December 9, 2014

From:

An Nguyen - IEEE Region 5 Robotics Competition Team 1

Cullen College of Engineering, University of Houston

Houston, TX 77004

To:

Dr. John R. Glover

Advisor, IEEE Region 5 Robotics Competition Team 1

N308 Engineering Building 1

Houston, Texas 77204

Dear Dr. Glover:

This is the final technical report for the IEEE Region 5 Robotics Competition Team 1, for the fall semester of 2014. This report serves to update you on our progress for the semester, and to explain issues that we encountered this semester that delayed some of our goals. We will also review our budgets for the semester.

Our drive train and chassis has been completed and tested. However, we are not able to have fully autonomous movement in the maze yet. Our sensor tests have showed that the line sensors are not accurate at the height of over 0.5 [in]. We will need to go back and re-design our sensor placement and selection.

For image recognition, the project is progressing as planned, there was no delay after the second progress report. However, due to this major delay, we are not able to recognize images yet.

Also, for the general readers, we will explain the background and goal of the project. We will go over the specifications and constraints of the project, and show the overview diagram of the robot’s design.

Respectfully,

An Nguyen

Totally unimpressive. This report has the tone of someone who just doesn’t care. I made many comments in your progress report that you simply ignored, and consequently you repeated those same mistakes here. Most importantly, the goal analysis is still not done well, and is unchanged even though I made many comments on your progress report about it, which you simply ignored. There are many grammatical errors that should have been caught in a proofread; very little information on critical components like sensors; poorly discussed specifications; missing information (you didn’t read my guidelines, or at least you ignored much of them); poorly explained information in an appendix that was not discussed in the body of the report… I could go on.

**University of Houston**

**Electrical and Computer Engineering Department**

**IEEE Region 5 Robotics Competition Team 1**

**Progress Report II**

**Dec 9th 2014**

**Written By**: An Nguyen

**Team members**

An Nguyen

Christopher An

Hector Reyna

Daniel Romero

## Abstract

The team’s project is a maze solving robot that has the capability of recognizing characters of the maze wall and displaying the results in a text file. Some constraints for the design and built for the robot is that the team needs to adapt to an environmental constraint and it needs to meet the specifications to clear the maze at a minimum speed of 6[in/s]. Our progress so far is that we have chassis and drive train designed and built, we have also tested the drive train to a forward and reverse speed of 1[ft/s]. The image recognition progress has the largest delay due to software issues, the maze algorithm is progressing as plan and will be ported soon to a Tiva-C board. The line sensor has been tested, while the ultrasonic tests are still in progress. Updates on the cost of the project is the man labor that each team member has placed in the project and updated parts that were not included in the previous progress report.

# Background and Goal

The University of Houston has participated in the IEEE Region 5 Robotics Challenge since 2000. We have a tradition of doing well and placing, most times 1st and 2nd, in the Challenge up to 2010-2011. This team intends to revive the tradition by placing 1st in the 2015 challenge in Apr 17th, 2015 in New Orleans.

This year’s challenge involves solving a square maze in two phases. The first phase is the search phase, where the robot will have 3 minutes to search and map a 5’x5’ maze, 4 minutes for a 6’x6’maze and 5 minutes for a 7’x7’ maze. During this phase, there will also be characters on the walls of the maze. The robot can recognize these characters and store their position on a flash drive for bonus points or a chance to go straight to the finals. In the second phase, or the critical path phase. The robot has 1 minute to go from the start to the end of the maze. No extra actions are required in this phase. Figure 1 shows and example 5’x5’ maze, with different color walls: Blue for punctuations and numericals, pink for punctuations and alphabetics, and green for no characters:

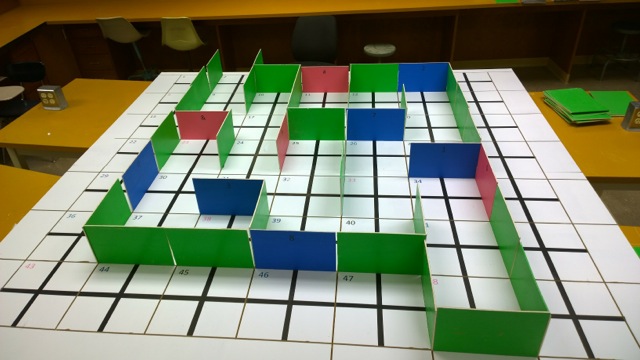
In the 3rd and final rounds, there will also be speed bumps placed along the path for an added challenge.

Figure -Example 5' x5' maze

# Problem, need and significance

Hazardous and frequently changing environments require robots that can traverse the environment and recognize hazardous features or features that require work along the way. This created a need for a robot that can map its environment, analyze for the quickest path to its goal and visually recognize features along the way.

This project will increase the prestige the University of Houston’s ECE department along its peers in Region 5. It will also increase the prestige of the IEEE Student Branch within the University of Houston.

# User Analysis

This project is intended for use by Engineering students, particularly Electrical, Computer and Mechanical Engineering students, who are interested in Robotics and Embedded system as a career path.

