Remote Home

System Design Document | Current Version [1.0.0]

Prepared By:

Joshua Kinkade

James Wiegand

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Comments** |
| 12/7/12 | James Wiegand | 0.0.0 | Rough Draft |
| 12/15/12 | Joshua Kinkade | 1.0.0 | Rewrote using template and added detail |
| 12/16/12 | Christopher Jensen | 1.0.1 | Made some style corrections and added details to base station-related elements. |

Table of Contents

[1.0Overview 6](#__RefHeading__2291_610263877)

[1.1Scope 6](#__RefHeading__2293_610263877)

[1.2Purpose 6](#__RefHeading__2295_610263877)

[1.2.1iPhone Application 6](#__RefHeading__2297_610263877)

[1.2.2Base Station 6](#__RefHeading__2299_610263877)

[1.2.3Resolution Server 6](#__RefHeading__2301_610263877)

[1.2.4Devices 6](#__RefHeading__2303_610263877)

[1.3Systems Goals 6](#__RefHeading__2305_610263877)

[1.4System Overview and Diagram 6](#__RefHeading__2307_610263877)

[1.4.1iPhone Application 7](#__RefHeading__2309_610263877)

[1.4.2Base Station 7](#__RefHeading__2311_610263877)

[1.4.3Resolution Server 7](#__RefHeading__2313_610263877)

[1.4.4Devices 7](#__RefHeading__2315_610263877)

[1.5Technologies Overview 7](#__RefHeading__2317_610263877)

[2.0Project Overview 7](#__RefHeading__2319_610263877)

[2.1Team Members and Roles 7](#__RefHeading__2321_610263877)

[2.2Project Management Approach 7](#__RefHeading__2323_610263877)

[2.3Phase Overview 7](#__RefHeading__2325_610263877)

[2.4Terminology and Acronyms 8](#__RefHeading__2327_610263877)

[2.5Requirements 8](#__RefHeading__2329_610263877)

[2.5.1Smart Phone App 8](#__RefHeading__2331_610263877)

[2.5.2Hardware Control 8](#__RefHeading__2333_610263877)

[2.5.3Base Station 8](#__RefHeading__2335_610263877)

[2.5.4Communication 8](#__RefHeading__2337_610263877)

[2.5.5Resolution Server 8](#__RefHeading__2339_610263877)

[3.0Design and Implementation 9](#__RefHeading__2341_610263877)

[3.1iPhone Application 9](#__RefHeading__2343_610263877)

[3.1.1Technologies Used 9](#__RefHeading__2345_610263877)

[3.1.2Component Overview 9](#__RefHeading__2347_610263877)

[3.1.3Phase Overview 9](#__RefHeading__2349_610263877)

[3.1.4Architecture Diagram 9](#__RefHeading__2351_610263877)

[3.1.5Data Flow Diagram 9](#__RefHeading__2353_610263877)

[3.1.6Design Details 9](#__RefHeading__2355_610263877)

[3.1.6.I.1.1.1.1First time registration (iOS) 9](#__RefHeading__1652_408602583)

[3.2Base Station 10](#__RefHeading__2357_610263877)

[3.2.1Technologies Used 10](#__RefHeading__2359_610263877)

[3.2.2Component Overview 10](#__RefHeading__2361_610263877)

[3.2.3Phase Overview 10](#__RefHeading__2363_610263877)

[3.2.4Architecture Diagram 11](#__RefHeading__2365_610263877)

[3.2.5Data | Logic Flow Diagram 11](#__RefHeading__2367_610263877)

[3.2.6Design Details 11](#__RefHeading__2369_610263877)

[3.3Resolution Server 11](#__RefHeading__2371_610263877)

[3.3.1Technologies Used 11](#__RefHeading__2373_610263877)

[3.3.2Component Overview 11](#__RefHeading__2375_610263877)

[3.3.3Phase Overview 11](#__RefHeading__2377_610263877)

[3.3.4Architecture Diagram 11](#__RefHeading__2379_610263877)

[3.3.5Design Details 11](#__RefHeading__2381_610263877)

[3.3.5.I.1.1.1.1SQLite Database 11](#__RefHeading__1656_408602583)

[3.3.5.I.1.1.1.2Finder Class 11](#__RefHeading__1658_408602583)

[3.3.5.I.1.1.1.3Server Class 12](#__RefHeading__1660_408602583)

[3.3.5.I.1.1.1.4Main.py 12](#__RefHeading__1662_408602583)

[3.4Devices 12](#__RefHeading__2383_610263877)

[3.4.1Technologies Used 12](#__RefHeading__1870_21348573)

[3.4.2Component Overview 12](#__RefHeading__1872_21348573)

[3.4.3Phase Overview 12](#__RefHeading__1874_21348573)

[3.4.4Architecture Diagram 12](#__RefHeading__1876_21348573)

