Smart Hub MDR Report

Patrick Lowry, EE, Aman Sardana, EE, Sidney Saycocie, EE, and Chris Mitchell, EE

***Abstract*—This report describes the world of smart home appliances and our solution to reducing their apparent complexity: Smart Hub. Smart Hub is a low-cost, high-functionality alternative to pre-existing devices that integrate technology around a home. From radio frequency communication to infrared, Smart Hub is capable of syncing with multiple home appliances such as air conditioners, window shades and smart outlets. A Raspberry Pi 2 is what gives it its processing power, while Atmel ATtiny85 microcontrollers control the peripheral devices. This report also discusses the project's progress thus far and what needs to be accomplished to reach project completion.**

I. Introduction

The past decade has seen a drastic rise in home appliance complexity. While this is widely an effort to integrate smart technology in our everyday lives, one cannot deny that there is a learning curve that must be overcome in order to utilize say, a modern day central air system or a smart television. Arguments can be made in favor of this complexity by saying that consumers who are not the most well-versed with technology are now compelled to change that and learn more about the technology that serves them well. Steve Cousins, a manager at the Palo Alto Research Center (PARC) makes a similar point:

"People are getting more proficient [with technology] because they have to, not because they want to" [1].

However, the above statement also proves that there is some reluctance when it comes to people welcoming technology into their homes. Most modern smart devices that serve nontrivial purposes—such as home heating, refrigeration, etc.—involve a setup process that is the biggest factor to turn people away from the investment. The Nest thermostat for example, is a slick and efficient device to suit one's heating or cooling needs. It comes with a setup process that aims to be simple but might discourage many customers who are not technologically literate or prefer not to be so involved with their technology [2]. There are many electronic products on the market that are remarkably easy to setup and use within a unique ecosystem. However, it is very complicated and often impossible for these devices to communicate with each other without having to use multiple, incompatible applications and hardware configurations.

People with physical disabilities are not accounted for either. While there are accessibility options on all modern day computing devices, home appliances are left out with no 'ease of access' solutions other than perhaps a bulky, often complicated remote controller.

The inspiration for Smart Hub arose from these very issues. We want to create a device that not only makes home appliances smarter, but also makes them simpler and more intuitive to use. Speech is one of the most instinctive human attributes and so we decided to incorporate voice control to send commands to various home appliances. We believe that this could have significant societal impact in terms of accessibility.

We also want a low-cost product to compete with the more expensive alternatives that are on the market. We primarily made comparisons with Amazon's Echo and found that Smart Hub can deliver similar functionality for cheaper. An in-depth comparison between Smart Hub and its competitors can be seen in Fig.8 in the Appendix. Additionally, we want to make Smart Hub compatible with pre-existing smart peripherals (Belkin WeMo, Philips Hue) to better utilize current retail devices.

In essence, we want this to be a customer's product; one that is cost effective and user friendly. This entails developing a lightweight device with a minimal setup time and an intuitive interface.

II. Design

A. Overview

The Smart Hub ecosystem can be seen in Fig. 1 on the following page. At the core of the system will be the Smart Hub itself, powered by a Raspberry Pi 2 [3]. It will be used as a base station to control all of the peripherals around the home. Among these will be a window air conditioning unit (IR), a temperature probe (RF), an automatic window shade (RF), and a Belkin WeMo wireless switch (Wi-Fi). The Smart Hub will broadcast and receive RF signals to most of the peripherals using an RF transmitter and receiver. It will control Wi-Fi connected smart devices through the built-in wireless router. Lastly, it will be able to control any IR device by recording and sending IR its commands. A summary of system specifications is shown in Table I.

In order to make the Hub simple to use, we will incorporate an open source voice recognition system called Jasper. Jasper will be capable of recognizing certain spoken voice commands

TABLE I
Smart Hub System Specifications

Specification	Value
RF Control	2 Tx/Rx channels (315 and 433 MHz)
Speakers	2.0
Operating Voltage	12 V
Operating Amperage	6 A (max)
RF Range	> 20m
Total Cost	<$150

