ANALYSIS OF SMARTPHONE EARTHQUAKE EARLY WARNING NETWORKS IN CHILE AND COSTA RICA

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

GEOLOGY AND GEOPHYSICS

MAY 2020

BY
Jonathan Avery

Thesis Committee:
James Foster, Chairperson
Ben Brooks
Neil Frazer

Keywords: Earthquakes; Earthquake Early Warning; EEW; Smartphone; GNSS; Accelerometer

Acknowledgements

I would like to thank my advisor, James Foster, for his help and advice in completing this thesis over the past 2 ½ years; Ben Brooks for allowing me to work on this project with the USGS, and for securing the funding that allowed me to complete the work; Neil Frazer for teaching me how to code in R, which allowed me to complete the data analysis presented in this work; Chris Duncan and Todd Ericksen for and helping me get to get our server setup to archive and process the data, and run ; our partners in Chile, Juan Carlos at Chile's Centro Sismológico Nacional and Hector Para at Chiles Instituto Geográfico Militar, for securing permissions to install our equipment around the country, and Marco Aros and Samuel Silva for driving me around and helping with the installation of our equipment in Chile; finally, I would like to thank our partners in Costa Rica, Marino Protti Qesada and Floribeth Vega Solano for helping us to install our smartphone EEW network in Costa Rica.

Abstract

Seismometers and continuous GNSS stations can be used to detect the ground motion from an earthquake, and issue an alarm before strong ground shaking reaches a population center. Although Earthquake Early Warning (EEW) systems have been around since 1991, they have been implemented in only a few countries due to the large costs associated with installing, maintaining, and monitoring them. The accelerometers and GNSS chips inside modern smartphones can detect the shaking and displacements from large earthquakes, and so smartphones can augment or even replace the high cost scientific grade instruments that are traditionally used in EEW networks to help lower costs, and make these systems more affordable for developing countries. Our team has developed an EEW system that uses smartphones for its sensors, and has installed networks in Chile and Costa Rica. Here we analyze the performance of the internal and external GNSS chips used in our system while installed in several building types, the latencies associated with the message transmission time, and the performance of our Costa Rica EEW system in a real-world application by simulating the 2012 Nicoya Earthquake. Our analysis suggests that the external UBLOX GNSS chip used in our system should be able to detect the displacements from large earthquakes, and that our network latency isn’t large enough to significantly reduce our ability to issue timely warnings.

Table of Contents

Table of Figures/Images/Tables

Introduction

Earthquakes account for 60,000 deaths annually on average, 90% of which are from developing countries [Kenny 2009], and have caused $2.9+ Trillion US-dollars in damage since 1900 [Daniell et al. 2012]. To help mitigate the impacts of these disasters, Earthquake Early Warning (EEW) systems have been in development since 1991. EEW systems detect the arrivals of the first seismic waves and send an alarm to users before the strong ground shaking hits them. The physics of earthquakes puts limits on the amount of warning times that users will receive from these systems, and it has been shown that users can expect warning times of 1 minute to a few seconds depending on the ground shaking thresholds that they set [Minson et al. 2018]. This isn't much time, but it can give users enough time to seek shelter, shut down delicate equipment, allow hospitals and first responders to prepare for shaking, and allow utilities companies to shut down gas and electrical lines that can rupture and lead to fires in large earthquakes.

There are several countries that are currently operating EEW networks, but the high costs associated with the installation, maintenance, and monitoring of these networks are a prohibitive factor for developing countries to be able to implement them. The cost of installing a continuous GNSS or seismic station that could be used in an EEW system can be around $25 thousand US-dollars, and for an EEW network to be effective hundreds of sensors must be deployed across a wide region. It is estimated that the Shake Alert EEW network that is currently being developed in California will cost $38 million to install, and $16.1 million annually to monitor and maintain [Strauss Allen 2016]. One way to reduce the cost of these networks would be if it were possible to take advantage of the unit price reductions that are created by the mass production of consumer grade electronics equipment. It has been demonstrated that the accelerometers and GNSS chips inside smartphones, coupled with a consumer grade external Satellite Based Augmentation System (SBAS) GNSS chip, would be able to detect the displacements from large Mw 6+earthquakes [Minson et al. 2015]. To take advantage of this capability we have developed a smartphone EEW system, and have installed two smartphone EEW networks in Chile and Costa Rica. The average cost of a sensor in our network is about $110 US-dollars, which will lower the costs associated with the installation and maintenance of an EEW network, and make EEW networks more affordable for developing countries.

The goal of this work is to analyze the performance of the sensors in our smartphone EEW network in various installation settings, and to analyze the possible warning times that our networks would be able to provide to users. To analyze the performance of the phone's internal GNSS chip and external UBLOX GNSS chip used in our system, 24 hours of data was collected in various building types ranging from a small wooden framed box to a poured reinforced concrete building. We also collected 24 hours of data collected with the sensor installed on a roof top.

All GNSS sensors show a drift in position over time because the noise in the observations is not white noise [Minson et al. 2015]. The impact of this drift on our GNSS sensors reported positions was calculated for each data set by taking the difference between each latitude, longitude, and altitude value and the next reported value at lags from 1 s to 10 minutes. The results of these difference tests were then implemented in a detection model to test if our sensors would be able to detect displacements from large earthquakes, and if so how accurately they are able to measure the displacements.

The small amount of warning time that EEW systems can produce even in best-case scenarios means that any latencies in the network will have a large impact on the effectiveness of the system. To analyze the latencies associated with the message transmission time, the latencies for 24 hours of data were collected. The recorded latency values were bootstrapped to create 10,000 new sample populations that were then used to get a 95% confidence interval of the mean latency associated with message transmission time for our network. After the mean network latency was calculated we looked at the distribution of the latencies across Costa Rica to test if the phones installed outside of city centers have larger latencies than the phones installed inside large cities and towns.

To analyze the effectiveness of our EEW network in Costa Rica we simulated the 2012 Nicoya earthquake to compare the results from a theoretical best-case EEW network, and our smartphone EEW network running the FinDer EEW algorithm. We followed the framework for a theoretical best-case EEW system, proposed by Minson et al. 2018, to calculate the warning times that users in Puntarenas, Liberia, and San Jose would have received for the ground shaking thresholds that were seen in those areas during the 2012 Nicoya earthquake. We then performed a field simulation of the 2012 Nicoya earthquake by programming the smartphones in our network to vibrate at times when the S-Waves would have reached each site. The resulting data was passed to the FinDer EEW algorithm to see how quickly it would be able to produce a warning.

Background

Earthquakes occur at plate tectonic boundaries where the interaction between the plates causes stress to build up in the Earth's mantle causing it to strain until it ruptures along fault lines. Ninety percent of the world's earthquakes occur along a ring of strike slip and convergent plate boundaries that circle the Pacific Ocean known as the ring of fire. These type of plate boundaries cause the largest earthquakes, and the largest earthquake that was ever measured occurred at the convergent plate boundary off the coast of Chile. When two tectonic plates converge one plate is subducted under the other plate. The friction of the two plates sliding over each other causes stress to build up which causes the mantle to strain and elastically and plastically deform. As the stress continues to build in the system the shear stresses on the interface between the plates will eventually exceed its shear strength and it will rupture along a fault line causing an earthquake. The initial rupture is called the earthquake nucleation, and occurs at a single point along the fault known as the hypocenter. After nucleation, the fault rupture evolves along the fault plane at a constant velocity which is typically around 3 km/s.

There are two seismic body waves that radiate out from the hypocenter. The first wave is called the P-wave, for primary wave, and it is a transverse wave that oscillates in the direction of propagation. The second wave is called the S-wave, for secondary wave, and it is a shear wave. The S-wave oscillates perpendicular to the direction of propagation, shearing the rock, and generally cause more damage that the P-waves. In addition to the P and S-waves, there are two types of surface waves generated by the interaction of the S-waves with the surface and near surface layers: Love Waves and Rayleigh Waves. These surface waves oscillate perpendicularly to the direction of propagation both horizontally and vertically, as well as back and forth along the axis of propagation. The surface waves decrease in amplitude with depth. The P-waves, S-waves, and surface waves all propagate at different velocities depending on depth and the medium through with they are travelling. On average, in the upper crust, the P-Waves propagate at a velocity of 5 km/s, and the S-Waves propagate at a velocity of 3.5 km/s. Surface waves propagate at slightly slower velocities than the S-Waves, and as the distance from the hypocenter increases the gap between the onset of the different wave fronts increases. However, for EEW the distances from the hypocenter that would receive a warning a relatively small, and it is assumed that the S-waves and the surface waves arrive at the same time.

EEW systems attempt to rapidly estimate the magnitude and location of an earthquake, and predict the level of ground shaking that will occur in the surrounding area (Figure 1). The first EEW network was established in Mexico in 1991. Since then EEW systems have been installed in the US, Japan, Turkey, Romania, China, Italy, and Taiwan (Shakealert.org). EEW systems work by detecting the seismic P-wave and S-wave signals that are generated during an earthquake, and the static offsets of the deformation caused by the fault slip, with a dense network of continuous GNSS receivers and accelerometers. The data from the network is streamed to a central location that is running one or several EEW algorithms that rapidly estimate the magnitude and location of the earthquake, and the predicted ground shaking in the surrounding region. If the predicted ground shaking for a given location is larger than a user specified threshold, a warning is sent out. Large earthquakes in coastal regions can also cause tsunamis that account for a large part of the fatalities associated with the event. The estimated magnitude and location output from EEW systems can be used as input into tsunami warning models as well.

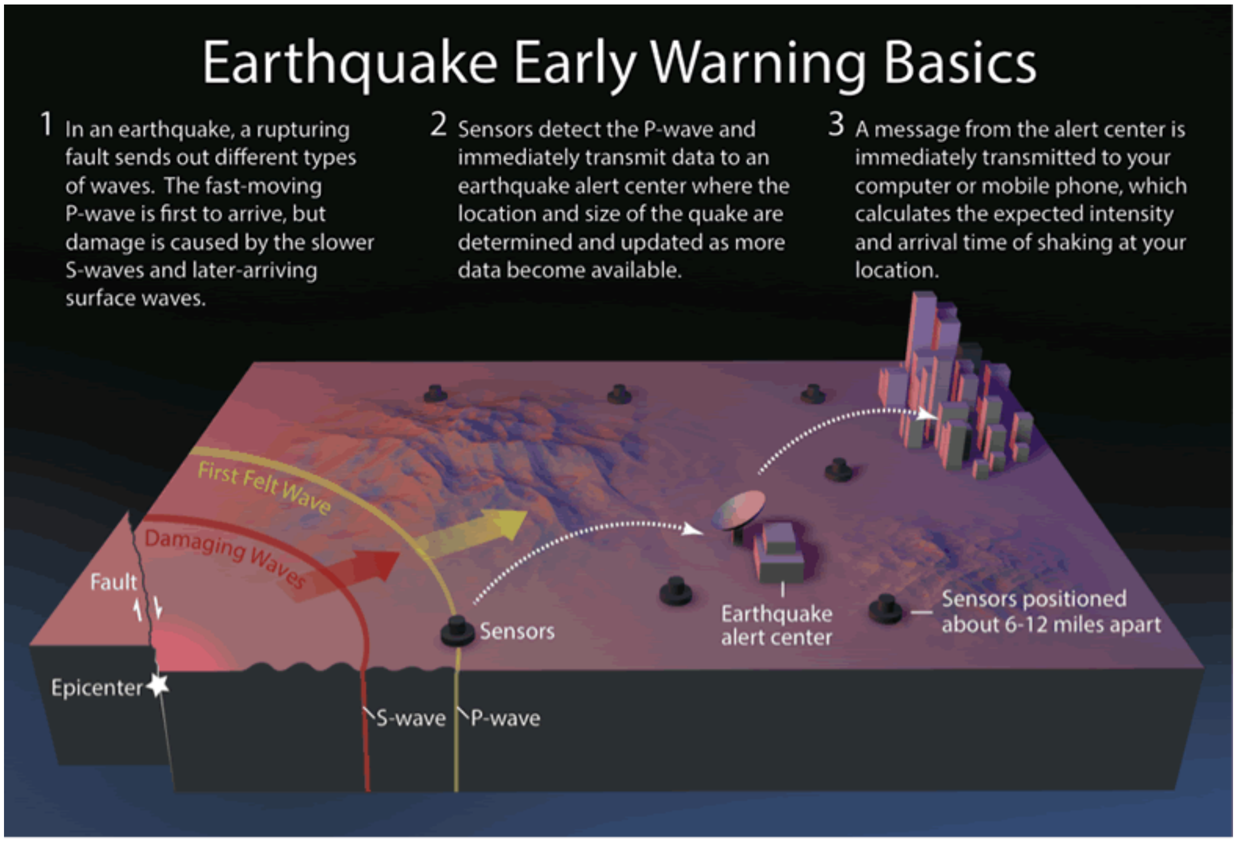

Earthquake Early Warning Basics
1 In an earthquake, a rupturing fault sends out different types of waves. The fast-moving P-wave is first to arrive, but damage is caused by the slower S-waves and later-arriving surface waves.
2 Sensors detect the P-wave and immediately transmit data to an earthquake alert center where the location and size of the quake are determined and updated as more data become available.
3 A message from the alert center is immediately transmitted to your computer or mobile phone, which calculates the expected intensity and arrival time of shaking at your location.
First Felt Wave
Damaging Waves
Fault
Sensors
Earthquake alert center
Sensors positioned about 6-12 miles apart
Epicenter
S-wave
P-wave

Figure 1 *Earthquake Early Warning Basics. After an earthquake P-waves and S-waves spread out from the epicenter. As they pass the sensors the sensors record the accelerations, and pass the data to a central location that is running one or several EEW algorithms that attempt to predict the location, magnitude, and shaking intensities in the surrounding area. If the predicted shaking intensities exceed user defined thresholds an alarm is sent out to users before the strong ground shaking arrives at their location.(Image from USGS Shake Alert web page.)*

In this paper we will be looking at the effectiveness of our EEW networks ability to deliver timely warnings to users at different ground acceleration thresholds. As earthquakes increase in intensity, the ground acceleration and potential damage that they cause will increase. Peak ground acceleration is measured in units of %g, which can be unintuitive for readers who have not worked with this unit before. Table 1 below shows the different levels of earthquakes using the Instrumental Intensity scale, shows the peak ground acceleration values that would be seen during that event, and describes the shaking that would be felt and the potential damage that would be caused in an earthquake of that size.

Instrumental Intensity	Acceleration (%g)	Velocity (cm/s)	Perceived Shaking	Potential Damage
I	< 0.05	< 0.1	Not Felt	None
II–III	0.3	0.1 – 1.1	Weak	None
IV	2.8	1.1 – 3.4	Light	None
V	6.2	3.4 – 8.1	Moderate	Very Light
VI	12	8.1 – 16	Strong	Light
VII	22	16 – 31	Very Strong	Moderate
VIII	40	31 – 60	Severe	Moderate to Heavy
IX	75	60 – 116	Violent	Heavy
X+	>139	> 116	Extreme	Very Heavy

Table 1 *This table shows the instrumental intensity scale linked with the corresponding ground acceleration as %g, and shows what the damage and perceived shaking would be for the different levels.*

We chose to install our smartphone EEW networks in Chile and Costa Rica due to the high earthquake hazard in those regions. Both Chile and Costa Rica have subduction zones a few 10s of kilometers off their coasts that are prone to large megathrust earthquakes. Chile is just east of the Nazca South American plate boundary. The Nazca plate is subducting below the South American plate at a rate of 6.7 cm/year [Eckerman et al. 2018]. The largest earthquake ever recorded was the Mw 9.6 1960 Valdivia earthquake that occurred off the coast of Valdivia, about 570 km to the south of Santiago. In 2010 the Mw 8.8 Maule earthquake struck about 100 km to the northwest of Concepcion and Talcahuano killing 525 people and causing an estimated $30 billion in damages or 17% of the country's GDP (Table 1)[USGS Maule report]. Costa Rica lies just east of the Central American and Cocos plate boundary. The Cocos plate is subducting under the Central American plate at a rate of 7.8 cm/year [Yue et al. 2013]. In 2012 a 7.6 Mw earthquake struck off the coast of the Guanacaste peninsula. There were two deaths that were attributed to the earthquake, about 30 houses were destroyed in the region, and the Monsenor Sanabaria Hospital in Puntarenas was significantly damaged causing 218 patients to evacuate [ticotimes.net]. The World Bank Group estimates that the annual average loss from earthquakes in Costa Rica is $407.5 million, or 0.82% of Costa Rica's GDP, and that the probable maximum loss for earthquakes over a 250-year period to be $9.7 billion or 20% of Costa Rica's GDP [World Bank Group].

Deaths	521 people
Missing Persons	56 people
Homes destroyed or damaged	~370,000 (11% of the total in the area)
Hospitals destroyed or damaged	73
Schools destroyed or damaged	3,049 schools, housing 1.25 million students
Bridges destroyed or damaged	221
Villages and rural and coastal communities affected	More than 900
Estimated Economic Loss	U.S. $30 billion. 17% of GDP

Table 2 Damage statistics for the 2010 Maule earthquake

Chile and Costa Rica Network Installations

Two different installation types have been implemented in our EEW networks in Chile and Costa Rica: roof top installations, and wall installations. A roof top installation consists of an Android smartphone with all of our apps installed, a 12V 5.2Ah Li-Po battery, a circuit board that contains the Bluetooth and UBLOX Neo-7 GNSS chips as well as a solar charging circuit, and a 25 or 30-watt solar panel (Image 1a). The phone, battery, and circuit board are housed in a small water tight electronics enclosure which is installed under a solar panel to keep the temperature down and minimize sun and weather damage. The total cost for the equipment in a roof top installation is about $250 depending on the phone that is used in the installation. A wall installation consists of an Android smartphone with our apps installed on it. The phone is then taped to a wall with double sided adhesive 3M VHB tape and plugged directly into a wall outlet. There is no external UBLOX GNSS chip in a wall type installation. Each installation in the Costa Rica network is a phone on the wall type of installation. An enclosure was added to the Costa Rica wall installations to house the phones, and to discourage theft of the devices. The phone is mounted to two pieces of aluminum square pipe using double sided 3M VHB tape, the aluminum stanchions are taped to the lid of the enclosure, and then the entire box is taped to the wall with double sided 3M VHB tape. (Image 1b-c). The total cost for a wall type installation is about $100 depending on the phone that is used in the installation. For more detailed information on how the two installations are installed see Appendix (**I)**. The maps of our two networks can be seen below (Figures 2-3). In the map of our Chile network roof style installations are denoted with blue triangles, and phone on the wall installations are denoted with green circles.

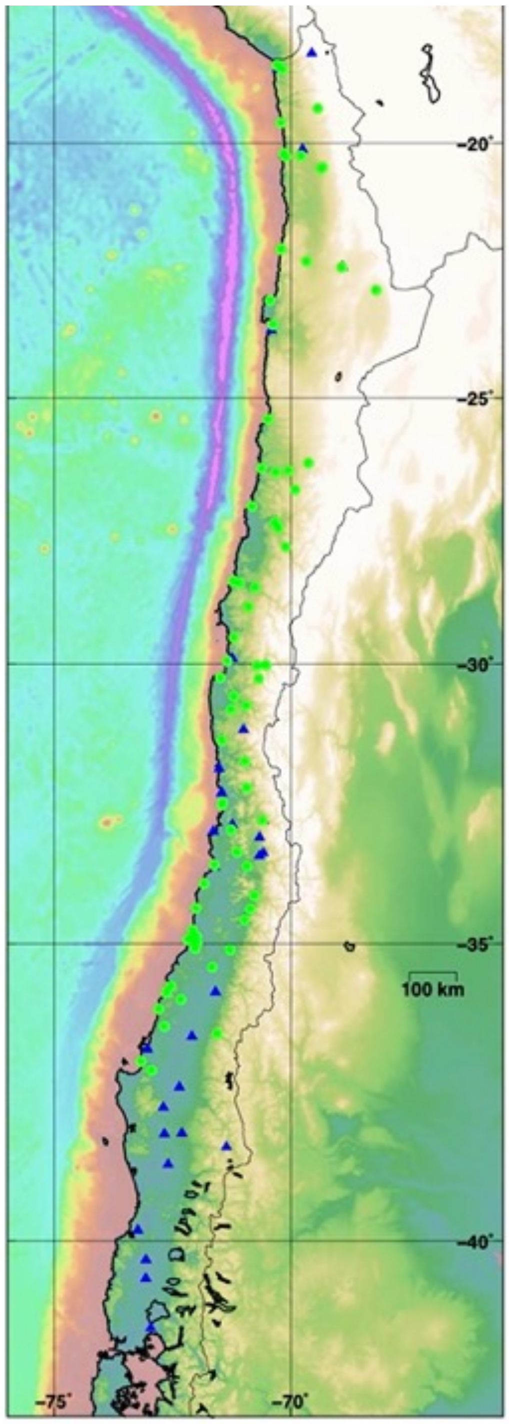


Figure 2 Map of our Chile Smartphone EEW network. The Blue triangles are roof top installations, and the green circles are phone on a wall type installation.

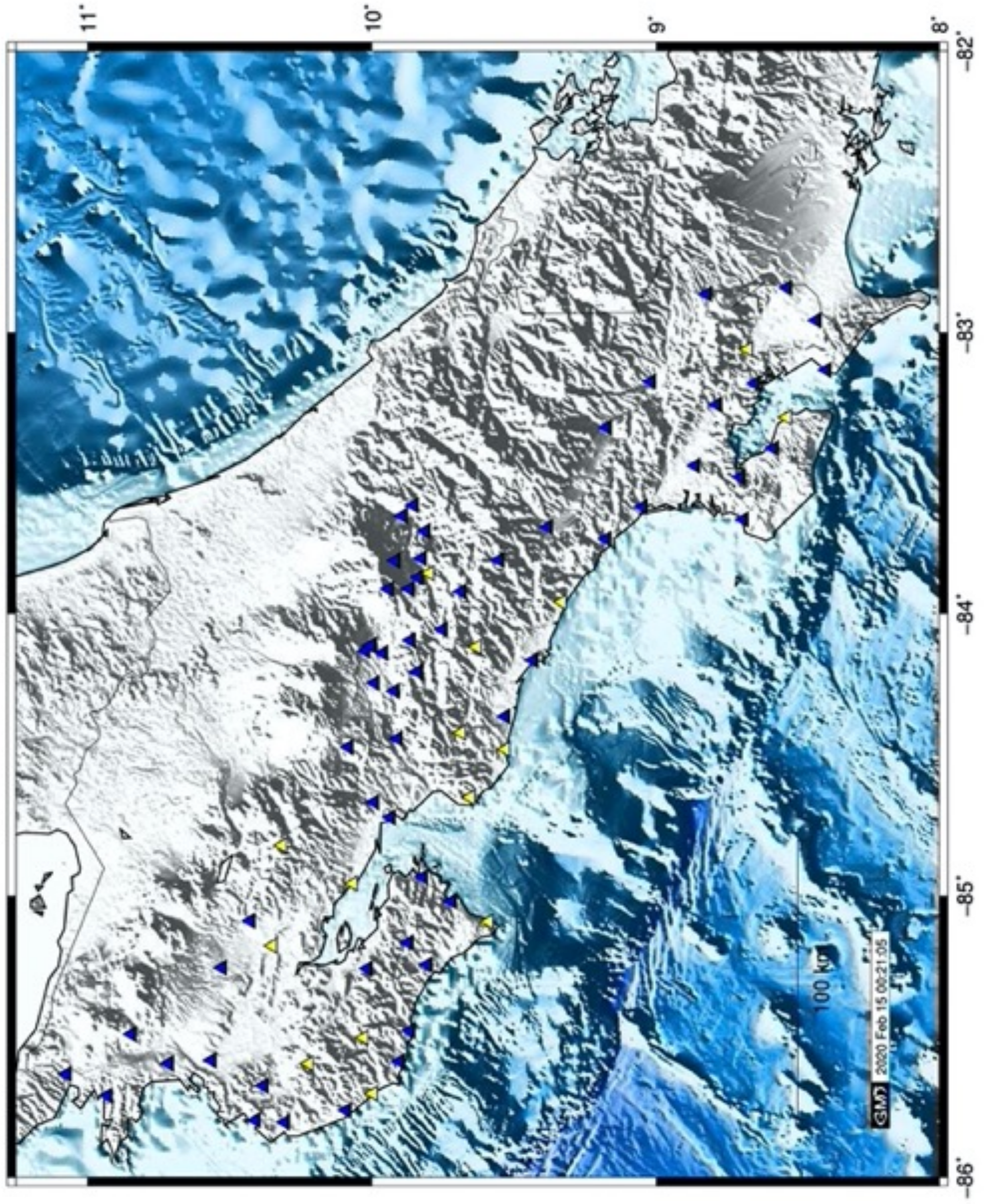


Figure 3 Map of our Costa Rica EEW network. The blue triangles are live stations, and the yellow triangles are stations that haven't sent data for the past 10 minutes (as of 2/14/2019).

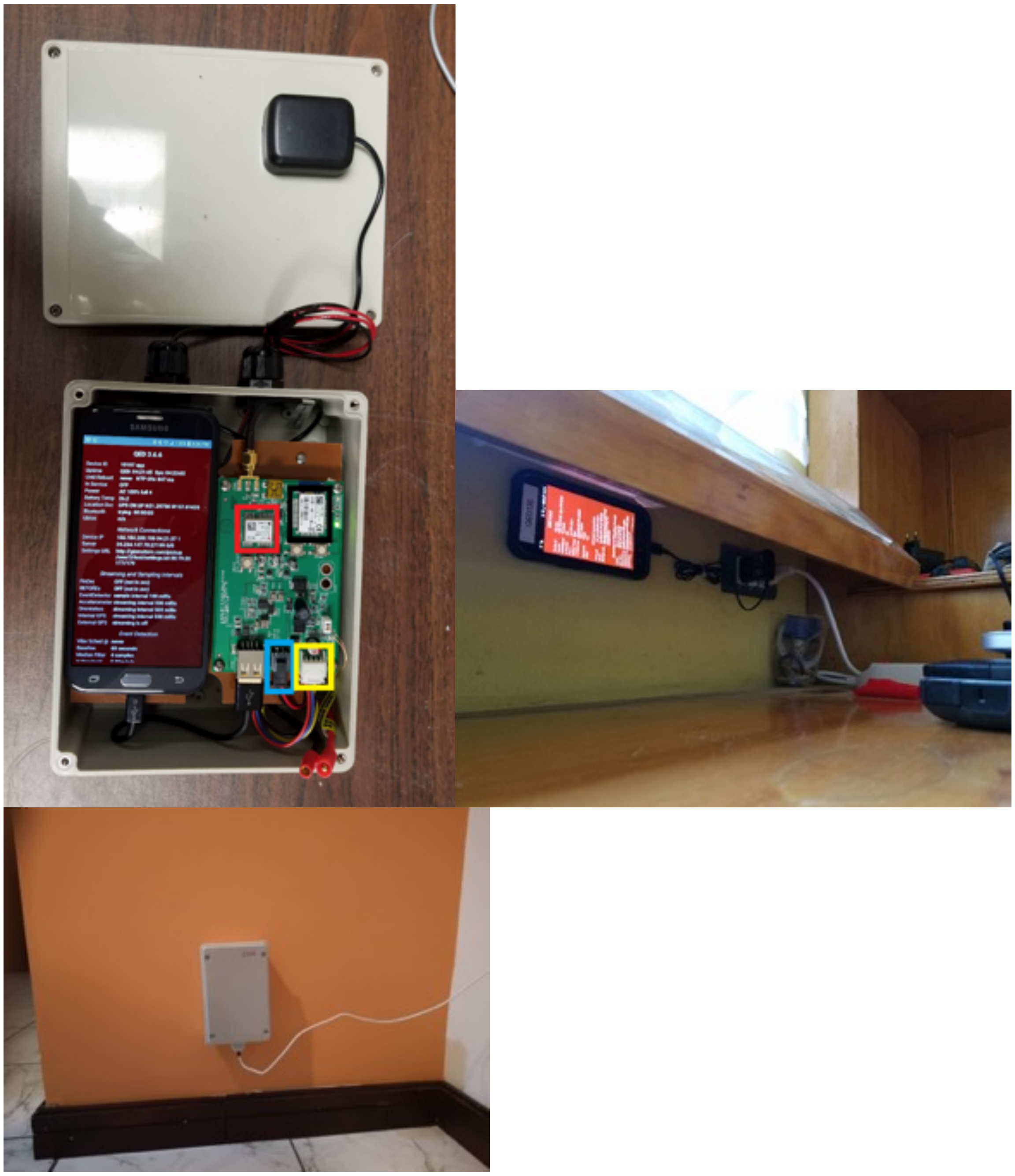


Image 1 **(a Top Left, b Top Right, c Bottom Left):** Images of the different installation types. Image 1a (top left) shows the equipment used in a roof top installation. The UBLOX external GNSS chip is circled in red, and the Bluetooth chip is circled in black. The LiPo battery is tapped to the bottom of the box below the phone and, and plugs into the charging circuit via the white balance charger connector circled in yellow. The solar panel, not shown here, is wired up via the red and black 18 AWG wires that are spooled up in the top right of the image, and connects to the charging circuit via the black connector circled in blue. Image 1b (top right) shows a wall

type installation. The phone is stuck to the wall with double sided 3M tape, and is plugged directly into an outlet. Many of the phones that were installed this way in Chile were stolen, and to prevent theft of the phones in Costa Rica an enclosure was added to the wall type installations Image 1c (bottom right). The phone is mounted on aluminum stanchions that are tapped to the lid of the enclosure with 3M tape. The enclosure is tapped to the wall with the same double-sided 3M adhesive tape, and the power supply for the phone is run into the enclosure through a cable gland in the bottom, and is plugged directly into a wall outlet. There is no UBLOX external GNSS in either type of wall mounted installation.

Our EEW network in Chile was installed over three campaigns from August 2016 to November 2017. During the first two campaigns, in September 2016 and January 2017, 52 stations were installed primarily at participating military bases. These installations were roof top installations. After the second round of installations, stations began to drop offline due to battery and temperature issues. There is no insulation in the enclosure that houses the LiPo battery, the charging circuit, and the phones, and the heat from the metal roof caused the phones and the LiPo batteries to overheat. In addition to the heat from the metal roofs there was also a problem with the charging circuit for the phones. The circuit was designed to sense the voltage of the phone's battery, and to stop charging when the battery was full. The sensor failed, and the charging circuit continued to charge the phone's battery even after it was full. This caused the batteries in the phones to swell either crushing the phone, or disconnecting the battery (Image 2). The batteries in the Samsung Ace 4 Neo phones that were installed tended to swell faster and to a larger size than the batteries in the Samsung J1 Ace phones, but the issue affected all the phones. Because of these issues it was decided to move the installations from the roof tops to inside the buildings. The third round of installations were moved to wall type installations, and 70 new sites were installed in November 2017. In total 122 stations were installed in our Chile EEW network. Our Chile EEW project has been completed, and data collection stopped on October 7th 2019.

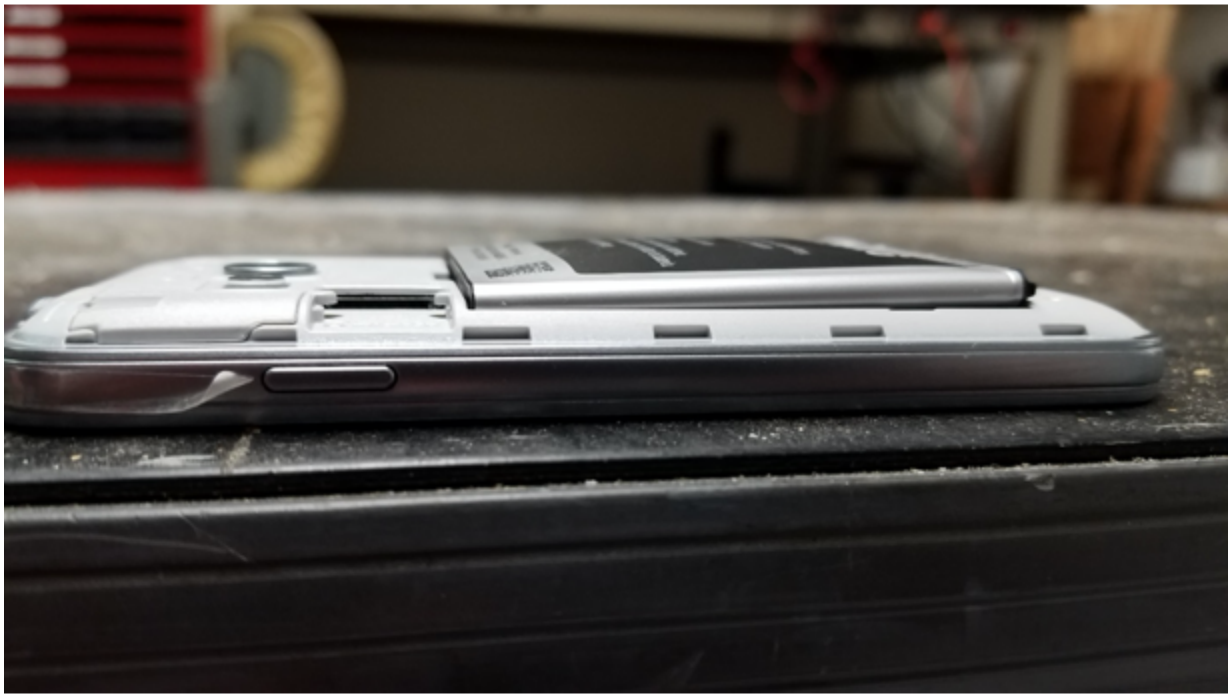


Image 2 A swollen battery in a Samsung Ace4 Neo phone. The battery is so swollen that it will not fit in its slot, and the battery pins can't reach the corresponding pins on the phone. When this happens the back of the phone can come off and the battery will disconnect itself, or if the back can't come off then the battery will swell into the phone and crush the phones circuit board.

To date 70 stations have been installed in Costa Rica. The installations have been conducted by our partners in the country, Marino Protti Quesada PhD and Floribeth Vega Solano of Costa Rica's Observatorio Vulcanológico y Sismológico de Costa Rica (OVSICORI). The installations began in May 2019. All of the installations in Costa Rica have been wall type installations with the phone housed in an enclosure to prevent theft.

Network Data Collection, Archiving, and Monitoring

The sensors in our Costa Rica and Chile EEW networks are Samsung Ace 4 Neo, J111M, and J337U smartphones running the Android Operating System. These smartphones were chosen because they have a 16-bit accelerometer, and there had already been work done in the android developer community to root the phones. Before our EEW apps are installed the phones need to be “rooted”, a process of unlocking root access for the phones, which allows restricted settings on the phones to be changed. Each smartphone has 4 apps installed for our EEW project: QED, QED Configurator, SSH Server Pro, and Keep Running, and an NTP client tomake sure that the phones are time-synced.

QED

The QED app is the main app of our Smartphone EEW system, and was developed by Chris Duncan at GIS Matters, and the USGS. The QED app collects data from the phone’s accelerometer, internal GNSS chip, and an external UBLOX NEO-7 GNSS chip if one is installed with the system. It also gathers general phone diagnostic data. The QED app splits the data from the different instruments and the phone diagnostic data into different message types. See Appendix(**II**) for more information on the different QED message types. Default sample rates of 1Hz and 10Hz are set for both of the GNSS chips and the accelerometer respectively. The QED app assigns a unique ID to each phone using the phones IMEI number, and each QED ID has a specific settings file that is stored on the Amazon Web Server. These settings files can be edited to change the frequency of any message that the QED app outputs without having to directly access the phones themselves. If an external UBLOX GNSS chip is included in the installation the phone connects to the UBLOX chip via Bluetooth.

QED Configurator

The QED Configurator app is used to pair the phone with the Bluetooth chip so it can collect the external GNSS data.

SSH Server Pro

The SSH Server Pro app allows us to remotely log into the phones with a secure connection to pick up error logs that are saved on the phones SD card, to reboot the phones, and to install updated versions of the QED app.

Keep Running

The Keep Running app periodically checks that all of the necessary apps are running, and restarts them if they are not. See Appendix(**III**) for a more detailed discussion of the Apps.

MQTT

The data from our smartphone EEW network is picked up by several EEW centers using MQTT. MQTT is an IoT protocol that uses a publish/subscribe philosophy. The instruments that we have in the field "publish" their data as different "topics" to a central server running a MQTT Broker. The topics in our project include the network, message type, and phone id. Individual MQTT Clients can connect to the central MQTT Broker to "subscribe" to different published topics. The broker acts as a middle man between the data stream and the client. This means that the sensors don't have to push their data to each EEW center, which cuts down on the data costs.

Data Transport and Archiving

The QED app outputs a binary UDP stream through the smartphone's cell network connection to a central Amazon Web Server that is running a MQTT broker. The binary stream is translated to ASCII by the UDP receiver on our Amazon Web Server, and is then passed onto the MQTT broker. The ascii MQTT stream can be picked up in real time by any server running a MQTT client. The MQTT client can request data from all the projects, data from a specific project, data of a specific message type, or data from a specific phone depending on the MQTT client's subscription request. In our data archiving process, all of the data for each project are collected independently, and written to a project specific log file. The Unix logrotate utility is used to rename, compress, and rotate each project's log file every hour. A Unix timestamp is appended to the log's filename which is later converted to a human readable timestamp, and is used to archive the file. The compressed hourly log files are moved to the archive every 24 hours. The archive directories use the following convention: ~/Projects/Smartphone_EEW/archive/"project_name"/YYYY/MM/DD/"project"_YYYY_MM_DD _HH.csv.gz. The data for the Chile Project, the Costa Rica project, and the development network are archived at both the PGF Lab at UH Manoa and at the USGS facilities in Menlo Park.

Data Flow Monitoring and Analysis

In addition to archiving the data, several scripts are run to monitor and map network latencies, and to make sure that the data collection and archiving process is live (Figure 4). The last_message.R script reads the live data stream, finds the last message sent by each phone in the different networks, and calculates the time between when that message was sent and the current time. These times are used to create an interactive network status map in our R Shiny Web app that is hosted on our server at UH Manoa. The Shiny Web app uses R and the leaflet package to create an interactive map of the network. Stations that have reported data in the past 10 seconds are marked as blue, any station with a last sent message older than 10 seconds is marked orange and is considered to be down. Users can enter a different threshold time than the default 10 seconds by changing the "Threshold Latency" setting in a drop-down menu. A histogram of the past hour of latencies for all the phones that are inside the map view is plotted in a drop-down menu to give users an idea of the overall health of the network. The age_alert.csh script is run every 5 minutes to make sure that the log files are being populated with the live data, and not error messages from the MQTT client. If there are any problems with the data logging it sends an email alert notifying us that there is an issue with the live log file.

