Department of Electrical and Computer Engineering

University of Victoria

ECE 499 – Design Project II

Final Report

Group No: 2

Project Title: A Basic Visual Guidance Sensor Subsystem for Racing Autonomous Robots

Web Presentation: http://www.ece.uvic.ca/~tyu/

Report Submitted on: April 4, 2013

To: Dr. Jim Collins

CC: Maryam Behrouzinekoo, Dr. K. Li and Dr. A. Zielinski

Team Members: Elaine Yan V00701459
Jeremiah Wilbur V00743587
Tiffany Yu V00704736
Ruben du Plessis V00709331

Table of Contents

Table of Figures

Project Summary

The objective of this project was to produce a robotic vehicle to navigate around a pre-set track numerous times with brightly coloured pylons. This project was from a pre-approved JC1 project on the course page. This project used the platform of the RC Car, Traxxas Slash Model. It drove autonomously on the track through a communication protocol, called I2C which is on the Raspberry Pi microcontroller. The car drove from pylon to pylon via image detection by the Raspberry Pi Camera, which was connected to the pi.

The car was powered through a Traxxas Power Cell of 8.4V, 3000mAh. Six AA batteries in series were also used to power the microcontroller, and the pi itself powers the PWM controller and the IMU Breakout board. The PWM controller board connected the servo and motor of the car to the pi via serial data line (SDL) and serial clock line (SCL) pins of the I2C. By writing the proper PWM commands to the servo and motor, the car could steer and drive properly. In order for the car to turn in the desired direction, the gyroscope from the IMU Breakout was used. This IMU Breakout board was also connected to the pi via SDL and SCL.

The software of the project was written in C++. To interface with the camera, the raspicam library was utilized. The camera ran on a low quality of 640 pixels by 480 pixels mode. Low quality of image detection was used to increase the speed of the image processing. The software code also used OpenCV libraries to process image frames. The frames were first converted to

HSV colour space, blurred with a Gaussian filter, and then turned into a binary image. The distance between the vehicle and the pylon were then calculated by using OpenCV's find contours function, which found the contours of the segments in the image. After the area of each contour was found, it was mapped to a distance using the areas of test pylons.

With the described implementation, the project was completed successfully. Although the car can now race with other autonomous robotic vehicles, it may have other practical features such as a moving home security system. This can also be great education material for anyone who wants to learn about mechatronics and perhaps build on top of the current project.

1.0 Introduction

The objective of this project was to produce a robotic vehicle that navigates around a pre-set track numerous times with brightly coloured pylons. This project was from a pre-approved JC1 project on the course page [1]. For this project, a chosen vehicle would drive autonomously, and using a camera, would process image frames of the pylons on the race track.

The platform used for this project was the RC Traxxas Slash Model, and the Raspberry Pi B microcontroller unit programmed the vehicle. The microcontroller contained an I2C communication protocol which connected with the pulse width modulation (PWM) controller and the IMU Breakout board for the system to work interchangeably. The PWM controller also connected the car's servo and motor to autonomously control the steering and driving of the car. The gyroscope from the IMU Breakout board was used to control the direction of turn of the car. A Raspberry Pi camera was used for image processing to detect the pylons and determine the distance from the car to the pylons on the track. The decisions of these components will be discussed in section 2.0 of this report.

The software used in this project includes C++ programming and OpenCV libraries. It is written to accommodate the I2C wiring, by utilizing the I2C addresses. OpenCV was used to process image frames. The distance between the vehicle and the pylon was calculated by using OpenCV's find contours function which finds the contours of the segments in the image. From this, the car could locate the pylon's location and the point to turn on the track. Details of the

software implementation will be discussed in section 3.0 of this report. Figure 1 shows the pylon's location and the ideal driving path for demonstration.

Although the project was to produce an autonomous vehicle for racing robots, it would be a good source of educational material for learning mechatronics. Furthermore, this project could potentially be implemented into a home security system for more practical uses.

