ECE 445
SENIOR DESIGN LABORATORY
PROJECT PROPOSAL

Any-Screen to Touch-Screen Device

Team No. 1
Sakhiyuvio Farsya Yunalfian
(sfy2@illinois.edu)
Muthu Ganesh Arunachalam
(muthuga2@illinois.edu)

TA: Chi Zhang
Professor: Arne Fliflet
September 19, 2024

1. Introduction

1.1 Problem

As touchscreens become an important method of user input, the demand for touch-enabled devices across industries has surged. However, retrofitting existing displays of any size, especially large ones, with touch capability is prohibitively expensive and technically challenging. Organizations like schools, businesses, and research departments use non-touch displays that lack the interactive functionality needed for modern applications, like collaborative learning, design work, or real-time marking up of digital documents. Upgrading these screens to support touch functionality would be a large cost and would require specialized hardware.

Although there are devices on the market that can overlay touch functionality on a non-touch screen, there are several drawbacks. Some of these devices only accept generic free-floating hand motions as input (instead of taps, clicks, and dragging), limiting user interaction with the screen. Other devices allow for direct interaction with the screen, but these devices can be inaccurate, resulting in user frustration. Several products are accurate and allow direct interaction with the screen, but use technologies like camera imaging which can cause faulty sensing if your hand is in the way; such sensing technologies can also limit the screen size on which the product can be used. Given the growing need for interactive interfaces and the lack of versatile products on the market, developing a cost-effective, scalable solution to this problem is crucial.

1.2 Solution

The proposed solution is a device that can convert any standard screen into a touchscreen using ultra-wideband (UWB) sensors to track a specially designed pen. This system relies on a network of UWB sensors placed at the corners of the display, which track the position of the pen in real time based on Time of Flight (ToF). The pen, equipped with a UWB tag and motion sensors (gyroscope), communicates its position to the sensors, allowing the device to detect movement across the screen. The pen communicates all sensor and location data via Bluetooth HID to the host device. This approach negates the need for expensive capacitive or resistive touch overlays, instead providing a wireless and modular solution that can be applied to any display size or type. By incorporating additional features such as buttons, the pen can offer more than basic touch functionality, enabling features like clicks, dragging, scrolling, and hotkeys. This solution not only provides an affordable way to retrofit non-touch displays but also expands the possibilities of user interaction in diverse settings and across multiple operating systems.

