**DWM1001 UWB Tutorial & Integration with CAARS project**

* The positioning and localization for the connected and autonomous drone was achieved with the help of an ultra wideband beacon(UWB).
* One of the important applications of using UWB technology is a RTLS network. RTLS stands for “Real time locating systems” which acts as a localization mechanism for indoor conditions or at the confined spaces like offices, airports, shopping malls, etc.
* We can use 2 methods to evaluate the UWB data. First method is through an android application and the second method is directly using the DWM1001’s UART Shell Mode. The UART Shell mode elaborates about how the DWM1001 module communicates with the Jetson Nano via UART(USB)

*Preparation of Anchors:*

* Anchors are the units that are placed as reference points at various locations in the indoor conditions.
* Mount the anchors at the high level and within the line of sight as mentioned in the Figure 10.
* Minimum 3 anchor units are required to establish an RTLS connection.
* Power them using batteries or a power bank or a USB power adapter directly plugged to supply.

*Preparation of Tag:*

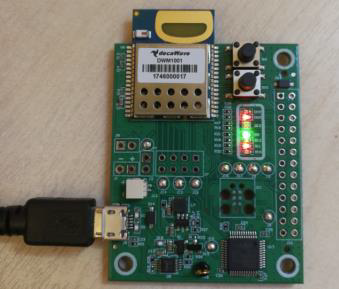
* Tag is a main unit deployed on the device whose location needs to be tracked in the inner conditions.
* Minimum 1 tag unit is required to establish a connection. Powering it is similar to anchor.



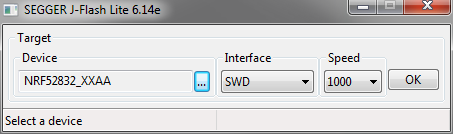
* Below are the steps to be performed when we power it up for the first time:

**DWM1001 Preevaluation steps**

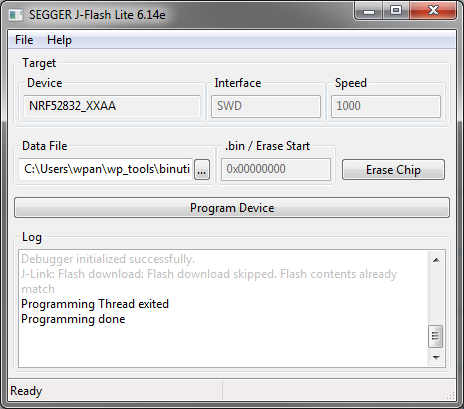
* DWM1001 comes with a predefined image inside the microcontroller. But we need to ensure that it is loaded with the factory image before any configuration can be done
* Download the DWM1001 Firmware package (<https://www.decawave.com/1001-license/>) and extract it into a workspace
* Download the J-Flash Lite from the Seggar website (<https://www.segger.com/products/debug-probes/j-link/technology/flash-download/>)
* Connect the DWM1001 module with a micro USB data cable



* Open J-Flash Lite and choose nrf52832\_XXAA as Device and SWD as interface, use default speed 1000. Click “OK”.



* Click “Erase Chip” to do a full chip erase.
* In Data File,browse to the hex file provided in the DWM1001 firmware package that we extracted earlier (/DWM1001/Factory\_Firmware\_Image/DWM1001\_PANS\_R2.0.hex) to flash, click “Program Device”



* Follow the same steps for all the DWM1001 modules that come in MDEK1001 Development Kit.

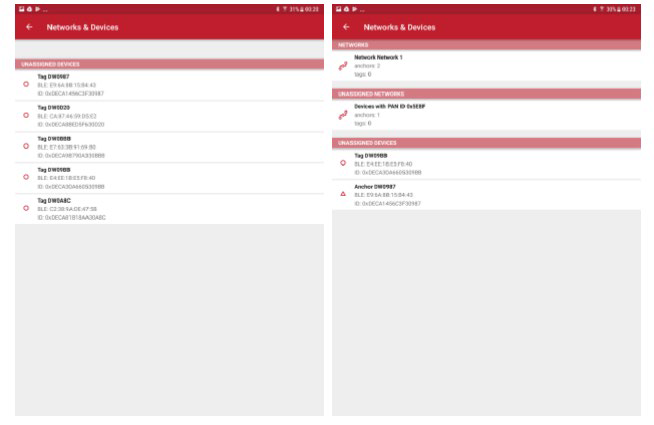
**DWM1001 evaluation: First Method (Android App)**