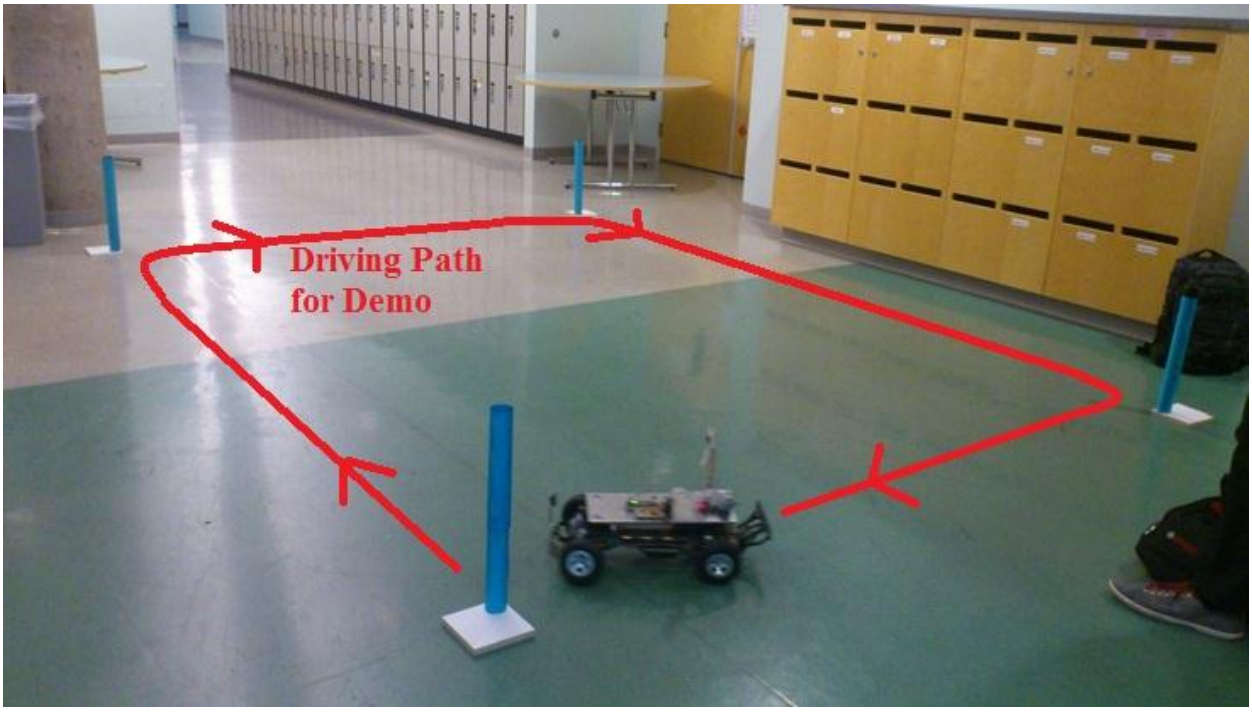

Figure 1: Demonstration Track

2.0 Hardware Design

2.1 Platform

In the starting phase of this project, a moving platform was chosen. A couple of weeks was spent on deciding between ROMO [2] and the RC cars. ROMO was a programmable, telepresence robotic toy for kids. It functioned fairly well with most recent models of iPhones and iPods. The original thought was to use an iPhone or iPod for image detection but ROMO already had an app on the Apple Store by Romotive for ROMO to do specific tasks as a toy for children. ROMO also had an SDK for developers. However, the team would need to hack into their software and break into ROMO's hardware to turn the platform into an autonomous racing robotic vehicle.

The team contacted Romotive for consultation about the project, and they could not confirm the feasibility of the project intention, using ROMO. Thus, the project resorted to the RC car that was lent to us from the course instructor, Dr. Kin Li. The benefit of using the RC Traxxas Slash Model (Figure 2) was that the team had control over all the hardware and software implementation. Size of the hardware components had not been an issue, and no software restriction existed in this project.

The RC car speed control model is a Traxxas XL5 Waterproof Electronic Speed Control (ESC). It has three modes [3]:

1. Sport Mode: 100% forward, 100% brake, and 100% reverse,
2. Race Mode: 100% forward, 100% brake, and no reverse, and
3. Training Mode: 50% forward, 100% brake, and 50% reverse.

For this project, training mode is used based on the desired PWM required to produce to the servo and motor of the car for demonstration.

