Indoor Navigation using the HTC Vive Base-Stations

Morten Kals, SINTEF

September 8, 2017

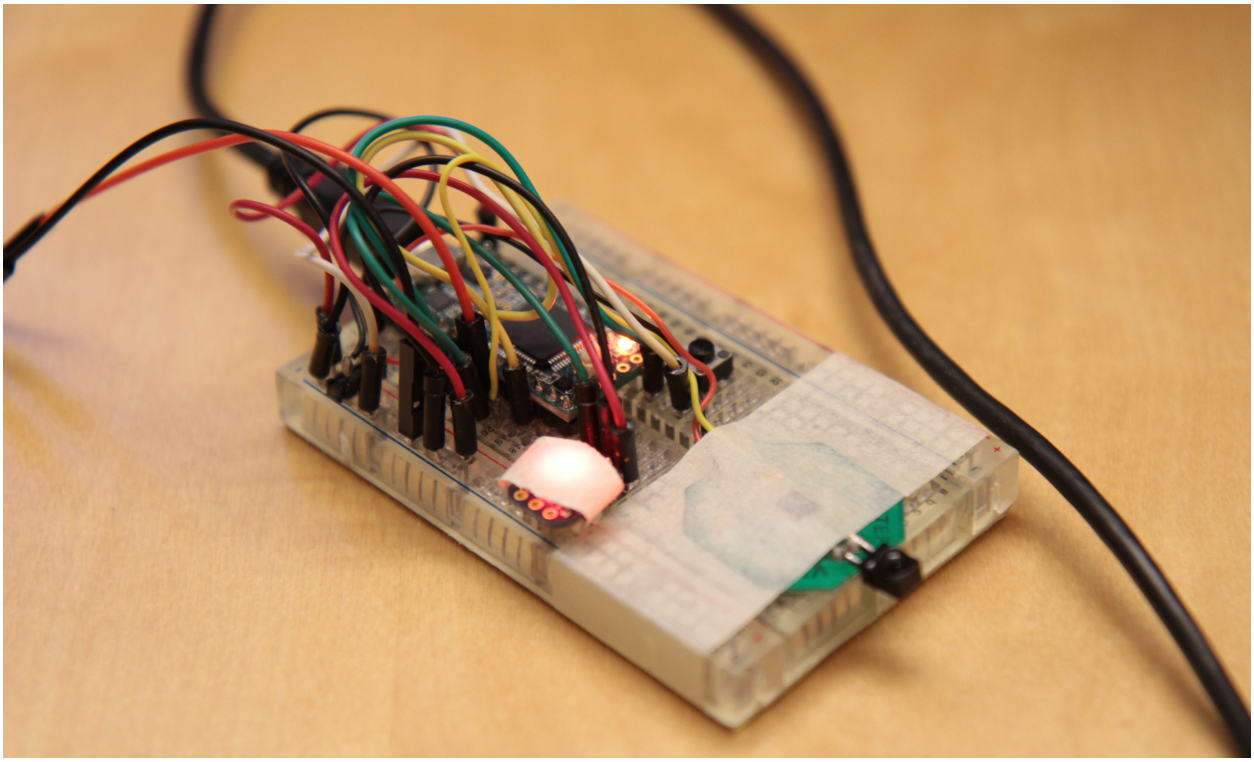


Figure 1: Teensy with Neopixel, Push-Button and IR Sensor Circuit Peripherals

Contents

1 Base-Station Signal Processing

The HTC Vive base-stations operate in pairs to send information that can be detected by an infrared (IR) sensor and used to the determine the position of the sensor. The functionality provided is thus similar to that of GPS, although the information transferred from the base-stations is relative angle measurements rather than distances. No processing occurs on the base-stations for computing the position of the sensor, resulting in a system that can be used for multiple sensors simultaneously with positive security-implications.

Each base-station unit consists of a line laser that rotates at 120 rpm in both the horizontal and vertical plane. By determining the time taken from the first calibration-pulse arrives to when the laser-pulse arrives, the relative angle between the sensor and the base-station can be determined. The calibration-pulses are also used by the base-station to synchronise the rotation of their line lasers. The pulse-sequences are summarised in Table 1.

Table 1: Pulse Characteristics

Pulse start (μs)	Pulse length (μs)	Source station	Meaning
0	65–135	A	Sync pulse (LED array, omnidirectional)
400	65-135	B	Sync pulse (LED array, omnidirectional)
1222-6777	$\sim$10	A or B	Laser plane sweep pulse (center=4000μs)
8333			End of cycle

One base-station is set to be the master of the system (A) and the other is set to be the slave (B). The calibration pulse of the master arrives before the calibration pulse of the slave, and it is as such easy to tell them apart. The duration of the calibration-pulse of the master and of the slave describes which angle-reading the following line-laser-pulse will correspond to in accordance with Table 2. Axis 0 is horizontal and axis 1 vertical. The data-bit is discarded.

Table 2: Ideal Package-Characteristics

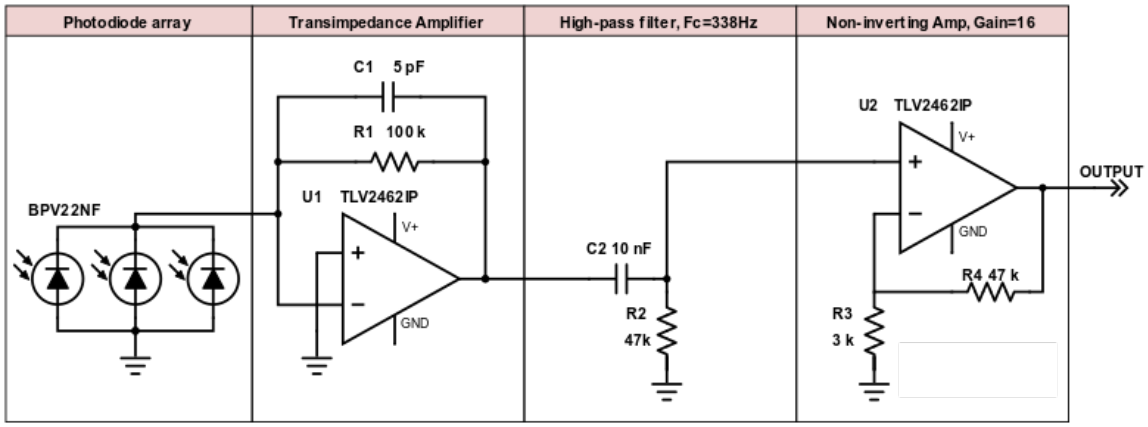
Name	skip	data	axis	length (ticks)	length (μs)
j_0	0	0	0	3000	62.5
k_0	0	0	1	3500	72.9
j_1	0	1	0	4000	83.3
k_1	0	1	1	4500	93.8
j_2	1	0	0	5000	104
k_2	1	0	1	5500	115
j_3	1	1	0	6000	125
k_3	1	1	1	6500	135

1.1 Hardware Implementation

Figure 2 shows the circuit we used to detect the IR-signal and process it for the micro-controller.

With the base-stations active and mounted below the roof, the sensor was positioned on the floor and various readings performed using an oscilloscope and Teensy serial-prints. The below graphs

Figure 2: Sensor Circuit

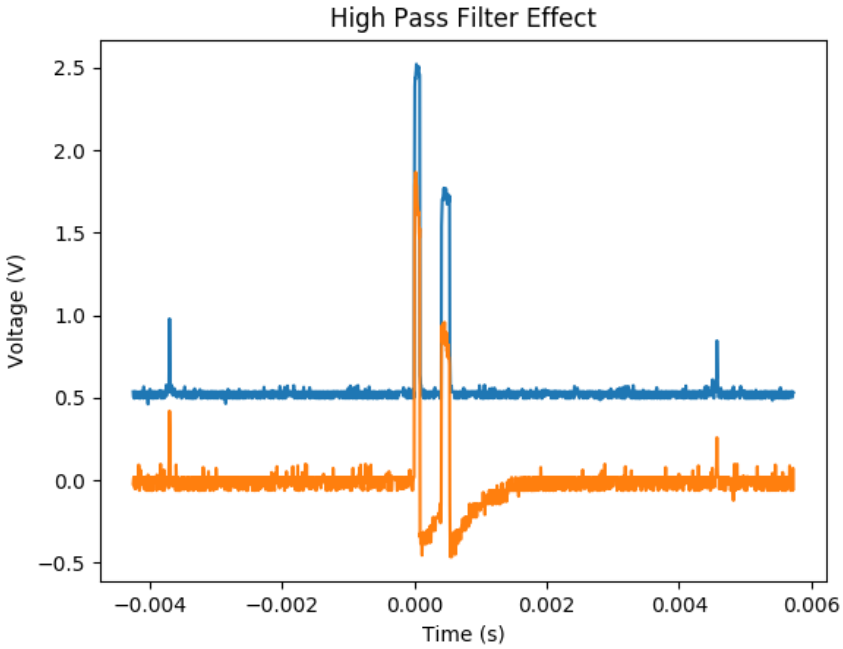


Source: `https://github.com/ashtuchkin/vive-diy-position-sensor`

seeks to outline some of the characteristics of the signal and to test the function of the different stages of electrical processing performed on the signal.

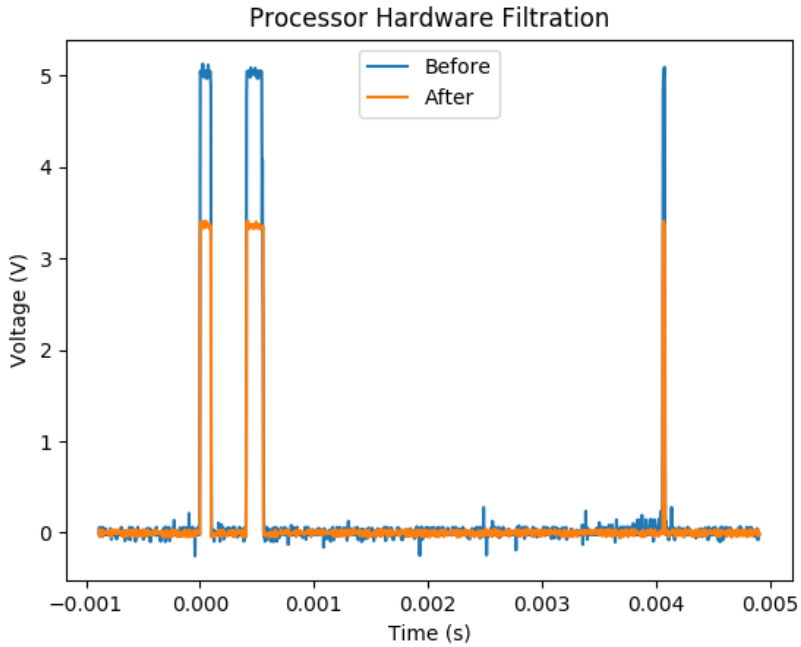
1.1.1 High Pass Filter

After the initial transimpedance amplifier comes the high pass filter. The high pass filter is seen to remove the DC component (orange line) form the signal coming out of the first Op-Amp (blue line) during normal operation. In the figure below, the sensor is located about 1.8 meters from the base-stations. We see the calibration pulses do not saturate the first Op-Amp as the signal coming into the capacitor has an amplitude less than VCC (5V). We also observe a large deviation in amplitude between the the calibration signals (located at 0.0000 and 0.0005 s) and the signals from the line-lasers (located at -0.0038 and 0.0048 s).



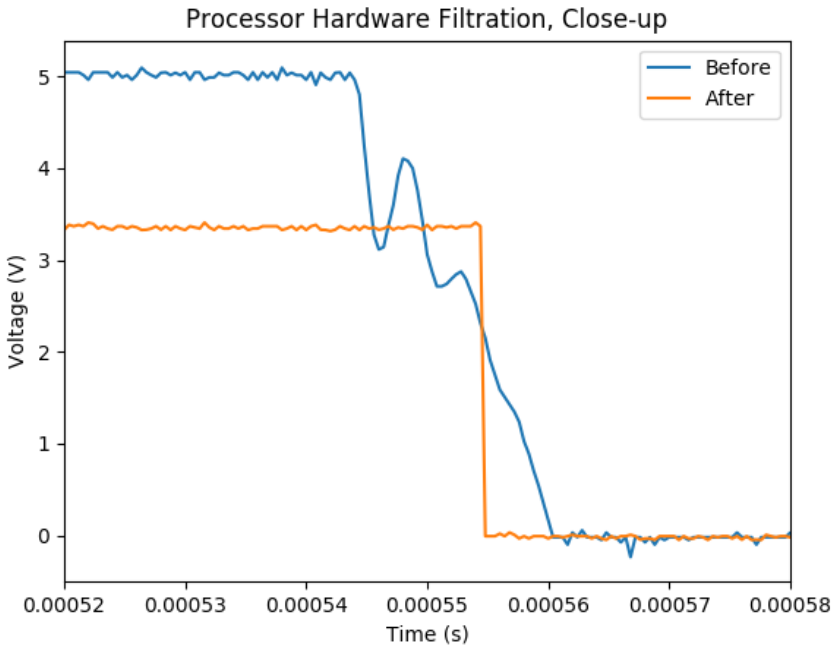
1.1.2 Processor Hardware Filtration

Using the Input Capture Filter Control mechanism on the MK20DX256VLH7 processor, the input signal from the IR sensor circuit is smoothed through a low pass filter and its voltage level decreased to 3.3 V as we can see in the figure below.



By zooming in on the end of the second peak, we see how this final layer of hardware filtering squares

the trailing edge of the signal without generating small pulses for the input oscillations.

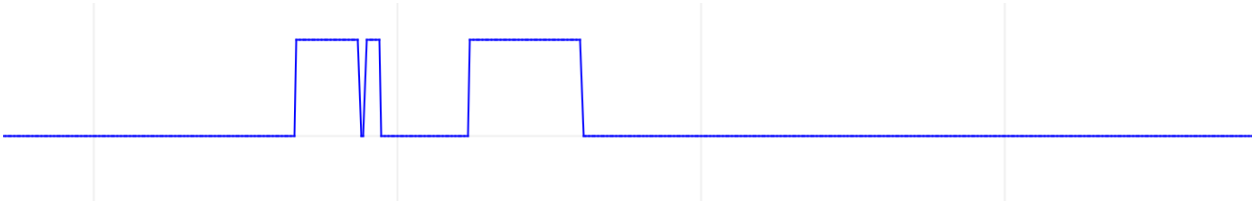


1.2 Software Implementation

The implemented algorithm uses the Teensy 3.1 processor running at 72 MHz. Flex-timers are used to determine pulse lengths and a falling edge interrupt routine collects this information and passes it to the main thread of the algorithm. The digital signal processing is broken into three steps: pulse detection, package detection and signal extraction.

