Draft of the Final Report for
“Development of Low Cost Seismometer System”

Submitted to

Mr. Jason Arthur Blink

AND

Mr. Wit Sumathavanit

By

Mr. Chanaropv Vichalai

Faculty of Engineering and Architecture

Rajamangala University of Technology Suvarnabhumi,

Suphanburi 72130, Thailand

Correspondence: chanaropv@gmail.com, mobile (66) (0)89-1831984

3rd October 2019

Table of Content

Exclusive Summary

The prototype of this seismometer was developed based on the commercial home/hobby seismometer system, Raspberry shake. The system was improved on the timing redundancy, temperature shield, long range data telemetry, and the solar power system.

This prototype can be improve to a commercial/research by reverse engineering process to make new system which included the self-timer, gps timing and radio data telemetry into a single board. The software part can be improve and adapt from the sources code that installed on the system

1 Introduction

This project was aimed to develop the inexpensive, affordable and believable seismic monitoring system as it can deploy remotely away from local internet connection of around 10 km++. The proposal was summited and proposed the seismic monitoring system is possible to develop from a single board computer with a commercial Data Acquisition Unit (DAQ) and other electronic modules available in the market.

The research agreement of this project was signed on 20th November 2018 and the system will be delivered to the funder within 31st March 2019. According to the in house official documents process, part lists ordering from overseas, customs process and system testing that caused delay of the project for a month.

At the first stage, while the first instalment claim to the University, the research conclusion of the DAC for this project is the RaspberryShake 3D as it is a personnel seismometer that available in the market and it recognizes by USGS [1-2] and other academic research institutes [3-4]. We have done the circuit analysis and improvement of it to make it run remotely.

Objectives

1) To development of affordable seismic monitoring system.

2) To implement and test the developed seismic monitoring system on data telemetry and real-time data monitoring.

3) To improve the design for further developed seismic monitoring system.

2 System Building

2.1 Digitizer

At the first stage, the research team searching for an affordable data acquisition system that can be used on this project. Many ADC devices were identified in depth follow and described in Table 1.

The conclusion of our team the DAQ for this project is the RaspberryShake 3D as it is a personnel seismometer that available in the market and it recognizes by USGS [1-2] and other academic research institutes [3-4] which we believed it is good enough for this project.

The RaspberryShake was developed based on OSOP Sixaola short-period six-channel seismograph [5], the name was changed to “RS Pro Seismograph” in 2018. In July 2016, the Raspberry Shake 1D was launched on Kickstarter and it was a resounding success, receiving an award for innovation in Panama from SENACYT and huge interest from consumers and institutes alike. Consumers could access a plug and play professional home science monitor and institutes could easily densify their existing networks with an affordable device that could hold its own against much more expensive models. The RaspberryShake data acquisition board was tested before shipped out from the factory.

This project the RaspberryShake 3D board was used as a digitizer. This board is an inexpensive 24-bit (144 dB) digitizer that is a three dimensional (3D) version capable of detecting and recording earthquakes and vibrations in all directions which are designed for serious earthquake aficionados with some extra cash, scientists and geophysical institutes.

The small current from geophone sensor is then amplified with some ultra-quiet state of the art op amps. Once amplified, the signal is digitized, then that data is shipped to the ARM processor and bundled into one-second packets that are shipped to your Raspberry Pi. The anatomy of RaspberryShake 3D board is shown in Figure 1.

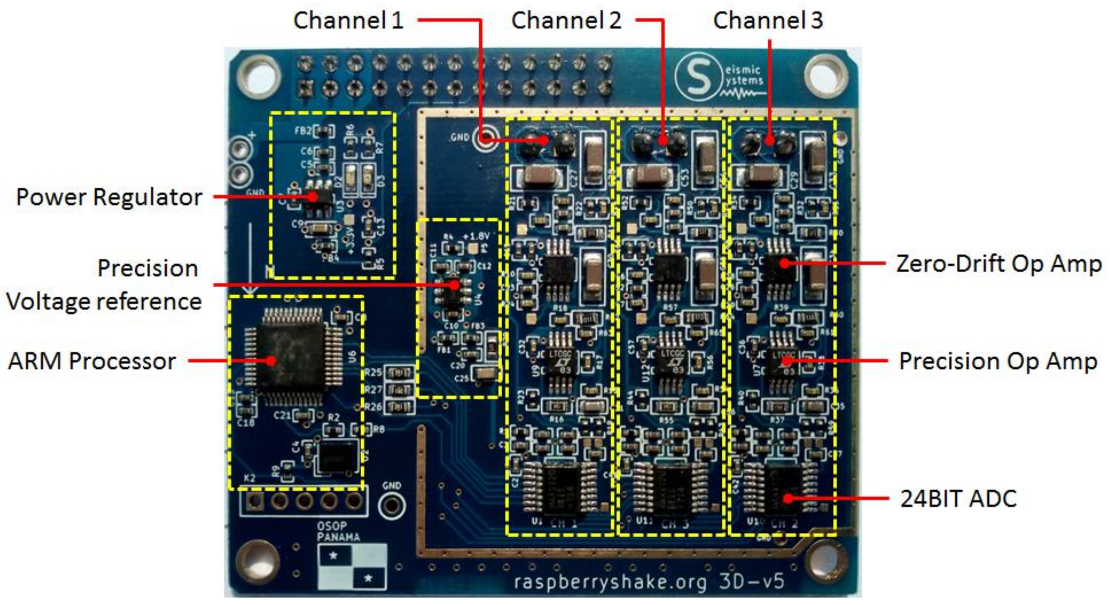


Figure 1 Raspberryshake 3D

Table 1 DAQ details

Model/Brand name	Advantage	Disadvantage
NI DAQ / Other device that similar to NI DAQ	Well known and easy to use.	Cost varies with the ADC resolution. Need licensed software to run.
Arduino and other 8-16 bits microcontrollers	Open-source and open hardware. Inexpensive and low power consumption.	Low computing power and memory. The accuracy depends on ADC module.
RaspberryShake	Became widely used by many organization and academic institute in a short time. Well designed for earthquake application and it came with source code that adapts for future work.	Non-open hardware which some part number was erased.
Other single board computer	More powerful than Raspberry pi	Difficult to find in the market and it is not stable.

2.2 Computer/Controller Board

RaspberryShake 3D board is a shield that compatible with almost all Raspberry Pi models. This single board computer is running its own OS based on the Debian Linux system to handle multiple programs at a time. Compare with other microcontrollers, such as Arduino, it can be run a single program code again and again.

Raspberry's advantage includes an easy connection with the internet, complete support from Linux community is available and the choice of programming language can be huge.

The disadvantages of Raspberry include no support for real-time hardware, the interfacing can be delayed if CPU is busy, inductive loads cannot be driven, analogue to Digital converter is not available and hardware design is not open source.

The Raspberry Pi 3 Model B+ (figure 2) was used for this project as the latest product in the Raspberry Pi 3 range. The specification of this single board computer is the following:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput of 300 Mbps)
- The extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port
- Micro SD port for loading your operating system and storing data
- 5V/2.5A DC power input
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)

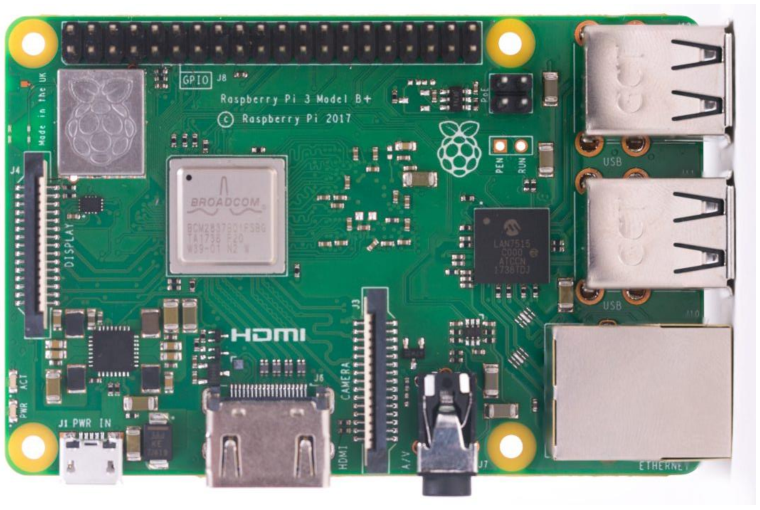


Figure 2 Raspberry Pi 3 Model B+

2.3 Geophone Sensor

The main sensor is a geophone, there is a microphone for the vibrations sensing of the earth movement. The coil inside a geophone that moves in relation to a magnet and creates a small current, as shown in figure 3.

There are three commercial 4.5 Hz (figure 4) geophone sensors for seismic explorations were used for this project that aimed to minimize the overall size of a seismic recording system reduces the cost of the instrument. The photo of these geophone sensors are shown below.

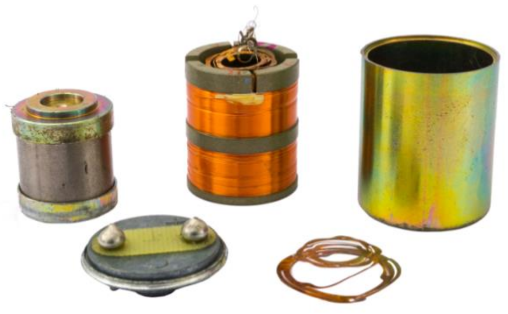


Figure 3 geophone components

Figure 4 The 4.5Hz geophone sensors

2.4 Timing

The important information rather than the vibration data is timing. The seismometer records seismic wave or ground motion and time synchronized so that the resulting seismograms can be analyzed systematically.

The earthquake epicenter determination can be analyses from seismograms from a few or many monitoring stations. One can determine the origin time and location of the event and estimate its magnitude (M) which is a measure of energy released by an earthquake.

This research the GPS time was used for the seismometer system. The network timing protocol (NTP) was disable, the GPS timing had set as timing server that allow this system can deploy remotely.

The GPS type is doesn't matter that is a consumer grade or survey grade due to the most important data is just the time not a position. But the most important is sensitivity of the GPS that allow us to receive more satellite. The Ublox NEO-M8N NEO M8N GPS Module was used for this project. This GPS module support BeiDou, Galileo, GLONASS and GPS / QZSS which has 72 channels.

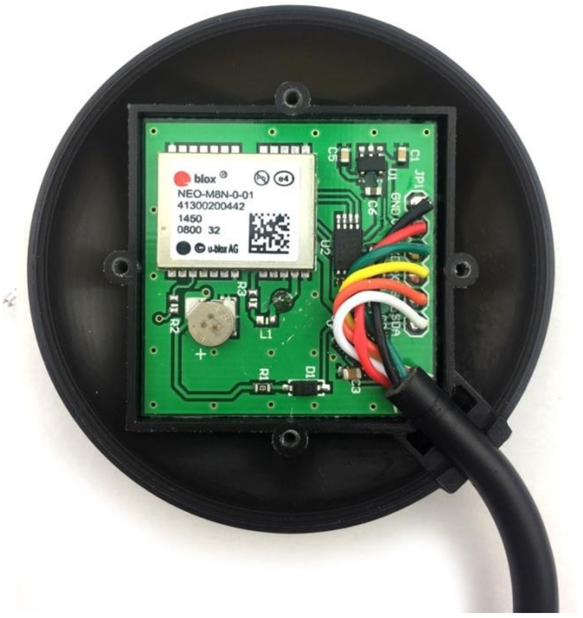


Figure 5 Ublox NEO-M8N NEO M8N GPS Module

2.5 Data Telemetry System

There is several telemetry architectures were taken into account, due to the bandwidth of data stream is about 512 kB the most suitable wireless telemetry system is WIFI. The details of several telemetries are described in the table below.