Figure 2: RC Traxxas Slash Model [4]

2.2 Microcontroller

The microcontroller chosen for this project was a Raspberry Pi B [5]. Although other microcontrollers on the market (for example, Beaglebone Black) may be better to use, the reason of the Pi choice was due to a low team budget. On top of that, one of the team members owned it for a hobby interest and volunteered it for the project use.

The Raspberry Pi is a small single board computer that is widely used by hobbyists for small projects. The model B uses a 700MHz ARM1176JZF-S processor. For memory, the model B Pi has 512 megabytes of SDSRAM. The Pi also has a Broadcom VideoCore IV GPU. For persistent storage the Pi uses a standard SD card slot. For I/O, the raspberry Pi B has 2 USB 2.0 ports, a 10/100 wired Ethernet RJ45 port, several GPIO pins, a Serial Peripheral Interface Bus, an I2C

bus, and a UART port. There is also a CSI-2 camera port. The Model B Pi runs off of 5V DC at 3.5W (700mA).

2.3 Camera

The camera used for this project was the Raspberry Pi camera module. A lower end webcam on the market would have been a better choice for image processing but again, due to the budget and because one team member had purchased the pi camera for interest, it was used for the project.

This camera module uses an OmniVision OV5647 Color CMOS QSXGA sensor. This is a 5 megapixel (2592 x 1944) camera [6]. The angle of view is 54 x 41 degrees and the focal length is 3.6mm. This means that the width of the view at any distance is approximately equal to the distance. The mounting of this camera (and other components) on top of the car is shown in Figure 3.

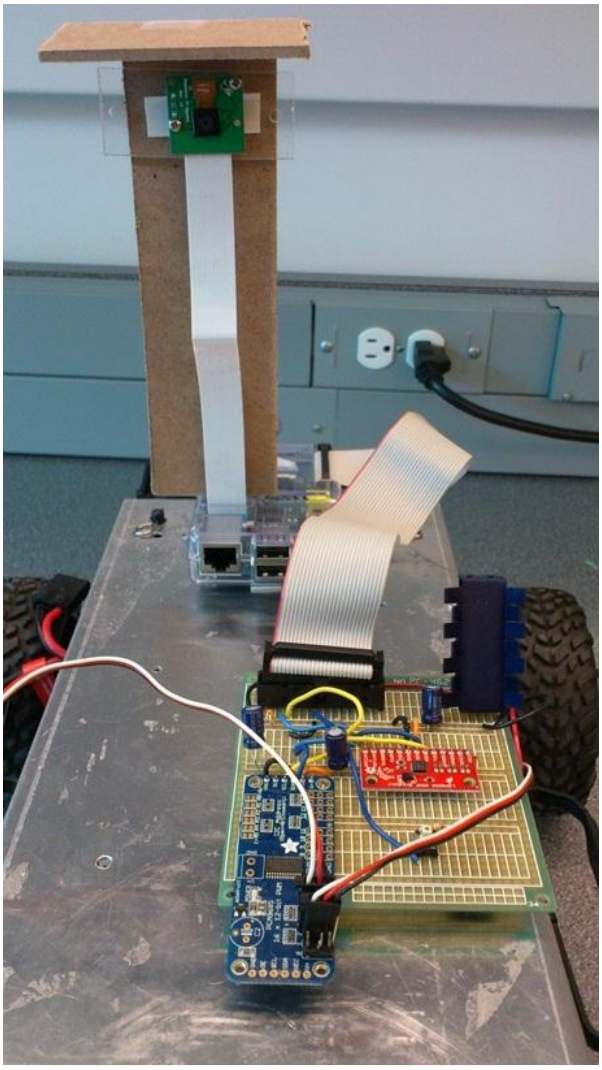

Figure 3: Component Layout on Top of the Car

2.4 PWM Controller

The PWM controller (model PCA9685) is a 16 channel 12 bit controller board which is controlled through I2C. The choice of this was because it was able to communicate through I2C, and it was at a fairly low cost with one of Sparkfun's supplier, Solarbotics [7].