1.2.1 Pulse Detection

Information from the separation of pulses is used to determine whether a pulse is correctly detected or whether the rising edge is caused by noise. Tests have revealed that there is a danger of a rebound pulse arising if the Op-Amps become to heavily saturated. This is handled by discarding pulses that arrive too close to the end of another pulse as seen at the end of the first pulse below.



1.2.2 Package Detection

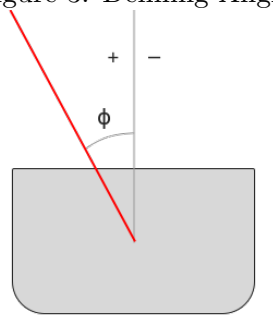
The information in Table 1 is used to identify the first and second calibration peaks as well as the following line-laser pulse and any associated noise. The calibration pulse durations and pulse arrival times are passed on to signal extraction.

1.2.3 Signal Extraction

We know the calibration pulses are being sent in a given order:

1. horizontal pulse from master
2. vertical pulse from master
3. horizontal pulse form slave
4. vertical pulse from slave

Figure 3: Defining Angle



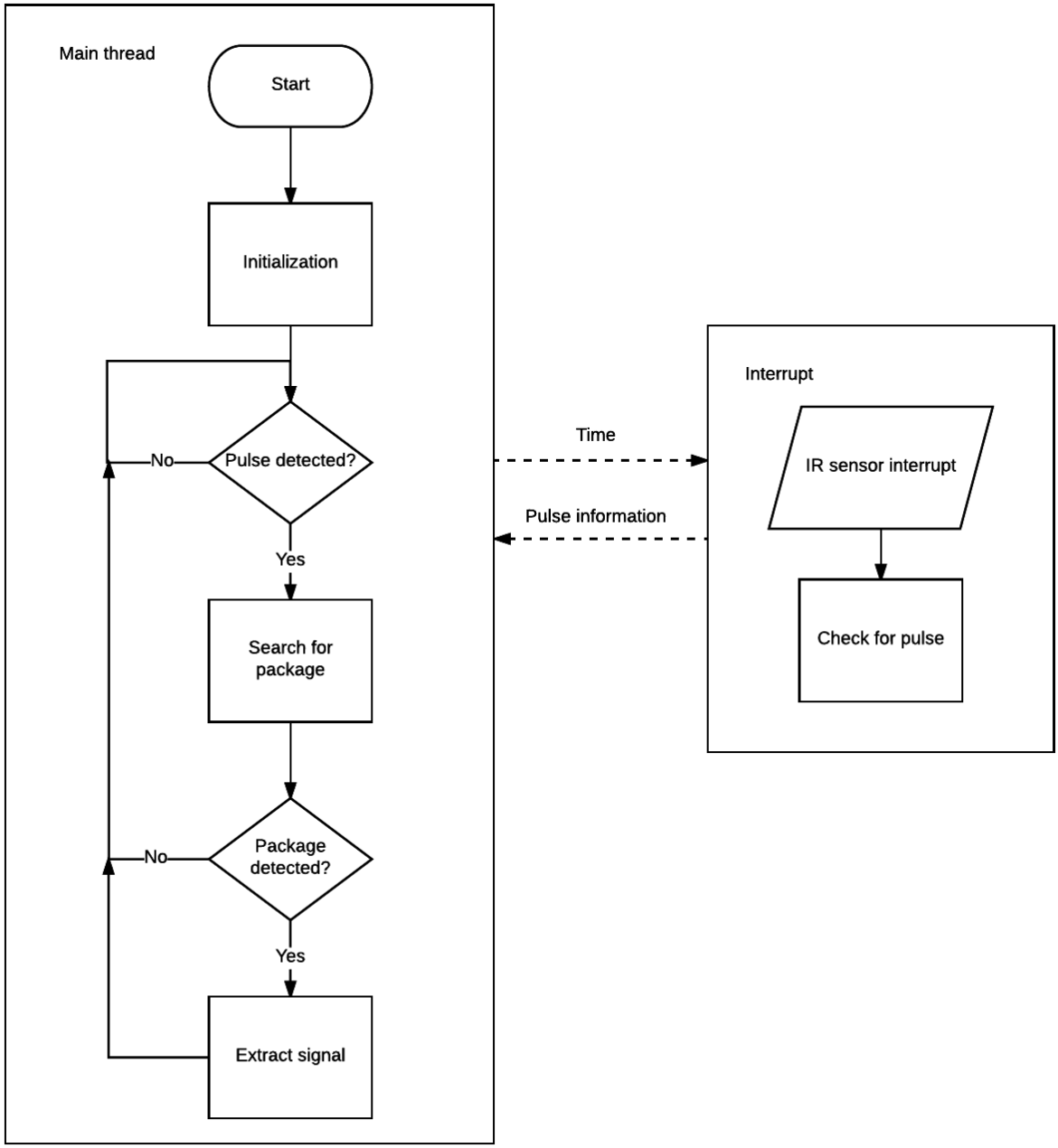
The fact that packages arrive sequentially enables the system to infer which signal type is going to arrive based on time after an initial calibration period where this time-cycle is determined. The main advantage with this is to enable the system to correctly identify the signal type despite some mechanical interference. In practice this makes a large difference to the accuracy of operation of the system. It is implemented so as to preserve the following properties:

- Thousands of consecutive packages can be lost due to mechanical interference without impeding the system's ability to regain the signal.
- The start-time of each pulse is used to ensure no drifts in timing between the base-stations and our processor.

After proper calibration, there is no chance for interpreting the signal type incorrectly. The functionality is summarised in Figure 4.

The signal output from the system is in radians and measured from the centre line of the front-face of each base-station unit as shown in Figure 3.

Figure 4: Angle-Reading Flow Chart



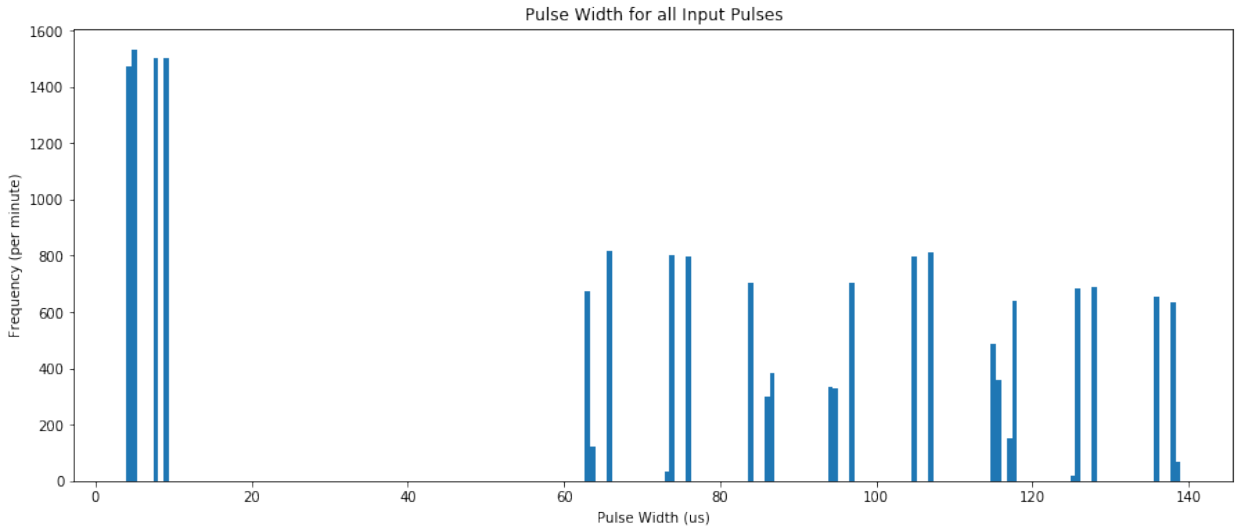
2 System Validation

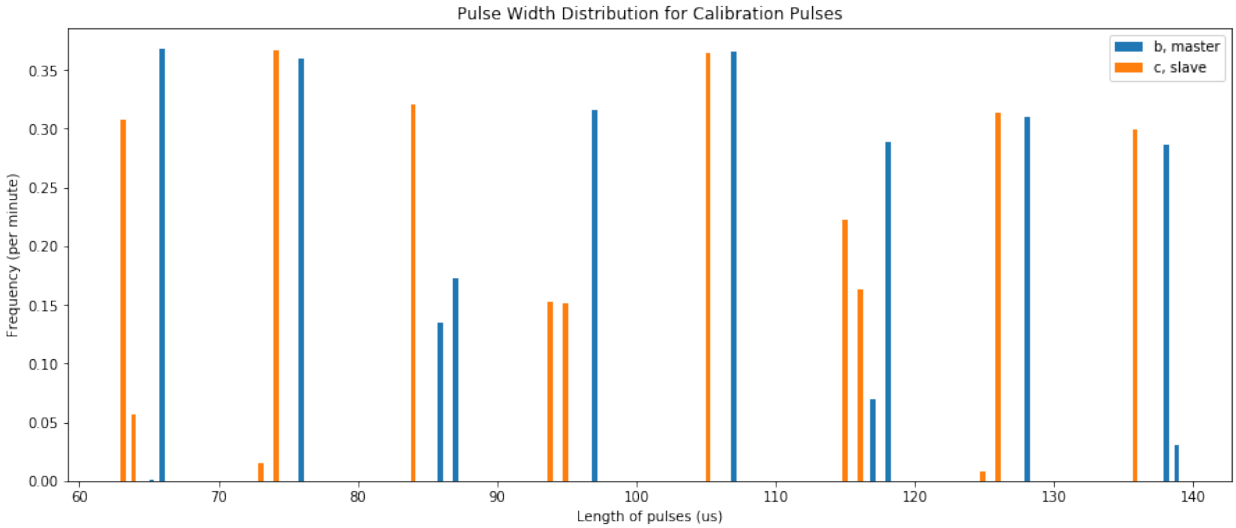
A Jupyter Notebook was written to validate the components of the Vive-based positioning system and estimate the overall accuracy obtained and limits of operation. The script is designed to take a file containing the serial-output from the verbose debug-mode of the Teensy program. To enter debug mode you plug the Teensy-board to power with a jumper between ports 11 and 12. When launched, double-press the button to enter verbose mode. The file Parsing reads the output format, while Visualising contains a series of helper-methods to make it easy to plot the data.

2.1 Validating Pulse Classification

For this initial test, we use the roof-mounted base-stations and position the sensor between them. The digital signal read by the sensor is monitored by falling/rising edge interrupts that record the timing-information for each input pulse. A histogram of the pulse-widths is seen below. The next graph uses the package-recognition functionality in the Teensy code to determine which pulses originate from the master and slave base-stations and plots the pulse-widths accordingly.

We see that pulses of varying length are detected in what corresponds to the range of the calibration-pulses as well as the signal pulses. In the first plot, there is not eight distinct pulse-widths to be observed, but a series of double spikes. When we in the second graph separate the signals from the master and the slave base-station, we see that this double-spike behaviour is caused by there being a constant shift between the pulse-widths from the two. This is something we account for in our classifier by determining the pulse-class from the longest/shortest calibration pulses received from the n most recent pulses.

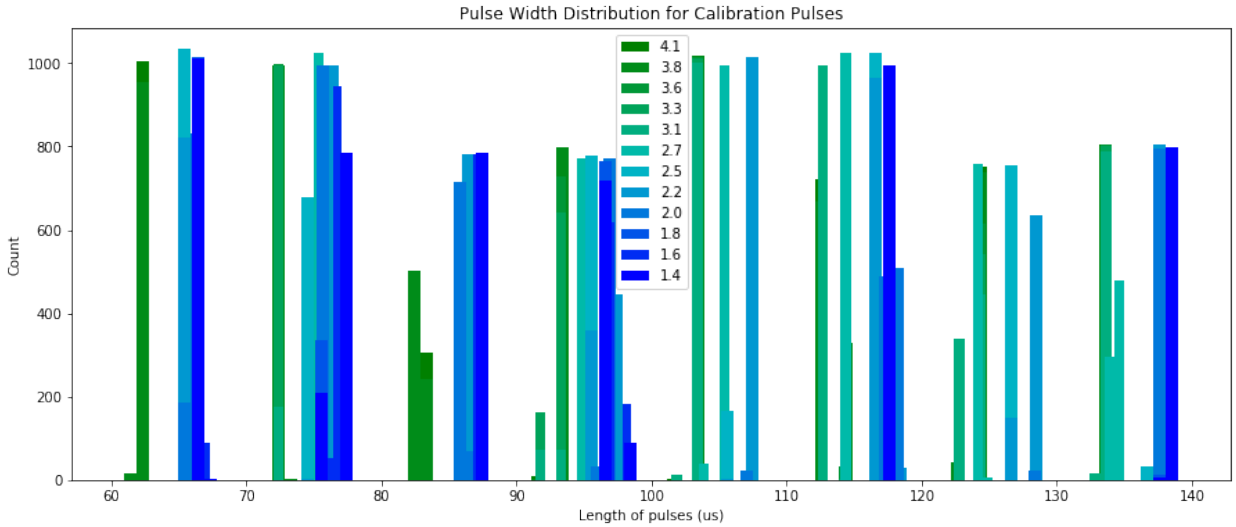




Note: The discrete values for pulse-length observed here is caused by an integer division to result in truncation and loss of accuracy. This has been updated to a floating point deviation operation, resulting in a much better resolution to the pulse-widths.

2.2 Normal Distance Variation

The following data was collected with the base-stations mounted in the roof at 6 meters distance and in opposite corners of the room. The sensor was mounted on a camera tripod 1.3 meters below the roof. The tripod was then positioned under one base-station and moved across the room in 30 cm steps. 1 minute of data was recorded for each position. For simplicity, we only consider the data from the master base-station (the results for the slave is very similar).



