

USB Controlled Appliances

ECE 445 Design Document

Nagarjun Kumar
Peter Jin
TA: Dean Biskup
Team 15

1 Introduction

1.1 Problem and Solution Overview

We will be 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 [10]. 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 [10]. So instead of relying on wireless technologies, our solution only ever uses wires for communication using much simpler protocols like USB and UART.

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.

1.2 Visual Aid

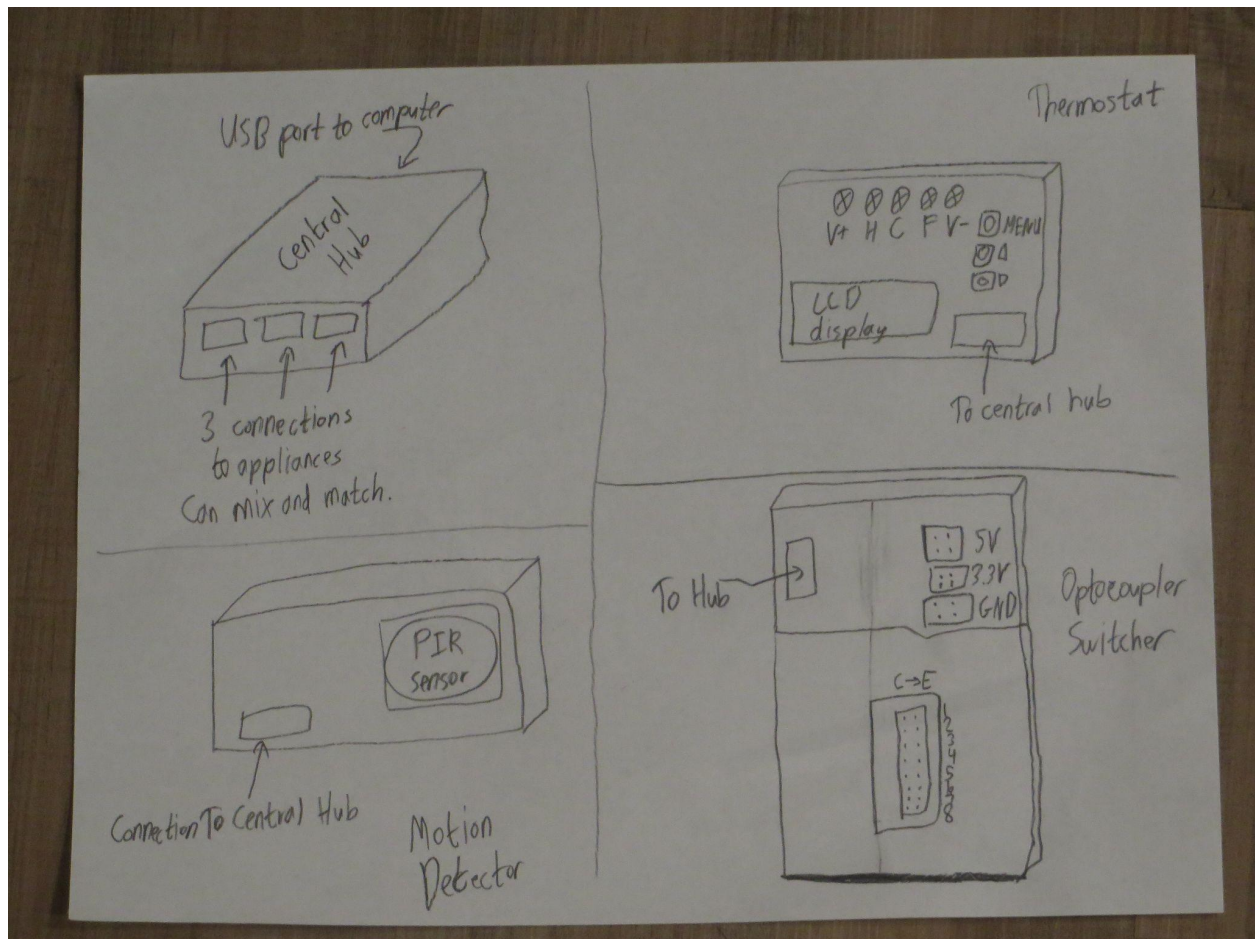


Figure 1a (top left), 1b (bottom left), 1c (top right), 1d (bottom right). Visual representations of hub and appliances.