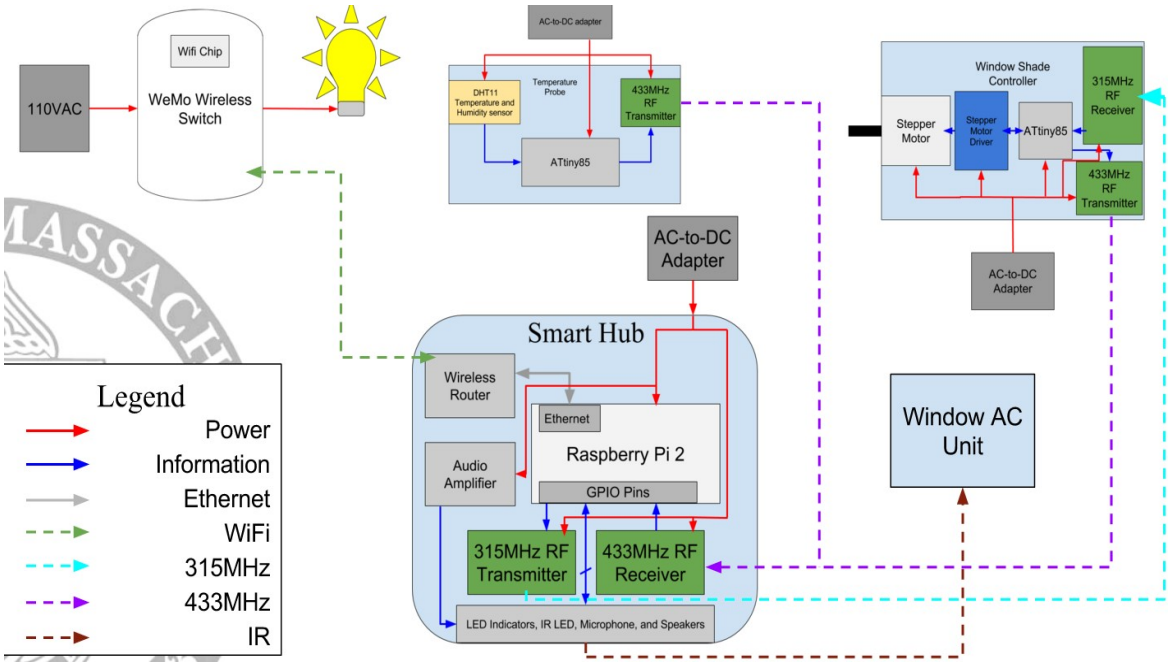

Fig. 1. Smart Hub system block diagram illustrating the Smart Hub and its wireless device ecosystem. The Smart Hub is at the center with an AC unit to the right, a window shade controller to the upper right, a temperature probe directly above, and a Belkin WeMo switch to the top left.

and communicating with the user's home appliances accordingly. A USB microphone will be used to collect voice input for this purpose.

There will also be a web companion to the hub, through the means of a web server installed on the Raspberry Pi 2. This web server will serve up a website containing a GUI through which the user can control all of their appliances from a computer (in home or remotely). For security, every Smart Hub will have unique login details (a username and password) when connecting via the website to prevent unauthorized control of devices.

B. Smart Hub

At the center of the whole system is the Hub itself. The Hub acts as the brain and central nervous system of the user's network of smart devices. It will be an easy way for the user to interact with and control the connected devices throughout their home. The Smart Hub connects to the user's home network using a built-in N150 wireless router [5], which will connect directly to a cable modem or fiber-optic equivalent (Fig. 2). All that's left is for the user to plug the Hub into a power outlet and it's ready to use (Fig. 2).

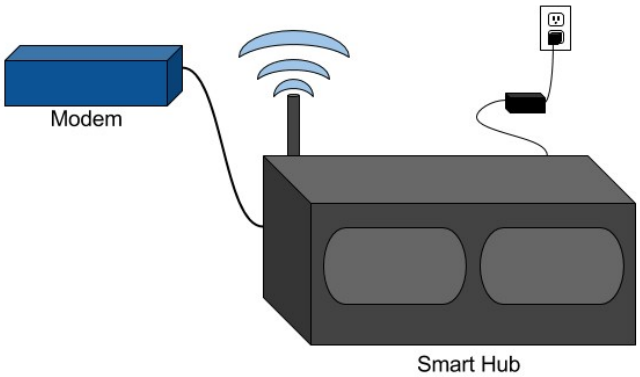

Fig. 2. Simple diagram of Smart Hub setup. The smart hub only needs a power source and an internet source.

