LOW COST SEISMIC DATA ACQUISITION SYSTEM BASED ON OPEN SOURCE HARDWARE AND SOFTWARE TOOLS

Arvid Ramdeane[1*], Lloyd Lynch[2]

[1,2]Seismic Research Centre, The University of the West Indies, Trinidad
[1]Email: arvid.ramdeane@my.uwi.edu *
[2]Email: llynch@uwiseismic.com

Abstract*:* The University of the West Indies Seismic Research Centre, Trinidad and Tobago, operates a network of over 50 stations for earthquake and volcanic monitoring in the Eastern Caribbean islands. These stations form a seismic network consisting of various types of instrumentation, and communication systems. Over a period of 11 years, the Centre has embarked on an initiative of upgrading and expanding the current network with combinations of broadband and/or strong motion sensors, high dynamic range digitizers and networking equipment to link each station to centralized observatories via high speed digital data transmission medium. To realize such an upgrade and expansion, the Centre has developed a seismic data acquisition system prototype built using open-source hardware and software tools. The prototype is intended to be low-cost using off the shelf hardware components and open-source seismic related software handling data acquisition and data processing in two separate modules. The prototype uses a three-channel accelerometer sensor and can process data into standard MiniSEED format for easy data archiving and seismic data analysis. A global position module provides network time protocol time synchronization within 1 millisecond for accurate timestamping of data. Data can be stored locally on the prototype in twenty-minute data files or securely transferred to a central location via internet with the use of virtual private network capabilities. The prototype is modular in design allowing for components to be replaced easily and the system software can be updated remotely thus reducing maintenance cost.

Keywords: *MiniSEED, MEMS (Micro Electro Mechanical Systems), NTP (Network Time Protocol), Open-Source Hardware and Software Tools, Digitizer.*

https://doi.org/10.47412/VYCB8830

1. Introduction

The University of the West Indies (U.W.I.) Seismic Research Centre (SRC) headquartered in St. Augustine, Trinidad and Tobago is an organization responsible for the monitoring of earthquakes and volcanoes for the English-speaking Islands of the Eastern Caribbean (from Trinidad and Tobago in the South to St. Kitts and Nevis in the North). The SRC as of 2019 operates a network of fifty-six seismic stations which includes a combination of both analog and digital stations. Since 2008, the SRC has embarked on a mission to fully modernize the current network by furnishing the existing stations with combinations of broadband and/or strong motion sensors, high dynamic range digitizers, and networking equipment to link each station to a centralized observatory via high speed digital data transmission medium. After accomplishing half the task, the Centre has come to the realization that it has neither the fiscal nor the technical capability to complete the full upgrade and provide maintenance to a network equipped with commercial instrumentation in a

sustainable manner. Several inhibiting factors include insular geographical layout of the network, giving rise to several access and communications related challenges and high cost of procuring and maintaining commercial instrumentation.

The SRC has decided to undertake a project to design and develop a Low-Cost Seismic Data Acquisition System (SDAS) based on open-source hardware and software tools that will be used to complete the upgrade of their seismic network. This paper describes the design of the SDAS prototype which is based on an Arduino Mega Microcontroller that is programmed to capture data from a 24-bit analogue-digital-converter (ADC) interfaced with a micro electromechanical system (MEMS) based accelerometer sensor. The digitized data is then continuously fed to a Raspberry Pi (RPi) microcomputer for processing and a global positioning system (GPS) module interfaced with the RPi provides time synchronization via network time protocol (NTP) for data time stamping. Data is processed in standard MiniSEED (MSEED) seismological data format and inserted into a ring buffer where it can be streamed out to subscribers via the seedlink protocol and/or also be fed to a local Earthworm or SeisComp installation for manipulation and processing. The SDAS prototype is intended to be low-cost and produce data quality and timing comparable to other systems. To verify correct functionality of the system and that it meets all objectives, it will be tested side by side with other fully functioning systems where the data produced from both systems will be compared for accurate timing and quality of data.

The paper is formatted as follows. Section 2 provides a brief literature review into low-cost seismic instrumentation. Section 3 discusses the implementation and design of the SDAS. Section 4 provides a discussion on the overall SDAS design and section 5 concludes the paper with provisions for future work.

