Development and Control of a swarm of nano quadcopters

Summer Internship Project Report by
Sagar Milind Supe

Supervisor: Dr. Jerome Le Ny

École Polytechnique de Montréal

Abstract

The position of a quadcopter should be known in order to control it. In this project, Ultra Wide Band system has been used to localize the quad-rotor using TOA and TDOA schemes. These schemes require synchronization of the clocks of anchors to establish a common reference of time and enabling to carry out the calculations of TOA and TDOA. The frequency of the crystal oscillators present in the anchors is not constant and thus becomes a difficult task to synchronize. Various methods to synchronize the clocks and the protocol to carry it out have been tested and the results are given in the report.

Index

1. Introduction

The Project aims at developing and controlling a swarm of nano quadcopters. In order to control each quad rotor in a particular pattern, its position should be known. The position can be estimated from the on board sensors like gyroscope and accelerometers but is not very accurate whereas the orientation can be measured accurately from these sensors.

The position of the quadrotor can be estimated from GPS or computer vision but are subject to conditions like GPS signal strength or light conditions respectively. The system used here to localize is ultra wide band.

There are different basis on which various methods can be implemented with Ultra Wide Band system like

1. Signal Strength
2. Angle of Arrival
3. Time of Arrival (TOA)
4. Time Difference of Arrival (TDOA)

TOA can be achieved using Two Way Ranging Algorithm which involves several message exchanges between the quad rotor and the anchor, but reduces the sampling rate of the position with the increase in number of the quadrotors as the base anchors have to communicate with each quadrotor one at a time.

This problem can be solved if the quadrotor only receives messages from these anchors' and is never required to send a message as a reply. In this was both TOA and TDOA can be implemented. But this requires all the clocks of the anchors to be synchronized. Various methods to achieve the same have been discussed in the report.

2. Experimental Setup

To perform localization, DW1000 module was used, which is a wireless Ultra Wide Band RF transceiver. DW1000 was connected to STM32F3 Discovery board to control it. DW1000 enables precise timestamping when the messages are sent and received.

2.1 DW1000 Clock Specifications

The frequency of the crystal oscillator present on DW1000 is 38.4 Mhz. The time is measured in a 5byte counter whose 1 tick is approximately equal to 15.65 pico seconds. Thus the counter or the clock resets approximately after every 17 seconds.

2.2 Timestamp sending problem

The timestamp recorded when the message is sent from the board, cannot be sent on the same message. Sending a delayed message with the timestamp is possible, but the accuracy of sending a delayed message is upto 4 nano-seconds. Thus the timestamp recorded at time t when message k is sent, can be sent on message k+1.

3. Clock Synchronization

3.1 Clock Characteristics

Two Boards with DW1000 were used to check the characteristic of the clocks. Board A was sending its timestamp after every 10ms and Board B was receiving the messages continuously and recording the timestamp when it received the message from Board A.

The difference between timestamps of Board A and Board B should be ideally constant if both the clocks had the same frequency. The graphs below show the characteristic of the clocks.

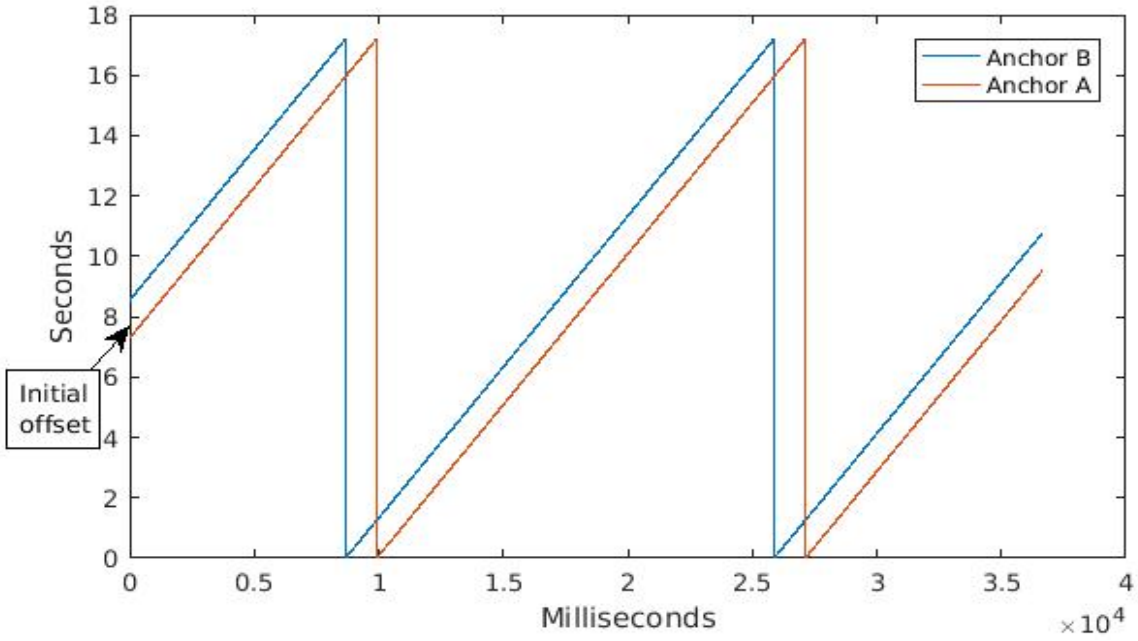


Fig.1 Clocks of Anchor A and Anchor B

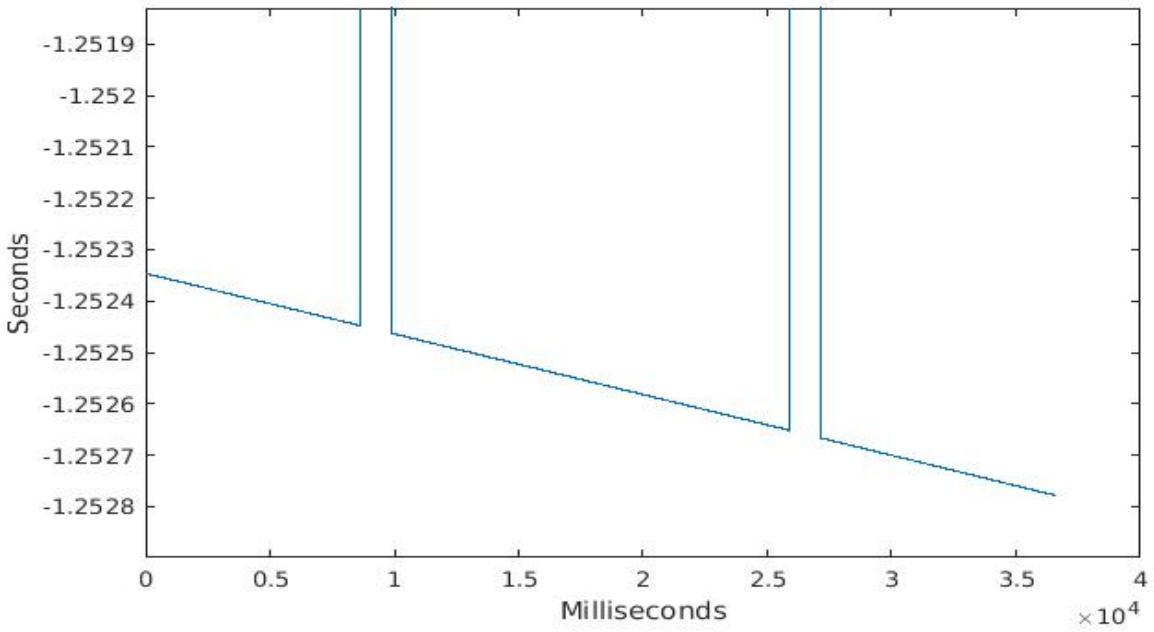


Fig.2 Difference Between the Two Clocks

The initial offset seen at the start in fig.1 is subtracted to calculate the difference. It is seen in fig.2 that the difference is not constant and it can be inferred that one clock is drifting away from the

other clock.

