HomeAlive
Team – 9
2014-2015

Andrew Jo
Okkar Myint
Hezkiel Nanda
Jeremy Ward

Engineering 339 Senior Design Project
Calvin College
December 08, 2014
i. Executive Summary

Most homes do not utilize the technology available in the modern age and the electrical devices within them are unnecessarily inaccessible. This project will show that the addition of a cost-effective, aftermarket home automation system to a house would result in productivity, security, and environmental conservation gains. The project’s system prototype shall allow the reliable control and monitoring of household devices via remote user interfaces.

The design of a home automation system that satisfies this need could take many various forms; the implementation that has been preliminarily chosen includes sample devices, a central hub, a webserver, and user interfaces. These components will form a network that allows information and commands to be sent from the user interfaces to a specific device, or from a device to the user interfaces. Each sample device will require hardware design, the webserver and user interfaces will require software design, and the central hub will require both. Additionally, the wired and wireless communication between these components will require significant hardware and software design with a focus on protocol selection and utilization.

The final prototype that will be designed and implemented throughout this project must be completed by May, 2015. In addition to the development of a functional prototype, this project includes giving presentations, recording documentation, writing reports, and maintaining a web presence. A project budget has not yet been established; however, it is expected that $500 will be allocated to the project by the Calvin College engineering department. This funding will be spent wisely in order to obtain the development tools and physical parts necessary to implement the design. The team responsible for this project consists of four electrical and computer engineering students, Andrew Jo, Hezkiel Nanda, Jeremy Ward, and Okkar Myint. Calvin College engineering professor Mark Michmerhuizen, Gentex employee Eric Walstra, and SpinDance Inc. employee David van Geest have generously agreed to offer support for the team.

This project was begun in September, 2014 and the progress to date is explained in detail throughout this feasibility report. The most notable progress areas involve system architecture and proposed project scope. A preliminary system architecture has been decided upon, but facets of the architecture that require research, testing, and refinement have also been identified. Since project conception, its proposed scope has been decreasing. The original idea was too ambitious for the available time and resources; however, its essential features have been identified and represent a more appropriate scope. Project scope will be reevaluated periodically throughout the project and nonessential features will be re-included as time allows. Additionally, preliminary implementation and design has been started for the central hub and two devices.
## Contents

i. Executive Summary .................................................................................. c

ii. Table of Figures ...................................................................................... iv

iii. Table of Tables ...................................................................................... iv

iv. Abbreviations ........................................................................................... v

1 Introduction ................................................................................................. 6

1.1 Report & Project Context ......................................................................... 6

1.2 Problem Statement ................................................................................... 6

1.3 Project Definition & Objectives ................................................................. 6

1.4 Scope & Constraints ............................................................................... 7

2 Project Management .................................................................................. 8

2.1 Team Assignments & Organization ......................................................... 8

2.1.1 Team Assignments ............................................................................... 8

2.1.2 Team Organization .............................................................................. 9

2.2 Deliverables .............................................................................................. 9

2.3 Schedule .................................................................................................. 10

2.3.1 Milestones .......................................................................................... 10

2.3.2 Schedule Management ......................................................................... 10

2.3.3 HoursLogged ...................................................................................... 10

2.4 Operational Budget ................................................................................. 12

2.4.1 Project Costs ......................................................................................... 12

2.4.2 Sources of Funding ............................................................................. 12

2.5 Task Specifications and Schedule ............................................................ 12

3 Design ......................................................................................................... 14

3.1 Design Norms .......................................................................................... 14

3.1.1 Stewardship ....................................................................................... 14

3.1.2 Justice ................................................................................................ 14

3.1.3 Trust ................................................................................................... 14

3.2 Requirements ............................................................................................ 14

3.2.1 Requirements Categories .................................................................... 14

3.2.2 Technical Requirements ..................................................................... 15

3.3 System Architecture .................................................................................. 15

3.3.1 System Separation & Conceptual Diagram ......................................... 15

3.3.2 System Modularization & Technical Diagram ..................................... 17

3.3.3 Hardware ............................................................................................. 20
3.3.4 Software ............................................................................................................. 20
3.4 Design Alternatives ............................................................................................... 20
  3.4.1 Central Hub OS and Networking ...................................................................... 20
  3.4.2 Terminal Device to Central Hub Communication Protocol .......................... 21
  3.4.3 Central Hub Processing Hardware .................................................................. 22
  3.4.4 Central Hub Power Supply ............................................................................. 23
  3.4.5 Server & Central Hub Integration ................................................................... 24
  3.4.6 Webserver Hosting ........................................................................................ 26
  3.4.7 Database Structure ....................................................................................... 27
  3.4.8 Web Application Framework ......................................................................... 28
  3.4.9 Central Hub to Webserver Messaging Protocol ............................................ 29
  3.4.10 Web Portal Programming Language ........................................................... 30
  3.4.11 User Interface ............................................................................................... 30
3.5 Design Alternative Selection Summary ............................................................... 31
3.6 Testing Definitions ................................................................................................. 31
4 Business .................................................................................................................. 33
  4.1 Mission for the Company .................................................................................. 33
  4.2 Vision for the Company .................................................................................... 33
  4.3 Potential Customers & Target Market ............................................................... 33
  4.4 Competitive Strategy ....................................................................................... 33
    4.4.1 Differentiation ............................................................................................. 33
    4.4.2 Response ..................................................................................................... 33
  4.5 Market Sizes & Trends ....................................................................................... 34
  4.6 Business Strategy & Distribution ....................................................................... 34
  4.7 SWOT Analysis .................................................................................................. 34
    4.7.1 Strengths ..................................................................................................... 34
    4.7.2 Weaknesses ................................................................................................. 34
    4.7.3 External Opportunities ................................................................................ 34
    4.7.4 External Threats ......................................................................................... 34
  4.8 Uniqueness & Existing/Potential Competitors .................................................... 35
5 Conclusion ................................................................................................................ 35
  5.1 Potential Risks & Issues .................................................................................... 35
  5.2 Project Status .................................................................................................... 36
  5.3 Project Feasibility ............................................................................................... 36
  5.4 Summary ............................................................................................................ 36
6 Acknowledgements .................................................................................................. 37
ii. Table of Figures

Figure 1. Team Hours as of Dec. 07, 2014 by Category, Totaling 382 Hrs.......................... 11
Figure 2. Team Hours as of Dec. 07, 2014 by Person & Week ........................................ 11
Figure 3. Conceptual System Diagram ................................................................. 16
Figure 4. Technical System Diagram ..................................................................... 18
Figure 5. Category Workload Graph ........................................................................ 39
Figure 6. MQTT Messaging Implementation ............................................................. 40
Figure 7. RabbitMQ Messaging Implementation ......................................................... 41
Figure 8. Thermostat Device System Integration ....................................................... 43
Figure 9. Thermostat Mechanical Interface ............................................................ 44
Figure 10. Thermostat Circuit ................................................................................. 44
Figure 11. Temporary Thermostat GUI ................................................................. 45
Figure 12. Thermostat Analog Circuit ....................................................................... 45
Figure 13. Outlet Adapter System Integration ......................................................... 48
Figure 14: Outlet Adapter Physical Separations ....................................................... 49
Figure 15: Receiver Photograph ............................................................................. 50
Figure 16: Transmitter Photograph .......................................................................... 50
Figure 17: Modified XBee Serial Adapter Schematic .............................................. 51
Figure 18: Wattcher.py Logical Diagram ................................................................. 52

iii. Table of Tables

Table 1. Course Instructors for Fall 2014 Engineering 339 ........................................... 8
Table 2. Project Milestones ..................................................................................... 10
Table 3. Projected Budget ....................................................................................... 12
Table 4. Work Breakdown Schedule Summary ....................................................... 13
Table 5. Decision Matrix for Central Hub OS ........................................................ 21
Table 6. Decision Matrix for Terminal Device to Central Hub Communication Protocol 22
Table 7. Decision Matrix for Hardware System for Central Hub .............................. 23
Table 8. Decision Matrix for Central Hub Power Supply .................................... 24
Table 9. Decision Matrix for Server & Central Hub Integration ............................. 26
Table 10. Decision Matrix for Server Hosting ....................................................... 26
Table 11. Decision Matrix for Database Structure ............................................... 27
Table 12. Decision Matrix for Web Application Framework .................................. 29
Table 13. Decision Matrix for User Interface ......................................................... 31
Table 14. Design Alternative Selection Summary ............................................... 31
Table 15. Existing Competitors in the Market with Devices ................................... 35
Table 16. Features Comparison among Commercially Available Home Automation Systems ... 35
### Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>AMQP</td>
<td>Advanced Message Queuing Protocol</td>
</tr>
<tr>
<td>DIY</td>
<td>Do It Yourself</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>JSP</td>
<td>JavaServer Pages</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>M2M</td>
<td>Machine-to-machine</td>
</tr>
<tr>
<td>MOM</td>
<td>Message Oriented Middleware</td>
</tr>
<tr>
<td>MQTT</td>
<td>MQ Telemetry Transport</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-View-Controller</td>
</tr>
<tr>
<td>ORM</td>
<td>Object-Relational Mapping</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PPFS</td>
<td>Project Proposal Feasibility Study</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RoR</td>
<td>Ruby on Rails</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real Time Operating System</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>WAF</td>
<td>Web Application Framework</td>
</tr>
<tr>
<td>WBS</td>
<td>Work Breakdown Schedule</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Report & Project Context

This home automation project is being worked on for Calvin College’s engineering senior design course. The senior design course is intended to both give its students experience working on a yearlong project and allow them to showcase what they have learned at Calvin. The capstone project will last from September 2014 to May 2015, and it includes everything from brainstorming project ideas through demonstrating the functionality of a prototype system. The project’s team members consist of four electrical and computer engineering students: Andrew Jo, Hezkiel Nanda, Jeremy Ward, and Okkar Myint.

This PPFS report is intended to explain the home automation project’s purpose, scope, and preliminary design implementation. It includes topics focused on scheduling, budgeting, and technical decisions; its overall conclusion regards the feasibility of the project.

1.2 Problem Statement

There are four problems and opportunities that are related to this project: house modernization rates, device inaccessibility, productivity improvements, and home monitoring.

House modernization rates refers to the problem that homes in the modern world are not taking advantage of new technologies. This is caused by house longevity and high renovation costs; the home automation system prototype should show that its mass production and installation would be possible at an affordable price.

Device inaccessibility refers to the problem that many household devices are unnecessarily difficult to interact with; it should be possible to control any electrical device one owns from anywhere in the world. The home automation system prototype should show that secure access to home devices is possible from remote locations.

Productivity improvements refers to an opportunity to introduce time-saving into the daily lives of those who would use the home automation system. Integrating household devices and allowing them to be automatically and remotely managed can increase ease-of-life and decrease wasted time.

Home monitoring refers to the opportunity to observe and control household devices more effectively. The home automation system should allow for greater security and reduced energy consumption by providing homeowners with a single interface for interacting with several devices. The system should also address this opportunity by providing automatic device control scheduling and home-mode settings.

1.3 Project Definition & Objectives

The home automation project consists of addressing the problems described above by designing and implementing a prototype version of the system. This system should allow for the wireless control of sample devices using an internet or smartphone interface; the system may also offer basic scheduling and automated control techniques. For example, it should be possible to check the power consumption of an integrated appliance and to schedule the toggling of overhead lights. The prototype will include a central hub, sample devices, host server, and user interfaces.