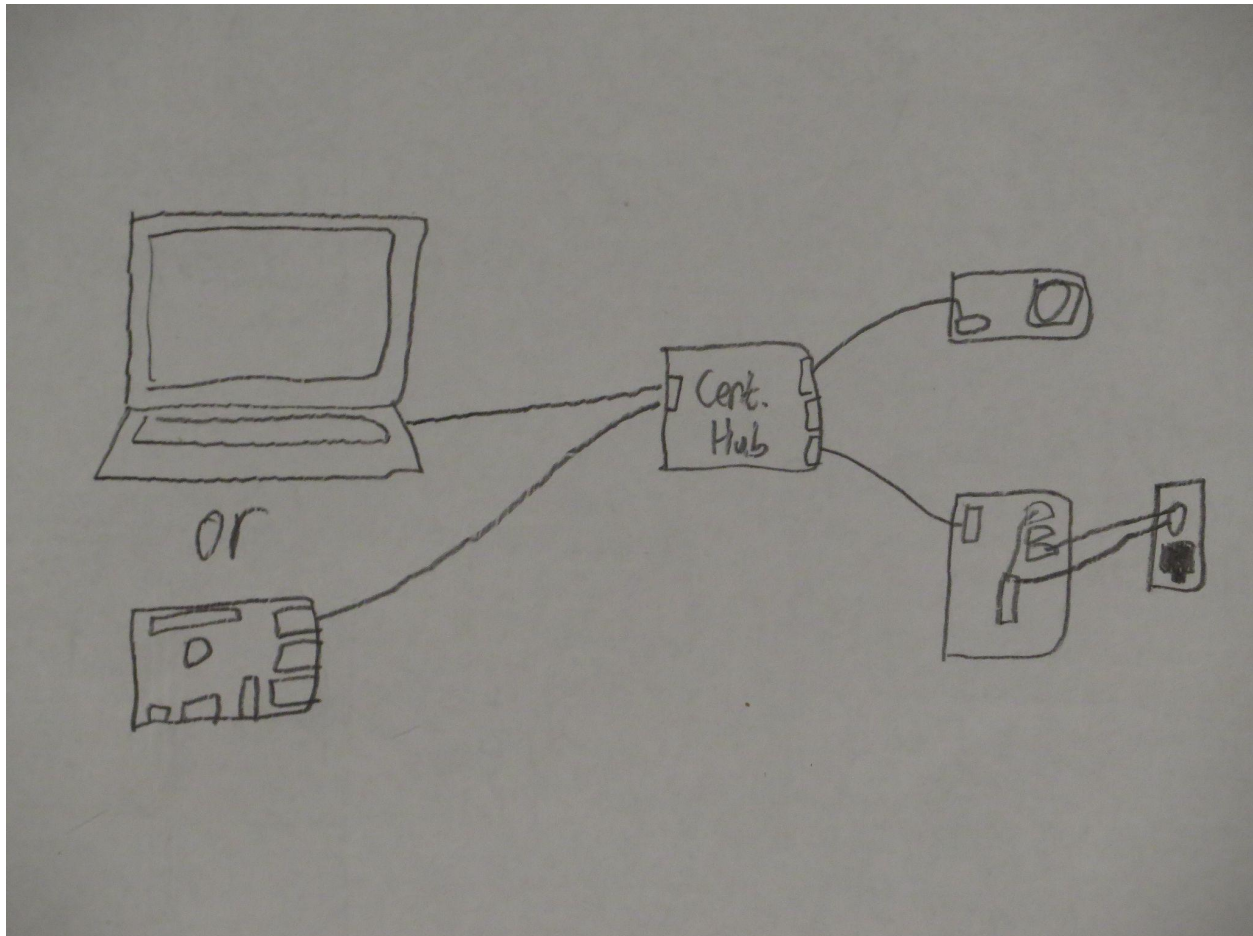


Figure 2 Example connection of devices by the end user.

1.3 High Level Requirements

- The most important objective 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.
- The optocoupler switcher needs to have at least 8 outputs, to allow multiple devices to be switched on at one time [9].
- 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.

2 Design

2.1 General Block Diagram

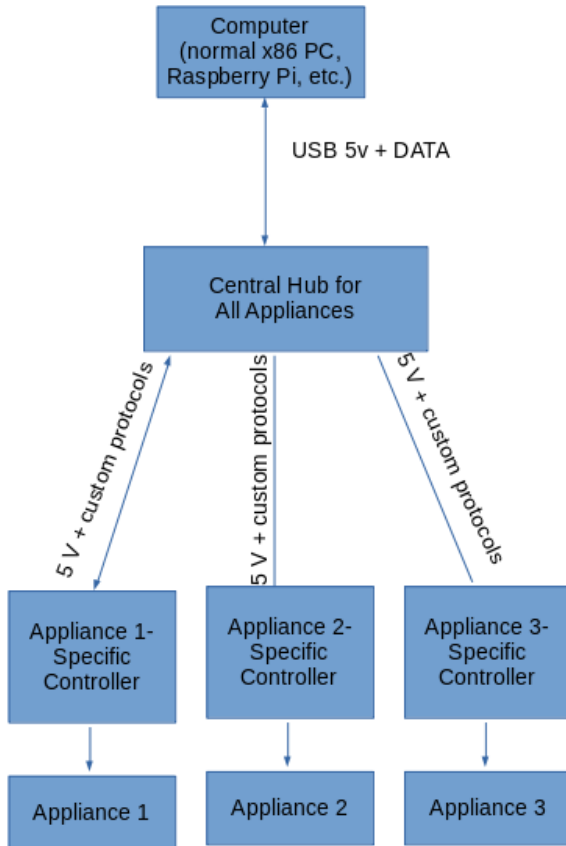


Figure 3: General block diagram of project.