Table 2 Telemetry type that possible for this project

Telemetry type	Data rate	Frequency	Range	Power consumption
Serial radio telemetry	250kbps	433MHz, 915MHz	40Km++	4.1W
Lora	21.9kbps	902-928MHz	15Km	412.5mW
Zigbee pro 900	9.6kbps	900MHz	40Km++	854mW
XTend 900 1W RPSMA	115.2kbps	900MHz	60Km++	3.65W
XBee Pro 63mW RPSMA	250kbps	2.4GHz	1.6Km	974mW
Bluetooth 5.0	1Mbps	2.4GHz	150m	124mW
Z-wave	100Kbps	900MHz	30m	124mW
WIFI	150Mbps+	2.4GHz	50m	49mw++

According to the wide band width but limited range of the WIFI data telemetry, we have designed to increasing the transmission power by using the high gain antenna (Yagi antenna) and the signal booster to increase the telemetry range up to 10 km++. The diagram of this telemetry is shown in figure 6 below.

In some cases to extend the range or to avoid the obstruction, the repeater unit can be easily setup by using a WIFI access point with two WIFI boosters and two Yagi antennas.

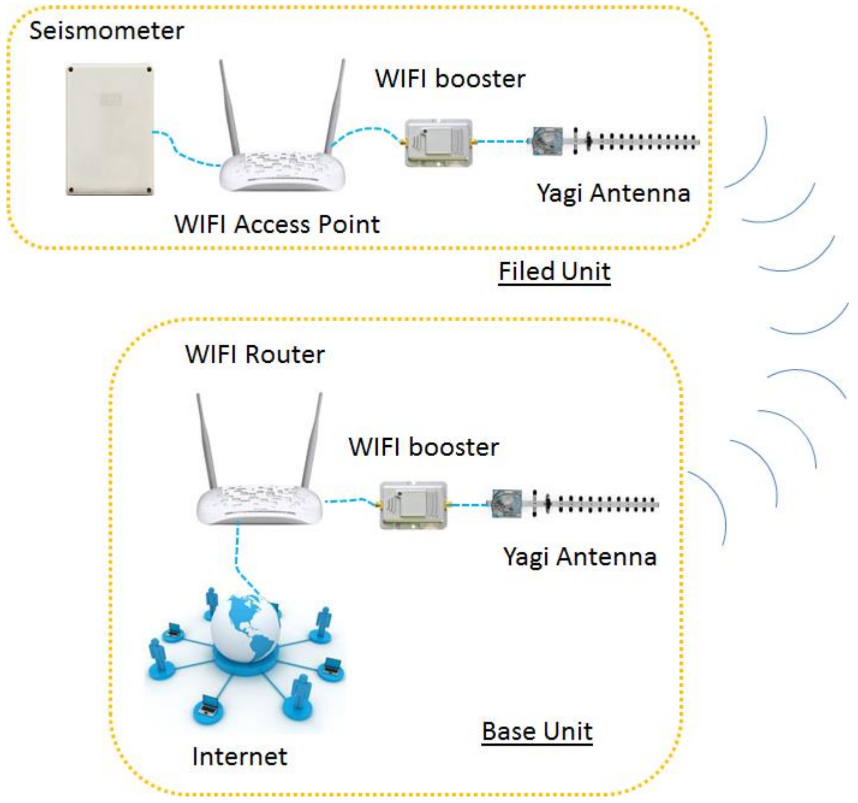


Figure 6 long range WIFI telemetry system

2.6 Solar Power System

The solar power unit was designed for standalone installation in the remote area that can continue working for 5-7 days with no sunlight. The system consists of the solar cell, battery, and solar charge controller.

The solar charge controller is a device that manage a power charge and working as a power supply for the seismometer system.

The DC-DC converter was used to reduce the DC voltage form battery prior input the device. The solar power diagram is shown in figure 7.

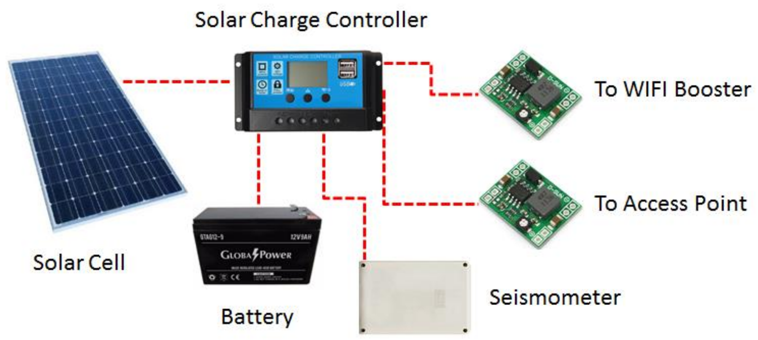


Figure 7 Solar power system diagrams

2.7 Complete Seismometer System

This prototype system is design to be easy to be deploy and robust. The system was divided into two units, one is field unit and the other is base transceiver unit.

A field unit is consisting of two boxes, one is seismometer and the other box is consisting of WIFI telemetry and solar charge controller. The diagram of the field unit is shown in figure 8. The photo of filed unit is shown in figure 9.

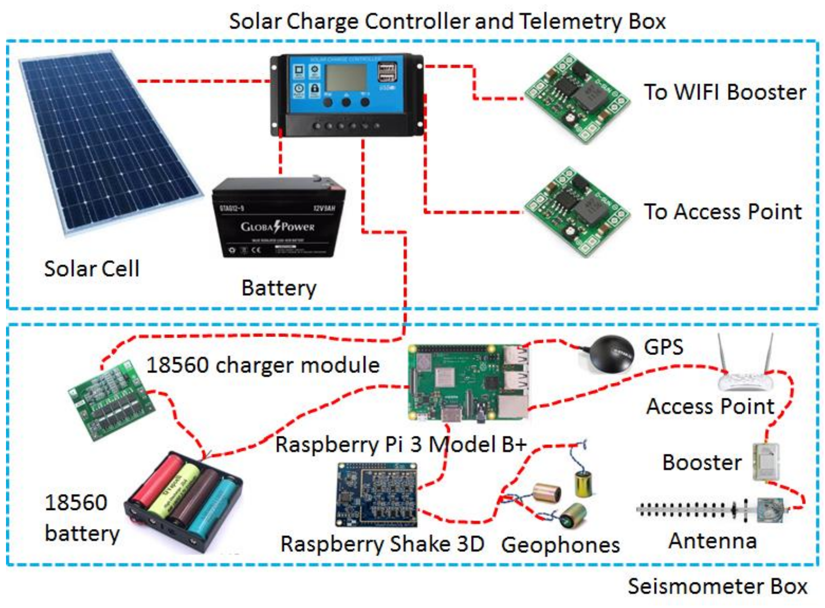
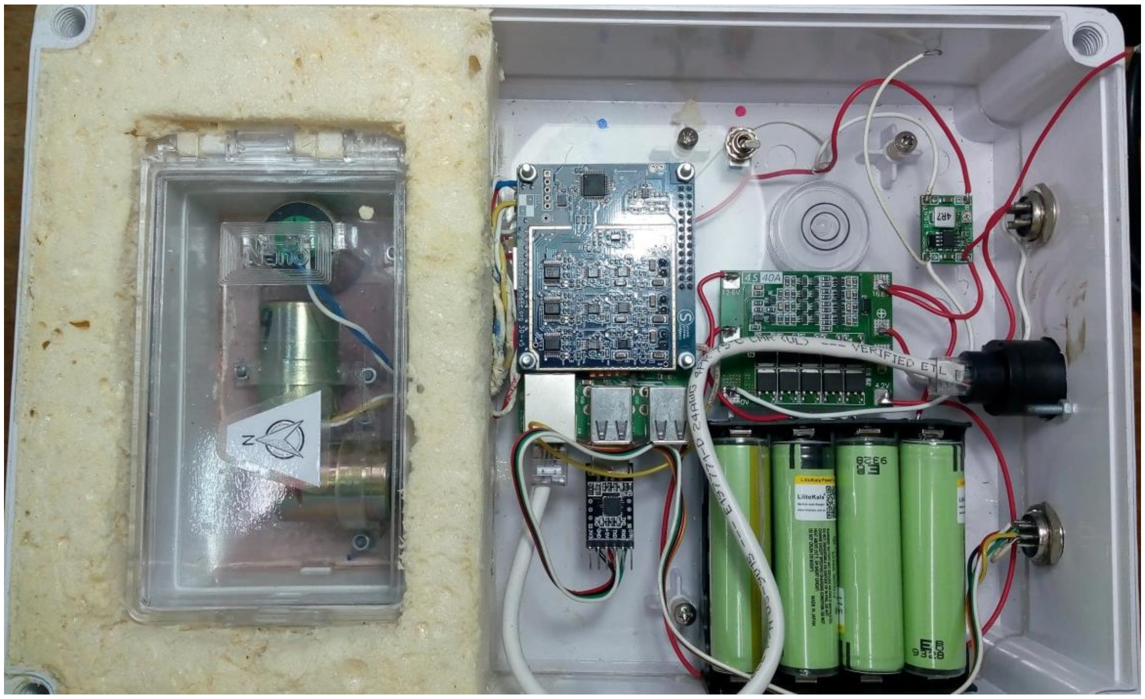
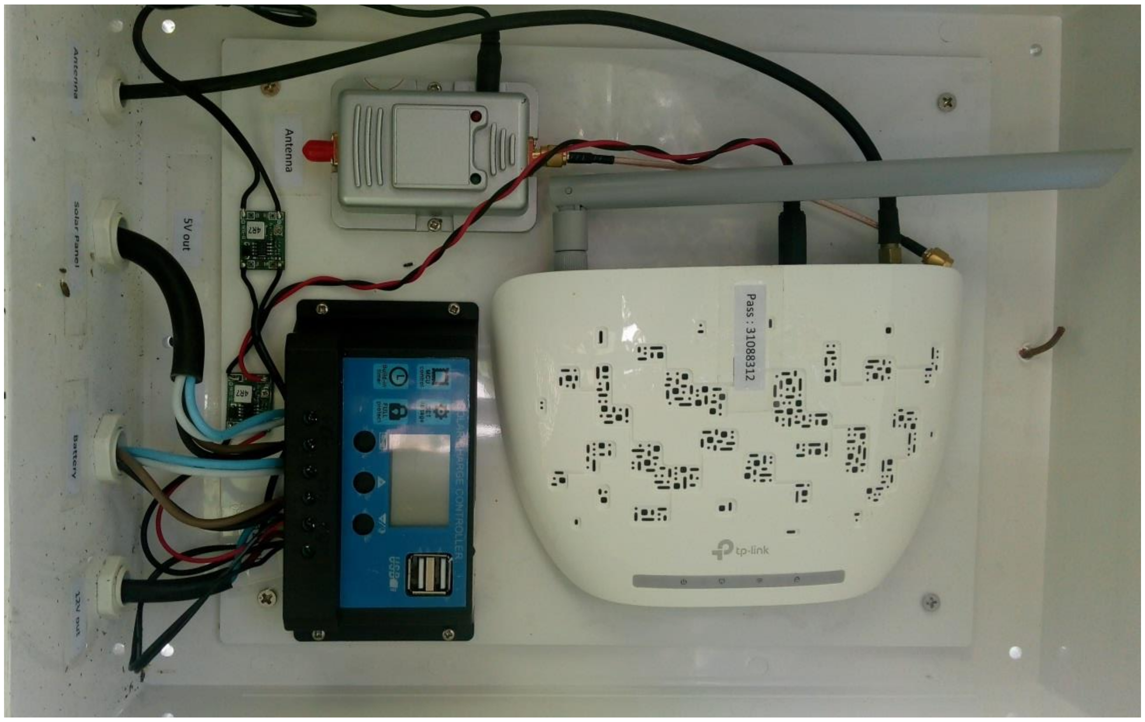


Figure 8 Field unit diagram


a) Seismometer box



b) Telemetry and solar charge controller box

Figure 9 Photo of the field unit

3 System Testing

The choice of which seismic instrumentation for a seismic network or field study is filled with trade-offs. Size, ease of deployment, power consumption, data format and storage, cost, and performance of the sensor must all be taken into account. Of these, performance of the system in terms of noise dynamic range, and ability to recover true ground motion across a broad range of frequencies may be the most difficult for users to intuit and assess. However, the best-performing sensors are not needed for all applications, which is advantageous because sensor performance is often a direct trade-off with sensor cost. Thus, to maximize use of available funds while still accomplishing the goal of recording the seismic signal of interest with acceptable fidelity, it is necessary for network designers and researchers to understand how sensor performance is measured and the meanings of the various metrics used to quantify this performance. Additionally, data users must understand any limitations in the fidelity of seismic waveforms that arise through the performance of different sensors.

