Team Antiprocrastinator Final Report

By
Kyle Chiu
Taylor Plummer
Brandon Wong

Final Report for ECE 445, Senior Design, Spring 2022
TA: Zhicong Fan

4 May 2022
Project No.19

Abstract

The Antiprocrastinator is a device that is able to connect to a computer via bluetooth and detect websites and applications that are running on it. Users can choose certain websites/applications to be blacklisted. Should any of these blacklisted items be run and detected, the device will emit an annoying buzzing sound to deter users from staying on those sites. Similarly, should the user try to leave the computer, the device will also be able to detect that, and will emit the noise as well in that circumstance. This buzzing noise will continue, if the behavior persists, until 3 minutes have passed, where it will wait another 3 minutes before buzzing again in order to conserve power and sanity. This interval pattern will proceed until the undesired behavior ceases.

Table of Contents

1 Introduction

Problem

Procrastination has been a huge hurdle in the lives of many students and workers. While some people have the self-discipline to get work done, many people like our group aren't great at staying focused. Some of our members have participated in procrastination and have also had firsthand experience of the consequences of procrastination (e.g. all-nighters, lower grades). Many procrastinators like ourselves push school and work to the side and instead pursue various forms of entertainment, such as computer games or internet videos, in order to avoid doing work. This forces procrastinators to put off their assignments until the deadline, to which they are forced to cram a few days' worth of work into a few hours, which consequently means that the procrastinator will be physically and mentally exhausted, and the work completed will be of mediocre quality at best.

Solution

The solution is to make procrastinating harder. Specifically, we will be hindering the access of any websites or computer programs that the user has blacklisted. There are browser extensions that can block websites, but those just leave the user waiting in anticipation until their websites are unblocked when the timer ends. Our goal is to use a wireless, battery-powered desk device with an integrated buzzer that, upon receiving a signal from the computer that the user is using blacklisted sites/programs, continuously plays an annoying sound until said sites/programs are closed. This desk device consists of a few components in order for it to work:

- Computer software that detects what programs are being run and what websites are currently open
- A Bluetooth module that will receive signals from the user's computer
- A sound-emitting device (i.e. a piezo buzzer)
- A small microcontroller that handles the signals from the Bluetooth module
- Power circuit for handling the different voltages and charging

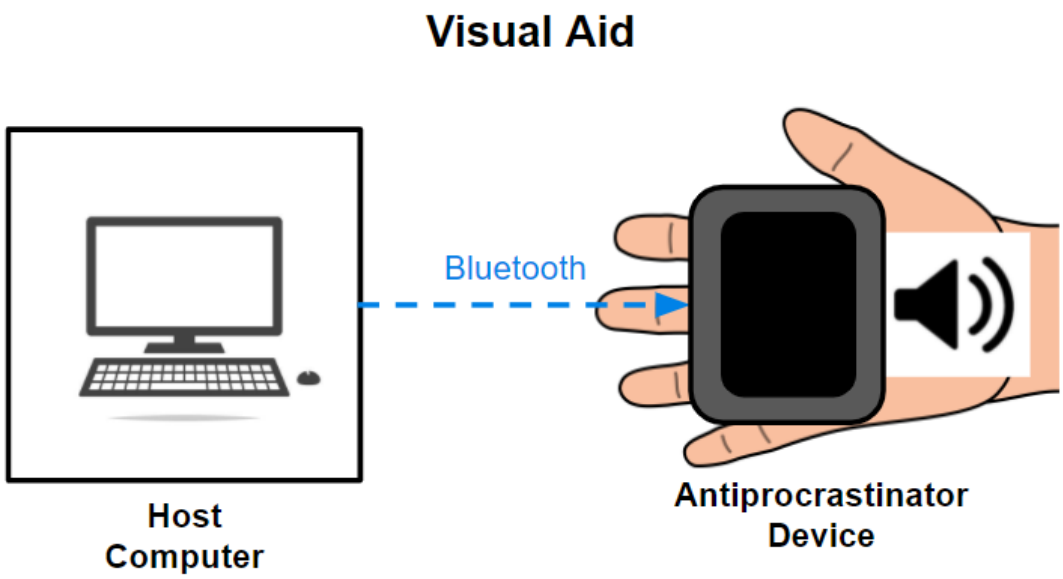

Figure 1: Visual Aid

High-Level Requirements

- The device must be able to connect to the user's computer over bluetooth.
- The device must be able to detect the presence of the user sitting in a chair close to the desk.
- The device must be able to emit a noticeable noise (40-65 decibels) to deter the users behavior when banned apps/websites are open or if the user leaves the desk for too long.

2 Design

2.1 Design Procedure

Based on our final idea of the antiprocrastinator desk device, we knew that we needed BlueTooth connectivity, a microcontroller, a buzzer/speaker to emit noise, a charging circuit to handle LiPo charging, and an IR sensor.

For BlueTooth, originally a TA informed us that while we could go with an ESP32, it is a bit overkill for our project. So we took some time exploring other options, mainly QFP BlueTooth chips that we could integrate into our device. But that proved to be extremely complex, since that required difficult soldering only possible with solder paste and a reflow oven, as well as a lot of other components like antennas that would have made the PCB design a lot more complicated. So we ended up going with the ESP32 in order to avoid all of these things. Since the ESP32 has an onboard microcontroller, that took care of that part as well. The ESP32 needs 3.3v, so we just took a 3.3v voltage regulator and regulated the battery voltage to 3.3v for the ESP32.

The buzzer/speaker was simple. While a speaker is more versatile and can play more sounds, it isn't the best when it comes to playing a tone, and since we only needed the device to be annoying and play a tone, the speaker was unnecessary, so we went with the buzzer which was great for playing tones which we can generate from the ESP32.

For the charging circuit, we knew that we could find an IC that could take care of charging our 3.7v LiPo battery for us. We settled on the MCP73831T-2ACI/OT since it was available at the time and the circuit diagram in the datasheet was fairly simple and straightforward to implement onto a PCB. The charging circuit can be seen in Figure 5.

For the IR sensor, it actually needs 5v to run, so we needed to supply it with 5 volts. After some research, we discovered that we could boost the battery voltage to 5v using a step up boost converter IC, and so we went with the UM3429S IC since it had adjustable output voltage simply by using 2 resistors and the rest of the circuit in the datasheet was pretty straightforward, so it would also be easy to implement onto our PCB. The boost converter circuit can be seen in Figure 5.