2.2 Central Hub

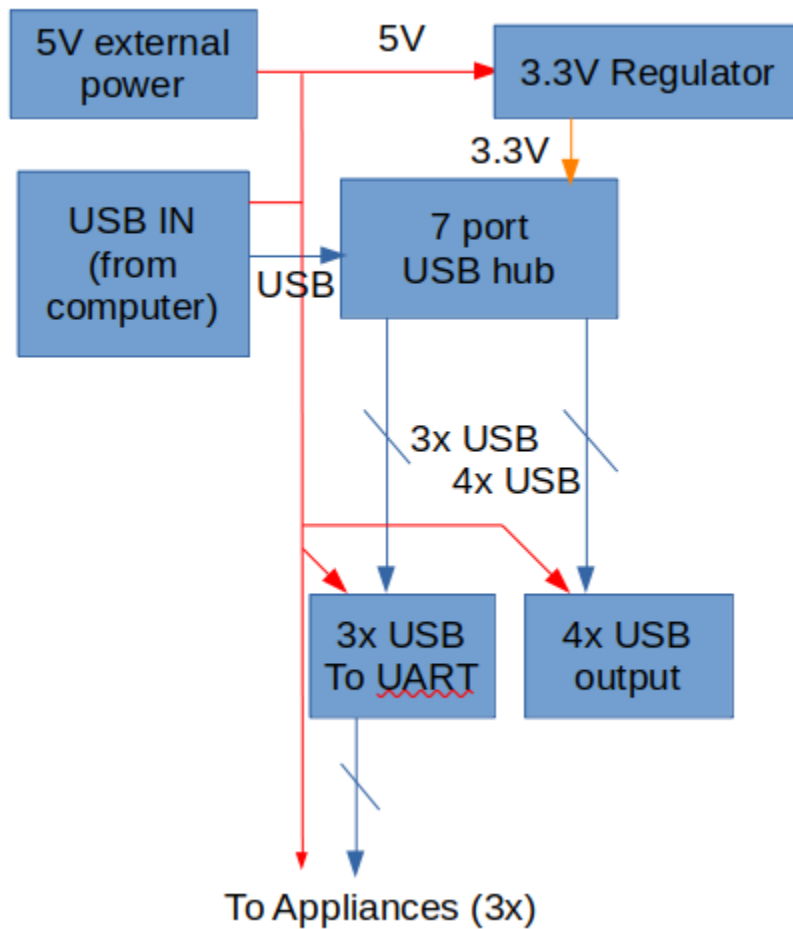


Figure 4: Block diagram of central hub.

The central hub consists of a 7-port USB hub with three of the ports connected directly to USB-to-UART ICs, with additional supporting ICs like voltage regulators. The four other ports are exposed externally to allow more central hubs to be daisy chained together, in case three appliances are not enough. (Devices other than another instance of the central hub may still be plugged into these USB ports, though they will only have a speed of 12 Mbps due to a limitation of the USB hub IC.) The main purpose of the central hub is to convert between the USB on the computer and the UART modules on the appliances. This reduces the complexity of the appliances such that if the central hub is bypassed, the USB protocol does not need to be

reimplemented by the end user, in case they're using a very memory or CPU-constrained device connected directly to the appliance.

Requirement	Verification
1. Central Hub must be able to relay UART data from the appliances to and from the computer, with a round-trip latency of less than 50 ms [7].	Send a command from a laptop running x86 Linux to the central hub that turns an optocoupler on, and wait for a response. Using the "time" Linux command, the "real" time of the action should be less than 50 ms, and the optocoupler should turn on [9].
2. Central Hub must be functional on Linux desktops, laptops, and single board computers.	Connect the Central Hub to each of the following computers, and verify functionality of the appliances: <ul style="list-style-type: none">• A laptop running x86 Linux• A single-board computer (e.g. Raspberry Pi) running ARM Linux• Another single-board computer running ARM Linux
3. 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.	Plug each appliance into one of the hub's output ports, and verify that the device is detected on the computer on the correct output port, as well as the functionality of the appliances. Repeat this on each of the other two output ports for every appliance, and the appliance should work in exactly the same way.
4. The central hub must be able to support three devices plugged into it at the same time, and in any combination.	<p>Plug all three appliances into the central hub. Verify that each appliance works independently of each other, as well as simultaneously. Repeat this process for all six possible permutations of three appliances.</p> <p>Randomly select 6 of the 64 possible permutations (nothing, motion detector, optocoupler switcher, or thermostat connected to each output). For each of the 6 permutations, verify that each appliance is detected correctly, and in the correct order.</p>

2.3 Appliance Subsystem (Common)

Common requirements applicable to all three appliances. (This heading is not an appliance in and of itself.) All appliances will have breakout headers and a removable microcontroller where appropriate to allow custom logic to be applied without any built-in microcontroller. In addition,

the user could even substitute in their own microcontroller, and the appliance should work in the exact same way, subject to the logic of that microcontroller.

