

**Self-Powered Automated Refuse Cart**

**(SPARC)**

|  |  |
| --- | --- |
| Group 12 | |
| Drayton, Alexis | Computer Engineering |
| McGarvey, Sean | Electrical Engineering |
| Schmitz, Wyatt | Computer Engineering |
| Vento, Chris | Computer Engineering |

**Funded by:**

****

[www.theselahgroup.com](http://www.theselahgroup.com)

Table of Contents

[List of Figures 4](#_Toc512578944)

[List of tables 5](#_Toc512578945)

[1.0 Executive Summary 6](#_Toc512578946)

[2.0 Project Description 7](#_Toc512578947)

[2.1 Project Motivation 10](#_Toc512578948)

[2.2 Objectives 10](#_Toc512578949)

[2.3 Requirements and Specifications 11](#_Toc512578950)

[2.4 House of Quality 12](#_Toc512578951)

[3.0 Design Constraints and Standards 14](#_Toc512578952)

[3.1 Standards Used 14](#_Toc512578953)

[3.2 Design Constraints 14](#_Toc512578954)

[3.2.1 Economic Constraints 15](#_Toc512578955)

[3.2.2 Environmental Constraints 15](#_Toc512578956)

[3.2.3 Sustainability Constraints 15](#_Toc512578957)

[3.2.4 Manufacturability Constraints 16](#_Toc512578958)

[3.2.6 Ethical Constraints 16](#_Toc512578959)

[3.2.7 Health Constraints 16](#_Toc512578960)

[3.2.8 Safety Constraints 16](#_Toc512578961)

[3.2.9 Social Constraints 17](#_Toc512578962)

[3.2.10 Political Constraints 17](#_Toc512578963)

[4.0 Relevant Project Research 17](#_Toc512578964)

[4.1 Similar Projects 17](#_Toc512578965)

[4.1.1 A.S.R (The Autonomous Sentry Robot) 18](#_Toc512578966)

[4.1.2 The Manscaper Autonomous Lawn Mower 18](#_Toc512578967)

[4.1.3 S.H.A.S Bot 19](#_Toc512578968)

[4.2 Pathfinding 20](#_Toc512578969)

[4.2.1 Shortest-Path Search Algorithms 20](#_Toc512578970)

[4.2.2 Collision Avoidance Technologies 22](#_Toc512578971)

[4.2.3 Absolute Positioning 23](#_Toc512578972)

[4.2.4 Relative Positioning 24](#_Toc512578973)

[4.2.5 Markov Decision Processes 25](#_Toc512578974)

[4.3 Computer Vision 26](#_Toc512578975)

[4.3.1 Image Recognition 26](#_Toc512578976)

[4.3.2 Edge Detection 27](#_Toc512578977)

[4.4 Hardware 28](#_Toc512578978)

[4.4.1 Microcontroller Options 28](#_Toc512578979)

[4.4.2 Serial Communication Protocols 29](#_Toc512578980)

[5.0 Hardware Design 30](#_Toc512578981)

[5.1 Hardware Overview 30](#_Toc512578982)

[5.2 Robotics Design 31](#_Toc512578983)

[5.2.1 Chassis Structure 31](#_Toc512578984)

[5.2.2 Motor and Gear Sizing 32](#_Toc512578985)

[Permission Pending from 2BFly 35](#_Toc512578986)

[5.3 Internal Electrical Design 35](#_Toc512578987)

[5.3.1 Battery Bank 36](#_Toc512578988)

[5.3.2 Microcontroller 40](#_Toc512578989)

[5.3.3 Voltage Regulators 42](#_Toc512578990)

[5.3.4 Motor Controllers 45](#_Toc512578991)

[5.3.5 Bluetooth Module 47](#_Toc512578992)

[5.3.6 GPS Module 47](#_Toc512578993)

[5.3.7 Pixy Camera 48](#_Toc512578994)

[5.3.8 RF Transmitters 51](#_Toc512578995)

[5.4 Home Base Design 53](#_Toc512578996)

[5.4.1 Charging Circuit 53](#_Toc512578997)

[5.4.2 RF Receivers 55](#_Toc512578998)

[5.4.3 Wireless Conduction Charging 57](#_Toc512578999)

[5.5 PCB Design 58](#_Toc512579000)

[6.0 Software Design 62](#_Toc512579001)

[6.1 Software Overview 62](#_Toc512579002)

[6.2 Microcontroller Programming 63](#_Toc512579003)

[6.2.1 Movement Control Functions 64](#_Toc512579004)

[6.2.2 Input Control 66](#_Toc512579005)

[6.2.3 Output Control 68](#_Toc512579006)

[6.3 The Mobile Application 77](#_Toc512579007)

[6.3.1 Mobile Application Diagrams 77](#_Toc512579008)

[6.3.2 Application to Microcontroller Interaction 80](#_Toc512579009)

[6.3.3 Mobile Application Design 81](#_Toc512579010)

[6.4 Pathing Algorithm 84](#_Toc512579011)

[6.4.1 Class Diagram 85](#_Toc512579012)

[6.4.2 Initial Path Priming & Graph Initialization 86](#_Toc512579013)

[6.4.3 Value Iteration 88](#_Toc512579014)

[6.4.4 Collision Avoidance Implementation 88](#_Toc512579015)

[7.0 Prototyping 89](#_Toc512579016)

[7.1 Hardware Prototyping 89](#_Toc512579017)

[7.2 Software Prototyping 91](#_Toc512579018)

[7.3 Bill of Materials 92](#_Toc512579019)

[8.0 Testing 97](#_Toc512579020)

[8.1 Hardware Tests 97](#_Toc512579021)

[8.1.1 Power Testing 98](#_Toc512579022)

[8.1.2 Microcontroller Testing 98](#_Toc512579023)

[8.1.3 Independent Module Testing 99](#_Toc512579024)

[8.1.4 Physical Movement Testing 99](#_Toc512579025)

[8.2 Software Tests 102](#_Toc512579026)

[8.2.1 Algorithm Simulation Testing 102](#_Toc512579027)

[8.2.2 Mobile Application Testing 105](#_Toc512579028)

[8.2.3 Processor Program Tests 108](#_Toc512579029)

[8.3 Integration Tests 109](#_Toc512579030)

[8.4 Testing Overview and Results 111](#_Toc512579031)

[9.0 Administrative Content 112](#_Toc512579032)

[9.1 Milestones 112](#_Toc512579033)

[9.2 Budgeting and Final Cost Analysis 114](#_Toc512579034)

[9.3 Project Group Roles 115](#_Toc512579035)

[10.0 Conclusions 116](#_Toc512579036)

[10.1 Future Revisions 116](#_Toc512579037)

[10.1.1 SPARC Weighing Scale 116](#_Toc512579038)

[10.1.2 Wireless Communication via WIFI 117](#_Toc512579039)

[10.1.3 GPS Tracking & Anti-theft 117](#_Toc512579040)

[10.1.4 Adverse Weather Exceptions 118](#_Toc512579041)

[Appendix A References 119](#_Toc512579042)

[Appendix B Permissions 120](#_Toc512579043)

# List of Figures

[Figure 1 Concept Block Diagram 9](#_Toc512579044)

[Figure 2 House of Quality Diagram 13](#_Toc512579045)

[Figure 3 Reflexive Algorithm Design Flowchart for SPARC 21](#_Toc512579046)

[Figure 4 Shape Recognition on Humans 27](#_Toc512579047)

[Figure 5 Edge Detection used on Coinage 28](#_Toc512579048)

[Figure 6 Concept Art 31](#_Toc512579049)

[Figure 7 Drive Motor Sizing Tool 33](#_Toc512579050)

[Figure 8 Brushed and Brushless DC motors 34](#_Toc512579051)

[Figure 9 Rotary Servo 35](#_Toc512579052)

[Figure 10 H-Bridge Representation 45](#_Toc512579053)

[Figure 11 Pixy CMUcam5 Board 48](#_Toc512579054)

[Figure 12 Pixy Color Code Detection 49](#_Toc512579055)

[Figure 13 Home Base PC Top 58](#_Toc512579056)

[Figure 14 Home Base PCB Bottom 59](#_Toc512579057)

[Figure 15 Main PCB Top Layer 60](#_Toc512579058)

[Figure 16 Main PCB Bottom Layer 61](#_Toc512579059)

[Figure 17 Software Use Case Diagram 62](#_Toc512579060)

[Figure 18 Movement Functions Flowchart 65](#_Toc512579061)

[Figure 19 Input Control Flowchart 67](#_Toc512579062)

[Figure 20 Output Control Flowchart 1 70](#_Toc512579063)

[Figure 21 Output Control Flowchart 2 71](#_Toc512579064)

[Figure 22 Output Control Flowchart 3 72](#_Toc512579065)

[Figure 23 Output Control Flowchart 4 73](#_Toc512579066)

[Figure 24 Output Control Flowchart 5 74](#_Toc512579067)

[Figure 25 Output Control Flowchart 6 75](#_Toc512579068)

[Figure 26 Output Control Flowchart 7 76](#_Toc512579069)

[Figure 27 Software Use Case Diagram 78](#_Toc512579070)

[Figure 28 Application Class Diagram 79](#_Toc512579071)

[Figure 29 SPARC Bluetooth State Diagram 80](#_Toc512579072)

[Figure 30 SPARC Mobile Application Splash Screen and Sign in Screens 82](#_Toc512579073)

[Figure 31 SPARC Mobile Application Sign Up and Device Control Screens 83](#_Toc512579074)

[Figure 32 Pathing Algorithm Class Diagram 85](#_Toc512579075)

[Figure 33 Initialization Flowchart 87](#_Toc512579076)

# List of tables

[Table 1 Requirements Specification 11](#_Toc512579332)

[Table 2 Standards Used 14](#_Toc512579333)

[Table 3 Prototype List of Materials 1 89](#_Toc512579334)

[Table 4 Prototype List of Materials 2 90](#_Toc512579335)

[Table 5 Bill of Materials 1.1 92](#_Toc512579336)

[Table 6 Bill of Materials 1.2 93](#_Toc512579337)

[Table 7 Bill of Materials 1.3 94](#_Toc512579338)

[Table 8 Bill of Materials 2 95](#_Toc512579339)

[Table 9 Bill of Materials 3.1 96](#_Toc512579340)

[Table 10 Bill of Materials 3.2 97](#_Toc512579341)

[Table 11 Hardware Test Case Results 111](#_Toc512579342)

[Table 12 Software and Integration Test Case Results an 112](#_Toc512579343)

[Table 13 Milestones SD1 113](#_Toc512579344)

[Table 14 Milestones SD2 114](#_Toc512579345)

[Table 15 Cost Analysis 115](#_Toc512579346)

[Table 16 Group Roles 115](#_Toc512579347)

# 1.0 Executive Summary

As the world’s technology progresses, the automation of mundane tasks is on the rise. Online shopping, food delivery services, self-checkout machines and self-driving cars are all examples of this progress and trash collection should not be an exception. In many domestic settings, the trash collection process includes having to move a large trash container to the curb side in front of your home for pick up at a specific time weekly. This task is simple and routine for a lot of people but again as we progress in technology, if the task is simple and easy why not made it automated and efficient.

To automate this process the Self-Powered Automated Refuse Cart (SPARC) will be designed as a rechargeable, motorized, load-bearing crawler that can mount a compatible garbage capable of automatically delivering the garbage bin and its contents to a predetermined location for it to be collected with accordance to a defined schedule. The process is automated by using computer vision and a path seeking algorithm to find a path through any obstacle it might encounter on its journey from the user’s garage to the curbside for pick up. A camera will be mounted on the device to give it the “sight” it needs for its collision avoidance capabilities. For charging the SPARC will have a “Home Base” which will wirelessly charge the SPARC when it returns from the curb. The SPARC will utilize GPS positioning technology to know the locations of the curb and the home base for return. The central control unit of the SPARC will be its microcontroller which will house its movement controls which will take input from its path finding algorithm with takes data from the camera. The movement functions can also take in input via Bluetooth from the user mobile application. Users will also be able to program their garbage collection schedule into the SPARC by using this companion smartphone app for the Android OS. Physically the SPARC will include waterproofing of its electrical components and will be able to withstand a garbage load up to 50 pounds and hold a charge capacity that will be enough to traverse an average driveway and make a return trip back to the home base for recharging and garbage reloading.

Ultimately, the SPARC is meant to facilitate the user’s day to day routine by eliminating the need to think about taking out the trash. The final design of the SPARC will be implemented in the most cost-effective way possible. This project will be funded by Selah Group Florida, Inc but kept on a tight budget. The SPARC’s end goal is to be a user-friendly device that will ease the user’s mind by eliminating the need for such a mundane task that will leave room for more important things.

# 2.0 Project Description

This section will provide an overview of the project, giving details of the project motivation, project objectives, requirements specification and an outline of our products wants versus the capabilities of the project.

There will be three units that comprise the SPARC system; the SPARC unit itself, and SPARC’s “Home-Base” platform, and an RFID “stake” to mark the approximate designation.  The SPARC unit will require charging through a NEMA 5-15 standard 120V wall outlet, and will ideally be charged later automatically and autonomously through solar panels built into Home-Base. When powered, the user can pair his/her smartphone to SPARC through Bluetooth. Then the user should select which days of the week their local garbage collection takes place. The SPARC will be designed to accommodate the square garbage bins utilized by semi-automated, side-loading garbage collection vehicles. This garbage can will be firmly attached to the SPARC unit, primarily via a clamp to the wheel axle of the garbage can in the rear, and grips in the front corners. The weight of the SPARC unit needs to be heavy enough to offset weight from the trash can to prevent toppling while moving.

Once a location for the Home Base has been chosen, the platform has been placed on the ground, and the SPARC unit has been initially charged, the user’s ideal path has been initialized, and programmed with the user’s desired delivery schedule, the SPARC will wait in standby mode until a pre-programmed time on the morning of the collection day. At that time, the device will power on and check its location compared to the first node in its path. The SPARC will pilot itself using an A\* pathing algorithm in order to reach the destination.  SPARC will utilize collision avoidance technology to prevent collisions with parked cars or other obstacles on its trip, and will automatically find new paths and will greedily choose one which puts it in a greater proximity to its destination. Figure 7 demonstrates the logic flow for the search Algorithm as a determines a path to its destination. Once the cart has arrived at the GPS coordinates of the drop-off point, the SPARC will then utilize its visual sensors to determine where the curb is in relation to the SPARC, at which point the SPARC will use this information to rotate itself, and move up against the curb. Then it shall switch to a low power state, and wait for garbage collection.

After garbage collection, during a user programmed time, the SPARC unit will power on again, and deploy the same pathing algorithm to return to the Home-Base safely. Once at the Home-Base, SPARC will deploy a “charging arm” on the underside of the unit, which will make contact with a wireless charging pad, to facilitate recharging of the battery built into the SPARC unit. At any given time, the user should be able to use the smartphone app to assume direct control of the SPARC unit, which is connected to the microcontroller via Bluetooth, using a GUI control system to pilot the SPARC at will.  The app will also give the user an interface for controlling the time of day the SPARC will deliver its payload and when it will return to its Home-Base based on their trash collection schedule. Figure 1 below shows the top level break down of all components of the SPARC and the division of labor amongst the group.

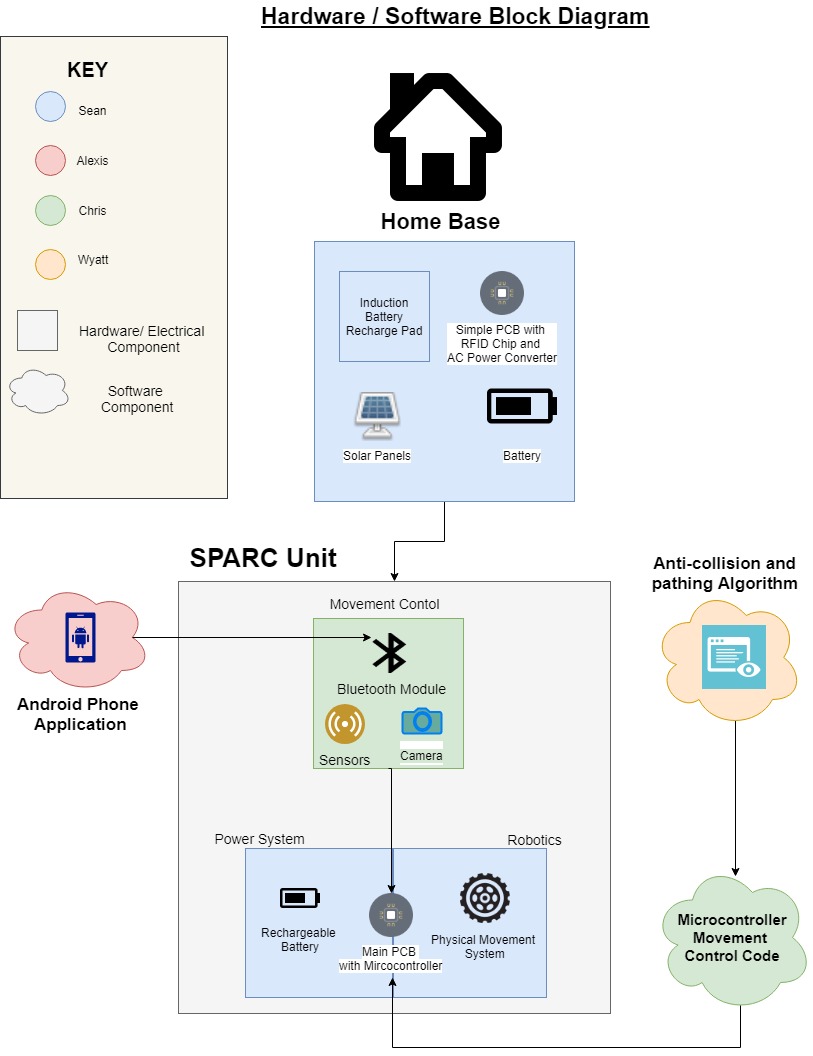
****

Figure 1 Concept Block Diagram

Figure 1 demonstrates the concept flow of both the home base and the main SPARC unit with both software and hardware components. The home base will be constructed as a purely hardware powering service for the SPARC unit, with a solar panel, battery, and a wireless induction charging pad. The home base will be connected using a simple PCB equipped with an RF chip for SPARC positioning on the return path. The SPARC unit has three major connecting components; the movement control, power system and robotics. The movement control components include the Bluetooth and GPS modules, which will connect to the main PCB equipped with a microcontroller. The main PCB also controls the physical movement system which will consist of the motors, chassis and treads. The software components connect include the mobile app, which controls the SPARC unit via the Bluetooth module, and the pathing algorithm and movement code which is loaded into the microcontroller directly.

## 

## 2.1 Project Motivation

Trash collection is a standard ritual in any domestic setting, and for many homes it involves a weekly schedule of days for trash collection by a municipal vehicle along the street. Despite its banality, the failure to remember to take out the trash can result in a number of problems; rotting food and other waste can draw the attention of unwanted pests, and the odor can make human living quarters quite uncomfortable.  The curbside trash collection system also poses problems of accessibility for those who cannot maneuver a heavy trash can to the street, as well as logistical problems for those who have abnormal working schedules. Automating this domestic chore prevents the chances of oversight, and answers accessibility and logistical concerns for those who struggle or are unable to deliver garbage bins to the curb.

Other than wanting to create a product that can be simply used in most domestic settings we wanted to create a project that combines the skill that have been cultivated in the Electrical and Computer Engineering major curriculums. Appealing to both the software and hardware skills of both majors. The electrical and computer hardware skills from the electrical engineering student together with the software designing and microcontroller programming skills from the computer engineering students will be used to full capacity to achieve completion of this project.

## 

## 2.2 Objectives

In creating this project, our main goal is to ease the users mind when it comes to remembering to take out the trash or in some cases help those who are not able to take out the trash on their own to aid in their individual independence. To achieve our main goal, we have some desired technical objectives for the SPARC:

* Autonomous Control
* Remote Control
* Device Scheduling
* Path Mapping and Localization
* Object and Collision Avoidance
* Wireless Charging

Ultimately the SPARC will be a fully functional transport cart that will map its local area using its camera to create a path to a desired location and autonomously move there at the desired time. The SPARC will be able to be remotely controlled as well via the mobile application. The device will also be scheduled on this mobile application. The path algorithm the SPARC uses to move to and from the desired location and the Home Base will have object and collision avoidance capabilities using computer vision. The device will be able to localize itself using GPS as well. Once the SPARC returns to the Home Base it will wirelessly charge in a low powered state until triggered by its set schedule.

## 2.3 Requirements and Specifications

Together with the projects technical objectives, the project’s basic requirements and specifications are shown below (Table 1):

Table 1 Requirements Specification

|  |  |
| --- | --- |
| **Requirement #** | **Requirement Specification** |
| 1.1 | The device will be a fully autonomous vehicle. |
| 1.2 | The device will be fully capable of carrying up to 50lbs. |
| 1.3 | The device will be fully rechargeable. |
| 1.4 | The device will have a mobile application that for the user to interface with the device. |
| 1.5 | The device will connect to the mobile application via Bluetooth connectivity. |
| 1.6 | The mobile application will allow the user to manually control the device. |
| 1.7 | The mobile application will allow the user to schedule the user to their desired times. |
| 1.8 | The mobile application will allow the user to get a notification when the device is out of range as an anti-theft precaution. |
| 1.9 | The device must have collision avoidance capabilities. |
| 1.10 | The device must be robust enough to handle trash collection. |
| 1.11 | The device must be waterproof. |
| 1.12 | The device must have the battery capacity to traverse the user’s driveway in both directions. |

## 2.4 House of Quality

The House of Quality diagram (Figure 2) is used to define the relationship between desires and the product capabilities. There was a plethora of designs considered for the software and hardware selections for the SPARC, and the House of Quality allows for a comprehensive analysis of the merits of using each of these designs. T the following page displays some of the key design options that were created for the SPARC and their estimated merits in conjunction with each other design choice.

The development of the SPARC had the qualities of being low-cost and easy to use as the top priorities. The utility of automating trash bin delivery is fairly low when compared to other household chores, so maintaining a low price point was key for considering designs for the SPARC system. Its unique ability to increase the level of independence experienced by accessibility challenged users made the ease of use of the SPARC another top priority; designs that made the SPARC too cumbersome or complicated to set up would severely depreciate this very unique merit of the SPARC and this heavily affected design selections.

Several different design selections were considered for the SPARC, each with different sets of merits. Design A required the use of expensive GPS and Pixy CMUcam5 modules, but provided a much higher quality of accuracy than Design B, which opted to do without these modules to reduce cost. While maintaining a low cost was a high priority for the design goals of the SPARC, the loss of reliability was too much of a liability in cost benefit-analysis. Design A was chosen despite its higher cost for its high reliability and ease-of-use. While keeping the cost down is of importance, maintaining the autonomy of the SPARC was of higher priority in order to keep its benefits in user accessibility.

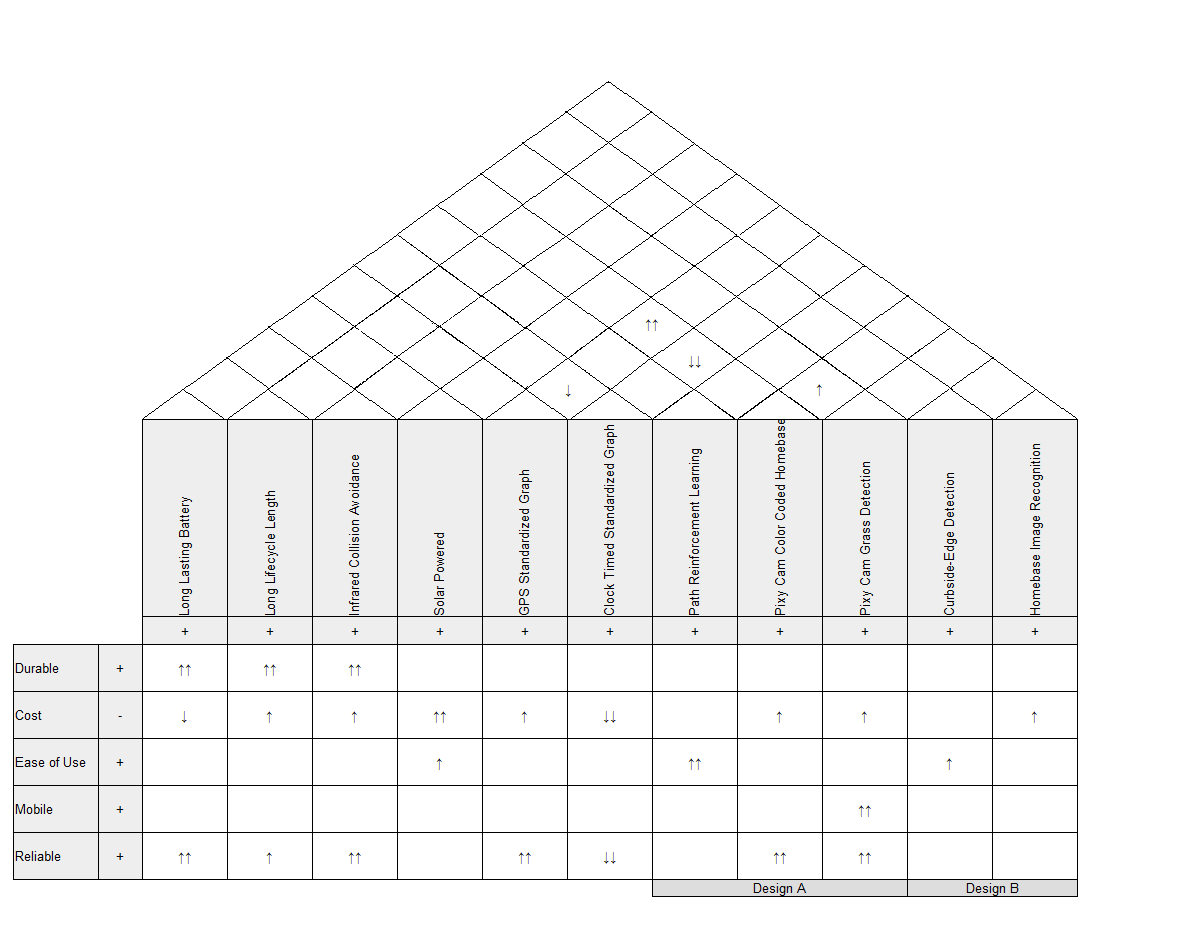


Figure 2 House of Quality Diagram

Several features, such as the inclusion of Solar Panels were not included in the final design. This house of quality served to weigh the merits of its inclusion, and was not simply a list of all the features included in the final design. Mobility was one of the design goals, but was a hard goal to satisfy. The inclusion of the Pixy CMUcam5 module allows for the detection of difficult terrain, which made it have a very high utility in cost-benefit analysis.

# 3.0 Design Constraints and Standards

In designing a product there are constraints and standards that we must consider. For the design constraints both Software and hardware constraints were taken into account in terms of the basic ABET constraints.

## 3.1 Standards Used

The table, Table 2, below demonstrate the standards used in this project.