2.2 Design Details

For the Power Subsystem, we knew that we will be using a 3.7v LiPo battery, but we needed 3.3v for the ESP32 and 5v for the IR sensor, and a charging circuit to handle charging. As previously mentioned, rather than trying to design a charging or boost circuit by ourselves, it would be in everyone's best interest to just use a premade IC designed by professionals. So for those circuits, we went off of the recommended implementation in the datasheets of the ICs. And for 3.3v, we just used a standard heatsink through-hole 3.3v voltage regulator to convert our battery voltage to 3.3v.

For the Control Subsystem, we knew that we had to connect the IR sensor, the piezo buzzer, and BlueTooth the ESP32, so we connected those pins to the ESP32's GPIO pins. We knew that the IR sensor and the piezo buzzer both needed analog inputs/outputs, so we had to make sure that the GPIO pins we were using supported analog I/O. But otherwise there wasn't much else to connect the ESP32 to.

For the Output Subsystem, it just consists of the piezo buzzer connected to the ESP32's analog output so it really just plays whatever sound wave that comes out of the analog output of the ESP32.

Lastly, for the User Interface Subsystem, we knew that we needed to detect the current active tab as well as any running applications. We decided that the best way to detect tabs was to use a Chrome extension which would have direct access to what tabs are running. But for applications, browser extensions don't have access to the list of running applications, so we need a desktop application running in addition to browser extension. In order to get browser data to the desktop application, we decided that the best way to connect the two would be to also run a Express.js server in the background. This way the browser extension could ping the backend server and the desktop application could read from the backend server.