These system components communicate with one another in a hierarchal fashion in order to facilitate the wireless control of devices from interfaces. Devices are the physical, electrical machines that are being monitored and controlled, for example overhead lights. They will communicate wirelessly with the central hub, which will organize information from all devices and pass it on to the host server via an internet connection. The host server will include a database of system information and be accessible from both the central hub and user interfaces. User interfaces include both a web
portal and smartphone application. They will allow for an understandable portrayal of system information and an intuitive console for sending system commands.

1.4 Scope & Constraints

This home automation prototype is meant to demonstrate the proper functionality and scalability of the system, not to include every desired system feature. There are countless features and devices that the system could eventually incorporate, but in order to maintain proper project scope, only basic user interfaces and a few sample devices are being designed. Determining an appropriate scope for this project is perhaps the biggest challenge that will be faced; however, the prototype will include at least: a sample device, the central hub, a web server, a web portal user interface, and communication structures between them.

These components were selected as the minimum set required to demonstrate appropriate system functionality because they make up the communication network between users and the devices that are being controlled. Once this structure is in place, additional devices, features, and user interface options can be added to the prototype in order to increase project scope.
2 Project Management

2.1 Team Assignments & Organization

The teams consists of four members: Andrew Jo, Okkar Myint, Hezkiel Nanda, and Jeremy Ward. All four team members are electrical and computer engineering students at Calvin College and expecting to graduate in May 2015.

2.1.1 Team Assignments

Team members are assigned the following roles in order to increase the efficiency of the team.

2.1.1.1 Andrew Jo: Project Manager

As a Project Manager, Andrew Jo’s assignment is to set milestones and schedule the project by keeping track of deadlines, turn-in assignments. His roles is essential in driving the team in a forward direction.

2.1.1.2 Okkar Myint: Research & Development Manager

As a Research and Development Manager Okkar’s assignment is giving team members specific topics to research while managing the team’s efforts in collecting research for the different components of the design. This assignment includes developing the research into usable concepts and designs.

2.1.1.3 Hezkiel Nanda: Website & User Interface Developer & Financial Manager

As the Website and User Interface Developer Hezkiel’s assignment is structuring the project website and releasing appropriate documents and reports to the public. He also gathers user feedback and assures the interface of the design is intuitive and accessible at every level. As a Financial Manager Hezkiel’s assignment is monitoring the budget of the project and estimating expenses.

2.1.1.4 Jeremy Ward: Professionalism Manager & Marketing Manager

As a Professionalism Manager Jeremy’s assignment is verifying that the deliverables comply with professionalism codes and ethics. This includes proofreading and formatting. As Marketing Manager Jeremy’s task is developing the marketing strategy and ensuring that the strategy is embodied in the finished product’s design.

2.1.1.5 Team Advisor

Mark Michmerhuizen is an electrical and computer engineering professor at Calvin College and serves as the project’s faculty advisor.

2.1.1.6 Course Instructors at Calvin College

The followings serve as course instructors for ENGR-339.

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ned Nielsen</td>
<td>Mechanical Engineering Professor</td>
</tr>
<tr>
<td>David Wunder</td>
<td>Civil and Environmental Engineering Professor</td>
</tr>
<tr>
<td>Jeremy VanAntwerp</td>
<td>Chemical Engineering Professor</td>
</tr>
<tr>
<td>Mark Michmerhuizen</td>
<td>Electrical Engineering Professor</td>
</tr>
</tbody>
</table>

2.1.1.7 Mentor

David van Geest is a Calvin College graduate from 2009. He works at SpinDance, a company that specializes in the development of technologies in the “internet of things” field and is based in Holland, Michigan. David graciously offers his expertise and feedback for the project.
2.1.1.8 Industrial Consultant

Eric Walstra works at Gentex, a company that mainly specializes in developing electronic products in the automotive and aerospace industries and is based in Zeeland, Michigan. Eric has met with the team and given insight to the project regarding project scope, task scheduling, risk assessment, and design testing. Eric will meet the team again in spring 2015 to offer more advice and project feedback.

2.1.2 Team Organization

2.1.2.1 Team Meetings

There is a scheduled team meeting every Friday from 1:00 PM to 1:30 PM. This weekly meeting gives the team a chance to go over the progress of the last week and schedule new assignments as needed. This meeting holds the team members accountable and ensures the project’s progress.

The team plans to add a three hour weekly meeting on Saturdays. This meeting would provide for time each week to work on issues that require the whole team’s attention.

2.1.2.2 Team Documentation & Task Organization

All of the team documents, including research notes, reports, diagrams, presentations, schedules, budgets, and test results, are kept on a shared Microsoft OneDrive account. These documents are only accessible by team members.

The team assigns tasks using an online scheduling tool called Asana. The tool allows reminders to be sent at set times, and it allows team members to view each other’s progress. This tool has proved very effective and useful in task organization, time management, and adherence to schedule.

2.2 Deliverables

This home automation project includes several deliverables including: a final showcase, a final report, a PPFS, team presentations, a project website, and design documentation.

The final showcase of the project is scheduled for May, 2015 and includes both a presentation and demonstration of the prototype system. The presentation will describe the project including its learning outcomes and significant challenges. The prototype demonstration will allow individuals in an open-house setting to interact with the user interface components in order to control the device components. This hands-on showcase is the project’s culminating deliverable.

The final report for this project will be submitted in May, 2015 and include all relevant project and design information. The technical document will explain project timeline, budgeting, and team organization. It will also describe the rationale behind the design decisions that were made and highlight how the design was physically implemented. Notice that this document is similar to the PPFS, but differs because it will focus more on implementation and testing instead of requirements and scheduling.

The PPFS report for this project will be submitted in December, 2014 and include all information relevant to a preliminary analysis of the project. The focus of this technical document is to establish feasibility for the project by evaluating the project’s scope, human resources, physical resources, and available time. In order to most accurately evaluate these aspects of the project, preliminary design decisions will have to be made and research will have to be done. Furthermore, early implementation attempts and proofs of concept will be conducted.

Team presentations are intended to communicate the state of this project as it progresses towards its deadline. The presentations are oral with slideshow visuals, and they will focus on changes since the previous presentation was given. Project status includes high-importance design decisions, schedule projections, and feasibility conclusions.
The project website will include all relevant project information and be updated fairly frequently. Relevant project information includes biographies on team members, copies of all deliverable documents, informal progress updates, and descriptions of the system and its components. In addition, the project website will allow access to the system’s web portal user interface.

Project design documentation will be delivered regularly and serve as justification for design decisions that have been made. The periodicity of its delivery is intended to highlight changes in project scope, continued progress, and significant challenges that have arisen. Design documentation can take many forms including text documents, spreadsheets, diagrams, budgeting worksheets, and test results. However, each will include a brief summary explaining its purpose and relevance to the project.

2.3 Schedule

In order to achieve its goals and make progress towards its milestones, HomeAlive breaks down their workload into manageable pieces, and assigns weekly tasks to every team member. Progress on tasks is reviewed every Friday, and the schedule is updated at this time. Large project milestones are tracked using Microsoft Project and include items such as writing the PPFS and completing the website. Weekly tasks are tracked using Asana and include shorter items that make progress towards project milestones, for example adding photographs to the website. This approach to scheduling is implemented as described in the task specifications section below.

2.3.1 Milestones

This project has several notable milestones in its schedule as shown in Table 2. These are both design process milestones, with goal completion dates, and deliverable content milestones, with assigned due dates.

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oral Presentation 1</td>
<td>10/17/2014</td>
</tr>
<tr>
<td>PPFS Rough Draft</td>
<td>11/10/2014</td>
</tr>
<tr>
<td>Oral Presentation 2</td>
<td>12/3/2014</td>
</tr>
<tr>
<td>PPFS Final Draft</td>
<td>12/8/2014</td>
</tr>
</tbody>
</table>

2.3.2 Schedule Management

The scheduling approach for this project involves a master WBS, task lists, estimations, and reevaluations. Originally, a master WBS was generated which includes large scale tasks. It incorporated rough time estimates and due dates. Next, the last three of these steps were done consecutively, and the process has been repeated at least one time per week since then. First, task lists are generated which include all relevant small and large scale tasks, the large scale tasks are taken from the master WBS. Second, each of these tasks are given an updated time-required estimation and due date. Third, estimations from the past iteration are examined for accuracy in order to improve the scheduling estimation process. Reevaluation also helps identify tasks that are behind schedule.

2.3.3 Hours Logged

The timing information in this section reflects project hours from 8/18/2014 to 12/07/2014. As seen in Figure 1, the majority of the time logged has been spent writing the PPFS (34%) and central hub work (15%). Brainstorming (8%) and creating the project website (8%) were also significant time uses. Total manhours to date are 382. Additionally, as seen in Figure 2, the hours spent each week on the project have been increasing steadily since August. This graph also displays the breakdown of hours for each team member each week; additional hour tracking graphics are available at <insert link once website content available>.
Figure 1. Team Hours as of Dec. 07, 2014 by Category, Totaling 382 Hrs.

Figure 2. Team Hours as of Dec. 07, 2014 by Person & Week
2.4 Operational Budget

The operational budget shown on Table 3 has a calculated financial range. The budget estimation uses a contingency factor of 1.5 account for the risk of part replacements and repairs. Table 3 provides estimated costs for the chosen hardware based on the design alternatives. HomeAlive has assigned Hezkiel Nanda to maintain its project budget. He will be responsible for updating budgetary information when unexpected repair or replacement occurs and for discussing financial issues with team supervisor and industrial mentor.

2.4.1 Project Costs

Table 3. Projected Budget

<table>
<thead>
<tr>
<th>Component</th>
<th>Unit Price</th>
<th>Quantity</th>
<th>Total Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Hub Transmitter: XBee [1]</td>
<td>$37.95</td>
<td>1</td>
<td>$37.95</td>
</tr>
<tr>
<td>Central Hub Wi-Fi Shield [2]</td>
<td>$19.95</td>
<td>1</td>
<td>$19.95</td>
</tr>
<tr>
<td>Central Hub Power Supply [3]</td>
<td>$7.89</td>
<td>1</td>
<td>$7.89</td>
</tr>
<tr>
<td>Central Hub Box Frame</td>
<td>$5.00</td>
<td>1</td>
<td>$5.00</td>
</tr>
<tr>
<td>Single Board Computer: Raspberry Pi¹</td>
<td>$0.00</td>
<td>1</td>
<td>$0.00</td>
</tr>
<tr>
<td>Cables &amp; Connectors [4]</td>
<td>$6.50</td>
<td>1</td>
<td>$6.50</td>
</tr>
<tr>
<td>Resistors, LED, and Transistors¹</td>
<td>$0.00</td>
<td>1</td>
<td>$0.00</td>
</tr>
<tr>
<td>Device Receiver¹</td>
<td>$37.95</td>
<td>2</td>
<td>$75.90</td>
</tr>
<tr>
<td>Device Frames [5]</td>
<td>$5.48</td>
<td>1</td>
<td>$5.48</td>
</tr>
<tr>
<td>Specific Device components (estimated)</td>
<td>$20.00</td>
<td>1</td>
<td>$20.00</td>
</tr>
<tr>
<td>Server Hosting²</td>
<td>$0.00</td>
<td>1</td>
<td>$0.00</td>
</tr>
<tr>
<td>Shipping Cost (estimated)</td>
<td>$20.00</td>
<td>1</td>
<td>$20.00</td>
</tr>
</tbody>
</table>

Subtotal: $198.67
Contingency Factor: x1.5
Total: $298.01

2.4.2 Sources of Funding

HomeAlive has one primary source of funding from Calvin College engineering department’s senior design fund. They will generously provide approximately $500.00 to fund the budget estimated in Table 3. Furthermore, they will also provide electrical equipment and components such as resistors, LEDs, transistors, cables, and connectors. There is one secondary source of funding, the Calvin College engineering department honors program. This budget should be used in order to purchase hardware components with an approximate amount of $100 per honors student. Two HomeAlive team members will do their honors projects in order to support the HomeAlive prototype. With these sources of funding, HomeAlive will have sufficient financial support for prototype design and implementation.

