University of Notre Dame

High Level Design
Infrared Remote Control

Group 11
Collin Finnan, Adrienne Niewiadomski, Nick Osborne, Elizabeth Gonzalez, Kara Garvey
Senior Design, EE 41430
Professor Mike Schafer
December 8, 2022

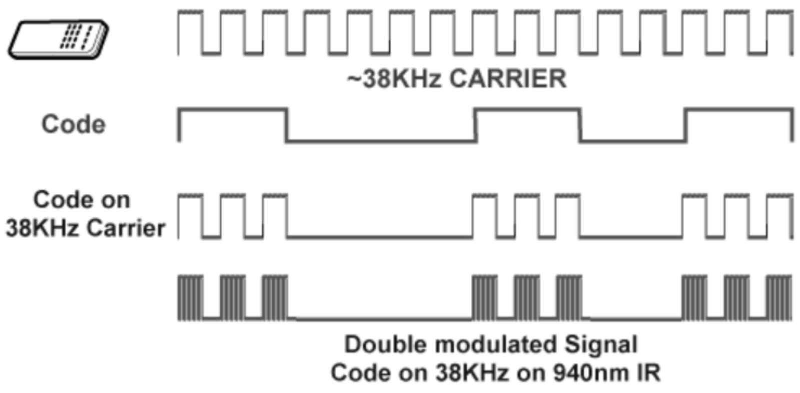
Table of Contents

1. – Introduction:

One need not think hard to find issues with their current remote control set up. Oftentimes there are too many remotes for different things, the remotes do not have intuitive hot-keys, the connection is too slow or doesn't have sufficient range, etc. By investigating the fundamental properties guiding a remote control, and building on these properties to create an advanced system, we will provide users with the ability to customize their own remote via an application and / or website that is functional for them specifically.

To begin, we need a brief overview of how a TV remote works. The remote is essentially a transmitter, sending infrared signals to a receiver on the TV. The infrared light is generated via an LED (or more correctly, an IRED since we cannot see the light) with a wavelength that corresponds to that required by the receiver (typically ~940 nm). The LED blinks on and off rapidly, creating binary data that the TV receives and interprets. Because infrared noise exists everywhere (sunlight, body heat, etc.), the IR signal must be modulated to a specific frequency set by the receiver, so that noise is filtered and the signal is amplified. This modulation will be in the range of 32-40 kHz typically (source 1).

Figure 1. Basics of Remote Control Signal



But how do we generate these signals? We can blink the LED on and off to make nice square waves like this, but how do we know which "code" to make? The answer is unsatisfying, but essentially it depends on the TV being used. For example, many IR receivers use a standardized system like NEC, while others like Samsung TVs have decided to make their own protocol. Luckily, these protocols are quite similar, and both transmitting and receiving are supported within the Arduino architecture (source 2).

We've now introduced the general concepts behind how a TV remote works. We have not, however, gotten into any of the nuances we plan to add to make this primitive model superior.

Source 1: https://www.laser.com/dhouston/ir-rf_fundamentals.html

Source 2:
http://techiesms.blogspot.com/2016/01/ir-protocol-decoding-and-transmiting.html#:~:text=NEC%20protocol%20is%20most%20commonly,mark%20and%204.5%20ms%20space.

2a. – Problem Descriptions:

The problems plaguing today's TV remotes have only been accelerated by the advent of smart TVs, where new user freedom has overwhelmed the aged model for controlling the system. The problem description is best seen as a list that will correspond to a secondary solution list in the next section. These are high level complaints that most people find themselves saying when operating a remote.

