Seismic Sensor From Hard Drive?

Paul Byrne 15th September 2021

Table of Contents

1 Background....3
2 Principles of Seismic Detection....4
3 Project Goal....4
4 Construction....4
5 Acquisition System....6
i Requirements....6
ii Design....6
Initial Problems....6
Acquisition System Design....7
Amplifier Board Circuit....9
Software....11
iii Testing and Calibration....12
Results....13
6 Conclusion....16
7 Appendix....17
i Software....17
ii Hardware....18
iii Applications Used....19
iv Github Folders....19

1 Background

Whilst dismantling some old IDE drives to scavenge any useful parts, motors etc. (*Figure 1*), it occurred to me that the actuator, which comprises of the voice coil and powerful magnet could be used to detect motion. This document describes the project to test this conjecture. All the work including circuit design, software, is in located at https://github.com/Starlight1856 .

Figure 1

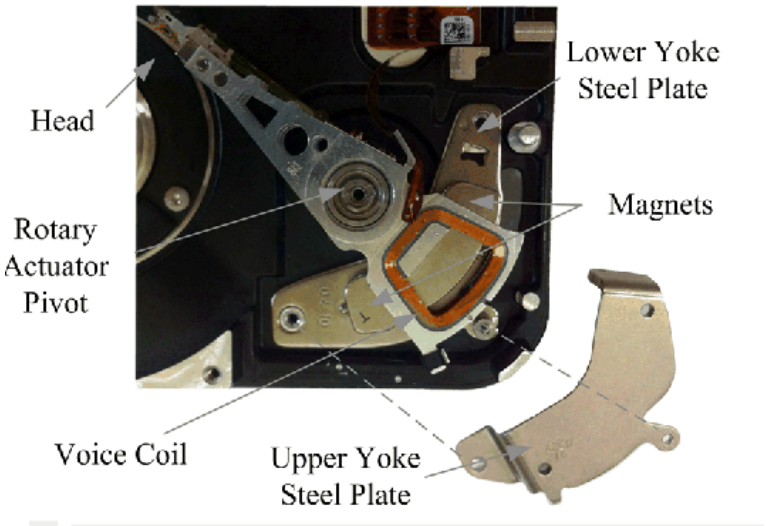


Figure 2

The motion would be detected by the movement of the voice coil across the magnet generating an electrical current which produces a voltage across the coil (*Figure 2*).

2 Principles of Seismic Detection

A basic seismometer for the up-down motion of the ground relies on a mass suspended on a frame mounted on the ground. When ground movement occurs, the frame moves along with the ground, the mass, however, because of its inertia does not move at the same time as the frame. This difference between frame and mass movement can be measured and is the principle behind seismic measurements.

3 Project Goal

The goal of this project is to determine in principle if the voice coil/actuator magnet combination from the hard drive could be used to detect any seismic events.

4 Construction

The mechanical design is based on a vertical motion seismometer. There are numerous examples; I have adapted a design based on mass suspended on a spring. The mass comprises of the actuator magnet on its mount, plus magnets for eddy current damping. The spring is a small toy ‘slinky’ type.

Figure 3

The voice coil has had to be mounted on a separate block to allow positioning against to actuator magnet. This would not be suitable for a permanent installation, but is adequate for the purpose of this experiment. The voice coil is positioned within 1mm of the magnet (*Figure 4*).

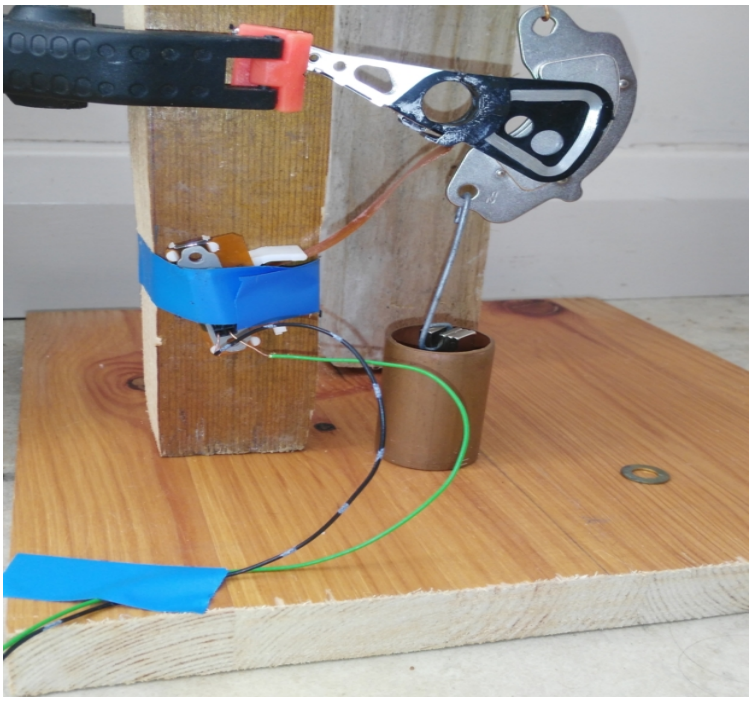


Figure 4

Attached to the bottom of the actuator magnet and within the copper tube are the magnets used for eddy current damping. The spring and magnet assembly have been tuned for 1Hz SHM. The complete unit is mounted on a solid floor, and whilst in use, screened.