* The first method for evaluation is described in detail in MDEK1001 Kit User Manual (<https://www.decawave.com/sites/default/files/mdek1001_system_user_manual.pdf>)
* DWM1001 can be configured as tag, anchor and listener and different types of configurations can be used depending on the applications
* For the sake of simple application, I am using 3 Anchor+1Tag configuration. Below are the step by step method to configure the DWM1001 and record the localization data
* Hence,we perform the pre evaluation steps, as explained above, for only 4 modules
* The anchors and tag communicate with each other via UWB. There is an android application that controls and configures these sensors over BLE.
* This application shall be used to monitor the relative distance between the tag and the anchors to pin-point the exact coordinates of the tag. Here are the **evaluation steps** that was performed using the android application:

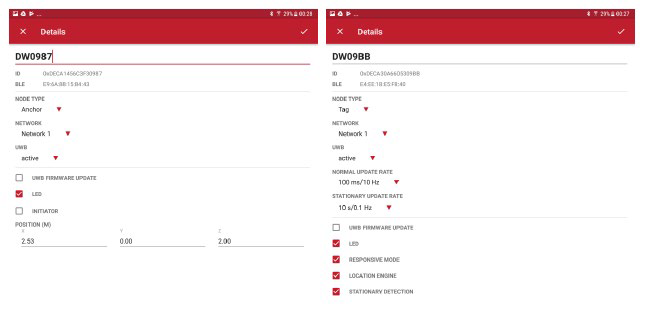
1. Mount at least 4 anchors on the wall as per the positioning mentioned in the instruction manual and assign the remainder as the tags whose location needs to be monitored.
2. Start the device discovery and perform the configuration
3. Auto position these units and measure their distances with respect to the initiator.
4. Upload the floor plan, modify the units and change the refresh rate using the android app and repeat the above steps to measure the sensor performance.

Detailed description of using an android app is as follows:

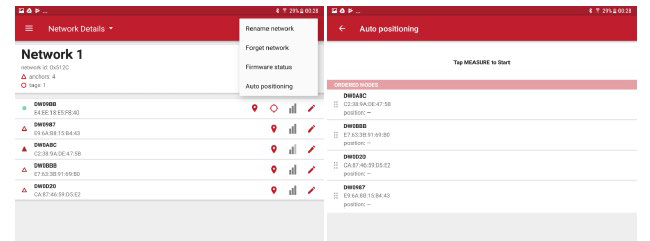
* Download the Decawave RTLS Manager apk file from the decawave’s website: <https://www.decawave.com/product/mdek1001-deployment-kit/>
* Power on all the modules at their factory settings and start the device discovery in app’s home screen
* The devices will be detected as tags and devices will be grouped as unassigned devices, unassigned networks and networks as shown below. Select the devices and provide a network name and assign all the unassigned devices to these networks:

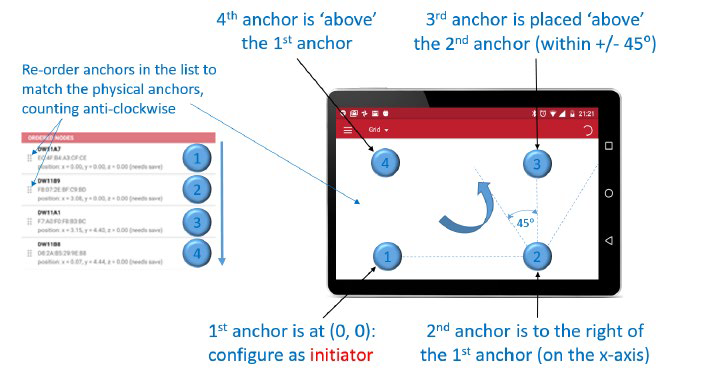


* The configuration of the anchor needs to be done as mentioned in figure 8 by clicking on the edit section. Select one of the anchors as an initiator and keep the value of x,y,z as (0,0,0). Ensure UWB is active.
* The configuration of the tag needs to be done as mentioned by clicking the edit button. Make sure the UWB and BLE is active.