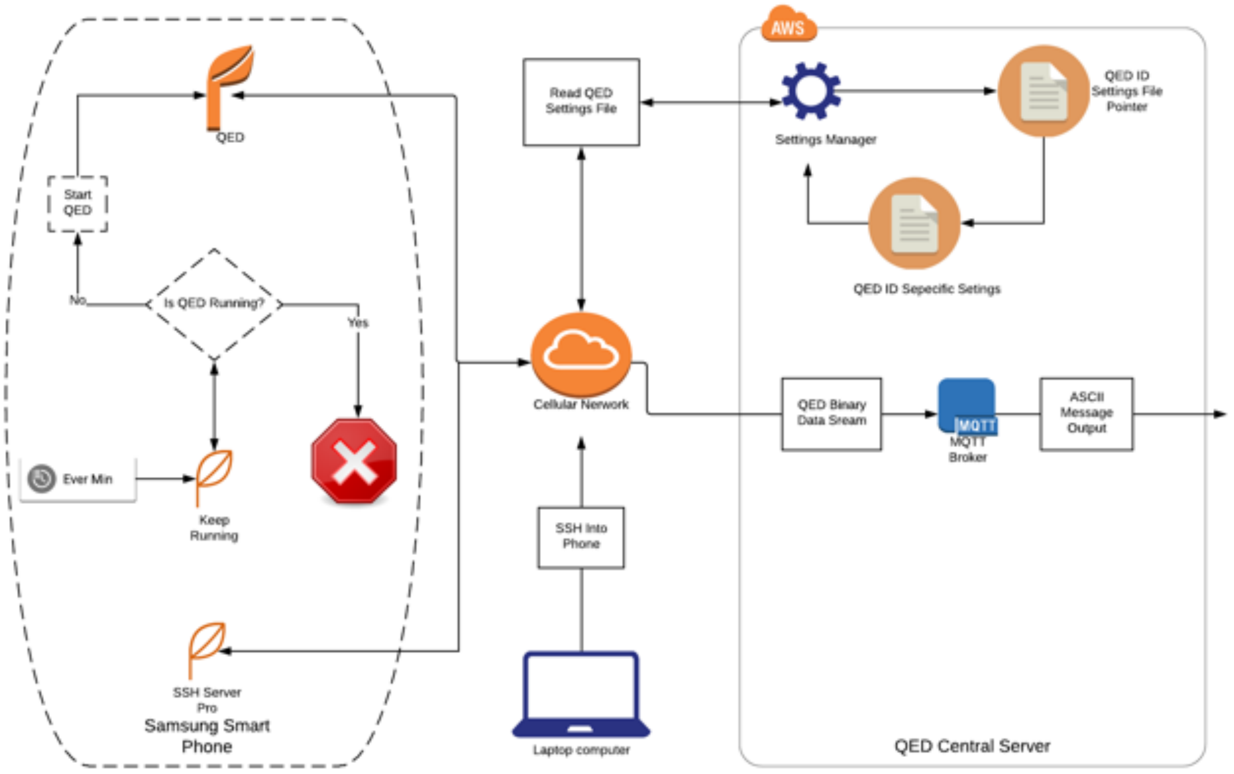


Figure 4 Layout of the phone and QED central server. The Keep running app checks if QED is currently alive every minute. If it is it does nothing. If it isn't it starts QED. Remote computers can SSH into the smartphones via the SSH Server Pro app to collect QED error logs which are stored on the phones SD card. QED connects to the QED Central Server to collect its settings file at specified intervals. It reads a pointer file which directs it to its specific settings file, and if there are any changes in the specific settings file QED implements the changes. QED pushes the Binary UDP stream of all the messages to the MQTT broker which is running on the central server. The server converts the Binary stream to ASCII outputs the converted data stream to MQTT Clients upon request.

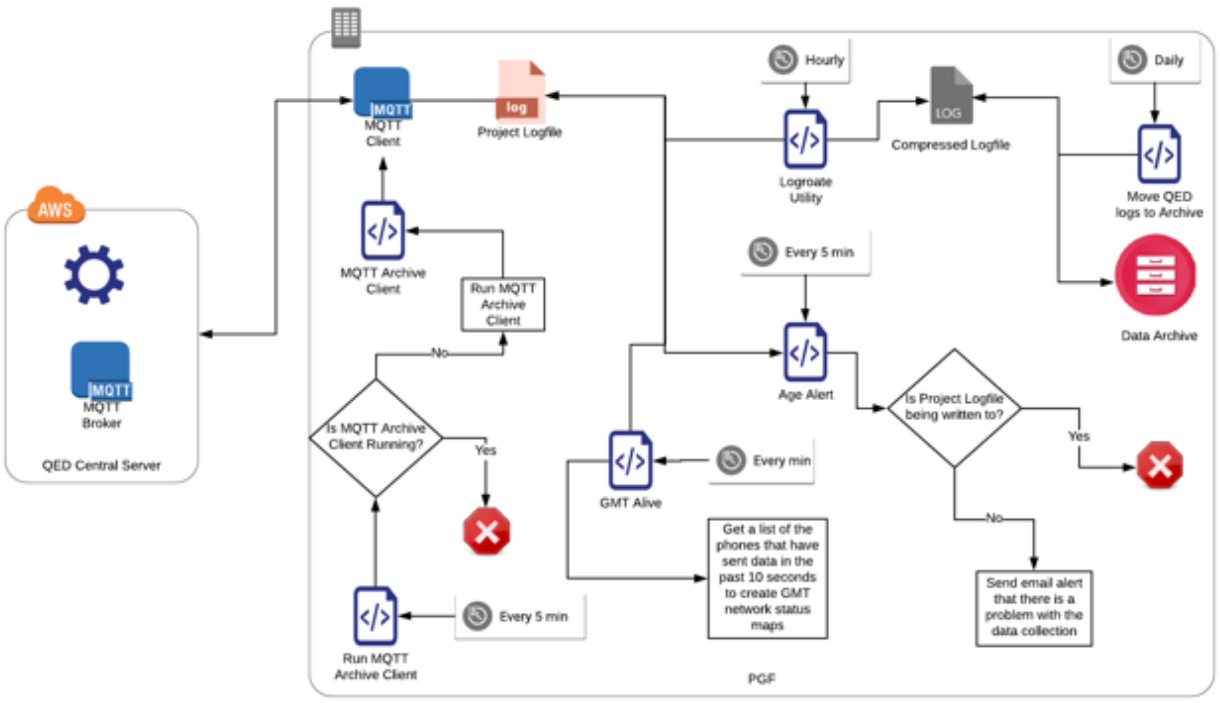


Figure 5 Overview of the PGF archiving and network monitoring scripts. The ASCII data stream is picked up with the MQTT client. The instructions for the MQTT client are sent with the Archive Client scripts which should only need to be called once. Each project has its own Archive Client script which outputs the MQTT ASCII data stream to a project specific log file. The different Run Archive Client scripts check that each project's Archive Client scripts are running every 5 minutes. If they have stopped running for any reason it starts them back up. The Unix logrotate utility rotates and zips the Project Log files every hour. The Move QED Logs to Archive script is run daily, and takes the past 24 hours of the Compressed Log files and moves them to their proper directories in the data archive. The Age Alert script runs every 5 minutes, and checks that the Project Log files are actively being written too. If nothing has been written to the Project Log files in the past minute an alert that there is a problem with the data collection is emailed. The GMT Alive script checks the Project Log files every minute and creates a list of phones that have sent data in the past 10 seconds. This list is used to create the Network status maps in the R shiny app which is hosted on the pgftsunami server at: 128.171.151.105:3838**.**

Data Analysis: Assessing the Quality of Consumer Grade GNSS Sensors

Rapid and accurate estimates of an earthquake's magnitude is one of the most important factors in EEW, and tsunami early warning. EEW algorithms use Ground Motion Prediction Equations (GMPE) that are primarily a function of magnitude and distance to the fault to predict the shaking that will be felt at a user's location. EEW algorithms are split into two categories, point source algorithms, and finite fault algorithms. Point source algorithms can saturate during large magnitude earthquakes and under predict both the magnitude and the ground shaking in the surrounding region. During the 2011 Tohoku earthquake Japan's point source EEW algorithm underestimated the initial magnitude of the earthquake by 0.9 (earthquake magnitudes are in a log scale, so a 1 unit magnitude increase equates to an 33x increase in earthquake power) , and the final magnitude by 1.5 units of magnitude [Ruhl el al. 2019]. Magnitude estimates and ground shaking predictions can be improved by using a finite fault model; however, these algorithms have traditionally taken a longer time to output solutions. One way to improve source estimation is through measurements of static displacements generated by an earthquake. It has been shown that combining the measured coseismic displacement from an earthquake with empirical relations between peak displacement and earthquake magnitudes developed by Gutenberg 1945 can yield quick and accurate fault length and earthquake magnitude estimates that do not saturate, and hold up to magnitude 9.0 events [Fang et al 2013].

Displacement can be calculated by integrating the measured velocities from seismometers, but these instruments tend to clip in large earthquakes or when they are near the fault. This makes them ill-suited for use in EEW networks. Accelerometers do not clip in large earthquakes, and displacements can be calculated by doing a double integration over the accelerations measured by these instruments. However, any tilt, rotation, or drift that the accelerometer experiences will add errors to the calculated displacement, making it difficult to use these instruments to constrain the magnitude of earthquakes where they are likely to experience tilting and rotation.

Historically ground displacement measurements have been acquired using expensive, scientific-grade GNSS receivers, however it has been shown [Minson et al. 2015] that the lower-grade GNSS chips inside cell phones, and cheap external GNSS chips using SBAS should be able to detect ground motions associated with Mw 8+ and Mw 6+ earthquakes respectively. In this section we will analyze the performance of the external UBLOX GNSS chip as well as the internal GNSS chip to determine what level of displacements they are able to accurately detect and measure.

Effect of Buildings on Internal and External GNSS Position Accuracy

Difference Test

In their 2010 paper *Indoor Positioning Using GPS Revisited* Kjaergaard et al. looked at the ability of the UBLOX GNSS chip and the internal GPS chip of a Nokia Phone to provide usable indoor positioning. They found that both the UBLOX GNSS chip and the Nokia’s internal GPS chip were able to get positions inside of wood and brick buildings. The UBLOX GNSS chip and the Nokia’s internal GPS were able to get positions in buildings with reinforced concrete, but their performance was much more inconsistent. They also found that the strength of the

GNSS signal depends on both the building material, and the distance that it travels after penetrating the building [Kjaergaard et al. 2010]. This suggests that we might still be able to use the UBLOX and internal GPS data while our EEW systems are installed inside wooden or brick buildings.

To analyze how installing our cell phone EEW systems inside of buildings would affect the accuracy of their internal and external GNSS chips, data was collected in three building types ranging from a plywood box to a poured concrete office building. Full systems, with the UBLOX external GNSS chip, were installed in the windows of the Pacific Ocean Science and Technology (POST) Building on the UH Manoa campus (Image 3), a residential home, a box that was designed to attenuate GNSS signal (Image 4), and on the roof of the HIG building at the UH Manoa Campus. Data was collected for 24 hours in each of the different building types in windows facing each of the four cardinal directions. The box that was built for the test had three of its walls, and its roof, made from plywood wrapped in several layers of aluminum foil and a fine steel mesh, 1 inch of concrete wall board, and 2-4 layers of corrugated Zinc-Aluminum siding, and its fourth wall was a single pane window.

Image 3 The Pacific Ocean Science and Technology (POST) building on UH Manoa Campus. It is poured reinforced concrete with reflective film on the windows. The external UBLOX GNSS receiver did not always obtain a position while it was installed in the North, South, and East facing windows. It was never able to obtain a position while it was installed in a West Facing window.

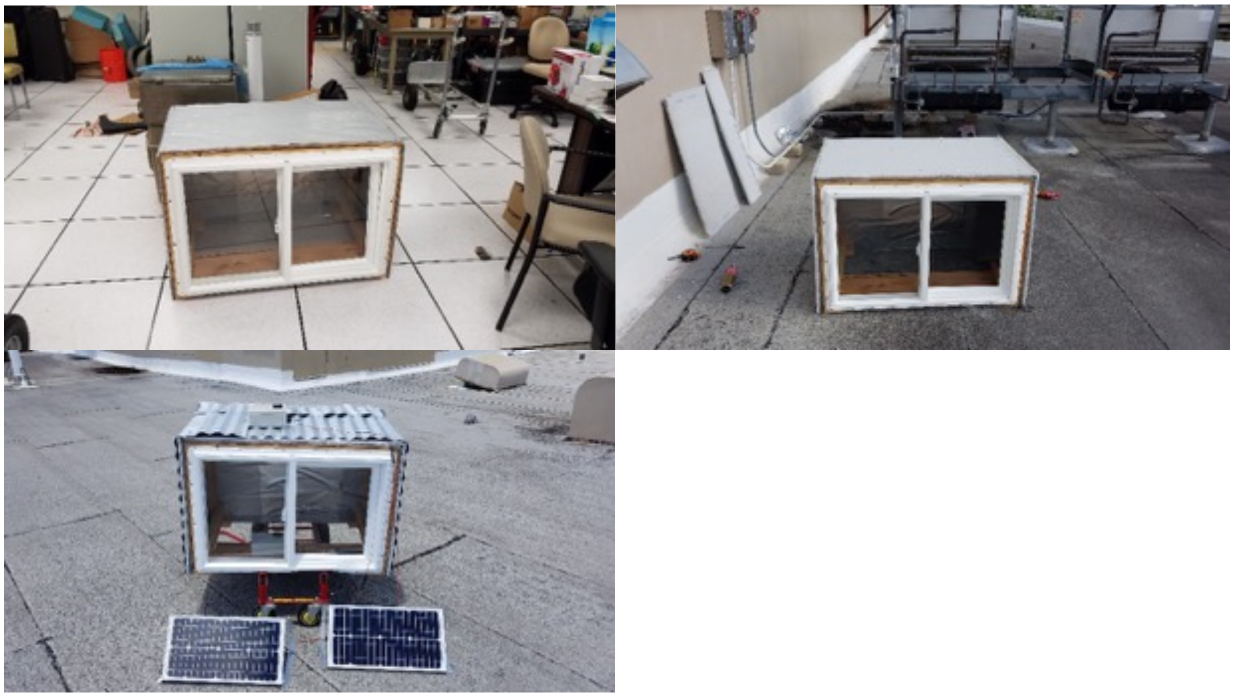


Image 4 The plywood box that was used to test signal attenuation, and the performance of the internal and external GNSS receivers of our EEW system. The box is framed with 2”x4”s and has walls of plywood. The plywood is wrapped in 5 layers of aluminum foil, and 2 layers of fine steel mesh (top left). Two ½” sheets of Durock concrete wallboard were screwed and glued to the plywood walls (top right). Three layers of corrugated Zinclum were screwed into walls, and another two layers were added to the corners (bottom left).

The data from the directional window tests was taken and processed using R to analyze the effect of the different construction types on the GNSS chips. The reported latitude and longitude values from the QED X (external GNSS data)and I (internal GNSS data) messages were converted from decimal degrees to UTM coordinates using the sp package in R. After the reported latitude and longitudes were converted to UTM coordinates, the difference between each reported latitude, longitude, and altitude value, and the next reported value with lags of 1 to 600 seconds were taken to create 600 vectors of differences for each aspect (latitude, longitude, and altitude). Each of the 600 difference vectors for each aspect were then bootstrapped. Samples were taken with replacement from each difference vector to create

1,000 new sample populations. The mean, the standard deviation from the mean, and two standard deviations from the mean were then calculated for each of the new 1,000 sample populations. Each calculated parameter (mean, SD, 2SD) was then appended to their own vectors. The mean of the resulting vectors were calculated for each parameter, and were plotted over 10 second and 10 minutes to see how the different buildings affect the GNSS drift over time (see Appendix IV for the code).

Latitude Differences

Control Sensor

The control sensor's UBLOX GNSS chip showed a mean change in latitude of 0 m over 10 minutes for all four days that data was collected. While the window was facing north 95% of the differences in latitude it detected were between +/-0.66m at 10 seconds, and were within +/- 5.81m at 10 minutes. While the window was facing south 95% of the differences in latitude it detected were within+/-0.28m at 10 seconds, and were within +/-2.49m at 10 minutes. While the window was facing east 95% of the differences in latitude it detected were within +/-0.44m at 10 seconds, and were within +/-3.86m at 10 minutes. While the window was facing west 95% of the differences in latitude it detected were within +/-0.32m at 10 seconds, and were within +/- 2.17m at 10 minutes. (Figures 6-7)

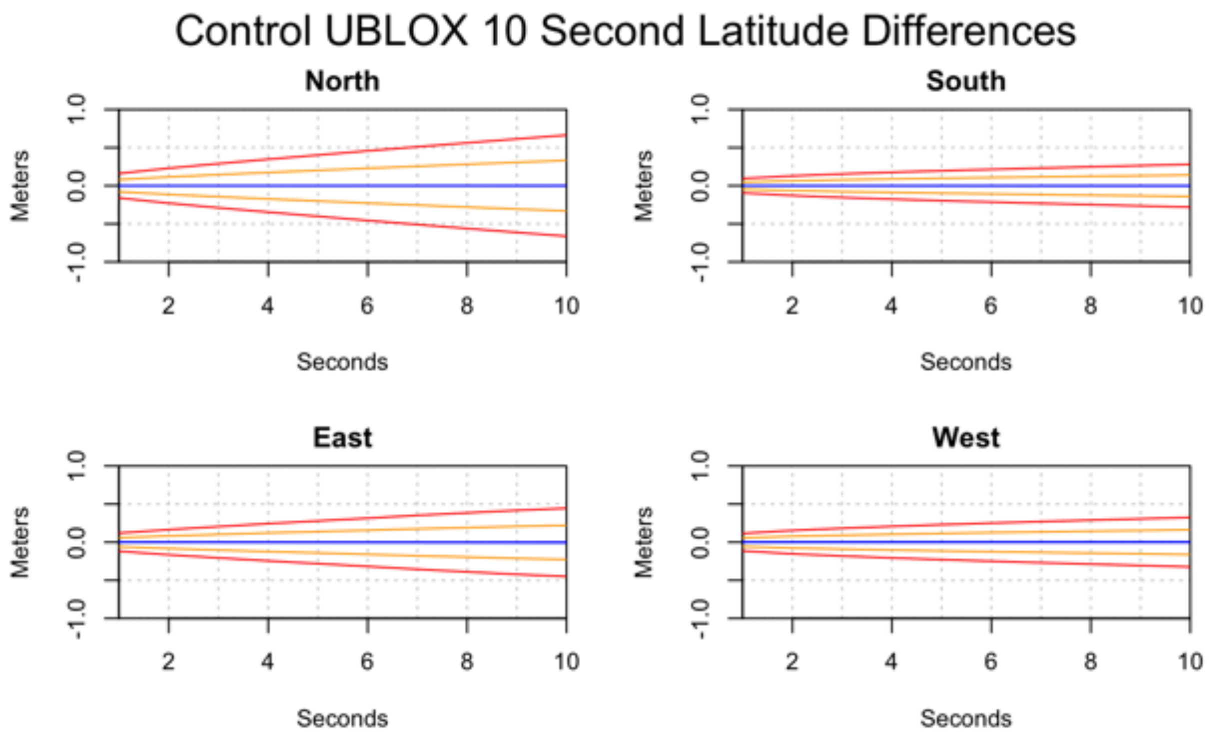


Figure 6 Latitude difference plots over 10 seconds for the UBLOX GNSS chip installed on the roof. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

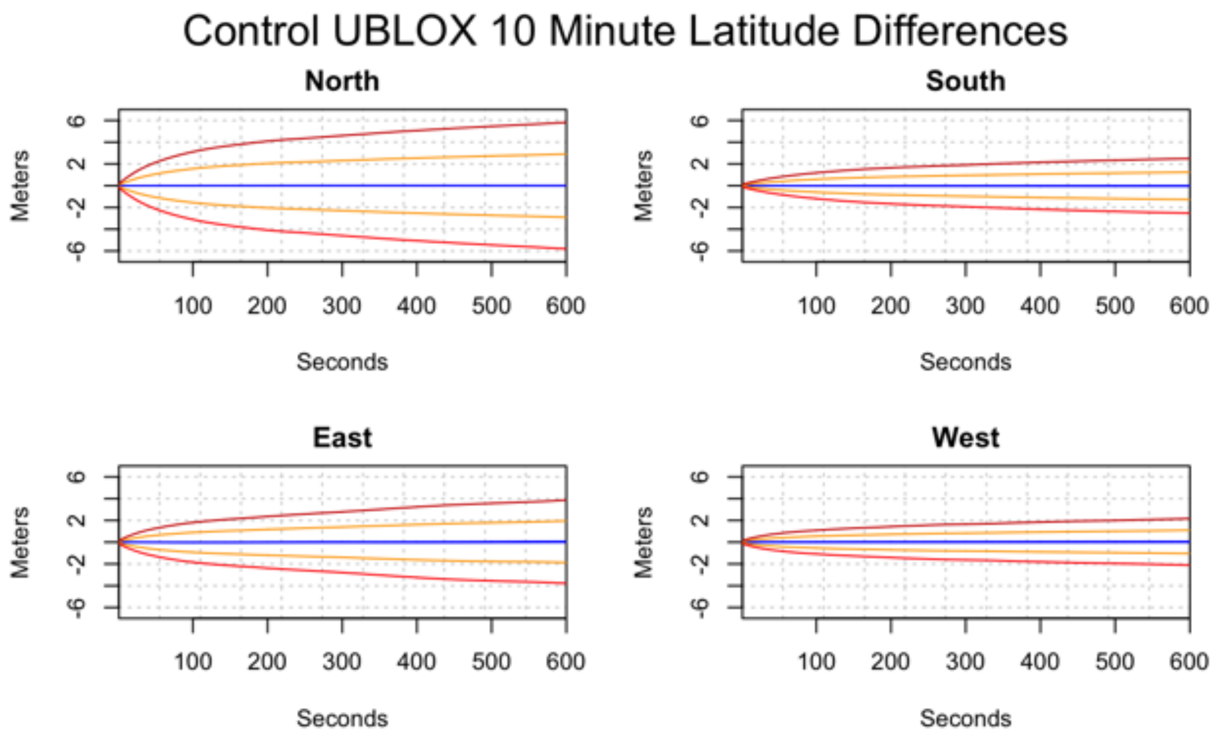


Figure 6 Latitude difference plots over 10 minutes for the UBLOX GNSS chip installed on the roof. *The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.*

UBLOX GNSS Chip in the Box

The UBLOX GNSS chip of the sensor installed inside the box recorded a mean difference in latitude of 0m over 10 minutes for all four days that data was collected. While the window was facing north 95% of the differences in latitude it detected were within +/-0.76m at 10 seconds, and were within +/-8.38m at 10 minutes. While the window was facing south 95% of the differences in latitude it detected were within +/-0.55m at 10 seconds, and were within +/-5.49m at 10 minutes. While the window was facing east 95% of the differences in latitude it detected were within +/-0.49m at 10 seconds, and were within +/-5.24m at 10 minutes. While the window was facing west 95% of the data were within +/-0.45m at 10 seconds, and were within +/- 4.5m at 10 minutes (Figures 8-9).

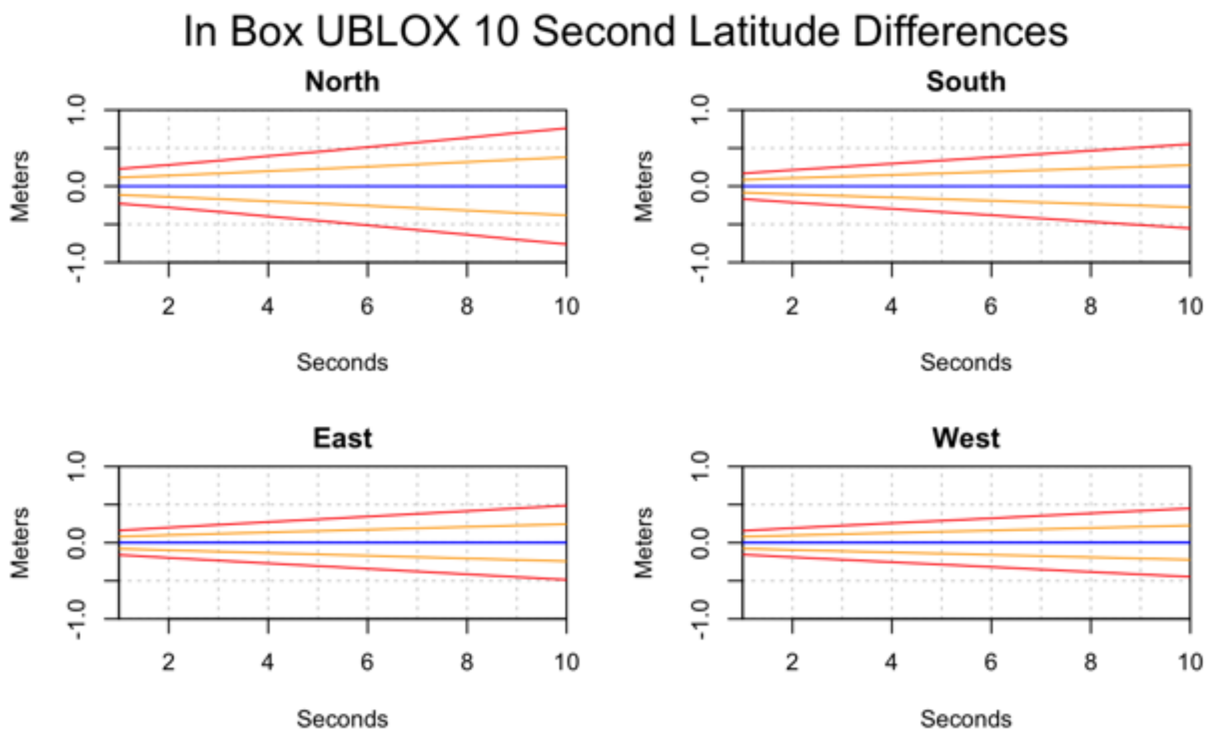


Figure 7 Latitude difference plots over 10 seconds for the UBLOX GNSS chip installed inside the box. The blue line shows the mean difference, the orange lines shows one sd from the mean, and the red lines show 2sd from the mean.

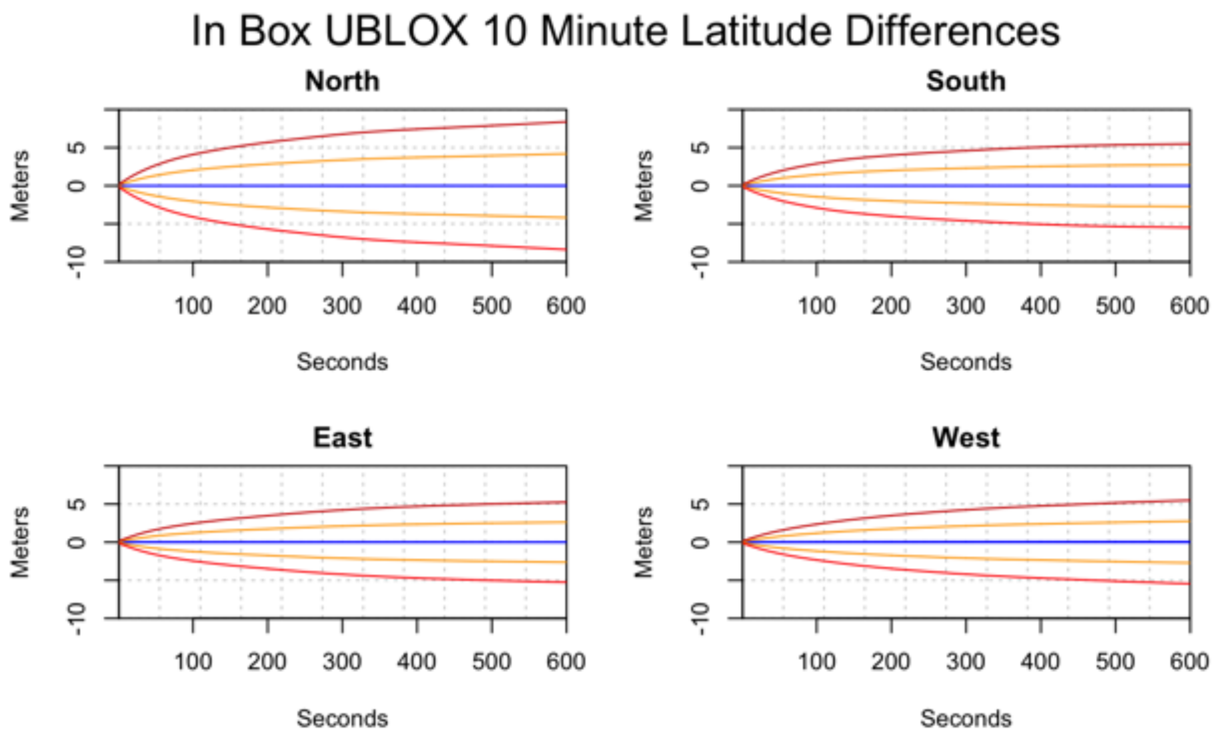


Figure 8 *Latitude difference plots over 10 minutes for the UBLOX GNSS chip installed inside the box. The blue line shows the mean difference, the orange lines shows one sd from the mean, and the red lines show 2sd from the mean.*

Internal GNSS Chip in the Box

The phone's internal GNSS chip installed in the box recorded a mean difference in latitude of 0m over 10 minutes. While the window was facing north 95% of the differences in latitude that it recorded were within +/- 1.20m at 10 seconds, and were within +/-8.21m at 10 minutes. While the window was facing south 95% of the differences in latitude that it recorded were within +/- 1.23m at 10 seconds and were within +/- 6.75m at 10 minutes. While the window as facing east 95% of the differences in latitude that it recorded were within +/- 1.23m at 10 seconds, and were within +/- 6.75m at 10 minutes. While the window was facing west

95% of the differences in latitude that it recorded were within +/- 1.23m at 10 seconds, and were within 6.75m at 10 minutes (Figures 10-11).

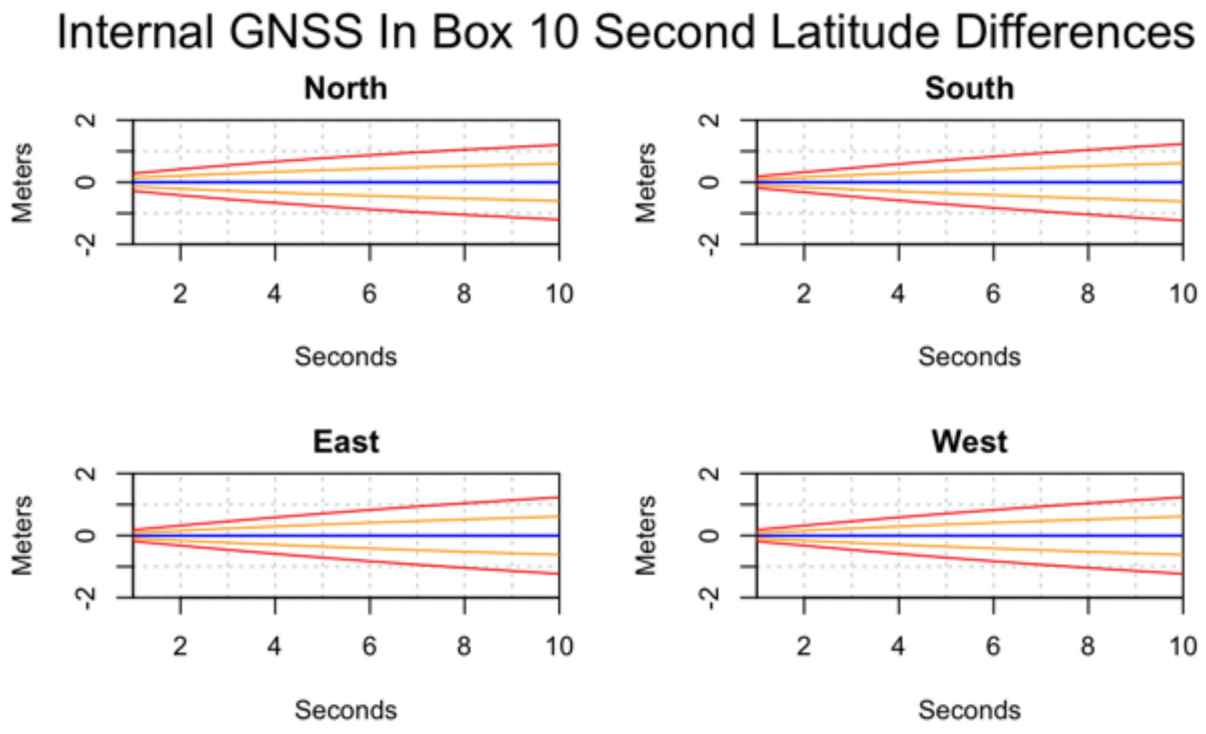


Figure 9 Latitude difference plots over 10 seconds for the phone's internal GNSS chip installed inside the box. The blue line shows the mean difference, the orange lines shows one sd from the mean, and the red lines show 2sd from the mean.

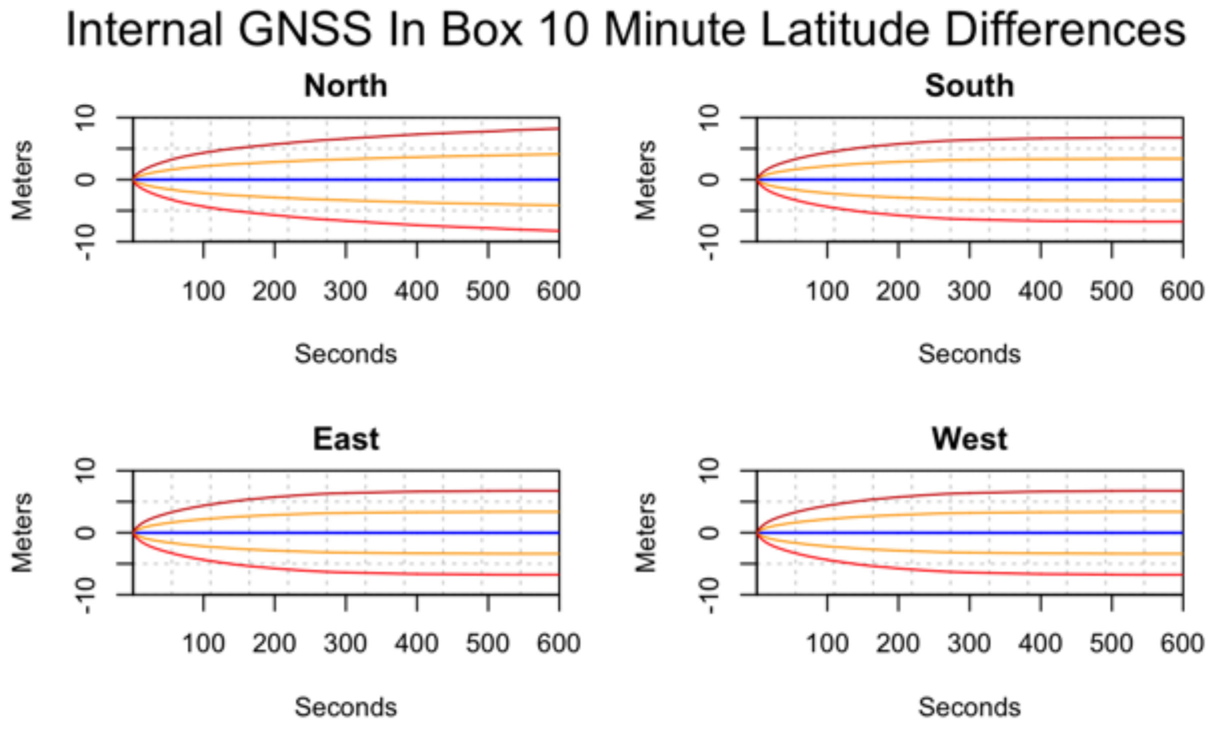


Figure 10 Latitude difference plots over 10 minutes for the phone's internal GNSS chip installed inside the box. The blue line shows the mean difference, the orange lines shows one sd from the mean, and the red lines show 2sd from the mean.

UBLOX GNSS Chip in Residential Home

The UBLOX GNSS chip of the sensor that was installed in the residential home recorded a mean difference in latitude of 0m over 10 minutes. The sensor was only installed in one window of the house which faced due south. While the sensor was installed in the window of the residential home 95% of the differences in latitude the UBLOX GNSS chip detected were within +/- 1.51 m at 10 seconds, and were within 8.61 m at 10 minutes (Figure 12).

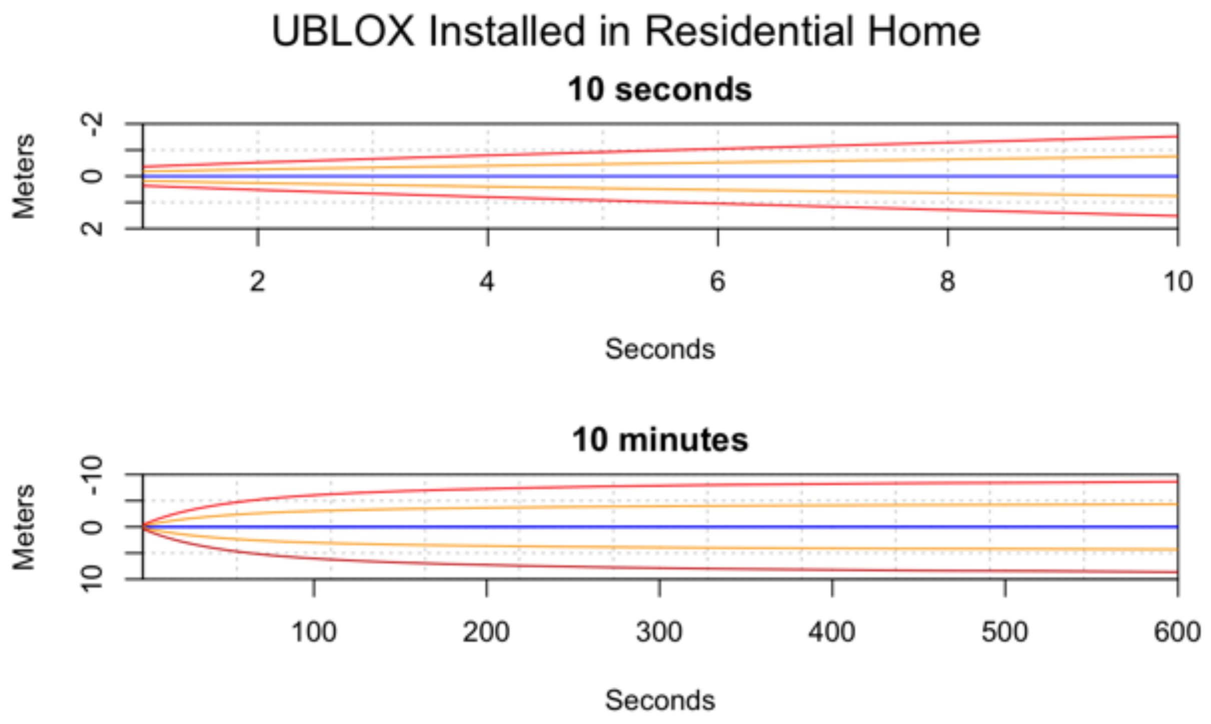


Figure 11 Latitude difference plots over 10 seconds and 10 minutes for UBLOX GNSS chip installed in a residential home. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

UBLOX GNSS Chip in POST

The UBLOX GNSS chip of the sensor installed in the windows of the POST building recorded a mean difference in latitude of 0m over 10 minutes. While the sensor was installed in the north facing window 95% of the differences in latitude its UBLOX GNSS chip detected were within +/-25.58 m at 10 seconds, and were within +/- 88.35 m at 10 minutes. While the sensor was installed in the south facing window 95% of the differences in latitude its UBLOX GNSS chip detected were within +/- 11.45 m at 10 seconds, were within +/- 38.55 m at 10 minutes. While the sensor was installed in the east facing window 95% of the differences in latitude its UBLOX GNSS chip detected were within +/-10.26m at 10 seconds, and were within +/- 37.73m at 10 minutes. Two attempts to collect data from a window facing west in the POST building were made, but the sensor's UBLOX GNSS chip was never able to obtain a position Figures 13-14).