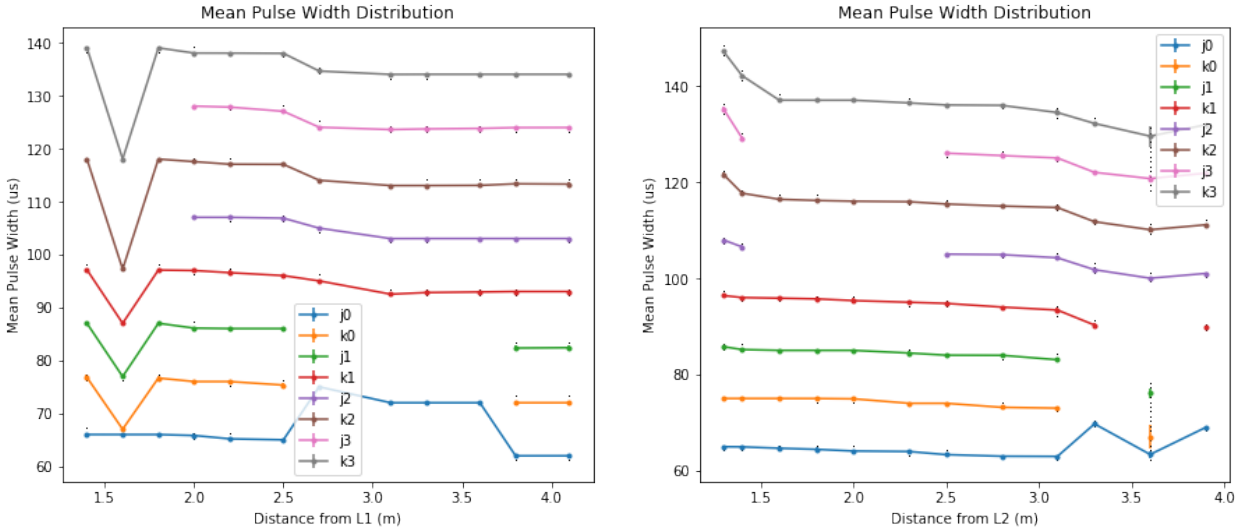
We see the pulse-widths fall into 8 distinct classes (despite variations between data-sets).

There should be 30 pulses/second $*$ 60 seconds $=$ 1800 pulses arriving per angle. Due to the data-bit, this will be split between two angle readings, giving on average 900 pulses that we expect in

each class. We know the data-pulse is the second most significant bit (see Table 2) of the signal-distribution, so the pattern where we see j_0 receive about 1000 pulses in the given period while j_1 receives about 800 pulses etc. makes sense - they sum to 1800, giving the expected number of pulses coming in for each angle.

Interestingly, we see the data-points collected in the middle of the two base-stations (eg. 2.5 meters) have a significant number of dropped packages. This was later discovered to be caused by the comparator being set too high so that large incident-angles on the sensor did not trigger the registration of a digital pulse. When lowering the comparator value the system is seen to work with significantly larger angles. For more information, see Section 2.8.

In order to more clearly asses the correspondence between pulse-width/sensor distance and classification-algorithm the following plot was made. The black dots correspond to the data-points collected, and the coloured lines to the mean pulse width of each pulse-class.



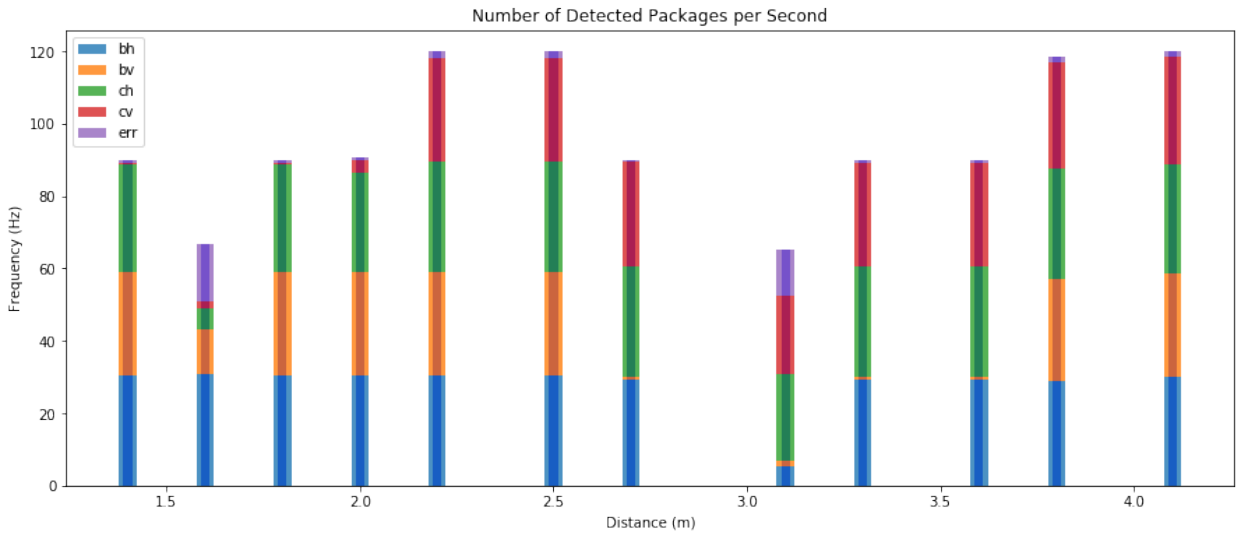
We see a clear classification for the bands where we have signals in all eight classes. Where the longest or shortest pulses are missing, this system breaks down which is to be expected given our implementation. This breakdown is acceptable, as we assume that the package-arrival-time will be calibrated in a region where it is able to receive readings from all angles.

If we look at the numerical error for this data-set, we confirm that there is reception-issues for certain angles. A large number of the positions lack about 1800 data-points, suggesting one of the four angles have not been measured at all. From this we again conclude that the issue is systematic and related to input-angle on the sensor and/or other factors.

When we use the measured-angle breakdown for this data-set we see that while bv (angle in the vertical plane from the master base-station) is the angle that is missing the most, this is not always the case.

Table 3: Error-Log for Normal Distance Variation

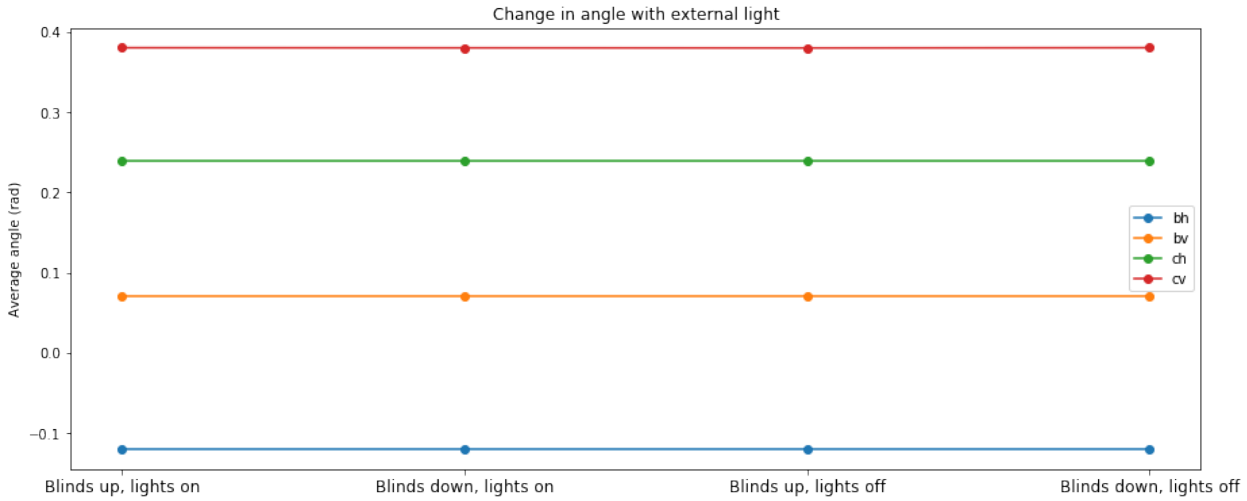
distance to L1 (m)	# collected points	# lost points	% error rate	# signal gaps
4.1	7199	1	0	2
3.8	7111	89	1	88
3.6	5396	1803	25	1802
3.3	5398	1799	24	1800
3.1	3920	3276	45	1799
2.7	5396	1803	25	1800
2.5	7196	4	0	4
2.2	7197	3	0	3
2.0	5445	1755	24	1755
1.8	5399	1800	25	1800
1.6	4013	3185	44	1799
1.4	5396	1803	25	1800



From this we conclude that the signal-classification seems to work well when we are measuring all angles. There is however a significant problem present with regards to signal reception (resolved, see Section 2.8).

2.3 Light Interference

To evaluate the performance of the system with interference, data-points are collected with/without light on and with/without blinds down. During the capture the sky was blue, but no direct sunlight came through the window.



The angles measured in the different situations are identical and the differences in pulse-widths seems to be very small - we can confidently conclude normal light sources will not disturb the system.

We do however observe that when direct sunlight falls on the sensor, the OpAmps get fully saturated so that the input comes to rest at a permanent high and no information is as a consequence communicated.

2.4 Minimum Operation Distance

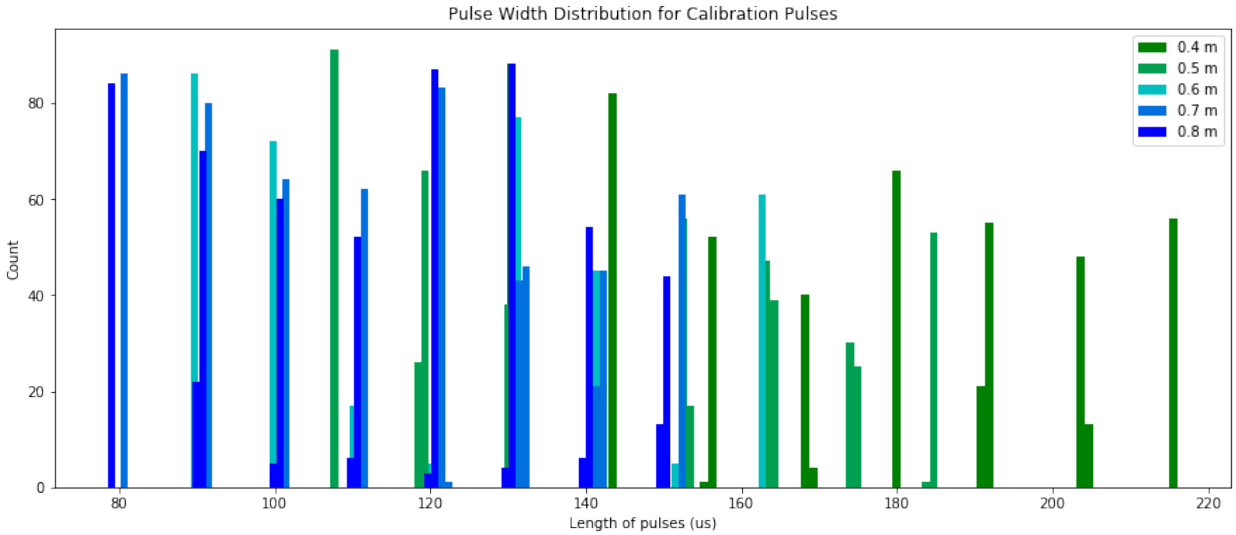
In order to determine the minimum operation distance of the system, two base-stations were positioned next to each other with a sync-cable (requiring one base-station to be in mode b while the other is switched to A). The sensor was mounted on a tripod located directly in front of the master and moved backward in 10 cm increments, starting at a 40 cm distance from the base-stations.

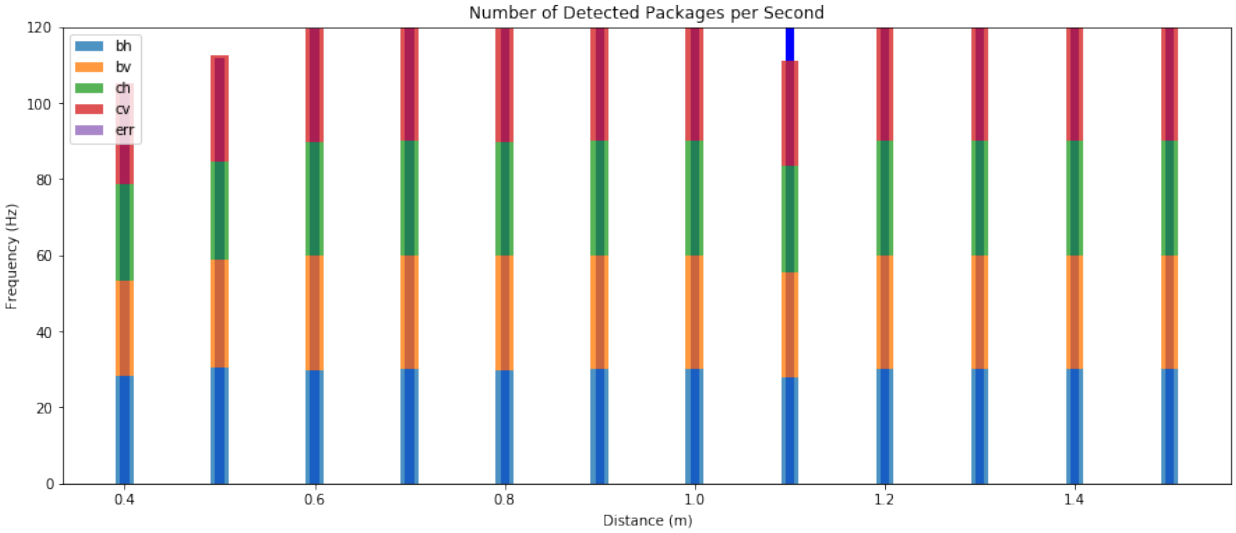
We see a very clear correlation between calibration pulse-width and base-station distance. When the sensor is very close, the difference is huge while the rate of change rapidly decreases as we reach a more normal operation distance.

All distances larger than 0.8 m is here seen to give clean data. The systematic change in signal with distance is not important, as the sensor was not kept directly in front of the base-station.