The board is able to operate at frequencies between 40Hz and 1000Hz, and at duty cycles between 0% and 100%. The 12 bit range provides 4096 different values for frequency and duty

cycle which is extremely precise given the servos applications we used it for. Since the servos operate at 50Hz and at duty cycles between 7.2% and 6.0%, the PCA9685 board was within the required operating range. Since the board is controlled using the I2C bus it only required the I2C data and clock line to communicate, and the device addressing could be handled by C++ libraries. The inputs and outputs of the board can handle a voltage range of 2.3V to 5V [8].

2.5 Accelerometer & Gyroscope

Initially, an accelerometer (model ADXL345 [9]) was purchased at the same time with the PWM controller. However, due to time constraints and missing gyroscope component at the time, the ADXL345 was not used in the final design. Instead, the IMU Breakout board (model LSM9DS0 [10]) was purchased because it was at a low cost and has a nine degree of freedom three-in-one accelerometer, gyroscope, and magnetometer. Each of these sensors has three degrees of freedom and several operational ranges. The accelerometer can be set to ±2/±4/±6/±8/±16 g. The magnetometer can be set to ±2/±4/±8/±12 gauss. The gyroscope can be set to ±245/±500/±2000 dps. The LSM9DS0 has a 16 bit output, and can be controlled through SPI or I2C. The supply voltage range is 2.4V to 3.6V. However, this component arrived one week later than expected, and due to time constraints, only the gyroscope was utilized and implemented to control the car's turning.

2.6 System Schematic

Power was supplied to the Raspberry Pi using Six AA batteries. This produced a 9V output and was then converted to 5V using a 7805 voltage regulator. The 5 volt input was connected to pin 2 of the Raspberry Pi. The PCA9685 PWM and LSM9DS0 breakout were powered using the 3.3

volt output from pin 1 of the Raspberry Pi. The channel select of the LSM9DS0 breakout was also connected in order to put it into I2C mode. The I2C data and clock pins of the Raspberry Pi (3 and 5) were then connected to both the PCA9685 PWM controller and the LSM9DS0. Initially there were two 10kOhm resistors connected between the 3.3V line and the I2C data and clock. These were removed when the LSM9DS0 breakout was introduced because it has two built in 10kOhm resistors.

The Traxxas ESC was connected to header 1 of the PCA9685 PWM controller and the servo was connected to header 3. Initially, the team had the 5 volt input which drove the servo and motor coming from the output of the voltage regulator. However, since the servo would draw a large current whenever it changed position, it would cause the Raspberry Pi to turn off. The team then discovered that the 5 volts input for the PCA9685 PWM controller can be from the Traxxas ESC instead of the voltage regulator. This allowed the team to prevent current limitation of the 7805 regulator.

