Facial-Recognition, Obstacle-Detection, Omni-Wheel Robot Platform

Final Report

(FRODOW Robot Platform)

Team Members: James Kolar, Caden DeRoche, Noble Koshy, Bryan Nestingen, Yao Yao

Advisors: Professor Chris Kim, Luke Everson

College of Science & Engineering

Department of Electrical and Computer Engineering

University of Minnesota-Twin Cities

EE4951W

Professor Douglas Ernie

16 December 2017

Table of Contents

Executive Summary

Facial-recognition software is the on the forefront of technology. In order to test out these developing software packages, it was our goal to enhance a pre-existing omni-directional robot platform to include obstacle-avoidance and pathing strategies. Our solution implements facial-recognition libraries, recognition-to-pathing algorithms, and obstacle-avoidance hardware/software onto the FRODOW robot platform. The final robot design tracks a designated person through a three-meter-radius static-environment containing blocking obstacles.

The robot is a self-contained unit equipped with cutting-edge software and hardware. It is operated by a 1.2 GHz Raspberry Pi processor that can execute open-source software packages. The Pi serially communicates movement commands to the motors' onboard Arduino processor. The robot is powered by onboard power supplies and is accessed through a 7" LCD touchscreen.

The previous robot platform could detect a face using a Pi Noir camera and the OpenCV library and would rotate until it detected a face. Once a face was found, the robot would center the face within its camera's frame of reference and move toward that person. The previous robot platform had problems running into obstacles while tracking the face. The robot also lacked the capability of recognizing and tracking specific faces.

The new design addresses these shortcomings and demonstrates these solutions by tracking a specified person through an 'obstacle course.' In the initial robot state, the camera takes a picture of an audience member. The neural network library learns the person's face while the person moves to a new place within the course. The robot scans the environment until the person is found and sets a path to get within 1 meter of the target. The robot enters its pathing and obstacle avoidance algorithms until it reaches its destination.

Our design implements dlib's state-of-the-art facial recognition function for tracking individuals. Since the facial recognition software is computationally intensive, the robot cannot maintain camera direction toward the tracked person. Therefore, the robot has been developed to utilize an 'A*' pathing algorithm (i.e., an efficient graph traversal function) using forward movements and 90 degree rotations. The robot utilizes the coordinates from the facial recognition function and maps a straight-line course to the target. If the robot encounters an obstacle, it will map the obstacle into its grid and recalculate the path to the target.

Our solution protects against forward-movement collisions by adding three ultrasonic distance sensors placed evenly across the front-face of the robot's custom-built chassis. A detected object and its distance are successfully conveyed to the robot's pathing algorithm. An inertial measurement unit (i.e., IMU, or accelerometer and gyroscope) is used to calculate all rotation movements so that the pathing algorithm can overcome motor drive-train inconsistencies.

The team has demonstrated a successful design through testing the robot's capabilities. Through 30 trials, the A* pathing algorithm has achieved an accuracy of ±1 cm per movement over 35 cm node distances. The ultrasonic distance sensors have demonstrated an average range of 1m, which is an acceptable range for our demonstration. The movement software and the IMU have been calibrated to ensure the robot's rotations have an accuracy of ±1◦ rotational error.

Our design is the first step toward creating a fully-autonomous obstacle-avoiding robot. The addition of the IMU and ultrasonic sensors have increased the pathing and avoidance capabilities of the robot. By becoming fully autonomous, this robot will serve as a platform to test future facial recognition software. Robot's like FRODOW have the potential to be used in healthcare, security, and military applications.

Future FRODOW project recommendations include:

- More robust sensor package for 360° obstacle avoidance
- Take advantage of omni-directional movements vs using 90° rotations using IMU
- Update PID algorithms (tune or replace wheel 3's motor)
- Real time facial-recognition and identification. Use a more powerful alternative to the Raspberry Pi such as the NVIDIA Jetson TK1. Also modify facial-recognition software for multiprocessing in order to further improve performance.

Problem Definition

Introduction and Background

The senior design team has been given a working robot prototype, depicted in figure 1, and is tasked to improve its capabilities. The robot can move in omni-direction (i.e. any direction) and includes a Raspberry Pi processor that can execute open source graphics software packages such as OpenCV to allow for analyzing photo data. The Pi also has many I/O pins that can be accessed to implement other peripherals. The Raspberry Pi serially communicates movement commands to the robot's onboard Arduino board. Listed below are the components of the existing robot system:

- Arduino based 4WD Omni-Directional Robot
- Raspberry Pi V.3 Model B
- Raspberry Pi Noir Camera
- Raspberry Pi 7" LCD Touchscreen Display
- 5V, 2.4A Power Bank
- Custom-built electronics chassis

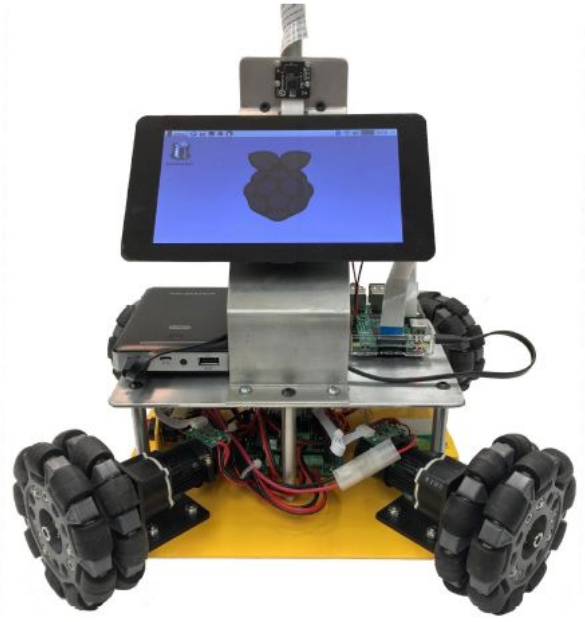


Figure 1: Original Robot

The previous senior design project was tasked with improving the robot design being Arduino based and communicated to MATLAB via Bluetooth. However, the group had difficulty getting the robot to perform as expected and decided to start from scratch following a list of achievable requirements. The team desired a robot that is completely autonomous with internal image processing capabilities that were more complex than the provided light intensity line scanner. Therefore, the team opted to switch to a Raspberry Pi for processing. Choosing the Raspberry Pi over the Arduino for processing allowed the team to access many impressive open source graphics software packages such as OpenCV to allow for analyzing photo data and put a 1 GHz computer onboard the robot, lifting the restraint of external image processing. The Pi, like the Arduino, also has many I/O pins that can be utilized to implement many other peripherals. With this, the previous design team decided to use the Raspberry Pi and the Pi Noir camera for image processing.

The previous design team successfully built an autonomous robot with internal image processing capable of object detecting and tracking utilizing facial detection software. The robot was created such that it can be easily used and modified based on the client's requirements. This robot platform serves as a solid foundation for future experimentation. Therefore, the previous senior design team recommended the following improvements:

- Create a more robust movement library to improve presentation of the robot.
- Decrease the image processing time in order to detect faces quicker and have quicker robot reaction times.
- Update the PID control for more fluid movements. Install more peripherals to advance the robot's capabilities.
- Implement a neural network based program on the robot.

Our Project Description and Goals

Our project aims to create a fully-autonomous robot, capable of obstacle-avoidance and tracking individuals, to serve as a testing platform for facial-recognition software. The robot demonstrates these added capabilities by being able to locate and plot a safe path within an obstacle course to a targeted individual.