[3.4.5Design Details 12](#__RefHeading__1878_21348573)

[4.0System and Unit Testing 12](#__RefHeading__2385_610263877)

[5.0Overview 12](#__RefHeading__1880_21348573)

[5.1.1iPhone Application 12](#__RefHeading__2387_610263877)

[5.1.2Base Station 13](#__RefHeading__2389_610263877)

[5.1.3Resolution Server 13](#__RefHeading__2391_610263877)

[5.1.4Devices 13](#__RefHeading__2393_610263877)

[5.2Dependencies 13](#__RefHeading__2397_610263877)

[5.2.1iPhone Application 13](#__RefHeading__2399_610263877)

[5.2.2Base Station 13](#__RefHeading__2401_610263877)

[5.2.3Resolution Server 13](#__RefHeading__2403_610263877)

[5.2.4Devices 13](#__RefHeading__2405_610263877)

[5.3Test Setup and Execution 13](#__RefHeading__2407_610263877)

[5.3.1iPhone Application 13](#__RefHeading__2409_610263877)

[5.3.2Base Station 13](#__RefHeading__2411_610263877)

[5.3.3Resolution Server 13](#__RefHeading__2413_610263877)

[5.3.4Devices 14](#__RefHeading__2415_610263877)

[6.0Development Environment 14](#__RefHeading__2417_610263877)

[6.1.1iPhone Application 14](#__RefHeading__2419_610263877)

[6.1.2Base Station 14](#__RefHeading__2421_610263877)

[6.1.3Resolution Server 14](#__RefHeading__2423_610263877)

[6.1.4Devices 14](#__RefHeading__2425_610263877)

[6.2Development IDE and Tools 14](#__RefHeading__2427_610263877)

[6.2.1iPhone Application 14](#__RefHeading__2429_610263877)

[6.2.2Base Station 14](#__RefHeading__2431_610263877)

[6.2.3Resolution Server 14](#__RefHeading__2433_610263877)

[6.2.4Devices 14](#__RefHeading__2435_610263877)

[6.3Source Control 14](#__RefHeading__2437_610263877)

[6.4Dependencies 15](#__RefHeading__2439_610263877)

[6.5Build Environment 15](#__RefHeading__2441_610263877)

[6.5.1iPhone Application 15](#__RefHeading__2443_610263877)

[6.5.2Base Station 15](#__RefHeading__2445_610263877)

[6.5.3Resolution Server 15](#__RefHeading__2447_610263877)

[6.5.4Devices 15](#__RefHeading__2449_610263877)

[6.6Development Machine Setup 15](#__RefHeading__2451_610263877)

[6.6.1iPhone Application 15](#__RefHeading__2453_610263877)

[6.6.2Base Station 15](#__RefHeading__2455_610263877)

[6.6.3Resolution Server 15](#__RefHeading__2457_610263877)

[6.6.4Devices 15](#__RefHeading__2459_610263877)

[7.0Release | Setup | Deployment 15](#__RefHeading__2461_610263877)

[7.1.1iPhone Application 15](#__RefHeading__2463_610263877)

[7.1.2Base Station 16](#__RefHeading__2465_610263877)

[7.1.3Resolution Server 16](#__RefHeading__2467_610263877)

[7.1.4Devices 16](#__RefHeading__2469_610263877)

[7.2Deployment Information and Dependencies 16](#__RefHeading__2471_610263877)

[7.2.1iPhone Application 16](#__RefHeading__2473_610263877)

[7.2.2Base Station 16](#__RefHeading__2475_610263877)

[7.2.3Resolution Server 16](#__RefHeading__2477_610263877)

[7.2.4Devices 16](#__RefHeading__2479_610263877)

[7.3Setup Information 16](#__RefHeading__2481_610263877)

[7.4System Versioning Information 16](#__RefHeading__2483_610263877)

[8.0End User Documentation 16](#__RefHeading__2485_610263877)

[8.1Getting the iPhone App 16](#__RefHeading__2487_610263877)

[8.2Setting up your Base Station 17](#__RefHeading__2489_610263877)

[8.3Adding a Base Station to Your Phone 17](#__RefHeading__2491_610263877)

# Overview

The purpose of this document is to give the reader an understanding of iPhone Home Remote. iPhone Home Remote is a system that will allow users to control devices in their house from anywhere in the world, using their iPhone. Although the system contains implementations for garage doors, the iPhone Remote Home system is easily extendable to allow any electronic device to be controlled with only minor changes to the system. The Remote Home system has four major components:

1. An iPhone application, which acts as a remote control,
2. A Base Station in the house, which connects the devices to the internet,
3. A Resolution Server, which allows the iPhone application to connect directly to a Base Station, and
4. The actual devices to be controlled.

