DWM1000 UWB Sensor Indoor Tracking Report

Peter Xuenan Fei

Abstract

In this project, I will go over my attempt to implement 2D indoor ranging with Arduino and three UWB Sensors from DECAWAVE(DWM1000). While the system sometimes produces satisfactory results, it has some concerning stability and reliability issues due to hardware limitations.

1. Motivations of the project

Rokid Inc. plans to implement hand gesture recognition on the company's latest product AR glasses. This project explores a possible alternative to the machine learning approach, in which we construct a model, acquire lots of data, and train the processor on the glasses. The UWB approach is easier to implement on the software level, easier to understand, and requires much less data compared to the machining learning approach.

2. How the system works

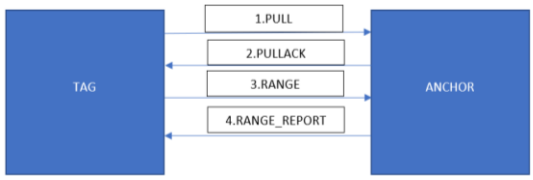
In this project, the UWB sensors are connected to Arduino UNO boards. They communicate with the SPI interface. There is a total of three UWB sensors in the system; two of them are marked as ANCHOR and the other as TAG. The ANCHORs and TAG have different .ino source codes loaded to their Arduino boards.

a. UWB sensors

UWB stands for Ultra-wideband. The sensor chips can communicate with each other using radio waves of a wide variety of frequencies, hence a great bandwidth, which gives its name. This system is proven successful to track a person walking in a room.[1]

The UWB chips we use in this project is DWM 1000 by DECAWAVE. This is one of the most accessible UWB chips in the market.

The chips communicate with each other to implement two-way ranging as shown in diagram 1 below.



(Diagram1: working mechanism of one-to-one ranging between two UWB sensors. The timestamps when each of the messages (PULL, PULLACK, RANGE, RANGE_REPORT) are sent are recorded and transmitted between the two chips. ANCHOR then uses these data to calculate the distance between itself and the TAG using a two-way ranging TDOA (time difference of arrival) algorithm.[2] The data for distance is then stored in TAG, and is used for further calculations.)

b. Projection onto 2D plane

In this project, we aim to compute the 2D coordinates of the TAG on a plane. One of the ANCHORs will be defined as the origin of that plane. The other ANCHOR is placed at a fixed distance a away from the origin, which has the coordinates (a, 0) in the plane. The tag is placed at a fixed distance z away from the plane of the two ANCHORs. Given the absolute distance from TAG to each of the ANCHORs, we can calculate the projected distance from TAG to the ANCHORs in the plane we are interested in using Pythagoras Theorem.

c. Trilateration

Now that we know the two projected distances between the TAG and the ANCHORS on the plane, we can find the coordinates of the TAG on the projected plane. Suppose we know that the projected distance from TAG to the first ANCHOR(defined as the origin) is r_1, and the projected distance from TAG to the other ANHOR is r_2, we can locate the TAG as the intersection of two circles: one centered at the origin with radius r_1, and the other centered at the second ANCHOR (a, 0) with radius r_2.

Expressing this idea mathematically, we get a set of simultaneous equations:

$$x^2 + y^2 = r_1^2 \ (1)$$
$$(x - a)^2 + y^2 = r_2^2 \ (2)$$

Where x and y are the x-coordinate and y-coordinate of TAG in the plane of our interest.

Subtract (2) from (1), we get:

$$2ax - a^2 = r_1^2 - r_2^2$$

Solve for x, we have:

$$x = \frac{a^2 + r_1^2 - r_2^2}{2a}$$

Substitute into (1) and solve for y, we have:

$$y = \pm\sqrt{r_1^2 - x^2}.$$

3. Source code implementations

Unless mentioned in this section, the source codes of this project are taken from the library for DW1000 developed by Thomas Trojer and Ludwig Grill.[3] This library also provides several simpler examples which are very helpful for getting to know the setup of the system.

In DW1000Ranging_TAG.ino:

Added algorithms to calculate projected ranges and coordinates.

Now supports the option of frames per second (fps) test. It counts the total ranges the TAG receives from any ANCHOR in the past 5 seconds and computes the average value in that 5 seconds. (It is able to achieve 15Hz fps, which means on average, 7.5 range data are sent from each ANCHOR every second.)

Added hard-coded values of the z-coordination of the TAG (z) and the distance separating the two ANCHORs (a).

Added mechanisms to store the most recent three projected ranges from each ANCHOR and calculate the average projected ranges.

Added mechanisms for the TAG to recognize each ANCHOR. Each ANCHOR is given a random 8-byte integer as its identifier or address. TAG recognizes the first ANCHOR it connects with as the origin. It must be able to remember the origin's address so that it doesn't confuse the ranges sent from two different ANCHORs, one at (0, 0) and the other at (a, 0).

In Anchor2.ino:

Almost identical to DW1000Ranging_ANCHOR.ino from the library; but supports fps test.

In DW1000Ranging.h:

Changed the values of DEFAULT_RESET_PERIOD, DEFAULT_REPLY_DELAY_TIME, and DEFAULT_TIMER_DELAY. This is because of the added workload on the Arduino microprocessor to store extra data and make extra computations. These values affect the ranging results since they are set based on less workload. Therefore, leaving them unchanged makes TDOA measurements inaccurate.