Objectives

To meet our goal, our team has added environmental sensors for obstacle avoidance, implemented a pathing algorithm, and created a demonstration algorithm to test facial-recognition software. The team defined the listed specifications and client needs to pursue during our iterative design approach:

Movement Patterns

- More robust and expanded movement library
- Optimize PID control

Obstacle Avoidance (Indoors)

- Additional peripherals and sensors
- Re-evaluate camera structure

Image Processing

- Face-tracking recognition and object tracking

Improve Testing Interface

- Remote controlling for debugging
- Documentation/Schematic of robot
- Storing live-feed data, display on LCD

Client Needs

#	Need	Importance
1	Optimize PID control	3
2	Expanded movement library	1
3	Additional peripherals for environmental sensing	2
4	More efficient camera/movement structure	1
5	Face-tracking neural network integration	2
6	Possible off-board image processing	4
7	Remote controlling for debugging	4
8	Documentation/Schematic of robot	3

9	Storing live-feed data, display on LCD	5
10	Handles for transporting robot	1

Design Specifications

#	Need #'s	Metric	Units	Ideal Value	Acceptable Range	Importance
1	1,2,3,4	Minimum distance from obstacles	m	0.1	>0.05	1
2	1,2,4	Minimum displacement from rest	m	0.05	<0.1	1
3	4,5,6	Facial recognition in group of N or more	N	5	2	1
4	1,2,6	Time to re-acquire facial tracking	ms	500	<5000	1
5	9	Refresh rate on displaying captured image to LCD	fps	24	>5	2
6	5	Time to run facial Recognition	s	5	<60	1

Further Design Specifications/Constraints

- Budget of $300
- End goal presentation is specified by the team
- Must be able to run in real time on a Raspberry Pi V3B
- Should be able to operate without collision in a static environment

Design Description and Component Evaluations

This project aims to create a fully-autonomous robot to serve as a testing platform for facial-recognition software. Figure 2 introduces the FRODOW robot platform, an acronym for our project (Facial-Recognition, Obstacle-Detection, Omni-Wheel Robot Platform). Our solution brings added-value to the robot platform by adding obstacle-avoidance, absolute-orientation, and facial-recognition capabilities. Distance and measurement data are extracted from the sensors to plot obstacle and terminal locations. The robot demonstrates its facial-recognition capabilities by being able to locate and plot a safe course to a targeted individual. The following subsections describe the design and evaluation of the components we added to the robot platform.

Figure 2: Meet FRODOW

Demonstration Algorithm

The team designed a demonstration algorithm to allow FRODOW to test facial-recognition software by programming the robot to locate and plot a safe course to a targeted individual. Our demonstration algorithm is highlighted by the flow chart depicted in Figure 3. The robot begins

the program by searching for faces using a Haar-Cascade method. The robot will rotate in increments of 60 degrees until it detects a face. Once a face is found, the facial recognition software is run to identify the proper target and returns the distance and angle to the individual's face. The angle is used to center the robot's camera on the target, and the distance is sent to the pathing algorithm. The robot moves toward the target following an optimal path using only forward movements and 90 degree rotations. If the robot encounters an obstacle, the pathing algorithm will re-route the robot around the obstacle to the targeted individual. Once the robot has reached its destination, it will check again for the person's face and if the person is within the tolerance distance, the robot will end its program.

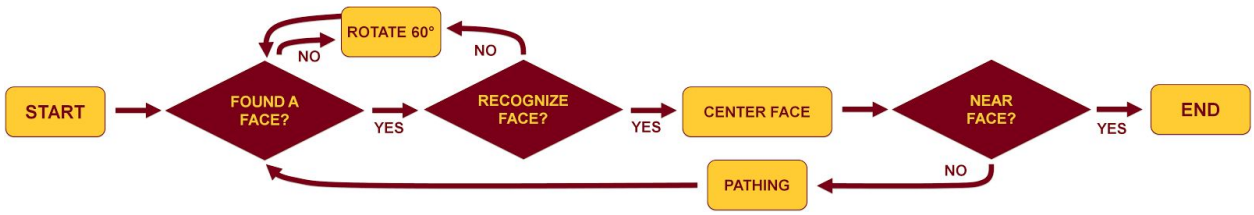


Figure 3: Demonstration Algorithm

Deciding upon this demonstration proved vital for the direction of our design and through it we were able to delegate tasks to accomplish our objectives. This algorithm is adaptable to enhancements in movement capabilities, environmental sensors, and facial recognition software, which was our ultimate goal for improving FRODOW's capabilities. This algorithm will be the basis for FRODOW's operations moving forward. The design and evaluations of the flowchart components are discussed in the following sections.

Ultrasonic Distance Sensors

The robot needed environmental sensors to detect objects. Our initial idea was to use the camera to detect objects and people but decided to use the camera only as a facial-recognition tool. We brainstormed a few options to add onto the platform such as ultrasonic distance sensors and lidar. The ultrasonic sensor option is low cost, ranging from $2.00 unit cost up to $100.00 unit cost whereas the lidar option starts at a $300 unit cost. Ultrasonic distance sensors work by

'pinging' the environment with an ultra-sound wave (approximately 40 kHz). When the response is detected, we can determine how far the nearest object by measuring the time latency and dividing by the speed of sound (see figure 4). Since we had a Parallax PING Ultrasonic Distance sensor in our inventory, we evaluated its functionality and it demonstrated an accurate detection range within 1 meter. Therefore, we decided to stick with the low cost option because our demo would be contained within a 3 meter-radius environment.

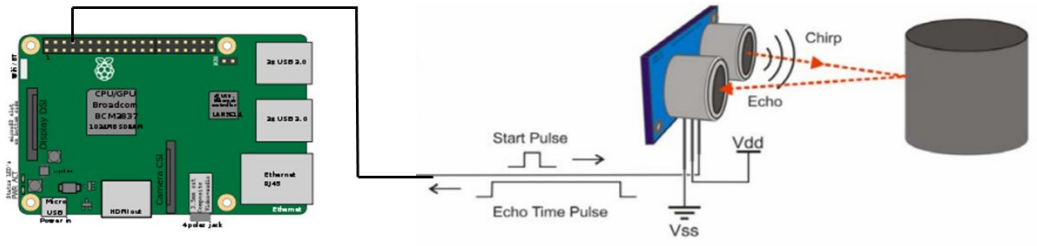


Figure 4: Ultrasonic Distance Sensors (borrowed and modified from Parallax)

A critical decision in our design was to implement three ultrasonic distance sensors, placed evenly across the front plane of the robot (see figure 5 for mounting prototype). This design constrains the robot to only forward movements and 90 degree rotations but this was deemed to be the best design path (i.e., to make the robot 'walk before it could run') because the A* pathing algorithm allows for this design. We purchased five Keywish HC-SR04 ultrasonic distance sensors for under $10 (i.e., a $2.00 unit cost) to meet this design. The sensor mount was created using Autodesk Fusion 360 and fabricated in the CSE machine shop.

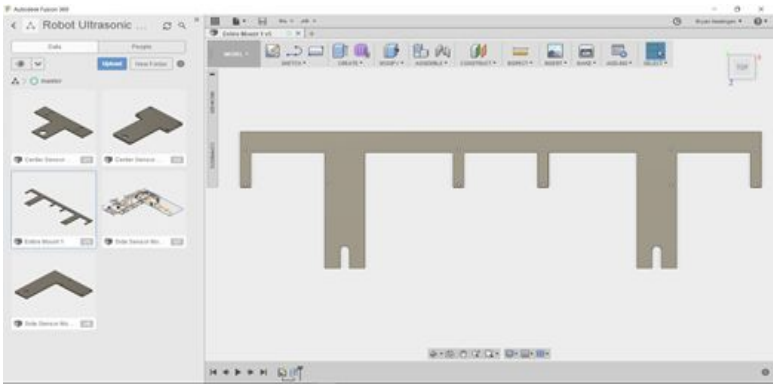


Figure 5: Sensor Mount Prototype

The ultrasonic distance sensors placed the biggest constraint on our design and proved difficult to debug. The sensors would detect false-positive objects at a distance of 50 cm. The root cause was deemed to be improper powering of sensors and the senior design lab had too much interference. Outside of the lab, the robot performed better but could only achieve up to 1m of accuracy. Moving forward it is recommended to change the environmental sensor design.

A Pathing Algorithm and Navigation*

Our navigation implementation takes in a relative distance and angle from the facial recognition software. We initialize both a continuous and discrete grid in the first quadrant of the cartesian coordinate plane. The continuous grid holds the destination location and the robots current location. The discrete grid consists of 35 cm x 35 cm blocks (i.e. the dimensions of the robot wheel base) and corresponds to our understanding of our environment. As we encounter obstacles by polling our sensors, we fill in these blocks or nodes, to record where obstacles were encountered. This means we have a discrete grid that consists of nodes and each node may or may not have an object.