## Scope

This document describes what the purpose and requirements of our system are and how it works, including details about the major components and what technologies are used to implement the system. It also discusses our team and development process. It does NOT act as a user guide, nor does it describe in full a standard or metric for evaluation of technologies.

## Purpose

The purpose of this project is to allow people to control electronic devices in their house over the internet using their iOS-compatible devices, and to create a formal protocol for passing data to such devices over TCP/IP.

### iPhone Application

The users will mostly interact with the system using an application on their smart phone. This application allows them to register a Base Station with their phone, view all of their Base Stations, view and control all of the devices associated with each Base Station.

### Base Station

The Base Station can connect all of its attached devicesin a house to the internet. It will register its IP address and unique identifier with the Resolution Server so the iPhone application will be able to connect directly to the Base Station. The Base Station will send commands from the iPhone to the appropriate device, and act as a basic authentication server to prevent unauthorized access. Configuration of the server can be done through a remotely-accessible web application running on an Apache server on the base station. It is also possible to manually configure the base station by editing the devices.ini and users.ini files with a text editor while locally logged into the base station.

### Resolution Server

The Resolution Server will store the unique identifiers and IP addresses of all Base Stations that have been setup, will handle requests from the iPhone application to get the IP address of a particular Base Station, and will receive updated IP addresses from all of the Base Stations.

### Devices

The devices to be controlled are electronic devices that can either receive commands from the Base Station and reply, or devices which can be controlled by such a device externally. A device could be anything from a lamp, which can be operated by opening or closing a switch, to a complex programmable thermostat that would allow users to adjust the temperature or create heating schedules from their iPhone. The present implementation includes a garage door opener as a sample device and example of how to implement the protocol that the system provides.

## System Goals

The goal of the iPhone Home Remote system is to provide a complete, end-to-end solution to allow people to control devices in their house from anywhere they can connect to the internet.

## System Overview and Diagram

The four major parts of the Remote Home system are the iPhone application, the Base Station, the Resolution Server and the actual devices that are controlled.

Figure System Diagram

### iPhone Application

The iPhone application allows users to view or add Base Stations and view and control devices associated with those Base Stations.. It will use serial numbers on each of the Base Stations associated to get an IP address from the Resolution Server and will use this data to connect directly with a Base Station using TCP. Communications are currently sent “In the clear”, although with some modification it would be possible to use TLS or other encryption technology. The application stores the serial number, password, and user-defined friendly name of each base station on the phone.

### Base Station

The Base Station will handle connections from the iPhone and send commands to the appropriate device using the device’s serial number. It will also send feedback from the devices back to the iPhone, and periodically update the Resolution Server with its global IP address. At present, the Base Station is only designed to handle communications over TCP/IP or Serial to communicate with devices, but it would not be hard to expand to other communication protocols in Python, such as Bluetooth/HCI connections.

### Resolution Server

The Resolution Server stores the serial numbers and associated IP addresses of various base stations and handles requests from the iPhone for the IP address associated with a given serial number. It also handles requests from the Base Station to update its IP address.

### Devices

The actual devices that will be controlled by the iPhone are regular electronic devices currently available on the market, or devices specially modified to be controlled by a microcontroller. The devices will be capable of receiving commands from the Base Station and sending information that needs to be sent from the iPhone to the Base Station.

## Technologies Overview

The system uses a variety of technologies. Each component’s technology is discussed in the component’s sub section in section 4. All communication except for communications from base station to device shall be based on TCP/IP and use JSON to structure information sent between components. Communications between devices and base station will be a serial format which will include methods to query data based upon an index system.

# Project Overview

## Team Members and Roles

James Wiegand is our team leader and is developing the core of the iPhone application. Joshua Kinkade is developing the individual Device controllers for the iPhone application and the Resolution Server. Christopher Jensen is working on the hardware and the Base Station software. Brian Vogel will join our team in January and will work with Christopher on the hardware and Base Station.

## Project Management Approach

Our project is using the Agile development process. Our sprints are mostly three weeks long. We are using Trello to keep track of our backlog.

## Phase Overview

In the first phase of our project we will focus on getting the system operational. To limit the amount of work in this, we will be focus on getting one device, the garage door, working. This will allow us to build each component of our system and ensure that they work together. After that we will start working on adding additional devices such as a sprinkler controller. If time permits, we may develop an Android remote control application as well.

## Terminology and Acronyms

Base Station – A lightweight server that will be in the users house. This device controls and manages devices and communicates directly with the iPhone app.

Cocoa Touch – Apple’s framework for iOS applications.

Device – A physical object that the user wishes to control with their iOS application. Examples would be garage doors, and sprinkler systems. The only device created by this team is a garage door opener.

iOS – The operating system that is present on iPhones and iPads. In this context iOS refers to the iOS 6.0

Resolution Server – A server operated by our client. The server has a database that will store associations between a serial numbers and IP address.

