Course: CEE-325 Digital System Design

Project Title: 24 Hour Clock Design

Student Name: Jacob Hillebrand

Instructor Name: Dr. Cheng Liu

University of Wisconsin-Stout

Menomonie, WI 54751

December 20, 2018

# 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

The end goal of this project is to create a functional 24-hour clock design. The design will run on the Basys 3 FPGA, and will utilize both the onboard 7-segment display as well as an external Pmod CLP LCD to display the clock. The clock will have 4 button controls; button one will reset the clock to 00:00:00, button two will increment the minutes on the clock, button three will increment the hours on the clock, and button four will switch the 7-segment display between displaying hours and minutes, and minutes and seconds. This entire design can be realized with a singular Verilog module, and will utilize several designs we used previously in class.

**1.1. *Introduction***

To begin designing this project, I first started by creating a clock design to run on the Basys 3’s 7-segment display. First, I created a clock divider to not only drive the design, but also as a means to control when the seconds on the clock would increment. I then set up a button to increment the minutes on the clock, and another button to increment the hours on the clock (both of which required special techniques to debounce the button inputs). In addition, I set up a third button to make the 7-segment display show minutes and hours (instead of the default hours and minutes), as well as a fourth button to reset the counter to 0. With this framework set up, I then set up several blocks of code to enable and use the seven segment display, so that it would display all of the proper values.

With the previous design complete, I then set out to add the Pmod CLP LCD to my design. Thankfully, the LCD could draw from most of my clock framework, so all I had to design for this part was the display block. This required adding the JB and JC package pins to my design, in order to power the display. The display had several connections; an enable pin, power and ground pins, a register select pin, a cursor select pin, and 8 data pins. After looking through some example code, I was able to realize a design that would properly display by clock. The entire design will be described below.

# BILL OF MATERIALS

|  |  |
| --- | --- |
| Part Name | Cost |
| Basys3 Board | $79 (Free for Class) |
| Pmod CLP LCD | $38(Provided by Instructor) |