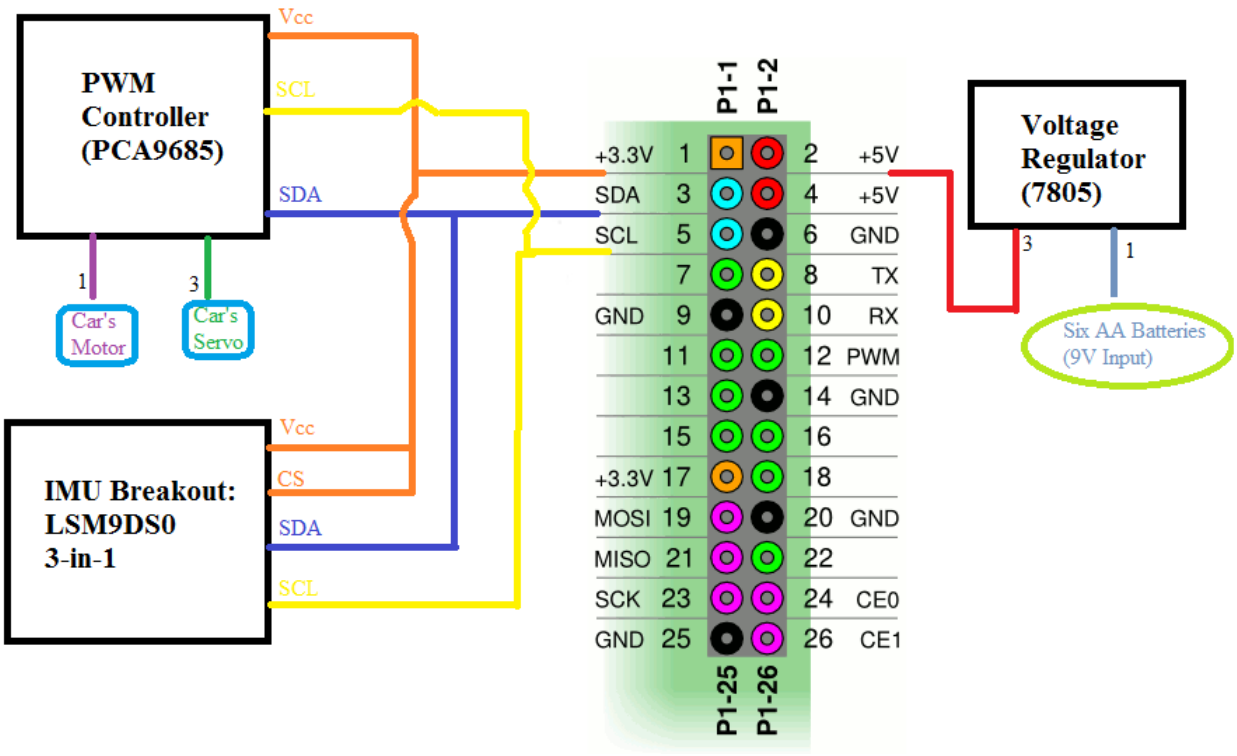

Figure 4: Schematic of Pin Connections [11]

3.0 Software Design

3.1 External Libraries

OpenCV [12] is an open source multiplatform computer vision library with many algorithms and structures that help with extracting meaning for images. The team decided to use OpenCV for a number of reasons. It has reputation of being fast which was important on this project. One of the group members had experience with it before, and there was not enough time to experiment individually.

Raspicam [13] is library for interfacing with the raspberry pi camera, using a library for this allowed us to focus more on the function of the vehicle then developing and testing the camera. This library was chosen because it works with OpenCV to take pictures as an OpenCV image format.

WiringPI [14] is a library for dealing with the Pi's IO pins, allowing more time to be spent working on the vehicle. Moreover, this library is versatile. I2C control, GIOP control and simplified timing functions are the features used.

3.2 Image Processing Thread

The image processing ran in a separate thread from the main program. Since it took slightly less than half a second to process a frame, it allowed the main program to update the steering or

check the gyro without having to wait for a full frame to be viewed. These thread first sets up the camera, and then continuously ran through a loop of the following steps:

1. Check to see if it should quit,
2. Capture a frame,
    - The larger the video format the more time it takes to process a frame. The precision for calculating the distance will increase. A 640x480 picture size was chosen because it is one of the formats the camera supports which reduced unnecessary cropping, as well as balancing the speed and precision.
3. Convert to the hue saturation value color space,
    - The HSV color space is useful because it is resistant to changes based on lighting as it should affect the value but not significantly affect the hue.
4. Gaussian Blur,
    - Smooths abrupt changes in the image, helps to keep segments continuous and reduce specular noise. A smallish 5x5 kernel was used to reduce processing load
5. Take all pixels in the HSV range (see Figure 5),
    - Currently the lower bound is (50,80,30) and the upper bound is (90,255,255)
6. Find all the contours in the resulting image,
    - Using the OpenCV function findcontours with a simple chain approximation
7. Find the area and center of mass of these contours,
    - Using the OpenCV moments function
8. Remove contours that are too large or too small to be a pylon,
9. Fit the area of pylons to an area vs distance curve, and
    - Linear interpolated between the nearest data points.

