Auto-Park for Social Robots

Collaborative Parking between Autonomous Vehicles Utilizing Point-to-Point Communications

Team Daedalus: Dorothy Kirlew, Mohak Bhardwaj, Pranav Maheshwari, Richa Varma, and Shivam Gautam
Masters of Science Robotics Systems Development
9/27/2015

Carnegie Mellon
THE ROBOTICS INSTITUTE

Table of Contents

1. Project Description

1.1. Keywords

- **Optimal Spot** – The optimal spot is the parking spot with the shortest optimal route between that spot and the exit. The optimal spot must also be unoccupied.
- **Optimal Route** – Also called **Optimal Path**. The route is from the vehicle's current position to the optimal spot or to the exit, depending on the vehicle's status. It is optimal because it takes the least amount of time to traverse.
- **Vehicle** – Also called **Mobile Platform** or **Robot**. This serves as the test platform to implement and showcase the collaborative and autonomous aspects of the project, such as path planning, navigation, communication, and obstacle detection.
- **Vehicle Status** – The vehicle can be in the following states:
    - **Free** – The user has not yet told the vehicle to park. The vehicle is waiting for a command.
    - **Parking** – The vehicle has been told to park by the user. The vehicle is heading towards the optimal spot, but has not reached it yet.
    - **Parked** – The vehicle is stationary within the parking spot.
    - **Returning** – The vehicle has been told to return by the user. The vehicle is heading towards the parking lot exit, but has not reached it yet.
    - **Returned** – The vehicle is at a complete stop in the parking lot exit and is waiting for the user.
- **Parking Lot** – The parking lot is a single-level testing area with a known entrance, exit, and known parking spots. The lot will fit the size of the vehicle.

1.2. Description

The imminent arrival of driverless cars has led to an increased focus on the development of an ecosystem that supports driverless cars. This project, "Auto-Park for Social Robots", aims at developing an autonomous system for collaboratively parking driverless cars. The project, as envisioned by the team and the sponsor, United Technologies Research Center, would allow a person with a driverless car to park the vehicle by simply pressing "Park" on their Android App.

The motivation for the project stems from key factors affecting the current parking system, such as poor parking safety standards, parking industry growth potential, and a competitive advantage of developing such a system. According to reports by Fayard and Stark, around 20% of all automobile accidents occur inside parking lots. About 90% of the people involved in these accidents are injured. The parking industry is worth $25-$30 billion and has potential to invest in upgrades [1] [2]. It regularly bleeds money to accident insurance claims, as well as public transportation due to people that take the bus rather than find a spot in a congested lot. The precious time wasted while searching for a parking spot, translated into profit, is also a motivation behind the project.

Auto-Park for Social Robots would consist of a scaled-down version of a car (mobile platform) that would be able to localize itself in a parking lot (see Figure 1 – Entering Parking Lot), collaborate with other cars to identify the best possible parking spot, navigate to the spot (see Figure 2 – Choosing Optimal SpotFigure 2), and avoid any obstacles in the process. The vehicle would then exit the parking spot and navigate to the parking lot exit (see Figure 3). The team's main focus will be on establishing a robust system for effective collaboration between vehicles.

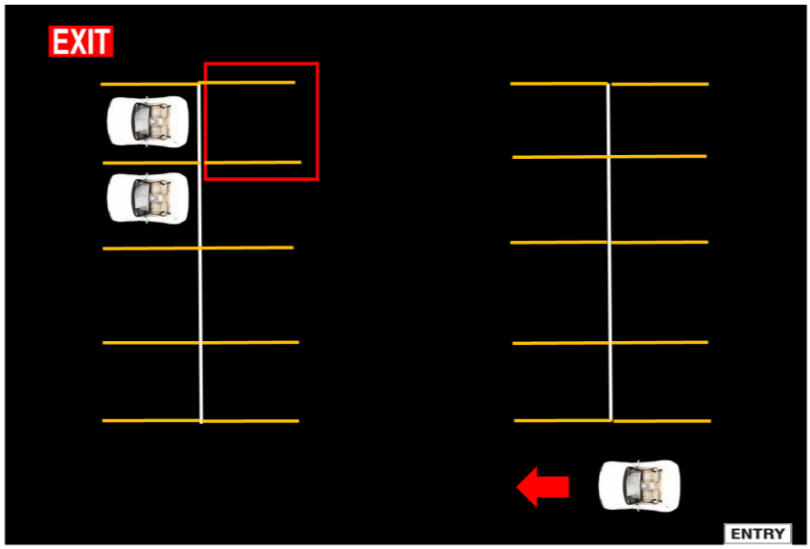
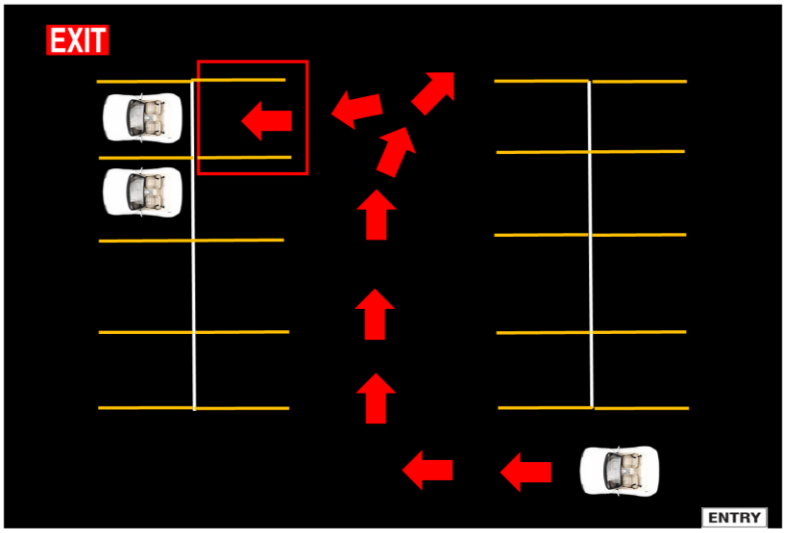
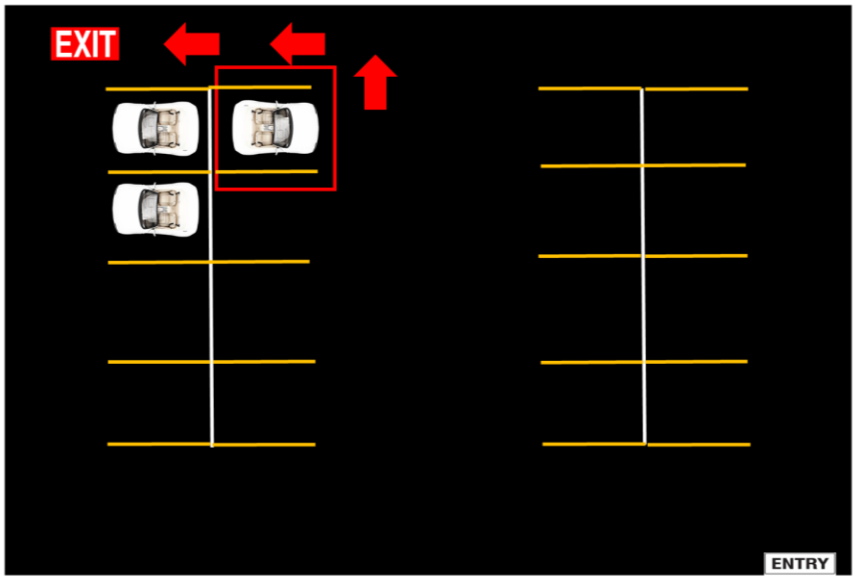


Figure 1 – Entering Parking Lot



Choice of most optimal spot and path while parking (closest to exit). Backing into spot for faster exit.