2.5 Task Specifications and Schedule

This home automation project includes numerous small and large scale tasks which are organized according to a flexible schedule. As the project progresses, the schedule becomes less flexible, so care will be taken to evaluate project status frequently. Task lists and several schedule versions have been generate and are available at <insert link here once content is uploaded>. A summary of the scheduling content, as of 12/6/14, exists in Table 4. Overall, the project is on-schedule. However, it is important to note that the ambitious and flexible scope of the project cause schedule estimation challenges.

The critical path through this project’s schedule is determined by the order in which future goals are pursued. The order of some future goals, for example server development and user interface development, is still being finalized but will most likely be: central hub, devices, server, user interfaces. Factors influencing future order include expected time required, dependent tasks, and assigned due
The most probable critical path is based on the debugging of intrasystem interfacing because it incorporates examinations of all system components.

Table 4. Work Breakdown Schedule Summary

<table>
<thead>
<tr>
<th>Tasks &amp; Milestones</th>
<th>Due Date</th>
<th>Due Date Type</th>
<th>Percent Complete</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Choice</td>
<td>9/10/14</td>
<td>Assigned</td>
<td>100%</td>
<td>Complete</td>
</tr>
<tr>
<td>Team Name &amp; Facilities</td>
<td>9/15/14</td>
<td>Assigned</td>
<td>100%</td>
<td>Complete</td>
</tr>
<tr>
<td>Scope Definition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentations &amp; Reports</td>
<td></td>
<td>Assigned</td>
<td>50%</td>
<td>On schedule</td>
</tr>
<tr>
<td>PPFS</td>
<td>12/8/14</td>
<td>Assigned</td>
<td>100%</td>
<td>Complete</td>
</tr>
<tr>
<td>Outlet Adapter Device</td>
<td>12/8/14</td>
<td>Milestone</td>
<td>75%</td>
<td>On schedule</td>
</tr>
<tr>
<td>Thermostat Device</td>
<td>12/8/14</td>
<td>Milestone</td>
<td>75%</td>
<td>On schedule</td>
</tr>
<tr>
<td>Central Hub</td>
<td>12/5/14</td>
<td>Milestone</td>
<td>25%</td>
<td>On schedule</td>
</tr>
<tr>
<td>User Interface</td>
<td></td>
<td>N/A</td>
<td>0%</td>
<td>Not started</td>
</tr>
<tr>
<td>Server</td>
<td></td>
<td>N/A</td>
<td>0%</td>
<td>Not started</td>
</tr>
<tr>
<td>Final Report</td>
<td>5/15</td>
<td>Estimated</td>
<td>50%</td>
<td>Not started*</td>
</tr>
<tr>
<td>Initial Prototyping</td>
<td>3/15</td>
<td>Milestone</td>
<td>15%</td>
<td>On schedule</td>
</tr>
<tr>
<td>Prototype Completion</td>
<td>5/15</td>
<td>Estimated</td>
<td>10%</td>
<td>On schedule</td>
</tr>
</tbody>
</table>

* uses PPFS content significantly
3 Design

3.1 Design Norms

Design norms are topics that are meant to facilitate the application of core Christian beliefs into the engineering design process. They provide guidelines for thinking about engineering in the holistic context of life, as opposed to just the product being designed. The design norms that are most important to this project are stewardship, justice, and trust. Together, they influence the development of the home automation system prototype by encouraging the careful consideration of how the system interacts with people during each stage of its product life-cycle.

3.1.1 Stewardship

Stewardship is defined as conserving finite economic, human, and environmental resources while preserving character and minimizing environmental degradation [6]. This home automation system adheres to the design norm of stewardship in three ways. First, economic resources will not be squandered in either the design or production of the system; care will be taken to use the available project budget as effectively as possible. Second, the system is intended to conserve human resources by increasing its consumers’ ease-of-life and improving the security of their homes. Finally, the system includes features that allow the monitoring and reduction of power consumption in consumers’ homes. In this way, the system contributes to the conversation of environmental resources and helps to minimize degradation.

3.1.2 Justice

Justice is defined as respecting the rights of all persons and considering all stake-holders, not just users [6]. This home automation system adheres to the design norm of justice in two ways. First, the system is designed to allow increased access to devices for people with limited mobility. By making the control of home devices possible through a smartphone application, it will be possible for many to achieve things alone that they previously needed help with. For example, a person who is paralyzed from the waist down and living in a multilevel, non-handicap-accessible home, may now easily control devices from any floor.

3.1.3 Trust

Trust is defined as ensuring that the design is trustworthy, dependable, reliable, and avoids conflicts of interests [6]. This home automation system adheres to the design norm of trust in three ways. First, the system functions entirely as intended and is not susceptible to inconsistent performance. This is especially important for a home automation system because users quickly come to rely on it and a lot of time is wasted doing things the old way if the system fails. Second, the issue of security is considered throughout the design of the system. Security is important because unintended access to the system could result in theft or other undesirable circumstances for the system’s owner. Finally, all conflicts of interest by the people involved in the design of this system are avoided. No one participating in the project has stake in any competing organizations or relationships with anyone in a contradictory circumstance.

3.2 Requirements

3.2.1 Requirements Categories

Requirements for this design can be divided and categorized in order to better ensure the generation of accurate specifications for the entire system. They are divided into system requirements and component requirements, each of which has categories as outlined below.

3.2.1.1 System Requirements

3.2.1.1.1 Capability Requirements

Capability requirements involve the ability of the system to achieve its intended purpose. They demand that the system allows its users to accurately accomplish the monitoring and controlling of their devices. Furthermore, capability requirements help ensure that the system satisfies its purpose and
addresses the problems described above. These requirements are likely influenced by the design norm of trust.

3.2.1.2 User Interface Requirements
User interface requirements involve the ability of the entire system to interact with its users. They demand that users are provided with certain methods to control and monitor the system, and they also demand restrictive features that guard against improper user interfacing. These requirements are likely influenced by the design norms of trust and justice.

3.2.1.2 Component Requirements
3.2.1.2.1 Functional Requirements
Functional requirements involve the ability of the component to achieve its intended use. They demand that the component accurately processes its inputs in order to produce appropriate outputs. Requirements of this sort are used to define the purpose of a component’s existence. For example, the central hub exists in order to relay wireless information from multiple devices through a single internet connection to the webserver. Thus, one of its functional requirements would be: can organize multiple information channel sources into a single information channel destination. These requirements are likely influenced by the design norm of trust.

3.2.1.2.2 Interface Requirements
Interface requirements involve the ability of separate components to communicate with one another. They demand that the component is able to send and receive information from other components in the appropriate mediums and protocols. For example, the central hub exists in order to relay wireless information from multiple devices through a single internet connection to the webserver. Thus two of its interface requirements would be: can communicate with a variable number of devices wirelessly according to an established protocol and can communicate with the internet either wirelessly or via Ethernet. These requirements are likely influenced by the design norms of trust and justice.

3.2.1.2.3 Performance Requirements
Performance requirements involve the ability of a component to accomplish its task relative to some metric, such as speed or power efficiency. They demand that the component operates in a quality manner and are crucial to making the system useful. For example, the central hub exists in order to relay wireless information from multiple devices through a single internet connection to the webserver. Thus two of its performance requirements would be: can relay the necessary information in X or fewer milliseconds and uses Y or less watts of power (where X and Y are values). These requirements are likely influenced by the design norm of stewardship.

3.2.2 Technical Requirements
System requirements will be stated explicitly and in great detail on the final project report. Each system module will be subject to requirements from each of the categories described above. Qualitative, capability-based requirements are explained in the system architecture and design alternatives sections below; these will yield specific technical requirements as the initial stages of prototype implementation occur. These initial implementation decisions are important for determining technical requirements because they provide a foundation from which to define specifications. For example, the decision between battery and corded power supplies must be made before central hub power consumption specifications can be declared.

3.3 System Architecture
3.3.1 System Separation & Conceptual Diagram
This home automation system has been designed modularly as several components according to both concrete and abstract distinctions between different parts of the system. At its broadest level, system inputs are the user inputs: device control and system control. Broad system outputs include device information and system information, which are displayed to users in order to allow them to make
informed control decisions. This can be seen in Figure 3, which also highlights the modular distinctions described below.

![Conceptual System Diagram](image)

**Figure 3. Conceptual System Diagram**

This diagram shows system inputs and outputs and some component distinctions within the system. Notice that internal system communications include RF-Wireless and internet (Wi-Fi or Ethernet).

The system shown conceptually in Figure 3 is divided into components according to perceived concrete and abstract distinctions. First, the phone application and web portal are separated from the remainder of the system using the concrete distinction that they are software modules running on arbitrary devices, and using the abstract distinction that they are the user interface modules for
interaction with the system. Second, the webserver is separated from the remainder of the system using the concrete distinction that it is a software module running on a server host machine. It is also separated using the abstract distinction that it is the computational and data master of the system, the webserver is the main component that is not dedicated to strictly interface translations. Third, the central hub is separated from the remainder of the system using the concrete distinction that its hardware exists in only place, as opposed to on several devices. It is also separated using the abstract distinction that it is responsible for relaying device information to the webserver. Finally, each device is separated from the remainder of the system using the concrete distinction that the hardware of each does not interact with any other part of the system. They are also separated using the abstract distinction that they are the source of the information within the system and the destination of commands given to the system.

3.3.2 System Modularization & Technical Diagram

3.3.2.1 Overview

The conceptual system diagram explained in the previous section can be expanded into a technical system diagram as seen in Figure 4. The technical diagram shows greater detail within each system component, from Figure 3, by separating them into even smaller modules. These modules are categorized as either hardware or software, and the interface protocols between them have been explicitly declared. Notice that the specific protocols and hardware labelled in the diagram show only one of several alternatives for each module, the estimated best alternative.
This diagram shows system component separation into hardware and software modules. It also clarifies that interface protocols will be needed for communication between components. Notice that the specific protocols and hardware labeled in the diagram show only one of several alternatives for each module, the estimated best alternative.
3.3.2.2 Devices

The devices component, seen in Figure 4, is separated according to each device; each device is then separated further into hardware, software, and interface modules. Hardware modules for each device include the essential device hardware, control circuitry, and an RF communication chip. The essential device hardware is comprised of the actual lighting system, thermostat, coffee machine, or other household object. The control circuitry is attached to the essential device hardware in such a way that it can execute the desired monitoring and controlling of the household object. The RF communication chip is responsible for relaying information wirelessly from the control circuitry to the central hub, and from the hub to the control circuitry. It also hosts the only device component software module, which includes basic configurations for the wireless communication chip. Finally, there is a wireless communication interface protocol module which is meant to show that the information being relayed by the RF communication module has had a logical structure imposed on it. Protocols provide a layer of abstraction between data and their meaning.