Requirement	Verification
1. For each appliance that is declared to have a removable microcontroller, basic functionality of the appliance must still be achievable in some way by adding external logic when the microcontroller is removed.	Design a simple external logic circuit on a breadboard and connect it to the appliance's PCB using headers on the PCB with the microcontroller removed. The appliance will be disconnected from the central hub, and for appliances without an external power source, an external 5 V +/- 0.25V power source capable of supplying at least 500 mA will be connected to it. Verify that despite these conditions, the external logic is able to control the appliance for both TTL and CMOS outputs. (Whether the logic implemented by the external circuit itself is correct is irrelevant.)
2. For appliances which do not have an external power source, the appliance must work with an input voltage between 4.4 and 5.5 V.	The cable connecting the appliance to the hub will be spliced in a way such that the appliance's power will come from an external power source, rather than from the computer's USB. Verify functionality of the appliance when the power source is set at 4.4 and 5.5 V.
3. Appliances should still work by manually using a separate USB-to-UART chip (such as the MCP2221A), without relying on the central hub. (This mechanism must be meaningful even for an end product, and should not be viewed as merely a convenience for debugging.)	Connect a USB to UART adapter directly from the computer to the appliance, without using the central hub [8]. Verify functionality of each appliance by manually sending commands to the tty or COM device on the computer, and verifying the result to be the same as if the command were sent from the central hub. This configuration will be referred to in later verifications as "standalone UART" or "STUART" mode.
4. (If a 3.3V microcontroller or output is used in the appliance) The voltage regulator to step down from 5 V to 3.3 V has an output voltage range of 3.1 to 3.4 V with an output current of 250 mA.	Apply a 13 ohm load on the voltage regulator's output to ground. Measure the voltage of the output using a multimeter; it should be between 3.1 to 3.4 V.

2.3.1 Appliance Subsystem (Motion Detector)

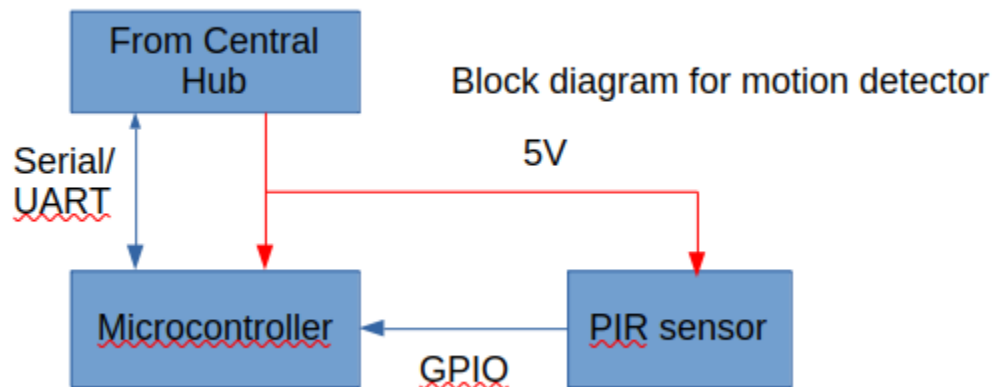


Figure 5: Block diagram of motion detector.

The motion detector is an example of a generic input device. It can be used with the computer or hub logic to cause any of the other outputs or something on the computer to be automatically turned on. For example, it can cause a camera on the computer to be turned on, under the discretion of the computer's software. For the purposes of the verifications below, "signal detected" means that the appliance is in STUART mode, and the computer's serial terminal detects the character "S" to indicate a signal. This occurs when there is a change in voltage on the GPIO line.

Requirement	Verification
1. The motion detector must be able to detect walking humans in the background for distances up to 3 m away.	Have humans walk (at a speed of 1 m/s, perpendicular to the motion detector's visibility) in front of the motion detector at distances of 0.5 m, 1 m, and 3 m (measured using a meter stick). Verification passes if the signal is detected at 3 m. Assume that the back is more than 6 m away.
2. When there is no object behind the motion detector, there must not be a signal for up to 5 minutes.	Keep the area of motion detection still for five minutes (measured using a stopwatch). If no signal is detected for this entire time, then this verification passes.
3. If an object is detected on the motion detector, it should report to the computer within 5 seconds.	Put something near the motion detector; use a stopwatch to determine the time from the object being placed and a signal being detected on the computer; verification passes

	if the time is < 5 s. Should be performed with the central hub, rather than in STUART mode.
--	---