Figure 2 – Choosing Optimal Spot



Choice of most optimal spot and path to exit

Figure 3 – Exiting Spot and Lot

2. Use Case

Benjamin is a retired Armed Forces veteran. He has been driving in Pittsburgh, his home city, for many years now. The only thing that has made him think twice before taking his car out is the nightmare of parking.

Recently, Benjamin was diagnosed with prostate cancer and is receiving treatment at UPMC, 10 miles away from his home. The parking lot at the hospital is always extremely congested and it sometimes takes him longer to find a parking space than to drive from his home. He has to leave an hour and a half before his appointment and is frequently late. After his appointment last week, Ben was backing out of a parking spot when a reckless driver sped into his bumper at 30 mph. He spent the next few weeks handling insurance claims and getting repairs done. After the accident, Benjamin started taking the Port Authority bus to the hospital, even though he had to transfer buses several times to reach the hospital. Benjamin missed driving, but the scare from the accident had made him give it up altogether.

Five years have passed. It is 2020 and Benjamin has purchased an autonomous car equipped with CMU-UTRC Auto-Park System. His new car now takes him to the hospital for his weekly appointments – efficiently, safely, and without any hassle. When Benjamin's car stops at the entrance of the hospital, he exits his car and presses "Park" on his smartphone app. The app sends the command to his driverless car. Benjamin enters the hospital and his car autonomously enters the queue at the entrance of the parking lot. As it enters, it connects to the network of cars already inside the parking lot. Benjamin's car receives a notification from the network containing information about the closest free parking spot to the exit. His car autonomously navigates to the spot, occasionally stopping to give way to exiting cars.

Benjamin is tired at the end of his appointment and wants to get home as quickly as possible. He hits the "Return" button on his app and walks towards the hospital exit. His car receives the command and alerts nearby cars that it about to exit the spot. An incoming vehicle stops to allow Benjamin's car to exit the parking spot safely. His car exits the spot and navigates

to the exit queue to wait for Benjamin. By the time Benjamin reaches the hospital exit, his car is waiting for him. He enters his car, grateful that he did not have to walk through the large parking lot after such a long day.

3. System-Level Requirements

The system-level requirements are categorized as Mandatory (M) or Desirable (D), as well as Functional (F) and Non-functional (N). The requirement titles are meant to be read as "The system shall…", followed by the title. The description offers more detail about how the requirement applies to the system, and the performance metric(s) detail how the requirements can be measured.

Table 1 – Mandatory Functional System Level Requirements

ID	Title	Description	Performance Metric(s)
MF.1	Receive commands from user via smartphone app	User will send "Park" and "Return" commands to the appropriate vehicle from Android app interface.	95% of messages will be received.
MF.2	Share data with other vehicles	Vehicles will send and receive basic information regarding location, available and occupied spots, and obstacles by joining the network.	They will establish communication with other vehicles within 30 seconds of approaching the parking lot. 90% of messages will be received.
MF.3	Navigate autonomously through parking lot	Robot will maneuver through parking lot without being controlled by an operator.	100% of navigation will be autonomous.
MF.4	Plan optimal route to exit	When the vehicle receives the "Return" command, the vehicle will plan the fastest path to the exit.	The vehicle will exit the parking lot within 90 seconds of receiving command.
MF.5	Follow optimal route to exit	The vehicle will follow the optimal route once it is planned. It will maintain a safe velocity.	The vehicle will maintain a velocity between 0 and 10 cm/sec.
MF.6	Park in optimal parking spot	The vehicle will park in the parking spot in a reliable manner.	The vehicle will park 100% within a parking spot within two attempts. When parked, the vehicle will be within 35º of parallel with the neighboring vehicles or the lines of the spot, as applicable.
MF.7	Exit parking spot	The vehicle will exit the spot without colliding with other vehicles or infrastructure.	The vehicle will exit the spot within two tries without colliding with other vehicles or infrastructure.
MF.8	Sense the environment	The vehicle will identify infrastructure, parking spots, static vehicles, and other static obstacles.	Obstacles are between 1-50 cm high and 2-120 cm wide.
MF.9	Avoid infrastructure	The vehicle will maintain a safe distance between a vehicle and infrastructure. Parked vehicles are infrastructure.	The vehicle will maintain a distance of 30.48 cm (1 ft.) between a vehicle and infrastructure.

ID	Title	Description	Performance Metric(s)
MF.10	Stop in the event of an emergency	The vehicle shall stop in the event of an emergency, such as an obstacle or internal vehicle error.	The vehicle shall stop within 3 seconds of an emergency, such as an obstacle or internal vehicle error.

Table 2 – Desirable Functional System Level Requirements

ID	Title	Description	Performance Metric(s)
DF.1	Identify optimal parking spot	Parking spots in lot are ranked based on proximity to exit. Vehicles in the lot will communicate this to each other when possible, as well as when a spot becomes occupied or free.	Optimal spot is identified 98% of time. If incorrect spot is chosen, it is within 5% of optimal spot.
DF.2	Plan optimal route to spot	When optimal spot is identified, the vehicle will plan the shortest path from its location to the spot.	Optimal path is chosen 90% of time.
DF.3	Follow optimal route to spot	The vehicle will follow the optimal route once it is planned. It will maintain a safe velocity.	The vehicle will maintain a velocity between 0 and 10 cm/sec.
DF.4	Avoid other vehicles	The vehicle will maintain safe distance between the front of one moving vehicle and the back of another moving vehicle.	The vehicle will maintain at least 60.96 cm (2 ft.) between the front of one moving vehicle and the back of another moving vehicle.

Table 3 – Mandatory Non-Functional System Level Requirements

ID	Title	Description	Performance Metric(s)
MN.1	Use smartphone app to communicate with user	The vehicle will communicate its status to the user. The app will show whether the vehicle is free, parking, parked, returning, or returned.	95% of messages are received.
MN.2	Communicate reliably between local vehicles	If the vehicle loses network connection, it will rejoin quickly without losing information.	The vehicle will rejoin within 30 seconds. No information will be lost in this time.
MN.3	Efficiently exits the parking spot	The vehicle will exit the parking spot as efficiently and quickly as possible.	The vehicle will exit the parking spot within two attempts. It will take no more than 45 seconds to exit the parking spot.
MN.4	Return to user as quickly as possible	When the "Return" command is received, the vehicle will arrive at the exit quickly.	The vehicle will arrive at the exit within 90 seconds of receiving the "Return" command.
MN.5	Make minimal changes to infrastructure	No changes will be made to the parking lot, nor will a central server be installed for the system to operate.	There will be 0 changes to the infrastructure.
MN.6	Be within $4000 budget	The cost of vehicles and equipment will not exceed the budget provided by the MRSD program.	The budget is $4000.

Table 4 – Desirable Non-Functional System Level Requirements

ID	Title	Description	Performance Metric(s)

ID	Title	Description	Performance Metric(s)
DN.1	Maintain scalable network of vehicles	The network will be able to accommodate several vehicles.	The network will be able to accommodate at least three vehicles.
DN.2	Efficiently maneuver throughout the lot	The vehicle will choose the fastest route possible to reach their destination.	The vehicle will choose the fastest route (in time) 98% of the time.
DN.3	Efficiently enter the parking spot	The vehicle will back into the parking spot quickly.	The vehicle will back into the parking spot within two attempts. It will take no more than 45 seconds to back into spot.

