# Software Requirements Specification

**for**

# Project Bowser

**Version 1.1**

**Prepared by**

**Group Name:** Let My People Code

| | | |
|---|---|---|
| Moshe Carmeli | 4770954 | moshecmeli@gmail.com |
| Chris Nelson | 5176169 | sciencectn@gmail.com |
| Adam Ehrlich | 4847885 | adamsehrlich191@gmail.com |
| Alex Sarraf | 4913604 | alex.sarraf@gmail.com |
| Daniel Imberman | 4301487 | danielryan2430@gmail.com |

| | |
|---|---|
| Instructor: | Chandra Krintz/Tim Sherwood |
| Course: | Computer Science 189 |
| Lab Section: | *Wednesday 6 PM* |
| Teaching Assistant: | *Geoffrey Douglas* |
| Date: | March 4, 2014 |

| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
|---|---|---|---|
| 1.0 | Moshe Carmeli, Chris Nelson, Adam Ehrlich, Alex Sarraf, Daniel Imberman | First version | 2/10/2014 |
| 1.1 | Moshe Carmeli, Chris Nelson, Adam Ehrlich, Alex Sarraf, Daniel Imberman | Updated user stories and other minor details | 3/3/2014 |

# 1 Introduction

*Project Bowser aims to seamlessly introduce Android cellphone users to the world of robotics by serving as an adapter for ubiquitous cellphones to function as autonomous robots. In this section of the documentation, the goals and abstract project outline are detailed. These serve to inform the reader of the objectives of Project Bowser and how these objectives will be met.*

## 1.1 Document Purpose

The purpose of this report is to fully document Project Bowser v1.0 which will be carried out by Let My People Code. This SRS aims to provide guidelines for how the project will be deemed successful and how these concepts and ideas will be created, effectively. Furthermore, this document will serve as an agreement between Let My People Code and Qualcomm as to what this project will entail such that both parties may understand the specifications, requirements, and design of Project Bowser.

Note: In the event of dispute between both parties (Let My People Code and Qualcomm), this document will serve as a reference for mediation. No significant changes in Project Bowser's plans or workflow will take place without proper revision of this document and consent from both parties.

## 1.2 Product Scope

Project Bowser aims to utilize the processing power of the host Android phone to perform useful tasks by using the rovers' physical equipment. There are a large number of applications which could be developed, many of which would be revolutionary for cell phone users everywhere. Project Bowser hopes to bring some of the most ambitious and promising ideas to the Mini-Dragon Rover so that it can be showcased and popularized for public consumption.

The developed code and features will all be open-sourced in order to facilitate manipulation and advancement by motivated contributors. The open-source environment and cost effective construction cost of the MDR will make Project Bowser a hotbed for innovation and education.

In particular, our team will be utilizing Qualcomm's Gimbal Bluetooth beacon and Vuforia computer vision library to create an autonomous navigation system for a 3D printed rover. In addition, the rover will be able to be controlled remotely through a web interface.

## 1.3 Intended Audience and Document Overview

This Document is intended for all readers interested in learning about Project Bowser. Section 1.4 serves as a reference for explanations and definitions of terms, abbreviations, and acronyms. Section 2 provides an abstract overview of the project and its mechanics while Section 3 provides details about how the functionality and features will be implemented and created. Other Non-functional requirements are detailed in Section 4 and Section 5.

## 1.4 Definitions, Acronyms and Abbreviations

**3D Printed** – Manufactured through Additive Manufacturing with a 3D printer
**Additive Manufacturing** – Layering materials in different shaped to create 3D objects
**Android** – An operating system based on the Linux kernel used in many smartphones
**Kernel** – a computer program that manages input/output requests from software and translates them into instructions for the CPU
**Mini-Dragon Rover (MDR)** – The 3D printed robotic carriage for Android cellphones on which Project Bowser is focused
**Bluetooth LE (BLE)** – Bluetooth Low-Energy, the technology used for location beacons
**IOIO** – Pronounced yo-yo. A library for Android allowing you to communicate with hardware with the help of a special accessory board (the IOIO board).

