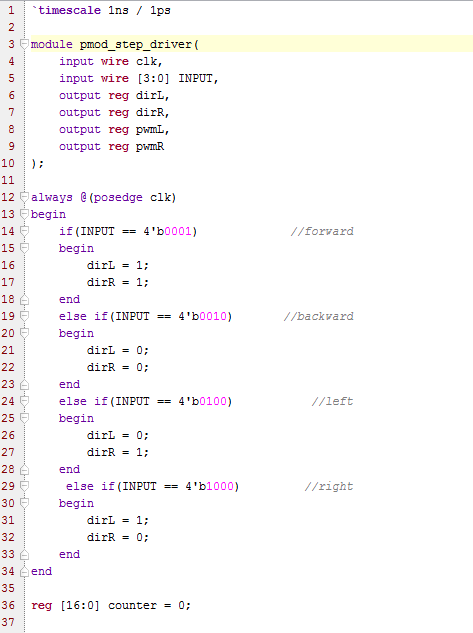
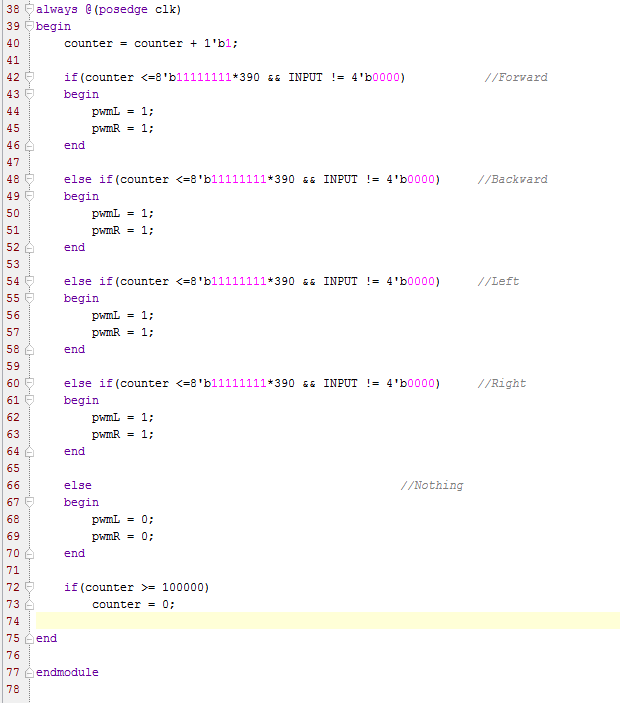