1.3 Visual Aid

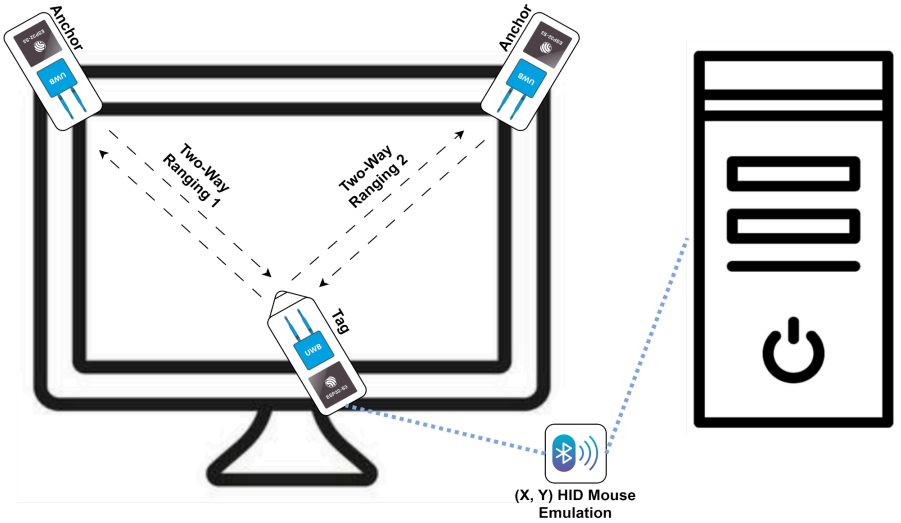

Figure 1: The Any-Screen to Touch-Screen System

1.4 High-Level Requirements

- Accuracy and Responsiveness: The most crucial success indicator for our project is touch emulation accuracy and responsiveness. Utilizing the Bluetooth module embedded in the ESP32-S3 microcontroller, we undoubtedly expect numerous sources of delay like propagation delay, lags, and queuing. We aim to quantitatively access this through continuous data logging through Python to monitor the time interval of **data transfe**r and keep it **under 250 milliseconds**. Aside from system responsiveness, another crucial requirement is for our location-tracking mechanism to be accurate. We aim to potentially reach a **5% margin of error in regards to the Euclidean distance of where the pen is touching the screen** and what (x, y) coordinates are digitized and processed. There are some factors that we need to consider that may intensify the error like sensor data fusion with the gyroscope to account for pen tilt, acceleration, and sensitivity.

- System Integration and Compatibility: The next crucial high-level requirement is that we want our system to be compatible with straightforward integration. Numerous compatibility aspects will be taken into account: **screen size and operating system**. Quantitatively, we will be **testing our system from small screens (5 inches) to large screens (20+ inches)**. We want to **keep our error rate consistent at 5% or less for any type of screen**. Lastly, we want to test our system integration and compatibility with **different operating systems like Windows, macOS, or Linux**. The OS compatibility

will be quantitatively measured through **success rate**, **aimed at 95%**, by **comparing location precision error rates** between different OSs.

➢ Extended Functionality: The last high-level requirement is that the device must reliably support discrete touch-based actions such as **tapping, clicking, scrolling, and dragging**. Each input type should be detected and executed correctly at least **99%** of the time, with a clear distinction between tapping (single-touch actions) and dragging (continuous motion) gestures. Scrolling actions must be executed smoothly, with a minimal jitter rate of less than **2%** when moving across the screen. The system must ensure that scrolling and dragging actions are continuous, with no unintentional interruptions.

