Mitchell Barton

ECE 578

Homework 2

11.1.12

**Parallax Laser Range Finder (LRF) Development Continued**

**Introduction**

For homework 1, I acquired and began testing the laser range finder (LRF) from Parallax. I discussed the operation of the sensor, my test setup and results, and the issues and future plans. For homework 2 I was able to begin imaging with a new test setup, mount the sensor to the MCECS bot upper body, and demonstrate functionality after integration within the planned architecture. Once this was complete, we determined that the wheel encoders needed to be integrated so I began planning that as well. Also, our team and the MCECS bot sonar team began using Git within the PSU computer action team’s Red Mine project page for version control of our code as well as project documents.

**Imaging with the LRF**

Previously I had been unable to image the LRF due to limitations with the Arduino software serial library. Because I was using an Arduino Duemilinove there were no dedicated hardware serial ports available to communicate with the LRF, given that I was using the primary serial port for communications with the PC. Using digital PWM I/O pins and the software serial library from Arduino limited me to communicating at only 9600 baud and the LRF cannot transfer image data at that rate with the included LRF image viewer application. Once I switched to an Arduino Mega 2560 microcontroller with multiple hardware serial ports, I was able to communicate at 115,200 baud which is required to transmit image data with the LRF viewer application. A test image with 160 x 128 resolution within that application is shown in figure 2.

Additionally, I was able to confirm the factory calibration with the LRF image viewer application. There are instructions to recalibrate if the laser diode is out of alignment with the focal plane, but thankfully that is not necessary because ours is in alignment, as shown in figure 1. If it were out of alignment, the laser spot would not be centered in that horizontal band of the focal plane.

\\pdx-hood\Advanced_Dev\Barton\School\ECE578\LRFverify.bmp

Figure 1 - LRF Calibration Check

The LRF is quite a powerful sensor given that it can use the camera in a general capacity as well, however we have chosen not to use the camera at this time. It requires a piece of hardware called the prop clip from Parallax to program their propeller processor. All of the existing code is available online, so this could make a very good project in the future.

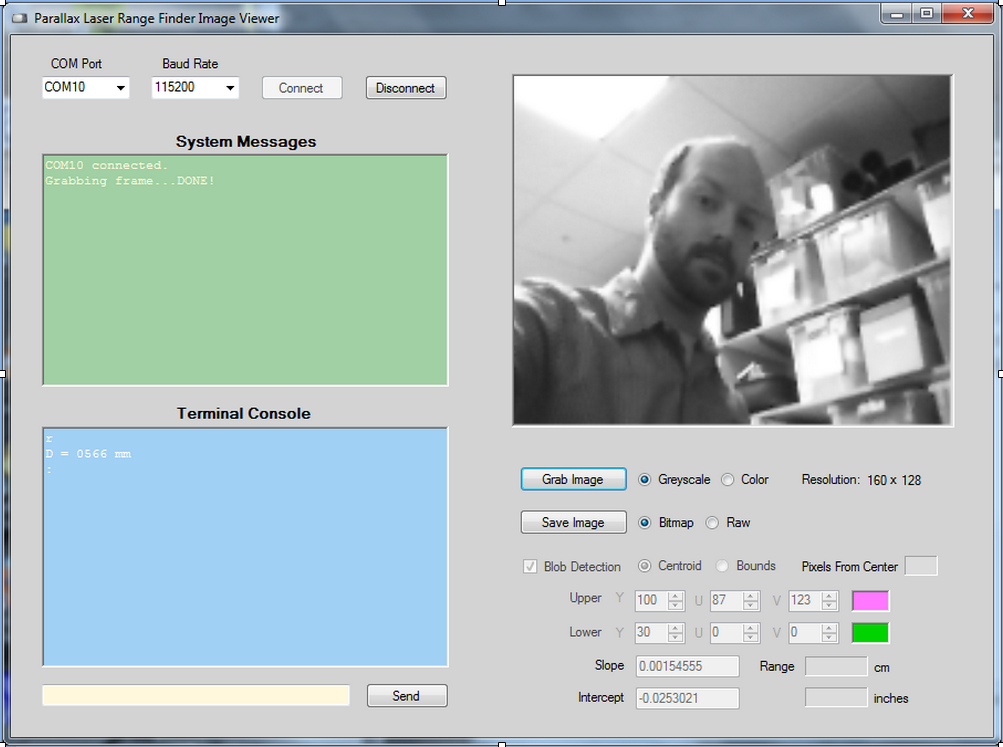


Figure 2 - LRF Viewer Application

**LRF Integration**

At this time, we wanted to install the LRF on the MCECS bot and verify operation once integrated. The location of the sonar sensors was chosen to be the four corners of the base, with 3 sensors at each corner aligned 45 degrees with respect to one another, giving the bot full coverage with the sonar. Given this setup, we determined the best location for the LRF to be below the middle plate which is part of the upper body, for several reasons:

1. We wanted a different location than the sonars vertically so that we could capture potential differences in distance measurements along the vertical axis.
2. We determined it would face forward because we have coverage behind and to the side with the sonar sensors, and it would mostly be moving forward so extra information there would be good.
3. The minimum distance the LRF can measure is within the space claim of the base so we don’t have to worry about an object being in front of the MCECS bot but not giving a return to the LRF.

In order to mount the sensor in this location, some hardware design was required. The middle plate already had a hole pattern for the L brackets used elsewhere, so we used 2 of those brackets, shown in figure 3.



Figure 3 - 2"x5/8" with 4-holes L bracket

Additionally, we needed an interface plate to mate the LRF sensor itself to those brackets. I designed and prototyped a custom aluminum plate shown below in figure 4. The final installation on the MCECS bot is shown below in figure 5.