## 1.5 Document Conventions

Some titles are in **Bold** text.

## 1.6 References and Acknowledgments

Mentor: Chad Sweet, Director of Software Engineering at Qualcomm

# 2 Overall Perspective

## 2.1 Product Perspective

The main purpose behind Qualcomm's Mini-Dragon Rover use existing technology that everyone has (smartphones) and interface it with a 3D printed rover. Our role in this product is to create interesting demos showing off the capabilities of the rover and smartphone combined.

## 2.2 Product Functionality

These are the possible major functions, (in order of most to least likely to be implemented):

- Indoor position awareness using Bluetooth LE beacons
- Target following, by tracking Vuforia markers
- Remote controlling of the MDR over the web (could be used as a security monitor)

## 2.3 Users and Characteristics

The initial users of Project Bowser's software will be early adopters who wish to get the most functionality out of their phone as possible and developers (innovators) who wish to implement their own apps and services on the MDR.

Qualcomm's ultimate goal with this MDR is to create a simple robot which takes advantage of the modularity of 3D printing. Users will be able to create their own APIs and custom parts to build cheap custom rovers that can perform a wide array of functions.

Eventually, these apps will reach the early majority which already own an Android phone, and will do more consumption than contributing. These users will wish to show their friends the newest technologies and take advantage of the newest offerings in entertainment/utility for their phones. However, with increased popularity, more developers will be inclined to take up developing Android apps which utilize the MDR (especially since the code is open-source).

## 2.4 Operating Environment

Project Bowser will mostly consist of Android OS 4.3 (minimum) applications which will utilize the functionality available on the Mini-Dragon Rover. The applications will interact with the MDR's IOIO board and communicate with other devices such as Bluetooth LE beacons to perform tasks autonomously and interactively.

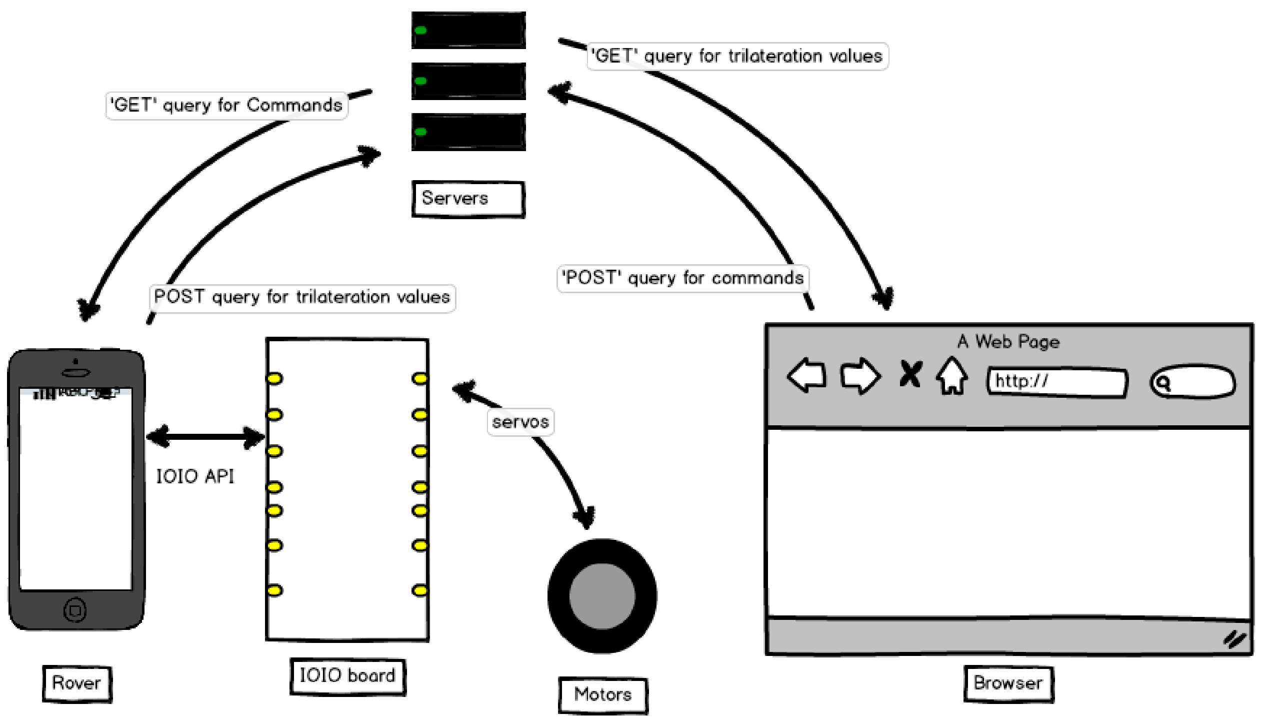
'GET' query for trilateration values
'GET' query for Commands
Servers
'POST' query for commands
POST query for trilateration values
A Web Page
http://
servos
IOIO API
Rover
IOIO board
Motors
Browser