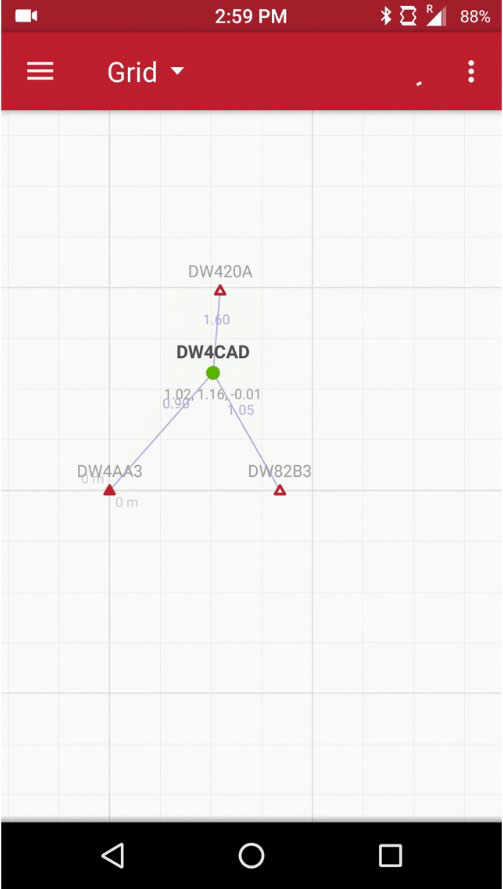
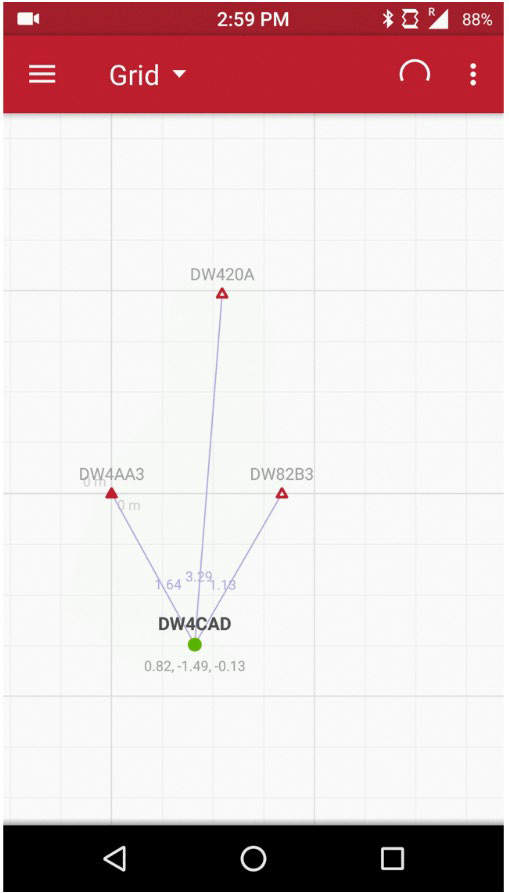


* **Auto positioning:** The auto positioning feature is used to automatically measure the distances of the anchors with respect to its initiator and set up an RTLS network. This may result in a small amount of error, but it is recommended that the line of sight between the anchors is maintained. Auto positioning can be selected as explained below. Once all the anchors are present in the list, reorder them from top to bottom and start measuring the distance by clicking on MEASURE. The auto positioning rules are explained below.