4. Functional Architecture

The functional architecture shown in Figure 4 – Functional Architecture depicts the process flow of the Auto-Park for Social Robots with the inputs on the left, outputs on the right, and the internal functions of the system in the center box. The direction of arrows represents the flow of material, energy and information from one function to another. The purple box represents the process when a vehicle parks and the blue box represents the process when a vehicle returns.

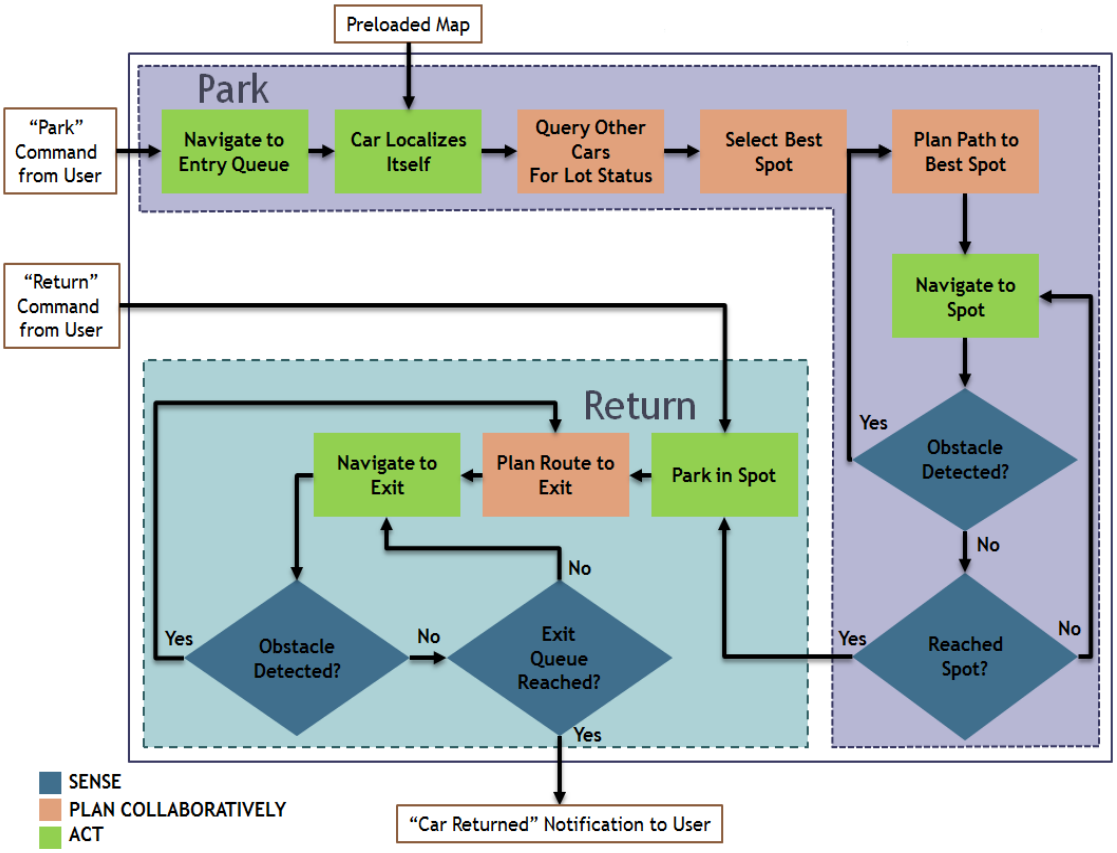


Figure 4 – Functional Architecture

4.1. User Interface

The smartphone app-based user interface is responsible for providing a reliable and simple method of communication between the user and the vehicle.

- **GUI**: The main screen in the android app shall have two buttons labelled "**Park**" and "**Return**". Pressing a button will trigger the corresponding command. See Figure 5 – Sample App Interface for an example of the app interface. Note: when the vehicle is parked, the user cannot press "Park". Similarly, when the vehicle has returned, the user cannot press "Park" or "Return".
- **Communication Service**: The app shall receive updates from the vehicle regarding the status of the vehicle via a background service running on the smartphone. This allowed for the app to send commands to and receive updates from the vehicle at any time.
- **Push Notifications**: Notifications shall be published on the android device whenever the status of the vehicle changes while the app is in the background. Specifically, the vehicle will have five possible statuses namely, "**Waiting**", "**Parking**", "**Parked**", "**Returning**", "**Returned**", where "Waiting" is the time while the vehicle is waiting for user input.

Figure 5 – Sample App Interface

4.2. Collaboration

- **Network of All Platforms**: All the mobile platforms shall be connected to each other on a single reliable network, which shall be scalable in order to simply incorporate more vehicles without failure.
- **Optimal Spot**: The vehicles on the network shall exchange information regarding availability of spots near them other vehicles on the network, so that the new vehicle can choose a spot that is closest to the exit without inspecting the entire parking lot itself.
- **Collaborative Path Planning**: The vehicles already on the network shall provide preemptive information to any new vehicle on the network regarding obstacles to avoid or preferred routes.

4.3. Navigation and Path Planning

- **Mapping**: The mobile platform shall be loaded at the entry queue with a map of the parking lot, which it will use to navigate the parking lot.

- **Localization**: The localization module shall use the pre-loaded map of the parking lot environment in order to accurately localize the mobile platform inside the parking lot.
- **Path Planning**: Once the optimal spot has been located, the mobile platforms shall plan the most efficient path to the spot based on the information exchanged with other vehicles.
- **Obstacle Avoidance**: The obstacle avoidance algorithm shall allow the mobile platform to avoid collisions with infrastructure and detect and avoid other vehicles, as well as any static obstacles in its path.

5. System-Level Trade Studies

5.1. Single Board Computer (SBC)

A powerful single board computer is a fundamental requirement to implement autonomy and create a collaborative network between various robots in a system. The processing unit needs to handle Ubuntu 14.04, ROS, Vision algorithms, and control an Arduino board, all running in parallel. To perform these tasks, it is important to choose a single board computer, which has capable enough hardware, in terms of processing power, RAM, etc., and supports other peripherals, such as a Kinect or Hokuyo Laser Scanner. For these reasons, performance, ease of integration, and support of peripherals have the highest weights in the trade study. Cost is an important factor because the system will require multiple units of these single board computers [3].

Table 5 – Single Board Computer Trade Study (Point Scale: 0-10)

	Weights	Raspberry Pi 2	Edison	BeagleBone Black Rev C	Odroid XU-4
Performance	25	6	10	4	10
Cost	10	10	4	7	6
Documentation	15	9	5	7	7
Ease of Integration	20	8	4	8	8
Peripheral	20	5	7	5	7
Availability	10	10	10	10	10
Total	100	7.5	6.9	6.4	8.2

5.2. Point-to-Point Communication

One of the core technologies that the project is intended to showcase is point-to-point communication between multiple vehicles. Additionally, the project will demonstrate how the vehicles can utilize shared data to collaborate in an effective manner. The network technology used to carry out this operation will have a huge impact on the implementation technique and overall performance of the system. Eliminating a central server is a non-negotiable requirement of the user, which is why cloud computing and wireless distributed computing are the two point-to-point communication options. Since this network will be supporting collaboration between real time systems, low latency is a major requirement. Owing to the short timeline of the project, ease of implementation will play a critical role in getting the system up and running within the given timeframe [4] [5].