5 Acquisition System

i Requirements

According to https://glossary.oilfield.slb.com/en/terms/s/seismic_wave the frequency range of seismic waves is approximately 1 to 100Hz. Whilst the full frequency range is beyond the capabilities of the design at present, the acquisition system shall be capable of this range for any future work. For a 100Hz frequency range a sample rate of 200Hz sampling frequency is required to meet the Nyquist rate for the maximum frequency in the range.

The acquisition hardware should operate as a remote autonomous system operating as a server storing at least a full days worth of data. This data can then be uploaded by a remote client via WiFi. For a 200Hz sampling rate and if stored as 32bit values (after any oversampling), the required memory would be 22Mb for a 30 hour period.

Data will be acquired at 16 bit accuracy, oversampling will also be used so accuracy will be above 16 bit.

Existing available hardware will be used a much as possible.

ii Design

Initial Problems

I had intended to use a 16 bit ADC with an SPI interface. However, as a result of the covid pandemic and associated chip shortages I have not been able to obtain any of the ADC's I had initially considered, certainly not at the prices I was prepared to spend. As a result I purchased an AD677 16 SAR ADC in a DIP package from Ebay, which was ex-development stock.

The AD677 is controlled by TTL level logic I/O rather than SPI.

Acquisition System Design

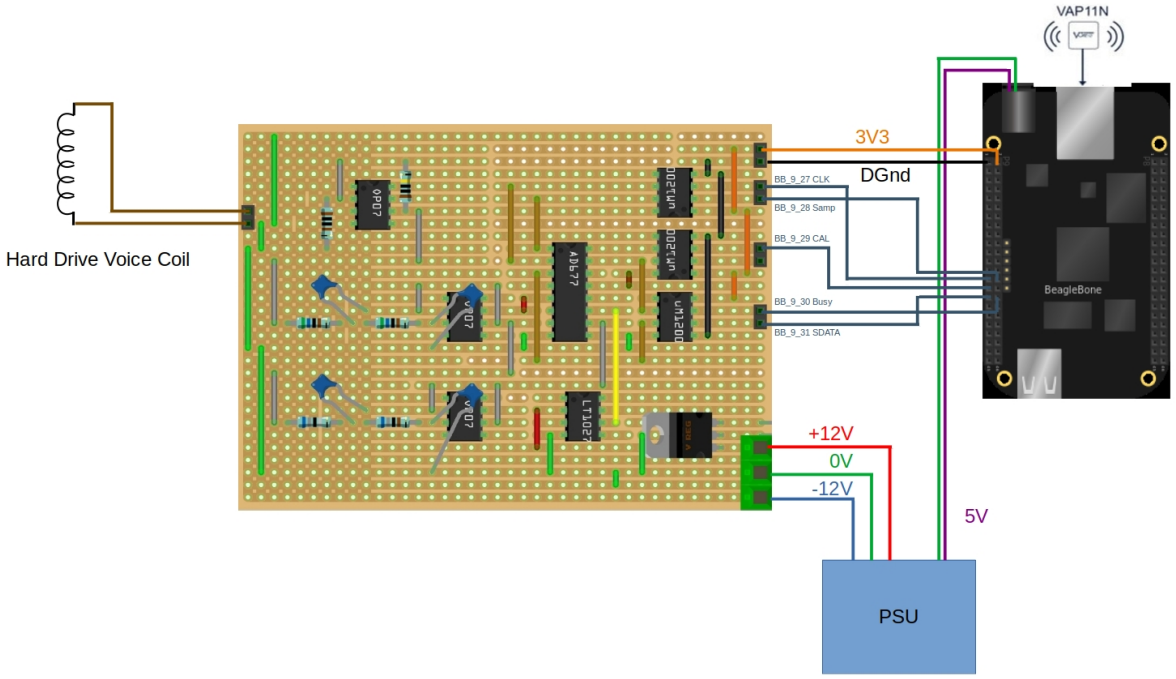


Figure 5

The acquisition system is shown in Figures 5 and 6. It comprises of a board containing the amplifier and ADC, and a BeagleBone Black which is used to drive the ADC, store the acquired data and when requested transmit via WiFi the accumulated results.

The BeagleBone Black has 512MB of memory capacity which is more than adequate to provide the 22MB required to store up to 30 Hours of data. The BeagleBone Black CPU is an ARM AM3358 Cortex-A8. This processor has two real-time processor units (PRU's) one of which is used to drive the AD667 ADC on the amplifier board using a bit-bang interface. The PRU drives the AD677 at a 10kHz sampling rate, then averages the result every 50 samples producing a final oversampled result with a rate of 200Hz. The results are stored in a shared memory area which is accessed by an application running in userspace that stores the data prior to transmission. This userspace application operates as a server awaiting requests from a remote client to transmit the data.

The WiFi interface is provided by a VAP11N bridge.

