March 4, 2014

Project Bowser Specification
Version 1.0

Prepared for Chandra Krintz
Let My People Code

Team Members
Alex Sarraf
Adam Ehrlich
Daniel Imberman
Christopher Nelson
Moshe Carmeli

Table of Contents

Introduction

Product Overview

Project Bowser is designed to introduce Android users to the world of robotics by providing a rover which transforms unused phones to powerful robots. These robots will be able to perform various tasks, such as indoor positioning, target following, and responding to remote commands offered via the internet.

Definitions and Initialisms

- **BLE:** Bluetooth Low Energy, part of the Bluetooth 4.0 specification. Very low throughput (at most 0.3mbps) but designed to use very little energy.
- **MDR:** Mini Dragon Rover, stuff
- **RSSI:** Received Signal Strength Indicator, a measure of wireless signal strength (usually in dBm).
- **Trilateration:** Not to be confused with *triangulation* (which is much easier to do). Trilateration is a method of determining your own location when you know your distance to at least 3 locations.
- **Servo Motor:** A motor with extra circuitry to control its speed. Comes in two flavors: *open loop*, or continuous rotation, where you can only control the speed, and *closed loop*, where you can control the exact position.
- **PWM:** Pulse Width Modulation. The details of this are unimportant; all you need to know is that PWM is the protocol used to communicate with servos and tell them how to move.
- **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).
- **Vuforia:** Qualcomm's computer vision library primarily designed for augmented reality. Can track targets and superimpose 3D images on top of them (we probably won't be using the latter feature).

Document Overview

This document begins by providing a concise overview of the components involved in Project Bowser. Following this, the document will provide design specifications, which provides a high-level overview in the form of flowcharts and diagrams.

Core Components

At the bare minimum, the rover will require these components in order to operate, no matter what the application is.

Android Phone on Rover

An Android phone will be mounted on the rover providing the computational power necessary to communicate with the server (if necessary) and forward the necessary signals to the IOIO board. It also provides all the sensory input, such as positioning using BLE or Vuforia, object awareness using the camera and Vuforia, and the digital compass to detect its heading.

Relevant User Stories	Acceptance Test (Phone Perspective)
As a user, I would like to set the heading of my rover (0-360º)	The phone should be able to use its built in compass functionality to check its current heading.
As a user, I would like a video feed of where my rover is facing	The phone needs to successfully transmit a video feed to the controller over the internet.
As a user, I would like to take pictures and store them for later	The phone should be able to take the picture and send it to a server.
As a rover, I would like to get the distance from 1 beacon	The distance matches what you would measure with a tape measure (approximately)
As a rover, I would like to reduce noise from beacon inputs	Trilateration results stay within a preset margin of error
As a rover, I would like to trilaterate from multiple Bluetooth beacons	Phone should be able to identify and distinguish multiple beacons based on a unique attribute
As a rover, I would like to recognize visual targets	The phone pinpoints the target on a video feed
As a rover, I would like to understand my current location in relation to a visual target	The phone can estimate where the target is
As a rover, I would like to move toward a visual target but not too close	The rover moves smoothly toward a target and doesn't go AWOL

As a rover owner, I would like secure access to my own rover	Rover should communicate over a secure path to the phone
As a user, I would like to calibrate my rover to new beacons.	Rover should detect the signal of a beacon correctly at a known distance accurately.

IOIO Board

The IOIO board's job is simply to accept incoming signals from the Android phone and convert them to PWM signals to control the servos.

Relevant User Stories	Acceptance Test (IOIO Perspective)
As a user, I would like to move my rover forward,backwards,left,and right remotely	A command (forward/backward/left/right) successfully received by the IOIO board should be observable on the servomotors.
As a user, I would like to control the motor speed of the servos	A setting for servomotor speed successfully received by the IOIO board should be observable on functioning motors of the MDR.
As a user, I would like to control the forklift to move up and down	A command successfully received by the IOIO board should be observable on the MDR's forklift
As a user, I would like to control the mirror facing the camera view left and right	A command successfully received by the IOIO board should be observable on the MDR's mirror

MDR

The MDR Rover is the physical carriage which holds an Android phone in it (interfaced with the rover through the IOIO board) and provides physical functionality through its structure. The motors included allow the ability to move the rover forward/backward/left/right, move the camera mirror left/right, and move the forklift up/down.

Relevant User Stories	Acceptance Test (MDR Perspective)
As a rover, I would like to move toward a target but not too close	The rover should be responsive enough to commands to be able to slow down instantly.

Ancillary Components

These components will probably be required for most applications that we do, but some will be optional depending on the application.

Remote Control Interface (User's End)

The remove control interface will allow the user to control the rover from any location through a web interface. The interface will be accessible from a computer or a mobile phone. Through the interface, the user will be able to control most of the features of the rover (basic movements, mirror panning, forklift) and initialize more complicated functions (such as target seeking, self-positioning).

Relevant User Stories	Acceptance Test (RC Perspective)
As a user, I would like to know where my rover is from its trilateration.	The interface reports the approximate location correctly
As a user, I would like to control the forklift to move up and down.	The user hits a button, the interface relays the necessary signals through the phone
As a user, I would like a video feed of the rover	The user hits a button, the interface relays the necessary signals through the phone
As a user, I would like to set the heading of my rover (0-360º)	The user hits a button, the interface relays the necessary signals through the phone
As a user, I would like to take pictures and store them for later.	The user hits a button, the interface relays the necessary signals through the phone
As a user, I would like to move my rover forward, backwards, left, and right remotely	The user hits buttons, the interface relays the necessary signals through the phone
As a user, I would like to control the motor speed of the servos	The user selects a speed, the interface relays the necessary signals through the phone

Server (Back End)

In order to allow the rover to be controlled remotely, a server will be set up to facilitate communication between the user and the mobile phone controlling the rover. The server will be running a WSGI application using Python (Pyramid framework) on top of a Nginx HTTP proxy utilizing uWSGI to mediate the two pieces of software. The server software will communicate to the user and the MDRs via GET and POST requests which will communicate with a back-end MySQL database and a hierarchical file for storing images. In addition the server will implement the websocket protocol for video streaming capability.