Table 6 – Point-to-Point Communication Trade Study (Point Scale: 0-10)

	Weights	Cloud Computing	Wireless Distributed Computing
Range	15	10	7
Reliability	15	10	8
Bandwidth	5	10	8
Latency	25	6	9
Cost	5	8	8
Ease of Implementation	25	7	9
Scalability	10	8	6
Total	100	8.0	8.2

5.3. Sensors

The mobile platform needs to actively sense the environment and avoid collisions with obstacles, infrastructure, and other vehicles. To satisfy the performance requirement of maintaining more than 2 ft. distance between moving vehicles and 1 ft. between the vehicle and static obstacles, the vehicles need sensors that can measure these distances with high accuracy and relay data back to the microcontroller at a high baud rate. As the vehicles will be using multiple sensors per mobile platform, cost is an important issue in the selection. Additionally, the output of these sensors need to be robust to noise and various other environmental factors [6].

Table 7 – Proximity Sensor Trade Study

	Weights	IR Sensor	Ultrasonic Sensor
Range	30	7	10
Signal/Noise	15	6	8
Cost	15	10	8
Accuracy	20	6	8
Ease of Interfacing	20	10	10
Total	100	7.7	9

6. Cyber-Physical Architecture

The cyber-physical architectures in Figure 6 – Cyber-Physical Architecture and Figure 7 – Cyber-Physical Architecture depict the flow of information within the system and the interaction between the different subsystems. Separate flows have been identified for data, communication signals, and control signals. The power supply provides two voltage levels, 5V and 12V, which are distributed to the entire system as needed. In order to avoid processing overload on one SBC, separate control boards will be used for sensors and actuators in addition to the central SBC. The mobile nodes communicate wirelessly and the network switch/router specifications will be decided in the future. The user interacts with the system via Bluetooth using a smartphone.

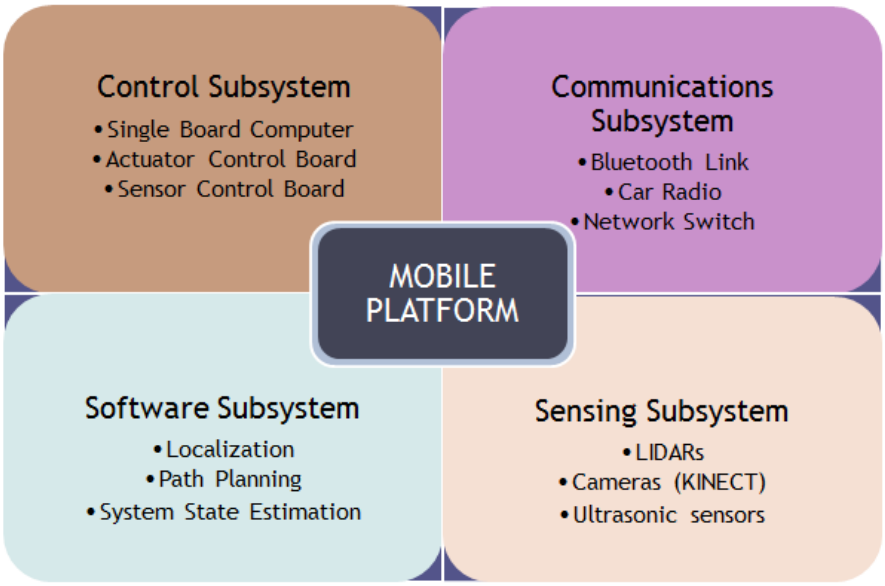


Figure 6 – Cyber-Physical Architecture

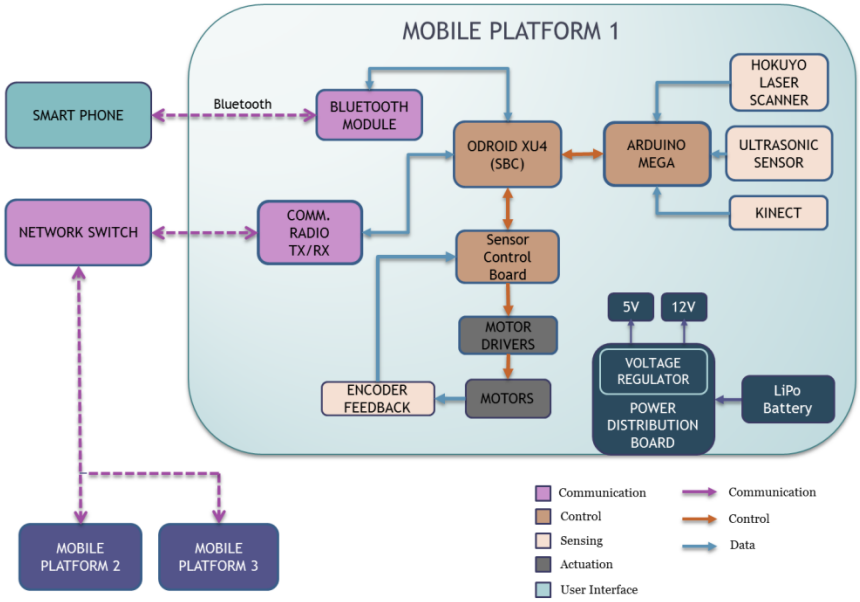


Figure 7 – Cyber-Physical Architecture

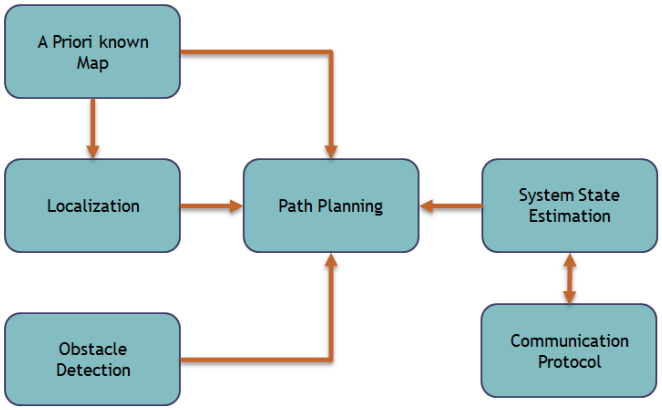


Figure 8 – Software Architecture

7. Subsystem Descriptions

7.1. Sensing

The sensing subsystem's main objective is to collect data to aid in localization, navigation, spot detection, and obstacle avoidance. The generation of the environment map is also a key aspect of the project, which the system will accomplish through LIDAR.

7.1.1. Laser Range Finder

Function: Map Generation, Localization

Laser Range Finding is a method by which topographical features are generated and used to map environments. The laser range finder usually consists of a laser and a scanner; the laser emits light at a certain frequency and the scanner detects this light reflected off an object. The phase difference between the reflected wave and the emitted wave leads to an estimation of the time difference of arrival. This can in turn be used to estimate the distance of the object being scanned. Laser technology is highly effective because the light wave is reflected from all solid surfaces with limited divergence, regardless of the nature of the obstacles [7].

This would aid in the estimation of the distance of objects such as walls, boxes, and vehicles in the environment. Ultimately, the Laser Range Finder will be used for generating the map of the parking lot. We further aim to use laser range finders to localize the vehicle with respect to its current surroundings. This is essential, as the robot needs to know its current position before planning and navigating along a path.

