USB Controlled Appliances

ECE 445 Final Report

Nagarjun Kumar
Peter Jin
TA: Dean Biskup
Team 15

Table of Contents

1 Introduction

1.1 Problem and Solution Overview

We have been introducing a series of “smart” devices that will be controlled mostly or entirely by a computer, without reliance on wireless technologies. The rationale behind this is that wireless IoT devices, in many cases, tend to have an inherent security and privacy risk, because the attack surface of a home network is quite high. For example, any device on the network could potentially conduct a spoofing attack to spoof the router, which would allow any website viewed from any device on the network to be intercepted [4]; in ECE 422 class, we demonstrated how to perform an ARP spoofing attack to hijack connections to websites [4]. Bluetooth is even worse, due to potential attacks during pairing. For example, someone could put another Bluetooth device close in range with the same name as an intended device, and this could lead to sensitive data being sent to the wrong device [4]. So instead of relying on wireless technologies, our solution only ever uses wires for communication using much simpler protocols like USB and UART [8].

Three types of appliances will be demonstrated with this idea in mind: an optocoupler switcher, a thermostat, and a motion detector. These three types of appliances were chosen to cover a good variety of potential IoT appliances in ways that don’t duplicate each other in terms of their general functionality.

2 Outline of Subject Matter

2.1 Introduction

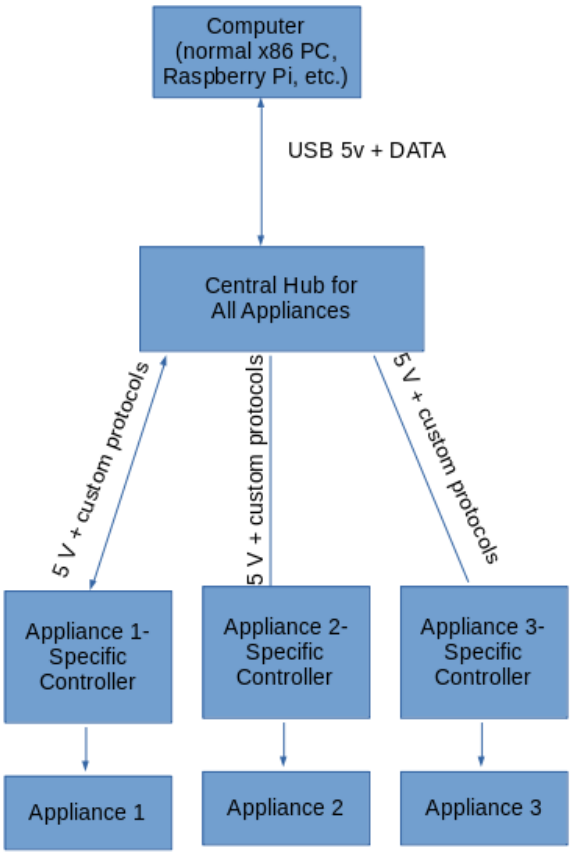

Figure 1: General block diagram of project.

We have created three different smart appliances to demonstrate our functionality, a optocoupler switcher, thermostat, and motion detector. These three appliances are split amongst two PCBs, the optocoupler switcher and motion detector on one and the thermostat on the other. These PCBs send UART protocol to the central hub via the GPIO headers present on both PCBs as well as on the central hub. The central hub will then convert these UART protocols to USB protocols via the 3 USB to UART adapters present on the central hub. The project has been divided into 3 subsystems; the thermostat subsystem, the optocoupler switcher/motion detector subsystem, and lastly the central hub subsystem. We will go into more detail on each of these appliances as we as the central hub further down in the report.

We have three major high level requirements that we had achieved upon completing the project. Our first and most important requirement is to create a set of IoT-style appliances that

can be controlled directly from a computer -- turn on and off devices using an optocoupler switcher, detect motion using a motion detector, and turn on and off devices by temperature using a thermostat. No wireless or TCP/IP technology may be used anywhere in the control path between the computer and the appliance. This requirement is important as this is the entire premise of our project; we want to develop smart appliances that promote the security of our customers by not incorporating wireless technologies that could easily be compromised by a network attacker. The next requirement is that the optocoupler switcher needs to have at least 8 outputs, to allow multiple devices to be switched on at one time. We wanted to meet this number so that the end user would be able to use this appliance for multiple devices around their home. The last requirement was that the appliances must work in all its capabilities even without the central hub. The central hub may facilitate communication between the computer and the appliances, but failure to do so should not make the appliances unusable on their own. This is to have a level of modularity incorporated in the project. If the end user does not have access to a central hub, each of the appliances are usable independently.

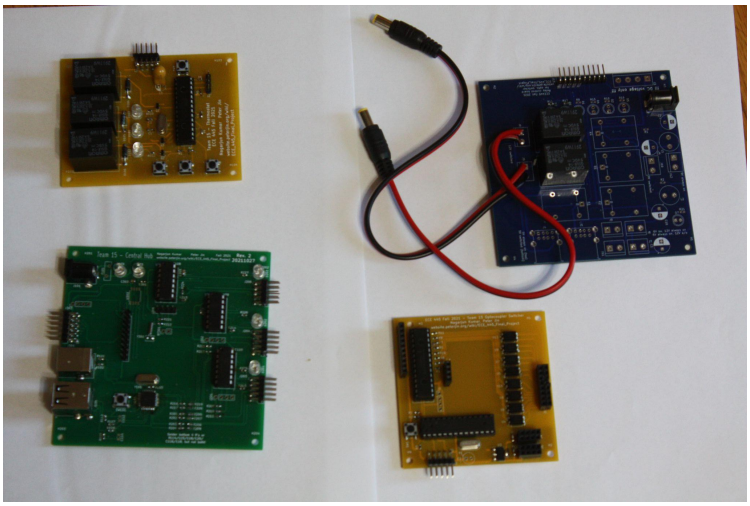

Figure 2: Final product of appliances and central hub.

2.2 Design

2.2.1 Design Procedure and Details

2.2.1.1 Thermostat

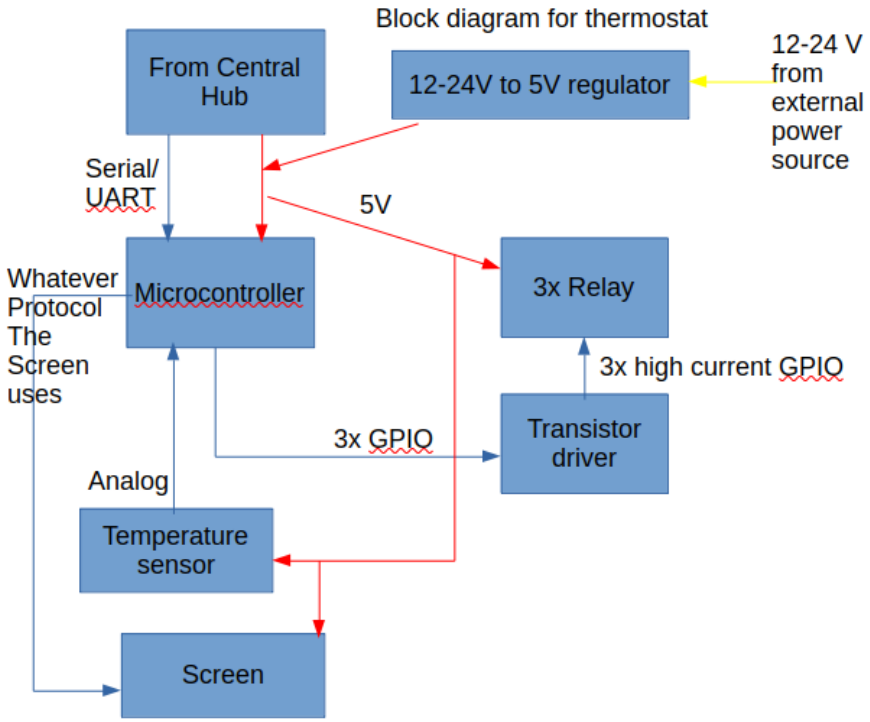

Figure 3: Block diagram for thermostat

The first appliance we will be covering is the thermostat. For this block, we didn't have that many major decisions from what was discussed in our block diagram (see Figure 3). One of the main decisions we had to make for this section was the temperature sensor we were going to use. We initially were deciding between various temperature sensors, but decided to stick with the MCP9804 due to its lower number of pins and accuracy over a wide range of temperatures (-40 to 125 degrees Celsius).