A* is a directed graph search algorithm very similar to Dijkstra's search algorithm, and is demonstrated in figure 6. The only main difference is that A* has a heuristic to improve the efficiency of the search. Everytime an obstacle is in our current path, we recalculate a new path using A*. This is done by simply treating our discrete grid of blocks as a directed graph where each block has four neighbors corresponding to the blocks or nodes surrounding it. What this means is that everytime we have a destination we can generate a list of nodes that we should move to to get to our destination if such a route exists. This list of nodes is then converted into a list of movement commands and these commands are then executed until we reach our destination or we encounter an obstacle that requires us to recalculate our path.

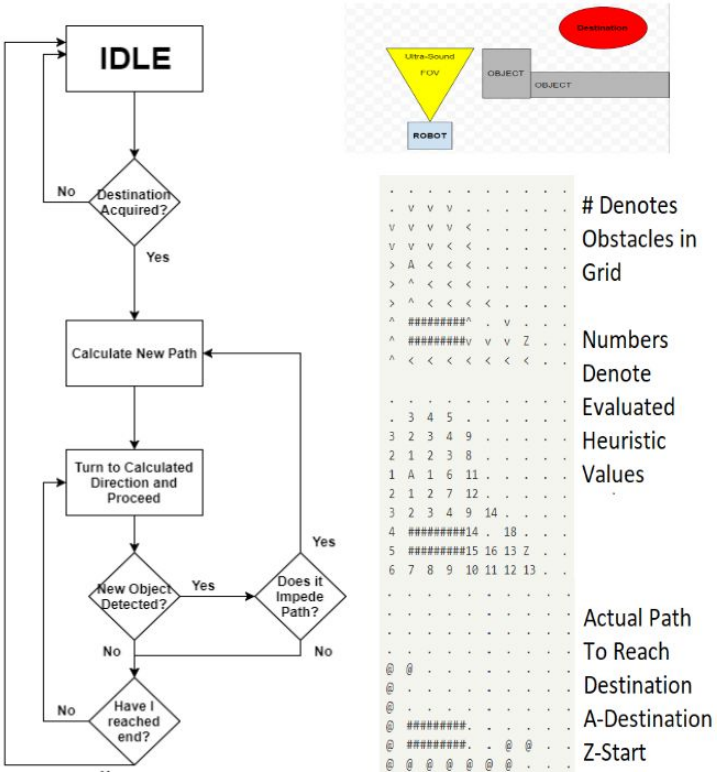


Figure 6: The above figure represents the navigation state diagram, an illustration of our basic navigation concept, and a full example of how we find a path from start to destination based on our current understanding of our environment

The A* pathing algorithm itself was tested with numerous scenarios and never failed. If a starting node and destination node is properly specified, a return path of nodes will be found if such a path does exist. Additionally, A* will always find an optimal or shortest path since it is an optimal algorithm. However, there are limitations to these statements. As stated earlier, our A* implementation contains square nodes that do not consider diagonal nodes to be neighbors. Therefore, our navigation system only allows us to navigate in 90 degree turns. This means that in reality, we do not travel in the shortest path, but the shortest straight lines along the x and y axis. Adding the ability to travel diagonally could decrease travel time. However, our current ultrasonic sensors do not pick up obstacles well at angles and therefore this can not be currently implemented.

If we assume complete knowledge of the environment, then A* is an optimal algorithm. However, we do not have the sensor capability to have complete understanding of the

environment at the start of our navigation and instead we add objects as we discover them. Therefore, there is no guarantee we will create an optimal path. There is only the guarantee that we will create the optimal node-based path based on our current information. The main way to improve performance would be to add a lidar based system to detect many more objects at the beginning. This would allow for an enhanced understanding of the environment at the start and decrease the need to plot a new path as we discover obstacles that render our old path obsolete.

Dlib Facial Recognition Software

In order to identify and locating a target person's face among multiple faces, the facial recognition software was implemented. The facial recognition uses a pre-trained neural network accessed using the Dlib toolkit with a Python API. This network is trained to recognize faces and generate a facial encoding of 68 landmark points (figure 8). The algorithm for the facial recognition is described in the figure 7 below.

The robot will scan the surrounding environment (i.e. take real-time images) and look for person to target. The 'dlib' library uses a pre-trained detector to extract 68 key points from a person's facial image. By feeding these landmark points into a pre-trained network, 128 measurements are returned, which are called "embeddings". Fortunately, each embedding is almost same for each specific person within different camera frames. In this way, it recognizes target person from others. The last step is to use a simple SVM classifier to compare the embeddings in the real-time image with our target's embedding. This classifier would yield the desired facial-recognition results.

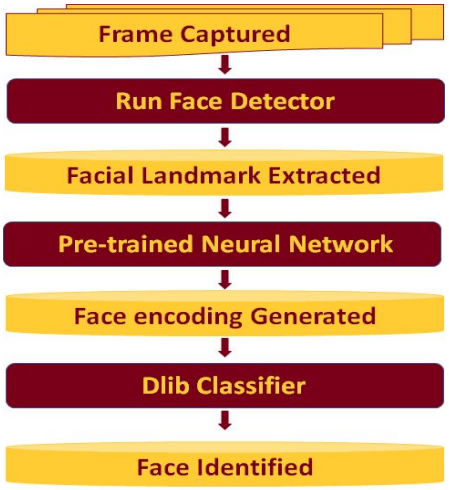


Figure 7: Face Recognition Algorithm

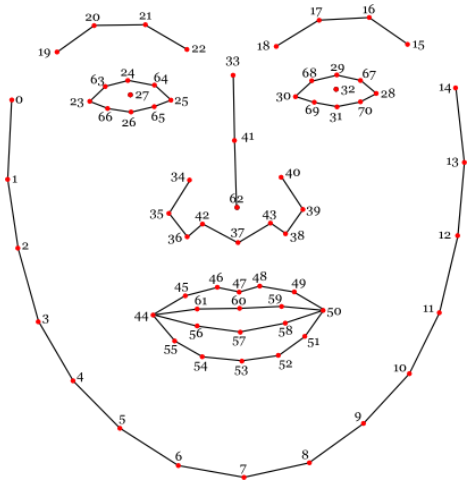


Figure 8: 68 face landmarks

Facial recognition software was also used to calculate distance and angle of the target face (see figure 9). This was accomplished by the target person taking their picture at a known distance, d0, which corresponds to the distance in this picture (e_frame_0) in number of pixels. The facial recognition software then calculated the distance（e_frame） between the eyes in real-time frame captured by camera, with the facial landmarks. The team implemented the following distance and angle algorithm in figure 9. So with these calculations, the robot is able to calculate the distance and angle relative to the center of the camera of a known-target person.

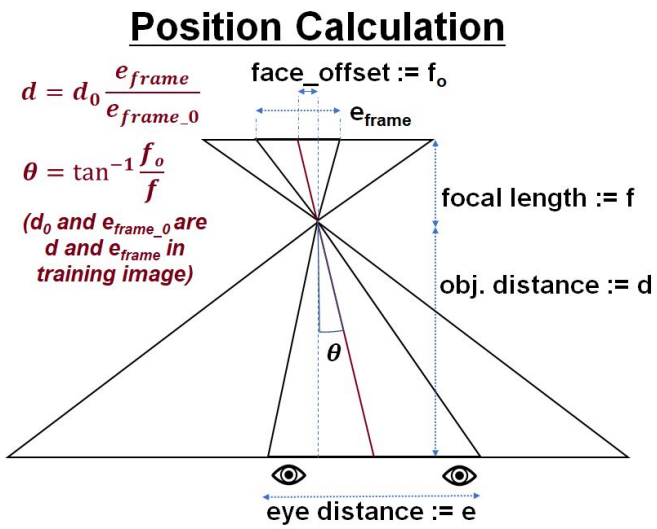


Figure 9: Face Distance and Angle Calculation

The precision of identification and position calculation is significantly determined by resolution of the face. The lower number of pixels representing the face, the inaccurate face-landmarks was extracted. However, this is a trade-off: high resolution caused more complex calculation, which drags down the computation speed. The solution we came up with is to crop only face regions on a reduced resolution frame and then raise resolution only in face regions to extract face landmarks; besides, facial-recognition was performed on every 4 frames.