Figure 6

Amplifier Board Circuit

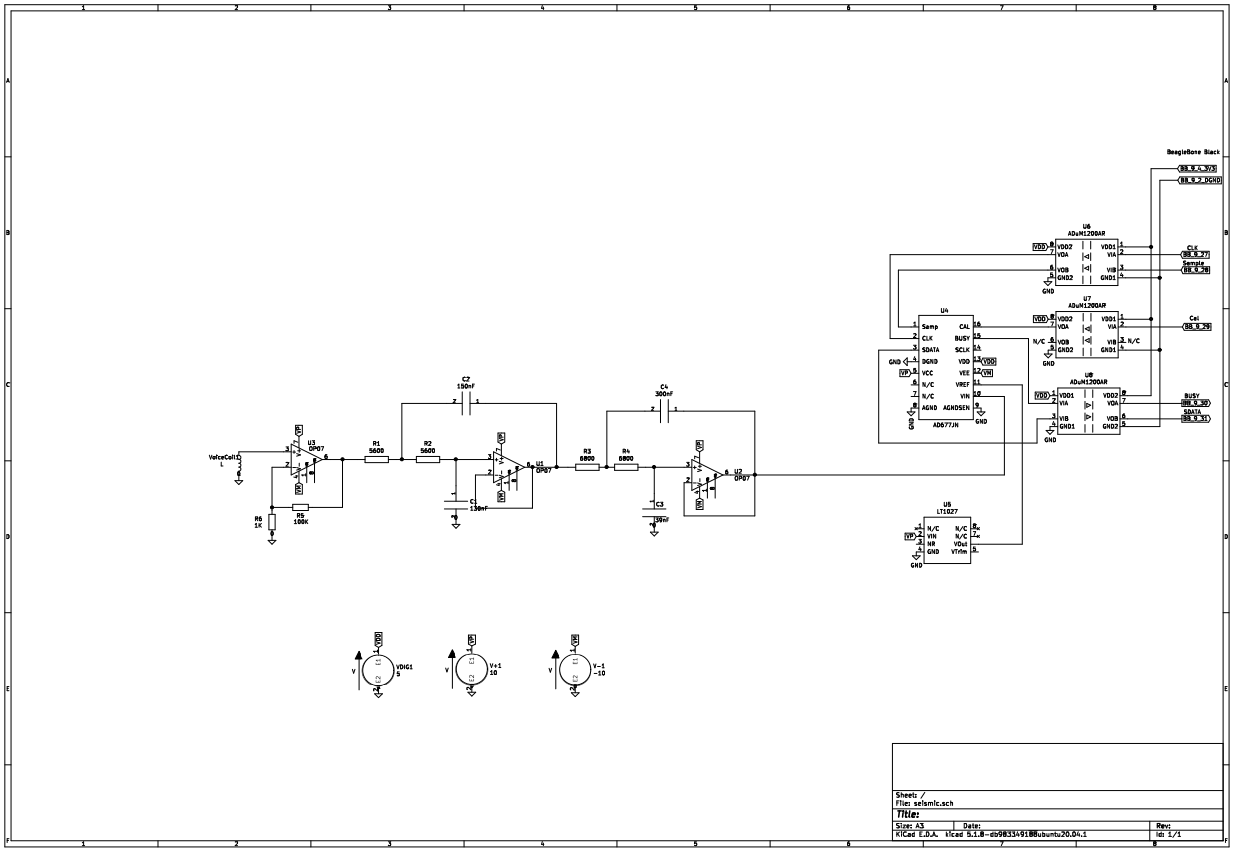


Figure 7

The amplifier circuit is described in Figure 7. The AD677 required bipolar supplies so the op-amps operate in bipolar mode. The first stage is a non-inverting op-amp operating with a gain of 1000. The output is fed into an anti-aliasing filter using a 4th order low-pass Butterworth filter comprising two, second order stages with a cut-off frequency of 200Hz. The original design was to achieve at least -40dBv at 1kHz which was the original oversampling rate. This design was kept for the 10kHz rate which now achieves -133dBv. The output from the filter stage if fed to the ADC input.

The AD677 also requires a reference voltage to operate; a LT1027 5.0V reference is used. The digital circuitry on the AD677 is TTL and requires a 5V supply, this is derived from an LM7805 regulator fed from the +ve component of the bipolar supplies.

The BeagleBone Black logic operates at 3.3V. To interface with the AD677 5V TTL control lines ADnM1200 digital isolators are used; this allows interfacing between the different logic levels and isolates the BeagleBone from the amplifier board. The BeagleBone side of the isolators are powered from the BeagleBones 3.3V supply.

Figures 8 and 9 below show the designed layout and actual implementation.