As shown in the Figure 4, the main components necessary for the thermostat PCB are the ATMEGA328P microcontroller, MCP9804 temperature sensor, the 3 LEDs, the 4 buttons, the 3 relays, and the 2x5 GPIO header pins. I will be going into more detail on each of these parts and their purpose in the circuit. The MCP9804 is a surface mount temperature sensor that is able to get room temperature readings from temperatures of -40 degrees Celsius to 125 degrees Celsius. The readings then get sent out via pin A0 to the microcontroller. The microcontroller we are using for this module is the ATMEGA328P. It has a program memory size of 32 KB and can operate from temperatures between -40 to 85 degrees celsius. The main purpose of our microcontroller is to acquire the temperature data from our MCP9804 and be able to increment or decrement the temperature to the liking of the user. Numerous parts are required for the microcontroller to function such as a 7805 voltage regulator, 2 220 ohm resistors, 1 10k ohm resistor, 2 22 pF capacitors, 3 LEDs, and a 16 MHz timing crystal. The 3 LEDs display in three different colors (Red, Blue, Green) and are connected to the ATMEGA328P via ports PD4, PD5, and PD6. Each LED is used to display a different functionality, red for heat, green for fan, and blue for AC. Each of these LEDs is then connected to a diode and a relay after finally connecting to a 1x4 GPIO header to implement an external HVAC system that will replicate heat, AC, and fan.

There are also 4 buttons connected to the ATMEGA328P as well via pins PB0, PB1, PB2, and PC6. The 3 buttons PB0, PB1, and PB2 are used to increment and decrement the temperature as well as enter the main menu on the CLI respectively. The 4th button acts as our programmer and is used to program the microcontroller.

Lastly, pins PD0 and PD1 are connected to our 2x5 GPIO pin header to send the UART protocols to our central hub where it will be converted to USB. This USB protocol will then be sent to the PC to display the readings coming in from the various appliances.

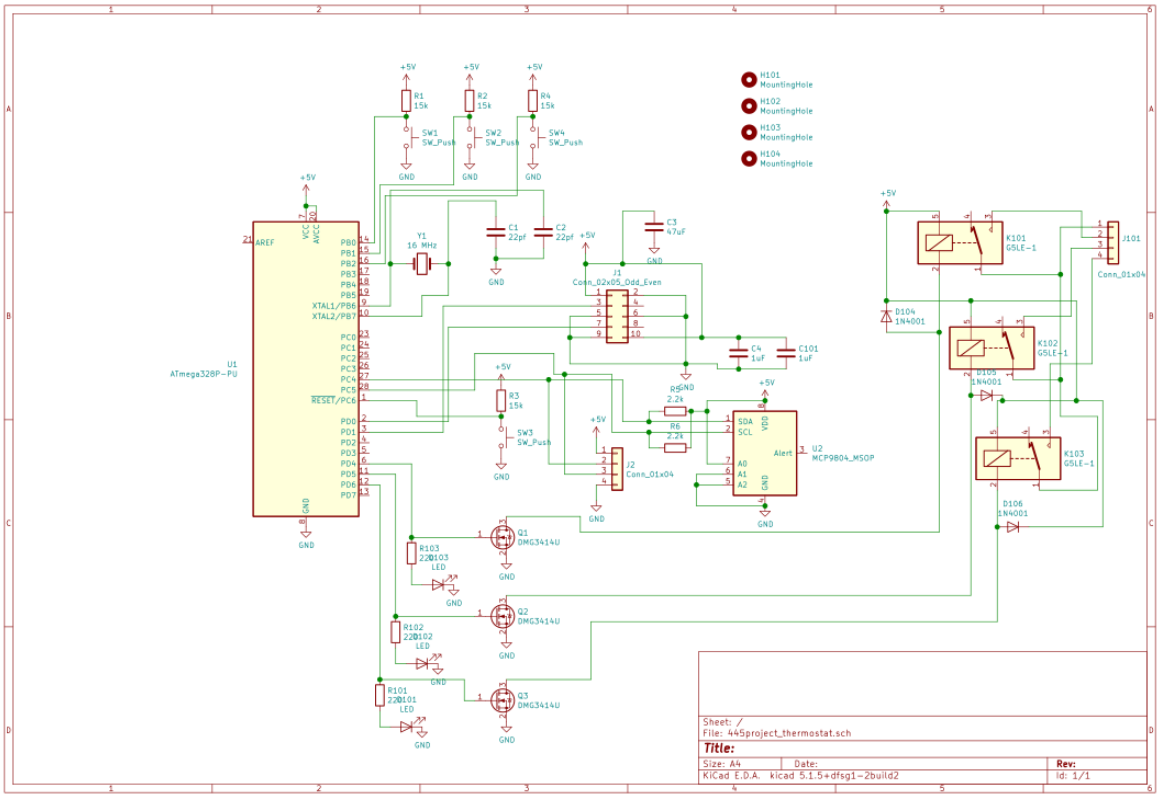

Figure 4: Circuit Schematics for thermostat

The code programmed on the ATMEGA328P chip allows us to get the full functionality of the thermostat. We are able to read data coming from the MCP9804 temperature sensor serially and also able to increment and decrement the set temperature using the buttons on the PCB as well as on the screen. We also implemented a menu screen that allows us to switch between different modes (AC, heat, fan) as well as fine set, and set the temperature via the computer. We were able to implement this menu by creating a state machine that switches between these different settings.

All in all, for the thermostat, there aren't any alternate designs that we could have done as all of the components on our board were essential to display the full functionality of the appliance.