Table 4: Error-Log for Minimum Distance Testing

distance to base-stations (m)	# collected points	# lost points	% error rate	# signal gaps
0.40	526	74	12	74
0.50	558	42	7	42
0.60	598	2	0	2
0.70	600	0	0	0
0.80	598	2	0	2
0.90	600	0	0	0
1.00	600	0	0	0
1.10	600	0	0	1
1.20	599	1	0	1
1.30	600	0	0	0
1.40	600	0	0	0
1.50	600	0	0	0





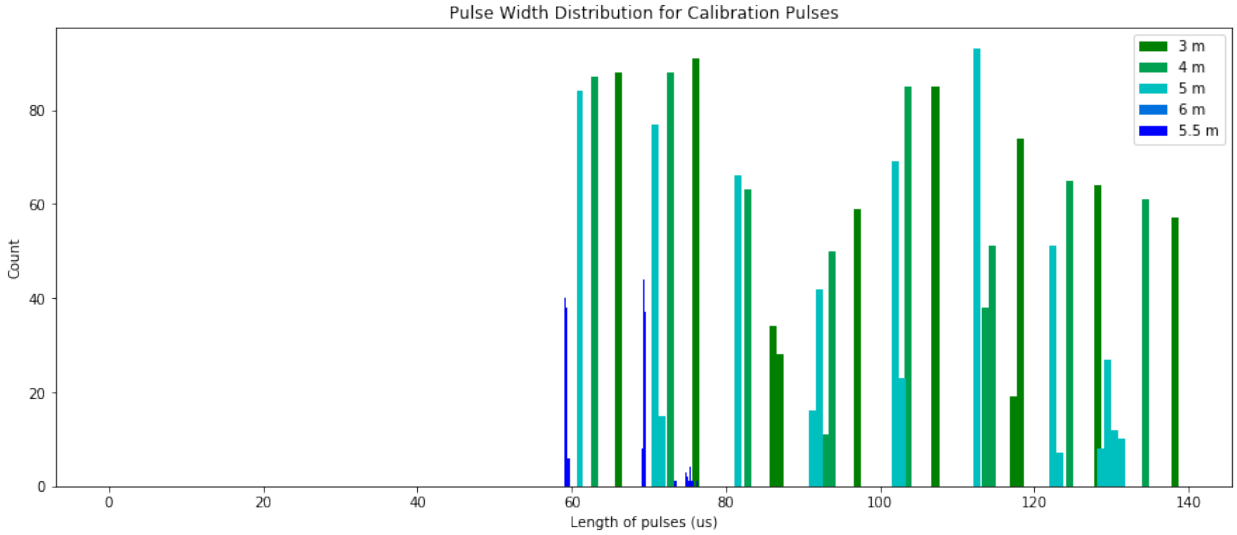
2.5 Maximum Operation Distance

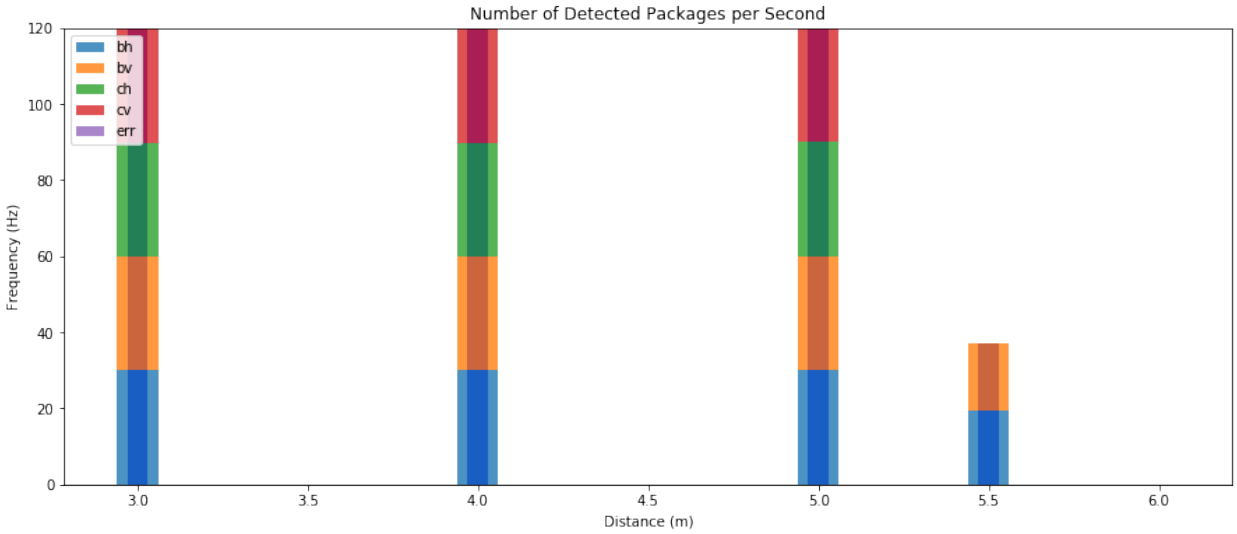
With procedure like to that used for min-distance, the max-distance of the system was determined.

We see there is no signals detected past 5.5 meters, and that only a few signals are collected at 5.5 meters.

Table 5: Error-Log for Maximum Distance Testing

distance to base-stations (m)	# collected points	# lost points	% error rate	# signal gaps
3.00	599	1	0	1
4.00	599	1	0	1
5.00	600	0	0	2
6.00	0	0	100	0
5.50	186	412	68	119

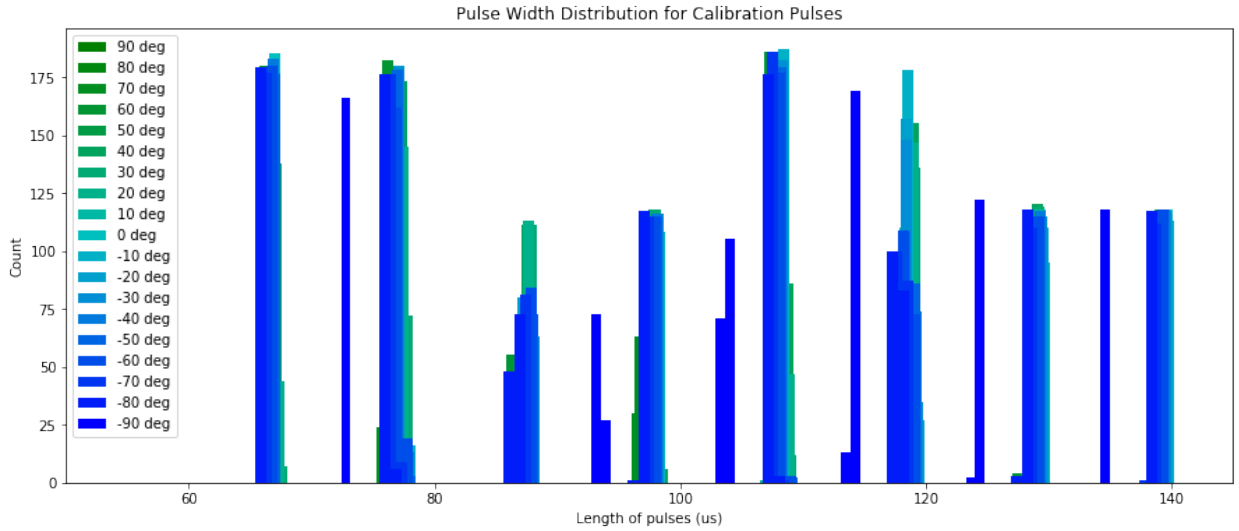


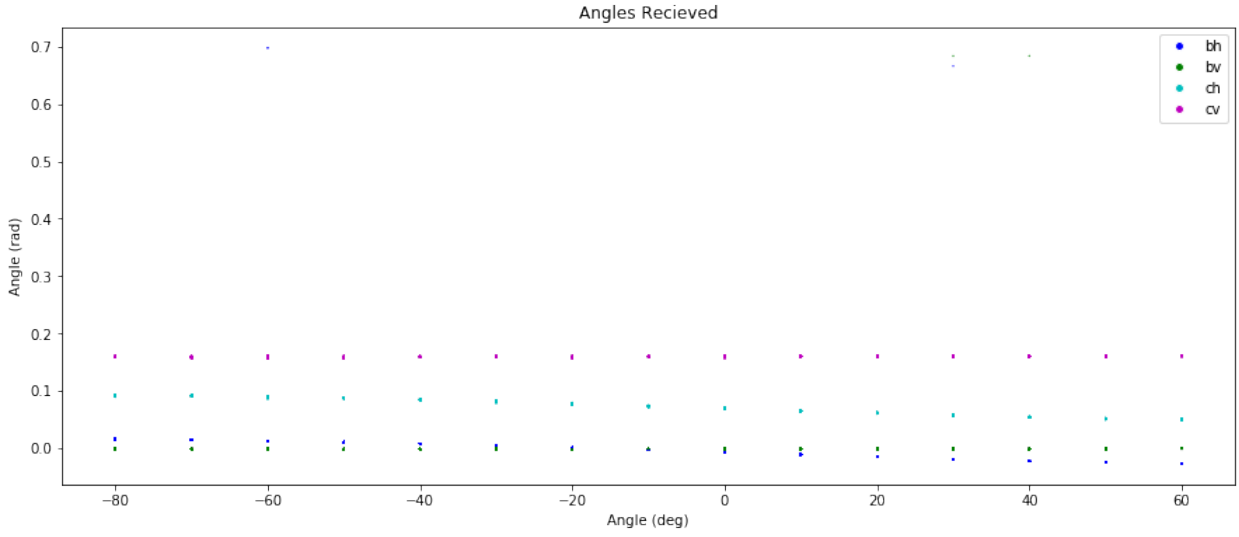


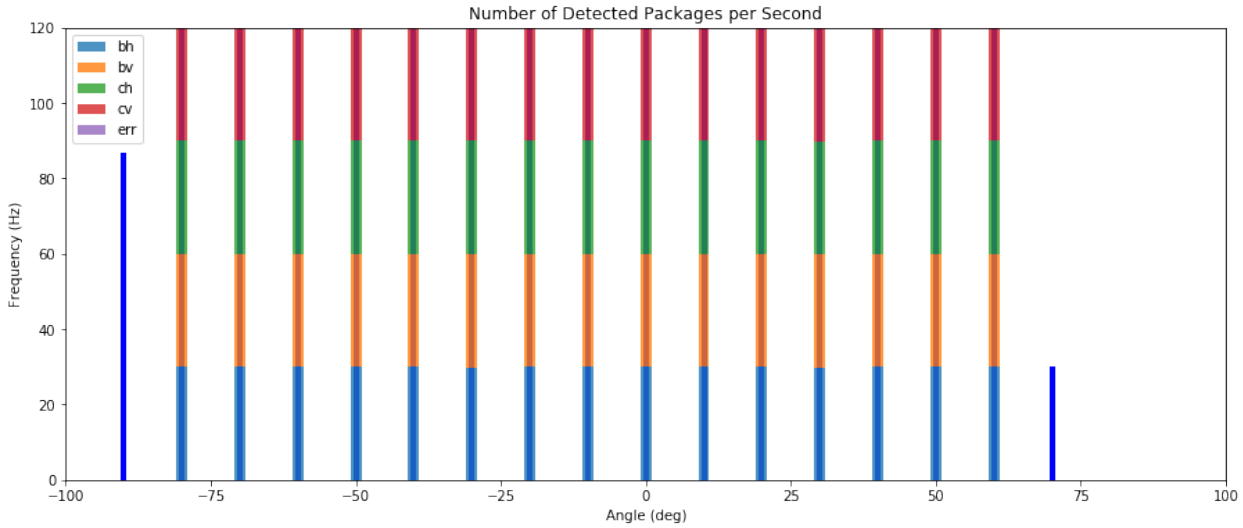
2.6 Sensor Angle

To determine the signal-integrity at different sensor angles, the sensor was taped to the angle-measurement device and rotated through 180° from 90° to −90° horizontally.

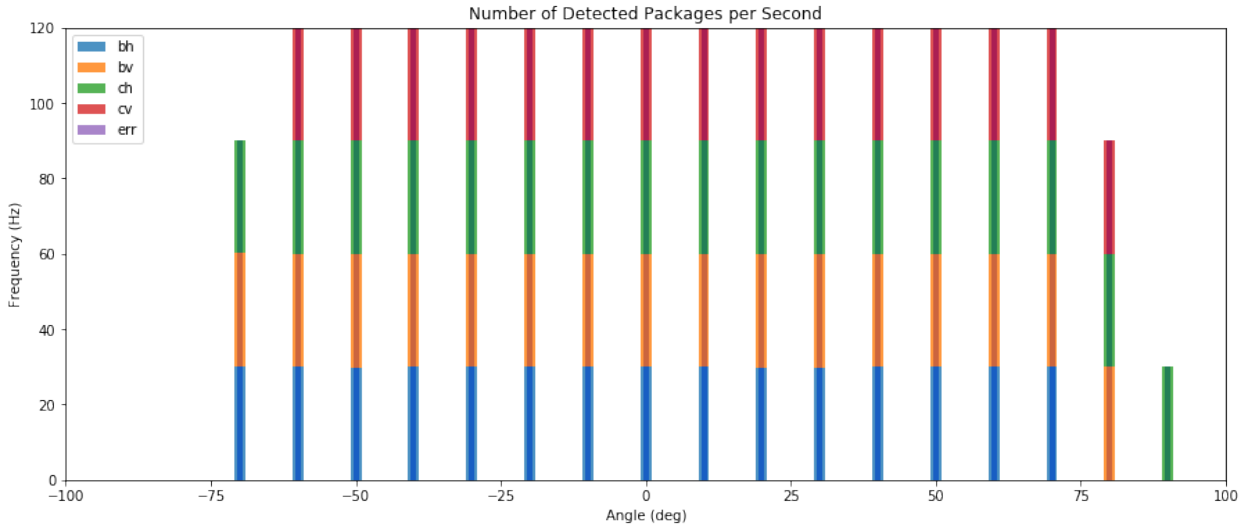
We see that sensor-rotation has little effect on the number of detected packages and pulse-durations when we operate within suitable bounds (±60°). There is an apparent asymmetry in the operation of the sensor from left to right, but this difference is likely primarily caused by mechanical shielding from misalignment between the base-stations and the sensor during the test-setup.