1. Missing remote –
    - There is nothing more frustrating than finishing all your work, sitting down on a comfortable couch, placing your food and drinks on the coffee table in front of you, only to discover the remote is nowhere to be found. Addressing this problem is paramount.
2. Effective range –
    - Although less prevalent in modern remotes, there is always the problem of having your buttons actually transmit. When the battery is low or the angle is just wrong, it can be frustrating to get the TV to do what you want it to.
    - Maybe you want to pause something while outside the room because the task is taking longer than expected. This is currently not possible.
3. Battery lifetime –
    - Transitioning from the previous point, one could also find themselves upset with the battery lifetime of remotes. AAs are used for packaging purposes, but they get used up fairly quickly.
    - There is also no built-in indicator for battery life, so it always comes as a complete surprise.
4. Hot-keys –
    - Today's remotes often come with hot-keys. Want Netflix to pull up right away? Click the Netflix button. But what if you want HBO? Or YoutubeTV? Sure there are ways of programming the remote to pin these to other, unmarked hot-keys, but wouldn't it be nice if I could customize the layout of the remote myself?
    - What if I want a simplistic remote for grandma, which has 4 buttons only, whereas I would like more options with several? Without hiring a universal remote guy to come in, this is impossible.
5. Access to control –
    - There is typically a single TV remote for a whole room. Want to put something on? "Pass the remote!". But tossing the remote across the room could damage the devices inside, and then you're completely out of luck. Fundamentally, you could have as many remotes as you wanted, so long as they weren't being used at the same time. Why limit yourself to having one person in control?
6. External Devices –

- Universal remotes were the fad of the early 2000s, but with new devices cropping up every month, the remote you just paid hundreds of dollars to have set up would become antiquated in less than a year.
- But the idea behind them is still solid. Why have a remote for your speaker and for your TV? Why not put it all in one place? Is there a way to do this that is Easily adjustable for new devices?

2b. – Proposed Solution:

The proposed solutions all revolve around a single idea: a virtual TV remote, controlled by the user via a website. Immediately, people will be hesitant to this solution. Why complicate everything with an app? But the beauty of TV remotes is in their simplicity: they are just transmitters. If you have someone over that doesn't want to go to a website or download an app, just hand them the remote! It will still work. In this solution, a PCB with a microcontroller and transmitter circuit will be mounted near the receiver. The microcontroller will have a codebase of various TV commands that generate signals corresponding to the TV's protocol. A front end for users will be built upon these lower-level operations. The issues provided in the previous section will be addressed with this solution as follows:

1. Missing remote –
    - The transmitter itself will be mounted, and the remote becomes your phone. No one ever loses their phone.
    - The physical remote that comes with the TV may get lost, but the set up we are providing only fails when you misplace your phone or move the PCB.

2. Effective range –
    - Once again, the transmitter is mounted, so it will be perpetually in front of the receiver (but with enough space open so that the actual TV default remote can still be used).
    - But if I leave the room, my phone is still on the WiFi, meaning I can send signals to the TV from anywhere in the house! Go upstairs and forget to turn off the TV? That's okay! Just refresh the website and press the power button.
    - The packaging for this device will have some sort of easy-clip that can attach to a standard, thin TV along the bottom. The only issue will be helping users identify where the TV receiver is for installation.

3. Battery lifetime –
    - This solution can go a couple ways, so we believe our PCB should give the user both options.
    - First, the device could simply plug in. It does not have to be mobile, like an actual remote, and because the TV is itself near an outlet, so too this device could be externally powered, removing the battery problem.
    - But this introduces a cable which may be unattractive to the user, so as a second option, we provide the following: a larger battery space. The setup is not limited by size like a handheld remote, so we could put several additional batteries in parallel, resulting in a longer lifetime.

    - An LED could be added on the PCB to indicate low battery if the second option were selected. This would address the lack of power indication in modern remotes. Alternatively, a power indicator on the user-interface could also be presented. This second solution requires less power consumption and seems more intuitive.

4. Hot-keys –
    - This is where the beauty of the idea comes in. Because there is not a physical remote in this design, the layout is completely up to the user.
    - Some sort of prompt could be provided which asks the user what features they want on their remote. Move "arrow" keys, on / off buttons, and volume would be default, but other buttons would be dependent on input from users. Only use your TV for Netflix? Make a remote with less than 10 buttons and have netflix take up half the board. Need everything on your remote? Have 200 buttons.
    - A GUI could even be devised that allows users to click and drag where they want the buttons to be, so not only the hot-keys were custom but the layout itself.
    - Clicking a button on the app / site would send an HTTP signal via WiFi to the microcontroller. The microcontroller converts this signal to the given protocol and actuates it via a GPIO pin to the LED subsystem.

