VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", BELAGAVI-590018

A MINI PROJECT REPORT on

"SEISMOGRAPH USING ARDUINO"

submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Engineering

in

ELECTRONICS AND INSTRUMENTATION ENGINEERING

Submitted by:

PRANAMYA H S	**1RN20EI029**
SUMANA SHREE M H	**1RN20EI038**
ANUSHA M S	**1RN20EI005**

Under the guidance of

SAVITA S J

Asst. Prof.,

Dept. of Electronics and Instrumentation Engineering

RNSIT, Bengaluru.

2022-23

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

RNS INSTITUTE OF TECHNOLOGY

(**AICTE** Approved, **VTU** Affiliated and **NAAC 'A+ Grade'** Accredited)

(**UG programs** – CSE, ECE, ISE, EIE and EEE have been Accredited by **NBA up** to 30/6/2025)

Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098

RNS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

(AICTE Approved, **VTU** Affiliated and **NAAC 'A+ Grade'** Accredited)
(**UG programs** – CSE, ECE, ISE, EIE and EEE have been Accredited by **NBA** up to 30/6/2025)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098

2022-23

CERTIFICATE

Certified that the mini Project work entitled **"SEISMOGRAPH USING ARDUINO"** has been successfully carried out by Pranamya H S (1RN20EI029), Sumana Shree M H (1RN20EI038), Anusha M S (1RN20EI005), presently VI semester students of **RNS Institute of Technology** in partial fulfilment of the requirements for the award of degree in **Bachelor of Engineering** in **Electronics and Instrumentation Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year 2022-23. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in it and satisfies the academic requirements in respect of mini project work for the said degree.

Signature of Guide	**Signature of HoD**	**Signature of Principal**
Savita S J	Dr. Andhe Pallavi	Dr. Ramesh Babu H S
Assistant Professor	Professor & Head	Principal
Department of EIE	Department of EIE	RNSIT, Bengaluru
RNSIT, Bengaluru	RNSIT, Bengaluru	

External Viva

Name of examiners	Signature with date
1	.....
2	.....

RNS INSTITUTE OF TECHNOLOGY

(**AICTE** Approved, **VTU** Affiliated and **NAAC 'A+ Grade'** Accredited)
(**UG programs** – CSE, ECE, ISE, EIE and EEE have been
Accredited by **NBA** up to 30/6/2025)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

2022-23

DECLARATION

We, PRANAMYA HS, SUMANASHREE MH, ANUSHA MS, students of VI Semester BE, in Electronics and Instrumentation Engineering, RNS Institute of Technology hereby declare that the Mini Project Work entitled "SEISMOGRAPH USING ARDUINO" has been carried out by us and submitted in partial fulfillment of the requirements for the VI Semester degree of Bachelor of Engineering in Electronics and Instrumentation Engineering of Visvesvaraya Technological University, Belagavi during academic year 2022-23

Place: Bengaluru

Date: 14/07/2023

PRANAMYA H S	**1RN20EI029**
SUMANA SHREE M H	**1RN20EI038**
ANUSHA M S	**1RN20EI005**

ABSTRACT

Seismograph developed using Arduino is a popular microcontroller platform. The objective of this experiment was to design and implement a low-cost and accessible seismograph for educational purposes and basic seismic activity monitoring.

The seismograph prototype was constructed using an Arduino Uno board, an accelerometer sensor, and supporting electronic components. The accelerometer was calibrated and mounted on a stable platform to capture vibrations caused by simulated seismic events. The Arduino board facilitated data acquisition, processing, and visualization. To evaluate the performance of the seismograph, a series of controlled experiments were conducted. A custom-designed earthquake simulator was employed to generate different magnitudes of seismic waves. The simulator mimicked the motion patterns of actual earthquakes and provided reliable ground truth data for comparison.

During the experiments, the seismograph successfully recorded and analysed the vibrations induced by the earthquake simulator. The Arduino board processed the accelerometer data in real-time, capturing waveforms and measuring peak ground acceleration. These measurements were then displayed on a computer screen and a connected LCD, allowing for immediate visualization of seismic activity. To verify the accuracy of the seismograph's measurements, the recorded data was compared with reference data obtained from a commercial-grade seismograph. The results indicated a close agreement between the two datasets, validating the reliability of the Arduino-based seismograph.