2. Literature Review

Low-cost seismic instrumentation and acquisition systems have been continuously researched and developed over the years. Dating back to 1988, the authors in [1] developed and installed a PC-based broadband digital seismograph network in Central California as part of an upgrade to the already permanent University of California seismographic stations. The design of the new stations was intended to be low-cost, with the use of personal computers for continuous recording of near and distant seismic events with the use of short period and broadband sensors. Modern day fully digital seismic stations can cost in excess of $100,000 USD [2, pp. 232] where the cost includes acquiring and installing the components and retrofitting the station site with proper power, communication, noise free, security, access, weather, topography, and geology considerations. But actual instrumentation costs are not cheap either. Some commercial sensors can cost upwards of $15,000 USD and multichannel digitizers can cost upwards of $6000 USD [3]. Thus, significant research has been conducted into low-cost seismic instrumentation incorporating the use of low-cost sensors and open source hardware and software tools. MEMS based accelerometers are one of the most widely used sensors in low cost-seismic instrumentation for measuring ground acceleration during seismic events. They are inexpensive, where substantial work with the use of these sensors in engineering seismology can be found in literature [4. pp. 118]. MEMS accelerometers are ideal for applications for dense seismic instrumentation use or microzonation where local site response of ground level during seismic events can be monitored. This type of application is evident in work done by [5] which assessed the performance of a low-cost MEMS based accelerograph for high seismicity monitoring in an urban area in Greece. In another study done by [6], a low-cost accelerograph was developed for monitoring the Alpine fault in New Zealand for anticipation of any large earthquake events along the fault. Current commercially available instruments were costly and incurred high maintenance costs, thus the low-cost accelerograph was realized by using easily available off the shelf components to keep development cost low.

In addition to sensors, other components comprise of seismic instrumentation, for example, the digitizer which is responsible for digitizing, handling and processing data from the sensors. The use of microcontrollers and/or microcomputers are becoming a popular choice in low-cost seismic instrumentation for handling such tasks. These components are inexpensive and easy to acquire with a wealth of software and hardware support available. The hardware itself is integrated with various communication protocols, general purpose input output (GPIO) pins and other features which allows for seamless integration with external hardware components required for seismic instrumentation such as sensors, ADC, GPS modules etc. The RPi microcomputer is one such component. It was used to build the Raspberry Shake 4D (RS-4D) discussed in [7] which is an all in one seismic data acquisition and processing system. The RS-4D captures and processes data from a vertical component 4.5 Hz geophone and 3-component MEMS accelerometer with 24-bit resolution sampling at 100 Hz where timestamping of data is provided using an NTP server. According to the authors, the overall cost of the RS-4D varies between a few hundred to $1000 USD which is significantly less than the cost of some commercial instruments. The authors concluded based on testing that the RS-4D can be used for building dense networks for monitoring local and regional seismic events. In another study done by [4], the primary objective was to develop a low-cost approach for designing a MEMS based accelerograph by using inexpensive components, such as a MEMS based accelerometer. The low-cost accelerograph, nicknamed the "seismobug" incorporated the use of Atmel ATmega328P microcontroller capturing data over the serial peripheral interface (SPI) from a digital 14-bit accelerometer sensor with variable data rates and user defined acceleration full scale ranges. The purpose of this study was to create a dense network of seismobug instruments for seismic monitoring in urban areas investigating local site response. Initial testing showed good correlation in terms of timing and data quality between the first seismobug prototype and a high-cost high accuracy strong motion instrument.

In terms of seismic related software, there are many libraries available for processing and viewing of seismic data. While some commercial seismic instrumentation have their own proprietary software, which can lead to high design and maintenance costs [6, pp. 45], the use of open source software can help reduce these costs allowing for flexibility in designing low-cost instrumentation in terms of communication and operation. The following software libraries developed by Chad Trabant of IRIS Data Management Centre are just some of the many libraries available for seismic related use. The "libmseed" library is used for processing data into standard MiniSEED (MSEED) format which is widely used in the seismological community (Refer to Section 3.4 for more detail on MSEED format). The MSEED data format can be converted to or from other data formats such as gcf (Guralp instruments data format) or seismic analysis code (sac) and is compatible with most MSEED readers. The "ringserver" library provides an internal ring buffer data structure for storing data packets in different formats for multiple channels allowing end users to access the data in the ring buffer through the seedlink and datalink protocol. The seedlink protocol allows for robust data communication between a client and server where data packets are typically 512-byte MSEED records and the datalink protocol allows for data packets mainly MSEED packets to be inserted into the ring buffer. The ring buffer can also be memory mapped to physical storage in situations where random access memory is low. Apart from predefined libraries used for designing the software aspects of low-cost instrumentation, there are many programs available for reading time series data, such as Swarm and Seisgram2k. These programs are compatible with many different data formats, including MSEED and allows for processing of seismic data, for example, event picking. The programs can also connect to a ringserver over the seedlink protocol allowing for data transfer from a seismic site to a central location where data can be archived and processed.