The Hokuyo laser range finder is selected at this point due to the availability of documentation, use cases, and its availability in the lab. [8]

Figure 9 – Hokuyo Laser Range Finder (URG-04LX-UG01)

7.1.2. Camera

Function: Spot Detection, Odometry, Obstacle Avoidance

We plan to use the Kinect to acquire images and generate point clouds. The device features an "RGB camera, depth sensor and multi-array microphone running proprietary software", which provide full-body 3D motion capture, facial recognition, and voice recognition capabilities. The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. The sensing range of the depth sensor is adjustable, and Kinect software is capable of automatically calibrating the sensor based on the physical environment [9].

The camera will primarily be used to detect the spot in which a platform will park. By combining the data from ultrasonic sensors and vision system, the exact location of the obstacle can be calculated. This will augment the path planning algorithm's obstacle avoidance capability.

Vision-based odometry using monocular or stereo vision cameras will help in localization and avoiding other moving vehicles in the lot. The data from encoders when coupled with visual feedback from the sensors will help in generating a reliable estimate of the robot's position.

Figure 10 – Microsoft Kinect v1

7.2. Control Subsystem

7.2.1. Single Board Computer

The central processing unit of the system is the single board computer, which is the ODROID XU4 for this project. The other control boards for sensors, actuators and communication hardware will be interfaced with the SBC, where the path planning and localization algorithms run.

7.2.2. Sensor and Actuator Control Boards

To avoid processing overload on the SBC, as well as to increase system reliability, two separate control boards will be used to interface the sensors and the actuators. The sensor control board will acquire sensor information and process it into a suitable format that can be used by the SBC to make path planning and localization decisions.

The actuator control board runs the motors of the mobile platform and processes the information provided by the encoders for odometry. The Arduino Mega 2560 will be used to do this, as it is highly suitable for interfacing multiple sensors and motors and comes with the ease of implementation standard to Arduino.

The tradeoff here is the use of multiple controllers for different interfaces. This increases cost and complexity but reduces the processing overload on the central processor.

7.3. Communications Subsystem

The design of the communication subsystem is a critical aspect of the project as it is required for the communication subsystem to be efficient, robust, reliable, and scalable. The challenge lies in designing a system that will house multiple communication channels without any intermittent interference and communication range reductions, i.e., the platforms will be able to communicate with each other throughout the parking lot without any loss in link.

We therefore intend to use two communication links per mobile platform

7.3.1. Bluetooth Link

A Bluetooth link will establish communication between the smartphone and the communication system in the mobile platform. The user will send park and return commands via this link, as well as receive status information from the vehicle.

The link between the smartphone app and the platform was chosen as a Bluetooth link due to ease of implementation on a smartphone interface, which reduces complexity. For the scope of this project, there are limited range requirements of this link, so the range of a Bluetooth link is not a problem.

7.3.2. Platform-To-Platform (P2P) Communication Link:

The P2P communication link will ensure that each vehicle is able to share data with the others. The data being shared will initially be limited to the availability of spots and the most optimal spot for the next vehicle in the queue. This information is essential for ensuring that vehicles park in a collaborative manner. By sharing this information amongst vehicles, the time taken for a vehicle to identify and park in an optimal spot is significantly reduced when compared to a system without collaboration.

Another key aspect of the system is to eliminate the need of a central server. Because this will have bandwidth limitations, the size of data being shared will be kept as minimal as possible. For example, the information regarding the state of the parking lot will be a Boolean matrix instead of a full-blown map.

The following approaches are being considered for implementing P2P communication:

- **The XBee Mesh Protocol**

  Shown in Figure 11 – Mesh Protocol Implementation for XBee Pro S2B, this is an apt fit for this application and eliminates the need for a central server [10].

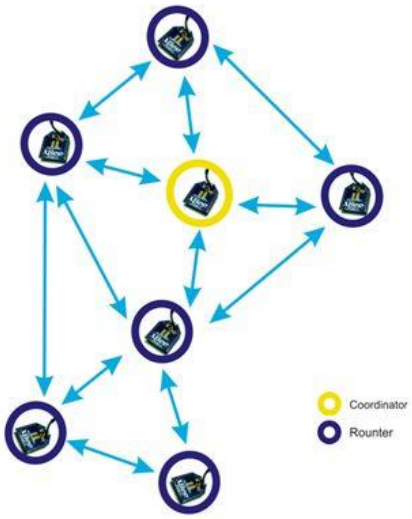


Figure 11 – Mesh Protocol Implementation for XBee Pro S2B

The main challenge with this implementation lies with assigning the first time vehicle that enters the lot. The vehicle needs to identify that it is the first to enter the lot and assign itself as the coordinator. In the event that the coordinator leaves the parking lot, the system will assign a new vehicle to be the coordinator.

- **Network Switch**

  A network switch ensures that each SBC is able to leverage the available Wi-Fi capabilities to connect to a single network. This method eliminates the need of a central server which would otherwise run all computations. The computations run individually on each SBC which are linked via a switch.

  This method, though it requires the need for a small change to infrastructure (by adding a switch), is still a better alternative than having a central server for communication and computational purposes.

  The network switch or hub is a networking device that receives, processes, and forwards data only to the devices that need to receive it, rather than broadcasting the same data out of each of its ports.

  This method ensures ease of implementation as it creates seamless communication between SBCs via the ROS interface.

7.4. Software Subsystem

7.4.1. Localization

A LIDAR-generated map will be pre-loaded on the robot to aid in localization. LIDAR will be used for odometry along with the camera. Region Growing, feature point tracking across closely spaced frames, or motion modelling are a few of the methods for visual odometry along with the point cloud from LIDAR.

Particle filter-based Monte Carlo localization is an effective method for determining the robot's pose and is based on the concept of particle filters. These filters work on the principle of Bayesian probability and use randomly generated particles to estimate posterior belief.

Possible ROS packages that can be used for this purpose are robot_localization package, which uses sensor fusion for robot localization, or the more general amcl (Adaptive Monte Carlo localization) package. The robot_localization package, shown in Figure 12 – robot_localization, can provide continuous localization utilizing multiple sources of odometry information, which would be the camera and LIDAR for this project.

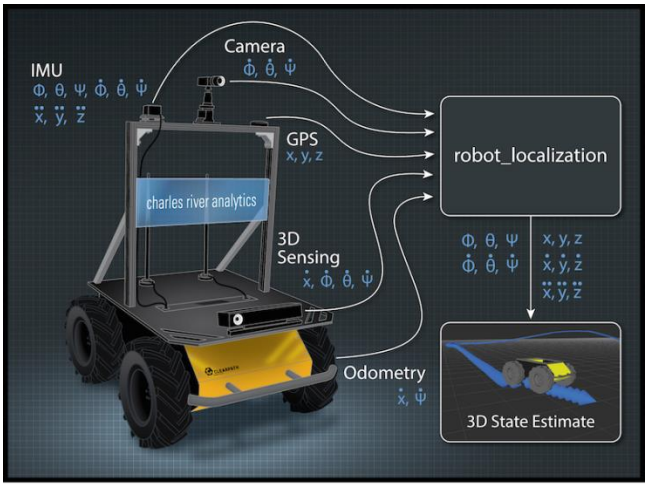


Figure 12 – robot_localization

7.4.2. Path Planning

At each point during the locomotion of the vehicle in the parking lot, planning an optimal path requires a map of the environment and the robot to be both aware of its location with respect to the map and capable of avoiding obstacles on its way. Obstacle avoidance at the low level can be performed by ultrasonic sensors, which will be used for emergency stops in critical cases. Either vision or LIDAR can be used for obstacle detection. The camera will also be used for detecting empty spots in the way.