Block Diagram

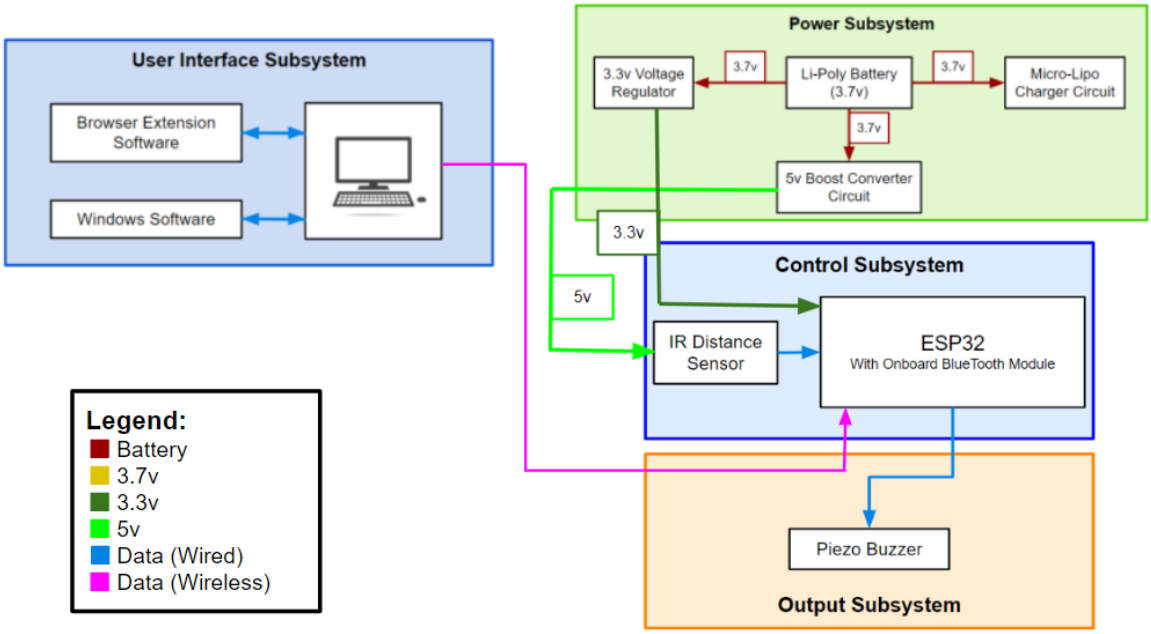

Figure 2: Block Diagram

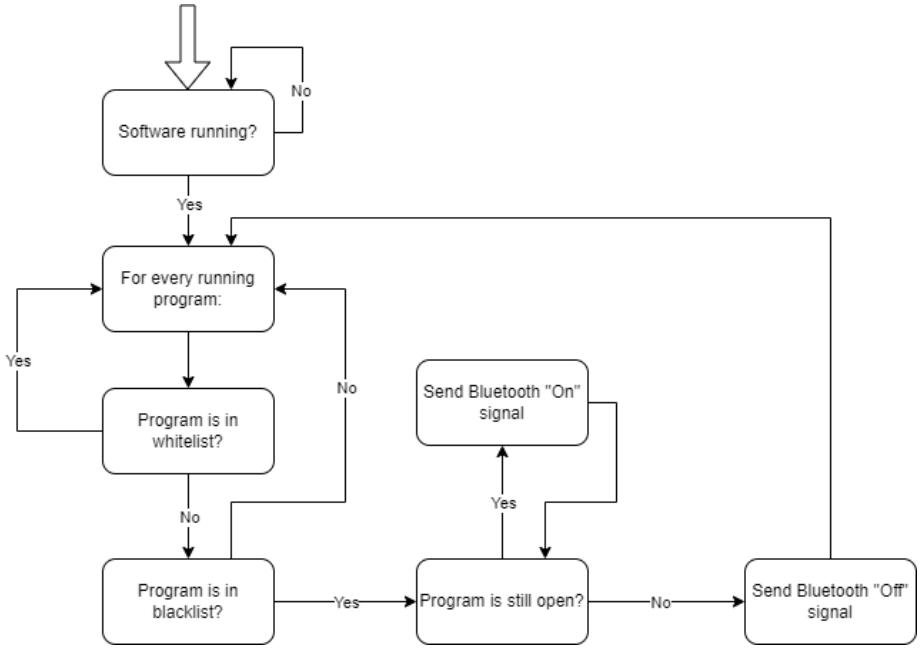

Figure 3: Windows Software Flowchart

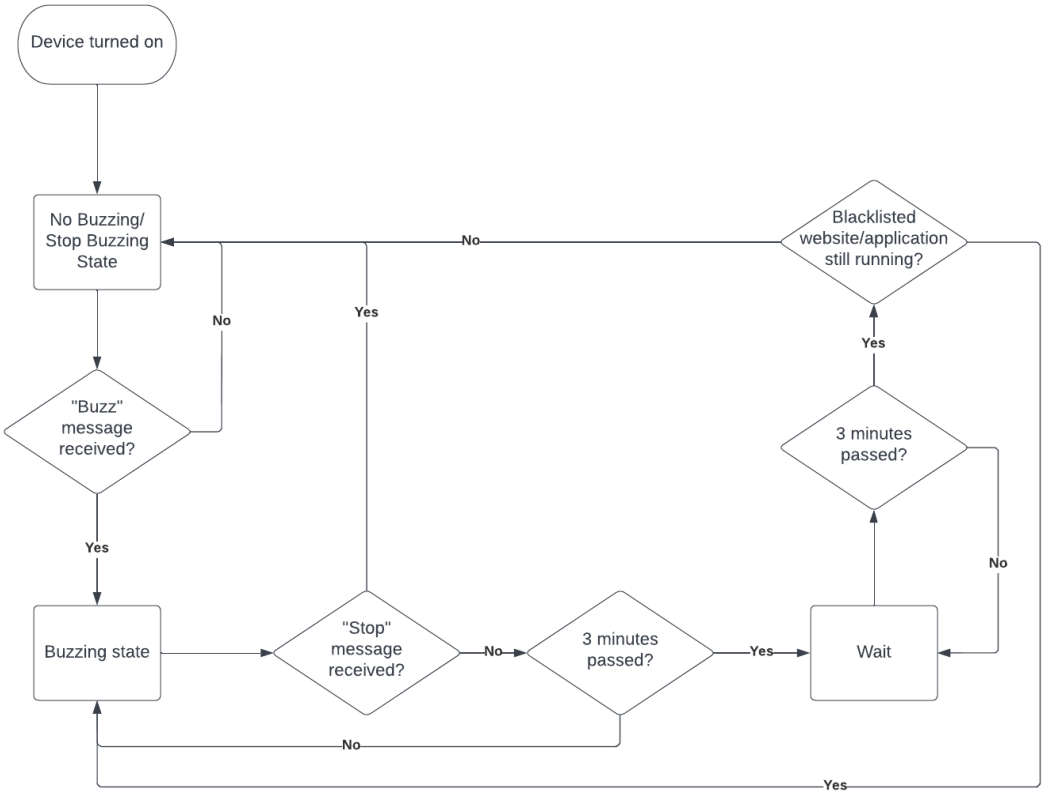

Figure 4: ESP32 Arduino Flowchart

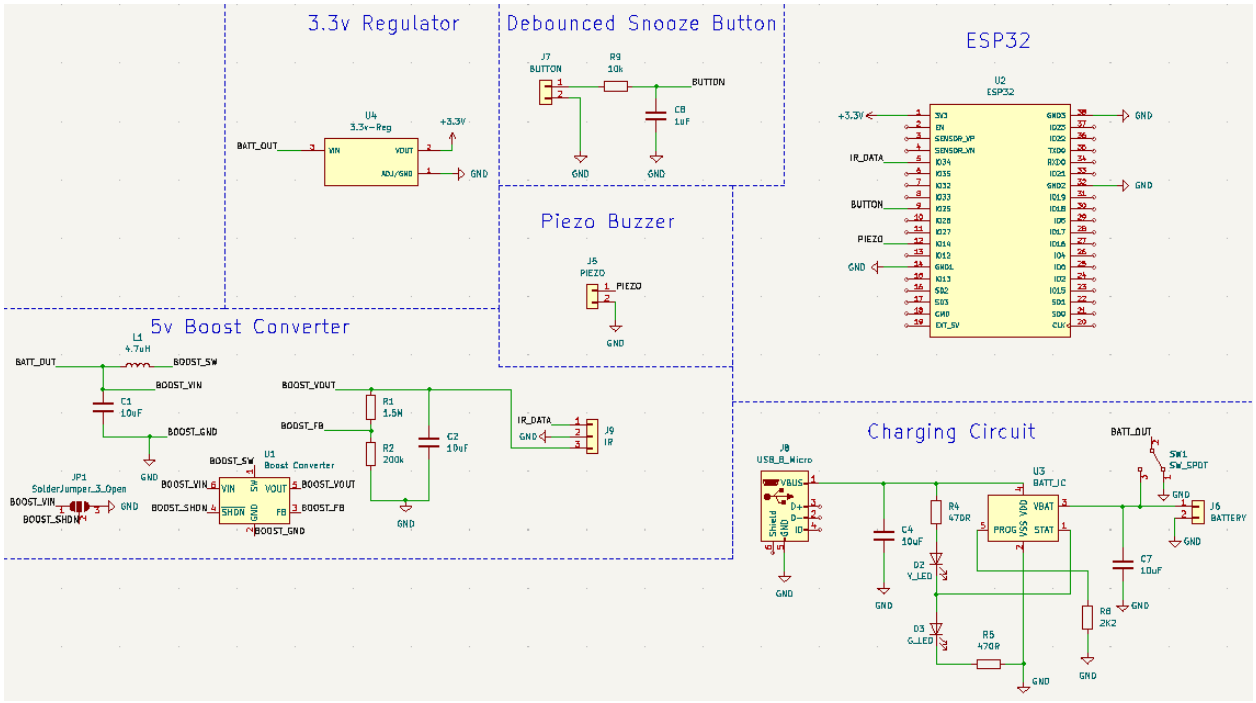

Figure 5: Circuit Schematic

2.1 Power Subsystem

The Power Subsystem provides all of the power for all of the components in our device. Using the 3.7v battery, the 3.3v voltage regulator, the 5v boost converter circuit, and the LiPo charging circuit, the Power Subsystem is able to provide all of the necessary voltages that the various components in our device use as well as a way to charge the battery. The Power Subsystem connects to the Control Subsystem by providing 3.3v to the ESP32 and 5v to the IR sensor. To simulate our power systems battery lifetime we were able to estimate this using the equation for total battery life

Battery life (h) = Capacity(mWh) / (I(Ah) * Vcc(V)) [22]

we could gather expected current loads of the parts we planned on using to determine how long the battery should last. Through a combination of datasheets and testing, we expected the current draw from each component to be - ESP32 (20mA @ 3.3V), PIR sensor (20mA @ 5V), and the buzzer (8mA @ 3.3V). The numbers for the regulator and converter are (3.7 Vin, 3.3 Vout, and 20mA) for the regulator and (87% efficiency, 5 Vout, 20mA) for the boost converter. Using these numbers we can determine the power draw of each component including the 5v boost converter and 3.3V regulator.

