# USB Controlled Appliances

## ECE 445 Project Proposal

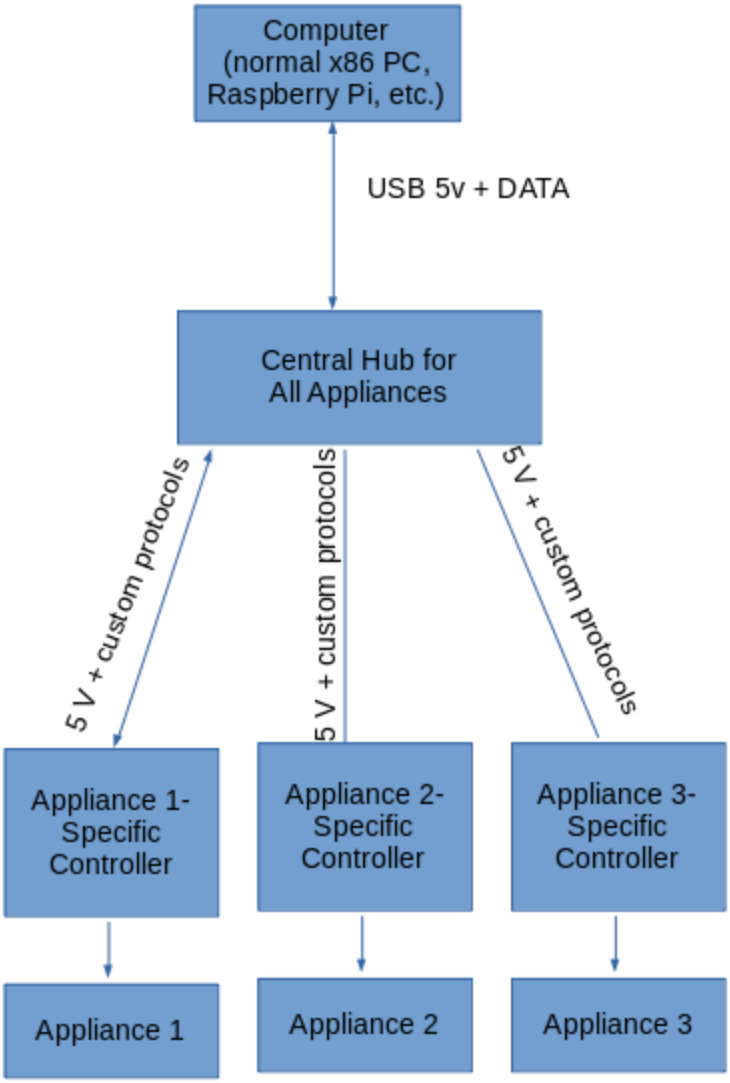
Nagarjun Kumar
Peter Jin
Riley Baker
TA: Dean Bishop
Team 15

**Problem:**
There are many kinds of IoT devices, ranging from smart plugs, thermostats, washers and dryers, garage doors, and refrigerators. However, the main issue with these kinds of devices is that the embedded systems that run on those devices are closed-source, out of date, or simply

have glaring security and privacy issues. Furthermore, since they usually connect via Wi-Fi to a home network, the attack surface of many of those devices is very large since any computer on the network could access it in some way, allowing attackers on the network to potentially control these devices maliciously. Building a smart plug or appliance by yourself also has challenges. For example, some appliances require switching of high voltages, which may not be safe on a breadboard. This effectively makes a “DIY” version of a smart plug that doesn’t connect to the Internet very difficult to create safely.

**Solution:**
Separate the part that connects to the Internet with the part that actually switches the appliance, and connect them with a safe and well-defined wired protocol. This is what our “USB Controlled Appliances” project intends to do. Instead of connecting to the Internet via Wi-Fi, this “smart plug” connects to a computer via USB. In this way, the computer can be used to control the smart plug, without inherently relying on wireless network protocols. Since this smart plug does not have any Wi-Fi capabilities, the attack surface will be greatly reduced.

**Visual Aid:**




Appliances may either be powered by the hub, which can either be powered by the computer or by an external power supply. Each of the appliances can also have their own power supply, depending on the power requirements of the appliance.