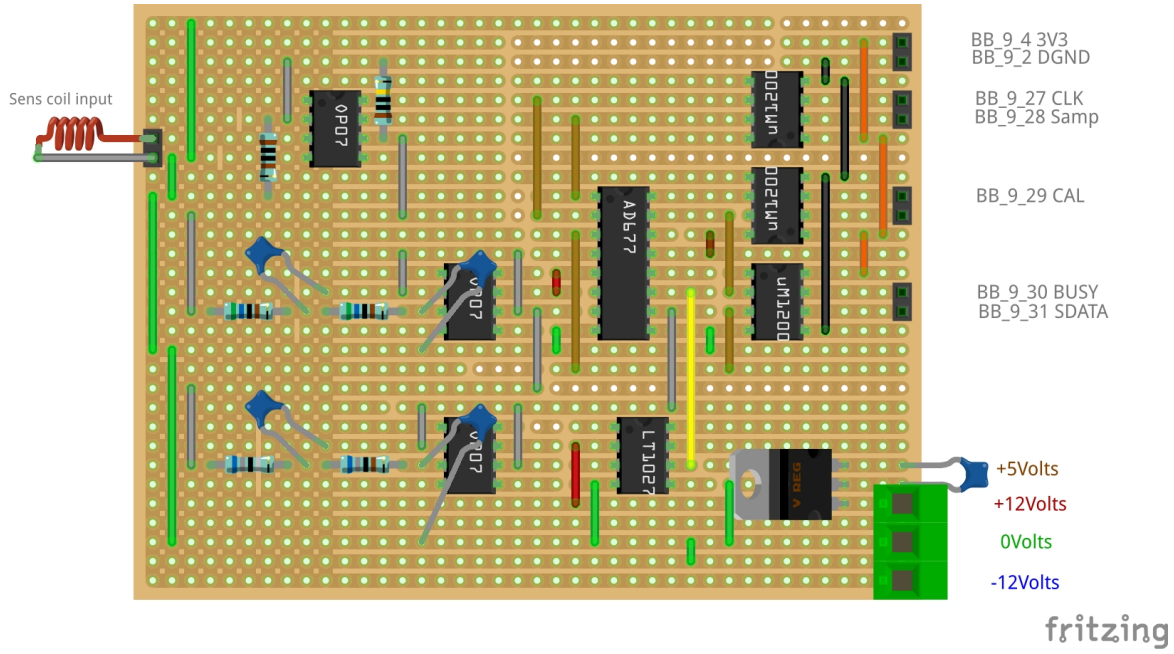


Figure 8 Board Layout

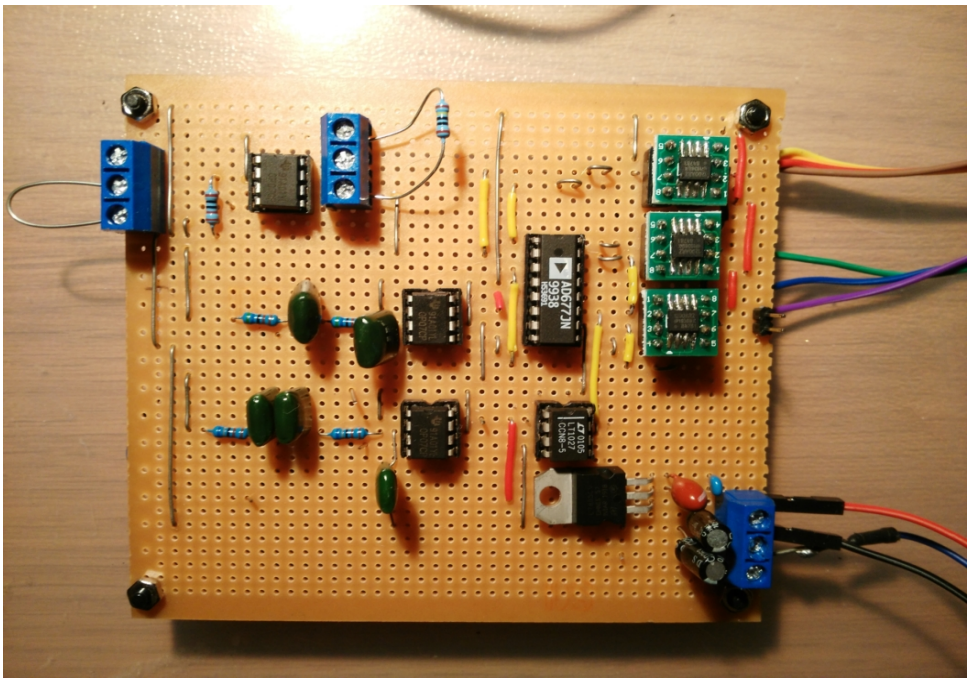


Figure 9 Board Implementation

Software

The system requires three software components. Two for the driving BeagleBone, and a remote client. While the eventual aim is to run the BeagleBone code automatically at bootup, at this stage all operations are carried out using PutTTY SSH Client, and PSFTP.

BeagleBone

The BeagleBone Blacks operating system is Beaglebone Linux 4.19.94-ti-r42. The I/O lines used for video have been disabled by editing the uEnv.txt file.

1. PRU Code. One of the Programmable Realtime Units is used to drive the AD677 ADC. The code is written in C using the Code Composer Studio 10.2.0 cross compiler. The BeagleBone I/O ports accessed by the PRU code are configured using a script with 'config-pin' commands rather than a device overlay tree which this operating system allows. Another script is used to load and run the PRU code. Interfacing with userspace code is via a shared memory area.
2. Server code. The BeagleBone operates as a server responding to commands from a client. The server code obtains the PRU acquired ADC data from a shared memory area configured as a buffer. As this PRU shared memory area is limited, the data is copied to a circular buffer in userspace in blocks of a seconds worth of data, plus a timestamp. The circular is buffer is configured to store up to 30 Hours of data. The code is written in C, mainly using **eclispe** with the arm-linux-gnueabifh-gcc cross compiler. The source code, with a build script was then transferred to the BeagleBone and any final editing was performed using VIM.

Remote Client

This is implemented as a console application with a keyboard interface, written in C using the GCC compiler. The client software sends commands to the server, including a request to transmit the buffer containing all acquired data. When this data is requested, it is stored in a binary file in blocks of a seconds worth of data with its associated timestamp. The client code also can convert on request this binary file to text files which are easier to use in applications such as python. A command is also available to shut down the server code on the BeagleBone.

Further information is contained in Appendix i.

iii Testing and Calibration

For testing and analysis python scripts are used using the *matplotlib* and *scipy* libraries.