The constructed seismometer system need to be tested both filed unit and base station unit. Test program included in house and field test follow.

3.1 Field testing

The assembled system was has tested on the field according to ensure the data telemetry is working properly. The first test we have done with a distance of 1.7 km using standard WIFI router with 6dB-Omni WIFI antenna which the base station located at point A and field unit is located at point B, as shown in figure 10.

The second test we need to achieve 10 km++ range of data telemetry, the base station still located at same position (Point A). We have move the filed unit to point C which has a distance between these stations of 10.8 km, as shown in figure 11. The WIFI data telemetry that had been used for the first location was not work for this second location. The first solution we had changed the antenna from Omni antenna to Yagi antenna (14E) which has gain of 20dBi on both base and field station. This setup still not success to achieve the range of 10 km+ as aimed.

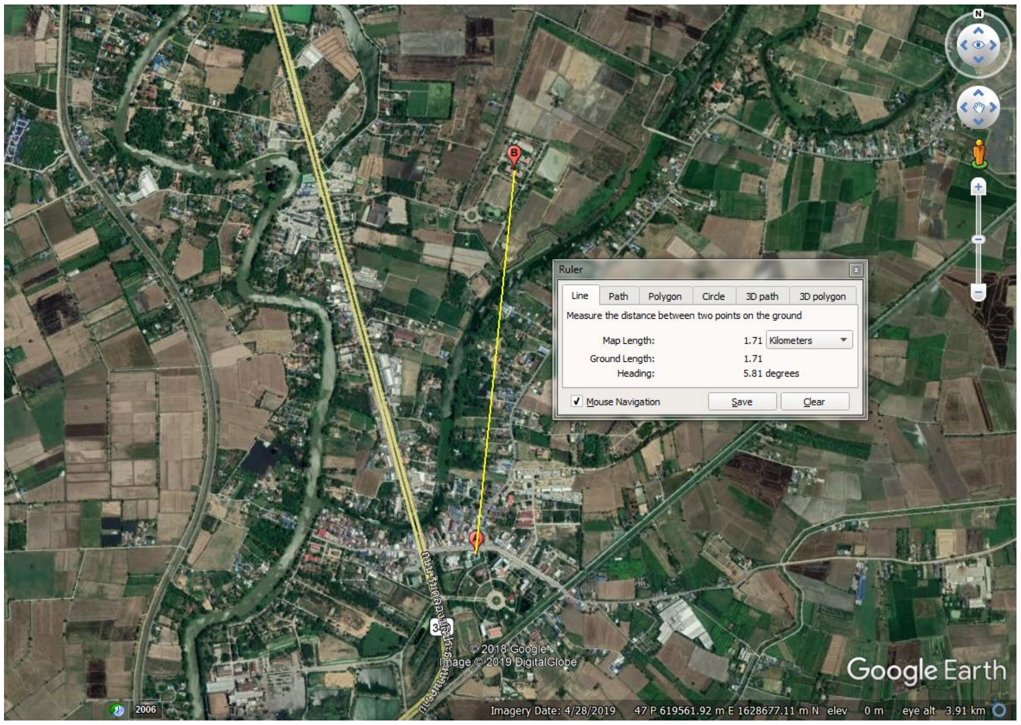
The new setup using the high power USB WIFI adapter, as shown in figure 12, was employed for this test. This WIFI adapter has 1W to replace with the WIFI router. This setup is not work.

The final setup was tested with improve the transmission power by using 2W WIFI signal booster on both base station and field station. The data transmission over 10km was successful with this setup. The photo of base and filed stations are shown in figure 12-13

According to increasing transmission power to achieve long range data telemetry, the power consumption of the system was increased. Table 4 shows the power consumption from those four setups that had been done for this project.

Table 3 Seismometer test positions.

Point	Latitude	Longitude	Easting	Northing	Type of station
A	14°43'17.24"N	100° 6'33.70"E	619424 E	1627812 N	Base
B	14°44'12.64"N	100° 6'39.36"E	619585 E	1629515 N	Field
C	14°48'59.43"N	100° 7'55.22"E	621809 E	1638339 N	Filed

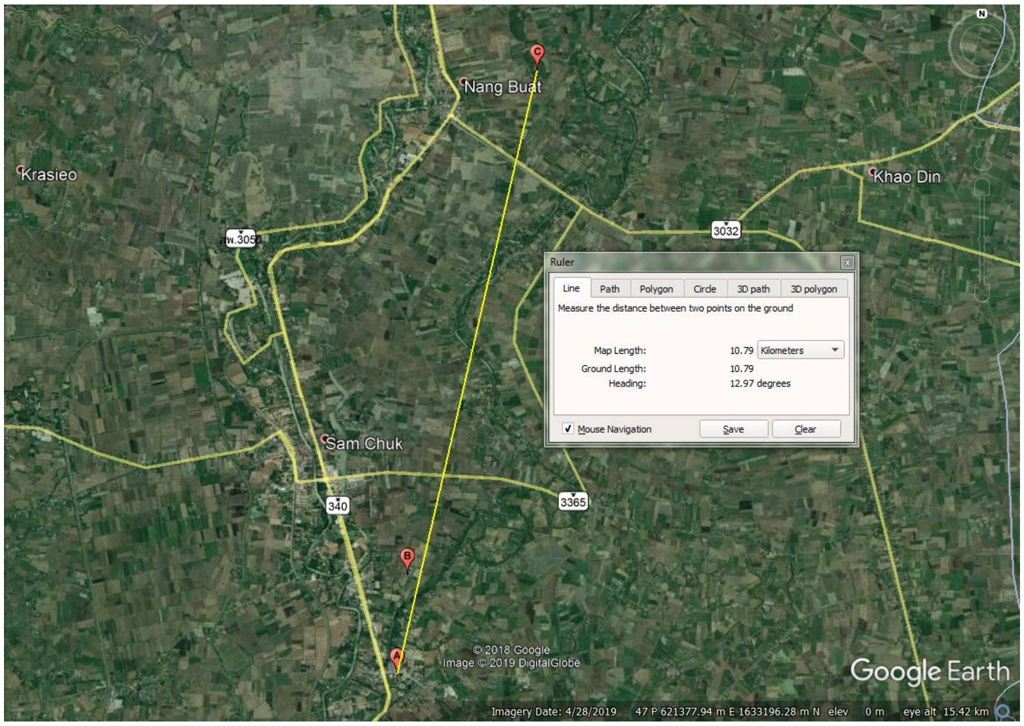


Measure the distance between two points on the ground

Map Length: 1.71 Kilometers
Ground Length: 1.71
Heading: 5.81 degrees

Imagery Date: 4/28/2019 47 P 619561.92 m E 1628677.11 m N elev 0 m eye alt 3.91 km

Figure 10 First location of field testing from A to B



Measure the distance between two points on the ground

Map Length: 10.79 Kilometers
Ground Length: 10.79
Heading: 12.97 degrees

Imagery Date: 4/28/2019 47 P 621377.94 m E 1633196.28 m N elev 0 m eye alt 15.42 km

Figure 11 Second location of field testing from A to C

Table 4 The power consumption of field testing for the Field unit only.

Setup No.	Details	Power Consumption (Watt)	Range (km)
1	Standard WIFI router+6dB-Omni antenna	6	1.7
2	Standard WIFI router+20dB-Yagi antenna	6	1.7+
3	High power USB WIFI adapter (1W) +20dB-Yagi antenna	8	1.7+
4	Standard WIFI router+20dB-Yagi antenna+WIFI booster	30	10.7+

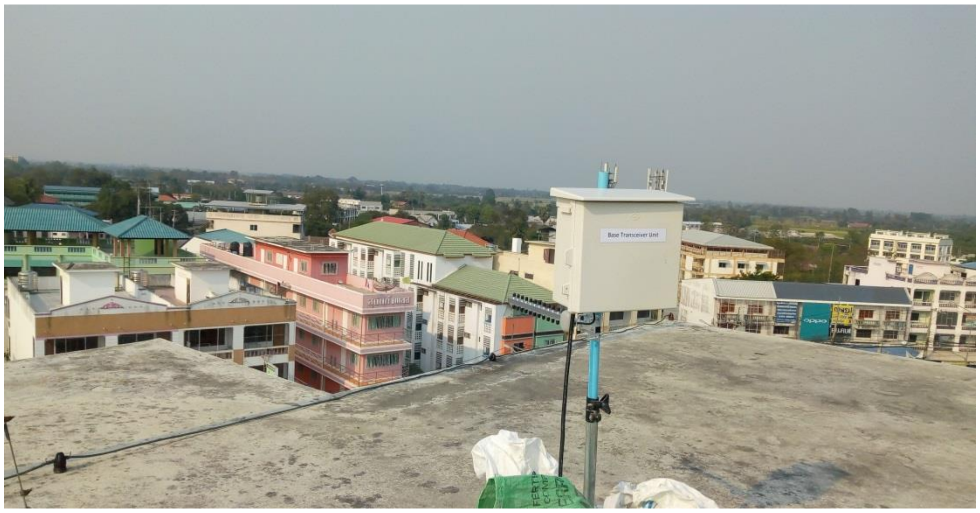


Figure 12 The Base station setup

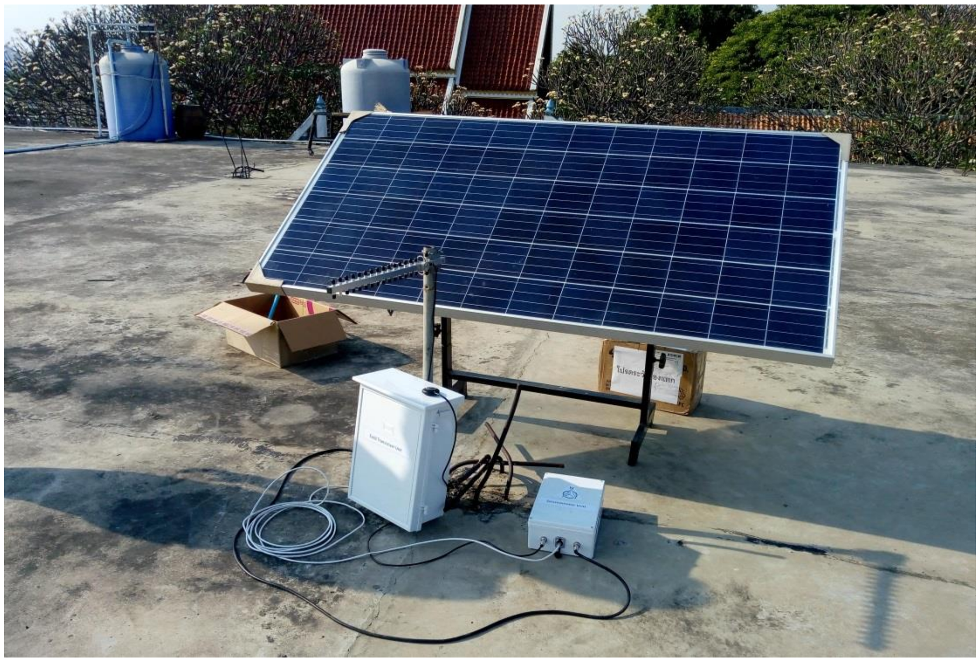


Figure 13 The field unit setup

3.2 In house testing

According to risk of equipment damaged or lost, after the data telemetry test over the range of 10 km was achieved, the field unit was moved to the campus for debugging and monitoring test. The system installation is shown in figure 13.