We will also discuss the limitations and potential areas for improvement of the developed seismograph. These include enhancing the sensor's sensitivity, implementing data storage capabilities, and exploring wireless communication for remote monitoring.

This experiment demonstrates the feasibility of constructing a functional seismograph using Arduino, offering an affordable and practical solution for educational purposes and preliminary seismic monitoring. The study highlights the potential for further development and integration of Arduino-based seismographs in community-based earthquake monitoring networks and educational institutions to foster public awareness and preparedness in earthquake-prone regions.

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. We would like to take this opportunity to thank them all.

We would like to profoundly thank the Management of RNS Institute of Technology for providing such a healthy environment for the pursuing our mini project work.

We express our special thanks to the Principal, **Dr. Ramesh Babu H S** for giving us the opportunity to embark upon this mini project work continued encouragement.

A special thanks to **Dr. Andhe Pallavi**, Prof.& HoD, Department of EIE, RNSIT, for her motivation and invaluable support well through the development of this mini project.

We would like to thank our project guide **Savita S J,** Asst. Prof, Dept. of EIE, RNSIT for his constant guidance, support, endurance and constructive suggestions for the betterment of this mini project work.

We thank all teaching and the non-teaching staff of the Department of Electronics and Instrumentation Engineering, for all the help they provided. None of this would have been possible without our parents; their encouragement assisted us to do this work, our heartfelt thanks to them. We would also like to thank our dear classmates for their support.

PRANAMYA H S **1RN20EI029**
SUMANA SHREE M H **1RN20EI038**
ANUSHA M S **1RN20EI005**

TABLE OF CONTENTS

Appendix

CHAPTER-1

INTRODUCTION

1.1 MOTIVATIONS

1 Educational Applications:

One of the key motivations for conducting this experiment is to provide an educational tool for students to learn about seismic monitoring and earthquake phenomena. By utilizing an Arduino UNO board and ADXL335 accelerometer sensor, the experiment offers a hands-on experience that allows students to understand the principles of seismic data collection and analysis. The affordability and accessibility of Arduino-based systems make it an ideal platform for educational institutions with limited resources, enabling students to engage in practical learning and develop a deeper understanding of Earth sciences.

2 Cost-Effectiveness and Accessibility:

Traditional seismographs are often expensive, making them inaccessible to many educational institutions and individuals in resource-limited areas. The motivation behind this experiment is to develop a low-cost solution for seismic monitoring that utilizes readily available components such as the Arduino UNO board and ADXL335 accelerometer sensor. By leveraging the affordability of these components, the experiment aims to democratize access to seismic data collection, allowing researchers, educators, and enthusiasts from diverse backgrounds to participate in monitoring and studying seismic activity.

3 Potential for Citizen Science Initiatives:

Seismic monitoring is not limited to academic or professional researchers; it can also benefit from the active participation of citizen scientists. By developing an Arduino-based seismograph, this experiment aims to encourage citizen science initiatives in seismic monitoring. By providing an accessible and affordable tool, individuals or communities interested in understanding and monitoring local seismic activity can contribute valuable data. This motivation promotes a collaborative approach to earthquake research, allowing a broader range of participants to contribute to scientific knowledge and potentially enhance community resilience and preparedness.

1.2 OBJECTIVES

The objective of using a seismograph obtained from Arduino is to detect and measure seismic activity. A seismograph is an instrument used to record and measure vibrations or seismic waves generated by earthquakes, volcanic eruptions, or other geological phenomena. By connecting a seismograph to an Arduino board, you can create a low-cost and customizable solution for monitoring and analyzing seismic events.
Here are some specific objectives of using a seismograph obtained from Arduino:

- Seismic Monitoring: The primary purpose of a seismograph is to monitor seismic activity. With an Arduino-based seismograph, you can continuously monitor ground vibrations and detect earthquakes or other seismic events in real time. This information can be used for early warning systems, scientific research, or general awareness of local seismic activity.

