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*

Table of Contents

1.0 Overview

The purpose of this document is to give the reader an understanding our project. Remote Home is a system that will allow users to control devices in their house from anywhere in the world using their iPhone. We will be focusing on controlling garage doors and sprinkler systems for this project. However, our system will be easily extendable to allow any electronic device to be controlled without any major changes to the system. The Remote Home system has four major components: an iPhone application to act as a remote control, a Base Station in the house that will connect the devices to the internet, a Resolution Server that will allow the iPhone application to connect directly to a Base Station, and the actual devices that will be controlled.

1.1 Scope

This document will describe what the purpose and requirements of our system are and how it works, including details about the major components and what technologies we are using. It will also discuss our team and development process.

1.2 Purpose

The purpose of this project is to allow people to control electronic devices in their house over the internet using their smartphone.

1.2.1 iPhone Application

The users will mostly interact with the system using an application on their smartphone. This application will allow 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.

1.2.2 Base Station

The Base Station will connect all of the devices in 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.

1.2.3 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.

1.2.4 Devices

The devices to be controlled will be electronic devices that can to receive commands from the Base Station as well as send information back to the sender. A device could be anything from a lamp, which could respond to a single toggle command, to a complex programmable thermostat that would allow users to adjust the temperature or create heating schedules from their iPhone.

1.3 Systems Goals

The goal of the Remote Home system is to allow people to control devices in their house from anywhere they can connect to the internet.

1.4 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.

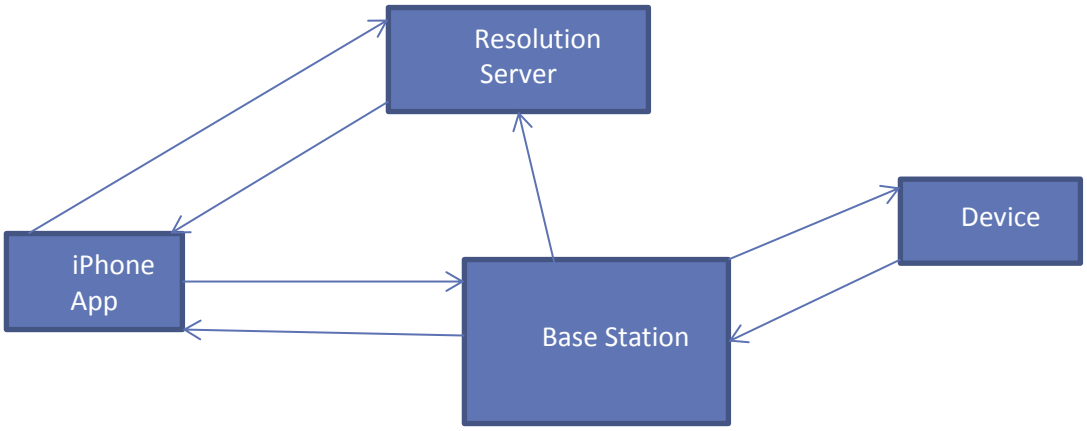


Figure 1 System Diagram

1.4.1 iPhone Application

The iPhone application will allow users to view and add Base Stations and view and control devices associated with those Base Stations. The application will connect directly with a Base Station using TCP. It will use serial number of the Base Station to get the IP address from the Resolution Server. All communication between the Base Station and the application will be secured using TLS. The application will store the serial number, password, and user friendly name of each base station on the phone.

1.4.2 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.

1.4.3 Resolution Server

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

1.4.4 Devices

These are the actual devices that will be controlled by the iPhone. They will be regular electronic devices 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 to the iPhone to the Base Station.

1.5 Technologies Overview

Our system will be using a variety of technologies. Each component's technology is discussed in the component's sub section in section 4. Our communication is based on TCP/IP and uses JSON to structure information sent between components.

2.0 Project Overview

2.1 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. Chris Jensen is working on the hardware and the Base Station software. Brian Vogel will join our team in January and will work with Chris on the hardware and Base Station.

2.2 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.

2.3 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.

2.4 Terminology and Acronyms

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

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.

iOS – The operation 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 will have a database that will store associations between a serial numbers and IP address.