2. Design

2.1 Block Diagram

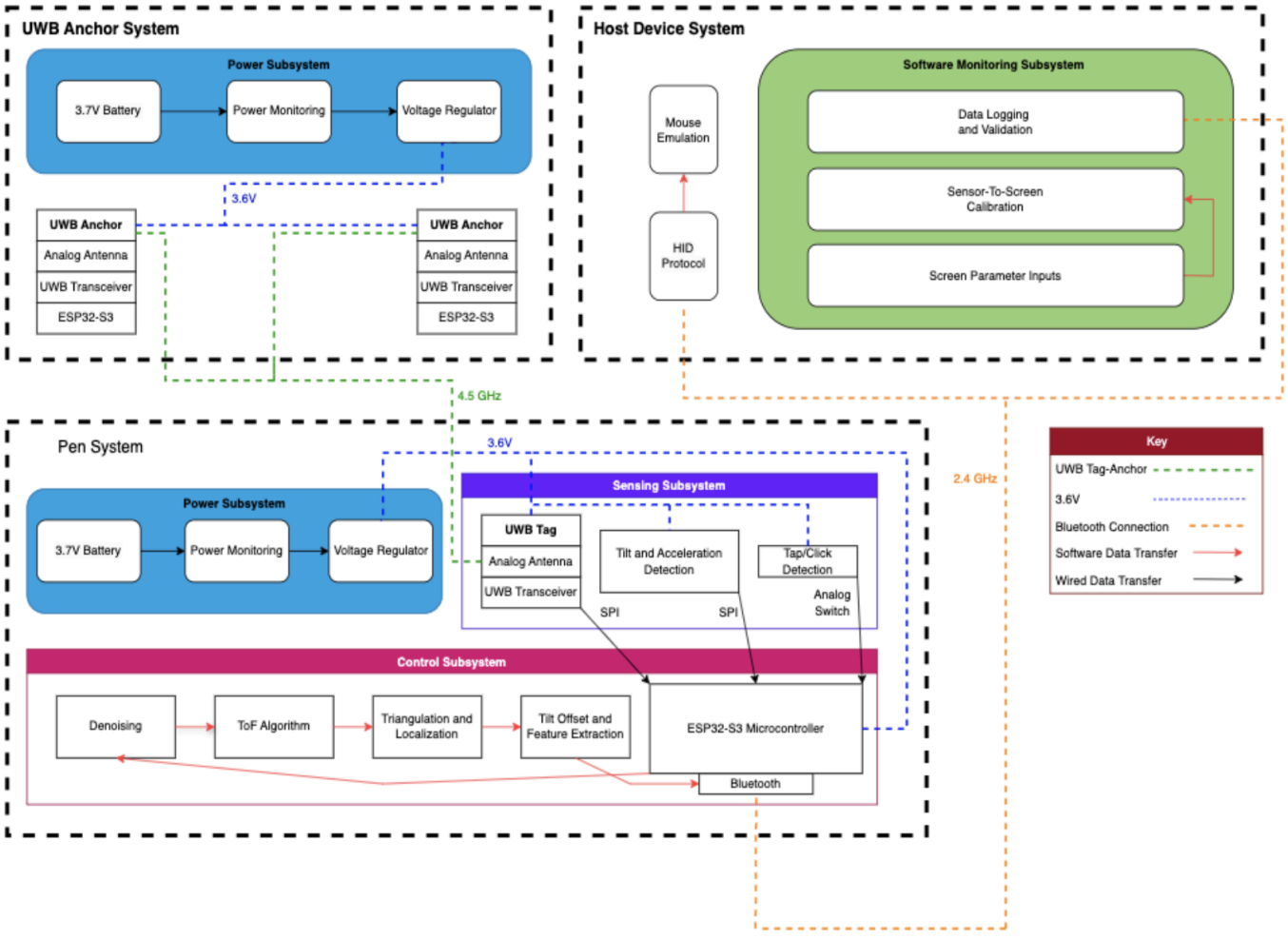

Figure 2: The Any-Screen to Touch-Screen Device Block Diagram

2.2 Subsystem Overview

Power Subsystem

Functionality: The Power Subsystem provides reliable and stable power to the entire device. It consists of a 3.7V battery, power monitoring, and a voltage regulator. The power monitoring system will detect when voltage dips below a certain level The voltage regulator converts the 3.7V battery output into the required 3.6V for the UWB modules and other components, ensuring stable operation.

Contribution: This subsystem ensures the device can function reliably and safely by providing consistent power to the sensors, microcontroller, and transceivers. The power monitoring and voltage regulator allow for the user to keep track of the state of the device and ensure that fluctuations in power supply do not affect the device's performance.

Interfaces:

- Inputs:
    1. 3.7V input from the rechargeable battery.
- Outputs:
    1. 3.6V regulated power to the UWB modules and ESP32-S3 microcontroller.
    2. Status signals from the BMS for potential monitoring of battery performance and safety.

Sensing Subsystem

Functionality: This subsystem captures important information on touch events between the pen and the screen using the concept of sensor fusion. Essentially, one UWB transceiver is embedded on the PCB that will contain the overall location data. The tilt and acceleration detection will use a gyroscope/IMU sensor, specifically the LSMGD2032 IMU. This gyroscope will contribute to the exact 3-dimensional position of the pen as it touches the screen, taking into account the tilting of the pen when finalizing location-tracking data. Lastly, the tap/click detection will be done with a small button embedded at the tip of the pen. This will be debounced to make sure that data transfer from the location and motion sensors occurs only when there is direct contact between the pen and the screen; in this case, through the button being depressed.

Contribution: The sensing subsystem will provide all pre-processed location data that is extremely crucial to create a distinctive judgment of the pen's location concerning the screen for touch-screen emulation. The pre-processed data will go through a DSP pipeline specifically to decode and assign the microcontroller to communicate with the host device of touch coordinates and HID functionalities via Bluetooth.