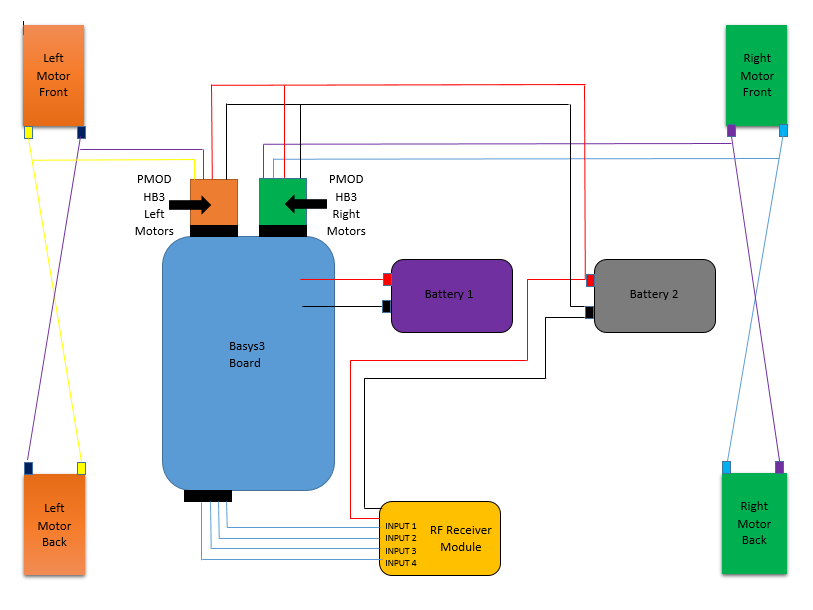
# 3.0. DEIGN PROCESS

# 3.1. HARDWARE DEVELOPMENT

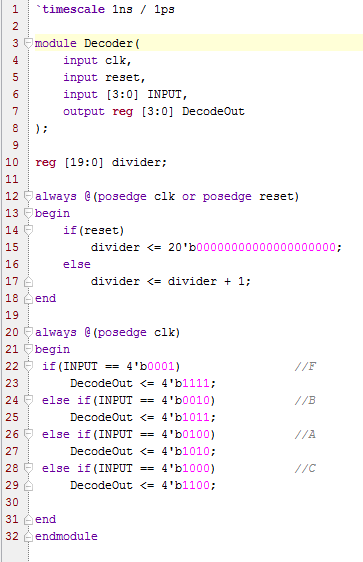
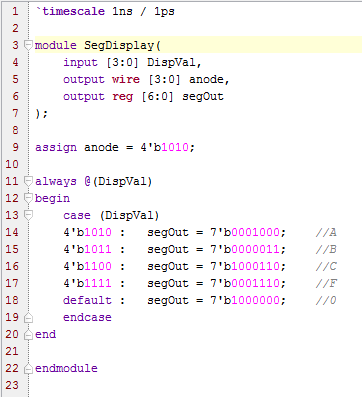
# RF Receiver Input

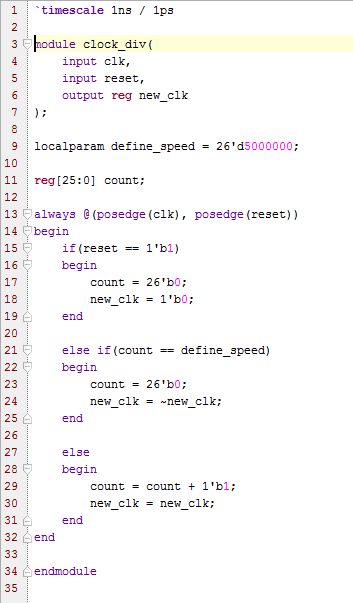
The receiver used in this project was a simple RF receiver that is triggered by a 315MHz Key-fob. The wiring of this receiver has the 4 inputs connected to the Basys3 board and the receiver being powered by a 9V battery supply. When programming this for testing, I set up that it would output on the 7-segment 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