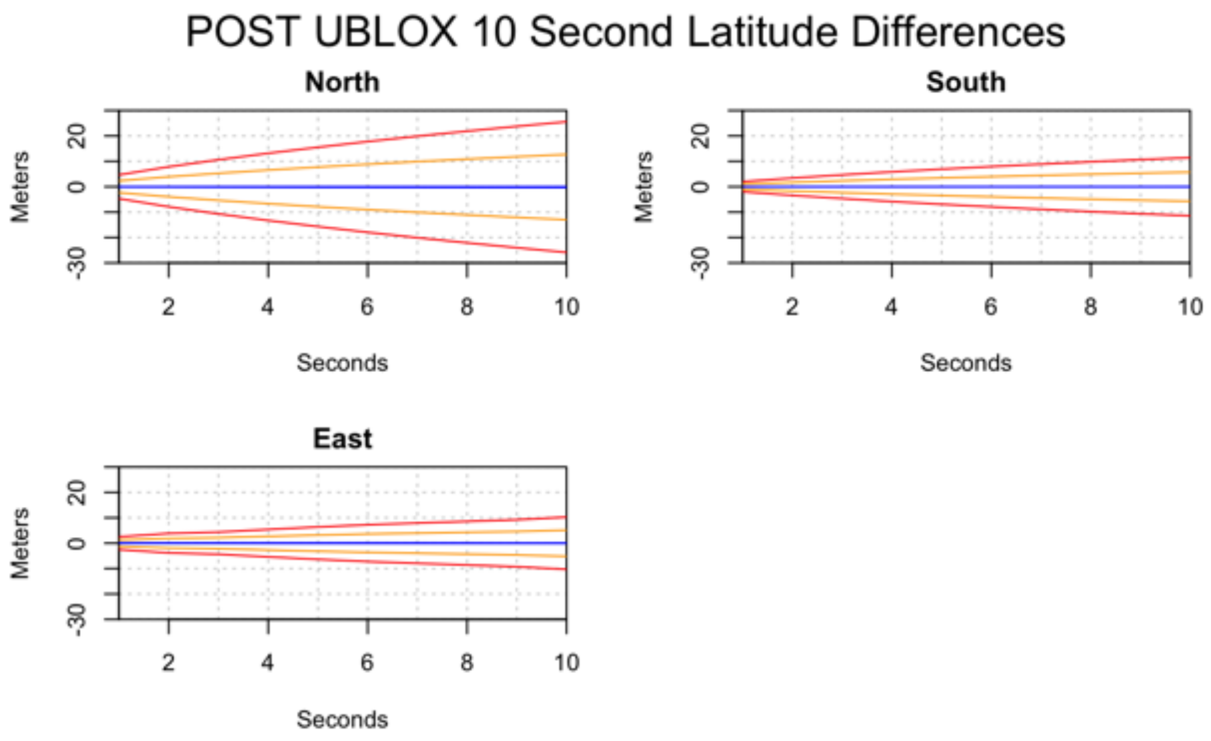


__Figure 12__ Latitude difference plots over 10 seconds for the UBLOX GNSS chip installed in the POST building. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

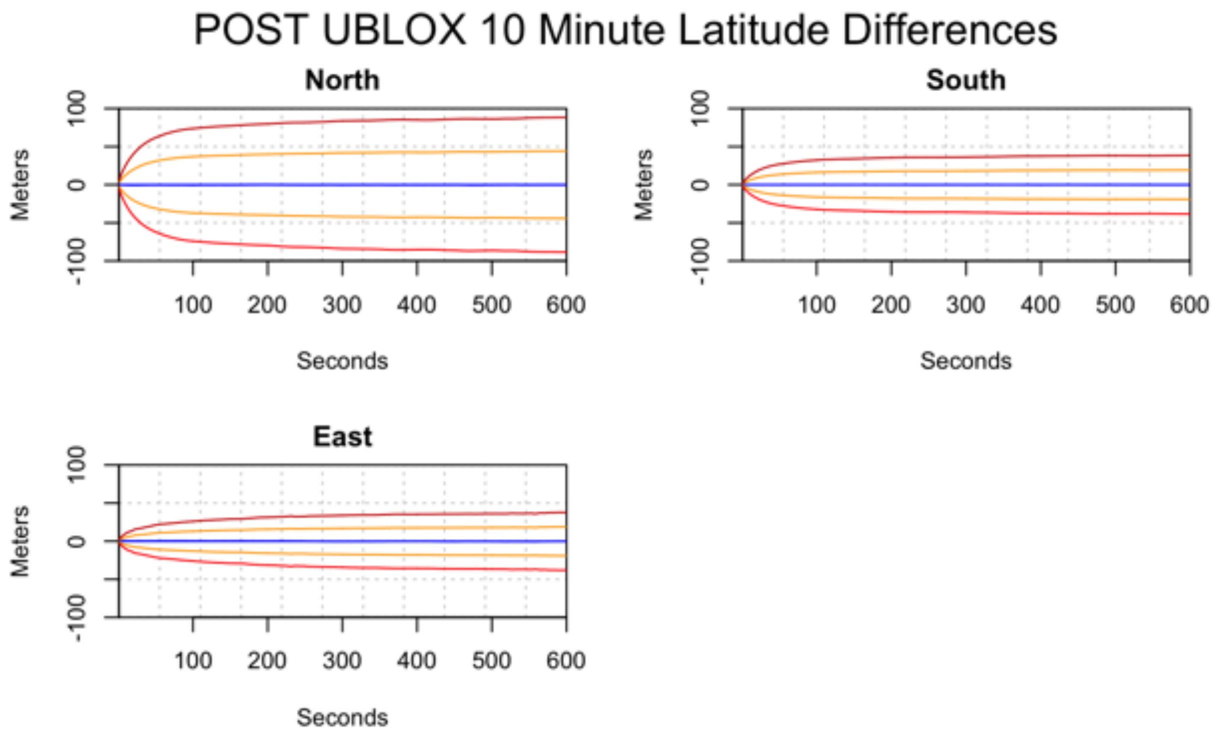


Figure 13 Latitude difference plots over 10 minutes for the UBLOX GNSS chip installed in the POST building. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

Altitude Differences

Control Sensor

The mean change in altitude recorded by the control sensor's UBLOX GNSS chip was 0m for all 4 days over the entire 10 minutes. On the day the window was facing north 95% of the differences in altitude it detected were within +/- 1.02m at 10 seconds, and were within +/- 8.68m at 10 minutes. On the day the window was facing south 95% of the differences in altitude it detected were within +/- 0.57m at 10 seconds, and were within +/- 5.45m at 10 minutes. On the day the window was facing east 95% of the differences in altitude it detected were within +/- 1.26m at 10 seconds, and were within +/- 8.64m at 10 minutes. On the day the window was facing west 95% of the differences in altitudes it detected were within +/- 0.61m at 10 seconds, and were within +/- 7.63m at 10 minutes (Figures 15-16).

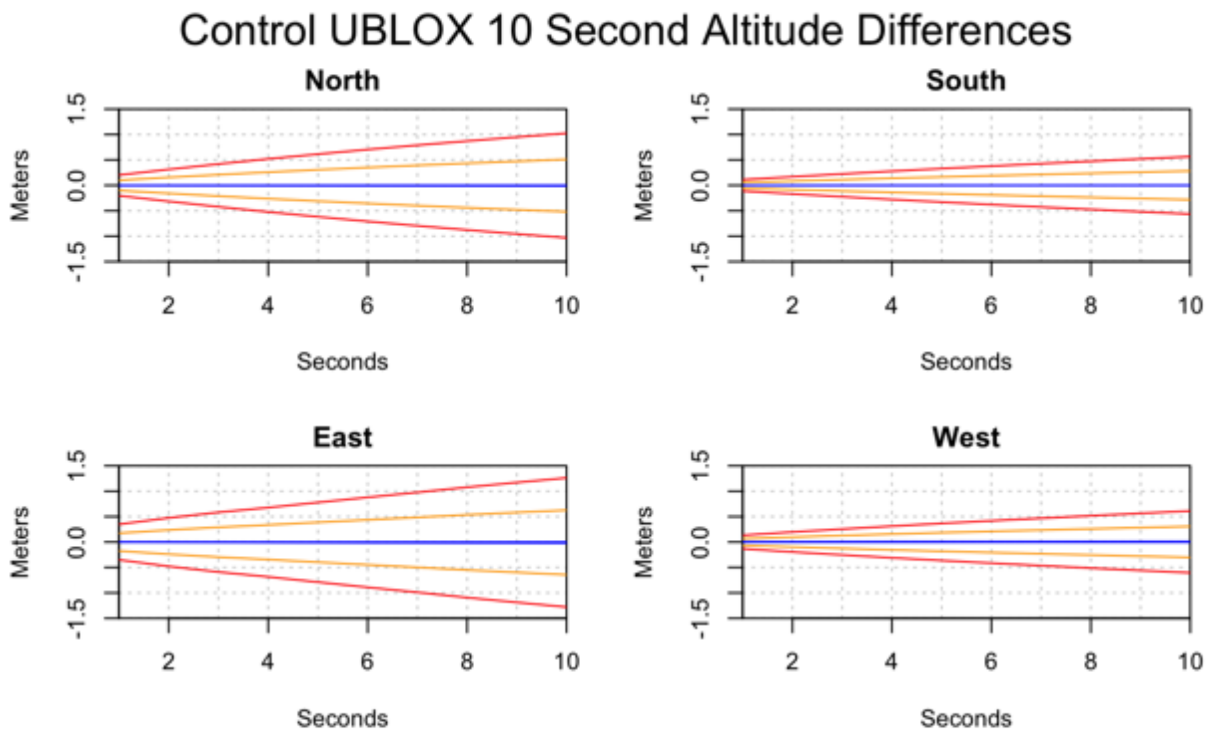


Figure 14 Altitude difference plots over 10 seconds for the Control UBLOX GNSS chip. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

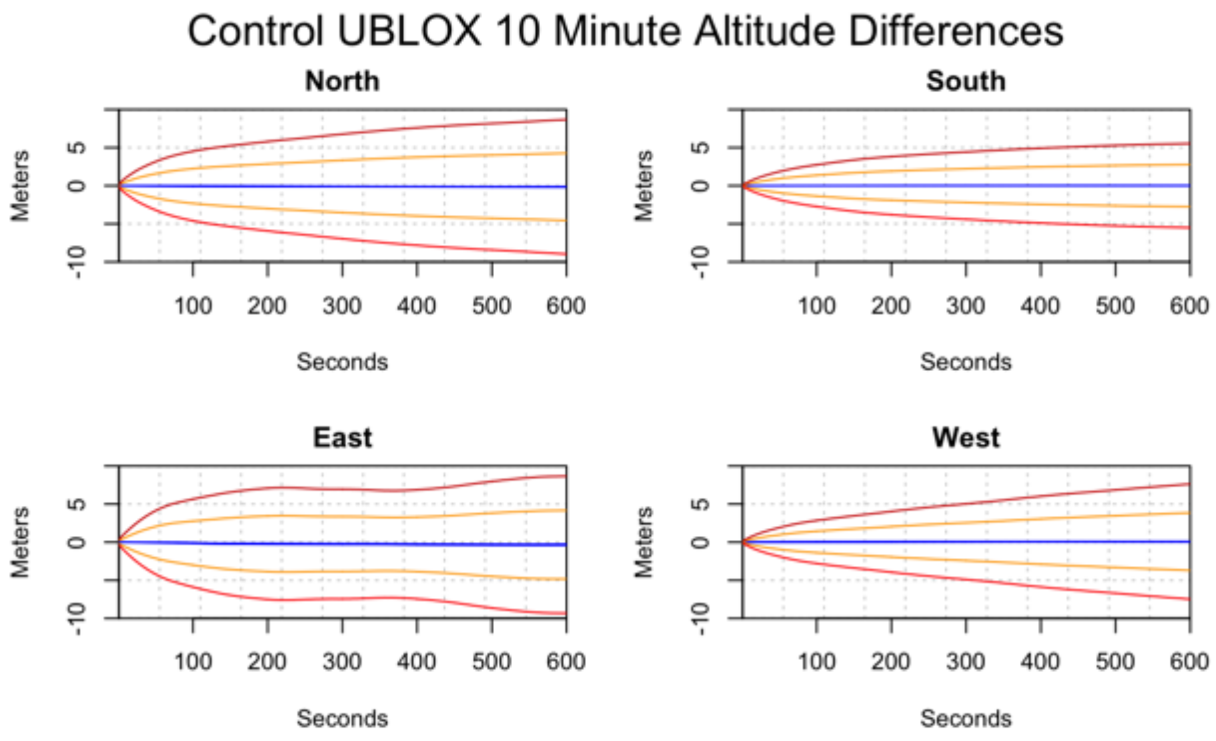


Figure 15 Altitude difference plots over 10 seconds for the Control UBLOX GNSS chip. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

UBLOX GNSS Chip in the Box

The mean change in altitude recorded by the UBLOX GNSS chip of the sensor installed inside the box was 0m over the entire 10 minutes for all four directions. While the window was facing north 95% of the differences in altitude it detected were within +/- 1.08m at 10 seconds, and were within +/- 14.78m at 10 minutes. While the window was facing south 95% of the differences in altitude it detected were within +/- 1.02m at 10 seconds, and were within +/- 13.39m at 10 minutes. While the window was facing east 95% of the differences in altitude it detected were within +/- 1.03m at 10 seconds, and were within +/- 13.00m at 10 minutes.

While the window was facing west 95% of the differences in altitude it detected were within +/- 1.08m at 10 seconds, and were within +/- 13.88m at 10 minutes (Figures 17-18).

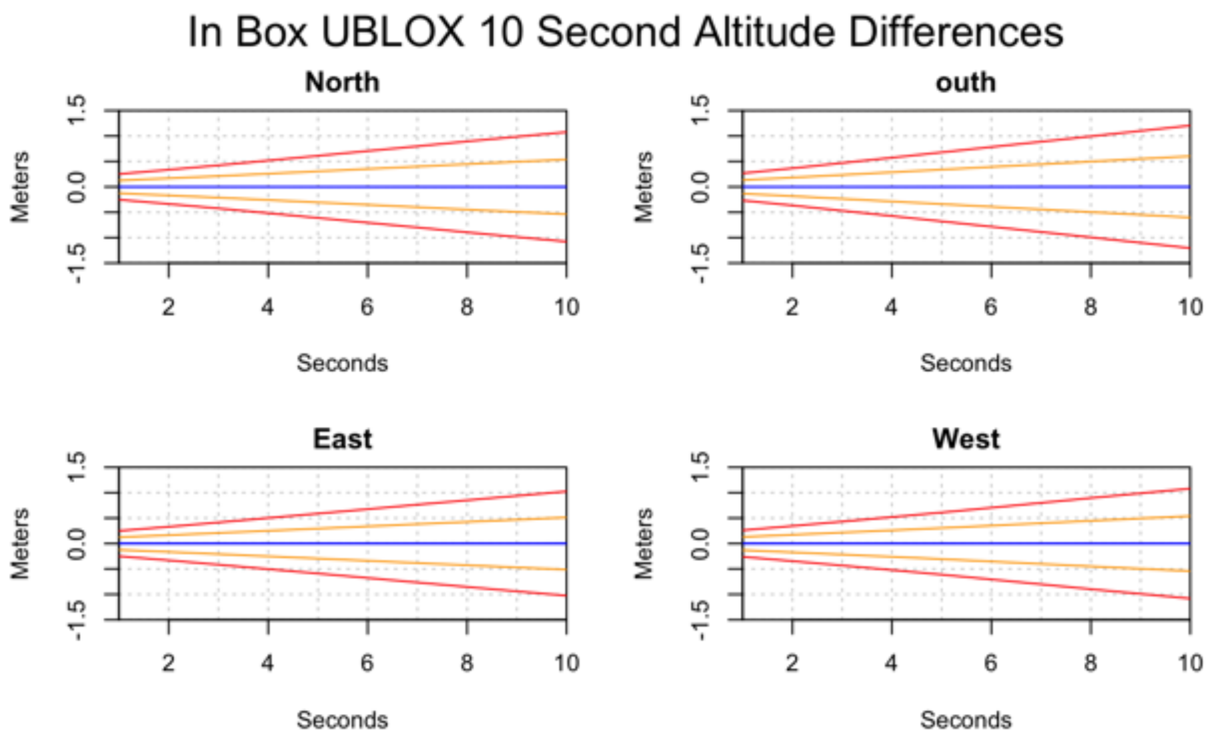


Figure 16 Altitude difference plots over 10 seconds for the UBLOX GNSS chip installed inside the box. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

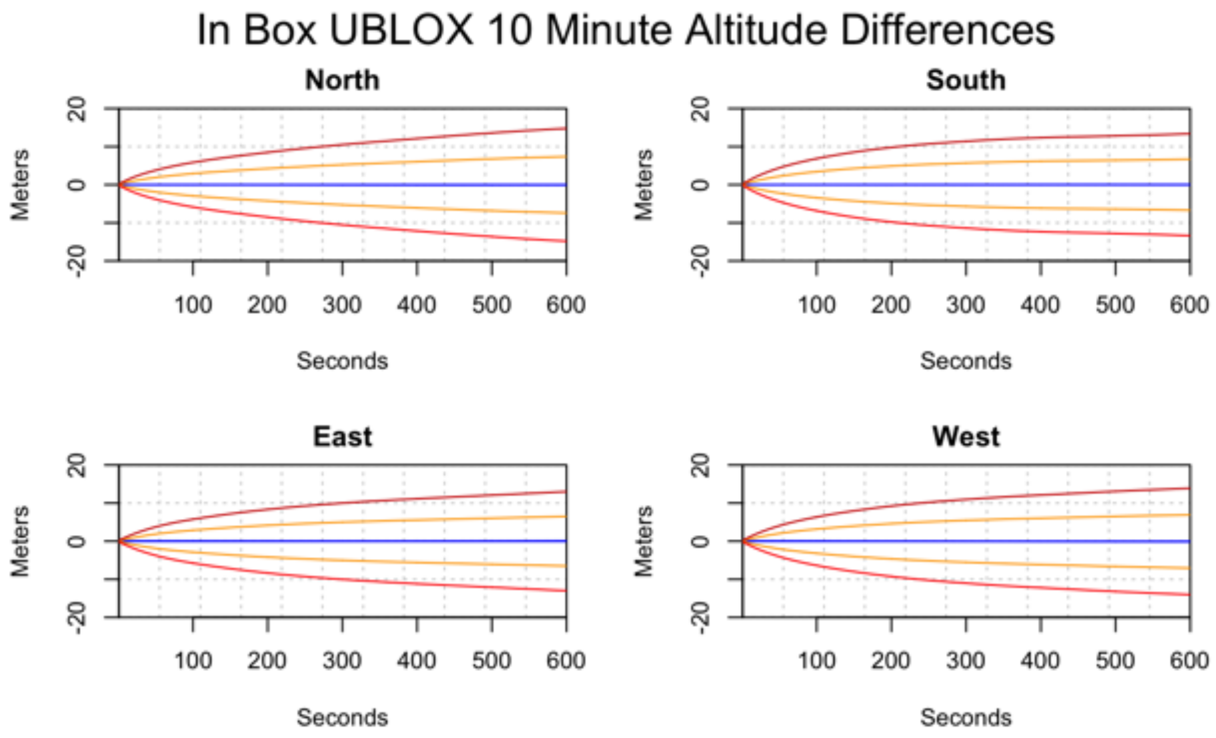


Figure 17 Altitude difference plots over 10 minutes for the UBLOX GNSS chip installed inside the box. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

Internal GNSS Chip in the Box

The mean change in altitude recorded by the phone's internal GNSSS chip while it was installed inside the box was 0m over 10 minutes in all four directions. While the window was facing north, 95% of the differences in altitude at 10 seconds were within +/- 1.82m, and were within +/- 13.39m at 10 minutes. While the window was facing south, 95% of the differences in altitude were within +/- 1.78m at 10 seconds, and were within +/- 12.21m at 10 minutes. While the window was facing east, 95% of the differences in altitude were within +/- 1.78m at 10 seconds, and were within +/- 12.25m at 10 minutes. While the window was facing west, 95% of the differences in altitude were within +/-1.78m at 10 seconds, and were within +/-12.24m at 10 minutes (Figures 19-20).

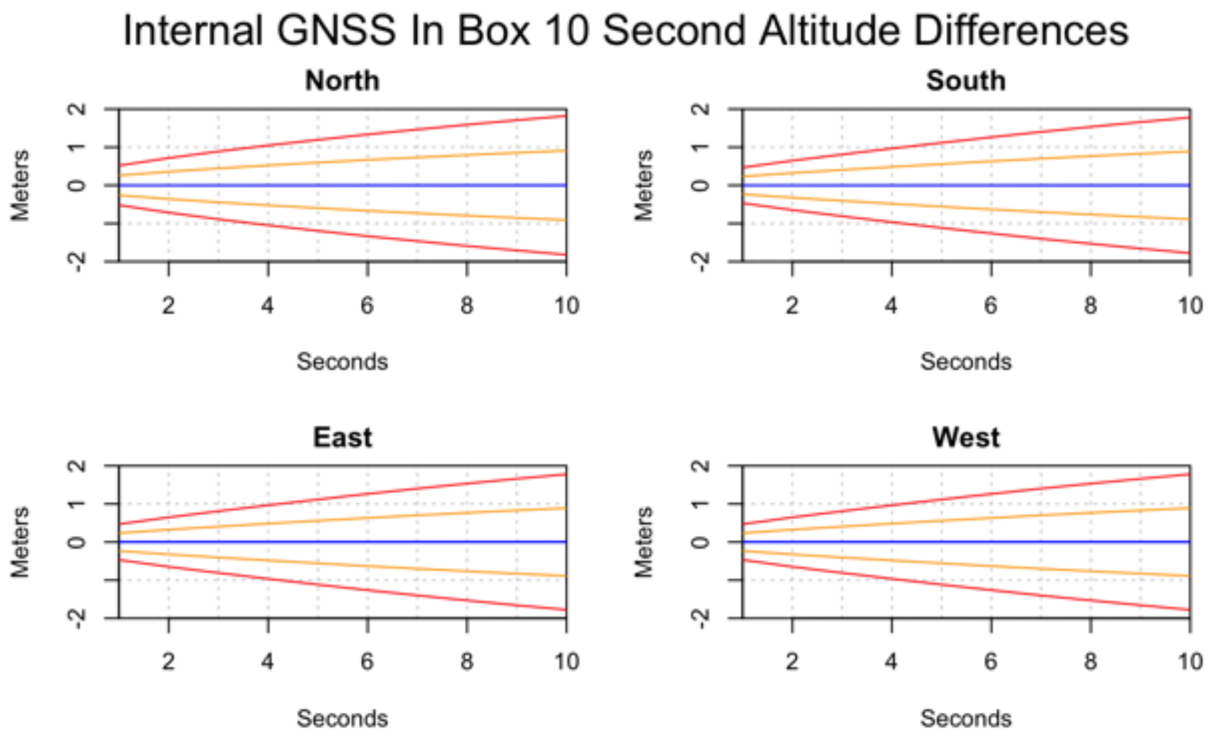


Figure 18 Altitude difference plots over 10 seconds for the phones Internal GNSS chip installed inside the box. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

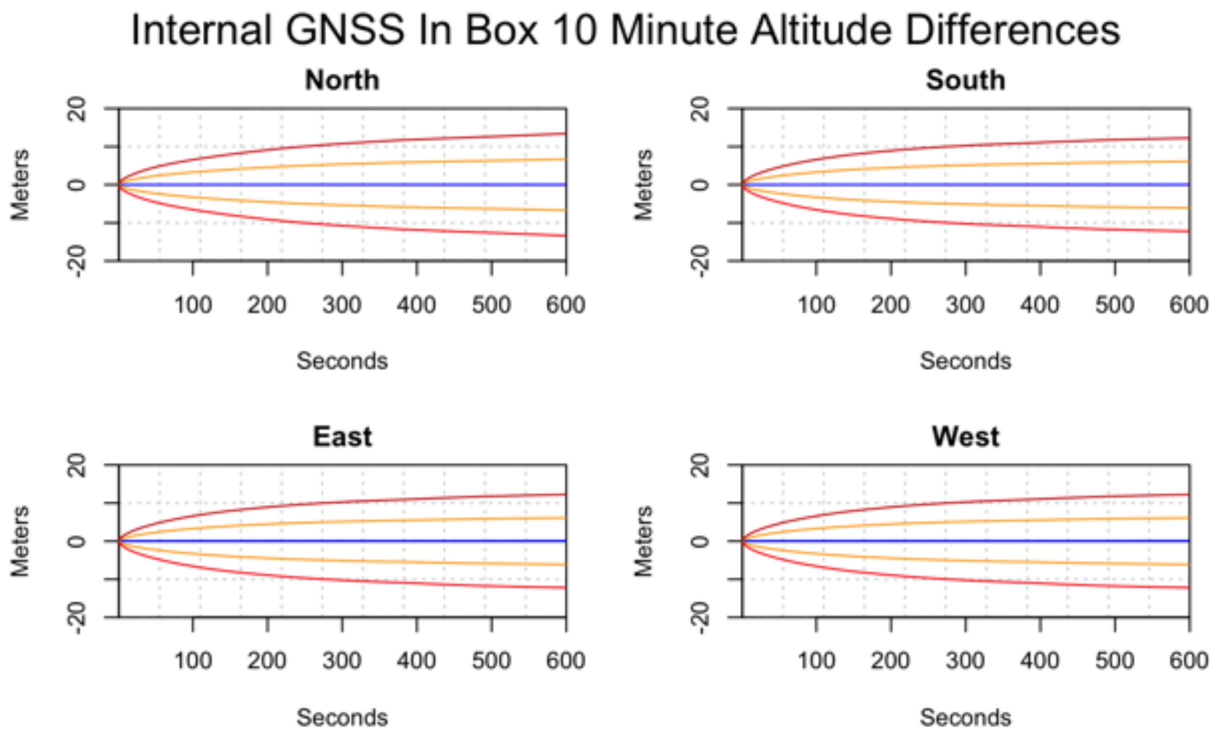


Figure 19 Altitude difference plots over 10 minutes for the phones Internal GNSS chip installed inside the box. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

UBLOX GNSS Chip in Residential Home

The UBLOX GNSS chip of the sensor that was installed in the south facing window of the residential home in California recorded a mean change in altitude of 0m over 10 minutes. While the UBLOX GNSS chip was installed in the home 95% of the differences in reported altitudes were within +/- 3.31 m over 10 seconds, and were within +/- 18 m over 10 minutes (Figure 21).

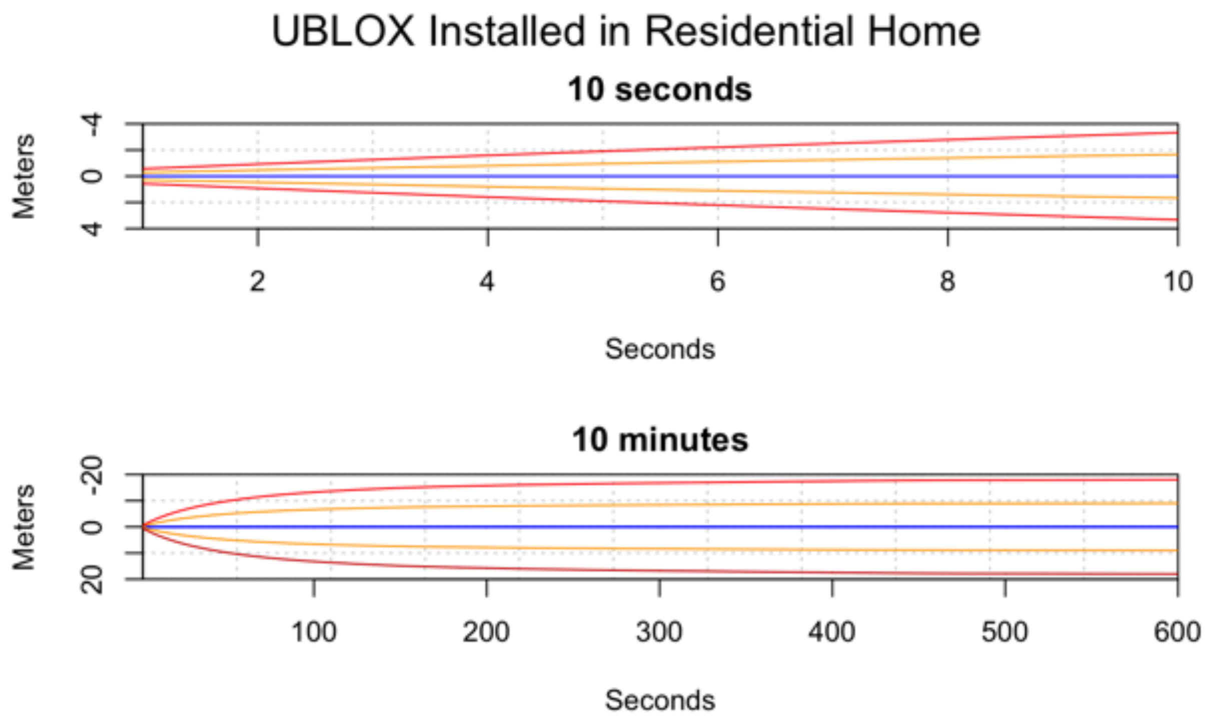


Figure 20 Altitude difference plots over 10 seconds and 10 minutes for the GNSS UBLOX GNSS sensor that was installed in a south facing window of a residential home in Novato California. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

UBLOX GNSS Chip in POST

The UBLOX GNSS chip of the sensor that was installed in the windows of the POST building recorded a mean change in altitude of 0m over the entire 10 minutes for all 3 days data was collected. While the sensor was installed in the north facing window 95% of the differences in altitude its external UBLOX GNSS chip detected were within +/- 10.48m at 10 seconds, and were within +/- 77.59m at 10 minutes. While the sensor was installed in the south facing window 95% of the differences in altitudes its external UBLOX GNSS chip detected were within +/- 12.91m at 10 seconds, and were within +/- 79.37m at 10 minutes. While the sensor was installed in the east fading window 95% of the differences in altitudes its external UBLOX GNSS chip detected were within +/- 11.21m at 10 seconds, and were within +/-81.18m at 10 minutes (Figures 22-23).

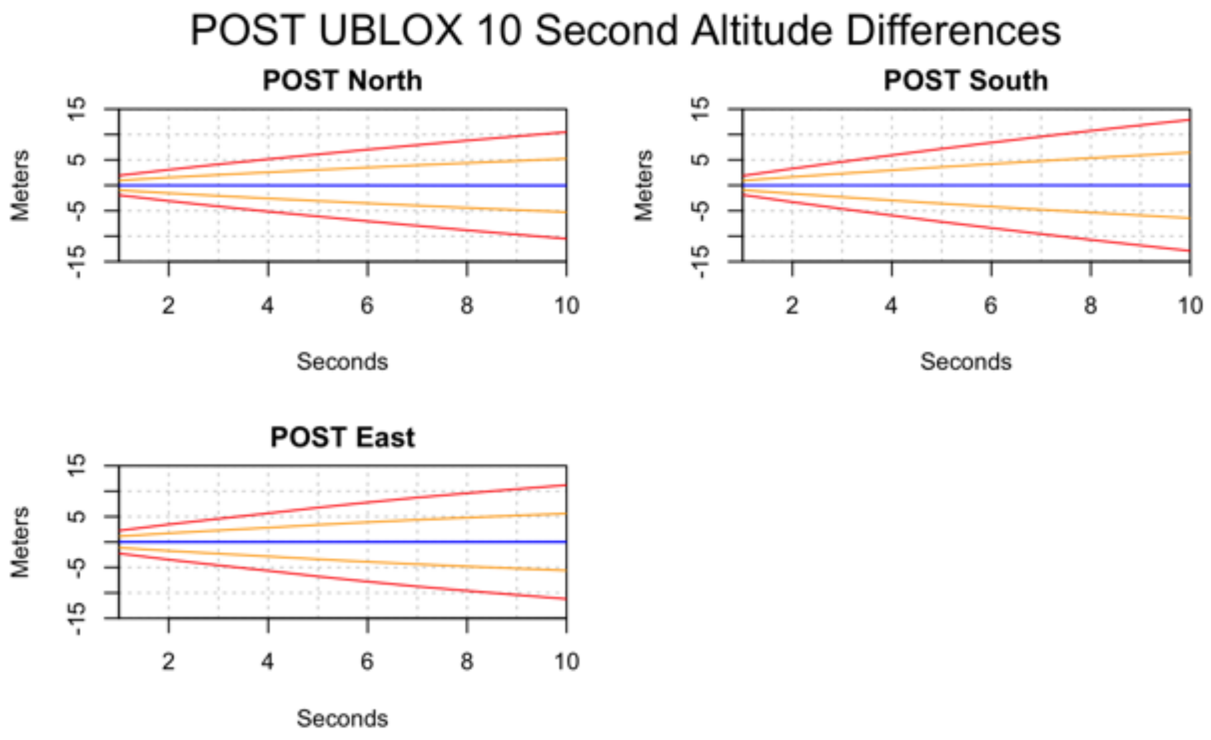


Figure 21 Altitude difference plots over 10 seconds for the UBLOX GNSS chip installed inside the POST building. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

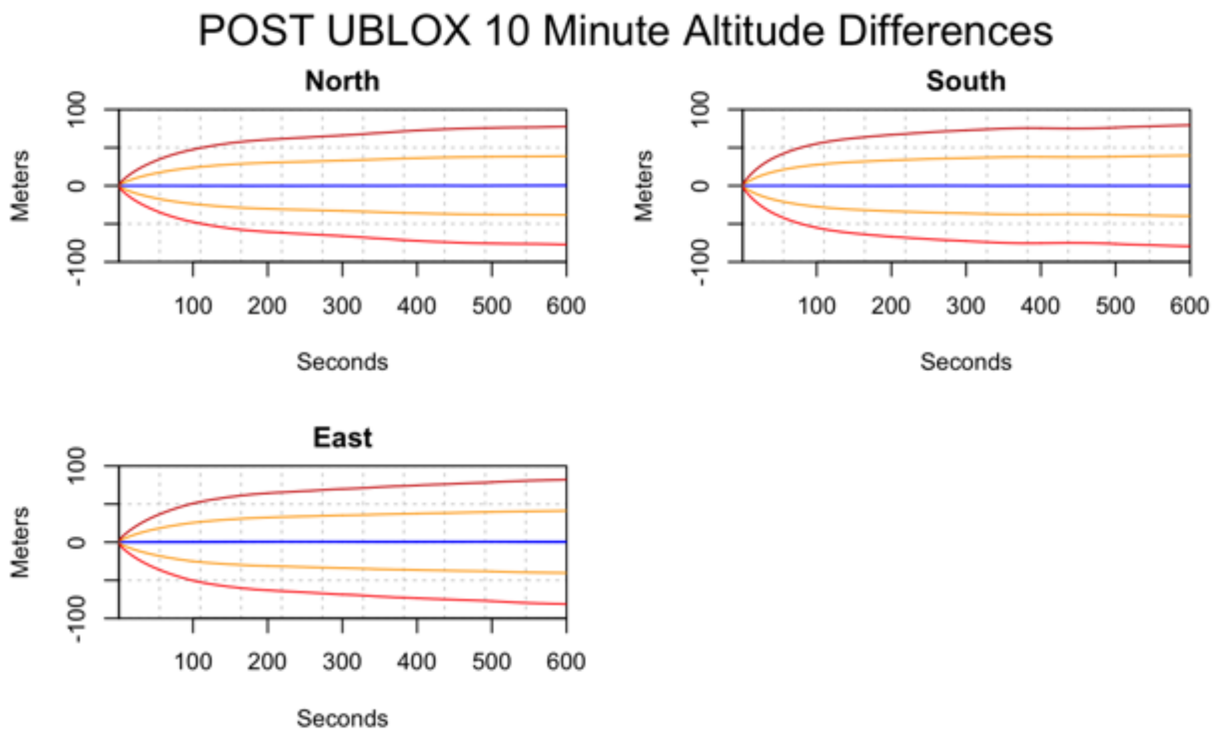


Figure 22 Altitude difference plots over 10 minutes for the UBLOX GNSS chip installed inside the POST building. The blue line shows the mean difference, the orange lines shows on sd from the mean, and the red lines show 2sd from the mean.

Analysis of GNSS Sensor Performance

The data from the control sensor's UBLOX GNSS chip and the data from the UBLOX GNSS chip installed inside the box showed consistent changes in latitude over 10 minutes on the days that the window was facing south, west, and east. Both sensors showed larger changes in latitude on the day that the window was facing north. The sensor that was installed in the windows of the POST building showed similar changes in latitude over 10 minutes on the days that it collected data in the windows facing south and east, and showed larger changes in latitude on the day it was installed in the north facing window. This discrepancy between the difference in latitude recorded by the UBLOX GNSS chips in all the installation scenarios while they were facing north and while they were facing the other directions was not seen in the altitude difference data or in the latitude difference data recorded by the phone's internal GNSS chip.

The GPS sky view plots for the days when data was collected with the control UBLOX GNSS chip and the UBLOX GNSS chip installed inside the box show that there was no difference in the GPS constellation over the 4 days that would account for the poor performance of the systems on the day they were facing north (Figures 24-31). Histograms of the Horizontal Dilution of Precision (HDOP), a measure of the strength of the geometry of the GNSS satellite constellation that the sensor can see, for the control UBLOX GNSS chip, the UBLOX GNSS chip installed inside the box, and the UBLOX GNSS chip installed in POST were consistent over all the days that data was collected as well (Figures 32-33). The median HDOP for the UBLOX GNSS chip in the control sensor for the days the window was facing North, South, East, and West were 0.81, 0.96, 1.01, and 0.81 respectively. The median HDOP for the UBLOS GNSS chip in the sensor installed inside the box while the window was facing North, South, East, and West were 0.83, 0.79, 0.79, and 0.77 respectively. The median HDOP for the UBLOX GNSS chip in the sensor installed inside POST in the North, South, and East facing windows were 2.04, 1.34, and 3.6 respectively. The similar HDOP values, and the consistency of the GPS constellation suggests that there is another reason for the poor performance of the sensor on the days they were facing North.