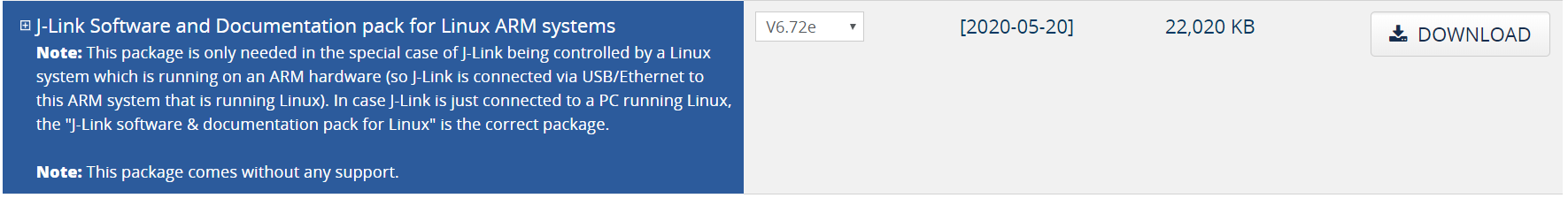
* Ranging starts in the auto positioning mode once all the anchors and tags are in the network and in the network settings, tap on the “GRID” option and see the dynamic update of the tag distance with respect to anchors and initiator. Below shows the demonstration of the RTLS network.



***Second Approach- UART Shell mode:***

We connect the DWM1001 module, which needs to be configured as a tag and placed in the quadcopter to monitor its location in the indoor environment, to the Jetson Nano via USB connection. Below is the step-by-step evaluation method in Jetson Nano’s Ubuntu 18.04.

* Go to the website:<https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack> and download the J-Link Software and Documentation Pack for LINUX ARM systems.



***Installation of J-Link Driver:***

* Once downloaded, extract the tgz file using tar -xvzf or installing a 64 bit .deb file can be done using the below command



* All the J-Link executables are installed in **/usr/bin**. In order to execute the J-Link driver installation, we need to add a .rules file specific to SEGGAR devices.
* In order to run JLinkExe with standard user rights you have to do the following:

- Copy the file "99-jlink.rules" provided with this J-Link software package in the /etc/udev/rules.d/ directory using this command:

**sudo cp 99-jlink.rules /etc/udev/rules.d/**

* Restart your system
* After restarting, open the terminal and navigate to the folder where it contains all the JLink executable files.
* Executable files can be recognized by the extension .run. Before the installation procedure can be started, the user rights of such files need to be changed.
* Type the following command:



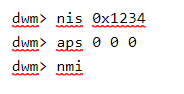
* When asked for, type the required password and press Enter. Now the file can be executed by the current user with root privileges.
* Now the file can be run in the terminal by typing the following command in the terminal:



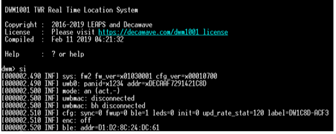
* If the installation file doesn't run, add “sudo” before the above command and it should start installing the file
* I followed the above steps for the JLinkEXE executable file and after the installation was completed, the JLink driver for SEGGAR devices was ready to use.
* After installing the driver and restarting the Nano, we will be able to see the SEGGER device at /dev/ttyACM0 using the command in below figure:

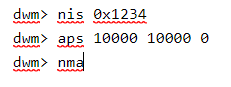


* Since the devices come with the factory settings, we go ahead and enter the shell mode by pressing “Enter key” twice.
* We connect the anchors to the Jetson Nano and configure them in Shell mode as explained in DWM1001 Gateway Quick Deployment Guide(<https://www.decawave.com/wp-content/uploads/2019/03/DWM1001_Gateway_Quick_Deployment_Guide.pdf>).
* We configure the DWM1001 connected to Jetson Nano using the DWM1001 Firmware API Guide (<https://www.decawave.com/wp-content/uploads/2019/01/DWM1001-API-Guide-2.2.pdf>)
* The first Anchor needs to configured as an initiator which can be achieved using the below commands:

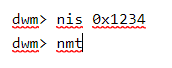


* “nis” sets up the PAN id to 0x1234, “aps” sets the initiator position to x=0,y=0,z=0, “nmi” configures the node as initiator and resets the device.
* Remaining anchors can be initiated directly in shell mode using the below commands:





* “nma” configures the node as normal anchor and “aps” set the distance as x=10000, y=10000, z=0(mm).
* Similarly, tag can be configured in shell mode using the below commands



* “nmt” sets the node as tag and resets the device and “si”provides the node status as shown
* After the nodes are configured, the system looks like the image as follows:

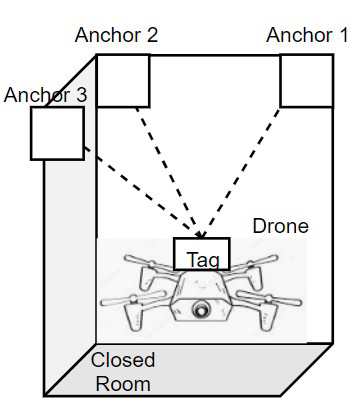


**Integration with CAARS project**

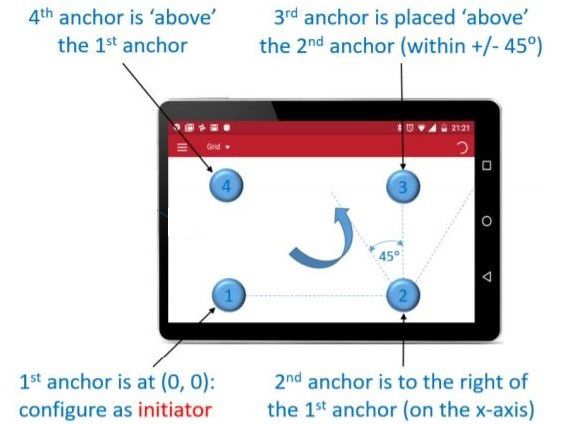
* DWM1001 interacts with Nvidia Jetson Nano through UART which works with the camera to provide object tracking.



* After powering up the tag and anchors, we configure them using UART Shell Mode when DWM1001 is connected to Jetson Nano. The configuration steps are mentioned in the previous section of the DWM1001 Deployment Guide.
* After configuring the tags and anchors, we place the anchors inside a closed room as shown in figure in previous page.
* The tag is connected to Jetson Nano(Drone). As per the DWM1001 instructions on the firmware guide, we are positioning the anchors to establish the RTLS network.
* Below is the image setup of the DWM1001 tags(drone) and anchors placed in closed room



* To evaluate the performance of DWM1001 in indoor conditions, we placed 3 DWM1001 at different corners of the room and configured them as the anchor as a part of an experimental setup. The DWM1001 along with the camera and drone is configured as a tag.



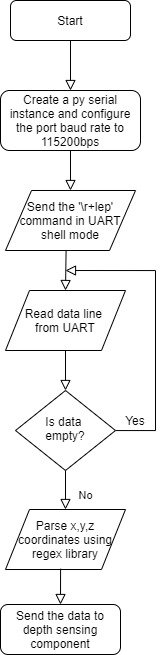
* As per the instructions above, we need to place all the anchors within their line of sight and at the same z-plane.

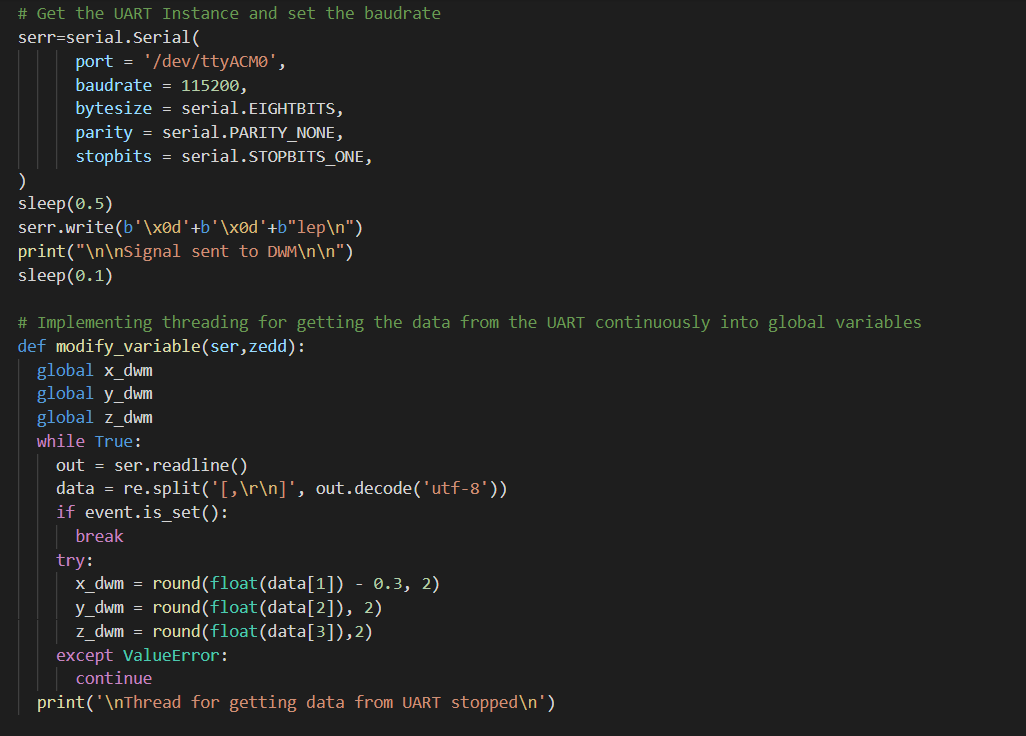
*“****Configuring them at different heights will lead to failure in anchor configuration and RTLS network will not be established.”***