Table 2 Standards Used

|  |  |
| --- | --- |
| **Component** | **Standard Title and Description** |
| Bluetooth | IEEE 802.15.1: WPAN / Bluetooth: Defines physical layer (PHY) and Media Access Control (MAC) specifications for wireless connections with static, portable and mobile devices in a personal operating space. |
| USB | IEC 62680-2-1:2015 :Universal serial bus interfaces for data and power - Part 2-1: Universal Serial Bus Specification, Revision 2.0 |
| GPS | Title 47 CFR Part 15 Code of Federal Regulations: GPS receivers are regulates under this section |
| Trash Can | Brevard County- Code of Ordinances Section 94: Defines the proper waste disposal methods and rules. |

## 3.2 Design Constraints

There were numerous constraints for designing the SPARC that were immediate in the minds of all of our group members such as the extreme necessity for low cost and for ease of use, though many constraints were taken for granted and had to be analyzed with in-depth analysis of the different meta-environments it would affect.

The following subsections review all of the design constraints that were encountered upon initially designing the SPARC, as well as all of the constraints that were noted when brainstorming the implications the SPARC could potentially have.

### 

### 3.2.1 Economic Constraints

Several constraints limited the potential design choices that could have been used to solve much of the SPARC’s search problem. Foremost of these was the specifications for an inexpensive and easy-to-use design. The SPARC is an automation device for trash delivery, so for such a device it must be convenient and inexpensive enough to be preferable to taking out the trash manually for the nominal user. Users are not likely to spend a large sum for a product that automates a choice that takes no more than 3-5 minutes of their time every week. Expensive component parts such as the GPS module, batteries, and motors were selected with cost foremost in mind.

### 3.2.2 Environmental Constraints

The SPARC has very little impact on the environment that is not already impacted by taking out the trash in a traditional manner. The SPARCs slow operating movement speed excludes any chance of injuring local wildlife or domesticated animals.

### 3.2.3 Sustainability Constraints

The SPARC utilizes lithium ion batteries for its rechargeable battery bank. As the life of the batteries wane, they will need to be disposed of or replaced. Batteries will corrode with repeated exposure to water and air, leaking toxic chemicals into the environment. Even batteries that are disposed of in the proper manner will not fully degrade into the environment in any reasonable amount of time, and will end up sitting in a landfill all the same, with protections against corrosion albeit. Despite the additional sustainability afforded by using rechargeable batteries the SPARC still bears this footprint on the environment.

The energy used to power the battery bank could possibly be generated through a clean and renewable resource such as solar, wind, or hydroelectric, or a clean and sustainable resource such as Nuclear Fission, but will likely be generated in large part from the burning of Fossil Fuels and/or Natural Gas. The burning these energy sources of course contributes to the strengthening of the greenhouse effect from the additional carbon dioxide (and methane gas, with Natural Gas) released into the atmosphere. The unchecked growth of the greenhouse effect contributes to a Global Warming which negatively affects the biodiversity of life on Earth, and with it the Earth’s ability to habituate Human Civilization.

### 3.2.4 Manufacturability Constraints

The parts selected for the SPARC were selected with cost in mind foremost, so prefabricated modules were sought over specialty components were selected to fit any specific need.

However, the SPARC was made to be compatible with garbage bins used with the Brevard County Waste Management. This heavily effects the manufacturability of the SPARC, as its use is limited to areas where users also use bins that are in accordance with this standard. Attempting to manufacture SPARC units for users in areas that are outside of this standard would be futile; the SPARC system would need to be customized to fit other standards of garbage bin if manufacturability across standards were desired, which it currently does not.

### 

### 3.2.6 Ethical Constraints

The ease-of-use of the SPARC was of particular concern to ensure that users with accessibility problems would not be overly inconvenienced when trying to set up the SPARC. The creation of the SPARC would impart a great convenience for those who would have trouble taking out the trash, and would allow them for a greater independence in their daily lives. Keeping in mind those who have challenges with accessibility, several design choices were affected. Setting up an RFID driven RTLS (Real Time Location System) would have been a very powerful solution to solving the SPARC’s SLAM (Simultaneous Localization and Mapping) problem, but the solution required the driving of several RFID-tagged stakes into the group, which would be abhorrent inconvenience for users and one that is very contrary to the SPARC’s specification for accessibility.

### 

### 3.2.7 Health Constraints

The automation of trash delivery via. the SPARC ensures that trash is delivered with accordance to the user’s delivery schedule. Omission or inability to take out the trash on a regular basis can cause waste to accumulate in trash bins. The stagnation of such refuse can become breeding grounds for infectious diseases, molds, pests, and other threats to personal health and property. This automation fail-safes many situations where the user may be unavailable to take out the trash, such as personal emergencies, unplanned overnights, etc.

### 

### 3.2.8 Safety Constraints

The SPARC is designed with safety for the user and pedestrians in mind. The collision avoidance technology prohibits the SPARC’s pathing algorithm from ever choosing to move in the direction of an obstacle; human, animal, object, or otherwise. The SPARC’s low operating speed ensures that there is ample time to react to sudden obstacles, and doubles as a fail-safe should the algorithm fail such that any collision would impart negligible harm to persons or property.

### 

### 3.2.9 Social Constraints

The impact social constraints had on designing the SPARC were very minimal. The idea of having household chores automated is a concept that is not an exotic one. The iRobot Corporation’s popular household robotic vacuum “Roomba” has been vacuuming since late 2002[1], and its innocuous size and behavior has allowed for most robust, imposing, and potentially dangerous automatons like the self-driving car to be introduced into the marketplace with moderate success. Having trash delivery automated is not something that should draw much social unease from neighbors with such products already having an established role in society. Neighbors may consider use of the SPARC a form of laziness, but considering the SPARC’s positive impacts in accessibility and user independence the prevalence and magnitude of this feeling would be very minimal.

### 

### 3.2.10 Political Constraints

The SPARC also had some unique constraints for its area of operation that is unlike many other autonomous vehicles; the proximity of private property own by entities other than the user. While other autonomous vehicles have the convenience of operating in most any space in their operation area, the SPARC is constrained by the unpredictability of property lines between homes. While mapping all of the obstacles in the area surrounding the user’s driveway may have been a powerful tool in the creation of the search algorithm in projects such as the A.S.R. as previously explored, the SPARC has no indication of where the user’s property ends and private property begins.

While current use of SPARC systems waves the designers of any liability to SPARC units operating on private property, it is a moral obligation that the SPARC be designed such that the risk of intruding on neighbor’s property should be absolutely minimal, and certainly not something that normal operations should encounter like mapping the surrounding area would without the use of a drone system with a sophisticated mapping algorithm or something similarly exceeding our budget and project scope.

# 4.0 Relevant Project Research

To successfully implement this design, extensive research must have been done. Firstly, similar past projects were looked at to see how other projects handled and executed their designs. Next was research on the on the core software on the project, the pathing finding algorithm and the computer vision aspects of the project. Next was research on the hardware and electronic aspects of this project. Lastly research was done on other software aspects, like wireless connection and the mobile application.

## 

## 4.1 Similar Projects

The following projects have similar descriptions or a similar premise as our project. All of these projects include an autonomous vehicle that uses sensors and/or computer vision for its movements. All of these and robots meant to aid humans in doing simple tasks by making it efficiently and accurately done.

* A.S.R. The Autonomous Sentry Robot from UCF
* The Manscaper Autonomous Lawn Mower from UCF
* S.H.A.S Bot from UCF

### 4.1.1 A.S.R (The Autonomous Sentry Robot)

This project’s main goal was to create a robot that is autonomous and can be used for sentry in enclosed areas. It was designed to map an unknown room and localize itself on a based its mapping of the area. It had capabilities that included object detection and collision avoidance similar to the SPARC. Researching this project assisted us in the design of an autonomous vehicle with a chassis that can hold all of its electronics, autonomous maneuvering of a robot, and remote control by users.

Despite the similarities in the design problem, there were many specifications of the ASR that were not applicable to designing the SPARC. The mapping of an enclosed area was a requirement of the ASR that necessitated sophisticated visual sensors in order to accurately map the area. The SPARC is designed to operate in an open space of undefined area and thus make mapping the area an unrealistic endeavour especially considering the small area of operation that the SPARC would potentially have to sense and traverse. The detection of motion was a feature that was important to the designers of the ASR for detecting intruders to the area, and this pushed them towards certain technologies such as the Kinect, which the SPARC has no need for as all obstacles are treated in the same manner whether they are ‘new’ to the system or are already known.

### 4.1.2 The Manscaper Autonomous Lawn Mower

The Manscaper was another device made to ease the day to day chores we may have by automating the mundane task of mowing your home’s front lawn. This product was made to efficiently and effectively mow the lawn for its user autonomously and finish its task without much need for interference from the user. In a similar fashion to our project, the Manscaper will return to its charging station after it finished ist pre-mapped lawn cutting area, which it maps when the device is initially used. The methodology of return to the charging station was particularly challenging, so this was an especially useful reference for designing our own solution.

The Manscaper had some differences in its path determination and collision detection. It had the additional constraints of being a lawnmower, with all of the danger of being an unmanned blade-spinning robot. For this purpose, the designers of the Manscaper provided many extra levels of redundancy into their design, such as a bumper fail-safe for when collision avoidance failed. The SPARC’s low top speed works as a general fail-safe for collisions, with the SPARC’s IR sensor being sufficient to avoid such close collisions in any case.

Another major deviation was that the Manscaper opted for a hybrid relative and absolute positioning system for their localization and mapping. Their design included sensors on the gearshafts of their motors as well as a magnetic compass in addition to their other visual input technologies to correct the instability in using a relative positioning system. Our team however had the advantage of being able to use GPS, so implementing such technologies was not necessary for the development of the SPARC. The centimeter precision so desired by the Manscaper in order to have evenly cut edges of the lawn made the use of GPS an impossibility for their design.

The Manscaper also provided much inspiration for additional features for the SPARC. The avoidance of adverse weather conditions was a key design principle for the Manscaper, as cutting wet grass has severe consequences for the functionality of lawnmowers and health consequences for the grass itself. While the SPARC did not include this feature such as this in the final design, it was noted that it would be a very important feature for future iterations of the SPARC to be able to avoid weather conditions that would prove hazardous for the system.

### 

### 4.1.3 S.H.A.S Bot

The S.H.A.S bot is again, another robotic device made to aid humankind. This device specifically was made as a surveying robot that wirelessly transmitted a video feed to a laptop PC, giving the user a first person view and utilizes gyroscopes to make sure the video displayed is non-disorienting. Researching this project gave insight on image recognition and processing, wireless connectivity, and object avoidance technologies.

S.H.A.S. bot was uniquely one of the autonomous vehicle projects that had implemented a GPS antenna into their design. Many of the other autonomous vehicles projects rejected designs utilizing GPS, as their need for precision was much too high to be able to implement such a system. The S.H.A.S. bot did not simply use the GPS for coordinate positioning however, but also for information on altitude, and meta-information for calculating the S.H.A.S. bot’s speed over a certain distance.

The SPARC did not choose a GPS utilizing design for quite the same reasons as the S.H.A.S. bot; the SPARC didn’t need to track information for speed calculations or for localization besides the latitude and longitude. The S.H.A.S. bot designers had also a larger constraint on the size of their GPS module, as their vehicle was the approximate size of a commercial RC car, while the SPARC has a much more generous profile to strap parts to.

## 4.2 Pathfinding

The pathfinding problem of the SPARC is to find and follow paths to the destination while avoiding collisions and remembering locations of recurring collisions for future path optimization. Broadly, the problem falls under the abstract class of search problems referred to as Simultaneous localization and mapping as the SPARC must keep track of its location relative to the destination in an area that has the potential to be constantly changing over long periods of time.

The SPARC utilizes elements of Shortest-Path algorithms such as weighted paths, and reflexive cost evaluation algorithms. It includes some elements of reinforcement learning; modifying the weights of nodes iteratively with each trip based upon the duration of individual trips and the frequency of encountered obstacles in certain nodes.

### 

### 4.2.1 Shortest-Path Search Algorithms

Of the search algorithms researched, Reflexive search and A\* search were very applicable to our design. Reflexive algorithms make decisions based on evaluation algorithms that assign weights to each action based upon the current state of the graph, and the state of the graph after the action. A Reflexive algorithm strategy had the appeal of a very simple implementation, as there would be no necessity to construct a search tree which would bring its own design challenges. The detection of, and maneuvering around is the most pivotal consideration for SPARC’s pathfinding, and this heuristic is not one that more effectively implemented by a more sophisticated planning agent, as the SPARC will have to react in real time to obstacles upon what would be a ideal path, albeit obstacles that are relatively slow-moving.. Reflex does however require some sort of heuristic in its cost-analysis algorithm that prohibits it from wandering, so some sort of positioning with relation to the destination would need to be established to ensure that it tends to move in that direction.

Below is an algorithm design for the SPARC utilizing a reflexive pathing policy. Nodes are expanded whenever imminent collisions are detected such that the SPARC finds a new path to the destination that does not have a such a collision.

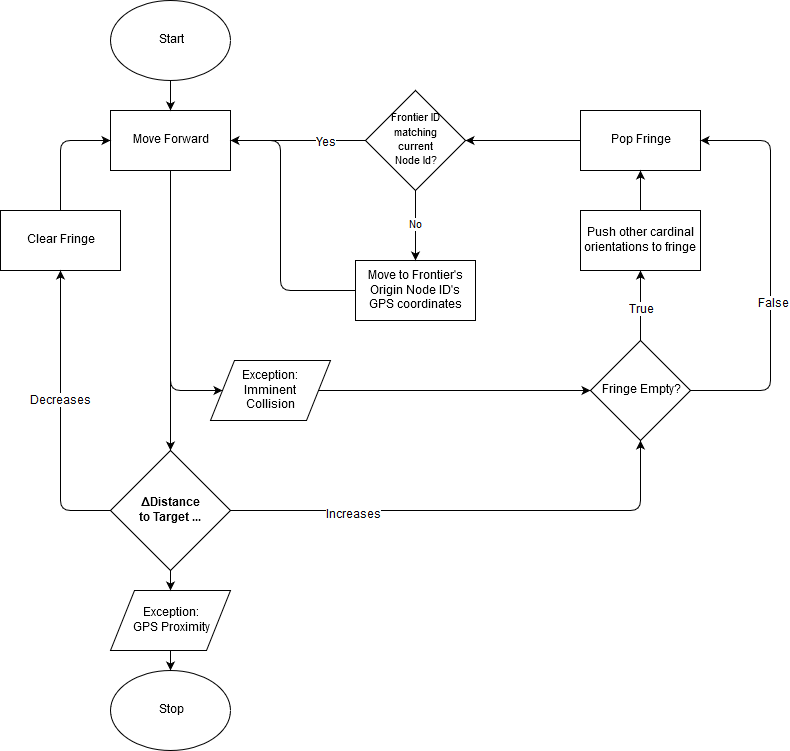


Figure 3 Reflexive Algorithm Design Flowchart for SPARC

A\* was appealing algorithm to use, as there are many useful heuristics that can be integrated for the SPARC’s path theoretically allowing for a more efficient path to the destination. A\* algorithm is one that assigns a certain weight to each path or node in a search graph similar to uniform cost search algorithms like Breadth/Depth-First search and modified uniform cost searches like Dijkstra's Algorithm. The A\* takes the idea of weighting search nodes, but also adds an additional heuristic to the weight depending on certain factors of the state of the graph. The theoretical “deep water / shallow water” problems that are commonly used to describe A\* search are readily translated into maneuvering on concrete vs maneuvering on grass for our own search problem, making A\* seem like a sure choice for the SPARC. Additionally, by planning paths in advanced, the SPARC would able to store important information about the graph of the users driveway, including obstructions and past pathways to the destination so that it does not encounter the same obstacles each delivery. The necessary construction of a graph for the A\* algorithm to search does necessitate a standard for measurement of distances between each node though, which necessitates some additional design considerations in order to implement them.

For A\*, the initial path construction would need to be created by the user to avoid having the SPARC wander in the manner as described previously. Unlike other similar projects such as the ASR, the SPARC by no means should be attempting to map out the user’s driveway. The SPARC’s operating environment is an open space with no potentially no discernable separation between the user’s and their neighbor’s properties; the ability to map out the area is constrained heavily by these factors.

By using the mobile application’s remote control feature, the user could be able to create such a path for the SPARC that it would initially try to adhere to while still avoiding collisions and re-finding its path to the destination if one was imminent. Despite adding more complexity for the first-time user experience, it was acceptable within our design parameters, as accessibility is not hindered so long as the user is able to control the SPARC remotely rather than being forced to manually push the SPARC to give it their preferred path for it to take to the destination.

The most meritorious features of A\* assume that the algorithm is able to fully predict the optimal path to follow before taking any action, which unfortunately is not fully possible in our particular problem. It would be a significant undertaking designing a system that could analyze the area around the SPARC crawler before it takes any movement into account, so creating a true A\* algorithm was not a complete solution for the SPARC, though the SPARC does borrow many elements from the A\* algorithm as the solution to its own search problem.

### 4.2.2 Collision Avoidance Technologies

Selection of collision avoidance technologies was a relatively simple choice for the purposes of the SPARC. The advent of Collision Detection as a staple feature in the automotive industry gave a large selection of mature technologies.

Modern self-driving vehicles most commonly deploy a powerful sweeping LIDAR sensor in combination with other localization technologies such as GPS, inertial sensors, and cameras. The sweeping scanning utilized by self-driving cars is not as imperative for the purposes of the SPARC, as the threat of imminent collisions to the SPARC will be from stationary or slow moving objects in most conceivable cases. Previous projects examined commonly used a multitude of detection agents, such as LIDAR, SONAR, and physical bumpers, though the SPARC does not have such a dire need for input as its top speed is low, and such detection measures such as a collision bumper would likely be redundant with an element such as a simple LIDAR rangefinder.

### 4.2.3 Absolute Positioning

The reflexive solution had a very dire design problem in that the reflexive cost function required the a positioning of the destination with respect to the SPARC’s current position; without this heuristic, the sparc would be doomed to wander aimlessly around the user’s driveway, exploring for an undefined amount of time. Thus, for this design it was necessary to explore the implementation of an absolute positioning system. The A\* solution also required some sort of standardization in order to construct the driveway graph, which would also resolve by allowing the SPARC to retrieve its coordinates and compare them with its destination.

#### 4.2.3.1 GPS

Foremost was GPS, which had the large benefit of being a localization service free of charge besides the antenna used for connection. The GPS module would solve the localization design problem by allowing the SPARC to access the location data on its current position, using this data as a reference to compare to past positions and the position of its destination. This information would be used to formulate system of absolute localization for each node created in the search graph. With the localization provided by GPS, search algorithms could have a standard of which to tell the SPARC how to move in order to reach the next node in the sequence.

The cost-benefit analysis of adding a GPS modules was found to be somewhat marginal among the different options; it would be have been among one the most expensive digital components to purchase. The additional power drain of the module was also of concern, as it might necessitate having a higher capacity and more expensive battery onboard the SPARC, which already was one of the largest price-points among the electrical and digital systems. GPS ended up being the technology we ended up moving forward with despite all these drawbacks for its reliability and supreme ease of setup. These two important goals for the SPARC were important enough to wave the increase in cost.

#### 4.2.3.2 Real-Time Location System (RFID Localization)

RFID looked to be a very promising technology to integrate and solve these design problems, with the exceptionally low cost of passive tags, but compared to to other similar projects using RFID technology the SPARC lacks a large number of points from which to reference to triangulate positions of the SPARC and its position with regards to its destination. A proposed solution was to planting a number of ‘stakes’ with RFID passive tags in the ground surrounding the user’s driveway, but this solution vastly increases the initial setup complexity which conflicted heavily with our accessibility design specifications. Having too few of these stakes would result in a very weak positioning system, more akin to a “marco-polo” relative positioning which was far too inefficient to be a desirable design.

There were also social and legal constraints that made this solution sub-optimal. It would be a very unappealing lawn feature that may turn off potential consumers from wanting to use the SPARC, and some homeowners associations might outright prohibit having such a device planted around the yard.

There were additional risks using the stake solution. There was a large chance of the stakes being damaged from being accidentally driven over by an automobile pulling out of the driveway, or even more dangerous; a lawnmower driving over them. The drawbacks far outweighed the small price differential it had over using a GPS system for the same purposes, so any use of RFID technology was dismissed moving forward.

### 4.2.4 Relative Positioning

While the Reflexive solution relied on positioning itself, for deterministic search algorithms such as A\*, it was only a means to the end of normalizing all the of the graph search nodes. With the prohibitive cost-benefit to implement any of the position technologies aforementioned, additional problem interpretations were explored for solving this problem for the A\* design.

#### 4.2.4.1 Odometer

As the graph would be one of discrete distances from the starting location, one of the foremost ideas to research was to include a odometer module to the design of the chassis to allow the SPARC to record its progress through the search graph in the same manner that an automobile would. This solution had several shortcomings, however. While researching, there was a lack of any odometer devices that were compatible with the type of motors that we selected for use by the SPARC. The motors have no digital or analog output which would be necessary for the odometer module to receive to track the distance tracked.Having no digital output from the motors meant that integrating an odometer would necessitate some very intensive mechanical modifications to the wheel axes or the motor, which none of our project team has any relevant experience in. The large overhead for research and the large and still yet unknown cost for making such modifications made designs using an odometer extremely unfavorable in cost-benefit analysis.

#### 4.2.4.2 System Clock

Another design choice considered was a design that used the system clock as a timer for how long each motor was running. While other solutions envision our search problem as a shortest-path problem, it is a Simultaneous localization and mapping problem. While most shortest-path algorithms assume that the composition of the graph is known, our search problem deals with an unexplored graph that has no assurances of a constant position of obstacles and of valid paths. Using the system clock as a means for timing the duration of motor impulses would have served as an alternate means of tracking the path the SPARC would take to the destination without using any kind of formal positioning. While this solution does remove the need for expensive localization technology like GPS or Active RFID, this solution would have significant problems impulse normalization. The state of the terrain would drastically affect the amount of impulse to reach the next node in the path. Even if the SPARC was able to detect if it were on grass or concrete, the variations with those terrains changes drastically with the inclusion of such variables as inclement weather, grass density, etc. The amount of additional modules that needed to be included made this solution infeasible without also allowing for some considerable degree of unreliability, which was not something under considerations for compromise in cost-benefit analysis.

### 4.2.5 Markov Decision Processes

Whereas Shortest-Cost Algorithms handle pathing in cases where the actions are fully deterministic of the choices made, Markov Decision Processes provide frameworks for which the outcome of each step is not fully controlled; there exists only a chance for the action to be taken. The name Markov refers the nature of Markov Decision Processes (MDPs) as algorithms that only take into account the current state of the system rather than referring to past events[2]. As only the current state is examined, a “policy” is constructed that will determine for the algorithm what action should be taken in each different situation the agent could be in. For example, for the SPARC we could design a policy that tells it to stop before colliding with obstacles; this notably should not have to depend on any past information.

Exploring MDPs was of interest to designing the SPARC because of its framework for Reinforcement Learning. For MDP’s one of the algorithms used to solve them is known as Value Iteration, where each search node is assigned a weight and direction for that weight indicating the most beneficial path to take at that moment given the reward for completion and the amount of time taken to get there. These values determine the policy for a given MDP. We can change these values dynamically with each iteration through the graph, and over time construct a more efficient policy for the graph.

For the SPARC, we could construct a default policy from initial user input, and over time evaluate which paths end up leading to the destination, and which paths end up encountering obstacles more often than not. Using reinforcement learning, the SPARC can behave in a more intelligent manner, avoiding common obstacles rather than having to encounter them each delivery routine. So, if a user had a new car parking in their driveway in the way of the SPARC’s usual path, the SPARC can self-correct and determine a new default path that avoids this new obstacle, removing any necessity for the user to intervene. This process was very integral to the final design of the SPARC, as it allowed for a very high level of ease-of-use and autonomy; our primary focus on developing the SPARC.

## 4.3 Computer Vision

One of the goals of the project team was to try to utilize the burgeoning field of computer science known as Computer Vision. Computer Vision (commonly abbreviated as CV) is broadly defined as the processing of visual information into workable computer models[3]. Computer vision has made some of the more complex works of robotics such as self-driving cars possible, with the ability to make sense of the immense amount of data provided by real-time visual input.

There were several relevant technologies of interest for designing the SPARC; image recognition, edge detection, and color detection all provided intuitive solutions to the unique pathing problem that the SPARC needed to solve.

### 4.3.1 Image Recognition

There existed a very important design problem for the SPARC, as it needed to proceed to its home base in order recharge its battery bank while it waited for its next delivery time. While the localization and mapping methods explored in the previous sections all have some measure of accuracy, none would reliably be able to orient the SPARC with enough precision to have it make contact with the home base with enough accuracy to allow it to begin charging. There needed to be some additional technology to enable the SPARC crawler unit to be able to identify the home-base unit.

Within the open source resources for computer vision, there were already very complex neural networks developed that could be trained to recognize common objects such as Google's TensorFlow library. The use of neural networks allows for the shape of objects to be analyzed and classified with a confidence rating based on its trained dataset. In figure 4 below, a neural network trained to identify pedestrians identifies people entering a storefront.



Figure 4 Shape Recognition on Humans

*“SRT Shape Recognition Technology” by wikipeida user QueSera4710, licensed under CC BY-SA 3.0*

The SPARC could theoretically use such a library and have its own neural network trained to recognize images of the home base, so that when the SPARC’s crawler is in line-of-sight of the home-base, it could lock-on to the location of the home-base and navigate towards it such that it will be able to charge.

Our language we chose for the SPARC system was C, though much of Open CV uses C++ so many of the libraries would not be so simple to utilize. The amount of processing power was limited by the size of our embedded system and the amount of system resources used in real time image recognition made it not such a ideal solution in cost-benefit analysis.

### 4.3.2 Edge Detection

One of the design possibilities explored for the SPARC was use the use of edge detection. Edge detection algorithms analyze a gradient for sharp differences in contrast or between brightness between pixels in order to determine the separation of objects. Figure 5 below is a before and after of an array of coins which have their edges analyzed by an edge detection algorithm. The sharpest differences in the color gradients is represented by a line in the image to the right.



Figure 5 Edge Detection used on Coinage

*“Edge Detection”* *by wikipedia user* Shaddowffax*, licensed under CC BY-SA 4.0*

Applying this algorithm to the output from a visual sensor, such information could be used to create useful information for path determination. Obstacles with recognizable patterns such as parked cars could be identified readily, and important traversal information such as the separation of road, sidewalk, driveway, and lawn could potentially very powerful tools for creating a robust pathing algorithm.

There were several drawbacks in trying to implement such an algorithm for the SPARC. The lack of a uniformity in potential obstacles was one of the major challenges. In the problem space for a SPARC obstacles could range anywhere from parked cars, small animals, and more problematic; large obstacles with uniform gradients such as buildings, sheds, or shipping containers. A large amount of computational power would be required to process these images in real time, as the determination of what gradients are important to collision avoidance or not would be exceptionally taxing from the amount of raw output a camera would contribute.