The rotation was then repeated with the sensor oriented vertically to much the same effect. The same range is obtained and a slight asymmetry between top and bottom is observed.



2.7 Base-Station Angle

To determine the signal-integrity at different sensor angles, the sensor was screwed/taped to a angle-measurement device and rotated both horizontally and vertically. We observe that the measurements are very accurate, but that there is a systematic error present.

The following factors have been determined to not impact this systematic deviation significantly:

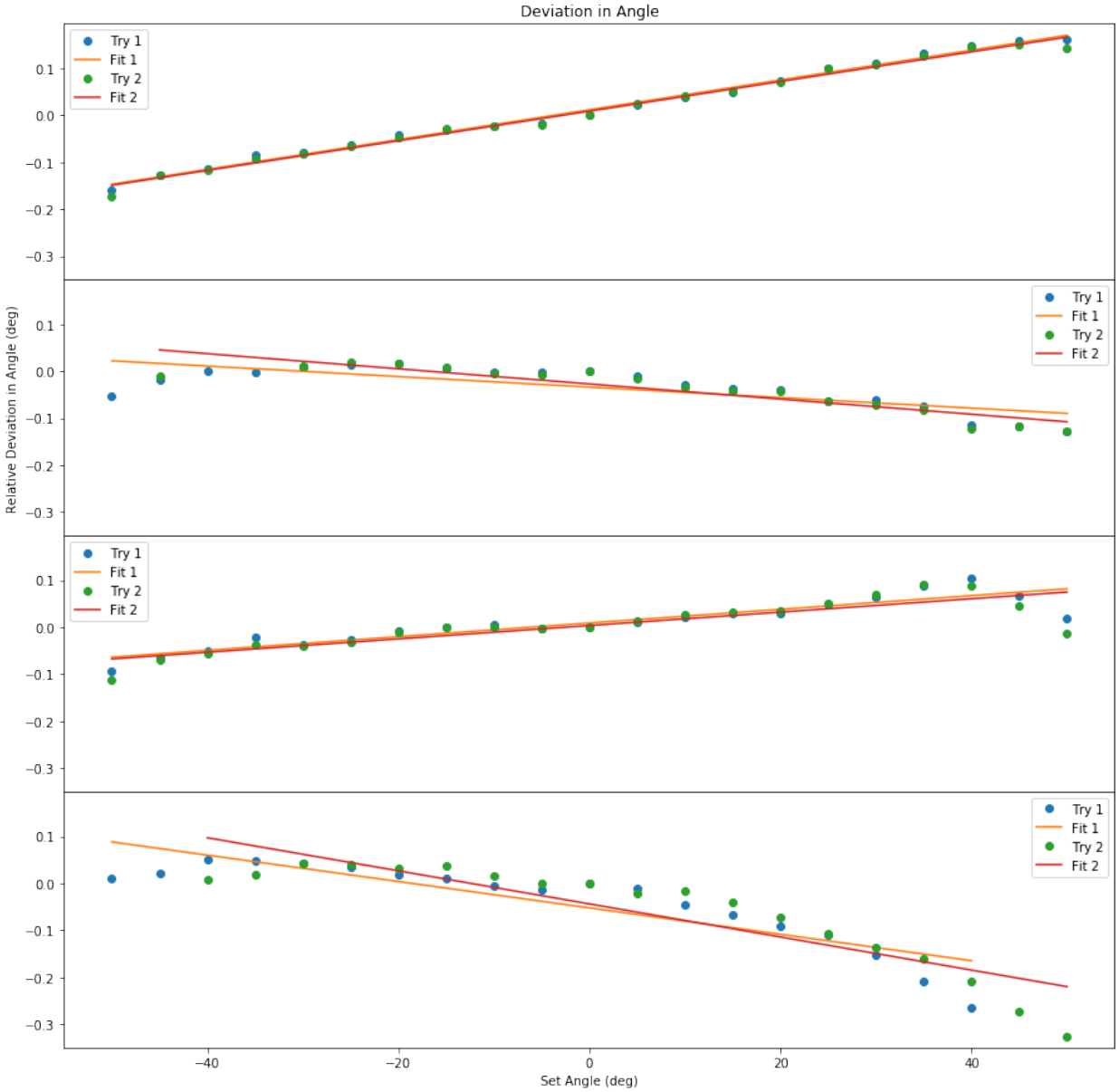
- different time/days
- different locations
- whether the base-station is operating as master or slave

whereas these factors do cause a noticeable change:

- which base-station is used
- which angle (horizontal or vertical) of the base-station is measured
- the orientation of the base-station

We have not been able to determine the root cause of this, but are left to assume there is some imperfection in the operation of the base-stations (a systematic non-constant rotational speed, imperfections in the line-laser mirrors, imperfection in the cover-class etc.).

The below graphs show the error in angle reading for two different base-stations for the two different angles. Each measurement was taken twice, with a separate curve-fit performed for each data-set.

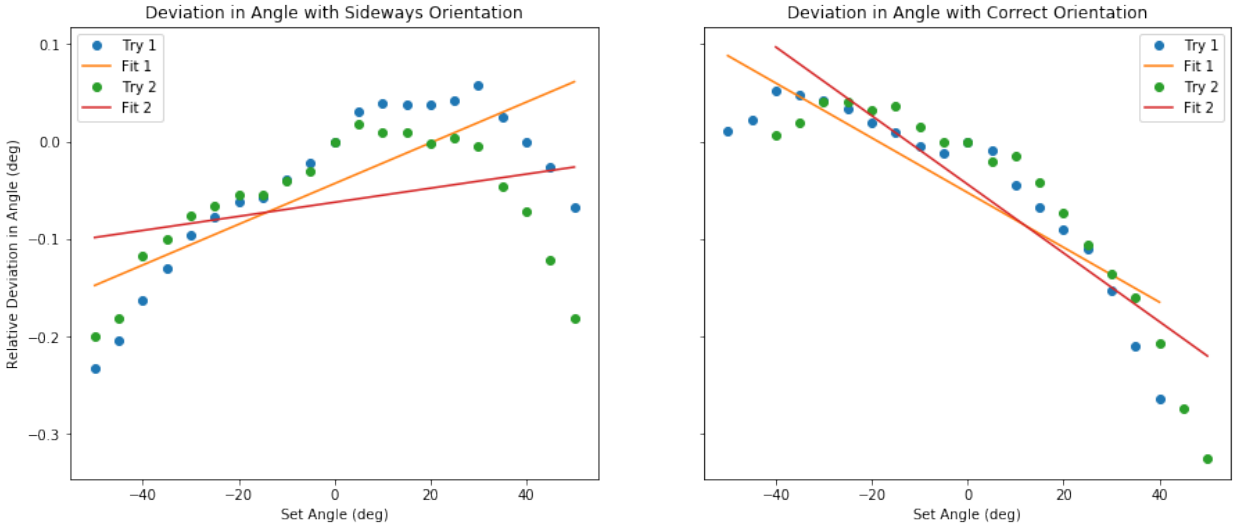


The slopes are computed to be:

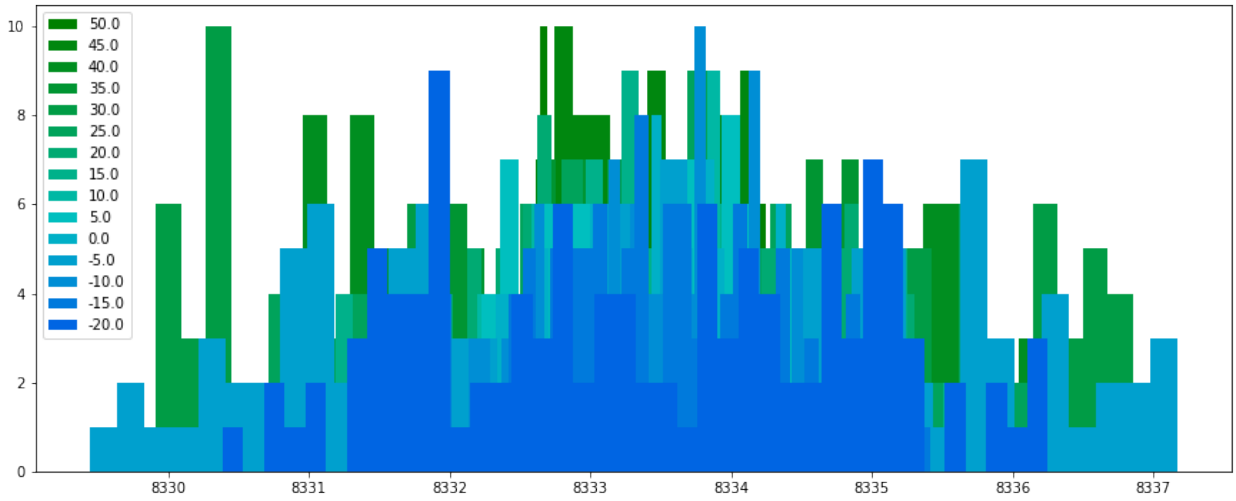
- Base-station 1 horizontal: 0.00317 and 0.00315, average: 0.00316
- Base-station 1 vertical: -0.00111 and -0.00161, average: -0.00136
- Base-station 2 horizontal: 0.00145 and 0.00142, average: 0.00144
- Base-station 2 vertical: -0.00280 and -0.00352, average: -0.00316

The average slope across all measurements is 1.8×10^{-5} with variance 6×10^{-6}

The below graphs are collected from the same base-station and for the same angle, but with the base-station in the correct orientation (standing normally) and the base-station laying on it's side. As we can see, there is a significant difference between the two, with the normal orientation yielding the most linear result. This issue is left as a loose point for further investigation.



In an attempt to investigate a possible correlation between the arrival-times of each package and the angle we obtain the following histogram. We see what appears to be a normal distribution with mean period of 8333.3909μs and with standard deviation of 2μs. This close to the expected 8333.33μs. There is no systematic drift with angle, so a systematic clock-drift can not be the cause of this error.



2.8 Continuous Motion

In order to evaluate the performance of the system during a scenario more reminiscent to that of the application, we move the sensor around in the room continuously while capturing data for approximately one minute over the serial-port.

The fist graph is plotted using data-sets collected with the initial system-configuration. We see a very clear issue arise with regards to lost readings, causing large gaps in the signal.

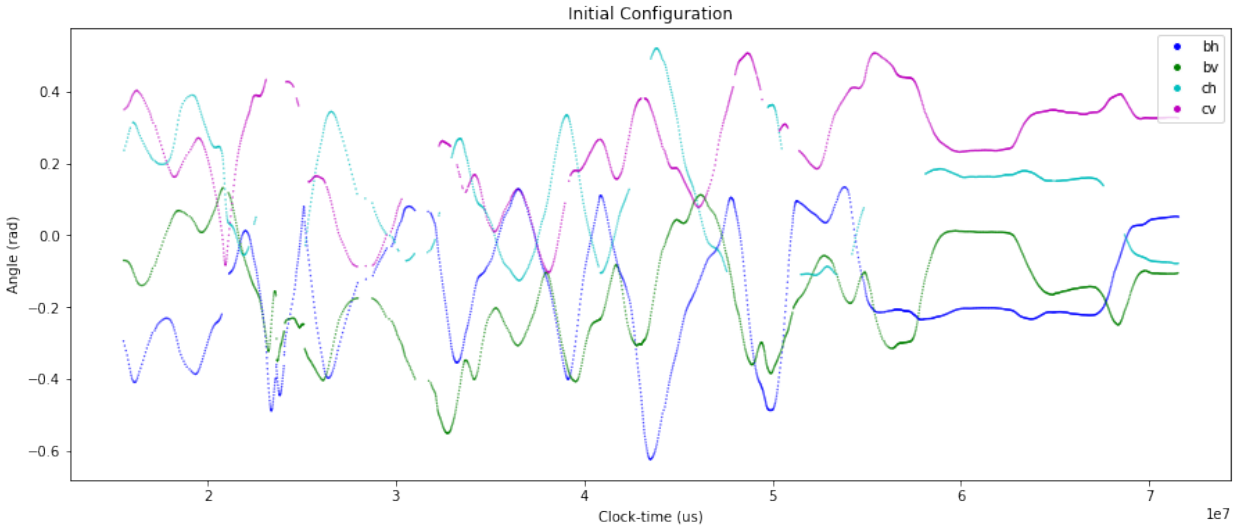
In order to investigate this issue we attempted to use three sensors simultaneously (all pointing in different directions), but this still left blind-spots. We next connected the sensor and comparator output to the oscilloscope and realised the issue would be solved by lovering the comparator-value. The potential issues with loss of accuracy from ramp on analog signal is small and this drastically improved reception as can be seen in the second graph below.

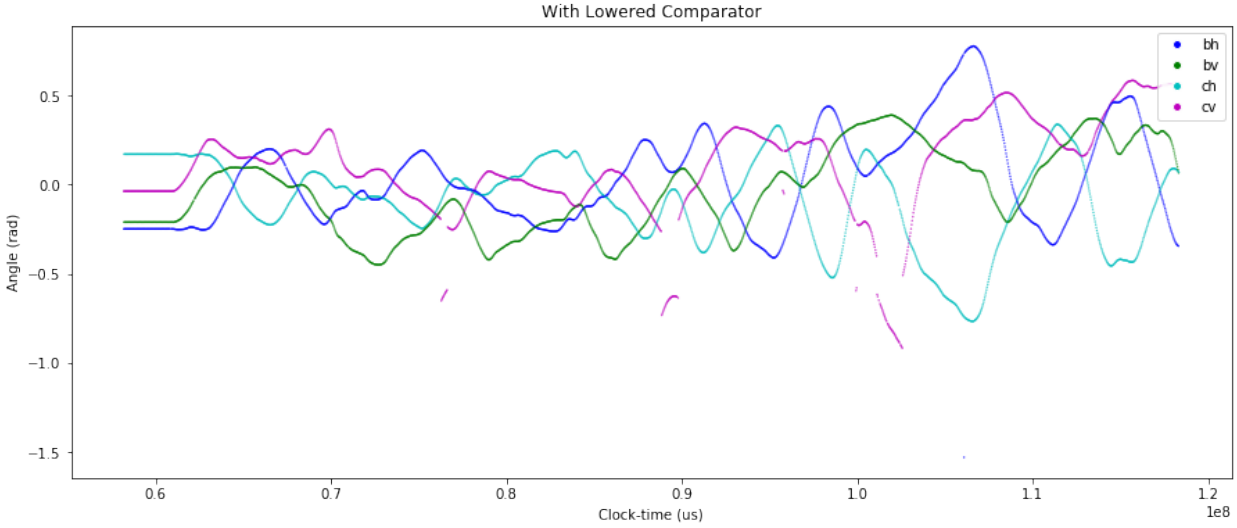
The second graph do however have issues with discontinuities for certain angles. This is caused by reflection, and implementing a system for selecting the widest (most powerful) signal pulse enabled us to pick out the right signal pulse when we have multiple arriving for a given package.