3.3.2.3 Central Hub

The central hub component, seen in Figure 4, is separated into two hardware modules, their associated software modules, and three interface protocols. It also includes two additional, minor hardware modules. The first main hardware module is the RF communication chip. It is responsible for relaying information from all of the devices to the hub’s processor, and for relaying information from the hub’s processor to the correct individual device. The second main hardware module is the processor which is primarily responsible for translating messages received from the RF communication chip into internet-based messages, these are destined for the webserver’s database manager module. The software module running on the RF communication chip includes basic configurations for the chip. The software module running on the processor includes both a small operating system and a networking structure that allows it to access the internet and pass messages to the webserver.

The two interface protocols include webserver communication and wireless communication. The webserver communication protocol allows the processor to make and receive understandable requests from the webserver. Next, the wireless communication interface protocol module is meant to show that the information being relayed by the RF communication module has a logical structure imposed on it.

The minor hardware modules include debug LEDs and a power supply. The debug LEDs are labeled on the diagram in order to convey one method by which the central hub, and other parts of the system, will be examined for proper functionality. The power supply is labeled in order to clarify that the central hub is the only component in the system that is being powered independently from any external power sources.

3.3.2.4 Web Server

The webserver component, seen in Figure 4, is separated into software and interface modules. The basis of the webserver is the database storage module which is responsible for containing device and system information. This software element is important because the storage of device information enables actions to be taken based on the past, instead of just the present. Additionally, the storage of system information is crucial to maintaining device pairings and security logins. Next, there is a database manager software module which is responsible for controlling reads, writes, and searches of the database. It provides an interaction API which will function as a communication protocol between it and its associated modules. Next, there is a translation software module which simply converts database manager API protocol information to and from the central hub’s internet communication protocol, this will likely be replaced by an AMQP broker. Finally, there is a web portal host software module which serves the web portal internet pages to client browsers. It is also capable of issuing commands to the database manager module.
3.3.2.5 User Interfaces
The user interfaces component, seen in Figure 4, is separated into two distinct software modules: a web portal client and a smart phone application. The web portal client describes any internet browser that is accessing the server’s web portal host module. The phone application functions similarly but interacts directly with the server’s database manager API protocol; it uses a GUI that has been optimized for smartphone screens. Both of these modules allow for users to interact with the home automation system.

3.3.3 Hardware
This home automation system’s hardware architecture will need to be determined and refined. Each hardware module present in Figure 4 will need an explicit physical structure and associated block diagram; however, this has not yet been completed. During prototype implementation, hardware system architecture documentation will be generated including: circuit schematics, block diagrams, and interface definitions.

3.3.4 Software
This home automation system’s software architecture will need to be determined and refined. Each software module present in Figure 4 will need an explicit logical structure and associated block diagram; however, this has not yet been completed. During prototype implementation, software system architecture documentation will be generated including: software flow charts, block diagrams, and interface definitions.

3.4 Design Alternatives
3.4.1 Central Hub OS and Networking
3.4.1.1 Criteria
The operating system for the central hub should support real time operations in order to provide seamless experiences to users. The operating system should also perform well with somewhat low system resources (e.g., 512 MB of RAM). In addition, it should allow optimal interfacing with other devices. This encourages compatibility and reliability within the design, which relates to the design norm of trust.

3.4.1.2 Options
3.4.1.2.1 UNIX variants – LiteOS & Debian
LiteOS and Debian provide a real time operating system with a Unix-like interface. LiteOS is developed by the Laboratory for Autonomous, Interconnected, and Embedded Systems at the University of Illinois. This alternative provides a familiar developing environment. But it was recently developed, and its newness to the RTOS field means that it only supports connecting to a few devices. Debian is optimized for use with Raspberry Pi processors and is has a wide developer support community.

3.4.1.2.2 Windows - Windows Embedded
Windows Embedded is an embedded operating system developed for terminal devices with low system configurations, such as small amount of memory. It allows real time operation and device connectivity support is also provided. Developers in the home automation industry typically avoid the windows platform because of its proprietary licenses status and other administrative difficulties.

3.4.1.2.3 Real Time Operating Systems – TinyOS
TinyOS is an RTOS written in nesC. It was developed jointly by the University of California, Intel Corporation, and Crossbow Technology. TinyOS provides fairly simple interface for developers, and has a rich support for many devices. This alternative is also well documented and well known in the RTOS field.
3.4.1.4 Decision Matrix

Table 5. Decision Matrix for Central Hub OS

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Unix Variants</th>
<th>Window Embedded</th>
<th>TinyOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability (RTOS)</td>
<td>38%</td>
<td>10</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>System Resource Requirements</td>
<td>31%</td>
<td>10</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Reliability (Device Connectivity)</td>
<td>31%</td>
<td>8</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>9.38</td>
<td>7.07</td>
<td>10</td>
</tr>
</tbody>
</table>

3.4.1.3 Choice

As depicted in Table 5, TinyOS appears to best satisfy the weighted criteria for a central hub OS. However, Unix variants were evaluated as similarly suitable and the chosen operating system must also be compatible with the central hub processor, alternatives for which are examined below. The Raspberry Pi processor choice influences the usage of Debian, a Unix variant, for the central hub OS.

3.4.2 Terminal Device to Central Hub Communication Protocol

3.4.2.1 Criteria

The communication protocol should have low latency times and low power consumption, in accordance with the design norm of stewardship. In addition, the protocol should include security measures in order to prevent unwanted manipulation of connected devices. The communication protocol should also be widely used in order to ensure reliability and continued support in the future.

3.4.2.2 Options

3.4.2.2.1 ZigBee

ZigBee is a commonly used communication protocol in the home automation industry. It provides low latency times and has low power consumption. The protocol itself has security architecture built into it. However, increasing numbers of developers in home automation industry are abandoning ZigBee due to its non-free, non-open source licensing status.

3.4.2.2.2 Z-Wave

Z-Wave is a widely used communication protocol in home automation industry. It has low latency times, low power consumption, and a decent security architecture. However, Z-wave chips are manufactured only by the company Sigma Designs. Some developers avoid supporting the Z-Wave chip protocol because of its monopolistic nature.

3.4.2.2.3 Bluetooth

Bluetooth is a widely used communication protocol in home automation industry. It is secure with low latency times and average power consumption. Bluetooth is widely used and supported by numerous developers and products.
3.4.2.4 Decision Matrix

Table 6. Decision Matrix for Terminal Device to Central Hub Communication Protocol

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>ZigBee</th>
<th>Z-Wave</th>
<th>Bluetooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability (Low Latency Time)</td>
<td>25%</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Stewardship (Low Power Consumption)</td>
<td>5%</td>
<td>10</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Trust (security)</td>
<td>19%</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Trust (Devoted Developers)</td>
<td>11%</td>
<td>5</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Range</td>
<td>40%</td>
<td>8</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>13.15</td>
<td>8.49</td>
<td>9.69</td>
</tr>
</tbody>
</table>

3.4.2.3 Design Alternative Selection

As depicted in Table 6, ZigBee appears to best satisfy the weighted criteria for a device to hub communication protocol. Zigbee will be used this wireless communication.

3.4.3 Central Hub Processing Hardware

3.4.3.1 Criteria

The central hub processor should support the OS alternative chosen above, this will dictate its platform architecture and other features. The hardware system should provide the ability to flash the board itself and have strong development tools. The processor should also align with the design norm of stewardship by having a low power consumption. Additionally, the hardware should be compact, and be readily available at a reasonable price.

3.4.3.2 Options

3.4.3.2.1 Arduino Uno

Arduino boards allow direct programming, but not the ability to flash the board. The Arduino developer GUI is well-developed. The operational voltage for Arduino Uno is 5 Volts\(^3\). The current draw varies based on the load attached to the board. The size of Arduino Uno is 75 mm x 53 mm\(^4\), and can be purchased easily at common electronic devices distributor with a price of $28.20\(^5\).

3.4.3.2.2 Raspberry Pi – Model B

Raspberry Pi boards can be flashed; however, they only support operating systems that have been optimized for use with their hardware. They have strong support and a widely used development environment. The Raspberry Pi is powered by 5V micro usb, and the current usage also varies based on the application. The Raspberry Pi measures 85.60mm x 56mm\(^6\), and is also readily available at Calvin’s engineering labs with no charge. It can also be purchased with $35\(^7\).

3.4.3.2.3 BeagleBone Black

Beagle boards can be flashed; however, they only support operating systems that have been optimized for use with their hardware. They have a fairly widely used development environment. The operational voltage is 5V, and the current drawn varies based on the load attached to the board. The BeagleBone Black board measures 86.36 mm x 53.34 mm\(^8\), and is available at a price of $55\(^9\).
### 3.4.3.4 Decision Matrix

Table 7. Decision Matrix for Hardware System for Central Hub

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Arduino Uno</th>
<th>Raspberry Pi Model B</th>
<th>BeagleBone Black</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>13%</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Reprogrammable</td>
<td>12%</td>
<td>8</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Cost</td>
<td>13%</td>
<td>10</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Development Environment Popularity</td>
<td>12%</td>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Appropriate OS Support</td>
<td>22%</td>
<td>5</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>15%</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Size</td>
<td>13%</td>
<td>10</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>8.66</td>
<td>9.24</td>
<td>8.46</td>
</tr>
</tbody>
</table>

### 3.4.3.3 Design Alternative Selection

As depicted in Table 7, the Raspberry Pi seems to best satisfy the weighted criteria for central hub processor type. Given this and their availability at no additional cost, a Raspberry Pi central hub processor will be used.

### 3.4.4 Central Hub Power Supply

#### 3.4.4.1 Criteria

The central hub should be portable, although it is designed for stationary use, in order to ensure that it can be placed in various areas. It should also be reliable in the event of a power outage, as suggested by the design norm of trust, and allow any battery operated device to continue communicating with the system. Additionally, and in accordance with the design norm of reliability, the central hub should not require any maintenance once installed.

#### 3.4.4.2 Options

##### 3.4.4.2.1 Battery Powered Only

A battery powered central hub will allow for portability if it becomes necessary to move the hub. Batteries would also give the hub a limited increase in reliability in the event of a power outage (i.e., the hub would continue to function as long as the batteries last). However, using solely batteries also significantly decreases reliability and increases maintenance needs because they must be checked frequently.

##### 3.4.4.2.2 Outlet Powered Only

A central hub powered only through an outlet would have low maintenance requirements. By allowing the hub to be plugged into the wall, this alternative provides power in the absence of a power outage. This is highly reliable, but not portable. Each time the hub would be moved, it would power down and cause system communication loss. This alternative also is not reliable in the event of a power outage.
3.4.4.2.3 Battery & Outlet Powered

A central hub that is both battery and outlet powered embodies the portability, and reliability of both alternatives described above. This alternative allows the hub to be easily relocated, and also makes the already highly reliable hub more trustworthy by reducing chances of communication loss in the event of a power outage. However, the appropriate interaction between two unique power systems involves increased design resource usage and monetary cost for the final prototype.

3.4.4.3 Decision Matrix

Table 8. Decision Matrix for Central Hub Power Supply

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Battery Powered Only</th>
<th>Outlet Powered Only</th>
<th>Battery &amp; Outlet Powered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>10%</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Reliability (Power Outage)</td>
<td>45%</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Reliability (Maintenance)</td>
<td>45%</td>
<td>2</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>6.4</td>
<td>4.5</td>
<td>10</td>
</tr>
</tbody>
</table>

3.4.4.4 Design Alternative Selection

As depicted in Table 8, using both a battery and outlet power supply in the central hub best satisfies the weighted criteria.