Relevant User Stories	Acceptance Test (Server Perspective)
As a user, I would like to know where my rover is from its trilateration.	The server relays trilateration location from rover to interface and interface settings to phone.
As a user, I would like to move my rover forward, backwards, left, and right remotely	The server relays movement signals from the interface to the rover
As a user, I would like to control the forklift to move up and down.	The server relays movement signals from the interface to the rover
As a user, I would like a video feed of where my rover is facing.	The server continuously relays video from the rover to the interface and interface interactions to the rover
As a user, I would like to set the heading of my rover (0-360º)	The server relays interface interactions to the rover
As a user, I would like to take pictures and store them for later.	The server relays photos from the rover to the interface and interface interactions to the rover
As a rover owner, I would like secure access to my own rover.	The server relays communications in a secure manner

BLE Positioning

Using an array of BLE beacons positioned at points known to the rover beforehand, the rover will be able to get its approximate location indoors using the signal strength to each beacon and trilateration.

Relevant User Stories	Acceptance Test (Phone/Beacon Perspective)
As a rover, I would like to get the distance from 1 beacon.	RSSI is accurately converted to distance
As a rover, I would like to identify with multiple Bluetooth beacons.	Beacons must transmit unique attributes and phone must identify these
As a rover, I would like to trilaterate from multiple Bluetooth beacons.	Beacons must have similar transmit powers and phone must run calculation accurately

Vision System

Using Qualcomm's Vuforia API, we intend to track special markers that it is designed to recognize. We have two possible use cases for this:

- Locate a target (something the forklift picks up, for instance) and move toward it.
- Use it for additional positioning information by knowing where the marker is beforehand

Relevant User Stories	Acceptance Test (Phone/Camera Perspective)
As a rover, I would like to recognize visual targets	Phone must process video feed and identify targets in video using Vuforia
As a rover, I would like to understand my current location in relation to a visual target	Phone must use Vuforia library functions to estimate distance from target
As a rover, I would like to move toward a visual target but not too close	The phone must generate signals to move the rover smoothly toward the target (avoiding collisions)

Design Specifications

Overview

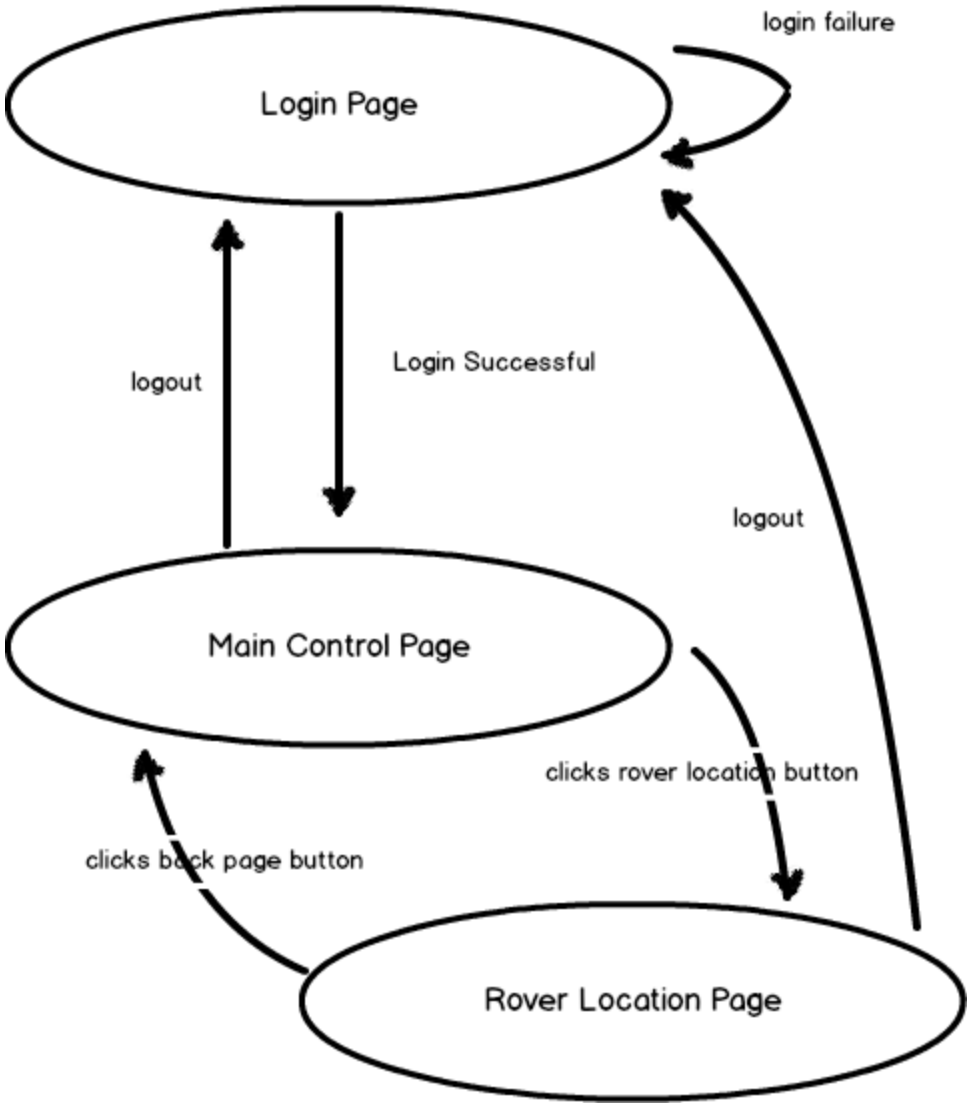


Figure 1: The figure above shows the high level expected interaction between the user and the Project Bowser service.The user is expected to visit our web portal and login with predefined user credentials (if they input the information correctly, they are presented with a dialog detailing the issue). Once the user successfully logins, they are redirected to the main control page where they are able to manually control the Rover and receive a live video feed from the phone in the carriage as well. From this main control page, the user can switch screens to the rover location page, where they can view the relative location of the rover between reference beacons (or switch back to the main control page). The user may also log out from both the main control page and rover location page.