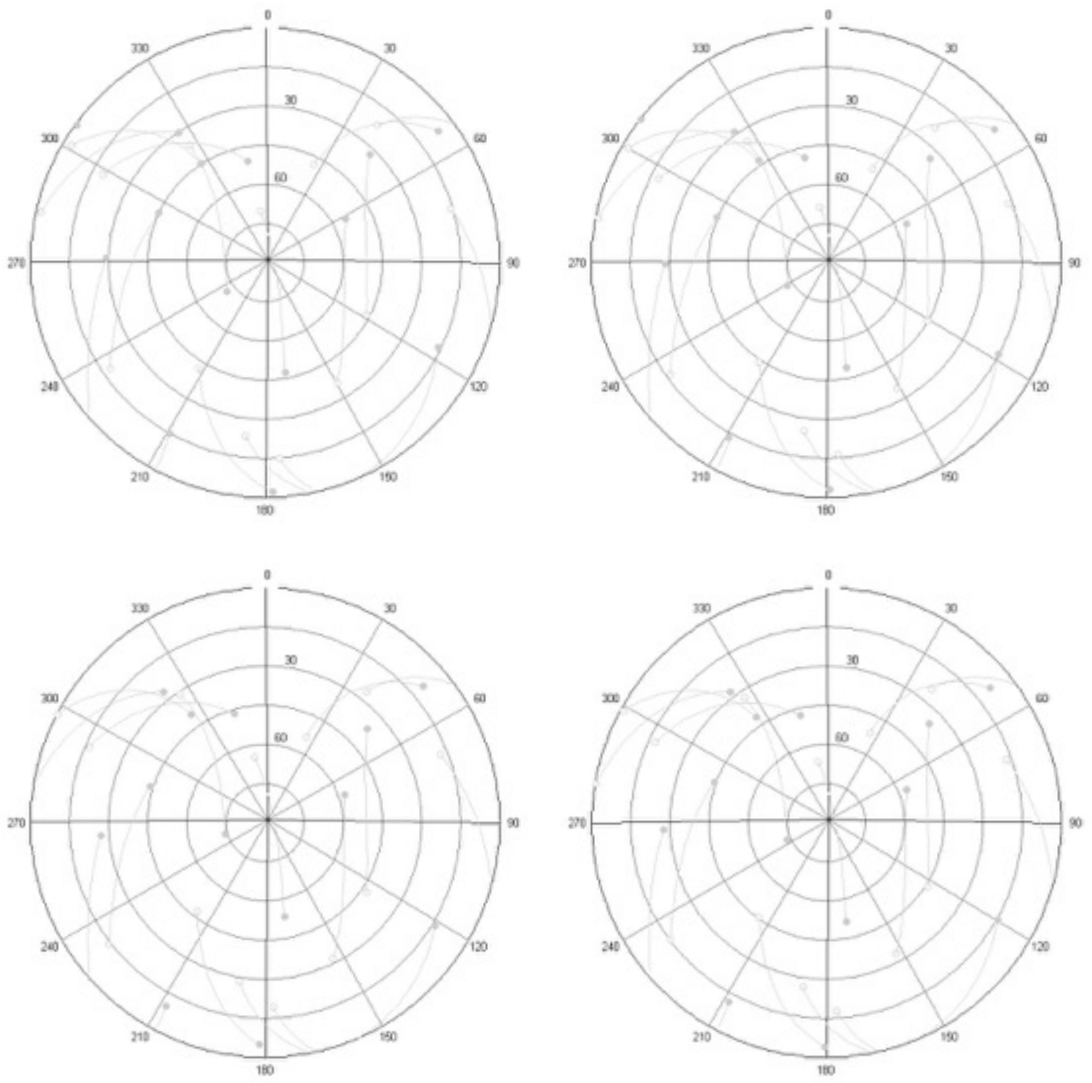


Figure 23 Control/Box GPS sky view for 0:00-3:00 UTC. (From top left to bottom right: North facing, South Facing, East Facing, West Facing)

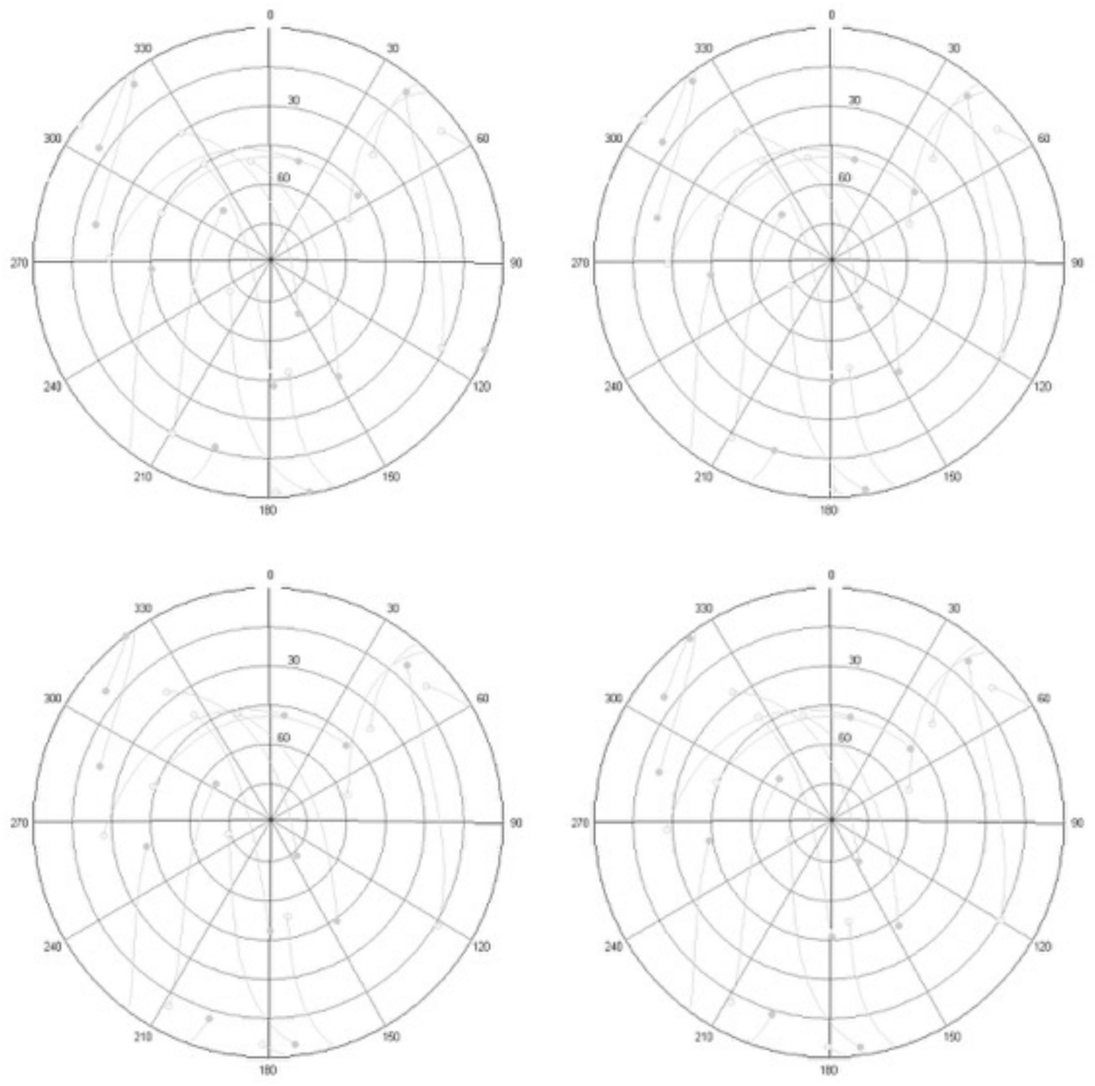


Figure 24 Control/Box GPS sky view for 3:00-6:00 UTC. (From top left to bottom right: North facing, South Facing, East Facing, West Facing)

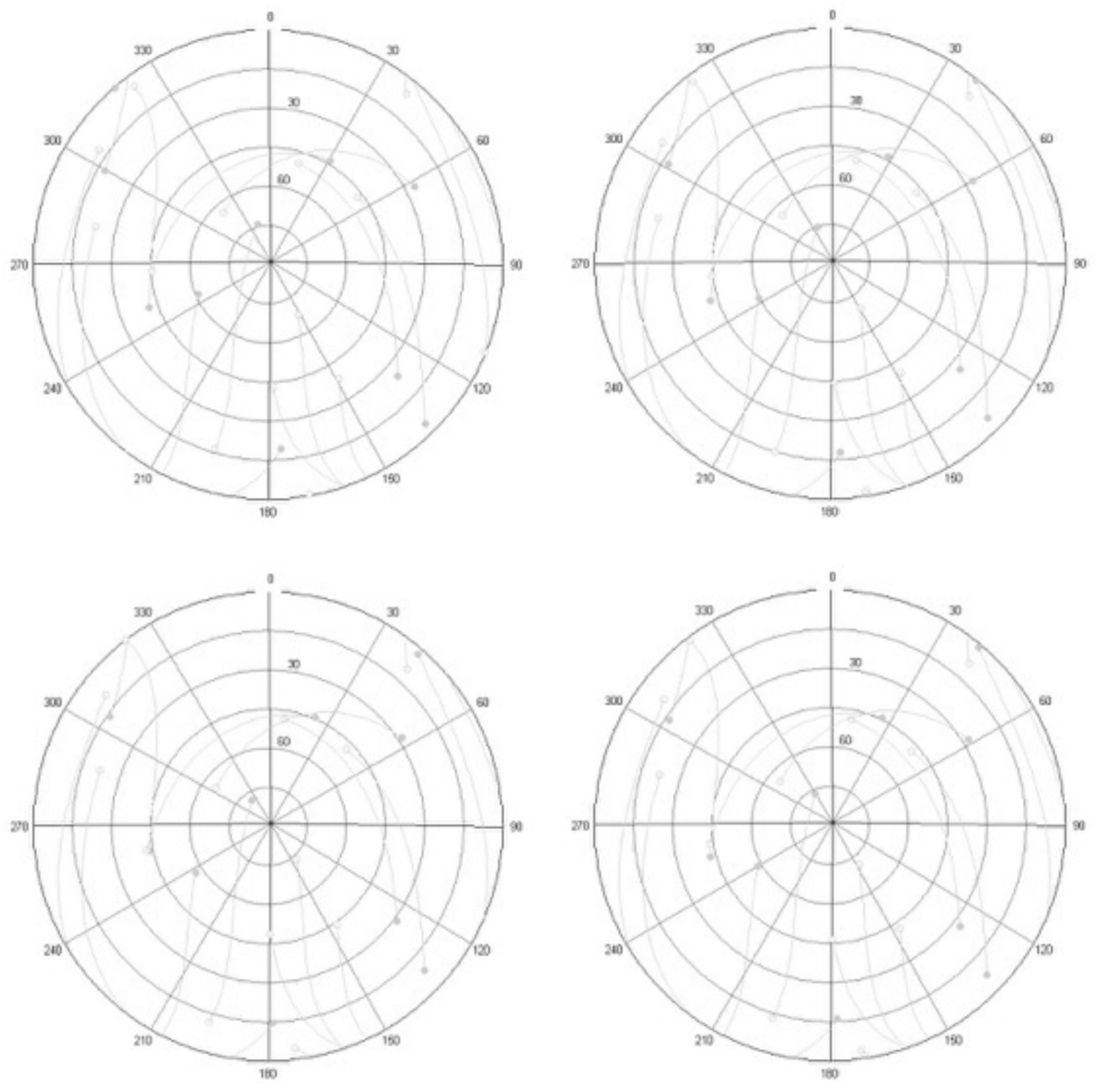


Figure 25 Control/Box GPS sky view for 6:00-9:00 UTC. (From top left to bottom right: North facing, South Facing, East Facing, West Facing)

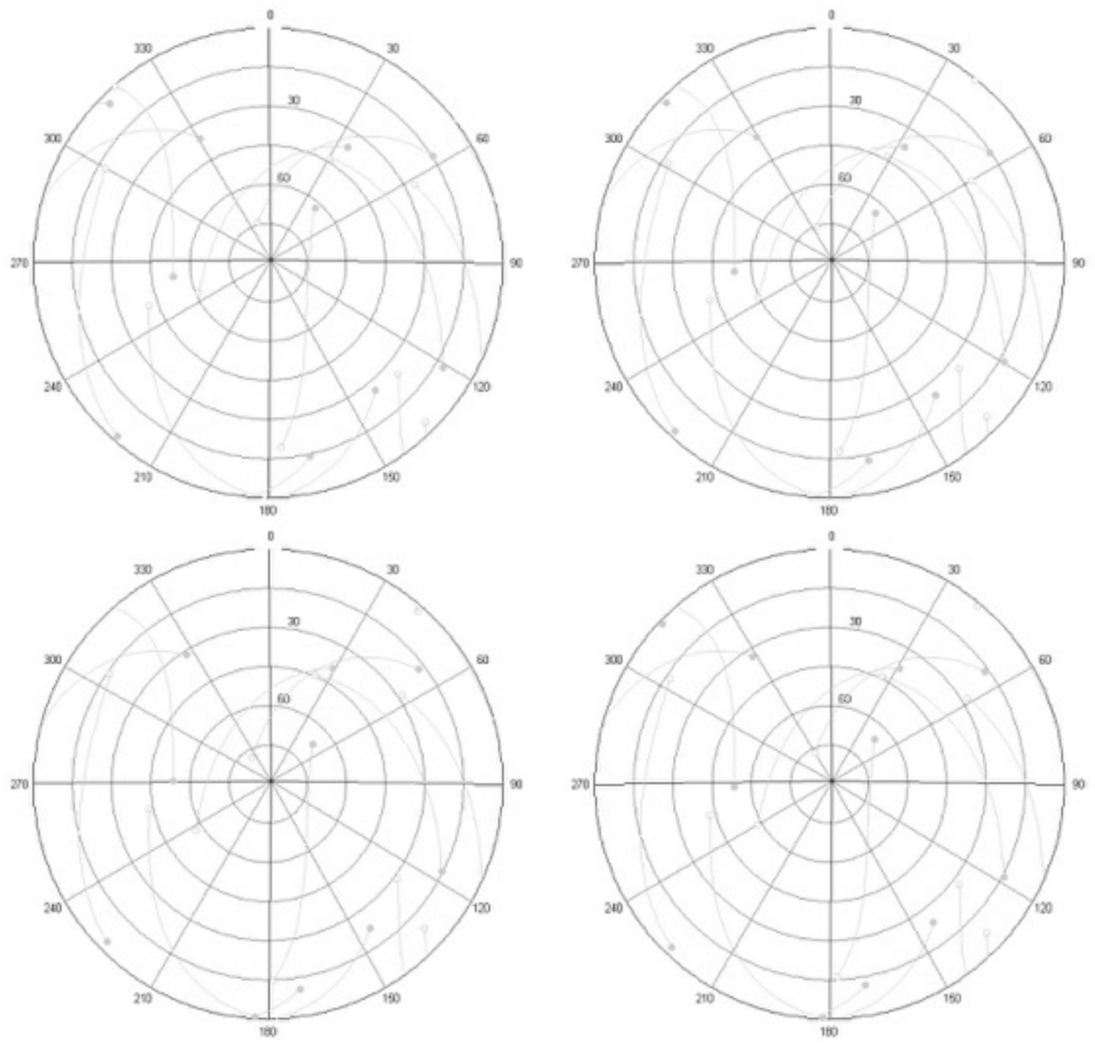


Figure 26 Control/Box GPS sky view for 9:00-12:00 UTC. (From top left to bottom right: North facing, South Facing, East Facing, West Facing)

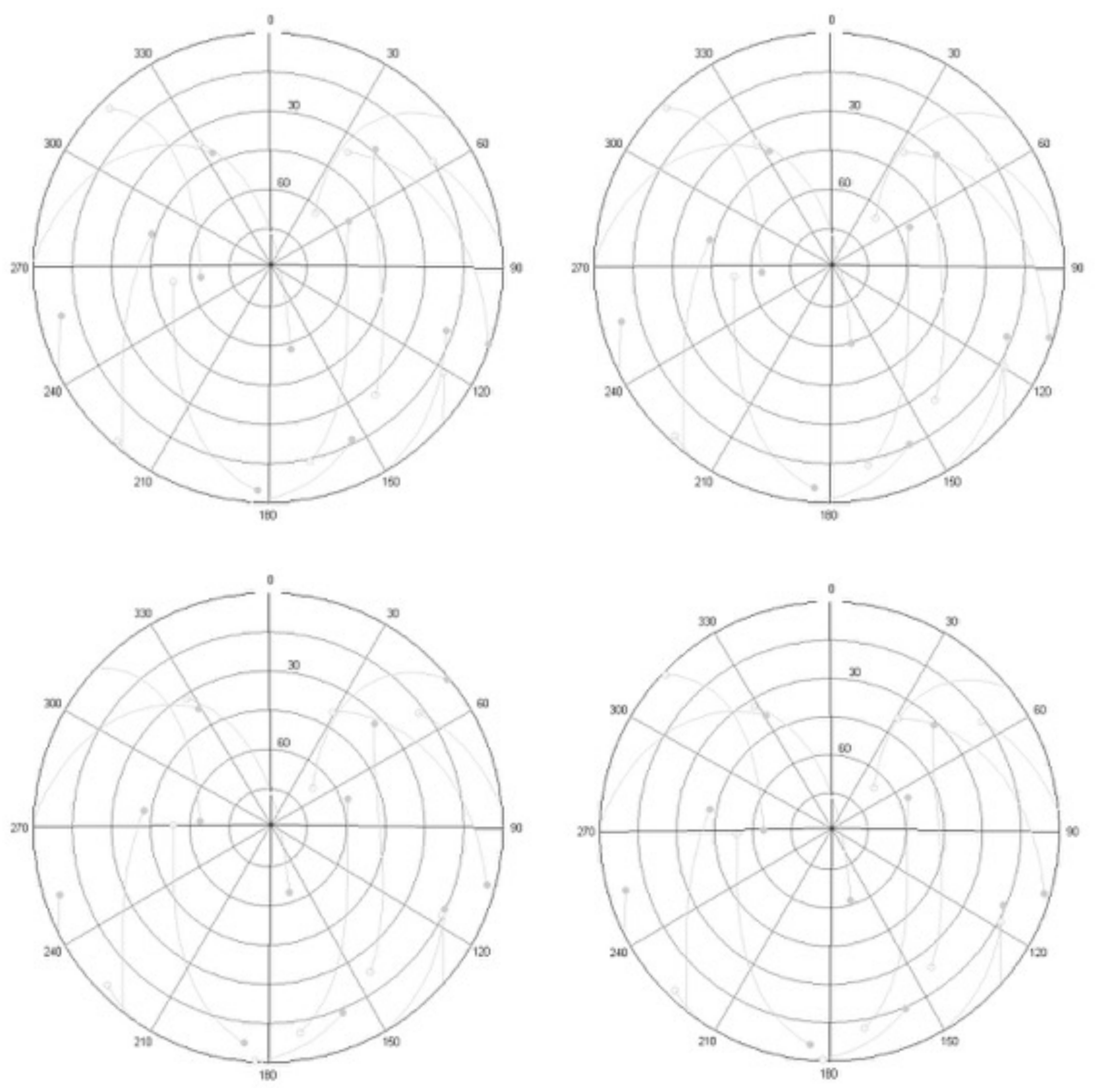


Figure 27 Control/Box GPS sky view for 12:00-15:00 UTC. (From top left to bottom right: North facing, South Facing, East Facing, West Facing)

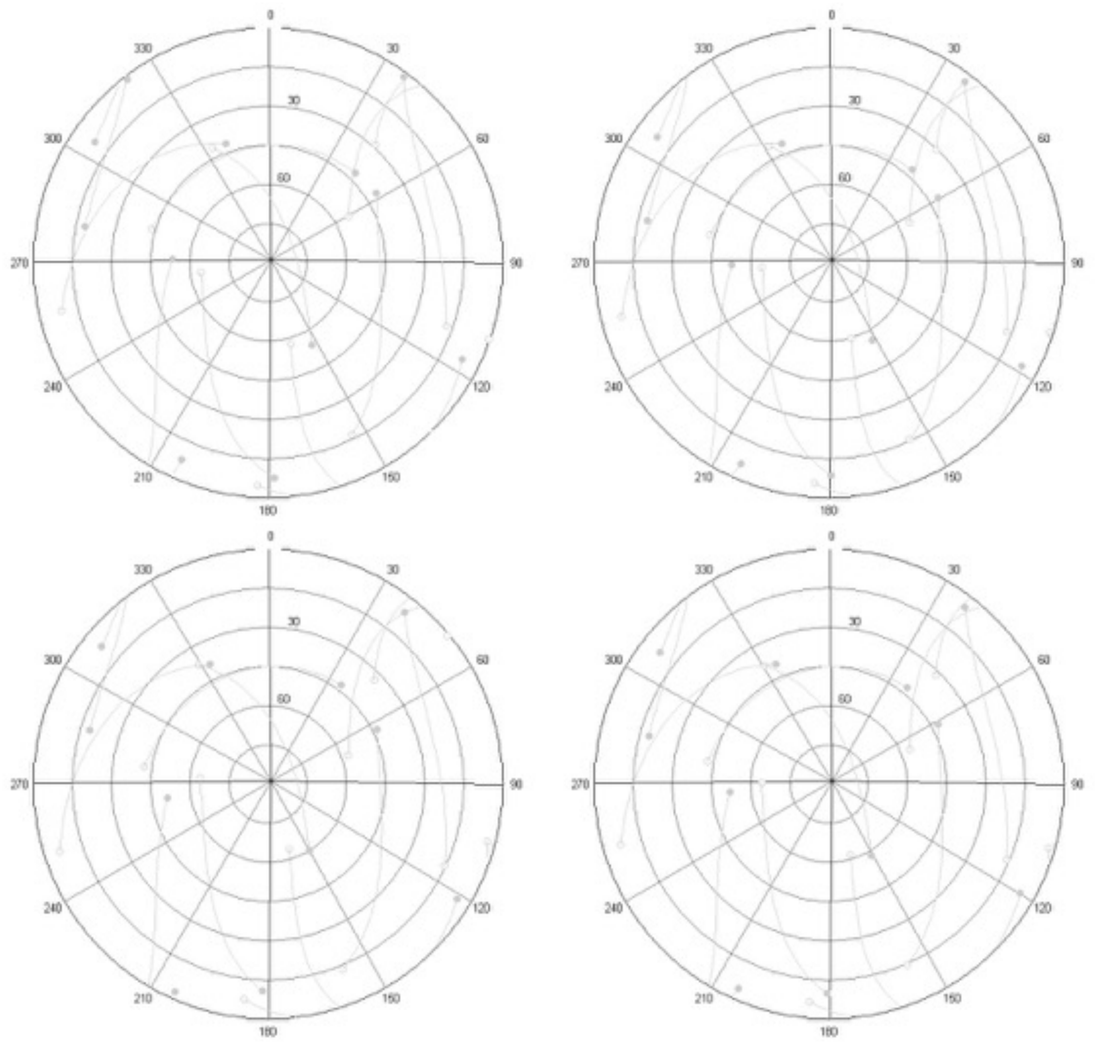


Figure 28 Control/Box GPS sky view for 15:00-18:00 UTC. (From top left to bottom right: North facing, South Facing, East Facing, West Facing)

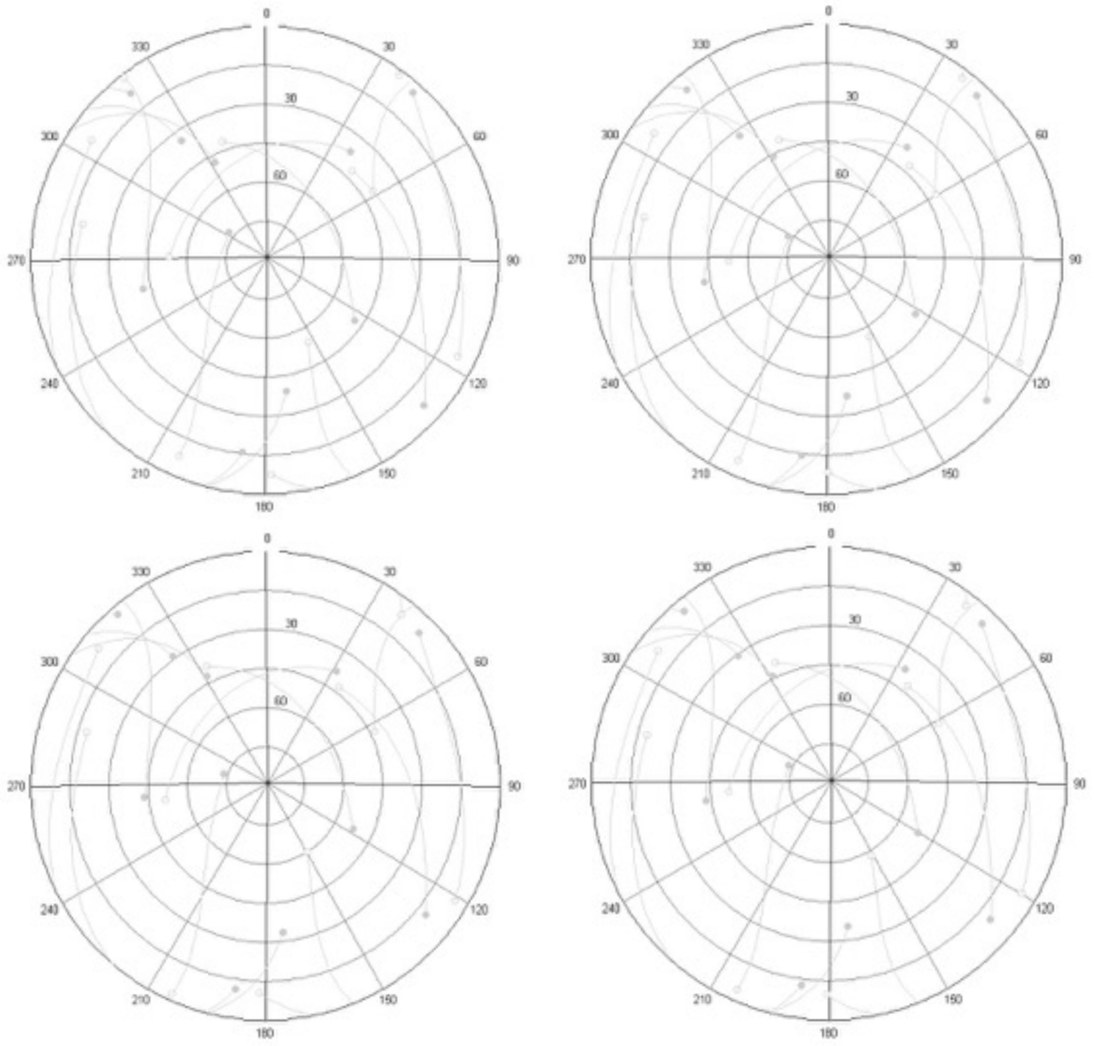


Figure 29 Control/Box GPS sky view plots for 18:00-21:00 UTC. (From top left to bottom right: North facing, South Facing, East Facing, West Facing)

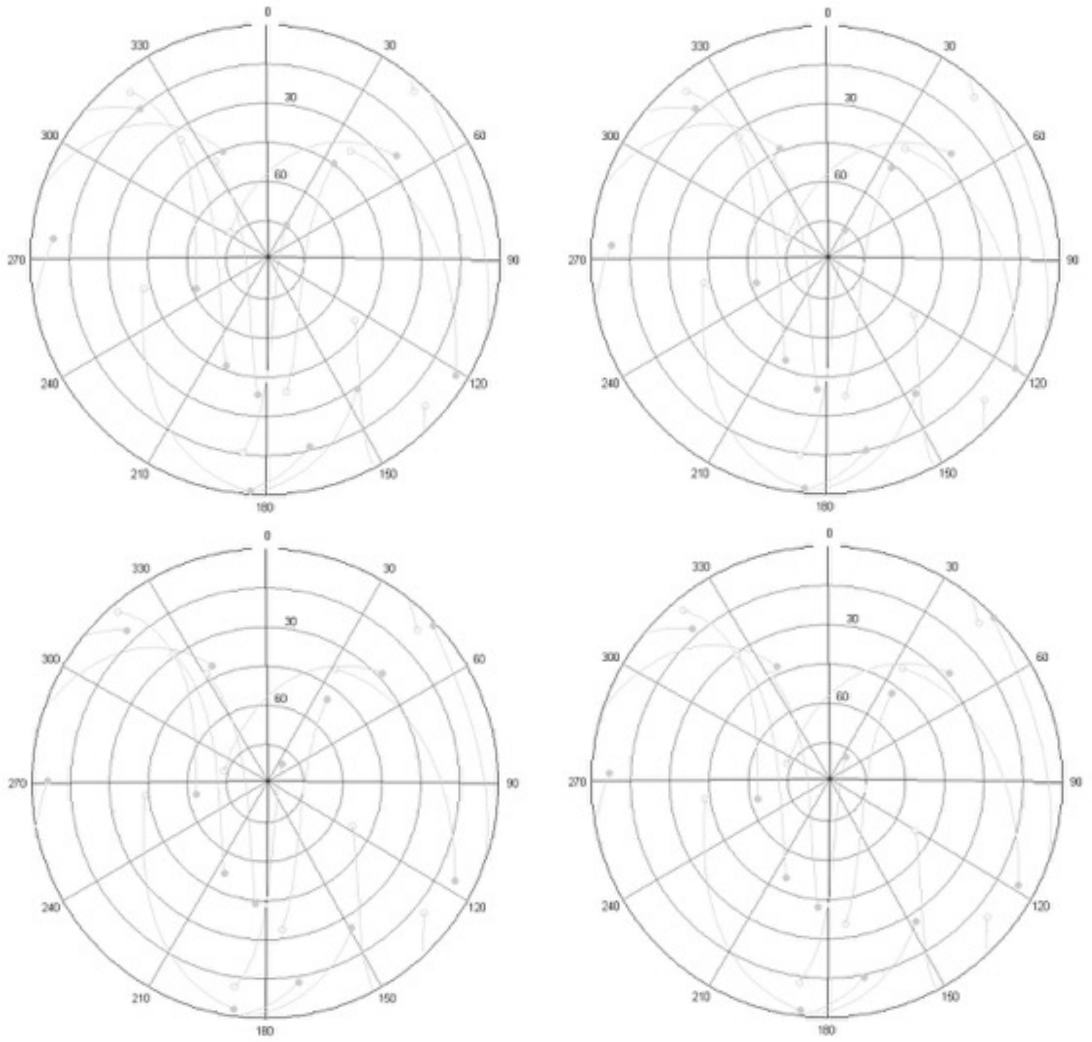


Figure 30 Control/Box GPS sky view for 21:00-24:00 UTC. (From top left to bottom right: North facing, South Facing, East Facing, West Facing)

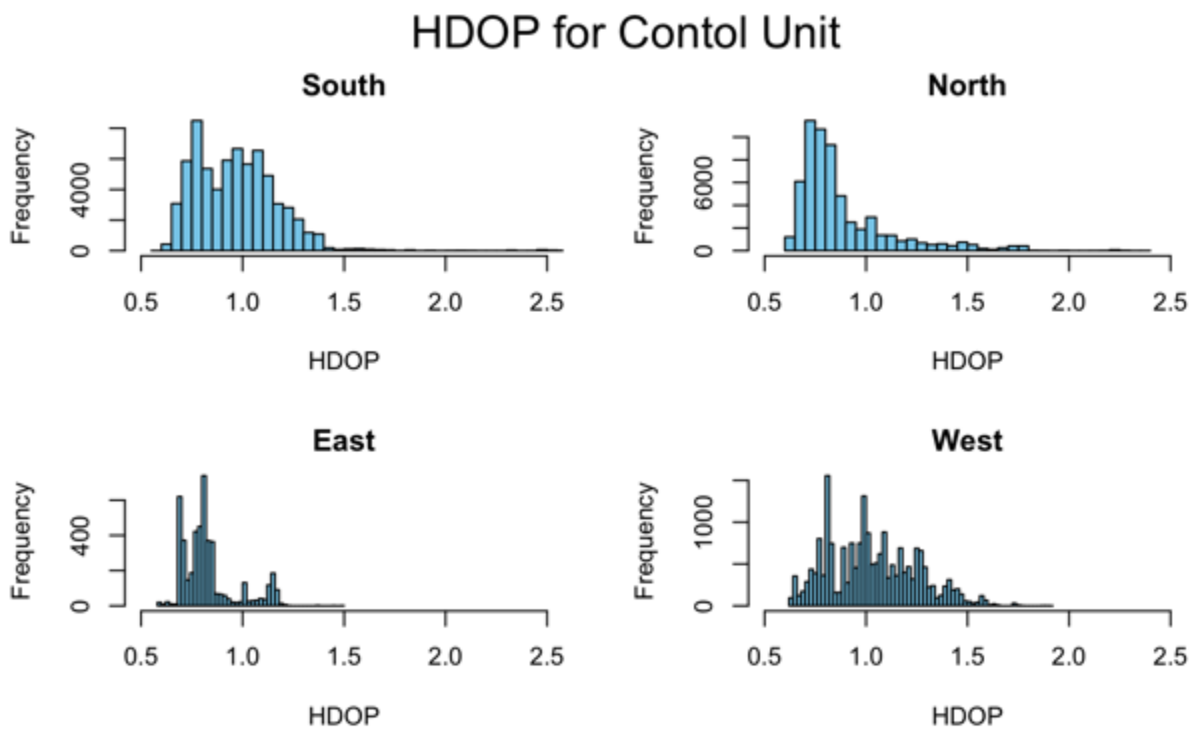


Figure 31 Histograms of the horizontal dilution of precision for the four days the UBLOX GNSS chip which was installed on the roof collected data. There isn't a large range in the values, and the day the window was facing North showed some of the lower HDOP values.

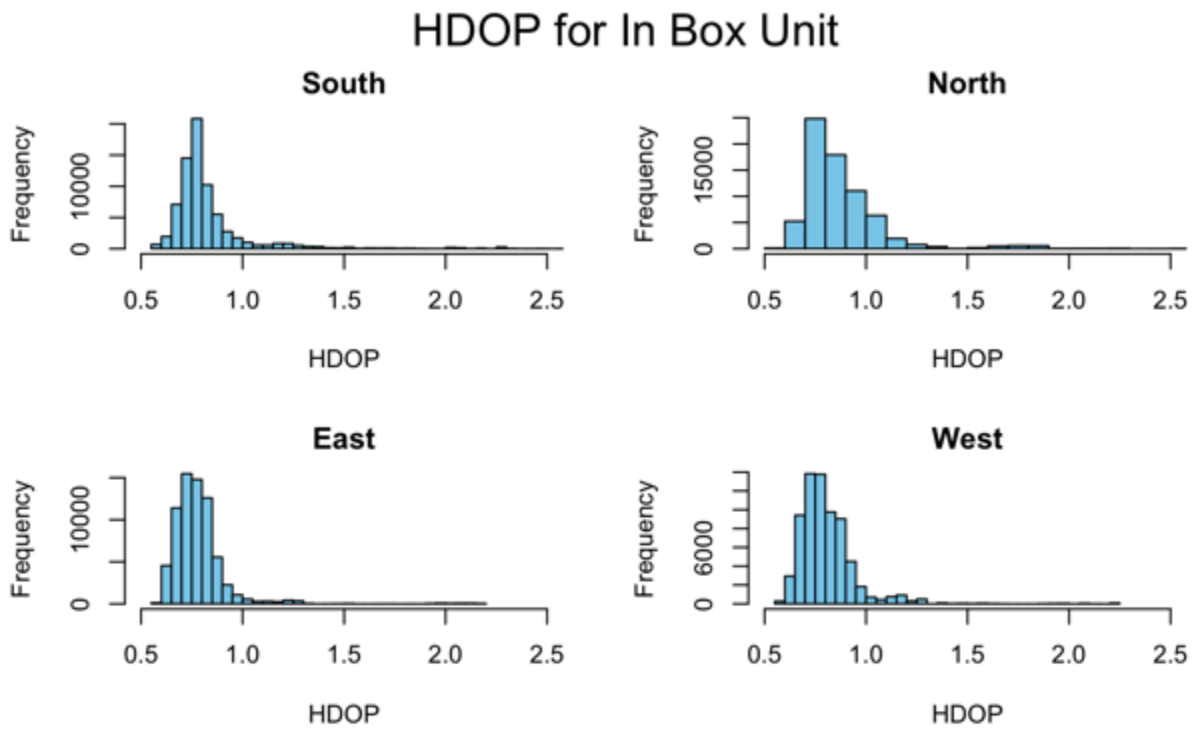


Figure 32 Histograms of the horizontal dilution of precision for four days the UBLOX GNSS chip that was installed inside of the box collected data. The histogram for the day the window was facing North doesn't seem to be any different from the rest of the histograms.

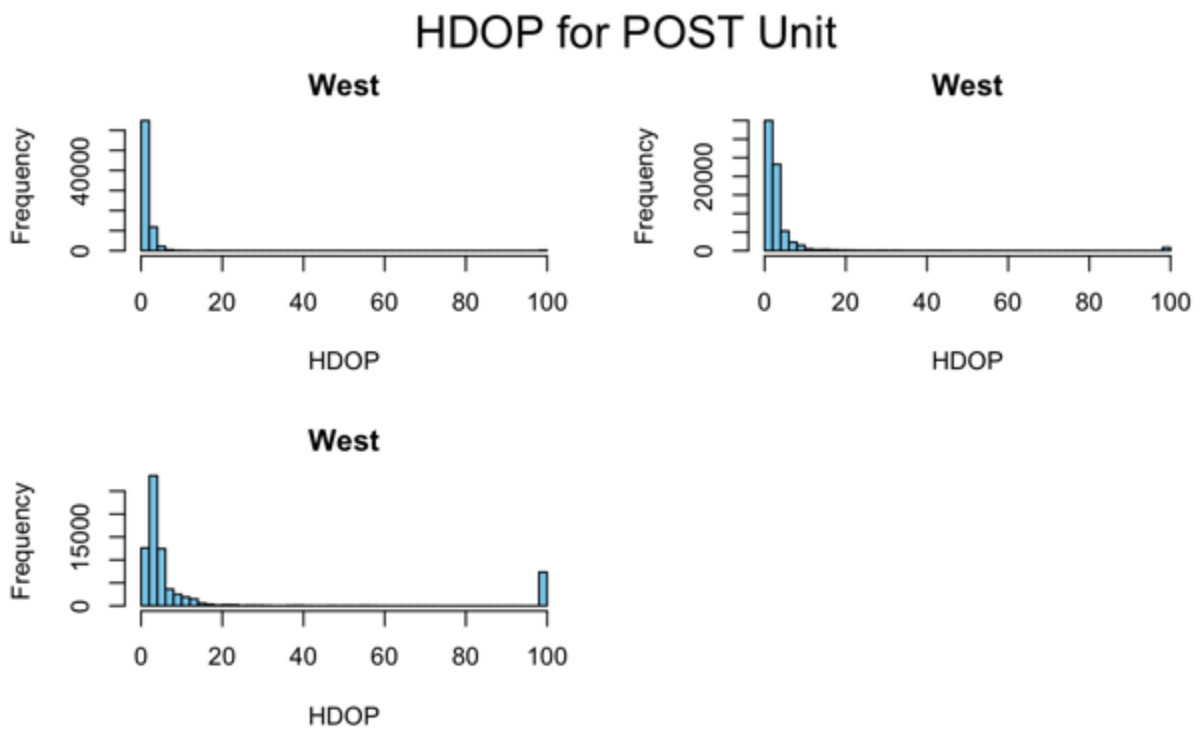


Figure 33 Histograms of the horizontal dilution of precision for the three days the UBLOX GNSS chip that was installed inside of the POST building collected data. The reported values are within the same range for all three days.