Thus there are two differences between the clocks

1) The initial offset

2) Drift

3) Sudden huge difference in between when 1 of the clock resets

The Drift occurs due to the difference in the frequencies of the clock.

3.2 Synchronization techniques

3.2.1 Data Logging

In order to Synchronize two clocks, Anchor A sends a Synchronization Message which has its timestamp on it. Anchor B uses this timestamps to Synchronize itself with Anchor A.

Data of Anchors was logged after every 10ms for 36670ms.

Message from Anchor A was taken after every 50ms as the Synchronization timestamp for Anchor B. Though the results shown below remained same even when data was logged up-to every 1ms and the Synchronization gap was kept up-to 200ms.

3.2.2 Synchronization Algorithm

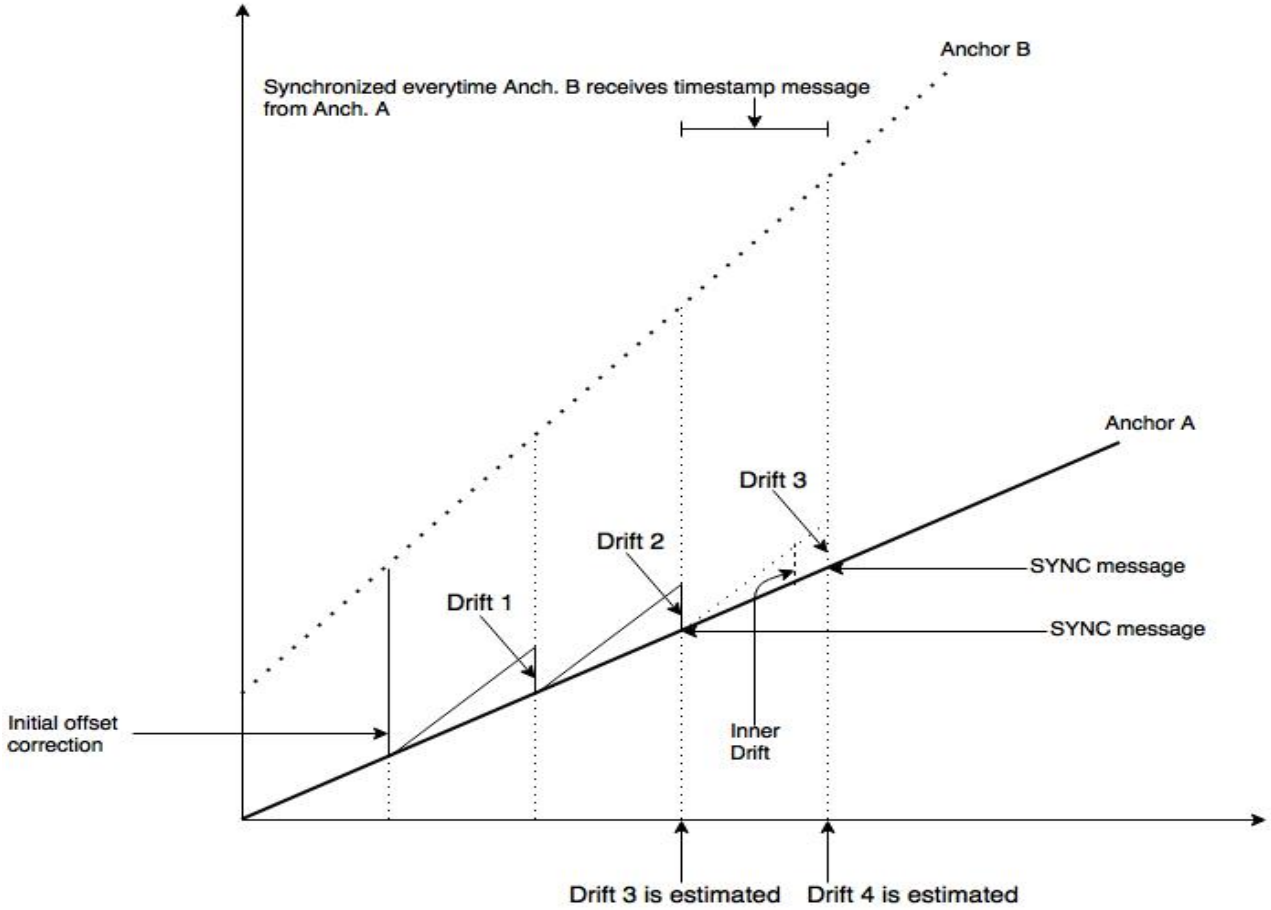


Fig.3 Synchronization Algorithm

Dotted line represents the actual clock of Anchor B and the dark solid line represents the clock of Anchor A (reference clock).

1) The initial offset is corrected at the start.

2) This offset is continuously subtracted from the subsequent readings of Anchor B till the next Sync message and Drift 1 is recorded.

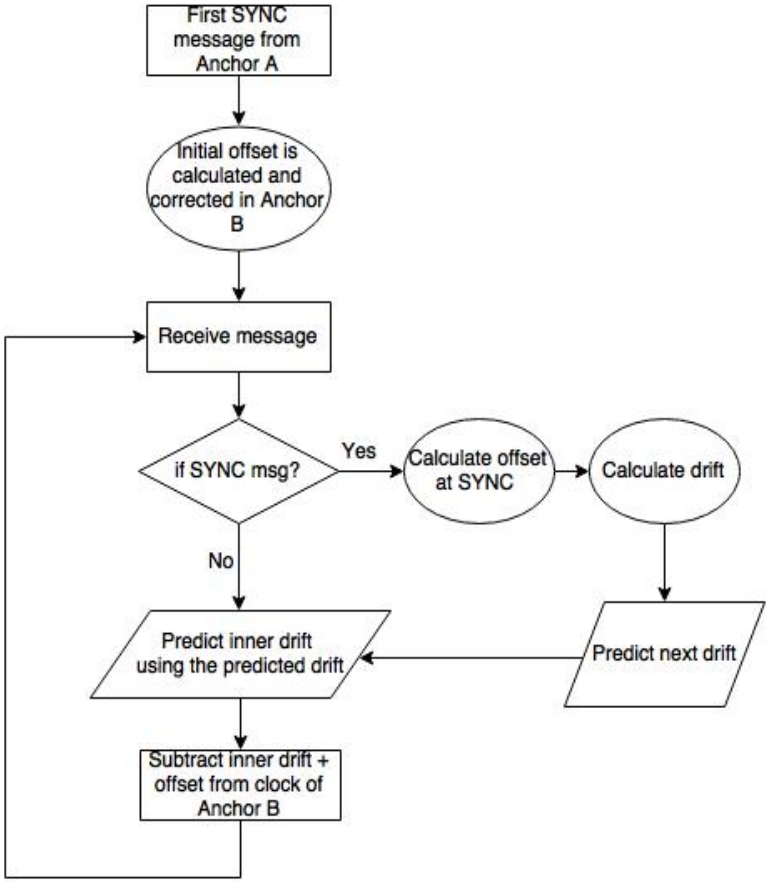
3) Drift 2 and the subsequent drifts are measured in a similar way by subtracting the new offset measured at sync message.

These drifts are used to estimate the future drifts and correct the clock according to it using two different methods

1) ARMAX model - Estimation starts after 6 drifts are calculated.
2) PI estimation - Estimation starts after 2 drifts are calculated as shown in fig.3

Once the future drifts are estimated the inner drifts shown in the fig.3 can be calculated simply from the predicted drift and the properties of similar triangle.

This innerdrift is subtracted from the timestamp of anchor B along with the offset.



The error after
synchronization as shown in fig.7, fig.9 and fig.14 has huge spikes in between. These occur when the clock resets and needs to be rejected.

Another method of Synchronization has been explained in 3.2.5 which does not use this scheme. It is based on Alpha – Beta filter.