One issue worth noting was that the original design was bent badly during machining and consequently torqued the LRF board enough to throw the laser diode out of alignment! After re-machining and carefully installing while checking alignment with the LRF viewer tool I was able to install the LRF without creating a misalignment.

Figure 4 - LRF Interface Plate Drawing

5.1”

3.7”

0.4”

0.7”

3.2”

3.62”

1.25”

Ø0.22”

Ø0.10”

0.15”

0.65”

2.7”

2.7”

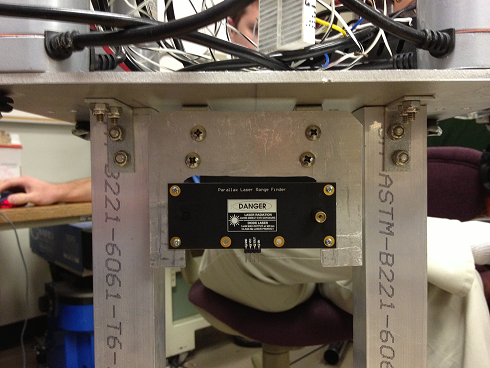


Figure 5 - Final LRF Installation

**Wheel Encoders**

Each wheel has an encoder attached to it via a gear train, and these will be used to determine the position and speed of the wheel during operation for closed loop control. The encoders and motors are connected to Basic Micro RoboClaw motor controllers, which are in turn connected to an Arduino Uno microcontroller. In order to proceed with integration and testing of the sensors, we need to be able to operate the wheels. Following are the tasks which I will accomplish in the next week to support encoder development for wheel operation:

Harness build – Each of the four encoders requires a connector to interface to the RoboClaw motor controller, and I have determined that a 4 pin 0.100” female header is needed, such as the one shown in figure 6.



Figure 6 - 0.100" female header

Initial Testing – the RoboClaw has an extensive command set when used in conjunction with their Arduino libraries, but we will be using a small subset of those commands, named ‘motor control by quadrature encoding’, which rely on feedback from the encoders in the RoboClaw mode called ‘packet serial’. I will test out the commands which drive with speed and distance, both individual and mixed mode, specifically commands 41-43. Because of the gears and the fact that the speed and distance are in units of encoder pulses, I will determine the relationship between encoder pulses and actual distance on the ground driven by the bot, and implement an abstraction layer in software such that commands are given to the arduino in feet rather than encoder pulses.

**System Architecture**

While this wasn’t a spectacularly productive couple weeks as far as coding and implementing control over the MCECS bot with sensor feedback, there was a significant amount of foundational development, namely meeting and coordinating with the sonar sensor group as well as integrating the sensors with the MCECS bot. By meeting with the sonar group we were able to determine where everything would be physically located as well as how the hardware and software architecture would be arranged. There were some requirements for pin count that led to the sonar and LRF being on an arduino Mega, while the RoboClaw motor controllers are controlled by an Arduino Uno, with the microcontrollers being connected to a master PC with the main program running on it. This arrangement is shown in figure 7 below.

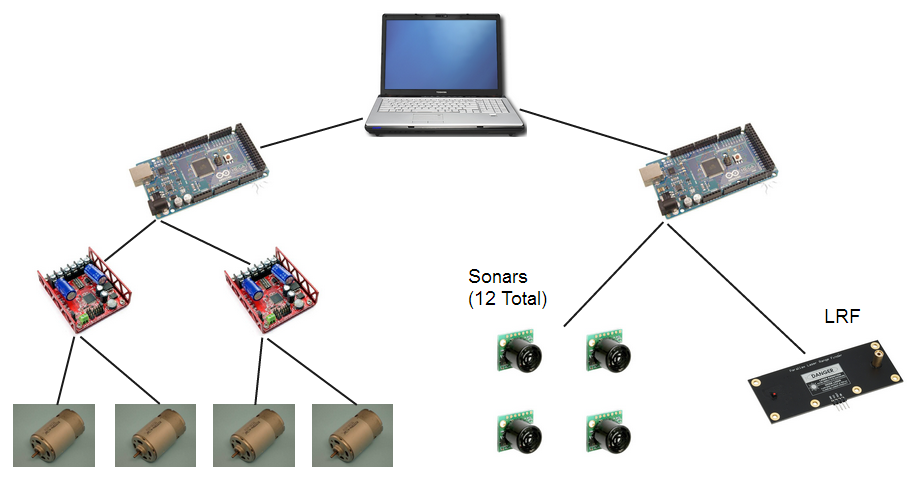


Figure 7 - MCECS Bot Electrical Architecture

**Software Repository**

One main challenge throughout this project has been obtaining the code from previous students, and in the condition that it was supposedly in. Additionally, working with several other programmers on the same software while we are all on different schedules would be extremely challenging without some sort of controlled repository. This should not be an issue at all given the multitude of version control suites available, and in fact the PSU CECS department hosts a project page for this very purpose called the CAT redmine system. It allows authenticated users the ability to upload and share files, as well as maintain a code repository using one of several available VCS suites. We have chosen to use Git, primarily because of it’s wide use and robust support. We have discussed processes which will allow us all to edit and maintain code, and the best part is that all our work will remain once we graduate in the exact same state we left it in. There will also be a history of all the changes and work we did since we have been using the repository. This is hard to show progress on, but the website is included in the references, and the accomplishment really is getting everyone setup on there, and learning how to efficiently use it, which is still in process.

**References**

1. MCECS bot project page; <https://projects.cecs.pdx.edu/projects/roboticsclub-mcecsbot>
2. Git Book; <http://git-scm.com/book>
3. Parallax LRF Homepage; <http://www.parallax.com/Store/Sensors/ObjectDetection/tabid/176/ProductID/774/List/0/Default.aspx?SortField=ProductName,ProductName>
4. Arduino Homepage; <http://arduino.cc/>