# Overview Diagram

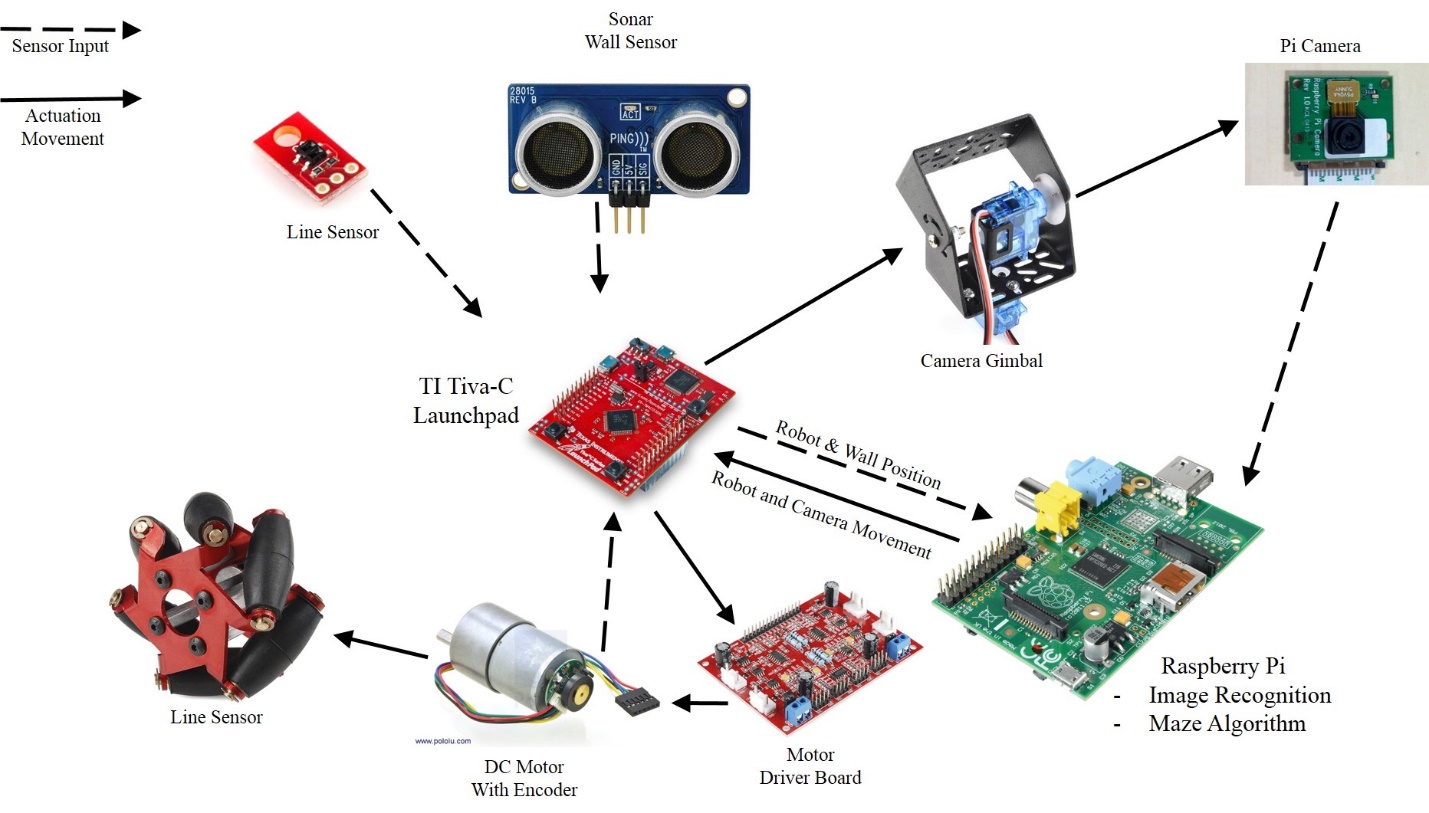
Figure 2 shows the team’s overall diagram in which the broken arrows represent a sensor or device input that goes into a microcontroller unit (MCU) board. Two MCUs that the team decided to use, according to Figure 2, are the Tiva-C and the Raspberry Pi. The team choose the Tiva-C as the unit to control sensors such as the wall sensor and the light sensor. The reason the team prefer the Tiva-C is due to familiarity and cost. Three team members are aware of how powerful the Tiva-C is (each pin may supply a voltage of 3.3[V]) and how to configure the board to do a certain task using the integrated development environment (IDE) Code Composer Studio. The cost of the Tiva-C may be found under the section “Budget”.

Figure - Overview Diagram

# Constraints

The Environmental constraint that was found in the project is the traversal of the maze. Since a maze is limited in size and the competition will be held indoors, the recommended length according to the rules of the competition is 7 [in] for width and length, this means the largest dimension for the robot is about 9.8 [in]. The height of the wall is 7.5 [in] therefore the team wants a maximum height of 7 [in] in order to take capture the best pictures for the Easter Egg. Also, the competition is time based and we are set at a limited time to be able to finish the course when necessary. Obstacles that may be found inside the maze are 0.5 [in] speed bumps, which constrains us to uese smaller wheels for traversing in the maze. Special constraints that were placed based upon the rules are the prohibition of wirelessly communicating with the robot and controlling its movement with wireless communication. According to Dr. Glover the IEEE Robot parts has a fixed budget of $1500.

# Specifications

For image recognition speed and accuracy, we are fixed with better resolution (a minimum of 240 x 104 resolution) than frames per second (FPS) since we are taking a picture of the colored wall and not live streaming the walls. The minimum FPS required for quicker storage of pictures is 10 FPS.

For navigation through the maze, we prefer acceleration over speed, since moving through the maze may require sudden stops when the robot reaches a turn or when the robot reaches a dead end. However, we do require a minimum speed when traversing the maze. To find the minimum speed we calculated the time we need to move around a maze cell in terms of inches per second by finding the total area of the maze and dividing the maximum time needed to search through the maze which is

|  |  |
| --- | --- |
| Minimum Speed = . | (eq. 1) |

Table 1 shows the maze size for each round, the maximum time to search and the minimum speed to traverse the maze.

Table : Table for Minimum Speed

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Round | Maze Size[ft x ft] | Cell # | Maximum Search Time[seconds] | Time per cell [seconds] |
| 1 | 5'x5' | 25 | 180 | 7.2 |
| 2 | 6'x6' | 36 | 240 | 6.7 |
| 3 | 7'x7' | 49 | 300 | 6.1 |

Therefore, the ideal minimum speed to traverse through the maze is 1 feet (or cell) per 5 second. Or 2.4 [inch/second]

# Target Objective

The target objective for the fall semester of 2014 is to have the three systems: chassis and drivetrain, image recognition and maze algorithm to work independently.

For the beginning of the spring semester of 2015, we expect to have some integration between the three systems. Namely chassis and drive train will be able to talk to the maze algorithm to simulate maze movement. Also the Tiva-C will be able to talk to the Rasberry Pi to request taking pictures and execute image recognition. The spring semester of 2015 will be used to debug the integration of the systems and devising strategies to win the competition.

# Goal analysis

Figure 3 is the team’s goal analysis for this semester. As discussed before in “Target Objective” for the fall semester of 2014, there are three systems being developed in parallel. The top system is the chassis and drivetrain, the middle is the image recognition and the bottom is maze solving and backtracking algorithm.

For the chassis and drivetrain, we have been able to assemble the final chassis design. We are in the process of testing the motors and writing drivers for the four motors to work together to move the robot.

For the image recognition system, we have been able to take a picture with the Rasberry Pi and OpenCV. However, we are having trouble compiling a C project with the OpenCV library. We have decided to recompile OpenCV on another Rasberry Pi board while exploring other algorithms on the original board.

For maze algorithm, we are able to implement the data structure for the maze algorithm using C programming. This step is important since we intend to run the algorithm on the Tiva-C, which will simplify the communications between the algorithm and the robot’s movement.

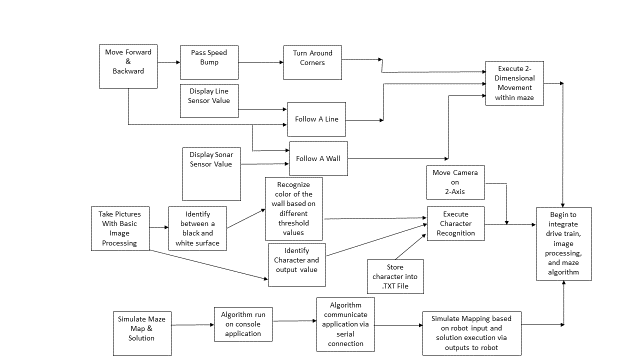


Figure – Goal Analysis

## Drive train and Chassis

The chassis for the robot has been designed, and the motors and chassis have been constructed. Figure 4 shows the chassis designwe drafted that the team will be using for the first layer of the robot. The first layer of the chassis consist of the motors and the mecanum wheels. The second layer will house the two MCUs and the camera gimbal.

