Lower Limb Exoskeleton

Tina Nguyen
University of California, Santa Cruz
Bionics Lab
Advisor: Jacob Rosen
Graduate Student: Jared Mednick

Introduction

The goal of the lower limb exoskeleton is to assist soldiers to carry their heavy loads in various terrains. The lower limb exoskeleton aims to reduce stress and fatigue on the body of the soldiers. The development of the lower limb exoskeleton will be lightweight, low powered, and wearable. The design concept will look at the human interface with the exoskeleton. The exoskeleton must react to the human interface quickly using several sensors to indicate where in the structure energy must be stored or released.

Figure 1. Solidworks of the lower limb exoskeleton prototype.

Concept

The human body stores and releases energy during the gait cycle and we want to do the same with the exoskeleton. The exoskeleton will use the soldier's weight and the weight of the load to store and release energy through the exoskeleton. The energy is stored and released during the gait cycle using the pneumatic air cylinders.

The soldiers wear the exoskeleton, so the exoskeleton must react to the human interface quickly. The use of feedback control system will allow the exoskeleton to integrate with the human interface. This will allow the movements between the system and the soldier to be smooth. The block diagram below shows the control system we planned to implement. The prescribed position will subtract the current position of the joint to compute the appropriate actuator commands that will give the desired motion. Take notice of the feedback from the joint sensors, which are needed to compute the torque required.

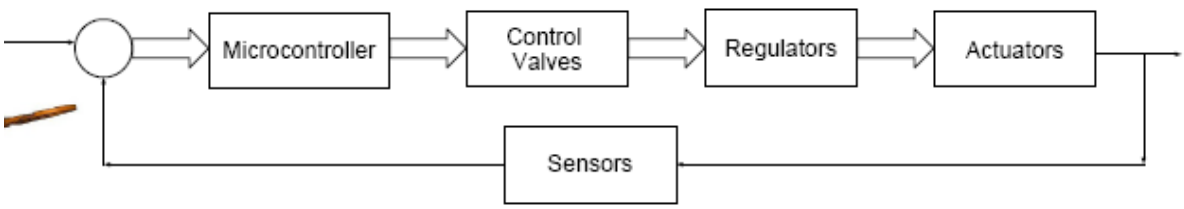

Figure 2. Block diagram of the control system planned

Hardware

For fast prototyping, we use the Explorer 16 Development Board with the dsPIC33FJ256GP710, a 16-bit microcontroller. The potentiometers are the angle sensors for the various joints on the lower limb exoskeleton. Potentiometers are a voltage divider having a three terminal resistor. The angle is linear to the voltage max and min. In our case, one of the potentiometers angles measures from 0 to 340 degrees with the voltage reference of 0 to 3.3V. At 0V the angle is 0 and at 3.3V the angle is 340, so at half the voltage, 1.65V, the angle is 170 degrees. Using the analog to digital converter, we measured the corresponding joint angle. The solenoid valves are the control valves for allowing airflow through. These solenoids are controlled by using several of the I/O port pins of the microcontroller. They will take a digital high or low from the microchip to open or close their valves using a solenoid. The pressure regulator will regulate the air pressure in the air cylinders. The air cylinders are the actuators that will provide the torque needed to help carry the heavy load. The Liquid Crystal Display (LCD) displays the sensor's angle for debugging purposes for early development of the lower limb exoskeleton, which uses the general I/O ports of the microcontroller.