**Figure 1** Interface diagram for phone on MDR

## 2.5 Design and Implementation Constraints

Some of the significant constraints on Project Bowser are:

- Since the MDR is still considered a part of Qualcomm's research branch, certain important aspects have not yet gained legal clearance for public use.
- Critical application for Bluetooth LE detection has not yet been released for android.
- Need to use Android 4.3 or above
- Need phones capable of Bluetooth LE
- Need to communicate with MDR peripherals through the IOIO board

## 2.6 User Documentation

While our app will be relatively simple to use, it introduces new technologies, new concepts, and new functionalities which users will need to learn. We will provide comprehensive documentation online such that the transition from simply Android phone to MDR will be quick and painless for users of all levels.

Online Help:

- Tutorial Videos
- Step-by-step guides for using the app
- FAQs and answers
- Javadoc

## 2.7 Assumptions and Dependencies

Project Bowser makes the following assumptions:

- The user needs to supply their own compatible Bluetooth LE beacons
- The user's phone needs to be compatible with IOIO boards
- The environment that the rover operates in needs to meet certain criteria in order to be navigated correctly (e.g. shape of room)
- The user needs to have an MDR

# 3 Specific Requirements

## 3.1 External Interface Requirements

### User Interfaces

**Object Retrieval App Screen**:
This screen will present the user with a red block indicating the location of their host MDR within a rectangle indicating the dimensions of the room. The dimensions corners are marked by purple circles indicating the reference beacons set up around the room (in a rectangular room). The other dark blue circles on the screen indicate the estimated location of target devices which the MDR can navigate to, pick up, and retrieve. The faint blue circles around all devices indicate the margin of error for location calculations.