## 4.4 Hardware

One consideration that we needed to account for was the vast set of options available to us in terms of hardware for this project. This research was critical for determining some of the logistical factors of the design, such as the budget and ease of use, and the compatibility of the components. The following subsections of Section 4.4 outline some of the different considerations and the options available.

### 4.4.1 Microcontroller Options

We primarily considered two different models for our microcontroller. The initial consideration was the MSP430 from Texas instruments. The MSP430 was designed for low cost and, primarily, low power consumption applications. In terms of budget restrictions, the MSP430 variants would provide good performance while still being an affordable option. Additionally, the MSP430 Launchpad, a development board, included multiple timers and systems to handle interrupts and low-power modes. The benefits of this chip also included the past experience of its use already gained through previous courses.

The other microcontroller we considered, and ultimately chose, was the Arduino Zero’s chip. This chip is comparable to the MSP430 in many ways, however it has better specs than that of the MSP430 and has a large open-source community with resources and references that may provide guidance or troubleshooting. These benefits come at a higher budgetary cost. Moreover, the camera we have chosen for the computer vision aspect of our design has built-in libraries specifically for use with Arduino chips.

### 4.4.2 Serial Communication Protocols

For the microcontroller to receive data from external sources a serial communication protocol must be used to send and receive data bit by bit spread over a length of time over a specified bus. The data can be sent synchronously, synced with a clock signal or asynchronously. For the purposes of this project, we need to send data directly to the microcontroller to change pin states. There are multiple serial communication protocols to choose from that is supported by our chosen microcontroller.

#### 4.4.2.1 SPI: Serial Peripheral Interface Bus

The Serial Peripheral Interface bus is a synchronous serial communication interface used in embedded systems for short distance communications . SPI uses a master-slave architecture with only one master. The master sends out the clock signal and sends data to the slaves on each clock burst. IT has signals with names SCK for the clock MISO for master in slave out and MOSI for MAster out Slave in. SPI is rather simple and will be considered for usage in the SPARC’s communication network.

#### 4.4.2.2 I2C: Inter-Integrated Circuit

Inter-Integrated Circuit or I-squared-C is another synchronous protocol that is a multi-master, multi-slave signal ended, packet switched serial bus. I2C has two line, Serial Data Line (SDA) and Serial Clock Line (SCL) that are bidirectional. I2C is slightly more advanced than the other serial protocols referenced and will not be considered due to its complexity.

#### 4.4.2.3 UART/USART

UART is the most widely used asynchronous serial protocol as it is very simple to use and most controllers come with a hardware UART on board. It uses one data line for transmitting and another for receiving 8 bit data both ways. Usually 1 start bit, 8 data bits and 1 stop bit are sent. To use UART the microcontroller must specify the transmission speed called the baud rate to synchronize with the start bits falling edge as this is asynchronous and does not take in a clock signal. This protocol is related to USART as this includes a synchronous option for a similar operation. The microcontroller we have chosen supports USART and it is the protocol we are heavily considering due to its ease in integration with our chosen Bluetooth module.

# 5.0 **Hardware** Design

Many variables came into play while determining how to design the Hardware of our SPARC Project. The first of many obstacles was that the hardware for the SPARC project needs to be of durable enough construction to support the weight of a fully loaded trash can, an estimated fifty (50) pounds. In addition to this, it needs to be able to keep out moisture well enough to where the circuits do not short themselves.

The Hardware Design aspect of this project began at a very early stage of the project, as our hardware has a major impact on the equipment needed. For example, our motors require 12 Volts with 10 Amps for each motor. This meant that we needed a battery bank that could safely output 20 Amps without overheating.

There were many changes that were made throughout the design process, which will be discussed in detail in the respective subsections throughout Section 5.

## 5.1 Hardware Overview

The Hardware for SPARC can be divided into 3 distinct categories: Robotics, Internal Electronics, and the Home Base. The Robotics aspect consists of the Chassis Design as well as the motors, gearboxes, and treads. The Internal Electronics section is comprised of the Power Source, Voltage regulators, the Microcontroller, Bluetooth Module, Pixy Camera, and the Motor Controllers. And finally the Home base is made up of the Solar Panels and the Conductive Charging contacts.

Simply put, the Chassis has to support the weight of a trash can without putting too much stress on the motor axles. The Internal Electronics need to deliver power to each component accurately, and within a safe margin of error. And the Home Base needs to provide enough voltage to successfully recharge the battery bank in side the main unit. Each of these pieces of hardware had to be carefully selected to ensure proper functionality, mobility, and charging, without excessive battery discharge, overheating, or overloading any of the Integrated Circuits.

## 5.2 Robotics Design

The robotics aspect of this project was the most difficult, as no one in our group had any real experience in the area of robotics, besides some minor childhood science experiments. Luckily there are a lot of videos and tutorials and tools around the internet that we could use as reference. Most of these tools will be outlined in the upcoming sections.

### 5.2.1 Chassis Structure

The design of the Chassis is still in a preliminary stage. As the Chassis is purely structural, aesthetics don’t matter too much. We are designing this cart to be functional, not stylish. That being said, the design of the structure needs to be able to support its own weight, and the weight of the trash can on top of it, which fully loaded we estimate to be 50-60 pounds.

As mentioned, this aspect has not been fully thought out yet, and all we have are a few concept sketches. We plan to built it out of sheet metal, probably from a local hardware store, along with some brackets. The PCB will be placed inside an enclosed area right underneath the trash can, and the edges will use a rubber gasket of some kind of to seal out water.

Figure 6 shows some preliminary concept art of what SPARC might look like. This design is subject to change before the final submittal.

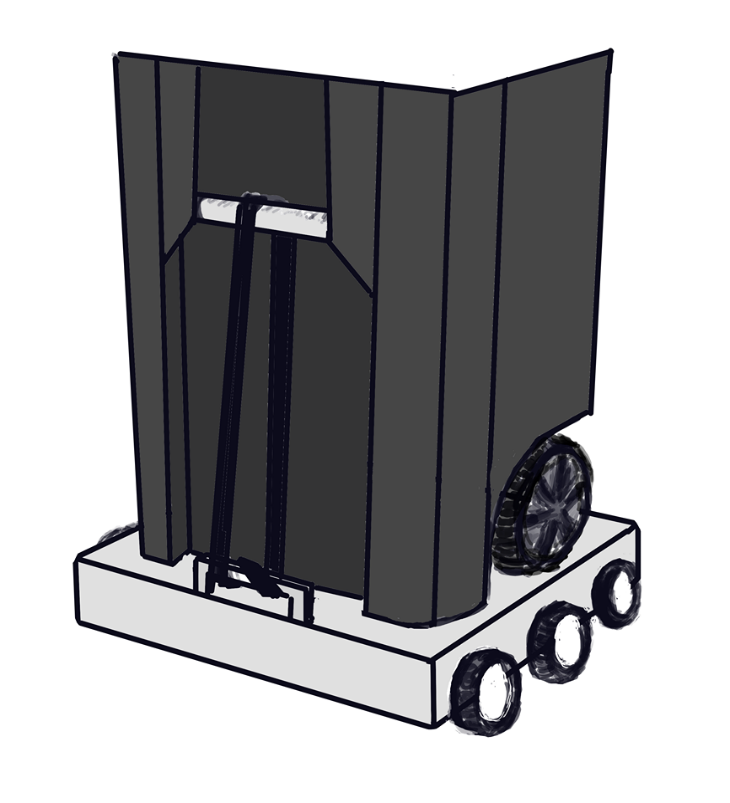


Figure 6 Concept Art

### 5.2.2 Motor and Gear Sizing

The Motor and gear sizes were one of the first aspects of the project that we decided to figure out. We knew that finding out which motor and gears to use would be a major impact in the Electrical Design of the project. One of the tools that was used was a “Drive Motor Sizing Tool” provided by Robotshop.com. Figure 7 shows the Drive Motor Sizing Tool. The output that the tool gives, shows how many Revolutions per Minute, and the Torque that each motor requires, given input requirements. We were a bit harsh on the requirements, to be sure that our motors will perform even under the most rigorous of conditions. For example, we put the maximum incline to be 20 degrees, even though it will probably never exceed 5 or 10 degrees. Also, the efficiency was an unknown variable, which only affected the required Amperage and Battery sizes required, however we calculated this ourselves later, and did not use that info from the Tool, so it has been omitted from the figure.

Using the Drive Motor Sizing Tool, we knew that the motor needed to output 248oz-in of force, at an angular speed of 153 revolutions per minute. This torque requirement is extremely high for a standard DC motor, and thus we had to use a more standard DC motor and a gearbox to achieve the desired specifications. The DC motor we chose is a “Banebots RS-540 12V 17200 RPM Brushed DC Motor”. On its own, this motor is rated for 4.47oz-in of force at 17200 RPMs. To adapt this motor to fit our specifications, we need to attach is to a gear box. A gearbox causes the motor to sacrifice RPMs to increase torque. The gearbox we chose, is a “Banebots P60 Gearbox: 1.5inch shaft, RS-540/550 Mount, 81:1”. The Motor and gearbox combined outputs a rated 362.07oz-in of force, at a rated 212.35 RPMs. The Stall current, meaning the max current the motor can pull, is 10A per motor. So the motors together need the ability to pull 20A from the rest of the circuit.

RS-54 DC Motor P60 Gearbox

*Permission Pending from BaneBots*

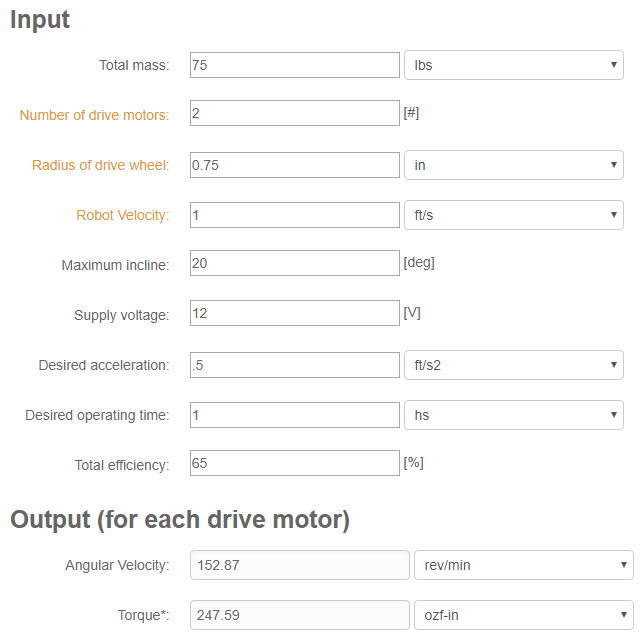


Figure 7 Drive Motor Sizing Tool

*Permission Pending from RobotShop*

#### 5.2.2.1 Brushed vs Brushless vs Servo

The RS-540 Motor is referred to as a “Brushed” DC Motor. A Brushed DC Motor uses an electromagnet system surrounded by permanent magnets, while a Brushless DC motor has a magnet surrounded by electromagnets (Figure 8). Basically the pieces into switch places.

The Brushless DC motor provides many advantages over a brushed DC Motor. These advantages include speed limiting, fine motion control, and holding torques. In addition they usually are more reliable, have a longer lifespan, and generate less electromagnetic interference.

The reasons we picked a Brushed DC motor though is cost and customizability. While the cost of a Brushless Motor isn’t extremely expensive, it is usually at least twice as much as a Brushed DC Motor. With a Brushed DC Motor, there are many options between gearboxes and motors, where brushless motors are not usually used with gearboxes, and do not have as many options to meet specifications that we need.

A third option for movement control is Servos (Figure 9). This is the option we looked into originally, however in the long run, servos would not have been feasible. What attracted us to the idea of servos originally is the ability to exactly control and measure the axle’s position and speed. However, after further investigation, we found out that most servos do not have a full 360 degree range of motion, and even less have the ability to continue turning like a wheel.

x

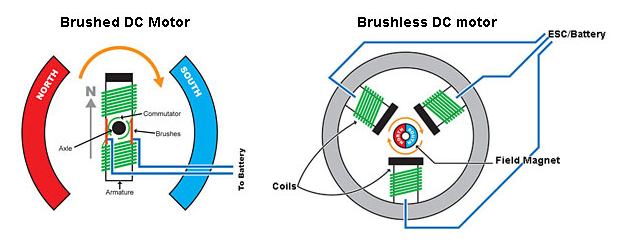


Figure 8 Brushed and Brushless DC motors

*Permission Pending from ThinkRC*

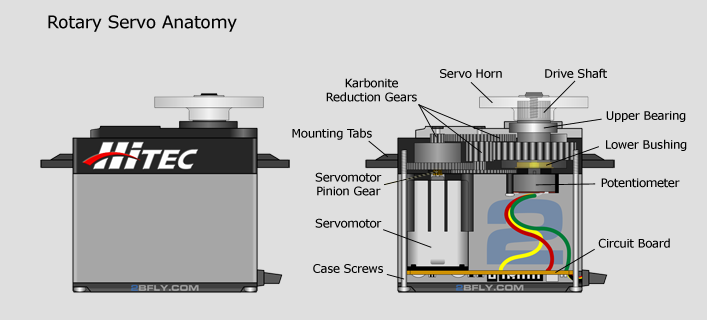


Figure 9 Rotary Servo

*Permission Pending from 2BFly*

## 5.3 Internal Electrical Design

The Electrical Design of this project was a complicated endeavor at first. Having never designed anything from scratch before, it was challenging to figure out how to power each individual device properly. We didn’t have a Lab Manual to tell us what to put where. We had to figure it out ourselves.

Also, having never worked with any PCB Software before, this step took days of research and planning and work to finally get a functional design laid out. Between built-in libraries not having the parts we needed, and making sure traces were wide enough, it was just a new experience for all of us.

The following sections will describe in detail the design decisions and procedures that have been made along the way.

### 5.3.1 Battery Bank

The Battery Bank was probably the second piece of hardware that was designed, immediately following the selection of the DC Motors and Gear Boxes. Since our motors require 20A of current at 12V, we knew that the demand of the battery bank would be quite high, especially if the device wanted to operate for any extended period of time.

To achieve this required voltage, we decided to go with 4 Lithium-Ion 18650 Batteries (3.7V nominal Voltage ea.) in series, making a total of 14.8V. This input then gets routed through the Voltage regulators that will be talked about in section 5.3.5 to lower the voltage to the 12V required for the Motors, 5V for the USB output to the Pixy Cam described in section 5.3.8, and 3.3V for the Bluetooth Module, RF Transmitters and the Microcontroller.

To achieve the acquired amperage, our first step was finding out the discharge rate of standard 18650 batteries. While the exact value differs between every battery manufacturer, that average is somewhere between 1 and 2 Amps. Also in the 18650 family of batteries is a high discharge variant, which can sustain up to a 35A continuous discharge. These are the batteries we chose to go with.

Our next step in the Battery Bank design was designing the system to last for a required amount of time. Most Lithium-Ion batteries have an Amp-hour rating of 2500mAh. This means that if the battery were to have a continuous current draw of 2500mA (or 2.5A), then the battery would last 1 hour. This also means that if we applied our 20A maximum current draw, then the battery would only last 7 and a half minutes. Clearly not long enough to drive down a driveway, wait 24 hours (albeit in a low-power mode), and then make a return trip.

We do not expect our device to move exceptionally, fast, so we wanted to budget the time to be 30 minutes in each direction, just to be on the safe side. This means that at Maximum current draw, we would need 20,000mAh. This was simple enough to achieve by simply adding 7 more rows of Lithium-Ion Batteries, with each row containing 4 batteries in series.

#### 5.3.1.1 Battery Charging Protection

When charging Lithium Ion Batteries, one must be careful to not overcharge the cells (each battery is considered to be 1 cell). A Fully Charged 18650 Battery Can safely have a voltage of 4.2V, while fully depleted, the voltage can be around 3.2V. Charging Lithium-Ion batteries (or any battery for that matter) is as simple as applying a current at a higher voltage across the battery.

Things get complicated however when one attempts to apply a normal charging current over a battery that is already fully charged. Doing so can cause Overheating and Overvoltage conditions. Both of these conditions can, and will, cause a battery to stop functioning, and in extreme circumstances can cause a cell to rupture and/or start a fire.

To prevent this from happening, a charging protection circuit is necessary. Each chip monitors the voltage across 1 row of batteries in parallel, constantly monitoring the voltage. If the row of cells (all tied together at the positive and negative ends) reaches the target voltage, the chip shuts off the output to the batteries that it is monitoring.

The devices we picked for this job are the MIC79050-4.2YMM by Microchip Technology. These chips monitor single cells (in parallel) up to 4.2V. Since each chip can have an absolute maximum current of 800mA, we designed the protection circuit to have 2 of these chips per cell in series, allowing each parallel array to receive the max amount of current possible from our design, for the quickest charging.

Under optimal conditions, the Battery bank receives 5A of current from the Home Base. Split between the 4 cells in series, that is 1.25A per parallel set of cells. And between the 8 cells in parallel, gives a nominal charging rate of 156.25mA per cell. At this rate, the entire battery bank should receive a full charge in 16 hours. Since SPARC Does not need to move on a daily basis, this charging rate is more than acceptable.

#### 5.3.1.2 Current Limiters

One possible issue that had been noticed that we wanted to address before testing if it would have actually become an issue, is “What happens when one of the cells in series is charged before the others in the series?”

When this happens, the Home base is still pumping 5A of unregulated current into the SPARC. If 3 of the 4 cells finish charging, leaving one cells still charging, we do not want 5A going through the two charging protection chips (which have a combined Absolute Maximum of 1.6A).

To prevent this disastrous scenario, we implemented a 1A Current Limiter, in the form of the AP2337SA-7 from Diodes Incorporated. This Device limits the current allowed through to 1A. While this does bottleneck the charging process from a 1.25A Nominal per cell in series down to 1A, it is a safety measure that must be implemented to avoid pumping too much current through traces that are too small. While yes, we could make traces wide enough, and get more protection chips to use in parallel, it would be unfeasible for a project of this scale.

This alteration to the charging current bumps up the time for a full cycle charge form 16 hours to 20 hours. WHich again, since the SPARC unit is not operating on a daily basis is an acceptable time frame for a full charge.

While bottlenecking the current is not an ideal solution, Ohm’s Law prevents us from directly limiting the current without also causing a decrease in voltage. This is the only solution we found that does not cause a significant decrease in voltage.

Schematic 01 Below shows the Battery Bank, along with the Battery Charging Protection Devices and the Current Limiters in place for the Main SPARC PCB.

REPLACE PAGE WITH

“Schematic 01 - battery bank.pdf”

AFTER PDF’ing

### 5.3.2 Microcontroller

For our Microcontroller Unit, we went through a few different design choices over the course of this project. Our main goal when selecting a Microcontroller was to select a microcontroller that we were familiar coding on. This limited our choices to Arduino and MSP chips.

Our first selection was the MSP430 chip. This was because we were all familiar with the chip, as well as the chip having a large amount of SRAM and Flash, in addition to having UART Channels for connectivity to our HC-06 Bluetooth Module. We ended up moving away from the MSP chipsets because the Pixy Cam has built in libraries for Arduino chips.

For our second (and current) Microcontroller Unit selection, we chose the ATSAMD21J18A-AUT. We chose this Microcontroller because it is from the same family of chips that control the Arduino Zero. We made the switch away from an MSP chip because of the Pixy Cam compatibility as stated above, however we selected this specific chip because of it is one of the highest-spec chips available in this company’s lineup. We figured that using image recognition would require more processing power and flash memory than a smaller chip would be able to process.

The specific chip we chose, the ATSAMD21J18A-AUT has 256Kb of Flash memory, with 32Kb of SRAM, and a processing speed of 48MHz. This chip has 64 pins, as opposed to the 48 and 32 pin options that were also available for the specified Flash and SRAM numbers given. We chose the 64 pin option because we have a lot of inputs and outputs given the bluetooth module, the two Motor Controllers, the Pixy Camera, and Proximity detection.

We did have a concern pop up closer to the end of the design stage that this chip does not list any UART Channels in it’s specifications. Because of this we went back to the drawing board to try and find a new chip, only to find out that this chip does indeed have UART capabilities, only it’s not as straightforward as on other chips. While the chip does not have any designated UART pins, it has six Serial Communication Ports (SERCOM) that can be assigned as USART (a more adaptive version of UART), SPI, I2C, SMBus, and PMBus connections.

This fits our needs for communication with Bluetooth, and then the chip itself should be compatible with the Pixy Cam, and all other devices that we have. Schematic 2 below shows the pin layout of the chip, and all appropriate connections.

REPLACE PAGE WITH

“Schematic 02 - mcu connections”

AFTER PDF’ing

#### 5.3.2.1 Computer Memory Types

As mentioned in the previous section, we chose this chip out of the family of Arduino chips because of its excessive SRAM and Flash sizes. These are important factors in programming as it directly limits the amount of object a Microcontroller can process.

To start off, SRAM is part of a family of memory called RAM, otherwise known as Random Access Memory. RAM is what is known as “Volatile Memory” meaning that when power is shut off to the RAM, all the data is lost forever. RAM itself can be divided into DRAM and SRAM. The D and S stand for Dynamic and Static, respectively. Static RAM is the more expensive type, because it holds its data forever as long as electric charge is maintained. Dynamic RAM, on the other hand, only holds its data for about 5ms before losing the data unless it is actively refreshed.

Contrary to RAM, ROM is “Read Only Memory”. ROM is non-volatile, meaning that the information stored on it will not be erased when the power gets turned off. This is a type of memory that needs specific ways to write new data to it. It is not as easy as just uploading a new code. The information in ROM devices is for all intents and purposes permanent, unless it is first completely reset, and then rewritten.

Finally, Hybrid Memory contains aspects of both RAM and ROM. In this memory type, You can Read and write freely to Hybrid memory, just like RAM, however when power is lost, the data remains, like ROM. Flash memory exists in this data type.

So when we say that our ATSAMD21J18A-AUT chip has 256kB of Flash Memory and 32kB of SRAM, this means that we have 256kB of space to write our program, algorithms, have the SPARC Learn and Remember its “idea path”, where it holds past data in an effort to “learn”. Whereas the 32kB of SRAM is just temporary short-term memory. It is where is will hold an image from the pixy cam to process it, only for it to be erased and have a new image pop in a moment later.

Because we want the SPARC to remember recurring obstacles, we need the large Flash Memory size. And because we need a place to hold pictures for processing, we need the large SRAM size. These factors, coupled with the 48MHz processing speed are what made us decide on this specific chip to use in our SPARC project.

### 5.3.3 Voltage Regulators

The voltage regulators in this circuit are the devices that allow us to take the 14.8V output from the batteries and dial it down to the voltages we need for devices. In our circuit we use 4 different voltage regulators, all from the same series.

To start off with we have three of the BAJ6DD0WHFP chip in the Home Base. While the home base will be discussed in further detail below in Section 5.4, it does contain a voltage regulator of the same series as the main SPARC unit.The Voltage regulator in the Home Base takes an 18V input from a wall charger, and converts it down to 16V, which is what our Battery Charging Protection Devices require.

On the main SPARC Circuit Board, there are three different voltage regulators to provide the different voltages needed to power the specific devices on the board. In our voltage regulation array, we have a total of 13 voltage regulators, all from the BAxxDD0WHFP chip family.

The main regulator chip that we use on the board is the BAJ2DD0WHFP. This particular regulator lowers the 14.8V output from the battery bank to 12V. Out of the thirteen voltage regulators on the board, eleven of them are the BAJ2DD0WHFP. The reason for this is because each regulator has a maximum output of 2A. Since each motor has a Stall Current of 10A, we needed ten regulators to allow the maximum current of 20A that could be pulled. Since we didn’t want to actually push the Absolute Maximum Current draw of the chips, we added an eleventh regulator to provide a bit of a buffer. One of the last things that we would want is to damage a chip during testing or operation, which would force extra current through the remaining chips, thus damaging more chips in a chain reaction effect.

The next chip on the board is the BA50DD0WHFP. We have one of these chips install to provide the 5.0V needed to power the USB Port to the Pixy Camera and the Lidar proximity detection device. Since there are only 2 devices, neither of which will draw a lot of current, a single regulator is all that was required for this purpose.

The last regulator chip is the BA33DD0WHFP. This chip provides the 3.3V source for the low power devices such as the RF transmitters, the Microcontroller, the HC-06 Bluetooth Module, and the GPS Module. All of these devices should have almost negligible current draw, and thus again, only one 2A regulator was needed.

Schematic 03 on the next page shows the Voltage Regulator Array that powers the devices on the SPARC PCB.

REPLACE PAGE WITH

“Schematic 03 - voltage regulation.pdf”

AFTER PDF’ing

### 5.3.4 Motor Controllers

The Motor Controllers were another one of the early design choices. To preface this section, it should be noted that to run a motor, a voltage needs to be applied the two terminals. Running a motor in reverse is as simple as reversing the polarity across the two terminals.

That is where the need to Motor Controllers come in. There isn’t really an intuitive Solution to reversing the polarity on a hardwired motor. The method needed to do this involves using an H-Bridge. An H-Bridge is a device usually made up of MOSFET’s that, given an input signal, can reverse to polarity of the outputs. An H-Bridge can also be made of switches in a more hardwired, non-chip, application. Picture 10 shows the two states of a simple H-Bridge.

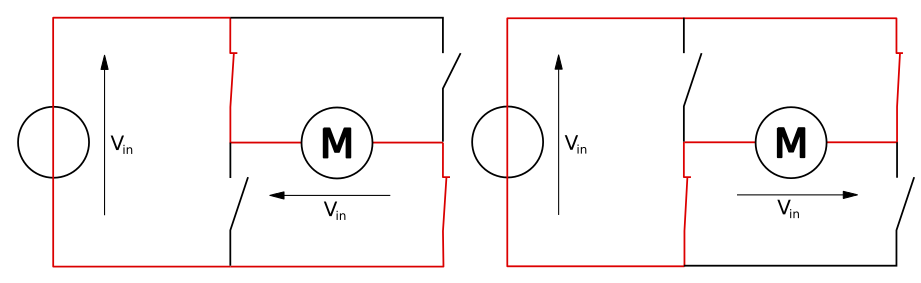


Figure 10 H-Bridge Representation

The Motor Controller that we selected for this project is the MC33HB2001 made by NXP. Figure XX below shows the internal block Diagram of the MC33HB2001 Motor Controller. We picked this chip specifically because it can handle high amperage situations. It is rated for 7A of continuous Input/Output while being able to reach 10A peaks as it’s Absolute Maximum Current Output, which lines up with the motor’s Stall Current.

The MC33HB2001 is able to report a fault status bit, and shut off output control. There is also a feature for some basic data reporting using an SLI interface, however we are not planning on using this feature.

Schematic 04 on the next page shows the connections for the two MC33HB2001 devices.

REPLACE PAGE WITH

“Schematic 04 - motor controllers.pdf”

AFTER PDF’ing

### 5.3.5 Bluetooth Module

The transmission protocol most suited for this device came down to using a Bluetooth connection. Bluetooth as a wireless technology standard made the most sense for the SPARC device and prototype because it is used in a close range and is rather simple to integrate into the device’s code base. Because we would like the user to interface and control the SPARC as easily as possible any user with a smart phone already has Bluetooth connectivity on their mobile devices and purchasing an external bluetooth module for the SPARC hardware would be a small simple addition.