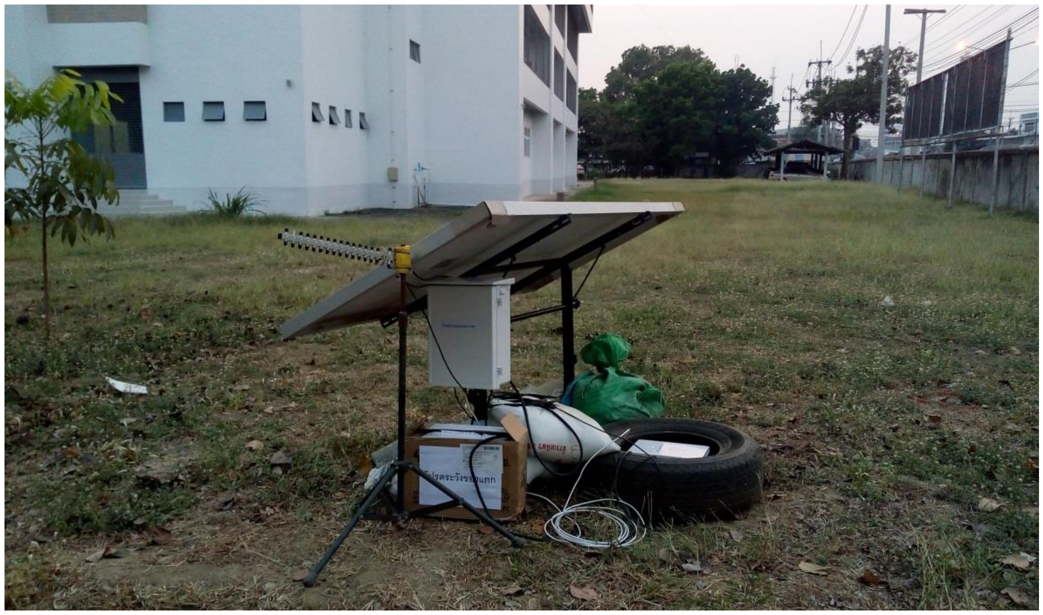


Figure 13 The in house field unit setup

4 Results

From this setup the power consumption of the field unit is 30W or 2.5Ahr and the solar power system has a 100 Ah battery. The solar power will be powering the system for around 5 days with no sunlight.

The recorded result is shown in Figure 14

5 Conclusions and Discussions

The prototype of this seismometer was developed based on the commercial home/hobby seismometer system, Raspberry shake. The system was improved on the timing redundancy, temperature shield, long range data telemetry, and the solar power system.

This prototype can be reverse engineering to make new system which included the self-timer, gps timing and radio data telemetry into a single board.

The power consumption of this system is quite high according to the WIFI signal boosters. The optimization design of radio transmitting power need to be made for each station.

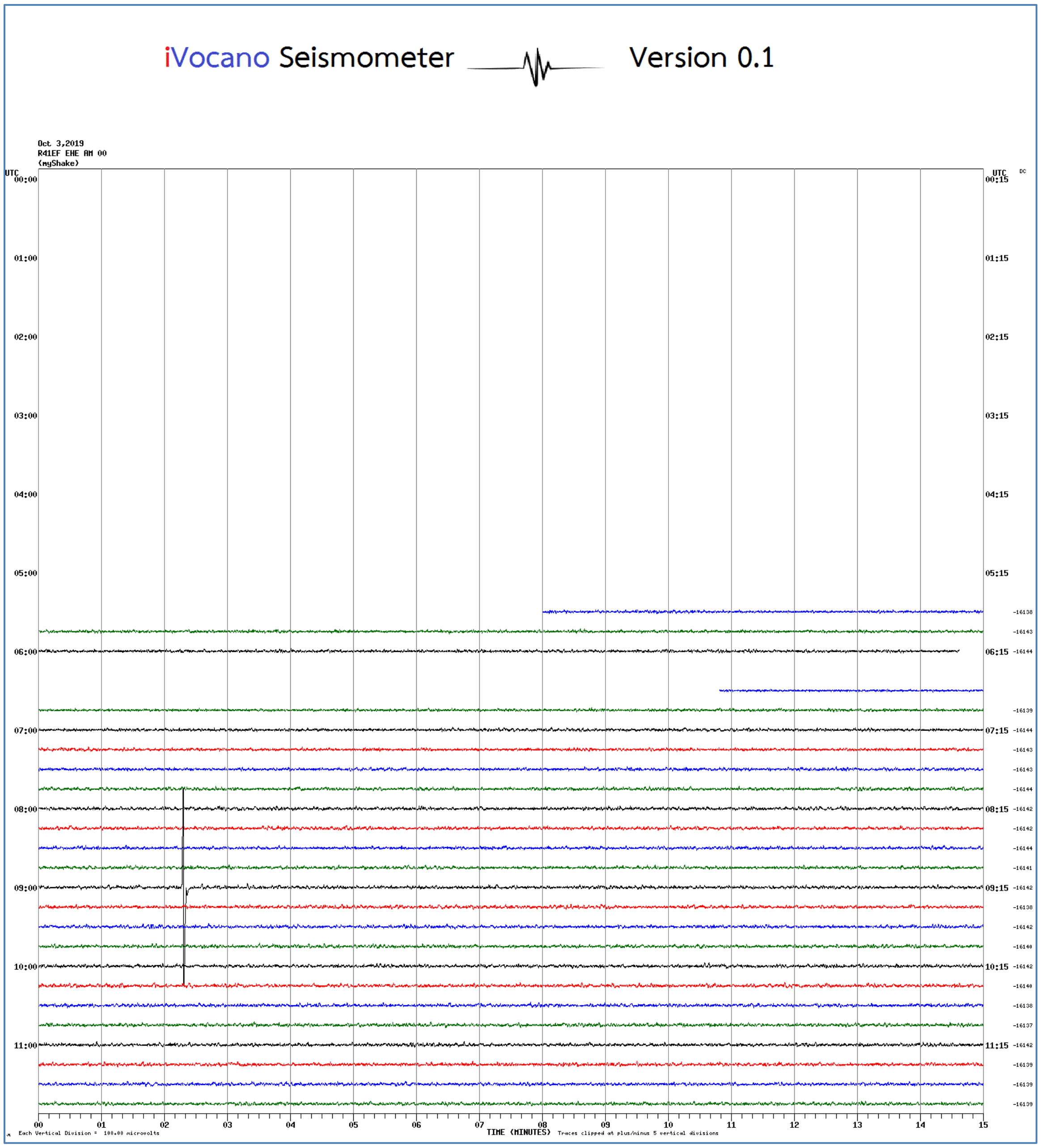


Figure 14 recorded data (Helicorder Displays)

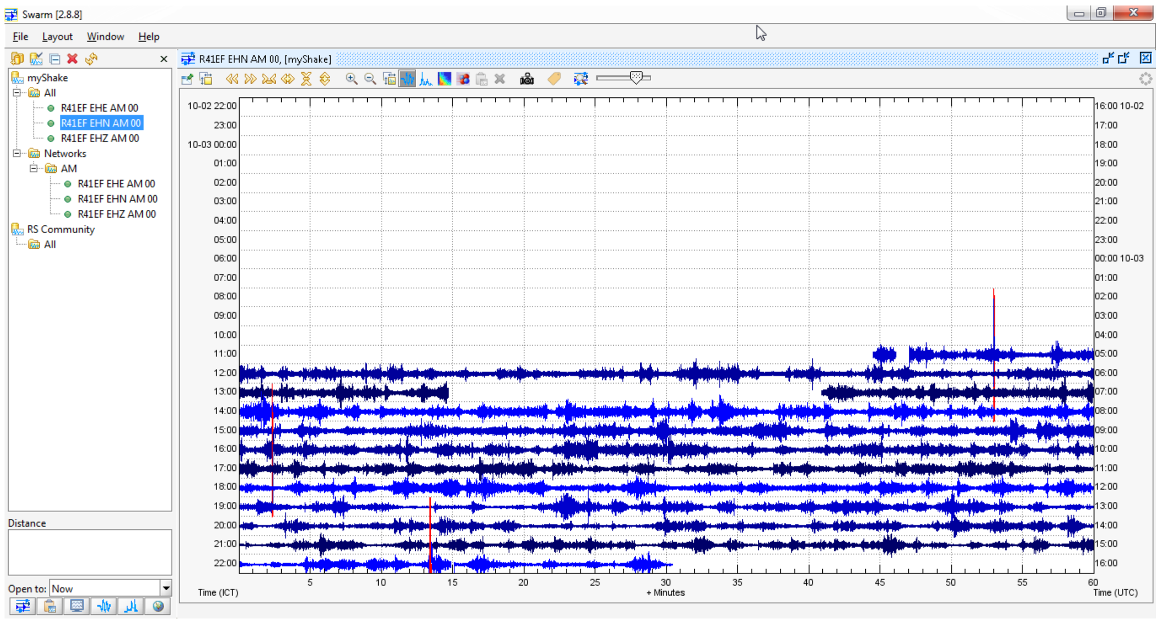


Figure 15 recorded data (SWARM Displays)

References

[1] USGS (2018) *New instruments installed to measure Arctic coastal erosion*. Raspberry shake 1D. Retrieved December 3, 2018, from https://www.usgs.gov/center-news/new-instruments-installed-measure-arctic-coastal-erosion?qt-news_science_products=1#qt-news_science_products

[2] Robert E. Anthony, Adam T. Ringler, David C. Wilson, Emily Wolin. (2018) *Do Low-Cost Seismographs Perform Well Enough for Your Network? An Overview of Laboratory Tests and Field Observations of the OSOP Raspberry Shake 4D*. Seismological Research Letters Vol. 90 (1) pp. 219-228.