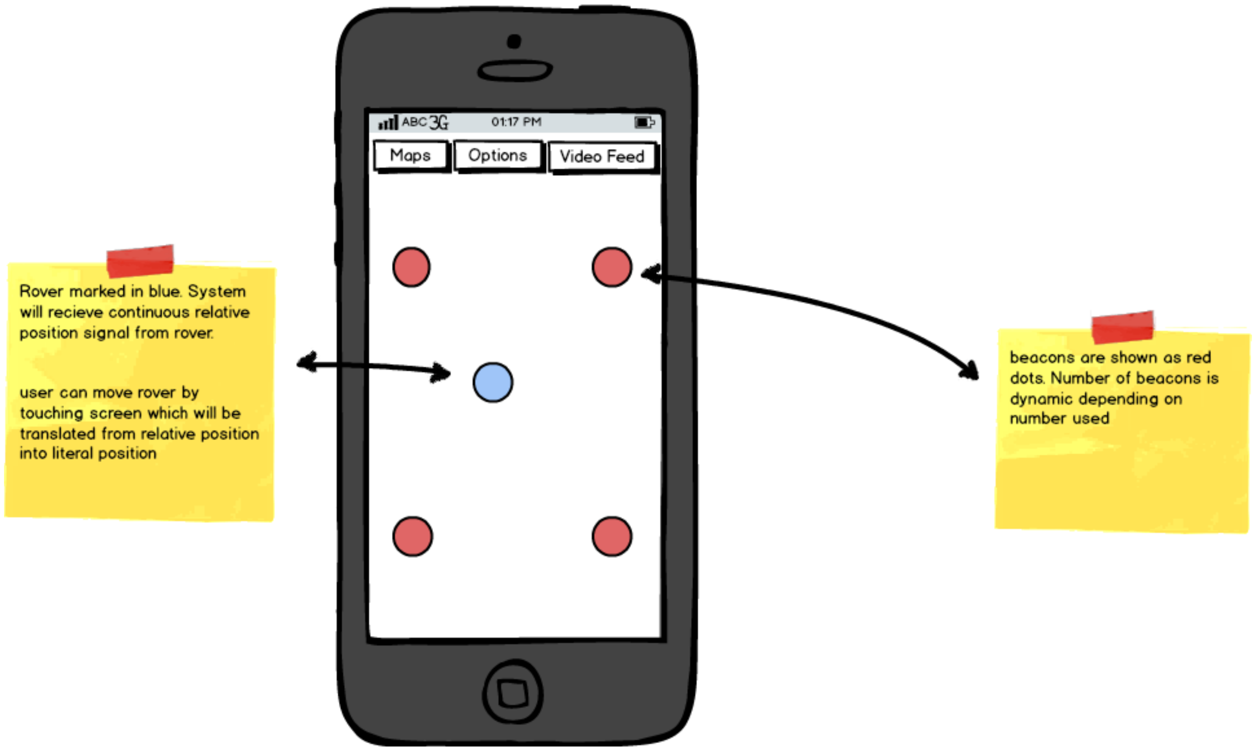
ABC 3G
01:17 PM
Maps
Options
Video Feed
Rover marked in blue. System will recieve continuous relative position signal from rover.
user can move rover by touching screen which will be translated from relative position into literal position
beacons are shown as red dots. Number of beacons is dynamic depending on number used

**Figure 2** Rover Location Screen

**MDR Remote Control**:
The MDR can be controlled via this remote control which provides an easy to use interface for all the MDR peripherals available. This will likely be the same screen presented to the user when they use the MDR for other features such as time trial games or picture taking.