Power (mW) = I(mA) * V(V) [23]

Regulator Power (mW) = $[V_{in} - V_{out}]$(V) * I(mA) [24]

Converter Power (mW) = (1- (η)(efficiency)) * V_{out}(V) * I(mA) [25]

Using the above equations we can find the power draw of each component - ESP32 66mW, PIR sensor 100mW, buzzer 26mW, regulator 8mW, and converter 13. This brings our total power draw to be 213 mW per hour and the total battery power is 1295mW. This brings us to 1295/213 hours, or 6.08 hours of life in the device. This was a calculation that was determining the worst case scenario given the high current draws of the IR sensor and ESP32.

2.2 Control Subsystem

The Control Subsystem receives any Bluetooth data coming from the host computer so it knows what to tell the ESP32 (e.g. to turn the piezo buzzer on/off). The ESP32's onboard Bluetooth module gets the Bluetooth data, and then relays it to the ESP32. The Control Subsystem connects to the Output Subsystem by the ESP32 generating a 1000 Hz square wave, which is sent to the piezo buzzer in the Output Subsystem. In order to preserve power, the device will be programmed to check for blacklisted websites/programs every 5 seconds and send the square wave if it detects a blacklisted item. It will send this wave while the website/program stays open until 3 minutes have passed, where it will stop and wait for another 3 minutes before sending the signal again for another 3 minutes if the website/program continues to stay open.

2.3 Output Subsystem

The Output Subsystem receives waves from the ESP32 and plays it out loud via the piezo buzzer in the Output Subsystem. The Output Subsystem's main function is to annoy the user (via the loud piezo buzzer) whenever a blacklisted website/program is opened, and it keeps playing until the blacklisted websites/programs are closed. The effectiveness of our project depends on the Output Subsystem.

2.4 User Interface Subsystem

The User Interface Subsystem reads programs and websites running on the user's host computer. The user can whitelist/blacklist websites/programs via Windows software and browser extension software. The detection of blacklisted websites/programs and the user interface to add websites/programs to the blacklist/whitelist is provided by the software in the User Interface Subsystem. When the software detects a blacklisted website/program running, it sends a wireless signal via Bluetooth to the ESP32, which is how the User Interface Subsystem connects with the Control Subsystem. The User Interface Subsystem will also have a user-friendly GUI that allows the user to add and remove applications/websites to their respective whitelists/blacklists.

3 Design Verification

The requirements and verification steps that were taken can be seen in the Appendix as the R/V tables for each individual subsystem. Overall we were able to verify every aspect of the design that we needed for it to function properly and maintain its core functionality, detecting when the user is procrastinating online or not at their desk and taking appropriate action in the form of buzzing loudly.

3.1 Power Subsystem

When thinking about the power subsystem as a black box, the requirements for the subsystem entailed our battery being able to maintain a 3.7V supply as the input of the system, the power subsystem must be able to output 3.3V to the control subsystem, the power subsystem must also be able to output 5V to the control subsystem, and the power subsystem must be able to do these functions for at least 6 hours on a fully charged battery.

As a group we were able to verify the first 3 requirements using the multimeter in the lab kit after we soldered the boards for the first time. After connecting all the components to the board we probed areas that would have the voltages we were looking for. To verify the 3.7V supply from the battery we probed the input to the 3.3V regulator and verified that we were receiving 3.7V. To verify that the voltage regulator was giving us 3.3V we probed the output of the voltage regulator as well as the 3.3V pin on the ESP32, both were verified as having 3.3V. Finally we needed to verify that the boost converter was giving us at least 5V for the IR sensor. We probed the input pin of the IR sensor and took a couple readings which gave us voltages of between 5.3 and 5.4, which is well within the tolerance of what we needed for the IR sensor to function properly.

For the final requirement we tested the battery life of the device by charging it fully, indicated by the green LED in the charging circuit, and then connecting it to the computer and going to a blacklisted application. Then the device was left running, beeping constantly, until it eventually died and we determined that the device lasted for approximately 7.5 hours. This exceeded the expected value and verified that the device would be able to last over 6 hours on a full charge.

3.2 Control Subsystem

The control subsystem requirements consisted mainly of being able to receive and interpret a bluetooth connection established with the host computer, being able to produce a square wave for the buzzer, and the IR sensor being able to give distinct output values for items at different distances in front of it.

The first requirement we were able to verify was when we first obtained the parts and started individually testing them. We programmed the ESP32 to just output a square wave and were able to adjust the frequency and duty cycle. We connected this with our buzzer and verified that the square wave was produced and the buzzer made a sound we were expecting. Along the same note we were able to test the IR sensors output here as well. Using the ESP32

we read the output of the IR sensor while sitting in front of it at varying distances and received distinctly different and fairly stable readings for each of the distances.

One of the most important parts of the project was the bluetooth connection though and we initially verified this by testing with an application called Windows Bluetooth Terminal. This application let us connect the ESP32 to the host computer and after putting code into the ESP32 that would light up an LED if a 1 was received or turn it off if a 0 was received we were able to verify that sending 1s and 0s through the bluetooth terminal would give us appropriate LED states on the board. Eventually we went on to directly connect a python script to the ESP32, but that worked in the same way and we were able to send data from the host computer to the ESP32 and it was able to interpret this data.