[3] Andrea Manconi, Velio Coviello, Maud Galletti, and Reto Seifert (2018) *Short Communication: Monitoring rockfalls with the Raspberry Shake*. Earth Surface Dynamics. An interactive open-access journal of the European Geoscience Union. 6, 1219–1227. Retrieved December 3, 2018, from https://www.earth-surf-dynam.net

[4] Jay Pulli (2019) *Array Processing of Raspberry Shake Data from Washington D.C. & Boston*. Conference: Monthly Meeting of the Potomac Geophysical Society. Retrieved December 3, 2018, from https://www.researchgate.net

[5] OSOP (2018) OSOP Sixaola short-period six-channel seismograph. Retrieved April 24, 2019, from https://raspberryshake.org/

Appendix 1 : Wiring Diagram

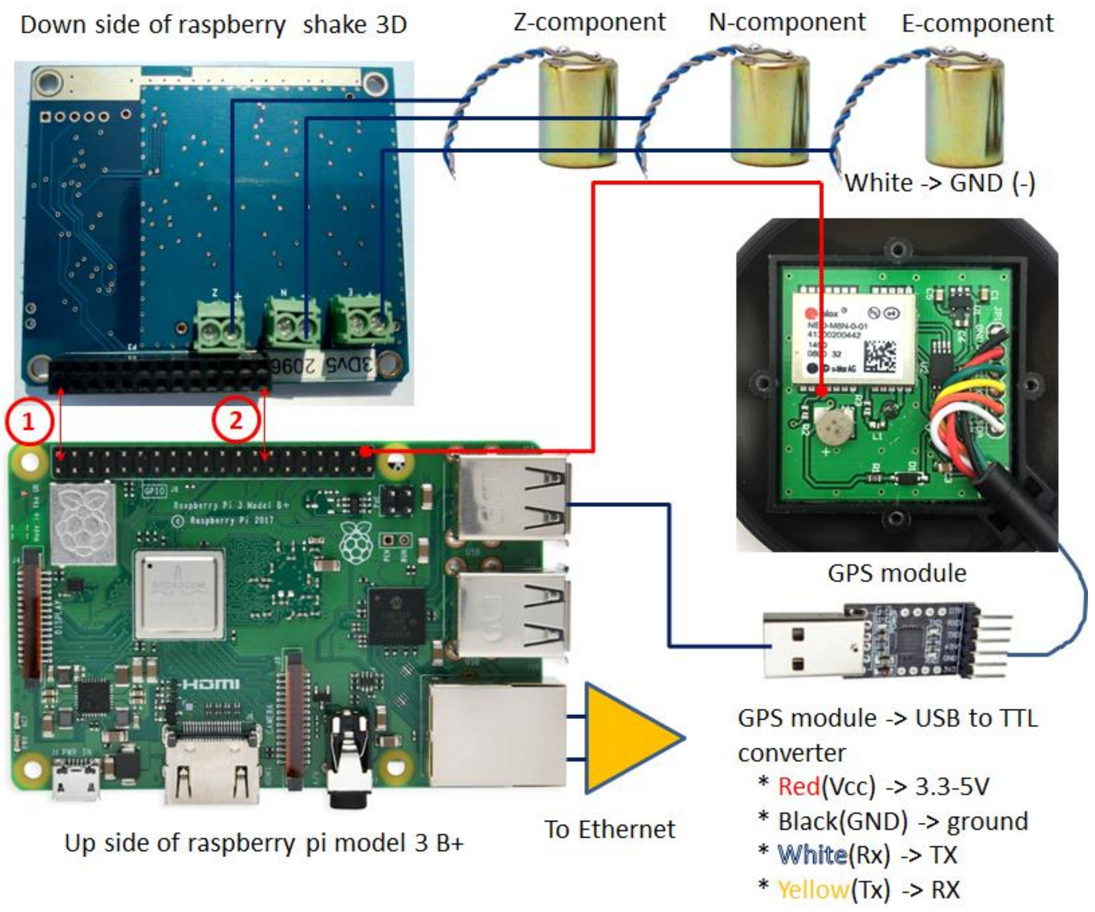


Figure A1-1 Seismometer meter unit (recording devices)

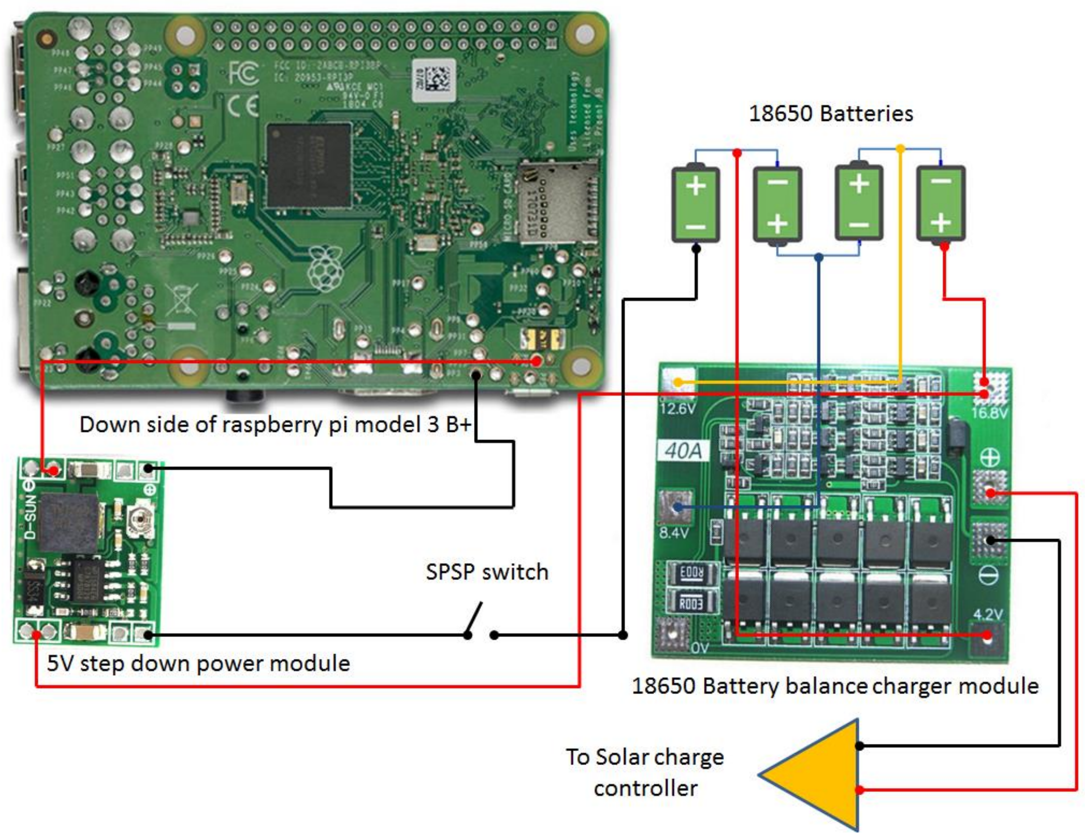

18650 Batteries
Down side of raspberry pi model 3 B+
SPSP switch
5V step down power module
12.6V
16.8V
40A
8.4V
4.2V
0V
18650 Battery balance charger module
To Solar charge controller

Figure A1-2 Seismometer meter unit (power supply)

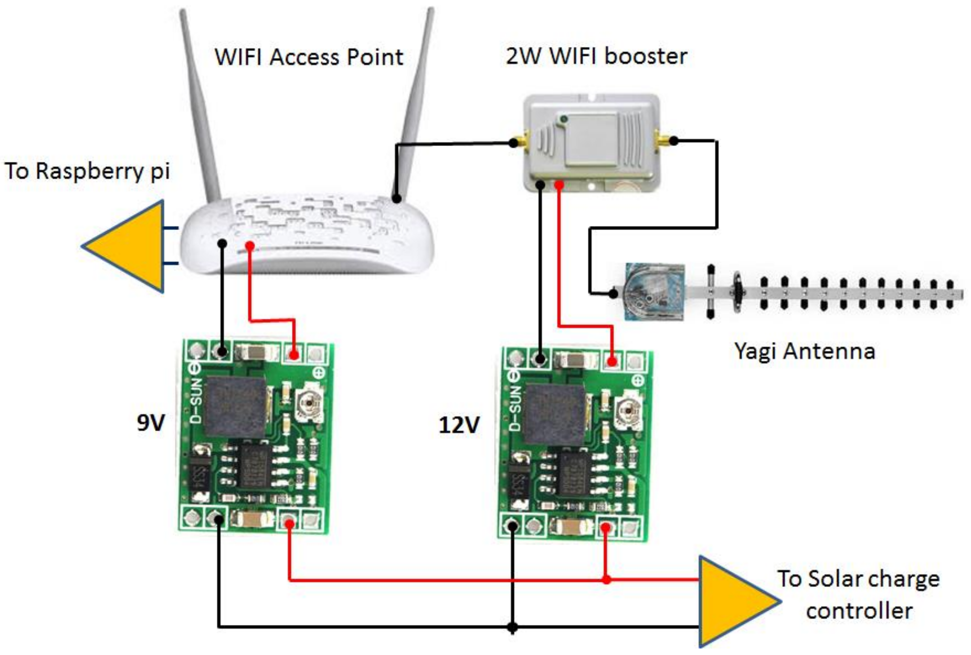


Figure A1-3 Seismometer meter unit (WIFI telemetry)

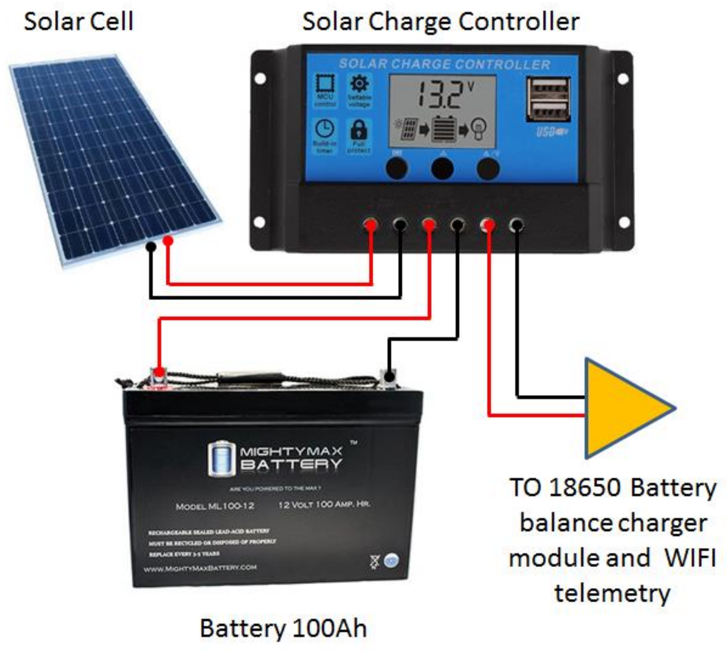

Solar Cell
Solar Charge Controller
SOLAR CHARGE CONTROLLER
13.2V
MIGHTYMAX BATTERY
MODEL ML100-12 12 VOLT 100 AMP. HR.
TO 18650 Battery balance charger module and WIFI telemetry
Battery 100Ah

Figure A1-4 Seismometer meter unit (Solar power system)

Appendix 2 : Firmware installation

1 Firmware setup

1.1 microSD cards

Use commercial-grade (MLC) or industrial-grade (SLC) microSD cards to prevent the memory damage due to lag of power supply. The consumer-grade TLC microSD card is **not recommended to use with this system**.

1.2 Micro SD card preparation

This project the seismometer system was run by "raspishake-release_V15" which can download at https://gitlab.com/raspberryShake-public/raspShake-SD-img.