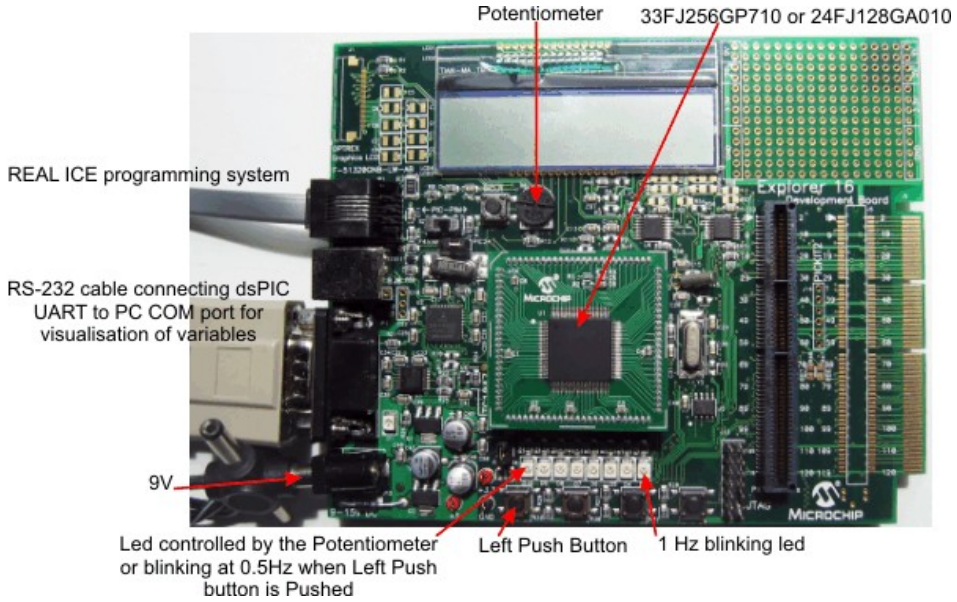

Figure 3. Explorer 16 Development Board

Software

There are two options for writing the software. First option is to use Matlab Simulink and the second option is to use MPLAB IDE that comes with the Explorer 16 Development Board. Matlab Simulink uses blocksets already prescribed to follow certain functions, so it is a quick drag and drop of boxes onto your model to compile, while MPLAB compiles C code written by the user. The PICkit 2 uploads the hex file onto the dsPIC. The choice of using Matlab Simulink is for fast prototyping. Shown below in Figure 4 is how Matlab Simulink compiles the model with the blockset. Create the model, configure to your device, then compile your model to produce the files needed to load onto your target device.

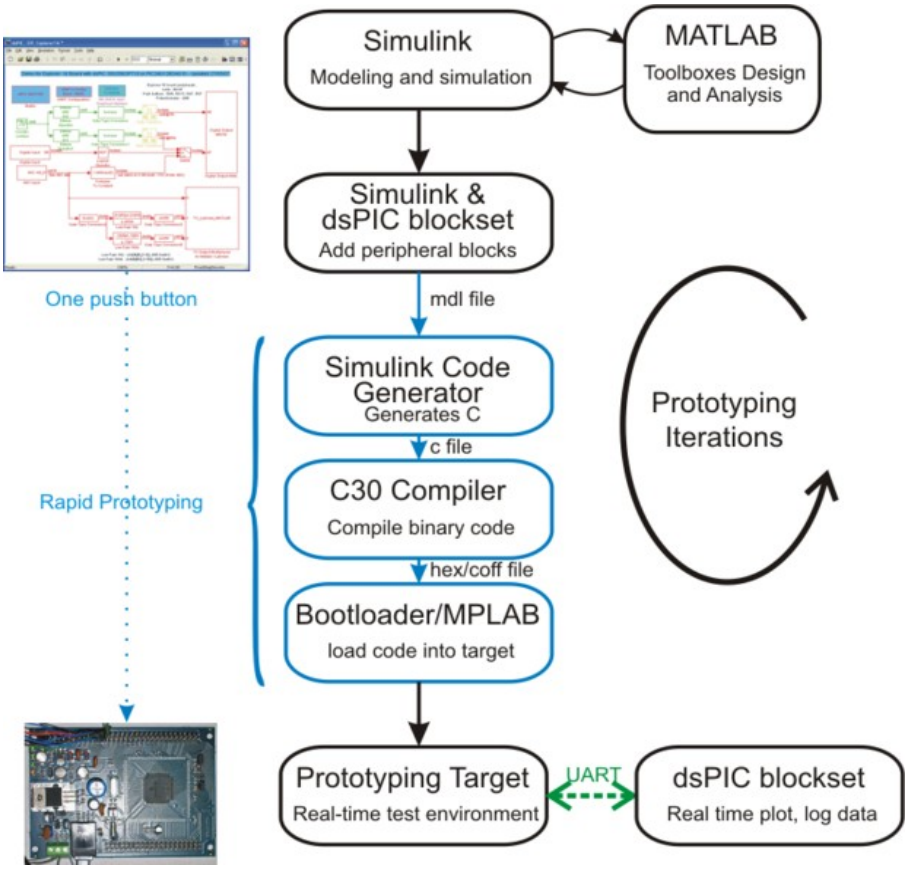

Figure 4. How Simulink works with the dsPIC blockset

Using the Explorer 16 Development Board, the initial setup was to print out the angle of the potentiometer on the LCD, which measures 0 to 340 degrees. The potentiometer would be one of the several sensors on the exoskeleton to indicate the joint angle. The potentiometer uses the analog to digital converter to decipher the angle of each joint. Taking this information, the microcontroller determines the commands needed for the actuators to give the desired motion.

Hysteresis is in place for the angle sensors to control the solenoid valves properly. At certain angles, the solenoid valves will be either closed or open, so there is a gap for noise margin errors and other sources. This will secure that the solenoid valves do not open and close quickly at one particular value. Recalling Figure 2 of the block diagram of the control system, the microcontroller will process the information from the joint sensors to control the solenoid valves.