The Bluetooth module that we decided on is the Bluetooth Transceiver Module HC-06. This module uses the Bluetooth 2.0 standard and has a transmit time of 0.5 seconds at 2.5 GHz ISM frequency band. Bluetooth has a range of approximately 100 meters which is a good enough range for controlling the SPARC from the garage to the curb from within your home. This module uses a serial interface that connects to the microcontroller UART serial communication protocol. The HC06 receives data in from the devices TX pin and transmits it and sends out data to the devices receive pin that it is transmitted.

The simple interface of the HC06 made it ideal to work with fro the mobile application. The mobile application will be using the Bluetooth module exclusively to send user inputted data to the SPARC.

### 5.3.6 GPS Module

Throughout this project, the decision on whether or not to include GPS has always been in contention. We have gone back and forth between GPS, RFID, or relying solely on Image recognition for the unit to know where it is.

GPS and RFID both have their own advantages and disadvantages, in terms of accuracy and ease of use. While the RFID has its merits, GPS is the superior choice when it comes to terms of ease of use. All that needs to be done is to program a location, and that’s all there is. RFID on the other hand did have the potential to be more accurate, however it involves placing a large number of markers at the boundaries of the operational area, which can be unsightly, a hassle to set up, and a nuisance every time a person goes to mow the lawn.

Because of this we decided that GPS was the more preferred method, as it requires less setup by the user. The specific chip we chose, the PAM-7Q-0 has an accuracy of about 2 meters, a built-in antenna, and an update rate of 10Hz.

### 5.3.7 Pixy Camera

The Pixy CMUcam5 is an open source embedded color detection visual sensor produced by Charmed Labs. The pixy frees up much of the system resources for the SPARC by processing of all of the visual data on its own on-board embedded system and returns the relevant data as digital input to the SPARC's own embedded system. The development an algorithm to process visual information by the SPARC team, as well as the cost of purchasing an independent camera to gather such information made utilizing the Pixy a very beneficial module to include in the SPARC’s design. The Pixy supports a wide variety of interfaces including UART, SPI, I2C, and USB; these features made the Pixy a very attractive module to include in even the initial designs of the SPARC. Below is a image of the Pixy's board, along with its components.

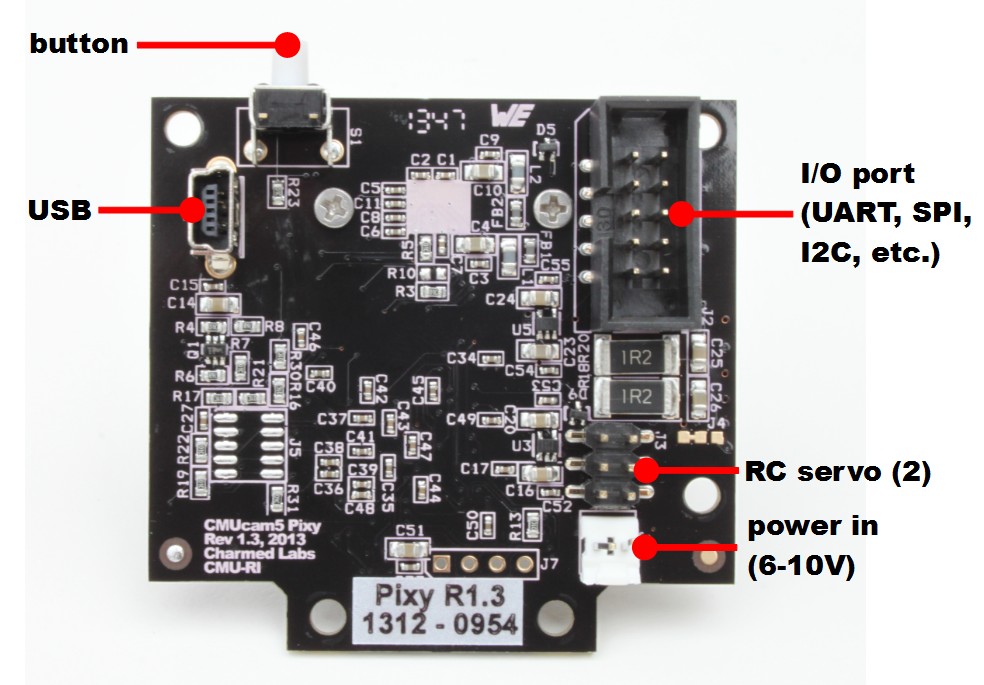


Figure 11 Pixy CMUcam5 Board

*Permission Pending from Charmed Labs*

The Pixy utilizes a color filtering algorithm that allows it to discern from up to seven different pre-programmed color signatures. The SPARC utilizes this to recognize grass as a 'difficult terrain', which the SPARC will have a negative preference for advancing into when other directions do not traverse through grass. This both allows for a more efficient expenditure of power by the SPARC from avoidant driving through this rough terrain, and protects the user's lawn from being trampled needlessly.

The Pixy can also additionally detect the presence of several unique 'color codes' as seen in the figure below, which can be marked on important objects to make them distinct from the detected colors in the area.



Figure 12 Pixy Color Code Detection

*Permission Pending from Charmed Labs*

The SPARC utilizes such a code on its home base, allowing it to be located easily and uniquely by the Pixy module. Utilizing color codes substituted the use of Image Recognition for the home base as a redundancy, and removed any necessity for the SPARC to have an algorithm to recognize the home base system in this way.

Schematic 05 on the following page shows the connection for the Bluetooth Module, the Pixy Camera, and the GPS Module.

REPLACE PAGE WITH

“Schematic 05 - pc-bt-gps connection.pdf”

AFTER PDF’ing

### 5.3.8 RF Transmitters

The RF Transmitters built into the Main SPARC PCB work in tandem with the RF Receivers that are built into the Home Base PCB that will be talked about in Section 5.4.2. There are two RF Transmitters that have been incorporated into our design. One is a 313MHz Transmitter while the other one is a 433MHz Transmitter.

These devices generate their signal using a crystal Oscillator tied to Pins 4 and 5. The chip takes the signal from the crystal, and amplifies it up to a working wavelength, which it then output through an antenna Module on the board.

As mentioned, we have two of these Transmitters. These transmitters are going to operate as the signal to allow power to flow from the Home Base to the Main unit, and the signal to disconnect power as well. We didn’t want to rely on having 1 transmitter do both the enable and disable in a toggle-style operation, because if there was a spotty signal, the circuit could not react while the MCU says “I sent signal, everything should be good.”

With two transmitters, we can send one long signal as an enable command, ensuring that the Home Base receives it, and same goes for the Disconnect signal as well. We just want to be sure that the signal is relayed properly, to avoid any possible ground faults that may occur due to rolling off the pad with power still flowing.

While these devices should be able to communicate a fair distance apart, these devices will only be operated while the SPARC is on the Home base, so there will be a distance of less than 1 foot between them, so communication reliability should not be an issue.

Schematic 06 on the next page shows the connections between the oscillators and the RF Transmitter Amplifiers in the Main PCB

REPLACE PAGE WITH

“Schematic 06 - rf transmitters.pdf”

AFTER PDF’ing

## 5.4 Home Base Design

Since the idea for this project came about, we wanted the device to be able to charge itself autonomously. For this to happen, the device needed a charging dock of some kind. That is where the idea of our “Home Base” comes into play.

The Home Base plugs into the wall, receives a signal from the main SPARC unit to enable charging, it regulates power from the wall, and it pushes power into the main SPARC unit to recharge the Battery Bank array. The Home Base is an important step in our design because it gives the SPARC unit a place to return to during it’s standby mode. The end-user can place this anywhere, and if initially set up properly, the Sparc unit should be able to find the Home base on it’s own by remembering where it came from.

### 5.4.1 Charging Circuit

Inside the Home Base is a circuit that sends power from a normal 120V AC Wall outlet to the SPARC Unit in a format that it is able to use to recharge the Battery Bank. The First step in this circuit is a standard AC to DC Wall Adapter for laptops. The Specific Adapter we chose has an output of 12V DC at 5 Amps.

The connector we used for the wall adapter is Part Number 694103107102 by Wurth Electronics. It is a standard 1.05mm DC Barrel Connector. This is a common adapter for homemade devices that plug into the wall.

This power is sent through 3.3V regulator to power the RF receivers and an SR Latch, both of which will be talked about in the next section. Also the power form the wall is routed through three 16V Voltage Regulators, Specifically the BAJ6DD0WHFP devices talked about in the Voltage Regulator section previously. These regulators are controlled by the outputs from the RF receivers. After the regulators, the power rail is attached to the Wireless charging system that will charge the main unit through metal on metal contact.

Schematic 07 on the next page shows the charging circuit in the home base, with the exception of the RF receivers, as those were too detailed to include on the same schematic as everything else.

REPLACE PAGE WITH

“schematic 07 - home base charging circuit.pdf”

AFTER PDF’ing

### 5.4.2 RF Receivers

The RF Receivers, coupled with the SR Nor Latch, in the Home Base are the closest thing to a brain the Home Base has. While not having a processor, the SR Nor Latch holds its state depending on the output of the RF Receivers.

Just like how there are two Transmitters on the Main SPARC Unit, there are two receivers in the Home Base. One for 313MHz, and one for 433 MHz. Both of these devices output their signals to an SR Nor Latch, as the enable and disable signals respectively.

When the Main Unit sends a signal, the SR Nor Latch will take it and hold a state depending on which signal was received. If the 433MHz signal is received, the SR Nor Latch will generate and hold an output state of 0, which will disable the 16V Regulators, and shutting off the 16V output from the Home Base, preventing charging to the SPARC unit.

However, if a signal comes in on the 313 MHz receiver, then the SR Nor Latch will generate and hold an output of 1, enabling the charging circuit, and allowing the battery bank to charge. This is a very important safety mechanism, because the risk of a ground fault occurring form the 16V charging rail could severely cripple the internal systems of both the Home Base and the main SPARC unit.

When building a project using this level of voltage and amperage capacity, safety is a serious concern. While 12V at 20A may not be able to kill a person, it definitely has the potential to inflict a great amount of pain if handled improperly. This is why we need to be sure that the Home Base is deactivated when not in use, as we don’t want any people or animals getting injured by coming into contact with the positive and ground terminals while the SPARC is doing its routine away from the Home Base.

REPLACE PAGE WITH

“schematic 08 - rf receivers.pdf”

AFTER PDF’ing

5.4.3 Wireless Conduction Charging

For the Home Base to work effectively, we needed to use a cordless method of charging, as autonomously finding, lining up, and plugging in a port does not seem like a feasible task given our current knowledge, resources, and expertise. This left wireless charging as the only viable solutions.

There are two main forms of wireless charging, inductive and conductive. Conductive has been commonplace in the consumer market for at least a decade now, as it is simpler. Conductive Wireless charging is as simple as making a metal contact touch another metal contact. A common example over the last decade is rechargeable battery packs. A battery pack has external contacts that line up with contacts on a charging dock, and you just drop the pack in. The metal contacts touch, and charging commences. The advantage of this method is quicker charging than that of inductive charging. The disadvantage however is that the contacts need to line up perfectly, along with the possibility of exposed metal contacts corroding, preventing power transfer.

The other form of wireless charging, that has gained popularity in the last 2 years, is inductive charging. This method has become more commonplace in the most recent models of cellular devices. It involves the transfer of power using electromagnetism instead of metal contact. The main premise is that AC current moves through a coil on the charging pad, which creates a shifting electromagnetic field. This Electromagnetic field then pushes current through a reciever coil on the device, creating AC current on the device. While the devices don’t necessarily have to touch, the efficiency of the devices drops off very quickly as the distance is increased.

If going from a wall outlet to DC power, an AC to DC converter is needed at some point in the circuit. For inductive charging, an AC to DC converter is needed after the wireless power transfer, while in conductive charging, the AC to DC converter is needed before the transfer of power.

In our project we chose conductive transfer back in the time where we were still planning on using Solar Power. Since Solar Panels already have an output in DC Voltage, it would have required a DC to AC converter betfore the transfer, along with an AC to DC afterwards. So in the case of Solar we would have needed 2 converters instead of zero, so it made more sense to use conductive. Once we moved away form solar, however, the idea stuck around, as we had no reason to revisit the charging process.

## 5.5 PCB Design

The Printed Circuit Board was an area that none of us had any experience in before this project. Between trace widths, clearances, layers, and mounting styles, none of us had any idea what to do. Luckily the internet has many great examples, and tips.

We decided to design the PCB using EasyEDA.com instead of Eagle, as EasyEDA has a much larger built-in library, and a much larger collection of user submitted libraries to help find devices that are not built in. Even with all this though, there were some devices which we simply could not find in either program. This required us to take an existing chip of a similar package, and rename pins, and double check pin spacing.

Picture 13 shows the top side of the PCB that was designed for the Home Base, while Picture 14 shows the bottom side of the same board. As can be seen, we only used the bottom layer for traces that were absolutely necessary for routing.

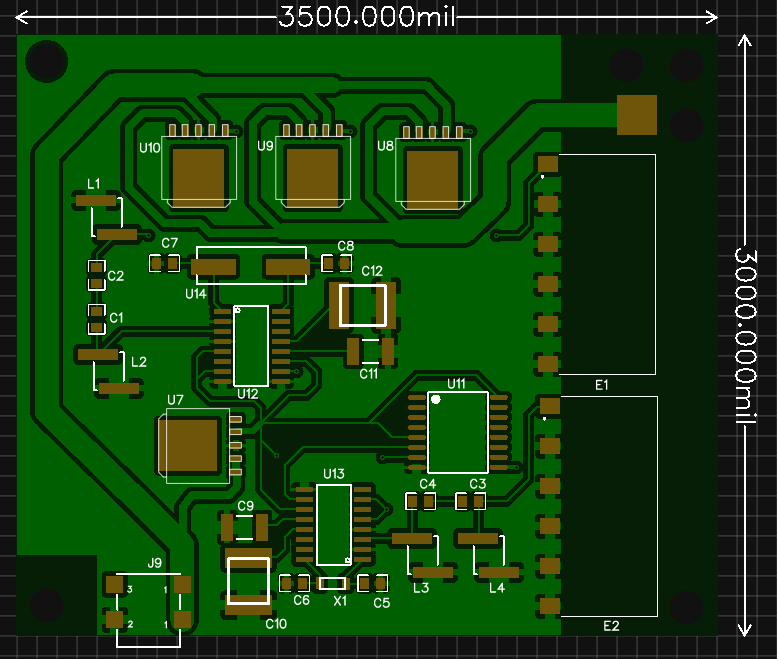


Figure 13 Home Base PC Top

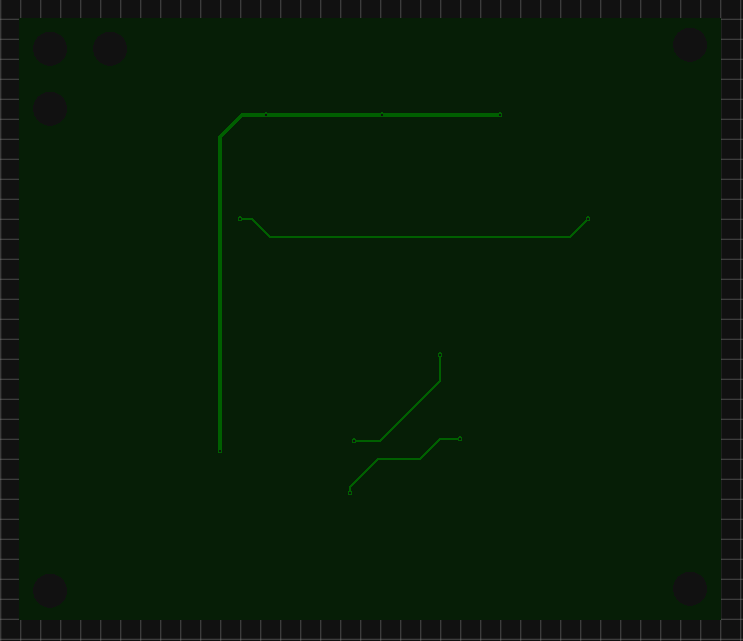


Figure 14 Home Base PCB Bottom

For simplicity sake, we felt as though keeping as many devices and traces on one side of the board as possible was the right thing to do. It should make testing and troubleshooting a lot easier in the long run too.

Pictures 15 and 16 on the next two pages shows the top and bottom layers of the main PCB in the SPARC Unit itself. It was very difficult figuring out the best way to route a 20A current through this board effectively, and in the end it came down to using single large traces, or multiple small ones. We ultimately chose single large ones just to make it simpler to follow.

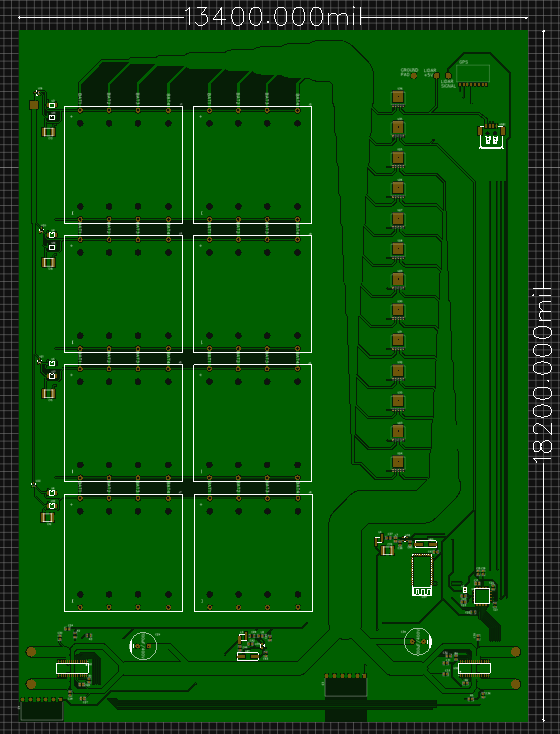


Figure 15 Main PCB Top Layer

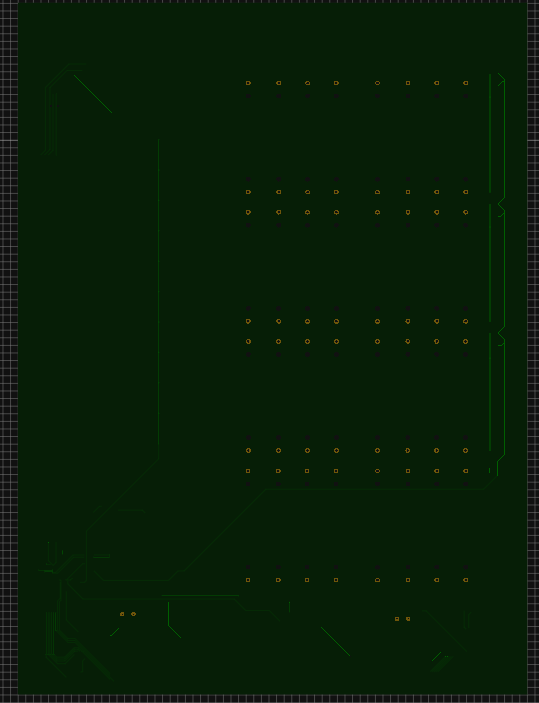


Figure 16 Main PCB Bottom Layer

# 6.0 Software Design

In order to interface with and control the SPARC, function software will be needed. The autonomous movement, vision and its over all ability to completely the basic task the SPARC is built for is directly depended on the device software. The following sections will go over every aspect of this crucial and integral part of the SPARC device.

# 6.1 Software Overview

The other main component of this project is all of the software needed to control this device. There are three main parts of the overall software of the SPARC device. Firstly, the most low level software of the SPARC is the programming of the microcontroller which will serve as the main processing unit of the SPARC. All of the basic movement control functions will be programmed into the microcontroller receiving input either from the path algorithm for autonomous movement or from the mobile application via Bluetooth for remote controlled movement. The next software aspect is the mobile application it self. This will serve as the main port for the user experience of the SPARC device. The user interface will be intuitive and easy to use to maximize user satisfaction. Lastly the most computationally intensive software component will be the pathing algorithm. This will be the brains from the autonomous control of the SPARC device.

The figure below is a use case diagram for the SPARC’s different software components. The user never interfaces with the algorithm or embedded code directly; all user input is delivered to the system through the Android Mobile Application.



Figure 17 Software Use Case Diagram

## 6.2 Microcontroller Programming

The microcontroller will be programmed using C language, which was one of the deciding factors for choosing our specific microcontroller, the ATSAMD21J18A-AUT, due to our familiarity with coding in C for microprocessors. The microcontroller will be receiving a variety of sensor inputs as well as input from a mobile app for manual controls through a Bluetooth module, and these inputs will be accounted for through the programming of the microcontroller.

That familiarity, however, has been gained using one specific model of microcontroller. We are most familiar with programming using the Texas Instruments MSP430 model, but we opted for the microcontroller that is on the Arduino Zero board. This change of platforms will require some adjustment, but the basic skills and concepts required for programming a microcontroller will likely carry over to the unfamiliar platform. Additionally, the benefit of using an Arduino microcontroller is that it is very beginner-friendly and has a large hobbyist community supporting it.

The Arduino community has a large collection of libraries, tutorials, and guides that will be utilized to make the software design process as smooth as possible. Along with standard libraries, there are other additional libraries that are controller-specific that can be used to perform more specific tasks on this specific microcontroller. There are also contributed libraries created by the community that are available for public use, along with numerous project tutorials that may provide explanations or good-practice methods for programming.

The SPARC unit’s motion is created by supplying power to the left and right tracks for a specific interval of time. This is controlled via signals given by the microcontroller that are based on both sensor and mobile app input. Additionally, the microcontroller will be constantly checking the sensors to determine if it can continue its designated path or if it needs to reroute its path to avoid an obstacle. These factors will be addressed using a combination of movement control functions, input control functions, and output control functions.

### 6.2.1 Movement Control Functions

The movement control functions encompass the core movements the SPARC device will use to traverse its path and correct its course. The device will move forward by having one motor run clockwise and the other anticlockwise, which will cause the tracks to move in the same direction. The device will reverse by running the motors in the opposite direction of the forward movement function. Due to the nature of using timings to determine how long a motor should run, the times for each track may need to be different to ensure straight motion (Figure 18).

The rotational motion for turning the SPARC unit can be done by having the tracks moving in opposite directions to each other, but having the motors run in the same direction. Turning left will require the left and right motors to rotate anticlockwise, turning the unit in place. Turning right would then be done by doing the opposite; signaling the motors to rotate clockwise (Figure 18).

The pseudocode shown in Figure18 was developed with the intention of using the C programming language. The functions themselves do not necessarily require an input parameter, since the movement programming will call these functions to execute as needed and send the appropriate signal to each motor controller to perform the movement action. This will allow for safer operation, as the program will gather input from the sensors or modules before calling the movement functions.

These movement control functions drive the core functionality of the SPARC unit, namely its ability to move itself to the curb and back. However, these functions alone can’t move the SPARC device reliably from point A to point B. Therefore, movement control will also rely heavily on the inputs it receives to traverse the path from the SPARC base to the curb on trash day.

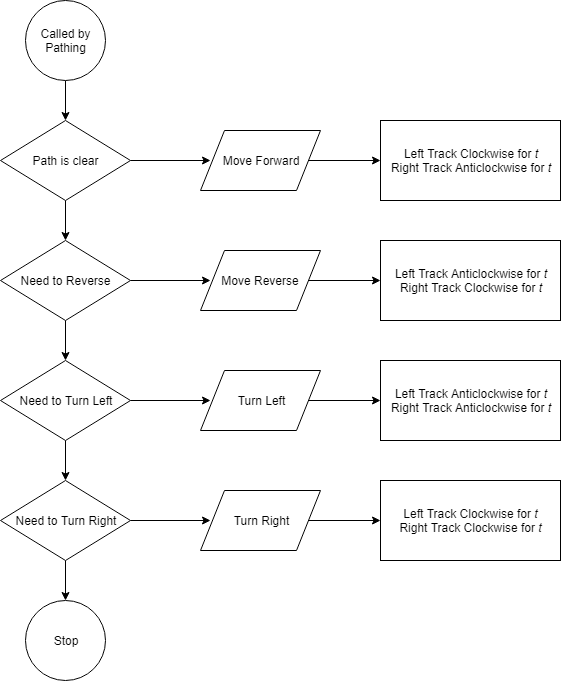


Figure 18 Movement Functions Flowchart

### 

### 6.2.2 Input Control

The input control for the device dictates what the SPARC will do based on what inputs it receives from its modules and sensors. The goal is for the unit to make it to the curb with the trashcan and back without having any collisions, so therefore the input control programming will need to know how to handle its sensor and module inputs and where to send them.

In the event of a signal from the sensors that indicates an imminent collision, the program will first ensure that the SPARC is not moving by triggering an interrupt condition to halt the unit’s movement. Once the unit has stopped moving, the program can begin pathing around the obstacle until it gets past it (Figure 19).

Another possible input condition is the case that the unit has reached its destination at the curb. Using a combination of inputs, the device will need to ensure that it is both close enough to the curb and that it has the correct orientation for garbage disposal. Upon detecting the curb, the SPARC device will begin a reorientation subroutine consisting of a combination of movement functions to determine its proximity to the desired location as well as determine the correct orientation before going into a standby mode until after garbage has been collected (Figure 19).

Similar to the condition that it reaches the curb, the SPARC also needs to know what to do when it reaches its home base on the return trip. Upon detection of the home base, the device will perform a subroutine to establish the correct orientation. Then, once the orientation has been established, the SPARC unit will then position itself on the home base to recharge and collect trash until the next garbage day (Figure 19).

These two conditions, reaching the curb and reaching the SPARC base, are key factors required for the device to function properly. If the SPARC is misaligned on the base, it could get shorted out or not be in the correct position to charge. If this happens, the device will lose power and no longer operate. Additionally, if the SPARC encounters a problem at the curb, it may either not get picked up by the garbage truck due to poor orientation, run off the curb and into the street, or topple over if it goes too far over the curb. This highlights the importance of correct input control.