* We evaluated the performance of the DWM1001 by comparing the values of (x,y,z) of the tag concerning the ground truth.

***Software integration:***

* DWM1001 interacts with Jetson Nano over UART communication. We used UART shell mode to retrieve data from the tag.
* I have developed a Python Script to parse the data from the string that is sent by DWM1001 to Nano.
* The following flow chart and the code snippet explains the step-by-step execution of our Python script. This python script is integrated with the depth-sensing component to achieve object tracking.

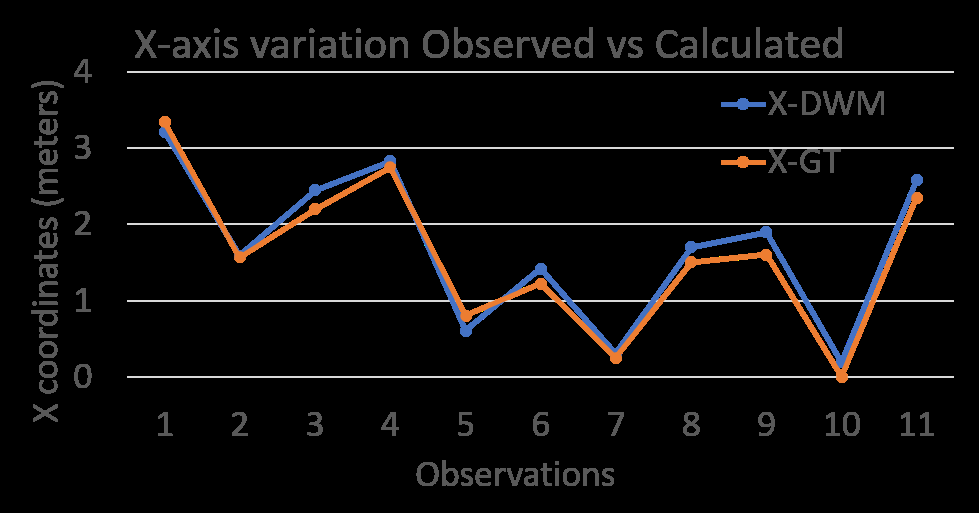
****

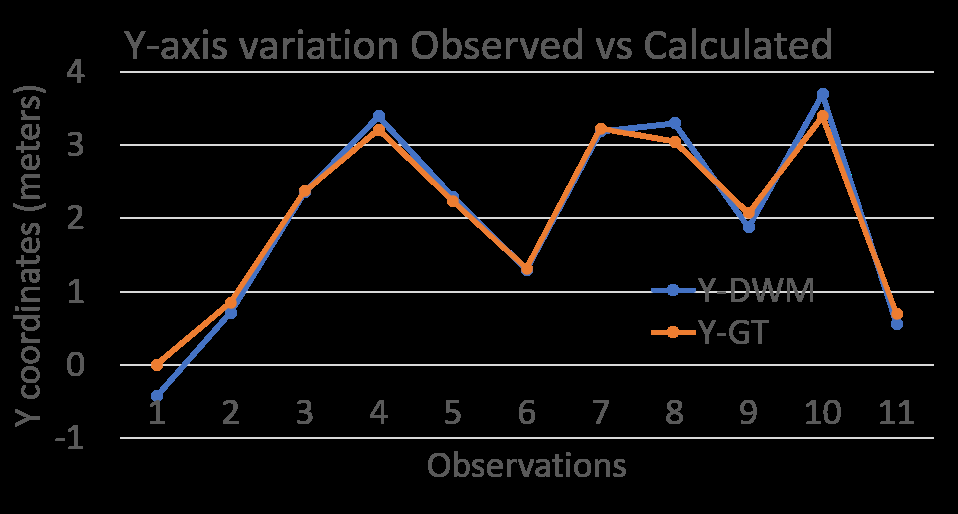


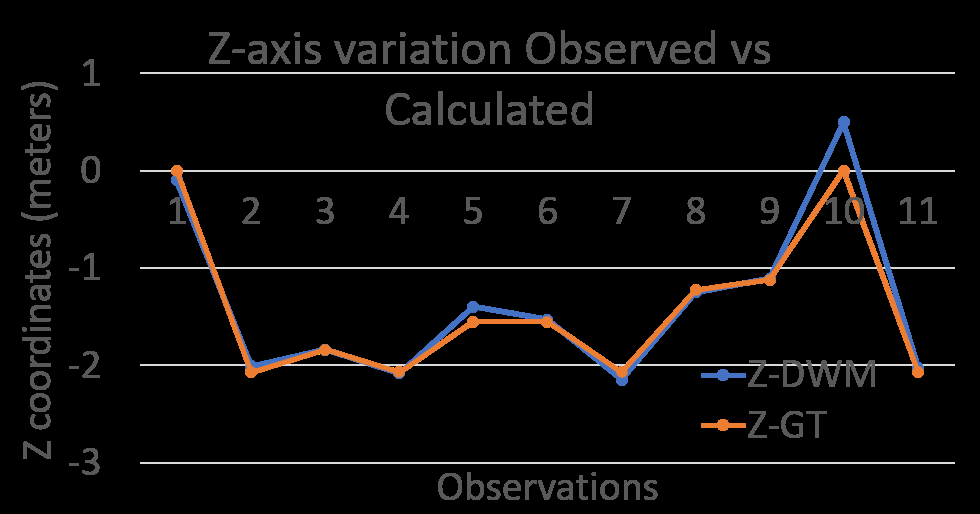
* We have created a serial object with configurations such as port,baud rate, byte\_size, parity and stop bits. This will also open the serial port.
* We send the command “double enter+lep” and dwm1001 starts sending data in a string format “**POS x,y,z, QF”**
* The regex library is used to parse the values and store in a list.
* Sometimes, we do not receive any data from UART to which regex throws a “ValueError” exception. We handle this exception using a “try-except” block.
* The dwm coordinates are taken to the depth sensing module which computes the absolute coordinates of an object based on the calculation with the camera coordinates.
* DWM UART communication takes place in a thread where another main thread captures this data and integrates it with the ZED Camera.