In case of a disruption in the planned path, a new path will be computed using information from the surroundings. The optimal path will be one with minimal time to the destination that also takes into possible hindrances.

7.4.3. System State Estimation

Information about position, surrounding obstacles, available spots, and other relevant data shall be shared by each vehicle in the parking lot. This information will be available to all the other vehicles on the network. This information will be stored to aid the process of planning.

This will be done in one of two ways, depending on whether the system utilizes cloud computing or distributed computing.

As suggested by the trade study, the choice between distributed computing and cloud computing will be one of the defining factors in the conceptual design of the system. Distributed computing allows for ease of implementation and lower latency, but the topology and the overall structure of the system are not known beforehand and the information is dynamic. It revolves around providing collaborative resource sharing by connecting users and resources.

On the other hand, cloud computing offers greater reliability, range, and scalability but is more difficult to implement along, and creates a single point of failure for the entire system. It focuses on delivering services and applications in a demanding environment with targeted goals of achieving increased transparency, security, monitoring, and management.

A careful consideration of system performance and functionality is needed to decide on one of these systems. It is also a possibility to explore and implement some combination of these two methods to achieve desired results.

7.5. Power Subsystem

The power subsystem consists of a Lithium Polymer battery connected to two voltage regulation units. The first regulator unit outputs a voltage of 5V to be used by the on-board electronics. The second regulator outputs a voltage required for the actuators to operate, usually 12V or higher. The capacity of the battery shall be estimated after the arrival of the mobile platform and the study of its actuator system.

8. Project Management

8.1. Work Plan and Tasks

The work breakdown structure depicts the subsystems that need to be developed, the tasks involved in their accomplishment, and system-level and sub-system-level testing that needs to be conducted. Each subsystem highlighted in the cyber-physical architecture has certain key tasks that need to be accomplished and modules that need to be developed. Further, it is imperative to conduct sub-system testing and validation experiments to ensure seamless integration.

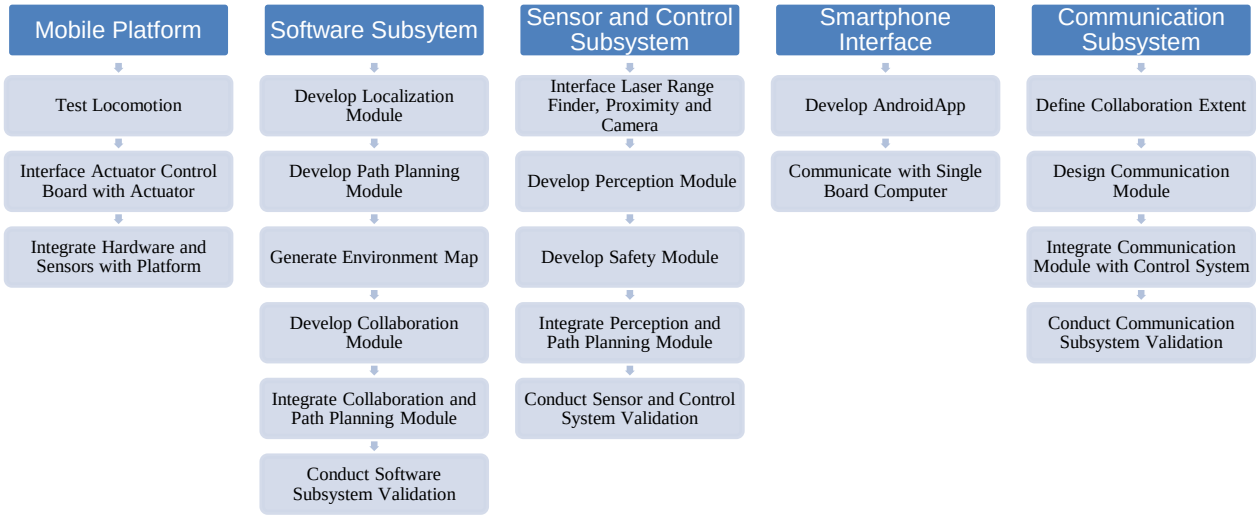


Figure 13 – Work Breakdown Structure

8.2. Schedule and Key Milestones

Table 8 – Schedule and Key Milestones

ID	Milestones	Start Date	End Date	Length (Days)	Predecessors
1	Obstacle Detection	14-Oct	4-Nov	21	
2	Literature survey	16-Oct	22-Oct	6	
3	Selection and procurement	22-Oct	24-Oct	2	2
4	Obstacle definition	20-Oct	22-Oct	2	3
5	Obstacle creation	3-Nov	4-Nov	1	4
6	Algorithm design				3
7	Algorithm formulation				6
8	Algorithm implementation				7
9	Testing				8
10	Car network setup	14-Oct	3-Nov	20	
11	Literature survey	16-Oct	22-Oct	6	
12	Selection and procurement	28-Oct	4-Nov	7	11
13	Hardware interface - network w/other routers	9-Nov			12
14	Hardware interface - SBC				12
15	Testing				13, 14
16	Mobile platform				
17	Trade study	10-Oct	13-Oct	3	

ID	Milestones	Start Date	End Date	Length (Days)	Predecessors
18	Selection and procurement	13-Oct	19-Oct	6	17
19	Actuator control board integrated	20-Oct	21-Oct	1	18
20	Actuator control board testing	21-Oct	22-Oct	1	19
21	Android App	15-Oct	11-Nov	27	
22	Research software and install	15-Oct	19-Oct	4	
23	Research Bluetooth capabilities	15-Oct	22-Oct	7	22
24	Create GUI sketch	15-Oct	22-Oct	7	22
25	Create GUI	23-Oct	30-Oct	7	24
26	Test GUI	1-Nov	11-Nov	10	25
27	Test Bluetooth interface with SBC	1-Nov	11-Nov	10	23
28	Progress Review 1	22-Oct			1, 10, 16, 21
29	Integration of actuator control board and single board computer	24-Oct	11-Nov	18	
30	Progress Review 2	29-Oct			28, 29
31	Collaboration between platforms setup	1-Nov	19-Nov	18	
32	Vision Software for obstacle detection developed	7-Nov	19-Nov	12	
33	Sensor Control Board tested	11-Nov	12-Nov	1	
34	Progress Review 3	12-Nov			
35	Integration of vision subsystem; Generation of map	20-Nov	17-Dec	27	
36	Progress Review 4	24-Nov			
37	Fall Validation Experiment and Demonstration	30-Nov			
38	Dedicated Verification, Integration Testing, and Verification	1-Dec	18-Dec	17	
39	Progress Review 5	3-Dec			
40	Progress Review 6	10-Dec			
41	Critical Design Review	15-Dec			
42	Integration of subsystems for all platforms and test rig development	10-Jan	31-Jan	21	
43	Localization, Navigation and Path planning Modules integrated	10-Jan	18-Feb	39	
44	Communication and Collaboration System Development and Testing	20-Feb	31-Mar	40	
45	Testing and Validation	1-Mar	2-May	62	

8.3. System Validation Experiments

The system validation experiments are used to ensure that all system-level requirements are met. Below, experiments are divided by the component that they will be validating, and then further divided into the semester they will be tested in. Additionally, there are outlines of demos that will be completed at the end of the fall and spring semesters.