Login Page

The login page allows the user to login using predefined credentials. If the user successfully logs in, then they will be redirected to the main control page, otherwise they will be prompted that their credentials were invalid.

User wants to Log In:

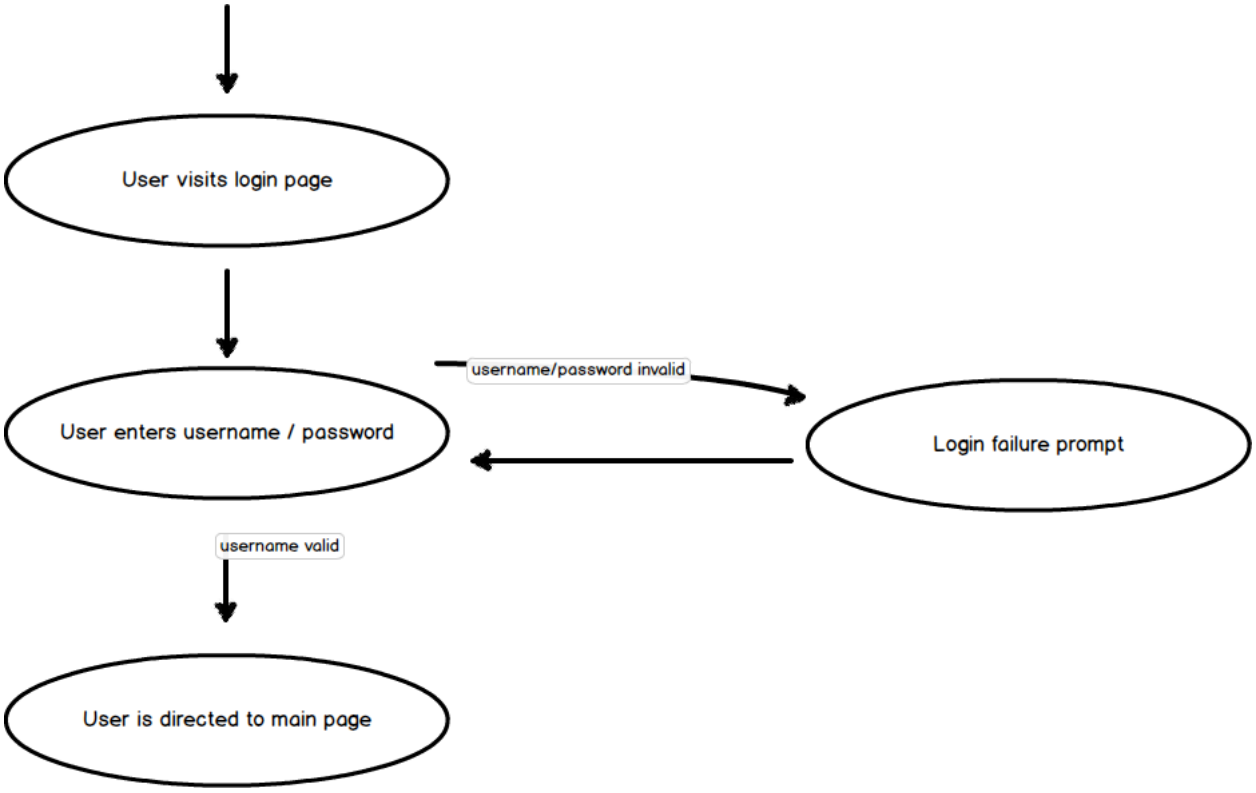


Figure 2: The Figure above portrays the expected state transitions a user will experience when logging into the Project Bowser web portal.

Remote Control Interface

The remote control interface allows the user to communicate with the phone on the rover, which is for rover control. The interface is accessed through a web server which allows you to perform operation on the rover but sending signals to the phone. This requires that the user logs in to the system first.

User wants Main Control Page:

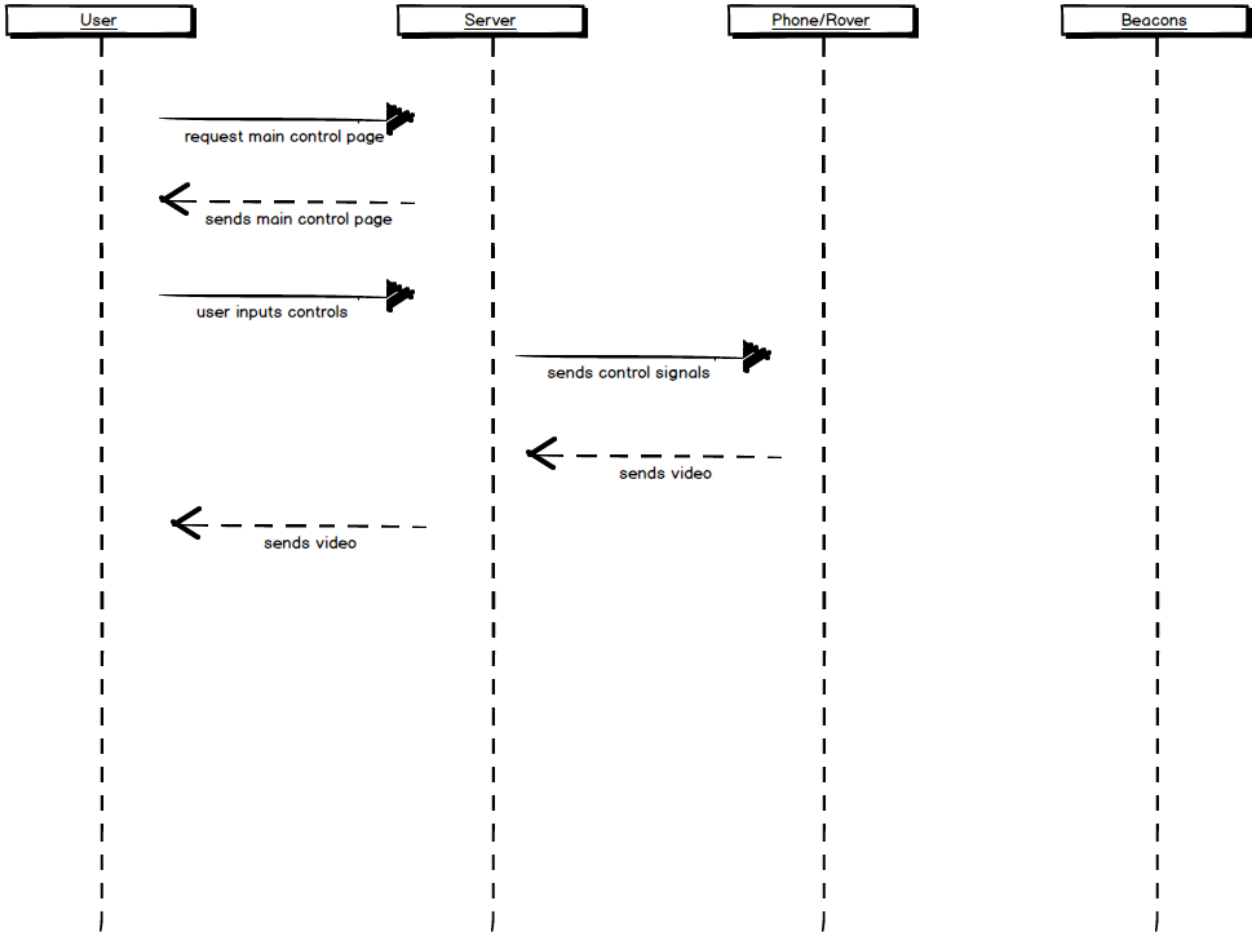


Figure 3: The figure above is the sequence diagram for a user who is successfully logged in and is either being redirected to the main page, or is switching back to the main control page from the rover location page. The video transmission (if enabled by the user) and control relaying is repeated endlessly while the user is on this screen.

Rover Control

The phone communicates with the rover through a USB cable connected to the IOIO board. Signals are received via the remote control interface, then relayed through the server to the phone, and then processed through the phone to the rover via the IOIO board.

Positioning Subsystem

To allow the rover to be able to get its approximate location, at least 4 bluetooth beacons will have to be placed in the vicinity and the coordinates in 3D space of each beacon will have to be provided to the rover beforehand. To perform positioning, the rover goes through these steps:

1. Determine the RSSI of each beacon
2. Convert the RSSI into distance, using previously measured values during a calibration step.
3. Perform trilateration

The measurements will all be noisy, so additional filtering steps have to happen.

Step 2 requires some clarification. 2 constants need to be accounted for with this system:

- **Path loss:** How much the signal degrades with distance. If this is low, you are in the middle of space with no obstacles. If this is high, then your house is probably made of tinfoil and signals are bouncing around everywhere and destructively interfering (so tinfoil houses may not be supported by our system). In theory, this constant stays the same for a given area.
- **Initial power:** The signal strength at a known distance. Beacons may transmit at different powers, and certain phones may have better antennas than others. This constant accounts for all this. It is different for each brand of beacon and different for each phone.

These constants need to be measured beforehand.

This entire positioning subsystem will be invoked when the user goes through the sequence detailed in Figure 4.

User wants Rover Location Page:

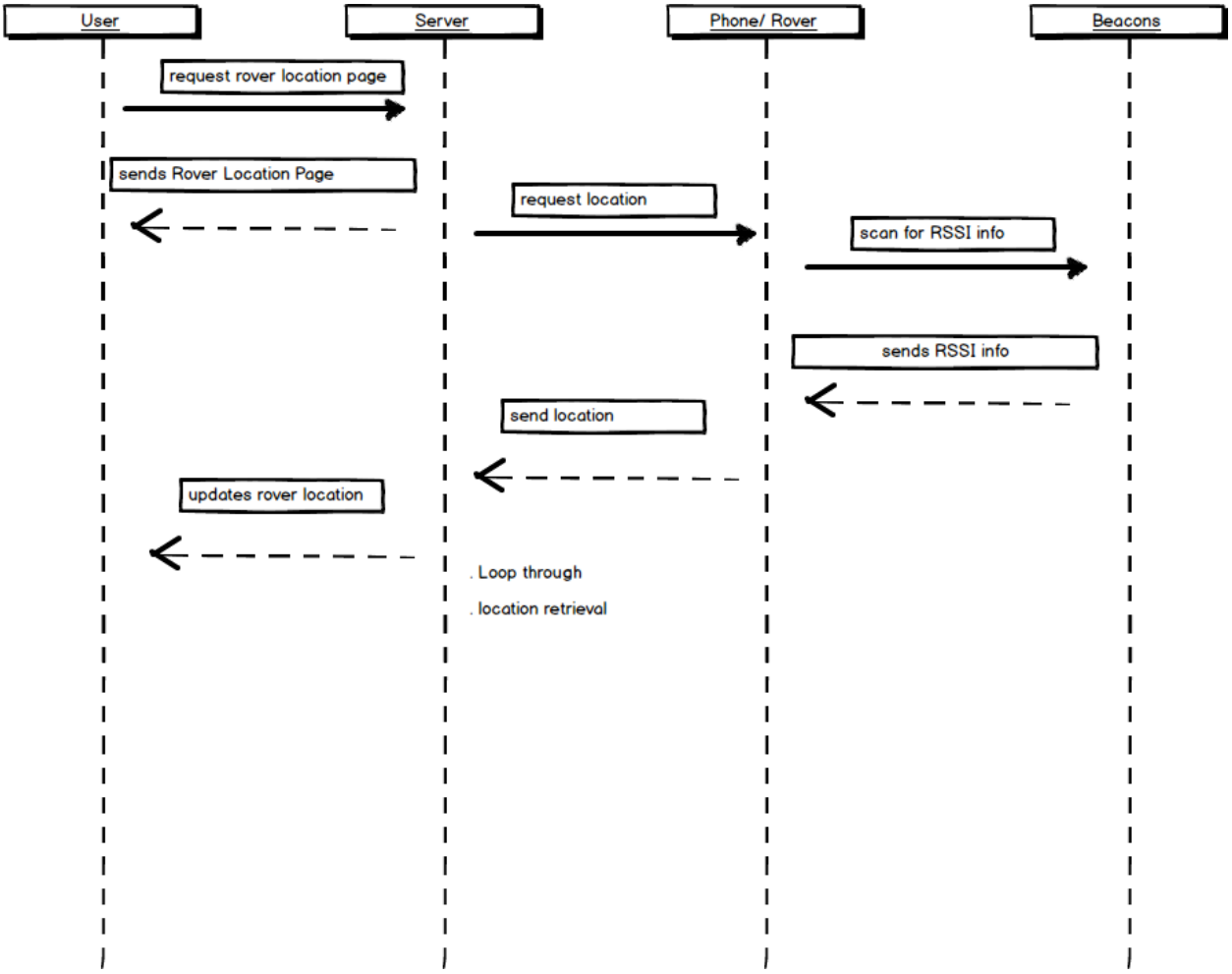


Figure 4: The figure above details the sequence the system will go through when the user requests the rover location page. The location retrieval process is repeated while the user is on this screen.

Class Diagram

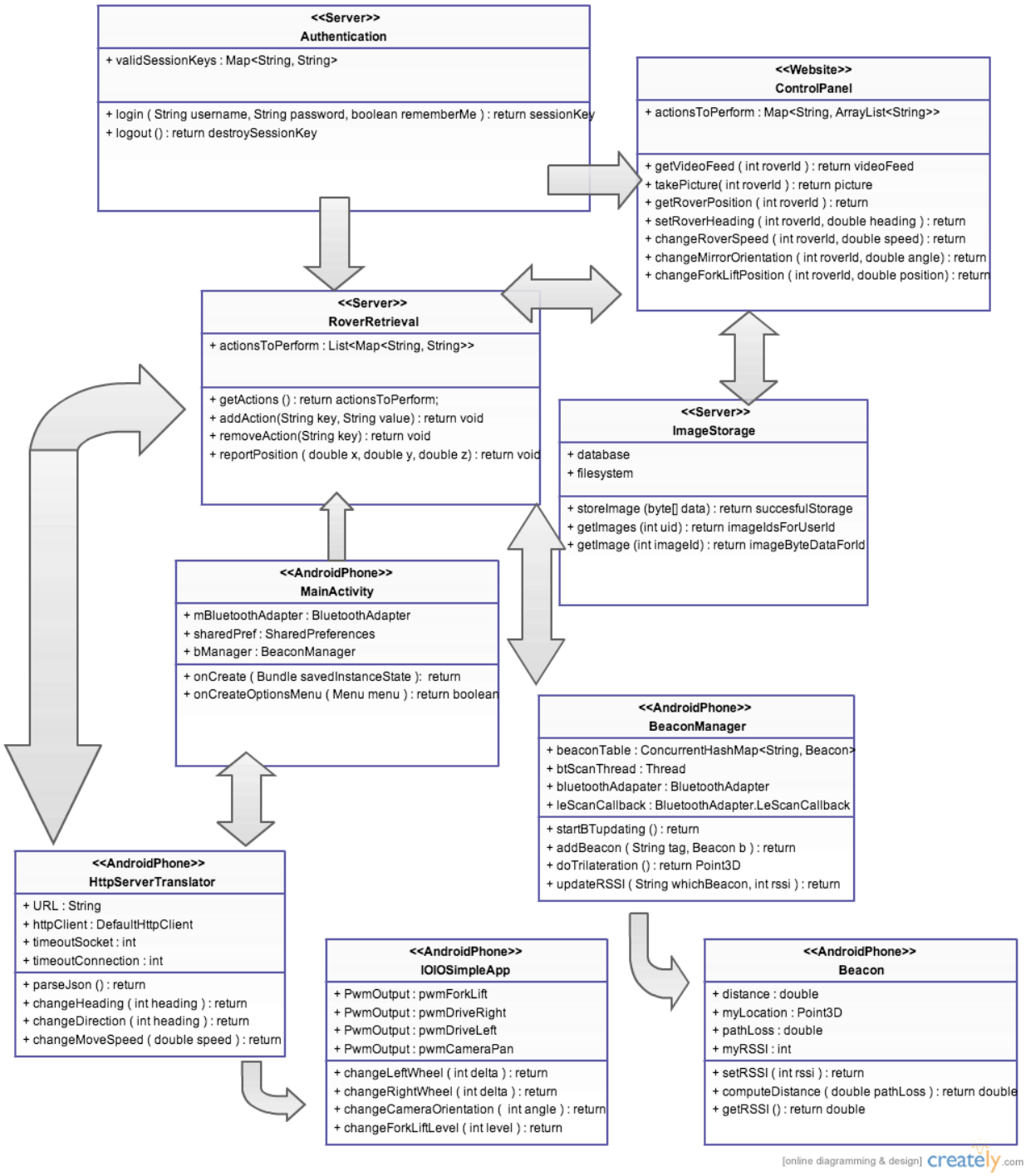


Figure 5: The figure above shows a UML diagram showing the various components of our system.

UI Mockups

Rover Remote-Control Interface (computer web version)
This is what users see when they want to manually control the rover. It contains a video feed, sliders to control parts of the rover, and a D-pad style interface to move the rover (forward, backward, turn left, turn right).