5. Access to Control –
    - Because it all occurs via WiFi, multiple users can send signals using different devices, eliminating the need to pass the remote or find it when it is lost.
    - To avoid issues with multiple users sending, there could be a ranked queue based on who joins / connects with the CPU first, and it is reset every time the TV turns on and off.
    - Some sort of methodology would need to be implemented that "remembers" users so that home-users do not need to reconnect / go through a long process each time they boot up their TV.

6. External Devices –
    - The icing on top is that because the TV is now controlled via a general purpose microcontroller, we have the freedom to extend the functionality of our application / website beyond IR protocols.
    - If we have a speaker that connects via bluetooth, we could add a feature to the program that has a bluetooth button. All it would require is additional libraries for whatever device types we want the user to be able to connect with.

---

3. – System Requirements:

The system requirements section has been broken into three sections: functional requirements, user-interface requirements, and safety requirements. The functionality requirements refer to what the system has to do and has nothing to do with the nitty gritty details of how the system actually works. These are reserved for the user interface requirements, which detail the specific needs of the way humans interact with the device. These requirements are more high-level, and assume that the basic functionality is there. Finally, there are safety requirements ensuring the device's production and usage isn't harmful (spoiler, for our project these are limited).

Functionality Requirements

1. Control a TV using the ESP32
   - The most fundamental and basic requirement of the project. A GPIO pin on the ESP32 will output to an IRED subsystem (amplifier + emitting diode, likely).
   - We need to use arduino libraries (reference: https://www.arduino.cc/reference/en/libraries/irremote/), which is compatible with the ESP32, to send codes to a TV receiver.
   - We can prototype this using a breadboard so that work can begin before the PCB is printed and ready.
   - Without being able to do this, the project does not go forward.

2. Control different TYPES of TVs
   - The product is useless if it only works with one type of television. One demonstrated feature is that this device must be quasi-universal.
   - The hardware is universal intrinsically: we just have an IRED subsystem, a microcontroller, and a power supply system.
   - The arduino library has robust support for different protocols: Denon / Sharp, JVC, LG / LG2, NEC / Onkyo / Apple, Panasonic / Kaseikyo, RC5, RC6, Samsung, Sony, (Pronto), BoseWave, Lego, Whynter, MagiQuest.New: Added NEC2 protocol.
   - The only limiting factor: what we do as programmers to implement different versions of code using these different available code bases.

3. Control Via WiFi Based Website
   - The ingenuity of the product comes from the ability to control it via a phone. A functionality requirement must then be that those GPIO pins sending signals are controlled by phones.
   - Our proposed method of doing this is a basic website being published by the ESP32 getting responses from users on their phones.
   - This means the range of the device is simply the range of the WiFi. No one is going to use a TV remote outside their home, so this makes sense for the implementation of the product.
   - The addition of a website means we will want a dual core microcontroller. One core works with the user-interface while the other broadcasts IR signals.

4. Control of External Device
   - Though this is the lowest priority, the proposed solution has immense freedom allowing for the development of universality. Because our ESP32 selection has bluetooth and WiFi, we could easily include support for external devices other than the TV.
   - This functional requirement essentially is a proof of concept: show this solution allows for easier universal remote capabilities than traditional implementations.

    - Bluetooth devices often stream data (music, for example). This may require a third core! Perhaps finding a device like an LED strip or lights system would be better, since they would not require a separate core for constant streaming.

5. Memory
    - The code base for this device may be quite large. When we boot up the system, we could either download the memory from an online library or just have it on the board. Depending on the size, we may have to go with the former.
    - Additionally, there will be saved layouts for returning users. If the device loses power, we don't want these users to have to remake their remote! Therefore, we should have some way of saving user data to an external cloud via WiFi, and additionally download code bases from the same cloud using WiFi.

User Interface Requirements

1. Power
    - Rather than have an additional power cable, the device will be battery powered.
    - Unlike a normal remote, we are not limited by size, so we can throw some additional battery packs onto the PCB. But we do want the design to stay relatively small so no need to go overboard.
    - The ESP32 requires 3.3 V to operate. Likely what we will do is have the batteries and lead them to a regulator that outputs 3.3 V. The input power we've been using this semester is ~5 V from a USB. We will use multiple 9 V batteries in parallel to yield a longer battery life and sufficiently power the regulator.
    - An indicator on the website will report to the user the battery power. Because of this, one of the pins on the ESP32 will need to be an analog input that reports the current value of the battery. When it starts to "droop", the website will alert the user.
    - There should also be a physical power switch. Typically, we propose the device will just always be on, hence the several batteries in parallel. The ESP32 can be put in an idle power-saving mode which will reduce consumption, but it would be a hassle for users to constantly have to turn a switch on and off. But, we want that option available (say the family goes for a long trip).