2.3.2 Appliance Subsystem (Optocoupler Switcher)

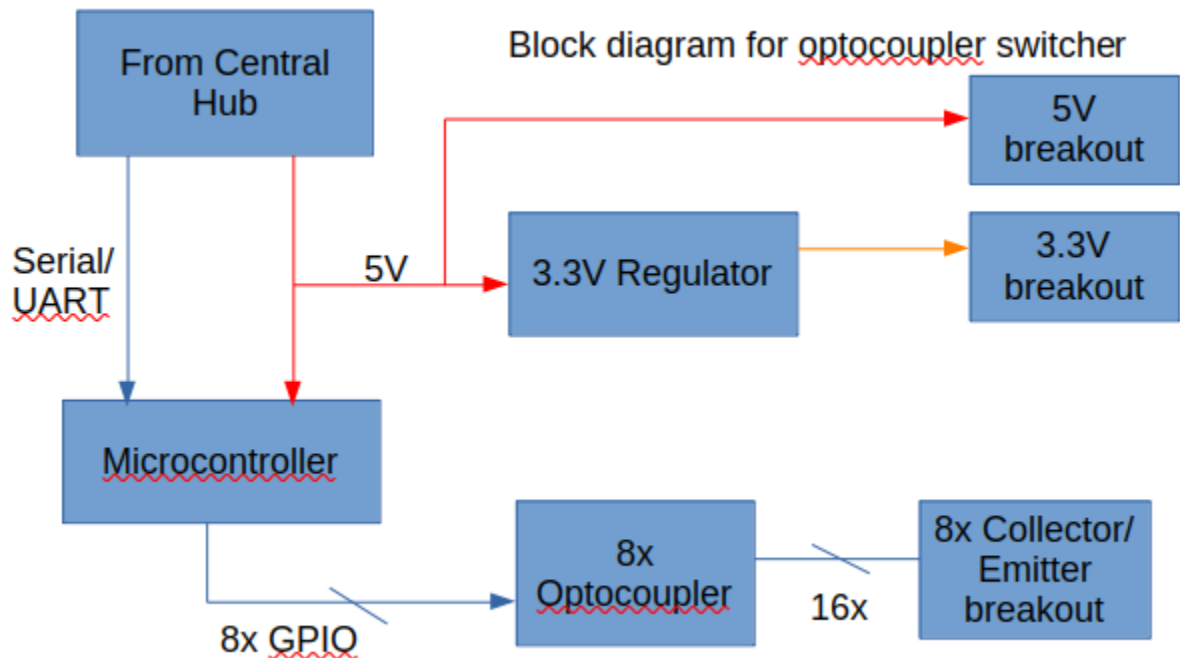
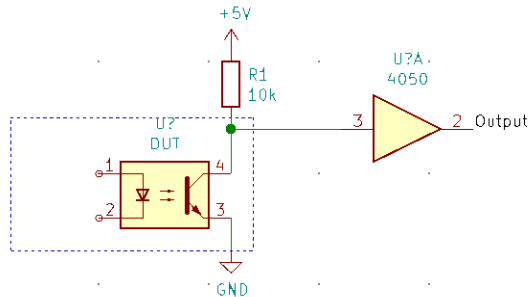
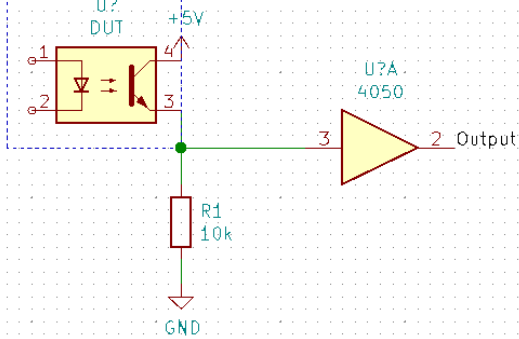


Figure 6: Block diagram of optocoupler switcher [9].

The optocoupler switcher is a generic means of switching on and off electronic devices by piggybacking off a remote control (garage door opener, TV control, light control), without any knowledge of the remote control's electrical wiring, by using optocouplers to simulate the low resistance of button presses [9]. It has a series of eight outputs, each with two pins for the optocoupler's emitter and collector [9]. Under normal operation, the two pins of each output are connected across a button of interest on the remote control (by the end user manually soldering the wires onto the control). When the optocoupler is switched on, the button on that remote control is "pressed". Commands exist on the computer to turn on and off each switch independently, as well as for a set time limit. In addition, there are output voltage terminals of 3.3 and 5 V for battery elimination purposes. (These voltages were chosen, rather than 3 and 6 V, because these voltages are still close enough to the typical battery voltages of a remote control, and 3.3 and 5 V are more convenient for logic anyway.) Unless otherwise specified, commands are to be sent directly from the computer in the STUART mode configuration.

Requirement	Verification
1. The computer is able to turn on and off the optocouplers reliably (i.e. < 10 ohms on, and > 100 kilohms off).	Send the appropriate commands (to be determined based on microcontroller code) (in STUART mode) to cause the optocoupler to be switched on and off. When switched on, the resistance across the optocoupler should be < 10 ohms. When off, the resistance should be > 100 kilohms.
2. The 3.3V output of the optocoupler switcher must have an output voltage between 3.15 and 3.45 V for the maximum load current of 250 mA at an input voltage of 4.4 V.	With a forced 4.4V input voltage as demonstrated in requirement 3 of “Appliance Subsystem (Common)”, apply a 13 ohm load onto the 3.3V output. Measure the voltage of the output with the load using the multimeter to verify that the output voltage is between 3.15 and 3.45 V.
3. The optocoupler switcher appliance must work with both an active-high and active-low remote control button model.	<p>Attach the optocoupler to the following remote control model, where the 5V source comes from the 5V power output pin on the optocoupler switcher (5V also used for the buffer’s power supply), and the optocoupler shown below is exposed as the collector and emitter output pins of the switcher:</p>  <p>Figure 7a: Active low configuration.</p> <p>Verify that in the active-low configuration the output reads “0” by a logic analyzer or oscilloscope if and only if the optocoupler is switched on by sending a command to turn it on in STUART mode.</p>