Difference Test Discussion

There were large sections of missing data from all four days that the UBLOX GNSS chip was installed in the POST building. While the UBLOX was installed in the south, north, and east facing windows there were 6,273 seconds, 15,504 seconds, and 13,879 seconds respectively where the UBLOX GNSS chip was unable to obtain a position. The reinforced concrete building also caused large drift in the reported latitudes and altitudes. Our EEW sensors would not be able to detect displacements from earthquakes while installed in a reinforced concrete building.

There was a decline in the performance of the UBLOX GNSS chip while it was inside the box. It is unclear why the drift in latitude was larger for both the control UBLOX GNSS chip and the UBLOX GNSS chip installed in the box on the day that the window was facing North. This larger range must be taken as our system’s threshold for detecting latitudinal displacement. Our EEW system should be able to detect displacements in latitude larger than +/- 0.76m, and vertical displacements larger than +/- 1.2m at 10 seconds while installed inside of a brick or wooden building.

Data was collected from only one window in the residential home in Novato, California, and more data would be needed to verify these results are a good representation of the UBLOX GNSS receivers drift while installed in a building of similar construction. However, the initial results are positive. As expected, the reported latitudes and altitudes were not as tightly constrained as they were in the control unit or the sensor installed inside the box, but we should be able to detect displacements in latitude larger than 1.51 m at 10 seconds, and vertical displacements of 3.31 m over 10 seconds.

Earthquake Static Ground Displacement Tests

The results of the difference tests suggest that one of our EEW systems, with a UBLOX GNSS chip, should be able to detect displacements from large earthquakes if it is installed on a roof or inside a brick or wooden building. We ran a displacement test on the roof of the HIG building at UH Manoa to test what displacements we would be able to detect in a real earthquake event. In their 2011 paper *the 2010 Mw 8.8 Maule Megathrust Earthquake of Central Chile, monitored by GPS* Vigny et al. found that the rise time along the Maule fault was a uniform 21 seconds, and that the maximum horizontal displacement during the event was 5m. To simulate the displacements associated with this event the box was pushed from north to south and then from west to east at distances of 1m, 1.5m, 2m, and 4m with a 15-minute pause between each push. The box was pushed at a rate of 0.25 m/s, which we arrived at by dividing the maximum horizontal displacement of the Maule earthquake by its rise time. QED X messages were collected from an EEW system mounted to the top of the box, and an EEW system installed inside of the box. The displacement test was repeated with the window of the box facing each of the four cardinal directions.

After the data was collected, it was analyzed using R to create a time series of the GNSS positions. The latitude and longitude values were converted from decimal degrees to UTM coordinates following the same method used in the difference analysis. The mean latitude and longitude for each push (north to south and east to west) were calculated, and then subtracted from each reported latitude and longitude value to demean the data. The displacement time series are show below (Figures 35-38). The red vertical lines mark the times when the box was pushed. The 4m and 2m Latitude displacements are clear in all of the tests, and the 1m and 1.5m displacements can be seen in the north facing run, the west Facing run, and the east facing run. There was more variability in the reported latitudes during the south facing run, and the 1m and 1.5m displacements are lost the noise.

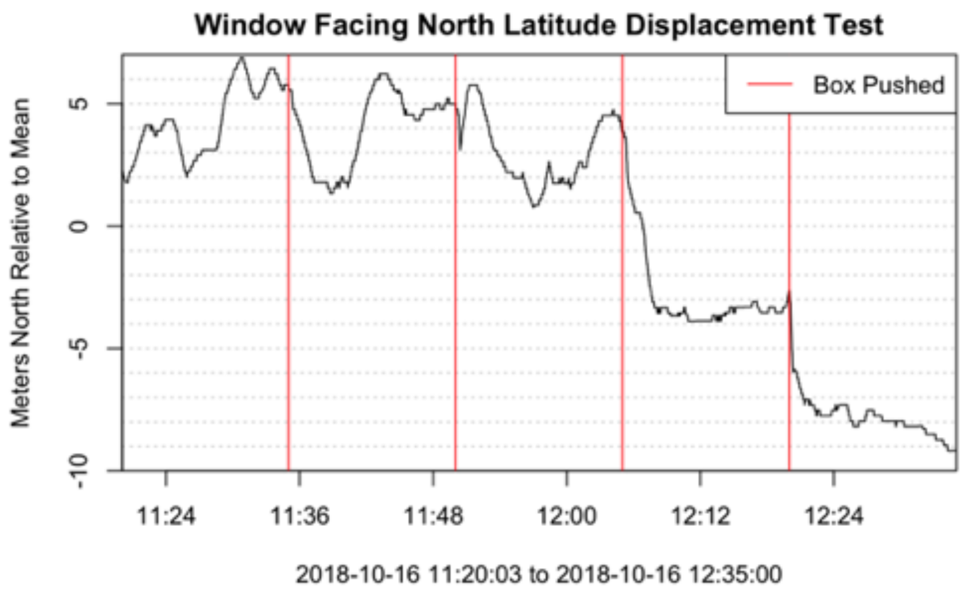


Figure 34 Latitude displacement time series pushed with the window facing North recorded by the external UBLOX GNSS chip.

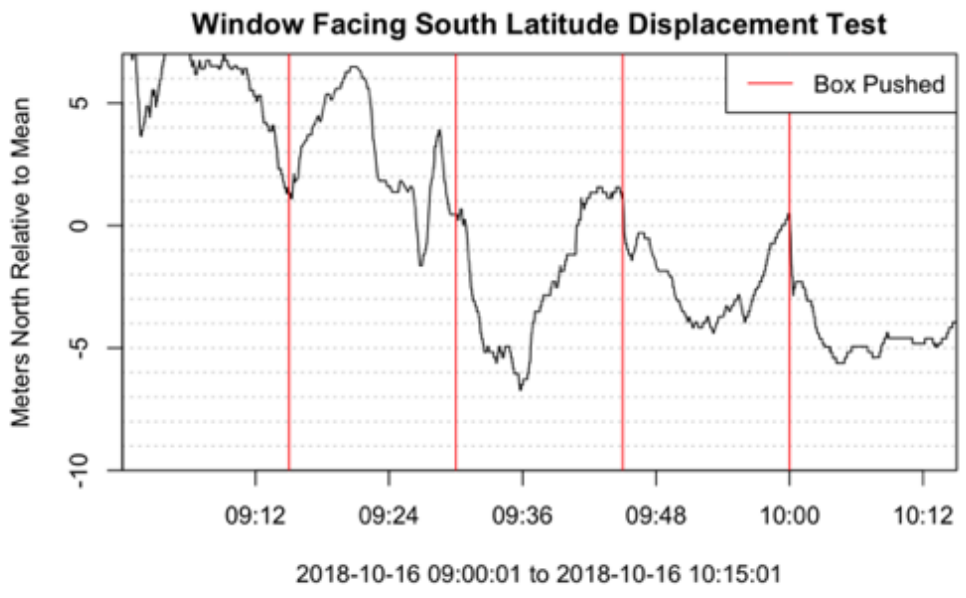


Figure 35 Latitude displacement time series pushed with the window facing South recorded by the external UBLOX GNSS chip.

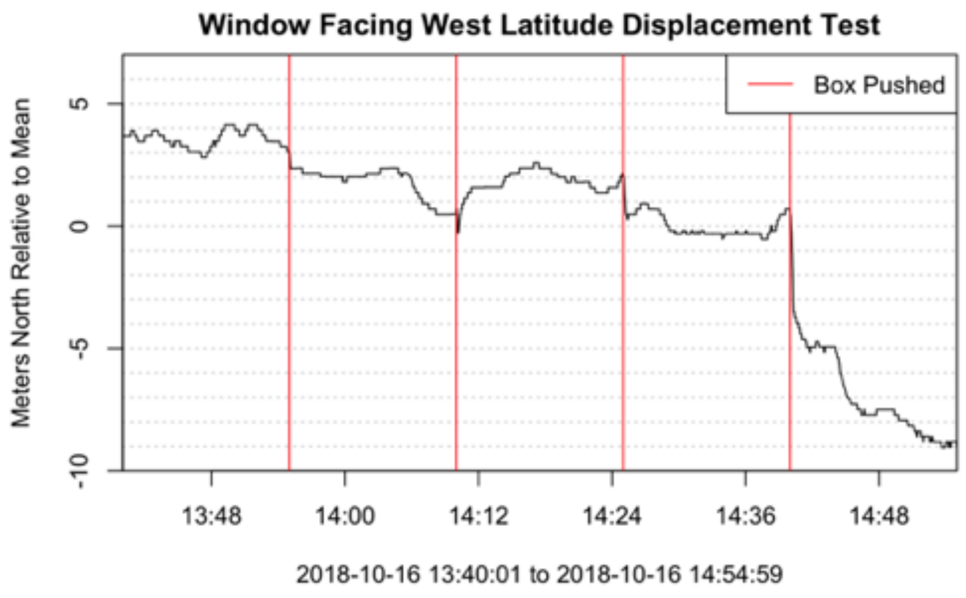


Figure 36 Latitude displacement time series pushed with the window facing West recorded by the external UBLOX GNSS chip.

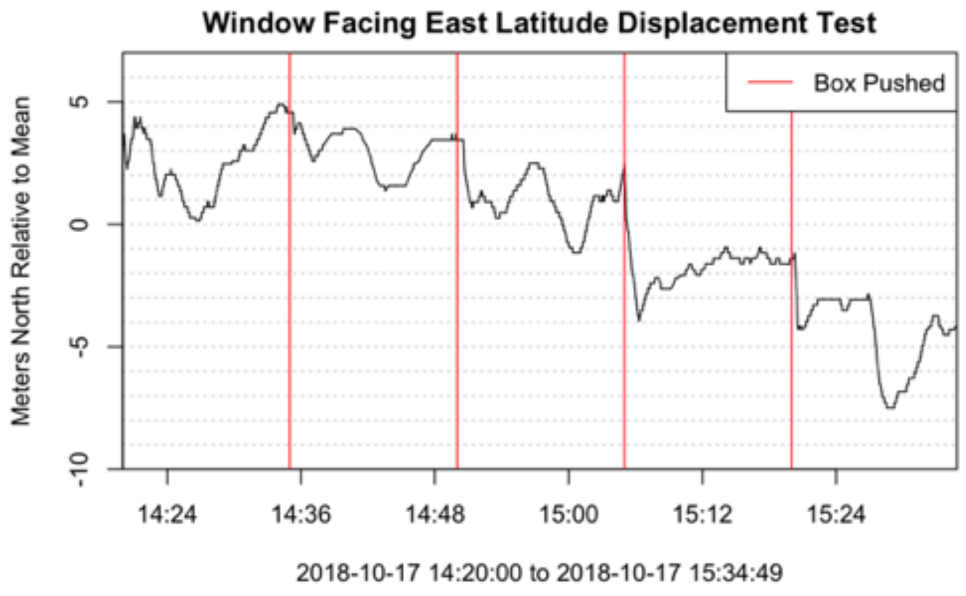


Figure 37 Latitude displacement time series pushed with the window facing East recorded by the external UBLOX GNSS chip.

Earthquake Static Ground Displacement Detection Model

To test how well we would be able to detect real displacements, the results of the difference test were used as inputs in a simple decision tree classification model: $if\ |lat_{n+l} - lat_n| > \frac{d}{1000}\ then\ Displacement\ is\ Real$, to test how well the real displacements can be detected in the data (See Appendix V for the code). Where l is the lag in seconds and d is four standard deviations from the mean of the latitude difference at lag l. Data from each of the displacement tests were run through the classification model using the d values calculated earlier in the drift test for the UBLOX GNSS chip inside the box at lags of 1 through 60 seconds. The displacements that the classification model detected as real displacements are plotted over the timeseries below (Figures 39-42).

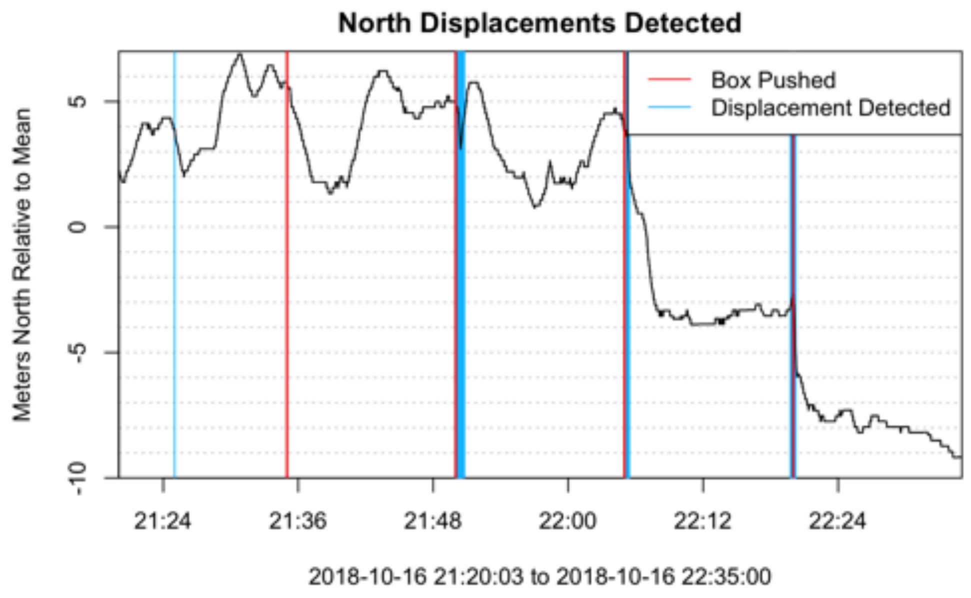


Figure 38 Detected displacements while the window was facing North. There was only one false positive displacement at the beginning of the test. The displacement model didn't pick up a 1 m push at the beginning of the test.

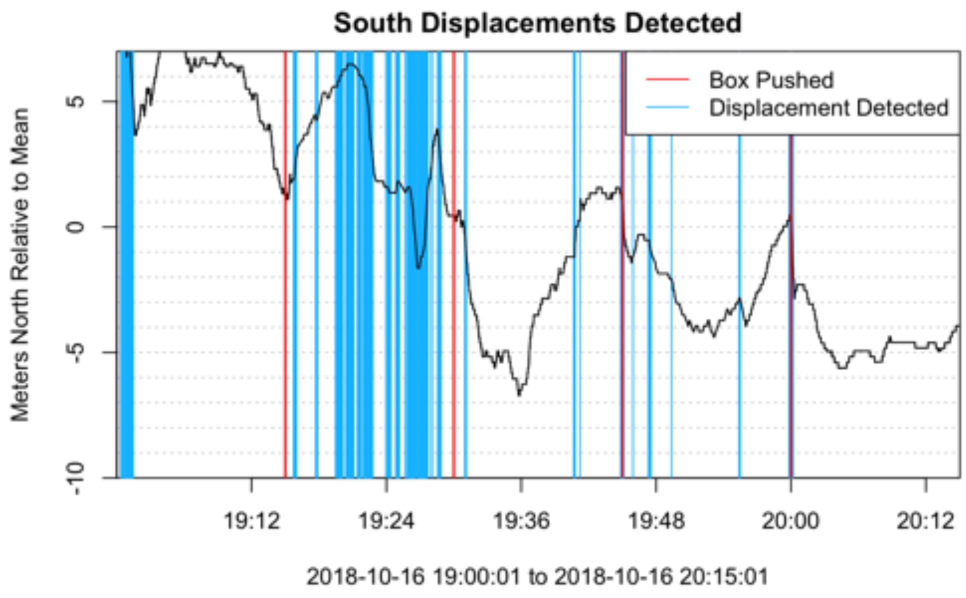


Figure 39 Detected displacements while the window was facing South. There were much more false positives during this test. The 1 m, 2 m, and 4 m, displacements were detected by the model however.

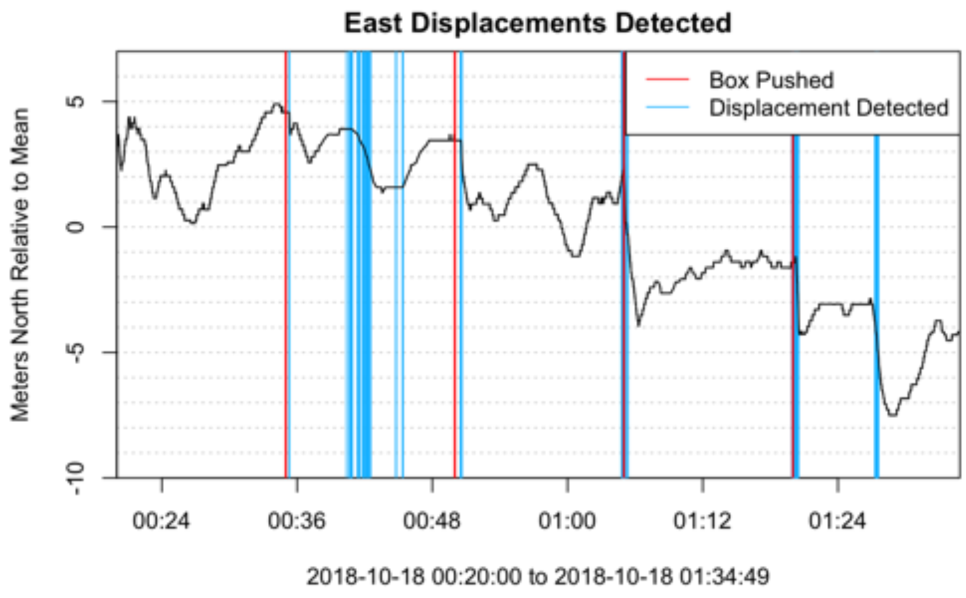


Figure 40 Detected displacements while the window was facing East. There were several false positives, but all four displacements were detected.

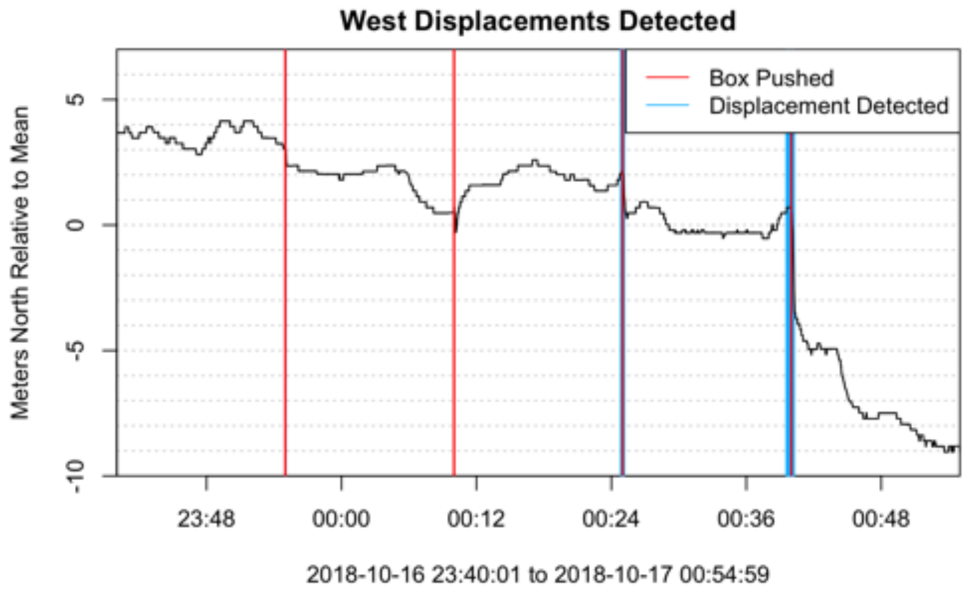


Figure 41 Detected displacements while the window was facing West. There were no false positives, and displacements were detected during the 1 m, 2 m, and 4 m displacements.

The classification model was able to detect all four latitude displacements in the test with the window facing east, the last three latitude displacements in the test with the window facing north, and first and last two displacements in the test while the window was facing west, and the last two displacements in the test with the window facing south. There were several false positives in the displacement run with the window facing east, and many false positives in the displacement run with the window facing south.

The displacement test data was processed in R to pull out the displacements that the UBLOX GNSS receiver measured during the real displacements. The full displacement cannot be measured while the period of the displacement is larger than the lag that we are measuring over. To make sure that the full measured displacement was pulled out, we loop over the displacement data from UBLOX GNSS chip at lags from 1 to 60 seconds over a two-minute interval centered on the time that the box was pushed. We then generated plots of the measured displacements at each lag, and output the real measured displacement (Figures 43-55) (See Appendix V for the code).

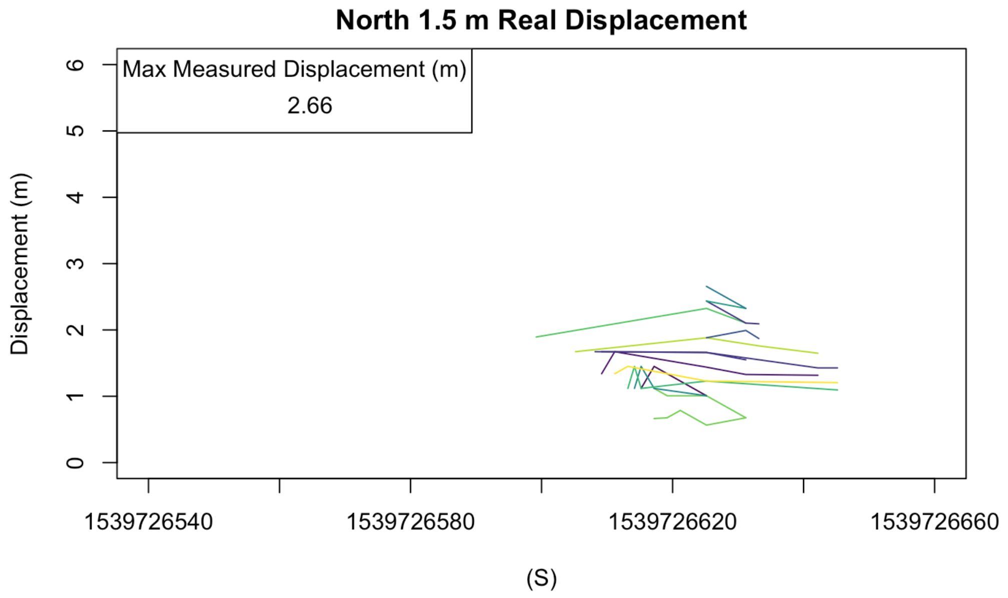


Figure 42 Measured displacements from the UBLOX GNSS chip while the window was facing North during the 1.5 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 2.66 m.

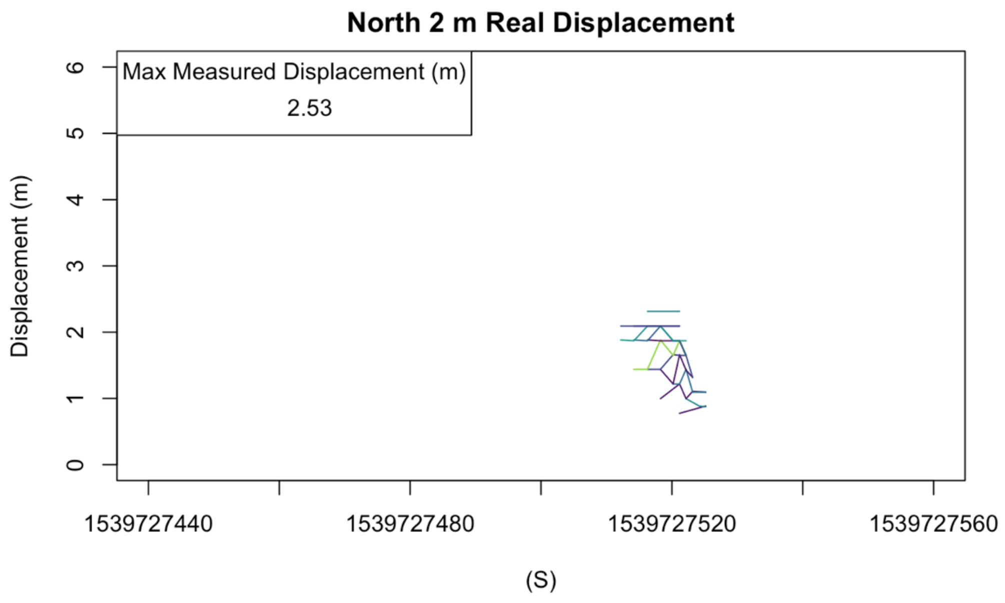


Figure 43 Measured displacements from the UBLOX GNSS chip while the window was facing North during the 2 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 2.53 m.

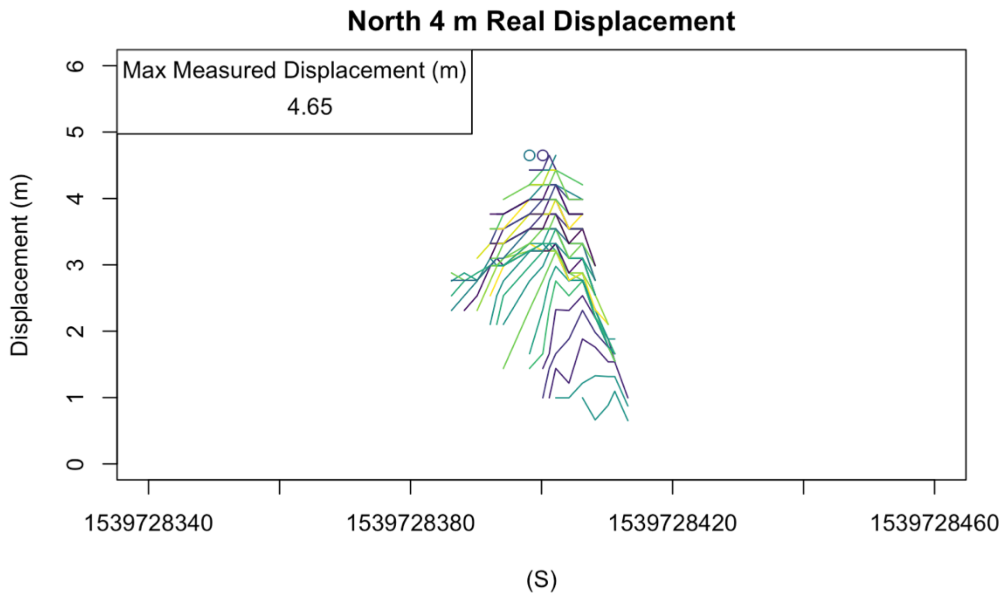


Figure 44 Measured displacements from the UBLOX GNSS chip while the window was facing North during the 4 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 4.65 m.

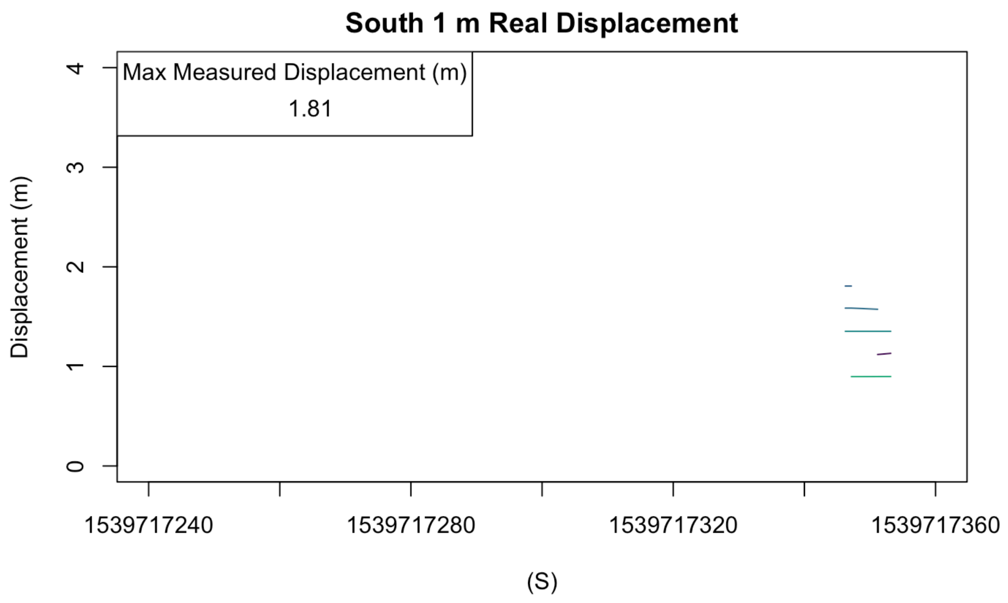


Figure 45 Measured displacements from the UBLOX GNSS chip while the window was facing South during the 1 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 1.8 m. These results are a false positive*.*

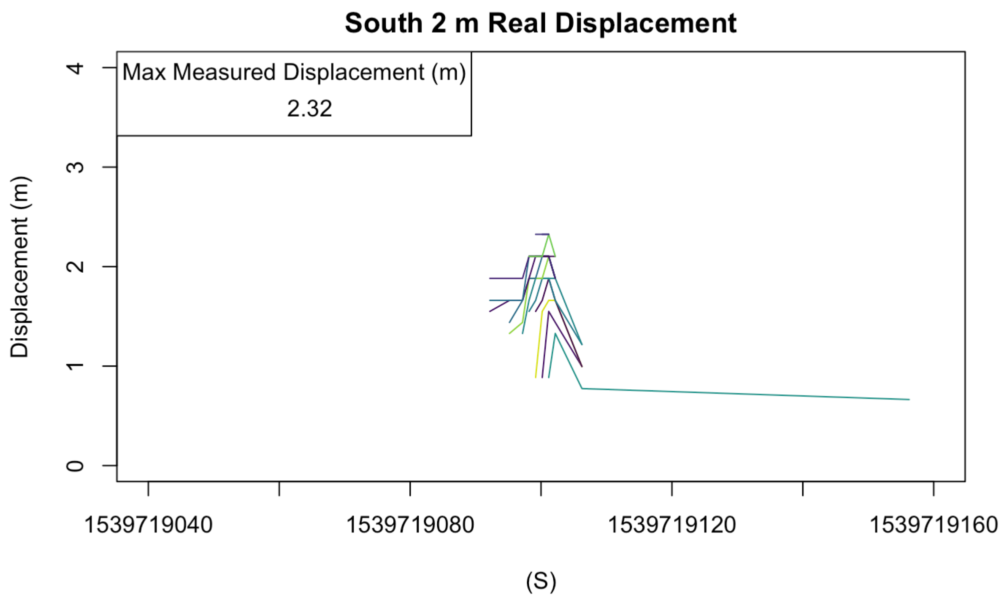


Figure 46 Measured displacements from the UBLOX GNSS chip while the window was facing South during the 2 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 2.32 m.

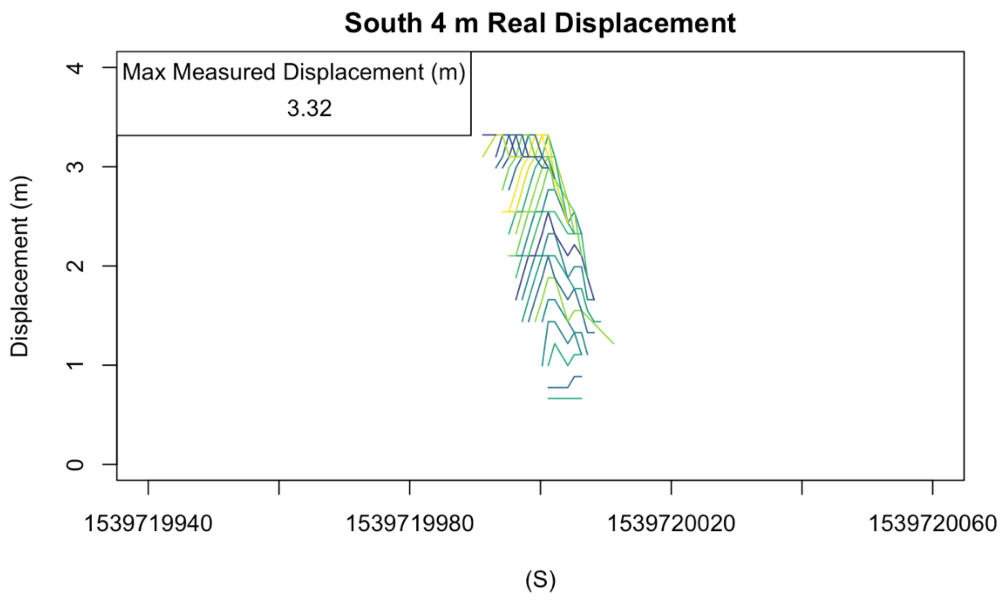


Figure 47 Measured displacements from the UBLOX GNSS chip while the window was facing South during the 4 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 3.32 m.

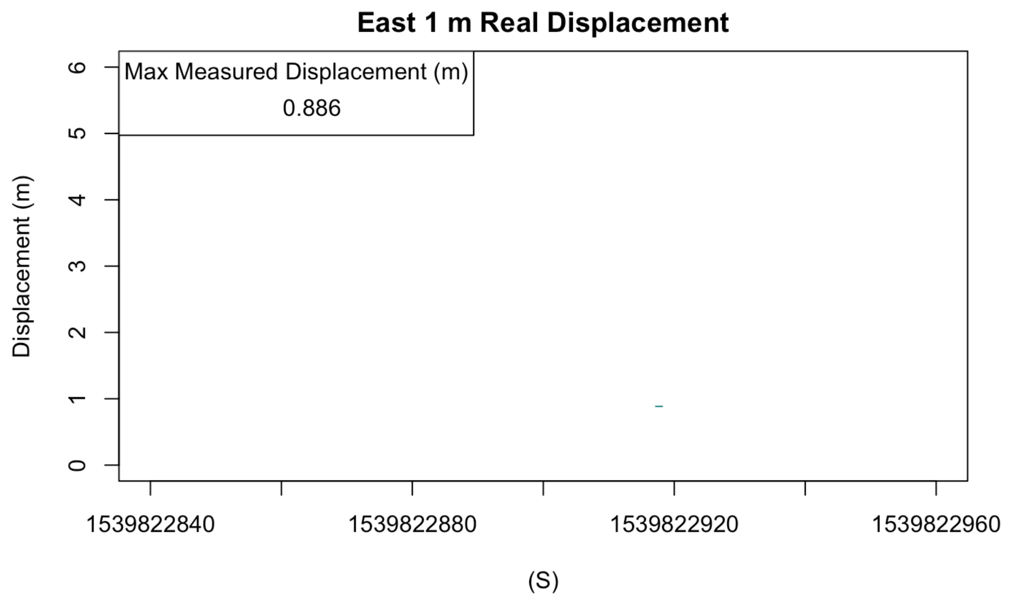


Figure 48 Measured displacements from the UBLOX GNSS chip while the window was facing East during the 1 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 0.89 m.

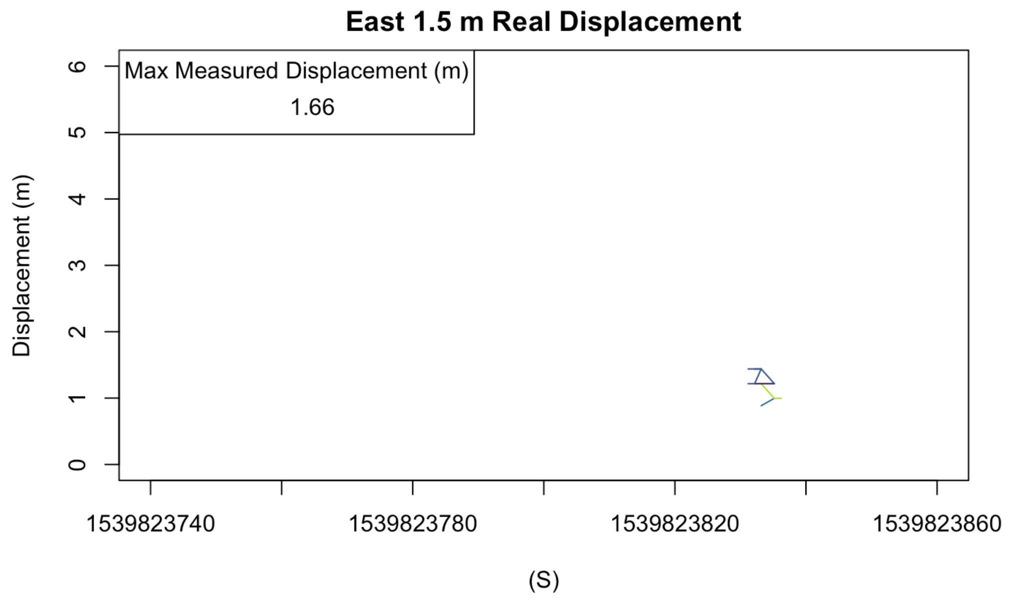


Figure 49 Measured displacements from the UBLOX GNSS chip while the window was facing East during the 1.5 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 1.66 m.

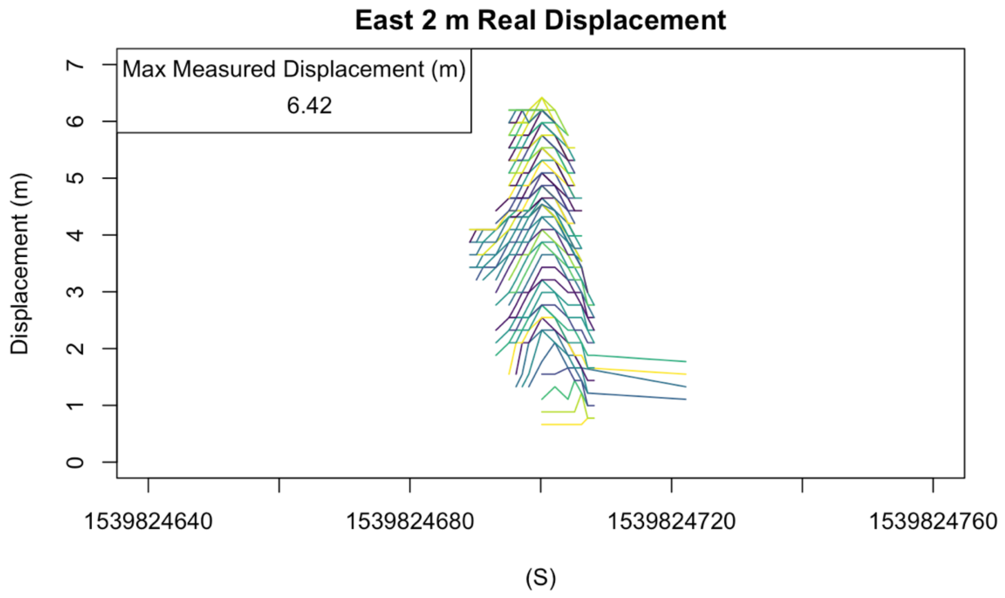


Figure 50 Measured displacements from the UBLOX GNSS chip while the window was facing East during the 2 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 6.42 m. This was the largest outlier seen during the test.