3.3 Output Subsystem

The output subsystem is fairly simple as it only consists of one part which is the piezo buzzer. The main requirements of the subsystem are that the buzzer must be able to produce a noise of between 40-60dB at around 1000 Hz by receiving a square wave from the ESP32. The wave requirement we were able to verify the same way as in the previous subsystem where we verified that the ESP32 was able to produce the wave. When we first obtained the parts we started individually testing them. We programmed the ESP32 to just output a square wave and were able to adjust the frequency and duty cycle. We connected this with our buzzer and verified that the square wave was produced and the buzzer made a sound we were expecting. The duty cycle was the “loudness” of the noise and could be tuned to get the appropriate dB of sound we needed. We used 1000Hz as the frequency for all tests and the final product. We verified that our buzzer was giving us between 40-60dB by using a smartphone app that could use the phone's microphone to give a dB reading. As seen in figure 5 we obtained a reading of around 56dB which was what we wanted.

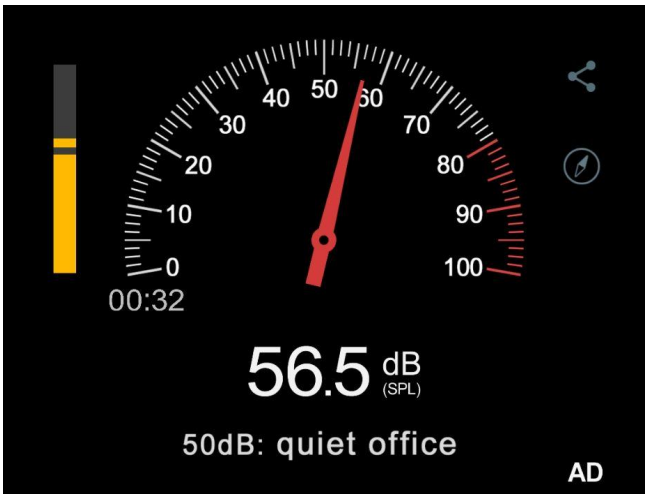

Figure 5: Decibel Reading

3.4 User Interface Subsystem

The requirements that we need to satisfy for the User Interface subsystem are that the user needs to be able to edit the blacklist and whitelist for applications and websites, the software must be able to detect open applications and websites, and finally that the software must be able to communicate with the ESP32 via bluetooth.

The verification of connecting the host computer software and the ESP32 was explained in the control subsystem verifications, but to add to that we followed basic example code to be able to program the ESP32 to connect via bluetooth and verified it with the Windows Bluetooth Terminal. Figure 6 below shows the device paired with the host computer.

Figure 6: Antiprocrastinator device paired with host computer

The verification for being able to detect open applications and websites was simple after it was finished because we had logs throughout that were able to tell if anything was detected. Figure 7 is an example of the logs that we used to verify that it could correctly detect when a website or application was open that was blacklisted. Youtube is a blacklisted website and Discord is a blacklisted application. Figure 8 shows the chrome extension which is uploaded into the user's browser to enable the website detection.

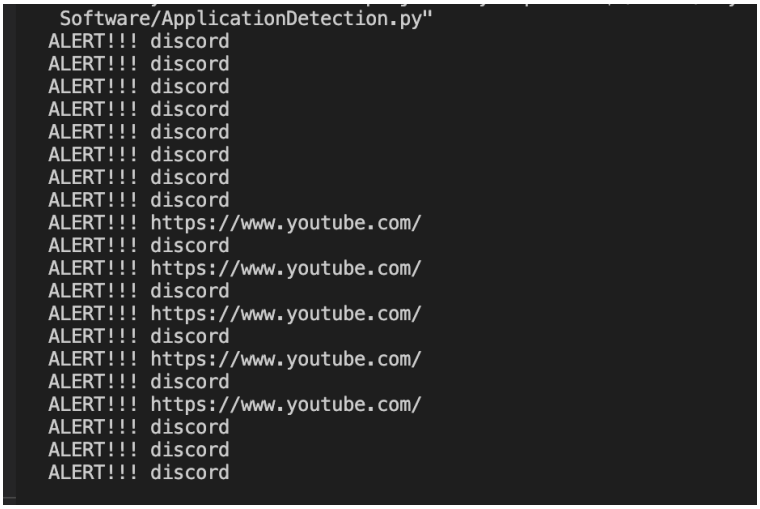

Figure 7: Logs showing apps or websites being detected

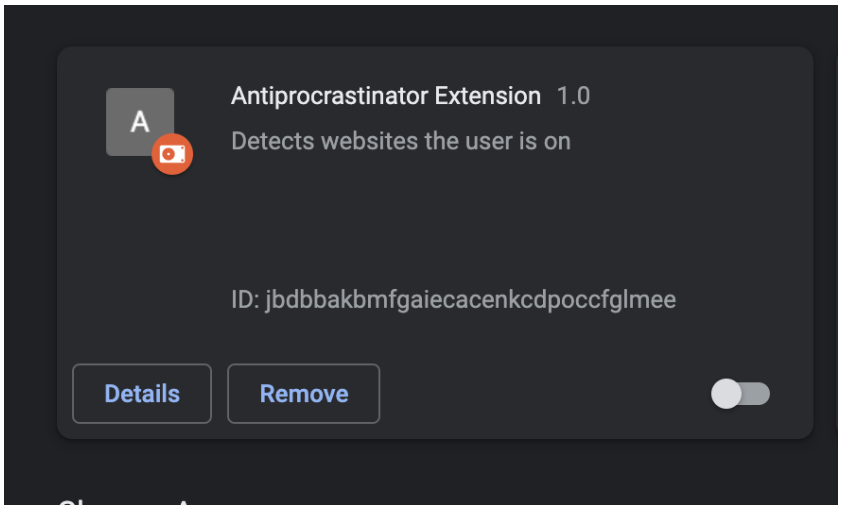

Figure 8: The Chrome extension which can be turned on or off

Finally we needed to verify that the user could easily add or remove websites and applications from the blacklist and whitelist. For this we implemented a GUI that would be intuitive to users as seen in figure 9. We verified that the GUI worked properly by adding and removing applications and websites while the device was on, there is no need to restart the device to apply changes to the lists due to how the GUI was coded, and verifying that we were printing the correct logs or in later stages of testing were producing the correct output of buzzing noise or not. The GUI explains itself, but to explain it anyways all you do is type the app name or website name into the bar and then choose from the dropdown menus what list you want to add it to, then simply hit the add button. To remove anything you just click the red remove button next to the name and it will update.