3. Implementation and Design

The architecture of the SDAS is shown in the block diagram in Fig. 1. The block diagram presents how the hardware components are integrated with each other and the flow of data from the sensors to storage. The subsequent sections discuss the hardware components in detail used in the block diagram of the SDAS.

3.1 Hardware Overview

3.1.1 Sensor

The sensor chosen for this project was the ADXL335 triple axis accelerometer. The ADXL335 is a MEMS based device with a +/- 3 g acceleration on each axis. The sensor is low power consuming 350 µA typical with a single supply range from 1.8 V to 3.6 V. We went with the Adafruit ADXL335 which came integrated on a breakout board providing a 3.3 V regulator for use with 5 V power supplies and integrated 0.1 µf capacitors for a low pass filter on each axis providing a 50 Hz signal bandwidth. The sensor was mounted in a small rectangular box attached to an aluminium plate which was bolted to the ground. The position of the sensor was placed such that the X axis points north, the Y axis points west, and the Z axis points upwards. The sensor is attached to the digitizer board on the SDAS through a RS-232 cable.

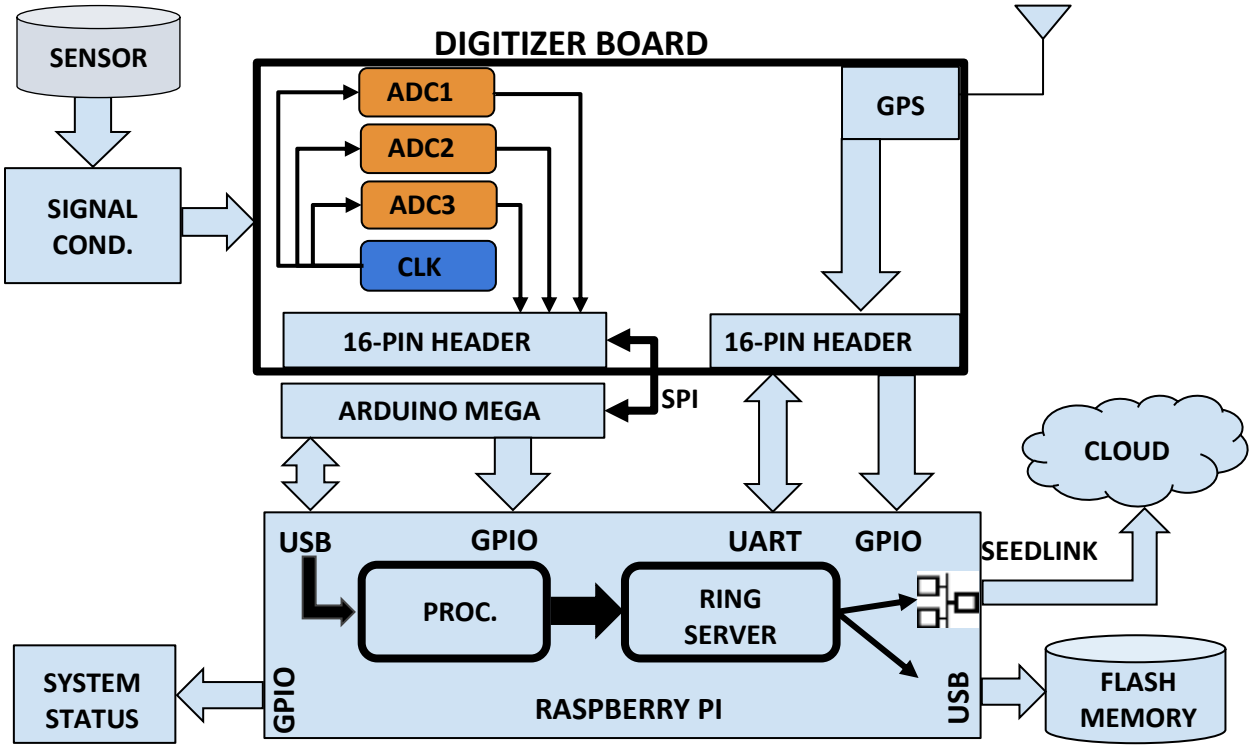

Figure 7: SDAS Block Diagram

3.1.2 Signal Conditioning

For signal conditioning, we used the integrated 50 Hz low pass filter on the breakout board of the ADXL335 accelerometer. The filter removes any high order frequency components that results in anti-aliasing during A-D conversion of the sensor signal. The ADC used for digitization is integrated with a digital finite impulse response (FIR) filter that performs both filtering and decimation of the digital data stream from the modulator.

3.1.3 ADC

The ADC chosen for this project was the ADS1220 24-bit delta sigma ADC providing low power consumption, wide supply ranges from 2.3 V to 5.5 V, programmable gains up to 128 V/V and programmable data rates up to 2 kSPS. To avoid skew delays with regards to sampling consecutive channels on a single ADC for a multi-channel system, we decided to use three ADC, one for each channel of the

sensor. Each ADC will sample and hold the data at the same time thus making it simple to read the data from each ADC output buffer with no delay. The ADC communicates with the microcontroller over the serial peripheral interface (SPI) communication protocol and have been programmed to sample data at 100 samples per second (SPS), using a voltage reference of 3.3 V and unity gain.

3.1.4 Clock

An external programmable MEMS oscillator was used to drive each ADC. An external clock was used for two reasons: 1, since we are using three separate ADC, each ADC should be driven by the same clock and 2, the ADC does not support 100 SPS data rate and since the data rate scales proportionally with clock frequency, we programmed the MEMS oscillator at 2.34057 MHz to achieve the 100 SPS data rate.

3.1.5 GPS