3.2.3 ARMAX

The difference between Anchor B timestamps and Synchronization timestamp as explained in 3.2.1 gave the following graph.

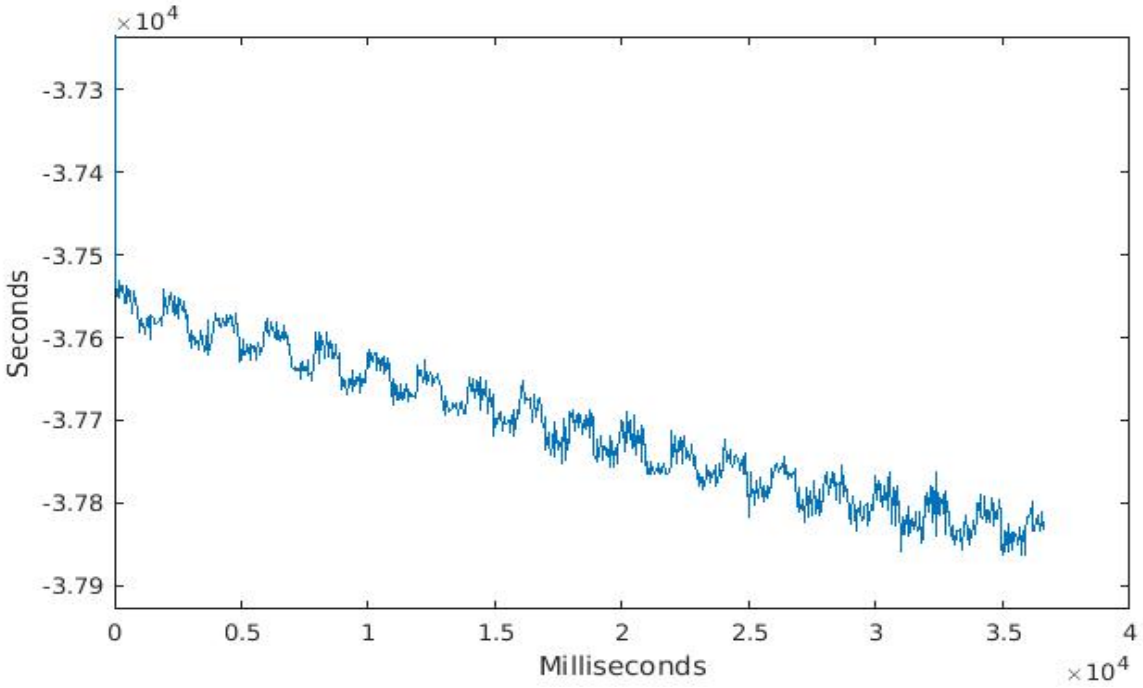


Fig.4 Difference between Anchor B and Anchor A timestamp at Synchronization.

Referring to fig.4 the difference between the timestamps can be seen to be periodic and a time series. Thus it was considered to use ARMAX to predict the future drift.

ARMAX model is a tool to understand a time series and thus predict the future value. The dataset of the drifts was available and thus the MATLAB function armax was used to estimate parameters of ARMAX model.

An order 5 Armax model i.e. previous 5 readings of drift were used to determine the next drift. No drift is estimated till 6 drifts have been measured. It has a model structure as follows:

$$y(t+1) = -(a_1 y(t) + \ldots + a_1 y(t-n+1)) + (b_1 u(t-1) + \ldots + b_1 u(t-n+1))$$

where, y(t+1) is the predicted output to be at t+1

y(t)...y(t-n+1) - are the previous output on which the current output is dependent

u(t)...u(t-n+1) - are the previous input on which the current output is dependent

In this case the predicted output is the future drift and inputs are the previous drifts

A dataset of 362 readings was used to find the model parameters. Fig.5 shows how well does the model respond to the same 362 readings.

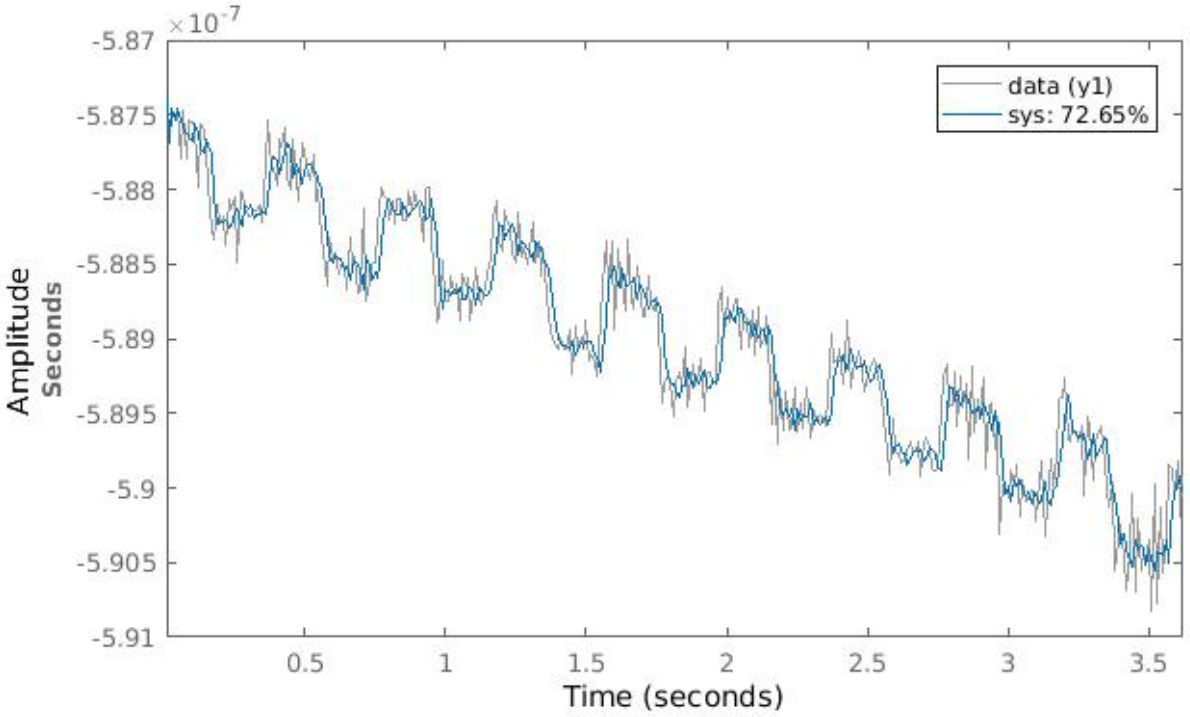


Fig.5 Trained Set to predict next drift

These parameters were used on the entire dataset to predict the drifts and synchronize Anchor B with Anchor A.