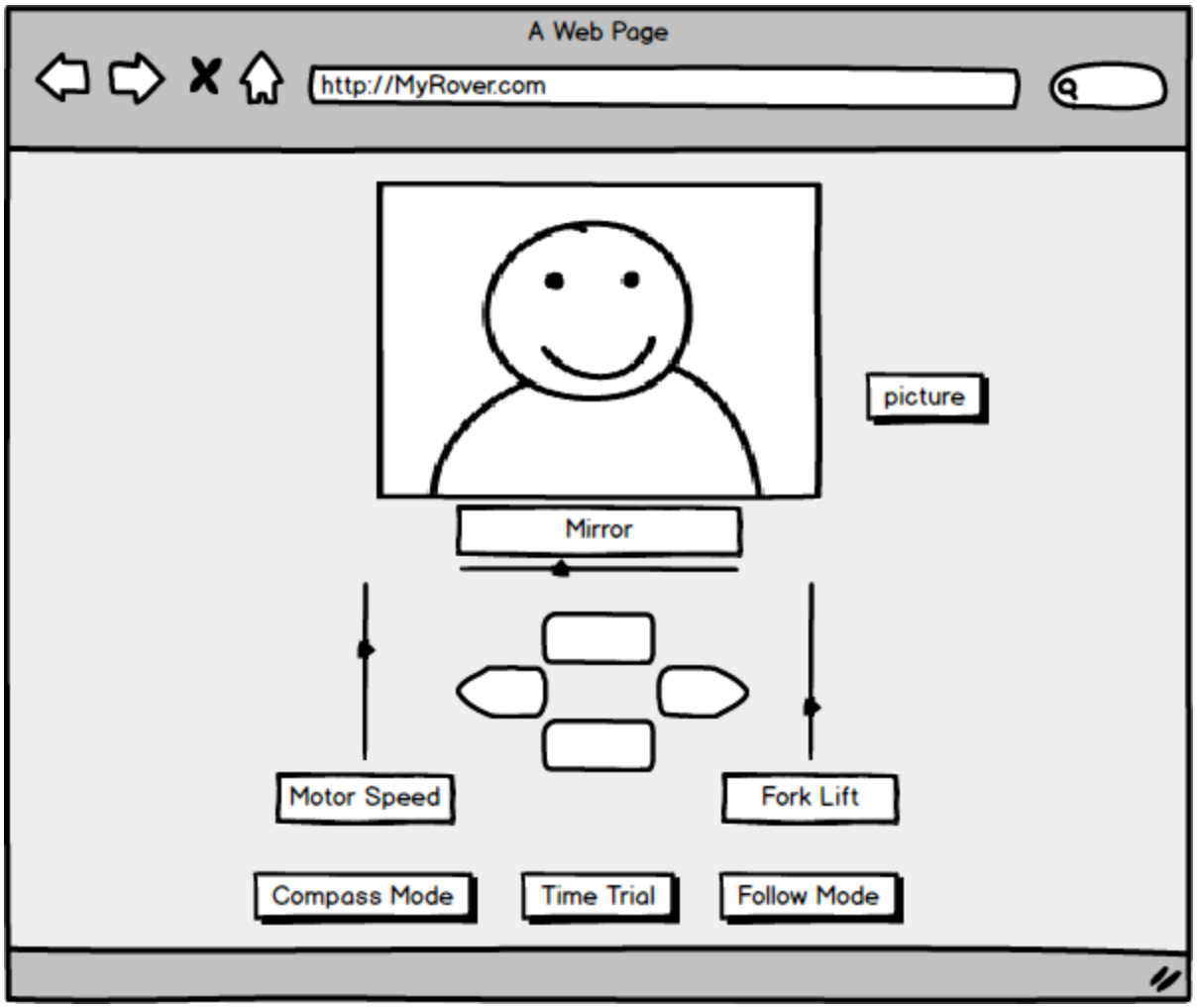


Figure 6: The web UI for the remote controller. The video stream will be shown at the top of the page. Arrow keys move the robot while separate sliders control mirror position, motor speed, and forklift level.

Rover Remote-Control Interface (smartphone web version)
This is what users see when they want to manually control the rover on a smartphone.

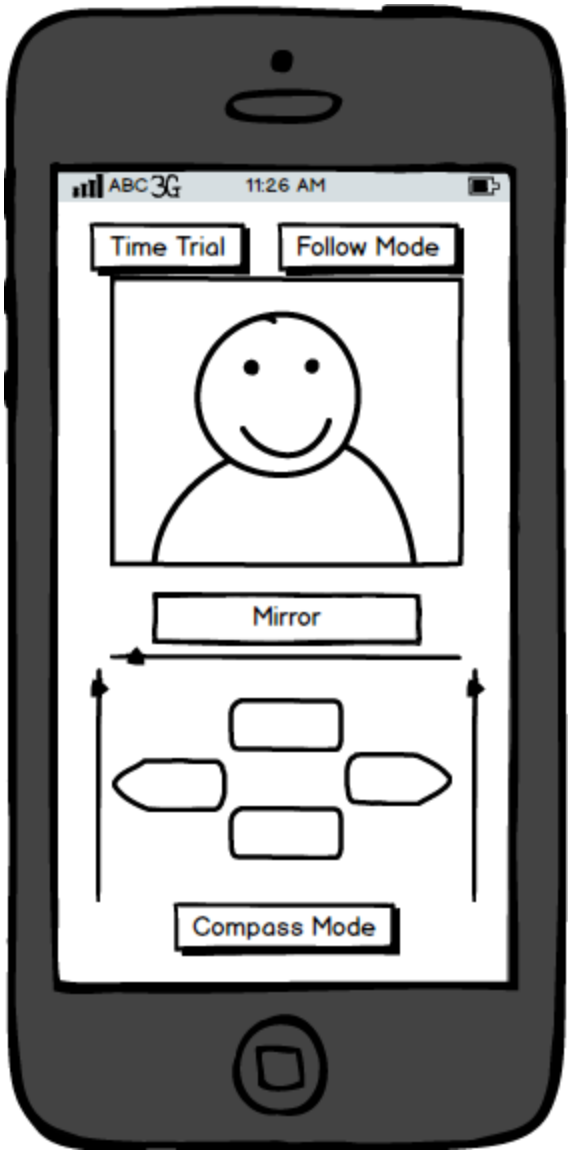


Figure 7: the mobile UI. all function are the same as in figure 5

Rover Location Interface

This interface provides a map showing where the rover is in relation to the beacons.