Serial Number – The MAC address of a base station's primary network card, as reported by python's uuid library.

TCP – Transmission Control Protocol – protocol that manages the transfer of data from one computer to another.

UIAlertView – The standard iOS dialog box for alerting the user with important information.

# Requirements

### Smart Phone App

The user should be able to control from anywhere in the world with internet access. The client gave specific requirements for garage doors and sprinkler systems. Our system will be able to handle both of these devices as well as any other that is designed for our system. For the garage door, the application should be able to tell if the door is open or closed or in between and control it with a button. Our garage door control handles the status display by showing a simple garage door graphic. For the sprinkler system, the application should allow the user to turn sprinklers on and off and give them schedules for when to run.

### Hardware Control

The client suggested that we use a Raspberry Pi, BeagleBone, or an Arduino to control the devices. We decided to use Arduinos because Raspberry Pi is more complicated than what is needed for our purposes and they are more readily available than Raspberry Pis are right now. Each of our devices will be controlled by an Arduino and will connect to a central Base Station.

### Base Station

Because we generalized what devices our system would control, we added a base station that will sit in the user’s house and coordinate communication between the application and all of the devices in the house.

### Communication

The client suggested that we use a custom TCP/IP protocol for communication between our devices. We are creating a protocol that uses a direct two way connection between the smartphone and the Base Station and also between the smartphone or base station and the Resolution Server. It uses JSON to send requests. The protocol is detailed in Appendix II.1.

### Resolution Server

Because the Base Stations will be located at the users’ houses, they will most likely have dynamically assigned IP addresses, we are creating a server that will associate a Base Station’s unique serial number with its current IP address. The Base Stations will periodically send updated IP addresses to the Resolution server. When the phone application wants to connect to a Base Station, it will first send the serial number of the Base Station to the Resolution server to get the Station’s IP address. This will allow the smart phone to establish a direct connection to the Base Station.

# Design and Implementation

This section is used to describe the design details for each of the major components in the system. This section is not brief and requires the necessary detail that can be used by the reader to truly understand the architecture and implementation details without having to dig into the code.

## iPhone Application

### Technologies Used

The iPhone application will be a native iOS application developed using the Cocoa Touch application framework.

### Component Overview

The iOS application will act as the remote control for the Remote Home system. The application will consist of views. The application will also use a SQLite database that will store the base stations that have been registered. When the application is first started the “first time registration” controllers will run, this will require the user to register a valid base station. If the user has at least one base station registered the application will start the “main view” controllers. This is a UINavigation view controller that will present a list of base stations. In addition a add button will be in the upper right hand corner of the list view. If the user presses this button they can add a new base station. The user can swipe across a cell of a base station to delete the base station from the SQLite database. If the user selects a base station a new list will populate with the individual devices the user will have an edit button in the upper right hand corner of the list view. If the user presses this button they will be presented with a form where they can modify the properties of the base station. If a user selects a device they will be presented with the correct device controller.

### Phase Overview

1. User Interface for viewing Base Stations and their associated devices
2. User Interface for registering a Base Station with the iPhone
3. Getting IP addresses for Base Stations from the Resolution Server
4. Device Specific view controllers

### Architecture Diagram

### Data Flow Diagram

Figure : The Architecture of the iPhone App

TODO: Get the flowchart from James' documentation

### Design Details

TODO: James and Josh: Not sure what you guys want here. Also need to fix the numbering below; OpenOffice is having a fit with it.

#### First time registration (iOS)

The purpose of the first time registration is to force the user to register a base station with the phone so that they can control devices. We will display this view if there are no base stations registered in the SQLite database.

**Instruction View Controller**

The purpose of this view controller is to display a scroll view with instruction embedded in it. These scroll view will instruct the user to set up their base station and connect devices. At the bottom of the view controller will be a button so that the user can advance to the registration view.

**Registration View Controller**

The purpose of this view controller is to allow the user to register a new base station. The view will consist of three text boxes and a register button. The view controller will check to see if all three fields are filled before connecting to the resolution server. If any of the fields are empty a UIAlertView will be displayed telling the user to fill out the empty field.

If all three fields are populated and the user clicks the register button the device will attempt to make a TCP connection to the resolution server on port 80. At this point the device will start a timeout timer. If the TCP connection fails to open before the timeout fires the system will close the connection and present a UIAlertView to the user. The alert will instruct the user to check their connection and/or try again later.

If the connection is successful the server will send the connection DDNSConnected (See “Bidirectional iOS to Resolution Server Communication”) signal to the phone. At this point the phone will send the HRHomeStationsRequest with the serial number provided by the serial number field. At this point the Resolution Server will look up the serial number. If the Resolution Server finds the serial number it will respond with HRHomeStationReply with the correct IP address and serial number. If the Resolution Server fails to find the serial number it will respond with HRHomeStationReply with ‘null’ for the IP address and the correct serial number.