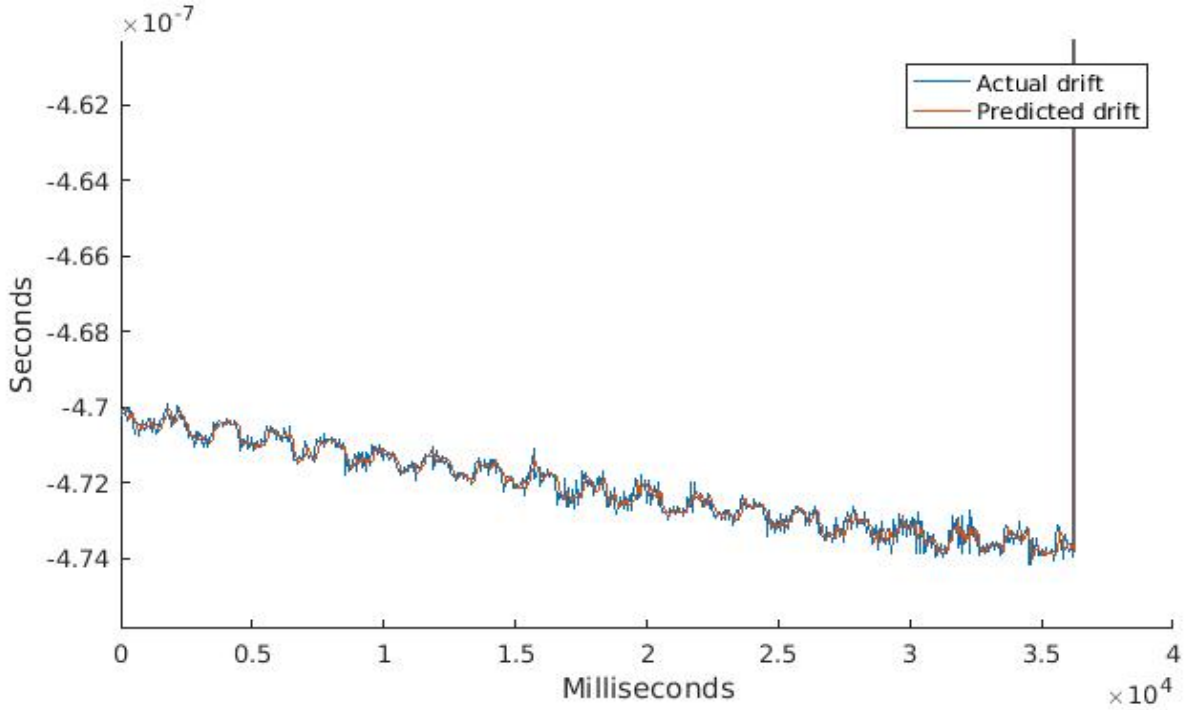


Fig. 6 Actual and the predicted drift for the entire set

Fig.6 shows the actual drift and the predicted drift. It is seen that the prediction has worked very well

Fig.7 shows the synchronization error throughout the time-span and is below 1 nano-second.

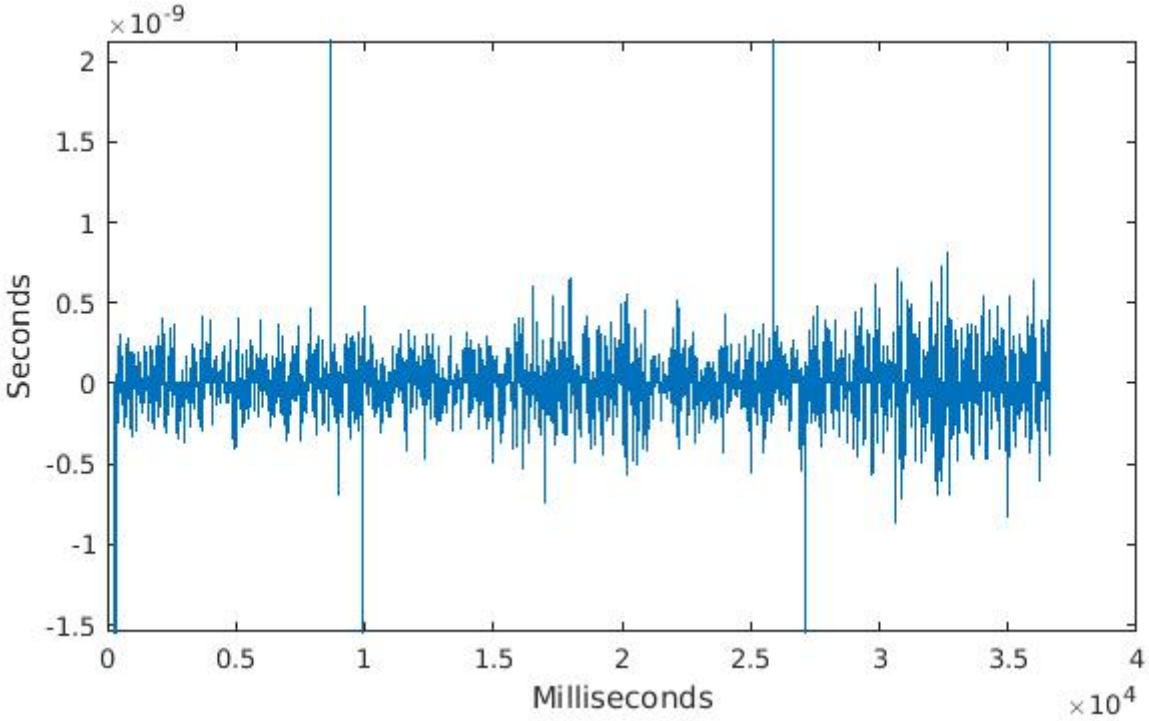


Fig. 7 Synchronization Error of ARMAX

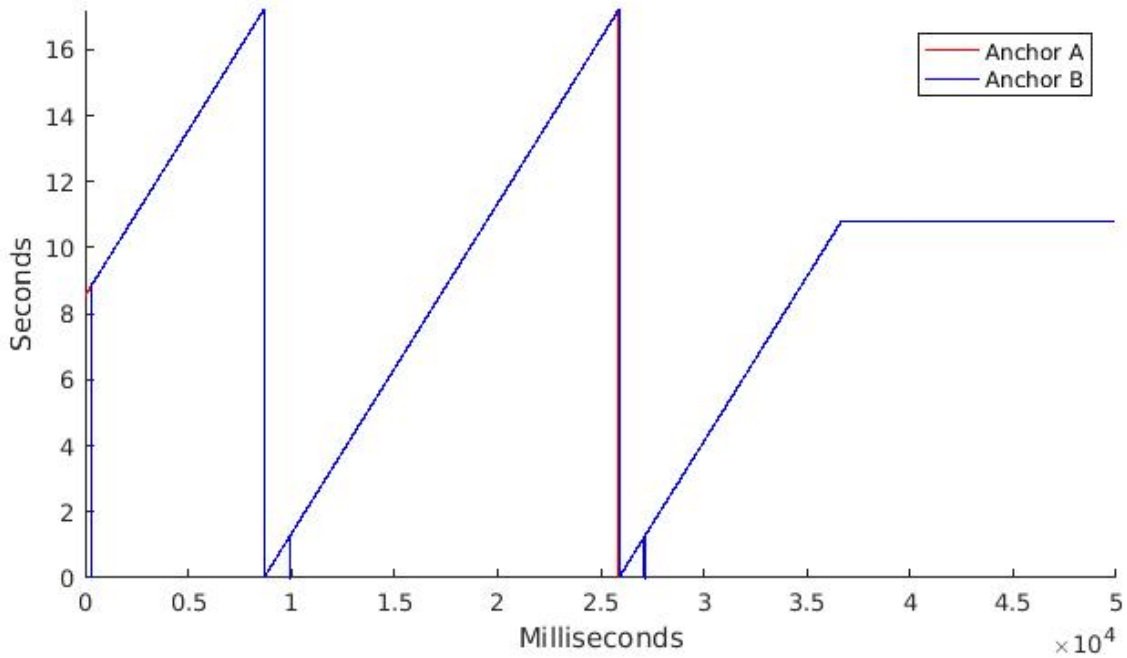


Fig. 8 Synchronization of Anchor A and Anchor B using ARMAX

3.2.4 PI Estimation

Proportional Integral estimation was used to estimate the future drift. The equations used for estimation are:

$error_n = drift_n - drift_{n-1}$

$errorsum_n = errorsum_{n-1} + error_n$

$estimated\_drift_{n+1} = Kp*error_n + Ki*errorsum_n + drift_n$

Drift at n is measured as follows:

$drift_n^B = T_n^B - Offset_{n-1} - T_n^A$

n is incremented on the arrival of a synchronization message.

T_n^B = Time stamp of Anchor B at n

Fig. 9 shows the synchronization error using PI estimation and is below 1 nanosecond throughout time-span.