The Hub uses the Pi to run the extensive code and communicate with the user's smart peripherals. On the Pi will be a host of software including a web server and voice recognition API. The web server will allow the user to conveniently control their devices from anywhere, on any internet connected device, using their unique login credentials. While they're home, the user can control their devices by talking to Jasper, an open source voice recognition API. The Pi will output Jasper's responses to the system's speakers and it will take input from a USB connected microphone.

The Raspberry Pi 2 is an extremely versatile tool in an inexpensive package. However, due to the low price, the audio quality is subpar. Because the audio quality is of the utmost importance for providing a rich user experience with Jasper, the Hub includes a built-in 2.0 speaker system with a stereo audio amplifier and volume control.

This block, shown in Fig. 3 on the next page, relies heavily on Circuits I and II as well as Electronics I and II. With three vital devices packed into one compact case, it's important to be able to supply the correct voltages and currents while maintaining circuit isolation to prevent shorts and electromagnetic interference. The Hub is fed 12V at a maximum of 6A from an AC/DC adapter with a barrel plug connection. From there, the voltage is stepped down using DC-to-DC buck converters with a maximum rating of 3A each. The first converter is set to 5V, which is used to power the wireless router and the Raspberry Pi 2 (as well as smaller, less power hungry circuitry). The other converter is set to 8.7V in order to power the audio amplifier.

The Smart Hub will connect to wireless devices using three separate methods. The first is over the Wi-Fi connection provided by the built-in wireless router. This will be used to connect with and control retail devices such as those from the Belkin WeMo and Philips Hue product lines (we will be using a Belkin WeMo switch). The Hub will also be able to control non-Wi-Fi connected devices that can be controlled with IR. Using IR LED's and an IR receiver, the Hub will be able to record IR commands from a device's included IR remote (we will be using a windowed air conditioner) and then transmit them. For the third method we will use RF transmitters and receivers to create and control our own wireless peripherals. The Hub itself will house a 315 MHz transmitter and a 433 MHz receiver in order to reduce congestion on the bands. The transmitter and receiver will be connected to the GPIO (general purpose input output) pins on the Pi 2 and will transmit commands or receive the current states of the devices. Using Manchester encoding, the transmitters will be able to send binary codes over RF to relay information and commands.

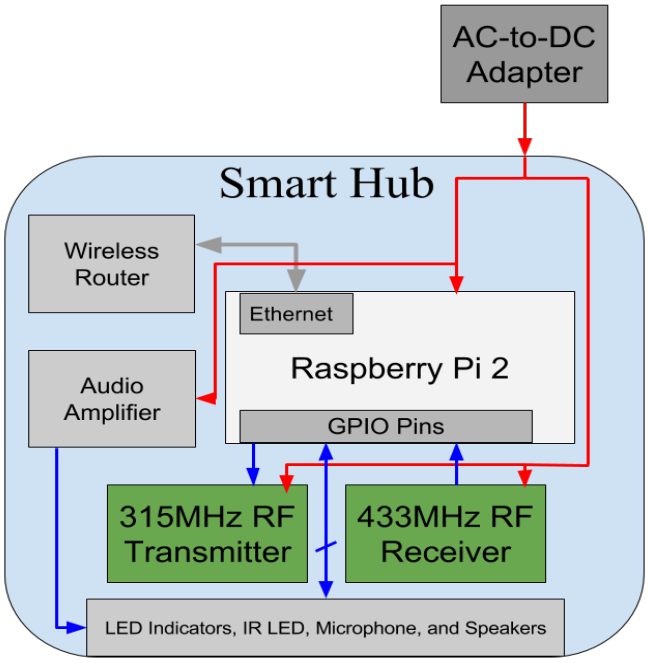

Fig. 3. Block diagram of the Smart Hub. The Hub has a Raspberry Pi 2, a wireless router, a stereo audio amplifier with speakers, and RF communication chips all built-in.

The two important topics for the Hub that will need to be studied and learned are RF communication and audio noise filtering. The RF will likely be the most complicated portion of the Smart Hub base station. This is largely due to the complexity of writing, filtering, and communicating with the Manchester code. The audio circuit is currently very susceptible to noise interference from the rest of the circuitry. It will be important to magnetically shield the speakers to reduce electromagnetic interference with the other components (especially the RF devices).

