CS 8803 Final Report

Hanrui Chen, Jongyoon Choi

1. Introduction

The problem we aim to address with our project revolves around the inefficiency of traditional WiFi activation approaches. Typically, WiFi is kept active for prolonged periods, leading to unnecessary power consumption. In contrast, our demonstration showcases the potential of Ultra-Wideband (UWB) messages to enable targeted WiFi activation for specific events. By initially turning WiFi off and activating it only when UWB messages targeted for the particular phone are received through our Android app, we explore the possibility of significant battery savings. This report delves into various aspects of our project. We provide an in-depth exploration of the core idea driving our approach, detailing the intricacies of the hardware and software components integral to its implementation. Additionally, we shed light on the methodologies employed in our project, elucidating the step-by-step processes that contribute to the efficiency of our WiFi activation model. Furthermore, the report presents an overview of the current status of our project and outlines the envisioned trajectory for its future development.

2. System Design

2.1. Overall Description

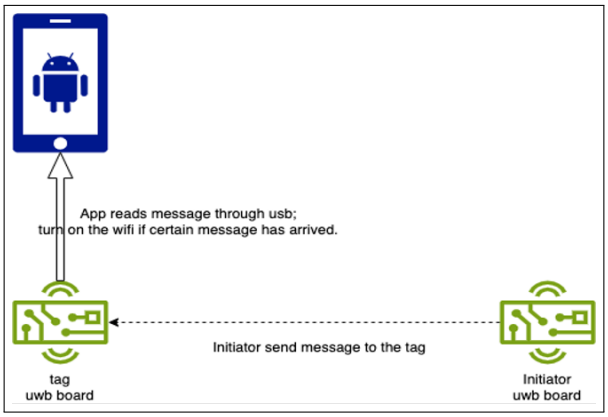

Figure 1: Design Diagram

Figure 1 illustrates the overall design of our project. The project comprises an initiator UWB board and a tag UWB board. We customized the provided Arduino codes and uploaded them to the two UWB boards, ensuring that the initiator periodically transmits UWB messages to the tag UWB board at regular intervals. The tag, connected to our Android phone via a USB cable, receives these messages. Then, the Android App reads the messages from the tag through the USB cable. Upon receiving specific messages intended for this particular phone, such as a request to turn on the WiFi, the app automatically activates the WiFi, simultaneously triggering the embedded YouTube video to play. For the purpose of our project, we chose to embed a 15-minute timer video that has no sound or intense graphics, in order to make battery consumption as simple as possible.

2.2. Arduino Code Design

The initiator is connected through the laptop. Then, we can send UWB messages to the phone, and we can also set how frequently it will send the message through Arduino. The tag is initially in *'UWB_DATA_EXPECTED'* state, and when the UWB message is received, it goes into *'STATE_RECEIVE_UWB_DATA'* state, where it checks whether the message is intended for the phone. If it is not, the message is ignored, and if it is, the app intelligently wakes up the WiFi to process the incoming data.