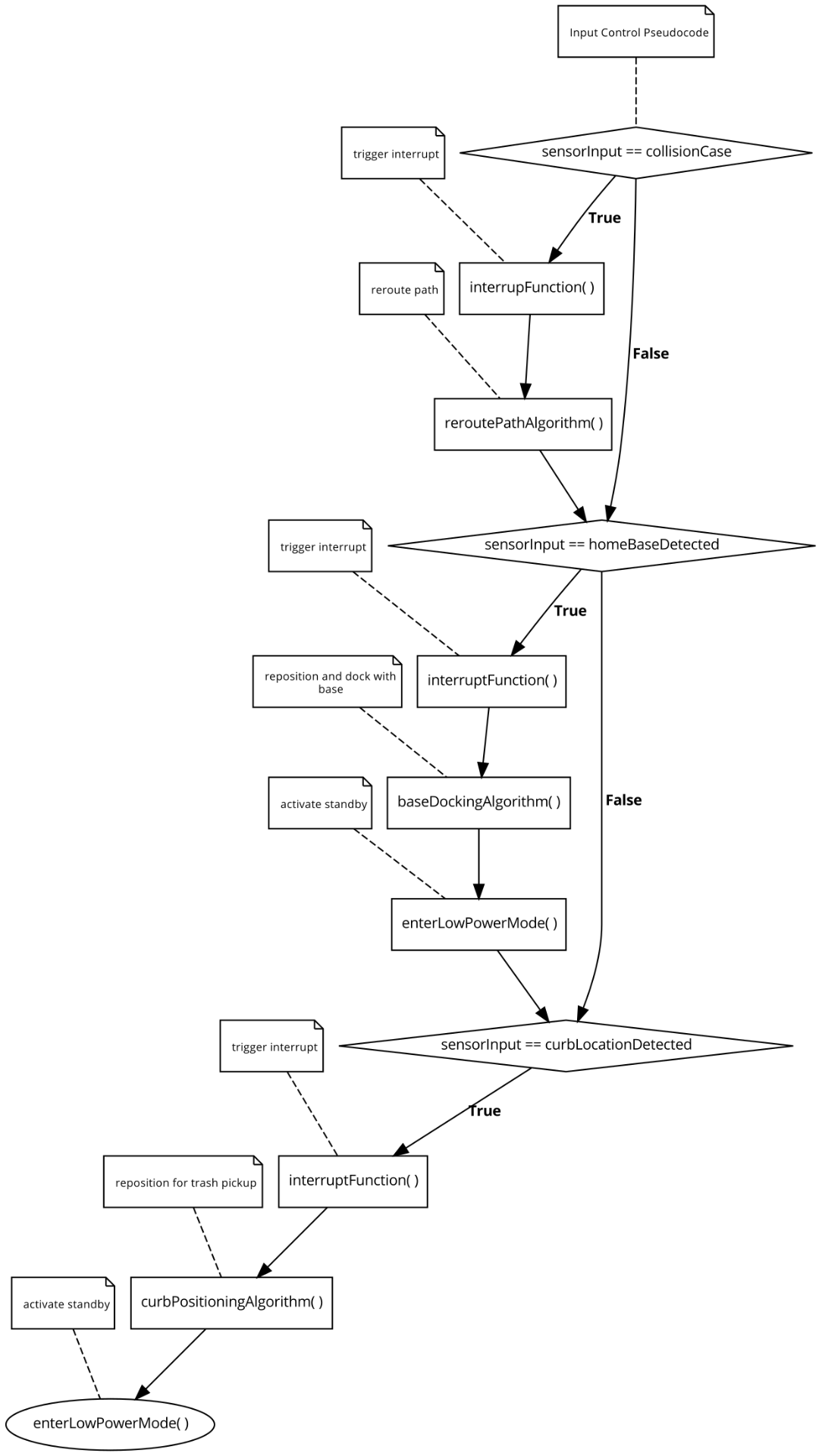


Figure 19 Input Control Flowchart

### 6.2.3 Output Control

The output control for the SPARC device will primarily serve as the structure for when the device will emit a signal or provide feedback via the app. This feature will allow for higher accuracy when determining the location of the device as well as allow it to handle the scheduling for when it is supposed to move to the curb. The output control will send status updates to the user as it performs its tasks and will work in tandem to the input control in order to output the correct responses to specific inputs.

On a predetermined interval, the SPARC device will check the current date and time. If this time matches with the scheduled time given by the user, the device will send a message to the user through the mobile app that it is beginning the trip to the curb. This will then link back to the input control to determine what function it needs to perform in order to properly navigate to the curb, where the output control will then use the Pixy Cam to detect a colored marker at the curb that will detect the proper location and orientation once it gets close enough (Figure20).

As an additional safeguard, the GPS module will check to make sure that the SPARC device is in the proper location for curbside garbage collection. If the GPS detects that the SPARC device is not in the correct location, it will continue to route until it reaches the correct location. If the GPS check result comes back positive, the output control will notify the user through the mobile app that it has successfully reached the curb and that it is ready for garbage collection. Finally, the SPARC device will enter a low power mode to conserve power (Figure 21).

Much like for the trip to the curb, the output control checks the current date and time against the scheduled return time provided by the user via the mobile app. When the times match, the user is sent a notification through the mobile app that the SPARC device is returning to the home base and the device begins its curb-to-base pathing algorithm. Once the device reaches the home base, it begins its docking algorithm using the Pixy Cam and a colored marker on the base for guidance (Figure 22).

After the unit executes the functions needed to dock with the home base, the SPARC will check its location using the GPS module. If the GPS location check results in a failure, the device will continue trying to path to the home base. If the GPS location check results in a success, the output control will notify the user that that device has successfully docked with its home base and that the SPARC unit is recharging. Finally, the SPARC device will enter a low power mode and charge while on the home base (Figure 23).

Additionally, the output control will need to have a copy of the user’s desired schedule for garbage collection. The output control’s scheduling function would establish a schedule through the accompanying mobile app using the Bluetooth module. This scheduling function will prompt the user for a desired time to go to the curb and a desired time to return to the base through the mobile app using the Bluetooth module. This information is used to set the departure and return times that will be checked against the current time, which determines the active times of the SPARC device (Figure 24).

Additionally, the GPS module can be used to ensure that the SPARC device remains in its proper operational area in case it encounters a problem in its pathing. If the device detects that it is out of its operational area, it will enter a while-loop that will constantly check its current location using the GPS module. If it is still out of the operation area, it will attempt to notify the user using the Bluetooth module and mobile app. It will then have a delay before it checks its location again and restarts the loop. If the GPS detects that it has returned to the operation area while in the while-loop, it will notify the user that it has returned and will continue its pathing as normal (Figure 25).

The output control will also need to handle cases where there is an obstacle in the path that needs to be avoided. The output control will get information from the input control in the event that the LIDAR sensor detects an obstacle in the path of the SPARC device. If the information does not match a collision case, the device will continue its pathing algorithm as normal. If the information does match a collision case, the output control will send a notification to the user stating that it has encountered an obstacle and is rerouting. The SPARC device will then begin a rerouting algorithm to get past the obstruction and return to its path, notifying the user through the mobile app once it has done so. Finally, the output control will message the user stating that it is resuming its path to its destination (Figure 26).