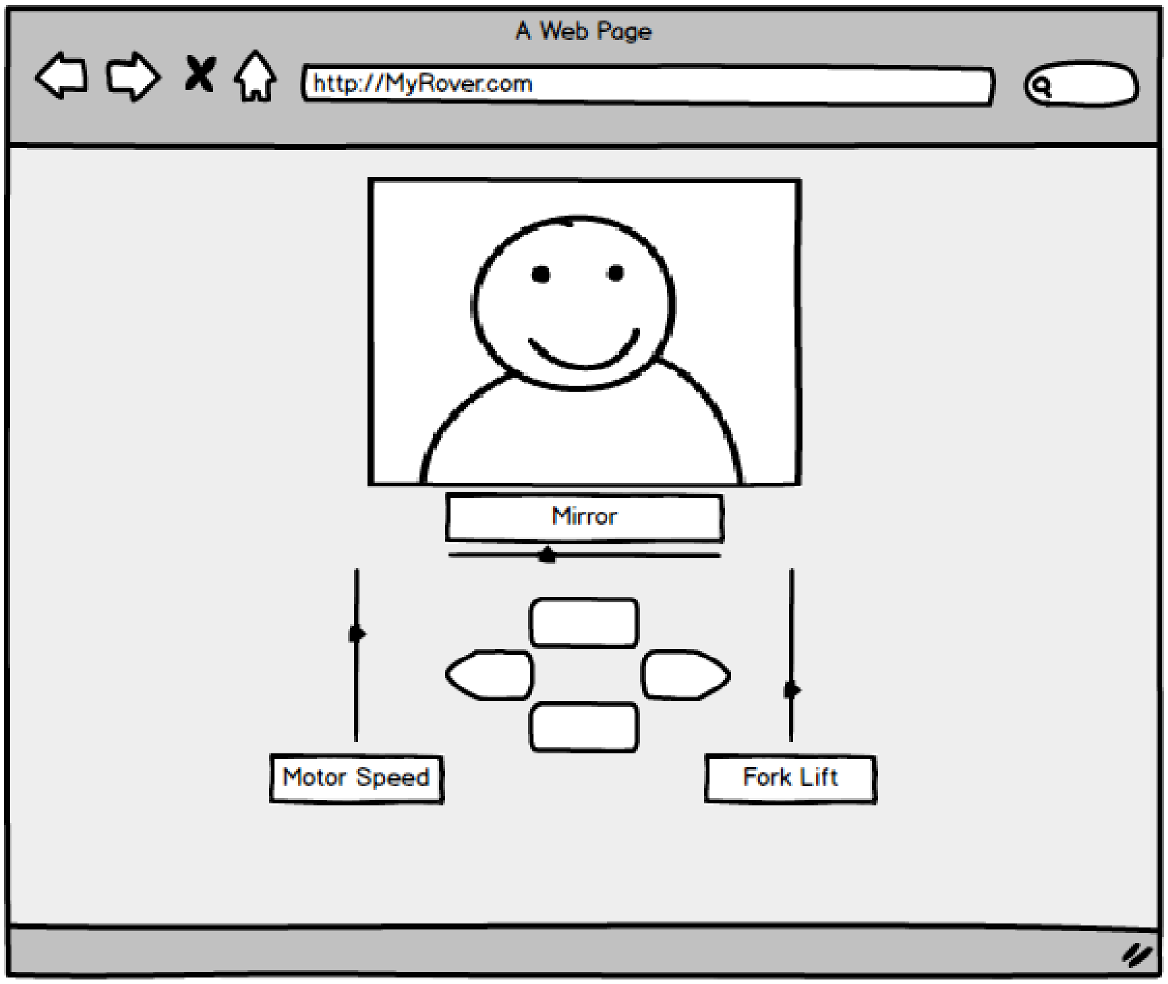

**Figure 3** MDR remote control screen

## Hardware Interfaces

There are three connectivity interfaces we will be using:

**App <--> Rover (through IOIO board):** This works over either Bluetooth or USB. We will be using USB, since the Bluetooth radio will be occupied with scanning for beacons. The basics will include sending PWM signals to control the speed of each wheel. Extra peripherals might require more methods of hardware communication.

**App <--> Bluetooth LE:** This is a standard Android library included with API 18 or higher. Currently, the only functionality we need is scanning to listen for the beacon broadcasts. Each broadcast consists of an ID an RSSI.

**App <--> Camera:** We will likely be using Qualcomm's Augmented Reality SDK, known as *Vuforia*, to analyze images from the camera. The robot will be able to recognize objects or obstacles with the help of special stickers that Vuforia is designed to recognize.