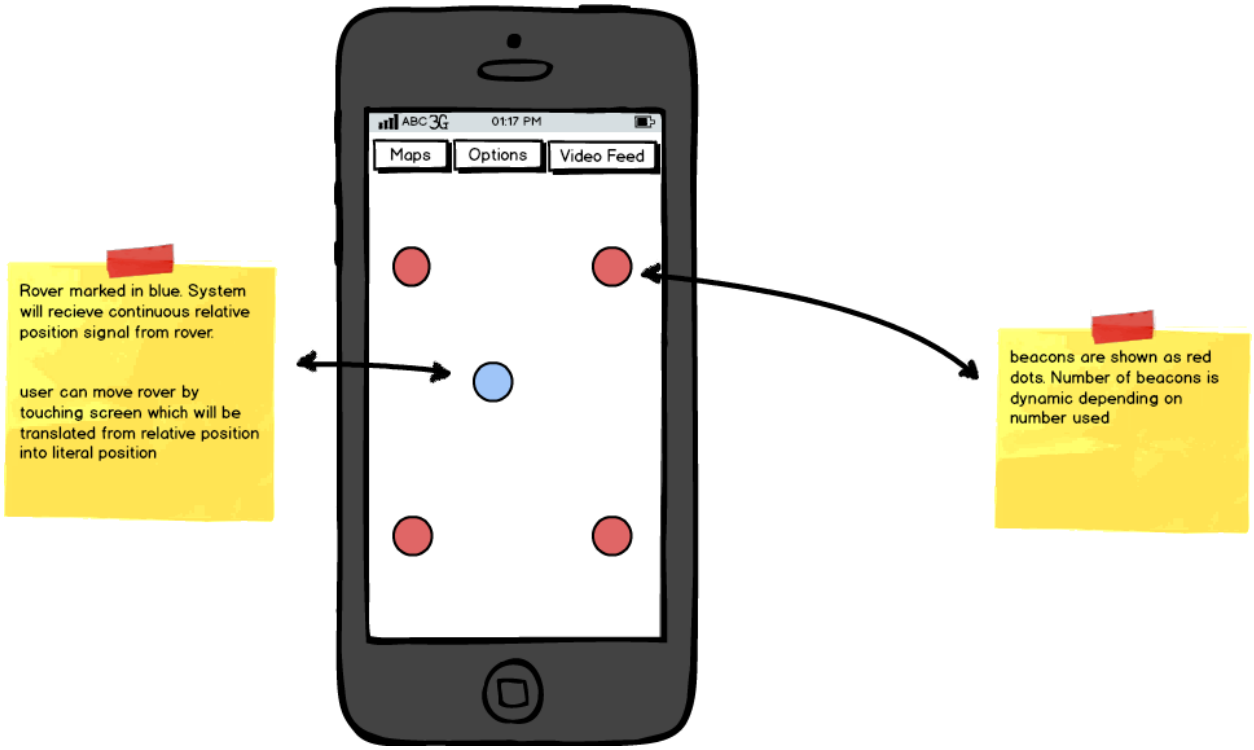


Figure 8: The mockup of what the rover location screen will look like. allowing the user to identify the position of the rover in between the reference beacons.

User Login Screen

This interface allows users to log in and access their MDR over the internet.