In DW1000.cpp:

Adjusted antenna time delay value in line 1086. Modifying this parameter can increase or decrease the magnitude of ranging data in general. To decrease distance data, increase this value slightly, and vice versa.

All these changes on the software level works as intended: The system is capable of producing sensible coordinates when all the key hardware parts are working properly.

The system is also well-calibrated. When testing one-to-one ranging accuracies, the mean distance acquired over a long period of time is less than 2cm adrift from the actual distance of 60cm, though there are fluctuations of the data ±5cm from the mean value.

4. Issues and limitations

As mentioned in the section above, there are considerable fluctuations of the data given the context of our goal. This is due to instabilities of the connection, which arises because of a combination of multiple possible causes like weak antenna signals and unstable power supply. The alleged timestamp resolution of the DWM1000 chips is 15.65 picoseconds, which translates to 4.5 millimeters in terms of distance calculated from TDOA algorithm. However, I was not able to reproduce this level of accuracy with the chips I have, not even with a TAG and a single ANCHOR. In addition, the producer of the chip, DECAWAVE, claims that the chips have the precision of 10 cm[4], which is not good enough for our purpose. However, this limitation can be mitigated by calculating the mean distance first then use the mean distance to calculate the coordinates. Nevertheless, it was not enough, as if we keep the TAG still, its x and y coordinates still vary by over 3cm over time, which is unacceptable given the expected values of x and y coordinates are approximately 30cm.

An even worse issue with the system is that it produces inaccurate results when multiple ANCHORs are connected to the TAG. The second ANCHOR that connects to the TAG always produces a range value greater than expected. I tried to set a bias value for the second ANCHOR and subtract that value from the range result given by the second ANCHOR, but the magnitude by which the result differ from the true

value is nondeterministic. I reckon this could be caused by the excessive workload on the Arduino microprocessor to store data and make computations, as this issue is not noticeable when we upload the original examples in the library to the Arduino boards.

Besides accuracy issues, the system also has hardware reliability issues. About once in 10 booting attempts, the second ANCHOR that connects to the system produces negative ranging data, which is nonsensical. To resolve this problem, we can unplug the first ANCHOR. The range values from the other ANCHOR will be accurate again after about 2 seconds. We plug in the other AHCNOR again, and it will work as we want. However, there is no software fix to this problem.

The issue mentioned in the paragraph above is always present if we attempt to connect 4 devices in the system (1 TAG, 3 ANCHORs), and the fix doesn't work. This caused me to switch to the current setup with 2 ANCHORs. If I could connect another ANCHOR, I would be able to get the 3D coordinates of the TAG, not needing to hard-code its z-coordinate.

5. Challenges

Most of the obstacles while doing this project was hardware related. At first, I struggled to get sending and receiving functions working. Either the chips can't recognize each other, or unidentified characters are sent in the protocol. I suspected that some pins on my Arduino boards are not working properly, so I reprogrammed the pin numbers myself, including the interrupt request (IRQ) pin. I found out later, thanks to some help from senior colleague Aiden, that the IRQ pin is set to be pin 2 on Arduino UNO, so I mustn't alter its assignment. The problem was resolved after I disconnected the UWB chips from the Arduino and rewired everything. I learned that to make sure the connection works, I need to plug every pin on the chip firmly into the pits on the Arduino board.

I also had a hard time setting the parameters to calibrate or tune the system. At first, I thought I shouldn't modify the constants in the original library, as I have limited understanding of how they are acquired. Later, as I realized that the ranging results are dependent on the execution time of the Arduino microprocessor, I started trying different values with these constants, with success to acquire accurate data in one-to-one ranging setup. It's worth noting that the optimal values of these parameters may differ from time to time with different environmental factors, so it's a good idea to calibrate the chips before further testing.

6. Possible Concerns

Even if all the limitations mentioned in the previous sections could be resolved, there are still concerns of this approach to implement hand gesture recognition.
Compared with the machine learning approach, this approach needs the user to wear a ring with UWB sensor on her finger. While this can make gesture recognition more specific (only movement on one finger is recognized), we need a physically smaller UWB chip and antenna so that the device fits onto the users' finger. This may further jeopardize the already weak and unstable signal strength of the antenna.
The UWB sensors also generate a lot of heat. This means it is probably not very power efficient and may affect battery life of the glasses. In addition, the heat on the UWB chip will be noticeable for the user as she wears it on her finger. After a long period of tracking, the heat will make the user uncomfortable.

7. Conclusions

After all these explorations with the possibility of implementing hand gesture recognition with UWB sensors, we can safely conclude that it is not the way to go. While the current generation of commercially available UWB sensors are competent of tracking objects meters away from each other (like a walking person in a room), it does not suffice in such a task as hand gesture recognition, whose demand for accuracy is too high. However, this approach does have its advantages, so it's worth looking back if there is a breakthrough in UWB technology.

8. References

1. "Localino: Open Source Indoor Location System (Arduino + Decawave DWM1000)" – Localino.net, https://www.youtube.com/watch?v=t1kuF8y7Lwg
2. "DW 1000 User Manual", page 227 – DecaWave, https://www.decawave.com/sites/default/files/resources/dw1000_user_manual_2.11.pdf
3. "arduino-dw1000" –Trojer, Thomas, https://github.com/thotro/arduino-dw1000
4. "DWM 1000 Datasheet", page1 – DecaWave, https://www.decawave.com/sites/default/files/resources/DWM1000-Datasheet-V1.6.pdf