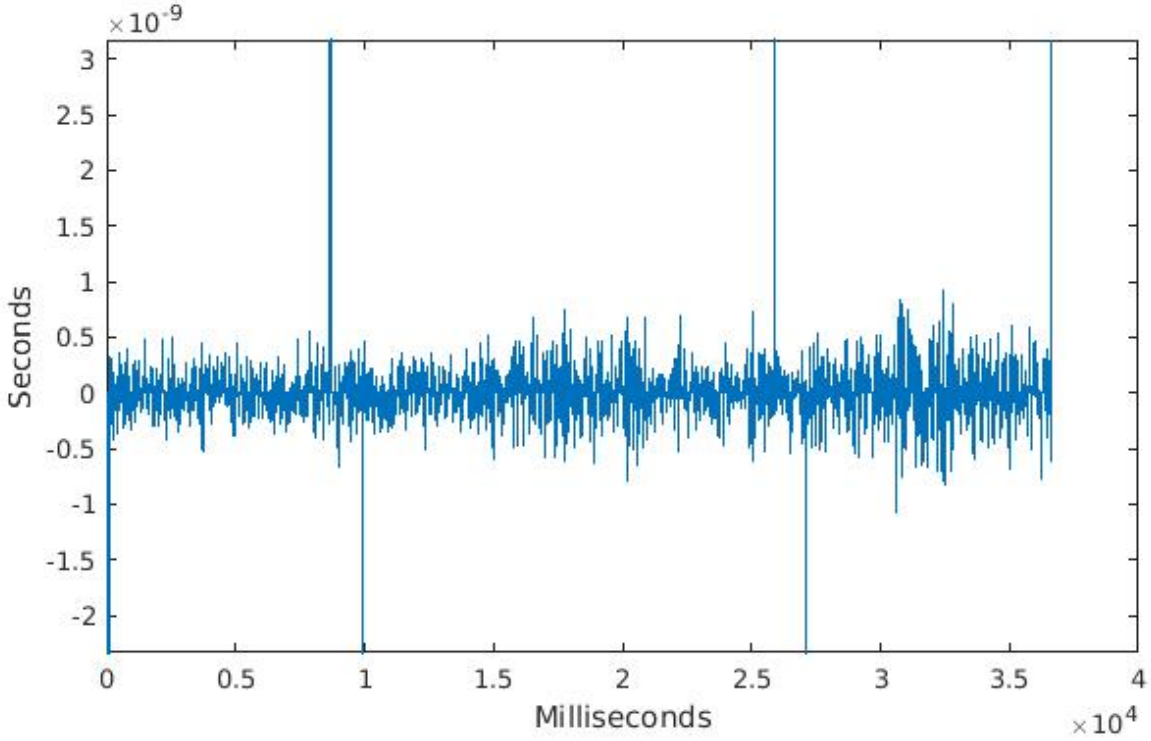


Fig. 9 Synchronization error of PI estimation

Had there been no subtraction and calculation of inner drift from the predicted drift in both ARMAX and PI estimation, the error would have been in the form of triangles as expected and shown in fig.10 .

It is in the range of 1e-6.

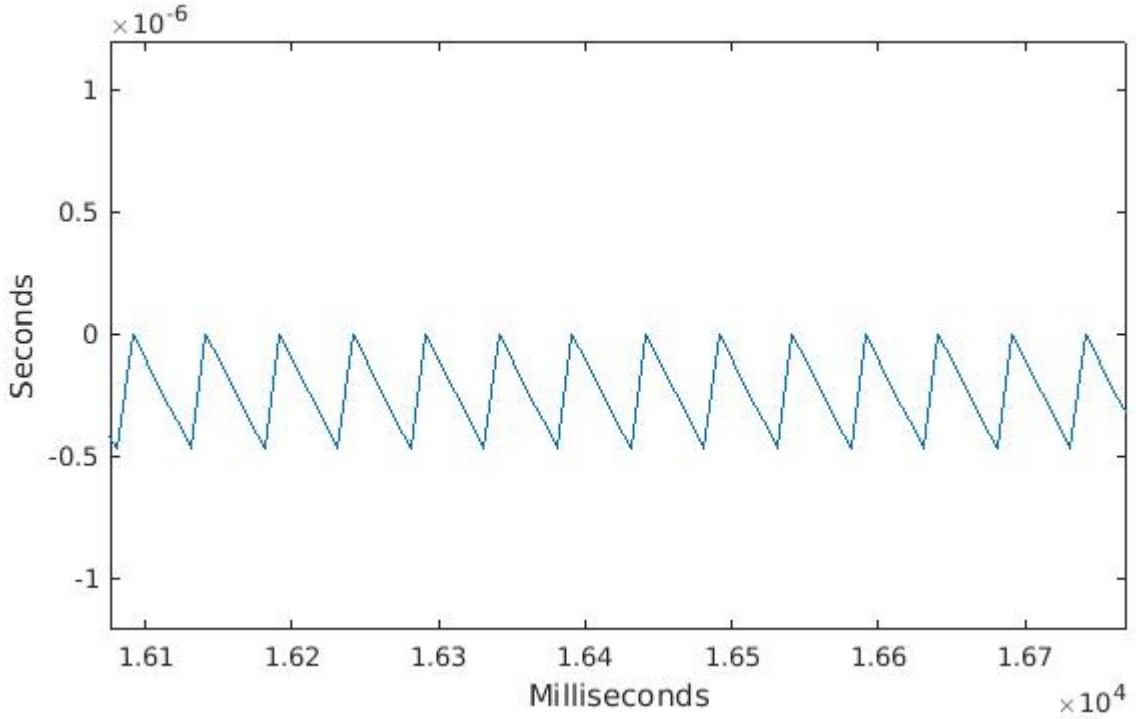


Fig. 10 Error in the case where inner drift is not subtracted

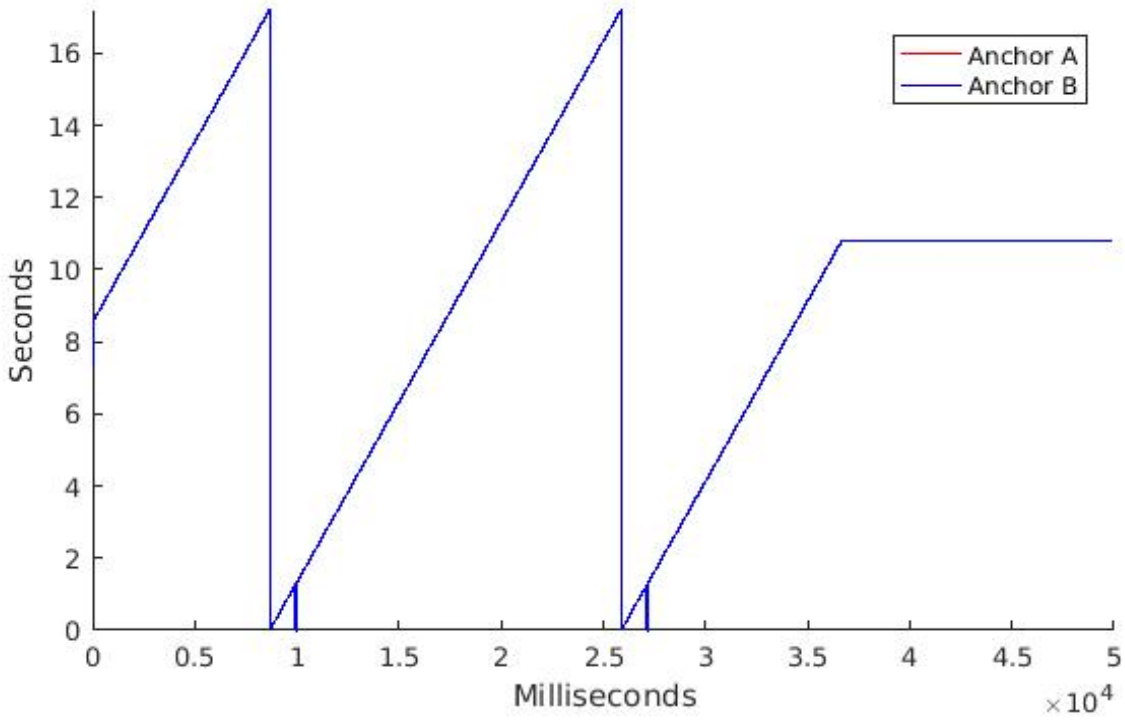


Fig.11 Synchronization of Anchor A and Anchor B using PI Estimation

3.2.5 Alpha Beta filter

This method is not dependent on the algorithm mentioned in 3.2.2 . Alpha Beta filter is a simplified form of observer for estimation. The model consists of two states, the time of arrival and the drift. Thus the time of arrival and the drift at an instant k can be given by the following equation, considering that drift remains constant over the interval Δt. These are known as time update equations.