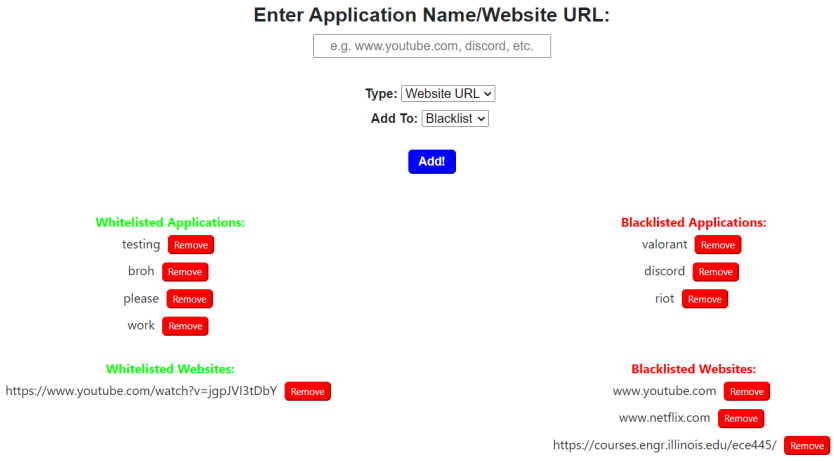

Figure 9: The GUI to update lists

4 Costs

4.1 Parts

Table 1: Parts Cost

Part	Manufacturer	Units Required	Retail Cost ($) (per unit)	Bulk Purchase Cost ($) (per unit in orders of 1,000 units)	Actual Cost ($)
LD1117V33 IC REG LINEAR 3.3V 800MA TO220AB [15]	STMicroelectronics	1	0.84	0.37030	0.37030
1051330001 CONN RCPT USB2.0 MICRO B SMD [1]	Molex	1	1.17	0.70762	0.70762
XZM2CYK54WA-8 LED YELLOW CLEAR CHIP SMD [21]	SunLED	1	0.57	0.19180	0.19180
XZDGK54W-8 LED GREEN CLEAR CHIP SMD [20]	SunLED	1	0.67	0.18508	0.18508
MCP73831T-2ACI/OT IC BATT CNTL LI-ION 1CEL SOT23-5 [17]	Microchip Technology	1	0.76	0.69	0.69
UM3429S STEP UP DC DC CONVERTER [18]	Union Semiconductor (HK) Limited	1	0.4175	0.38	0.38
258 BATTERY LITHIUM 3.7V 1.2AH [3]	Adafruit Industries LLC	1	9.95	9.95	9.95
GP2Y0A21YK0F SENSOR OPTICAL 10-80CM ANALOG [13]	SHARP/Socle Technology	1	13.50	6.63307	6.63307
1739 BUZZER PIEZO 12V 30MM FLANGE [2]	Adafruit Industries LLC	1	0.95	0.95	0.95

ANT11SEBQE SWITCH TOGGLE SPDT SOLDER SEAL [5]	CIT Relay and Switch	1	1.69	1.12167	1.12167
ERJ-HP6F4700V RES 470 OHM 1% 1/2W 0805 [8]	Panasonic Electronic Components	2	0.40	0.05558	0.11116
ERJ-HP6J222V RES 2.2K OHM 5% 1/2W 0805 [10]	Panasonic Electronic Components	1	0.27	0.03690	0.03690
ERJ-P06J204V RES SMD 200K OHM 5% 1/2W 0805 [11]	Panasonic Electronic Components	1	0.13	0.01980	0.01980
ERJ-6ENF1504V RES SMD 1.5M OHM 1% 1/8W 0805 [9]	Panasonic Electronic Components	1	0.10	0.01512	0.01512
LQM21PN4R7MGHL FIXED IND 4.7UH 1.2A 275MOHM SMD [16]	Murata Electronics	1	0.29	0.11684	0.11684
C2012X7R1A106K125AE CAP CER 10UF 10V X7R 0805 [6]	TDK Corporation	4	0.44	0.11495	0.4598
ESP32-DEVKITC-32U EVAL BOARD FOR ESP-WROOM-32 [12]	Espressif Systems	1	10.00	10.00	10.00
Total Cost:					**31.93916**

4.2 Labor

We are basing our analysis on the formula: ($/hour) x 2.5 x (hours to complete). Using a conservative estimate of $25 per hour and about 80 hours to complete, each team member would cost $5,000. And since we have a total of 3 team members, the total cost for labor is $15,000.

4.3 Grand Total

Total cost including parts and labor per 1,000 units is $46,939.16.

4.4 Schedule

Week 1

- **All Team Members:** Finished design document
- **Kyle and Brandon:** Completed rough draft of PCB design and parts layout
- **Taylor:** Started high-level software design

Week 2

- **Kyle and Brandon:** Started and finished rough draft of PCB design in CAD
- **Kyle:** Ordered all parts
- **Taylor:** Started creating software for Windows and Chrome extension

Week 3

- **Kyle and Brandon:** Finalized PCB CAD design and ordered PCB
- **Taylor:** Continued working on software functionality for Windows and Chrome extension software
- **All Team Members:** Started breadboard prototyping the device

Week 4

- **All Team Members:** Continued working on software functionality for Windows and Chrome extension software
- **All Team Members:** Tested basic functionality of all parts
- **All Team Members:** Soldered parts to PCB

Week 5

- **All Team Members:** Finished software functionality for Windows and Chrome extension software
- **All Team Members:** Tested Bluetooth functionality and pairing with host device

Week 6

- **Kyle:** Started designing and creating software GUI
- **All Team Members:** Tested Bluetooth signal reception from device and microcontroller response

Week 7

- **Kyle:** Finished software GUI
- **All Team Members:** Began testing device functionality

Week 8

- **All Team Members:** Finished testing device functionality
- **All Team Members:** Finalized software and the device

5 Conclusion

5.1 Accomplishments

The feat we were most proud of was being able to get the basic functionality of our project working before the deadline of our final demo, being able to successfully connect to a laptop/ PC via bluetooth, detect websites and applications, and buzzing when the user opens up blacklisted items or is away from the computer. Being able to turn in a complete product despite all the delays we encountered was a relief to us. We can attribute this success to the diligent work we put into testing and finalizing the software side of our project while parts were delayed. The ease of mind knowing that our website and application detection was working as intended allowed us to put all our focus into working on incorporating the hardware side of the project into it as soon as possible. As everyone in our team is software focused computer engineering majors, we were also glad that the circuit schematic we made worked as intended despite the little knowledge of CAD designing PCBs we all had coming into this project.