Because there are so many different devices with different power requirements in the enclosure, it was very important to find an AC/DC adapter that could supply enough current with little to no extra noise. The initial adapter chosen was rated for 5A, however it wasn't able to power the wireless router on its own (0.7A demand). Using 9, 1W 200Ω resistors in parallel, I was able to ensure that the issue was with the adapter and not the router. With these results it was determined that a higher quality adapter would need to be used. The adapter has since been changed to a 12V 6A supply that can fully power all of the internal devices of the Hub with 2-3A to spare (for higher current due to increased demand from the router and Pi 2).

C. *Window Shade Controller*

The window shade controller is made up of a stepper motor, a stepper motor driver, an ATtiny85 microcontroller [6], a 433MHz RF transmitter [7], and a 315MHz RF receiver [8]. The stepper motor is a RioRand motor that has 125oz.in of torque and 200steps/revolution [9]. This motor will be connected to the shade by a gear on the motor and one attached to the shade spindle. The motor will be driven by a basic stepper motor driver that will be supplied with between 15-20V and 2-3A. The driver is rated for 6-30V but anything above 22-25V range has a chance to cause a short in the board that could damage its main IC. The driver will be controlled by the ATtiny85 running Arduino IDE. It will receive commands from the Hub through a 315MHz RF receiver which it will then turn into signals for the driver to control the motor in the correct direction and number of steps with a 5V High/0V Low or 0-5V rising edge digital signal respectively. The ATtiny85 will keep track of the position of the shade (whether it is up or down) and will relay this to the Hub through the 433MHz RF transmitter. The ATtiny85 as well as the RF transmitter and receiver will all be powered with 5V that the EasyDriver will supply. A block diagram of the system is shown in Fig. 4

D. *Temperature Probe*

One of the peripheral devices we are building to interact with the Smart Hub is a temperature probe. The purpose of this temperature probe is to monitor indoor temperature and

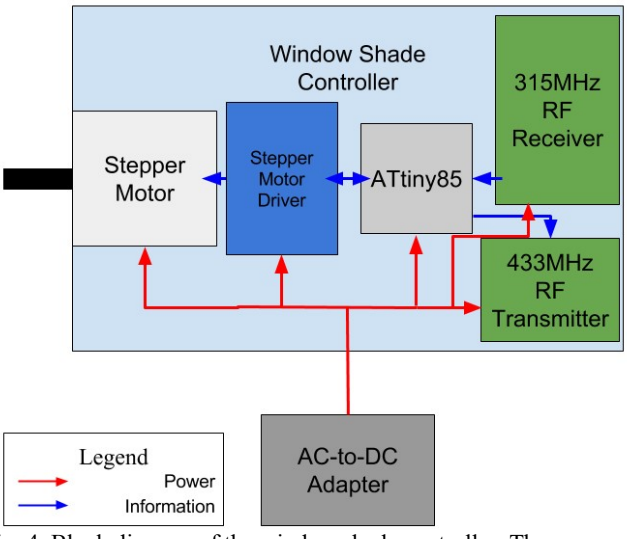

Fig. 4. Block diagram of the window shade controller. The controller uses an ATtiny85 to process commands from the Smart Hub and to control a stepper motor driver powering a stepper motor. The ATtiny85 receives commands via 315MHz RF and transmits via 433MHz.

humidity. This data will be displayed on the web server or it can be reported from the Smart Hub's speakers by prompting Jasper. The final version of the temperature probe will have 3 parts: a temperature/humidity sensor, a microcontroller, and an RF transmitter. The block diagram for the probe is shown in Fig. 5.We want a sensor that is small, cheap, and accurate. The DHT11 temperature and humidity sensor meets these three constraints [10]. It costs less than $2, it is accurate for temperature readings between 0°C and 50°C, and it is accurate for humidity readings between 20-80% [10]. It is accurate enough for indoor readings, which is all we need because outdoor temperature and humidity data can be obtained from the internet by the Smart Hub. The DHT11 uses a capacitive