Interfaces:

- Inputs:
    1. 4.5 GHz RF signal communication between the UWB transceiver embedded on the pen PCB and the static UWB transceivers on the ends of the screen. Tx-Rx communication will use concepts like triangulation and time-of-flight (ToF) for the microcontroller to process the final coordinates of the pen.
    2. 3-dimensional angular velocity (rad/s) and linear acceleration (m/s^2) for tilting mechanism detection.
    3. The sensor power source comes at 3.6 V at around 150 mA.
- Outputs:
    1. Pre-processed data from the sensors through SPI communication with the ESP32-S3 microcontroller.
    2. The button's voltage (HIGH/LOW) to indicate contact with the screen.

Control Subsystem

Functionality: The first crucial step is denoising; the UWB transceivers will first go through the denoising step for filtering and smoothing. This would involve software algorithms like the outlier rejection for pre-processing and simple moving average (SMA) smoothing for random noises. Subsequently, the control unit is in charge of feature extraction, to only process useful data especially when performing sensor fusion between the sensors. Lastly, the final coordinates estimation will be done through a triangulation algorithm of the three UWB transceivers and the tilt offset will be decoded through IMU's data. The control subsystem will depend hugely on real-time operation capabilities like timers and interrupts as well as utilizing FreeRTOS from ESP-IDF for real-time data processing.

Contribution: This subsystem is the core of sensor data processing. The ESP32-S3 will receive digitized sensor signals that are sent via SPI protocol by the UWB transceivers and the IMU sensor. The UWB transceivers need to be accurate; therefore, the control subsystem includes a DSP pipeline embedded in the microcontroller to pre-process the signal data for reliable transfer to the host device by using Bluetooth HID protocol and specifications.

Interfaces:

- Inputs:
    1. Pre-processed data from the sensors via SPI involving UWB and IMU signal data.
    2. The button's node voltage, indicating contact with the screen.
    3. The sensor power source comes at 3.6 V at around 150 mA.
- Outputs:
    1. Post-processed sensor data, Tx transmission to host device via Bluetooth HID protocol.

Software Monitoring Subsystem

Functionality: The host device uses the GUI to link the pens buttons to software hotkeys for extended functionality. The GUI will take in system parameters, including screen size and screen resolution to start the calibration process. The host device communicates these parameters with the ESP32-S3 so it can calibrate the distance between the UWB anchors and linearize the location data. The GUI will display points on the screen to touch the pen to and map these points to the UWB sensor data as well. Once the sensors are calibrated to the screen, the microcontroller can start accurately computing and smoothing location data. It will send the mouse emulation data through Bluetooth to the host device using the HID protocol. The MCU will have a pre-defined HID descriptor and will send HID reports to the host device detailing mouse movements and clicked buttons.

Contribution: The software subsystem interprets data from the control subsystem and processes mouse emulation. The GUI then allows users to start calibration, map the pen's hotkeys, and configure system parameters like screen resolution and size. Additionally, the GUI will be used for data logging, helping validate system accuracy and track touch input over time for analysis.

Interfaces:

- Inputs:
    1. HID reports from ESP32-S3
    2. Screen Resolution and Screen Size, Computer Environment details
    3. Hotkey Mappings
- Outputs:
    1. Calibration parameters to microcontroller
    2. Location/mouse data logging
    3. Clicks, drags, and cursor movements

2.3 Subsystem Requirements

➢ **Power Subsystem Requirements:**

Power Monitoring:

1. Alert microcontroller and user when voltage dips to 3.0V-3.3V range

Voltage Regulator:

1. Must convert 3.7V input to a stable 3.6V output.
2. Must support a current draw of at least 500 mA.

➢ **Sensing Subsystem Requirements:**

1. The UWB sensors must provide useful location information within $\pm$ 1 centimeter of the true pen location.
2. The UWB sensors must be able to transfer and receive signals to one another at a 4.5 GHz frequency channel wirelessly.
3. The IMU sensor must provide data on angular velocities ($\pm$ 0.2 $\frac{rad}{s}$) and linear acceleration ($\pm$ 0.2 $\frac{m}{s}$)