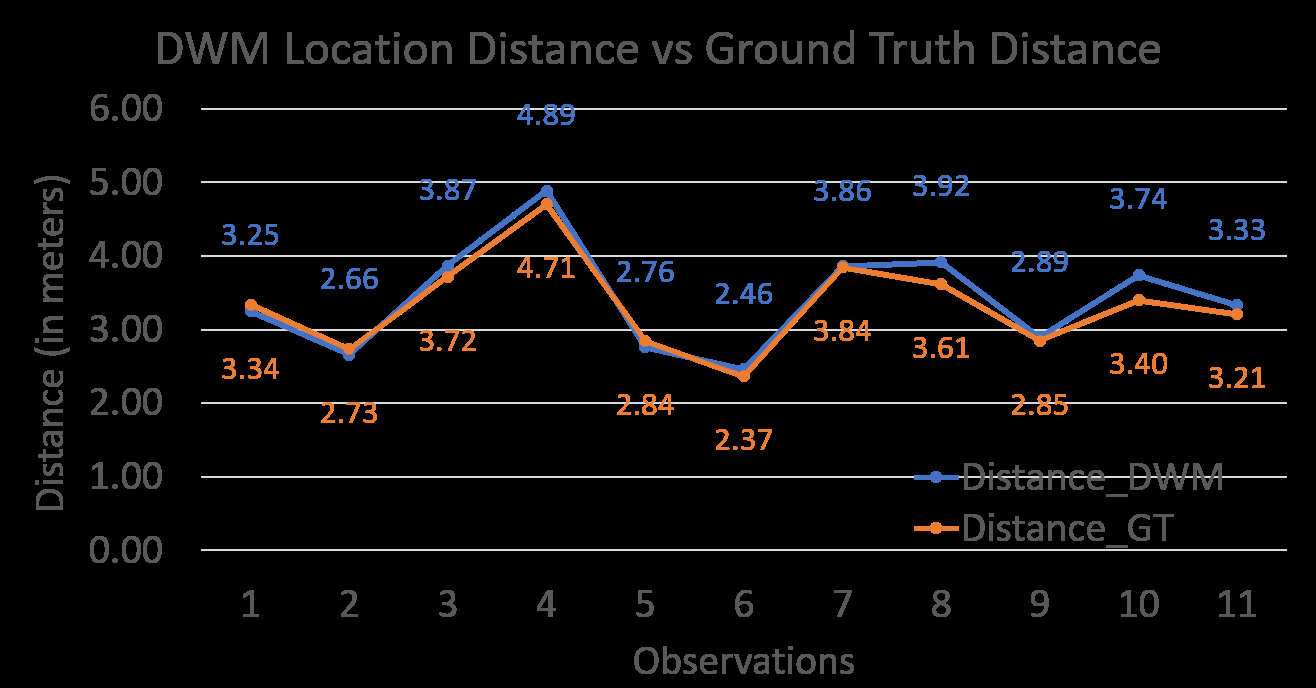
***Results:***

* During the experimental setup, we placed anchors at different locations inside the room and measured the coordinates.
* The coordinates of Anchor1 is (0,0,0), Anchor 2 is (3.61,0,0) and Anchor 3 is (3.8,3.61,0). Here, Anchor 1 acts as an initiator(origin) and the measurement unit is in meters.
* We also evaluated the values of (x,y,z) of tag at different positions inside the closed room and measured the corresponding ground truth and analyzed the data in excel by calculating the percent error parameter.
* Below is the graph showing the comparative analysis of DWM1001 coordinate values versus the actual ground truth values.









* We also computed the Euclidean distance of tag to Anchor 1 and evaluated the variations to calculate the percentage error. The accuracy of the DWM1001 distances measured relative to the anchors was around 96 percent.
* We also evaluated the variations of x y and z and calculated the average percentage errors. The percentage error for x,y,z came out to be 11.16%, 6.6%, and 2.1% respectively. In summary, we were able to achieve the maximum error of ±0.41m for the x-axis, ±0.25m for the y-axis, and ±0.04m for z-axis movements of the DWM1001 tag attached to Jetson Nano.

**Challenges and Solutions**

1. The android app was not able to keep the tags and anchors configured into the network.The UWB signal was lost and the tag/anchor fail to communicate

**Ans: Set the refresh rate inside the configuration setting at 100ms/10Hz and stationary update rate at minimum (10s) solved the problem**

1. The Linux could not identify the DWM1001 device/ couldnt find the device since there is just a number “006” getting popped up inside “/dev/bus/usb”. However connecting UART pins on Jetson Nano works well.

**Ans: Since UART pins were used by other modules in our project, my only option was to connect the DWM1001 through USB. After some research, I found that The DWM1001-DEV has a JLINK OB which allows debug and a serial connection (VCOM).Hence, I need to have the J-Link Segger Driver installed on your Linux machine. I solved it by directly installing the driver from the Seggar website with the correct version and platform and it solved the problem.**

1. The dwm1001 data was not getting parsed correctly at UART and it was throwing a ValueError in python code

**Ans: Since we were getting the blank values on UART, the regex module was unable to parse the blank values based on the parameters. So we implemented a “try-except” module which was used to continue the loop when we got the blank value. The incorrect regex parsing parameters caused the incorrect parsing of the UART data.**