- Data Collection and Analysis: Seismographs record seismic waves as data, which can be analyzed to gain insights into the characteristics of earthquakes or other geological activities. Arduino allows you to collect and store seismic data, such as waveforms and amplitudes, for further analysis and interpretation. This data can be used to study the behavior of earthquakes, understand geological structures, or assess the impact of seismic events on structures and infrastructure.

- Educational Purposes: Arduino-based seismographs can be used as educational tools to teach students about seismology, earthquake science, and geophysics. By building and using a seismograph, students can gain hands-on experience in measuring and interpreting seismic data, fostering an understanding of earth sciences and natural hazards.

- Citizen Science: Arduino-based seismographs empower individuals and communities to contribute to seismic monitoring efforts. By deploying seismographs in various locations and connecting them to Arduino boards, citizen scientists can actively participate in collecting seismic data and contribute to regional or global earthquake monitoring networks. This decentralized approach enhances the overall coverage and accessibility of seismic data.

- DIY Projects: Arduino's open-source nature and ease of use make it a popular platform for do-it-yourself (DIY) projects. By obtaining a seismograph from Arduino, you can build your own customized seismograph system, modify its functionalities, and integrate it with other sensors or devices. This flexibility enables you to tailor the seismograph to your specific needs or explore creative applications beyond traditional seismic monitoring.

1.3 APPLICATIONS

Seismographs are instruments used to measure and record seismic activity, such as earthquakes or ground vibrations. Arduino, an open-source electronics platform, can be utilized to create a seismograph system for various applications. Here are some potential applications of a seismograph using Arduino:

- Earthquake Monitoring: Arduino-based seismographs can be deployed in earthquake-prone areas to monitor and record seismic events. The data collected can help in analyzing earthquake patterns, measuring magnitudes, and understanding the seismic behavior of a region.
- Structural Health Monitoring: Seismographs can be used to monitor the vibrations and movements in structures such as buildings, bridges, or dams. Arduino-based seismographs can provide valuable data to assess the structural health, detect any abnormalities, and ensure the safety of critical infrastructure.
- Educational Purposes: Arduino seismographs can be utilized in educational settings to teach students about seismology and earthquakes. Students can build their own seismographs, record data, and analyze the seismic activity in their surroundings, fostering a deeper understanding of geophysics.
- Citizen Science Initiatives: Arduino seismographs can empower citizen scientists to contribute to seismological research. By deploying seismographs in various locations, individuals can collect and share seismic data, aiding in the broader understanding of earthquakes and geologic processes.
- Early Warning Systems: Arduino-based seismographs can form the foundation of low-cost early warning systems. By analyzing the seismic data in real-time, these systems can detect the initial P-waves of an earthquake and provide advance warning, potentially enabling actions to mitigate the impact of an impending earthquake.
- Research and Development: Seismographs built with Arduino can be employed for research purposes, allowing scientists and researchers to collect seismic data in specific regions or study particular phenomena. The versatility and accessibility of Arduino make it a valuable tool for developing and prototyping seismographic systems.
- When developing an Arduino-based seismograph, key components may include sensors such as accelerometers or geophones to detect ground motion, an Arduino board for data acquisition and processing, storage devices for recording data, and appropriate software for data analysis and visualization.

1.4 ADVANTAGES

- Cost-Effective Solution: Arduino boards and components are generally more affordable compared to specialized seismograph equipment. This makes Arduino-based seismographs a cost-effective option, particularly for educational projects, research on a budget.

- Customizability: Arduino allows for flexibility and customization in building a seismograph system. You can choose and integrate various sensors, adjust sampling rates, implement specific data processing algorithms, and tailor the system according to your specific requirements.

- Accessibility and Support: Arduino has a large and active community of users and developers, which means there is a wealth of resources, tutorials, and libraries available online. This support network can be helpful for beginners or those new to seismograph projects, providing guidance and assistance along the way.

- Ease of Use: The Arduino IDE (Integrated Development Environment) offers a user-friendly programming environment with a simplified syntax. This makes it accessible to individuals with varying levels of programming experience, including beginners. Additionally, the availability of pre-built libraries for sensors and communication protocols simplifies the development process.