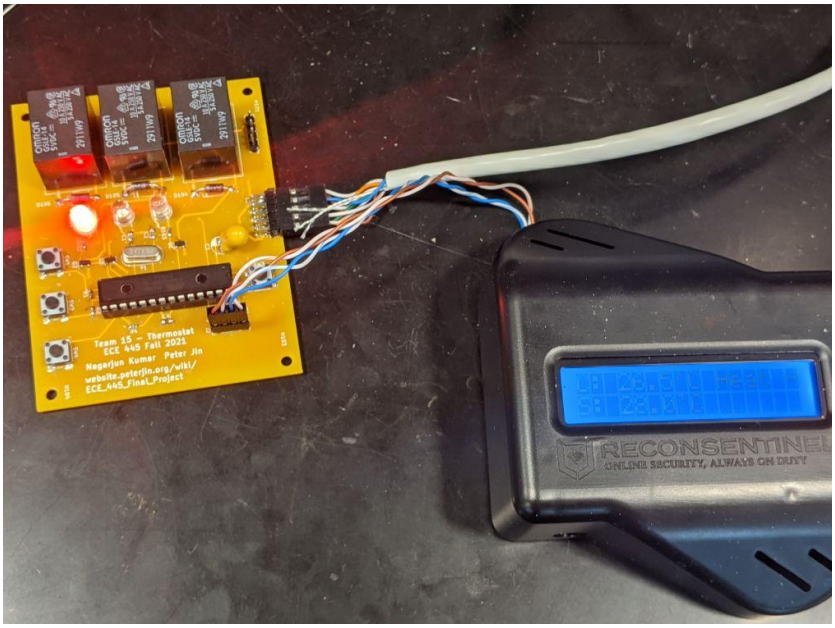

Figure 5: Final rendition of thermostat

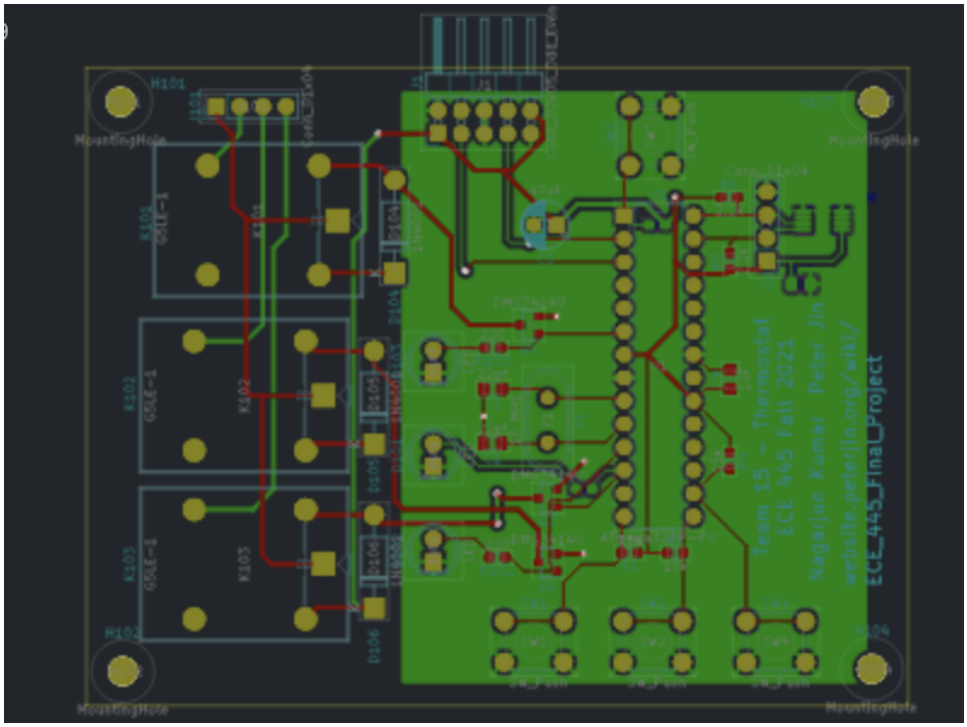

Figure 6: Final PCB Design for Thermostat

2.2.1.2 Optocoupler Switcher

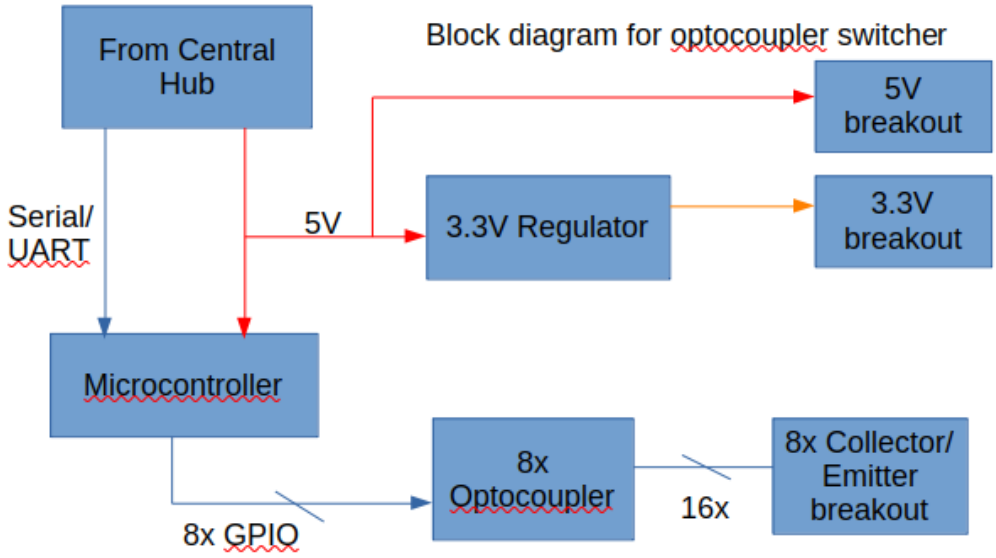

Figure 7: Block diagram for Optocoupler Switcher

The optocoupler switcher is a rather simple device that was originally intended to be a permanent “tether” to a remote control, such that button presses on that remote control can be simulated electronically [9].

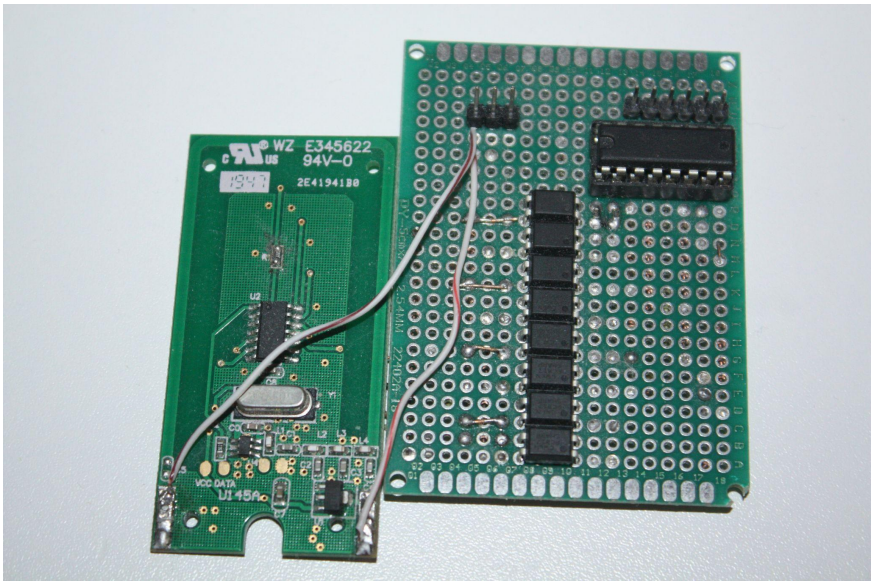

Figure 8: An old prototype of an “automatic controller for a remote control”, which ultimately led to the creation of the optocoupler switcher [9].