The GPS module chosen for this project was the NEO-6m/7m from WaveShare. The module provides pulse per second (PPS) support and is interfaced with the microcomputer over the universal asynchronous receiver transmitter (UART) communication interface. The GPS synchronizes the NTP server on the microcomputer which is used for timestamping of data.

3.1.6 Microcontroller

The Arduino Mega was the choice of microcontroller used to capture data from the sensor through each 24-bit ADC. As stated previously, it communicates with the ADC over the SPI communication protocol. The Arduino Mega is set up as a master device which uses dedicated chip select and data ready pins along with the standard SPI pins for communication with each ADC setup as slave devices. The Arduino Mega in addition to the digitizer board comprise of the data acquisition sub-system of the SDAS.

3.1.7 Microcomputer

The RPi 3 Model B was the choice of microcomputer used to handle data processing. The RPi collects data from the Arduino Mega over a universal serial bus (USB) serial cable connected on both devices. The received data is buffered, processed into 512-byte MSEED packets and inserted into an internal ring buffer which can be accessed from any seedlink client. The RPi comprise of the data processing sub-system of the SDAS.

3.1.8 Power Supply and Power Board

The SDAS is operated from a dedicated power supply unit providing DC 13.5 V 5 A and a 12 V 7.0 Ah battery backup. The power board on the SDAS is connected to the power supply unit providing power to the main components which includes the digitizer board using DC 3.3 V 1 A, the Arduino Mega using DC 9 V 1 A, and the RPi using DC 5 V 3 A.

3.2 Software Overview

A simple software overview flow chart for the operation of the SDAS is shown in Fig. 2. The operation is broken down into two main sub-systems; data acquisition sub-system handled by the Arduino Mega and data processing sub-system handled by the RPi. We begin with the data processing sub-system. When the program runs, it first reads a configuration file and sets up certain configuration parameters where any failure in doing these two processes results in the program exiting and logging errors to a log file. It is up to the user to restart the program manually. Once the setup parameters have been configured, two processes happen. First, the RPi sends a start command to the Arduino Mega over the USB serial interface to begin the data acquisition sub-system. Once the start command has been received, the processes outlined in Fig. 2a are executed. Once finished executing, the Arduino Mega is constantly writing digitized data from the sensor to the USB serial interface. Secondly, after the start command has been sent to the Arduino Mega, the RPi waits for data to be read over the USB serial interface. The data is read and stored in a temporary

first in first out (FIFO) queue. From the queue, data is read and processed into 512-byte MSEED packets where each packet is 1 second long containing 100 data samples. The processing involves creating three MSEED packets every second, one for each channel. After the packets have been created, they are inserted into a ring buffer which is managed by the ringserver software.