The downloaded firmware image file needs to write to the SD card. Etcher was use for this project which is a graphical SD card writing tool that works on Mac OS, Linux and Windows, and is the easiest option for most users. Etcher also supports writing images directly from the zip file, without any unzipping required. Etcher can download at https://www.balena.io/etcher/.

To write your image with Etcher:

- Download Etcher and install it.
- Connect an SD card reader with the SD card inside.
- Open Etcher and select from your hard drive the Raspberry Pi .img or .zip file you wish to write to the SD card.
- Select the SD card you wish to write your image to.
- Review your selections and click 'Flash!' to begin writing data to the SD card.

1.3 Firmware installation

Insert the micro-SD card into the Raspberry Shake's Raspberry Pi computer and power up the device. The Raspberry Pi computer will begin the Raspberry Shake software image unpacking task automatically. Wait until the unpacking task completes and your Raspberry Shake has re-booted.

The unpacking time is depending on the card size and type, this will take between 10 and 17 minutes. During this process the LED lights will display Blue-solid on the Raspberry Shake board, and red & green solid on the Raspberry Pi computer. When completed, the LED's will be blue solid on the Raspberry hake board, with red solid and green blinking on the Raspberry Pi computer.

Next step is to find out an IP address of raspberry pi on the network belonging, the device named "Raspberry Shake".

1.4 Changing the password

This can only be done when connected to the Pi over an internet connection and can access the desktop screen.

1. Connect the Shake to the router using an Ethernet cable
2. Open the Raspberry Shake's webpage at ***http://raspberryshake.local/*** or type ***http://***<IP_Address_here>

Changing the Password

1. Click on the gear icon and navigate to 'Actions'
2. Change ssh password
   Username: myshake
   Default ssh password: shakeme

1.5 Waveforms on disk

On the Raspberry Shake, continuous waveform data, as that seen with Swarm, are saved in miniSEED format to: "/opt/data/archive"

The archive structure is: YEAR/NETWORK/STATION/CHANNEL/<DAILY MINISEED FILES>

Each daily miniSEED file is generally about 15 Mb in size, though the files can be bigger or smaller depending on the actual amplitudes recorded which affect the compression.
Changing the retained data on the SD card
Click the settings icon and navigate to "Settings/Advanced/Waveform Data Retention/Waveform Files"
Change the "Retained for **x** days" to the number of days you wish for data to be retained for. The maximum is around 330days.

Note: Be careful when configuring this parameter! You risk filling up the disk space. If this happens, you will no longer be able to access your Raspberry Shake. It is estimated that the OS and software consume 3 Gb of disk space.

Please refer to https://manual.raspberryshake.org/traces.html#waveforms-on-disk for data retaining information.

2. Installing the necessary services on the on the Shake

Access the raspberry pi over the network via SSH using remote terminal.

Log into the raspberry pi using default Username and Password:

Username: myshake
Password: shakeme

Please refer to https://manual.raspberryshake.org/hacked.html#hacked for secure purpose.

Update program lists on the raspberry pi using the following command:

```
$ sudo apt-get update
```

Install xRDP, which is used for remote accessing the Shake, using the following command:

```
$ sudo apt-get install xrdp
```

When prompted if you want to continue press "**y**" and enter

Install gpsd (GPS daemon), gpsd-clients, python-gps, and pps-tools; a suite of tools which will enable the RPi to read the satellite data coming from the GPS receiver

```
$ sudo apt-get install gpsd gpsd-clients python-gps pps-tools
```

Install sg3-utils which will enable the use of a 3G dongle to send data in remote locations.

```
$ sudo apt-get install sg3-utils
```

3. Installing the GPS for setting the time

The GPS time will not synchronise with the NTP time if the date/time on the Shake is considerably different from that being received by the GPS. It is therefore necessary to crudely set the time before starting this procedure. To manually set the time by type the following command using your date/time:

```
$ sudo date --set "15 Mar 2019 16:00:00"
```

Run the following command to see all currently connected USB devices; the USB device should be listed. If it is there then the GPS has been recognized as a serial device.

```
$ sudo lsusb
```

Example output:

```
Bus 002 Device 002: ID 8087:0024 Prolific Technology, Inc. PL2303 Serial Port
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

The TTL to USB converter that GPS had been attached may come up as shown above.

View the system log to see how the GPS device has been listed:

```
$ dmesg | grep tty
```

This will search the ***dmesg*** output for any mention of ***tty***. This is used as it is the precursory information to either USB or ACM (see below) and will therefore narrow down the output to the information you require.

Near the bottom of the system log there will be lines confirming that the USB GPS device has been listed. The device should be listed as one of the following: ***ttyUSB0*** or ***ttyACM0***. Note which of these the device is listed as for later.

At this step, the necessary services needed for the raspberry shake to use the GPS timing has been installed.

Next, install *gpsd* (GPS daemon), *gpsd-clients, python-gps,* and *pps-tools;* a suite of tools which will enable the RPi to read the satellite data coming from the GPS receiver.

```
$ sudo apt-get install gpsd gpsd-clients python-gps pps-tools
```

Start **gpsd**; use ACM or USB depending on how the device was listed in the system log.

```
$ sudo gpsd /dev/ttyUSB0
```

Input your device parameters into the **gpsd** configuration file by entering:

```
$ sudo nano /etc/default/gpsd
```

In the file there will be the following titles in CAPITALS, the information after the titles needs adding to the file. In the 'DEVICES' section ACM can be changed to USB if this does not work, however ACM should work if this is how the device was listed in the system log (*dmesg*):

```
START_DAEMON="true"
USBAUTO="true"
DEVICES="/dev/ttyUSB0 /dev/pps0"
GPSD_OPTIONS="-n -G"
GPSD_SOCKET="/var/run/gpsd.sock"
```

Check that the GPS has signal before proceeding. A GPS signal lock will be indicated by the flashing light on the blue LED on GPS:

```
Flash every second = No lock
Flash every 1 seconds = GPS lock
```

With your RPi and GPS in a position where it can view the sky, check the data feed. **–s** tells the cgps to only show processed. Press '**q**' to exit the 'cgps' screen

```
$ cgps -s
```

The ***ntpd*** (Network Time Protocol daemon) should already be installed on the Shake. You can view NTP information by using the command:

```
$ ntpq -p
```

Edit the ***ntp.conf*** file using the following command to access the file.

```
$ sudo nano /etc/ntp.conf
```

nano refers to the text editing program installed on the raspberry pi and ***/etc*** is the directory to the file. The file need to modified as below.

```
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help
#
# Allow large time difference time changes
tinker panic 0
#
driftfile /var/lib/ntp/ntp.drift
#
# Enable this if you want statistics to be logged.
#statsdir /var/log/ntpstats/
#statistics loopstats peerstats clockstats
#filegen loopstats file loopstats type day enable
#filegen peerstats file peerstats type day enable
#filegen clockstats file clockstats type day enable
#
# Enable remote servers if the PI has access to the Internet
server 0.debian.pool.ntp.org iburst
server 1.debian.pool.ntp.org iburst
server 2.debian.pool.ntp.org iburst
#
# GPS Serial data reference
server 127.127.28.0 minpoll 4 maxpoll 4 noselect
fudge 127.127.28.0 time1 0.500 refid GPS
#
# GPS PPS reference
server 127.127.28.1 minpoll 4 maxpoll 4 prefer
fudge 127.127.28.1 refid PPS
#
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
#
# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1
restrict ::1
#
# Depending on your network you may need to change the IP address below
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap nopeer
restrict 192.168.0.0 mask 255.255.255.0 nomodify notrap nopeer
# End
```

The configuration above will use both the GPS receiver and 3 NTP servers over the Internet. If the raspberry pi is not connected to the Internet you should comment out these lines. The system need to reboot or use the following command to restart NTP:

```
$ /etc/init.d/ntp restart
```

Or

```
$ sudo service ntpd restart
```

After restarting NTP you can use the following command to test your configuration: **ntpq -p** The output should look something like this:

```
$ ntpq -p
```

Example output:

```
remote   refid    st   t    when     poll    reach     delay     offset     jitter
==============================================================================
*SHM(0)  .GPS.    0    1    2        16      377       0.000     -511.76    18.917
 SHM(1)  .PPS.    0    1    -        16       0        0.000      0.000      0.000
```

If the time has not synced then type the following commands to restart gpsd and ntpd:

```
$ sudo killall gpsd
$ sudo gpsd /dev/ttyUSB0
$ sudo service ntpd restart
```

If necessary check the raw data streams from the GPS unit using the following commands to Check processed GPS data:

```
$ gpsmon /dev/ttyUSB0
```

Press ***q*** and then ***enter*** to quit the *gpsmon* screen back to the command line

4. Setting up the Pulse Per Second (PPS) for better time accuracy

Edit the **ntp.conf** file: Use the following command to access the file.

```
$ sudo nano /etc/ntp.conf
```

Add the following lines after the # GPS_time lines that you added in the previous section:

This will enable the NTP clock to receive data from the PPS (Pulse Per Second) stream you are about to setup. The PPS allows the Shake to keep the timing of the clock's seconds more accurate but does not supply an actual time itself. ***22.0*** will use ***/dev/pps0*** to get PPS timestamps and correct the kernel clock.

The ***flag 4*** fudge factor used below records a timestamp once for each second allowing the construction of Allen deviation plots

```
# PPS_time - kernel-mode PPS ref-clock for the precise seconds
server 127.127.22.0 minpoll 4
fudge 127.127.22.0 refid PPS stratum 0 flag3 1 flag4 1
```

Make an ntp working directory

```
$ mkdir ntp
```

Enter that directory (*cd* = change directory)

```
$ cd ntp
```

Install the following to prevent a file not found error later on in the process

```
$ sudo apt-get install libcap-dev
```

Install the following

```
$ sudo apt-get install libssl-dev
```

Download the following NTP (network time protocol) tar file:

```
$ wget http://archive.ntp.org/ntp4/ntp-4.2/ntp-4.2.8p11.tar.gz
```

Unpack tar file

```
$ tar xvfz ntp-4.2.8p11.tar.gz
```

Enter the directory that was just created:

```
$ cd ntp-4.2.8p11
```

Once in the directory run '*configure*' which will find the configuration script and build the '*makefile*'.

```
$./configure --enable-linuxcaps
```

Build the new NTP software from the 'makefile'

```
$ make
```

Once you have a new set of NTP binaries, you need to copy them to the correct directory and then restart NTP. And make Copy:

```
$ sudo cp /usr/local/bin/ntp* /usr/bin/ && sudo cp
/usr/local/sbin/ntp* /usr/sbin/
```

Restart:

```
$ sudo service ntpd restart
```

Move into the "/usr/local/sbin" and check the new files have been copied

```
$ cd /usr/local/sbin
```

Check listings:

```
$ ls- s
```

Check the listings, files labelled ntp* should have the date that you built the new ntp binaries.

Check for version and basic function

```
$ ntpq -crv -pn
```

Edit the boot configuration file:

```
$ sudo nano /boot/config.txt
```

Add the following on a new line:

```
dtoverlay=pps-gpio,gpiopin=21
```

Save and close (***Ctrl-x, y, Enter***)

Edit the modules file

```
$ sudo nano /etc/modules
```

Add the following on a new line

```
$ pps-gpio
```

Save and close (***Ctrl-x, y, Enter***) and reboot the raspberry pi.