3.4.5 Server & Central Hub Integration

3.4.5.1 Criteria

Including the webserver as integrated into the central hub or as hosted off site, separate from the central hub, will not change the user interface of the system. Nonetheless, these design alternatives greatly affect the quality of the whole system and, indirectly, the user’s experience. These alternatives should be evaluated according to the criteria: reliability, latency, user support, stewardship of resources, and design simplicity.

The design solution should build trust by minimizing the probability of system failure, and by reducing both user involvement and the need for expert help in the event of a failure. The system should be highly reliable.

The system should also minimize the latency time between a user interface command and the target device’s response. This will improve user experience significantly.

Customer support should be considered during system design because it impacts the user’s experience and perception of product quality. The alternative selected should maximize the ability for customer support to handle incidents, such as a user needing help setting the system preferences or a third party device not interfacing well with commands.

The design shall embody stewardship through efficient hardware usage. This will result in relative cost savings, which can be passed on directly to customers. It will also result in decreased energy consumption, an environmentally friendly aspect.

Finally, the design’s complexity shall be minimized, especially if doing so impacts the stewardship of the design by reducing required human, material, and manufacturing resources.

3.4.5.2 Options

3.4.5.2.1 Unique Server & Central Hub

In this alternative the central hub would still mediate bidirectional communication between the devices and server through translating the information passed using different communication protocols.
But different from the alternative below, this one separates the database structure housing, database operation performance and the processing of commands from the central hub to a server. This alternative assumes that the server is not at the user’s house, but instead offsite where it is managed and maintained by the system designers.

This alternative has a high reliability in the event of a system failure. The system designers would be in charge of maintaining the server and would be more capable of doing so.

It is possible that this might have a high the latency for the response of commands sent by the user. This would be due to the location of the server being offsite from the user’s house. There would likely be a farther distance for commands to travel.

This design alternative gives the system designers great ability to support the users. If a user needs help with system preferences or third party device support the system designers would be in a great position to help because they are in control of the server.

Separating the server allows the system designers to effectively and efficiently manage the database structure’s size. This also impacts the user because they would not have a limit in devices or system preferences they may have.

Implementing a database structure with a server offsite will give greater flexibility in software choices and low complexity in hardware implementation.

3.4.5.2.2 Server Integrated Into Central Hub

This alternative integrates the server and its functions into the combined central hub. This would be accomplished by embedding a database structure in the central hub and giving the central hub the capability of processing commands and performing database operations.

Integrating the server into the central hub gives low reliability to the system if the server were to crash due from high activity. The user would be required to reset anything needing resetting. This calls for work or expertise out of the user and negatively affects the user’s experience.

With the server on central hub the latency of response time for the system would most likely be low, but a user in need of help might have difficulties in receiving user support.

For this alternative, the size of the database cannot be changed. If a user has a large variety of devices or system preferences, they will need a larger database structure to accommodate this. Thus, the design would need to plan for the worst case and give a large allocation of memory. This alternative would not allow for minimizing the server’s size and potentially wastes hardware. This would lead to an increase in the products price.

This alternative has a high complexity in the system design. Creating a database structure on the central hub might prove difficult and performing operations on an in house database structure might be challenging.
3.4.5.3 Decision Matrix

Table 9. Decision Matrix for Server & Central Hub Integration

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Unique Server &amp; Central Hub</th>
<th>Server Integrated Into Central Hub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>10%</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Latency</td>
<td>10%</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>User Support</td>
<td>40%</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Hardware Usage</td>
<td>20%</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>Complexity</td>
<td>20%</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>9.5</td>
<td>3.6</td>
</tr>
</tbody>
</table>

3.4.5.4 Design Alternative Selection

Table 9 clarifies the decision. Under the criteria listed, with the weights and scores given to each alternative, the design should implement a server separate from a central hub.

3.4.6 Webs Server Hosting

This design alternative assumes that the server is chosen to be separate from the central hub as explained above.

3.4.6.1 Criteria

The webservice should not require high levels of maintenance, be capable of supporting required program sizes, and be financially feasible given the project’s budget.

3.4.6.2 Options

3.4.6.2.1 Server Rented Online

A rented server from an online provider would probably not take much time to implement (there would be no need to set up the database structure) and would not be limited in size. But renting is typically expensive and most likely not possible given the operating budget of the project.

3.4.6.2.2 Calvin College Server

Using one of Calvin College’s servers would not take much time to implement, but there may be limitations on program size. This alternative would most likely be free; it would not require additional financial investment to use one of the college’s already existing servers.

3.4.6.3 Decision Matrix

Table 10. Decision Matrix for Server Hosting

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Server Rented Online</th>
<th>Calvin College Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to Maintain</td>
<td>10%</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Size</td>
<td>10%</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Cost</td>
<td>80%</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>2</td>
<td>9.7</td>
</tr>
</tbody>
</table>
3.4.6.4 Design Alternative Selection
As depicted in Table 10, using a Calvin College server to host the home automation system’s webserver seems to best meet the weighted criteria.

3.4.7 Database Structure
3.4.7.1 Criteria
The database structure must perform its duty well in order to maintain consumer trust. The three basic criteria for the database structure are reliability, database management efficiency, and database development efficiency. Reliability is an important consideration when the system will require multiple data access attempts simultaneously. Management efficiency directly affects how fast the HomeAlive system handles data storing and loading. However, comparing database structure performance times is complex. The final criterion involves project scope and consist of designer familiarity with the database’s programming language.

3.4.7.2 Options
3.4.7.2.1 SQL Server
Microsoft SQL server was first launched for commercial use and could only be used by machines running a Windows operating system. SQL Server is the first database structure that became widely known and has a long history of use beginning when it was released in 1989. This server-based database structure uses Microsoft relational database management system (RDBMS). Although implemented in C++, SQL supports the .Net, Java, PHP, Python, Ruby, C++, and Visual Basic programming languages [7].

3.4.7.2.2 MySQL
MySQL was developed by Oracle in 1995. It is a server-based, open-source database structure, and it is compatible with Linux, Windows, OS X, Solaris, and FreeBSD operating systems. MySQL uses client server and an open source RDBMS. Although implemented in C and C++, MySQL supports C, C#, C++, Java, PHP, Python, and other programming languages [8,14].

3.4.7.2.3 Oracle
The Oracle database structure was first published in the year 1980 for commercial use. Oracle uses RDBMS to manage its data storage, and it is compatible with Linux, Windows, OS X, Solaris, HP-UX, AIX, and z/OS operating systems. Although implemented in C and C++, it supports C, C#, C++, Python, PHP, Ruby, Visual Basic, and other programming languages [9].

3.4.7.3 Discussion
Microsoft’s SQL Server has a significant downside if HomeAlive uses an operating system besides Linux, OS X, or Solaris for its webserver. In terms of reliability, all of the database structures discussed are widely used and famous for their accuracy; however, the Oracle database has the longest tradition of reliable widespread use (30 or more years). When considering development efficiency, Oracle and MySQL support the languages C and C++ which are familiar to system designers [1].

3.4.7.4 Decision Matrix

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>SQL</th>
<th>MySQL</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>70%</td>
<td>9</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Easy to Manage</td>
<td>30%</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>8.4</td>
<td>8.4</td>
<td>9.8</td>
</tr>
</tbody>
</table>

Table 11. Decision Matrix for Database Structure
3.4.7.5 Design Alternative Selection

As depicted in Table 11, the Oracle database structure seems to best meet the weighted criteria. Selecting it as the database structure is advised. However, SQL and MySQL were evaluated to be suitable options if any problems arise with Oracle.

3.4.8 Web Application Framework

3.4.8.1 Criteria

The Web Application Framework (WAF) is software that will be used to design the web portal and other web services associated with this system. When choosing a WAF, there are three key criteria: protocol compatibility, development efficiency, and unique feature sets. Protocol compatibility is important because the web framework should be able to effectively communicate with the remainder of the HomeAlive system. Developer efficiency is important due to project scope considerations and is usually based on designer familiarity with the supported programming language. Routing and templates are also issues common to web framework evaluation [10].

3.4.8.2 Options

3.4.8.2.1 Ruby

Ruby on Rails (RoR) is an open source WAF built in the widely used programming language Ruby, uses an ActiveRecord model-view-controller (MVC), which allows it to represent its data in various ways. RoR manages MVC using push techniques, and its object-relational mapping (ORM) data conversion tool is also ActiveRecord. RoR adds inheritance and associations features; it also incorporates testing, database migration, security assurances, template generation, caching, and form validation frameworks. This alternatives security framework is plug-in [11, 12].

3.4.8.2.2 Python

Django is a commonly used, open-source WAF built in the programming language Python. Django uses a full stack type MVC and has its own integrated ORM. Like most WAF models, Django supports testing, database migration, security assurances, template generation, caching, and form validation. Django’s security framework uses access control list (ACL), and its template and form validation frameworks use Python. The caching framework that Django uses is a dynamic one [10].

3.4.8.2.3 Java

There are several WAF models built in the Java programming language, but the most widely used one is Spring. The Spring WAF is one of the most commonly used open source frameworks. Its MVC is a push-typed one, but it is not highly unique. The ORM that Spring uses is called Hibernate; it solves ORM impedance mismatch problems. Like most Java-based WAF models, Spring does not include a database migration framework. Spring’s security framework is unique, like its unit-testing test framework. JavaServer Pages (JSP), software that dynamically generates web pages, is used as its template framework. Spring’s caching framework employs the Ehcache technique, and this validation framework utilizes Bean Validation strategies [13,14].

3.4.8.3 Discussion

The WAF model associated with the Java programming language is the most familiar to system designers, giving it the highest expected development efficiency. Compatibility considerations must be examined in tandem with webserver choices. The Spring and Django WAF models include several unique features which adds to both their flexibility and complexity.
3.4.8.4 Decision Matrix

Table 12. Decision Matrix for Web Application Framework

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
<th>Ruby</th>
<th>Python</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Familiarity</td>
<td>60%</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Compatibility</td>
<td>30%</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Uniqueness</td>
<td>10%</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>7.4</td>
<td>8.1</td>
<td>8.8</td>
</tr>
</tbody>
</table>

3.4.8.5 Design Alternative Selection

As depicted in Table 12, the WAF model that seems to best meet the weighted criteria is Java’s Spring WAF. However, each alternative evaluated as similarly suitable and decision matrix estimates require refinement before a WAF is chosen.

3.4.9 Central Hub to Webserver Messaging Protocol

In order for the central hub to properly function it must consolidate wireless communication from many devices into a single internet-based communication path. It includes two critical components: a wireless communication module and an internet-enabled processor. The communication protocol for messages passed between the hub and the webserver will be chosen from several alternatives as discussed below.

3.4.9.1 Criteria

Each messaging protocol should be evaluated according to its reliability, computational requirements, compatibility, and complexity. Reliability refers to the protocol’s assurances that its messages are transmitted successfully, decoded accurately, and received from a trusted source. Computational requirements refers to the idea that minimal hardware and software resources should be needed in order to utilize this protocol; its primary purpose is to transmit information, not process it. Compatibility refers to the fact that the protocol must be capable of interacting with both the wireless communication module and the hub’s main processor. Finally, complexity refers to choosing the simplest available alternative that satisfies the preceding criteria satisfactorily.