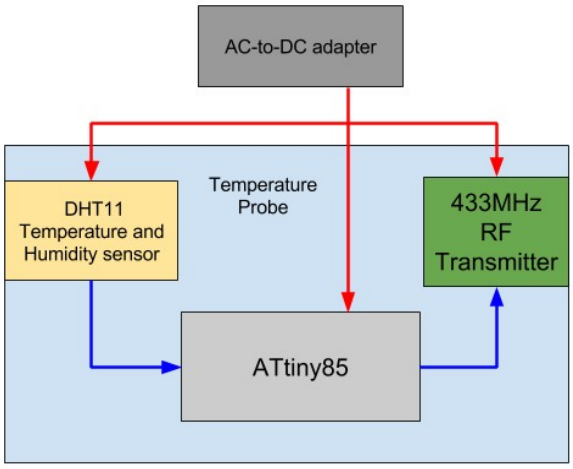

Fig. 5. Block diagram of temperature probe. The probe is controlled by an ATtiny85, which processes temperature and humidity values from the DHT11 sensor and transmits it to the Smart Hub via a 433MHz transmitter.

humidity sensor and a thermistor to measure the surrounding air, producing a digital signal on the data pin [10]. The digital signal will be read by a microcontroller and then it will be sent to the Smart Hub wirelessly through an RF transmitter. The microcontroller that we chose for the temperature probe was Atmel's ATtiny85. Again, we wanted a component that was small, cheap, and simple enough to be able to read and send data. The ATtiny85 is also less than $2, has 8KB of flash memory, 6 general purpose input/output pins, and an 8-bit CPU [6]. Since we are just using this microcontroller to read and send two numerical values, there is no need for a bigger, more complex microcontroller. The ATtiny85 runs by using the Arduino programming language and the code is loaded onto the ATtiny85 by using an in-system programmer (ISP). The DHT11 sensor and ATtiny85 microcontroller were tested to see if they worked together by using an Arduino Uno. The Arduino was used so that we could visually see the output of the ATtiny85 through its serial communication port. Using the Arduino IDE's serial monitor we were able prove that the DHT11 and the ATtiny85 successfully work as a pair by displaying the humidity and temperature values on the computer screen. We are going to implement a 433MHz RF transmitter to the probe so that we can send the temperature and humidity values wirelessly to the 433MHz RF receiver in the Smart Hub. The Raspberry Pi 2 in the Smart Hub will then be able to interpret the data and upload it to the web server, while Jasper can recite this information when prompted to. The same can be done with outdoor temperature by using the Raspberry Pi 2 to gather local weather information from the internet. Using both the indoor and outdoor readings, the Smart Hub can use this information to regulate an air conditioning unit near the Smart Hub by using IR to send commands.

III. Project Management

According to our evaluators, the MDR deliverables we initially promised—working subsystems and partial integration—were too ambitious given the complexity of the project. After receiving the feedback from the professors, we therefore adjusted our MDR deliverables. The new goal for MDR was to focus on getting each block of the block diagram working on its own. Table II on the next page displays our MDR deliverables. For MDR we designed and 3D-printed an enclosure and installed major components into it, including wireless router, Raspberry Pi 2, audio amplifier circuit and speakers. We designed, assembled, and tested the temperature and humidity sensor that will be placed in the user's home, we created a Pi-based web server along with HTML and CSS formatted PHP web pages for the user to control future connected devices, and we designed and tested a circuit configuration (stepper motor and driver) for the automatic window shade to be implemented for CDR.

There is a significant amount of work remaining before CDR arrives, as illustrated in the Gantt chart shown in Fig. 7. The Smart Hub must be completed, with all internal components properly assembled and functioning. This entails adding the RF circuitry (315MHz transmitter and 433MHz receiver), adding a fan and controller circuit to allow the internal devices to remain at an acceptable temperature, and adding an (ideally) internal microphone to the case before printing the top cover. For the window shade peripheral, the control circuit needs to be finalized so that the shade can temporarily be controlled by buttons or a timer, the frame needs to be made to hold the shade as well as the stepper motor, and the RF transmitter and receiver need to be added so that it can interact with the Hub. For the temperature probe, the RF transmitter needs to be added to the circuit (and the code modified for transmission), and the code for the Pi to control the air conditioner unit based on the indoor temperature needs to be written and tested. For the web server, the functionality needs to be improved and other functions need to be added so that the site is ready when the peripherals are completed, and the design needs to be improved with content that explains the project and gives info about each team member. The control code for the Pi needs to be written and added: this includes Jasper and the software to control the WeMo switch. Finally, all of these subsystems need to be integrated sufficiently by CDR so that the remainder of time before FDR can be spent troubleshooting and fine-tuning.