	 <p>Figure 7b: Active high configuration.</p> <p>Verify that in the active-high configuration, the output is “1” when read by a logic analyzer or oscilloscope if and only if the optocoupler is switched on by sending a command to turn it on in STUART mode.</p>
<p>4. When an output is only switched on for a time limit between 0.1 and 5 seconds, the accuracy of the time limit is within 10% of the set time limit.</p>	<p>Set an output to be switched on for 0.1, 0.2, 0.5, 1, 2, and 5 seconds, attached to the active-high model described above. Use an oscilloscope to verify that the time that the output is turned on is within 10% of the set time.</p>
<p>5. The reaction time between when a signal is sent from the computer to the optocoupler should be less than 0.1 seconds.</p>	<p>Send a signal from the computer at the same time a button connected to an oscilloscope is pressed (to indicate “start” on an oscilloscope). Hook up the output of the active-high model to a second channel on the oscilloscope. If the rise time between these signals is within 0.1 seconds, then the verification passes. This verification shall be done with the central hub, rather than in STUART mode.</p>
<p>6. Multiple outputs from the optocoupler switcher must have their own independent time limits.</p>	<p>Set one output to have a time limit of 1 second, then use the “sleep” command on the computer to wait for 0.5 seconds, then set another output to have a time limit of 2 seconds. As measured by an oscilloscope using the active-high model, the first output should turn off at $t=1 \pm 0.1$ s relative to when the first output turns on, and the second output should turn off at 2 ± 0.2 seconds relative to when the second output turns on. (In order for this verification to be useful, the second output must turn on before the first</p>

	output turns off.)
7. When an output is only switched on for a time limit, it should not be possible for a computer crash to keep the output stuck high.	Send commands from the computer to turn one of the optocouplers on for a time limit of 5 seconds. Within that time limit, press alt-sysrq-c on the computer to manually crash the system (set kernel.sysrq=1 on the computer beforehand). Despite this, the output signal from the optocoupler switcher should not be stuck high past the time limit.