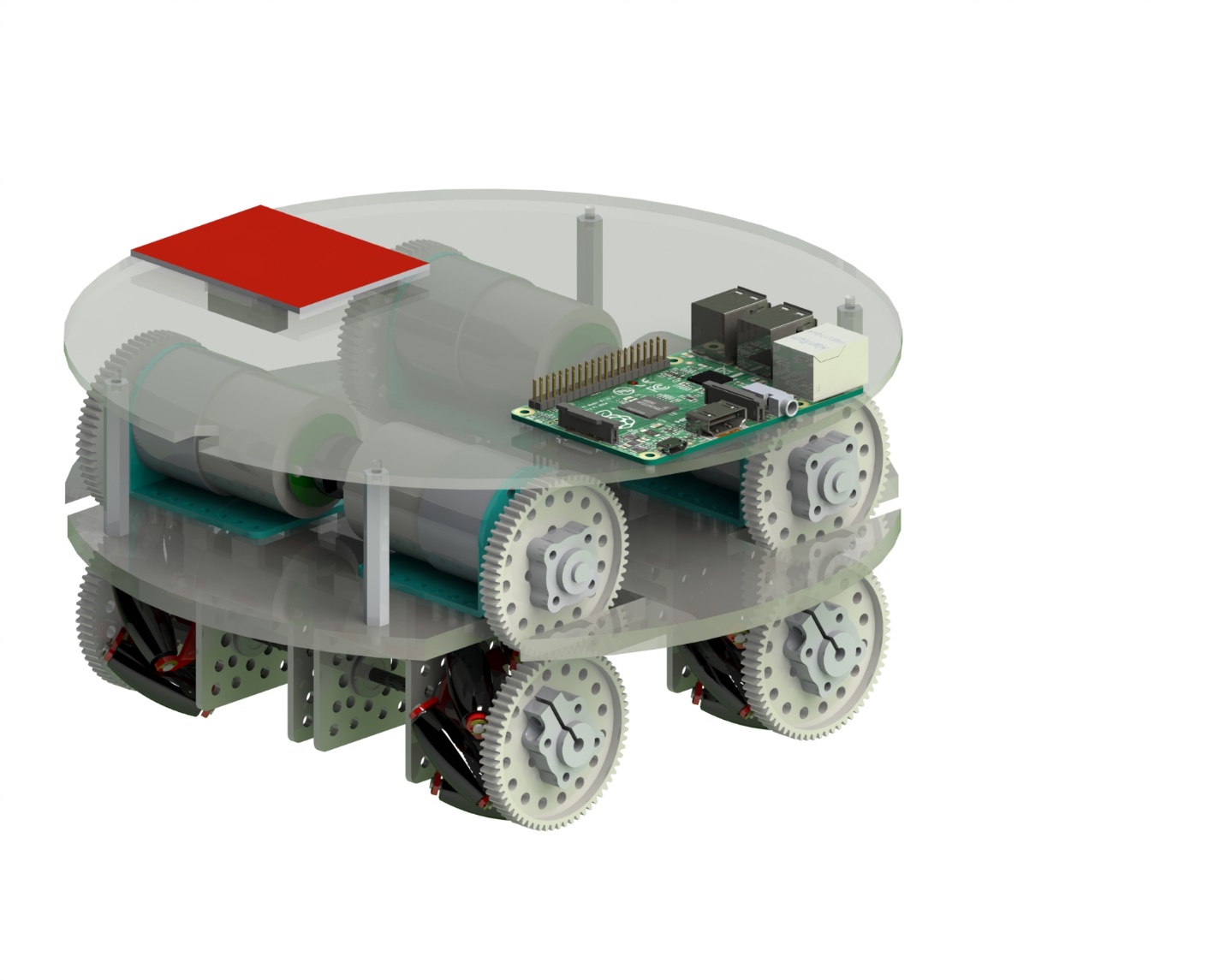


Figure : Chassis Robot Design

The chassis and drive train has been built (see figure 5). Currently I am in the progress of writing drivers for the motors to move the robot.

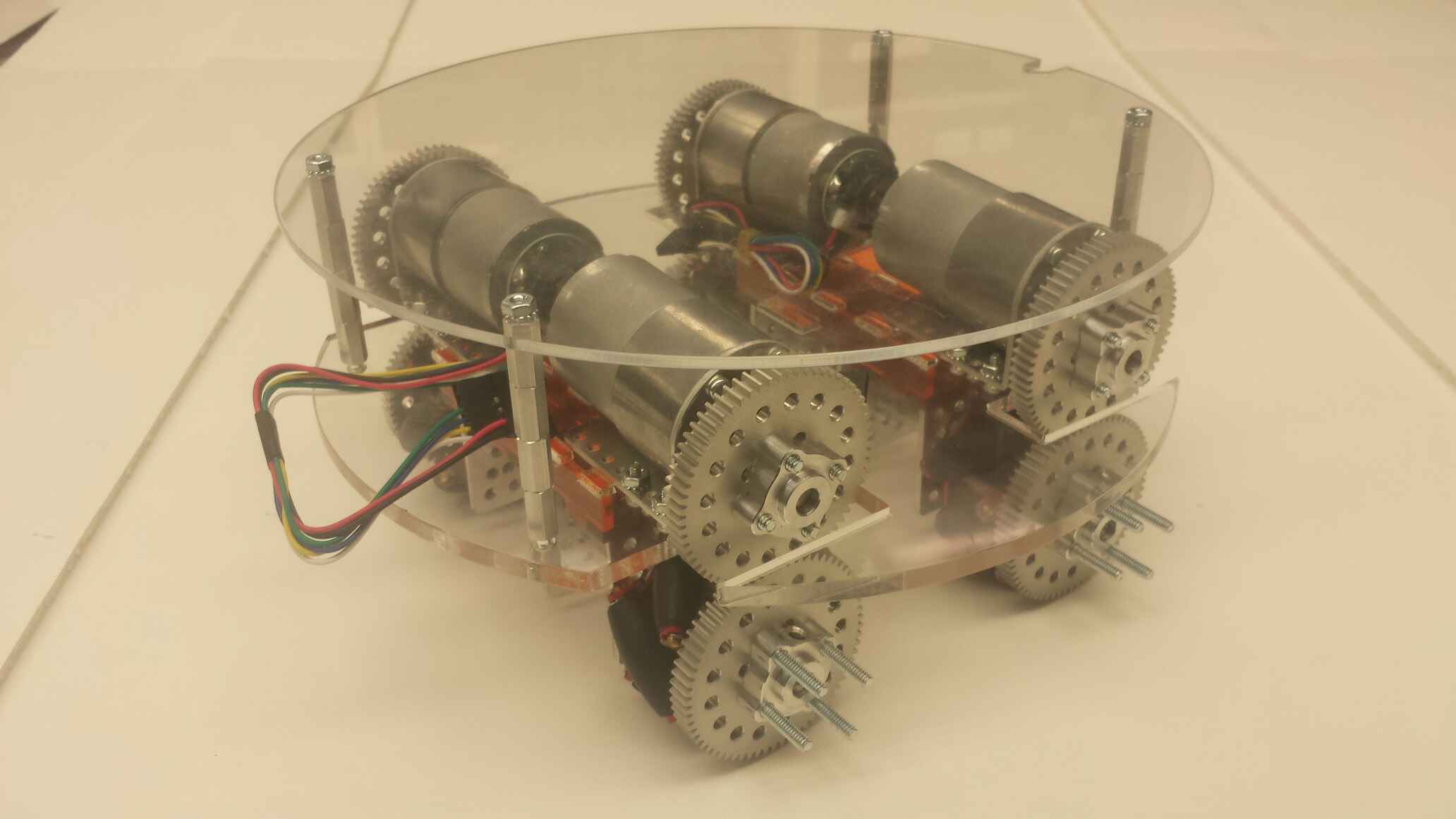


Figure : Built Chassis and Drive train

We have been able to test the individual motor with the intended motor driver and Tiva C board.