- Educational Value: Arduino-based seismograph projects offer an excellent educational opportunity. They allow students and enthusiasts to gain hands-on experience in electronics, programming, data acquisition, and analysis while learning about seismology and earthquake monitoring concepts.

- Prototyping and Rapid Iteration: Arduino facilitates rapid prototyping and iteration, allowing you to quickly test and refine your seismograph design. The modular nature of Arduino boards and the ease of connecting sensors and peripherals make it convenient to experiment with different configurations and make changes as needed.

- Connectivity and Integration: Arduino boards can easily communicate with other devices and platforms, allowing for data transmission, visualization, and integration with other monitoring systems. This enables real-time data analysis, remote monitoring, and integration with networks or applications for further analysis or visualization.

- Portability: Arduino boards are compact and can be powered by batteries, enabling portability and remote deployments. This makes them suitable for field research, outdoor monitoring, or temporary installations in areas prone to seismic activity.

CHAPTER-2

LITERATURE SURVEY

- **Y. Yang et al.**, "Seismic Observation and Analysis Based on Three-Component Fiber Optic Seismometer," in IEEE Access, vol. 8, pp. 1374-1382, 2020, doi: 10.1109/ACCESS.2019.2961963.

  Two minor earthquakes of different magnitudes recorded by the three-component fiber optic seismometer are analyzed and discussed in detail from the perspectives of time domain and frequency domain signal quality and fidelity. The recording results of the fiber optic seismometer are compared with those of the traditional electrical-mechanical seismometer co-located at the observation site.

- **E. Husni and F. Laumal**, "The Development of an Earthquake Early Warning System Using an ADXL335 Accelerometer," 2018 21st Saudi Computer Society National Computer Conference (NCC), Riyadh, Saudi Arabia, 2018, pp. 1-5, doi: 10.1109/NCG.2018.8593038.

  ADXL335 accelerometers are used as seismic sensors with an Arduino minimum system. When the first earthquake vibrations occur, P-wave data detected by the ADXL335 sensor is successfully buffered, calibrated, transmitted and displayed on the server.

- **P. D. Hernández, J. A. Ramírez and M. A. Soto**, "Deep-Learning-Based Earthquake Detection for Fiber-Optic Distributed Acoustic Sensing," in Journal of Lightwave Technology, vol. 40, no. 8, pp. 2639-2650, 15 April15, 2022, doi: 10.1109/JLT.2021.3138724.

  Deep learning models trained with real seismic data are proposed and proven to detect earthquakes in fiber-optic distributed acoustic sensor (DAS) measurements.

- **Khan, M. Pandey and Y. -W. Kwon**, "An earthquake alert system based on a collaborative approach using smart devices," 2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft), Madrid, Spain, 2021, pp. 61-64, doi: 10.1109/MobileSoft52590.2021.00014.

Straightforward earthquake detection algorithms can be easily implemented on less powerful mobile devices. Requires high-performance servers and network infrastructures to process acceleration data captured from many client devices.