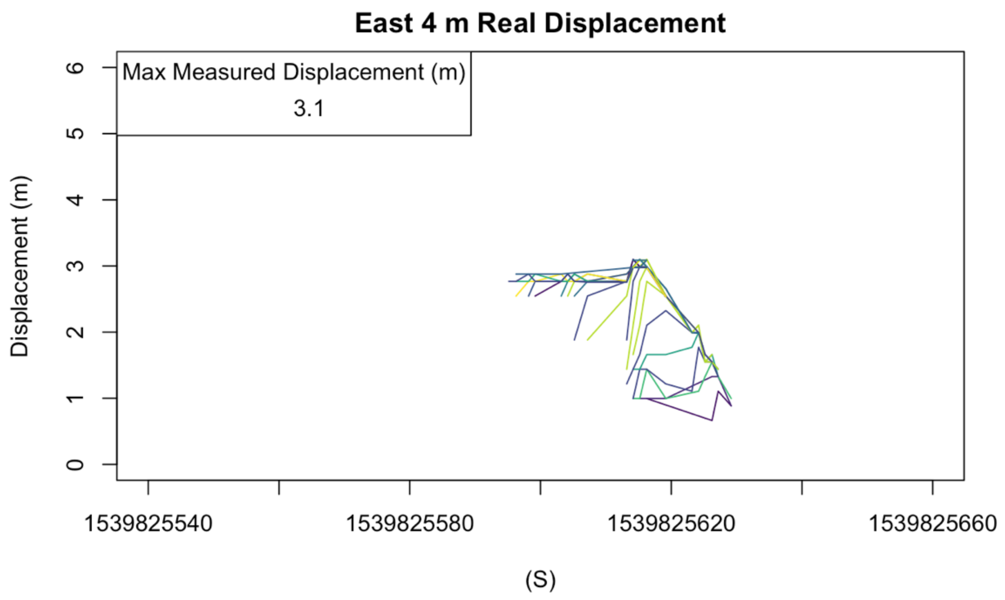


Figure 51 Measured displacements from the UBLOX GNSS chip while the window was facing East during the 4 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 3.02 m.

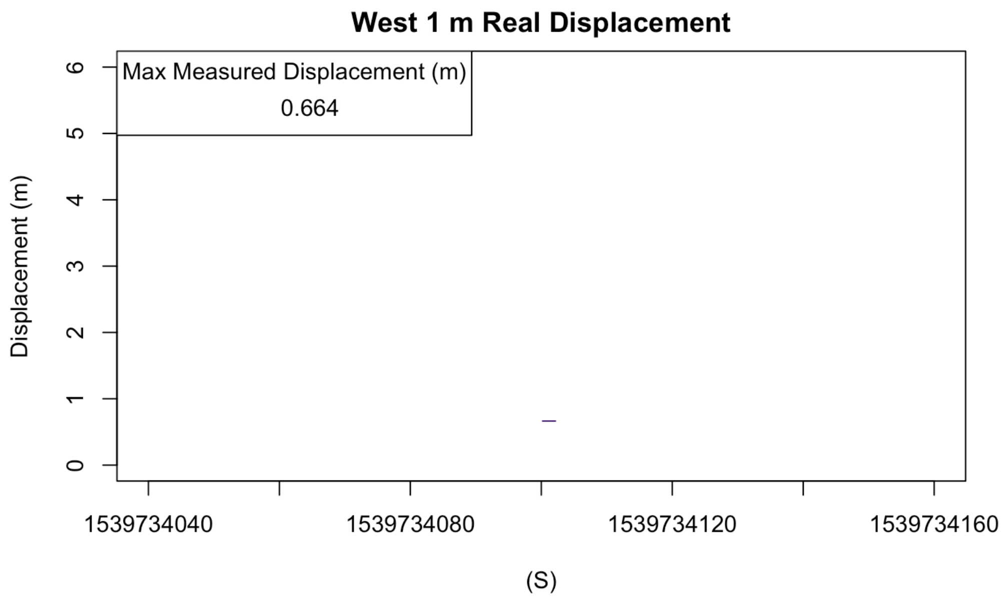


Figure 52 Measured displacements from the UBLOX GNSS chip while the window was facing West during the 1 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 0.66 m.

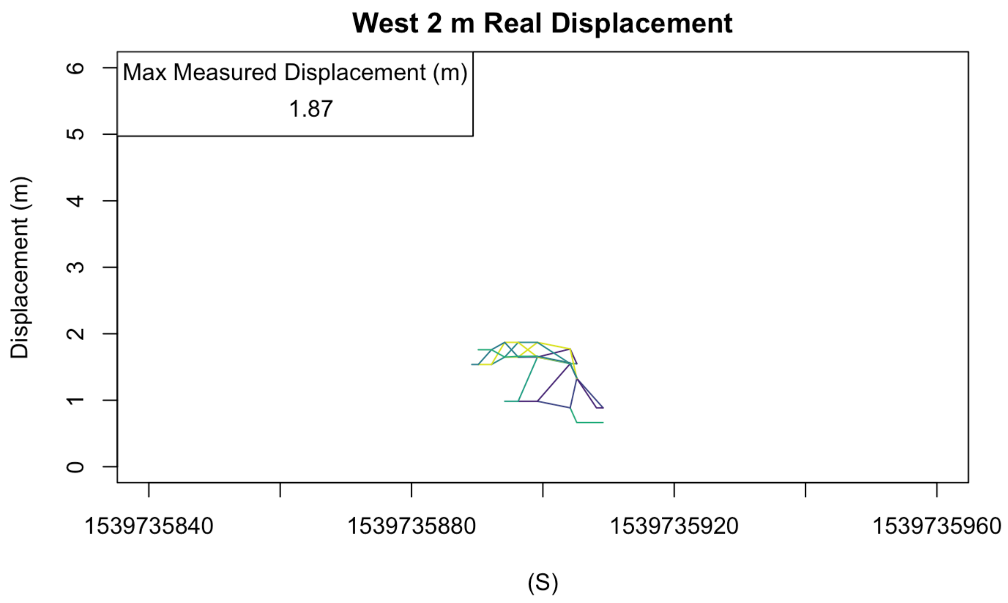


Figure 53 Measured displacements from the UBLOX GNSS chip while the window was facing West during the 2 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 1.87 m.

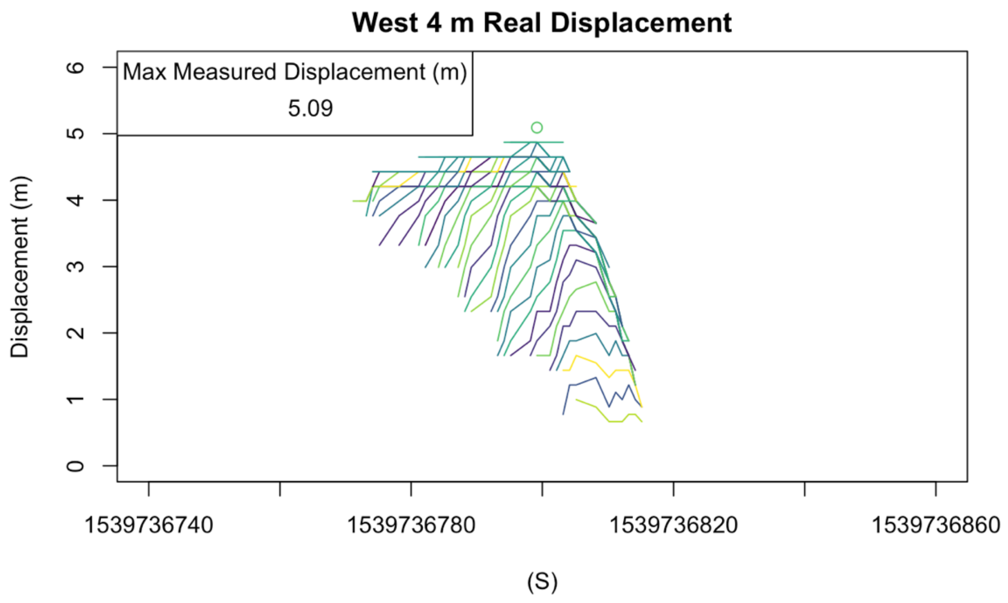


Figure 54 Measured displacements from the UBLOX GNSS chip while the window was facing West during the 4 m real displacement. The max measured displacement after looping through lags of 1-60 seconds was 5.09 m.

Table 2 shows the real latitude displacements and the corresponding latitude displacement estimates that were measured during each of the four displacement tests. Only the 2m and 4m displacements were detected in all four tests. The UBLOX GNSS chip measured an average displacement of 1.11 m, 2.16 m, 3.29 m, and 4.04 m during the real 1 m, 1.5 m, 2 m and 4 m displacement tests respectively. If we throw out the outlier value of 6.42 m during the East facing 2 m displacement the average measured displacement during the real 2m displacement test comes down to 2.24m.

Real Disp. (m)	North Disp. (m)	South Disp. (m)	East Disp. (m)	West Disp. (m)
1	NA	NA	0.89	0.66
1.5	2.66	NA	1.66	NA
2	2.53	2.32	6.42	1.87
4	4.65	3.32	3.10	5.09

Table 3 Real vs measured displacements during the 4 displacement tests.

Analysis of the Detection Model's Performance

The confusion matrix below (Table 3) shows the performance of our simple decision tree classification model in picking out the real displacements. When we run the model over all the data, the true positive rate, or recall, of the model is ~97%, however the precision of the model is only ~42%. This means that we are detecting almost all of the real displacements. However, we are also getting a lot of false positive results, which means that when we predict there has been a displacement we are only correct 42% of the time. This is largely due to the large number of false positives in the displacement tests while the window was facing south and east. The lack of large numbers of false positives in the other displacement tests suggests that we could improve the precision of the model by cross referencing positive results with other nearby stations. With a dense network of phones, we could run a more sophisticated version of the model to detect displacements in real time.

Another option would be to go back and apply the classification model to the data after accelerations past a certain threshold have been detected – indicating a large earthquake has actually happened. Applying the model over smaller time windows would decrease the likelihood of false positive results, and improve the precision of the model. To test how well the model would perform in this scenario we went back and applied the model over 2-minute time windows centered around the times we know the box was pushed. As expected, the model performed much better. Table 4 shows the confusion matrix for the model applied over the shorter 2-minute time windows. Over these smaller time windows, the recall of the model stayed the same, ~97%, but the precision of the model increased from ~42% to ~98%.

N: 635,639	Predicted No	Predicted Yes
Actual No	630,369	3,007
Actual Yes	56	2,208

Table 4 Confusion matrix for the displacement classification model when looking at all of the data. Over this timescale the model has an accuracy of 99.5%, a true positive rate (recall) of 97.5%, a false positive rate of 0.4%, and a precision of 42.3%.

N: 5215	Predicted No	Predicted Yes
Actual No:	2,918	34
Actual Yes:	56	2,207

Table 5 Confusion matrix for the displacement classification model when only looking at 2-minute time windows centered around the times we know the box was pushed. Over this timescale the model has an accuracy of 98.2%, a true positive rate (recall) of 97.5%, a false positive rate of 1.5%, and a precision of 98.4%.

We ran four displacement tests for each real displacement (1 m, 1.5 m, 2 m, 4 m), and the UBLOX GNSS chip had four chances to detect and measure the same displacement. Taking the average of the reported displacements from each of the 4 phones allowed us to get within 0.04-1.28 m of the real displacement. If we remove the outlier displacement measured by the UBLOX GNSS chip during the 2 m push while the window was facing East (6.42 m), we would have been within 0.04-0.66 m of the real displacements. If we only look at the displacements that were detected by all four of our sensors, the 2m and 4m displacements, we would have been within 0.04-0.24 m of the real displacements.

This is a small sample size, and more work would need to be done to better constrain the real accuracies that would be possible to attain by average the displacements measured by multiple units. However, these initial positive results suggest that if we were to install groups of our smartphone EEW sensors in a location with external UBLOX GNSS we would be able to average their measured displacements to get a quick and accurate measurement of the real displacement to help constrain the magnitude of the earthquake. During our test we were able to detect displacements as small as 1 m. The max horizontal displacement measured during the 2012 Mw 7.6 Nicoya earthquake and the 2014 Mw 8.2 Iquique earthquake were ~60 cm and ~90 cm respectively. This suggests that the lower magnitude limit that we would be able to detect with our equipment would be around Mw 8.0+. Although we did not test the ability of the UBLOX GNSS chip to detect displacements smaller than 1 m, we would expect that as long as the slope of the displacement rate is larger than the slope of the drift in the UBLOX GNSS chip we would be able to detect the displacement with our model, and it is possible that we would be able to detect displacements from smaller earthquakes.

Longitude Data Problem

During the displacement test it became clear that there was an issue with the longitude values that were being sent in the QED X and I messages (Figure 56). The smallest change in longitude that we were able to measure was 1m. The problem was the QED app was sending the longitude and latitude values as 4-bit doubles which truncated the longitude value after the fifth decimal place. Chris Duncan created a new version of QED that sends a 16-bit double for the longitude and latitude values and this fixed the problem. This is why an analysis of the longitude data was not included in the GNSS drift and displacement detection sections.

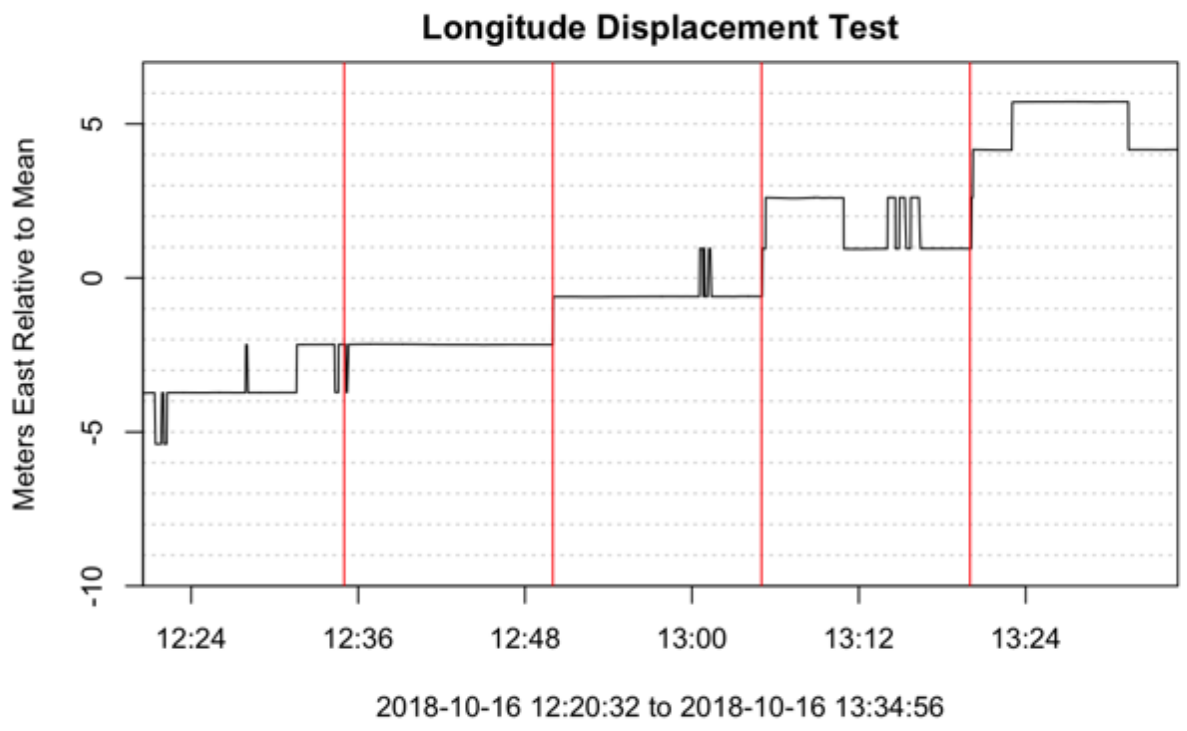


Figure 55 Longitude displacement time series. The smallest difference in latitude that we were able to measure was 1 m. This problem was tracked down to how QED was reporting the longitude value. QED was originally programmed to send the longitude data as 4-bit doubles. This was cutting off the last few decimal places of the longitude value leaving us with a 1 m resolution. After this issue was discovered a new version of QED was built that sent longitude values as 16-bit doubles which fixed the problem.

Internal Vs External GPS Displacement Roof Test

All our EEW installations in Costa Rica are wall type installations, and do not include an external UBLOX receiver. It has been shown that GNSS data from a smartphone with an onboard Kalman filter should be able to detect displacements from very large earthquakes (Mw 8+ events) [Minson et al. 2015]. In the difference tests we ran, the phone's internal GNSS chip installed inside the box performed similarly to the external UBLOX GNSS chip installed inside the box, which suggests that we should be able to detect displacements from large earthquakes with the internal GNSS chips in the phones. To compare the performance of the external UBLOX GNSS chip with the internal GNSS chip we ran a second displacement test on the roof of HIG.

This time the box was pushed distances of 1 m, 2 m, 4 m, and 8 m from North to South, and then from West to East at the same rate of 0.25 m/s that we calculated from the 2011 *Vigney et al.* paper. Between pushes the box was stationary for 30-minutes. The new version of QED with the fix for the 4-bit longitude issue was installed before the test was run, and we ran an analysis of both the latitude displacements and the longitude displacements. Directionality did not seem to have an impact on the performance of the system in the previous displacement and difference tests, and this displacement test was only one run with the window facing south.

The QED X messages and I messages from one of our sensors installed inside the box were collected and processed in R following the same method as the original displacement tests. The latitude and longitude time series for both message streams were plotted with red vertical lines indicating when the box was pushed, and blue lines indicating real displacements detected by the detection model. The results from the phone's internal GNSS chip difference tests were used as inputs for the detection model to pick out the real displacements recorded by the internal GPS chip.

Results of Internal vs External Analysis

The data from the phone's internal GNSS chip appears to be heavily filtered. The displacement time series for the internal GNSS chip shows that the GNSS chip is reporting the same latitude or longitude for long periods of time, which is not seen in the time series for the UBLOX GNSS chip. The 8 m displacement can be seen in both the latitude and longitude displacement time series for the internal GNSS chip, and the 2m displacement can be seen in the latitude displacement time series for the internal GNSS chip (Figures 57-58). All four of the displacements can be seen in both the latitude and longitude data from the external UBLOX, and the detection model was able to pick out all of the displacements as real displacements with only a few false positives (Figures 59, 64). The UBLOX GNSS chip measured latitude displacements of 1.2 m during the 1m displacement, 1.01 m during the 2m displacement, 4.8 m during the 4m displacement, and 7.08 m during the 8m displacement (Figures 60-63). The UBLOX chip measured longitude displacements of 0.92 m during the 1m displacement, 1.59 m during the 2m displacement, 3.44 m during the 4m displacement, and 7.64 m during the 8m displacement (Figures 65-68).

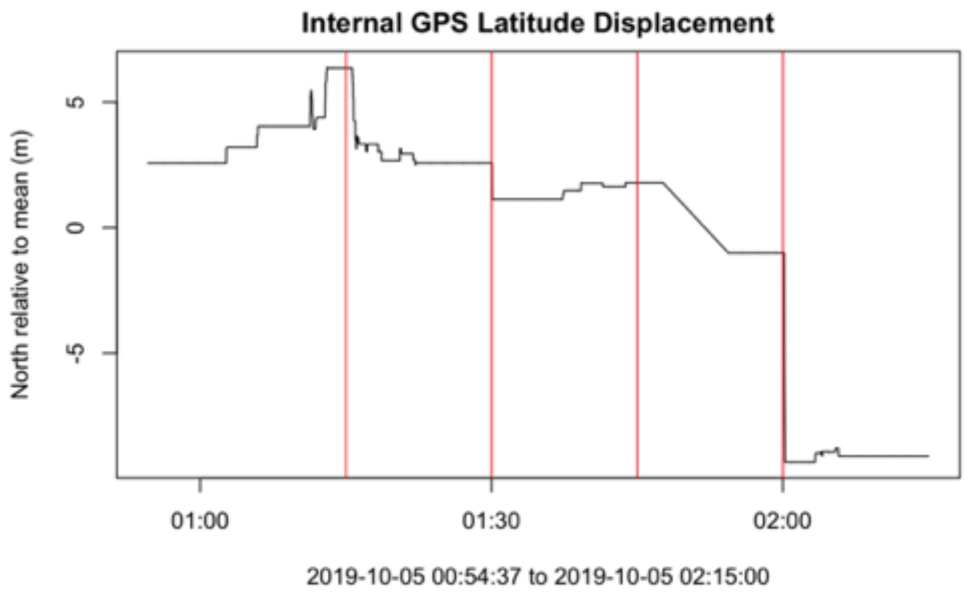


Figure 56 Latitude displacement time series for the phones internal GNSS chip. The red lines denote the times when the box was actually pushed. The internal GNSS chip is filtering the data to keep the reported position as consistent as possible. This is useful for navigation apps that are installed on phones, but means that we would likely miss the displacements from an earthquake. During this displacement test the box was pushed distances of 1 m, 2 m, 4 m, and 8 m. Only the 2 m and 8 m displacements were picked up during the test.

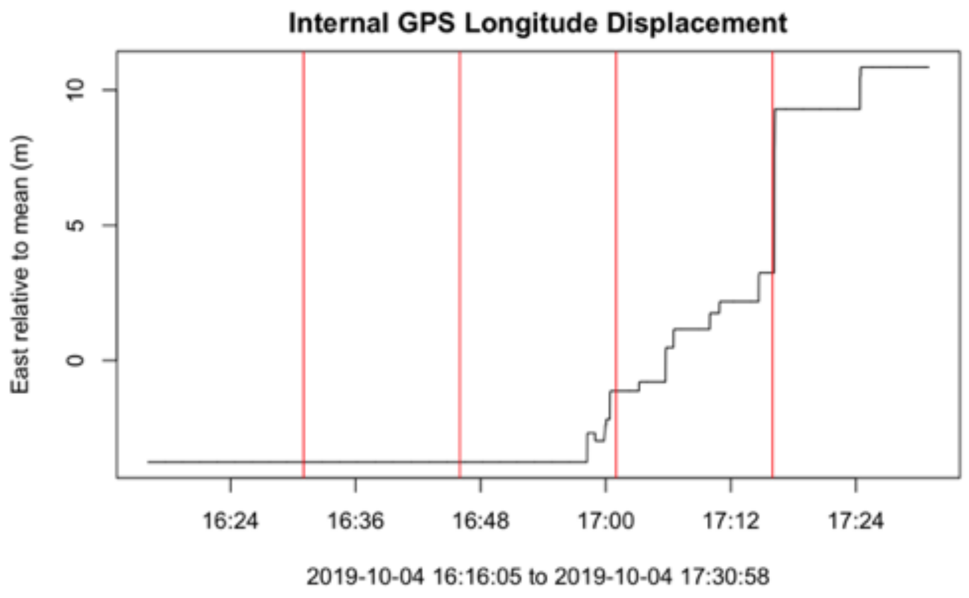


Figure 57 Longitude displacement time series for the phones internal GNSS chip. The red lines denote the times when the box was actually pushed. The internal GNSS chip is filtering the data to keep the reported position as consistent as possible. During the longitude displacement test the box was pushed 1 m, 2 m, 4 m, and 8 m. Only the 8 m displacement was detected during the longitude displacement test.

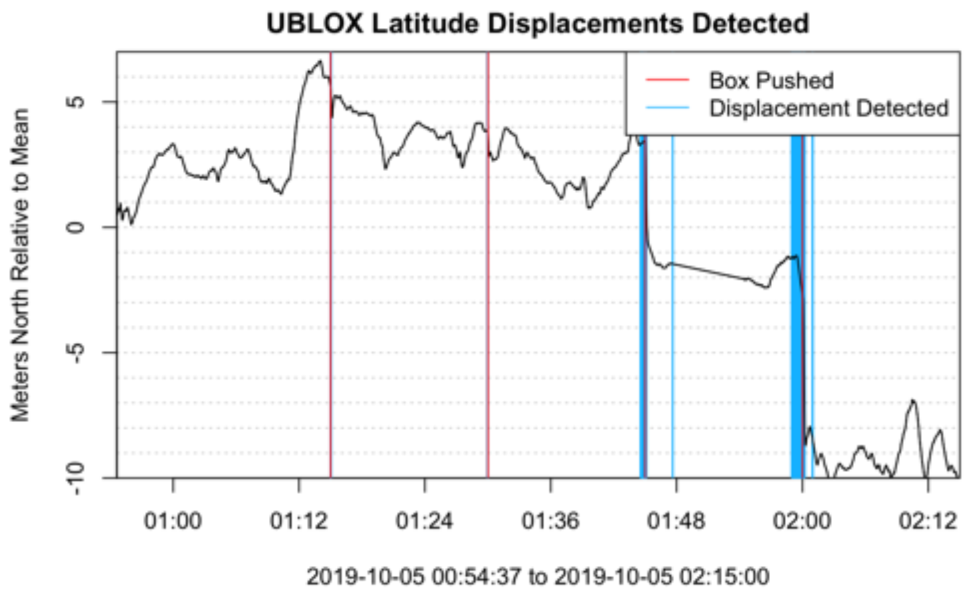


Figure 58 The UBLOX GNSS chip latitude displacement time series. The displacement detection model was run over this second latitude displacement run, and all 4 displacements of 1 m, 2 m, 4 m, and 8 m, were detected with only a couple of false positives.

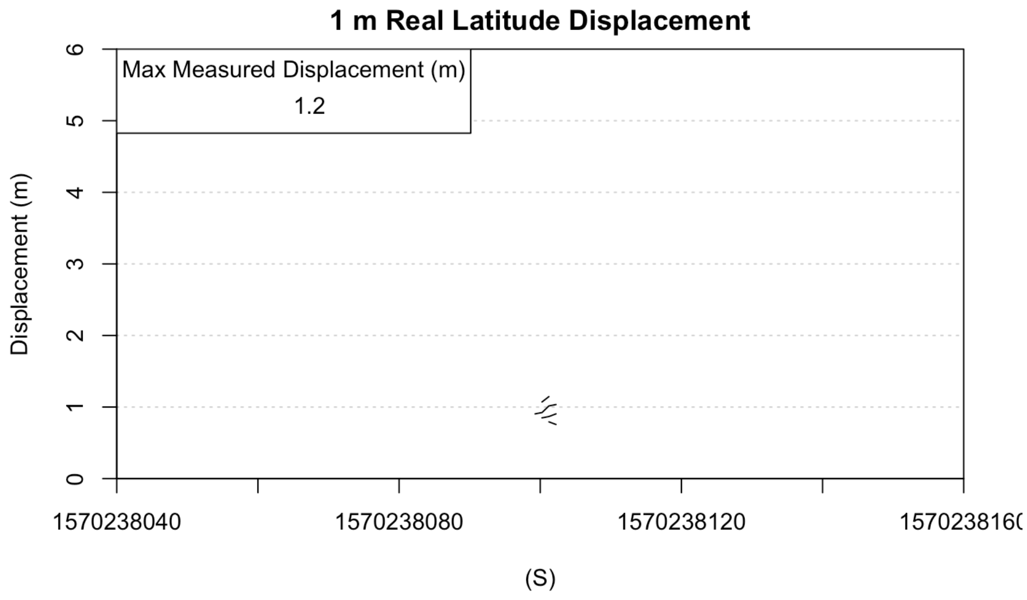


Figure 59 The measured latitude displacement measured by the UBLOX GNSS chip during the real 1m displacement. The max measured displacement was 1.20 m.

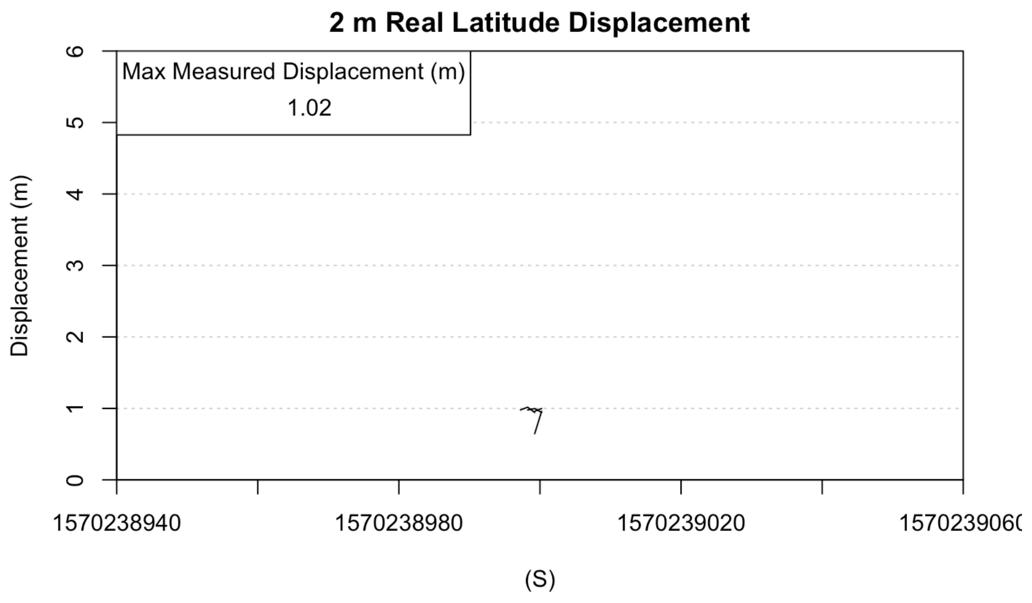


Figure 60 The measured latitude displacement measured by the UBLOX GNSS chip during the real 2 m displacement. The max measured displacement was 1.02 m.

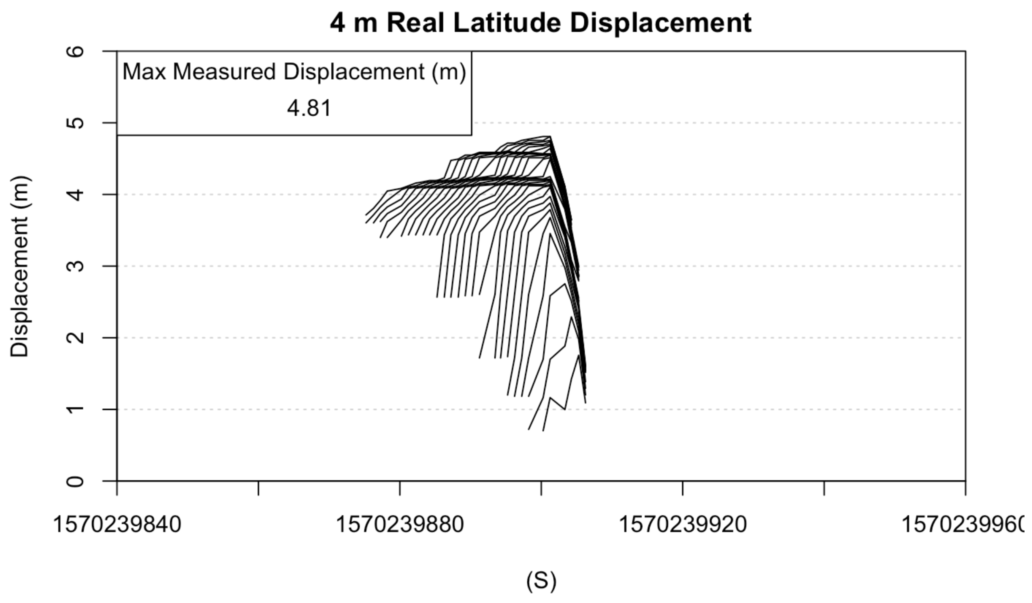


Figure 61 The measured latitude displacement measured by the UBLOX GNSS chip during the real 4 m displacement. The max measured displacement was 4.81 m.

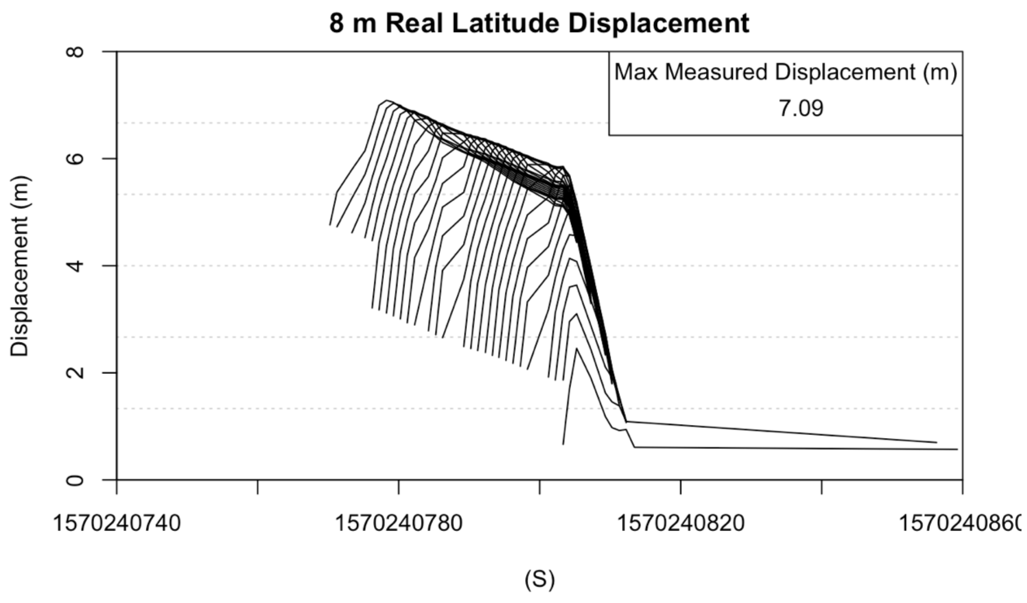


Figure 62 The measured latitude displacement measured by the UBLOX GNSS chip during the real 8 m displacement. The max measured displacement was 7.09 m.

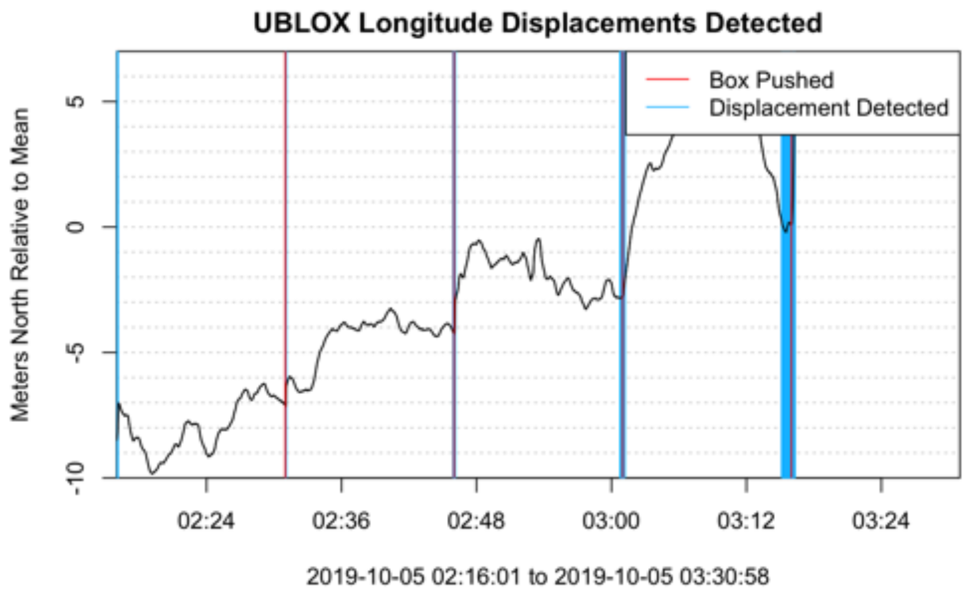


Figure 63 The UBLOX GNSS chip longitude displacement time series. The displacement detection model was run over this second latitude displacement run, and all 4 displacements of 1 m, 2 m, 4 m, and 8 m, were detected with only a couple of false positives at the beginning of the test.

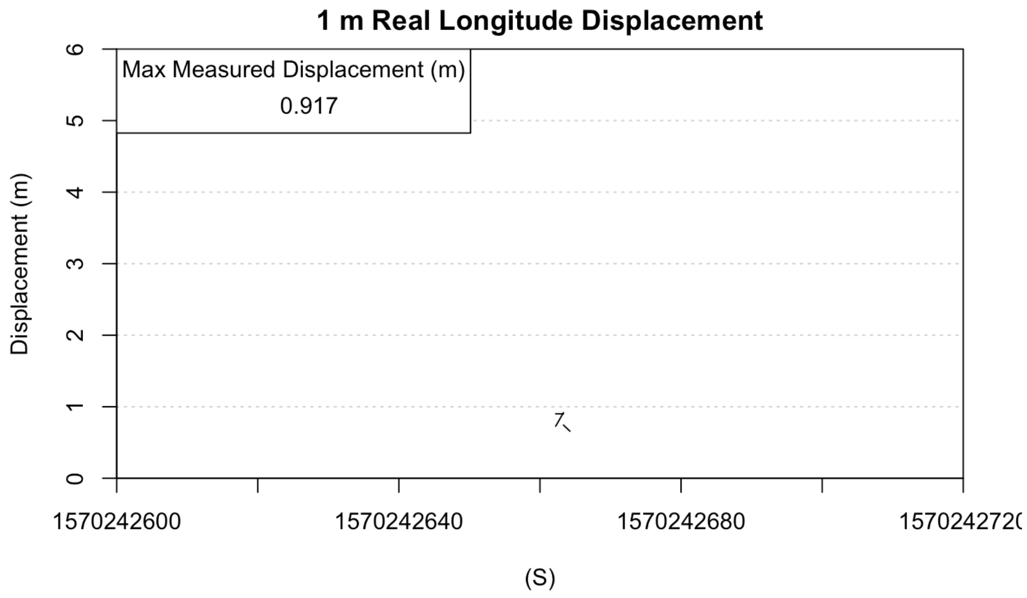


Figure 64 Measured longitude displacements during the real 1 m displacement by the UBLOX GNSS chip. The max longitude displacement that it measured was 0.91 m.

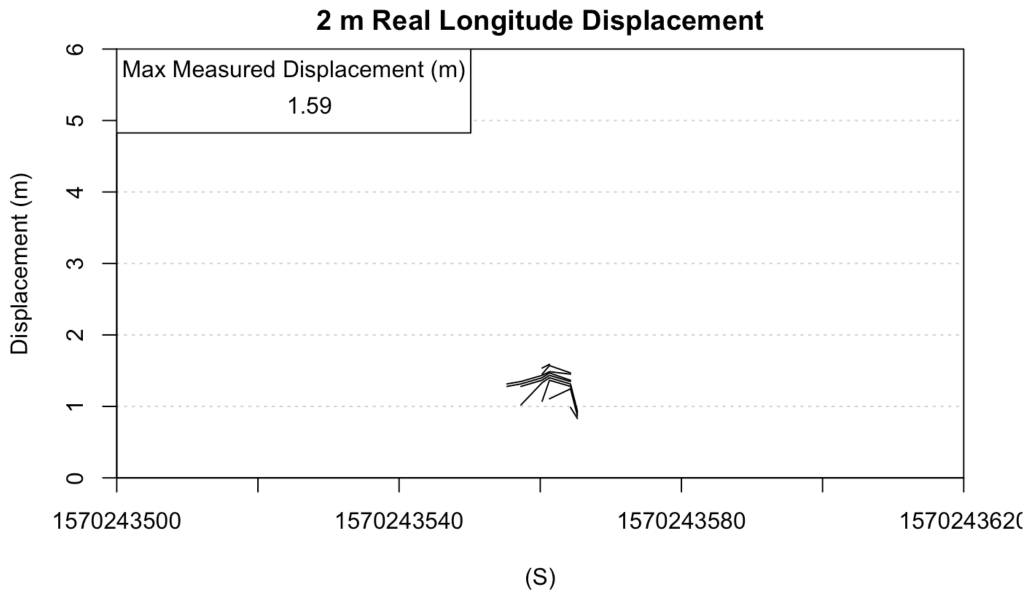


Figure 65 Measured longitude displacements during the real 2 m displacement by the UBLOX GNSS chip. The max longitude displacement that it measured was 1.59 m.