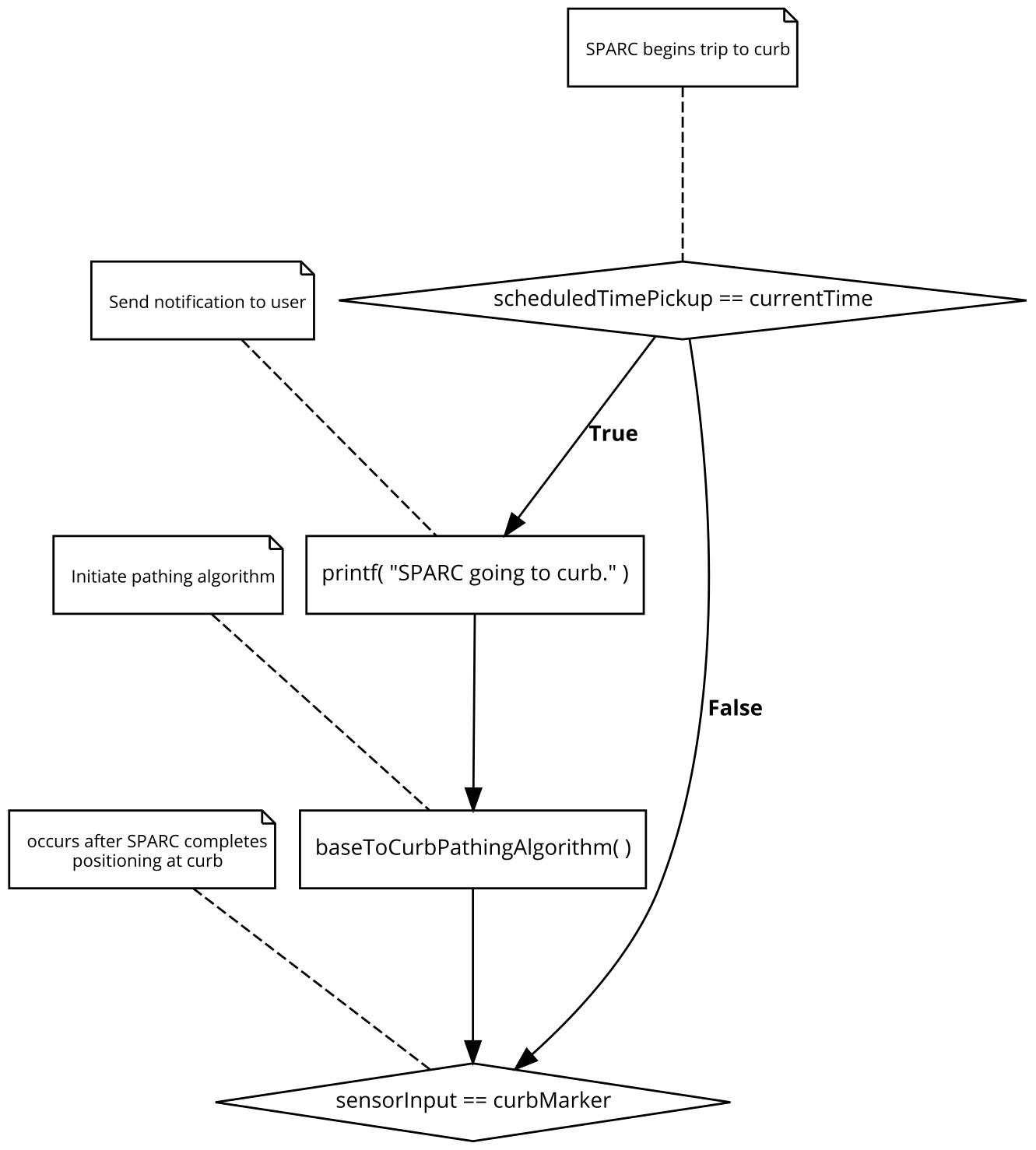


Figure 20 Output Control Flowchart 1

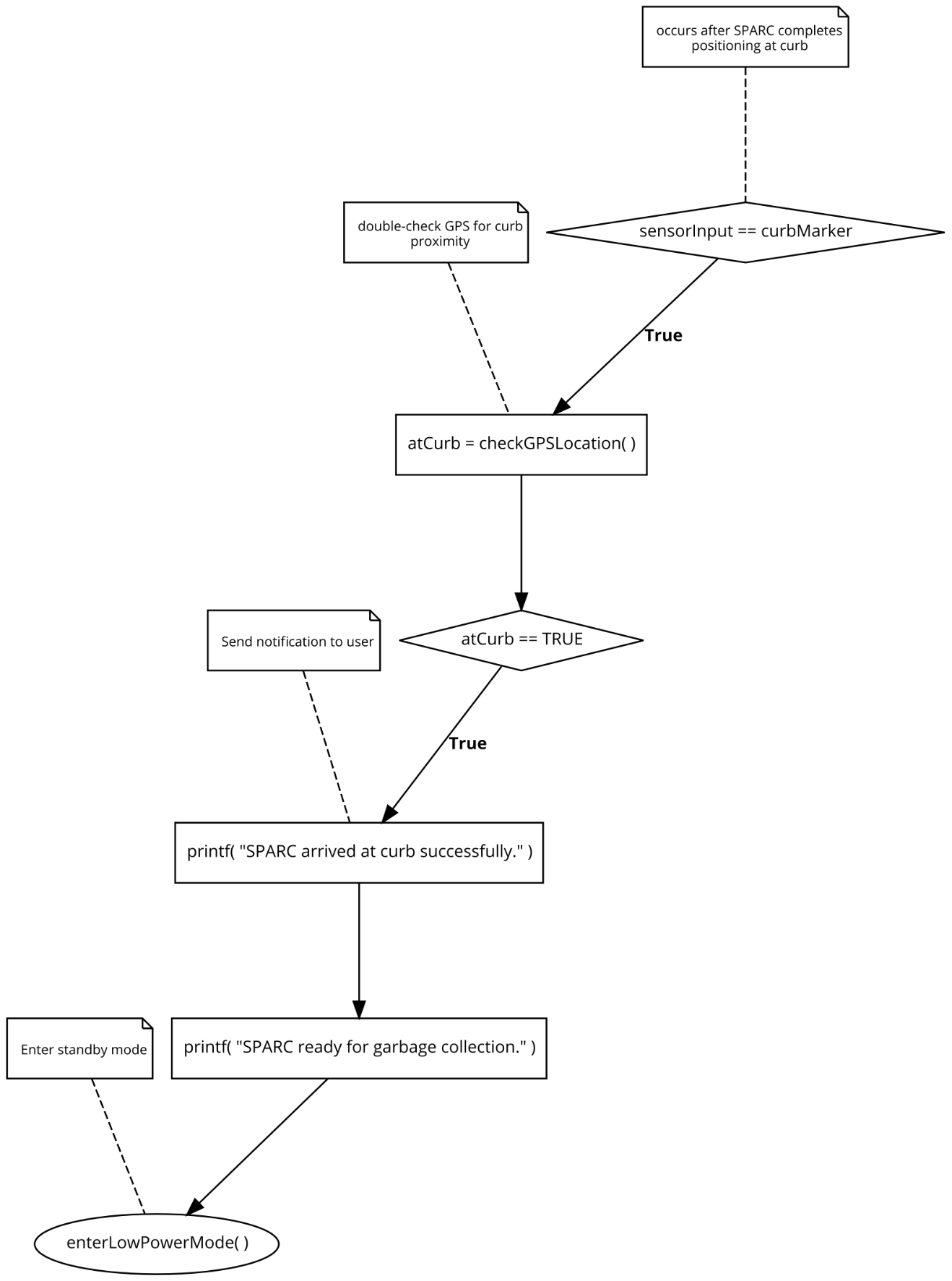


Figure 21 Output Control Flowchart 2

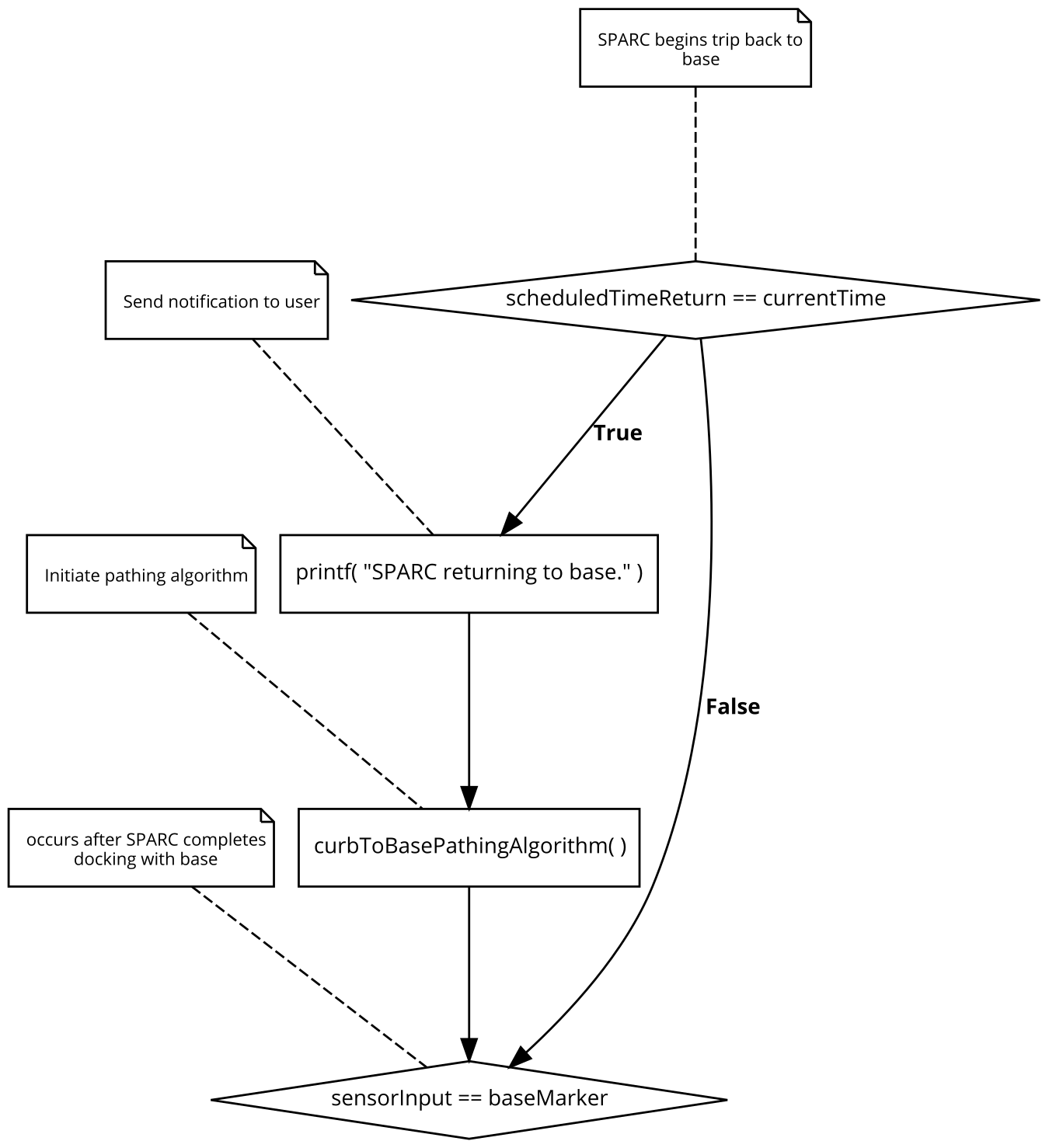


Figure 22 Output Control Flowchart 3

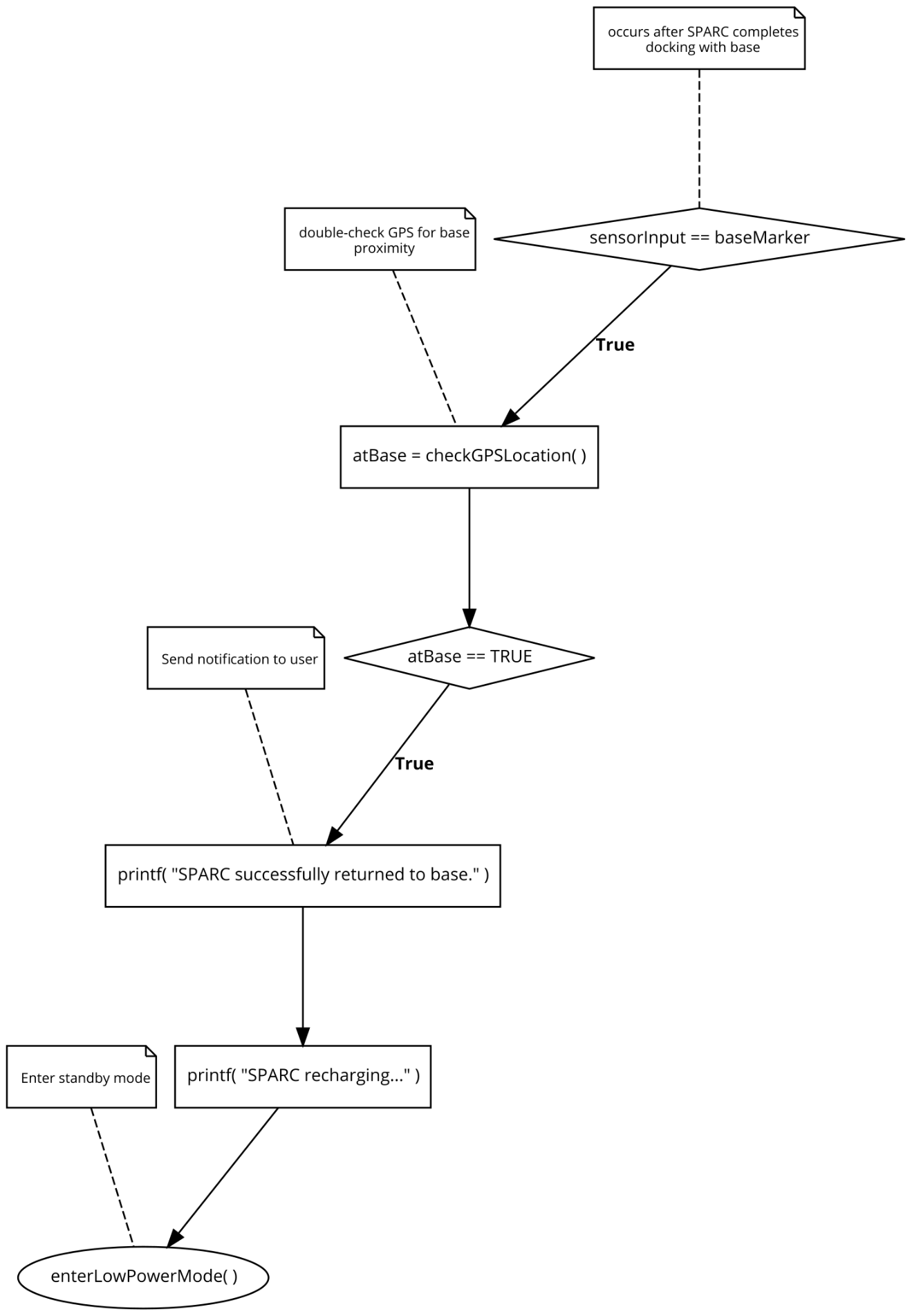


Figure 23 Output Control Flowchart 4

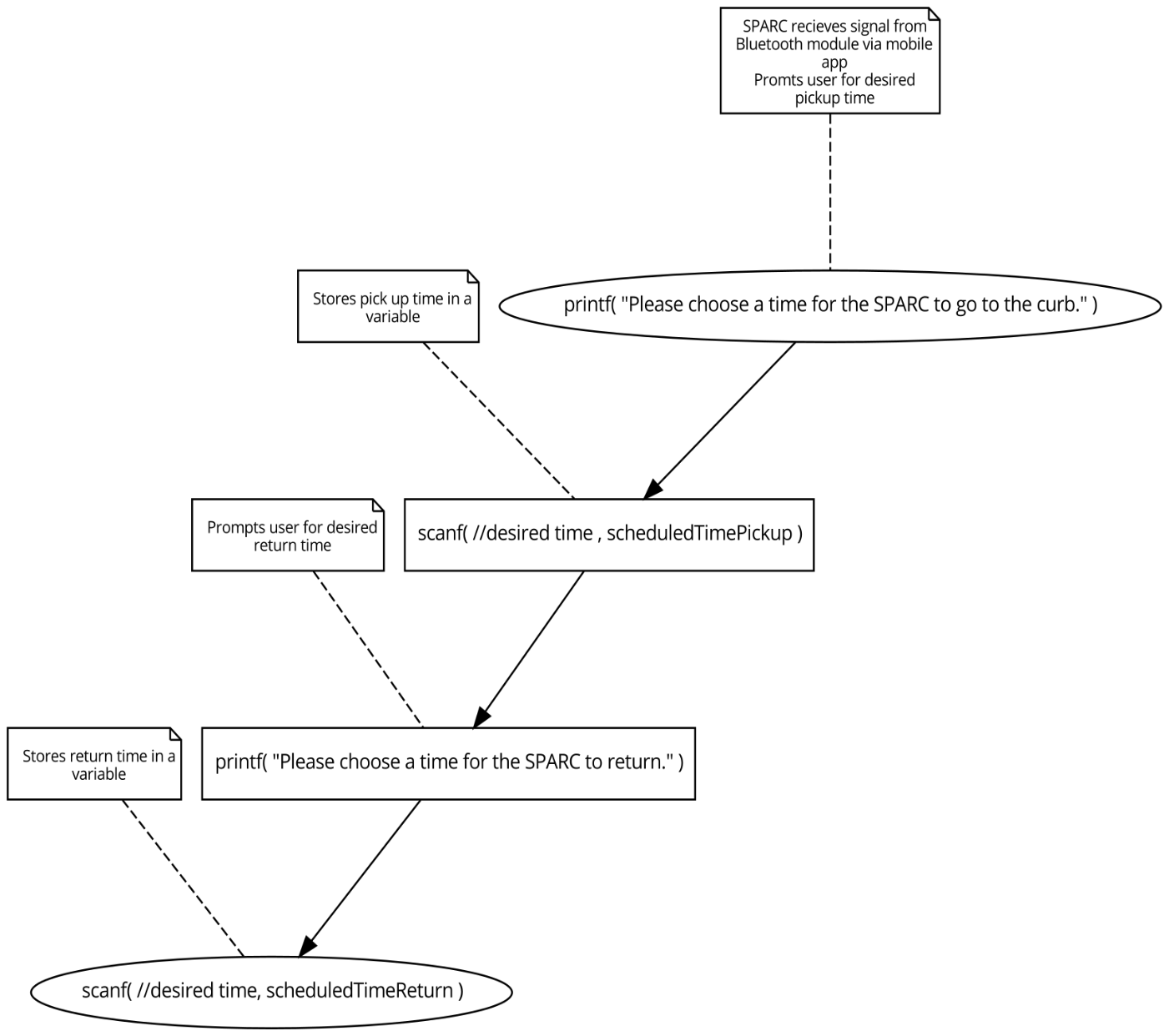


Figure 24 Output Control Flowchart 5

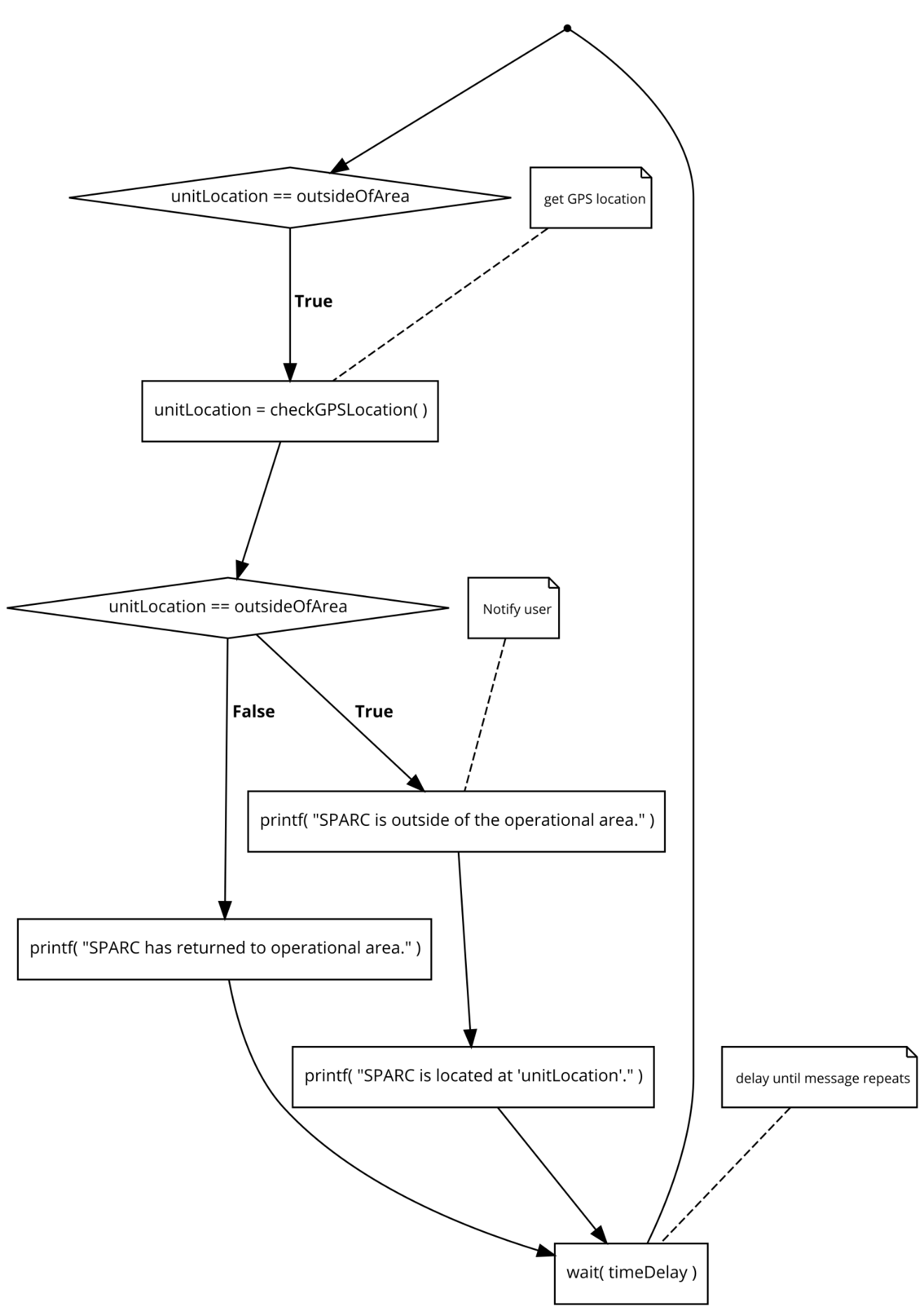


Figure 25 Output Control Flowchart 6

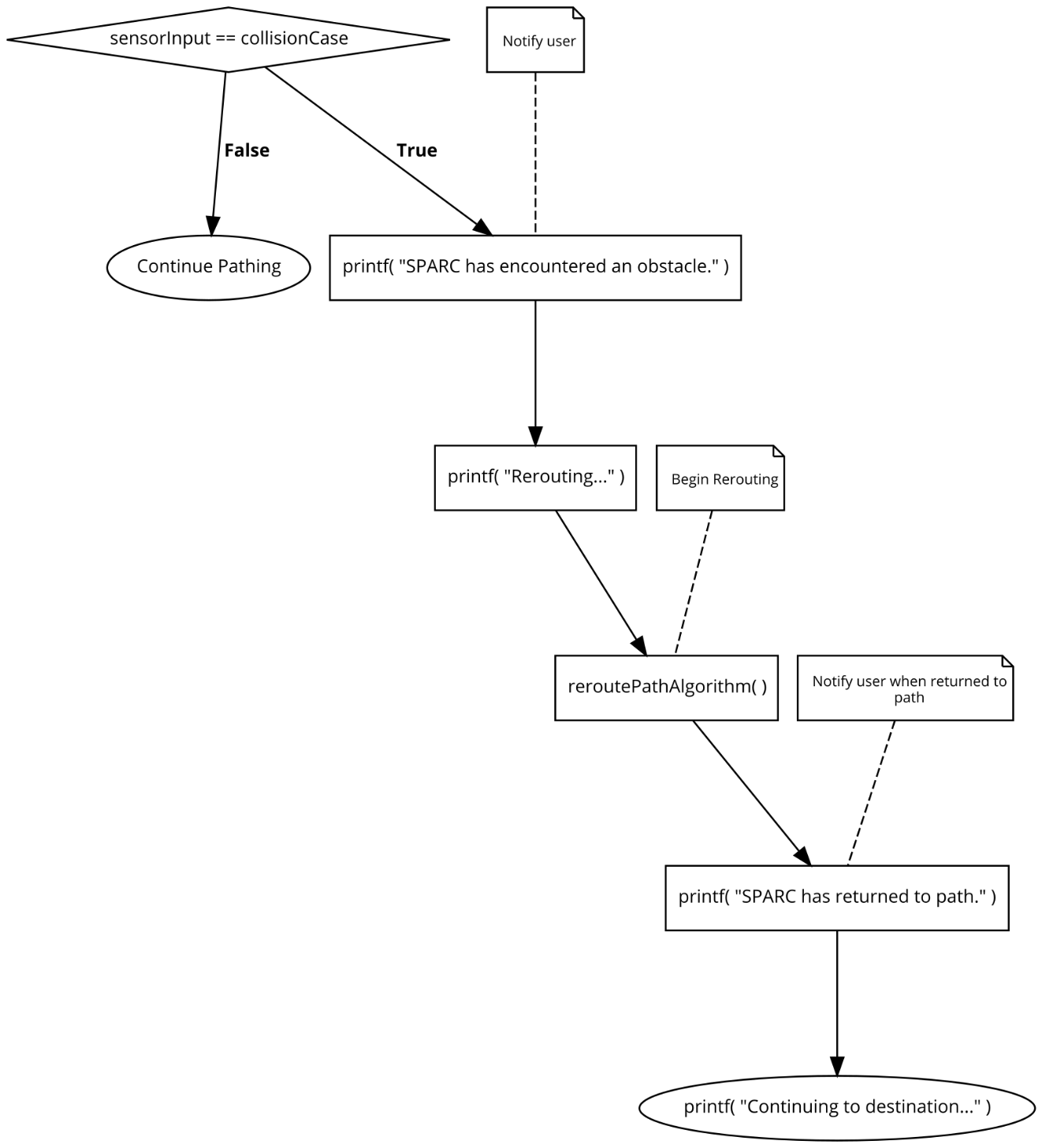


Figure 26 Output Control Flowchart 7

## 6.3 The Mobile Application

The user experience with the SPARC remains one of the number on priorities in the construction of this product. The main haven for this user experience will be the mobile application. Users will need an easy way to “talk to”, monitor and control the SPARC. The Mobile Application will be that main connection between User and Device. The application will be a simple Android application with options for the user to manually control the SPARC, to schedule the service in time intervals, and to view its GPS location. The application will also be vital in the initial setup of the SPARC.

### 6.3.1 Mobile Application Diagrams

When designing software, to ease the process, multiple UML diagrams are made for control flow and software organization. *Figure 23 Software Use Case Diagram* is the high-level use case diagram for the mobile application. This diagram demonstrates the basic control flow for the mobile application that will aid the development process. Every action the users will make on the application is a case and the diagram demonstrates the possible outcomes of each task.

When the user downloads and opens the application, they will be prompted to sign in or to create an account. The Log in page will ask users to input a previously registered email and password. This page will check the validity of the email and password which will be saved on our application server. If the email and password correctly match an existing user account, that user’s settings will be loaded into the application. If the User does not have an account they will be prompted o create a new one. When creating a new account, the Users name and a valid email address is asked for and they are prompted to give a password with a minimum length of 4 and max of 10. The email address is asked for in order to create unique user accounts on the Application server.

Once the new account is created and verified, the device set up instructions will be given. The user will have to set up the Home Base and plug in the SPARC so that it is on. The user will then have to turn Bluetooth on their mobile devices and pair it with the SPARC’s Bluetooth module (the HC06 module has been chosen for our prototype). Once the user’s device is paired with the SPARC, the mobile application home page will display the device options. The device options will include manual SPARC device controls and service scheduling. Manual SPARC device control will allow the user to move the SPARC forward, backward, left and right. The Service Scheduler will allow the user to schedule when they want the SPARC to run. For example, if garbage collection is every Friday morning at 7 AM, the user would want to schedule the SPARC to go to the curb at 4AM on Friday morning. The user can also schedule when the SPARC returns to the home base. To be sure that the SPARC does not miss the garbage collection the user is prompted to choose some return time multiple hours after the collection time. If the user does not choose a return time, they will have to manually return the SPARC to the Home Base to charge. On the application the user will also be able to view their user profile setting which includes their notification settings. The user can choose to be notified when the SPARC moves and when the SPARC is out of range.

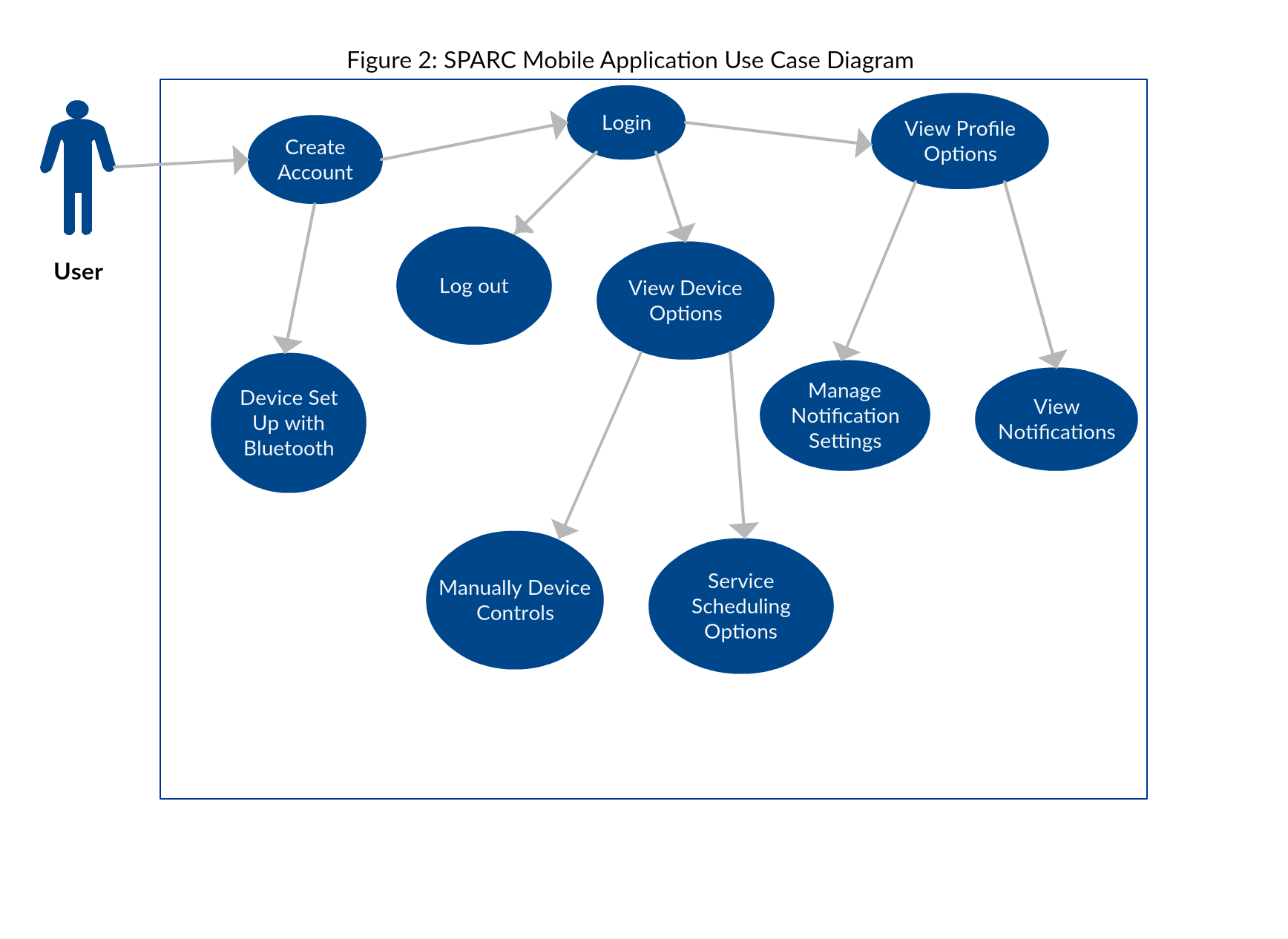


Figure 27 Software Use Case Diagram

For a more specific and detailed layout of the software, a class diagram is used to expand on each class of the mobile application which are the building blocks of the software. The below figure demonstrates this by splitting the software into eight main classes which will also serve as each main screen the user will be presented with. Firstly, is the Login/Sign up page where the user chooses to create an account or sign into the account that was previously made. If create account was chosen the user will be prompted to set up their new device on the screen. There the user will go through the steps needed to correctly calibrate the SPARC and make a Bluetooth connection. Once the user is signed in the screen shown will be the home page that has two options: User Settings and Device Options. User Setting will include an on and of option for notifications whenever the SPARC is scheduled to move and also to view those notifications. Device Options will allow the user to choose between “Schedule Service” and “Manual Control”. The schedule service option will prompt the user to choose the time and day and the frequency. The manual control option will have a “Joystick” control pad that allows users to control the SPARC, left, right, forward and backward. Once the user is completely set up, the Main home page will be the Manual Device Control page.

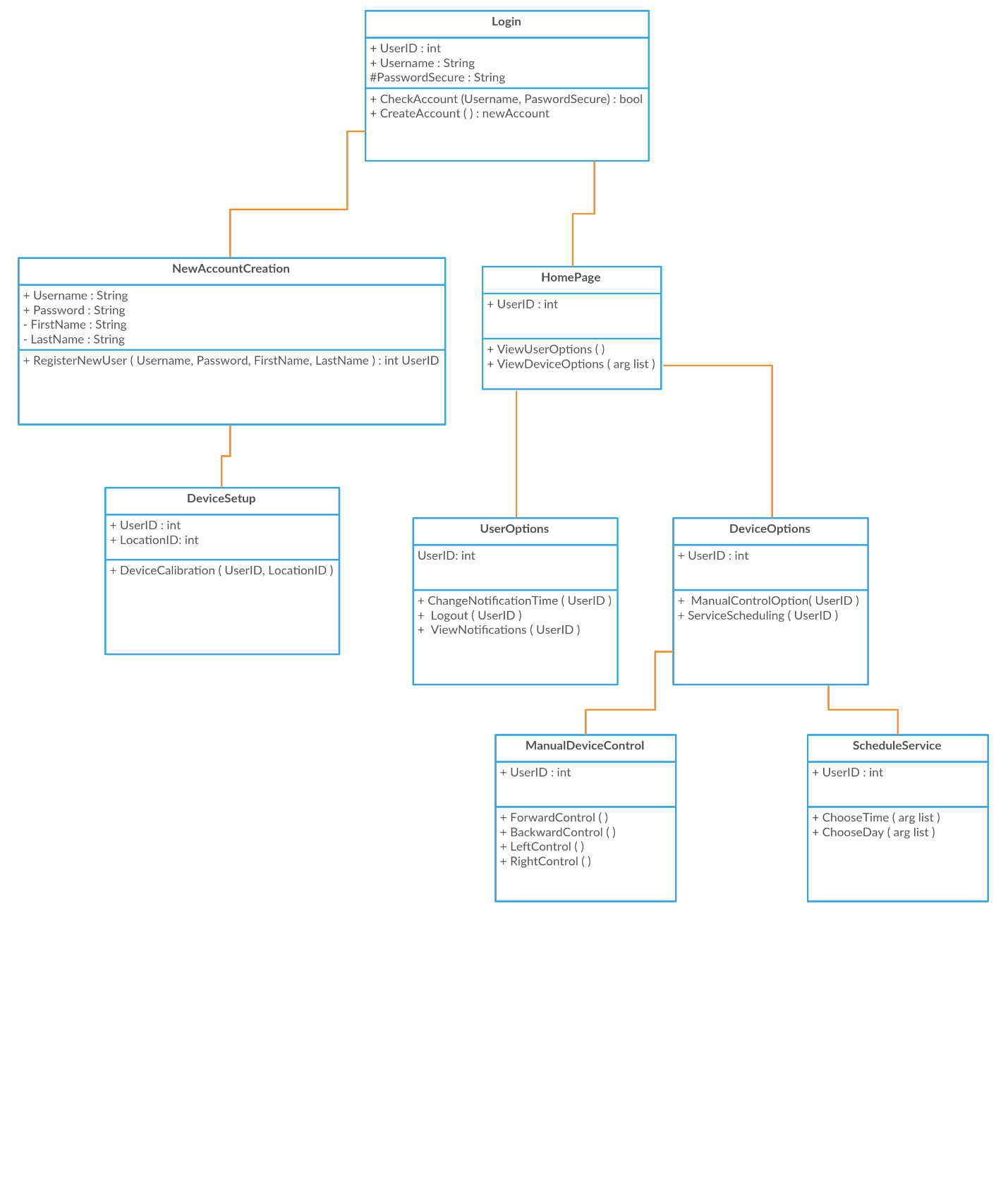


Figure 28 Application Class Diagram

### 6.3.2 Application to Microcontroller Interaction

To control the SPARC, the Mobile Application will have to connect to the device in such a way that it can wirelessly transmit commands to it. As shown in the Section 3: Relevant Project Research, multiple options were considered to transmit these commands. Ultimately Bluetooth was chosen due to is simple connection and despite its low transmit range as the SPARC device, for the prototype, the user will be in close proximity to the SPARC. Bluetooth connectivity can be easily achieved on Android Applications via Android Bluetooth APIs. These APIs (Application Programming Interface) allow applications to wirelessly connect to other Bluetooth devices, enabling point-to-point and multipoint wireless features.

The Application, using Bluetooth APIs, can scan for other Bluetooth devices, query the local Bluetooth adapter for paired Bluetooth devices, establish RFCOMM channels, connect to other devices through service discovery, transfer data to and from other devices and manage connections. For this application The SPARC will only need to pair to the User’s phone once and that connection will be remembered. Once connected the application can start sending data. In the device set up, when a new account is created, the application will show a list of scanned devices and a list of paired/connected devices. The SPARC will utilize the HC06 Bluetooth module which can be paired with the phone in the user’s phone settings. The pairing code for this module is simply “1234”. If not paired already, the application will pair with the HC06 once it is selected from the list of scanned devices and receives the code. The HC06 will appear on the list of scanned devices as long as the SPARC is on, receiving power. Once connected, data transfer can begin. Commands will be sent via Bluetooth to the microcontroller as ASCII characters. A buffer of characters can be sent to the microcontroller and read out as a string. Data that will be sent to the microcontroller will include, the movement commands, the schedule and on/off controls. The Bluetooth state diagram is shown below.

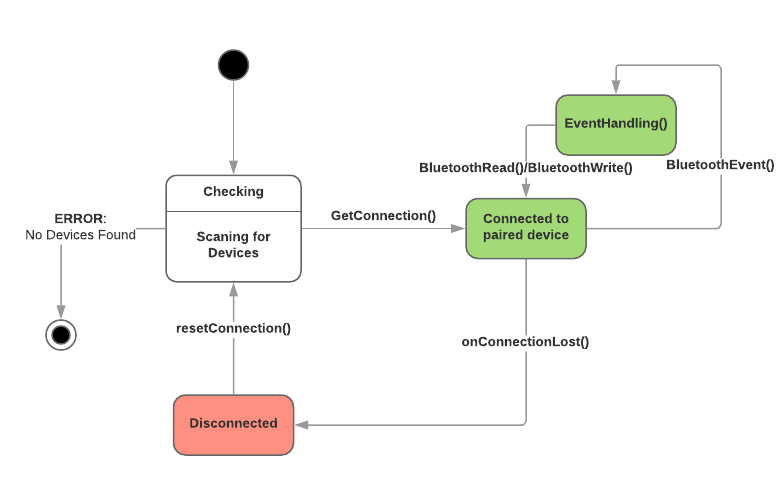


Figure 29 SPARC Bluetooth State Diagram

To send User Commands, each option will simply send a single ASCII character through Bluetooth to the Microcontroller which will then go through it’s receive buffer, and when it revives a specific character it will execute the specified command. For Movement controls the Application will send the SPARC a character to say that it is ready to send movement commands, once the SPARC sends back a ready character to the application, then User can begin pressing the controller like buttons to move the SPARC in real time. Each button press will send a unique character that is encoded to specific movements on the microcontroller. It will be a matter of the microcontroller changing certain pin states on the hardware to execute the movement. When setting up or changing the schedule for the SPARC to automatically begin its journey, the application will send another unique character to say that it is ready to send the schedule setting. Once received, the SPARC will send back the ready response. The User will input the day, the time and the frequency it wants the SPARC to go to the curb. The application will encode this into a string and send it character by character to the microcontroller. Once the string is finished it will end the finished signal, easily the null terminator for string, and the string will be saved and decoded on the microcontroller into the schedule. The microcontroller will implement its own internal clock to abide by the schedule. As an option the user can receive notifications when the SPARC is scheduled to move and deploys correctly. The on/off controls will be implicitly implemented as when the device is not on schedule to move the SPARC should be set to be in a low powered state. Before the scheduled deployment begins the application sends a power-up signal and the SPARC will send back a confirmation once its is back on. After the SPARC reaches the curb, it sends a finished movement signal to the application and the application will send back a power down command. The same process is used when the SPARC will make its journey back to the Home Base. If the User loses Bluetooth connectivity with the SPARC during its deployment to send the power on signal, the SPARC will automatically do this as long as it is preset to do so.

### 6.3.3 Mobile Application Design

The user interface of the mobile application will define the user experience and must be designed with that in mind. The UI design will be very simple for this application where every main class is a screen for the user to see and interact with. The mobile application is designed and programmed using the Android Studio 3.1.2 IDE and tested using the built in Android phone emulator. This application was tested on an emulated Google Pixel phone. Using this IDE, each class is an “Activity” that includes a .java file for the functional code and an associated .xml file for user interface design.

The first screen the user will see is a “Splash Screen”. This is used as a buffer to load any user data in the background as the app is opened. If the user has already signed in the Splash Screen will load all of that user’s saved data so that the application will open directly to the user’s home page. If not, the Splash Screen, shown in Figure 25 on the emulated Google Pixel phone, will open the Sign In Activity. The Sign in activity with prompt the user to input their email and password. If the email and password match information stored on the application server then the user will be logged in and their device settings will be loaded. For new users, there will be a prompt that says “ No account yet? Create One” that once clicked, will open the Sign up activity. Sign in Screen also shown below.

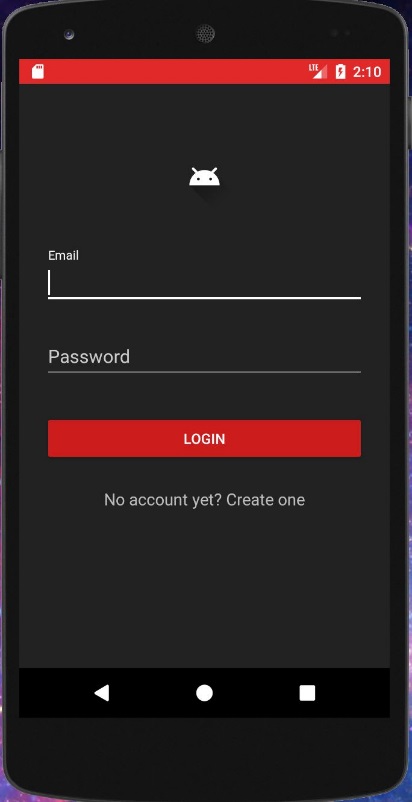
 

Figure 30 SPARC Mobile Application Splash Screen and Sign in Screens

On the Sign Up screen, shown in Figure 27, the users name, email, a password and to re enter the password to make sure the user inputted it correctly. The email is used to create a unique key for the user and is stored in the application server. The application server is essentially an authentication service that securely stores user accounts and private user information. There are validation checks for the information inputted on the sign up screen. The email address must be a standard format email address and the password must be between 4 and 10 characters long. Once all of the user info is valid and “Create Account” is pressed the user data is parsed into a new user account stores in the server.

The next major screen on the SPARC mobile application is the Device Controls Screen, shown in Figure 27. This is where the user will take up manual control of the SPARC device either in the initial device set up or at anytime the user requires. The controls include forward, back, left and right and the simple user interface is designed to be similar to video game or RC car controls. There will be a “Schedule Device” button for easy access to rescheduling the SPARC service. On the top right of the screen, Settings for the SPARC service is split into Device Settings and User Settings. The device setting include the device controls and schedule service options but in another place other that the home page. Also on/off controls can be specifically uses under these settings and in case the SPARC has disconnected from the user’s device, device settings will be where the Bluetooth options will be. The User settings include regular account settings like Sing Out or Change Password and notification settings as the user can receive notifications for SPARC events.

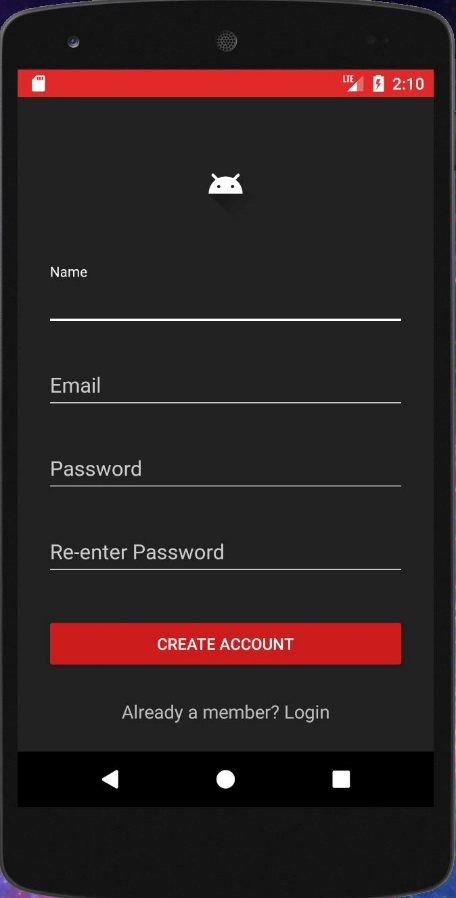
 

Figure 31 SPARC Mobile Application Sign Up and Device Control Screens

## 6.4 Pathing Algorithm

The software onboard the SPARC crawler will enable it to drive itself from the home-base to the user’s designated drop-off point and vice versa while detecting and maneuvering around obstacles it finds in its way.

The features of the algorithm prioritize autonomy from the user so that after the user performs the initial path priming and setting their desired delivery schedule, the SPARC will perform its duties without any input from the user whatsoever. The SPARC is ultimately a trash-bin robot, and ensuring that the user does not have to give it any additional consideration is the algorithm’s priority.

Unlike other similar projects, the SPARC makes no attempt to fully map its problem space; it would be a difficult endeavor give the openness of the operating environment, and ultimately a futile one as new, potentially permanent obstacles may be added between each delivery. The SPARC instead only explores new areas when its preferred path is blocked by obstacles and will stick with its preferred path if at all possible. The optimal path for any one delivery is never assumed at start-up; new obstacles may always be encountered so the decision making is done at run-time in each individual node.

Thus, the pathing algorithm implements reinforcement learning to iteratively change the preferred path when obstacles are encountered. The policy created from this should create preferred paths that encounter no obstacles.

### 6.4.1 Class Diagram

The class diagram for the pathing algorithm is outlined in figure 32 below. Separate graph can be stored in the SPARCs memory, and each graph is host to many node linkages which comprise the path the SPARC will take in order to reach its destination. The Frontier structure is the list of possible nodes that the SPARC may add to the graph in order to reconnect to the destination or an existing path, which are pushed onto the Fringe stack structure where they are selected.