The first stage of the project largely required compartmentalizing responsibilities and working individually on the various components in preparation for full system integration. The team manager arranged weekly meetings with our advisor to review our progress and plan further development of each subsystem. We also met periodically to improve our designs, solve issues with the in-progress designs, or discuss how the peripherals will interact once finished. The team manager included all team members in email correspondence with our advisor and evaluators to ensure that each team member was informed of any project developments, part orders, or other updates.

Because we are all Electrical Engineering students, the heavy amounts of code will be challenging. However, our collective programming experience from various courses, including Computer Systems and Data Structures, will be particularly useful. We have experience with Java, C, Python, JavaScript, and PHP. More importantly, we have experience integrating hardware with computer languages. This skill will be vital for integrating the subsystems into the larger system.

In addition to our team meetings and the heretofore mentioned email correspondence, our team typically communicates either before or after a shared class or over the internet using a shared drive and instant messaging services. The Gantt chart for the tasks leading up to MDR are shown in Fig. 6 below; tasks leading up to CDR are shown in Fig. 7.

TABLE II
MDR DELIVERABLES

Deliverable	Completed
Smart Hub Enclosure	95%
Smart Hub Internal Assembly	85%
Apache Web Server on RPi2	100%
Smart Hub Website	30%
Temperature Probe (w/o RF)	100%
Window Shade Controller (w/o RF)	20%
Window Shade Mount	0%