The team has implemented facial recognition software with both the Raspberry Pi hardware and a cuda-based linux OS laptop. At beginning, we planned to wrap up all functions, including facial recognition, on the Raspberry Pi with a multi-processing structure. Because the computation capability is limited on the Raspberry Pi, it does not have GPU for speeding up neural network computation. Instead, it only has a 4-core CPU. However, multi-processing structure had never been worked on separate cores, which directly resulted in an unacceptable low speed for facial-recognition computation on Raspberry Pi. So, only as a part of the entire program, the facial recognition function can only use 1 core in CPU (only 25% of that limiting capability was used). We chose not to include the facial recognition software for our design show presentation due to its lengthy procedure. In order to demonstrate the performance, facial-recognition function was implemented on a cuda-based laptop. On the laptop, the software

runs smoothly in real-time on the laptop. We recommend implementing a more capable processor to handle the onboard facial recognition and image processing software.

Nexus Robot Omni-Wheel Drivetrain

FRODOW is equipped with the Nexus robot omni-wheel drivetrain. The Raspberry Pi serially communicates movement commands to the motors' onboard Arduino processor (movements are depicted in figure 10). The Arduino uses the OmniWheel library to set the individual wheel speeds in millimeters per second and the function PIDregulate() to monitor wheel speeds. Our software does not take advantage of the omni-direction capabilities due to the sensor mounting.

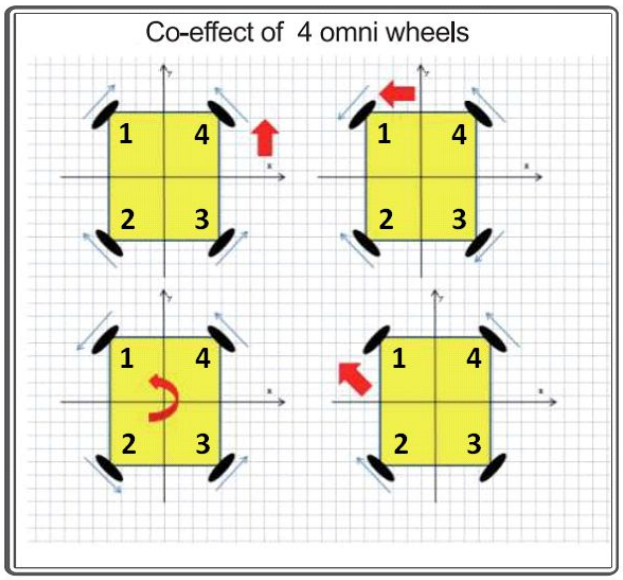


Figure 10: Wheel commands for respective robot movement (borrowed from Nexus)

FRODOW's drivetrain had troubles translating over a set distance, demonstrating left-drift and stopping-rotation during initial testing of the A* pathing algorithm. We decided to evaluate the root cause of the issue because the demonstration relied on accurate 90 degree rotations and forward movements. Individual wheel tests were performed by sending the Arduino to execute wheel#.setspeedMMPS() for each wheel separately. During the individual wheel tests, wheel 3's operation was audiably different than the others. For wheel 3, it was found that the steady-state error to a 300 mm/s step response, with sampling the wheel speed at 2 Hz, produced an

oscillating response (see figure 11). Whereas, wheel 1 (which is representative of wheels 2 and 4) did not demonstrate the same oscillating steady-state response. We were not able to fix wheel 3 due to time constraints and recommend individual PID tuning (i.e., exploring and reworking the PIDregulate() function) or replacing the motor for wheel 3.

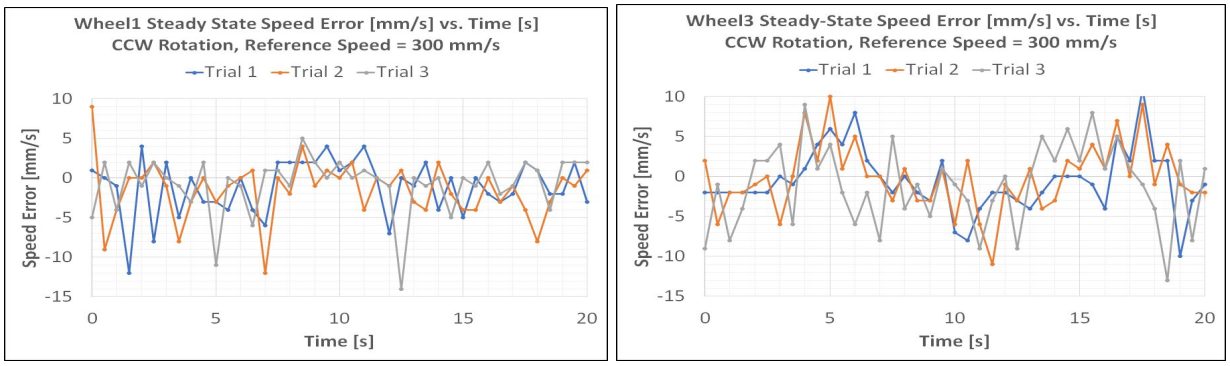


Figure 11: Steady-State Error Testing Results for Wheel 1 and 3

FRODOW's drivetrain also has some intricacies with initializing and stopping the motors leading to rotational errors. Currently, the Arduino can only initialize and stop one motor at a time which causes differential torques on the robot as it comes to a stop and begins its movements. We rearranged the order of wheel initialization and found that this had an effect on which way the robot would drift through its forward movement. This due to an initial rotation of the robot before it would translate forward. Since we did not change the motor stopping sequence, each of these tests resulted in the same ending clockwise rotation. Therefore, we concluded that the Arduino initializing/stopping sequences as the main source for translational error and rotational error. The current stopping sequence results in an added 4 degrees of CW rotation from a 90 degree CW command and an added 2 degrees of CCW rotation from a 90 degree CCW command. Therefore, we decided it would be best to program movement functions using an absolution-orientation sensor (i.e., inertial measurement unit) in order to protect future FRODOW projects from drivetrain inconsistencies and intricacies.

Adafruit Inertial Measurement Unit

We added an Adafruit BNO005 Inertial Measurement Unit (i.e., IMU) onto the platform to protect future FRODOW projects from drivetrain inconsistencies and intricacies. We researched absolute-orientation sensors and found a high performing unit for low cost. The IMU is marketed as a 9-Axis Abs. Orientation sensor outputting euler vector orientation, quaternion, angular velocity, acceleration vector, magnetic field strength vector, linear acceleration vector, gravity vector, and temperature. Our current design solely uses the heading output for rotational commands because we were late into the semester installing this hardware (see figure 12). The IMU was used to correct the rotational error due to the stopping sequence. We calibrated our rotational command to have +/- 1 degree of rotational error over a 90 degree rotation. An interesting note is how the heading angle is provided from the IMU, positive angles follow a clockwise rotation about the heading axis.

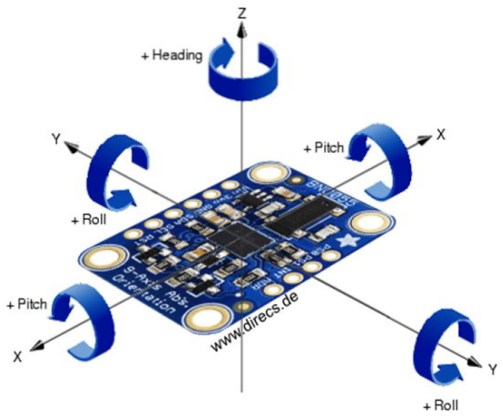


Figure 12: Rotational Angles Available using Adafruit BNO055 IMU

Our biggest challenge was installing the IMU software onto the Raspberry Pi. The standard communication protocol for the sensor is I2C, but the Pi has difficulty with this timing. Therefore it was suggested to use UART communication. In order to make this work on the Pi, the Bluetooth functionality needed to be turned off. There are still some errors in communication due to wrong bytes being received, but this is all a timing error. After putting the Pi into turbo mode, it reduced the number of communication errors when initializing the demonstration

function. Future software should keep this in mind and set a tolerance level of communication errors before shutting the entire demonstration down.