An operational test was performed by removing the eddy current damping and allowing the spring to oscillate, which also included a slight pendulum motion. The results in *Figure 10* is as expected, producing both positive and negative pulses as the magnet moves across the axis, and to and fro with the pendulum motion. Note that the voice coil has had to be moved further away from the magnet to prevent saturation. A 1Hz low pass filter was applied to the data.

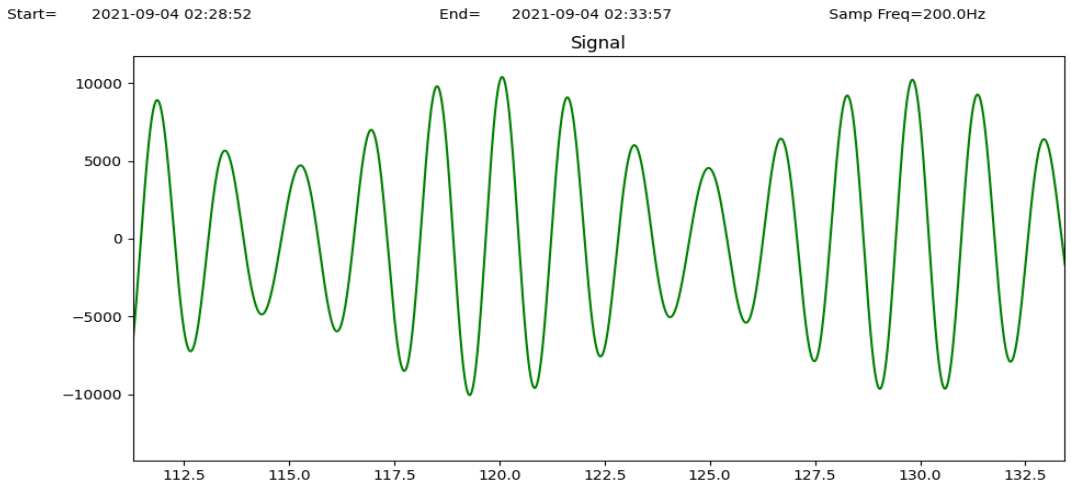


Figure 10

For calibration, the analysis software's low-pass filter cut-off frequency was set to 100Hz and the result analysed with an FFT. The voice coil was replaced with a large transformer with its core removed, so as to pick up a mains signal.

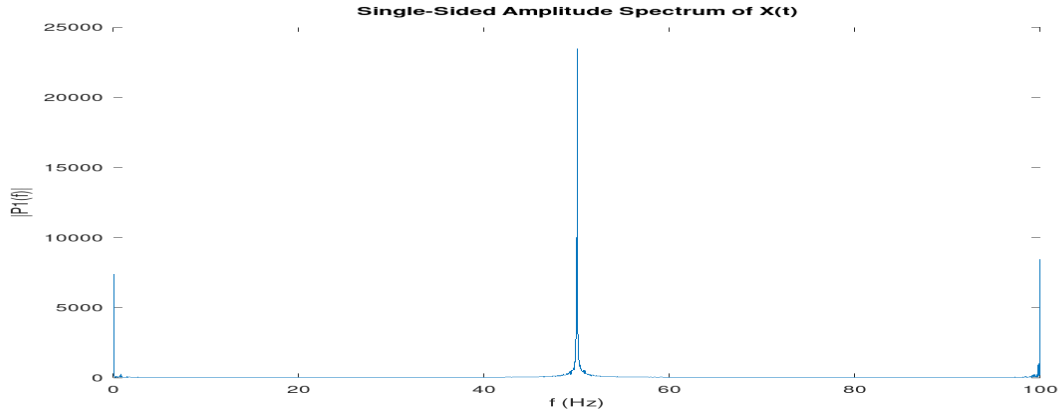


Figure 11

In Figure 11 above the frequency of the mains signal was 50.089Hz. Calling up an online national grid status site https://www.gridwatch.templar.co.uk , the frequency was 50.016Hz. Close enough!

Results

A number of runs were performed over a number of days. Shown below (*Figure 12*) is a typical run.