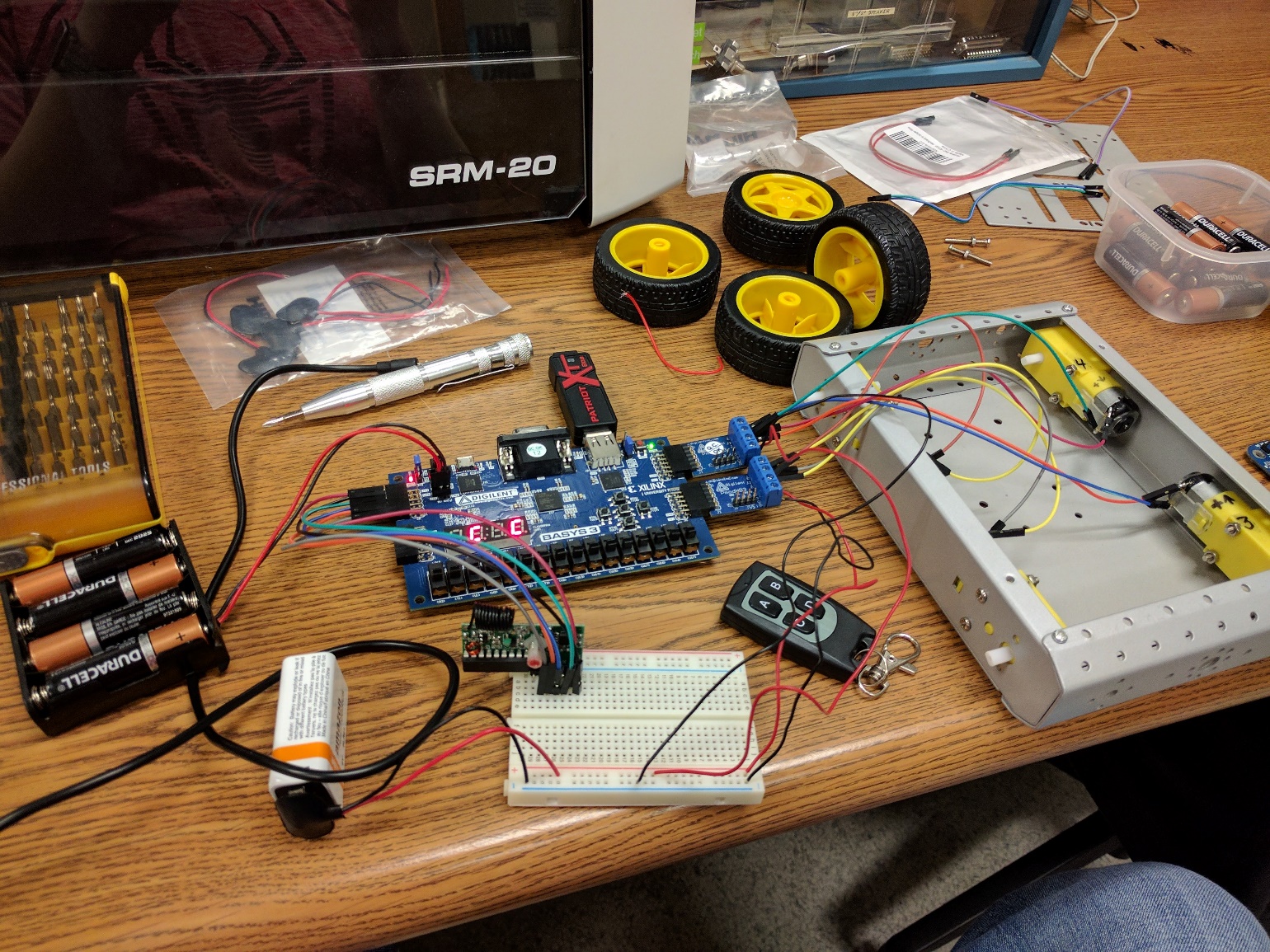
## 3.3. SOFTWARE DEVELOPMENT





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

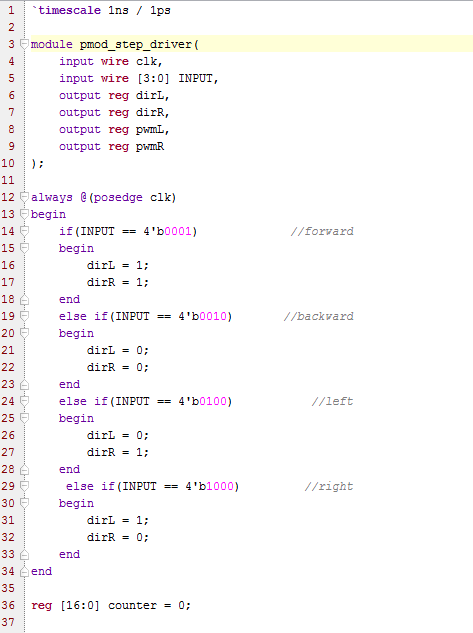
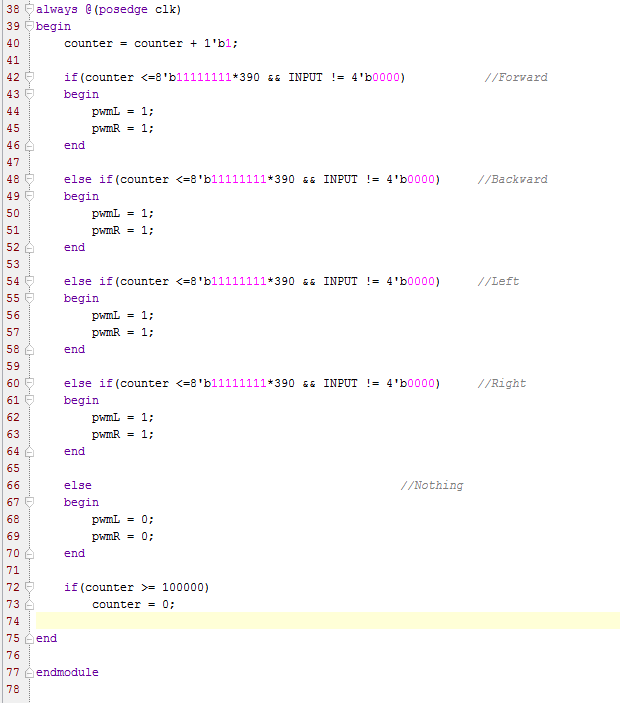