If the phone receives a null for the IP address it will present the user with an UIAlertView. This view will inform the user to check the serial number and make sure that they set up the base station correctly. If an IP address is sent the device will register the device in the SQLite server and present a UIAlertView to the user. This view will inform the user that the device was successfully registered. At this point the device will go into the main view.

Garage Door View Controller

This view controller will allow a user to control a single garage door. The user interface consists of a simple animated garage door graphic at the top of the screen and a large button at the bottom. The user can open and close the door by pressing the button. The label will change to fit the current action. It will say ‘open’ when the door is closed and ‘close’ when the door is open. In addition, the user can swipe the garage door picture up to open the door and down to close it. If there is an object preventing the door from closing, the garage door graphic will stop halfway down and the application will alert the user using a UIAlertView. Once the object has been removed the user can finish closing the door.

## Base Station

### Technologies Used

The base station leverages the same Python interface used by the Resolution Server, as well as an interface for sending serial data. The exact carrier is not specified, and will be left open to individual implementations of the system. The sample is expected to leverage PySerial to communicate via USB with the individual device.

### Component Overview

The Base Station will run on an Intel x86 or amd64 architecture CPU, with at least 1 MB RAM and 50KB of disk space. The base station can be deployed to any system which supports Python and PySerial, so theoretically other CPU architectures could be targeted if Python and PySerial are available.

### Phase Overview

* 1. Create an executable to handle basic network communications with the iPhone app: Authentication, General Receive, General Send
  2. Create or expand an executable to communicate with devices: General Receive, General Send, ACK, NACK
  3. Have the ability to populate a device pool and credentials from the base station
  4. Get the web interface which integrates database management and basic registration tasks constructed.

### Architecture Diagram

### Data | Logic Flow Diagram

### Design Details

Presently, the base station works using the Python built-in SocketServer in threaded mode to handle requests. It also starts a daemon thread that occasionally sends a data parcel to the resolution server to keep the system updated. The serial number is generated by the python uuid library, which queries the MAC address of whichever network card is presently active. This allows for a reasonably unique identifier which SHOULD not change.

The different devices are stored in an associated ini file. ini format was selected for its simplicity and small size; XML or similar verbose systems may not fit easily on a small footprint device and require additional effort to parse. Each entry in the file contains a key of the device name, an associated group, Device ID, and the interface on which the device should be contacted. It is theoretically possible to communicate with other devices on port 8128 by specifying an IPv4 address (xx.xx.xx.xx) as the interface, otherwise a serial device is assumed. The server will attempt to open said interface using the PySerial library.

Users have a separate ini file which contains the user's password (unhashed) and group. The special group “All” (case-insensitive) is considered an alias for all devices attached to the station, and can be considered an administrative group. Beyond that, any user's group is matched against the group of each device before a response is sent from the base station, so any client apps will only be aware of devices the user is permitted to access.

## Resolution Server

### Technologies Used

The Resolution Server is written in Python 2.7. It uses the SQLite database to store data and TCP to communicate.

### Component Overview

The Resolution Server stores the IP address of each Base Station, along with a unique identifier for that station, and allows the iOS application to ask for the IP address of a Base Station. The server will run as a background daemon on a computer with a domain name or a static IP address. It will use port number 8128. The IP address and identifier of the Base Stations are stored in a SQLite database. The server itself will be written in Python 2.7. It will be written in a few classes. The Finder classes will handle the SQLite database. The Server class will handle connections.

### Phase Overview

1. Make the server correctly respond to requests for IP addresses from the iOS application.
2. Make the server correctly respond to requests to update an IP address from a Base Station.
3. Make the server run as a daemon, so the computer running the program doesn’t need a terminal open all of the time

### Architecture Diagram

### Design Details

#### SQLite Database

This database has a single table, called devices, with two columns: ID and IP. Both columns are text and the column ID is the primary key.

#### Finder Class

The Finder class abstracts database interaction with the rest of the program. It will have methods that roughly correspond to the possible requests made to the server. It may also have some utility methods.

#### Server Class

The Server Class handles connections with the client. It waits for a connection and when it gets one immediately responds using the DDNSConnected protocol described in Appendix II.1.1.1. This is required for the iPhone application to connect properly. Then it will wait for the actual request from the client. Once it has the request, it will call the appropriate method of the Finder class to retrieve or store information.

#### Main.py

The main Python file contains the function main. The main function is just the entry point for the program. It instantiates a Server Object and calls its main loop. Once the loop returns, the program exits. This function will also be responsible for converting the process into a daemon.

## Devices

TODO: Brian, this is your section. I've tried to put some stuff in here for what I did on the design phase, but large portions of this are you.

# Technologies Used

The Garage Door Opener uses a Seeed Studio Relay Shield, a Seeed Studio Prototype Shield for the limit switches, and an Arduino Uno microcontroller board.