➢ **Control Subsystem Requirements:**

1. Must be able to detect screen coordinates with an accuracy of 99%. The metric varies due to differing screen sizes, and the protocol of measuring accuracy will be done by calculating the Euclidean distance between actual and measured coordinates.
2. To ensure real-time feedback, complete required sensor data processing with a total communication delay of 60ms to the host device.
3. The system must be able to only initiate processing needs when there is direct contact between the pen and the screen.
4. Operate continuously with a consistent power consumption of 250 mW.

➢ **Software Monitoring Subsystem Requirements:**

1. Must be able to send GUI input parameters and hotkeys to the ESP32-S3 for calibration/setup process
2. Output data logs of location and mouse data for data and accuracy analysis
3. Have a functioning GUI that can potentially work on multiple OSs and display animations on the screen for the calibration process

2.4 Tolerance Analysis

The most critical aspect of our device is the **UWB-based positioning and communication system**, which uses Time of Flight (ToF) to eventually locate the pen's position on the screen. The accuracy of this positioning is essential, as even a small error in detecting the pen's position could lead to significant usability issues, such as incorrect touch points on the screen, poor gesture recognition, and overall frustration for the user.

The primary risk in this system is the error in the position calculation due to timing inaccuracies in the UWB sensors. To achieve accurate touch-screen emulation, the positioning error must remain below a threshold of 5% of the screen size. Given the short timing intervals involved in UWB signal transmission and reception, any noise, propagation delay, or timing drift could result in larger positioning errors.

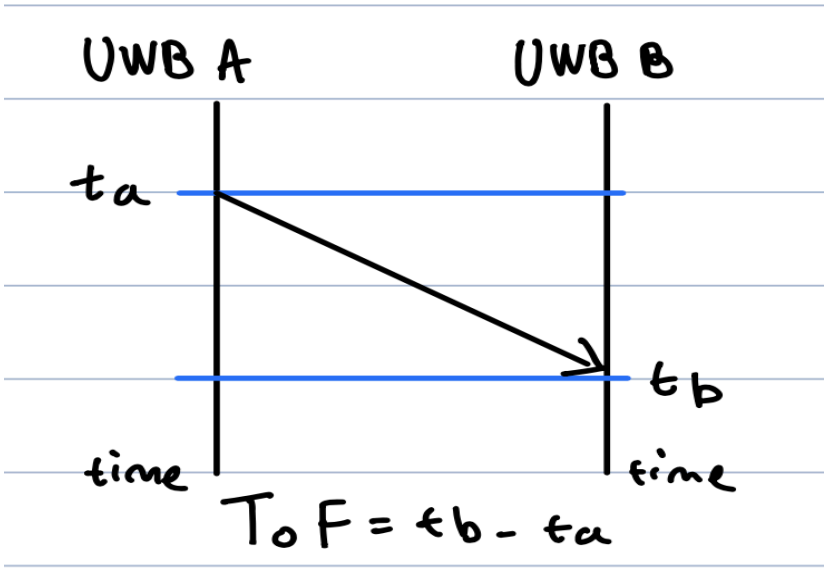

Figure 3: The SSR Time of Flight Algorithm

Figure 3 shows the simple Single-Sided Ranging (SSR) ToF. The ToF, which can be decoded to calculate the distance between the pen and the screen, will be given by this expression:

$$ToF = t_b - t_a$$

Although this looks like a simple technique, it holds a high potential for error when used for the UWB communication system: **the clock offsets and drifts of the UWB transceivers are not synchronized**.

A better way would be to use a Two-Way Ranging (TWR) ToF algorithm. Figure 4 shows the TWR in action between two UWB transceivers. Mathematically, the expression of the ToF becomes:

$$ToF = \frac{r_a - d_b}{2}$$

Note: r_a is the total time it takes for UWB a to transmit and receive back the signal from UWB b. d_b is the process time UWB b takes between receiving the initial signal from UWB a and sending it back.