2. Physical Design
    - So far, everything we've discussed could be done on a breadboard. But this is not very durable or elegant. We will make a PCB for our final design.
    - The PCB will have an IRED that must overhang the edge so it is visible.
    - A 3D printed, rectangular enclosure will surround the PCB and batteries. The majority of the space will be the power supply, meaning the enclosure will be the same size as the number of batteries that we choose to use.
    - The 3D printed enclosure must have a hole that allows the IRED to poke through.
    - The size of the enclosure and PCB must be such that a clip (as seen below) can hold it and attach to the TV.

    - The consequence of this size restriction is that we will likely use 3-4 9V batteries in total. Estimated size will be slightly larger than a hand-held remote, but not by much.

3. Website
    - Quite simply, the website has to be flawless, otherwise this alternative is not worthy of being selected. Therefore, the website needs to be attractive and easy to use.
    - The website needs to be able to build profiles. If there is a new user, there needs to be a flow which walks users through which hotkeys they want, where they want to put them, what size they want them, etc. The output will be a GUI of their personalized remote. This will be saved under their name.
    - The website needs to be able to remember profiles. As discussed in the functional requirements, if someone lives in the home and wants to have their personal remote booted up, the website needs to remember that, otherwise this device becomes a hassle.
    - The website needs to allow multiple users at once. A priority queue of sorts will be made which values one user's input above others so interference isn't allowed. This would only arise when both users try to input at the same time.
    - The website needs to be secure. Although WiFi is already a barrier of entry, we also want some sort of password for the remote so that not all guests can control the TV.
    - The website needs an intuitive bootup. When we first install the ESP32, it won't be on the WiFi. We need an initial way in which the first user enters the network and yields the password. Then, the standard bootup will show two options: "load existing remote, create new remote".
    - Potential idea: remember devices? It may be possible for the ESP32 to recognize a certain person's device via the HTTP request (it is not secure so …?). This is a half-baked idea in its current form, but it would allow a user to avoid the bootup screen and just log onto the website with a remote already connected.

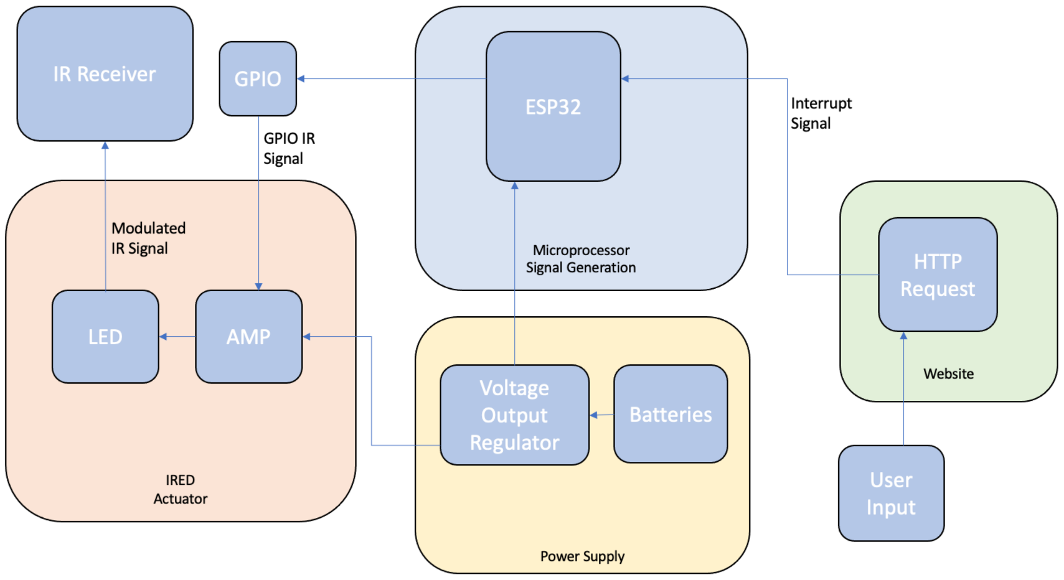
Safety Requirements

- This is a low-power device, so there are no dangerous currents or voltages in the development or operation of the device.
- There are no dangerous pieces of equipment nor machines needed to create this product.
- Lead-based solder may be the only dangerous part of this project.