# Component Overview

The relays close and open the garage door's “wall button” circuit to operate a standard garage door opener. The device listens on its USB connection for a signal from the base station, then reports its status (or an error, if applicable). If it receives a signal ASCII 1, it returns an integer value indicating the state of the door. TODO: Brian, what are the values possible? If it receives a signal ASCII 0, it actuates relay 4 open and closed, causing the garage door to operate.

# Phase Overview

Acquire hardware for opening garage door

Set up hardware to operate as specified.

# Architecture Diagram

TODO: Brian, this is all you.

# Design Details

# System and Unit Testing

# Overview

Provides a brief overview of the testing approach, testing frameworks, and general how testing is/will be done to provide a measure of success for the system.

### iPhone Application

Testing on the iPhone application will be done using the unit testing framework that is built into Cocoa Touch.

### Base Station

The Base Station is tested by sending it requests and looking at the onboard logs to determine how it handled the request. To facilitate message passing, a dummy client was written which can handle batch requests and record returned data.

### Resolution Server

The resolution server will be tested using a script to simulate the iPhone and Base Station. It will make all of the requests using both valid and invalid data.

### Devices

The device controllers can be tested using the standard Arduino testing tools and serial communication programs, like PuTTY.

## Dependencies

### iPhone Application

The iPhone application depends on Apple’s Cocoa Touch framework.

### Base Station

The base station depends on the same version of Python as the Resolution Server, TCP for internet communication and PySerial for USB communication. The web configuration frontend requires PHP 5.4 or later and an Apache server with PHP enabled.

### Resolution Server

The Resolution Server depends on Python 2.7 and SQLite.

### Devices

The devices will each require an Arduino control board or that the device itself implement the necessary logic and communication to a base station.

## Test Setup and Execution

### iPhone Application

### Base Station

A small python script was created to send command strings to the base station. This script reads commands, one line at a time, from a plaintext file and prints the server's response. With redirection, it is possible to generate files for executing and validating batches of commands. This dummy client is available in the dummy directory of the base station, and is available for use as a sample for communicating in addition to its test role.

### Resolution Server

### Devices

# Development Environment

### iPhone Application

The iPhone application must be developed on Mac computer using Apple’s Xcode IDE. Xcode can be downloaded from Apple’s website for free after creating a free Apple developer account. The application can be developed and tested on the simulator with a free account, but to test the application on hardware and distribute it a paid subscription to the Apple iOS Developer Program is required.

### Base Station

The Base Station was developed in a POSIX-compliant environment using Python with PySerial, using a plain text editor.

### Resolution Server

The Resolution Server is being developed on a Mac using a plain text editor, but any computer with a Linux or UNIX operating system would work as well. It requires Python 2.7.

### Devices

The Arduino based device controllers were developed using the free Arduino development tools.

## Development IDE and Tools

### iPhone Application

The Xcode IDE was used.

### Base Station

Development was done using a plain text editor.

### Resolution Server

Development was done in a plain text editor.

### Devices

The Garage Door controller was written in C++ using the Android IDE.

## Source Control

Github was used for source control. It has a Documents directory to store documentation and a src directory for project code.

## Dependencies

The system depends on the iPhone API and Python with the PySerial module. The optional web frontend depends on PHP 5.4 or later and apache.

## Build Environment

### iPhone Application

The iPhone application must be built using Xcode on a Mac.

### Base Station

Python 2.7 with PySerial module is necessary for building. The web frontend is built on Apache and PHP 5.4 or later.

### Resolution Server

Python 2.7 or compatible interpreter is necessary.

### Devices

## Development Machine Setup

### iPhone Application

The iPhone application was developed on a Mac with Apple’s developer tools installed.

### Base Station

The Base Station was developed in various Linux distributions using vim, emacs and cat. The machine had Python 2.7, PySerial, Apache and PHP installed.

### Resolution Server

A Linux or UNIX computer with Python 2.7, SQLite and a text editor.

### Devices

The Garage Door controller was developed using the Arduino IDE.

# Release | Setup | Deployment

### iPhone Application

The application must be distributed through Apple’s AppStore to be installed on users’ phones.

### Base Station

To setup a Base Station, the software must be installed on a computer and the network must be configured to allow incoming connections to reach the Base Station.

### Resolution Server

The Resolution Server software should be installed on a computer with a domain name or a static IP address so the iPhone application and Base Stations know where to find it.

### Devices

## Deployment Information and Dependencies

### iPhone Application

### Base Station

### Resolution Server

### Devices

## Setup Information

## System Versioning Information

TODO: Brian, Josh, James, back me up on this? Does this sound good to everyone?