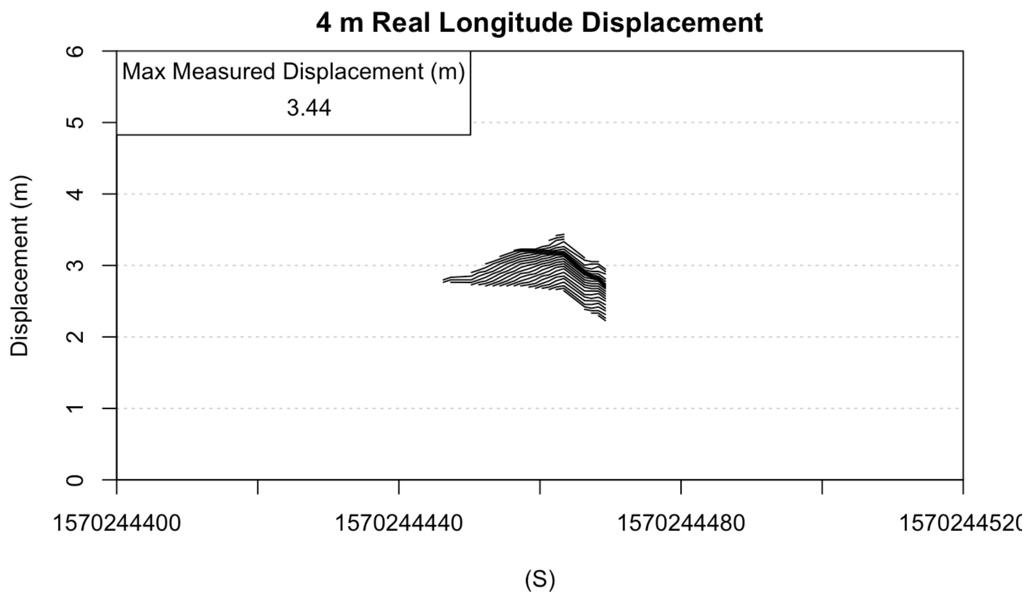


Figure 66 Measured longitude displacements during the real 4 m displacement by the UBLOX GNSS chip. The max longitude displacement that it measured was 3.44 m.

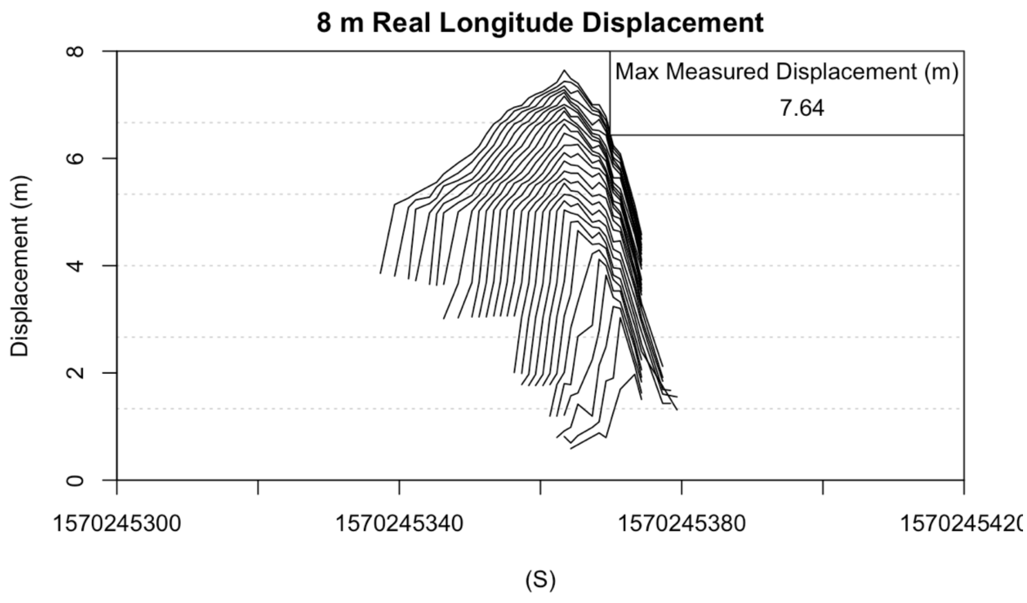


Figure 67 Measured longitude displacements during the real 8 m displacement by the UBLOX GNSS chip. The max longitude displacement that it measured was 7.64 m.

Discussion

The results of the difference tests we ran for the drift in latitude and altitude showed that the phone's internal GNSS performed at the same level as the UBLOX GNSS chip. The small drift in the internal GNSS chip's reported latitude and altitude that we measured are due to the Kalman filter that the raw GNSS data is passed through. This Kalman filter is most likely designed to improve the performance of the phone's internal GNSS chip for navigational apps, and it resists outputting small changes in position in an attempt to keep the reported position more consistent. The data from the phone's internal GNSS chip is too heavily filtered to be able to detect displacements, and we would not be able to use our current Costa Rica EEW system to accurately detect and measure the displacements of earthquakes. New smartphones, like the Xiaomi Mi8, are coming out with dual frequency GNSS chips that should perform similarly to the UBLOX chip we used in our analysis. Combining these phones with our EEW apps in the future should allow us to detect the displacements from earthquakes in the future without the need for the external UBLOX GNSS chip.

Effectiveness of Our EEW Networks

Latencies in Our Chile and Costa Rica EEW Networks

The effectiveness of any EEW system depends on its ability to deliver a warning to a population before the strong ground shaking reaches them. In their 2018 paper *The limits of earthquake early warning: Timeliness of ground motion estimates* Minson et al. examined the possible warning times that users would receive in an ideal case of an EEW system with zero latency. They noted that both large and small earthquakes look the same at nucleation, and therefore the only way that an EEW network can know that an earthquake is large is to continue to monitor the earthquake's moment release. They found that warning time, which they define as the time between when the alarm was sent and when the strong ground motion reaches a population center, depends on the ground shaking thresholds that users set. Users that want to receive a warning at low ground acceleration thresholds (2% g) could have warning times greater than 1 minute. Users that have a low tolerance for false alarms, and only want to receive warnings when strong ground motions (20% g) are predicted would have less than 10 seconds of warning time. These warning times do not improve for users that are further away from the epicenter because larger earthquakes are needed to cause ground accelerations of these thresholds at distances farther away from the fault, and the event would need to be monitored for a longer period of time to verify that the earthquake had grown large enough to cause shaking at these thresholds. This means that large latencies in EEW systems could render them ineffective.

To analyze the effectiveness of our EEW system, the latencies associated with message transmission were calculated. Each QED message contains a 5th column populated with a POSIX time, the number of seconds since midnight January 1st 1970 UTC. These timestamps represent the time when the message began to be compiled by the QED app. The moreutils Linux package has a command (ts %s) that can be used to get the current POSIX time and append it to strings. The MQTT output is piped to the ts %s command to append a POSIX timestamp to each incoming message. This timestamp is used as the arrival time of each message. To calculate the latency associated with the message travel time the timestamp in the 5th column of each QED message is subtracted from the timestamp that was appended to the message representing the arrival time of that message. For each QED message, the latency is calculated and is combined with the QED message flag, the phone's IMEI number, the arrival time, and the sent time, and is archived using the same manner as the full data stream.

Mean Network Latency of Our Costa Rica and Chile EEW Networks

To find the average latency associated with the message transmission time in our EEW networks 24 hours of latency data for the QED I, X and A (acceleration) messages were collected for the Chile and Costa Rica networks. Histograms of the latencies for both networks were plotted to see what the network latency data looked like (Figures 69, 71). The latency data for both networks were bootstrapped, the data from each network was sampled with replacement to generate 10,000 new sample populations of latencies for each network. The mean of each of the 10,000 new sample populations were taken and appended to a vector which was then used to calculate a 95% confidence interval for the mean network latency of each network (See Appendix **VI** for the code).

The 95% confidence interval for the mean network latency of our Costa Rica network was between 566.3ms and 570.6ms (Figure 70). The 95% confidence interval for the mean network latency of our Chile network was between 767.6 ms and 777.9 ms (Figure 71). The 95% confidence interval for the mean network latency of our development network was between 481.1ms and 510.9ms (Figure 71).

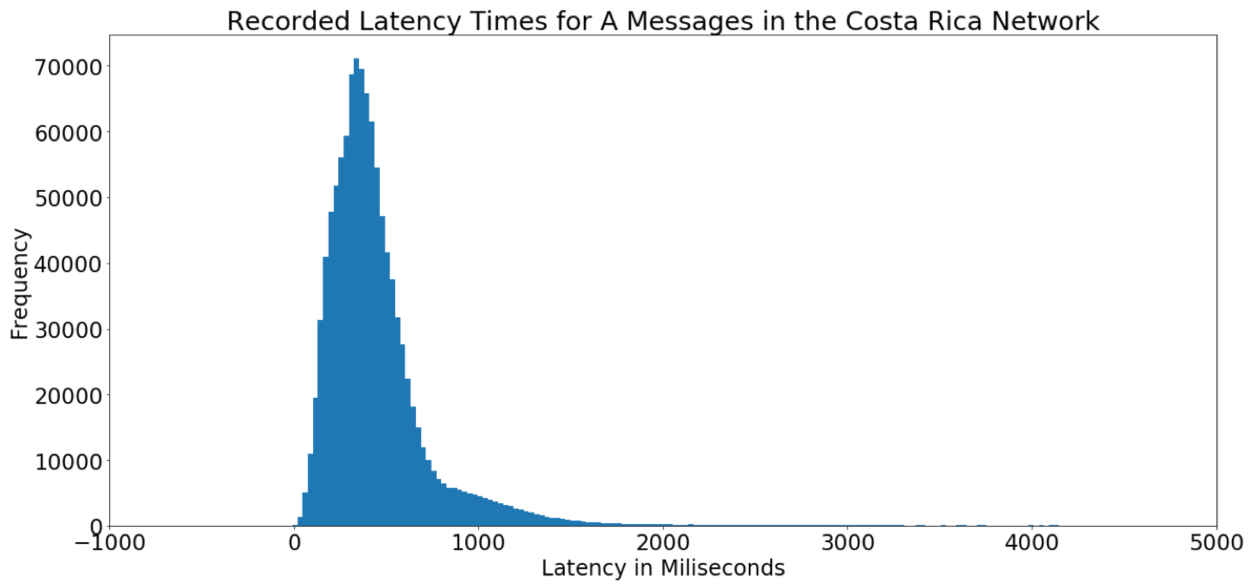


Figure 68 Recorded latencies form the Costa Rica Network.

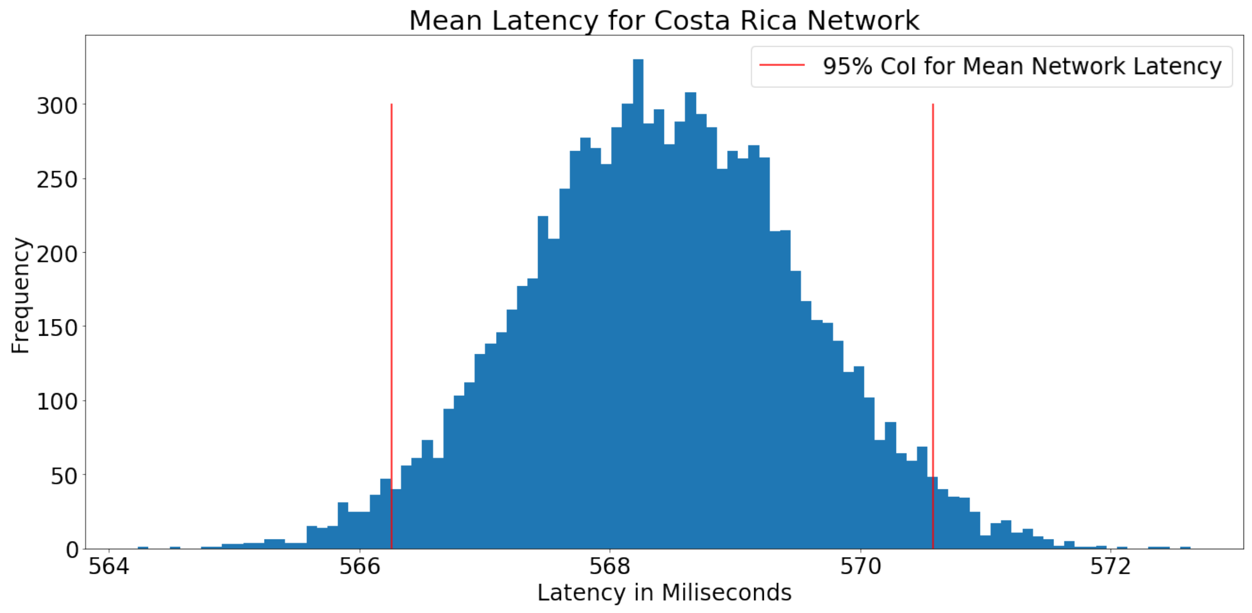


Figure 69 95% confidence interval for the bootstrapped mean latencies of the Costa Rica network.

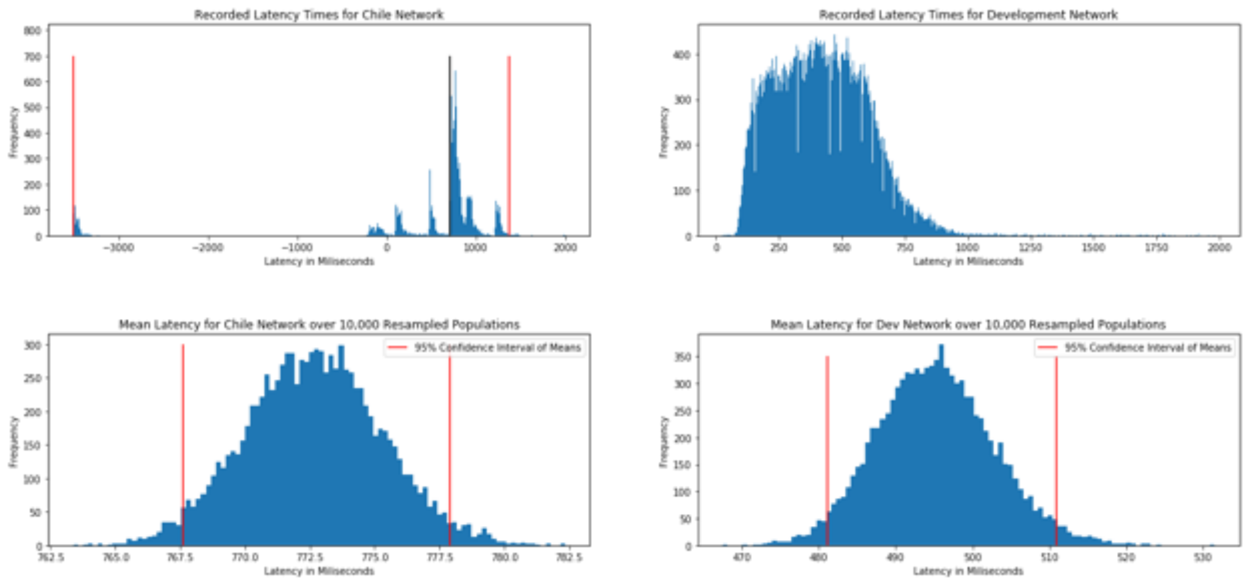


Figure 70 Recorded latencies for the Chile network (top left) and development network (top right). The 95% confidence interval of the Chile network (bottom left) and development network (bottom right).

Analysis of Latency Distribution in Costa Rica

The sensors in our EEW network transmit data over the cellular network. To test if the phones installed in the countryside, where the cellular signal is likely to be weakest, have larger latencies than phones that are installed in and around city centers in Costa Rica, we analyzed two days' worth of network latency data for the Costa Rica network. The data totaled a combined 128,359,526 latency measurements for all 80 sites. The two days of latency data were split by site into a list of 80 data frames, each of the 80 data frames were bootstrapped to generate 1000 new sample populations for each site, and the median latency of each of the 1000 sample populations was calculated. The mean of the resulting 1000 calculated medians for each site was calculated, and plotted over a map of Costa Rica in GMT with a blue to red color scale showing the range in latency from 0 to 1.5 seconds (Figure 72). The means of the bootstrapped median latency values for the 80 sites that reported data were centered around

0.24 seconds. The resulting map shows that the phones that are installed in rural areas are performing at the same level as phones that are installed in large towns and cities.

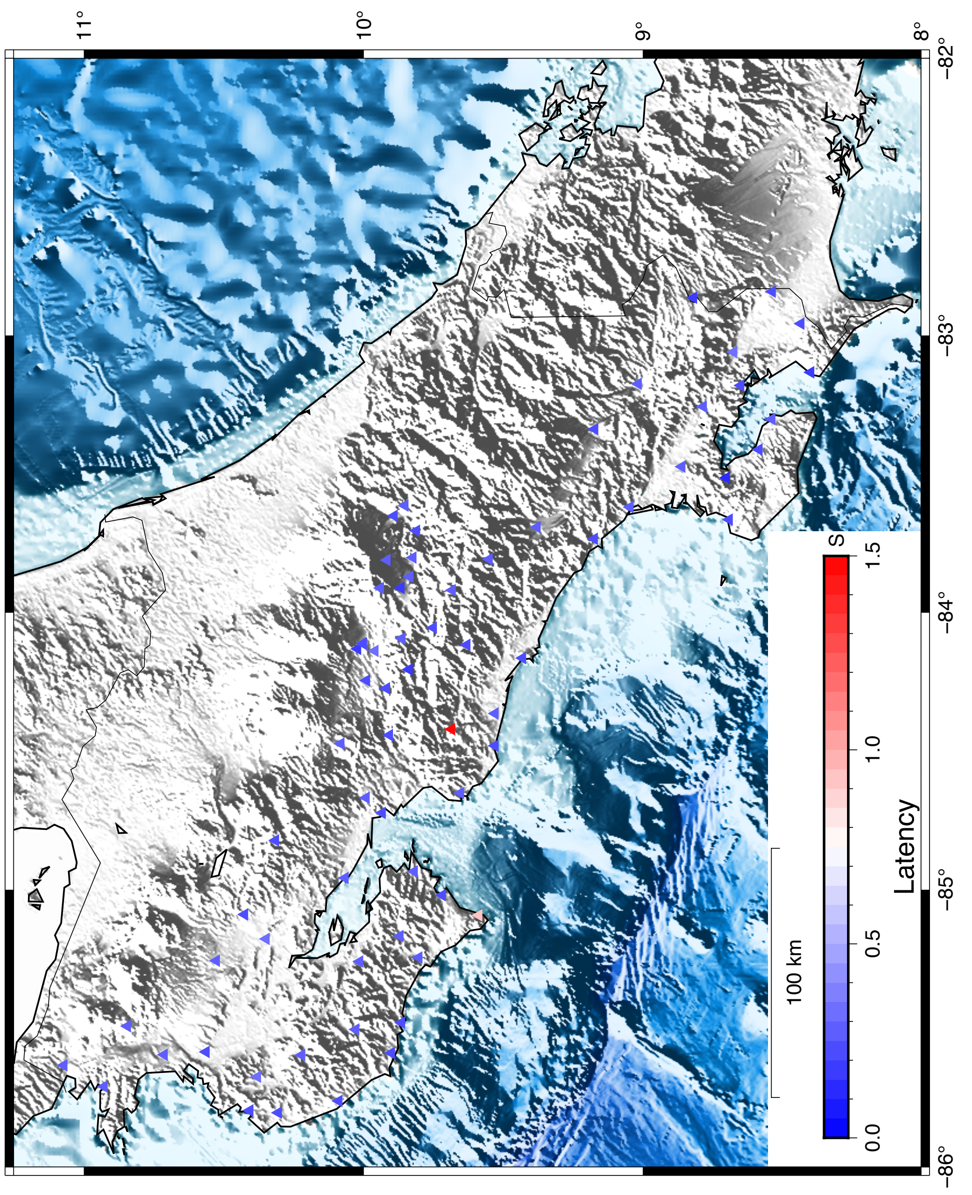


Figure 71 Map of the bootstrapped network latencies for the Costa Rica Network. There doesn't seem to be a difference in the reported latencies between sites installed in cities, and sites installed in the country side.

Analysis of Costa Rica EEW Performance

2012 Nicoya Earthquake

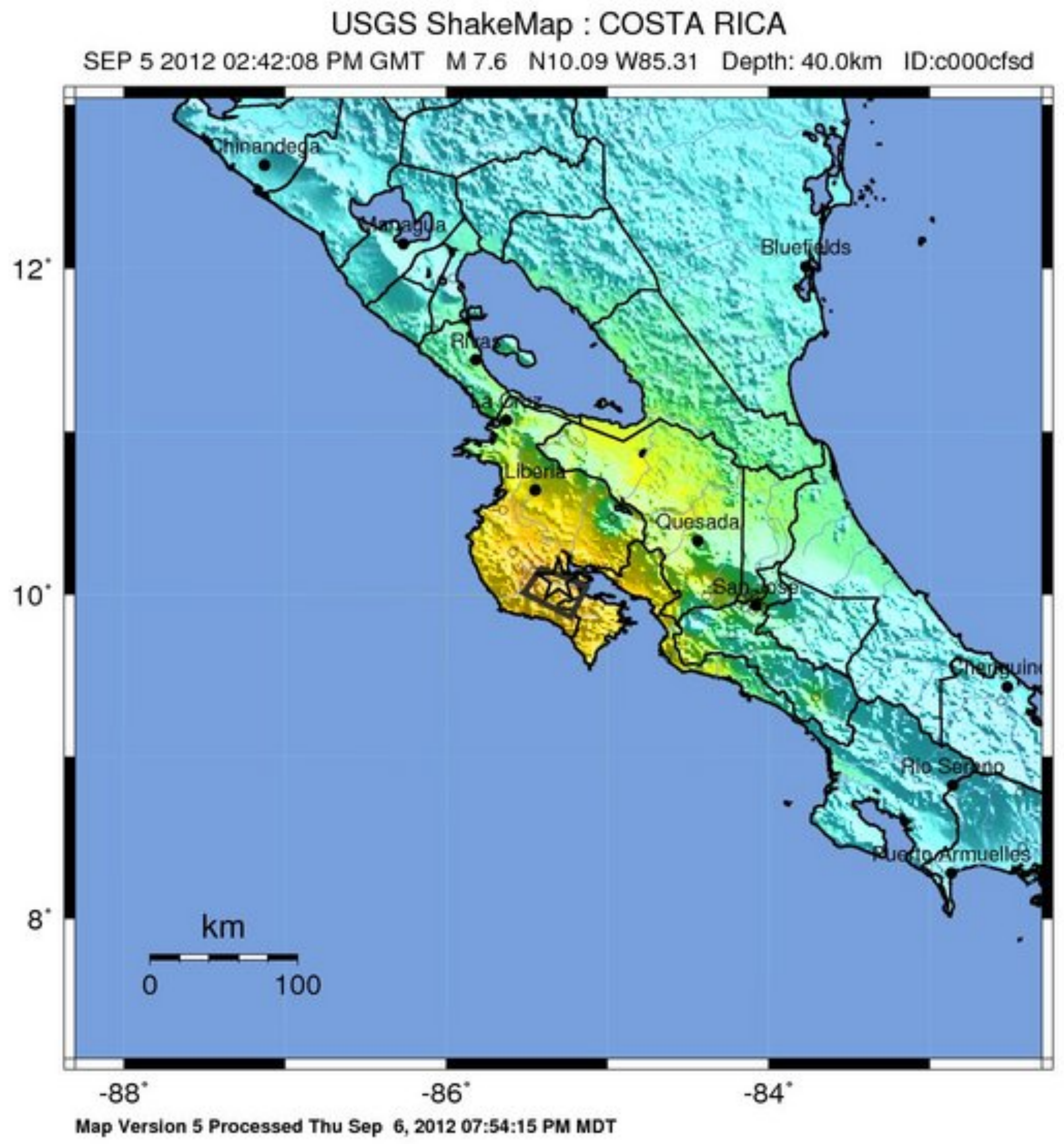
On September 5, 2012 The Nicoya Earthquake struck 10 km off the coast of Costa Rica's Nicoya Peninsula at 14:42:04.4 UTC, causing strong ground shaking in Puntarenas, Liberia, and San Jose. The earthquake was responsible for the deaths of 2 people, the destruction of over 200 homes, and an estimated $45 million dollars in damage according to the Costa Rican newspaper the Tico Times [TicoTimes.net]. The hypocenter was located at 9.76 N, 85.56 W about 13 km below sea level, the fault ruptured in the center and propagated 80 km in both directions from the hypocenter along a strike of 307 degrees, and about 30-50 km down dip at a velocity of 3 km/s. The source duration of the event was 21 s [Liu et al. 2015] [Yue et al. 2013].

Theoretical Best-Case Performance

In their 2018 paper *The limits of earthquake early warning: Timeliness of ground motion estimates*, Minson et al. looked at the best possible warning times that users might receive using a theoretical zero-latency EEW system that can instantly and accurately monitor the moment release of an earthquake. In the paper they define warning time as the difference between the time when the alarm is sent, and the arrival of the strong ground shaking at a user's location. It has been argued that possible EEW warning times would be limited by the fact that the final magnitude of an event cannot be determined at the time of nucleation, and that the only way to predict the final magnitude is by observing the evolution of the moment release [Kanamori et al. 2005][Rydelek et al 2006][Rydelek et al 2007][Yamamoto et al 2008]. In addition, it has been shown that an earthquake's final magnitude can be estimated from data at one-half of the source duration [Meier et al 2017] . To calculate the times at which an alert could be sent, Minson et al. used ground motion prediction equation's (GMPE's) to figure out what magnitude earthquake would cause ground shaking above 2, 5, 10, and 20%g thresholds at different distances. Then they used a circular crack model and a magnitude-log area scaling model from Hanks and Bakun 2014 to calculate how long the source duration would be for earthquakes ranging from Mw 5 to Mw 8. The calculated source durations were divided in half and were used as the zero-latency alarm times.

Minson et al. used two different approaches to model the arrival times of the strong ground motions, a point source model, and a finite line source model. In their point source model, they assume that the strong ground motion waves propagate out from the hypocenter at the S-wave velocity. In their finite line source model, they assume that the strong ground motion waves propagate along the fault at the fault rupture velocity until the end of the fault where they propagate out at the S-wave velocity. The fault rupture velocity is slower than the S-wave velocity so the strong ground motion waves would take longer to arrive at locations in the forward rupture direction. For locations in the backwards rupture direction the arrival time of the strong ground motion waves is modeled as a point source.

We ran a similar analysis using the 2012 Nicoya earthquake to see what warning times we could expect in Liberia, Puntarenas, and San Jose from a zero-latency system. The USGS Shakemap that was produced the day following the earthquake shows that there was very strong shaking (~22%g) in the area of Puntarenas, strong shaking (~12%g) in the area of Liberia, and moderate to light shaking (~2.8-6.2%g) in the San Jose area (Figure 73). The source duration of the 2012 Nicoya earthquake was shown to be 21 seconds [Yue et al. 2013]. Following the relationship between source duration and final magnitude we used an alarm time of 10.5 seconds. This means that users in the Puntarenas area with a warning threshold of ~22%g, users in the Liberia area with a warning threshold of ~12%g, and users in the San Jose area with a warning threshold of ~2.8-6.2%g would receive an alarm 10.5 s after nucleation.


USGS ShakeMap : COSTA RICA
SEP 5 2012 02:42:08 PM GMT M 7.6 N10.09 W85.31 Depth: 40.0km ID:c000cfsd
Chinandega
Managua
Bluefields
Rivas
La Cruz
Liberia
Quesada
San Jose
Changuinola
Rio Sereno
Puerto Armuelles
12˚
10˚
8˚
km
0
100
-88˚
-86˚
-84˚
Map Version 5 Processed Thu Sep 6, 2012 07:54:15 PM MDT

PERCEIVED SHAKING	Not felt	Weak	Light	Moderate	Strong	Very strong	Severe	Violent	Extreme
POTENTIAL DAMAGE	none	none	none	Very light	Light	Moderate	Mod./Heavy	Heavy	Very Heavy
PEAK ACC.(%g)	<0.05	0.3	2.8	6.2	12	22	40	75	>139
PEAK VEL.(cm/s)	*<0.02*	*0.1*	*1.4*	*4.7*	*9.6*	*20*	*41*	*86*	*>178*
INSTRUMENTAL INTENSITY	I	II-III	IV	V	VI	VII	VIII	IX	X+

Scale based upon Worden et al. (2011)

Figure 72 USGS shake map for the 2012 Nicoya earthquake. The colors represent the various shaking intensities that were felt at different locations. The gold star represents the estimated

epicenter for the earthquake. The Nicoya peninsula experienced strong to very strong shaking equivalent to 12-22%g. San Jose experienced light to moderate to strong shaking equivalent to2.8-6.2%g.

To model the arrival of the strong ground motion waves from a point source, we calculated the distance from the hypocenter to each city center. To calculate the distance, we make the assumption of a perfectly spherical earth, and used the Haversine function to calculate the arclength between the latitude and longitude of the epicenter and the latitude and longitude of each city center. The angle between the lines from the center of the earth and the hypocenter and the center of the earth and the city centers was calculated by dividing the arclength by the radius of the earth (6,378 km). Once we calculated each angle, we were able to use the law of cosines to calculate the linear distance from the hypocenter to the city centers. An S-wave velocity (Vs) of 3.2 km/s was used to calculate how long it would take the strong ground motion to arrive.

We calculate that for a point source model it would take 26.73 s, 30.93s, and 50.81s for the strong ground waves to reach Puntarenas, Liberia and San Jose respectively. Given an alarm time of 10.5 seconds, users in Puntarenas with a warning threshold of 22%g would receive a 16.23 s warning time, users in Liberia with a warning threshold of 12%g would receive a 20.43 s warning time, and users in San Jose with a warning threshold of 2.8-6.2%g would receive a 40.31 s warning time. These warning times assume that the EEW system can instantly and accurately monitor the moment release. The epicenter of the 2012 Nicoya earthquake was about 10 km offshore, and it would take ~3.12 seconds for the S-waves to reach the shore and be detected any real world EEW system. If we modify the alarm time for our theoretical zero-latency EEW system to reflect the location of the epicenter, then Puntarenas, Liberia, and San Jose would receive warning times of 13.11 s, 17.31 s, and 37.19 s respectively. Users that had set warning thresholds lower than the 22%g, 12%g, and 2.8-6.2%g at Puntarenas, Liberia, and San Jose could expect longer warning times.

During the 2012 Nicoya earthquake, the fault ruptured in the center, and propagated for 80km in both directions at a rupture velocity (Vr) of 3 km/s. To follow the Minson et al. finite line source model, we allow the strong ground motion to follow the rupture at Vr=3 km/s for the length of the fault before accelerating to Vs=3.2k m/s. The fault ends can be treated as a point source, and their latitudes and longitudes were calculated using the Haversine formula shown below [Equation 1]. After calculating the location of the fault ends, the S-wave travel time between each fault end and the cities in their respective directions were calculated following the same method as the point source model above. We calculate that the strong ground motion would arrive in San Jose, Liberia, and Puntarenas at 63.8 s, 55.5 s, and 50.2 s after nucleation respectively. Using the same alarm time of 10.5 s would give 53.3s, 45 s, and 39.7s of warning time for San Jose, Liberia, and Puntarenas respectively. Subtracting the approximate 3.1 second travel for the S-Wave to hit the coast would leave us with a warning time of 50.2 s for users in San Jose who had a warning threshold of 2.8-6.2%g, 41.9 s for users in Liberia who had a warning threshold of 12%g, and 36.6 s for users in Puntarenas who had a warning threshold of 22%g.

Equation 1: Haversine formula for the destination location given bearing and distance.

$$\varphi_2 = \operatorname{asin}(\sin\varphi_1 * \cos\delta + \cos\varphi_1 * \sin\delta * \cos\theta)$$
$$\lambda_2 = \lambda_1 + atan2(\sin\theta * \sin\delta * \cos\varphi_1\ , \cos\delta - \sin\varphi_1 * \sin\varphi_2)$$

Finite Fault Rupture Detector (FinDer)

To simulate the 2012 Nicoya earthquake, the data were run through the Finite Fault Rupture Detector (FinDer) EEW algorithm. FinDer takes the observed peak ground accelerations (PGA) values and spatially interpolates them over the entire EEW network. FinDer then uses acceleration thresholds to perform a near/far classification on the spatially interpolated map. If the PGA on the spatially interpolated map is above the acceleration threshold the algorithm assumes that area is near to the fault, and it gets a unitless value of 1. If the PGA in that area doesn't pass the acceleration threshold the algorithm assumes that area is far away from the fault, and it gets a unitless value of 0. These values are used to create a 2D binary map in cartesian space. FinDer then compares the binary map against 12,000 preexisting templates of ground acceleration to find the rupture length, and strike. A divide and conquer algorithm is used to enable FinDer to find the strike without comparing the binary map to every strike from 0 to 180 degrees. Once the fault length has been determined a moment magnitude is estimated using empirical fault length to magnitude relationships. These relationships can vary depending on the depth, and type of the fault, and so magnitude uncertainties of Mw 0.1-0.2 are introduced. [Böse et al. 2012] [Böse et al. 2018]. FinDer allows the fault length to grow over time as new data comes in, and more areas in the spatially interpolate map pass the acceleration thresholds for near/far classification. This prevents the magnitude estimates from saturating, and means that algorithms using displacements measured with GNSS are not as necessary for our network as they would be for an EEW network running a point source EEW algorithm.

Simulation of 2012 Nicoya Earthquake

In their 2020 paper, *How Often Can Earthquake Early Warning Systems Alert Sites with High-Intensity Ground Motion?* Meier et al. analyzed the real-world performance of three EEW algorithms, FinDer, PLUM, and EPIC in 219 historic Japanese earthquakes ranging from Mw 4 to Mw 9.1. Japan maintains excellent records of all their seismic strong motion data, and make it publicly available. They were able to replay the data from 219 different earthquakes for the different algorithms to see what warning times locations close to the epicenter of the earthquakes, where the ground shaking will be strongest, users would be able to receive. They found that about 50% of sites where the ground shaking was strong to extreme would have received a warning of at least 5 seconds if their warning threshold was set low enough.

The sensors in our EEW network are smartphones, and we can simulate real events by programming the smartphones in our network to vibrate at specific times to simulate the arrival of the S-waves and thus field test our network's response. To simulate the 2012 Nicoya event in our vibration tests the phones were programmed to vibrate at the times when the S-waves would reach each site. The S-wave arrival times for each site were calculating by taking the distance between the hypocenter and each site, that were obtained using the same method that was previously used to calculated distances from the hypocenter to the various city centers, and dividing them by a Vs of 3.2 km/s. Each phone was programed to vibrate for the length of the 2012 Nicoya earthquakes source duration of 21 s [Liu et al. 2015].

Vibration Test Results

Twenty-two vibration tests were run to simulate the Nicoya earthquake (Table 5). The times that FinDer output the first alerts were bootstrapped to create 10,000 new sample populations, and we calculated the 95% confidence intervals of the mean first alert time to be between 11-12.48 s (Figure 74). On average, FinDer predicted ground shakings of 0.5-1%g in San Jose, 1%g in Liberia, and 3%g in Puntarenas in the first alert. Assuming that users set their ground acceleration thresholds accordingly, and that it takes 10 seconds to assemble and deliver a warning, users in Puntarenas, Liberia, and San Jose would have received warning times of 4.25-5.73 s, 8.45-9.93 s, and 28.33-29.81 s respectively for the earthquake we simulated.

First Alert Sent (s)	Estimated Magnitude (Mw)	Puntarenas Warning Time (s)	Liberia Warning Time (s)	San Jose Warning Time (s)
11.505	6.3	15.225	19.425	39.305
12.228	6.6	14.502	18.702	38.582
12.348	6	14.382	18.582	38.462
11.121	6.4	15.609	19.809	39.689
10.03	6.2	16.7	20.9	40.78
11.03	6.6	15.7	19.9	39.78
11.593	6.5	15.137	19.337	39.217
10.682	6.4	16.048	20.248	40.128
12.242	6.2	14.488	18.688	38.568
10.874	6.5	15.856	20.056	39.936
10.995	6.5	15.735	19.935	39.815
12.266	6.6	14.464	18.664	38.544
9.976	5	16.754	20.954	40.834
10.154	6.6	16.576	20.776	40.656
11.577	6.6	15.153	19.353	39.233
11.953	6.5	14.777	18.977	38.857
10.613	6.6	16.117	20.317	40.197
12.329	6.1	14.401	18.601	38.481
11.559	6.5	15.171	19.371	39.251
11.505	6.1	15.225	19.425	39.305
18.931	6.7	7.799	11.999	31.879
10.037	5.9	16.693	20.893	40.773

Table 6 First solutions from FinDer during the Vibration Tests. Warning times assume a zero-latency delivery system.

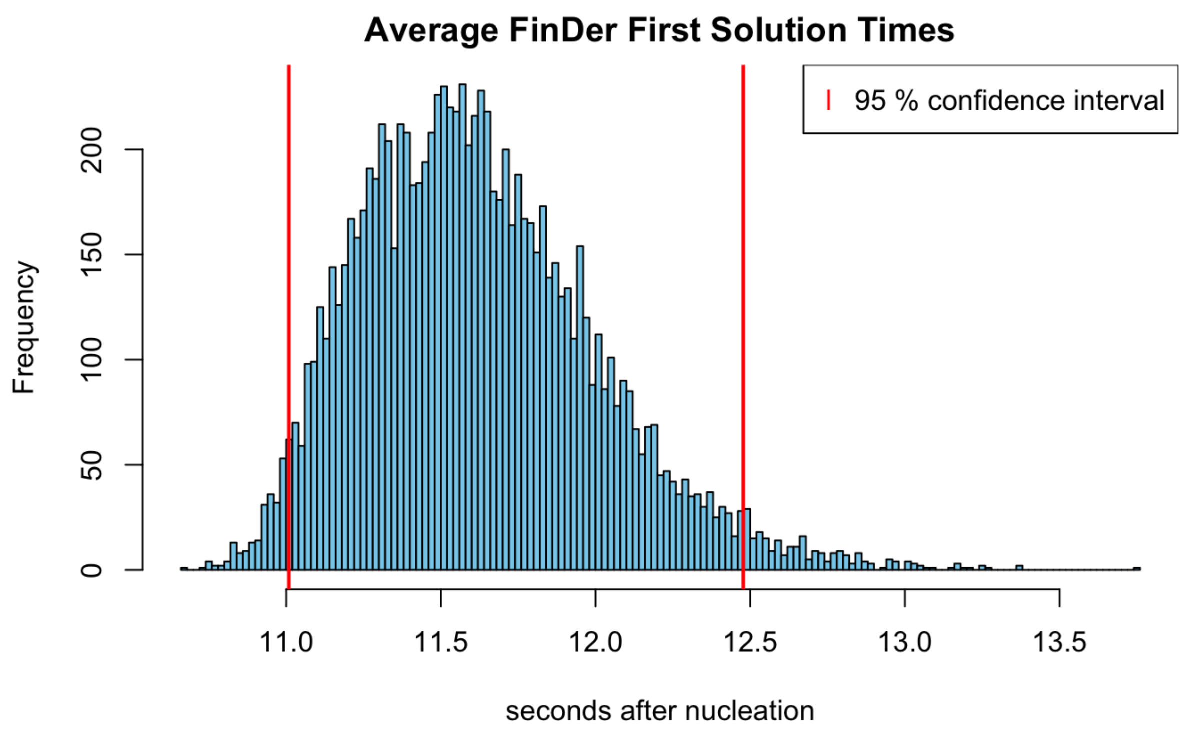


Figure 73 The histogram of the mean solution times from FinDer over 10,000 bootstrapped samples. The 95% confidence interval for the first solution we would expect to see from the 2012 Nicoya earthquake is between 11 s, and 12.48 s. This would give warning times of 14.25-15.73 s, 18.45-19.93 s, 38.33-39.81 s for Puntarenas, Liberia, and San Jose respectively.