5.2 Uncertainties

A few issues that arose throughout this development process mainly came in the form of hardware difficulties. The first problem that was brought to our attention after numerous feedback was the complexity of our design. Originally, we had planned to use a bluetooth module called an HC-05 to establish our bluetooth connection and a standard charging circuit off the market. Luckily, since our project was still in its early stages, we were able to redesign our PCB to what it is currently, increasing the complexity by coding the serial bluetooth connection ourselves with the ESP32 and designing our own charging circuit on the PCB. As expressed above, our inexperience when it came to designing schematics led to many redesigns of our PCB, which caused a significant delay to actually soldering the parts we ordered and testing the analog devices. It got to the point where we had ordered a third PCB out of pocket only to get delayed further as we were sent someone else's PCB. The one function we had to scrap due to time constraints was a snooze button that would stop the buzzer if the user chose to press it. We had difficulty with the debounce circuit being always logic level high, so we were forced to focus on making sure the main features were working properly instead. While we were checking our circuit with a multimeter, we also found that our boost converter outputted 6V instead of the intended 5V. This worried us as it could have affected other components such as our IR sensor; while the values the IR sensor outputted were sometimes inconsistent (Spiking in the 2000s while no object was in front of it) we were able to adjust the thresholds so that the overall function was working as intended at the cost of its sensitivity forcing users to stand rather close to the sensor in order to be detected.

5.3 Ethical Considerations

In the interest of following principle 1.2 of the ACM Code of Ethics, we will ensure that the possibility of being injured or harmed by our project is minimal by selecting a piezo buzzer with

a low decibel range that match our requirements of being 40-60 dB as to not harm the ears of our users [19]. We hope to respect privacy and honor confidentiality of our users as stated in principles 1.6 and 1.7 by having users input the desired websites and programs to blacklist rather than automatically collecting data on the users browsing habits and having the device decide [4]. We will act in accordance with principle 2.9 by thoroughly testing and updating the functions of our device to make sure it works as intended, focusing on specifically making sure that the device is able to detect user-inputted websites and programs, and emitting a safe level of noise when doing so. In the same vein, we will ensure that this data remains secure so that there are no potential leaks of information of any kind. In order to follow the guidelines of the IEEE Code of Ethics, we will keep the safety, health, and welfare of the public as a top priority when designing our project [14]. Our project involves a lot of potentially dangerous electrical components. Should any of these components prove to endanger a user in any way, we will make revisions to our design in order to remove the threat. However, since we are dealing with very low voltages (3.7v), our project is not as dangerous as other projects dealing with higher voltages. But since we are dealing with lithium-ion batteries, we will link a University lab safety document to mitigate any possible issues with batteries [7]. A few potential issues that we noticed now that our project is complete was that certain components on our PCB would spike in temperature while the device was turned on. This could result in a malfunction later on in its lifespan and could cause a hazard in conjunction with our charging circuit and battery, so we will work towards modifying our design and thoroughly testing it so that this does not happen.

5.4 Future Work

A few features we planned implementing but weren't able to due to difficulties came up in the form of not being able to add a snooze button when the buzzer goes off. Continuing this project would most likely start with making sure that gets added next. Some other considerations we planned on incorporating into our design were being able to allow the user to adjust the volume of the buzzer, how long the intervals are between buzzing and waiting, and being able to adjust the IR sensor threshold. All three of these features work towards the client's ease of use by giving them control over their overall experience, making it so that users want to participate and engage with our project rather than despising it. As mentioned in the ethical considerations, we would also like to improve our design on a hardware level by modifying our PCB to ensure safety as well potentially designing a custom enclosure for our device to further increase the ease of use and aesthetics.

References

[1] “1051330001,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/molex/1051330001/3313407?s=N4IgTCBcDaIIwAYCscDMqGbiAugXyA. [Accessed: 04-May-2022].

[2] “1739,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/1739/5824413?s=N4IgjCBcoLQdIDGUBmBDANgZwKYBoQB7KAbXAFYAmADhAF0BfAmSqUZSAFwCcBXfIqRDl6DJiFaQyABwCWOAF6EABACNeChTm6igA. [Accessed: 04-May-2022].

[3] “258,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/258/5054544. [Accessed: 04-May-2022].

[4] “ACM Code of Ethics Booklet - Association for Computing Machinery.” [Online]. Available: https://www.acm.org/binaries/content/assets/about/acm-code-of-ethics-booklet.pdf. [Accessed: 04-May-2022].

[5] “ANT11SEBQE,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/cit-relay-and-switch/ANT11SEBQE/12503360. [Accessed: 04-May-2022].

[6] “C2012X7R1A106K125AE,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/tdk-corporation/C2012X7R1A106K125AE/5809181. [Accessed: 04-May-2022].

[7] “Division of Research Safety,” *Illinois*. [Online]. Available: https://drs.illinois.edu/Page/SafetyLibrary/BatterySafety. [Accessed: 04-May-2022].

[8] “ERJ-HP6F4700V,” DigiKey. [Online]. Available: https://www.digikey.com/en/products/detail/panasonic-electronic-components/ERJ-HP6F4700V/13878173. [Accessed: 04-May-2022].

[9] “ERJ-6ENF1504V,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/panasonic-electronic-components/ERJ-6ENF1504V/282713. [Accessed: 04-May-2022].

[10] “ERJ-HP6J222V,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/panasonic-electronic-components/ERJ-HP6J222V/13877635. [Accessed: 04-May-2022].

[11] “ERJ-P06J204V,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/panasonic-electronic-components/ERJ-P06J204V/525235. [Accessed: 04-May-2022].

[12] “ESP32-DEVKITC-32U,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/espressif-systems/ESP32-DEVKITC-32U/9357002. [Accessed: 04-May-2022].

[13] “GP2Y0A21YK0F,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/sharp-socle-technology/GP2Y0A21YK0F/720159. [Accessed: 04-May-2022].

[14] “IEEE code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 04-May-2022].