2.3.3 Appliance Subsystem (Thermostat)

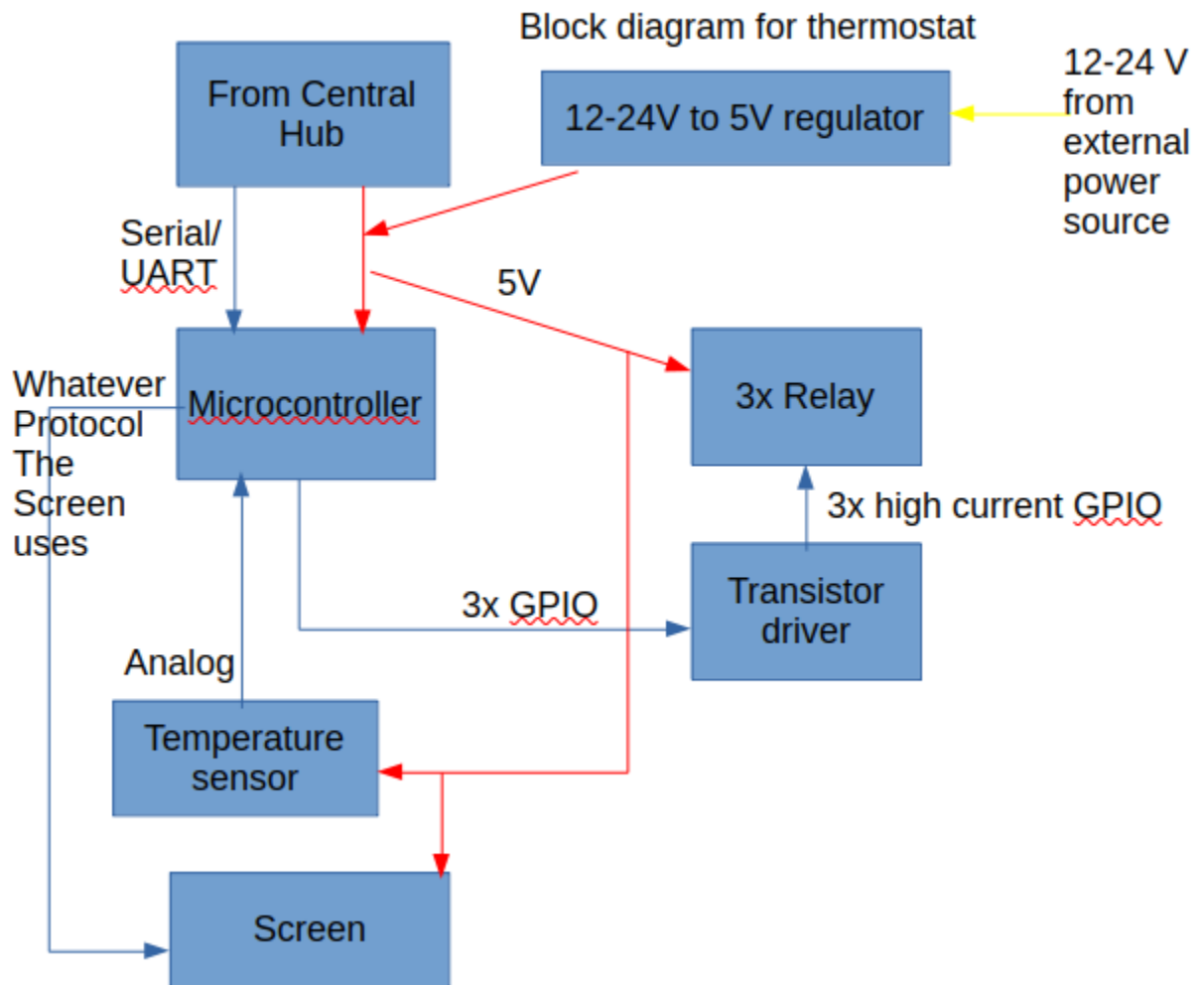


Figure 8: Block diagram of thermostat.

The thermostat is fundamentally a device with three relays for heat, cool, and fan, plus a temperature sensor. It can optionally be powered by the HVAC system itself (i.e. the external power source). In a normal mode of operation, the “heat” is turned on when it is too cold, “cool” is turned on when it is too hot, and “fan” can be turned on and off manually. These output signals can be controlled as part of the normal thermostat logic on the appliance itself, or the signals can be manually switched on or off by the computer. The thermostat will also have a screen to display the current and set temperature, as well as buttons to control it (like a normal thermostat). Unless otherwise specified, all commands for verification are to be sent using the STUART mode configuration.

Requirement	Verification
1. The computer is able to control the “hot”, “cold”, and “fan” outputs manually.	Send commands from the computer to turn on and off each of those three outputs. Verify that the relays switch on and off as a result of these commands.
2. All three outputs should automatically turn off if no “ping” packet was received in 15 to 20 minutes.	Send a command from the computer to turn on the “hot” signal, and don’t send any other commands. The “heat” output should turn on within 1 second, and turn off after 15 to 20 minutes.
3. When “ping” packets are sent, the output should be able to stay on for a 1-hour period.	Send a command from the computer to turn on the “hot” signal, and send “ping” commands every minute. The “heat” output should turn on and stay on for the entire 1-hour period.
4. The temperature sensor on the thermostat must be able to read ambient temperatures between 0 and 40 degrees Celsius with 1 degree accuracy.	Bring the thermostat to rooms of 0, 10, 20, 25, 30, and 40 degrees Celsius. Check that the temperature sensor reads the temperature correctly to the computer within a 1 degree accuracy.
5. When the logic is in the thermostat, the computer should be able to control the temperature the thermostat should be set to from a range of 10 to 35 degrees Celsius.	Send a command from the computer to the central logic to set the temperature of the system, and whether or not heating, cooling, or the fan is requested. Verify that when heating, the “heat” output is on if and only if the temperature measured by the selected temperature sensor is less than the set temperature; ditto for “cool” output for temperatures greater than the set temperature. This verification is to be performed for set temperatures at 10, 15, 20, 25, 30, and 35 degrees Celsius.
6. Buttons on the thermostat must switch between heat, cool, and fan modes, as well as increasing or decreasing the temperature.	Verify that pressing buttons increases or decreases the set temperature. There must also be a sequence of buttons to change between heat, cool, and fan modes.

2.4 Software Subsystem

The software subsystem refers to any software that runs on the computer. The main functionality of this program is to read any data coming from the three appliances as well as write that data back to each of them.

Requirement	Verification
1. Appliances must work for cable lengths for up to 10 m from the central hub.	Construct cables that connect the appliance to the hub for lengths of 1, 3, and 10 m, as measured by a meter stick. Verify functionality for each appliance with each length of cable. The cable will be subject to conditions such as EMI (put near a wireless router) and bending (extend the wire completely, then crush the wire into a ball) during testing.
2. The LCD display must show the temperature as reported by the selected temperature sensor, as well as the status of “heat”, “cool”, and “fan” lines.	Compare the temperature reading from the computer and the temperature reading on the LCD screen, to see that they are the same within a whole number rounding error.
3. Thermostat readings must be visible on the LCD display.	Compare readings acquired from the thermostat with that displayed on the computer to make sure that they are both accurate to display precision.
4. The program is able to write information to the appliances, (switch or temperature or heat/cool/fan status of thermostat)	Change these appliance settings through the program and verify that changes are applied to each of the appliances.

2.5 Tolerance Analysis

The main risk here is the need to transmit UART signals over long lengths of wire. (USB signals already have a maximum cable length of 5 m, according to [1].) Generally, if running signals over long distances, the voltage of the signals needs to be increased, and the frequency of the signal needs to be decreased. We will initially test our prototype with a 9600 baud UART with 5 V signaling, which is still sufficient for simple switches like the optocoupler switcher and the thermostat [7]. If it turns out that there are still signal losses, we could have an optional external power supply for each appliance. On the other hand, if the transmission quality is still good, then the UART baud rate could be increased to as high as possible. Note that this baud rate has to be specified in advance and be the same for all three appliances, because the UART protocol does not have a mechanism to negotiate baud rate [7].