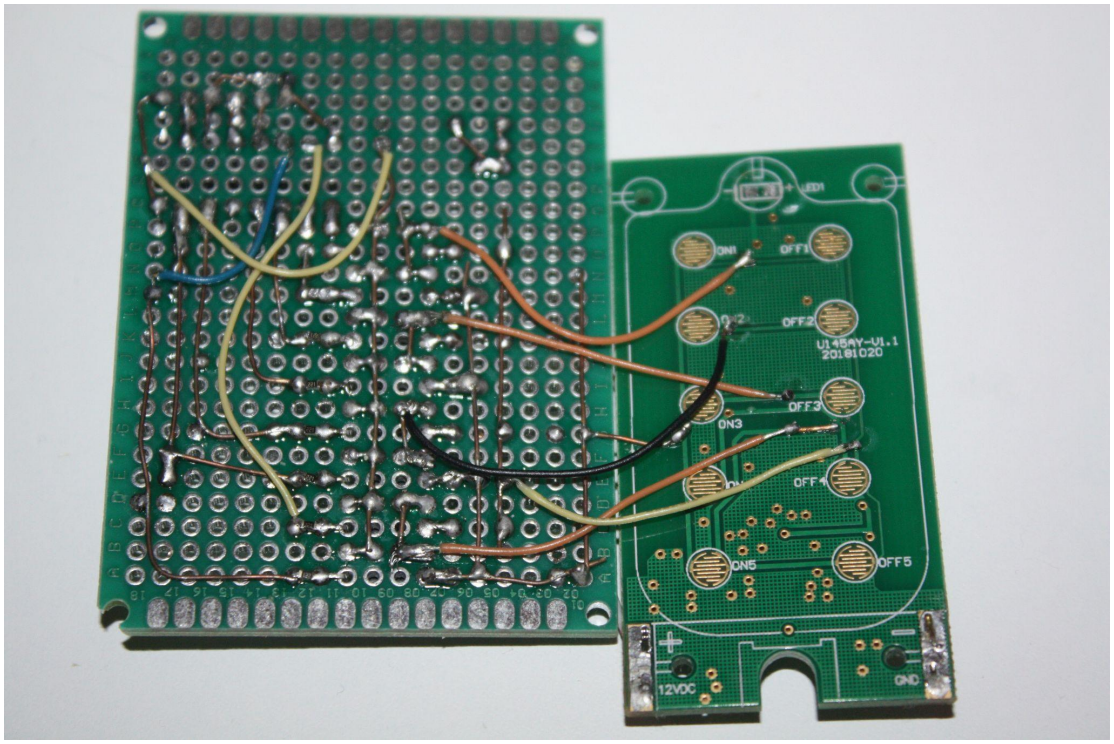

Figure 9: The back side of the same prototype as in Figure 7.

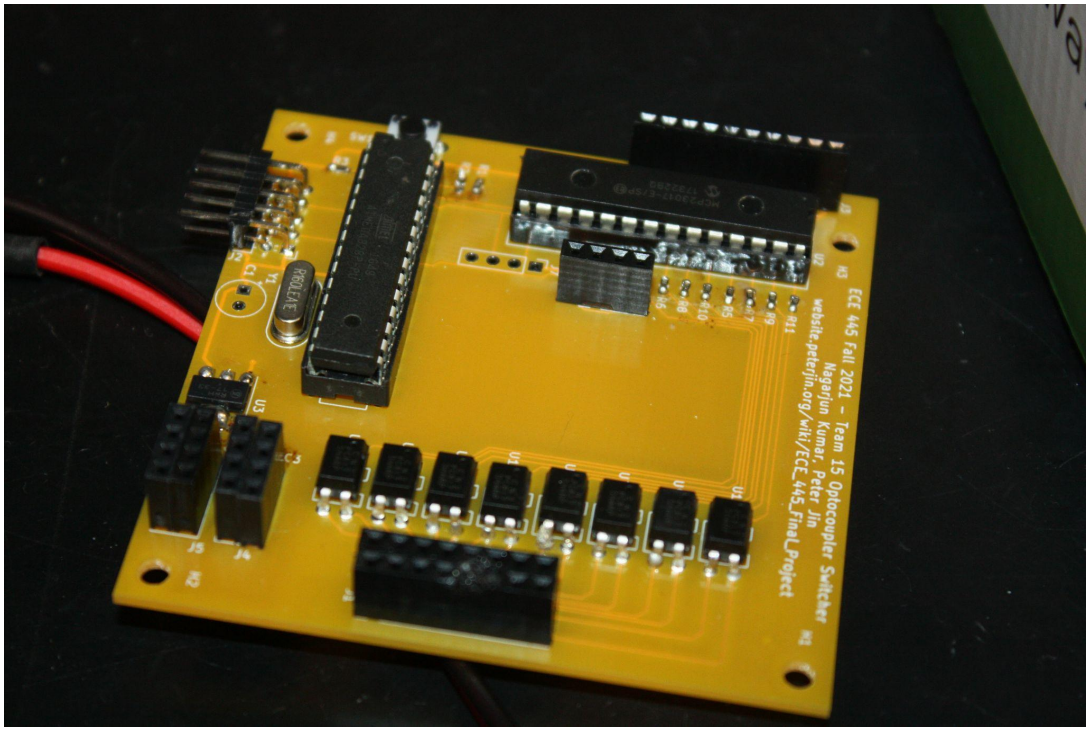

Figure 10: Final rendition of optocoupler switcher

The optocoupler switcher has its origins in Peter's old lab experiments a long time ago, way before this project was even conceived in this final form. As such, although the optocoupler switcher has not been formally verified in its entirety, the concept of using optocouplers to switch buttons on and off is still reasonable.

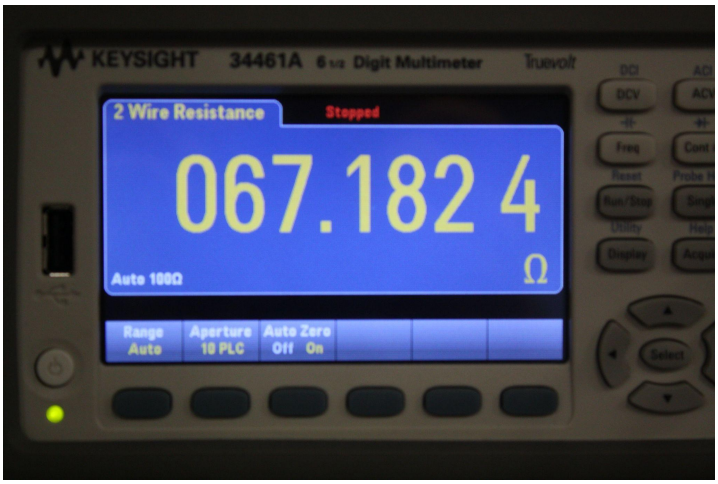

Figure 11: Multimeter readings that checks resistance between C and E is less than 100 ohms

The optocoupler switcher uses the MCP23017 GPIO expander to provide 16 outputs. While it was certainly an option to use the GPIO pins on the ATMEGA328p itself, we were uncertain to the fact that some of the GPIO pins on the microcontroller might not actually be GPIO pins. To prevent any irregularities as to the electrical characteristics of the GPIO pins, we decided to only use the I/O expander's output pins to control the optocouplers [9].

In addition, from the original design document, there was a specification to create a "motion detector" in addition to the optocoupler switcher. That was ultimately merged into the optocoupler switcher, rather than being its own type of appliance.