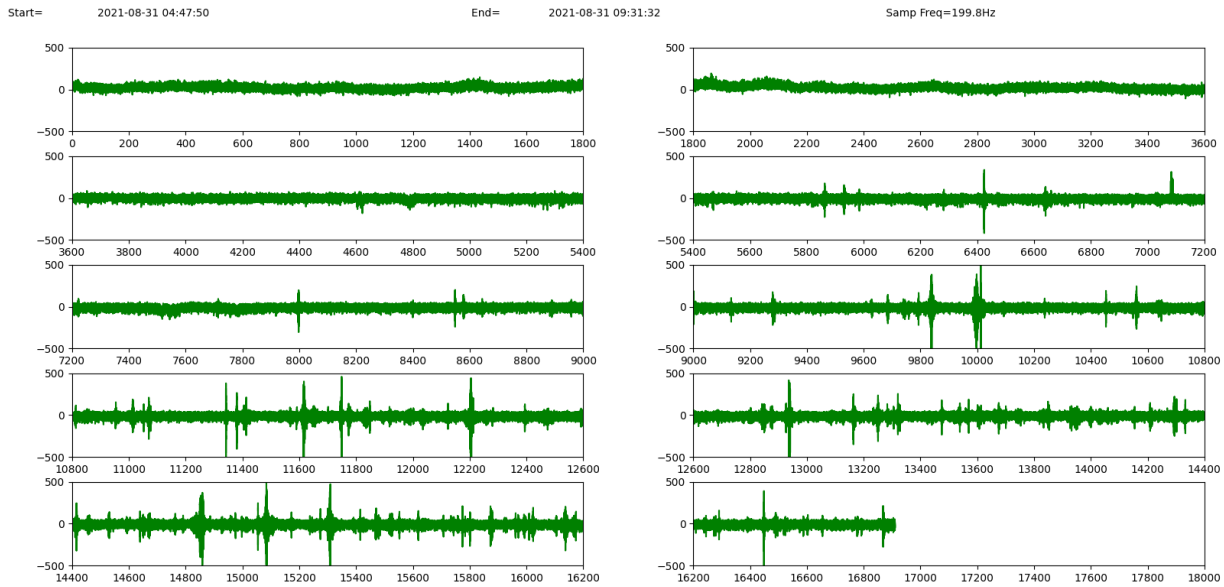


Figure 12: Results

Each graph is 1800 secs, 30 minutes long. It can be seen that from 04:47 to about 06:30 there is little activity, then a great deal of seismic activity begins. This seismic activity is however man-made. There is a road close by, and the sensor is located in rural East Anglia where mostly arable crops are grown and it's harvest time. From about 07:30 large signals start to appear; these are combine harvesters moving slowly along the road with their associated equipment. Also, tractors pulling trailers full of grain, straw bales and so on. The data displayed has had a low-pass filter applied with a 20Hz cut-off frequency. If the same data is processed this time with a 0.5Hz low-pass filter applied the results are shown in Figure 13.

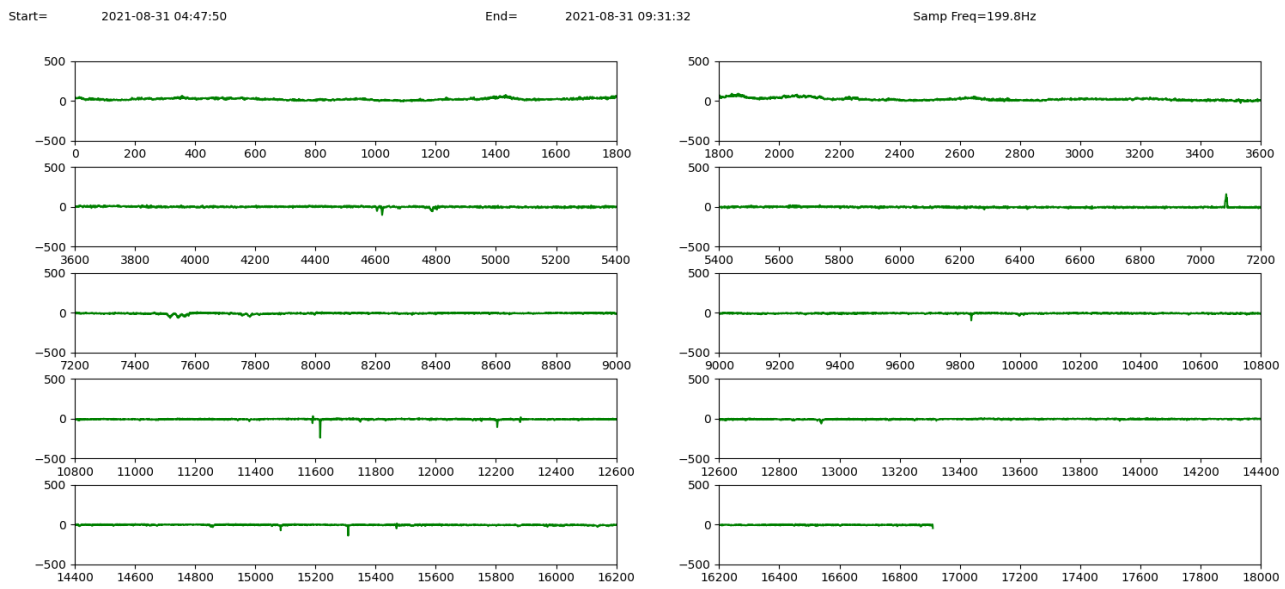


Figure 13: Results 0.5Hz Cut off

From the results when a 0.5Hz low-pass filter applied it can be seen that effects caused by traffic are still apparent, though the noise level is significantly reduced.

Comparison with Seismic Monitoring Stations

There is a British Geological Survey station Elmsett, 20 miles SSE of my location. These were the seismograms obtained during the same time.