$$T\hat{O}A_B{}^A{}_k = T\hat{O}A_B{}^A{}_{k-1} + \Delta t * \hat{d}rift_{k-1}$$

$$\hat{d}rift_k = \hat{d}rift_{k-1}$$

$TOA_B{}^A{}_k$ = Time of Arrival at B according to clock A at k

$TOA_B{}^B{}_k$ = Time of Arrival at B according to clock B at k

On the basis of mechanical system analogy the time of arrival is the position and the drift is the velocity.

The timestamp of Anchor A is sent after every 50ms as mentioned earlier as a Synchronization message. The measurement update should ideally happen in every loop but as we do not get the measurement update continuously, it takes place whenever a synchronization message is received by Anchor B.

The measurement update is given by the following equations:

1) $\hat{r}_k = TOA_B{}^A{}_k - T\hat{O}A_B{}^B{}_k$: The prediction error is the difference between the timestamp A and timestamp B at the time of Synchronization.

2) $T\hat{O}A_B{}^B{}_k = T\hat{O}A_B{}^B{}_k + \alpha * \hat{r}_k$

$\hat{d}rift_k = \hat{d}rift_k + (\beta/\Delta t)\hat{r}_k$

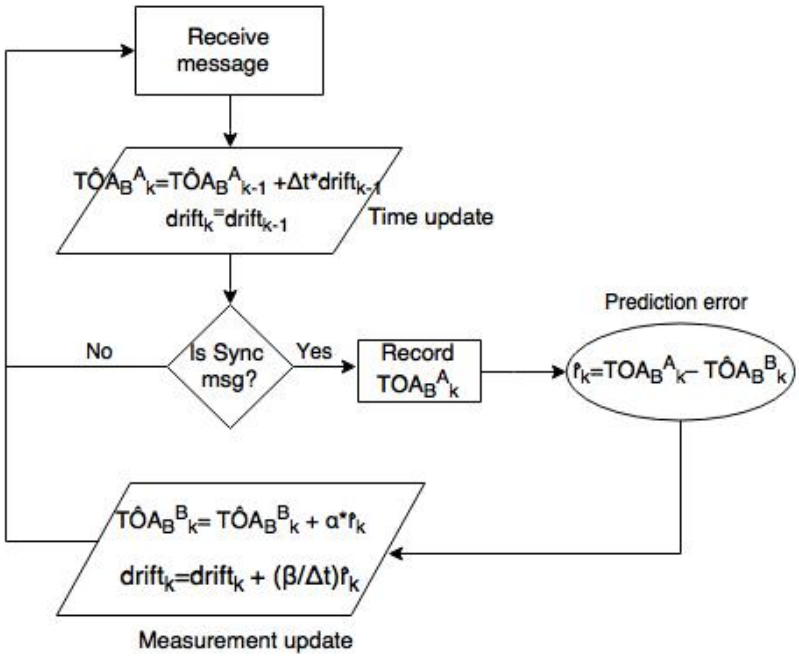
Alpha and beta are calculated from process variance and noise variance.

$\lambda = \sigma_w * t^2$

$r = (4 + \lambda - \sqrt{(8\lambda+\lambda^2)})/4$

$\alpha = 1 - r^2$

$\beta = 2(2-\alpha) - 4\sqrt{(1-\alpha)}$



Applying the following steps gave the following results:

Fig.12 shows the sync between Anchor A and Anchor B.

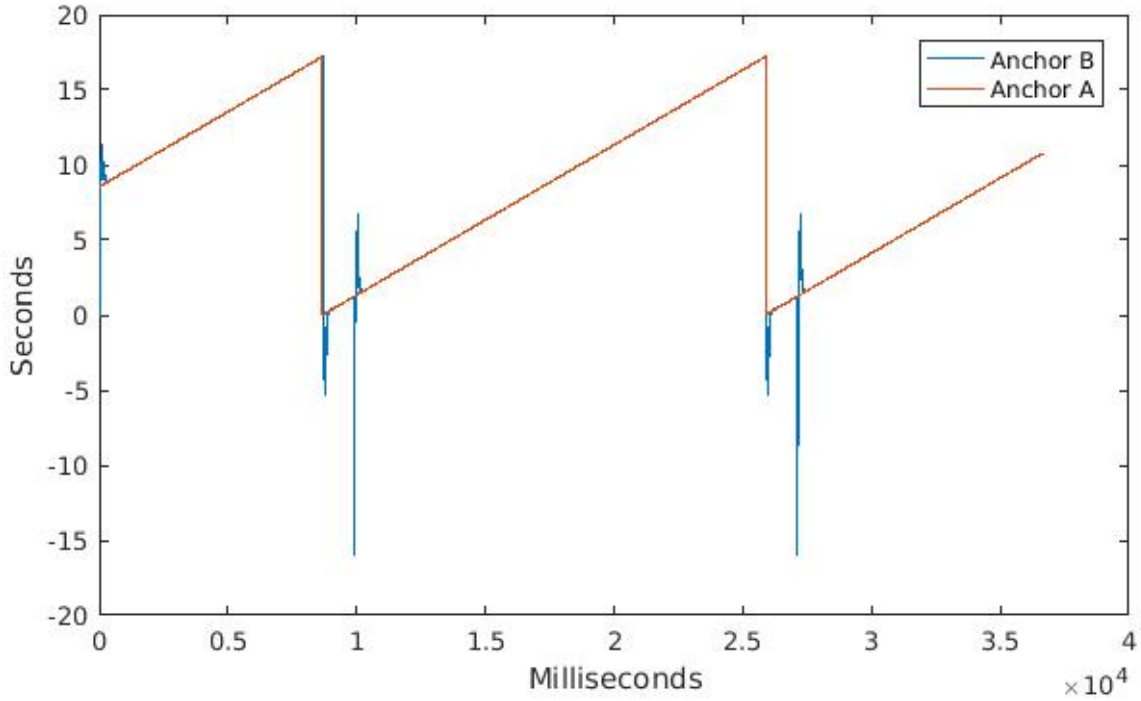


Fig.12 Synchronization of Anchor A and Anchor B using Alpha Beta

Fig.13 shows the error between Anchor A and Anchor B.