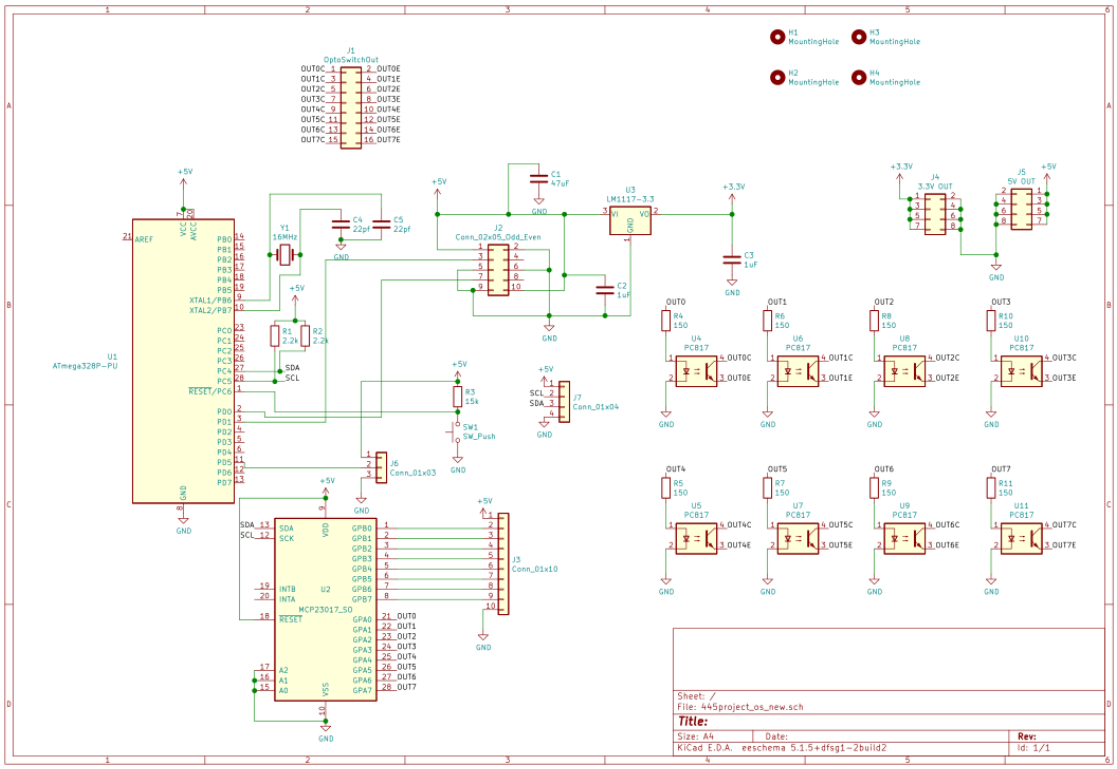

Figure 12: Schematics for optocoupler switcher

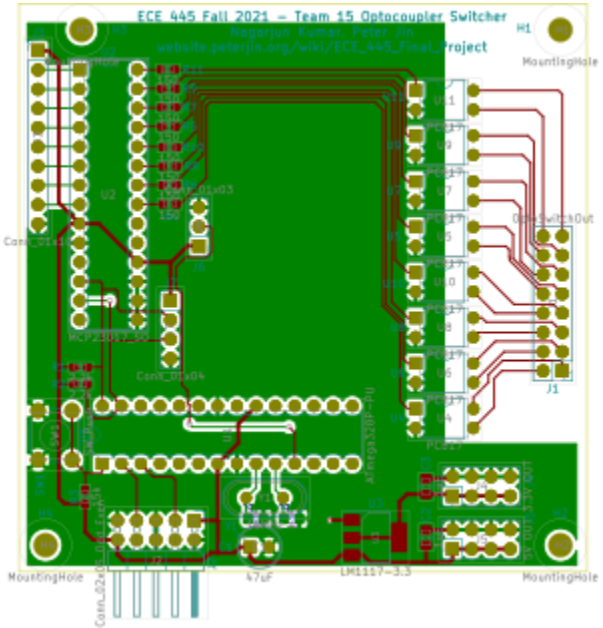

Figure 13: PCB design for optocoupler switcher

2.2.1.2 Central Hub

2.2.1.2.1 Communications between Central Hub and Appliances

For the communications between the central hub and appliances, we used a 10-pin connector (2x5 pins with 2.54mm pitch) with 8 loaded positions, such that we can use regular Ethernet (Cat5e) cable to connect the appliances to the central hub. This connector is identical for both appliances, and this allows the appliances to be permuted in any order (i.e. the central hub must not assume that the thermostat is always plugged into slot 1, for example.)

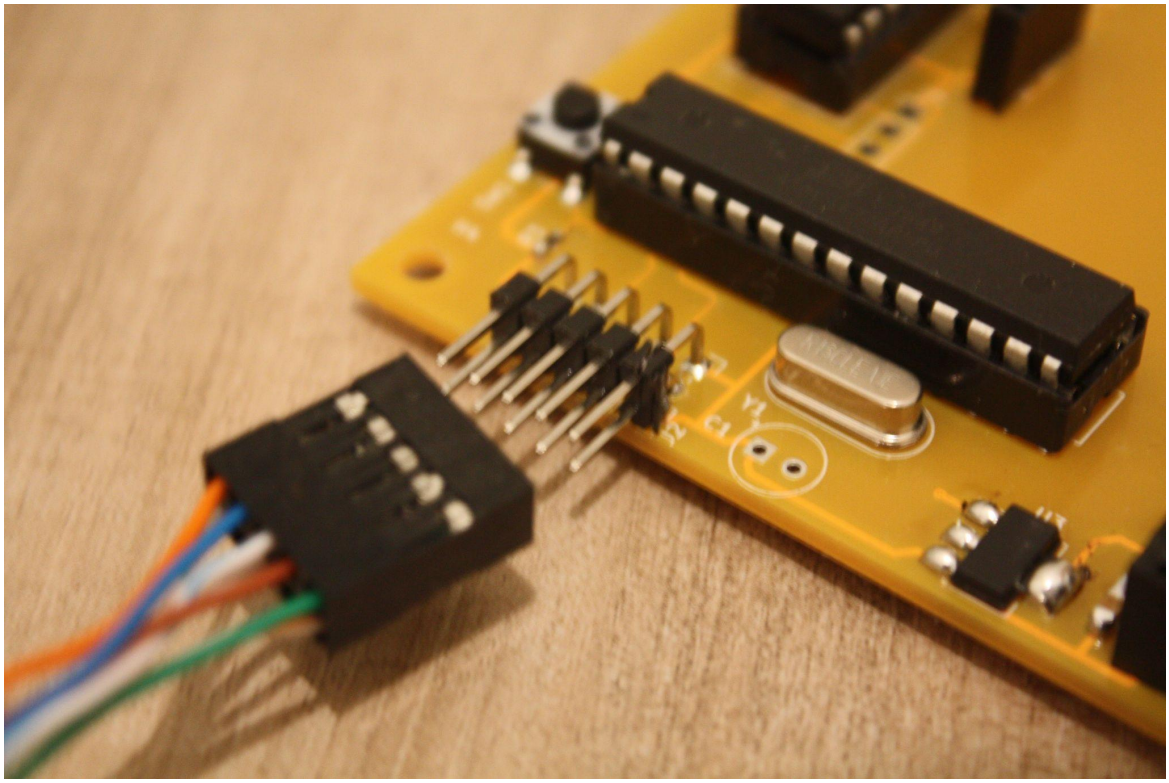

Figure 14: Optocoupler Switcher with 10-pin connector and cable

The pinout of that connector is as follows:

Top	Bottom
+5V (Green)	GND (Orange)
RX (Blue on central hub)	[Empty]
Presence Detection (Striped Blue)	GND (Brown)
TX (Striped Brown on central hub)	[Empty]
GND (Striped Orange)	+5V (Striped Green)

Table 1: Pinout of 10-pin central hub-to-appliance connector

- +5V and GND are used to provide power to the appliance.
- RX and TX form a 9600 baud UART with 5 volt signal level [3]. They are swapped on each end of the connector, essentially forming a “null modem” cable.
- The “Presence Detection” pin connects to a GPIO on the central hub MCP2221A with a pull-up resistor. On the side of the appliance, this pin is tied to GND. The read out of this presence detection pin was ultimately not implemented in the software on the computer.

