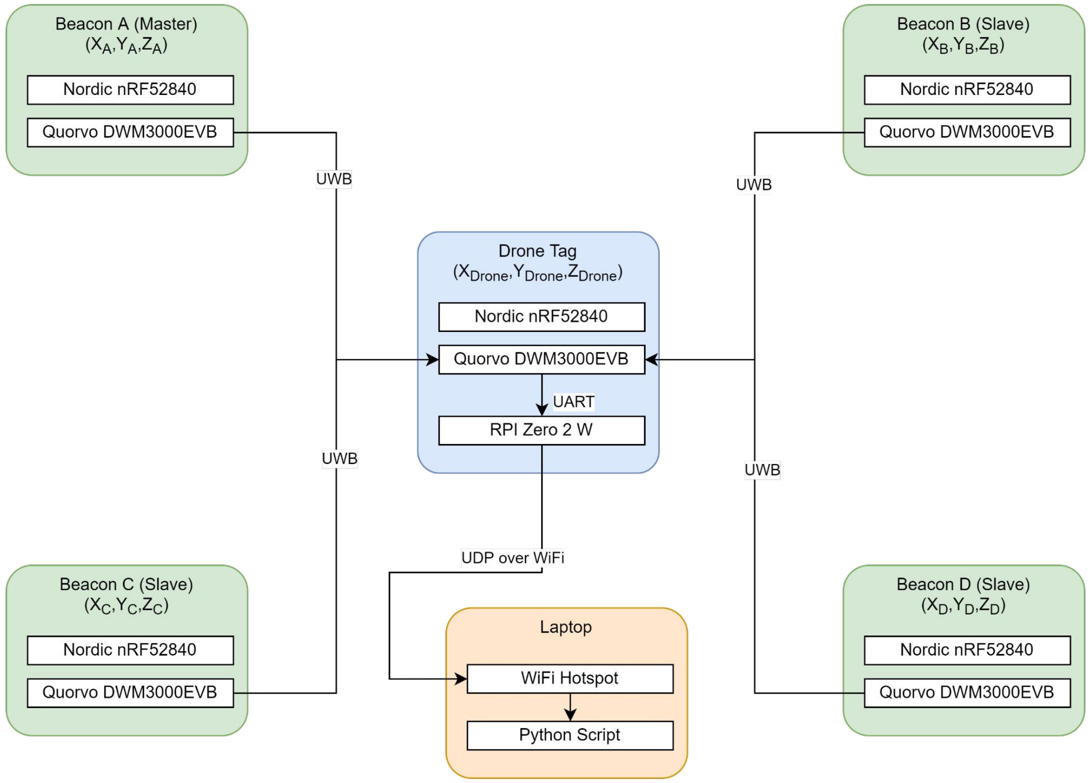
DLPS Proof Of Concept

Overview

System overview (4 beacons, 1 tag)