2.3. Android App Design

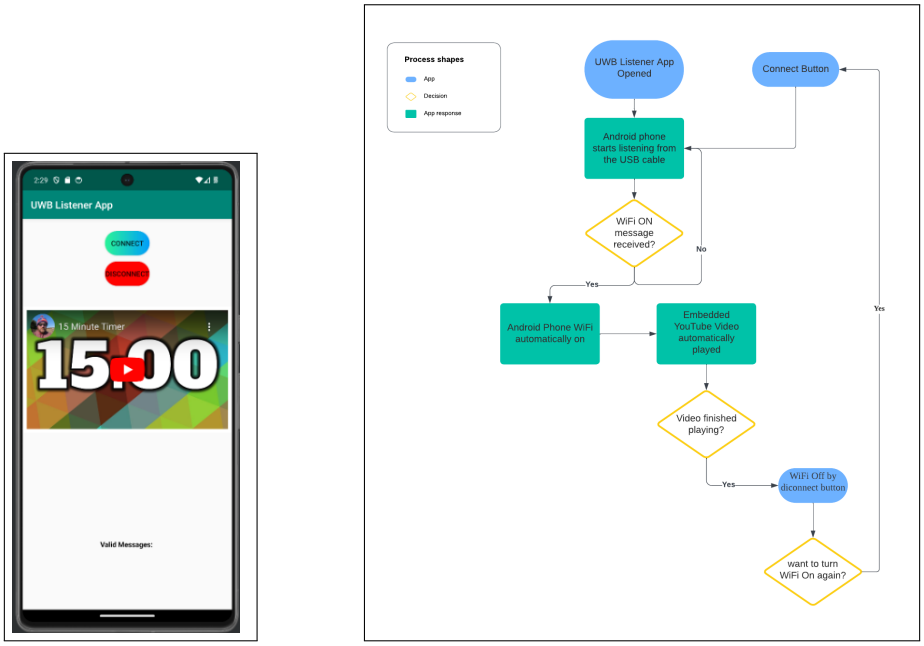

Figure 2: left to right: UWB Listener App & App Flow Diagram

Figure 2 displays the user interface of our UWB Listener App along with the corresponding flow diagram. Once the app is opened, it initiates communication with the tag UWB board through the USB cable using UsbSerial [1]. Upon receiving a message prompting the activation of WiFi, the message appears in the Valid Message area, the WiFi is activated, and the YouTube video begins playing automatically. To automate the process of turning on the WiFi, we utilized the *WiFiManager* class and its *setWifiEnabled* API.

Regarding the other two buttons, the Connect button re-establishes USB communication between the Android app and the tag UWB board. The Disconnect button, on the other hand, deactivates the WiFi. After the YouTube video completes, the WiFi can be disconnected, and the UWB message reception can be restarted by clicking the Connect button. The two buttons can also be used for handling errors, which will be discussed under the Discussion section in detail.

The inclusion of the embedded YouTube video serves a specific purpose. Our initial baseline scenario involves activating the WiFi for a duration of 30 minutes, while the comparative scenario entails keeping our app open for the same duration. Through our investigation, we observed that the WiFi consumes significantly less battery power than our app due to its inherent optimization features. Consequently, we aim to enhance WiFi usage by incorporating the YouTube video.

3. Evaluation and Discussion

3.1. Evaluation

To measure the effectiveness of our approach, we used Battery Historian, which is a tool developed and provided by Google as default, as shown in Figure 3. With Battery Historian, we conducted 3 experiments comparing battery usage between the traditional continuous WiFi activation and our UWB-triggered WiFi activation method. One example can be seen in Figure 4. As can be seen

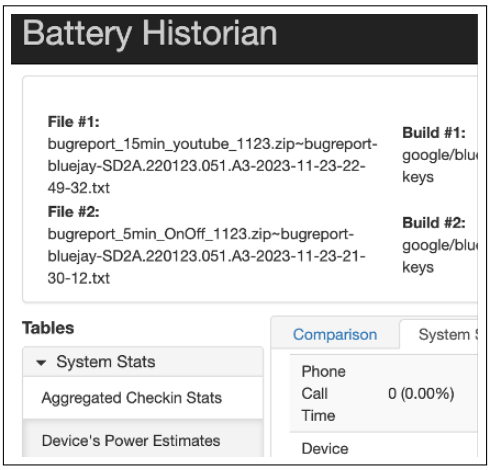

Figure 3: Battery Historian Comparison

Name	Diff Battery (%) Consumed	#1 Battery (%) Consumed	#2 Battery (%) Consumed
WIFI	0.10	0.17	0.07
SHELL	0.03	0.03	0.00
com.mci.wakeonuwb	-0.02	0.71	0.74
com.google.android.gm	0.02	0.02	0.00

Figure 4: Comparison in details

from Figure 4, Traditional continuous WiFi activation utilized 0.88% (0.17% (wifi) + 0.71% (app) = 0.88%) of the battery over the 30-minute timeframe while our app utsed 0.81% (0.07% (wifi) + 0.74% (app) = 0.81%) of the battery. Similar results have observed in two subsequent tests. Our results indicate potential energy savings, prompting us to further explore this innovative approach.

We tested the traditional method by first watching embedded 15-minute YouTube video and then leaving WiFi on for the next 15 minutes. We tested the UWB method by alternating between 1) WiFi on & watching YouTube video and 2) WiFi off & not watching YouTube video, with both 1) and 2) with leaving UWB tag communication open.