Serial Number – The MAC address of a base station.

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.

3.0 Requirements

3.1.1 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.

3.1.2 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.

3.1.3 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.

3.1.4 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.

3.1.5 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 smartphone to establish a direct connection to the Base Station.

4.0 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.

4.1 iPhone Application

4.1.1 Technologies Used

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

4.1.2 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.

4.1.3 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

4.1.4 Architecture Diagram

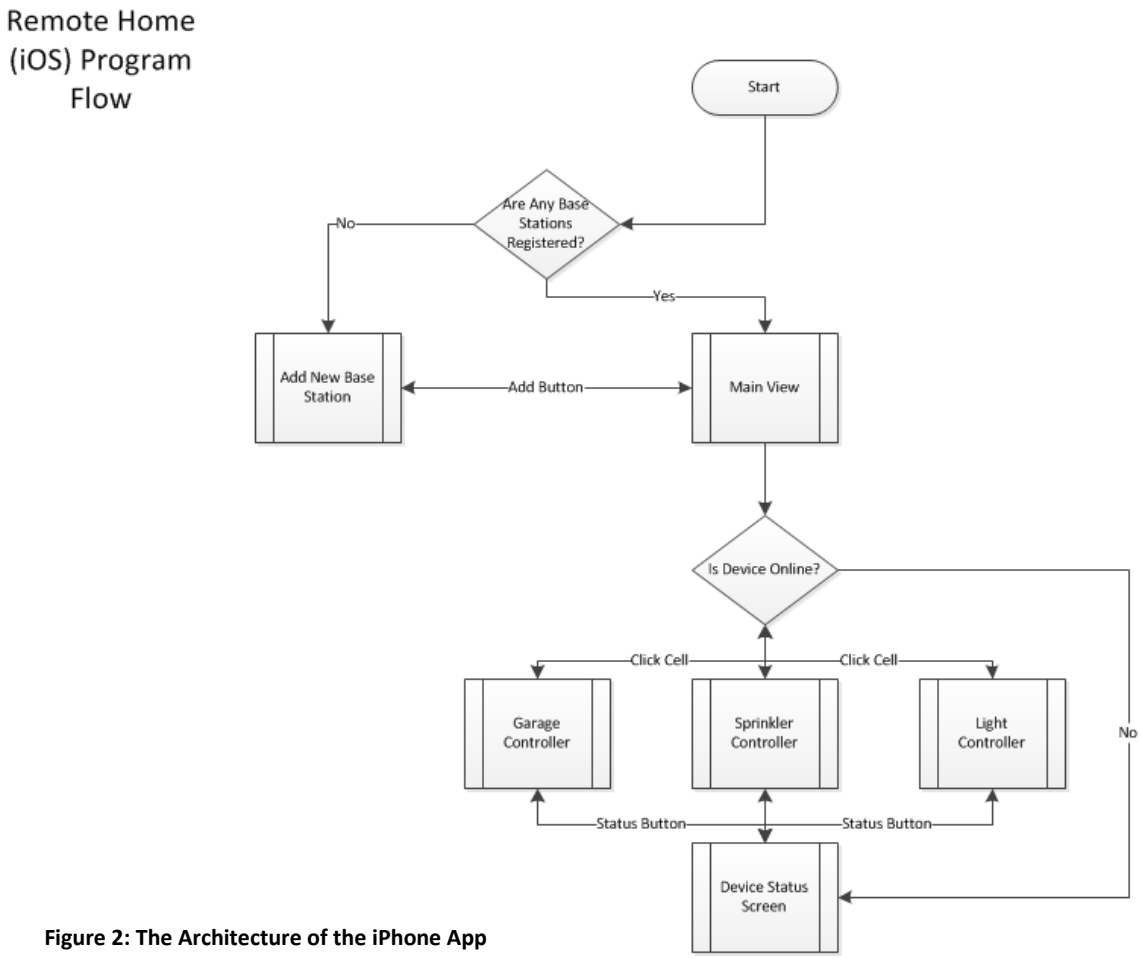


Figure 2: The Architecture of the iPhone App

4.1.5 Data Flow Diagram

It is important to build and maintain a data flow diagram. However, it may be that a component is best described visually with an architecture diagram.