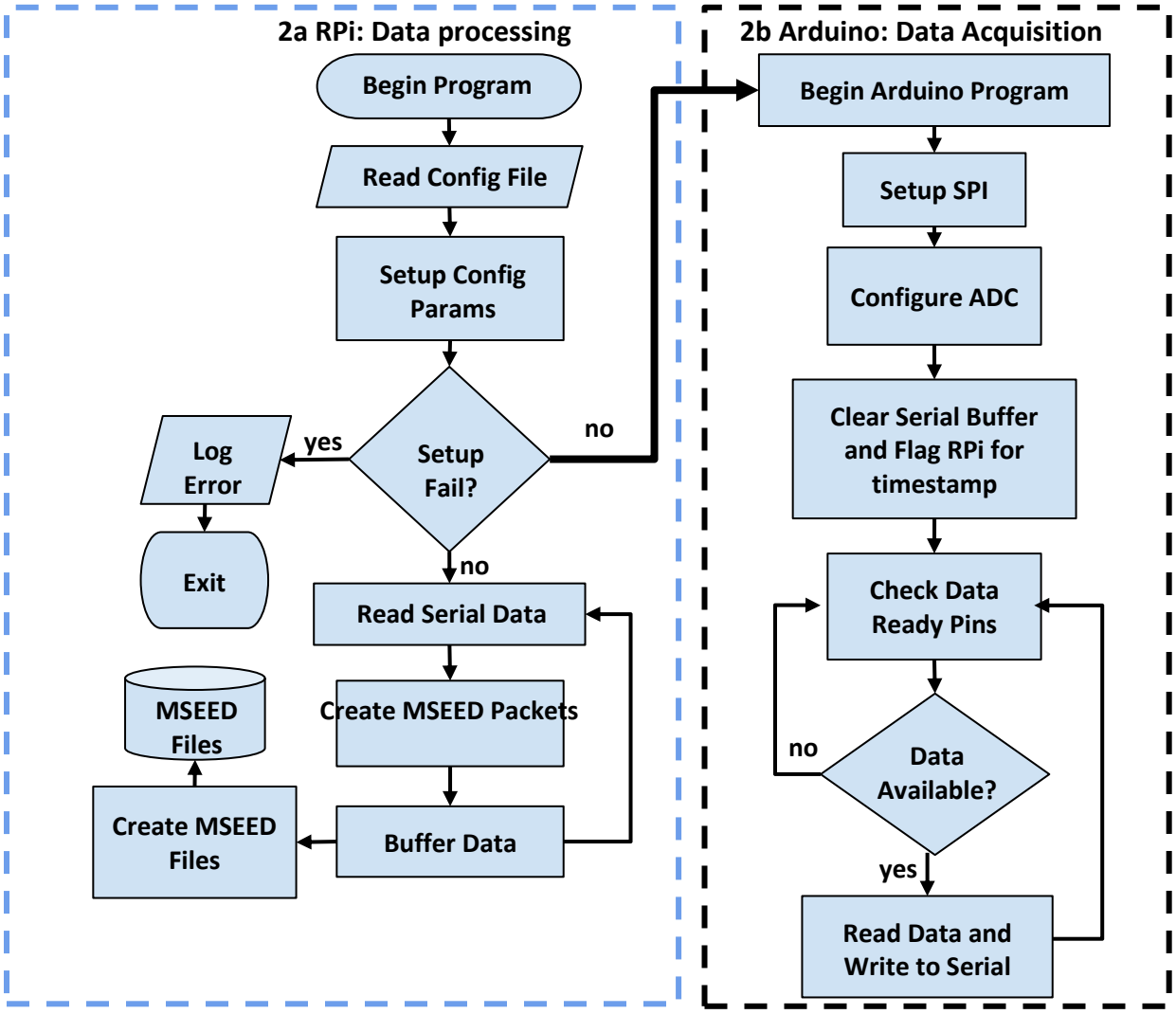

Figure 8: SDAS Software Overview Flowchart

The ringserver sofware allows for seedlink clients such as Swarm or Seisgram2K to connect and stream data packets out. The data is also written in twenty-minute MSEED files which are stored on a USB flash drive. The main software that handles the data processing is written in C which uses threads to handle certain processes and uses a combination of predefined libraries.