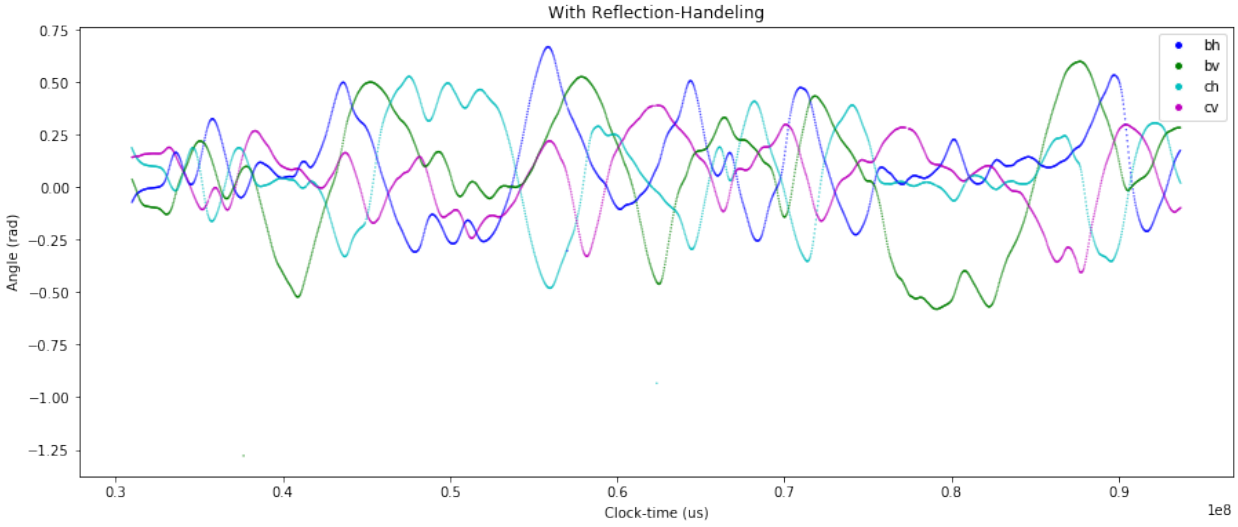
The third graph is the performance of the system as of 18. Aug 2017. There is few packages lost and there is no reflections present.

Table 6: Error-Log for Different Configurations During Continuous Motion

Description	# collected points	# lost points	% error rate	# signal gaps
Initial configuration	5881	1260	17	786
With lowered comparator	7200	17	0	19
With reflection-handling	7497	33	0	32


Initial Configuration
bh
bv
ch
cv
Angle (rad)
0.4
0.2
0.0
−0.2
−0.4
−0.6
2
3
4
5
6
7
Clock-time (us)
1e7


With Lowered Comparator
bh
bv
ch
cv
Angle (rad)
0.5
0.0
−0.5
−1.0
−1.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
Clock-time (us)
1e8

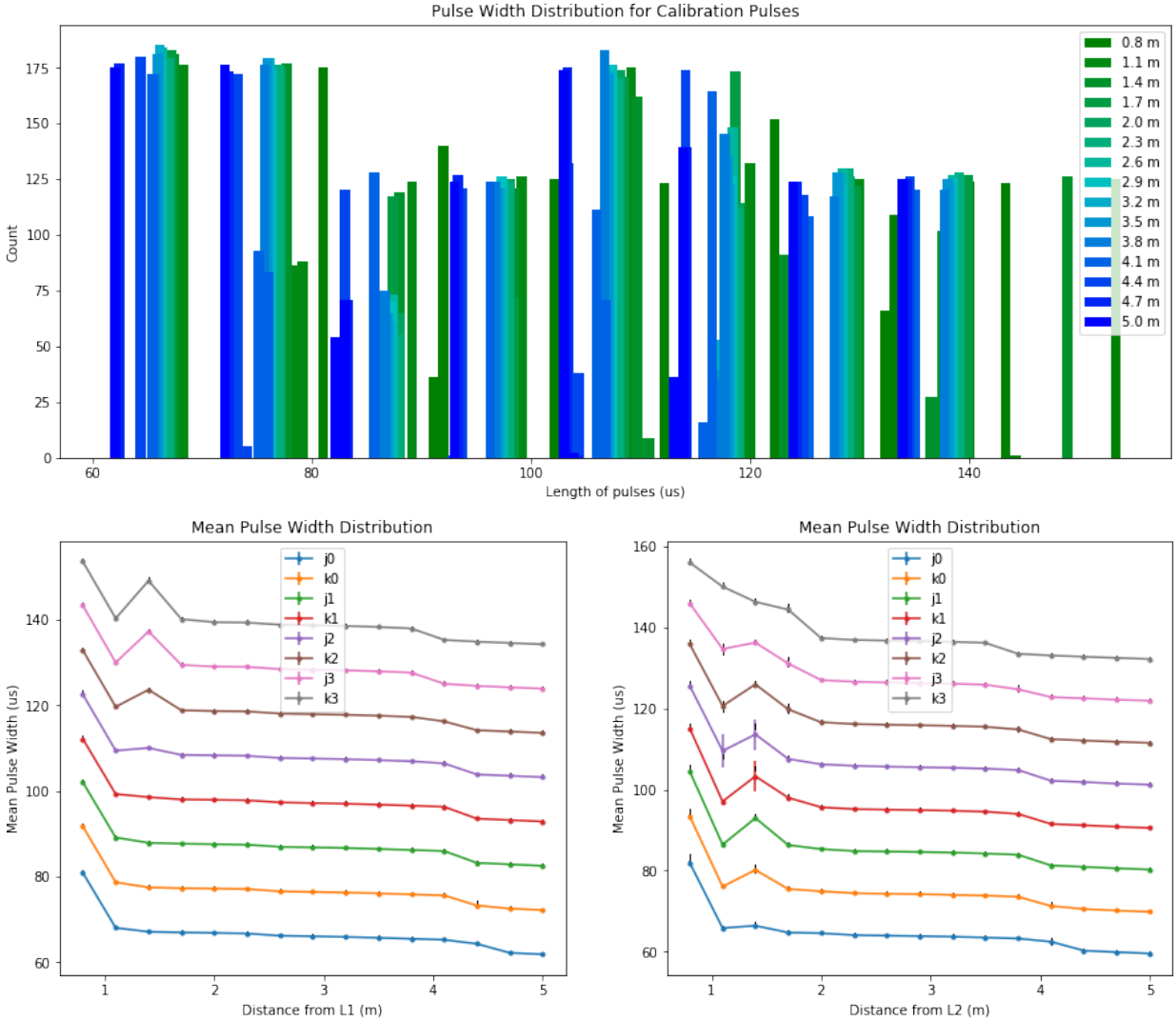

With Reflection-Handeling
bh
bv
ch
cv
Angle (rad)
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Clock-time (us)
1e8

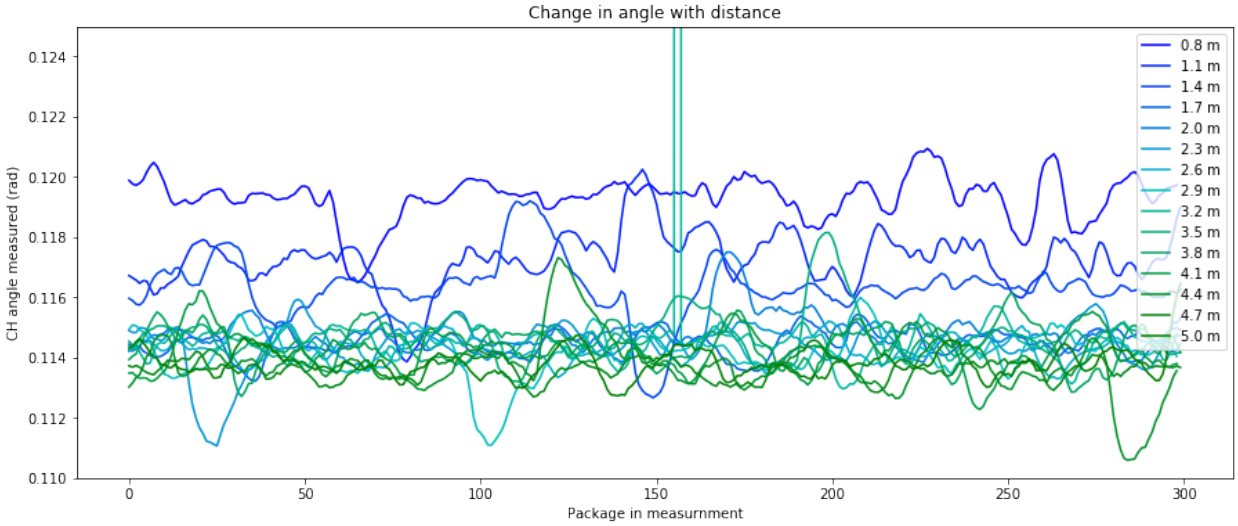
2.9 Dependence of Angle-Reading on Sensor Distance

In order to determine whether the distance has an impact on the angle-reading as could be indicated by the earlier distance-readings, a distance-measuring laser is positioned at the exact centre of a base-station and the laser-beam pointed to make a line going through the sensor. When moving the sensor away from the base-station, the sensor is moved so that the laser is always kept on the vertical axis going through the sensor.

The pulse-widths decrease with increasing length from the base-stations as expected. We do however see a decrease in pulse-length at around 1 meter - the cause of this is unknown but it is a minor point as the classifier still functions.

There is a discernible increase in angle as the sensor moves away from the base-station, but the difference is on the order of 0.006 rad $\approx 0.3°$ and is as such fairly insignificant. We also see that a significant change in angle only occurs for readings performed closer than 2 meters. We therefore conclude that the angle does not depend on distance when operating in the normal range.

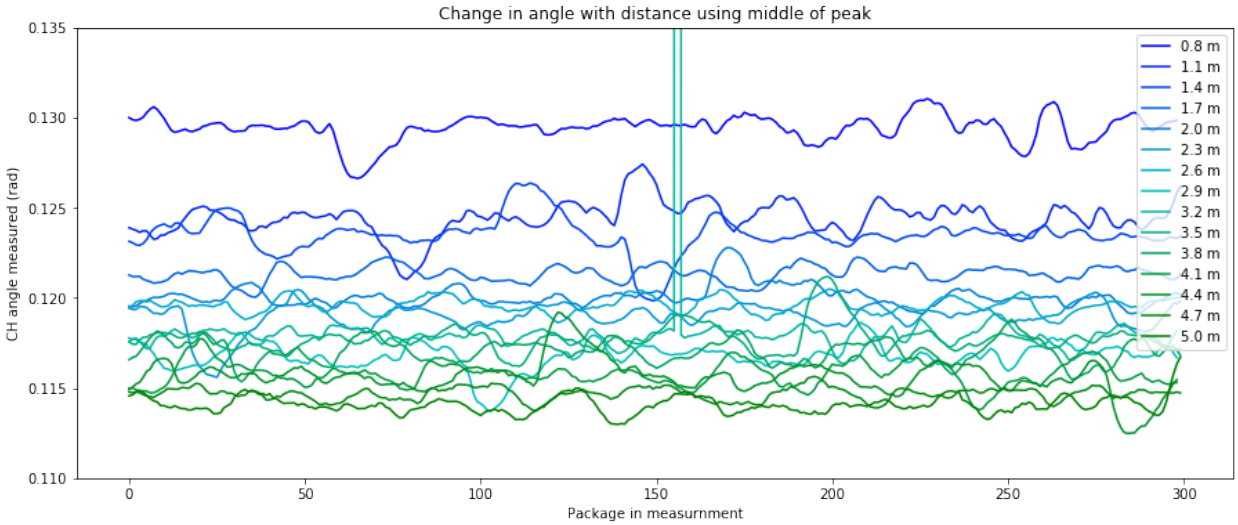




2.9.1 Measure Angle from Middle of Signal Peak Instead?

Due to the potential change in start time, the question was raised as to whether it would be beneficial to determine the angle reading not from the rising edge of the appropriate line-laser peak, but form the middle of this peak. The following computes the middle distance of the data used to plot the graph above so that we can compare the two methods.

We see that there is significantly better accuracy from not taking the width of the pulse into account (here, difference is 0.015 rad $>$ 0.006 rad for rising edge measurement above). This could be caused by the saturation of the OpAmp causing a delay for the signal to go back down again. We are left considering it beneficial to continue referencing the rising edge as we have been doing.