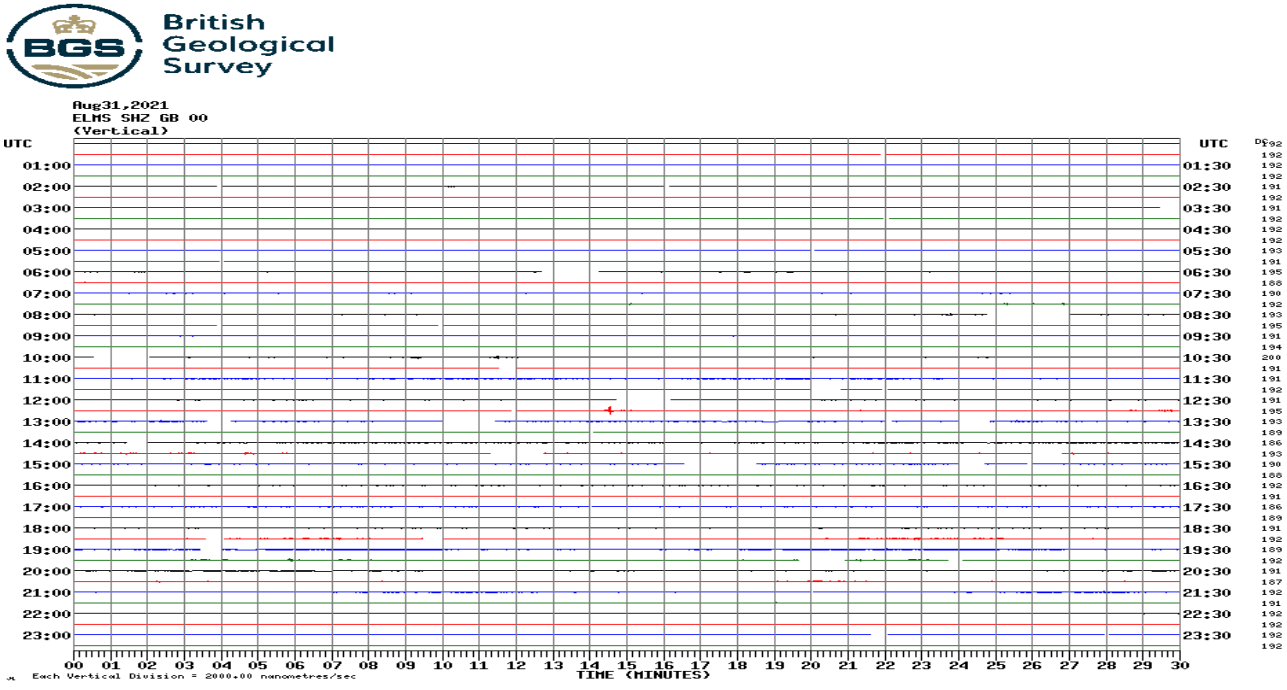


Figure 14: Short Period

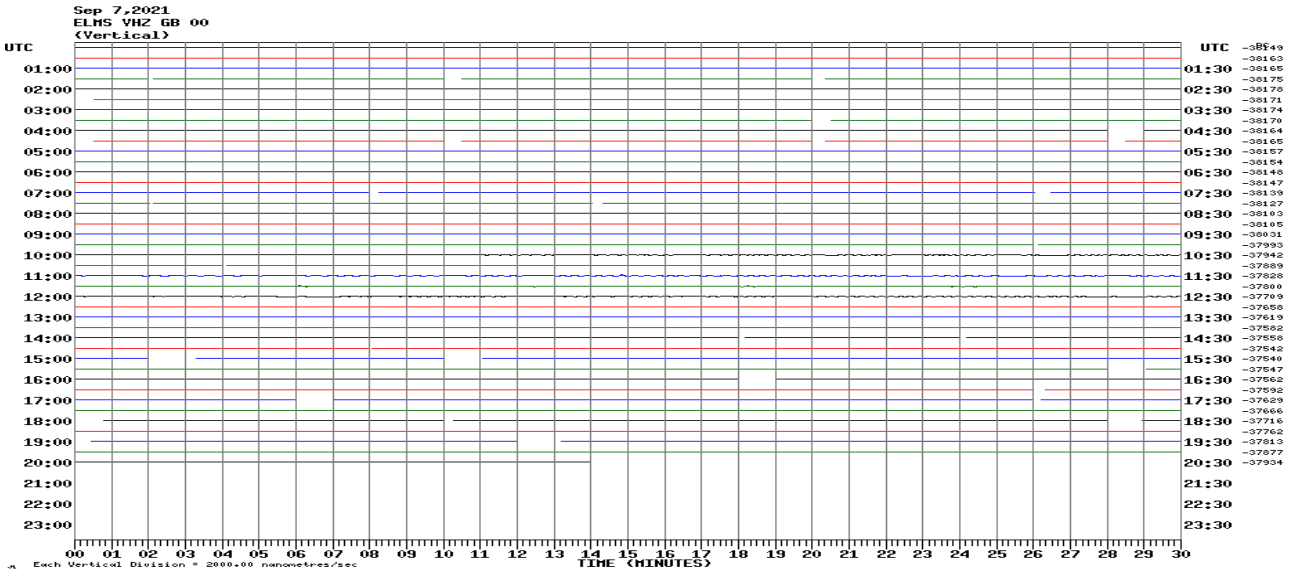


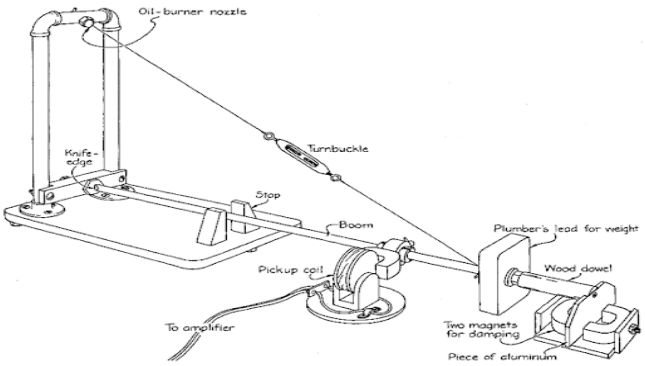
Figure 15: Very Long Period

From the above seismograms in Figures 14,15 it can be seen that no noticeable activity took place during the data acquisition.