3.3 Timing

A GPS module is used to synchronize an NTP server installed on the SDAS to provide timestamping of data. The GPS module interfaces with the RPi over the UART interface. The Tx, Rx, and PPS pins of the GPS module connects to the Rx, Tx, and GPIO18 pins on the RPi. The PPS is very important in setting up time synchronization as it allows the SDAS to achieve 1 ms time accuracy. An NTP configuration file is used to configure the NTP software to synchronize the operating system (OS) clock of the RPi to the GPS

and PPS. A snippet of the configuration file is shown in Fig. 3. The configuration tells the NTP server where to locate the GPS and PPS source to be used in time synchronization. After the NTP server have been configured, a polling table shown in Fig. 4a is generated which list all possible time sources that can be used for time synchronization. *127.127.28.0 with refid .GPS. indicates the GPS signal is being used as the reference time source to synchronize the RPi OS clock. The second line, o127.127.22.0 with refid .PPS. indicates the PPS signal from the GPS is being used to discipline the reference GPS signal.

```
# GPS .. shared memory with gpsd "time1 0.130"
server 127.127.28.0 mode 24 minpoll 4 maxpoll 4 iburst true prefer
fudge 127.127.28.0 time1 0.140 flag1 1 refid GPS stratum 1

# PPS
server 127.127.22.0 mode 24 minpoll 4 maxpoll 4 iburst true
fudge 127.127.22.0 flag1 1 refid PPS
```

Figure 9: Snippet of NTP Configuration File

The third line, 0.ubuntu.pool.n is an internet time source which can be used for synchronization if the GPS fails. The column reach indicates if data can be read from the GPS time source. Once this value reaches 377, it indicates the reference time source was synchronized and the NTP server can achieve time accuracy within 1 ms as shown in Fig. 4b. The column delay shows the time it takes for the NTP server to query the GPS time source. The column offset shows the difference between the reference time and the RPi OS clock time and the column jitter indicates the variance in latency between several queries. Figure 4c shows the SDAS is NTP synchronized with time being formatted in universal time coordinated (UTC).

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*127.127.28.0    .GPS.            1 l    1   16  377    0.000   -5.606   2.133
o127.127.22.0    .PPS.            0 l   14   16  377    0.000    0.000   0.001
 0.ubuntu.pool.n .POOL.          16 p    -   64    0    0.000    0.000   0.001
```

(a)

```
synchronised to atomic clock at stratum 1
   time correct to within 1 ms
   polling server every 16 s
```

(b)