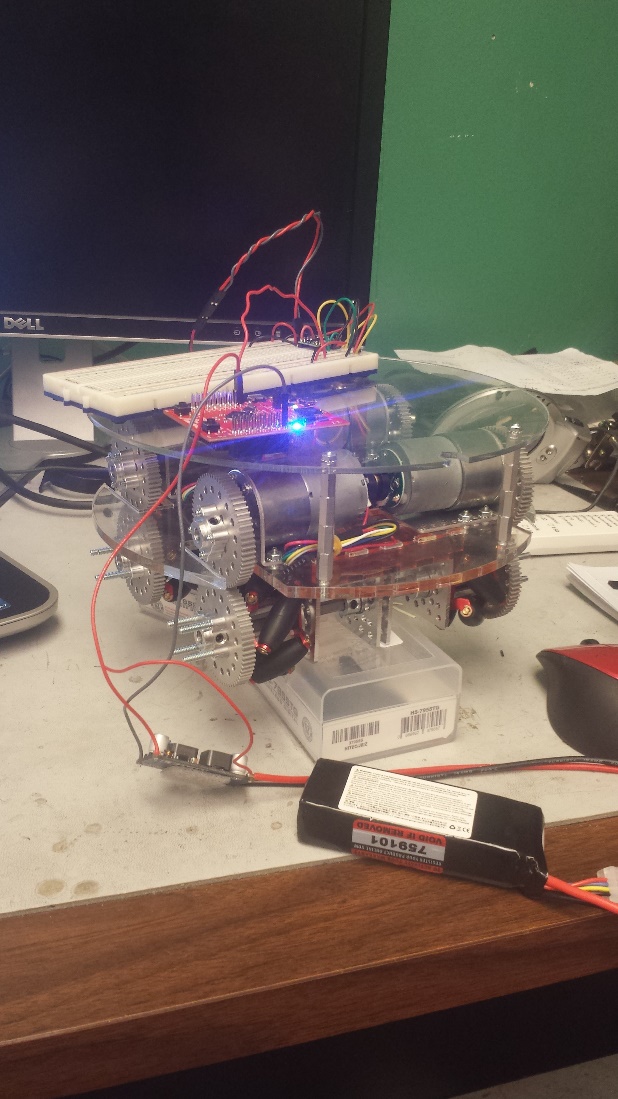
Figure 6 show the test set upwe had for each motor. We manually counted the revolutions in an interval of 10 seconds. In no load situations, the motor runs at about 150 ±2 [RPM]. As the wheels have 2 [in] diameter, we get:

Figure – Motor test set up

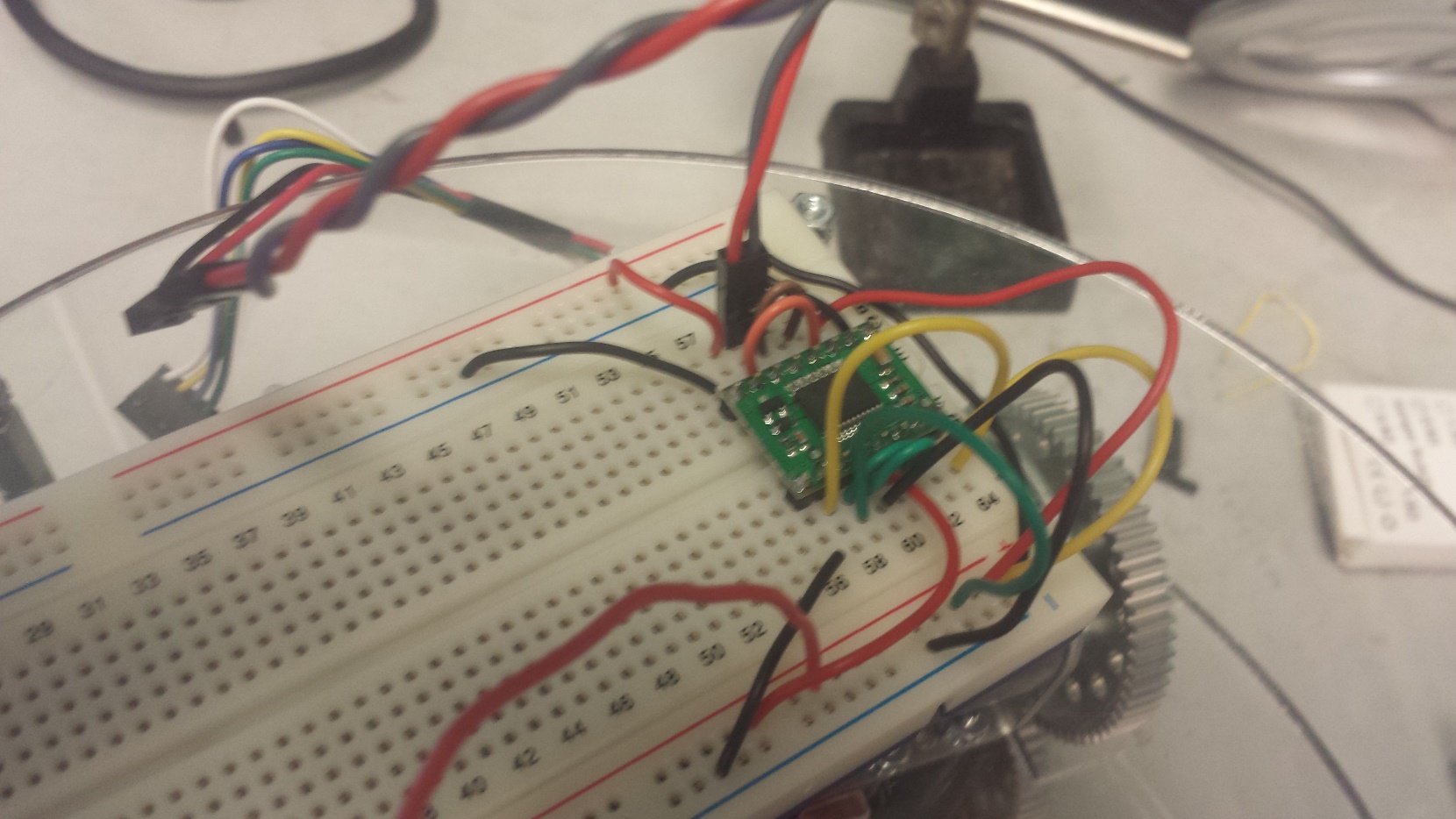


Figure – Motor Driver Board

Figure 7 shows the dual motor driver board. These driver boards have a peak current of 3[A] per driver. However the drivers can be paralleled to get a peak current of 6[A], well over the 5[A] peak of our motors.