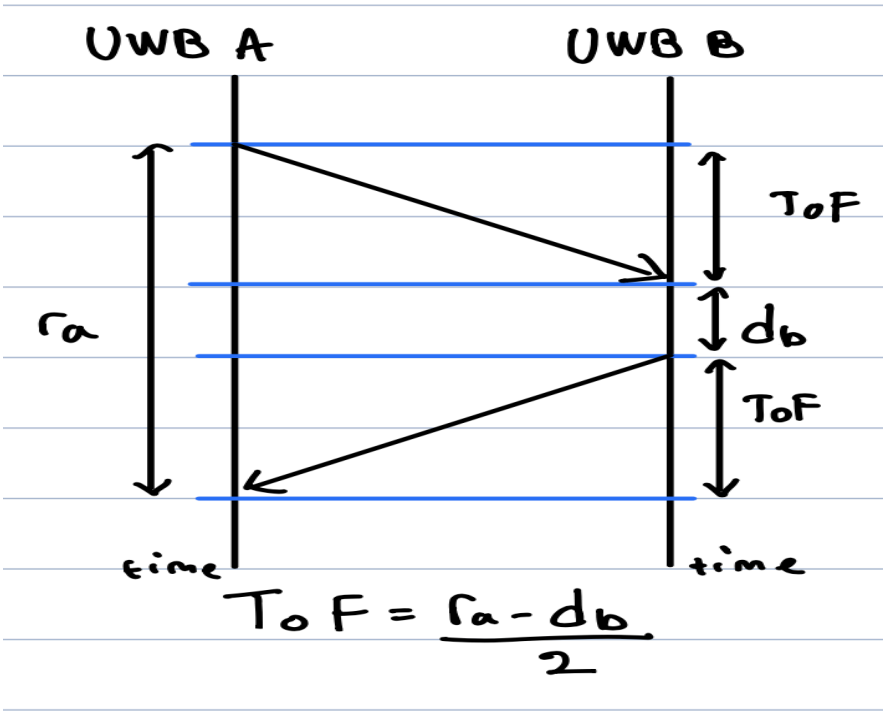

Figure 4: The TWR Time of Flight Algorithm

Now, the clock offsets and drifts will be completely local. This means the error of distance calculation no longer depends on strict synchronization. The error analysis for the TWR algorithm is the following:

Let e_a and e_b be the respective errors due to clock drifts, then:

$R_a = (1 + e_a)r_a$, $D_b = (1 + e_b)d_b$ where R_a and D_b are total time measurements accounting for errors. We then have:

$$ToF_{estimate} = \frac{R_a - D_b}{2}$$

$$ToF\ error = ToF_{estimate} - ToF = \frac{R_a - D_b}{2} - \frac{r_a - d_b}{2} = \frac{1}{2}(e_a r_a - e_b d_b) .$$

However, we know that $r_a = 2\,ToF + d_b$. Hence, we have:

$distance\ error = c * (e_a ToF + \frac{d_b}{2}(e_b - e_b))$, where c is the speed of light.

We have reduced the distance error to be completely independent of clock synchronization between the UWB transceivers. However, an issue remains. Notice that $e_a ToF$ is in the magnitude of nanoseconds due to the relatively high speed of light. However, $\frac{d_b}{2}$ is still in the **milliseconds ToF error magnitude, which makes our distance error relatively concerning**.