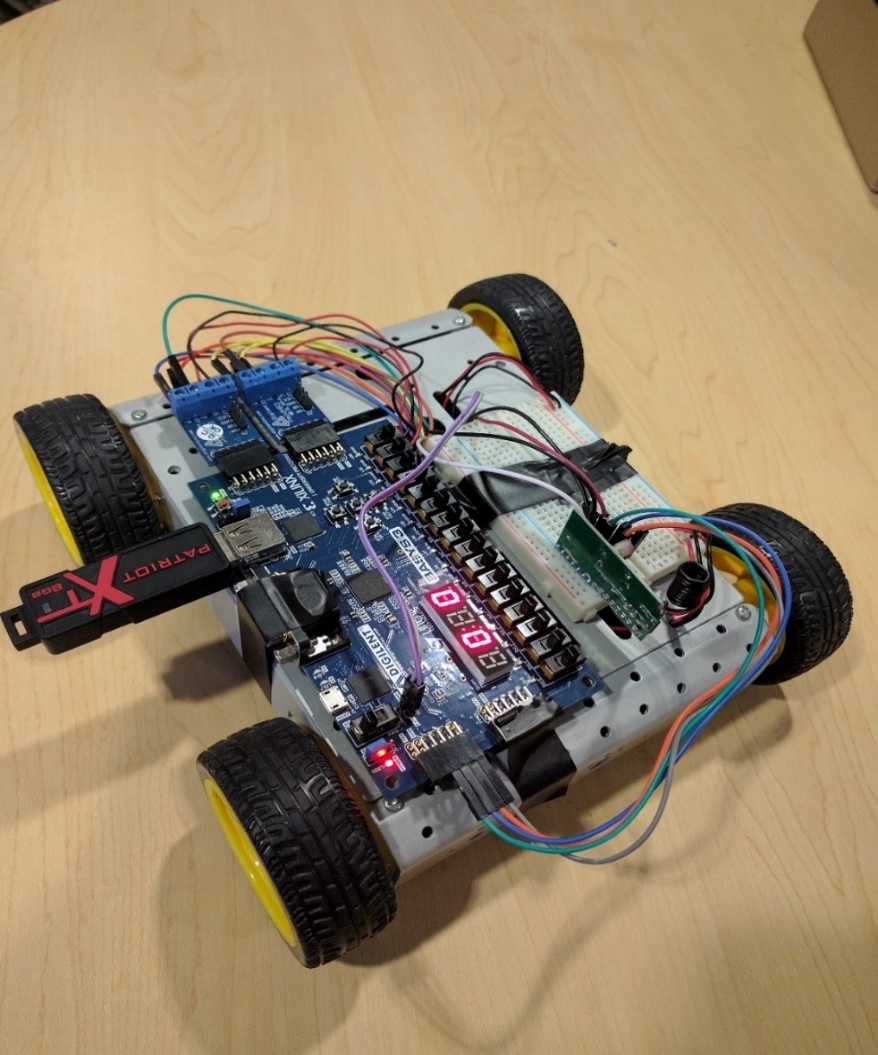
# Motor Controls

With controlling the motors, I first wanted to reimplement the dc motor lab that we had used in the past and set it up to use the toggle switches for each of the four directions that the car will have. Doing this required to reset up how it decided to activate the motors. To do this, I used a system of if-else statements to test if the switches were activated or not.

The 4-switch system would not only turn the motors on from the switches, but it would also set the direction of the motors spinning. The way I programmed this, the drive system is similar to a tank drive. This means that for forward and backward, the left and right motors spin in the same direction. Furthermore, for left and right directions spin the left side motors in one direction and the right side motors in the opposite direction. This helps to give the RC car the ability to turn on itself very tightly. Then, I used more if-else statements to test the if a switch was activated and then turned on the PWM for the left side and the right. If there isn’t a switch activated, the PWM signal is deactivated, turning the motors off. The code explained here is displayed below:

#### Motor Driver Module:

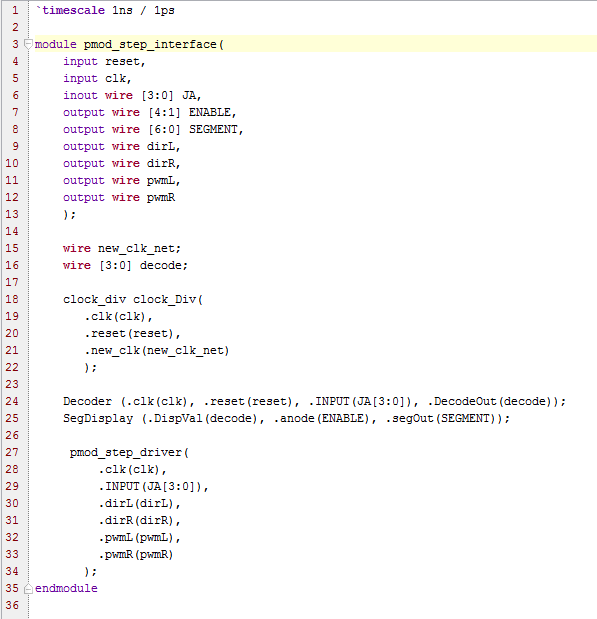


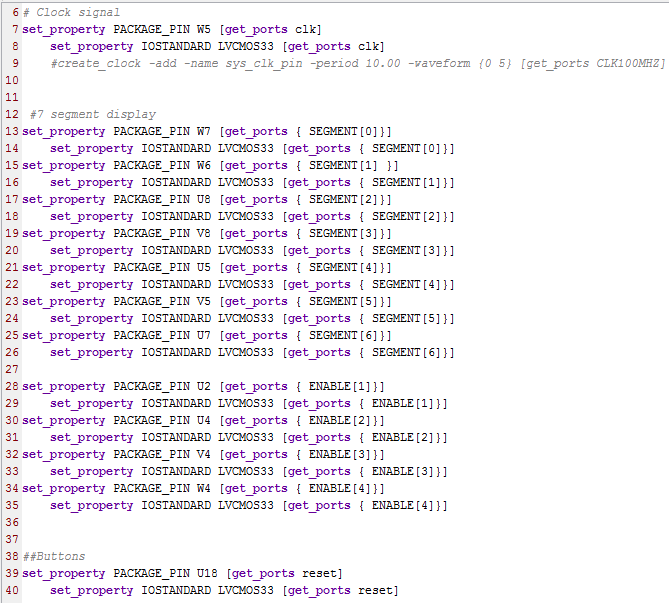


# Combination

For the combination of these two different sections of code, I used an interface module that used all of them inside it, allowing them to all work and communicate effectively with one another. We can see that using this together was the simplest way to combine these codes and still have them maintaining functionality that they have. The code is listed below:

#### Top Module:

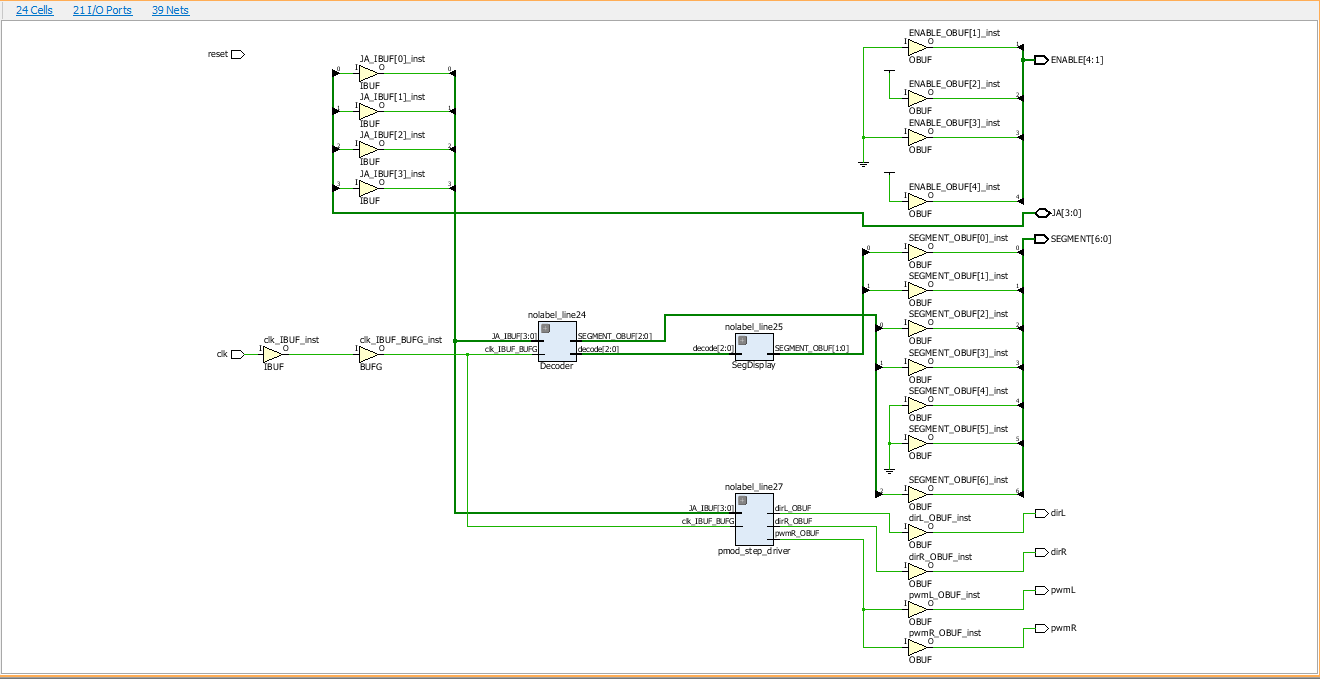




#### Constraint File:

#### 

## 3.4. Schematic of the code



# 4.0. STUBLING BLOCKS AND CONCLUSION

In summary, this project was quite challenging overall. There were a lot of unknowns to me when I started this and I learned through process of trial and error to get this project functioning properly. I learned quite a lot from this project and a lot about how I will approach these projects in the future. I also realized how much I took away from this class in learning how to code in this language. It is a great feeling finally solving the problem for a project and getting it all working correctly. The most challenging part of this project was the wiring of how to connect the board to the motors. Originally, I was trying to use a part that came with the car kit as I thought it would be easier. But when that was not working and Dr. Liu recommended I try out the PMod HB3 to control the motors, it turned out to be easiest to use these parts that were built for the Basys3 and other boards with these motors than to try to use something not designed specifically for our boards. I will take this into the future when using thinking of parts to put together for a project. Overall, I think I spent over 30 hours plus working on this project to bring it together into a working invention.