10. Write the distance, angle and time the picture was taken for the main thread to read.

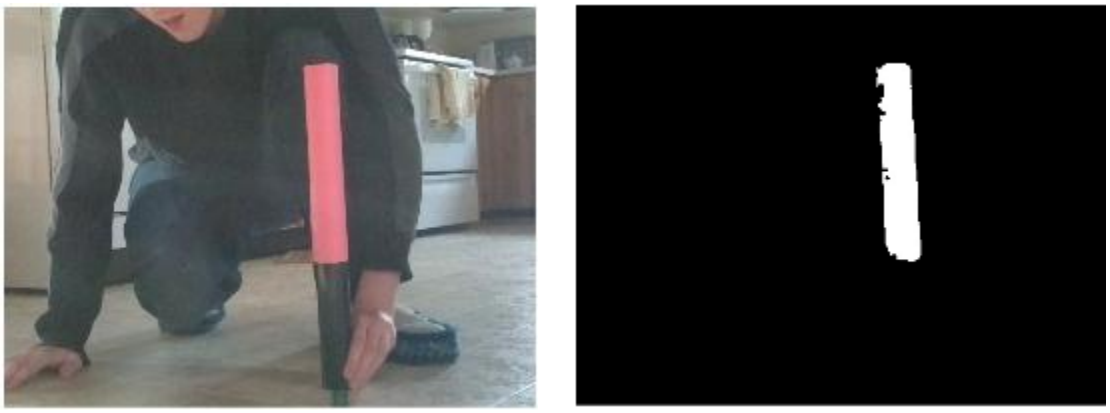

Figure 5: Testing Pylon Image (left) and its Binary Image (right)

3.3 Gyro Control

Only the z-axis of the gyro is used by this project. It is setup to poll at 100 Hz at ±245◦/s. This is done by sending the command 0xBC to register 0x20 of the gyro. These measurements were set to bypass the internal FIFO queue so that when it obtains a new measurement, it is always the most recent. To get the next measurement, the Pi reads registers 0x2C and 0x2D into a 16 bit integer After getting new data the controller conditions it to remove the offset that the imperfections in the gyro introduces, then converts it from the 16 bit integer into degrees per second which can be multiplied by the amount of time since the last measurement, and added to the current turn.

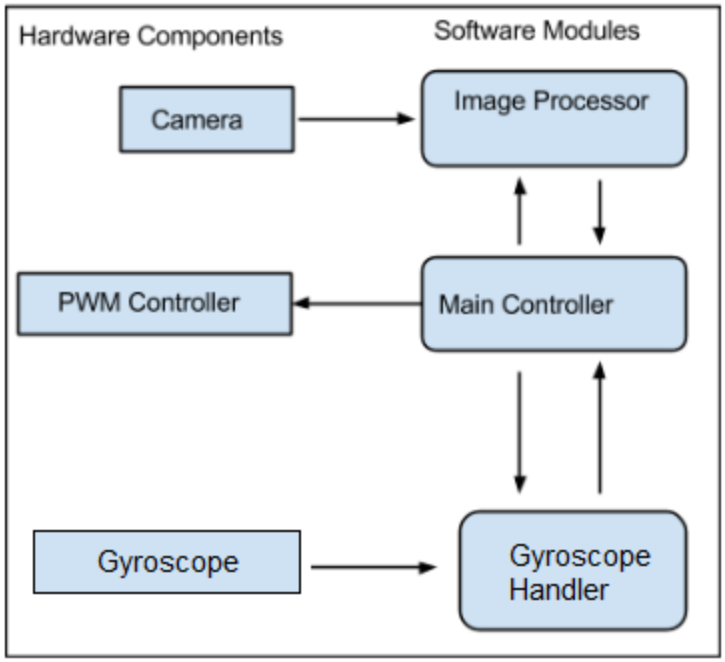

Figure 6: Dataflow Between Hardware and Software Components

3.4 Main Thread

The main thread of the program is in control of the steering, the motor and checking the gyro. This reduces the overhead of thread switching and locking/unlocking as data is transferred. The main program has two states. In the first state, the car drives towards a pylon that is in its field of view. It then turns towards it based on the angle and distance from the pylon. Once it sees the pylon is within 120 cm, the code “zeros” the gyro and then switches to the second state where the car will turn until the car sees the next pylon that should be in its field of view, and then switches back to state one. The gyro was used for the turns because of the low processing rate of the frames done by the camera. To do a sharp turn, the car would need to move slowly with the speed depending on the level of the car battery. The car would often stop during the turn if the battery was slightly drained.