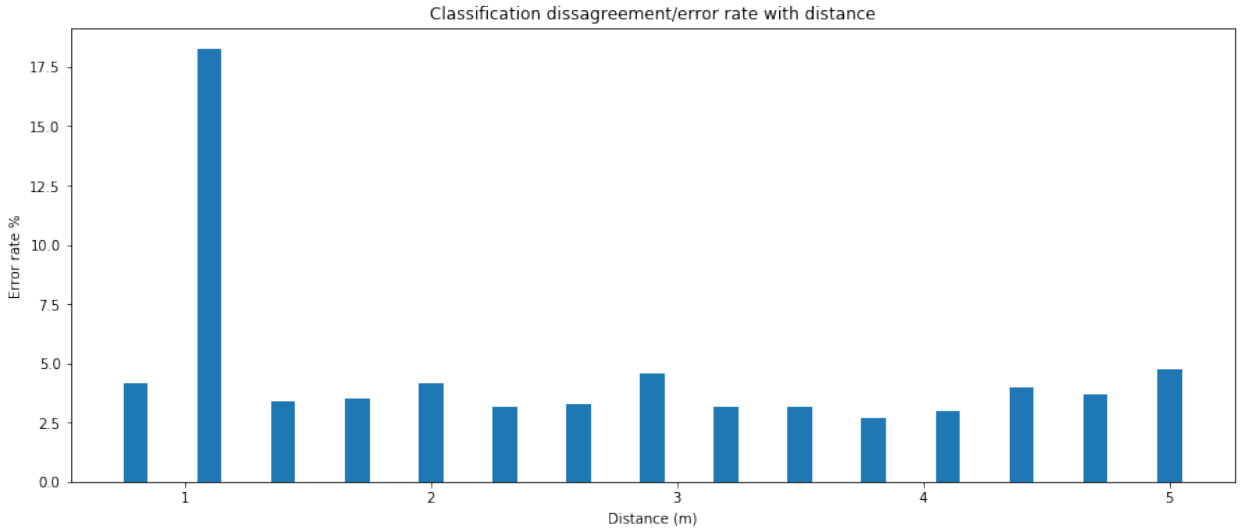


2.9.2 Verifying Pulse Classification

In order to verify that the pulses are classified correctly, we use two methods:

1. comparing the two classifiers we have at hand, and
2. ensuring the angle-measurement for each angle makes sense (no jumps, discontinuities or switching of values).

We begin by comparing the classifications on the data-set of varying distances form the base-station. We see there is a high degree of agreement between the two. This is sufficient, as the calibration only needs to get a given sequence calibrated correctly before the time-based classifier takes over. This initial pulse-width classification process will typically be done in a good operating-rage of the system, and will as such not have to be able to handle too large a range of distances.



2.10 Improvements and Suggestions

Based on the investigations performed in this section, the following improvements could be implemented:

- The accuracy of the system is limited when moving away from the immediate volume surrounding the calibration-points. The cause of this is unknown and needs further investigation.
- The calibration could be done using more point for increased accuracy.
- There occasionally arrives stray angle-readings with a value significantly different from what they should be. What this is caused by is unclear, one possible source is when the timer overflows (occurs about once a minute).
- A companion script could be written to read the verbose output and dynamically display the information to get a real-time diagnostic tool for the system.

2.11 Conclusion

The system perform is able to indicate positions with limited accuracy when moving away from the immediate calibration-volume. It is very possible to use the system for navigation, but the user must be aware of this limitation and take it into account for the application.

Differences in signal strength (and thus pulse-width) is:

- little affected by external interference
- little affected by base-station angle
- little affected by sensor angle
- little affected by distance between sensor and base-station

The operational parameters of the system is here demonstrated to be as follows:

- Range: 0.8 to 5 meters
- Base-station angle range: $\pm 55^\circ$
- Sensor angle range: $\pm 60^\circ$ (in both horizontal and vertical axis)
- Direct sunlight stops the system from working, but indirect sunlight and indoor lighting is generally not an issue

3 Base-Station Pose Calibration

In order to be able to determine the sensors position from the angle-measurements, we need to know the pose (position and rotation) of both base-stations. This is difficult to measure directly, so we seek to determine the base-station positions by taking angle-measurements with the sensor in known positions.

There are several possible strategies for determining the base-station positions. The first we attempted involved taking many readings from random points in space and using this information to determine the poses of the base-stations relative to one another. Thereafter, points on the floor can be collected to define a xy-plane etc.

We did however experience issues getting this method to converge. There is known issues with the accuracy of the system, and under the current conditions there is a great number of local optima that makes small changes in initial conditions lead to large differences in the solution.

The strategy we have implemented thus far is one in which four points in a square on the ground is collected and each base-station pose fitted to this calibration-square separately. The method has fair accuracy and is very easy to perform, especially when using a physical frame with the appropriate square geometry.

3.1 Vive Position Calibration using 4 Points

With this calibration approach we attempt to use the smallest number of data-points to reduce the number of local optima that can cause our minimisation-problem to converge to an incorrect solution. We define a coordinate-system in the room and position a square with sides of known length d in the xy plane such that:

- $\mathbf{p}_1$ is at the origin $(0, 0, 0)$
- $\mathbf{p}_2$ is along the positive y-axis $(0, d, 0)$
- $\mathbf{p}_3$ is along the positive x-axis $(d, 0, 0)$
- $\mathbf{p}_4$ is on the last corner of the square $(d, d, 0)$

Record angle-measurement $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ and $\mathbf{a}_4$ respectively from these points.

3.1.1 Strategy

The calibration is performed on each base-station separately. The algorithm can be outlined as follows:

- Convert base-station angles to vector directions.
- Define lines from vector directions that go through the same point (the base-station, currently defined to be the origin). We know the points will fall somewhere on these lines.
- Search for points on lines 1, 2 and 3 that conform to the known geometric relations between points 1, 2 and 3. Two solutions will exits.

- Perform this for both base-stations and compute position $\mathbf{p}_4$ using all 4 combinations of base-station positions and the base-station angles $\mathbf{a}_4$ for this point.
- Choose the base-station positions that make the computed $\mathbf{p}_4$ fall closest to $(d, d, 0)$.

Refer to Calibration.py for an implementation of this algorithm in Python. This implementation has been translated with only minor changes to C++ for use directly on the Teensy module.

3.1.2 Mathematical foundation

Using the two angles measured for a point we are able to compute the line relative to the base-station on which the point has to be located. Let the lines $\mathbf{L}_1$, $\mathbf{L}_2$, $\mathbf{L}_3$ correspond to these lines passing through points $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ respectively, relative to the given base-station.

A line can be parameterised by

$$\mathbf{L}(t) = \mathbf{a} + t\mathbf{b}$$

We define the centre of the base-station as the origin such that $\mathbf{a} = (0, 0, 0)$, leaving us with

$$\mathbf{L}_n(t) = t\mathbf{b}_n$$

as the general equation of a line, where b is defined as the direction-vector derived from the intersection of the planes described by the base-station angle measurements.

We are seeking points $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ on $\mathbf{L}_1$, $\mathbf{L}_2$ and $\mathbf{L}_3$ respectively such that

$$|\mathbf{u}| = |\mathbf{v}| = 1 \quad \cup \quad \mathbf{u} \cdot \mathbf{v} = 0$$

where

$$\begin{aligned} \mathbf{u} = (u_x, u_y, u_z) &= \mathbf{p}_2 - \mathbf{p}_1 \\ &= \mathbf{L}_2(t_2) - \mathbf{L}_1(t_1) \\ &= t_2\mathbf{b}_2 - t_1\mathbf{b}_1 \end{aligned}$$

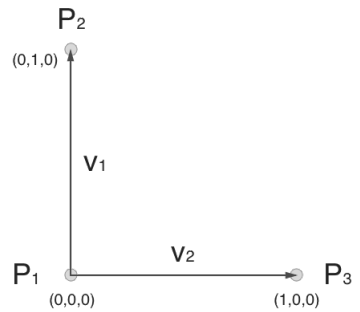
$$\begin{aligned} \mathbf{v} = (v_x, v_y, v_z) &= \mathbf{p}_3 - \mathbf{p}_1 \\ &= \mathbf{L}_3(t_3) - \mathbf{L}_1(t_1) \\ &= t_3\mathbf{b}_3 - t_1\mathbf{b}_1 \end{aligned}$$

Expanding the constraining equations into components, we get three equations.

$$\begin{cases} {u_x}^2 + {u_y}^2 + {u_z}^2 = 1 \\ {v_x}^2 + {v_y}^2 + {v_z}^2 = 1 \\ u_x v_x + u_y {v_y}^2 + u_z {v_z}^2 = 0 \end{cases}$$

Breaking these into components we reduce the system

Figure 5: Problem Geometry



$$
\begin{aligned}
|\mathbf{u}| =&(t_2 b_{2,x} - t_1 b_{1,x})^2 + (t_2 b_{2,y} - t_1 b_{1,y})^2 + (t_2 b_{2,z} - t_1 b_{1,z})^2 \\
=&(t_2 b_{2,x})^2 - 2 t_2 b_{2,x} t_1 b_{1,x} + (t_2 b_{1,x})^2 \\
&+(t_2 b_{2,y})^2 - 2 t_2 b_{2,y} t_1 b_{1,y} + (t_2 b_{1,y})^2 \\
&+(t_2 b_{2,z})^2 - 2 t_2 b_{2,z} t_1 b_{1,z} + (t_2 b_{1,z})^2 \\
=&1
\end{aligned}
$$

Factoring out t_2

$${t_2}^2({b_{2,x}}^2 + {b_{2,y}}^2 + {b_{2,z}}^2) - 2t_2(b_{2,x} t_1 b_{1,x} + b_{2,y} t_1 b_{1,y} + b_{2,z} t_1 b_{1,z}) + (t_1 b_{1,x})^2 + (t_1 b_{1,y})^2 + (t_1 b_{1,z})^2 - 1 = 0$$

we get a second order polynomial on the form $m{t_2}^2 + nt_2 + o = 0$ with solution

$$t_2 = \frac{-n \pm \sqrt{n^2 - 4mo}}{2m}$$

where

$$
\begin{cases}
m = {b_{2,x}}^2 + {b_{2,y}}^2 + {b_{2,z}}^2 & = |\mathbf{b}_2|^2 \\
n = -2(b_{2,x} t_1 b_{1,x} + b_{2,y} t_1 b_{1,y} + b_{2,z} t_1 b_{1,z}) & = -2t_1(\mathbf{b}_2 \cdot \mathbf{b}_1) \\
o = (t_1 b_{1,x})^2 + (t_1 b_{1,y})^2 + (t_1 b_{1,z})^2 - 1 & = {t_1}^2 |\mathbf{b}_1|^2 - 1
\end{cases}
$$

$$t_2 = \frac{2t_1(\mathbf{b}_2 \cdot \mathbf{b}_1) \pm \sqrt{4{t_1}^2(\mathbf{b}_2 \cdot \mathbf{b}_1)^2 - 4t_1 |\mathbf{b}_2|^2 |\mathbf{b}_1|^2}}{2|\mathbf{b}_2|^2}$$

Similarly, solving $|\mathbf{v}| = 0$ for t_3 gives coefficients

$$
\begin{cases}
m' = |\mathbf{b}_3|^2 \\
n' = -2t_1(\mathbf{b}_3 \cdot \mathbf{b}_1) \\
o' = {t_1}^2 |\mathbf{b}_1|^2 - 1
\end{cases}
$$

so that

$$
\begin{cases}
t_2(t_1) = \frac{t_1(\mathbf{b}_2 \cdot \mathbf{b}_1) \pm \sqrt{{t_1}^2(\mathbf{b}_2 \cdot \mathbf{b}_1)^2 - t_1 |\mathbf{b}_2|^2 |\mathbf{b}_1|^2}}{|\mathbf{b}_2|^2} \\
t_3(t_1) = \frac{t_1(\mathbf{b}_3 \cdot \mathbf{b}_1) \pm \sqrt{{t_1}^2(\mathbf{b}_3 \cdot \mathbf{b}_1)^2 - t_1 |\mathbf{b}_3|^2 |\mathbf{b}_1|^2}}{|\mathbf{b}_3|^2}
\end{cases}
$$

Now, consider the last equation in the system:

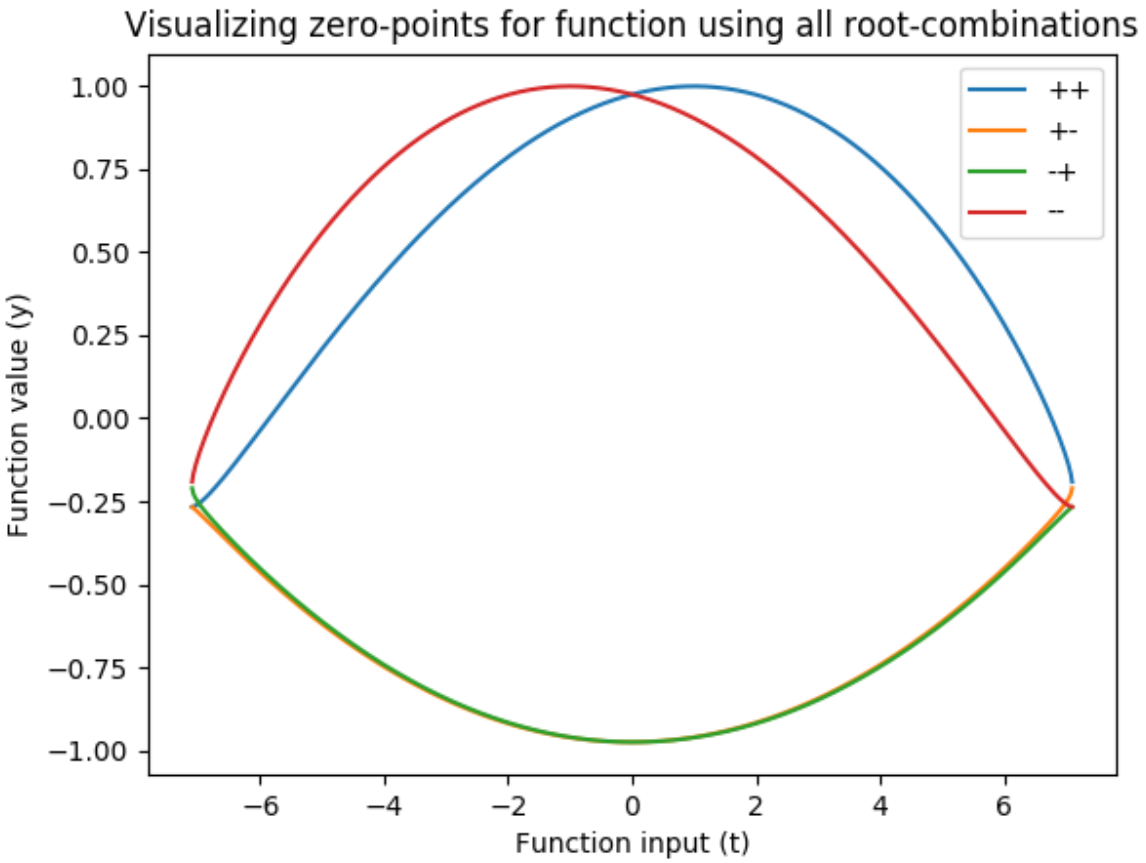
$$
\begin{aligned}
\mathbf{u} \cdot \mathbf{v} =&(t_2 b_{2,x} - t_1 b_{1,x})(t_3 b_{3,x} - t_1 b_{1,x}) + \\
&(t_2 b_{2,y} - t_1 b_{1,y})(t_3 b_{3,y} - t_1 b_{1,y}) + \\
&(t_2 b_{2,z} - t_1 b_{1,z})(t_3 b_{3,z} - t_1 b_{1,z}) \\
=&t_2 b_{2,x} t_3 b_{3,x} - t_2 b_{2,x} t_1 b_{1,x} - t_3 b_{3,x} t_1 b_{1,x} + (t_1 b1, x) + \dots \\
=&t_2 t_3(\mathbf{b}_2 \cdot \mathbf{b}_3) - t_2 t_1(\mathbf{b}_2 \cdot \mathbf{b}_1) - t_3 t_1(\mathbf{b}_3 \cdot \mathbf{b}_1) + {t_1}^2 |\mathbf{b}_1|^2 = 0
\end{aligned}
$$

We can now substitute t_2 and t_3 to get one equation with one unknown. There will in general be zero, one or two roots for each of t_2 and t_3 such that there could be from zero to four solutions to the general equation. In practice, it is observed that exactly two solutions exist for each equation. Point 4 is used to determine which solution is correct for the given setup.