4.1.6 Design Details

4.1.6.1 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.

I.1.1.2 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.

4.2 Base Station

4.2.1 Technologies Used

4.2.2 Component Overview

4.2.3 Phase Overview

4.2.4 Architecture Diagram

4.2.5 Data | Logic Flow Diagram

4.2.6 Design Details

4.3 Resolution Server

4.3.1 Technologies Used

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

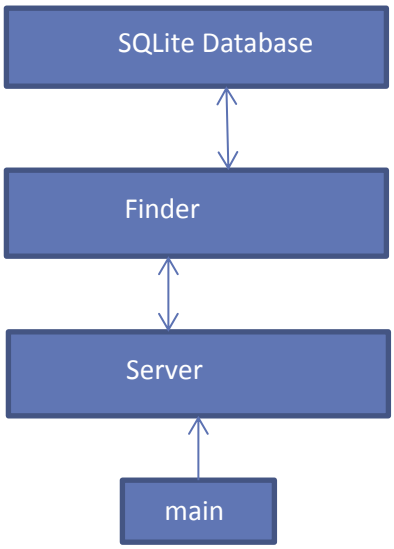
4.3.2 Component Overview

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.

4.3.3 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

4.3.4 Architecture Diagram



4.3.5 Design Details

4.3.5.1 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.

4.3.5.2 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.

4.3.5.3 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.

4.3.5.4 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.

4.4 Devices

5.0 System and Unit Testing

5.1.1 iPhone Application

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

5.1.2 Base Station

The Base Station will be tested by sending it requests and looking at the onboard logs to determine how it handled the request.

5.1.3 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.

5.1.4 Devices

The device controllers can be tested using the standard Arduino testing tools.

5.2 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.

5.3 Dependencies

5.3.1 iPhone Application

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

5.3.2 Base Station

5.3.3 Resolution Server

The Resolution Server depends on Python 2.7.

5.3.4 Devices

The devices will each require an Arduino control board.

5.4 Test Setup and Execution

5.4.1 iPhone Application

5.4.2 Base Station

5.4.3 Resolution Server

5.4.4 Devices

6.0 Development Environment

6.1.1 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.

6.1.2 Base Station

6.1.3 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.

6.1.4 Devices

The Arduino based device controllers require the free Arduino development tools.

6.2 Development IDE and Tools

6.2.1 iPhone Application

The Xcode IDE will be used.

6.2.2 Base Station

6.2.3 Resolution Server

Development can be done in any plain text editor.

6.2.4 Devices

6.3 Source Control

We are using Github for source control. It has a Documents directory to store our documentation and a src directory for our project code.

6.4 Dependencies

Describe all dependencies associated with developing the system.

6.5 Build Environment

6.5.1 iPhone Application

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

6.5.2 Base Station

6.5.3 Resolution Server

Since Python is an interpreted language, it does not need to be built. It will run on any Linux or UNIX computer with Python 2.7 installed.

6.5.4 Devices

6.6 Development Machine Setup

6.6.1 iPhone Application

A Mac with Apple's developer tools installed.

6.6.2 Base Station

6.6.3 Resolution Server

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

6.6.4 Devices

7.0 Release | Setup | Deployment

7.1.1 iPhone Application

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

7.1.2 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.

7.1.3 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.

7.1.4 Devices

7.2 Deployment Information and Dependencies

7.2.1 iPhone Application

7.2.2 Base Station

7.2.3 Resolution Server

7.2.4 Devices

7.3 Setup Information

7.4 System Versioning Information

8.0 End User Documentation

8.1 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.

8.2 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.

8.3 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

Appendix I: List of Figures

Appendix II: Supporting Information and Details

II.1 Communication Protocols

II.1.1 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.

II.1.1.1 DDNSConnected

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

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

II.1.1.2 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)"}, ...] }

II.1.1.3 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}, ...] }

II.1.2 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.

II.1.2.1 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)”}}

Appendix III: 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.

III.1 Sprint 1 Progress Report

This would be the first sprint report.

III.2 Sprint 2 Progress Report

This would be the second sprint report.