Summary of Week 7

We should be in Week Eight now, but we have not yet picked a time to make up our lost session. If we don’t find a spot for a makeup, next week will be our next session.

## Weekly Summary

We didn't come to a consensus on continuing the programming meetings, so I will be preparing a summary to send out that talks about what we have accomplished and what we would have to have added to complete our project as we originally conceived of the idea.  I will also be writing up the whole project and uploading it to my own github account.  I will send out a note with this is done.

Looking at what we did, in no particular order.... we set up a project structure using the ant utility - ANT is an acronym for Another Neat Tool.  This involved creating an xml file with instructions on how to create the file and send it over to the target.  We decided that we were going to use the tool to create an executable Java Archive (jar) file.

Our target was a system running Raspbian and accessible via Secure Shell.  We downloaded the Remote System Explorer to allow Eclipse to move files back and forth between our PC and the Raspberry Pi.  In order for ant to be able to communicate with the Pi, we had to add jsch-0.1.50.jar from JCraft and the tools.jar file from the Java Development Kit to the Global Entries for the Classpath in the Ant Runtime Properties.

The build.xml file had a section for each of several steps, including initialization, compilation, distribution, deployment, and cleaning up the system.  In order for the compiler (javac) to work, we had to specify libraries on the host system.  In order for the jar file to be created, we had to add entries for the libraries where they sit on the target.  Since we were creating multiple classes and adding them to the Jar, we also had to create an entry modifying the manifest of the archive.

In running a program that had to deal with the hardware of the target, we had to deal with the possibility that the system would not respond.  We did this by enclosing some of our classes and methods in try { } catch, and exception clauses.  We discovered, in one case, that having the SPI on the Raspberry Pi enabled meant that some of the Gpio pins were not available to our program.

During the very last session, we were just reaching the last great error that we would face.  If we wanted to use network tables, in ntcore.jar, we have to be able to get to the Java Native Interface modules for the native platform.  I wrote up the class that I posted to the Google Document share assuming that I would be able to get the development source to compile on the Pi. I failed.  You always have to be prepared for failure.  Java knows how to deliver it.  Looking at the code posted on the github, you will see it rewritten to meet the requirements of Network Tables 3.1.7 - the latest version that has been ported to Raspbian.

It is not impossible to do this task on our own and I will prepare a couple of lessons on how to do the cross compiling and/or building the necessary files on the Raspberry Pi itself.

If you take nothing else from this summer, take that lesson.  Save your work, you might have to go back to an earlier version.

What do we have left to do?  We have the code that is necessary to listen to the DAGU Hall Effect Encoders, we have the code that is necessary to control the Motor Drivers, but we don't have code to talk to either the Gyro or the Arduino - this will use the I2C protocol.  We have a disconnect between the control mechanisms and the motor driver.   There is a decision to be made there, which we would be taking up in the next session, if we ever hold it.  How do we picture this working?  We talked about the three conditions you find in a differential drive system.  If the motors are run at exactly the same speed, the vehicle goes forward or backward in a straight line.  If the left motor is slower than the right motor, the vehicle turns in an arc to the left, if the right motor is slower than the left motor, the vehicle turns in an arc to the right.  If one motor is going one direction and one is going in the other, the vehicle will pivot in place.  Our problem is to set up an algorithm that accomplishes this task.

Look at the spreadsheet I have loaded into the class artifact folder. It illustrates how one formula for preparing inputs for Arcade Drive would work on the robot. The file is “Arcade Drive.xlsx”