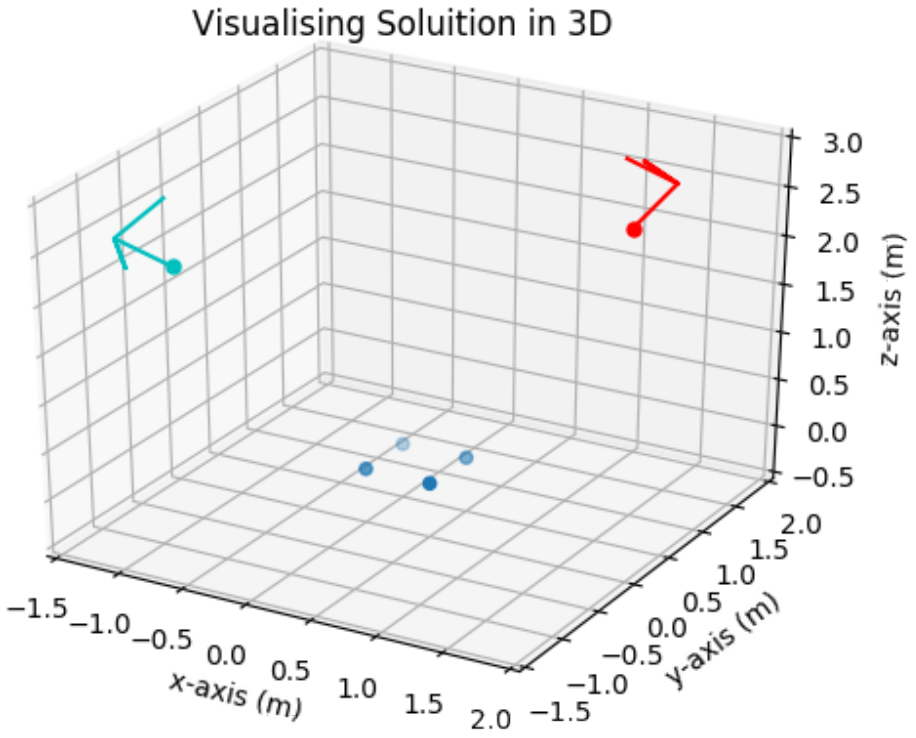
3.1.3 Algorithm

A Python script was written where the above computation is performed, see Calibration.py.

This is a visualisation of the 1d solution space. We solve for the zero-intercepts to determine the distance from the origin to point p_1 along line L_1. From this it is possible to infer t_2 and t_3 which makes the system fully defined.



Below is a representation of the solution for a sample setup. The base-station location and orientation is indicated with the 3d axis embedded in the main coordinate system, and the points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$ are located by reverse-engineering their position given the poses of the base-stations and their respective horizontal and vertical angles.



Calibrated poses for base-stations:

$$\text{Master} = \begin{cases} x = 1.471\,\text{m} \\ y = 1.616\,\text{m} \\ z = 2.580\,\text{m} \\ r_x = 2.521\,\text{rad} \\ r_y = -0.381\,\text{rad} \\ r_z = 2.083\,\text{rad} \end{cases}$$

$$\text{Slave} = \begin{cases} x = -1.351\,\text{m} \\ y = -0.924\,\text{m} \\ z = 2.452\,\text{m} \\ r_x = -2.473\,\text{rad} \\ r_y = 0.575\,\text{rad} \\ r_z = -0.897\,\text{rad} \end{cases}$$

The accuracy of these measurements is difficult to determine directly, as we are operating with significant distances in three dimensions. We therefore ended up using measured points to evaluate the accuracy of the fit (and therefore also the accuracy of the system). Points 1, 2 and 3 should

coincide perfectly with their theoretical locations as they are used to perform the fit. Point 4 can give us a sense of the quality/accuracy of the fit.

When calculating the points based on the angle-measurements and the computed base-station poses we get the following results:

$$\begin{aligned}
\mathbf{p}_1 &= (0.000\,\text{m}, -3.885 \times 10^{-16}\,\text{m}, 0.000\,\text{m}) \\
\mathbf{p}_2 &= (8.034 \times 10^{-13}\,\text{m}, 5.000 \times 10^{-1}\,\text{m}, 9.157 \times 10^{-13}\,\text{m}) \\
\mathbf{p}_3 &= (5.000 \times 10^{-1}\,\text{m}, 4.440 \times 10^{-16}\,\text{m}, -1.554 \times 10^{-15}\,\text{m}) \\
\mathbf{p}_4 &= (0.501\,\text{m}, 0.497\,\text{m}, 0.008\,\text{m})
\end{aligned}$$

We can see that $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ are very close to their theoretical locations, indicating our calibration-computation works. The slightly larger deviation from ideal in point $\mathbf{p}_3$ is caused by the fact that this point's location is updated slightly so as to make angle $\mathbf{u}$ and $\mathbf{v}$ perfectly perpendicular.

Point 4 gives us a sense of the performance of the actual calibration, as it was not used directly in the calibration. It is here seen to be within 0.01 m of its intended location at $(0.5\,\text{m}, 0.5\,\text{m}, 0\,\text{m})$, which is decent for several applications.

4 User-Guide

The prototype was developed using ThingML which is a code-generation language built around state-machines and asynchronous communication-messages.

4.1 Architecture

The functionality is split into 3 levels:

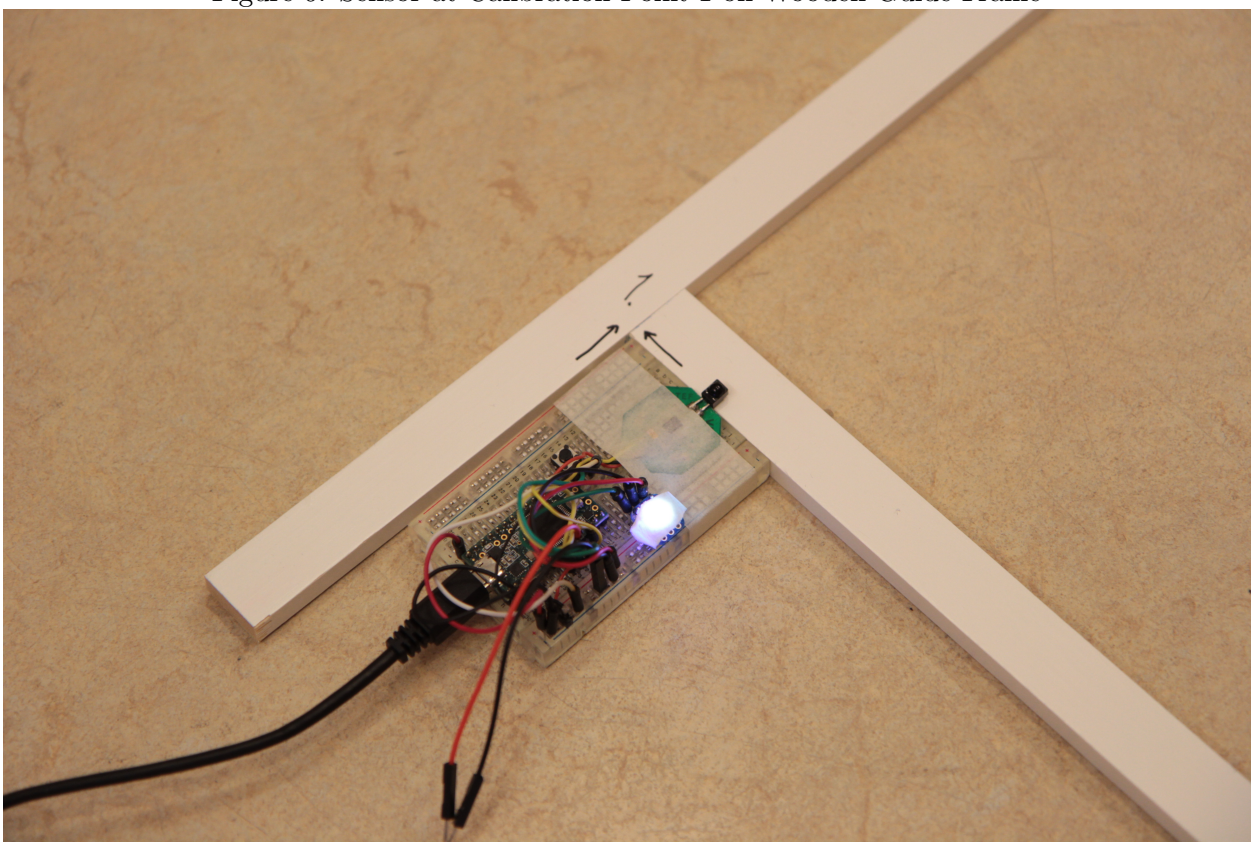
- The backbone represents the controller of the software - all messages passing form one part of functionality to another is handled by the backbone.
- The applicative models are abstractions for the backbone. These models transform raw data and compute required elements like position.
- The accessories level handles implementation-details for inputs and outputs. This includes sensors data, persistent storage in EEPROM and LED status indicators.

4.2 Operation

The Teensy can be put into two different operating modes - "Run" and "Calibrate". On launch, the system will try to load a calibration from EEPROM.

- If it succeeds, it will go directly to "Run" where it computes and outputs position.
- If if fails, it will go to to "Calibration" where the base-station poses are to be computed.

Figure 6: Sensor at Calibration Point 1 on Wooden Guide Frame



The system functions as follows:

- Calibrate by placing the base-station at the corners of a square that you use to define your coordinate-system and pressing the button to record a point. Record points in the following order:
  1. Point at origin (0, 0, 0)
  2. Point at 0.5 meters distance on the positive y-axis (0, 0.5, 0)
  3. Point at 0.5 meters distance on the positive x-axis (0.5, 0, 0)
  4. Point at 0.5 meters distance on the x-axis and y-axis (0.5, 0.5, 0)
- A green light indicates the point has been recorded, a red light indicates the point must be recorded again.
- When all 4 points have been recorded the LED will:
  - flash dark green if calibration is successful.
    - The calibration will be stored to EEPROM automatically before the system transfers to the running mode.
  - red if the calibration failed and must be redone.

- Long-pressing the button at any time will start a new calibration (this can also be used to restart the calibration if an incorrect data-point has been collected).

4.3 Debugging and Verbose Mode

There is debugging-functionality in place using serial-prints to communicate computed values, states and alert about events. The verbose-mode can be used for outputting data in a format that can be read by Parsing.py in order to produce an array of Data-classes that contain all the information and that can be used to diagnose the operation of the system. The file Visualizing.py has multiple methods to quickly produce useful output (see Validation for details).

- Use a jumper-cable to short pins 11 and 12 on startup to ender "debug" mode and get serial-prints.
- Double press the button to toggle verbose printing.

4.4 LED Status Messages

The base colour of the LED indicates state as follows:

Table 7: States

ID	Color	State
1	ORANGE	Load_Calibration
2	YELLOW	Calibration
3	ORANGE	Validate_Calibration
4	GREEN	Runner

The LED will flash for events according to Table 8. The most important takeaway is that

- red: recalibrate
- green: things are working well
- blue: remove obstructions and ensure sensor is in operating-volume
- others: check above table and serial-print

Table 8: Status Messages

ID	Color	Priority	Information	Sentiment	Required action
0	RED	1	Computed lines do not intersect	Severe	Recalibrate
1	BLUE	0	One or more angles not received	Fatal	Remove obstruction or reduce angles
2	LIME	1	Recording data-point for calibration	Good	Wait with sensor stationary
3	LAVENDER	-1	Reflection present	Handled	None
4	DARK_GREEN	4	Base station calibration successful	Good	None
5	DARK_GREEN	3	Loading calibration from EEPROM succeeded	Good	None
6	DARK_RED	3	Loading calibration from EEPROM failed	Not ready	Perform manual calibration
7	DARK_RED	4	Calibration failed	Retry	Recalibrate
8	GREEN	2	Calibration point collected	Good	Proceed with calibration on next point
9	RED	2	Calibration point not collected	Retry	Retry at this point
10	YELLOW	0	Unexpected package arrival time	Handled	None
11	BROWN	1	Consistently unexpected package arrival time	Severe	Wait for automatic recalibration of package arrival times
12	CIAN	0	Calibrating package arrival times	Not ready	Wait and ensure good reception
13	DARK_GREEN	0	Package arrival times calibrated	Good	System is operating normally
14	PURPLE	-1	No signals received	Fatal	Turn on/move closer to base stations/check sensor
15	GREY	0	Signal type not recognised (12 not done)	Severe	Move sensor away from base stations, check for interference
-	-	-	LED stops flashing - from loop or crash	Fatal	Restart Teensy, debug if persistent

5 External Sources

- https://github.com/ashtuchkin/vive-diy-position-sensor
- https://github.com/nairol/LighthouseRedox/blob/master/docs/Light%20Emissions.md
- http://www.nxp.com/assets/documents/data/en/reference-manuals/K20P64M72SF1RM.pdf