Discussion of Smartphone EEW Network Performance

During the twenty-two vibrations tests that were run to simulate the 2012 Nicoya earthquake, FinDer generated initial solutions using 4-5 phones with magnitude estimates of Mw 5 to Mw 6.7 . The lower initial magnitude estimate of Mw 5 suggests that we would be able to detect an earthquake of this magnitude with our system. FinDer's initial solutions predicted ground accelerations of ~3%g for the first solutions for the Puntarenas area. The final ground shaking estimates that FinDer output during the vibration tests were ~18%g which equates to strong shaking and moderate damage. If users in Puntarenas had set their ground acceleration thresholds at or below 3%g they would have received a warning with enough time to react and take shelter before the strong ground motion arrived. Users that are near a fault are going to experience the strongest ground shaking, and are the people most in need of EEW. Our results show that our smartphone EEW network is capable of delivering alerts to users that are near the fault before the strong ground shaking will reach them, provided that they set their ground accelerations low enough. Users that require larger ground accelerations before a warning can be delivered are unlikely to receive a warning in time to take any action.

The vibration tests are a complete real-world test of the performance of our network. The smartphones detected accelerations that were above FinDer's threshold, FinDer estimated a fault length, strike, and moment magnitude for each event, and updated the solution several times before issuing a final solution. The network performed well, and FinDer was able to output solutions before the strong ground shaking would have reached San Jose, Liberia, and Puntarenas. Our secondary EEW algorithm, PLUM, was also able to detect the simulated earthquake, and it sent out email alerts to our team for each of the vibration tests. PLUM sent the alerts ~14 seconds after the simulated earthquakes nucleation, which would have given positive warning times for users in all three cities as well. Moving forward, these vibration tests can be run up and down the fault to simulate historic earthquakes, or in seismic gaps where earthquakes are likely to occur to get a better idea of the warning times we can expect from our network under a wide range of earthquake scenarios.

Conclusions

Our analysis of the drift of the external UBLOX GNSS chip shows that we are able to detect displacements from large earthquakes while our sensors are installed in wood or brick buildings. Our findings show that we would be able to detect displacements of 0.76 m – 1.51 m over 10 seconds in buildings of similar construction to our box and the residential house. Each structure is likely to have its own effect on the drift of the GNSS precision. Calculating the unique drift over time for each phone should allow us to apply those values as the input for our displacement detection and measurement model.

Using the results of the difference tests we ran on the instruments installed inside the box as inputs for our detection model we were able to detect the displacements of 1-8 m with the external UBLOX GNSS chip. The recall of our displacement detection model is ~97% over both large and small time periods. This means that we are able to detect 97% of the real displacements from 1-8m. However, the precision of the model greatly increases as the time period it is applied to decreases. Applying the displacement detection model over the entire time period of the displacement tests yielded a precision of about 42%, when the model was applied to 2-minute time periods centered around the times that the box was pushed increased the precision of the model to about 98%. The simple detection model presented here would not be useful in detecting displacements in real time, but could be used to go back after an event has been detected to obtain displacements that could be used to estimate an earthquakes magnitude.

Individually, the UBLOX GNSS chips are not able to accurately measure displacements. The displacements measured by the UBLOX GNSS chips were 0.87 m away from the real displacement values on average when we include large outliers, and 0.51 m away from the real displacements if we throw out the large outliers. When 3 or more of our UBLOX sensors measured the same displacement the average of their measured displacements was within 0.4-0.24 m of the true displacement. When 2 or more of our UBLOX sensors measured the same displacement the average of their measured displacement was within 0.4-1.2 m of the true displacement. If 3 or more sensors are installed in a single area, our network would be able to measure the displacements from Mw 8+ earthquakes. Theoretically, the UBLOX GNSS sensor would be able to detect displacements smaller than 1 m as long as the rate of displacement was increased faster than the rate of the sensor's drift over time. This would allow the network to detect smaller earthquakes, but more testing would need to be done to verify that the sensors are capable of detecting the smaller displacements.

Our analysis of the performance of the phone's internal GNSS chip showed that the data from the internal GNSS chip was too heavily filtered to be able to detect displacements from large earthquakes. Without generating our own position solutions using the raw GNSS observables we would not be able to use the internal GNSS data in our displacement detection model. There are currently a few Chinese smartphones coming out with dual frequency GNSS chips that would have a similar performance to the external GNSS chips. As these chips become more common in smartphones, we can implement them in our EEW networks without needing the external UBLOX GNSS chips.

Moving forward, additional displacement experiments to analyze the performance of the dual frequency GNSS chips in the new Chinese smartphones should be run to compare their performance to the external UBLOX GNSS chip used in our networks. This will yield a better understanding of the ability of future phones to replace the external UBLOX GNSS component of our sensors. In addition to experiments on the dual frequency GNSS chips in new smartphones, more displacement experiments should be run using multiple sensors to better constrain the accuracy of their mean measured displacements.

The bootstrap analysis of the latency associated with message transmission times for our Costa Rica, Chile, and development networks showed that the mean network latency ranges from 481.1-777.9 ms. These latencies are unlikely to adversely affect the ability of our EEW networks to deliver timely warnings. Our smartphone EEW sensors rely on the cellular network of the countries they are installed in to transmit the data back to the central locations that are archiving the data, and passing them through EEW algorithms. Our analysis of the latencies across the Costa Rica network show that there isn't a difference between sites that are installed in and around large city centers and those installed in the countryside.

Following the methods presented in Minson et al. 2018 for analyzing the best possible warning times an EEW network can produce, we found that users in San Jose, Liberia, and Puntarenas would be able to receive warning times of 37.19 s, 17.31 s, and 13.11 s for an point source, and 50.2 s, 41.9 s, and 36.6 seconds respectively if a finite line source is used. These warning times are large enough to allow users to take action to protect themselves before the strong ground motion arrives.

The vibration tests we ran that were designed to simulate the 2012 Nicoya earthquake showed that our smartphone EEW network can detect accelerations, estimate the location and strike of the fault, and the magnitude of the event, and issue an alarm in time for users in San Jose, Liberia, and Puntarenas to take action. The first solutions that FinDer output during the vibration tests used 4-5 sites, and showed estimated moment magnitudes of Mw 5-Mw 6.7. This suggests that we would at least be able to detect a Mw 5 earthquake with our system. The tests showed that our EEW network is capable of issuing alerts to users in Puntarenas, who would have been the hardest hit by the simulated earthquake, with enough warning time to take action before the strong ground shaking reached them.

Bibliography

Yue, H., Lay, T., Schwartz, S. Y., Rivera, L., Protti, M., Dixon, T. H., Owen, S., & Newman, A. V., (2013). The 5 September 2012 Nicoya, Costa Rica Mw 7.6 earthquake rupture process from joint inversion of high-rate GPS, strong-motion, and teleseismic P wave data and its relationship to adjacent place boundary interface properties. *Journal of Geophysical Research: Solid Earth, Vol. 118*, pp 5453-5466.

Colombelli, S., Allen, R. M., & Zollo, A., (2013). Application of real-time GMS to earthquake early warning in subduction and strike-slip environments. *Journal of Geophysical Research: Solid Earth*, Vol. 118, pp. 3448-3461.
Böse, M., Heaton, T. H., & Hauksson, E., (2012). Real-time Finite Fault Rupture Detector (FinDer) for large earthquakes. *Geophysical Journal International*, Vol 191, pp. 803-812.

Böse, M., Smith, D. E., Felizardo, C., Meir, M. A., (2018). Heatoh, T. H., & Clinton, J. F., FinDer v.2: Improved real-time ground-motion predictions for M2-M9 with seismic finite-source characterization. *Geophysical Journal International,* Vol 212, pp. 725-742.

Sahu, K. N., Naidu, C. D. PhD, & Sankar, K. J. PhD, (2014). Study of RF Propagation Losses in Homogeneous Brick and Concrete Walls using Analytical Frequency Dependent Models. *Journal of Electronics and Communication Engineering,* Vol 9, Issue 5, pp. 58-66.

Dabove, P., Pietra, V. D., (2019). Towards high accuracy GNSS real-time positioning with smartphones. *Advances in Space Research, Vol 63*, pp. 94-102.

Fang, R., Shi, C., Song, W., Wang, G., & Liu, j., (2014). Determination of earthquake magnitude using GPS displacement waveforms from real-time precise point positioning. *Geophysical Journal International*, Vol. 196, pp 461-472.

Liu, C., Zheng, Y., Xiong, X., Wang, R., Lopez, A., & Li, J., (2015). Rupture processes of the 2012 September 5 Mw 7.6 Nicoya Costa Rica earthquake constrained by improved geodetic and seismological observations. *Geophysical Journal International*, Vol 203, pp. 175-183.

LaBrecque, J., Rundle, J., Bawden, G., (2018). Global Navigation Satellite System to Enhance Tsunami Early Warning Systems. GTEWS 2017.

Kjærgaard, M. B., Blunck, H., Godsk, T., Toftkjær, T., Christensen, D. L., & Grønbæk, K., (2010). *Lecture Notes in Computer Science*. Vol 6030, pp. 38-56.

Hein, G., Teuber, A., Theirfelder, H-J., Wolfe, A., (2008). *Inside GNSS*, Working Papers insidegnss.com, pp 47-53.

Melgar, D. & Hayes, G. G., (2017). Systematic observations of the slip pulse properties of large earthquake ruptures. *Geophysical Research Letters,* Vol 44, pp. 9691-9698.

Melgar, D., Crowell, B.W., Geng, J., Allen, R. M., Bock, Y., Riquelme, S., Hill, E. M., Protti, M., Ganas, A., (2015). Earthquake magnitude calculation without saturation from the scaling of peak ground displacement. *Geophysical Research Letters*. Vol 42, pp. 5197-5205.

Minson, S. E., Wu, S., Beck, J. L., & Heation, T. H., (2017). Combining Multiple Earthquake Models in Real Time for Earthquake Early Warning. *Bulletin of the Seismological Society of America.* Vol 107, Issue No. 4, pp. 1868-1882.

Minson, S. E., Brooks, B. A., Glennie, C. L., Murray, J. R., Langbein, J. O., Owen, S. E., Heaton, T. H., Iannucci, R. A., Hauser, D. L., (2015). Crowdsourced earthquake early warning. *Sci Adv.* 1, e1500036.

Blewitt, G., Kreemer, C., Hammond, W. C., Plag, H-P., Stein, S., (2006). Rapid determination of earthquake magnitude using GPS for tsunami warning systems. *Geophysical Research Letters*, Vol. 33 L11309.

Vigny, C., Socquet, A., Peyrat, S., Ruegg, C., Métois, M., Madariaga, R., Morvan, S., Lancieri, M., Lacassin, R., Campos, J., Carrizo, D., Bejar-Pizarro, M., Barrientos, S., Armijo, R., Aranda, C., Valderas-Bermejo, M-C., Ortega, I., Bondoux, F., Baize, S., Lyon-Caen, H., Paves, A., Viollet, P., Vevis, M., Brooks, B., Smalley, R., Parra, H., Baez, J-C., Blanco, M., Cimbaro, S., Kendrick, E., (2011). The 2010 Mw. 8.8 Maule Megathrust Earthquake of Central Chile, Monitored by GPS. *Science*. Vol 332, pp. 1417-1421.

Eckerman, L., Agüero, A., Spagnotto, S., Martinex, P., Nacif, S., (2018). Seismic-gravimetric analysis of the subducted Nazca plate 1 between 32 S and 36 S. *Geodesy and Geodynamics*, Vol 9, pp. 57-66.

Strauss, J. A., Allen, R. M., (2016). Benefits and Costs of Earthquake Early Warning. *Seismological Research Letters*, Vol 87, Issue No. 3, pp. 765-772.

Allen. R. M., Gasparini, P., Kamigaichi, O., Böes M., (2009). The Status of Earthquake Early Warning around the World: An Introductory Overview. *Seismological Research Letters*. Vol 80, Issue No 5, pp 682-693.

Ruhl, C. J., Melgar, D., Grapenthin, R., Allen, R. M., (2017). The value of real-time GNSS to earthquake early warning. *Geophysical Research Letters,* Vol 44, pp. 8311-8319.

Minson, S. E., Meir, M-A., Baltay, A., Hanks, T. C., Cochran, E. S., (2018). The limits of earthquake early warning: Timeliness of ground motion estimates. *Science Advances,* Vol 4, eaaq0504.

Simmons, M., Minson, S. E., Sladen, A., Ortega, F., Jiang, J., Owen, S. E., Meng, L., Ampuero, J-P., Wei, S., Chu, R., Helmberger, D. V., Kanamori, H., Hetland, E., Moore, A. W., & Web, F. H., (2011). The 2011 Magnitude 9.0 Tohoku-Oki Earthquake: Mosaicking the Megathrust from Seconds to Centuries. *Science*, Vol 332, pp 1421-1425.

U.S. Geological Survey. (2011). *Report on the 2010 Chilean Earthquake and Tsunami Response* (Open-File-Report 2011-1053, version 1.1)

Meir, M-A., Kodera, Y., Böes, M., Chung, A., Hoshiba, M., Cochran, E., Minson, S., Hauksson, E., & Heaton, T., (2020). How Often Can Earthquake Early Warning Systems Alert Sites with High-Intensity Ground Motion? *Journal of Geophysics Research: Solid Earth,* 125, e2019JB017718.https://doi.org/10.1029/2019JB017718

Daniell, J. E., Khazai, B., Wenzel, F., (2012). The Worldwide Economic Impact of Earthquakes, Paper No. 2038. *Conference Paper*

Kenny, C., (2009). Why do People Die in Earthquakes? *The World Bank Policy Research Working Paper*, 4823

Appendix

Appendix I

Phone Setup

The sensors in our Chile Earthquake Early Warning (EEW) network are Samsung Ace4 Neo and Samsung J111M phones. In order to be able to use these phones in our network they must be properly configured to ensure that the standard defaults do not interfere with their operation as an EEW sensor. When the phone is first turned on it begins the initial setup, and the following steps need to be taken.

- It is important to skip setting up a Google account, and to unselect all of the Google location settings so the phone will not periodically send Google position data or try to improve location accuracy.
- After the initial phone setup, the internal settings on the phone are changed. In the settings menu auto update is turned off in the about phone, system updates folders. This ensures that the phone does not automatically update itself to a newer version of Android that may not be compatible with the QED app.
- Developer options, which are hidden options that need to be enabled to be able to install/remove our applications, are opened by tapping “build number” in the “About phone” folder seven times.
- After the developer options folder is unlocked USB debugging is enabled and disable verify apps via USB is turned on. This allow apps to be installed on the phone and allows the phone to be programmed.
- In the security folder unknown sources are enabled and verify apps is disabled so that the QED app which is from an unknown source can be installed on the phone.
- To stop the phone from searching for new Wi-Fi networks the network notification is turned off. Keep Wi-Fi on during sleep is set to always to ensure that if a phone is connected to a Wi-Fi network it will stay connected while it is sleeping.
- The phones in Chile have SIM cards from the Entel Cellular provider, and their network settings need to be configured before the can get access to the internet and start streaming data. The SIM card does not automatically send the correct Access Point

Name (APN) to the phone. The APN is the name of the gateway that the phone uses to access the internet. The APN in the phone is set to be imovil.entelpcs.cl with username entelpcs, and PAP Authentication type.

When all of this has been completed the phone is ready to be rooted and to have the QED apps installed.

The Android Operating System was developed by Google, and is version of the Linux kernel. Android is provided to the end user without root access. Without access to the root user it is not possible to remotely update, remove, or install apps, or modify other restricted files. To get around this problem the phones need to be “rooted” to enable to root user. We do this using the Odin rooting program and the ROOT(English).bat script on a laptop. The phone is connected to the laptop via a USB cable and the ROOT(English).bat script is run. In the Odin program a Team Win Recovery Project (TWRP) boot.tar file is loaded and flashed to the phone which will give us root access. The TWRP file is model specific, and will not work for phones it was not specifically written for. After the TWRP file has been flashed to the phone the phone will reboot, and the Super SU app which controls what apps will have root access is installed on the phone.

Phone Installation Method

When all of the applications have been installed and properly configured the phones are ready to be installed at a site. There are two types of installation that have been deployed in our Chilean smartphone EEW network that we refer to as “phone on the roof” and “phone on the wall” installations. These two installation types require different tools, and times to complete the installations. It is possible to complete a “phone on the roof” installation in 2-3 hours. A “phone on the wall” installation can be completed in 1 hour.

A “phone on the roof” installation consists of a smart phone with our applications installed on it, our enclosure, a 12 V 5.2 Ah Li-Po batter, our GPS/charging circuit, and a 25- or 30-Watt solar panel. The enclosure is installed underneath the solar panel. A piece of white plastic is bolted onto the back of the solar panel mount so the enclosure is shielded from direct sunlight which helps to regulate its temperature.

The enclosure that houses the phone, the drone battery, and our GPS/charging circuit is a small 7 ½”x6”x3” plastic box with two ½” cable glands that allow the solar panel wires and GPS antenna cable to be routed into the box. The Li-Po battery is secured to the bottom of the enclosure with double sided adhesive tape. A non-conductive substrate board that is used as a platform to mount the phone and our GPS/charging circuit is elevated above the drone battery on circuit board stanchions. The phone is mounted to one side of the non-conductive substrate board opposite the GPS/charging circuit with two strips of Velcro. Two strips of foam are glued to the lid of the enclosure box so that when the lid is screwed down into place they press up against the phone and keep is secure.

Our GPS/charging circuit takes the input from the GPS antenna which is glued to the lid of the enclosure box and passes it to a UBLOX Neo-7P GPS chip. A Bluetooth chip in the circuit allows the phone to pull the external GPS data from the UBLOX chip. The other half of the circuit is a solar charge controller. It regulates the solar panels output, and charges the Li-Po battery through a balance port and the phone through a USB port. The charging circuit includes a battery temperature sensor and will stop charging the Li-Po battery if it gets too hot**.**

The first step in a “phone on the roof” type of installation is to find a building with a suitable roof. The installation is designed to be bolted down onto a flat concrete surface with 4 ¼-20 stainless steel concrete wedge anchor bolts. The roof needs to be accessible, and has to have room for the solar panel to be installed facing North.

When a suitable installation location has been found the next step is to assemble the solar panel mount. Two holes are drilled into each of the shorter sides of the solar panels frame using a drill template. The template has three settings that allow for the panel to be set at the proper angle for the installation latitude. After the solar panel frame has been drilled, 1”x1”x8” lengths of aluminum angle are bolted onto the sides of the panel using 8-32 x 1/2” bolts with nylock nuts. The aluminum angle forms the base and the back of the solar panel mount, and allows for the panel to be bolted down directly onto a roof.

After the solar panel mount has been assembled the solar panels wires need to be connected to the charging circuit. The enclosure comes with 20 AWG black and red wires that are pre-soldered to a 2-pin clip that is connected to the circuit board. The solar panels wires are routed into the enclosure through the open ½” cable gland, and spade crimp connectors are attached to both sets of wires. This allows for the box and solar panel to be decoupled if needed.

When the solar panel has been wired up to the charging circuit the phone can be placed into the box and plugged into our circuit board with a USB cable. Before the phone can pull the external GPS data from the UBLOX chip it needs to be paired with the Roving Bluetooth microchip on the board. To pair the phone with the Bluetooth, microchip the Bluetooth menu on the phone is opened and Bluetooth must be enabled. Once Bluetooth has been enabled on the phone, devices that are able to be paired will be listed in the Bluetooth menu. Each of the Roving Bluetooth chips on our GPS/charging circuit has a unique serial number, and will show up as a device named RNBT- followed by the last four digits of the serial number. When the phone has been paired with the Bluetooth microchip the QED Configurator app is opened on the phone and is run according to the instructions laid out above in the Applications section.

The enclosure and the solar panel are mounted to the roof with ¼-20 concrete wedge anchors. A second aluminum drill template is used to mark the location of the holes that need to be drilled into the roof to mount the solar panel and the enclosure. After the holes have been marked a cordless hammer drill with a ¼” concrete bit. After the holes are drilled, they are cleaned and the wedge anchors are tapped into the holes. The enclosure is placed down onto the anchors and is secured in place with a ratchet. After the enclosure is secured the lid is screwed down, and the solar panel is placed over its anchors and is bolted down in the same manner.

A “phone on the wall” installation is much simpler, and only consists of smartphone, a smartphone case, and an AC wall charger. Because there is no external GPS that the phone needs to connect to in this type of installation there is no need to install the QED Configurator app. The rest of our EEW applications are installed in the same way, and the phone settings remain the same. After the phone has been properly configured and the applications have been installed the phone is placed in a smartphone case. A piece of 3M double sided adhesive tape is applied to the back of the case and the case is stuck to the wall. In a “phone on the wall” installation the phone is charged using the wall charger that comes with the phone, and so the phone needs to be mounted within 3 feet of a power outlet.

After the installation there are two site specific settings files that must be added to Chris Duncan’s GIS-Matters server before the phone will start streaming its data. Both of the settings files follow the naming convention settings-xxx.txt where xxx is the phones QED Id number. The first settings file is located in the http://gismatters.com/pickup/eew/ directory and has one line that points the phone to a network specific sub directory where its unique QED settings file is located. For example, a phone in the Chile network would be pointed to the http://gismatters.com/pickup/eew/3/chile directory, and a phone in the Test network would be pointed to the http://gismatters.com/pickup/eew/3/test directory. This assigns the phone to a specific network. In each of the network sub directories there is a QED settings template named settings-xxx.txt. This template is copied and renamed with the x’s replaced with the QED Id number of the phone, and the first line of the new settings file, SET IN SERVICE ON, is uncommented. After SET IN SERVICE ON is uncommented, and the phone picks up its new settings file, QED will begin to stream its data to the Server running the MQTT Broker.

Appendix II

QED Messages

QED streams the following messages (depending on settings – any one of these can be disabled/enabled via a setting; also depending on availability -- e.g., if no Bluetooth then no external GPS messages):

A -	accelerometer raw values in device coordinates: A,id,systime,x,y,z
R -	event-detector inputs in earth coordinates: R,id,systime, ,h1,h2,v
C -	combined accelerometer and event-detector values: C,id,systime,x,y,z, ,h1,h2,v
O -	orientation of device: O,id,systime,a,p,r
X -	external GPS: X,id,systime,gpstime,lat,lon,alt,qual,numsat,hdop
I -	internal GPS: I,id,systime,lat,lon,alt
F -	FinDer: F,id,systime,zhpeak,zhpeaktime
B -	BEFOREs: B,id,systime,prelat,prelon,prealt,ofslat,ofslon,ofsalt
V -	SVINFO satellite count and status: V,id, systime,[SVI:FLAGS:QUAL:CNO]
S -	status: S,id,systime,long-string-of-|-delimited-info
E -	error E,id,systime,longish-string-of-maybe-|-delimited-info
D -	debug: D,id,systime,longish-string-of-|-delimited-info

Appendix III

Applications

After the phones have been setup, their settings have been properly configured, and they have been rooted, the apps needed for the EEW application are installed. The four Android package (apk) application files for EEW that we use are QED, QED Configure, SSH Server Pro, and Keep Running. QED and QED configure are custom apps built by our team for the EEW application. KeepRunning and SSH Server Pro are commercial packages that we use to ensure the QED app is always running on the phones, and to enable remote access to the phones respectively. The installation is managed using the Android Debug Bridge (ADB) command line tool. The ADB client which issues the commands to the phone and the ADB server which manages the connection between the phone and the laptop are run on the laptop. The ADB daemon that executes the commands on the phone is run in the background of the phone. To program the phone, it is connected to the laptop via USB. The phone will prompt the user to authorize the laptop for ADB.

After the apk files have been installed the apps are started with the adb monkey command. A specific ssh configuration file that is used for all of the Chile phones is pushed to the phone's SD card. The SSh Server Pro settings need to be configured to pick up its configuration file from the SD card and to start on boot. To finalize the changes to the SSH Server Pro the app is restarted. In the Keep Running app's setting menu the monitor interval is set to 2 minutes, and the QED and SSH Server Pro apps are added to the list of apps that Keep Running monitors. QED should automatically have root access, but SSH Server Pro and the ADB daemon must be given root access manually. When we ssh into the phone and enter the su command the phone prompts the user to verify root access to SSH Server Pro. Similarly, the ADB daemon is given root access by issuing the adb shell command to start a shell on the phone. When the su command is given in the shell the phone will prompt the user to verify root access for the ADB daemon**.** The final step is to configure the phone to boot upon connection to power. The file that triggers the battery charging animation to begin when the phone is connected to power is altered so that instead of the animation beginning the phone will turn on. The file is replaced by opening a shell in the phone and becoming the super user. The /system folder is remounted as read write, and the original battery charging animation file is replaced with the altered file.

QED

The main app in the Chilean Smartphone EEW system is QED. QED collects data from the phone’s accelerometer and internal GPS. In addition to the internal GPS QED collects data from the external UBLOX NEO-7P gps chip via a Bluetooth connection. The default sample rate is 1Hz. QED streams the binary data with UDP to a central server that is running a MQTT broker. QED configures it’s settings by reading a settings configuration file that is hosted on a remote server. QED connects to the server every 60 seconds to check if there has been any change to the settings file, so we can change the sampling rate and other settings on the app remotely. When QED first connects to the settings server it receives a specific ID that is saved on the SD card. The ID’s were originally just an indexed number, but as of November 2018 they have been switched to be the phones IMEI number.

QED Configurator

QED connects to the UBLOX chip via Bluetooth. In order for the Bluetooth chip to be able to communicate with the UBLOX chip they must both be configured to operate at 38,400 bps. The QED Configurator app sets the UART ports for both chips to operate at 38,400 bps. To run the QED Configurator app the phone must be paired with the Bluetooth chip, and QED must not be running. After these conditions have been met the user opens the QED Configurator app and hits configure. The app will communicate with the Bluetooth chip to temporarily set it to 9,600 bps so it can communicate with the UBLOX chip. Then it configures the UBLOX chip’s UART port to operate at 38,400 bps, and switches the Bluetooth chip’s UART port to 38,400 bps. The app only needs to be run once. After the configuration process is complete the QED Configurator app can be closed and the QED app will be able to pull the external GPS data.

SSH Server Pro

The SSH Server Pro app gives the phone ssh capability which allows for remote access for updating the QED app, downloading QED debug logs for troubleshooting, and changing QED ID's on phones. The phones are setup with a settings file that is read from the SD card. The settings file specifies the user, password, and port.

Keep Running

The Keep Running app periodically checks to see if apps are running, and starts them if they are not. It is important that QED and SSH Server Pro are running at all times. If QED is down the phone will stop transmitting data, and if SSH Server Pro is down we lose the capability to update the apps and troubleshoot the phones. Both apps are designed to start when the phone boots up and to remain on. The Keep Running app is installed as a safety net. Keep Running has a list of all the apps installed on the phone. If the app in the list is selected Keep Running will periodically check if the app is running, and if it is not it will start it. We have Keep Running set to check that QED and SSH Server Pro are running every 2 minutes.

Appendix IV

R Code for Boostrapping Data to Analyze Drift in GNSS Sensors

~~~
```{r}

diff_lats <- function(DF) {
  means <- rep(0, length.out = 600)
  means_2sd <- rep(0, length.out = 600)
~~~

```
  oneSD_mean <- rep(0, length.out = 600)
  oneSD_2sd <- rep(0, length.out = 600)
  twoSD_mean <- rep(0, length.out = 600)
  twoSD_2sd <- rep(0, length.out = 600)

  for (i in 1:600){
    difference <- diff(DF$lat, lag=i)
    boot_means <- rep(0, length.out = 1000)
    boot_sd <- rep(0, length.out = 1000)
    boot_2sd <- rep(0, length.out = 1000)
    for (b in 1:1000){
      samp_pop <- sample(difference, length(difference), replace = TRUE)
      samp_mean <- mean(samp_pop, na.rm=TRUE)
      samp_sd <- sd(samp_pop, na.rm=TRUE)
      boot_means[b] <- samp_mean
      boot_sd[b] <- samp_sd
      boot_2sd[b] <- samp_sd*2
    }
    means[i] <- mean(boot_means)
    means_2sd[i] <- sd(boot_means)

    oneSD_mean[i] <- mean(boot_sd)
    oneSD_2sd[i]<- 2*sd(boot_sd)

    twoSD_mean[i] <- mean(boot_2sd)
    twoSD_2sd[i] <- 2*sd(boot_2sd)

  }
  data_list <- list(means, means_2sd, oneSD_mean, oneSD_2sd, twoSD_mean,
twoSD_2sd)
  return(data_list)

}
```

```{r}

diff_alts <- function(DF) {
  means <- rep(0, length.out = 600)
  means_2sd <- rep(0, length.out = 600)
  oneSD_mean <- rep(0, length.out = 600)
  oneSD_2sd <- rep(0, length.out = 600)
  twoSD_mean <- rep(0, length.out = 600)
  twoSD_2sd <- rep(0, length.out = 600)

  for (i in 1:600){
    difference <- diff(DF$alt, lag=i)
    boot_means <- rep(0, length.out = 1000)
    boot_sd <- rep(0, length.out = 1000)
    boot_2sd <- rep(0, length.out = 1000)
    for (b in 1:1000){
      samp_pop <- sample(difference, length(difference), replace = TRUE)
      samp_mean <- mean(samp_pop, na.rm=TRUE)
      samp_sd <- sd(samp_pop, na.rm=TRUE)
      boot_means[b] <- samp_mean
      boot_sd[b] <- samp_sd
      boot_2sd[b] <- samp_sd*2
    }
    means[i] <- mean(boot_means)
    means_2sd[i] <- sd(boot_means)

    oneSD_mean[i] <- mean(boot_sd)
    oneSD_2sd[i]<- 2*sd(boot_sd)

    twoSD_mean[i] <- mean(boot_2sd)
    twoSD_2sd[i] <- 2*sd(boot_2sd)

  }
  data_list <- list(means, means_2sd, oneSD_mean, oneSD_2sd, twoSD_mean,
twoSD_2sd)
  return(data_list)

}

```
```

Appendix V

R Displacement Detection and Measurement Model

```
North_gausian_noise <-
read.csv("/Users/jon/Documents/R_Workspace/EEW_Data/Bootstrapped_Data/nort
h/north_inbox_lattitude.csv")
North_gausian_noise <- North_gausian_noise[c(1:100),c(6,7)]
North_gausian_noise$foursd <- 2*(North_gausian_noise$mean_2sd +
North_gausian_noise$twosd_of_2sd)
#get the mean for normalizing v
mean_ns_lat=mean(north_south_UTM$lat)
mean_ne_lon=mean(north_east_UTM$lon)

mean_ss_lat=mean(south_south_UTM$lat)
mean_se_lon=mean(south_east_UTM$lon)

mean_ws_lat=mean(west_south_UTM$lat)
mean_we_lon=mean(west_east_UTM$lon)

mean_es_lat=mean(east_south_UTM$lat)
mean_ee_lon=mean(east_east_UTM$lon)

north_south_UTM$lat=(north_south_UTM$lat-mean_ns_lat)
north_east_UTM$lon=(north_east_UTM$lon-mean_ne_lon)

south_south_UTM$lat=(south_south_UTM$lat-mean_ss_lat)
```
```

```
south_east_UTM$lon=(south_east_UTM$lon-mean_se_lon)

west_south_UTM$lat=(west_south_UTM$lat-mean_ws_lat)
west_east_UTM$lon=(west_east_UTM$lon-mean_we_lon)

east_south_UTM$lat=(east_south_UTM$lat-mean_es_lat)
east_east_UTM$lon=(east_east_UTM$lon-mean_ee_lon)
timeseries_lat=function(DF, shift, model, title, ylable,xgrid=NA,
ygrid=NULL, uplim, lowlim)
{
DF$gpstime=(DF$gpstime/1000)
DF$gpstime=as.POSIXct(DF$gpstime, origin="1970-01-01")
tmin=min(DF$gpstime)
tmax=max(DF$gpstime)
shift_time=as.POSIXct(strptime(shift, "%Y-%m-%d %H:%M:%S"))
model_time=as.POSIXct(strptime(model, "%Y-%m-%d %H:%M:%S"))
plot(x=DF$gpstime,
     y=DF$lat,
     col="black",
     ylab=ylable,
     xlab=paste(tmin,"to",tmax,sep=" "),
     main=paste(title),
     type="l",
     panel.first=grid(nx=xgrid, ny=ygrid, col="lightgray",lty="dotted"),
     xaxs="i",
     yaxs="i",
     ylim=c(lowlim, uplim)
     )

abline(v=model_time, col="deepskyblue")
abline(v=shift_time, col="red")
lines(x=DF$gpstime, y=DF$lat, col="black", type="l")
legend("topright", c("Box Pushed", "Displacement Detected"), lty=1,
col=c("red", "deepskyblue"))


}

plt.measured.lat.disp <- function(xmin, xmax, diff_data, noise, title=NA,
ymax,seconds=100){
  indx <- which(abs(diff(diff_data$lat, lag = 1)) > noise[1,3])
  xmin <- as.POSIXct(xmin)
  xmax <- as.POSIXct(xmax)
  max_measurements <- rep(0, length.out=seconds)

  if (length(indx) <= 1){
      diffs <- data.frame("displacement" = abs(diff_data$lat[indx] -
                                              (diff_data$lat[indx+1])),
                           "start time of displacement" =
.POSIXct(diff_data$gpstime[indx]/1000), origin="1970-01-01")
```

```
  min_indx <- min(which(diffs$start.time.of.displacement > xmin))
  max_indx <- max(which(diffs$start.time.of.displacement <= xmax))

  plot(diffs[c(min_indx:max_indx),2],
     diffs[c(min_indx:max_indx),1],
     main=title,
     xlim=c(xmin, xmax),
     ylim=c(0,ymax),
     type="l",
     xlab="(S)",
     col=sample(viridis(20)),
     ylab="Displacement (m)")
  max_measurements[1] <- max(diffs[c(min_indx:max_indx),1])
  } else {
    plot( x=NA, y=NA,main=title,
     xlim=c(xmin, xmax),
     ylim=c(0,ymax),
     type="l",
     xlab="(S)",
     col=sample(viridis(20)),
     ylab="Displacement (m)")
  }
  for (i in 2:seconds){
    indx <- which(abs(diff(diff_data$lat, lag=i)) > noise[i,3])
    if (length(indx) >= 1){ #don't run if there aren't any diffs at all
    diffs <- data.frame("displacement" = abs(diff_data$lat[indx+i] -
diff_data$lat[indx]),
                       "start time of displacement" =
.POSIXct(diff_data$gpstime[indx]/1000), origin="1970-01-01")
    min_indicies <- which(diffs$start.time.of.displacement >= xmin)
    max_indicies <- which(diffs$start.time.of.displacement <= xmax)

    if (length(min_indicies) >=1 && length(max_indicies) >= 1){ #don't run
if there aren't any diffs in our window
      min_indx <- min(min_indicies)
      max_indx <- max(max_indicies)
    if (nrow(diffs) == 1) {
      points(diffs[c(min_indx:max_indx),2],
             diffs[c(min_indx:max_indx),1],
             col=sample(viridis(20)))
      max_measurements[i] <- max(diffs[c(min_indx:max_indx),1])
    } else {
      lines(diffs[c(min_indx:max_indx),2],
            diffs[c(min_indx:max_indx),1],
            col=sample(viridis(20)))
      max_measurements[i] <- max(diffs[c(min_indx:max_indx),1])

      }
    }
    }
  }
  total_max <- max(max_measurements)
  legend("topleft", legend = total_max, title="Max Measured Displacement
(m)")
}
```

Appendix VI

Python Code for Bootstrapping Network Latency

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import scipy.stats as stats

%matplotlib inline

cr_data=pd.read_csv('data/cr_latency_06182019.log', names=list(range(10)))
cr_data.columns=["arrive","phone","na1","sent","message","na2","na3","na4"
,"na5","na6"]

new=cr_data["arrive"].str.split(" ",n=1,expand=True)
cr_data["arrival"]=new[0]
cr_data["type"]=new[1]
cr_data.drop(columns=["arrive"],inplace=True)
cr_data.head()

cr_1=cr_data.loc[cr_data['phone'] == 351764100160056]
cr_2=cr_data.loc[cr_data['phone'] == 351764100160239]
cr_3=cr_data.loc[cr_data['phone'] == 351764100161112]
cr_4=cr_data.loc[cr_data['phone'] == 351764100169784]
cr_5=cr_data.loc[cr_data['phone'] == 351764100175435]
cr_6=cr_data.loc[cr_data['phone'] == 351764100181987]
cr_7=cr_data.loc[cr_data['phone'] == 351764100192406]
cr_8=cr_data.loc[cr_data['phone'] == 351764100192745]
cr_9=cr_data.loc[cr_data['phone'] == 351764100193867]
cr_10=cr_data.loc[cr_data['phone'] == 351764100220074]
cr_11=cr_data.loc[cr_data['phone'] == 351764100221197]
cr_12=cr_data.loc[cr_data['phone'] == 351764100221221]
cr_13=cr_data.loc[cr_data['phone'] == 351764100221270]
cr_14=cr_data.loc[cr_data['phone'] == 351764100222344]
cr_15=cr_data.loc[cr_data['phone'] == 351764100223177]
cr_16=cr_data.loc[cr_data['phone'] == 351764100224522]
cr_17=cr_data.loc[cr_data['phone'] == 351764100237862]
cr_18=cr_data.loc[cr_data['phone'] == 353780102160025]
cr_19=cr_data.loc[cr_data['phone'] == 353780102163326]
cr_20=cr_data.loc[cr_data['phone'] == 353780102169166]
cr_21=cr_data.loc[cr_data['phone'] == 357665081645250]
```

```
a_messages=cr_data.loc[cr_data['type'] == 'A']
e_messages=cr_data.loc[cr_data['type'] == 'E']
i_messages=cr_data.loc[cr_data['type'] == 'I']
o_messages=cr_data.loc[cr_data['type'] == 'O']
s_messages=cr_data.loc[cr_data['type'] == 'S']
t_messages=cr_data.loc[cr_data['type'] == 'T']

cr_data.groupby('type').count()

times_cr=cr_data.loc[:,['arrival','sent']]

arrival_cr=times_cr['arrival'].values

sent_cr=times_cr['sent'].values

arrival_cr=arrival_cr.astype(float)

arrival_cr=arrival_cr*1000

diffs_cr=np.subtract(arrival_cr, sent_cr)

diffs_cr=np.around(diffs_cr)


bootstrap_meandiffs_cr=np.array([(np.random.choice(diffs_cr,len(diffs_cr),
replace=True).mean()) for _ in range(10000)])

diffs_cr_ci_mean=np.percentile(bootstrap_meandiffs_cr, [2.5,97.5])
diffs_cr_ci_mean
```