MEDSense: Accessible Pill Dispensing Device- Week 6 Report

Work Completed

After careful review of the performance and programming of servo motors and stepper motors the group came to the conclusion that it would be easier to use servo motors instead of the stepper motors we purchased. The circuits to drive the stepper motors add increased problems since the circuit is more intricate and thus more places for there to be mistakes. I also reviewed extensively the complexity of code for stepper motors versus servo motors. The code is much simpler for servo motors and thus more attractive for this project. Apart from the overall simplicity of servo motors is the space they save. Using the linear actuator would take up a large amount of space which would alter the shape of the overall device. Using the servo motor will save space which is the ultimate goal of the project.

Once the final decision on which motor to use was made, I began investigating the theory behind the servo motors. Servo motors are controlled by using a square wave. At the top of the square wave is the operating voltage which will be 5V in our case. At the bottom of the square wave is 0V. When the voltage is at 5V the motor is moving. When the voltage is at 0V the motor is stopped. Knowing the theory behind this allows us to control the angle to which the motor moves. To move it to a designated angle we must make the Up-Time the correct time. Figure 1.1 outlines how the servo motor is powered.

\[ \text{frequency} = 50 \text{ Hz} \]

\[ \text{Period} = \frac{1}{\text{frequency}} \]

\[ = \frac{1}{50 \text{ Hz}} = 20 \text{ mS} \]

\[ \text{Period} = 20\text{mS} \quad \text{Down-Time} \quad \text{Up-Time} \]

\[ = 10 \text{mS} \quad 10 \text{mS} \]

Figure 1.1- Controlling Servo Motors
Sending a time to the microcontroller does not work. Telling the PIC microcontroller to power the motor for a period of time must be done indirectly. Instead of sending a time to the microcontroller you must send an amount of instructions. Each processor has an instruction frequency. This means that instructions are sent once every few cycles done by the clock. For our particular microcontroller there is an instruction sent every 4 clock cycles. The clock frequency of the PIC16F747 is 20MHz. Knowing these two numbers I was able to find that the instruction frequency is actually 5 MHz.

I found documentation online that said that an Up-Time of 1.0 ms would result in moving the motor 0 degrees. An Up-Time of 2.0 ms results in a movement of 90 degrees. I also found that the frequency of the motor is 50 Hz which results in a period of 20 ms. If the Up-Time is 1.0 ms the Down-Time must be 19.0 ms. Knowing all of these numbers is critical in controlling the motor accurately.

Since the instruction frequency and the delay I can find the amount of instructions. This number is then put into a delay to form a square wave. The amount of instructions in thousands is placed in Delay1KTCYx function and the amount of instructions in hundreds is placed in the Delay100TCYx function. For example a delay of 92500 instructions would be programmed as follows:

```
Delay1KTCYx (92);
Delay100TCYx (5);
```

Apart from programming the actual square wave I know how to program the servo to hold for a period of time. This is done with a simple for loop. To hold for 1 second the for command would look like:

```
for (count=0; count < 50; count++)
```

The count variable counts each time the loop is run. If the period is 20 ms then 50 periods would result in 1 second. I plan on varying this time in the actual application because we may want to hold the blade in place after cutting for an extended period of time. The code has been finished and we have learned how to connect the motor to the PIC. Figure 1.2 shows a schematic on how the motors are attached to a microcontroller.

---

**Figure 1.2- Schematic of Motor Circuit**
Future Work

Over the coming weeks I plan on testing the motors and optimizing their performance. This is an important element to finish because the rest of the mechanical aspects of the project hinge on it. Since the code and schematic are done this process should not take a long time. In the laboratory session tomorrow I plan on finishing the motors and turn my attention elsewhere. Once the motors are finished I want to begin programming the real time clock and the text-to-speech modules. I have learned a lot about I2C programming and feel I can accomplish a lot of programming quickly. Following the programming of the real time clock and the text-to-speech modules I will finish the Bluetooth programming. Once all of these elements are done the programming of the various elements of the design will be done. The final element of programming will involve creating an alarm system. This is the basis of the project and will tie everything together.

Project Overview

I believe that we are ahead in some aspects of the project and behind in other aspects of the project. I think we have accumulated a lot of information on all of the electronic elements of the project but we haven’t applied our information as much as we would like. This changed late this week when we applied our knowledge to controlling motors. The actual programming of the microcontroller will take place tomorrow and the motors should be working by the end of lab tomorrow. Testing the motors will be an important step in finishing the project. I think all of our learning will prove to be beneficial over the next two or three weeks. I feel more confident than ever that the programming will go smoothly and progress will be tangible. I will have a greater grasp of how the project is going after these next two weeks. What the group does this week, over spring break and the following work will determine the quality of the finished project. I believe the group is up to the task and we will be greatly ahead of schedule in three weeks.

Hours Worked

14