2.2.1.2.2 Central Hub Design and Procedure

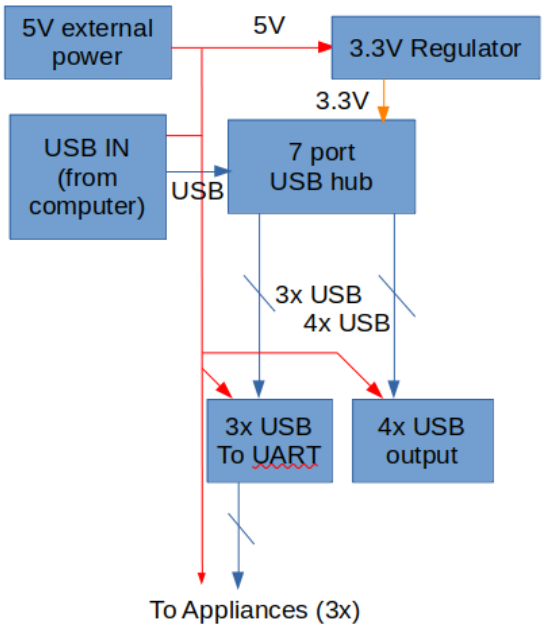

Figure 15: Block diagram for Central Hub

The central hub is fairly simple in terms of its design. It has no programmable microcontroller, but is rather a USB hub with three USB-to-UART ICs connected to it [1]. This simplified the design a lot because we could see this as just a regular USB hub with three USB

devices attached to it, only that in our case, the USB connections (“cables”) are internal and on the same PCB [1].

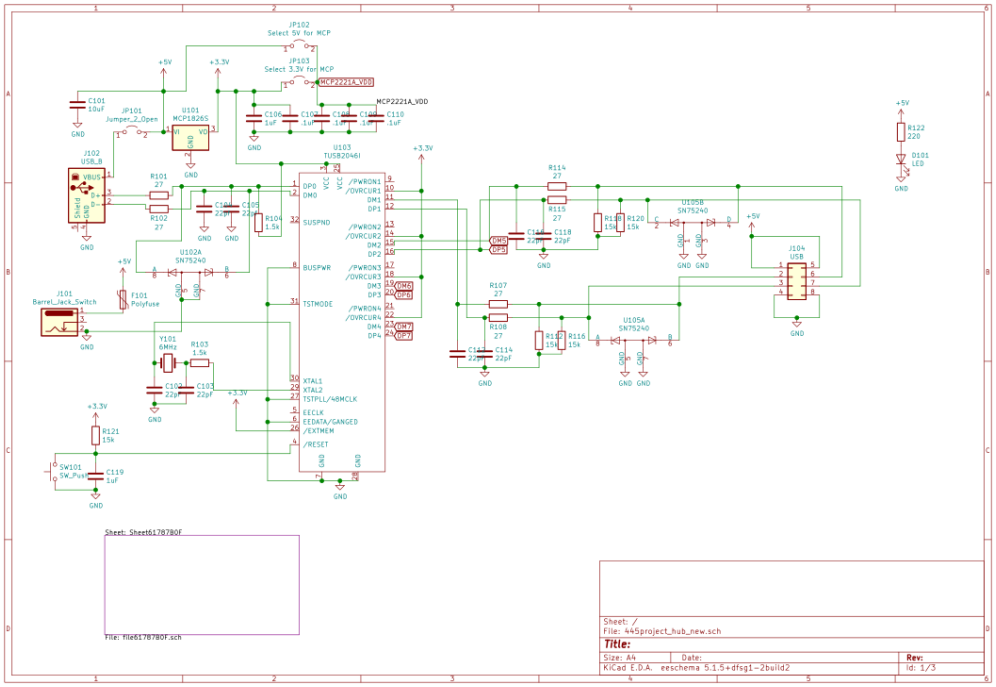

Figure 16a: Schematic of central hub (USB hub portion)

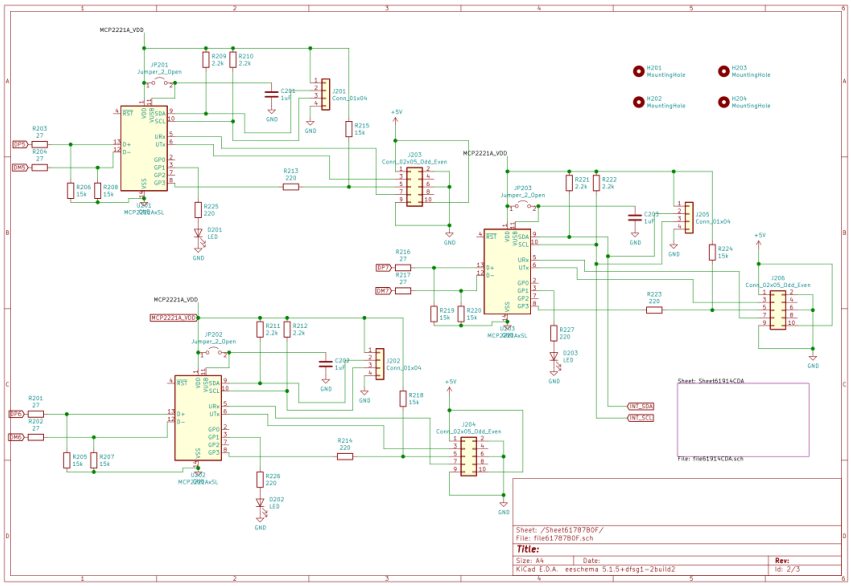

Figure 16b: Schematic of central hub (USB-to-UART portion)

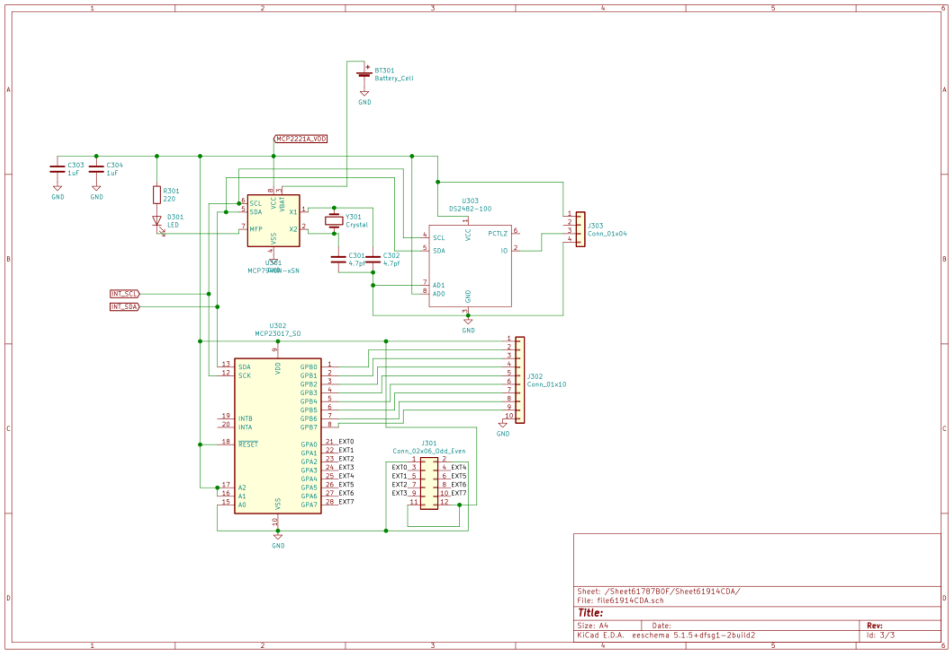

Figure 16c: Schematic of central hub (extra I2C functionality portion)