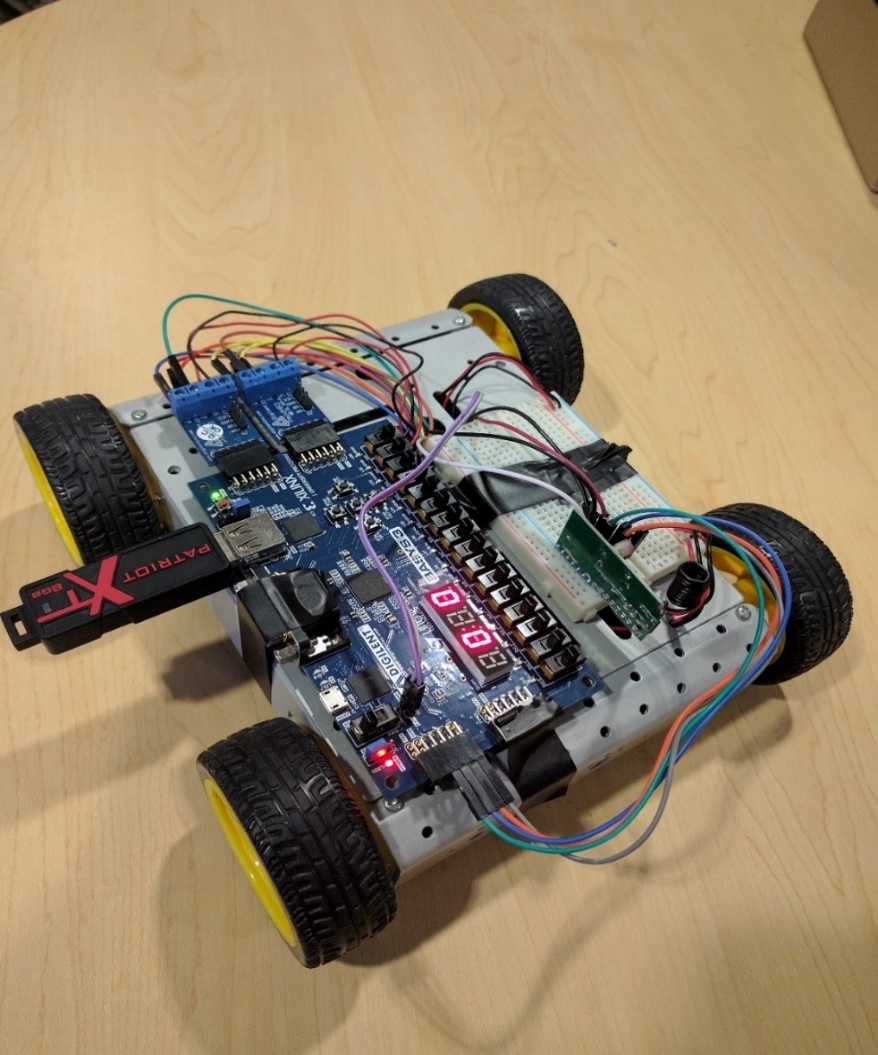
# Motor Controls

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:

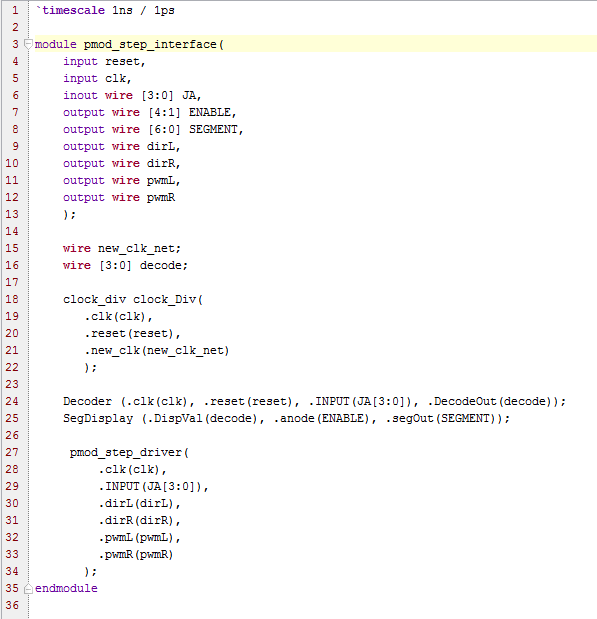


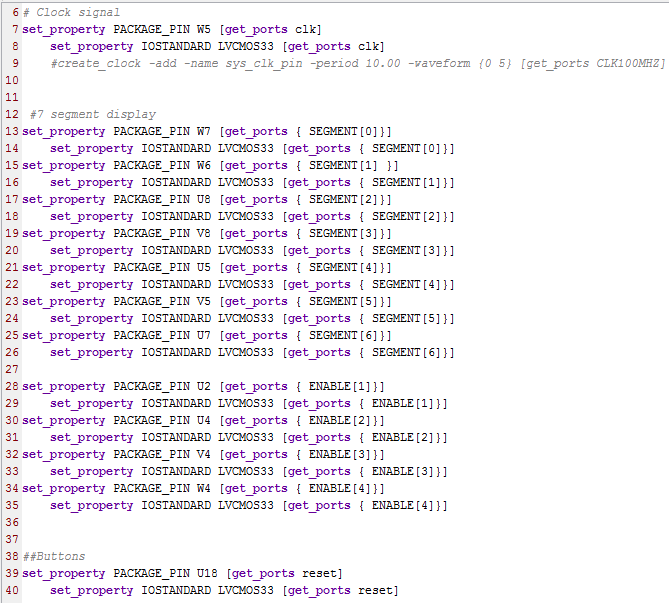


# Combination

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:

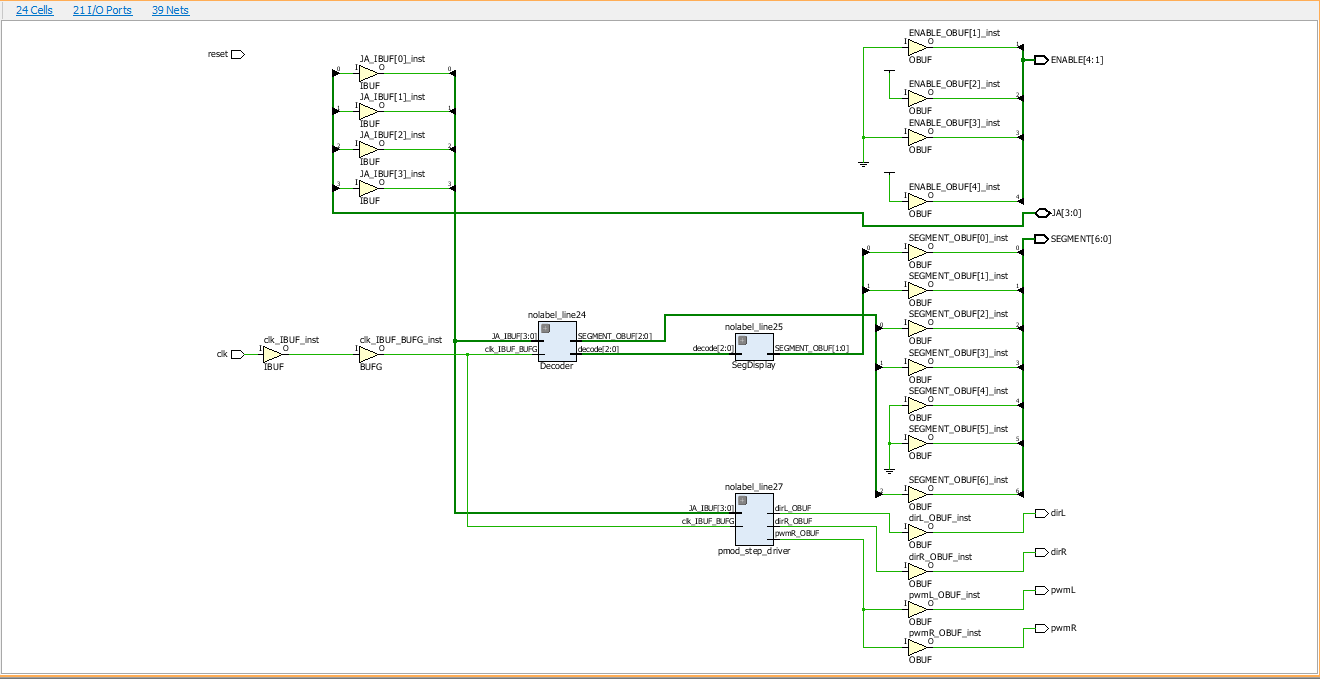




#### Constraint File:

#### 

## 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.