## Sensors

We were able to test our line sensors to confirm their effectiveness at the height of 0.5 [in]. Due to the speed bumps, we will need to position the line sensors higher than 0.5 [in]. We confirmed that the line sensors gives no distinction between black and white at the given height.

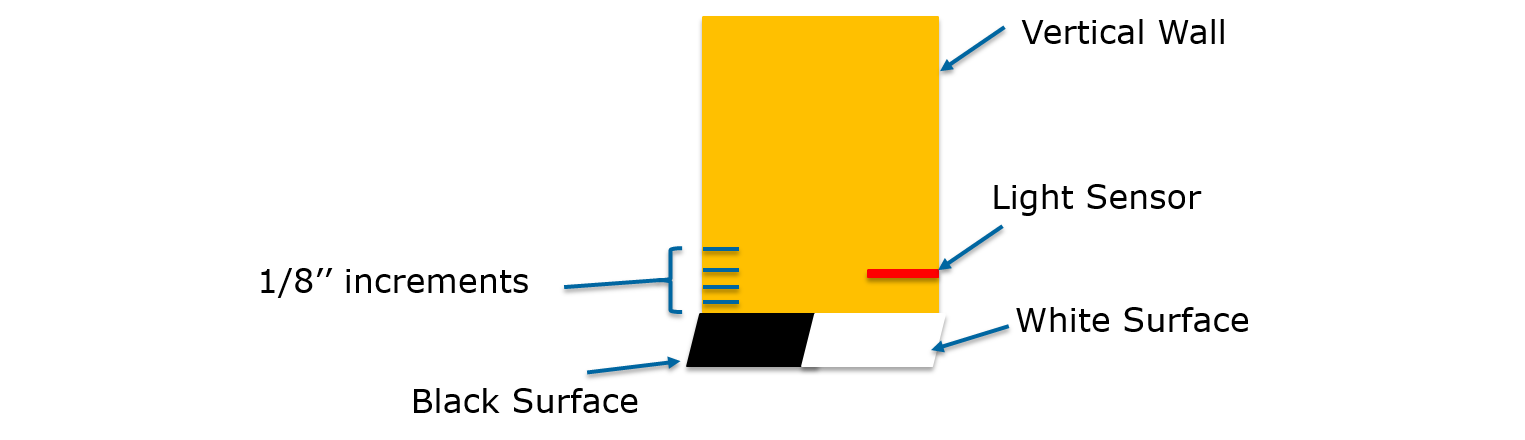
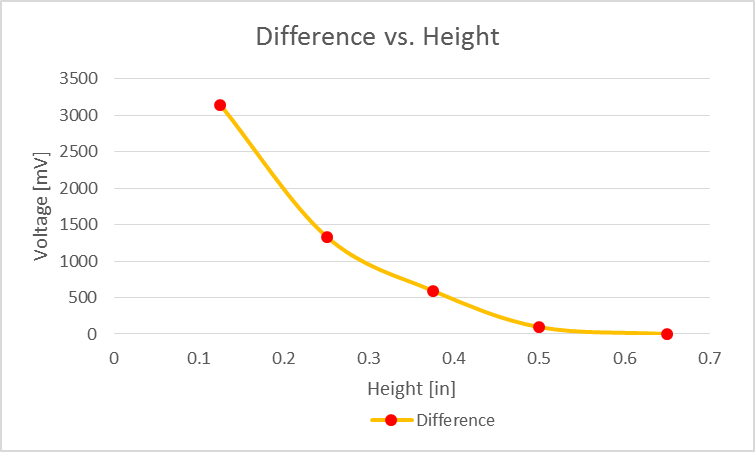
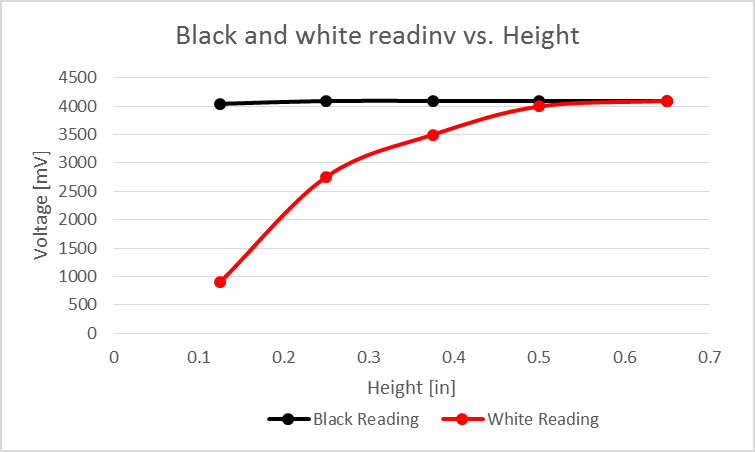
Figure 8 shows the test set up Chris used:

Figure – Line sensor Test

Figure 9 and 10 show the data we collected for the line sensor. The difference between white and blacked is practically 0 at 0.5 [in].

Figure

Figure

We are currently looking at design to lift the line sensor up when they are not in use. This will let us use the line sensor when calibrating the robot in a maze cell, and move them above 0.5 [in] while we are moving through the maze.

## Image recognition

At the moment, we have been able to install Rasbian, a port of the Debian Linux distribution, to the Rasberry Pi. We have also compiled and tested openCV as an application of the Rasberry Pi. However, he is having trouble compiling a program utilizing the openCV library at the moment.

Daniel was able to take a picture with resolution of 2592 x 1944 pixels (16:9) in 3.3 seconds. This is a problem and we will need to speed up this process.