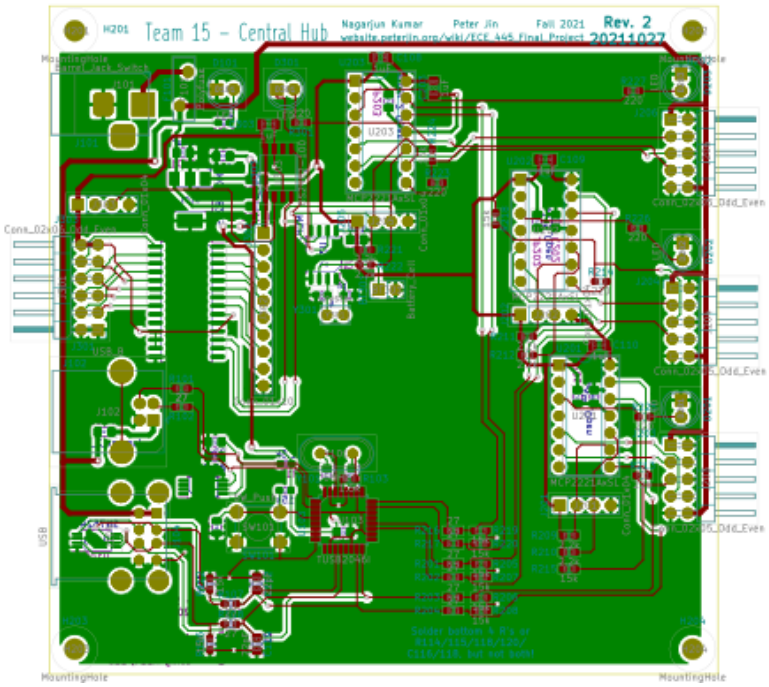

Figure 17: PCB Design for Central Hub

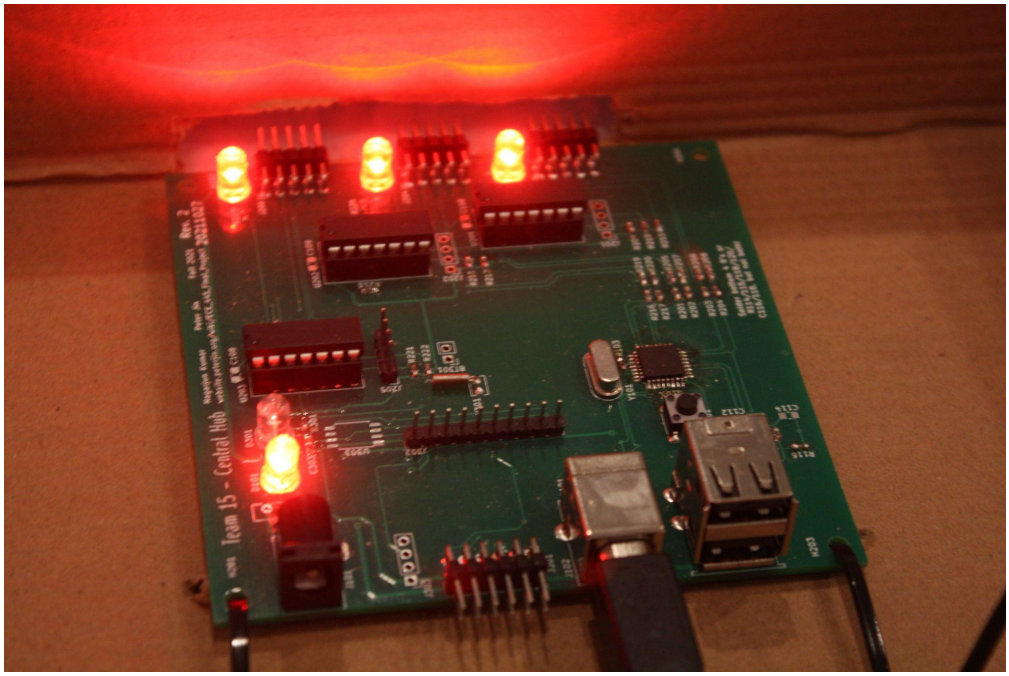

Figure 18: Final product of Central Hub

2.3 Verifications

2.3.1 Thermostat

In terms of the electrical connections and interfaces, the thermostat was fully functional. We were able to read the button states correctly using the Arduino IDE. We were also able to control the relays using digitalWrite on the GPIO pins. Finally, we were able to write text to the screen using the I2C Wire.write() function with the desired text [2].

Requirements	Verifications	Pass/Fail
The computer is able to control the “hot”, “cold”, and “fan” outputs manually.	Using the menu screen, we were able to turn on and off each of the 3 LEDs (red, blue, green) relating to heat, AC, fan.	Pass
Buttons on the thermostat must switch between heat, cool, and fan modes, as well as increasing or decreasing the temperature.	Assuming the ambient temperature is at 62 degrees Fahrenheit, we increase the set temperature to 70 degrees Fahrenheit. This triggers the heat LED to light up as the ambient temperature of the room is lower than the set	Pass

	temperature. This LED continues to be lit until the ambient temperature reaches the set temperature, or if the set temperature gets changed to a value less than or equal to the ambient temperature.	

Table 2: High level requirements for thermostat

2.3.2 Optocoupler Switcher

Our optocoupler switcher seemed to work. When the optocouplers are switched on per the microcontroller code, we observe that the resistance across the C and E pins is about 67 ohms (see Figure 10)

Requirement	Verification	Pass/Fail
The optocoupler switcher appliance must work with both an active-high and active-low remote control button model.	Check that the resistance between C and E is less than 100 ohms	Pass

Table 3: High level requirements and verifications for optocoupler switcher

2.3.3 Central Hub

The central hub met all of our requirements without any issues [9]. We even added a few I2C chips for testing, and those also worked without any issues [7].

Requirements	Verifications	Pass/Fail
For each appliance connected to the central hub, the appliance must work regardless of which output port on the hub the appliance was plugged into.	Connected each of the appliances to each of the available GPIO extenders on the central hub.	Pass
The central hub must be able to support three devices plugged into it at the same time, and in any combination.	Connect all three appliances to each of the GPIO extenders present on the central hub. Passed verification by being able to read from and write to each of the appliances simultaneously.	Pass

Table 4: High level requirements and verifications for the central hub

2.4 Cost

Fixed

Description	Fixed Cost
$20/hr 8 hr/day 5 days/week 10 weeks 2 people X 2.5	$40,000

Table 5: Fixed cost of overall project

Variable

Component	Individual cost	Bulk cost
PCB x3 (hub and 2 appliances) (PCBWay)	$20	$10
ATMEGA328P-PU x2 (Microcontroller for appliances) (Digikey)	$8.46	$7.02
497-4257-1-ND x2 (3.3v regulator) (Dlgikey)	$1.86	$0.82
IC HUB CONTROLLER USB 48LQFP	$6.07	$5.45
RELAY GEN PURPOSE SPDT 1A 5VDC x3	$3.18	$3
Temperature Sensor Analog, Local -40°C ~ 125°C 10mV/°C TO-92-3	$1.71	$1.53
IRA-S210ST01 PYROELECTRIC INFRARED SENSOR	$3.38	$2.58
JFET N-CH 35V 625MW TO92-3	$0.50	$0.38
Category 5e Patch Cord, 10.0 ft, Blue x3	$9.99	$9.30
MOSFET N-CH 20V 4.2A SOT23-3	$0.433	$0.4