```
      Local time: Mon 2019-12-09 23:00:09 UTC
  Universal time: Mon 2019-12-09 23:00:09 UTC
        RTC time: n/a
       Time zone: Etc/UTC (UTC, +0000)
 Network time on: yes
NTP synchronized: yes
 RTC in local TZ: no
arvid@arvid-desktop:~$
```

(c)

Figure 10: (a) NTP Source List. (b) NTP Time Accuracy. (c) NTP Synchronization and Time Format

3.4 Data Format

MSEED is the seismological data format used to process data from the sensors. SEED which is a standard for the exchange of earthquake data is a data format primarily used for archival and exchange of seismological time series data. MSEED is a strip down version of SEED containing only seismological

time series data primarily used for processing. A MSEED file contains several data records which are concatenated together. The data records are fixed in length typically being 512-bytes or 4096-bytes long. The 512-bytes long records are typically used for near real time data stream while 4096-bytes long records are used for archiving. The first 48 bytes of a MSEED record is called the fixed section of the data header and provides important information about the data such as the network, station, location, channel, data rate, data encoding format, etc. Following the fixed section of the data header, additional information can be collected into blockettes which are 56-bytes long. Blockettes are basically grouping of information and allows for additional information to be present which cannot be described easily elsewhere. One of the important blockettes is blockette 1000 which contains information about the time series data length and how the data is encoded. Blockette 100 is also important which allows for floating point data rates to be specified. Following the blockettes, which is the remainder of data record contains the encoded time series data. Data can be encoded in different formats such as integers, floating point, ASCII, STEIM1 and STEIM2 and can be formatted in either 16-,24- or 32-bits. Libmseed is a C/C++ library used in the SDAS to process data into MiniSEED files. The library provides a framework for manipulating SEED data records in which it contains functionality to read and write MSEED data records and reconstruct time series data.

3.5 Data Storage and Telemetry

The SDAS stores data temporarily in a ring buffer until it is overwritten or stored permanently on a USB flash drive. The local storage on the flash drive was implemented in the event the SDAS was to be installed at a location without internet connectivity. If the SDAS is installed at a location with internet, then internet would be used as the telemetry method to pull data packets from the remote location to a server at a central location. The central location in this case would be a server installed at SRC headquarters. Figure 5 shows a block diagram on how internet telemetry is configured for the SDAS.

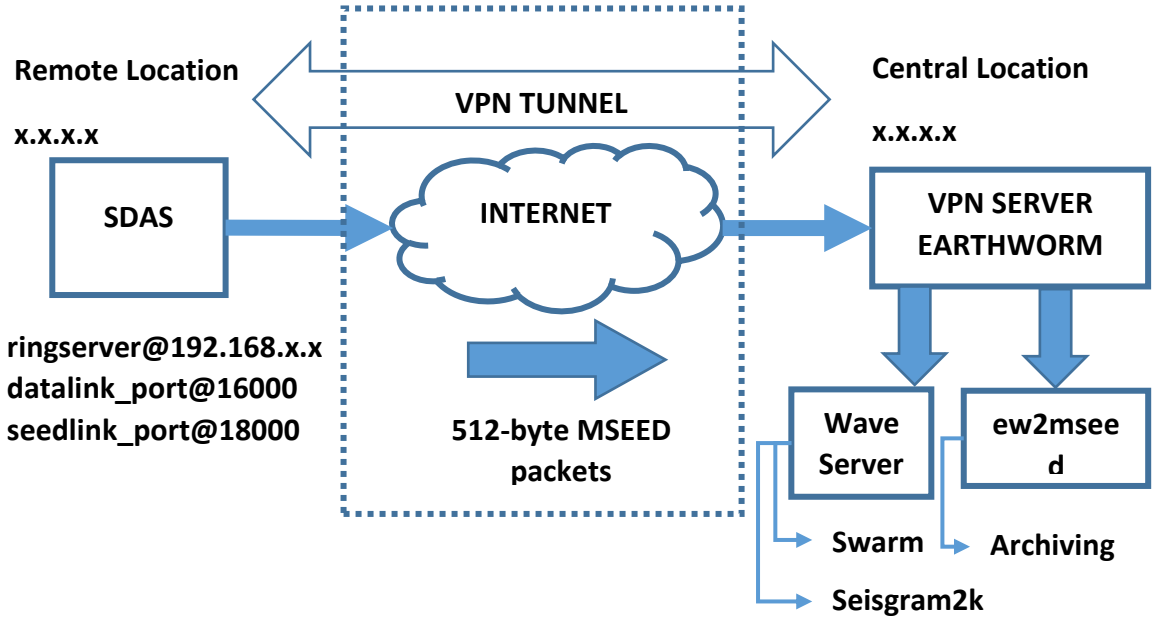

Figure 11: SDAS Telemetry Setup

A virtual private network (VPN) is used to establish a network connection between the SDAS installed at the remote location and a VPN server installed at the central location. When the SDAS is installed and powered on, a VPN client is used to establish a connection with the VPN server at the central location. Once a connection has been established, a 'virtual tunnel' is created between the SDAS and the VPN server allowing for communication. The virtual tunnel allows a user at the central location to easily access the

stored data on the USB flash drive of the SDAS by using any secure shell (SSH) file transfer program. Also, any MSEED reader can connect to the ringserver on the SDAS through the virtual tunnel internet protocol (IP) address and the seedlink port of the ringserver pulling data packets providing near real time data streaming. The use of the VPN allows for easy communication between the SDAS and central location bypassing any complicated network setup and dynamic public IP addressing schemes at most remote locations allowing for a constant reliable connection. Once the data packets arrive at the VPN server, they can then be forwarded to an Earthworm server for processing. The data from Earthworm can then be archived into one day MSEED files or forwarded to a waveserver which allows for MSEED readers such as Swarm