Conclusions and Recommendations

Our design is the first step toward creating a fully-autonomous obstacle-avoiding robot. The addition of the IMU and ultrasonic sensors have increased the pathing and avoidance capabilities of the robot. By becoming fully autonomous, this robot will serve as a platform to test future facial recognition software. FRODOW can be used in other industries such as military and healthcare applications. Seniors and people with disabilities can benefit from having a service robot that is capable of recognizing people, serving as a platform to distribute medication and for safety. Another application includes security in public places, being able to target person's of interest, bombs and protecting surrounding individuals. The future is bright for the FRODOW platform.

We were able to meet most of our self-prescribed specifications and remain under the budget of $300. We were able to create a platform that was capable of detecting objects and stopping before it ran into them. We were also able to successfully path to a targeted individual. The team has demonstrated a successful design through testing the robot's capabilities. The A* pathing algorithm has achieved an accuracy of ±1 cm per movement over 35 cm node distances. The ultrasonic distance sensors have demonstrated an average range of 1m, which is an acceptable range for our demonstration. The movement software and the IMU have been calibrated to ensure the robot's rotations have an accuracy of ±1° rotational error. The facial recognition software was not implemented on the Pi because it is computationally intensive.

Future FRODOW project recommendations include:

- More robust sensor package for 360° obstacle avoidance
- Take advantage of omni-directional movements vs using 90° rotations using IMU
- Update PID algorithms (tune or replace wheel 3's motor)
- Real time facial-recognition and identification. Use a more powerful alternative to the Raspberry Pi such as the NVIDIA Jetson TK1. Also modify facial-recognition software for multiprocessing in order to further improve performance.

References

Adafruit BNO055 Absolute Orientation Sensor. (n.d.). Retrieved December 14, 2017, from https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview

Dlib Face Recognition API for Python. (n.d.). Retrieved December 15, 2017, from https://github.com/ageitgey/face_recognition

Image Recognition | TensorFlow. (n.d.). Retrieved December 14, 2017, from https://www.tensorflow.org/tutorials/image_recognition

Implementation of A*. (n.d.). Retrieved December 16, 2017, from https://www.redblobgames.com/pathfinding/a-star/implementation.html#python-astar

Keywish HC-SR04 Ultrasonic Distance Sensor. (n.d.). Retrieved December 14, 2017, from https://www.amazon.com/Keywish-Ultrasonic-Distance-Duemilanove-Raspberry/dp/B071W9689R

Nexus 4WD Omni-directional Robot. (n.d.). Retrieved December 14, 2017, from https://www.robotshop.com/en/4wd-omni-directional-arduino-compatible-mobile-robot.html?gclid=Cj0KCQiA38jRBRCQARIsACEqIeuI0jPS9KxSJuvRcSidKeYVArKMNfTch9uWFYcA6yHpFltqrMIqxYkaAsssEALw_wcB

Parallax PING))) Ultrasonic Distance Sensor. (n.d.). Retrieved December 14, 2017, from https://www.parallax.com/product/28015

Appendices

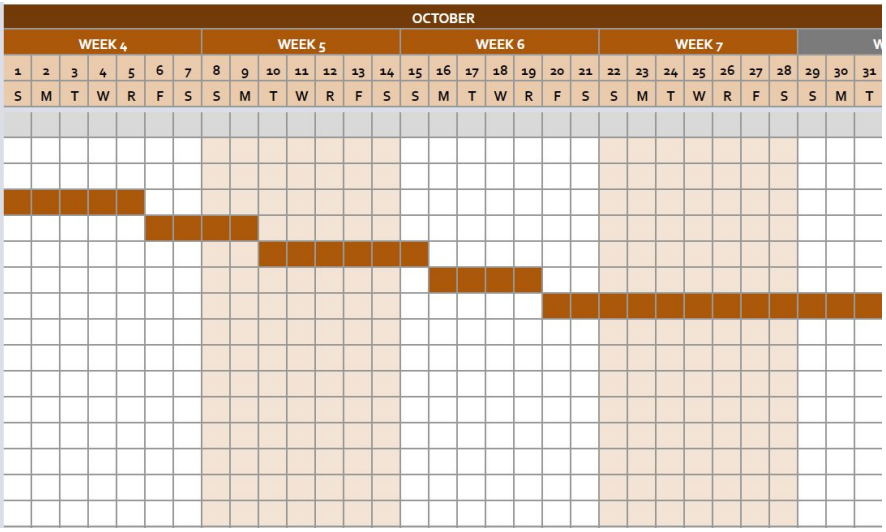
Appendix A: Budget

Item #	Product Description	Vendor	Vendor Part #	Per Unit Cost ($)	Qty	Total Cost
1	Raspberry PI 3 Model B 1.2GHz 64-bit quad-core ARMv8 CPU, 1GB RAM	Amazon	See description	35.20	1	35.20
2	Raspberry Pi 7" Touchscreen Display	Amazon	See description	69.99	1	69.99
3	Ultrasonic Sensor HC-SR04 (5-pk) - 8.99	Amazon	See description	8.99	1	8.99
4	Adafruit IMU Fusion Breakout BNO055	Microcenter	See description	37.62	1	37.62
5	Breadboard	ECE Depot	See description	2.36	3	7.08
6	Wire with pre-crimped connectors (long)	ECE Depot	See description	0.45	6	2.70
7	Wire with pre-crimped connectors (med)	ECE Depot	See description	0.35	23	8.05
8	Wire with pre-crimped connectors (short)	ECE Depot	See description	0.20	9	1.80
9	Connector housing (large)	ECE Depot	See description	0.40	2	0.80
10	Connector housing (1x4)	ECE Depot	See description	0.05	9	0.45
11	Connector housing (1x1)	ECE Depot	See description	0.02	28	0.56
12	Cable, USB A male to USB B female	ECE Depot	See description	2.45	1	2.45
13	Plano Case for electronics	ECE Depot	See description	8.99	1	8.99
			Total Cost			184.68

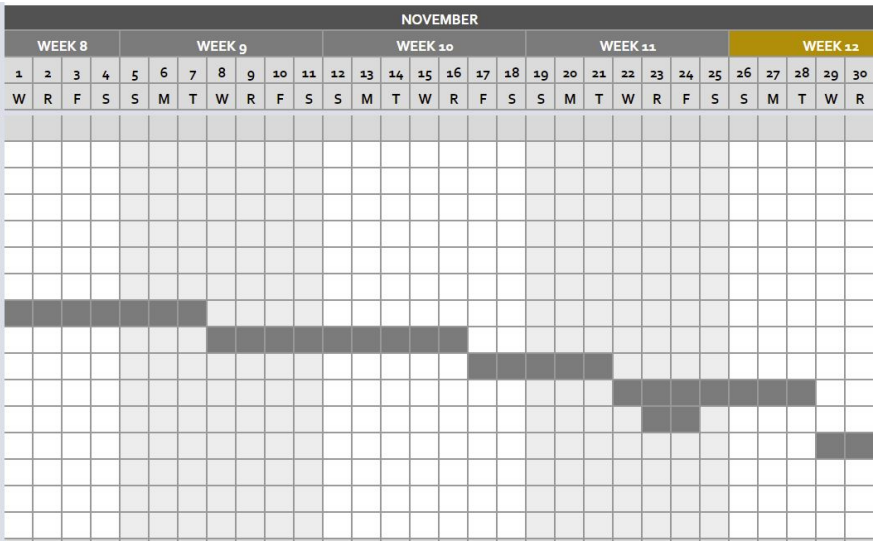
Appendix B: Project Due Dates Gantt Chart