[15] “LD1117V33,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/stmicroelectronics/LD1117V33/586012. [Accessed: 04-May-2022].

[16] “LQM21PN4R7MGHL,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/murata-electronics/LQM21PN4R7MGHL/7595494?s=N4IgjCBcpgbFoDGUBmBDANgZwKYBoQB7KAbRAGYBOcgJkpAF0CAHAFyhAGVWAnASwB2AcxABfAjQAMADgDsCEMkjps%2BIqRAAWAHSyABAFaAEoxbtIIAKoC%2BrAPIoAsjjRYArjxxiJM6QqUquATEkGRg2jR6AIKmIGwc1rYOzq4eXqLiIDQaggAmboishDyMokA. [Accessed: 04-May-2022].

[17] “MCP73831T-2ACI/OT,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/microchip-technology/MCP73831T-2ACI-OT/964301. [Accessed: 04-May-2022].

[18] “Um3429s,” *UM3429S Union Semiconductor (HK) Limited | Integrated Circuits (ICs) | DigiKey Marketplace*. [Online]. Available: https://www.digikey.com/en/products/detail/union-semiconductor-hk-limited/UM3429S/13926054. [Accessed: 04-May-2022].

[19] “What noises cause hearing loss?,” *Centers for Disease Control and Prevention*, 07-Oct-2019. [Online]. Available: https://www.cdc.gov/nceh/hearing_loss/what_noises_cause_hearing_loss.html#:~:text=Common%20Sources%20of%20Noise%20and%20Decibel%20Levels&text=A%20whisper%20is%20about%2030,immediate%20harm%20to%20your%20ears. [Accessed: 04-May-2022].

[20] “XZDGK54W-8,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/sunled/XZDGK54W-8/8259042. [Accessed: 04-May-2022].

[21] “XZM2CYK54WA-8,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/sunled/XZM2CYK54WA-8/8571167. [Accessed: 04-May-2022].

[22] “Battery life calculation formula,” *Electrical 4 You* [Online]. Available: https://www.electrical4u.net/calculator/battery-life-calculator-formula-example-formula/. [Accessed: 30-March-2022].

[23] “Calculating electric power,” *All about circuits* [Online]. Available: https://www.allaboutcircuits.com/textbook/direct-current/chpt-2/calculating-electric-power/ . [Accessed: 04-May-2022].

[24] “Power and thermal dissipation,” *Sparkfun* [Online]. Available: https://www.sparkfun.com/tutorials/217 . [Accessed: 04-May-2022].

[25] “Boost converter efficiency,” *Maxim Integrated* [Online]. Available: https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/110.html [Accessed: 04-May-2022].

Appendix A Requirement and Verification Tables

Power Subsystem

Requirements	**Verification**
The Power Subsystem must be able to supply 3.7V from the battery once it has been charged.	Use an oscilloscope to determine voltage across battery
The Power Subsystem must be able to supply at least 3.3V to power ESP-32.	1A. Connect battery to 3.3 voltage regulator input. 1B. Connect 3.3V regulator output to ESP-32 input. 2A. Use an oscilloscope to determine voltage between regulator and ESP-32. If the red light comes on on the ESP-32 it is receiving power.
The Power Subsystem must be able to supply at minimum 4V and at maximum 7V to power the IR Sensor. Ideally should be around 5V.	1A. Connect battery to 5V boost converter input. 1B. Connect boost converter output to IR Sensor input. 2A. Use an oscilloscope to determine voltage

	between boost converter and IR Sensor.
The Power Subsystem must be able to supply power to the device for at least 6 hours on a full charge	1A. Charge the device battery fully using microUSB. 1B. Connect device to host computer and use normally, visiting at least one blacklisted site an hour. 1C. Determine what time the battery stops being sufficient to power the system by monitoring when the device does not continue working.

Control Subsystem

Requirements	**Verification**
The Control Subsystem must be able to receive a bluetooth signal sent from a host computer to the bluetooth module	Connect bluetooth module to correct power and connect with computer. Send training data through bluetooth and verify at the bluetooth module data pin that data was received using a microcontroller program to display output.
ESP32 must be able to interpret the signal received by the Bluetooth module to turn on/off output	Connect bluetooth module to correct power and connect with computer. Write microcontroller code that uses data sent by the host computer through bluetooth to turn on and off an LED.
ESP32 must be able to control how long it sends a square wave to the buzzer such that it turns on and off in 3 minute intervals	After coding the behavior, open a banned application. As soon as the buzzer starts buzzing, time the duration of the sound. After it stops, time the duration up until it starts buzzing again. Compare the times to see if they are within ± 5 seconds of 3 minutes.
IR sensors must be able to give distinct voltages for something being about 6 inches from it and for nothing being in front of it for at least 2 feet.	After the IR sensor has adequate power,read the voltage at the output pin and verify that it is distinct for the IR sensor with a person standing 6 inches from it and nothing being in front of it.

Output Subsystem

Requirements	Verification
ESP32 must be able to send a 1000 Hz square wave to the buzzer	Load code to produce the wave on the ESP32 and run it while using an oscilloscope to verify that the wave is of the correct form and within ±50Hz.
The piezo buzzer must be able to output a noise of about 40dB-60dB whenever the board microcontroller sends a 1000 Hz square wave to the piezo buzzer	Once the square wave has been verified, wire that wave into the input of the buzzer and measure the sound using a decibel sensor.

User Interface Subsystem

Requirements	Verification
The User Interface Subsystem must be able to allow a user to add/remove applications/websites to a blacklist/whitelist managed by the browser extension software and Windows software	1A. Use the interface to add and remove websites from the blacklist/whitelist. 1B. Visit those sites and check if the device is performing the appropriate action.
The User Interface must be able to detect what applications and websites are running/open	1A. Give a list of applications and websites to the program to check. 1B. Make sure that some are open and the others are closed. 1C Read the program's output and verify that it is correct about which programs are open and closed.
The User Interface must be able to send a 'start'/'stop' signal via Bluetooth to the ESP32	1A. Code the ESP32 to be able to interpret data from the host computer. 1B. Send signals from the host computer to the device over bluetooth and verify that the ESP32 can discern between the two signals 'start' and 'stop' (which can be any arbitrary signal such as 0 or 1).