or Sesimgram2k to display the data. In terms of data storage, the SDAS stores 20-minute MSEED files onto a 32 GB flash drive mounted on the RPi microcomputer. Each MSEED file is 594 KB in size and contains 1187 MSEED records each 512-bytes long and containing 100 samples. Each record consists of metadata which identifies important properties of the MSEED records such as sampling rate, record length, encoding, number of samples, start time etc. MSEED files are named according to: YYYY_MM_DDTHH_MM_SSx.mseed where YYYY is the year, MM is the month, DD is the day, HH is the hour, MM is the minute, SS is the second and x is for channel orientation (e – EW, n – NS, z – Z).

4. Discussion

The first prototype of the SDAS has been completed based on the design described in section 3. We are currently in the process of testing which will be done in two phases as described in section 5. The hardware aspects of the SDAS was heavily based on low-cost, easily available off the shelf components which include the ADXL335 accelerometer, Arduino Mega, and Raspberry Pi. In terms of software, the SDAS used the libmseed library for processing seismic data into MSEED format which is used by most commercial seismic instrumentation and compatible with most programs used for earthquake location and magnitude calculation and the ringserver library used for creating a ring buffer to store MSEED packets and allowing TCP/IP connections from clients to access the ringserver to stream the data out achieving near real time data streaming. These two predefined libraries in addition to others being open sourced and easily accessible provided easy integration with the software written in house to operate the SDAS. The overall design of the SDAS is expected to change as testing commences. The changes whether it be hardware or software will facilitate the resolution of issues that may arise from testing.

5. Conclusion and Future Work

In this paper, we have discussed the design of a low-cost SDAS which is going to be utilized in upgrading SRC seismic network. One of the main objectives of the SDAS is to keep cost low by utilizing open source hardware and software tools in the design. With a completed first prototype of the SDAS, we will begin the testing plan in two phases. The first phase of testing will be done in the lab which includes a combination of shake table testing and comparison test with commercial high-cost high accuracy instruments where we will compare data for accurate timing and quality. The second phase of testing will involve field testing where the SDAS will be installed at a seismic station site in Trinidad with an already installed commercial instrument. Data from both instruments will be compared for any seismic events analysing timing and data accuracy. Once satisfied with the testing results, we will build multiple units of the SDAS and deploy them at various seismic sites across Trinidad as part of the upgrade of SRC seismic network. To end the discussion of this report, the following will be included in future development of the SDAS; instrumentation response, PCB design, weak motion sensor integration, durable housing, and web interface.

References

[1] B. A. Bolt, J. E. Friday, R. A. Uhrhammer. A PC-based broadband digital seismograph network. *Geophysical Journal International*, 93, no. 3, (1988) 565–573.

[2] J. Havskov, G. Alguacil, 2016. Seismic Stations, in *Instrumentation in Earthquake Seismology*. Cham: Springer International Publishing, pp. 231-259.
[3] L. Lynch, 2019. Seismic Instrumentation Cost. Interview.
[4] V. K. Papanikolaou, C. Z. Karakostas. 2013. A low cost MEMS-based accelerograph, *Proceedings of Experimental Vibration Analysis for Civil Engineering Structures (EVACES'13)*, pp. 116-123
[5] C. Z. Karakostas, V. K. Papanikolaou. 2014. A low-cost instrumentation approach for seismic hazard assessment in urban areas. *WIT Transactions on Information and Communication Technologies*, *47*, pp.97-107.
[6] H.R. Avery, J.B. Berrill, M.B. Dewe. 2004. Design and development of a low-cost, high-performance, strong-motion accelerograph. In *Proc. Of the 2004 NZSEE Conference*, pp. 44-50
[7] R. E. Anthony, A. T. Ringler, D. C. Wilson, E. Wolin. 2019. 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*, 90, no. 1, pp. 219–228.