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
1	Project Due Dates			
1.1	Draft Design Specs & Timeline	9/11/17	9/17/17	7
1.2	Final Design Specs & Timeline	9/18/17	9/24/17	7
1.3	Written Design Proposal	9/25/17	10/5/17	11
1.4	Design Review Presentation	10/6/17	10/9/17	4
1.5	Team and Peer Performance Evaluations	10/10/17	10/15/17	6
1.6	Team Meetings with Coordinator and TA	10/16/17	10/19/17	4
1.7	Peer Review of Executive Summaries	10/20/17	11/7/17	18
1.8	Executive Summary Draft	11/8/17	11/16/17	9
1.9	Written Individual Reports	11/17/17	11/21/17	5
1.10.	Executive Summary Final	11/22/17	11/28/17	7
1.11	Thanksgiving	11/23/17	11/24/17	2
1.12	Poster Preview Session	11/29/17	12/5/17	7
1.13	Product Launch Presentations/Demos	12/6/17	12/7/17	2
1.14	Design Show (2:00-4:30pm)	12/8/17	12/12/17	5
1.15	Final Written Report/Peer Perf Evals	12/13/17	12/18/17	6

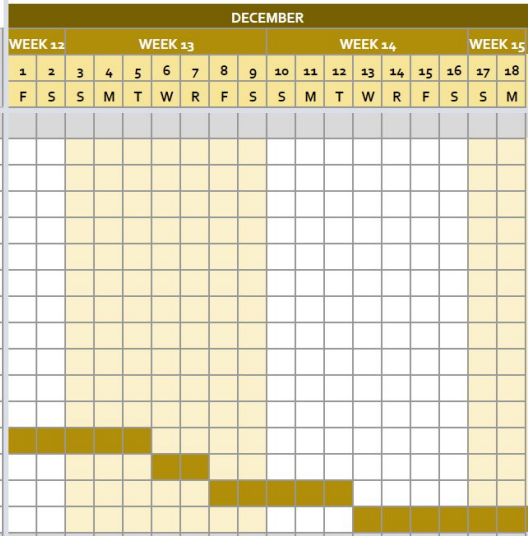
WBS NUMBER	TASK TITLE
1	Project Due Dates
1.1	Draft Design Specs & Timeline
1.2	Final Design Specs & Timeline
1.3	Written Design Proposal
1.4	Design Review Presentation
1.5	Team and Peer Performance Evaluations
1.6	Team Meetings with Coordinator and TA
1.7	Peer Review of Executive Summaries
1.8	Executive Summary Draft
1.9	Written Individual Reports
1.10.	Executive Summary Final
1.11	Thanksgiving
1.12	Poster Preview Session
1.13	Product Launch Presentations/Demos
1.14	Design Show (2:00-4:30pm)
1.15	Final Written Report/Peer Perf Evals



WBS NUMBER	TASK TITLE
1	Project Due Dates
1.1	Draft Design Specs & Timeline
1.2	Final Design Specs & Timeline
1.3	Written Design Proposal
1.4	Design Review Presentation
1.5	Team and Peer Performance Evaluations
1.6	Team Meetings with Coordinator and TA
1.7	Peer Review of Executive Summaries
1.8	Executive Summary Draft
1.9	Written Individual Reports
1.10.	Executive Summary Final
1.11	Thanksgiving
1.12	Poster Preview Session
1.13	Product Launch Presentations/Demos
1.14	Design Show (2:00-4:30pm)
1.15	Final Written Report/Peer Perf Evals

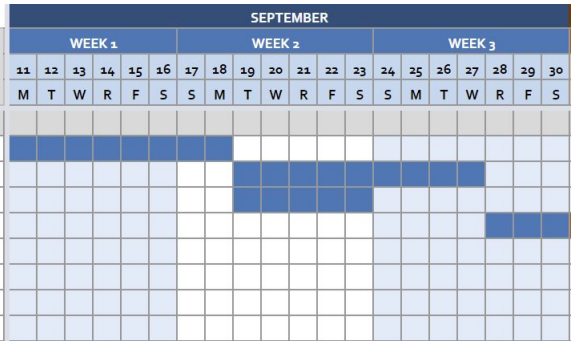


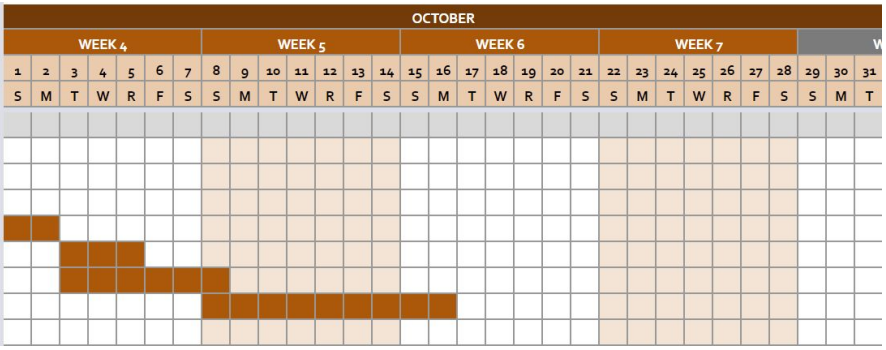
WBS NUMBER	TASK TITLE
1	Project Due Dates
1.1	Draft Design Specs & Timeline
1.2	Final Design Specs & Timeline
1.3	Written Design Proposal
1.4	Design Review Presentation
1.5	Team and Peer Performance Evaluations
1.6	Team Meetings with Coordinator and TA
1.7	Peer Review of Executive Summaries
1.8	Executive Summary Draft
1.9	Written Individual Reports
1.10.	Executive Summary Final
1.11	Thanksgiving
1.12	Poster Preview Session
1.13	Product Launch Presentations/Demos
1.14	Design Show (2:00-4:30pm)
1.15	Final Written Report/Peer Perf Evals

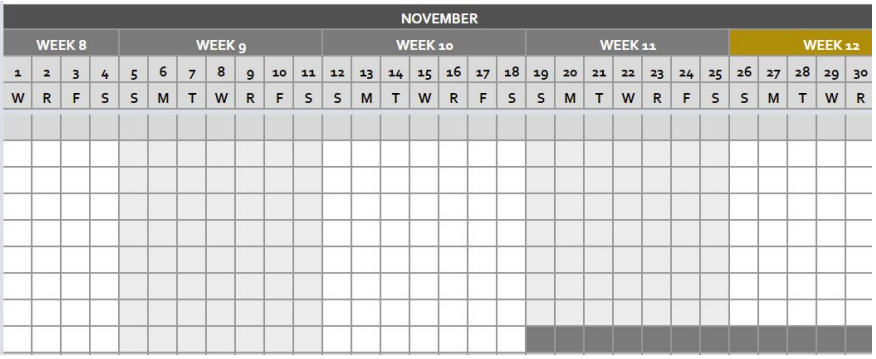


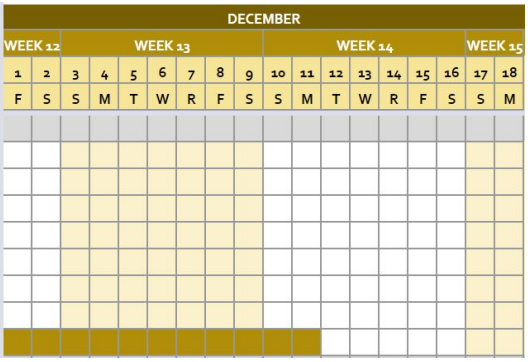
Appendix C: Project Definition and Planning Gantt Chart

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
2	Project Definition and Planning			
2.1	Scope & Goal Setting	9/11/17	9/18/17	8
2.2	Learn Pre-existing code base	9/19/17	9/27/17	9
2.3	Collect Inventory and Documentation	9/19/17	9/23/17	5
2.4	Draft Written Design Proposal	9/28/17	10/2/17	5
2.5	Final Written Design Proposal	10/3/17	10/5/17	3
2.6	Demonstration Planning	10/3/17	10/8/17	6
2.7	Demonstration Algorithm Creation	10/8/17	10/16/17	9
2.8	Demonstration Algorithm Finalization	11/19/17	12/12/17	24