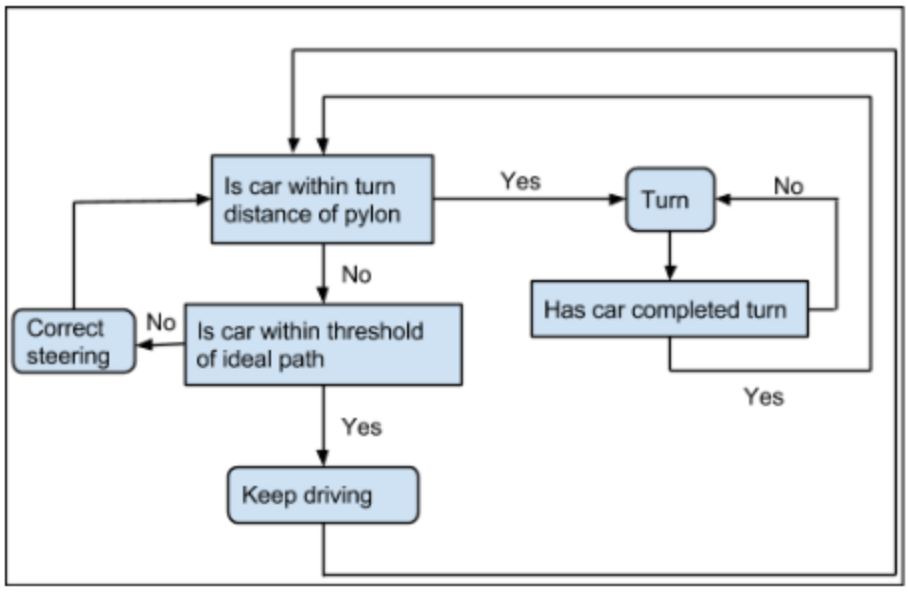

Figure 7: Decision Logic for Steering

3.5 Additional Programs

Additional programs were created to streamline the process of acquiring and testing images at known distances. The first one writes a series of images to a folder, and names them according to some parameters. This allows a picture to be taken every 10 cm from 10 to 500. The other program opens every image in a folder and runs an identical algorithm to find the area of each image. It then creates a file with all the area and their respective distances from the camera for the vehicle to use when fitting the area.

4.0 Conclusion

The project was successful and went smoothly throughout the term. The team worked very well together and was able to meet up as a group and also individually with minimum constant reminders from other team members. Throughout the term, progress was put into the project at a good pace, leaving enough time to troubleshoot and work out problems before demo day. The choice of equipment worked well with budget limitations, and the software and OpenCV libraries worked nicely with the project's intention. The project finished with all goals accomplished, and an autonomous robot was able to navigate its way around a track, using image processing to avoid brightly coloured pylons. The robot ran very well in all the test runs and navigated its way through the track with minimal errors on demo day. Some limitations (covered below in section 5.0) prevented the project from reaching its full potential, however the main goal was accomplished.

5.0 Limitations & Future Implementation

There were a number of limitations to this project, the first one being the budget. The team was fortunate enough to have the course instructor lend the Traxxas Slash RC car and cover the cost of the car battery and charger. One of the team members owned a Raspberry Pi camera and Pi MCU. The budget mainly covered additional parts, like the gyroscope, accelerometer, 26-pin ribbons, PWM controller, and AA batteries but the budget would not have been enough to cover higher quality and/or better functioning equipment.

The Beaglebone Black microcontroller would have yielded faster performance, versus the Raspberry Pi MCU, and an average webcam on the market would have yielded higher resolution images than the Pi camera. The Raspberry Pi camera is capable of 2592x1944p of images, but the project used 640x480 pixels as a trade-off for higher image processing speed. A higher resolution image on the Pi camera may provide more frames per second but will decrease image processing speed. With more frames per second, the system will have more image data to calculate the distance from the car to the pylon; yielding higher accuracy but the camera's processing speed will be too low to accommodate the accuracy of the image detection.

The robot currently utilizes only the gyroscope from the 3-in-1 IMU Breakout board, and does not utilize the accelerometer and magnetometer. Another way to determine the car's location from the pylon could be the implementation of the accelerometer and magnetometer. This would allow the robot to determine its location on the track to a higher degree of accuracy and allow for

minimal error correction. This is also a future implementation that could improve the robots driving ability.