4. – System Block Diagram:

4.1 – Overall System:

Figure 2: Overall System Block Diagram



4.2 – IRED Actuator

- The IRED needs to be within an appropriate wavelength to properly send the signal to the TV receiver. (~940 nm)
- It needs to have a high current tolerance in order to achieve a higher output signal.
- INPUTS: Power for the amplifier and a GPIO signal driving the amplifier.
- OUTPUT: Modulated IR signal.

4.3 – Power Supply

- The power supply has to provide at least 3.3 V of power to run the ESP32.
- Physically there needs to be a safe and stable container to hold the 9 V batteries within that isn't too obstructing as it would defeat the purpose of getting rid of the hindrance of a cable for power.
- Power must be monitorable.
- INPUT: 9V batteries, oriented in some form of rack external to the PCB leading to the regulator.
- OUTPUT: Regulated 3.3 V power for the whole board (ESP32 and amplifier for IRED subsystem).

4.4 – Microprocessor Signal Generation

- It must take in the HTTP information given by the website and send the appropriate corresponding IR signal to the IRED actuator.

- Analogous to the LED-WiFi individual coding assignment: this system is mostly interrupt / reaction based code. Something from the website loop triggers an action: “Increase the volume”! The job of this subsystem is to react to that command by drawing on the stored codebase and converting the HTTP command into an output on the GPIO.
- INPUT: HTTP request /interrupts from website.
- OUTPUT: GPIO IR signal for given protocol and required data.

4.5 – Website

- The website needs to have a clean and easy to understand user interface.
- There needs to be a clear way to create a new profile or choose an existing one.
- The website needs to be able to remember or access the information of previous users in order to allow for quick access to already designed user remote layouts.
- It needs an input queue in order to prioritize different users in order.
- There must be a password of some sort in order to secure the site from outside users not intended to be using the TV.
- It must provide user customization that appears in a clean button layout.
- These required features of the website essentially provide the tool in which users generate high-level signals like “volume up”. These signals manifest as interrupts which get sent to the previous stage.
- INPUT: user button pushes on website GUI.
- OUTPUT: interrupt signals that go to the ESP32.

4.6 – Future Enhancement

- More universality with other TV types
- The inclusion of other physical remote controls other than the phone/website interface that would allow for connection with the same signal generation system.

---

5. – High Level Design Decisions:

This section is guided by the requirements section, and will therefore also be organized into the three main sections: functional requirements (5), user-interface requirements (3), and safety requirements (1, kind of). Based on these needs, we will make decisions about how our project will be actuated. Because there is a lot of repetition from sections 3 and 4, this section will be shorter / more condensed.

Functional Decisions Regarding …

1. Control a TV using the ESP32:
    - We need to pick a library to do this. Since we are in the arduino framework, it needs to be ESP32 friendly and arduino based.
    - Our decision: the IRRemote library through arduino.
    - Link: https://www.arduino.cc/reference/en/libraries/irremote/
2. Control different TYPES of TVs:
    - Our previous choice of library supports this requirement.

   - This requirement demands different codebases from our end of the project. A Samsung TV versus an Apple TV may have its protocol's abstracted by the library, but we still need to choose the data being sent.
   - Our decision: limit the scope of the project to 2 types of TVs, since IR data sent will be different for different TVs.
3. Control Via WiFi Based Website:
   - Decision: Use ESP32 with WiFi and dual core so the website can run concurrently with signals being sent.
4. Control of External Device:
   - We discussed that a bluetooth device may require constant streaming, and therefore another core. We want a device that can just take a single command at a time, and is potentially WiFi connectable.
   - Our decision: control some sort of external lighting device (simple on / off commands) to prove universality.
5. Memory:
   - As previously mentioned, user data is paramount for custom remotes. The ESP32-WROOM we plan to use has onboard memory, though it may not be enough.
   - Decision: try to store codebase and userdata onboard. If this fails, store the data in a designated cloud and download when needed. This cloud will be a simple file system accessed through WiFi (maybe as simple as downloading a .txt file from google drive!).