```
$ sudo reboot
```

To check that the module is loaded, you can use the `lsmod` command, for example:

```
$ lsmod | grep pps
```

The output should be similar to:

```
pps_gpio 2529 1
pps_core 7943 2 pps_gpio
```

You should now be able to run the ppstest command and see that pps data is coming through to the Shake. ***You may need to wait upwards of 30 mins if you don't have good GPS signal.***

```
$ sudo ppstest /dev/pps0
```

With an example output of:

```
trying PPS source "/dev/pps0"
found PPS source "/dev/pps0"
ok, found 1 source(s), now start fetching data...
source 0 - assert 1351501153.999956346, sequence: 47481 - clear
0.000000000, sequence: 0source 0 - assert 1351501154.999954601, sequence: 47482
- clear 0.000000000, sequence: 0
```

press ***Ctrl-C*** to exit

Check the status of the pps input source in the kernel by using the following command:

```
$ dmesg | grep pps
```

Example output:

```
[ 0.000000] Linux version 3.2.27-pps-g965b922-dirty (root@bt) (gcc version
4.6.2 (Ubuntu/Linaro 4.6.2-14ubuntu2~ppa1) ) #1 PREEMPT Sat Sep 22 16:30:50 EDT
2012
[ 1.866364] usb usb1: Manufacturer: Linux 3.2.27-pps-g965b922-dirty dwc_otg_hcd
[ 12.797224] pps_core: LinuxPPS API ver. 1 registered
[ 12.803850] pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo
Giometti <giometti@linux.it>
[ 12.824858] pps pps0: new PPS source pps-gpio.-1
[ 12.832182] pps pps0: Registered IRQ 194 as PPS source
[ 133.043038] pps_ldisc: PPS line discipline registered
```

Check that the GPS and PPS source are now being used to keep time on the Shake

```
$ ntpq -p
```

Example output:

```
Remote   refid    st  t   when    poll   reach   delay    offset    jitter
==============================================================================
*SHM(0) .GPS.     3   l   16      16     377     0.000    -10.318   26.352
```

```
oPPS(0) .PPS.     0    1     14        16       377        0.000        0.113      0.009
```

The * symbol next to SHM indicates that the time is being set from the GPS on the shared memory driver (SHM). The o symbol next to PPS indicates that there is a good PPS signal that is being used to keep the seconds of the SHM synchronized.

Additional useful tool:

Install the GPIO Zero tool to view the GPIO pin layout on screen in the command window:

```
$ sudo apt-get install python3-gpiozero
```

Use the ***`pinout`*** command to view the pin layout on the RPi.

5. Real Time Clock (RTC) Installation

The Raspberry shake has no onboard clock that is requires the internet connection to update the internal clock via an NTP - Network Time Protocol – server. The installed GPS will not provide the timing to the system if the system time is much different from the GPS time. Thus, the external hardware clock is needed to provide the timing for the system to run on boots. This add-on boards plug on the GPIO pins and contains a clock chip and a battery that tell the Raspberry Pi what time it is.

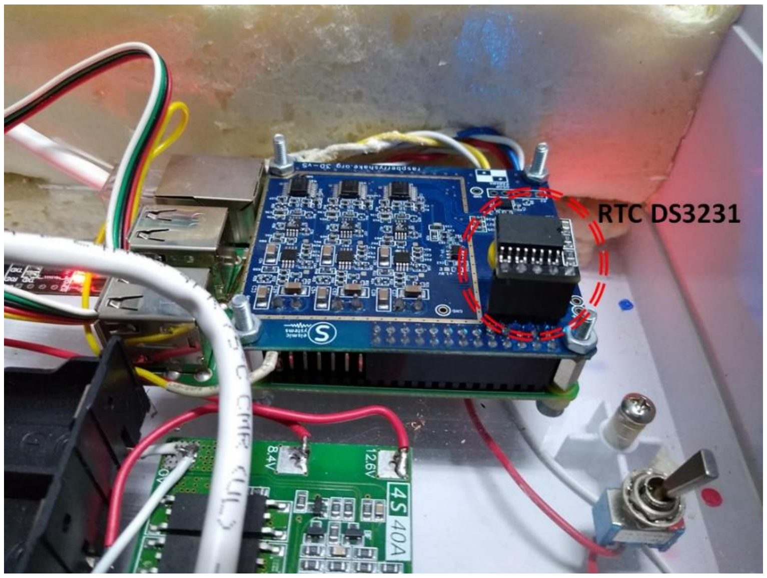


Figure A2-1 installation of the RTC

Step 1 : Install the RTC module

The RTC board uses what is known as the I2C (pronounced 'eye squared see') protocol to communicate between itself and the Raspberry Pi. The I2C allows the Raspberry Pi to communicate with a lot of different devices at the same time that is also connected to the same I2C pins on the Raspberry Pi. Five pins were ironed to the shake 3D hat, which is pins 1 , 3 , 5 , 7 ,and 9.

Step 2 : Setting up I2C by Install i2c-tools using:

```
sudo apt-get install i2c-tools
```

Step 3: Once the i2c-tools have been installed, check that the RTC Pi has been detected using:

```
sudo i2cdetect -y 0  # (if using v1 Raspberry Pi or)
```

```
sudo i2cdetect -y 1  # (if using v2 Raspberry Pi)
```

Then the RTC Pi should appear on channel 68. If the RTC Pi does not appear to check that the RTC battery is installed correctly and is fully charged.

Step 4 : Edit config.txt to add the following: dtoverlay=i2c-rtc,ds1307

```
sudo nano /boot/config.txt
```

At the end of the file add

```
dtoverlay=i2c-rtc,ds1307
```

Step 5 : Add the module to /etc/modules:

```
sudo nano /etc/modules
```

Add at the end of the file

```
rtc-ds1307
```

Save your changes

Step 6 : Next edit edit /lib/udev/hwclock-set

```
sudo nano /lib/udev/hwclock-set
```

Comment out the following lines with #

```
#if [ -e /run/systemd/system ] ; then
#exit 0
#fi
```

Reboot the Raspberry Pi.

```
sudo reboot
```

If this is the first time you have run the RTC Pi it will display a date of January 1st 2000.

Step 7 : If the Raspberry Pi is connected to the internet the correct date and time should be set automatically otherwise you can set the current date and time using:

```
sudo date -s "2 APR 2019 18:00:00"
```

You can check the current Linux date with the command (“**date”**). To save the date onto the RTC Pi use the following command:

```
sudo hwclock -w
```

Verify the date has been saved onto the RTC Pi with:

```
sudo hwclock -r
```

If everything worked correctly the RTC Pi should be initialized on boot and the current date and time will be loaded into Linux.

Step 8: If for some reason the date is not loaded from the RTC Pi on boot, you can get around this problem by creating a script which runs when the Raspberry Pi boots.

Create a script called hwclock in the /home/pi folder.

```
sudo nano /home/pi/hwclock
```

Add the lines below:

```
#!/bin/sh
/sbin/hwclock –hctosys
```

Save the file and set the permissions using chmod to make the script executable.

```
sudo chmod 755 /home/pi/hwclock
```

Next we will use crontab to run the hwclock script on boot.

```
sudo crontab -e
```

If this is the first time you have used crontab it may display the following message asking you to select an editor.

```
Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        <---- easiest
  3. /usr/bin/vim.tiny
```

Choose 2 to open the file in nano.

At the end of the file add the following line.

```
@reboot /home/pi/hwclock &
```

Save and reboot your Raspberry Pi.

```
sudo reboot
```

The hwclock script should run when your Raspberry Pi boots loading the date from the RTC into Linux. You can check that the date is correct using the date command.

```
date
```

6. Setting up a Static IP address for the Shake

Once setup the static IP will not allow further services to be installed via an Ethernet connection to a router, so make sure all necessary services (xrdp, gpsd etc.) have been installed before this process is undertaken. The Static IP that will be created is specific to the laptop from which it is setup, therefore this will need to be considered when taking a laptop into the field to download data.

1) Open Windows Command Prompt line by searching 'command prompt' in the windows search option. Find the IPv4 IP address of the laptop's Ethernet adapter in Command Prompt:

```
$ ipconfig
```

2) Under the Ethernet adapter Ethernet configuration settings find the ***Autoconfiguration IPv4 Address***; it will look something like: 192.168.0.100
3) Power on the Shake with a screen and keyboard attached. Find the default gateway IP of the Shake using the following command:

```
$ route -ne
```

Write down the number from the **Gateway** column on the line which has **eth0** in the **Iface** column. It may look something like 10.0.0.1 or 0.0.0.0

4) Configure the Static IP in the **dhcpcd.conf** file. Open the file using the command:

```
$ sudo nano /etc/dhcpcd.conf
```

5) Scroll to the bottom of the file and add the following lines of code to the bottom of the file.

```
interface eth0
static ip_address=12.168.0.100
static routers=0.0.0.0
```

For the **static ip_address** use the first three numbers from the laptop's IPv4 address and select a number between 0-255 for the final number.
For the **static routers** use the **Gateway** number noted down earlier. Exit and save the file using **ctrl + X** followed by **ctrl + Y**

6) Reboot the raspberry pi

```
$ sudo reboot
```

After the reboot, connect your laptop to the raspberry pi using an Ethernet cable. Open Remote Desktop Connection on the laptop and type in the static IP address that was set. Remote desktop should then prompt you for the Username and Password for the raspberry pi and allow a user to login to the Shake command line.

7. Downloading data from shake offline via Filezilla

1) Ensure Filezilla is installed on the laptop you want to use to download the data. This will be the laptop which the Static IP has been setup on. Filezilla can down load at https://filezilla-project.org/
2) Open Filezilla and enter the following information into the toolbar at the top of the screen

   Host: 12.168.0.100 (this is your chosen static IP)
   Username: myshake
   Password: shakeme
   Port: 22 (may not be required in order to login)

3) Once remote access is acquired through FileZilla you will need to navigate to the appropriate files on the Shake using the pane on the right hand side of the screen. The file directory for the waveform data is as follows:

```
/opt/data/archive/YEAR/NETWORK/STATION/CHANNEL/<DAILY MINISEED
FILES>
```

In reality this may look something like:

```
/opt/data/archive/2019/AM/RE**D/SHZ.D/AM.RE**D.00.SHZ.D.2019.095
```

4) The section reading ***AM.RE**D.00.SHZ.D.2019.095*** is the daily miniseed file. The number at the end (095 in this case) refers to the sequential Day of the Year (DOY) e.g. 095 would be April 5th.

   FileZilla defaults to accessing ***/home/myshake*** folder on the Shake, you will need to go up two directories (the directory above home) in order to find the ***/opt*** directory and locate the data using the directory structure above.

5) Select all of the miniseed files in the folder and drag them across to the desired folder on the laptop. The directory structure for the laptop is displayed on the left hand side of the screen.

6) Once the transfer is complete you can close FileZilla and stop the remote connection.

8. Remote login to the Shake

Note: requires a laptop and Ethernet connection to the laptop

Use this function if you want to remotely login to the raspberry pi using a laptop

1) Connect an Ethernet cable between the laptop and rasp berry pi. Open the Windows Remote Access software on the laptop and enter the Static IP address which has been set up.

2) This should open up the login screen for the raspberry pi, type in the Username and Password

3) The raspberry pi will login and you will have the familiar command line view to enter commands.

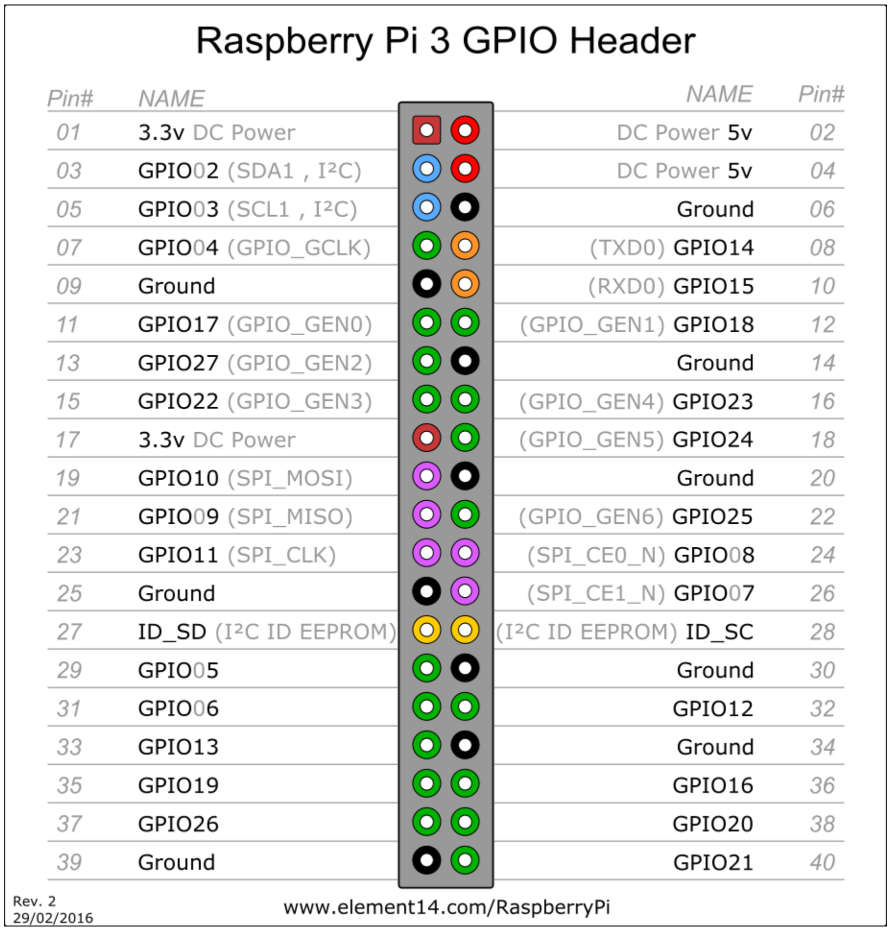
9. Raspberry Shake GPIO pins

The Raspberry pi 3 model B+ has 40pins GPIO header. The raspberry shake 3D board ("hat") sits on **pins 01 through 26**. So GPIOs and grounds on pins 27 to 40 are easily available to end users for other applications, as shown in figure below.

All of the pins from 01 to 26, the following pins are used as described in table A-1. All other pins are available for use. Some have specific functions such as pins 27 and 28, but the others can all be used for general purpose or their specific functions.

Table A2-1 The raspberry shake 3D board pins

GPIO PIN	Function
02, 04	5v
06, 09, 14, 20, 25	Ground
08	Tx
10	Rx
15	RESET
17	3.3v



Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

Figure A2-1 Raspberry pi 3 model B+ has 40pins GPIO header

10. Preparation of the Adafruit Ultimate GPS

1) Cut the row of pins down to a set of 9 and insert them into the 9 pin holes on the GPS board.
2) Use a soldering iron and solder to affix the pins to the board.
3) If wanted, solder the battery holder to the reverse of the GPS board.
4) Attach the colored USB-to-TTL cables and a female-to-female jumper cable, if some GPS module has no PPS pin, manual soldering was needed to wire out the cable to raspberry pi GPOI 21 in the following manner:

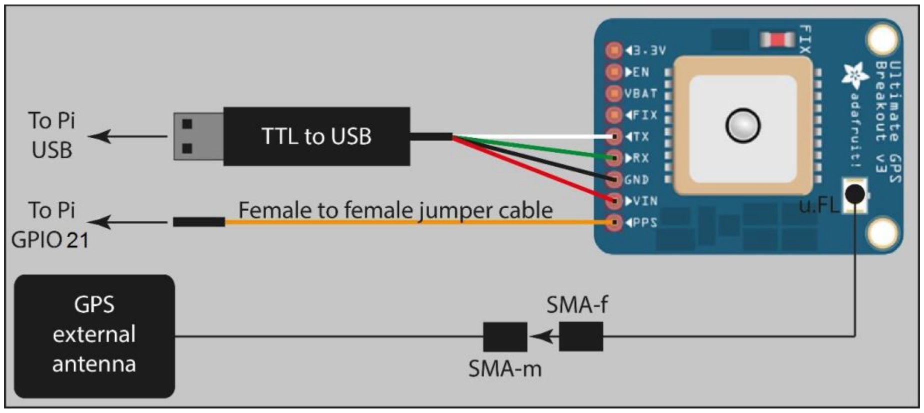


Figure A2-2 Adafruit GPS Module

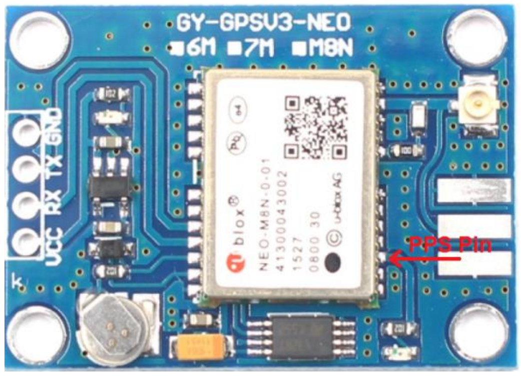


Figure A2-3 Ublox NEO-8M GPS Module

5) The USB cable should be put into one of the raspberry pi's USB slots, whilst the jumper cable should be attached to GPIO pin 21.
6) Connect the male-SMA connector of the external aerial to the female-SMA connector of the SMA-to-u.FL cable.
7) Connect the u.FL connector to the mount on the GPS board

11. Constructing Allen deviation plots for PPS

http://www.reeve.com/Documents/Articles%20Papers/Reeve_GpsNtp-Pi_Setup.pdf http://doc.ntp.org/4.2.4/monopt.html

When flag4 is set, the driver records a timestamp once for each second. The data are stored in clockstats file in directory /var/log/ntpstats/.

The clockstats and other statistics may be accessed using the cat command or WinSCP.

12. Real time data over UDP port

The real time data over data to one or more UDP ports located on any computer that the raspberry pi is able to communicate with. UDP functionality is not intended for production purposes but for hobbyist applications. This is because UDP by definition does not guarantee 100% receipt of data packets.

Because of the way the UDP protocol is designed to forward data packets to a specific IP address rather than other side computer requesting the packets from the Raspberry pi. This is different from most of the other protocols used in Seismology (SeedLink, etc.) which rely on handshakes and confirmations that data has been delivered. These protocols use the more standard internet protocol called TCP.

Since UDP is designed for real-time systems which prioritize data getting where it's going fast, and do not really care about "dropped packets" like TCP does, there are no requests or handshaking. There is only sending and receiving. The sending end is the Raspberry pi seismometer. The receiving end is the port on your desktop computer.

Since version 0.15, it is possible to configure destination machines and ports through the front-end configuration interface rs.local. Go to Settings::UPD Streams and define the combination of IP addresses and UDP ports you wish to send the data to. Sample programs are available for download, directly from the rs.local web interface, which can be run on the destination machine(s); these will provide a good tutorial and starting point for doing something interesting with the data.

Connecting to the UDP port to read the values can also be done using any programming language with the ability to make socket connections, basically all of them. Without getting into the details of socket programming, the choice of UDP was made to minimize the programming requirements to make a connection, get the data, and then do something interesting with it.

Below is a sample Python program that does only three things: **connects** to the default socket on the Shake Pi, **reads** the data off the socket, and **prints** it to the terminal.

```
import socket as s
with open('/opt/settings/sys/ip.txt', 'r') as file:
    host = file.read().strip()

port = 8888                                 # Port to bind to
sock = s.socket(s.AF_INET, s.SOCK_DGRAM | s.SO_REUSEADDR)
sock.bind((host, port))
print "Waiting for data on Port:", port
while 1:                                    # loop forever
    data, addr = sock.recvfrom(1024)        # wait to receive data
    print data
```

This sample program can be found in the directory **'/opt/settings/user'**, named **'shake-UDP-local.py'**. To execute from the command line (control-C to quit):

```
bash> python shake-UDP-local.py
```

In a real-world scenario, your program should separate out the individual fields of the data packet, compute any needed derivable values at program startup, place the data points into an array, and process the incoming data.

Configuration of Additional UDP Ports

If you would like to create additional data streams, either on the Shake Pi or a different computer altogether, a simple configuration file can be created to define these. Once the system has been updated to v0.8, a new directory will exist, named **'/opt/settings/user'**. Contained in this directory is a file named **'UDP-data-streams.conf.tpl'**, which serves as the template for the actual configuration file used by the data producer service. Editing this file will do nothing, as it is only a template. Read on for how to edit and save this as a working config file.

The format of this UDP configuration file is JSON and must conform by specifying the following entries:

```
{
    "UDP-destinations" : [
        { "dest" : "UDP-NAME-1"},
        { "dest" : "UDP-NAME-2"},
        { "dest" : "UDP-NAME-..."}
    ],

    "UDP-NAME-1" : {
        "Hostname" : "IP address or Fully-qualified DNS Name",
        "Port" :         "Port Number"
    },

    "UDP-NAME-2" : {
        "Hostname" : "IP address or Fully-qualified DNS Name",
        "Port" :         "Port Number"
    },

    "UDP-NAME-..." : {
        "Hostname" : "IP address or Fully-qualified DNS Name",
        "Port" :         "Port Number"
    }
}
```

The first section, **"UDP-destinations"**, defines the name of each UDP port to send data to. Subsequent sections must be named and represent each of the entries of the first section and define only two elements: the computer to send the data to, either by IP address or fully-qualified DNS name, and the port number. Create one or as many destination entries as you like.

An example configuration file sending data to two additional ports, one located on the local Shake Pi and another located on a different computer located on the local network might look like:

```
{
    "UDP-destinations" : [
        { "dest" : "UDP-SHAKE-PI"},
        { "dest" : "UDP-NETWORK-PI"}
    ],

    "UDP-SHAKE-PI" : {
        "Hostname" : "UDPIPFILE:/opt/settings/sys/ip.txt",
        "Port" :         "54321"
    },

    "UDP-NETWORK-PI" : {
        "Hostname" : "192.168.1.203",
        "Port" :         "11335"
    }
}
```

(Notice that for the local Shake Pi entry, the value "UDPIPFILE:/opt/settings/sys/ip.txt" is provided instead of an IP address or hostname. This is because for some Shake units their IP address is not fixed and can change when rebooted. This entry guarantees that the data producer will always output the data to the Shake Pi regardless if the IP address changes between reboots. If your Shake Pi has a fixed IP, the IP address can be provided directly if so desired.)

Once you have defined the computer and port combinations, and have created a configuration file, copy this to the file named **'UDP-data-streams.conf'**, located in the same directory where the template file itself is located: **/opt/settings/user**

Once the configuration file exists, the data producer service will automatically begin writing data to the configured hosts and ports, there is no need to restart the data producer service. As well, any time this file is modified, the new configuration will be taken on by the data producer service immediately, including if the file is removed, and will stop data transmission to all previous configurations.

Appendix 3 : User’s Manual

(To be continued)