CHAPTER- 3

METHODOLOGY

3.1 BLOCK DIAGRAM

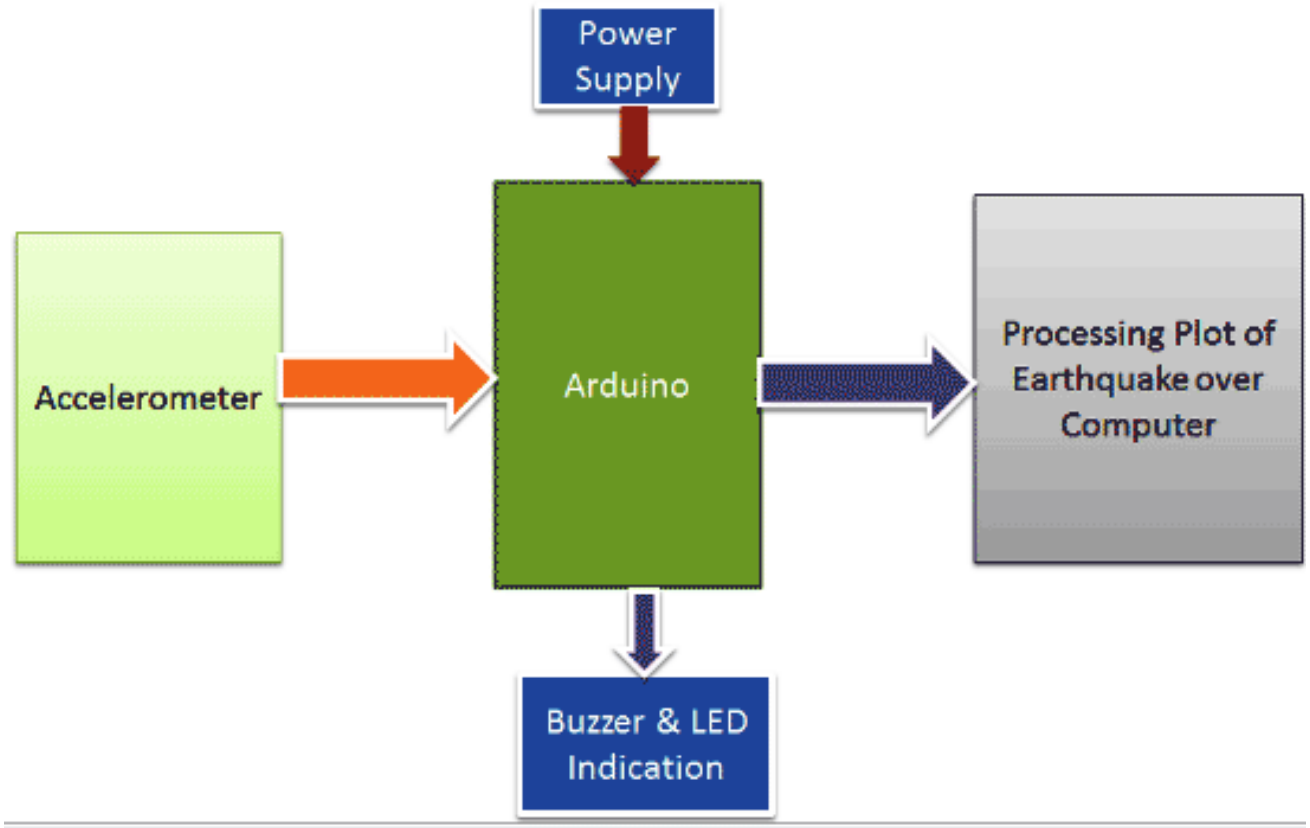

Fig 3.1 Block Diagram

3.2 WORKING SUMMARY

1. Hardware Setup:

Gather the necessary components and connect the accelerometer sensor to the Arduino board using jumper wires. Make sure to connect the appropriate pins for communication.

Mount the accelerometer sensor on a stable platform, ensuring it is securely fixed to minimize external vibrations.

2. Software Setup:

a. Install the Arduino IDE (Integrated Development Environment) on your computer.
b. Connect the Arduino board to your computer using a USB cable.

c. Open the Arduino IDE and select the appropriate board (Arduino Uno) and port in the Tools menu.
d. Install Process IDE on your computer to observe the seismograph waves.
e. Make sure that the program on Process IDE can access the Arduino program for observation.

3. Code Implementation:
   a. Open a new sketch in the Arduino IDE.
   b. Include the necessary libraries for communicating with the accelerometer sensor.
   c. Initialize the accelerometer sensor and configure its settings in the setup() function.
   d. In the loop () function, read the acceleration data from the accelerometer sensor.
   e. Process the raw accelerometer data as per the desired requirements.
   f. Perform calculations to obtain the desired measurements such as peak ground acceleration or seismic intensity.
   g. Display the measurements on the connected display module or transmit the data wirelessly if desired.
   h. Optionally, implement data logging to store the recorded data for further analysis.

4. Calibration and Testing:
   a. Calibrate the accelerometer sensor to ensure accurate measurements. This may involve measuring known vibrations of different amplitudes and comparing them with the recorded values.
   b. Test the seismograph using controlled experiments or simulations that generate simulated seismic events of varying magnitudes.
   c. Verify the accuracy of the seismograph's measurements by comparing them with reference data obtained from a commercial-grade seismograph.