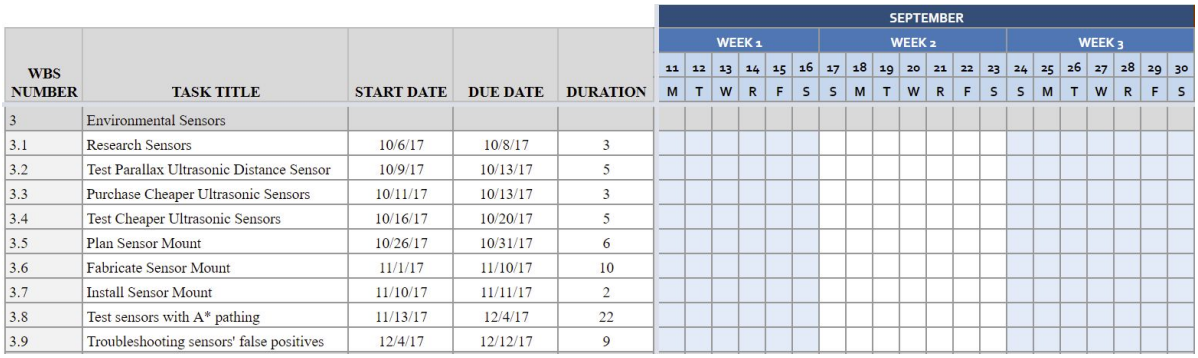


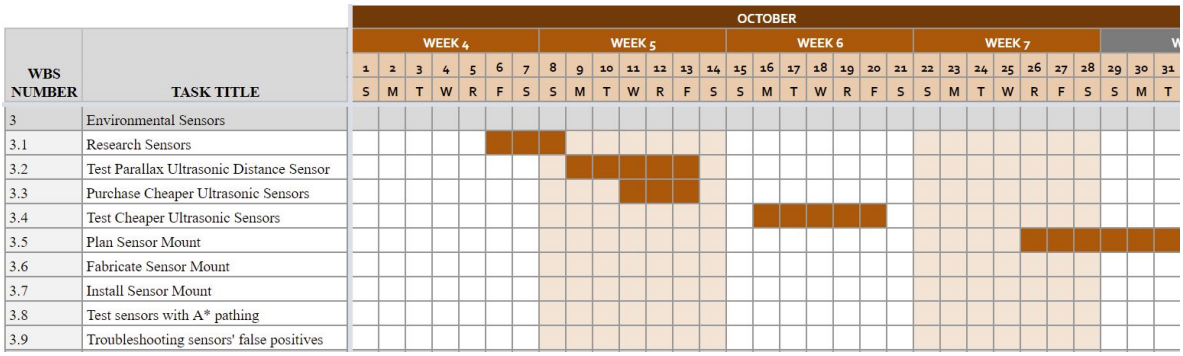


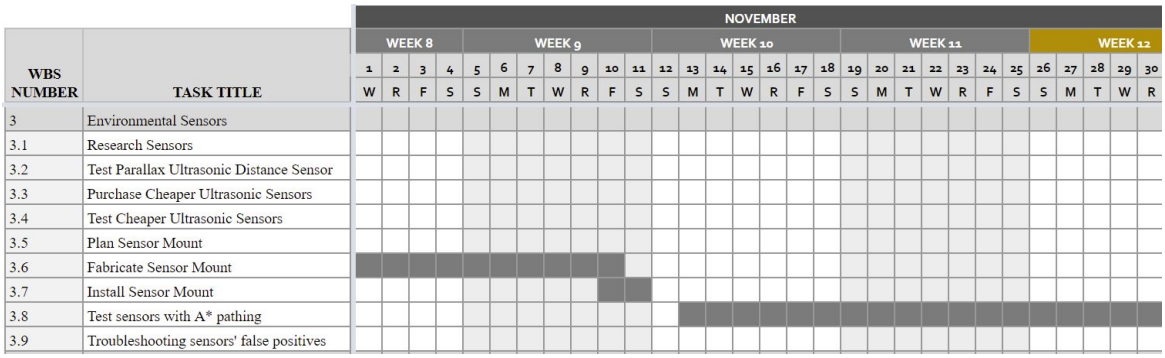


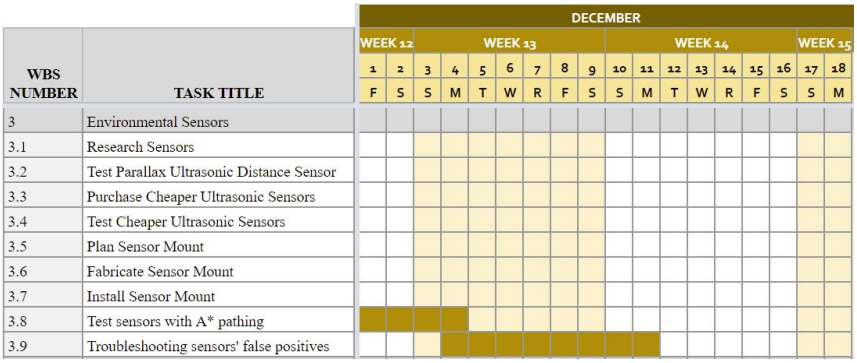
Appendix D: Environmental Sensors Gantt Chart

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
3	Environmental Sensors			
3.1	Research Sensors	10/6/17	10/8/17	3
3.2	Test Parallax Ultrasonic Distance Sensor	10/9/17	10/13/17	5
3.3	Purchase Cheaper Ultrasonic Sensors	10/11/17	10/13/17	3
3.4	Test Cheaper Ultrasonic Sensors	10/16/17	10/20/17	5
3.5	Plan Sensor Mount	10/26/17	10/31/17	6
3.6	Fabricate Sensor Mount	11/1/17	11/10/17	10
3.7	Install Sensor Mount	11/10/17	11/11/17	2
3.8	Test sensors with A* pathing	11/13/17	12/4/17	22
3.9	Troubleshooting sensors' false positives	12/4/17	12/12/17	9