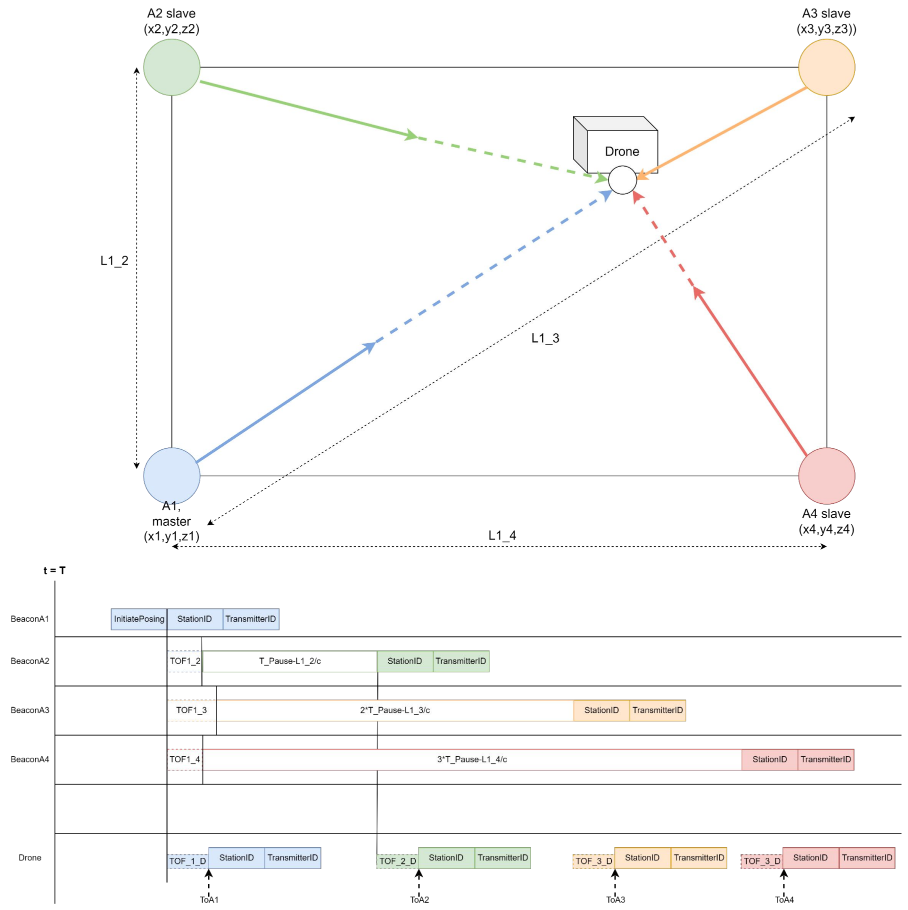


UWB Positioning principle

System components:

- 1 master beacon
- 3 or more slave beacons
- Any number of drone tags

Each beacon in turn transmits a data frame. Drone tag records the time difference of the reception and thus the difference in distances to each beacon (compared to the master). From this it calculates its XYZ coordinates



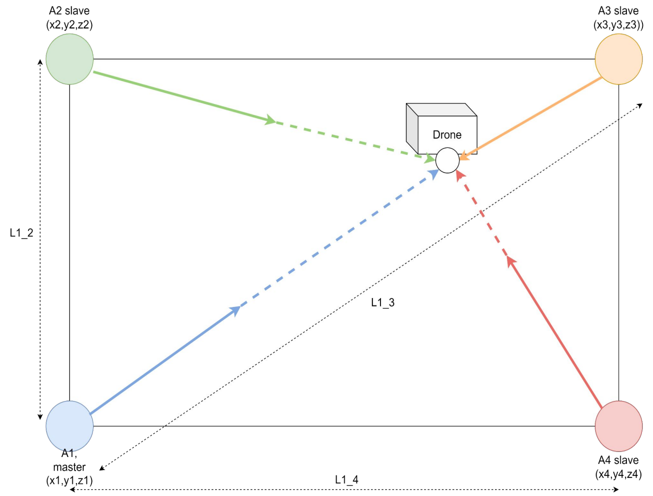
UWB Sequence

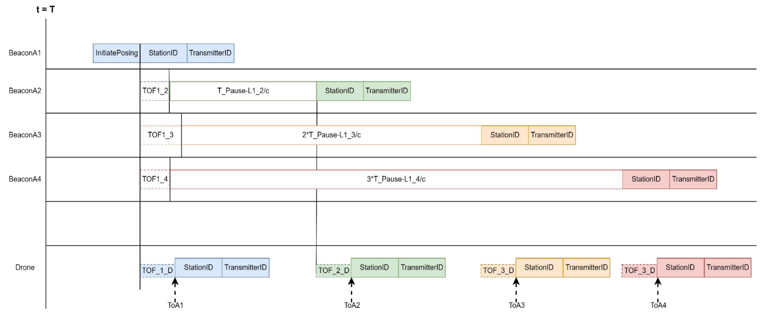
Ranging sequence initiated by Master beacon node (A). On reception of initiation signal, each beacon waits:

$$(x-1)*\text{min\_delay}-\text{Dist}_{\text{x-Master}}*c \text{ seconds}$$

where x its the beacons number (A=1, B=2 etc).

Each ranging signal contains the sender beacon id, its delay relative the master signal, and a sequence number (to discard whole sequence in case of missing messages)