In order to have a voltage drop which is $< 0.5 \text{ V}$ through a 10 m cable from the central hub to the appliance with a current of 0.5 A, the maximum allowed resistance across the wire must be less than $0.5 \text{ V} / 0.5 \text{ A}$, or 1 ohm. This current runs across 20 m (forward and back through the cable), so the maximum allowed per-length resistance is $1 \text{ ohm} / 20 \text{ m} = 50 \text{ milliohms per meter}$. According to [2], this would mean that the wire gauge for the power lines would need to be 21 AWG or higher. 20 AWG wire sounds sufficient, as it has a resistance of 33.31 milliohms per meter, which translates to a total voltage drop of $0.03331 \text{ ohms per meter} * 20 \text{ m} * 0.5 \text{ A} = 0.3331 \text{ V}$, which is still within the requirement.

As an alternative, we could potentially make the 10 m cable not have any power lines, thus requiring the use of an external power supply instead, regardless of the appliance. (Because the appliances without an external power supply use 5 V, a USB type B connector on the appliance for power should be sufficient because the voltage is the same)

Finally, to get around the baud rate issue, we could convert the UART signal into a differential signal. The differential signaling should be simple to implement (simply send both the normal and inverted signal on the wire at the same time, and use a differential amplifier on the receiving end to cancel out any noise; see [3]).

We believe that a maximum cable length of 10 m is sufficient for versatility -- since there will be power outlets near remote appliances anyway, power delivery through the transmission wire is less important. Our tolerance analysis indicates that that length of wire is manageable even with the voltage drop.

3 Cost and Schedule

3.1 Cost Analysis

Fixed

Description	Fixed Cost
\$55/hr 8 hr/day 5 days/week 10 weeks 2 people	\$44,000

Variable

Component	Individual cost	Bulk cost
PCB x4 (hub and 3 appliances) (PCBWay)	\$20	\$15
ATMEGA328P-PU x3 (Microcontroller for appliances) (Digikey)	\$8.46	\$7.02
497-4257-1-ND x2 (3.3v regulator) (Digikey)	\$1.86	\$0.82
C 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
Total	\$55.15	\$45.08

3.2 Schedule

Week	Peter	Nagarjun
10/4/2021 Design	Complete E2E Circuit Schematics	Order Parts, Begin PCB Design
10/11/2021 Circuit	Circuit Design	Circuit Design
10/18/2021 Central Hub PCB	Design PCB for central hub	Fix bugs in central hub PCB

10/25/2021 Appliance PCB	Design PCB for each appliance sub system	Fix bugs in appliance PCB
11/1/2021 Connections	Implement UART protocols between central hub and appliances	Implement UART protocols between central hub and appliances
11/8/2021 Mock	Complete design of prototype	Complete design of prototype
11/15/2021 Final	Prepare for final demo and final report	Prepare for final demo and final report

4 Safety and Ethics

One particularly important safety concern here is because we expect a high degree of modularity from the project, there are cases where end users could potentially e.g. put in an IC backwards in its socket or hook up the wrong connectors. This could result in component damage especially if high currents end up in low-current paths. Since low-level modularity is usually expected to be performed by advanced users only, a simple warning about these dangers would suffice, as those users generally understand the consequences of incorrect wiring. In addition, we also have connectors that are meant to be plugged in by non-advanced users. We would theoretically be responsible to some degree for those kinds of incidents. To mitigate this, we could either create a custom connector for the interconnects between the hub and appliances, or we could reuse an existing connector. (USB connectors to the computer are already standardized.) If we choose to reuse an existing connector, we should be careful not to damage devices that use that same connector, where the devices use protocols that are normally associated with that connector (e.g. if we use 6-pin mini-DIN, then we should design the protocol in the connector such that the devices should not be damaged by plugging in e.g. a PS/2 mouse or keyboard. The protocol will very likely be incompatible, but the mouse or keyboard should not be damaged as a result.)

Finally, the switcher could potentially be able to switch high voltages (e.g. 120V AC). Although failure to do so is not critical to the success of our project, we may still need additional training with high voltage if we ever decide to extend our project to switch high voltages [6].

There are also ethical issues with this project. For example, our project is touted to preserve the end user's privacy. In order for this to be relied upon, the end user should be able to verify this independently (i.e. the privacy aspects should not be present solely because we, as the creators of the project, are known to respect people's privacy) [5]. In order to do so, we must be transparent about the designs and make any microcontroller code open source, such that the end user is notified about the actual source code present in the microcontroller [5]. In addition, the number of "mandatory" parts (i.e. the parts that are required for essential functionality of the

appliances to be used in any meaningful way) is minimized, and should not rely on any microcontroller, thus the requirement for the breakout header for the appliances.

5 Citations

- [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].
- [10] Grainger Engineering Office of Marketing and Communications, "Course websites," *The Grainger College of Engineering | UIUC*. [Online]. Available: <https://courses.grainger.illinois.edu/ece439/fa2021/>. [Accessed: 30-Sep-2021].