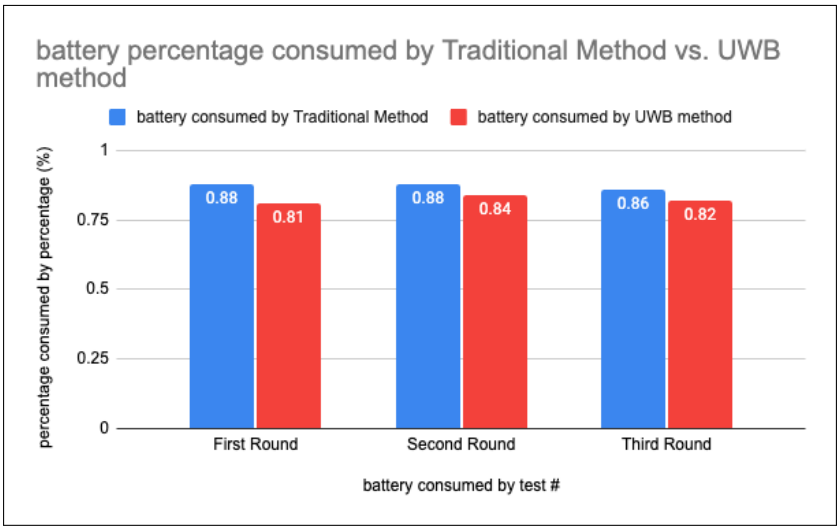

Figure 5: Battery Comparison Graph

As can be seen from Figure 5, after averaging the battery consumption for these three runs, the traditional method accounted for 0.873% and our method accounted for 0.823%, resulting in 0.05% difference over the 30-minute time frame. As professor mentioned in the project presentation, our method is not as optimized as it should have been. Thus, there is not as much battery consumption difference as we would have hoped.

Not only that, it is worthwhile to note that Battery Historian was not able to determine the amount of battery consumed by USB receptors. Thus, we hypothesize that accounting for USB connector used by the phone, the battery consumed for the second method is actually higher than what is reported on the Figure.

3.2. Discussion

3.2.1. Scenarios

1) **Normal Case**: The app successfully opens with WiFi off, initiating USB communication with the tag board. Upon receiving the "wake-up" message, the WiFi status is turned on, and the embedded YouTube video plays successfully.

2) **WiFi On Initially**: The app is successfully opened, starting USB communication with the tag board. The embedded YouTube video begins playing. Valid messages do not alter the WiFi status. The Disconnect button can be used to deactivate WiFi. The embedded YouTube video continues playing until there is no more cached content. Subsequently, WiFi can be reactivated upon receiving a valid message.

3) **App Fails to Communicate with USB When Opened**: In the event of a communication failure upon app opening, the Connect button may be utilized to rebuild the USB communication.

4) **USB Connection Failure During Video Playback**: If there is a sudden failure in the USB connection while the YouTube video is playing. We may also use the Connect button to rebuid the communication.

3.2.2. Potential Foreseen Issues

1) **The *'WifiManager.setWifiEnabled'* API was deprecated in Android SDK version 29**: This is the only method for automating the WiFi activation process. Alternative methods would necessitate varying degrees of human intervention. However, API levels higher than 28 no longer have access to this API due to privacy concerns. While it functions properly with API level 28, Google Play now mandates that apps target API level 31 or higher. Therefore, to ensure the future usability of our app for the public, a solution to this limitation needs to be addressed.

4. Related Work

As of now, we have successfully developed a demo app that incorporates the UWB-triggered WiFi activation feature. The app has been used to measure battery usage, providing preliminary insights into the potential savings achievable through our approach. Moving forward, our road map involves the following steps:

1) **Integration with Built-in UWB**: We plan to transition from using external UWB boards connected via USB to utilizing an Android phone that has built-in UWB capabilities. This transition aims to enhance the practicality and accessibility of our solution, as phones only have one Type C port and current method uses this port for connecting to the UWB board. Also, as only certain phone models have a built-in UWB software, it is critical to note that utilizing UWB software require correct phone models and that UWB integration is not possible for all types of phones.

2) **Extended Battery Measurements**: To gain a more comprehensive understanding of the battery savings potential, we intend to conduct experiments over longer periods of time. This will help us better understand the sustainability and real-world applicability of our UWB-enabled WiFi activation approach.

5. Conclusion

In conclusion, our project represents a novel exploration into optimizing smart WiFi activation through UWB-triggered events. By intelligently leveraging UWB technology, we aspire to contribute to the development of energy-efficient practices in mobile devices. The current demo app and initial battery usage measurements lay the foundation for further advancements and validations in our quest for sustainable and efficient WiFi activation methods.

References

[1] F. Herranz, felhr85/usbserial, `https://github.com/felHR85/UsbSerial`, accessed on: Nov. 15, 2023 (n.d.).