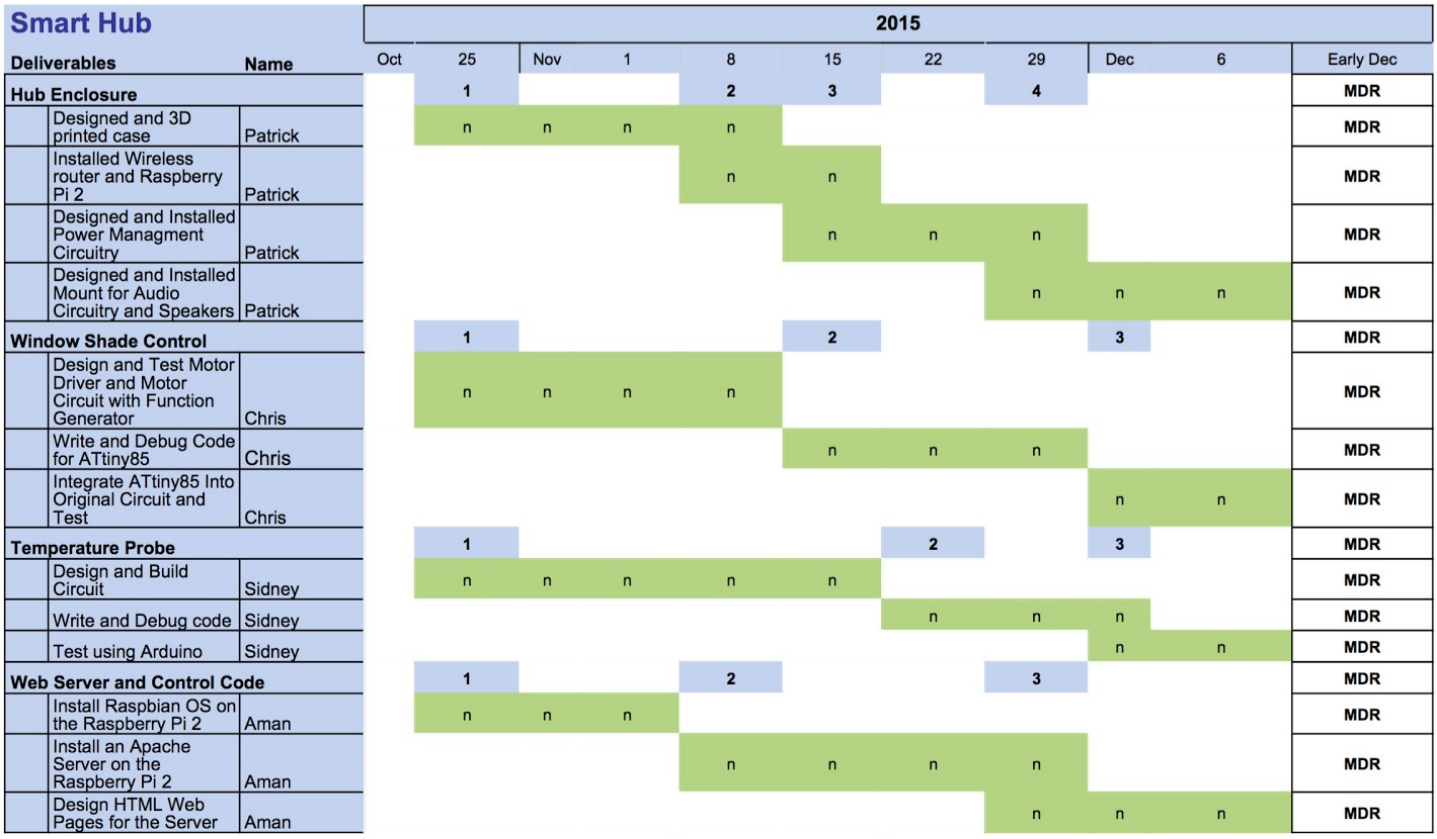

Fig. 6. Gantt chart showing responsibilities up until MDR with start and end dates denoted by sequences of green "n" blocks.

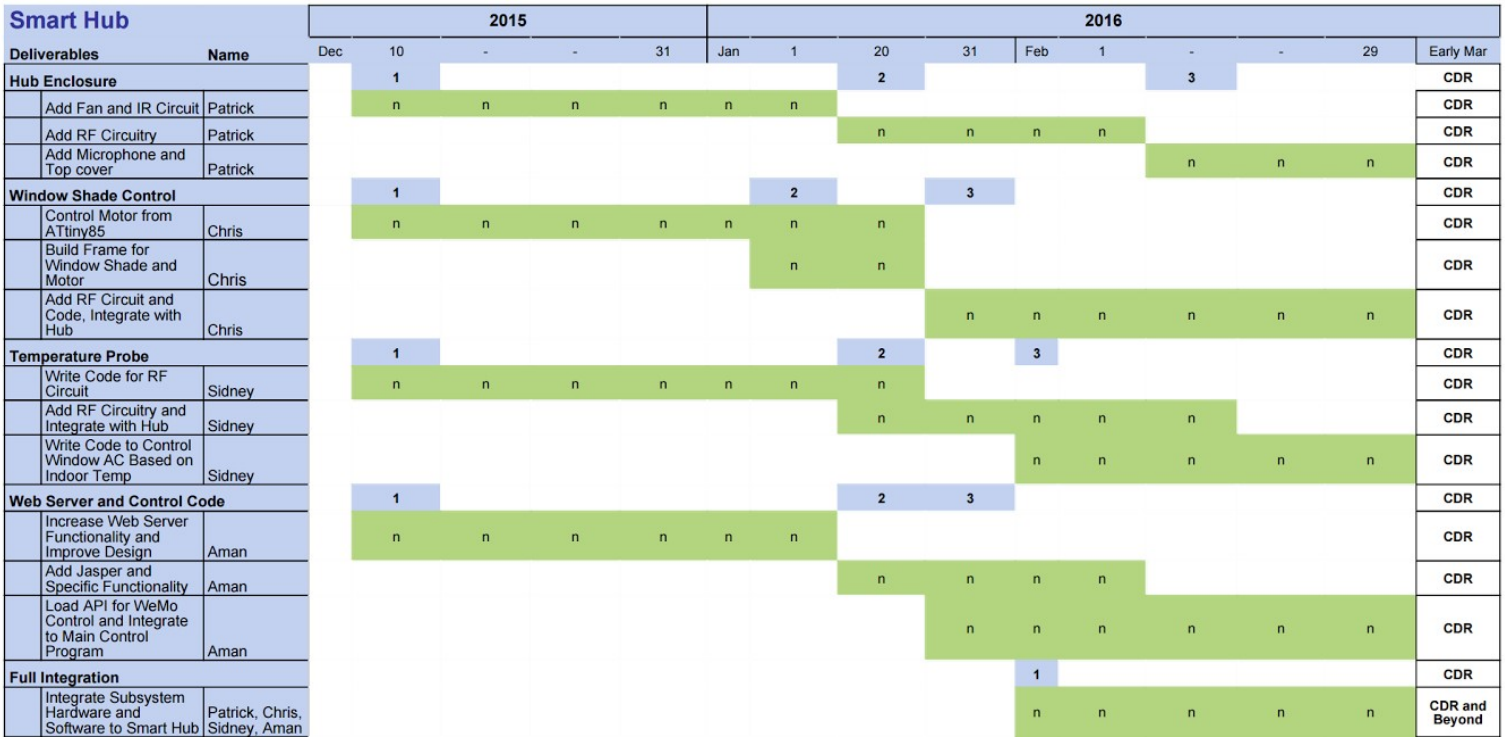

Fig. 7. Proposed Gantt chart showing responsibilities up until CDR in early March with start and end dates denoted by sequences of green "n" blocks.