User-Interface Decisions Regarding ...

1. Power:
   - Without knowing the model of TV and what available USB ports it has, it is hard to justify going with anything other than battery power. We have no size constraints so we can load a ton of batteries in parallel for longer life.
   - Decision 1: use battery power.
   - Additionally we want to monitor this power. This could be as simple as measuring the voltage of the batteries as an analog input. The ESP32 doesn't allow large DC voltages as inputs, but a simple voltage-divider solves the problem.
   - Decision 2: Connect the power supply to a voltage-divider whose output is monitored by an ESP32 pin, and harness the ADC on the ESP32 to notify use of droops in power.
2. Physical Design:
   - Decision 1: Size is larger (due to extra batteries), but roughly comparable to standard remote. Batteries will take up the majority of the design space.
   - Decision 2: Enclosure will be 3D printed, as this is the easy way to fit non-standard dimensions. A rectangle will do for this project.
   - Decision 3: Separate the batteries from the PCB. A battery holder like this one will be wired to the PCB. This will reduce the size of the PCB and the risk of damaging the PCB while swapping batteries. An example image is shown below of a battery carrier (this is for AA but we will find one for 9V.)

3. Website:
    - Decision 1: We need a welcome page that prompts whether you are an existing user (so it would boot up using existing user data) or a new user (so it would save and then use this user data as a new profile).
    - Decision 2: Customization occurs through checklist prompt, not a click and drag GUI. You get to pick what buttons you want, and the relative size of those buttons / relative location, but the click and drag GUI approach would be too hard. Simplistic customization is best.
    - Decision 3: One webpage. Rather than redirect to multiple different pages when navigating between a welcome page and the actual remote, the ESP32 will just “clear” the original page and update with the remote. This means that when a second user joins, they will automatically join on the remote of the first user, and be dropped to a lower priority.
    - Decision 4: A log off button will return the page to the welcome prompt for the next user.

There are no explicit safety inspired requirements, and therefore no decision regarding safety. This is a very safe project.

---

6. - Open Questions:

This is, generally speaking, a well documented project. Lots of people have made remotes. The difficulty comes from two distinct areas that we have addressed below:

1. *Propriety*. Different TVs have different protocols, and these are handled by the arduino library we selected. But to ping “volume up”, you need to send specific data to the TV. It is not just the format of this data that is specific to the TV, but the content of it. Is this information available for actual TVs? Is it propriety? How would we find this information?
2. *Website.* As EE majors, we do not have a ton of website knowledge. While the implementation seems straightforward from the outside, our ignorance of webdev could mean that this project is impossibly hard. This is especially true given the website is going to be implemented using HTML and runs off of a microcontroller (rather than a 3rd party site that abstracts a lot of the hard stuff for us). The website is so critical to the project, but could be the bottleneck.

7. – Major Component Costs:

PCB - $10-50
https://blog.svtronics.com/how-much-does-a-pcb-cost-factors-that-will-affect-the-price/

3-4 9 V batteries - ~$6-10
Home Depot 9V Batteries

ESP32-WROVER-E-N4R8, dual-core, wifi, bluetooth - $3.3/unit
Digikey Dual-core

9v battery holder/case (preferably cased; preferably multiple; up to 4; can connect together) - $2-7
Vetco Dual 9V Battery Connector
Amazon Battery Holder

IRED - $1.81/unit
Digikey IRED

Amplifier - $2.70
Digikey Amp

8. – Conclusions:

The modern TV deserves a modern remote. Creating a fully customizable experience via a smartphone will eliminate several of the problems currently existing in the remote control space. By having the transmitter fixed and communicating via personal devices, traditional issues like lost remotes, low battery, accessibility, and general interfacing will vanish. To implement will require intelligent PCB design using heavily researched components, thorough library building for effective communication between microcontroller and TV, and flawless high level design for a near perfect UI. The end result will be a completely redesigned remote control that has aspects needed for the progression of television watching.

References:

1. Arduino Library
2. Infrared TV Remote Background
3. See DigiKey references in the previous section.