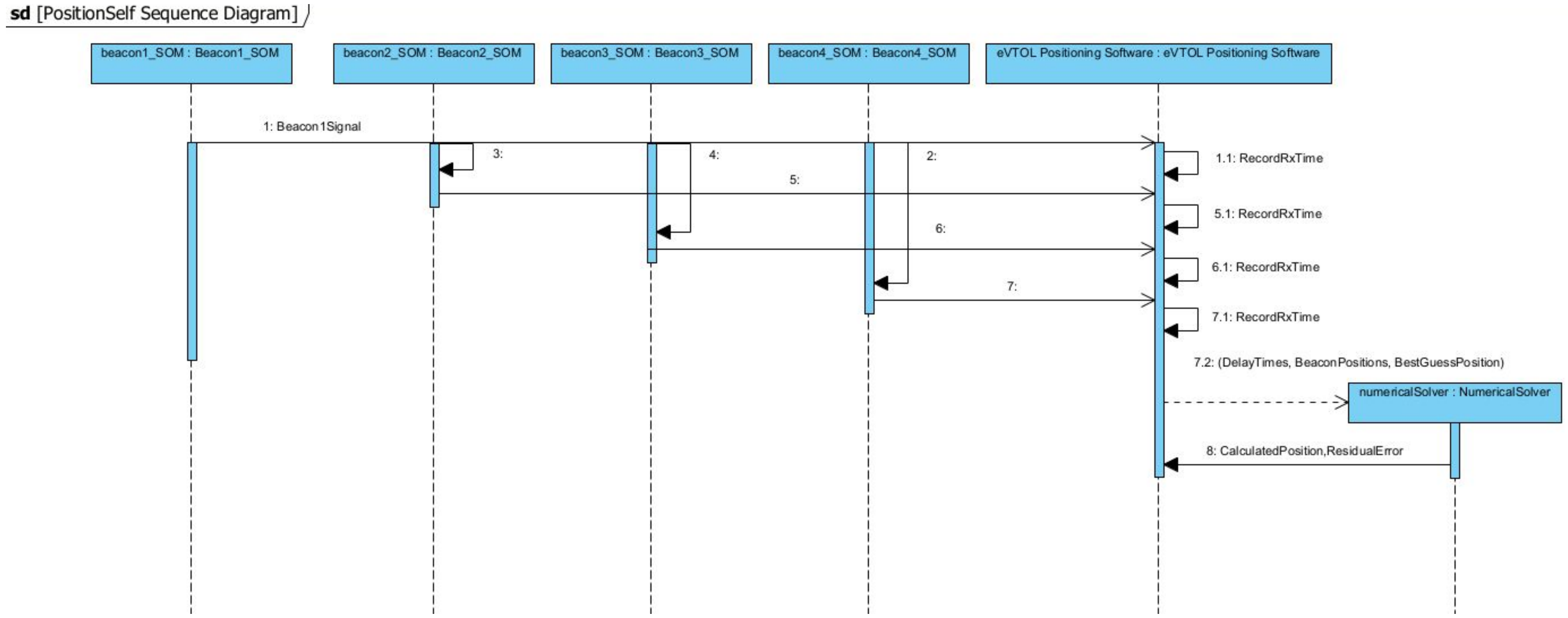




UWB Sequence, startup

- On Beacon A startup:
  - Do double sided to-way ranging for all slave beacons
  - When completed, regularly initiate drone positioning sequence
- Slave Beacons:
  - When master ranging signal received, wait (x-1)*min_delay-Dist$_{x\text{-Master}}$*c seconds before sending own ranging signal (using same sequence number as master signal)
- Drone Tag:
  - On reception of each beacon ranging signal (if sequence number matches master message): Record reception time.
  - On reception of final signal (nr 6 in POC): If all signals have been received (with matching sequence nrs), compute TDoAs and distance differences, (avg 5 latest), package message and send on UART pins to RPI.

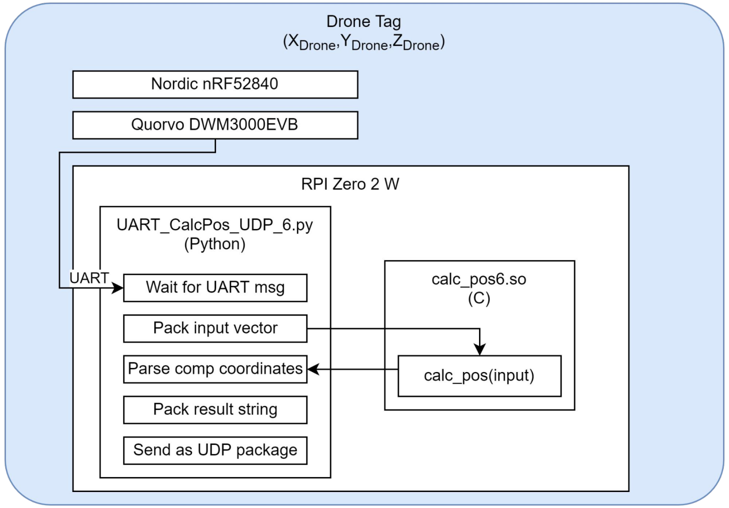
Sequence diagram


sd [PositionSelf Sequence Diagram]
beacon1_SOM : Beacon1_SOM
beacon2_SOM : Beacon2_SOM
beacon3_SOM : Beacon3_SOM
beacon4_SOM : Beacon4_SOM
eVTOL Positioning Software : eVTOL Positioning Software
1: Beacon1Signal
3:
4:
2:
1.1: RecordRxTime
5:
5.1: RecordRxTime
6:
6.1: RecordRxTime
7:
7.1: RecordRxTime
7.2: (DelayTimes, BeaconPositions, BestGuessPosition)
numericalSolver : NumericalSolver
8: CalculatedPosition,ResidualError