3.4.9.2 Options

3.4.9.2.1 MQTT

One protocol option for internet-based messaging is MQ Telemetry Transport (MQTT). It was designed for machine-to-machine (M2M) integration and mobile applications by focusing on the minimization of required computational resources and reliable assurances of message delivery [15]. It utilizes established connections between publishers and subscribers in order to perform message passing. This is examined in greater detail in Appendix 8.2.

3.4.9.2.2 AMQP

Advanced Message Queuing Protocol (AMQP) allows messaging providers to send information to a message-oriented middleware (MOM) broker by implementing an open standard application layer. MOM supports messaging distribution over similar platforms. AMQP is utilized for M2M communication, and it is similar to the familiar Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP). It utilizes established connections between clients and brokers in order to perform message passing. This is examined in greater detail in Appendix 8.3.

3.4.9.3 Design Alternative Selection

The internet-based messaging protocol for communication between the central hub and webserver has been tentatively determined as AMQP. This is based on the fact that the Raspberry Pi processor has
the required complexity to support an AMQP application, which offers greater capabilities than an MQTT one.

3.4.10 Web Portal Programming Language

In order to provide a web portal user interface the most ideally satisfies the criteria below, a programming language for designing the software interface must be chosen. Several alternative languages are identified and evaluated in this section.

3.4.10.1 Criteria

Each web portal programming language should be evaluated according to its compatibility, graphical capability, runtime performance, and developer familiarity. Compatibility refers the fact that the web portal must be able to interact with the database system and interface protocols that are chosen when designing the webservice. Graphical capability refers to ease with which the programming language can be used to develop intuitive and aesthetically appealing user interfaces. Runtime performance refers to the efficiency and speed with which programs written in the language typically run. Finally, developer familiarity refers to selecting the language that is best known by the web portal developer and also satisfies the preceding criteria.

3.4.10.2 Options

3.4.10.2.1 Ruby on Rails

One programming language option for the web portal user interface is Ruby on Rails. Its chief advantage seems to be an integrated web framework that facilitates the development of website user interfaces. It will be evaluated in greater detail when the topic of web portal design increases in priority for this project.

3.4.10.2.2 Additional Options

Additional alternatives for the web portal programming language will be identified and evaluated when this topic increases in project priority.

3.4.10.3 Design Alternative Selection

A protocol for web portal programming language will be decided upon based on carefully evaluated alternatives when this topic increases in project priority.

3.4.11 User Interface

3.4.11.1 Criteria

The user interface is the point where the product and consumer interact, so it plays a large role in the user’s experience and should seek to maximize the quality of that experience. This project is being conducted with limited resources including time and cost. In light of this, the user interface alternative that is selected should maximize quality of experience while remaining within project budget and scope.

3.4.11.2 Options

3.4.11.2.1 Web Portal Only

The web portal would use project resources efficiently by building upon an already required web framework. Developing a web portal interface that is intuitive for users may present a challenge, and the user experience for a web portal is not seamlessly available in daily life.

3.4.11.2.2 Mobile Application Only

A mobile phone application user interface would require development of an Android or an IOS application using Java. Developing a mobile application may cost money to gain a license to use an environment provided by the related OS. Designing a GUI of this nature would be time consuming, due to team inexperience with mobile development. However, this alternative could use project resources efficiently by utilizing connectivity capabilities in the server’s web framework. A mobile application
would be a great asset for maximizing the user’s experience; it would serve as an important part of the system’s overall accessibility.

3.4.11.2.3 Web Portal & Mobile Application
Developing both the web portal and mobile application interfaces, in comparison to only doing one, would increase the time required and possibly the cost of this project, but would significantly increase the user’s quality of experience.

3.4.11.3 Decision Matrix

**Table 13. Decision Matrix for User Interface**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality of User’s Experience</td>
<td>20%</td>
<td>2</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Cost</td>
<td>5%</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Time Required</td>
<td>75%</td>
<td>9</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Totals:</td>
<td>100%</td>
<td>7.65</td>
<td>5.45</td>
<td>2.65</td>
</tr>
</tbody>
</table>

3.4.11.4 Design Alternative Selection
As depicted in Table 13, it seems that developing only a web portal user interface for the system best meets the weighted criteria. However, it is important to note the criteria weights favor scope considerations far more than the quality of user’s experience and cost criterion. Because of this, a mobile application user interface will be developed if the project is ahead of schedule and time permits it.

3.5 Design Alternative Selection Summary
Table 14 summarizes all of the design alternative selections. Some selections are not determined yet because more research is need to make the decision.

**Table 14. Design Alternative Selection Summary**

<table>
<thead>
<tr>
<th>Design Alternative</th>
<th>Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Hub OS and Networking</td>
<td>Debian</td>
</tr>
<tr>
<td>Terminal Device to Central Hub Communication Protocol</td>
<td>ZigBee</td>
</tr>
<tr>
<td>Central Hub Processing Hardware</td>
<td>Raspberry Pi</td>
</tr>
<tr>
<td>Central Hub Power Supply</td>
<td>Battery &amp; Outlet Powered</td>
</tr>
<tr>
<td>Server &amp; Central Hub Integration</td>
<td>Unique Server &amp; Central Hub</td>
</tr>
<tr>
<td>Webserver Hosting</td>
<td>Calvin College Server</td>
</tr>
<tr>
<td>Database Structure</td>
<td>Oracle (or SQL)</td>
</tr>
<tr>
<td>Web Application Framework</td>
<td>Java</td>
</tr>
<tr>
<td>Internet-based Messaging Protocol</td>
<td>AMQP</td>
</tr>
<tr>
<td>Web Portal Programming Language</td>
<td>Ruby on Rails*</td>
</tr>
<tr>
<td>User Interface</td>
<td>Web Portal Only</td>
</tr>
</tbody>
</table>

*Indicates the alternative currently has only one option and needs research to determine more options.

3.6 Testing Definitions
Tests for this system will be used in order to ensure proper functionality of the system and its components throughout the development process and in order to evaluate the final prototype. These tests can be categorized as hardware tests, software tests, interface tests, user tests, or some
combination, and each category can be further divided into multiple types. Two specific tests that have been established are the end-to-end communication test and the weekend-loop test. Additionally, a ‘fail fast’ methodology will be used whenever possible.

A ‘fail fast’ methodology is based on the notion that the earlier in the design process a failure is, the more capable designers are of remedying the issue. This is because less resources have been invested, more time remains before deadlines, and less consecutive design decisions have been made that depend on the failing component. The methodology is acted upon by scheduling frequent, small-scale tests as early as possible during the design of the system. Even a basic proof of concept test can be invaluable when evaluating feasibility and making design decisions.

Hardware tests focus on confirming the appropriate functionality of the system’s physical enclosures and electrical circuitry. Physical enclosure tests include basic stress analysis and simulated drop-impact evaluation. Electrical circuitry tests can be divided into the types: functional, temperature independence, noise resistance, and stability. Functional tests will ensure that circuit drives appropriate voltage and current levels in response to known input signals; each circuit should always pass these tests. Temperature independence tests will ensure that the circuitry continues to pass functional tests when exposed to various ambient conditions; each circuit should pass these tests within reasonable ranges of ambient temperatures. Noise resistance tests will ensure that circuitry continues to pass functional tests when exposed to electromagnetic noise. This will primarily be a check of wireless communication acknowledgements or checksums. Stability tests will ensure that the circuitry continues to pass functional tests when powered for long periods of time, when power is toggled rapidly, and when powered for the first time after a long period of inactivity. The passing of these tests will likely be determined by electronic component temperature, max voltage, and max current ratings.

Software tests focus on confirming the appropriate functionality of the system’s program implementation and algorithmic design. Program implementation tests will consist primarily of software unit tests; these are tests that verify the correctness of each software function by calling it with known inputs and expected outputs. Program implementation tests should cover typical cases and all edge cases. They are used to identify programming mistakes. Algorithmic design tests vary from similar to program implementation tests to similar to user tests, and are meant to validate the logical structure of the program under test. The ones that are similar to program implementation tests use a set of known inputs and expected outputs to confirm the correctness of a segment of code larger than a single function call. As these code segments increase in size, algorithmic design tests become focused on increasingly high-level inputs and outputs. The highest level inputs and outputs are similar to what a user experiences in a user test.

Interface tests focus on confirming the appropriate functionality of the system’s inter-component communications. These tests are crucial because points of component interaction tend to be both the hardest to develop correctly and the most likely to fail over time. Interface tests include both hardware and software tests, as described above. Hardware interface tests are performed on circuitry that connects multiple components by transmitting signals between them. One key aspect they check is the matching between the maximum output signal amplitude and the maximum input rating of the targeted component. Software interface tests are performed to ensure that different programs and platforms are communicating digitally as intended. This is often done by sending a target component a known signal and expecting a certain forthcoming response.

User testing has two main aspects, unexpected use-cases and the user experience. User tests typically involve taking a third-party individual or sample of individuals with limited knowledge of the product, and allowing them to interact freely with the system or component under test. This is typically most helpful when the sample users are taken from the target demographic population. User tests can reveal unexpected use-cases for the system. New, unknowledgeable individuals are likely to attempt to interact with the system in ways that system designers did not foresee. Additionally, the user experience...
can be evaluated using user tests. By watching sample users interact with the system, and by surveying or interviewing them afterwards, it is possible to gain valuable insights into the development of the product.

One specific, holistic test that will be used to evaluate this system is the end-to-end-communication test. This test consists of simply sending a recognizable signal, such as “hello world” or a pattern of bit voltage levels, from device hardware through the communication modules, hub, and server to the phone application. This test should also be performed in the reverse direction. If successful, the end-to-end-communication test indicates that all relevant parts of the system can feasibly be used as intended in the system. This test should be done as early as possible in keeping with the ‘fail fast’ methodology described above.

Another specific tests that will be used to evaluate components in this system is the weekend-loop test. It is primarily an interface test, and consists of monitoring the communication between two modules over a period of time. For example, the central hub should be able to send an integer value to a device and the device should be capable of responding with that same value. Upon receiving the matched acknowledgement, the hub will increment the integer and send it again. If this continues successfully over the course of a weekend, it is likely capable of running stably for much longer.

4 Business

4.1 Mission for the Company

HomeAlive’s mission is to provide to homeowners accessible control of any electronic device, smart or non-smart, through a convenient, elegant, and intuitive remote-interface.

4.2 Vision for the Company

HomeAlive’s vision is to make its way into the home automation market and claim a significant portion of market share. Upon meeting this goal, HomeAlive will seek to expand its product base, which will further HomeAlive’s presence in the market. HomeAlive seeks to become one of the top competitors in the home automation market within ten years.

4.3 Potential Customers & Target Market

HomeAlive targets both new and existing homeowners as potential customers. The specific target market focuses on homeowners who have disposable income, familiarity with electronic devices, or margin to gain from increased accessibility to devices.

4.4 Competitive Strategy

HomeAlive seeks to compete based on differentiation. Competing based on cost would not be feasible due to the presence of large-company competitors already in the market. Competing based on response would not be as applicable to this product as differentiation for the reasons explained below.

4.4.1 Differentiation

Customers will want to buy home automation product based on up-to-date technology, ease of access, and reliability. HomeAlive provides ease of access by creating intuitive and elegant interfaces. Other key differentiation that HomeAlive focuses especially on customer experience and environmental sustainability of HomeAlive products.