**High-level requirements list:**

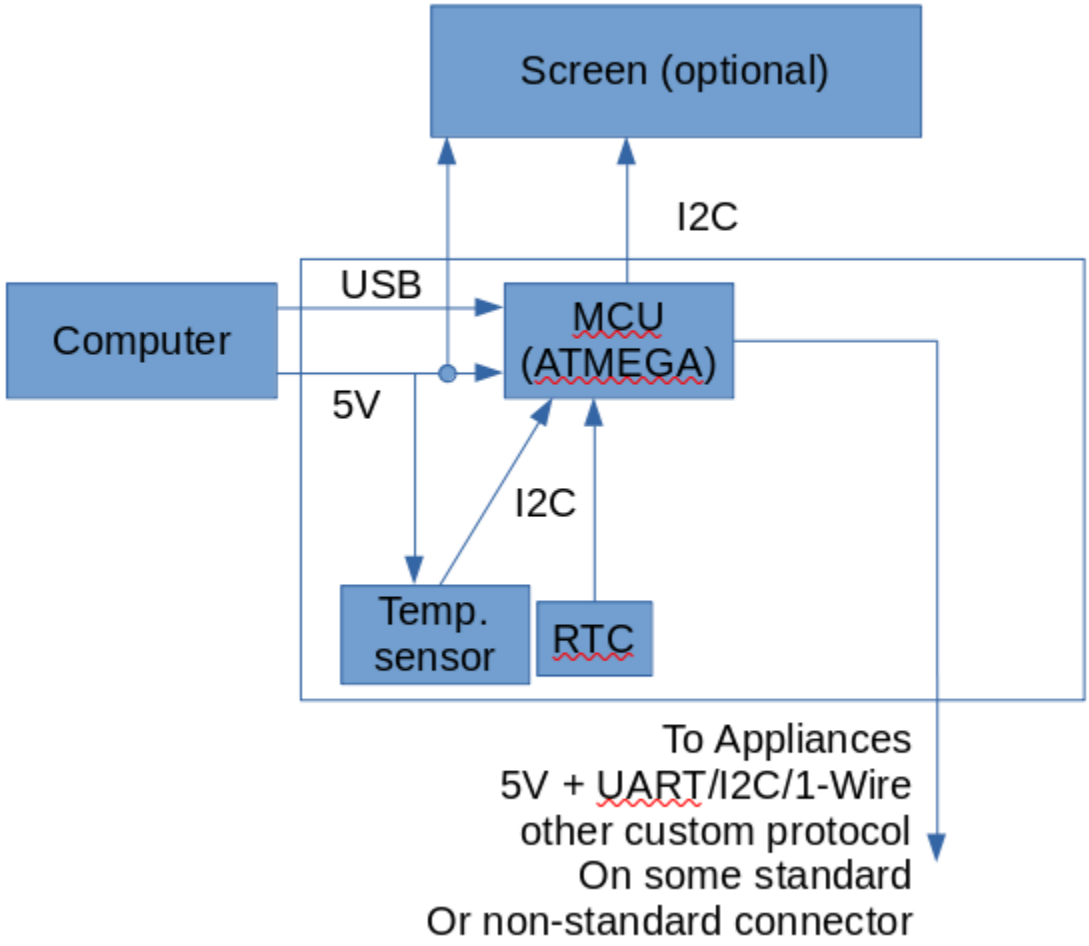
- The **central hub** must connect to the computer via USB and must also connect to each of the appliances. The central hub will relieve us from having to write a USB protocol for each appliance.
- The **computer subsystem** with a USB port is essential for this project as it is needed to run the software and also to act as a bastion host for the appliances.

- Design about 3-4 “smart” appliances that should not have any complex logic in them. Nothing wireless can be allowed and if the appliance uses a microcontroller, it should be able to be removed from the end circuit.

Block Diagram:

Voltage regulators are not shown unless the output is broken out. If a device requires 3.3v, a standard switching or linear regulator is implied.

For computer and hub:

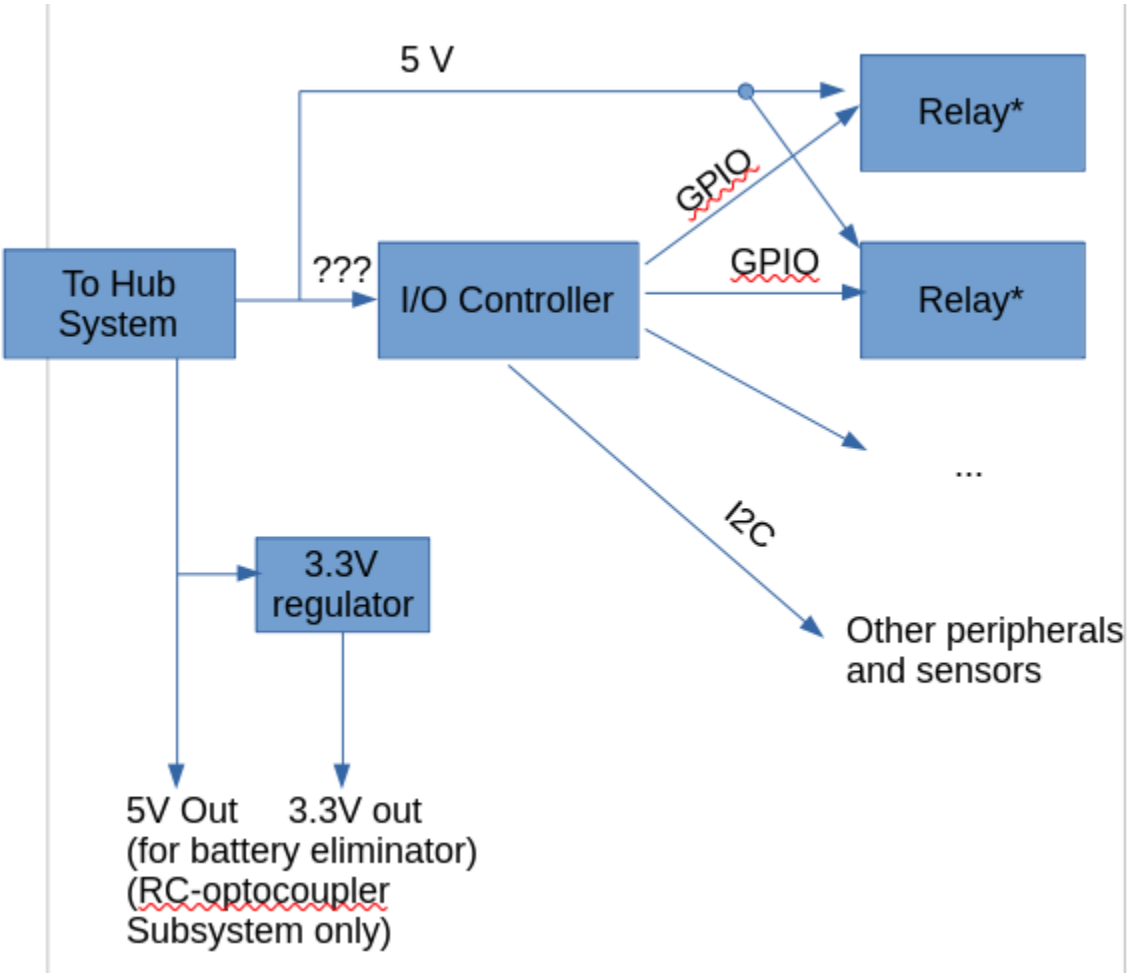



This subsystem is used to interface the simple protocols used by the appliances with the USB on the computer. There are a few other chips, mostly to provide auxiliary logic to the microcontroller where it is required (e.g. temperature sensor for a thermostat appliance, and an RTC to support schedules (e.g. to turn on and off an appliance at a certain point in time))

Requirements for the hub subsystem are as follows:

- Hub must connect to the computer via USB, as that is a universal protocol found on nearly all computers.
- Temperature sensor must read the temperature correctly from -20 to 50 degrees Celsius with 1 degree accuracy.
- RTC must keep time within a 1 second per day accuracy. (The software on the computer may have a means of synchronizing this RTC to the computer's clock (which in turn may be synchronized to an external NTP server -- details regarding that are out of scope).)

Example appliance controller system for a generic appliance:




The protocol that connects the hub subsystem to the I/O controller may vary; it is a decision mainly determined by the hub system.

The battery eliminator for the RC-optocoupler subsystem is intended for the use case of using optocouplers to simulate pressing buttons on a remote control. Since the remote control is now

hard-wired, it might be useful to provide power to the remote control directly instead of using batteries.

(*) “Relay” is just an example output device. It could also be an optocoupler, a solenoid, or even an input device to provide feedback. In the case of powering a relay or solenoid, an additional drive transistor may be required. Additional power supplies may also be required, depending on the current and voltage requirements of the appliance itself. The relay and optocoupler are hooked up in a way that controls the end appliance by opening and closing the contacts.

Requirements for appliance controller subsystems:

- The general concept of the appliance controller must be reasonably extendible to various kinds of appliances -- optocoupler switcher, door lock, motion detector, thermostat, and generic switches.
- Where appropriate, an appliance may break out 5V, 3.3V, and/or other voltages for the convenience of being able to eliminate the battery in a remote control. Voltage tolerances are generally 5% or better for output currents from 0 up to 100 mA (ideally up to 500 mA).
- The GPIO pins that connect the appliances should be broken out onto some kind of header wherever possible, in case an advanced end user would like to bypass the logic entirely.
- [For the motion detector] The motion detector should be able to detect large objects (human sized) which are moving in the background.
- [For the optocoupler switcher] The optocoupler switcher should be able to press various buttons on remote controls in a generic manner, and have an interface to connect wires from the optocouplers to the remote control.
- [If applicable for the type of appliance] The I/O controller must be designed such that it will never keep a switch on for too long, in case the hub or computer stops responding.
- If the appliance requires external power supplies, then they must be safe to use with the appliance (i.e. have correct isolation/grounding and voltage, within a 5% tolerance of the expected voltage).

Tolerance analysis:

Our USB Controlled Appliances project was designed to be very robust, such that the failure of one part of the system does not catastrophically cause the other parts to fail. For example, if the central hub fails, then the appliances can still be hooked up directly to an alternative hub, or can be wired to some other custom logic. That being said, within the appliances themselves, the main source of failure is the microcontroller. Since the microcontroller will be able to be removed by the end user, this part can still be easily replaced and reprogrammed without any negative impact on functionality.

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.

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). 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. 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.

A different ethical issue is if the optocoupler switcher is used with something like an i-clicker, and the i-clicker is used in an irresponsible manner. While we can't necessarily account for every abuse of our product, we should also at least not facilitate these kinds of abuse. The optocoupler switcher is still for advanced users only, and the end user will still be notified upon receiving the product that they should only use the remote control switcher in a responsible manner.