Conclusion

I have grown through this program. I gained experience in learning a new software by myself, an area that was usually taught to me by professors or teaching assistants. My patience in learning a new program and overcoming obstacles is a lesson I have learned through this program. I was able to display messages on the LCD and implement hysteresis for the solenoid valves using the software. I was able to help put together some of the parts on the mockup and learned how to use the drill press. I learned about control systems and how each component will work with each other. I saw how setting up a project is like, like obtaining the correct parts to installing the software. In the end, I gained a new perspective and appreciation for this project and program.

References

1. Rosen, Jacob. Lightweight Wearable Lower Limb Exoskeleton. January, 2009.

2. Krivts, Igor L. and Krejnin, German V. Pneumatic Actuating Systems for Automatic Equipment: Structure and Design. Boca Raton: CR C/Taylor & Francis. 2006.

Acknowledgements

I would like to give thanks to Jacob Rosen who was my faculty advisor on this project and Jared Mednick who was my graduate student who helped me through this project. I would also like to give thanks to the SURF-IT 2009 program and the National Science Foundation (NSF) for their funding (NSF grant CNS-0732604).

Installation of Matlab and dsPIC Blockset

1. Acquire the following components (do not install yet):

a. **Matlab 2007a** (Any Matlab version is fine as long as you have the right dsPIC Blockset version)

b. **dsPIC Blockset** for Matlab 2007a. A free version of dsPIC Blockset is here, but this version is limited to what you have access to.

c. **C30 compiler** for PIC24, dsPIC30, and PIC33 family from here.

d. **C32 compiler** for PIC32 family here.

e. **MPLAB**

f. **PICkit 2**

Note: Download the dsPIC Toolbox corresponding to your Matlab. If you don't it will not work!

First Installations

1. Acquire Matlab, in this case Matlab 2007a due to the Lubin's blockset only works for Matlab 2007a. Install Matlab 2007a.

2. Install MPLAB and Pickit2 in your **C:\Program Files**.

3. Install C30 compiler to **C:\Program File\Microchip\MPLAB C30** (for C30).

4. Install C32 compiler. Add the compiler Path to your variable environment : Click yes when Installer ask to modify environment.

- When installing C32 included in MPLAB, the path is not added to the path. Add manually the bin subdirectory of the C32 compiler in your computer path.
- Compiler for PIC30 and Compiler for PIC24 share the same directory. This is not a problem, both are working.

5. Install dsPIC Toolbox to Matlab.

6. Type in these commands in Matlab's command window.

- !pic30-gcc –v

Make sure you get these messages or something simular:

- ```
  Using built-in specs.
  Target: pic30-coff
  Configured with: ../../src/gcc-4.0.2/gcc-4.0.2//configure --target=pic30-coff
  --program-prefix=pic30-coff- --enable-languages=c --host=mingw32 --
  build=i386-pc-cygwin
  Thread model: single
  gcc version 4.0.3 (dsPIC30, Microchip v3.00) Build date: Feb 28 2007
  Microchip Language Tool Shell Version v3.00 (Build date: Feb 28 2007).
  Copyright (c) 2006 Microchip Technology Inc. All rights reserved
  ```

Installation of the dsPIC Blockset in Matlab

1. Start Matlab

2. Go to the directory where you unzipped the installation .m file of the dsPIC Blockset

3. Execute it by typing its name at the matlab prompt **>>install_dsPIC.** Another way is to rick click on the .m file and run it. The dsPIC Blockset will be installed into the current directory.

4. Once installed, go to configuration parameters in the simulation tab. Go to the Real-Time Workshop tab and make sure that dspic.tlc is in the System Target File.

5. Go to Matlab's Menu File

Double check that the following files are added to the Path:

- **examples**
- **blocks**
- **dspic**

6. Run one of the examples or all to see if they compile and there are no errors.

How to setup your model for the dsPIC

1. Start Matlab and type **Simulink** in the command window.

2. A library window should pop up to start a new model.

3. Include the following blocks:

- Configure Model of dsPIC
- Generate Code
  ```

- Master
- Configuration

These blocks are located in the 'Embedded Target for Microchip dsPIC' library.

4. Double click Configure Model of dsPIC first.

5. Double click on the dsPIC MASTER block and choose your corresponding PIC you are using. In addition, change any configurations for your dsPIC.

6. You can now start adding any peripherals to your model. Once done with your model you can compile by clicking on the 'Increment Build' button on the tool bar or double clicking on the Generate Code block. Check on the Matlab prompt that it was compiled correctly.

NOTE: If error messages occur, a box will appear but majority of the useful information to debug the error is on the Matlab Prompt.

How to upload hex file to PICkit 2

1. Power on the Explorer 16 Development Board.

2. Plug the PICkit 2 into the computer.

3. Plug the PICkit 2 onto the Explorer 16 Development Board shown below.

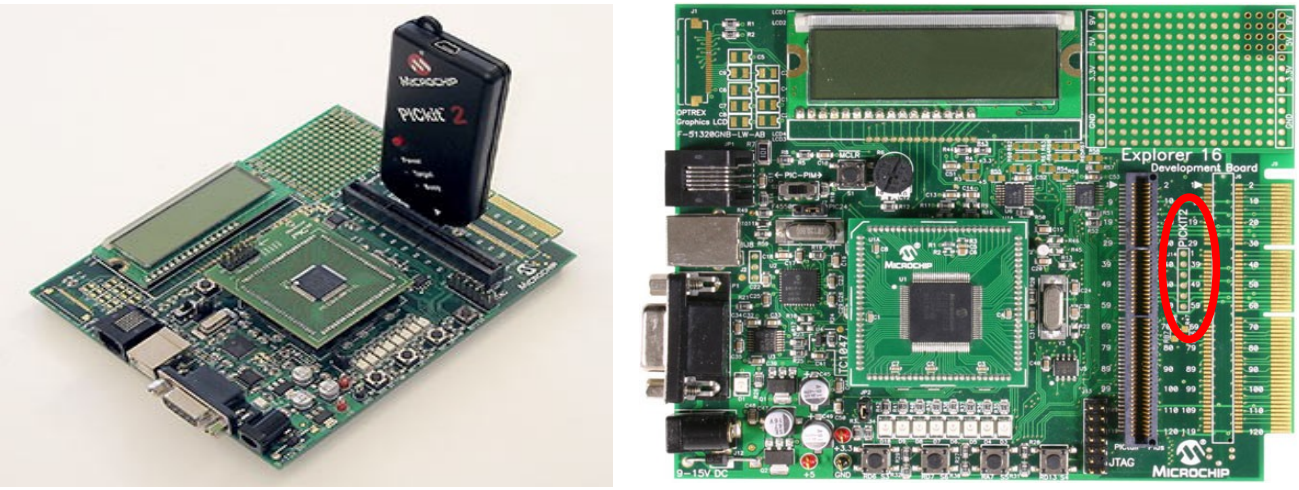

Figure . Placement of the PICkit 2.

3. Open PICkit 2 in the Microchip folder.

4. Choose dsPIC device family that corresponds to the Explorer 16 Development Board.

5. Click on the **Erase** button.

6. Go to **File Menu → Upload Hex File.** Choose the hex file created by Matlab Simulink. The file should be in the directory you specified on Matlab's window.

7. Click on the **Write** button to write the hex file to the dsPIC. A message should indicate that it was written properly to the dsPIC.

Examples/Demos of how to install and use the dsPIC Toolbox

1. http://www.kerhuel.eu/download/dsPicScreencastFromMariano.mov
2. http://www.kerhuel.eu/wiki/Category:Example

Project

Option 1: Matlab Simulink

Model File Name: TestLCDWrite.mdl in the **LCD folder**

LCD

Using the C Function Call block

- Name the block the function name going to be called
- Add the source file to the model
- There are outputs and inputs

To add the source files:

- Add files by: Simulation → Configuration Parameters → Real Time Workshop → Custom Code → Include list of additional: Source files → add files here

To display messages on the LCD include these files:
defineLCD.c
TinaLCD.c
OpenLCD.c
WriteCmdLCD.c
WriteDataLCD.c
BusyLCD.c

delay.c

These files should be in your current directory where your model is. Use Init_LCD and lcd_data. Lcd_data block displays your messages onto the LCD. These functions are located in TinaLCD.c. Using the ADC block, you just need to indicate which channel it is at and the output of the block is the aftermath of using the ADC. The output value is saved in a memory block that the lcd_data uses to compute the angle and the voltage. In addition the value is also read into the relay.

The Relay block is for hysteresis. The output is not a Boolean value, so you need to add a Bitwise AND block and use a Conversion block and change it to Boolean value. This value can be put to an Output Write block to your peripheral, in this model the led at the bottom will light up. The led is driven high.

Option 2: MPLAB IDE

Microchip has a tutorial on using this program for the Explorer 16 Development Board.

Click here for User Guide

It is located in Chapter 2. Explorer 16 Programming Tutorial

1. Create the project workspace and add your files to your workspace. Always include the gld file for your targeted microchip.

2. Add your header and source files.

3. To build your project, click on build project.

NOTE: Common places are program file -> microchip -> mplab c30 has most of the gld and h files for different dsPICs

The project is in the dsPIC33F Code Examples2. This includes all the files needed to compile the project. This code implements the LCD to display the angle and voltage of the potentiometer already set on the board. LED 7, furthest on the left is to indicate the solenoid being open or closed.