Raspberry PI Zero 2 W

Pythonscript listening for UART messages. On reception from nRF52840:

- Unpack individual differences in meters (and beacon coordinates)
- Feed input to compiled C-function (main5.c, main6.c uses data from 5 resp. 6 beacons for comparison).
- C-function solves equation system using GNS least-square numerical solver (for z>0, soft constraint)
- Packs results and some data into a string and sends via UDP to laptop over WiFi (hotspot)
- Receiving python script on laptop unpacks, records and plots data



Coordinate Calculation

Example 2D coordinates (3 beacons):

- ToA = [1,2,3] ns (rel. drone clock) => TDoA = [1,2] ns ([$\text{TDoA}_{B\text{-}A}$, $\text{TDoA}_{C\text{-}A}$])
- Beacon positions (known) = [X_A,Y_A], [X_B,Y_B], [X_C,Y_C]
- Drone position (unknown) = [X_{Drone},Y_{Drone}]
- Distance difference equations are:
  - $\text{TDoA}_{B\text{-}A} = \text{Dist}_{B\text{->}Drone}/c - \text{Dist}_{A\text{->}Drone}/c = (\text{Dist}_{B\text{->}Drone} - \text{Dist}_{A\text{->}Drone})/c = 1$
  - $\text{TDoA}_{C\text{-}A} = \text{Dist}_{C\text{->}Drone}/c - \text{Dist}_{A\text{->}Drone}/c = (\text{Dist}_{C\text{->}Drone} - \text{Dist}_{A\text{->}Drone})/c = 2$

$$\Leftrightarrow$$

$$TDoA_{B-A} = \left(\sqrt{(X_{Drone} - X_B)^2 + (Y_{Drone} - Y_B)^2} - \sqrt{(X_{Drone} - X_A)^2 + (Y_{Drone} - Y_A)^2}\right)/c$$

$$TDoA_{C-A} = \left(\sqrt{(X_{Drone} - X_C)^2 + (Y_{Drone} - Y_C)^2} - \sqrt{(X_{Drone} - X_A)^2 + (Y_{Drone} - Y_A)^2}\right)/c$$

Devices

- Beacon A - Master (Nordic nRF52840, Qorvo DWM3000EVB Shield)
    - Application running: beacon_master_6()
- Beacons B,C,D,F,G - Slaves (Nordic nRF52840, Qorvo DWM3000EVB Shield)
    - Application running: beacon_responder_6(x,y) (x = 2, 3, 4 etc for B,C,D... y = antenna delay calibration in DWTU, not fine-tuned)
- Tag E - Drone Tag (Nordic nRF52840, Qorvo DWM3000EVB Shield, RPI Zero 2 W)
    - Application running: drone_tag_6() (nRF), UART_CalcPos_UDP_6.py (RPI)

Notes

- **Nordic boards:**
  - **Go to** https://www.qorvo.com/products/p/DWM3000EVB#documents, download DWS3000 API Software and API Guide
  - Unzip, follow instuctions in \DWS3000_Release_v1.1\Software\DW3000_API\Sources\DW3000_API_C0_rev4p0.zip\README_nRF52840-DK.txt
  - Replace nRF52840-DK folder in API structure with Avalon code
  - Modify PositionProj.emProject with correct path names
- **RPI notes**
  - **Add new wifi credentials:**
    - Create a file in the root of boot with name: `wpa_supplicant.conf` with content (modify data):

```
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="NETWORK-NAME"
    psk="NETWORK-PASSWORD"
}
```

  - **File transfer, ex FileZilla SFTP:** ip = ?, port = 22, l/p = pi/Avalon
  - **SSH/Terminal (ex PuTTY):** ip = ?, port = 22, l/p = pi/Avalon
    - **Modify autostart of script (comment out if f.ex. debugging):** `crontab -e`
    - **Start python script (if not started):** `python3 UART_CalcPos_UDP_6.py (from DLPS folder)`
    - **Modify script:** `nano UART_CalcPos_UDP_6.py (from DLPS folder)`
  - **Modify position calculation algorithm:** `nano main6.c (from DLPS folder)`
  - **Compile modified position calculation algorithm:** `cc -fPIC -shared -o calc_pos_6.so main6.c -lgsl -lgslcblas`
  - **Update ip address in** `UART_CalcPos_UDP_6.py!`
  - **Update ip address in receiving pcs python script! (PC)**