Lighting limited the robot to turn only right in the demo track (the hallway in ELW behind Nibbles & Bytes), instead of having the option to turn left. In the track environment, the colour of the lighting, reflection on the floors, walls, and light from the ELW lobby decreased the pylon detection accuracy. The natural lighting in the ELW lobby, along with the blue carpet and sofa seating, gave off a blue temperature colour that interfered with the camera and the image processing. During testing, it was found that sometimes the robot would mistake certain objects or reflective spots from the lobby and think that it was a pylon. If the car was to directly face the lobby entrance (Figure 8) and turn counter clockwise around the track. However, turning in the clockwise direction allowed the robot to avoid directly facing the lobby, but face a wall instead (Figure 9).

Figure 8: Counter-Clockwise Direction, Robot Directly Facing the Blue Lighting

Figure 9: Clockwise Direction, Robot Avoids Reflection and Lighting of Lobby

Along with a higher MCU, and camera, the robot could also benefit if IR sensors were implemented. Using both image processing and IR sensors would allow the robot to become less reliant on proper lighting and rely on brightly coloured objects. The IR sensors could also be treated as a backup in case the image processing failed in those cases. This would be very beneficial as a home security system and would allow higher accuracy in avoiding objects, in any lighting. The robot can be navigating its way through a certain portion of the house, autonomously, creating movement inside the home to ward off intruders. Live image data streaming of the robot's camera will allow home owners to check up on their vacation.

References

[1] Department of Electrical and Computer Engineering, Faculty of Engineering, University of Victoria. (Mar. 24, 2014). ELEC/CENG/SENG 499 Design Project. [Online]. Available: http://www.ece.uvic.ca/~elec499/2014-summer/projects.shtml Retrieved Jan. 7, 2014.

[2] Romotive. (2014). Meet Romo Your Robotic Friend. [Online]. Available: http://www.romotive.com/ Retrieved Jan. 21, 2014.

[3] Traxxas. (2014). XL-5™ Waterproof FWD/REV ESC with Low Voltage Detection (LVD). [Online]. Available: http://traxxas.com/products/parts/escs/xl5waterprooflvd Retrieved Jan. 31, 2014.

[4] Burnside, Joe. (2013). Traxxas Slash 4x4 Review. [Online]. Available: http://www.8-lug.com/tech/1301_8l_traxxas_slash_4x4_review/ Retrieved Mar. 8, 2014.

[5] eLinux. (Mar. 6, 2014). RPi Hardware. [Online]. Available: http://elinux.org/RPi_Hardware Retrieved Apr. 5, 2014.

[6] eLinux. (Apr. 3, 2014). Rpi Camera Module. [Online]. Available: http://elinux.org/Rpi_Camera_Module Retrieved Apr. 5, 2014.

[7] Solarbotics Ltd. (2014). 16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685. [Online]. Available: https://solarbotics.com/product/19000/ Retrieved Feb. 7, 2014.

[8] NXP B.V. (Sep. 2, 2010). PCA9685 16-channel, 12-bit PWM Fm+ I2C-bus LED controller Production data sheet. [Online]. Available: http://www.adafruit.com/datasheets/PCA9685.pdf Retrieved Feb. 21, 2014.

[9] Solarbotics Ltd. (2014). SparkFun Triple Axis Accelerometer - ADXL345. [Online]. Available: https://solarbotics.com/product/50656/ Retrieved Feb. 7, 2014.

[10] ST Life Augmented. (2013). LSM9DS0 iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer Datasheet - production data. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00087365.pdf Retrieved Mar. 24, 2014.

[11] Advamation mechatronics. Raspberry Pi. [Online]. Available: http://www.advamation.com/knowhow/raspberrypi/ Retrieved March 7, 2014.

[12] OpenCV. (2014). Open Source Computer Vision. [Online]. Available: http://opencv.org/ Retrieved Apr. 4, 2014.

[13] AVA Aplicaciones de la Visio nArtificial. (2014). RaspiCam: C++ API for using Raspberry camera with/without OpenCv. [Online]. Available: http://www.uco.es/investiga/grupos/ava/node/40 Retrieved Apr. 4, 2014.

[14] Gordon Henderson. (2012-2013). Gordons Projects WiringPi. [Online]. Available: https://projects.drogon.net/raspberry-pi/wiringpi/ Retrieved Apr. 4, 2014