## Software Interfaces

- **IOIO Libraries:** The Android applications need to communicate with the MDR peripherals via the IOIO board in the MDR carriage. The signals passed will control the servomotors of the MDR.
- **Bluetooth LE:** The Android applications must identify its location via Bluetooth LE signals. These signals will provide a certain strength, which, when compared with reference signals (thought some complicated modeling), will reveal the relative location of rover between reference beacons..
- **WiFi:** The Android phone will speak to a server which will link it to a web based remote. This will offer the incredible benefit that users on any device can control their robot remotely via the web.
- **Android:** The Android app running on the rover's phone must communicate with the Android operating system in order to execute the desired low-level operations.
- **Python/Nginx Server**: The Android app communicates with a web server which is built on top of the Pyramid Python framework and Nginx HTTP web server. This web server is also the main interface that the user uses to give commands to in addition to controlling the rover through the web interface.

## Communications Interfaces

- **WiFi -** Used for the remote control interface
- **Bluetooth LE -** Only receives but never sends; used to get the signal strength of each beacon and tell which one is which
- **IOIO Board -** Android can't directly communicate with the rover's hardware, so it needs the IOIO board as an intermediary

## 3.2 Functional Requirements

### User Interface

- The user interface must be able to take in task requests from the user
- The user interface must allow the user the option independently manipulate the MDR

### Autonomous Execution

- The features must process user task requests and perform them as autonomously as possible (e.g. locating itself without walking the user through some procedure)
- The features must be able to forward human interface requests in real time to provide dynamic maneuvering of the MDR

## 3.3 Behaviour Requirements

### User Stories

*Difficulty Key:*
1 - can be done in 1 day/really quickly
2 - can be done in 1 or 2 days
3 - can be done in 3 to 5 days
4 - can be done in a week or more

| User Stories | Difficulty |
|---|---|
| As a user, I would like to move my rover forward,backwards,left,and right remotely. | 2 |
| As a user, I would like to control the motor speed of the servos. | 2 |
| As a user, I would like to set the heading of my rover (0-360º) | 2 |
| As a user, I would like a video feed of where my rover is facing. | 3 |
| As a user, I would like to control the forklift to move up and down. | 1 |
| As a user, I would like to know where my rover is from its trilateration. | 2 |
| As a user, I would like to control the mirror facing the camera view left and right. | 1 |
| As a user, I would like to take pictures and store them for later. | 4 |
| As a rover, I would like to get the distance from 1 beacon. | 3 |
| As a rover, I would like to identify with multiple Bluetooth beacons. | 4 |
| As a rover, I would like to trilaterate from multiple Bluetooth beacons. | 4 |
| As a rover, I would like to reduce noise of beacons. | 4 |
| As a user, I would like to calibrate my rover to new beacons. | 4 |
| As a rover, I would like to recognize visual targets. | 4 |
| As a rover, I would like to understand my current location in relation to a visual target. | 3 |
| As a rover, I would like to move toward a target but not too close. | 3 |
| As a gamer, I would like to start a time trial clock for gaming. | 1 |
| As a rover owner, I would like secure access to my own rover. | 3 |

# Other Non-functional Requirements

## 4.1 Performance Requirements

Project Bowser has the following performance requirements:

- Position resolution with Bluetooth beacons should be completed in less than 5 seconds
- Connecting over WiFi should take less than 10 seconds
- The MDR will not move fast enough to cause serious physical harm to furniture or people
- The MDR should be able to maneuver in different terrains (carpet, tile, etc.)

## 4.2 Safety and Security Requirements

**Safety**

- The MDR shouldn't contain any toxic materials
- *(Other than that the MDR is not capable of causing much damage)*

**Security**

- Some basic authentication might need to happen if the MDR is able to be controlled over the Internet to prevent intrusion attempts.

## 4.3 Software Quality Attributes

**Reliability**

- Since the MDR will be navigating based on radio measurements, which are inherently noisy, it needs to be robust in the face of noisy values or unreliable data
- As far as we know, the user is supplying information on the location of each beacon, so the MDR must respond gracefully in the face of bad input.

**Reusability**

- We want a somewhat generic navigation framework that will work in a variety of scenarios, as opposed to being hardcoded for our specific testing setup
- It needs to be pretty easy to set up, otherwise no one will want to use it