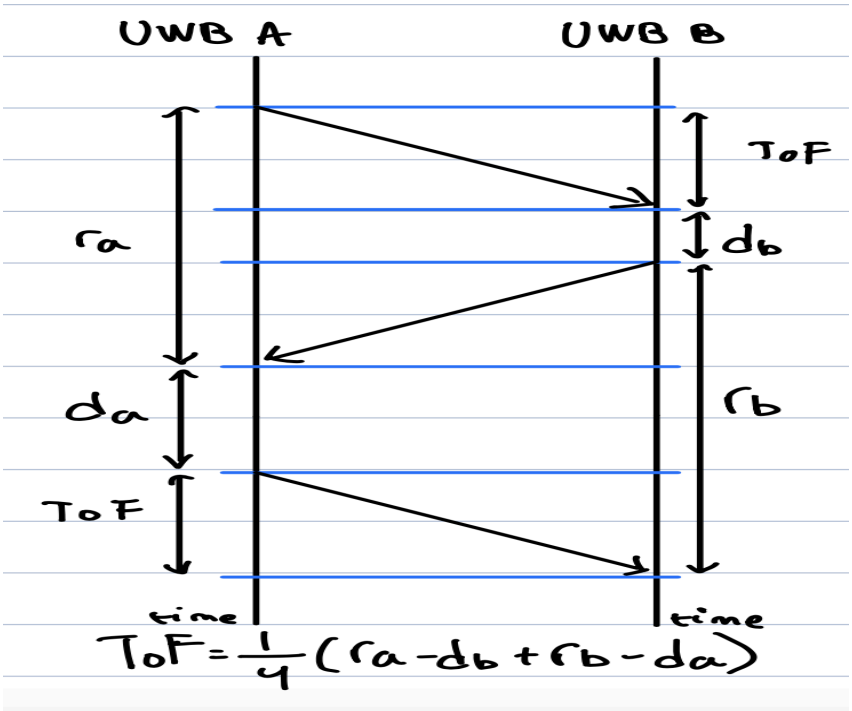

Figure 5: The ADS TWR Time of Flight Algorithm

Note: r_a is the total time it takes for UWB a to transmit and receive back the signal from UWB b. d_b is the process time UWB b takes between receiving the initial signal from UWB a and sending it back.

Further improvements need to be made to make our touch-screen emulation accurate. The Alternative Double-Sided TWR (ADS TWR) is an extension of the TWR to minimize the distance error by eliminating the $\frac{d_b}{2}$ error magnitude, as shown in Figure 5. The algorithm is as follows:

$$ToF = \frac{1}{2}(ToF_a + ToF_b) = \frac{1}{4}(r_a - d_b + r_b - d_a).$$

With this, the error analysis for our ADS TWR algorithm is the following:

$r_a = 2T + d_b$, $r_b = 2ToF + d_a$. Then,

$$r_a r_b = (2ToF + d_b)(2ToF + d_a) = 2ToF\,(2ToF + d_a + d_b) + d_a d_b$$

Then, $r_a r_b - d_a d_b = 2ToF\,(2ToF + d_a + d_b)$. Hence, the time of flight can be expressed as follows:

$ToF = \frac{1}{2}\frac{r_a r_b - d_a d_b}{2ToF + d_a + d_b}$. Substituting $r_a = 2ToF + d_b$, $r_b = 2ToF + d_a$ the denominator to omit ToF, our final expression will be:

$ToF = \frac{1}{2}\frac{r_a r_b - d_a d_b}{2(r_a + d_a)} = \frac{1}{2}\frac{r_a r_b - d_a d_b}{2(r_b + d_b)}$. Accounting for error,

$ToF_{estimate} = \frac{1}{2}\frac{R_a R_b - D_a D_b}{2(R_a + D_a)} = \frac{1}{2}\frac{R_a R_b - D_a D_b}{2(R_b + D_b)}$, where $R_a = (1 + e_a)r_a$, $D_b = (1 + e_b)d_b$

Finally,

$$ToF_{estimate} = \frac{(1 + e_a)(1 + e_b)}{(1 + e_b)}\ \frac{1}{2}\frac{r_a r_b - d_a d_b}{2(r_b + d_b)} = (1 + e_a)\,ToF.$$

$$ToF\ error = ToF_{estimate} - ToF = (1 + e_a)\,ToF - ToF = e_a ToF.$$

Our distance error will then be:

$distance\ error = c * ToF\ error = c * e_a ToF$, where c is speed of light.