Figure 11 show the Rasbian environment and Daniel taking a picture with the Raspi Cam on a terminal emulator:

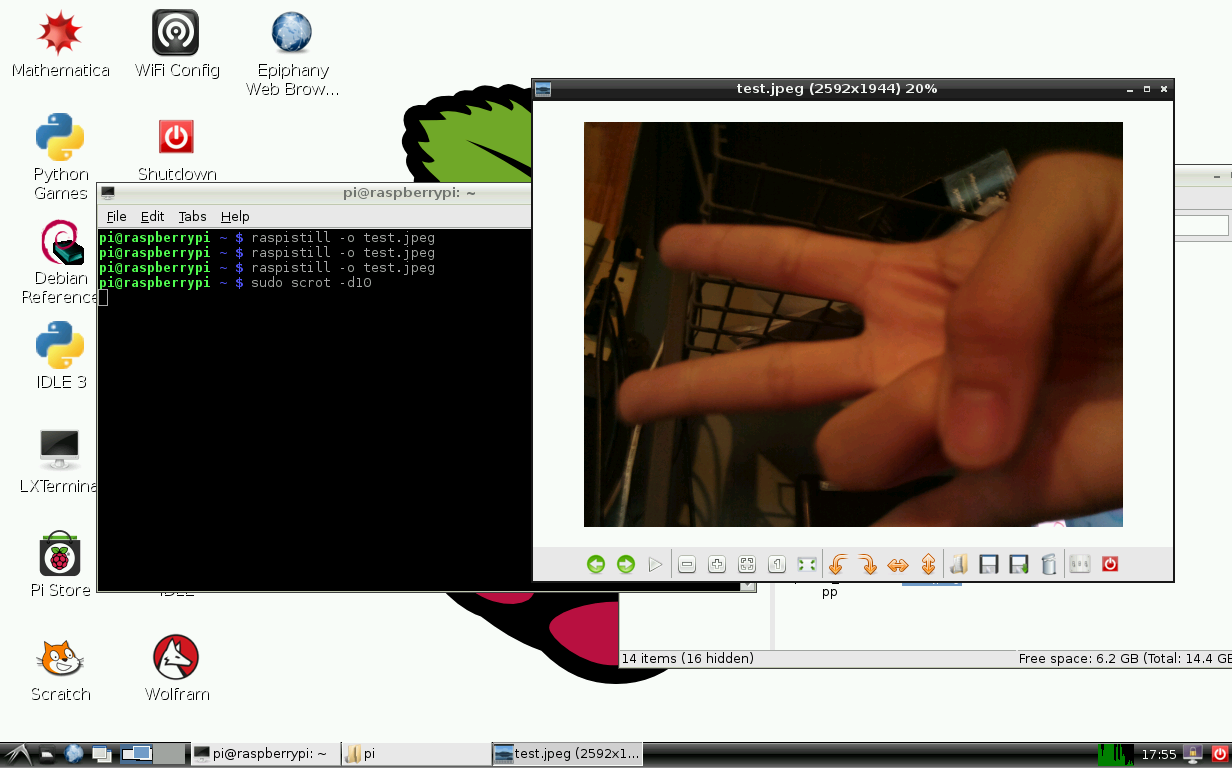


Figure – Taking picture with the Raspi Cam

Next Daniel would work on differentiating between and black and white background. Output “Black” and “White” based on what the program sees.

As Daniel is having trouble compiling a program utilizing the openCV library. He will be investigating other possible libraries for Visual Character Recognition and Image Recognition while waiting for openCV to recompile.

We have included Daniel’s documentation on how to install the Rasbian OS and openCV in Appendix I.

## Maze Algorithm

We were able to write a recursive backtracker algorithm that will recursively eliminate all dead-end, leaving the only critical path in place.

He was able to read the maze parameters, or the maze map, from a text file.

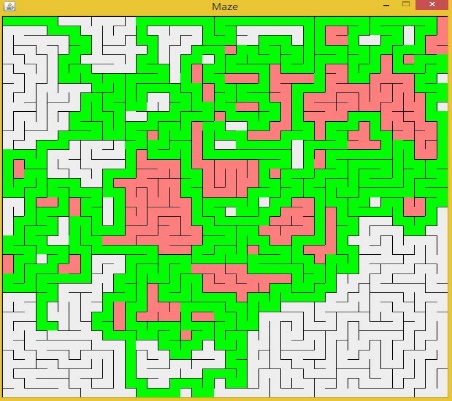
As we are implementing the maze solving algorithm on the Tiva-C, he need to write a linked list and stack data structure for the maze algorithm. He has finished this.

Figure – Recursive Backtracker GUI

Currently he is working on porting the recursive backtracker utilizing the data structure that he wrote.

Figure 12 show the recursive backtracker GUI written in Java. The red cells are cells that has been visited more than once. The white cells are cells that have not been visited yet.

# d Budget

|  |  |  |  |
| --- | --- | --- | --- |
| **System** | **Engineer** | **Hours** | **Cost\*** |
| Drivetrain | An Nguyen | 80 | $6,000.00 |
| Sensor | Christopher Ahn | 60 | $4,500.00 |
| Maze Algorithm | Hector Reyna | 80 | $6,000.00 |
| Image Recognition | Daniel Romero | 50 | $3,750.00 |
| Report and Documentation | All | 40 | $3,000.00 |
| **Total** | | **310** | **$23,250.00** |
| **Projected to date** | | **340** | **$25,500.00** |

Table – Personnel Budget

Table 2 show the Labor, or Personnel budget for the project. We are currently under budget. However as we are behind schedule. I would say that we will be over budget for this semester.

|  |  |  |  |
| --- | --- | --- | --- |
| **Robot Parts and Maze** | | | |
| **Parts** | Quantity | Unit Cost | Total |
| **Motors** | 4 | $40.00 | $160.00 |
| **Motor Driver** | 1 | $30.00 | $30.00 |
| **Wheels (4 Pack)** | 1 | $80.00 | $80.00 |
| **Acrylic Sheet** | 1 | $20.00 | $20.00 |
| **Tiva-C** | 2 | $15.00 | $30.00 |
| **Raspberry Pi Camera** | 1 | $26.95 | $26.95 |
| **Raspberry Pi** | 1 | $35.00 | $35.00 |
| **Line Sensors** | 6 | $20.00 | $120.00 |
| **Sonar Sensors** | 3 | $40.00 | $120.00 |
| **Maze** | 1 | $300.00 | $300.00 |
| **Total** |  |  | $921.95 |
| **Expected** |  |  | $1,150.00 |
| **Remaining** |  |  | $228.05 |

Table – Parts budget

Table 3 Show the Parts budget. The budget has not changed since the second progress report since all of our parts have come in and we are not expecting hardware changes.

# Conclusion

In conclusion, the project presented more difficulties than we anticipated, which created delays in our goal analysis. We are planning to work during the winter break to be up to date with the original schedule. By the beginning of the Spring 2015 semester, we plan to have our three system ready to be integrated. We expect to have the chassis and drive train integrated with the maze algorithm.