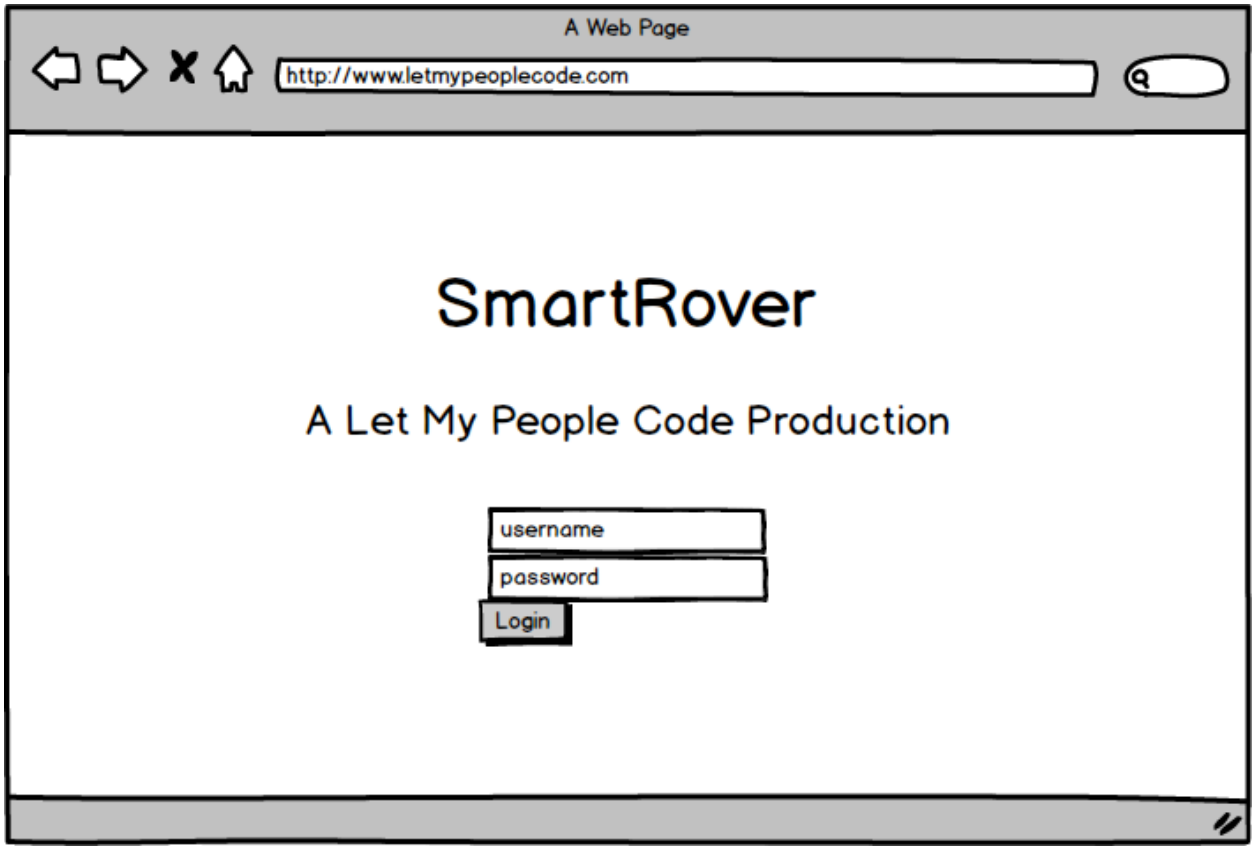
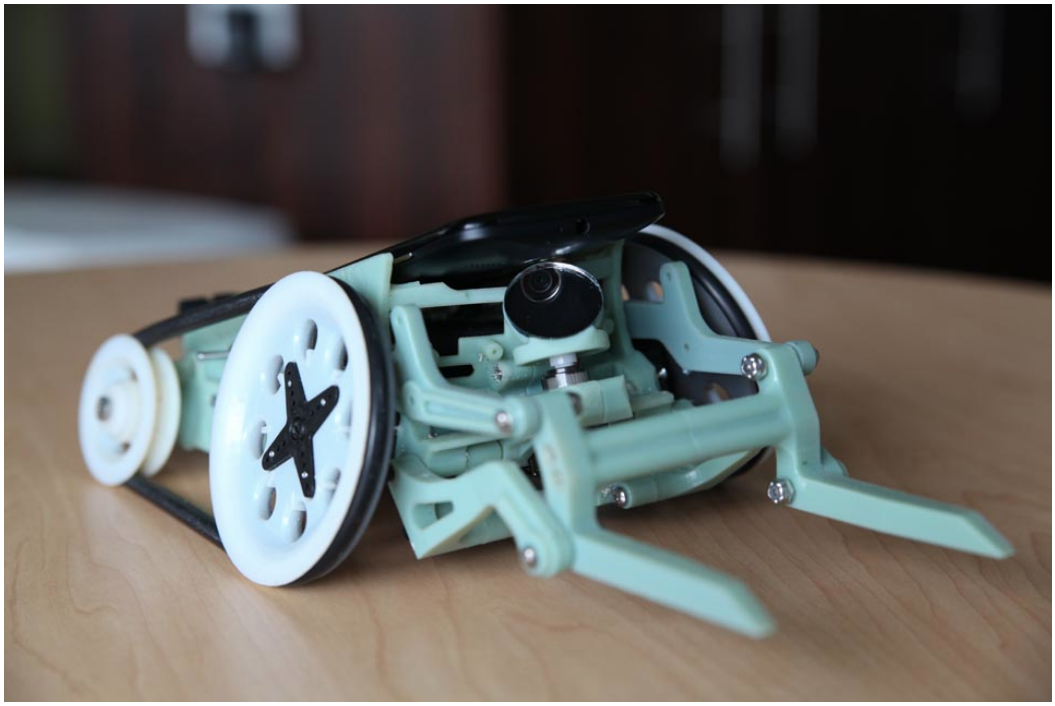


Figure 9: The login screen for the MDR's online remote control (desktop version)

Pictures of Our Hardware Setup

The Rover

Front view: Rover, forklift and mirror



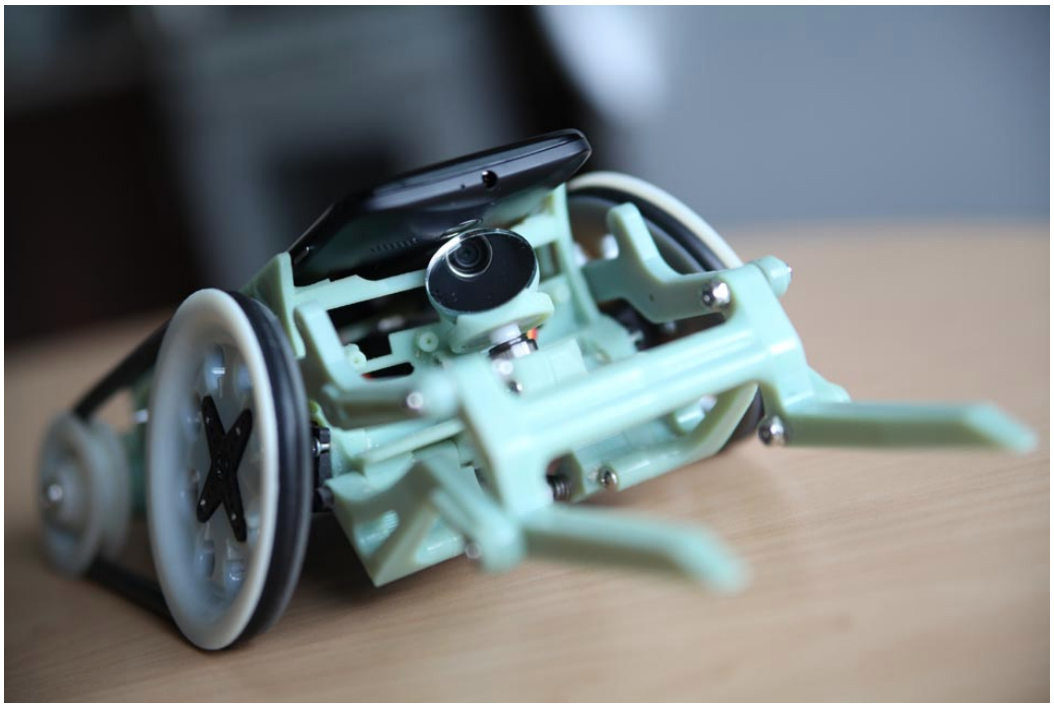


Figure 10: You can see the Moto G's camera reflected in the mirror.

Top view: a Moto G mounted on the rover

Figure 11: The rover carries the phone, connected through a USB cable.

Bluetooth Beacon Arrangement on the ceiling

Figure 12: Our bluetooth beacon setup so far in an apartment. There are 7 beacons total: 6 on the ceiling and one on the wall.