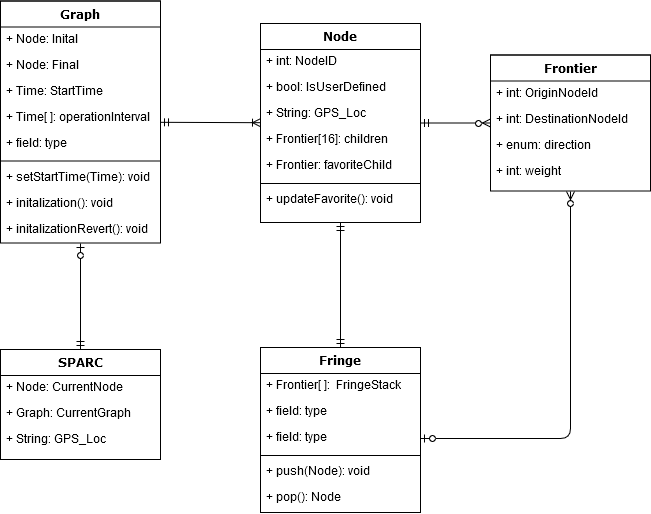


Figure 32 Pathing Algorithm Class Diagram

**6.4.2 Initial Path Priming & Graph Initialization**

The graph’s initial path is initialized by the user through use of the remote control feature in the mobile app. Once initialization is started, the user is instructed in the app to drive the SPARC with the remote control feature down their desired path from the SPARC’s initial location at the home base towards their desired destination where their trash will be collected. The user will then also be able to perform the same procedure for pathing from the trash collection point to the home base as well. The construction of two distinct graphs allows the user to customize their ideal path for the SPARC based on their knowledge of what obstacles will exist during each time of operation, which could vary greatly with the parking schedule of automobiles and other obstructions. When the user begins or ends the initialization protocol, the algorithm will create unique nodes signifying the initial and terminal nodes for the graph, respectively.

While the user is guiding the SPARC to their desired drop-off point, the SPARC will be constructing a sequence of nodes for it to follow during its future trips; every 5 seconds measured as by the system clock it will attempt to construct another node in the search graph. If the SPARC’s current position is at least 1.0 meters away from another node’s stored GPS coordinate, it will construct a new node with the SPARC’s current GPS coordinate. It will continue this behavior until the user has finished guiding the SPARC towards their chosen drop-off destination and has terminates the initialization protocol. When the initialization ends, the last node created is marked as the destination node.

After performing the initialization(s), the SPARC will require no further input from the user. The SPARC will automatically follow this path for its routine operations and will be able to find its way to the destination node through use of the stored GPS coordinate should obstacles obstruct the user-defined path during its trip to its destination. The GPS will also provide a generalized time schedule for operations based on the GPS localization should the user activate it and neglect to set the schedule through the mobile app or the app unable to receive this information from their mobile phone.

Below is a flowchart outlining the initialization protocol. Each loop the algorithm checks the location of the SPARC with respect to each other node to determine whether it has moved enough to warrant the creation of another reference node in the user’s path. The Algorithm will loop until the user terminates the protocol through the mobile app.

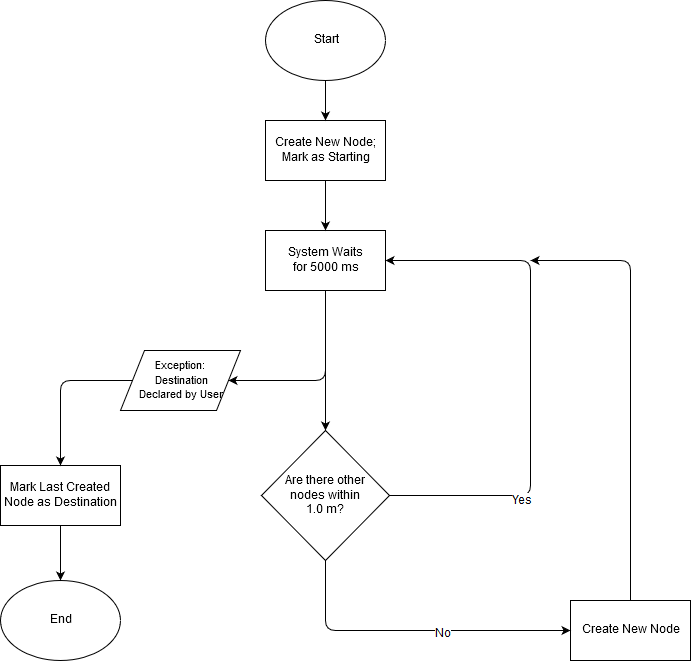


Figure 33 Initialization Flowchart

### 6.4.3 Value Iteration

The pathing algorithm is guided by the weights assigned to the nodes on each graph. With each traversal to the destination, the SPARC will update the weights on each node depending on the paths taken, and the collisions encountered with each iteration. The objective of having the SPARC modify the weights of each node over time is to remove any necessity for the user to have to re-initialize the SPARC each time they have a new car parking in their driveway. The SPARC will also with a small probability check paths that have low but non-zero weights to ensure that old, but more efficient paths are not being neglected due to an obstruction that only existed temporarily. This functionality in the algorithm allows for the SPARC to be truly autonomous, which was the primary specification for its design.

The initial path that the user creates through the initialization protocol will be the only nodes that will exist in new graphs, and they will have a particularly larger starting weight than nodes created during normal operation of the SPARC. The justification is that the user has the best knowledge of what path will be best for the SPARC to take, and that this knowledge should take precedence over any iterative changes the SPARC makes to the graph over time.

Every node that the algorithm creates has a flag that marks if it was generated by the user’s initialization path. Should the user be dissatisfied with the SPARC’s current path, they are enabled to revert any graph to its initial nodes, deleting any additional nodes that the SPARC had made creating new paths to the destination upon encountering obstacles. Reverting a graph also changes all the weights of the initial nodes that the SPARC may have modified to their default values. The user is also free to override all data in the current graph by re-initializing it.

### 6.4.4 Collision Avoidance Implementation

The SPARC is capable of detecting imminent collisions to the system by utilizing Infrared technology to detect the proximity of objects near the SPARC. While it is finding its way to the destination, the SPARC’s rangefinder IR device will be projecting IR light and be detecting reflected light off of nearby surfaces. When the SPARC is close enough to a nearby obstacle, the rangefinder will trigger an exception in the pathfinding algorithm.

Whenever the SPARC would attempt to move forward in the direction of a known obstacle within the range threshold, it instead marks the node as impassable, giving the path to what would have been the next node an infinite cost for this trip to traverse to that node. The SPARC then backtracks to the previous node and will begin searching for a path that will lead back to its preferred path to the destination, and will forge new paths if there are no pre-explored ones available. The SPARC will push attempts at movement into other cardinal, intercardinal, or secondary-intercardinal directions and will pop & explore these frontiers until it has found one that is not blocked by obstructions, wherein it will proceed with creating new nodes as described in the initialization. The SPARC will have preference for the directions that would reduce the manhattan distance to the nearest node. Nodes previously explored this trip and their unchosen children will not be taken into consideration for the manhattan distance calculation. The SPARC will continue creating new nodes until it has reconnected with a previously created, unexplored node or the destination node, whichever is closer.

# 7.0 Prototyping

The next part of the development cycle after research and development is building a prototype of the device. For the purposes of this project, a full prototype of the SPARC concept design will be made, hardware and software components included. The hardware prototype will be the biggest part of the SPARC prototype and might not exactly be the final product as it will be the basic functional concept design for the SPARC. The software prototype on the other hand will be the base code of the SPARC device that will only require some, hopefully small, revisions if the SPARC were to be a product.

## 7.1 Hardware Prototyping

Spark Prototyping for Hardware will be very limited, mainly consisting of testing the resistances, capacitances, and inductances of the acquired parts to assure that they fall within their specified tolerances. The only output that needs to be tested it the motor control, whether we can successfully make each motor go forward and backwards independently of each other. To prototype a rough drafter of the parts were made and follow in table 3.

Table 3 Prototype List of Materials 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Item | Details | Vendor | Price / Unit | Quantity | Total Price |
| 18650 Li-Ion Battery | 4 x 3.7V, 2500mAh | Walmart | $9 | 8 | $72 |
| Battery Holder | 4 x 18650 Cells | Digikey | $6.30 | 8 | $50.40 |
| Protection Circuit | PCB-S4A5-GS | BatterySpace | $6.75 | 1 | $6.75 |
| Trash Can | 32-gallon | Home Depot | $60 | 1 | $60 |
| Chassis | Steel, support ~60lbs | Research | Research | 1 | $150 |
| 12V DC Motor | 4.47 oz-in torque | RobotShop | $6.25 | 2 | $12.50 |
| Gear Box | 81:1 gear ratio | RobotShop | $68 | 2 | $136 |
| MicroController | ATSAMD21J18A-AUT | TI, Digi-Key | $3.40 | 1 | $3.40 |
| Voltage Regulators | 2A rated, 12V, 5V, 3.3V | Digi-Key | $1.73 | 15 | $27.68 |
| Motor Controller | MC33HB2001EK | Digi-Key | $6 | 2 | $12 |
| PCB | Research | Research | ~$40 | 1-2 | ~$80 |
| Copper Wire | 80’, bare, 14 gauge | Amazon | ~$15 | 1 | ~$15 |
| Bluetooth Module | HC-06 | eBay | $2.50 | 1 | $2.50 |
| RFID Tx/Rx | 315MHz link | Amazon | $5.22 | 1 | $5.22 |

Table 4 Prototype List of Materials 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RFID Tx/Rx | 433MHz link | Amazon | $6.66 | 1 | $6.66 |
| GPS Module | Research | Research | Research | 1 | Research |
| IR Sensor | GP2Y0A710K0F | RobotShop | $17.00 | 1 | $17.00 |
| Acrylic | 2’ x 2’ | Research | ~$20 | 1 | ~$20 |
| Bike chain | Research | Amazon | ~$10 | 4 | ~$40 |
| AC-DC Converter | 18V 5A | Amazon | $9 | 1 | $9 |
| Pixy Cam | Arduino Camera | Amazon | $69 | 1 | $69 |
|  |  |  |  | Total | ~$795.12 |

## 

## 7.2 Software Prototyping

The SPARC prototype device will require each component of the software to be development, tested and integrated. Once the full mobile application and the pathing algorithm is development that can be integrated with the processor program which will take in the inputs from both software components to control the device output. An initial mobile application prototype will be developed for use in testing the hardware components of the SPARC.

The initial prototype will be designed to work in both manual control mode and autonomous mode. For manual control, the embedded code on the prototype will receive the commands from the prototype application and will have to decode them to act on the corresponding movements for the SPARC prototype. For autonomous control, the embedded code will take in the specified schedule from the mobile application and store it. This initial prototype is utilized for the stress testing the SPARC in different operating environments.

The final prototype will have the rest of the mobile application's functionality, such as the schedule setting and path initialization. When the schedule is triggered for this prototype, the embedded code will control the turn on from a low powered state to on and begin its movement. The movement controls will then receive input from the pathing algorithm which reads from and grabs data from the camera for its vision and GPS for its spatial awareness. The input received from the algorithm will control the movement in accordance to the embedded movement controls. No materials will need to be purchased for the software side of the SPARC prototype.

## 7.3 Bill of Materials

The following table is the final draft of the parts, either purchased of in the process of being researched and will be purchased. The bill of materials is large and will be split into multiple tables.

Table 5 Bill of Materials 1.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | value | quantity | price | Supplier | Manufacturer Part | Cost |
|  | ARM Cortex M0+ Microcontroller | 1 | $3.39 | Digi-Key | ATSAMD21J18A | $3.39 |
|  | 100nF (0.1uF) Capacitor | 9 | $0.11 | Digi-Key | CL21B104JBCNNNC | $0.99 |
|  | 10uF Capacitor | 4 | $0.88 | Digi-Key | C0805X106J9RACAUTO | $3.52 |
|  | 1uF | 1 | $0.40 | Digi-Key | CL21B104JBCNNNC | $0.40 |
|  | 10uH | 1 | $0.21 | LCSC | LBM2016T100J | $0.21 |
|  | H-Bridge Motor Controller | 2 | $6.71 | Digi-Key | MC33HB2001EK | $13.42 |
|  | 10kOhm Resistor | 4 | $0.79 | Digi-Key | ERA-6ARW103V | $3.16 |
|  | 200Ohm Resistor | 2 | $0.93 | Digi-Key | RG2012N-201-W-T1 | $1.86 |
|  | 0.01uF Capacitor | 2 | $0.57 | Digi-Key | GRM2195C1H103FA01D | $1.14 |
|  | 100uF Capacitor | 2 | $0.89 | Digi-Key | 35TRV100M8X10.5 | $1.78 |
|  | 33nF Capacitor | 4 | $0.41 | Digi-Key | CGA4J2C0G1H333J125AA | $1.64 |
|  | 4 Cell 18650 Holder | 8 | $6.30 | Digi-Key | BK-18650-PC8 | $50.40 |
|  | IC Battery Charger Li-Ion | 8 | $2.53 | Digi-Key | MIC79050-4.2YMM | $20.24 |
|  | IC Load Switch | 4 | $0.50 | Digi-Key | AP2337SA-7 | $2.00 |

Table 6 Bill of Materials 1.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 4.7uF Capacitor | 7 | $2.15 | Digi-Key | C5750JB2A475M230KA | $15.05 |
|  | DC Barrel Jack | 1 | $1.56 | Digi-Key | 694103107102 | $1.56 |
|  | BA33DD0WHFP-TR | 2 | $1.92 | Digi-Key | BA33DD0WHFP-TR | $3.84 |
|  | BAJ6DD0WHFP-TR | 3 | $1.92 | Digi-Key | BAJ6DD0WHFP-TR | $5.76 |
|  | SR Nor Latch | 1 | $0.50 | Digi-Key | MC14043BDG | $0.50 |
|  | BAJ2DD0WHFP-TR | 11 | $1.92 | Digi-Key | BAJ6DD0WHFP-TR | $21.12 |
|  | BA50DD0WHFP-TR | 1 | $1.92 | Digi-Key | BAJ6DD0WHFP-TR | $1.92 |
|  | USB Receptacle | 1 | $2.84 | Digi-Key | 1-292303-6 | $2.84 |
|  | HC-06 | 1 | $4.02 | eBay | HC-06 | $4.02 |
|  | RF Transmitter | 2 | $0.83 | Digi-Key | MICRF113YM6 | $1.66 |
|  | 10pF | 6 | $0.68 | Digi-Key | CBR08C100JAGAC | $4.08 |
|  | 82nH Inductor | 1 | $1.16 | Digi-Key | 744912182 | $1.16 |
|  | 470nH Inductor | 2 | $0.12 | Digi-Key | CK2125R47M-T | $0.24 |
|  | 433 MHz Antenna | 2 | $2.26 | Digi-Key | ANT-433-SP | $4.52 |
|  | 18pF Capacitor | 4 | $0.77 | Digi-Key | 08055A180FAT2A | $3.08 |
|  | 13.56MHz Crystal | 1 | $0.41 | Digi-Key | ABLS-13.560MHZ-10-R30-D-T | $0.41 |
|  | 9.84375MHz Crystal | 1 | $1.61 | Digi-Key | ECS-98.4375-CDX-0314-TR | $1.61 |
|  | 313MHz Antenna | 2 | $2.26 | Digi-Key | ANT-313-SP | $4.52 |
|  | 6.8uF Capacitor | 1 | $0.55 | Digi-Key | CBR08C689B1GAC | $0.55 |
|  | 150nH | 1 | $0.33 | Digi-Key | MLF2012DR15JTD25 | $0.33 |
|  | RF Receiver | 2 | $2.03 | Digi-Key | MICRF219AYQS | $4.06 |
|  | 13.52127MHz Crystal | 1 | $0.66 | Digi-Key | ABLS-13.52127MHZ-10-J-4Q-T | $0.66 |
|  | 33nH Inductor | 1 | $1.16 | Digi-Key | 744912133 | $1.16 |
|  | 68nH Inductor | 2 | $0.74 | Digi-Key | AIAC-1812-68NJ-T | $1.48 |

Table 7 Bill of Materials 1.3

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 1.2pF Capacitor | | 2 | | $0.79 | | Digi-Key | GQM2195C2A1R2BB01D | | $1.58 |
|  | 1.5pF Capacitor | | 2 | | $1.31 | | Digi-Key | | | GQM2195C2E1R5BB12D | $2.62 |
|  | 39nH Inductor | | 1 | | $0.74 | | Digi-Key | | | AIAC-1812-39NJ-T | $0.74 |
|  | 9.81563MHz Crystal | | 1 | | $1.06 | | Digi-Key | | | ABM7-9.81563MHZ-10-R50-D4Q-T | $1.06 |
|  | 0.1uF Capacitor | | 2 | | $1.71 | | Digi-Key | | | C1812C104J2GACAUTO | $3.42 |
|  | 18V 5A AC-DC Adapter | | 1 | | $13 | | Amazon | | | B0771MMYV9 | $13.00 |
|  | RS-540 12V 17200 RPm Brushed DC Motor | | 2 | | $6.25 | | RobotShop | | | RB-Ban-68 | $12.50 |
|  | P60 81:1 Gear Box | | 2 | | $67.95 | | RobotShop | | | RB-Ban-235 | $135.90 |
|  | GPS Module | | 1 | | $41.42 | | Digi-Key | | | PAM-7Q-0 | $41.42 |
|  |  | |  | |  | |  | | |  |  |
|  |  | |  | |  | |  | | | Total = | $402.52 |

Table 8 Bill of Materials 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | value | quantity | price | Supplier | Manufacturer Part | Cost |
| 1 | 100nF (0.1uF) Capacitor | 9 | $0.11 | Digi-Key | CL21B104JBCNNNC | $0.99 |
| 2 | 10uF Capacitor | 4 | $0.88 | Digi-Key | C0805X106J9RACAUTO | $3.52 |
| 3 | 1uF Capacitor | 1 | $0.40 | Digi-Key | CL21B104JBCNNNC | $0.40 |
| 4 | 10uH Capacitor | 1 | $0.21 | LCSC | LBM2016T100J | $0.21 |
| 5 | 10kOhm Resistor | 4 | $0.79 | Digi-Key | ERA-6ARW103V | $3.16 |
| 6 | 200Ohm Resistor | 2 | $0.93 | Digi-Key | RG2012N-201-W-T1 | $1.86 |
| 7 | 0.01uF Capacitor | 2 | $0.57 | Digi-Key | GRM2195C1H103FA01D | $1.14 |
| 8 | 100uF Capacitor | 2 | $0.89 | Digi-Key | 35TRV100M8X10.5 | $1.78 |
| 9 | 33nF Capacitor | 4 | $0.41 | Digi-Key | CGA4J2C0G1H333J125AA | $1.64 |
| 10 | 4.7uF Capacitor | 7 | $2.15 | Digi-Key | C5750JB2A475M230KA | $15.05 |
| 11 | 10pF Capacitor | 6 | $0.68 | Digi-Key | CBR08C100JAGAC | $4.08 |
| 12 | 82nH Inductor | 1 | $1.16 | Digi-Key | 744912182 | $1.16 |
| 13 | 470nH Inductor | 2 | $0.12 | Digi-Key | CK2125R47M-T | $0.24 |
| 14 | 18pF Capacitor | 4 | $0.77 | Digi-Key | 08055A180FAT2A | $3.08 |
| 15 | 150nH Inductor | 1 | $0.33 | Digi-Key | MLF2012DR15JTD25 | $0.33 |
| 16 | 33nH Inductor | 1 | $1.16 | Digi-Key | 744912133 | $1.16 |
| 17 | 68nH Inductor | 2 | $0.74 | Digi-Key | AIAC-1812-68NJ-T | $1.48 |
| 18 | 1.2pF Capacitor | 2 | $0.79 | Digi-Key | GQM2195C2A1R2BB01D | $1.58 |
| 19 | 1.5pF Capacitor | 2 | $1.31 | Digi-Key | GQM2195C2E1R5BB12D | $2.62 |
| 20 | 39nH Inductor | 1 | $0.74 | Digi-Key | AIAC-1812-39NJ-T | $0.74 |
| 21 | 0.1uF Capacitor | 2 | $1.71 | Digi-Key | C1812C104J2GACAUTO | $3.42 |
|  |  |  |  |  |  |  |
|  |  |  |  |  | Total | $49.64 |

Table 9 Bill of Materials 3.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | value | quantity | price | Supplier | Manufacturer Part |  |
| 1 | ARM Cortex M0+ Microcontroller | 1 | $3.39 | Digi-Key | ATSAMD21J18A | $3.39 |
| 2 | H-Bridge Motor Controller | 2 | $6.71 | Digi-Key | MC33HB2001EK | $13.42 |
| 3 | 4 Cell 18650 Holder | 8 | $6.30 | Digi-Key | BK-18650-PC8 | $50.40 |
| 4 | IC Battery Charger Li-Ion | 8 | $2.53 | Digi-Key | MIC79050-4.2YMM | $20.24 |
| 5 | IC Load Switch | 4 | $0.50 | Digi-Key | AP2337SA-7 | $2.00 |
| 6 | DC Barrel Jack | 1 | $1.56 | Digi-Key | 694103107102 | $1.56 |
| 7 | BA33DD0WHFP-TR | 2 | $1.92 | Digi-Key | BA33DD0WHFP-TR | $3.84 |
| 8 | BAJ6DD0WHFP-TR | 3 | $1.92 | Digi-Key | BAJ6DD0WHFP-TR | $5.76 |
| 9 | SR Nor Latch | 1 | $0.50 | Digi-Key | MC14043BDG | $0.50 |
| 10 | BAJ2DD0WHFP-TR | 11 | $1.92 | Digi-Key | BAJ6DD0WHFP-TR | $21.12 |
| 11 | BA50DD0WHFP-TR | 1 | $1.92 | Digi-Key | BAJ6DD0WHFP-TR | $1.92 |
| 12 | USB Receptacle | 1 | $2.84 | Digi-Key | 1-292303-6 | $2.84 |
| 13 | HC-06 | 1 | $4.02 | eBay | HC-06 | $4.02 |
| 14 | RF Transmitter | 2 | $0.83 | Digi-Key | MICRF113YM6 | $1.66 |
| 15 | 433 MHz Antenna | 2 | $2.26 | Digi-Key | ANT-433-SP | $4.52 |
| 16 | 13.56MHz Crystal | 1 | $0.41 | Digi-Key | ABLS-13.560MHZ-10-R30-D-T | $0.41 |
| 17 | 9.84375MHz Crystal | 1 | $1.61 | Digi-Key | ECS-98.4375-CDX-0314-TR | $1.61 |
| 18 | 313MHz Antenna | 2 | $2.26 | Digi-Key | ANT-313-SP | $4.52 |
| 19 | RF Receiver | 2 | $2.03 | Digi-Key | MICRF219AYQS | $4.06 |
| 20 | 13.52127MHz Crystal | 1 | $0.66 | Digi-Key | ABLS-13.52127MHZ-10-J-4Q-T | $0.66 |

Table 10 Bill of Materials 3.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 21 | 9.81563MHz Crystal | 1 | $1.06 | Digi-Key | ABM7-9.81563MHZ-10-R50-D4Q-T | $1.06 |
| 22 | 18V 5A AC-DC Adapter | 1 | $13 | Amazon | B0771MMYV9 | $13.00 |
| 23 | RS-540 12V 17200 RPm Brushed DC Motor | 2 | $6.25 | RobotShop | RB-Ban-68 | $12.50 |
| 24 | P60 81:1 Gear Box | 2 | $67.95 | RobotShop | RB-Ban-235 | $135.90 |
| 25 | GPS Module | 1 | $41.42 | Digi-Key | PAM-7Q-0 | $41.42 |
|  | 4 Pack 18650 High Drain Li-Ion Batteries | 8 | $9.80 | Walmart |  | $78.40 |
|  |  |  |  |  |  |  |
|  |  |  |  |  | Total = | $430.73 |

# 8.0 Testing

In this section, an outline of the test cases for each component of SPARC will be presented. The SPARC prototype testing will be divided into Hardware testing and Software testing. The hardware tests will include testing the device’s power systems, movement, microcontroller functionality and each independent module. Software testing will include testing the mobile application and the path algorithm. Lastly a full integration test will be done.

## 8.1 Hardware Tests

This section will include all of the basic testing procedures for both the mechanical and electrical components for the SPARC. The tests with overarchingly cover four broad components that have the biggest priority in the hardware functionality. These components include the power, the microcontroller, the independent modules and the physical movement functionality itself.