There is currently no formal versioning system, but since this release is largely complete, an arbitrary initial release value of 0.8 will be selected. Releases will be divided into major and minor releases, with even-numbered minor releases indicating a “stable” variant and odd-numbered indicating “unstable” or “beta” releases. Major releases on a maintained system will occur every two years or as specific milestones are met, while minor releases should contain bug fixes and similar modifications.

# End User Documentation

TODO: I'm going to strip this section; it belongs in the actual End User Documentation.

## Getting the iPhone App

To get the Remote Home App on your iPhone, download it from the AppStore on your phone or download it from iTunes on your computer and sync your phone.

## Setting up your Base Station

To setup your Base Station plug the power cord in and then plug the network cable into your router. You will have to configure your router’s firewall to allow your phone to connect to your Base Station when you are away from your network. If you do not know how to do this, consult your router’s user manual or you can contact us at (555) 123-4567 and one of our representatives will help you.

## Adding a Base Station to Your Phone

The first time you open the Remote Home App on your phone you will be asked to add a Base Station. Fill in the text boxes with the Station’s serial number, which is printed on your Station, a memorable name for your device( it doesn’t matter what it is), and the password for your Station.l

* + - 1. List of Figures

TODO: Brian, Josh, James, I can't see anything here. If you can, groovy. Otherwise, could someone with a formal Word edit this?

* + - 1. Supporting Information and Details
         1. Communication Protocols

Bidirectional iOS to Resolution Server Communication

This communication protocol defines the messages that will be passed between an iOS client and the Resolution Server. This will allow the iOS client to loop up IP address for a base station from a serial number.

DDNSConnected

This message is passed when the Resolution Server acknowledges a connection from a iOS client.

{ "DDNSConnected": [ { "Connected": true } ] }

HRHomeStationsRequest

This message is sent from the iOS client to the Resolution Server. This message is a request for IP addresses based on serial number. “(StationDID)" field will be replaced by a base station serial number.

{ "HRHomeStationsRequest" : [ { "StationDID" : "(StationDID)"}, { "StationDID" : "(StationDID)"}, ... ] }

HRHomeStationReply

This message is sent from the Resolution Server to the iOS client. This will tell the iOS client the association between serial numbers and IP addresses. “(StationDID)” is the station serial number. "xxx.xxx.xxx.xxx" is the IPv4 address of the base station. If the station cannot find the IPv4 address it will fill the "xxx.xxx.xxx.xxx" field with a null.

{ "HRHomeStationReply" : [ {"StationDID" : "(stationDID)", "StationIP" : "xxx.xxx.xxx.xxx"}, {"StationDID" : "(stationDID)", "StationIP" : null}, ... ] }

Unidirectional Base Station to Resolution Server Communication

This communication protocol defines the messages that will allow the Base Station to update Resolution Server with its current IP address.

HRHomeStationUpdate

This message is sent from a Base Station to the Resolution Server. It contains the station’s unique identifier and its current IP address.

{“HRHomeStationUpdate”:{“StationDID”:”(StationDID)”,”StationIP”:”(xxx.xxx.xxx.xxx)”}}

* + - 1. Progress | Sprint Reports

This section will contain a complete list of all of the period progress and/or sprint reports which are deliverables for the phases and versions of the system.

TODO: Brian, James, Josh, Ugh. This formatting is awful, and OpenOffice keeps failing me. Anybody okay with mucking the formatting into consistency?

* + - * 1. Sprint 1 Progress Report
  + Description: L-3 Communications is a company that develops, command and control, avionics, and communications technology for commercial and government customers.
  + Project Goal: The goal of this project is to allow people to control devices in their house using an application on their smart phone.
  + Needs:
    - An application to send commands to devices remotely
    - A way to physically control devices
    - A way to connect the two things listed above
    - Project Overview: We are creating a system that allows users to control devices in their house, such as a garage door opener, from an application on their iPhone. Our system will have a gateway in the house that coordinates communication to and from the app with the numerous devices in the house. The devices will be connected to the gateway using Bluetooth wireless communication. We will use a centralized dynamic domain name server to allow users to connect their app with the specific gateway in their house. Our software is designed to allow new types of devices to be easily included in the system.

Project Environment:

* + Mobile Application: We are developing our mobile application for iOS, primarily targeting iPhones. We may also create and Android application if we have time.
  + Gateway: Our gateway will be a small Intel based Linux server.
  + Device Controllers: The controller for each device will be an Arduino board with a Bluetooth shield to communicate with the gateway.
  + DNS server: The DNS server will store the IP addresses of every gateway, so the app can communicate directly with the gateway.

Project Deliverables:

* iPhone application
* Gateway software
* DNS software
* Prototype Arduino control board
* Documentation

Backlog:

* Server Configuration
* iOS Register Device
* Communications Protocol
* iOS Main View
* iOS Status View
* iOS Garage Door opener Controller View
* Garage Door opener
  + - * 1. Sprint 2 Progress Report

Hardware

All hardware components except the Bluetooth Shield have been acquired for the garage door. An additional Arduino will be necessary for the sprinkler system, but one of the requisite valves has been purchased. Wiring will commence and a working opener over Serial communication will be ready by the end of next sprint.

iOS Interface

A rudimentary prototype for the opener app has been completed. The app will be updated when more information on server-phone communications is available for the iOS developer to work with.