8.3.1. Sensor

- **Ultrasonic**
  - **Fall Test:** Place pre-defined obstacles at different orientations, distances, and angles with respect to the sensor and log the readings. Confirm that the readings match the correct orientations, distances, and angles.
    Requirement Validated: MF.9
- **LIDAR**
  - **Fall Test:** Map a known environment and note the discrepancies between the actual and logged data. Note the resolution of the map.
    Requirement Validated: MF.8

8.3.2. Communication

- **Bluetooth**
  - **Fall Test:** Ensure Bluetooth device is recognized by the smartphone. Transmit serial data between phone and the microcontroller and confirm that it receives the transmissions correctly.
    Requirement(s) Validated: MF.1
- **Wireless Network**
  - **Fall Test:** Transmit a list of 10 elements between multiple ROS-masters over a Wi-Fi network. Each ROS master will change a different element in the list. Each ROS will receive the updates and will have identical lists.
    Requirement(s) Validated: MF.2, MN.1, MN.5, DN.1
- **Fall Test 1:** Send vehicle to parking spot. Turn off Bluetooth and Wi-Fi. Vehicle attempts to reconnect while maintaining course. After 25 seconds, turn on Bluetooth and Wi-Fi. Vehicle reconnects successfully and maintains course.
  Requirement(s) Validated: MN.2
- **Fall Test 2:** Send vehicle to parking spot. Turn off Bluetooth and Wi-Fi. Vehicle attempts to reconnect while maintaining course. After 30 seconds of no connection, vehicle changes course to exit queue. Vehicle continues to attempt to connect at longer intervals.
  Requirement(s) Validated: MN.2
- **Parking Spot Matrix**
  - **Fall Test:** Transfer a multidimensional array representing the parking spots in the lot between the single board computers and populate it with relevant data. Make some changes to the matrix and log how that affects the contents of the matrix. Ensure that the changes made on one board are carried over to the other boards.
    Requirement(s) Validated: MF.2, DF.1

8.3.3. Android App Interface

- **Phone to Car**

- **Fall Test:** Transmit a command, either "Park" or "Return" from the Android App to the microcontroller by pressing the appropriate button from the app. Ensure that the command is received at the other end and that the appropriate vehicle responds in the appropriate manner.
  Requirement(s) Validated: MF.1

- **Car to Phone – Vehicle Status and Notification**
  - **Spring Test:** Press "Park" from the Android App. The vehicle moves towards a parking spot and the app displays the status "Parking". When the vehicle is parked in a spot, the app displays the status "Parked". Press "Return" from the Android App. The vehicle exits the parking spot and moves towards the exit and the app displays the status "Returning". When the vehicle stops at the exit, the app displays the status "Returned" and sends a notification to the user.
    Requirement(s) Validated: MN.1
- **Android App Prevents User Error**
  - **Spring Test:** The user can only press "Park" when the vehicle status displays "Waiting" on the app. The user can only press "Return" when the vehicle status displays "Parked" on the app.
    Requirement(s) Validated: MN.4, MN.1

8.3.4. Locomotion

- **Acceleration Control**
  - **Fall Test:** Make the vehicle move slowly, normally, and quickly. Note the time taken and distance travelled. Make the vehicle come to a complete stop quickly and slowly. Note the nature between speed control commands and how they affect actual speed of the system.
    Requirement(s) Validated: DF.4
- **Deviation between instructed and actual motion**
  - **Fall Test:** Give a combination of commands to the platform and note the deviation from the desired position. Iterate multiple times and see if there is some predictable trend.
    Requirement(s) Validated: MF.3

8.3.5. Parking Spot – Entering and Exiting

- **Fall Test 1**: Send the vehicle to a spot with both adjacent spots occupied. The vehicle backs into the space within two tries. When the vehicle is parked, it is 100% within the parking spot boundaries and within 35º of parallel to both the neighboring vehicles. Send the vehicle to the exit. The vehicle exits the spot within two attempts without coming into contact with nearby vehicles or the infrastructure.
- **Fall Test 2**: Send the vehicle to a spot with one adjacent spot occupied. The vehicle backs into the space within two tries. When the vehicle is parked, it is 100% within the parking spot boundaries and within 35º of parallel to the neighboring vehicle on one side and the parking spot line on the other side. Send the vehicle to the exit. The vehicle exits the spot within two attempts without coming into contact with nearby vehicles or the infrastructure.
- **Fall Test 3**: Send the vehicle to a spot with both adjacent spots unoccupied. The vehicle backs into the space within two tries. When the vehicle is parked, it is 100% within the

parking spot boundaries and within 35º of parallel to the parking spot boundaries. Send the vehicle to the exit. The vehicle exits the spot within two attempts without coming into contact with nearby vehicles or the infrastructure.
Requirement(s) Validated: MF.6, MF.7, MN.3, MN.4, DN.3

8.3.6. Power Subsystem

- **Connections to PCB are secure**
  - **Fall Test:** Use a multi-meter to check all the connections.
    Requirement(s) Validated: MN.7
- **Power is constant with no fluctuation**
  - **Fall Test:** Note the nature of voltage being supplied through an oscilloscope. Ensure there are no fluctuations.
    Requirement(s) Validated: MN.7

8.3.7. Mapping

- **Fall Place markers at specific intervals**
  - **Test:** Place markers at predefined locations and map the area. Compare their position in the map to their actual physical location and ensure it is correct.
    Requirement(s) Validated: MF.8
- **Ensure map can be loaded into vehicles**
  - **Fall Test:** Upload the map generated to the SBC and visualize with RViz.
    Requirement(s) Validated: MF.8

8.3.8. Localization

- **Fall Test 1:** Have the vehicle circle the parking lot. Compare actual location to computed location every 30 seconds to ensure there are minimal errors. Continue to have the vehicle circle for approximately 10 minutes to ensure there are no compounded errors.
  Requirement(s) Validated: MF.3, DN.2
- **Fall Test 2:** Place the vehicle at different locations and note the localization readings. Make the vehicle move to different locations and note the localization readings.
  Requirement(s) Validated: MF.3, DN.2

8.3.9. Path Planning – From Entrance to Parking Spot to Exit

- **Spring Test 1:** Place the vehicle at the entrance of the parking lot. Have the vehicle plan a path to a spot and ensure that this is the most optimal path.
  Requirement(s) Validated: MF.4, MF.5, DF.2, DF.3, MN.4, DN.2
- **Spring Test 2:** Place the vehicle at the entrance of the parking lot. Have the vehicle plan a path to a spot. Then, introduce an obstacle in the path and see how the path gets altered. Ensure that the new path is the most optimal
  Requirement(s) Validated: MF.4, MF.5, DF.2, DF.3, MN.4, DN.2