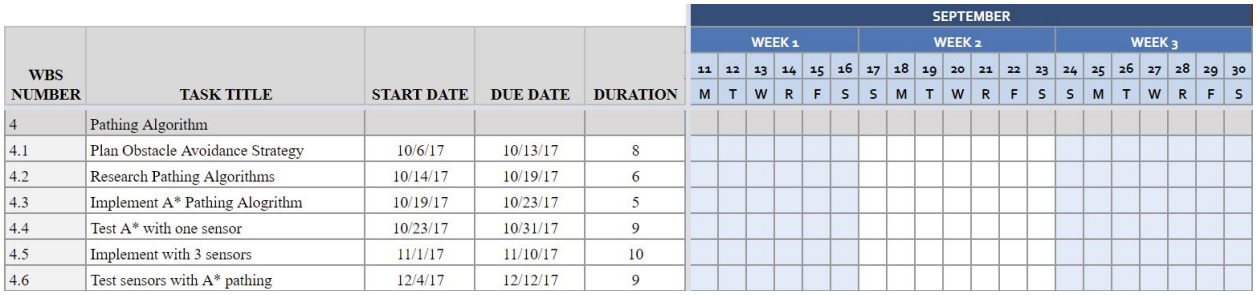


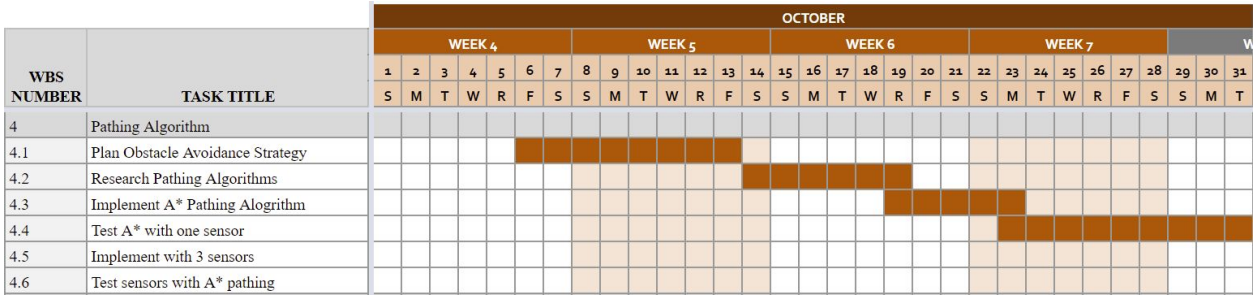


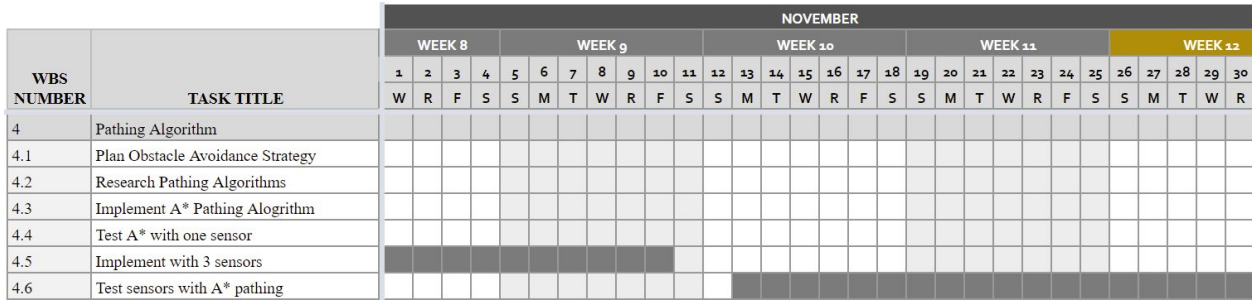


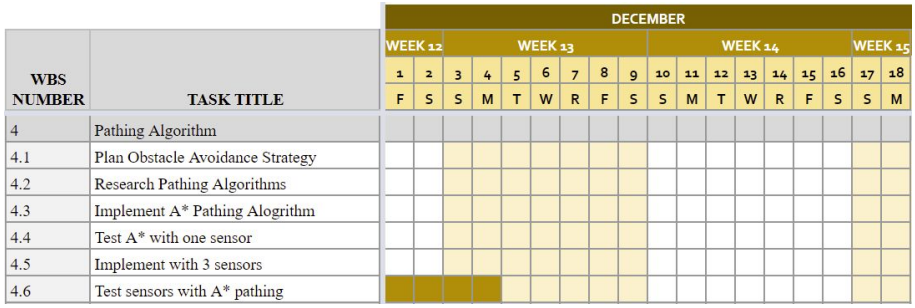
Appendix E: Pathing Algorithm Gantt Chart

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
4	Pathing Algorithm			
4.1	Plan Obstacle Avoidance Strategy	10/6/17	10/13/17	8
4.2	Research Pathing Algorithms	10/14/17	10/19/17	6
4.3	Implement A* Pathing Alogrithm	10/19/17	10/23/17	5
4.4	Test A* with one sensor	10/23/17	10/31/17	9
4.5	Implement with 3 sensors	11/1/17	11/10/17	10
4.6	Test sensors with A* pathing	12/4/17	12/12/17	9

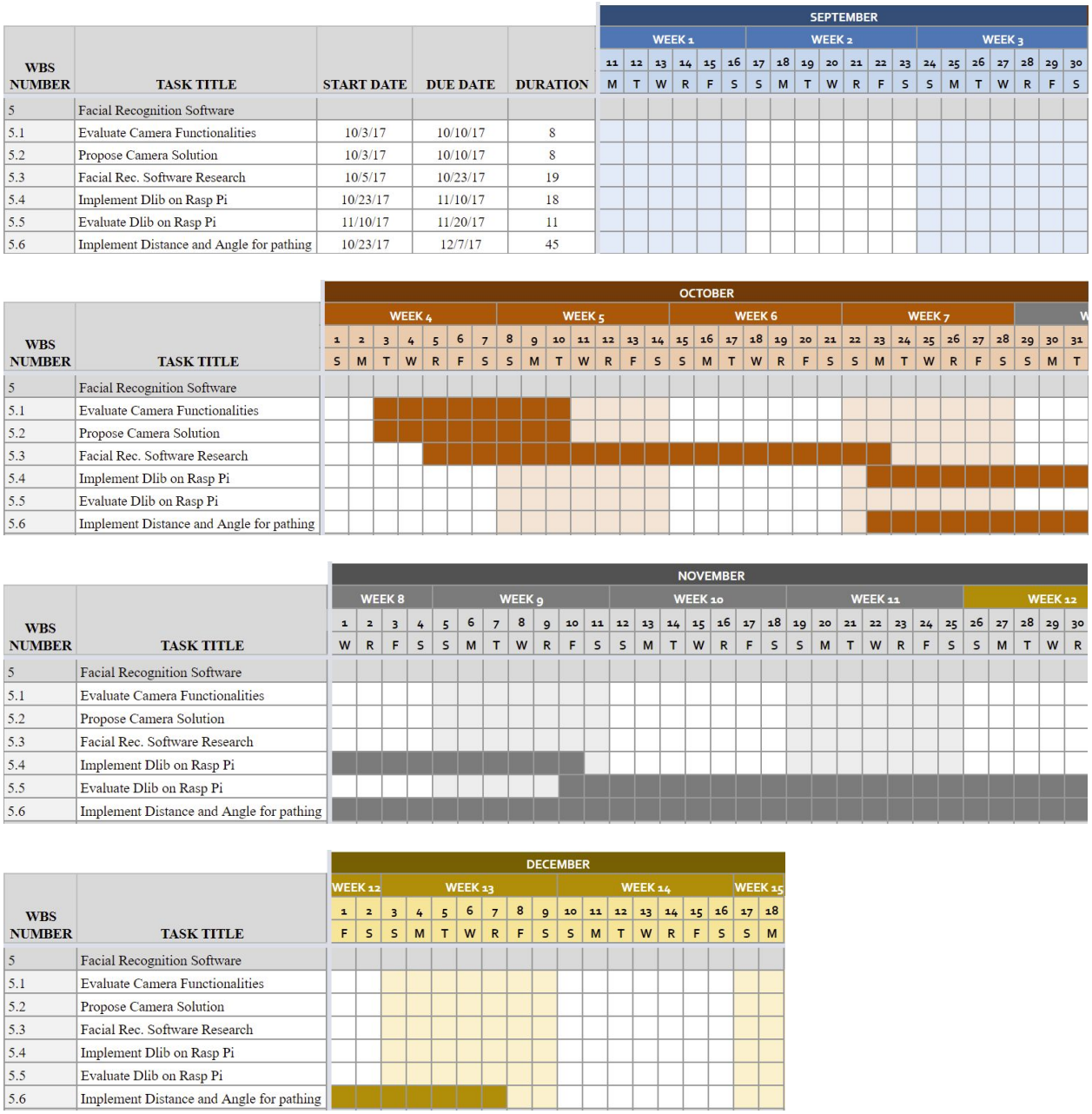








Appendix F: Facial Recognition Software Gantt Chart



WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
5	Facial Recognition Software			
5.1	Evaluate Camera Functionalities	10/3/17	10/10/17	8
5.2	Propose Camera Solution	10/3/17	10/10/17	8
5.3	Facial Rec. Software Research	10/5/17	10/23/17	19
5.4	Implement Dlib on Rasp Pi	10/23/17	11/10/17	18
5.5	Evaluate Dlib on Rasp Pi	11/10/17	11/20/17	11
5.6	Implement Distance and Angle for pathing	10/23/17	12/7/17	45

Appendix G: Nexus Robot Drive Train Gantt Chart

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
6	Nexus Robot Drive Train			
6.1	Learn Software Function s for motors	10/6/17	10/16/17	11
6.2	Evaulate/Plan Drive Train Movements	10/16/17	11/4/17	19
6.3	Implement IMU to Prevent DT Issues	11/4/17	11/16/17	13
6.4	Implement Rotation Function with IMU	11/16/17	11/25/17	10
6.5	Test and Calibrate rotate_gyro()	11/25/17	12/4/17	10