OPTOISOLATR 5KV TRANSISTOR 4-DIP	$0.30	$0.24
CONN RCPT USB1.1 TYPEA STACK R/A	$1.06	$0.89
OPTOISOLATR 5KV TRANSISTOR 4-DIP	$0.39	$0.24
CRYSTAL 6.0000MHZ 18PF TH	$0.32	$0.26
IC REG LINEAR 3V 1A SOT223-3	$0.83	$0.83
CONN PWR JACK 2X5.5MM SOLDER	$0.58	$0.50
SWITCH TACTILE SPST-NO 0.05A 24V	$0.14	$0.10
RES 220 OHM 5% 1/10W 0603	$0.04	$0.01
RES 15K OHM 1% 1/10W 0603	$0.02	$0.01
RES 1.5K OHM 1% 1/10W 0603	$0.02	$0.01
RES SMD 2.2K OHM 5% 1/10W 0603	$0.02	$0.01
CAP CER 22PF 50V C0G/NP0 0805	$0.04	$0.02
CRYSTAL 16.0000MHZ 20PF TH	$0.25	$0.15
CAP CER 0.1UF 25V X7R 0603	$0.02	$0.012
CONN RCPT USB1.1 TYPEB 4POS R/A	$0.77	$0.77
PTC RESET FUSE 30V 2.5A RADIAL	$0.7	$0.524
DIODE GEN PURP 1KV 1A DO41	$0.17	$0.123
IC I/O EXPANDER I2C 16B 28SOIC	$1.6	$1.47
IC RTC CLK/CALENDAR I2C 8-SOIC	$1.10	$0.95
MCP2221A	$2.92	$2.92
Total	**$66.873**	**$50.519**

Table 6: Variable Cost of Overall Project

Overall, we approximate that the overall cost to build this project would be around $45. Since our project is fully functional and can be ready to be built in bulk we believe that we can bring this cost to around $35 to build. We believe that this product can be sold for around $50 to consumers for the central hub and the 3 appliances. Considering we are selling in bulk this would give us approximately 15 dollars in gross margin per product sold.

2.5 Conclusions

2.5.1 Accomplishments

All in all, this was a very successful project. We were able to interface with all of our appliances using only USB and UART protocols along with a central hub to the computer. We were also able to read from and write to all of the appliances simultaneously when plugged into the central hub. Lastly, we were also able to have all appliances be functional without the use of the central hub, rather we connect directly to the computer via a UART to USB adapter. This was to ensure a high level of modularity in the project as an end user could use any of the appliances without the need of owning a central hub.

Furthermore, we were able to test and prove that each of our appliances were functional and were giving accurate readings. We were able to test that the thermostat worked by being able to switch between different modes (heat, AC, fan) manually as well as being able to set and fine set the temperatures from the computer. We were able to determine that the optocoupler switcher worked by using a relay board and turning on and off a lamp from it. The motion detector was also able to get accurate readings, although a bit erratic at times. Lastly, we were able to determine the central hub was fully functional as well as we were able to receive readings on our computer from all appliances hooked on simultaneously.

2.5.2 Uncertainties

While we were not able to implement or test every functionality of the optocoupler switcher and thermostat, we still allowed these functionalities to exist in a future version of the software, given that the hardware interfaces have already been verified to be functional.

Something that we had totally overlooked while designing the central hub was the lack of impedance control and differential pair matching on the USB data lines. We believe that the clock speeds for full-speed USB were slow enough that the differential pair matching was not required. If we had used a high-speed hub, then it would have been more of a concern.

Finally, currently, when plugging the central hub into a computer, it takes about 34 seconds for all three USB-to-UART ICs to be detected on the computer. The proper fix for this is to implement power-on-reset sequencing, such that the USB-to-UART adapters are reset about 1 second after the hub is reset, and the USB hub detects the three USB-to-UART adapters

correctly in a proper reset state. However, due to cost considerations and because the central hub still works as-is, this feature does not currently exist.

2.5.3 Ethical Considerations

Our system seeks to replace wireless IoT devices with wired alternatives. There are a few safety and ethical considerations to note with this general approach:

- Since we are using wires to communicate between the central hub and appliances, we have to be careful not to have anyone strangling on those wires [5].
- Since we are seeking to replace wireless IoT devices with wired alternatives, this may lead people to throw out their existing wireless devices; the ethical concern is that this may lead to significant environmental damage as a result of increased e-waste [5].

In terms of this specific implementation, the safety and ethical considerations are as follows:

- We intend for this project to be used in conjunction with software on the computer. In many cases, this will require a specific USB driver, and the end user may not feel comfortable running external code on their computer in kernel space, particularly if it's closed source. We chose the MCP2221A mostly because it is natively supported by Linux for both the I2C and UART interfaces with open-source drivers [5].
- The 10-pin connector between the appliances and the central hub could be mistakenly plugged in backwards, and if the power and ground were to be swapped, then this could result in damage to the central hub or the appliances [6]. This was mostly mitigated by designing the connector such that the power and ground would stay in the same position even if the connector were to be inserted upside down.

2.5.4 Future Work

In the future, we plan on extending this general concept of USB controlled appliances to be applicable to a variety of different appliances such as washers, dryers, refrigerators, security systems, and door locks. We also plan to improve the functionality of the central hub for our existing appliances. We will be adding an ethernet controller with our 4 port USB 2.0 hub. This will allow us to have fine-grained control of the network connectivity of an arbitrary appliance, by plugging in a switch and/or wireless access point into the ethernet controller. We also plan on incorporating an HVAC system to our thermostat to be able to actually increase or decrease a room temperature and also be able to switch between heat, AC, and fan.

2.6 References

[1] “USB hardware,” *Wikipedia*, 25-Sep-2021. [Online]. Available: https://en.wikipedia.org/wiki/USB_hardware#Cabling. [Accessed: 30-Sep-2021].

[2] “American Wire Gauge,” *Wikipedia*, 28-Sep-2021. [Online]. Available: https://en.wikipedia.org/wiki/American_wire_gauge. [Accessed: 30-Sep-2021].

[3] “Differential signalling,” *Wikipedia*, 29-Sep-2021. [Online]. Available: https://en.wikipedia.org/wiki/Differential_signalling. [Accessed: 30-Sep-2021].

[4] “ARP spoofing,” *Wikipedia*, 24-Jul-2021. [Online]. Available: https://en.wikipedia.org/wiki/ARP_spoofing. [Accessed: 30-Sep-2021].

[5] “The code affirms an obligation of computing professionals to use their skills for the benefit of society.,” *Code of Ethics*. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 30-Sep-2021].

[6] “IEEE code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 30-Sep-2021].

[7] “RF Wireless World,” *UART vs SPI vs I2C | Difference between UART,SPI and I2C*. [Online]. Available: https://www.rfwireless-world.com/Terminology/UART-vs-SPI-vs-I2C.html. [Accessed: 30-Sep-2021].

[8] “RF Wireless World,” *Advantages of UART | disadvantages of UART*. [Online]. Available: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-UART.html. [Accessed: 30-Sep-2021].

[9] “How an optocoupler works: Eagle: Blog,” *Eagle Blog*, 02-Feb-2021. [Online]. Available: https://www.autodesk.com/products/eagle/blog/how-an-optocoupler-works/. [Accessed: 30-Sep-2021].