### 8.1.1 Power Testing

Getting the power management of the SPARC device correctly is crucial in its functionality and will require extensive testing.

|  |  |
| --- | --- |
| **Test Name** | **PwT01: Voltage Testing** |
| Objective | Verify proper voltages at required PCB Pads after regulator installation |
| Requirements | * 14.8V input to regulators * 12V input to MCCs * 3.3V input to GPS, MCU, RF Transmitters * 5V input to USB and Lidar |
| Procedure | Apply Multimeter leads to pads and ground plane |
| Results Verification | Proper voltage within acceptable tolerance |

### 8.1.2 Microcontroller Testing

The SPARC device’s microcontroller needs to have correct pin mapping in order to function properly with the input and output controls required for the design. This test will ensure that the pin mapping of the microcontroller is correct for the purposes of the design.

|  |  |
| --- | --- |
| **Test Name** | **MT01: Pin Mapping Test** |
| Objective | Verify that the microcontroller’s pin mapping is correct. |
| Requirements | * Microcontroller * SPARC processor pin test program * External LED indicator |
| Procedure | Run a test program that tests each required pin and ensures proper function. Each pin that will be used will be initialized as input or output. Then the program will check to make sure that there is a signal sent or received from each pin. |
| Results Verification | All pins required in the design are properly assigned and functional, signified by the LEDs. |

### 

### 8.1.3 Independent Module Testing

|  |  |
| --- | --- |
| **Test Name** | **IMT01: Bluetooth Module Testing** |
| Objective | Verify that SPARC’s bluetooth module functions correctly once connected to the system. |
| Requirements | * HC06 Bluetooth Module * Mobile Application with Bluetooth connection * External LED indicator |
| Procedure | The Mobile application prototype will send a command character to the bluetooth module. The module will receive the character and insert it into its receive buffer. The character received will map to an I/O pin and set the pin to high. |
| Results Verification | The pin will be checked by connecting it to an external LED. If the pin is high the LED will turn on. |

### 8.1.4 Physical Movement Testing

The SPARC crawler system will have its mobility be tested in a large number of terrain types. Initial testing will simply be on a concrete surface, which will be the usual operating space for the SPARC. Simple maneuvers will be attempted in order to verify that the SPARC will be able to operate under these conditions.

|  |  |
| --- | --- |
| **Test Name** | **PMT01: Movement Testing - Concrete** |
| Objective | Validate the SPARC’s ability to perform simple maneuvers on concrete. |
| Requirements | * Android Remote Control Feature * Chassis Prototype * Concrete Testing Environment |
| Procedure | Using the remote control feature of the Android Mobile App, test the SPARC’s capabilities on concrete surface. Forward and reverse motion, and a 360° turn to validate functionality in this terrain. |
| Results Verification | The SPARC’s motors respond to commands from the mobile app, and have no trouble performing the movements in the procedure. |

While the pathfinding algorithm has a strong aversion to areas that the Pixy CMUcam5 identifies as grassy terrain, the SPARC will be enabled to traverse over these areas. The following test is to ensure that the motors provide enough torque to traverse grass in a reasonable amount of time.

|  |  |
| --- | --- |
| **Test Name** | **PMT02: Movement Testing - Grass** |
| Objective | Validate the SPARC’s ability to perform simple maneuvers on grass |
| Requirements | * Android Remote Control Feature * Chassis Prototype * Lawn Testing Environment |
| Procedure | Using the remote control feature of the Android Mobile App, test the SPARC’s capabilities on grass. Forward and reverse motion, and a 360° turn are to be attempted to validate functionality in this terrain. |
| Results Verification | The SPARC’s motors respond to commands from the mobile app, and have little to no trouble performing the movements in the procedure. |

#### 8.1.4.1 Adverse Conditions

The following tests are to test the limits of how the SPARC handles in terrain that is atypical for its normal function.

|  |  |
| --- | --- |
| **Test Name** | **PMT03: Movement Testing - Gravel** |
| Objective | Validate the SPARC’s ability to perform simple maneuvers in gravel |
| Requirements | * Android Remote Control Feature * Chassis Prototype * Gravel Testing Environment |
| Procedure | Using the remote control feature of the Android Mobile App, test the SPARC’s capabilities in gravel. Forward and reverse motion, and a 360° turn are to be attempted to validate functionality in this terrain. |
| Results Verification | The SPARC’s motors respond to commands from the mobile app, and is able to complete the movements in the procedure. |

|  |  |
| --- | --- |
| **Test Name** | **PMT04: Movement Testing - Wet Concrete** |
| Objective | Validate the SPARC’s ability to perform simple maneuvers on concrete. |
| Requirements | * Android Remote Control Feature * Chassis Prototype * Concrete Testing Environment * Spray Bottle with Water |
| Procedure | Using the remote control feature of the Android Mobile App, test the SPARC’s capabilities on a wet concrete surface. Forward and reverse motion, and a 360° turn to validate functionality in this terrain. |
| Results Verification | The SPARC’s motors respond to commands from the mobile app, and has at worst only minor difficulties performing the movements in the procedure. |

|  |  |
| --- | --- |
| **Test Name** | **PMT05: Movement Testing - Wet Grass** |
| Objective | Validate the SPARC’s ability to perform simple maneuvers on wet grass |
| Requirements | * Android Remote Control Feature * Chassis Prototype * Lawn Testing Environment * Spray Bottle with Water |
| Procedure | Using the remote control feature of the Android Mobile App, test the SPARC’s capabilities on wet grass. Forward and reverse motion, and a 360° turn are to be attempted to validate functionality in this terrain. |
| Results Verification | The SPARC’s motors respond to commands from the mobile app, and is able to complete the movements in the procedure. |

## 

## 8.2 Software Tests

Every aspect of the SPARC must be tested to ensure quality in the brains of the SPARC device. The test cases will be slipped into the three main components of the software.

### 8.2.1 Algorithm Simulation Testing

In order to isolate the functionality of the pathfinding algorithm and receive accurate date from testing the algorithm will simulate its functionalities in a virtual environment. The GPS information that the algorithm would receive from the SPARC’s GPS module will be represented by the cartesian coordinate location in the graph. Frontiers will only have the cardinal and intercardinal directions represented (8 directions instead of 16). Though the Frontiers in the virtual test will not have the additional secondary-intercardinal directions, this test should be sufficient to verify that the proof of concept is valid. To mimic the graphs that the SPARC will create, the virtual environment must also be able to be self-expanding using dynamic memory allocation. For these requirements, the following test will be performed:

|  |  |
| --- | --- |
| **Test Name** | **AST01: Virtual Driveway Functionality Testing** |
| Objective | Verify that that an agent can traverse through the data structure, and is impeded by obstacle cells. |
| Requirements | * Virtual Driveway Prototype |
| Procedure | Initialize the data structure and reference each cell to ensure no null pointers are declared.   Loop random movement for a large number of time-steps within the bounds of the structure to ensure no stack-overflow exceptions occur while executing.  Check that cells marked as obstacles are not able to be traversed by the search agent, and that attempting to do so will throw an exception. |
| Results Verification | The data structure initializes and allocates its size without encountering a runtime error.  The agent performs the arbitrarily large number of actions while in the virtual environment without running into any data overflows.  Exceptions are thrown when attempting to move into an obstacle cell. |

Being a C program, there is a large margin of error that could be encountered with dynamically reallocating the memory size of the data structure at runtime, so an additional functionality test is necessitated to ensure that

The initialization protocol will able to be tested by using the arrow keys in the console, which will display the current information about the graph such as: whether or not a coordinate square is a node, whether or not the SPARC is in a certain coordinate square. Only coordinate squares with created nodes and coordinate squares will that have been explored will show up on the console in this way. The functionality will tested automatically however, to ensure that convergence of test results is reached swiftly.

Each coordinate square will also carry a bit of information that will signify whether or not there exists an obstacle at that location. Obstacles will be added to the graphs to test the algorithm’s response to imminent collisions. To replace the IR rangefinder’s input, the algorithm will simply proceed as if such input has been received such that if it tries to move into a square that has an obstacle, it will instead continue with the behavior of trying to find or create a path to continue towards known paths to the destination. Obstacles will be able to removed from the virtual graph in the same manner, which will be used to test the value iteration functionality of the pathfinding algorithm. The following test is designed to ensure that the algorithm avoids encountering any obstacles in the graph.

|  |  |
| --- | --- |
| **Test Name** | **AST02: Virtual Driveway Recallocation Testing** |
| Objective | Verify that that the virtual driveway data structure is able to be resized successfully |
| Requirements | * Virtual Driveway Prototype |
| Procedure | Manually move the agent towards the edges of the virtual driveway such that the structure expands using dynamic memory allocation. |
| Results Verification | The data structure initializes and reallocates its size without exceeding a given constant memory cap. |

After ensuring that the data structure can be reallocated at runtime successfully, a more general testing of the Algorithm is required, which can be observed on the following page. This test will ensure that the SPARC is able to navigate around each obstacle, and in no cases will the algorithm ever choose to move in the direction of a known obstacle.

|  |  |
| --- | --- |
| **Test Name** | **AST03: Algorithm Functionality Test** |
| Objective | Verify that SPARC’s pathing algorithm navigates around virtual obstacles while navigating to its destination cell. |
| Requirements | * Virtual Driveway Completed * Algorithm Prototype Completed |
| Procedure | Initialize a virtual driveway with randomized locations for obstacles, size, and preferred path.  Run the agent through the graph with the SPARC’s pathfinding algorithm. Re-run test through numerous pre-generated graphs to ensure convergence of results. |
| Results Verification | The agent avoids impending collisions and expands its path out to the surrounding cells to attempt to navigate around the blocking cell. The agent expands its search to new nodes with closer manhattan distances to the destination first to simulate GPS locations. |

Once the algorithm is sufficiently tested and its ability to obstacles in the virtual environment is verified, the value iteration is added into the algorithm. Values are initialized into cells as either the value 16 for preferred paths, or the value 0 for others. This value is not affected by the nature of having an obstacle in this cell or not. When the agent attempts to move into a cell and detects that an obstacle resides in this cell, it should reduce the weight of the cell by 1 after marking it as explored for this iteration. Each new cell it explores trying to find a new path to the destination will have their respective weights raised by 1, such that a new path should emerge from the repeated encountering of the same obstacles.

|  |  |
| --- | --- |
| **Test Name** | **AST04: Value Iteration Test** |
| Objective | Verify that SPARC’s pathing algorithm iteratively modifies its preferred path |
| Requirements | * Virtual Driveway Completed * Algorithm Prototype Completed |
| Procedure | Initialize a single virtual driveway with randomized locations for obstacles, size, and preferred path.  Run the agent through the graph using the SPARC’s pathfinding algorithm with value iteration enabled for post-path analysis until the policy converges where no obstacles are encountered. Reduce value for paths that frequently encounter collisions, and increase value for paths that are not yet preferred but are chosen frequently. |
| Results Verification | The graph is updated with a new preferred path from the agent iterating through the graph such that obstacles are avoided completely in the new preferred path. |

### 8.2.2 Mobile Application Testing

The Mobile Application will be developed in the Android Studio IDE environment that includes an Android phone for testing and debugging purposes. The following are the test cases for testing the mobile application in this environment and when downloaded to a physical android phone and in that environment.

|  |  |
| --- | --- |
| **Test Name** | **MAT01: Sign in Authentication Test** |
| Objective | To test the authentication service on the mobile application. |
| Requirements | * The SPARC Mobile Application |
| Procedure | A premade user account will be used to sign into the mobile application. |
| Results Verification | The user must be signed in correctly with all the user data loaded. An automated test will return 1 for this case if successful. |

|  |  |
| --- | --- |
| **Test Name** | **MAT02: Sign Up Authentication Test** |
| Objective | Verify that new user can sign up for a new user account |
| Requirements | * The SPARC Mobile Application |
| Procedure | On the emulated Android phone, a new user account will attempt to be created. Once a valid email and password are entered a user key is created and the user data is stored in the authentication service. |
| Results Verification | The account created will be verified by attempting to sign into the account and checking if the user data can be retrieved. |

The Sign in and up test cases will mainly test the authentication service on the mobile application. The authentication service should be able to securely hold and create user data accounts and store them on a server.

|  |  |
| --- | --- |
| **Test Name** | **MAT03: Bluetooth connectivity test** |
| Objective | Verify that the application can view and scan for nearby Bluetooth devices. |
| Requirements | * The SPARC Mobile Application * Bluetooth connectable device * HC06 Bluetooth Module |
| Procedure | On a physical android device, the application will be loaded up. Once signed into a valid user account navigate to device settings. On the device settings activity the user will press the find devices button. Once the list appears the user will click on the HC06 module to connect. |
| Results Verification | All scanned device will be shown on the Device setting Activity. The HC06 module should be in the list. When the user attempts to connect the HC06 requires a pairing code and the Activity will say “Connected” it successful and “Not Connected” is unsuccessful. |

|  |  |
| --- | --- |
| **Test Name** | **MAT04: Device Controls Test** |
| Objective | Verify that the Mobile Application can send command for each of the controller buttons. |
| Requirements | * HC06 module * LED indicator * SPARC Mobile Application |
| Procedure | Once signed into the Application that is paired with the HC06, the user will input the four controller options, up, down, left and right. |
| Results Verification | Each output should be received by the HC06 module which will be connected to an LED indicator. When a command is received the LED will light up indicating the command receive. |

### 8.2.3 Processor Program Tests

The processor programming will be developed in the Arduino Software IDE environment. The following are the test cases for testing the processor program in this environment and when downloaded to a physical processor and in that environment.

The Movement Control test will ensure that the movement control functions operate properly and are synchronized.

|  |  |
| --- | --- |
| **Test Name** | **PPT01: Movement Control Test** |
| Objective | Verify that the processor programming can send a signal to and activate the motors with both clockwise and anticlockwise rotation. |
| Requirements | * Microcontroller * SPARC processor program * Motor controllers |
| Procedure | Run a test program that procedurally tests one motor at a time in each direction, then both in the same direction, and finally in opposite directions. |
| Results Verification | First, each motor should receive a signal from the microprocessor. Then for each motion test, the motors should operate simultaneously in the prescribed combination of directions. |

Additionally, the input control needs to be tested to ensure that information can be sent to the processor program. The input control will handle Bluetooth, Pixy Cam, LIDAR, and GPS input data and as such needs to be functioning properly for the design to succeed.

|  |  |
| --- | --- |
| **Test Name** | **PPT02: Input Control Test** |
| Objective | Verify that the processor programming can correctly receive and handle input from its sensors and modules. |
| Requirements | * Microcontroller * SPARC processor program * HC06 module * Pixy Camera * GPS module * LIDAR module |
| Procedure | Run a test program that procedurally tests one input component at a time and displays its input information. |
| Results Verification | The microcontroller should receive information appropriate to each sensor type. |

The output control will also be tested for proper functionality based on test inputs that will simulate all of the possible conditions that the SPARC unit might encounter.

|  |  |
| --- | --- |
| **Test Name** | **PPT03: Output Control Test** |
| Objective | Verify that the processor programming can correctly receive and handle input from its sensors and modules. |
| Requirements | * Microcontroller * SPARC processor program * HC06 module |
| Procedure | Run a test program that procedurally tests one output situation at a time using hard coded data for inputs. |
| Results Verification | The microcontroller should output motor functions and user notifications through the Bluetooth module correctly. |

### 

### 8.3 Integration Tests

Once each part is tested the entire system together must be tested. The following are the test cases that combine each software component and combine the high level software to the hardware device.

|  |  |
| --- | --- |
| **Test Name** | **IT01: Software Integration Test: Device Commands** |
| Objective | Test the ability of the mobile application to send commands to the Microcontroller |
| Requirements | * Microcontroller * HC06 module * SPARC Mobile Application * LED indicators |
| Procedure | Similar to the Device Controls Test Case, commands will be sent to the HC06. Instead of an LED the module will be connected to the microcontroller. Each command sent from the application will be inputted through the HC06 into the microcontroller and mapped to a specific pin for each command. |
| Results Verification | When a command is entered the LED indicator connected to the mapped I/O pin will light up is success. |

|  |  |
| --- | --- |
| **Test Name** | **IT02: Software to Hardware Integration Test: Device Commands** |
| Objective | Test the device commands directly to the motors |
| Requirements | * Microcontroller * HC06 module * SPARC Mobile Application * SPARC prototype |
| Procedure | The device commands will be entered on the application, sent to the microcontroller via the HC06. The maped pins will turn on motors as per describe by the processor program. |
| Results Verification | When the Up arrow is pressed the SPARC will move forward, Down the SPARC will move backward, and Left and Right accordingly. |

|  |  |
| --- | --- |
| **Test Name** | **IT03: Device Scheduling and Autonomous Control** |
| Objective | Verify that when the device is scheduled on the mobile application, the device will autonomously move via the set path algorithm. |
| Requirements | * Microcontroller * HC06 module * SPARC Mobile Application * SPARC prototype |
| Procedure | The schedule will be set on the application to move in one minute. The schedule will be sent to the microcontroller and stored |
| Results Verification | The SPARC will begin its movement in five minutes, moving to a specified area which it is taught before hand. A Notification is received on the application that the SPARC has moved according to schedule. |

## 8.4 Testing Overview and Results

The following are the results for each test case listed in the sections above. Test cases will be mark as Passed, Failed and Pending. At this point in the development cycle of the SPARC most of the test cases will be in pending as the prototype has not yet been built.

Table 11 Hardware Test Case Results

|  |  |
| --- | --- |
| **Test Case** | **Results** |
| **Hardware Tests** |  |
| PwT01: Voltage Testing | Pending |
| MT01: Pin Mapping Test | Pending |
| IMT01: Bluetooth Module Testing | Pending |
| PMT01: Movement Testing - Concrete | Pending |
| PMT02: Movement Testing - Grass | Pending |
| PMT03: Movement Testing - Gravel | Pending |
| PMT03: Movement Testing – Wet Concrete | Pending |
| PMT03: Movement Testing – Wet Grass | Pending |

Table 12 Software and Integration Test Case Results an

|  |  |
| --- | --- |
| **Test Case** | **Results** |
| **Software Tests** |  |
| AST01: Virtual Driveway Functionality | Pending |
| AST02: Virtual Driveway Reallocation | Pending |
| AST03: Algorithm Functionality | Pending |
| AST04: Value Iteration | Pending |
| MAT01: Sign In | Pass |
| MAT02: Sign Up | Pass |
| MAT03: Bluetooth Connectivity | Pending |
| MAT04: Device Controls | Pending |
| PPT01: Movement Controls | Pending |
| **Integrated Tests** |  |
| IT01: Software Device Commands | Pending |
| IT02: Hardware Device Commands | Pending |
| IT03: Scheduling and Autonomous Control | Pending |

# 

# 9.0 Administrative Content

## 

## 9.1 Milestones

The project milestones have been recorded from the beginning of the product development cycle and the projected project milestones for the next semester where the prototype will be developed. The following, Table 13, documents these milestones for Senior design 1:

Table 13 Milestones SD1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Task | Finish By | Status | Responsibility |
| Senior Design 1 | | | | |
| 1 | Brainstorming Assignment | 1/16/2018 | Completed | N/A |
| 2 | Group Formation | 1/18/2018 | Completed | N/A |
| 3 | Initial Project Documentation | 1/28/2018 | Completed | N/A |
| 4 | Microcontroller Selection | 1/26/2018 | Completed | N/A |
| 5 | Finalize Parts List | 2/2/2018 | Completed | Sean |
| 6 | Microcontroller Pin Mapping | 2/9/2018 | Researching | Sean |
| 7 | Mobile Application Class Diagram | 2/9/2018 | Completed | Alexis |
| 8 | Interface sensors and modules to Microcontroller | 2/16/2018 | Researching | Chris |
| 9 | Final Documentation: Table of Contents Finalized | 2/23/2018 | Completed | N/A |
| 10 | Pathing Algorithm Flowchart/UML Diagram | 3/2/2018 | Completed | Wyatt |
| 11 | SPARC Chassis Design | 3/9/2018 | Researching | Sean |
| 12 | Final Documentation: Draft 1 | 3/23/2018 | Completed | N/A |
| 13 | Mobile Application interface with Microcontroller via Bluetooth Module | 4/6/2018 | Pending | Alexis |
| 14 | Basic Movement Control via Microcontroller | 4/6/2018 | N/A | Chris |
| 15 | Proof of Concept Prototype Finalized | 4/6/2018 | N/A | N/A |
| Working on Final Documentation | | | | |
| 16 | Final Documentation: Finished Copy | 4/26/2018 | Completed | N/A |

The projected milestones and basic timeline for the next semester of the product development cycle is documented in table 14 below:

Table 14 Milestones SD2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Senior Design 2 | | | | |
| 17 | Finalize PCB Designs and Order | 6/1/18 | N/A | Sean |
| 18 | Design Home Base and Build | 6/8/18 | N/A | Sean |
| 19 | Finalize Physical Movement System Design and Build | 6/15/18 | N/A | N/A |
| 20 | Integrate Pathing Algorithm to Microcontroller Movement Control Code | 6/29/18 | N/A | Wyatt |
| 21 | Finish Build Initial Prototype | 7/6/2018 | N/A | N/A |
| Testing and Revisions | | | | |
| 22 | Finalize Prototype for Presentation | TBA | N/A | N/A |

## 9.2 Budgeting and Final Cost Analysis

The SPARC project has a generous maximum budget of $2000 from Selah Group Florida Inc. This budget is just a maximum that we do not expect to exceed. Below is the basic cost analysis for each major section or component of the SPARC. Each value is an estimate based on the parts list.

Table 15 Cost Analysis

|  |  |
| --- | --- |
| **Main Components** | **Estimated Budget** |
| Chassis and Trash Can containment parts | ~$200.00 |
| Wheels and other Mechanics | ~100.00 |
| Printed Circuit Boards | ~$50.00 |
| External Hardware Modules | ~70.00 |
| Miscellaneous Hardware parts | ~$100.0 |
| Camera | $69.00 |
| Power Supply Components | ~$200.00 |
| Microcontroller and other electronics | ~$20.00 |
| Estimated Total: | ~$809 |

## 

## 9.3 Project Group Roles

For a project like this to succeed, the division of labor was crucial. Each member of the group chose a part of the project to manage based on their individual strengths. The computer engineering majors managed the development of the three software aspects of the project including the low level base microcontroller programming, developing a reflexive pathing algorithm that utilizes computer vision and the development of the mobile application as the main interface for the user to control and monitor the SPARC. The electrical engineering major managed all of the hardware and electrical aspects of the SPARC which includes the chassis, motors, power management and PCB designing. Though the project is split between the team each member will assist in each part of the SPARC to maximize success and learning.

Table 16 Group Roles

|  |  |
| --- | --- |
| Drayton, Alexis | Mobile Application |
| McGarvey, Sean | Electrical and Hardware Components |
| Schmitz, Wyatt | Computer Vision and Pathing Algorithm |
| Vento, Chris | Microcontroller Programming |

# 10.0 Conclusions

The SPARC overall is made with easing the user in mind. It is a product meant to fix a simple problem by completely a simple task autonomously and efficiently with very little user input or interference in the vain of a present and future moving closer full autonomy of mundane tasks and chores to increase human progression and productivity.

The design and development process in creating a fully functional product is the basis of every engineering process. Researching by finding similar products, and weighing the benefits versus the consequences of using on device or component over the other, working as a group and dividing the development management in terms of individual strengths, building a prototype and testing it; all basic components of the engineering process. Upon finalizing the design and prototype some revisions to further the project evolution in the future were thought of and documented in the following section.

## 10.1 Future Revisions

The following feature concepts for the SPARC were created during the design after the requirements specification were laid out. In order to create a minimum viable product that fully meets the requirements specification laid it for the SPARC, no efforts were made to including these features in the final design. The time and financial budget for the development of the SPARC were not given to include these features, and to make a large effort to append them to our final product would be an egregious feature creep on the SPARC. These features were included in this section in the case that future iterations of the SPARC would benefit from having them implemented into its design.

### 10.1.1 SPARC Weighing Scale

Trash collection schedules for Brevard County waste management are not cut-and-dry, and the actual trash collection occurs during a range of times and not at discrete intervals. Including a digital weighing scale in the SPARC would allow the SPARC to tell with accuracy when the trash has been collected from its bin. The SPARC would record the weight of its trash-load before moving to the delivery point, and would periodically awaken from low power mode to check to see if the current weight on its scale is significantly less than the weight initially recorded for that collection cycle.

Having this input allows the SPARC to return to its home base promptly after the trash has actually been collected rather than waiting for the entire range of time that the trash could possibly could be collected ends. Shortening this time frame could potentially allow future iterations of the SPARC to have a smaller battery capacity, as this version of the SPARC would spend less time outside and more time on the home-base system charging. This would drive down the overall cost to produce the SPARC, and in turn reduce the cost for consumers to purchase SPARC systems. Having a low cost system was one of our main design goals, and so while this feature would have had a very high utility in our design for the SPARC it was abandoned regardless to avoid feature creep to focus on our main specifications.

### 10.1.2 Wireless Communication via WIFI

The final design of the SPARC currently only supports Bluetooth communication between the Android Mobile App and the SPARC system. In this build there is currently is no way for the user to control the settings of the SPARC unless they have a Bluetooth connection to it. Naturally, with the limited range of Bluetooth this makes it impossible to do so when the user is not at home. This could potentially cause problems should they forget to disable the SPARC before going on vacation, for example. Ease-of-use is one of the main design goals of the SPARC, and situations where the user will have to reset the SPARC’s pathing are to be avoided.

This feature would implement an additional method of wireless communication, by creating a local area network between an access point in the user’s home and the SPARC through WIFI. The user would then be able to push commands to the SPARC through an internet connection to alter settings such as delivery schedules, and toggling the general operations of the SPARC in case the user is not home for unexpectedly long durations of time.

Despite the utility in this feature, the added time workload for researching and developing system like this was high, as none of the members of our project team were experienced in wireless communication networking. The cost-benefit analysis of including this feature were not great enough to warrant adding this additional overhead when it was not even one of our specifications for the SPARC.

### 10.1.3 GPS Tracking & Anti-theft

The inclusion of the GPS module was a later addition to the design of the SPARC, far after solidifying our design specifications. It was not considered early in our initial brainstorms, as GPS was not a technology used in many of the previous projects examined for its lack of relative accuracy in measurements in small spaces. However, the SPARC has a large profile and works in a more generous space more similar to self-driving vehicles which utilize this technology in their array of input sensors.

As mentioned in the previous subsection, the SPARC system has no methods of wireless communication besides Bluetooth. For SPARCs that are outside the range of Bluetooth communication with the user’s smartphone, no GPS data can be relayed. Implementing a GPS tracking system would have likely implemented a connection to a cellular network or a similar wide area network. With this connection established, the SPARC’s current GPS location would be pushed to the user’s smartphone on command. This would enable the user to determine whether their SPARC has been stolen, or simply lost due to some critical error in the pathing and/or sensors. The user would also be able to disable the SPARC’s delivery schedule in either of these cases using this remote connection to ensure that they do not need to do any additional set-up when retrieving the device besides placing it back near its home-base unit.

This feature was not implemented in the final design for similar reasons to the WIFI communication feature. None of our group members were sufficiently experienced in long range communication networks, so implementing the feature would have necessitated a large amount of research and development. Cost-benefit analysis determined that including this feature was far too burdensome given it not even being one of our specified requirements. With the GPS module though, this feature was determined to be one to keep in mind for possible future iterations of the SPARC system.

### 

### 10.1.4 Adverse Weather Exceptions

Conceivably, the user would not want the SPARC to roll headlong into a category 5 hurricane, and in Brevard County FL it’s not unlikely that one may occur during the lifecycle of a SPARC system. Trash would likely not even be collected during such inclimate weather.

Another useful feature not included in our final design would be pushing reports on inclimate weather to the SPARC. Should services like Waste Management be suspended due to adverse weather or otherwise, the mobile app will have a background function that pushes this information through bluetooth to the SPARC, which will cause it to postpone any deliveries that would take place during that gap in trash collection. Additionally, the app would be able to push general information about tropical storms, tornado warnings, and smaller hurricanes. Of these weather anomalies, the user would be able to select settings in the smartphone whether or not they would like the SPARC to operate during these conditions.

These sort of weather conditions are not exceptionally common, and the combining of information from the internet on the smartphone Android application was not familiar to any one of our team members, so the workload would have been non-trivial. With these factors and its lack of being on the requirements specifications, the feature to push adverse weather conditions to the SPARC was not included in our final design.

Appendix A References

[1]"History | iRobot", *http://www.irobot.com*, 2018. [Online]. Available: http://www.irobot.com/About-iRobot/Company-Information/History.aspx. [Accessed: 26- Apr- 2018].

[2]D. Klein and P. Abbeel, “Lecture 8: MDPs I,” *UC Berkeley CS188 Intro to AI*. [Online]. Available: http://ai.berkeley.edu/lecture\_slides.html.

[3]J. Tompkin, J. Hays, and M. Black, “Introduction to Computer Vision,” *CSCI 1430: Introduction to Computer Vision*. [Online]. Available: https://cs.brown.edu/courses/cs143/.

Appendix B Permissions

Shaddowffax, “Edge Detection”, (https://commons.wikimedia.org/wiki/File:Edge\_detection.png), <https://creativecommons.org/licenses/by-sa/4.0/legalcode>

QueSera4710 (https://commons.wikimedia.org/wiki/File:SRT\_Shape\_Recognition\_Technology.png), „SRT Shape Recognition Technology“, <https://creativecommons.org/licenses/by-sa/3.0/legalcode>