6 Conclusion

The seismometer arrangement using the hard drive voice coil and actuator magnet can detect seismic events although those detected are all man-made. The location is not really suitable for a seismometer due to significant interference from heavy agricultural machinery on a nearby road. A natural seismic event could only really be detected during the ‘quite time’ of early morning, and this would have to be cross referenced with a seismic station.

A more suitable design using the voice coil/actuator magnet combination would be a horizontal mounted one such as below.



The main problem with the vertical arrangement was maintaining the 1mm separation between coil and magnet. The horizontal arrangement would allow these to be fixed. No conclusion can be reached as yet whether the actuator magnet/voice coil arrangement is sensitive enough to actually measure natural occurring seismic events. The work continues…….Paul Byrne 15th September 2021

7 Appendix

i Software

All the software produced plus hardware information is stored in https://github.com/Starlight1856 .

Beaglebone Black

1. **PRU Code**. This is in github folder */Beaglebone Code/PRU Code/pru_ad677d/* . This was built using the Code Composer Studio 10.2.0 Beaglebone Black PRU cross compiler. The compiled PRU code *ad677d.out* was copied to the Beaglebone via PSFTP to the */lib/firmware/* folder.
2. **Scripts**. Two scripts are required for running the PRU code and controlling the ADC I/O. These are *setup.sh* which configures the Beaglebones I/O ports, and *setup4.sh* which starts the PRU code. The scripts are run from the */home/debian/* folder on the Beaglebone. These scripts are in the */Beaglebone Code/scripts/* github folder.
3. **Server Code**. The code is in the github */Beaglebone Code/Server Code/* folder. The */server3/* folder contains the eclipse cross compiled project built using the arm-linux-gnueabifh-gcc cross compiler. The */Beaglebone Code/Server Code/Beablebone Build/* folder contains the same source files as */Server Code/* that have been copied to the Beaglebone and built using the native ARM gcc using the included *Makefile*. The source files are *netcom.c (main), netcom.h* and *server3.c* . The header file *netcom.h* defines the server/client messaging interface.

Remote Client

This code is in the */Client3/* folder. It is written in C and compiled using a native desktops x86 GCC cpmplier. It is a console based application with function selection by keyboard. Options available are;

i. ‘d’ Download data. Select to download the seismic accumulated seismic data from the remote server, and store as binary data file *data1.bin* in */tmp/* folder. It sends the download instruction to the server and waits for up to 5 seconds for a response to the message after which it will time out and return to the options menu, otherwise it will return to the menu after the last data block has been received.

ii. ‘p’ Ping remote server. Tests communications by sending a ‘ping’ message to remote server, server responds by sending ‘ping’ back to client. If no ping response is received it will time out after 5 seconds and return to the menu.

iii. ‘c’ Convert binary file. The binary file present in */tmp/data1.bin* is converted to a text format that is easier to use in python and octave. Two files are created *data1.txt* which contains the analogue data and *data2.txt* which contains absolute times of first and last data blocks plus acquisition rate and number of samples.

iv. ‘s’ Stop application. This halts the application running on the remote server.

v. ‘q’ Quit. Exits the client application.

ii Hardware

The selection of the hardware used has been constrained by availability during the covid pandemic. For example the original SPI 16bit ADC selected could not be obtained (not at prices offered) so an alternative was found. Another requirement was to use as many components which were at hand.

Analogue Board

- U1,U2,U3 0P07 Op-amps
- U4 AD677JN SAR ADC
- U5 LT1027 5V voltage reference
- U6, U7,U8 ADuM1200AR isolators 8SOIC soldered onto DIP carriers.
- C1 120nF n.p.f for 130nF
- C2 150nF
- C3 47nF n.p.f for 39nF
- C4 300nF 2x150nF in parallel
- R1, R2 5600
- R3, R4 6800
- LM7805 regulator 0.1uF ceramic, 2x100uF 30V Electrolytic

BeagleBone Black running Linux Beaglebone 4.19.94-ti-r42

VOTNETS VAP11N ethernet bridge.

Triple PSU +5V, ±12V

iii Applications Used

CodeBlocks : for client C code on desktop.

Code Composer Studio 10.2.0 : PRU code using the cross compiler.

eclipse : Primary development of Beaglebone server code using arm-linux-gnueabifh-gcc cross compiler on desktop.

Fritzing: Layout of circuit on veroboard.

KiCad : Circuit schematics and simulation of low pass filter design.

LibreOffice: Documentation.

PSFTP: Transfering files to and from Beaglebone.

PutTTY SSH Client: Opens remote console on BeagleBone.

VIM: For final editing of Beaglebone server code on the Beaglebone.

Visual Studio Code 1.51.1 : Writing Python scripts used for analysis of data.

Wireshark: Network comms diagnostics.

iv Github Folders

Beaglebone code: PRU code and server/acquisition code seismic3.

client3: Desktop remote client code.

datasheets: Datasheets used in project.

Kicad: Amplifier circuit schematics, and low pass filter simulation.

lpfilter: Low pass filter design and simulation results.

python: Python scripts used for analysis.

misc: Miscellaneous diagrams, pictures.

results: Contains data files produced from run shown in *Figure 12*.