We reduced **the ToF error magnitude to nanoseconds** with the ADS TWR algorithm for UWB transceivers communication protocol.

Furthermore, to help the ADS TWR minimize the distance error, we plan to commit to these extra implementation strategies:

1. **Sensor Calibration**: Regularly calibrating the UWB sensors can help mitigate timing drift and errors, ensuring that the ToF calculations remain accurate over time. Furthermore, our calibration process involves taking sensor data at several predefined locations on the screen with the help of the GUI. We can use these data-to-location mappings to help with error correction and outlier detection in real-time.
2. **Error Correction Algorithms**: Implement filtering techniques or error correction algorithms to smooth out data points that are outside the expected range of variation, reducing the likelihood of large positioning errors

3. **Gyroscope Location Adjustment**: The UWB module may not be located at the tip of the pen, causing the UWB sensors to incorrectly sense where the pen's tip is located if the pen is held at an angle. Offsetting the location with the gyroscope data will allow accurate pen tip location.

3. Ethics & Safety

RF Exposure and UWB Compliance: The UWB transceivers emit radiofrequency (RF) signals, which must comply with FCC regulations (Part 15 Subpart F) concerning RF exposure limits and interference control. We will ensure that our device operates within the permitted frequency bands (4.5 GHz) and maintains safe RF power levels to avoid harmful exposure to users and prevent interference with other devices. The Code for Federal Regulations details an entire section on Ultra-Wideband operation:

Data Privacy: Our software subsystems compile data logs and take in the user's screen information and computer environment. Following the ACM Code of Ethics section 1.6-1.7, we must ensure that this data is handled with the highest regard for privacy and confidentiality. Any data collected during calibration or logging must not be misused or shared without explicit user consent.

To mitigate this risk, we will ensure users are informed about what data is collected, aligning with principles of transparency and accountability. We will also only collect data that is strictly necessary for location/mouse data logging and accuracy analysis.

Battery Safety: Our device uses 3.7V rechargeable batteries, which can pose risks like overheating or fire if mishandled. We will follow IEEE Standard 1725 for rechargeable battery safety, mitigating these risks by implementing a power monitoring system that will prevent excessive discharging of the battery

Campus and Lab Policies: We will adhere to the University of Illinois laboratory safety guidelines, which emphasize the proper handling of electronic components and the safe use of tools/materials. This includes wearing appropriate PPE and ensuring that workspaces/lockers are kept organized and free of hazards.

4. References

[1] Espressif Systems. (n.d.). ESP32-S3 products. https://www.espressif.com/en/products/socs/esp32-s3

[2] Hossain, S., & Khawaja, B. A. (2019). A survey on ultra-wideband communication technologies and applications. IEEE Access, 7, 29714-29730. https://doi.org/10.1109/ACCESS.2019.2892430

[3] Qorvo. (n.d.). Getting back to basics with ultra-wideband (UWB). https://www.qorvo.com/resources/d/qorvo-getting-back-to-basics-with-ultra-wideband-uwb-white-paper

[4] RRC. (n.d.). Real-world algorithms for IoT and data science. https://rrc-uiuc.notion.site/Real-World-Algorithms-for-IoT-and-Data-Science-74d8f612f74a4c1689760dafa31ef93d

[5] Raghavan, S., Kittipiyakul, S., & Srikant, Y. (2021). Ultra-wideband channel modeling for wireless communication systems. IEEE Transactions on Wireless Communications, 20(3), 1789-1801. https://doi.org/10.1109/TWC.2020.3041958

[6] U.S. Government Publishing Office. (2022). Title 47—Telecommunication, chapter I—Federal Communications Commission, subchapter A—General, part 15—Radio Frequency Devices, subpart F—Unlicensed National Information Infrastructure Devices. https://www.ecfr.gov/current/title-47/chapter-I/subchapter-A/part-15/subpart-F