# Appendix I – Guide to installing Rabsian OS and openCV on the Raspberry Pi

## Why we choose the raspberry pi as our image processing board?

* Cheaper than other ARM boards (this one was $35)
* We need it only for image processing, it is not needed to use for anything else (unless we decide to) therefore have only a certain amount of GPIO pins is fine (since we are not going to use many anyway)
* There are many forums so we may reference (I am sure there is forums for using OpenCV and raspberry pi for image processing)
* Lower power therefore we won’t consume as much battery power
* Smaller than many other ARM Boards (this one is 3.3 x 2.2) giving us more options for placing it on our robot
* Specs for the Pi
  + 512 MB RAM
  + 700 MHz Broadcomm BCM2835
  + Storage: MicroSD
  + USB: 4
  + Power Draw/ Voltage: 600[mA] – 1.8 [A] @ 5[V]
  + GPIO: 40
* Raspi Cam Hardware:
  + Width: 2592
  + Height: 1944

## What steps were needed to be done

For installing os:

Download Noob into a microsd directly

You need internet connection (Ethernet cable) and also a good amount of time

When you power it up it wil prompt you to what OS you would like (I choose Raspbian since the most popular OS for raspberry pi)

Next we need to install opencv

Command Line:

* + $ sudo apt-get update
  + $ sudo apt-get upgrade
  + $ sudo apt-get install cmake
  + sudo apt-get -y install build-essential cmake cmake-curses-gui pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlib1g-dbg zlib1g zlib1g-dev pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools libeigen3-dev
  + $ git clone <https://github.com/Itseez/opencv.git>
  + $ cd /home/pi/opencv
  + $ mkdir release
  + $ cd release
  + $ ccmake ../
  + Configure what you want
  + Input ‘C’
  + Input ‘G’
  + $ make
  + $ sudo make install
  + $ sudo apt-get update
  + $ sudoe apt-get upgrade
  + Lsusb
  + Sudo apt-get install guvcview (to test openCV)
  + Sudo apt-get install autoconf (for auto configuration)
  + Gucview( test the camera, it is not going to work because it is not a USB camera)
  + Next: thing get pi camera compatible with openCV
  + $ wget <http://ww.linux-projects.org/listing/uv4l_repo/lrkey.asc> && sudo apt-key add./lrkeyasc
  + Cd /etc/apt/sources.list (does not work)
  + $ sudo nano /etc/apt/sources.list
  + $sudo apt-get update
  + (for the uv4l you need to put it in the sources.list, read the sections I have below to do this, I have a link)
  + $ sudo apt-get install uv4l uv4l-raspicam
  + $ sudo apt-get install uv4l-server
  + $ sudo apt-get install uv4l-uvc
  + $ sudo apt-get install uv4l-xscrenn
  + $ sudo apt-get install uv4l-mjpegstream
  + $ sudo apt-get update
  + $ sudo apt-get upgrade
  + $sudo apt-get install opencv-dev
  + (From the link I am currently working with)
  + $ cd /opt/vc
  + $cd /opt/vc/userland (use git clone for userland to place the userland directory here)
  + $ sed –I ‘s/if (DEFINE CMAKE\_TOOLCHAIN\_FILE)/if (NOT DEFINED CMAKE\_TOOLCHAIN\_FILE)/g’ makefiles/cmake/arm-linux.cmake
  + $ sudo mkdir build
  + Cd build
  + Sudo cmake – DCMAKE\_BUILD\_TYPE=Release ..
  + Sudo make
  + Cd /opt/vc/bin
  + ./raspistill –t 3000 (to test if it worked)
  + Cd /home
  + Sudo Mkdir camcv
  + Cd camcv
  + Sudo cp /opt/vc/userland/host\_applications/linux/apps/raspicam/\* .
  + Sudo Mv RaspiStill.c camcv.c
  + Sudo chmod 777 camcv.c
  + Sudo leafpad (used for super user to edit text files)
  + Cmake
  + Make (will not work as of now, need to learn to compile and link all file in the camcv directory) (Fixed issue: it was that the Link\_Libraries needed additional .cpp files in order to link with those .cpp files)

## Components Needed

What we need for the raspberry pi:

* The board itself
* Possible case (for now until the chassis for the robot is built)
* Micro SD card (Preferably 2 GB +)
* HDMI: To display results
* Rasberry Pi Camera (for Taking pictures of course)

## Getting Started

## Choosing the right OS

* For starters, this is my first time using a raspberry pi (virgin) so for a beginner I would recommend using Raspian (Debian Wheezy) which is a Linux OS.
* Therefore, this is the OS I am going to start with (until I am more familiar with the raspberry pi) on 11/4/2014 9:04 p.m.
* Need a micro sd card reader (will get one Wed 11/5/2014)
* Install Raspian