IV. Conclusion

The Smart Hub project is progressing steadily with the Hub being mostly assembled and two of the peripherals being completed, with the exception of the RF components. These peripherals are the window shade controller and the temperature and humidity sensor. The web server is running on the Raspberry Pi 2, which is installed in the Smart Hub enclosure. In the coming months, the RF will be incorporated into the two peripherals as well as the Hub. The Hub itself will also be able to receive and send IR commands. Code will also be written to allow the Hub to control the shade, an AC unit, and the WeMo plug. The web server will receive a lot of work allowing it to control the Hub and the peripherals. Jasper will be integrated into the system as well and will be given the same access. The hardest part of this process will be ensuring that all of the different parts have uninterrupted and clean communication whether via RF, Wi-Fi, or IR.

APPENDIX

	Smart Hub	Amazon Echo	Belkin WeMo	Philips Hue
Companion App	✓	✓	✓	✓
Internet Connectivity (Wifi) w/ Remote Access	✓	Only Certain Services Through Alexa App	✓	✓
Light Control	✓	✓	✓	✓
Outlet Control	✓	✓	✓	
Voice Control w/ Integrated Audio	✓	✓		
Connected App Capabilities (Social Media, Weather, News)	✓	✓		
RF Control	✓			
IR Control	✓			
Wireless Router	Integrated	Separate	Separate	Separate
Price	$118.18 - 139.18 (plus peripherals)	$179.99 (plus peripherals)	>$39.99 - 169.98	>$199.99

Fig. 8. In-Depth Smart Hub competition comparison.

REFERENCES

[1] K. Hafner, "Techies by Necessity, Not by Choice," The New York Times, July, 2003. Available: http://www.nytimes.com/2003/07/24/technology/circuits/24boot.html?pagewanted=all

[2] Nest Labs, "Installing the Nest Learning Thermostat," Web, Sep., 2015. Available: https://www.youtube.com/watch?v=dHKD-9uI24I

[3] Raspberry Pi Foundation (2016). *Raspberry Pi 2 Model B*. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[4] Jasper Project (2016). *Control Anything with Your Voice.* Available: https://jasperproject.github.io/

[5] Netgear (2016). *G54/N150 Wireless Router.* Available: http://www.netgear.com/home/products/networking/wifi-routers/WNR1000.aspx#tab-techspecs

[6] Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash, 1st ed. San Jose: Atmel, 2015, pp. 2-5.

[7] Amazon.com (2016). *SMAKN 433Mhz Rf Transmitter and Receiver Link Kit for Arduino/Arm/McU.* Available: http://www.amazon.com/SMAKN%C2%AE-433Mhz-Transmitter-Receiver-Arduino/dp/B00M2CUALS/ref=sr_1_1?ie=UTF8&qid=1453753998&sr=8-1&keywords=433+mhz+transmitter

[8] Amazon.com (2016). *SMAKN 315Mhz Rf Transmitter and Receiver Link Kit for Arduino/Arm/McU.* Available: http://www.amazon.com/SMAKN%C2%AE-315Mhz-Transmitter-Receiver-Arduino/dp/B00UWBJ7GK/ref=sr_1_1?ie=UTF8&qid=1453754132&sr=8-1&keywords=315+mhz+transmitter

[9] Amazon.com (2016). *RioRand Stepper Motor – 125 oz.in (200 steps/rev)-JK57HS51-2804.* Availabe: http://www.amazon.com/RioRand-Stepper-4-Phase-5-Wire-ULN2003/dp/B00JHS2AH2/ref=sr_1_1?ie=UTF8&qid=1453754226&sr=8-1&keywords=riorand+stepper+motor

[10] DHT11 Humidity & Temperature Sensor, 1st ed. London: D-Robotics, 2015, pp. 2-3.