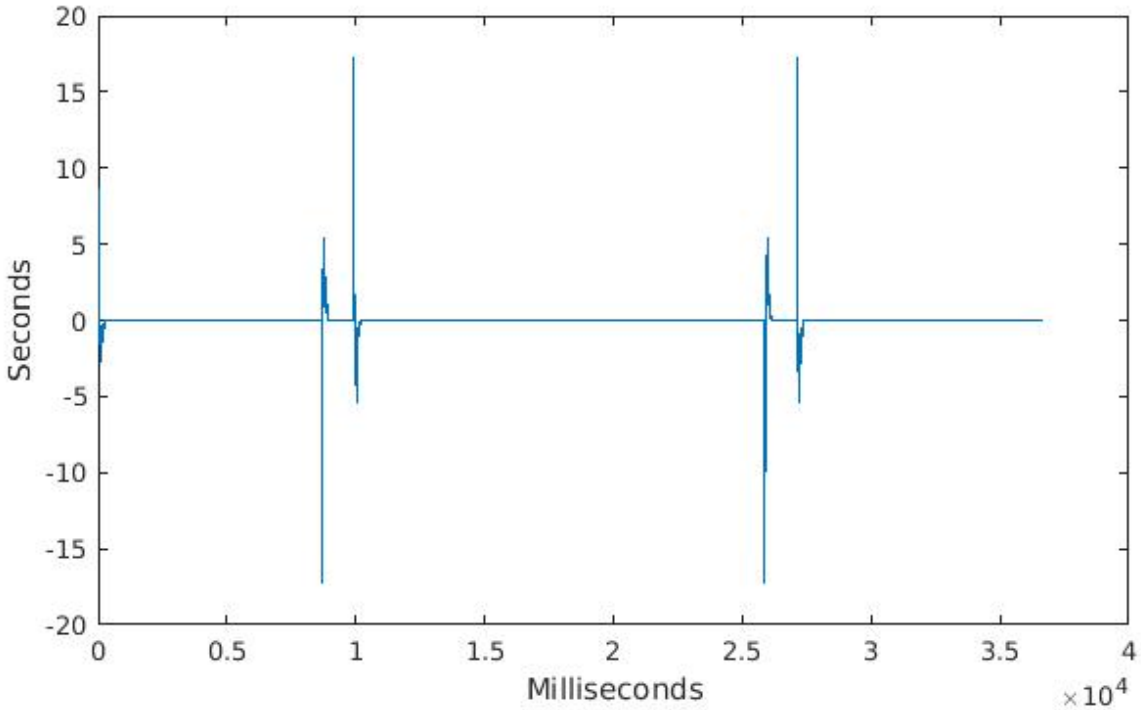


Fig.13 Alpha Beta filter error

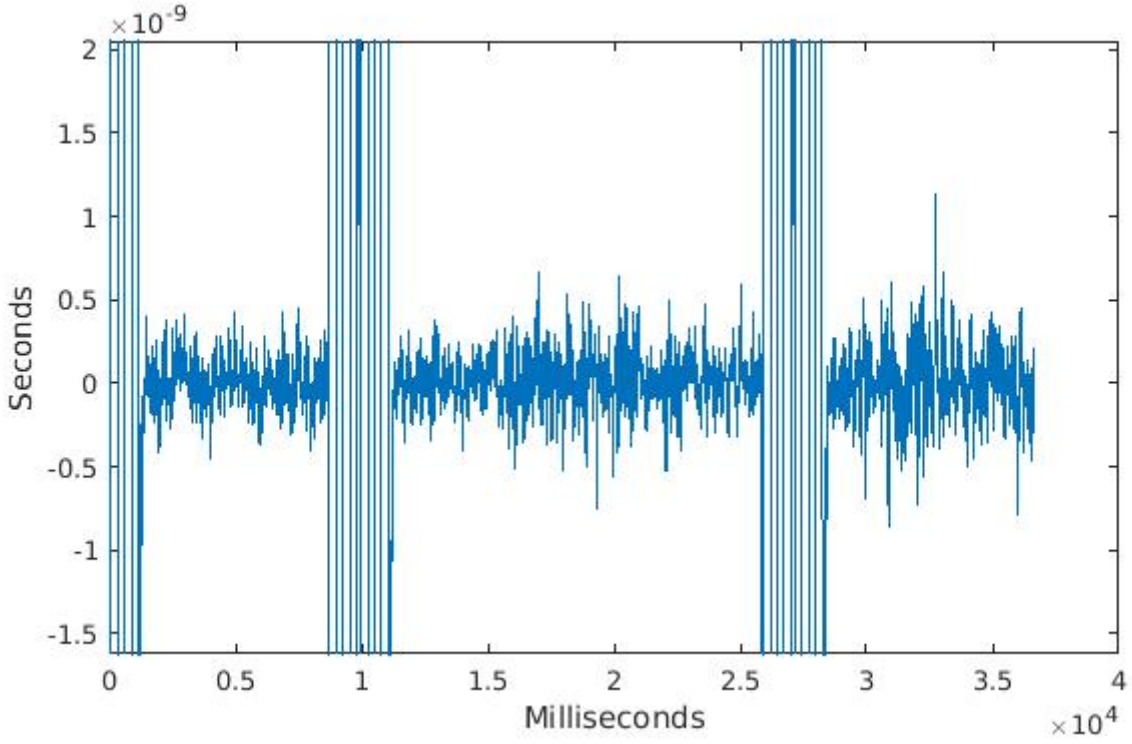


Fig.14 Synchronization error of Alpha Beta filter

4. Synchronization protocol

In-order to practically implement the above methods to synchronize the clocks a protocol was designed.

As explained earlier that the transmission timestamp recorded cannot be sent on the same message, and this has been considered in the protocol. Thus a message at time k carries the timestamp of k-1.

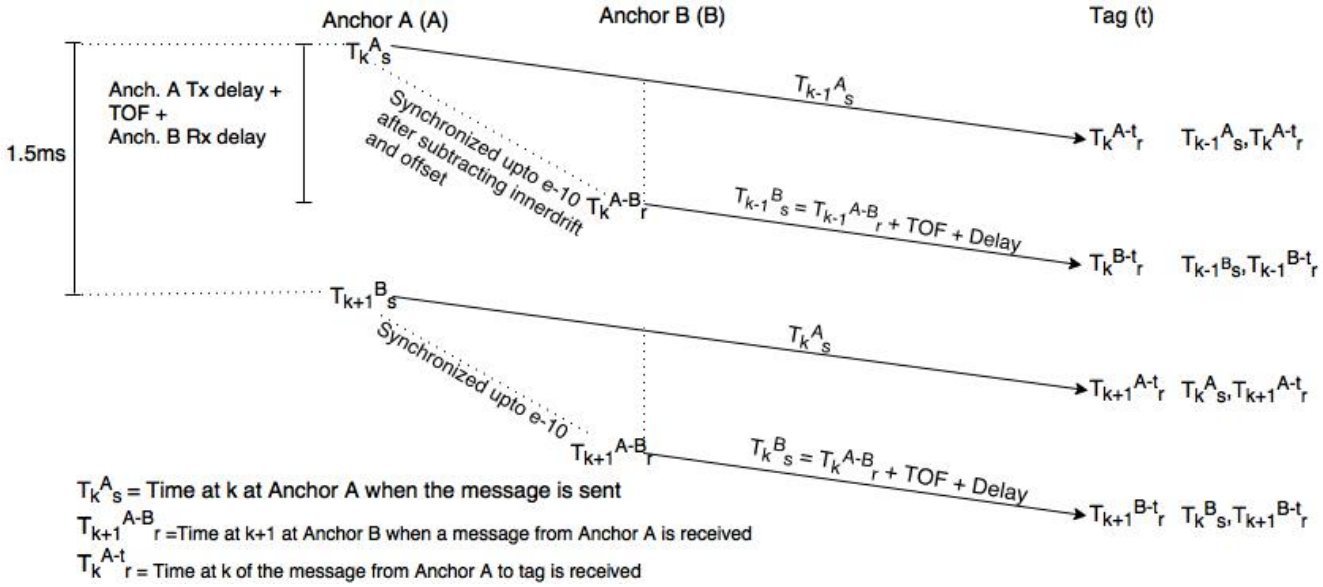


Fig.15 Protocol

-First subscript, k is incremented on every new message.
-Superscript A,B,t are Anchor A, Anchor B and tag respectively.
-Second subscript s,r signify whether the timestamp is stored when the message is sent or received respectively.

As shown in the fig.15, Anchor B sends its timestamp after it receives the message from Anchor A, to avoid collision of messages as they are on the same channel. Moreover, Clock A already moves ahead till the Synchronization message is received by the other Anchors. Thus the time of flight(TOF) from Anchor A to the other anchors needs to be measured and added to the synchronized time. An additional delay is also to be added to the synchronized time which is explained ahead.

The synchronization takes place in the same way as explained in 3.2.2 .