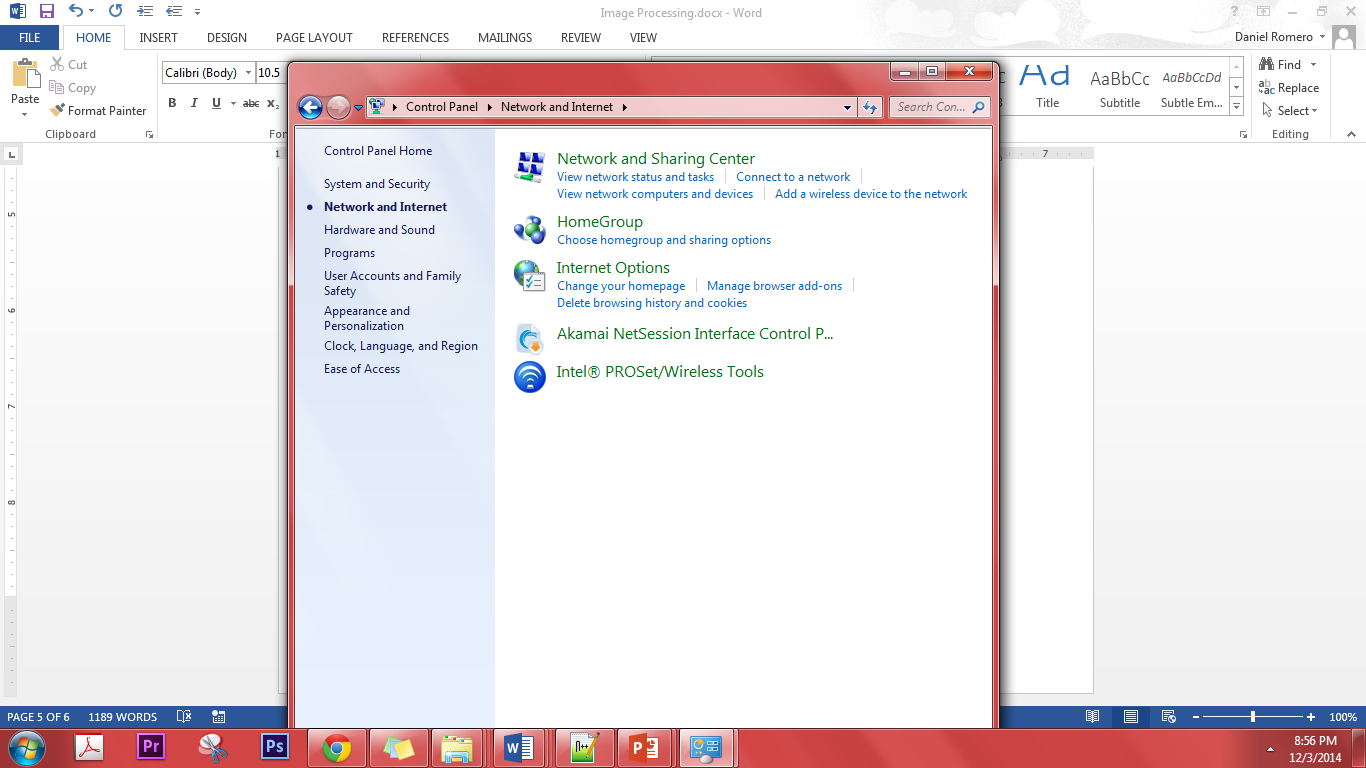
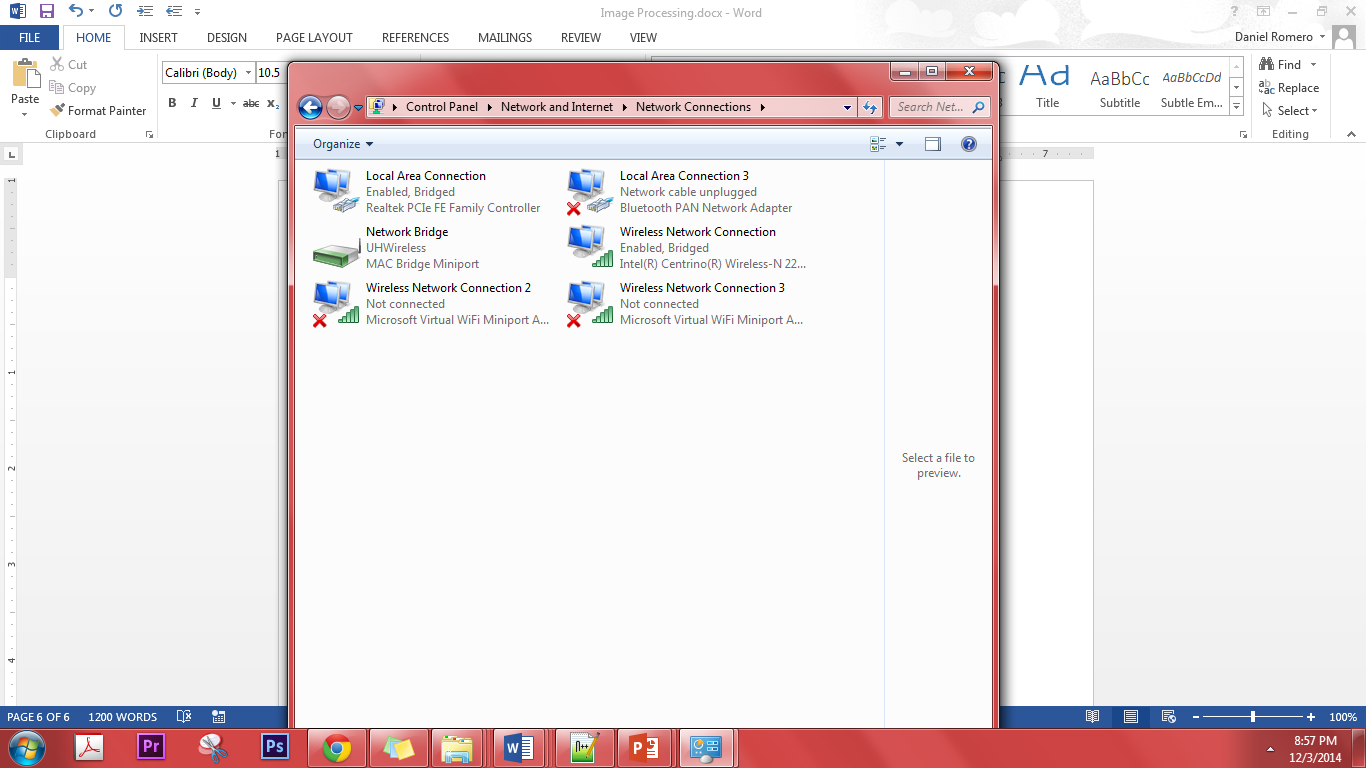
## Play with the OS (or get know the operating System and the raspberry pi)

* Seems like any OS
* Has Minecraft (WOW!)
* Several Things:
  + This is my first time using linux, therefore I needed to learn a little bit before getting started
  + So first of: Unlike windows where things “magically” are done for you “such as unzip”, the user needs to unzip things using a command line and to integrate a program into linux, they need to configure the files in order to “build” and use in raspbian (this takes a long time)
* Essential legends:
  + cd : opening a directory
  + sudo: for super user privelages
  + su: to be a super user
  + git: to put thing into the pi (from github)
  + sudo cmake .
    - makes its own in the current directory
  + make
    - makes the file or cpp

## Installing OpenCV

* What needs to be done
* Updates/Upgrades (must be connected to the internet):
  + $ sudo apt-get update
  + $ sudo apt-get upgrade
* Next you need to get cmake (in order to configure, generate, and build OpenCV)
  + sudo apt-get -y install build-essential cmake cmake-curses-gui pkg-config libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlib1g-dbg zlib1g zlib1g-dev pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools libeigen3-dev
* Next, you need to get open cv:
  + Either
    - git clone <https://github.com/Itseez/opencv.git>
    - or wget [http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.8/opencv-2.4.8.zip/download opencv-2.4.8.zip](http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.8/opencv-2.4.8.zip/download%20opencv-2.4.8.zip)
    - if chosen the second method then you must do the following:
    - First, look for your opencv and cd the location of the opencv
    - Next use: unzip opencv-2.4.8.zip
    - Follow by: cd opencv-2.4.8
    - mkdir release (creates that folder)
    - cd release
    - finally, ccmake ../
    - this opens up a command line gui looking thing that you may enable or disable certain things, therefore toggle things you want
    - Hit ‘C’ when done and ‘g’ to generate the Makefile
    - Finally: do make(10+ hours)
    - Then: sudo make install (? Min)
* This section is getting to take awhile, installation of OPENCV takes 10 hours, delayed build for our raspberry pi, build took 24 hours.
* <http://robertcastle.com/2014/02/installing-opencv-on-a-raspberry-pi/>
* <http://www.raspberrypi.org/forums/viewtopic.php?t=72796&p=524149>
* <http://thinkrpi.wordpress.com/2013/05/22/opencv-and-camera-board-csi/>

## Other Necessary installations

* How to bridge on Windows 7:
  + Go to Control Panel-> Network and Internet
  + 
  + Search for “Network Connection” -> Click on View Network Connections
  + 
  + Ctrl click the LAN connection for the pi and your wireless connection-> right click->connection bridge
  + Shut off pi and reconnect LAN cable (or connect) and turn pi on.
  + Works!
* The raspberry pi camera is a serial connection, therefore it will not work directly with OPENCV since OpenCV is supported for USB Cameras. However there is several ways to go about this.
  + For video: go to raspberrypi.stackexchange.com/questions/17068/using-opencv-with-raspicam-and-python
  + For right now, I think it works for both: link
    - Thinkrpi.wordpress.com/2013/05/22/opencv-and-camera-board-csi/
    - Progress: Currently at step 2. (the raspberry pi is building)
    - Been going with the thinkrpi website been working so far, however issues that I currently have:
      * Linking problems: I always get something about “unresolved…” and that is because I have not successfully compile the .cpp files so that is the issue
      * This is what I needed to do so far:
        + Install several different things