Gateway Server

No progress was made on the Gateway Server interface for this sprint, in favor of getting the iOS and hardware applications resolved. Gateway will not likely see further development until after initial hardware and full communication protocol is complete.

* + - * 1. Sprint 3 Progress Report

iOS

James continued working on the top level framework for the iOS application and created a prototype for the first run of the device where the user will register their device with a base station.

Resolution Server

Josh began working on the resolution server and created the first draft of the base station to resolution server communications protocol that allows base stations to update their IP addresses.

Base Station

We did not made any progress on the Base Station this sprint.

Hardware

Chris continued experimenting with a garage door opener to find a way to control it with an Arduino board.

* + - * 1. Sprint 4 Progress Report

Overview

This sprint report is to give an update on the progress of the Remote Home team on the development of home devices controlled by a user’s smart phone. This report will cover a recap of progress that has been made last year. Additionally, it will cover the progress made this year in the top level iOS framework, the iOS mobile application, the domain name resolution server, and the garage door hardware. Finally this report will give a list of prototypes for this sprint.

Sprint 1-3 Progress

Last year the Remote Home team was given the task of developing a system that allows users to control devices in their home, such as a garage door, from an application on their iPhone. The system was designed to have a “base station” in the house that coordinates communication to and from the application with numerous devices in the house. The system was also designed to have a centralized dynamic domain name resolution server to allow users to connect their application with a specific base station in their house, without the user needing to have a static IP address. The devices in the house will be connected to the base station using Bluetooth wireless technology. The system was designed to allow new types of devices to easily be included into the system.

James started developing a top level iOS framework that will link a user with a specific base station and all of the devices associated with the base station. The framework also manages the devices associated with the base station and allows the user to register new devices and remove devices that are currently on the system.

Joshua created a prototype garage door opener application that the user will use when controlling their garage door. He also began developing a dynamic domain name resolution server that will store the IP addresses of every base station, so that a user’s application can communicate directly with the base station located in their home. He finally started developing a communication protocol that will allow a base station to update its IP address on the dynamic domain name resolution server.

Chris started by gathering the necessary hardware for the project, e.g. garage door opener, and Arduino microcontroller. After gathering the hardware Chris then began working on controlling the garage door opener over serial communications.

iOS Framework

James completed developing and debugging the top level frame work over the Christmas break. base stations and devices can now be registered on the system.

Resolution Server

Joshua continued developing the dynamic domain name resolution server, running into some issues when connecting to the iOS framework. These issues are currently being resolved and the resolution server should be completed before the next sprint.

Hardware

This sprint Chris acquired the Bluetooth shield, the final piece of hardware need required for the project. Chris also successfully controlled the garage door opener over serial communication using the Arduino. This shows a proof of concept in opening a garage door using an Arduino. The next step is to control the garage door opener using the Bluetooth shield.

Prototypes

The Remote Home team produced two prototypes this sprint. The first prototype was the iOS framework developed by James. The client is now able to register both base stations and devices on the system. The second prototype is a garage door opener controlled over serial communication using an Arduino.

* + - * 1. Sprint 5 Progress Report

iOS

The top-level framework and Garage Door Controller have reached a beta state. The final testing will occur over the next sprint, with emphasis on ensuring integration between iOS components and the rest of the system.

Resolution Server

The Resolution Server is undergoing initial testing. Revisions are expected as integration commences, but it should be ready for deployment.

Base Station

The Base Station has reached an alpha state, with a threaded server and basic configuration files for controlling User and Device data. Currently only USB serial communication is supported, with TCP/IP and HCI/Bluetooth support being deferred.

Hardware

The garage door opener is ready for more intensive testing. It currently supports opening devices over USB. The HCI chip was discovered to not work with the relay board, so some modifications to the hardware will be necessary for Bluetooth support. Bluetooth support will be deferred.

* + - * 1. Sprint 6 (Final) Progress Report

iOS

James did unit testing for the iOS application framework. Josh modified the garage door view controller to work with the revised garage door messages and did unit testing for the garage door view controller.

Base Station

Chris completed the base station, including the web based front end, and did unit testing for the base station.

Garage Door Controller

Brian converted the garage door controller to a state based system and completed the controller. He also did unit testing for the controller.

Resolution Server

Josh has added error handling to the resolution server and has completed unit testing of the resolution server.

Integration

The team has integrated the different components of the system and has done most integration testing. Some additional testing will be done this weekend.

Poster

The team has put the poster for the design fair together.