More anchors can be added in the above scheme. For instance, Anchor C would start sending when it receives the message from Anchor B and would be synchronized at the same time when Anchor B is synchronized.

5. Time of flight and Delay Calculation

5.1 Delays

There is a transmission or reception delay whenever an Anchor sends or receives a message.

- Tx delay : A short delay between the timestamp recording and the time when the message leaves the antenna.
- Rx delay : A short delay between, when the antenna receives the message and when the timestamp is recorded.
  According to the datasheet of DW1000, Tx delay and Rx delay are the same value.

These delays need to be measured accurately and added or can be considered as measurement noise.

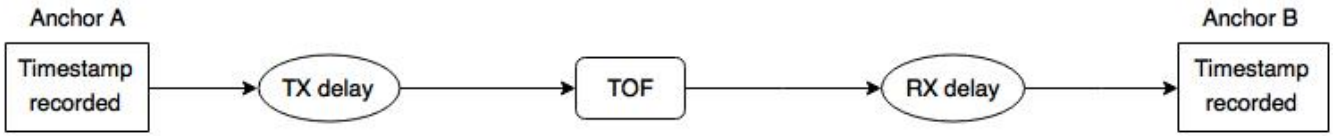


Fig.16 Tx and Rx Delay

5.2 Delay Calculation

Messages are sent and received in the following manner between Anchor B and Anchor A to calculate the delay.

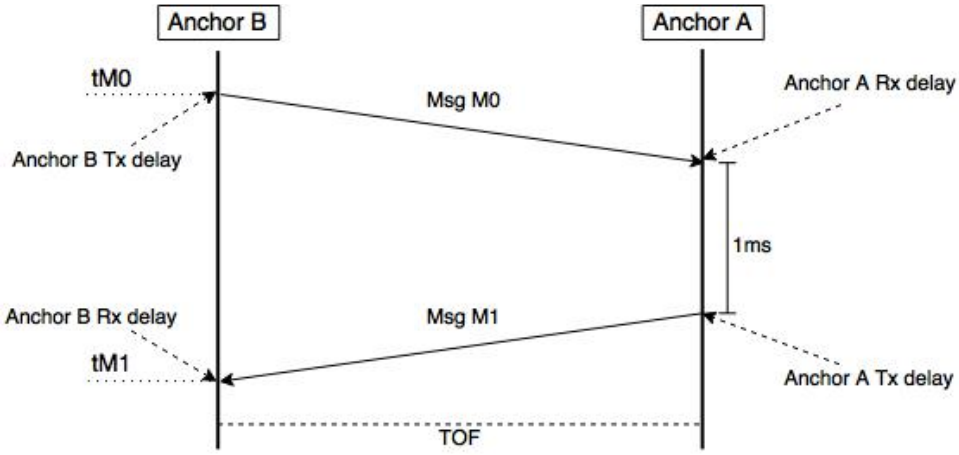


Fig.17 Delay calculation protocol

According to fig.17

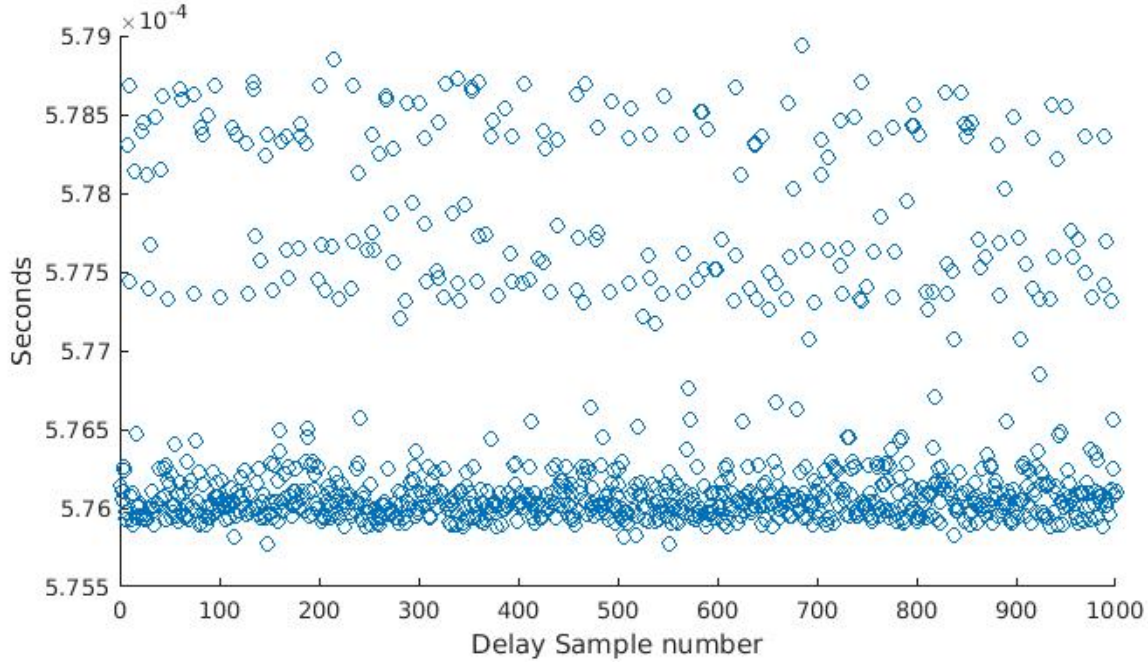
tM1 = tM0 + Anch. B Tx delay + TOF + Anch. A Rx delay + 1ms + Anch. A Tx delay + TOF + Anch. B Rx Delay

Here, 2*delay = Anch. B Tx delay + Anch. B Rx delay + Anch. A Tx delay + Anch. A Rx delay

Therefore, 2*delay= tM1 – tM0 – 1ms – 2*TOF

and TOF = (Distance between Anch. A and Anch. B) / (Speed of light)

The above method was tested with two anchors and the graph shown in fig.18 shows the value of 2*delay for 1000 samples.



The delay does not seem to be constant and is very high, and thus a better method is required to calibrate the anchors.

5.3 Correction to Synchronized timestamp

Thus the synchronized timestamp obtained from 3.2.2, needs to be added with the TOF and the delay calculated in 5.2 .

Corrected Synchronized timestamp = Synchronized timestamp + TOF + delay

6. Conclusion

Thus, the various methods to synchronize the clocks which were tested are explained and the results for the same have been shown. All the methods gave proper synchronization i.e below 1 nano-second. But the error when the clock resets needs to be corrected. The delay calculation explained in the end was not accurate as shown in the graph and needs to be measured in an efficient way. The protocol explained in 4 has been tested and works without any collision of messages of different anchors.

7. References

[1] Ciarán McElroy, Dries Neirynck, Michael McLaughlin, “Comparison of wireless clock synchronization algorithms for indoor location systems”

[2] Yik-Chung Wu, Qasim Chaudhari, Erchin Serpedi, “Clock Synchronization of wireless sensor networks”

[3] Jiming Chen, Qing Yu, Yan Zhang, Hsiao-Hwa Chen, Youxian Sun, “Feedback-Based Clock Synchronization in Wireless Sensor Networks: A Control Theoretic Approach”

[4] Mark W. Mueller, Michael Hamer, and Raffaello D’Andrea, “Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation”

[5] Anton Ledergerber, Michael Hamer and Raffaello D’Andrea, “A Robot Self-Localization System using One-Way Ultra-Wideband Communication”

[6] Mihail L. Sichitiu and Chanchai Veerarittiphan, “Simple, Accurate Time Synchronization for Wireless Sensor Networks”

[7] Ali Burak Kulakli, Kayhan Erciyes, “Time Synchronization Algorithms based on Timing-Sync Protocol in Wireless Sensor Networks”

[8] Decawave Ltd, “DW1000 Device Driver Application Programming interface (API) guide”

[9] Decawave Ltd ,“Sources of error in DW1000 based two-way ranging (TWR) schemes”