8.3.10. Obstacle Detection

- **Send and Receive Movement Information**
  - **Fall Test 1:** Send a Return command to a vehicle. Ensure that the vehicle publishes a message to other vehicles on the network that it is exiting the parking

spot. Nearby vehicles that are parking will yield right of way to the exiting vehicle, allowing it enough space to exit the spot safely and efficiently.
    - **Fall Test 2:** When a vehicle is entering a parking spot, nearby vehicles maintain a safe distance to avoid collision and give the parking vehicle adequate space to park safely and efficiently.
      Requirement(s) Validated: MF.2
- **Maintain Safe Driving Distance**
    - **Spring Test**: Send two vehicles to park in neighboring spots in quick succession. The first vehicle will maintain a constant speed. The second vehicle will maintain a safe distance between itself and the first vehicle. When parking, the second vehicle will stop at a safe distance so as not to interfere with the first vehicle parking.
      Requirement(s) Validated: DF.4
- **Avoid Infrastructure**
    - **Fall Test**: Have the vehicle circle the parking lot. The vehicle maintains a safe distance from the infrastructure of the parking lot and parked vehicles. Place an object in the vehicles path. The vehicle stops within 3 seconds of detecting the obstacle.
      Requirement(s) Validated: MF.8, MF.9, , MF.10, DF.4

8.3.11. Demonstrations

- **Fall Demo 1**: The user will send the "Park" command to their vehicle using the Android app. Upon receiving the command, a vehicle will start locomotion. When the user sends the "Return" command via the Android app, the vehicle will return to its original position.
- **Fall Demo 2**: The user will send the "Park" command to their vehicle using the Android app. Upon receiving the command, the vehicle will navigate from the entrance of the parking lot to another spot, based on the direction set by another vehicle. While navigating to the spot, the vehicle will encounter an obstacle and stop within a safe distance of the obstacle.
- **Spring Demo**: The vehicles receive park commands in quick succession. They will enter the parking lot and collaborate with other vehicles to choose the optimal parking spots and the optimal paths to them. When one vehicle encounters an obstacle, it will plan a path around it. A notification will be sent to each user when their vehicle is parked. When each user sends the command to return, the robot will exit the parking spot and navigate towards the exit spot. When it reaches the exit, it will send the user a notification stating that it is at the exit.

8.4. Team Member Responsibilities

Team Daedalus is composed of five members with a wide range of abilities. However, all team members are all interested in expanding their skills. Table 9 – Team Member Responsibilities depicts each team member, their area of expertise, and the area for which they will be responsible. This way, each member will be able to support each other while expanding skillsets.

Table 9 – Team Member Responsibilities

Team Member	Area of Expertise	Areas of Responsibility
Dorothy Kirlew	Software	Sensors, Control Subsystem, and Android App Development
Mohak Bhardwaj	Vision and Navigation	Collaboration and Android App Development
Pranav Maheshwari	Collaboration and Navigation	Vision
Richa Varma	Vision, Sensors, and Control Subsystem	Navigation
Shivam Gautam	Collaboration and Vision	Navigation and Path Planning Algorithm

8.5. Parts List and Budget

Table 10 – Parts List

Part	Part Number	Quantity	Cost/Unit	Total Cost
Hokuyo Laser Scanner	Hokuyo URG-04LX-UG01	3	$1,140	
ODROID-XU4	N/A	3	$74	$222
ArduRover				
Arduino Mega 2560 Rev 3	A000067	3	$39.21	$117.63
LiPo Battery	Turnigy 9SIA62H2KZ3269	3	$11.79	$35.37
LiPoBattery Charger	imaxRc B3 Pro Compact charger	1	$11.12	$11.12
AA Rechargeable Batteries	N/A	12		$15
AA Battery Charger	Duracell CEF14	1	$19	$19
Ultrasonic Distance Sensor	Parallax 70372368	9	$29.99	$269.91
MS Kinect Sensor	N/A	3	$149.99	$449.97
Miscellaneous (Mounts and electronics)	N/A	N/A	N/A	
XBee Pro S2B	N/A	3	$35	$105
Total Budget:				$1244.97

8.6. Risk Management

The risks for the project are outline in Table 11 – Risk Analysis. The categories are as follows:

- **ID**: Number used to reference risk
- **Description**: Brief description of the risk
- **Responsible Party**: Indicates who is in charge of handling the risk.
- **Risk Analysis**: The numbers 1-4 representing:
    - **1**: Risk is very likely with big impact
    - **2**: Risk is unlikely, but with big impact

    - **3**: Risk is very likely, but with little impact
    - **4**: Risk is unlikely with little impact
- **Area of Impact**: Time, cost, quality, other
- **Handling Strategy**: How the risk will be handled?
- **Status**: What is currently being done to handle the risk?

Table 11 – Risk Analysis

ID	Description	Responsible Party	Risk Analysis	Area of Impact	Handling Strategy	Status
1	Mobile platforms may not be provided by sponsor	Sponsor	2	Time, Cost	Check Status with sponsor regularly, Set final deadline for platforms to arrive	In Progress
2	Mobile Platform may not comply with requirement	Team and Sponsor	2	Time, Cost	Share requirements with Sponsor	In Progress
3	Smartphone Interface may be unsuitable	Team	4	Quality	Extensive Testing	Open
4	Incompatibility of Sub-systems while systems integration	Team	2	Quality, Functionality	Literature Survey, Use Case studies of equipment before purchase	In Progress
5	Accomplishing all desired and mandatory requirements within 9 months	Team, Sponsor, and Advisor	1	Time, Functionality	Progress Reviews with Advisor, Sponsor	In Progress
6	Parking lot/ Obstacles may not be an accurate model of the real world	Team	3	Quality	Create “proof of concept” test parking lot which is as close to the real world as feasible	Open

9. Bibliography

1. Stark, John. "Parking Lots." University at Albany, 23 Apr. 2012. Web. 25 Sept. 2015. <http://www.albany.edu/ihi/files/Parking_Lots_Where_Motorists_Become_Pedestrians.pdf>.
2. Fayard, GM. "Work-related Fatal Injuries in Parking Lots, 1993-2002." *National Center for Biotechnology Information*. U.S. National Library of Medicine, 10 Jan. 2008. Web. 25 Sept. 2015. <http://www.ncbi.nlm.nih.gov/pubmed/18325411>.
3. "Comparison of Single-board Computers." *Wikipedia*. Wikimedia Foundation, 1 Oct. 2015. Web. 25 Sept. 2015. <https://en.wikipedia.org/wiki/Comparison_of_single-board_computers>.
4. "Cloud Robotics." Wikipedia. Wikimedia Foundation, 23 Sept. 2015. Web. 25 Sept. 2015. <https://en.wikipedia.org/wiki/Cloud_robotics>.
5. "Distributed Computing." Wikipedia. Wikimedia Foundation, 28 Aug. 2015. Web. 25 Sept. 2015. <https://en.wikipedia.org/wiki/Distributed_computing>.

6. "Infrared vs. Ultrasonic - What You Should Know." Infrared vs. Ultrasonic - What You Should Know | Member Robot Tutorials. Society of Robots, 27 Jan. 2008. Web. 25 Sept. 2015. <https://www.societyofrobots.com/member_tutorials/node/71>.
7. "What Is LIDAR?" *National Oceanic and Atmospheric Administration*. NOAA. Web. 25 Sept. 2015. <http://oceanservice.noaa.gov/facts/lidar.html>.
8. "Various Techniques for Positioning a Mobile Robot in a Space." *Localize with a Hokuyo Laser Range Finder*. Generation ROBOTS. Web. 25 Sept. 2015. <http://www.generationrobots.com/en/content/52-localize-with-a-hokuyo-laser-range-finder>.
9. "Kinect." *Wikipedia*. Wikimedia Foundation, 18 Sept. 2015. Web. 25 Sept. 2015.<https://en.wikipedia.org/wiki/Kinect>.
10. "XBee." *XBee*. XBee. Web. 25 Sept. 2015. <http://xbee.wikispaces.com/>.