4.4.2 Response

HomeAlive’s system design has similar scale compared to other home automation products. This fact leads to a need for better control and focus. HomeAlive brings more flexibility to customers who have prior home technology devices than other products do. HomeAlive offers embedded technology that connects all home devices with the central hub, which helps with increasing ease of access and the apparent quickness of HomeAlive’s products.
4.5 Market Sizes & Trends
HomeAlive is targeted towards both new and previous homeowners. Several smart home products already exist, such as Google’s Nest, Philip’s Hue light bulbs, Apple’s HomeKit, and others. However, there is a lack of seamless system integration that would allow the connection of all devices. The home automation market has been changing in order to address that gap. Lowe’s Iris is an example of such an attempt; HomeAlive’s product will be another.

The smart home market was valued at $33 billion in 2013, and is expected to boom to $71 billion by 2018 according to Juniper Research. The recent market entrants such as Apple (HomeKit), Google (Nest), and BestBuy (Peq) have strengthened the market; it is clear that the smart home market is increasingly valuable.

4.6 Business Strategy & Distribution
HomeAlive will distribute products in two ways. First, the product will be available via internet sellers such as Amazon and eBay, and will be shipped to customers using their services. Second, the product will be distributed through a mutually beneficial partnership with a strategic partner. The current trend is that most electronics related companies are attempting to develop a smart home system. For example, GE has its own Wink smart home system and BestBuy has Peq. However, there are a few retailers and department stores who have interest in the smart home market, but are not actively developing a competing system. HomeAlive will partner with such a strategic partner in order to increase sales and accessibility.

4.7 SWOT Analysis
4.7.1 Strengths
HomeAlive is built by a passionate and motivated team, which is reflected in the product. The home automation market has a large potential for new product growth and HomeAlive will use this to gain market traction throughout development and support of new devices. Large company bureaucracy will not slow the team down; instead, necessary design modifications will be made quickly and effectively.

4.7.2 Weaknesses
HomeAlive is not a recognized name and its executives are not experienced in the market. In the early stages of production, due to lack of name recognition and a small customer base, the cost for HomeAlive will be more expensive. This will be true until demand reaches high enough levels to use economies of scale to lower production costs.

4.7.3 External Opportunities
The market that HomeAlive focuses on is relatively new and broad. Rapid technology improvements help the current generation to become familiar when handling technology. This trend is an excellent opportunity for HomeAlive to emerge into the market. New homeowners with higher disposable incomes and who have gadgets are the main target. The uniqueness of HomeAlive’s products will attract additional customers.

4.7.4 External Threats
The competition with the established brand products offered by competitors is a strong threat to HomeAlive. They are well funded and have more resources in their research and development branches. Another threat is the lack of differentiation between the home automation products currently on the market. Customers who have home automation devices might not be interested in buying HomeAlive’s products. In order to maintain market share, HomeAlive needs to build customer loyalty and focus on having a flawless product.
4.8 Uniqueness & Existing/Potential Competitors

The home automation market has multiple established large-company competitors; each of whom offer similar device support and features. To complete in this market, HomeAlive will need to differentiate its features from those competitors. Table 15 and Table 16 show existing competitors their product features.

Table 15. Existing Competitors in the Market with Devices

<table>
<thead>
<tr>
<th>Devices</th>
<th>Iris</th>
<th>Connect</th>
<th>Wink</th>
<th>Peq</th>
<th>Smart things</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Hubs</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Lights Control</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Contact Sensors</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Motion Sensors</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Water Sensor</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Door Open/Close Sensor</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Keypad</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Outlet Adapter</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Power Strip</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Thermostat</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Range Extender</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Locks</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Smoke Detector</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Camera</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>3rd Party Support</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table 16. Features Comparison among Commercially Available Home Automation Systems

<table>
<thead>
<tr>
<th>Features</th>
<th>Iris</th>
<th>Connect</th>
<th>Wink</th>
<th>Peq</th>
<th>Smart things</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Control</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Notification</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Video Camera</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Customizing Rule Set</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Voice Control</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

5 Conclusion

5.1 Potential Risks & Issues

The biggest risk for this project is underestimating project scope. According to research and mentor guidance, the original scope of the project was much too ambitious. However, scope reductions consisting of: fewer devices, fewer features, and more purchased components should render this project feasible. The removed devices and features will be reinserted into the project design as time and resources allow.

Another potential risk is the budget. Financial estimates are naturally inexact, and significant unexpected costs could cripple this project’s feasibility.

Unit and component testing also poses a potential risk to this project. The proposed system design is highly focused on network communication, so creating tests for components isolated from the
remainder of the system is challenging. However, doing only holistic system testing is not feasible, components and software modules will need to be tested independently as well.

A final risk inherent in any project, is uncontrollable delays of progress. These could include situations like waiting for parts to arrive or scheduling dependency bottlenecks. A more thorough estimation of unexpected situations is not within the scope of this PPFS document.

5.2 Project Status
As of December 6, 2014, the team has performed preliminary research into different design alternatives such as communication protocols, messaging protocols, processors, and web frameworks. A website has been created to make important project documents and information publically available. The team has also met with two different industrial mentors, receiving helpful feedback on project scope, scheduling, and technical design decisions. Additionally, two Calvin College honors projects, in conjunction with a computer architecture class, have helped further project progress. These include connecting an outlet adapter to the central hub and developing a thermostat control device. Furthermore, two minor lab projects have been used to begin work on the central hub. These include simulating MQTT and AMQP messages between the central hub and server. Finally, two oral presentations and the PPFS document, significant deliverables, have been completed. The team is currently meeting their schedule, but schedule estimates are difficult to make due to frequent scope changes (usually reductions).

Appendices 8.2, 8.3, 8.4, and 8.5 offer details on the MQTT lab project, AQMP lab project, thermostat honors project, and outlet adapter honors project respectively.

5.3 Project Feasibility
The feasibility of the project is evaluated using the factors: scope, schedule adherence, task completion capability, and budgetary restrictions.

Recent mentor feedback regarding the scope of the project has facilitated a significant narrowing of proposed system functionality. After the elimination of devices and features from the prototype design, the project is indeed feasible in terms of scope. It is also important to note that eliminated features will be re-inserted into the project as time allows.

The ability of team members to consistently adhere to the project schedule is crucial to the feasibility of this project. So far the team has done well in this area by completing tasks on time and communicating any unexpected delays. Next semester, spring 2015, the team members will have lighter academic schedules and heavier project schedules. The balance of these changes should allow the team to continue adhering to their schedule; the project is indeed feasible in terms of schedule adherence.

The capability of team members to accomplish design tasks is important to the feasibility of the project. The team’s technical knowledge and available research resources, as well as mentor guidance, indicate that any design task can be completed. However, the ability to complete a task does not always reflect an ability to complete that task within an acceptable amount of time. Therefore, tasks should be assigned to individuals based on their strengths, and this aspect of feasibility coincides with scope size feasibility.

The operational budget explained in section 2.4 of this report shows an estimated cost of $298, using a contingency factor of 150%. Considering that this value is lower than the expected project funding, the project appears to be financially feasible.

5.4 Summary
This project, after its scope reduction, seems to be feasible according to its scope, schedule adherence, task completion capability, and budgetary restrictions. There are some significant risks
associated with each aspect of feasibility; however, these can be mitigated with careful planning that leaves room for unexpected challenges.

6 Acknowledgements

There are several organizations and individuals that influenced this project and deserve both credit and thanks. First, the Calvin College engineering department for its administrative and financial contributions to the project. The department provided primary funding, quality workspaces, and necessary equipment for the project; it also organized project showcase and set the final prototype due date. Second, SpinDance Inc. for generously providing David van Geest as an industrial mentor. Also, David himself for offering guidance on several technical issues and helping to clarify appropriate project scope. Next, Eric Walstra for serving as an industrial consultant and providing assistance with technical issues and project risk assessment. Additionally, Professor Mark Michmerhuizen for his consistent feedback on schedule management and project deliverables. Then, Nathan Terschak for his contribution to preliminary design simulations. Finally, Professors Brouwer, Kim, VanAntwerp, Wunder, and Nielsen for their varied guidance. Without these organizations and individuals, this project would not have been possible.
7 References & Bibliography


8 Appendices

8.1 Category Workload Graph

8.2 MQTT Simulation

8.2.1 Objectives

The goal of the lab was to achieve bidirectional communication between a Raspberry Pi, and a device using the MQTT messaging protocol, and to learn more about the performance of MQTT.

8.2.2 Platform Chosen

An MQTT broker (server) resides on the Raspberry Pi board. A Raspberry Pi was chosen since it runs a Linux-variant OS (Debian), and had extensive networking capabilities. In addition, the Raspberry Pi was compatible with I/O devices such as a keyboard and mouse, this was essential for timely project completion.

A HomeAlive system device was simulated using an MQTT client on a desktop PC; a Windows PC was chosen because of the availability of MQTT client programs for that platform.

8.2.3 Lessons Learned

The lessons learned during this lab involved Raspberry Pi configuration and the MQTT protocol. They included the fact that a Raspberry Pi (B model) can run a full Linux operating system with 512 MB of RAM, two USB ports, and a 100 Mbits/s Ethernet port. They also included the fact that this processor is produced by ARM, has a frequency of 700 MHz, and is equipped with a VideoCore IV GPU. Lessons regarding MQTT included its publish-subscribe messaging techniques and the grouping of message topics. An MQTT client can either publish or subscribe to a topic in order to send or receive messages. The role of MQTT brokers (servers) are to route messages between clients as indicated by their subscriptions. For example, client A publishes “Hello World” message under “testing” topic. The
broker will broadcast that message, and it will direct it to any other clients who are subscribing to that topic. Any clients that subscribe to the “testing” topic would receive the “Hello World” message.

Figure 6. MQTT Messaging Implementation

8.2.4 Resources
For the project, a readily available Raspberry Pi and its default Debian OS were used. The open-source Mosquito program was used to set up the MQTT server on the Raspberry Pi. Then, the MQTT-fx program was used as a Windows PC client application. Another resource that was used is the C library for writing MQTT server/client programs.

8.2.5 Customizations
After booting the Raspberry Pi’s Linux OS successfully, the Mosquitto package was installed and an MQTT server was configured. Next, research on the MQTT architecture and supported server/client programming languages was completed.

8.2.6 Demonstration
Ultimately, the MQTT broker was able to communicate with multiple devices bidirectional – from broker to client and from client to broker. In order to test this, different messages under varying topics were published from several PC clients. On another PC, a client that subscribed to all these topics was checked for proper message receipts on every topic.

8.3 AMQP Simulation
8.3.1 Objective
The goal of the lab was to achieve internet-based bidirectional communication between two distinct computers using AMQP as the messaging protocol. This was done in order to simulate communication between the central hub and server components of the HomeAlive system, described above.
8.3.2 Platform Chosen

In accordance with the processor choices for the central hub above, and because of price and availability considerations for this lab, a Raspberry-Pi computer was used with a Debian OS.

For server simulations, an Ubuntu OS and Asus laptop were chosen because due to availability and accessibility. Ubuntu is a freely available Linux OS that supports the required AMQP libraries. Additionally, both the Raspberry Pi and laptop required stable internet connections to the same network.

8.3.3 Lesson Learned

Lessons learned during this lab involved the AMQP messaging protocol’s performance and installation. They included the fact that AMQP uses a connection-based communication in order to establish a channel for continuous, bidirectional communication. As seen in Figure 7, AMQP requires a broker, service, and client in order to run. The lessons also included examinations of several different message broker programs. JBoss, Maven, and RabbitMQ on a Windows PC were all attempted as broker programs; however, they were not successfully installed due to varying installation and configuration complexities. Finally, RabbitMQ on a Linux OS was utilized in order to successfully configure a broker program.

Figure 7. RabbitMQ Messaging Implementation

8.3.4 Resources

Numerous tutorials that were used for each message broker attempted install, especially to setup RabbitMQ (https://www.rabbitmq.com/configure.html & https://www.rabbitmq.com/install-debian.html). Additionally, RabbitMQ provides standard node module examples such as send and receive node (https://github.com/squaremo/amqp.node/tree/master/examples/tutorials). The send node publishes data and the client obtains the data by using the receive node. Furthermore, the WiringPi scripts for GPIO control were used in order to implement button pressing capability.

8.3.5 Customization & Future Work

Customizations of the downloaded open-source resources explained above included: broker configuration (adding users and setting permissions), client and server configuration (identifying appropriate IP addresses), and the adjustment of messaging content. Messaging content was altered to show a timestamp upon sending and receiving the message, effectively calculating the time required for message passing. Furthermore, Raspberry Pi GPIO control scripts were used in order to facilitate the choice of different messages. Two hardware buttons were used with an infinitely looping script in order to do this.
In the future, several changes will need to be made in order to achieve a connection suitable for the purposes of central hub and server communication. These changes include allowing the client to publish a conditional message to a remote database.

8.3.6 Demonstration
Two types of communication paths were successfully demonstrated. The first path used the Raspberry Pi as both the source and destination of a message passed through the laptop’s broker. The second path used the Raspberry Pi as a source and the laptop as a destination for message passing through the laptop’s broker. This shows that the client can send messages to itself and other clients through the broker. Additionally, the GPIO script and hardware buttons described above were used as basic conditional inputs for a multiple message content demonstration.

8.4 Thermostat Device
This appendix describes the details of the thermostat device, which represents a controllable household machine. It is intended to allow for the remote control of the thermostat through a GUI. As of 12/6/14, this section contains information on a preliminary implementation of the thermostat device including relevant diagrams and schematics. For a full version of the design including the procedure and results see <link to website when content available>.

8.4.1 Objectives & Requirements
It is not an objective for HomeAlive to design a thermostat device; this may be a later project of the proposed company. The objective for this device is merely to prove a concept (i.e., a thermostat can be controlled by HomeAlive’s system). The implications of this must be made clear: any limitations of the user interface in controlling of the thermostat, due to not being its designer, should not translate as failing to meet any requirements. (e.g., the temperature, set point of the thermostat, and other signals are not readable because they are encoded in a complex digital pulse wave signal only know to the designer. Also, the program in the thermostat’s on-chip user interface will be simulated in the GUI provided by HomeAlive, but any failures in perfectly simulating that program are not a failure to meet any requirements).

There is one simple overarching requirement for this implementation, the ability to remotely control a thermostat using a Windows PC GUI. Another requirement was to choose a digital thermostat that had multiple controllable features. Eventually, the device must incorporate the wireless communication protocol chosen above, but this is not a requirement for this implementation. Lastly, the thermostat adaptations should not inhibit the original design (i.e., not allowing for a mechanical button to be pressed).

8.4.2 Architecture
8.4.2.1 System Integration
The thermostat device implementation fits into the larger HomeAlive system according to the diagram seen in Figure 8. This implementation includes a preliminary device and communication with a simulated central hub. It includes modified purchased components (red with green stripes) and simulated units (red with blue stripes).
8.4.2.2 Device Architecture

8.4.2.2.1 Thermostat Button Interface

The thermostat has six buttons. Each button has a positive and negative terminal (0-3V) which are shorted upon the pressing of the button. See Figure 9 for the mechanical button interface, Figure 10 for the electrical circuit interface, and Figure 11 for the preliminary GUI (the GUI for the broader HomeAlive system has not yet been created).
Figure 9. Thermostat Mechanical Interface

Figure 10. Thermostat Circuit
8.4.2.2.2 Analog Circuit Layout

The buttons are remotely pressed using BJTs. Upon receiving the button press signal from the GUI, the Arduino sends out 5V to the base of a BJT. This shorts the emitter and collector of the BJT and actuates a button press. The circuit schematic can be seen in Figure 12.

![Figure 12. Thermostat Analog Circuit]

8.4.2.2.3 Arduino Program

An Arduino Uno is being used to receive the simulated ZigBee communication from the central hub and actuate the button presses. The following program, listed below as Arduino code, is used to control the Arduino. The pin outs are found in Figure 12 above.

```
#include <Arduino.h>

// Pin Numbers
int Up = 13;
int Hold = 12;
int Down = 11;
int Left = 10;
int Center = 9;
int Right = 8;
```
int incomingByte = 0;

// This runs when the reset button is pressed
void setup(){
  // initialize the ports
  pinMode(Up, OUTPUT);
  pinMode(Hold, OUTPUT);
  pinMode(Down, OUTPUT);
  pinMode(Left, OUTPUT);
  pinMode(Center, OUTPUT);
  pinMode(Right, OUTPUT);

  Serial.begin(9600); // Baud rate
}

// Infinite looping program
void loop(){
  if (Serial.available() > 0){
    incomingByte = Serial.read(); // Read the command from the GUI
    Serial.println(incomingByte); // This is for troubleshooting
    if (incomingByte == 48) {
      digitalWrite(Up, HIGH);
      delay(500);
      digitalWrite(Up, LOW);
    }
    if (incomingByte == 49) {
      digitalWrite(Hold, HIGH);
      delay(500);
      digitalWrite(Hold, LOW);
    }
    if (incomingByte == 50) {
      digitalWrite(Down, HIGH);
      delay(500);
      digitalWrite(Down, LOW);
    }
    if (incomingByte == 51) {
      digitalWrite(Left, HIGH);
      delay(500);
      digitalWrite(Left, LOW);
    }
    if (incomingByte == 52) {
      digitalWrite(Center, HIGH);
      delay(500);
      digitalWrite(Center, LOW);
    }
    if (incomingByte == 53) {
      digitalWrite(Right, HIGH);
      delay(500);
      digitalWrite(Right, LOW);
    }
  }
}

8.4.3 Future Work
The next step for this device is achieving the requirement to use ZigBee for communication with the central hub. In addition, the analog circuit, which is currently on a breadboard, will need to be constructed on a PCB. Lastly, a GUI simulating the thermostat’s mechanical interface will need to be developed and incorporated into the HomeAlive system GUI.

8.5 Outlet Adapter Device
This appendix describes the details of the outlet adapter device, which represents a controllable household machine. It is intended to allow increased power consumption monitoring and automated power toggling in order to support household energy use reductions. As of 12/6/14, this section contains information on a preliminary implementation of the outlet adapter device including relevant
diagrams and schematics. For a full version of the design including step-by-step assembly procedures see [link to website].

8.5.1 Objectives & Requirements
The outlet adapter device is capable of monitoring the current, voltage, and wattage for any household machine using it. It can be inserted into a standard American wall outlet and provides an identical female socket for machine use. Furthermore, the adapter reports its power monitoring information wirelessly to the system’s central hub and will be able to receive toggling commands from the hub. Toggling commands allow the remote activation of the outlet.

8.5.2 Source
This device utilizes a commercial power monitoring unit called a Kill-a-Watt and XBee wireless communication modules. These resources were modified and combined according to the DIY how-to guide found at [www.adafruit.com](http://www.adafruit.com), project name Tweet-a-Watt. Note that some portions of the guide were adjusted for this application.

8.5.3 Architecture
8.5.3.1 System Integration
The outlet adapter device implementation fits into the larger HomeAlive system according to the diagram seen in Figure 13. The implementation includes a preliminary device and wireless communication with a simulated central hub. It includes modified purchased components (red with green stripes) and simulated units (red with blue stripes). The details of each component are described below.
8.5.3.2 Device Architecture

The outlet adapter implementation includes the components shown in Figure 13 and discussed above; however, these components are combined into just three functional elements. First, the Kill-a-Watt power monitoring element includes the interface hardware component. Second, the XBee communication network element includes the RF communication and Zigbee protocol components. Finally, the central hub simulation element includes the processor and networking software components.

The outlet adapter implementation can also be separated according to physical divisions into the receiver, transmitter, network configuration, and scripting software. These relate as seen in Figure 14 and are discussed in detail below.
8.5.3.3 Receiver

The receiver consists of an XBee module, an XBee serial adapter, and a USB connection. The XBee module is responsible for wireless communication with the device transmitter and uses the serial adapter in order to perform wired communication with the simulated central hub. When included in the final central hub, this receiver will be shared among all device transmitters, not just the outlet adapter’s transmitter. It can be seen in Figure 15 fully assembled.
8.5.3.4 Transmitter

The transmitter consists of an XBee module, Kill-a-Watt unit, modified XBee serial adapter, and power regulation circuitry. The XBee module is responsible for wireless communication with the central hub receiver, and the Kill-a-Watt unit is responsible for monitoring the voltage, amperage, and wattage of its associated standard American outlet. The modified XBee serial adapter uses resistors and diodes in order to passively read voltage and amperage measurements from the main Kill-a-Watt microchip using hardware. It is energized by the power regulation circuitry which includes several capacitors and a voltage regulator. The power regulation circuitry ensures that the XBee module receives the appropriate DC voltage consistently despite variations in the Kill-a-Watt power output, its power supply. The transmitter can be seen in Figure 16 and the modified serial adapter schematic in Figure 17.
8.5.3.5 XBee Configuration

The XBee modules used in this outlet adapter implementation were configured to use a point-to-point Zigbee network topology. The network includes only two nodes, the receiver and transmitter, and communication between them is currently unidirectional. Default configuration modifications included PAN identification numbers, sleep cycle durations, and analog-to-digital converter (ADC) settings. The ADCs are integrated into the XBee module hardware. Configuration changes were performed using Digi’s X-CTU program.

8.5.3.6 Python Scripts

Code listings are available online at [insert link here when content uploaded].

The Python scripts used by this implementation of the outlet adapter device simulate code that will eventually be run on the central hub. They retrieve packets of serial information from the receiver via wired USB communication and parse those packets for power monitoring information. The information is then reported every few seconds (depending on XBee configurations) and will eventually be submitted by the central hub to the system server. The main script, titled watcher.py, runs a loop which repeatedly polls for new serial packets. It relies on several Python libraries including the third-party pyserial library and the project-provided sensor history and XBee libraries. A diagram of the main script’s looping logic can be seen in Figure 18.
Future Improvements

Additional improvements will need to be made in order to achieve all the outlet adapter device goals. These include bidirectional communication, network topology adjustments, central hub integration, and calibration. Bidirectional communication refers to the ability to receive power toggling signals in addition to the ability to send power monitoring data. Network topology adjustments will need to be made in order to handle multiple devices communicating with the same receiver. The central hub simulation computer running the scripts for this implementation will need to be integrated into the actual central hub. Finally, calibration of the software will be needed in order to ensure that accurate power monitoring information is calculated.
1 Donated by Calvin Engineering Department
2 Provided by Calvin Information Technology Department
3 http://arduino.cc/en/Main/ArduinoBoardUno
6 http://www.raspberrypi.org/help/faqs/