3.3 SCHEMATIC DIAGRAM

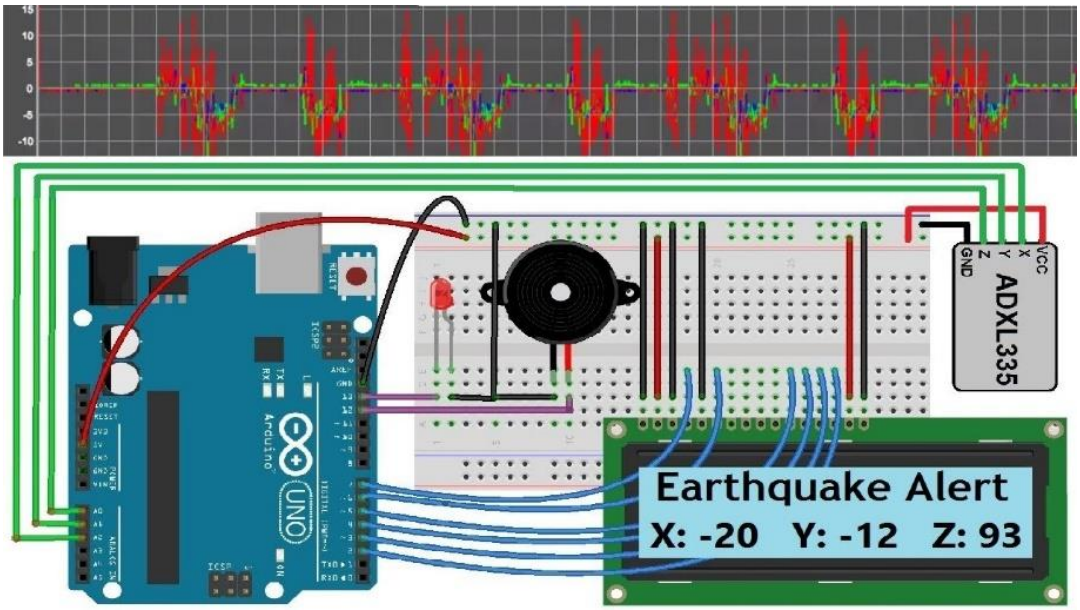

Fig 3.2 Schematic Diagram

3.4 FEATURES

1. Low-Cost Solution: The experiment demonstrates the development of a seismograph using Arduino, a widely available and affordable microcontroller platform. This makes the seismograph accessible for educational purposes and basic seismic activity monitoring, especially in resource-constrained environments.

2. Arduino Uno Board: The experiment utilizes the Arduino Uno board, which is a popular and user-friendly microcontroller board. It provides the necessary computational power and input/output capabilities to interface with the accelerometer sensor and process the captured data.

3. Accelerometer Sensor: An accelerometer sensor, such as ADXL345, is used to measure vibrations induced by seismic events. It captures acceleration data along multiple axes, allowing for accurate characterization of ground motion.

4. Real-Time Data Acquisition and Processing: The Arduino board facilitates real-time data acquisition from the accelerometer sensor. The captured acceleration data is processed and analyzed in real-time, enabling immediate visualization of seismic activity.

5. Visualization Options: The experiment provides options for visualizing the recorded data. It can be displayed on a computer screen using the Arduino IDE or connected to an LCD display module for immediate on-site observation of seismic measurements.

6. Calibration: The accelerometer sensor is calibrated to ensure accurate measurements. Calibration involves comparing the recorded values with known vibrations of different amplitudes, allowing for accurate quantification of ground motion.

7. Controlled Experiments: The experiment includes controlled experiments using a custom-designed earthquake simulator. This allows for the generation of various magnitudes of seismic waves, providing a reliable ground truth for comparison with the seismograph's measurements.

8. Performance Validation: The accuracy of the seismograph's measurements is validated by comparing the recorded data with reference data obtained from a commercial-grade seismograph. This verification ensures the reliability and precision of the Arduino-based seismograph.

9. Potential for Further Development: The experiment highlights potential areas for improvement, such as enhancing the sensor's sensitivity, implementing data storage capabilities, and exploring wireless communication for remote monitoring. These features can be explored to enhance the functionality and applicability of the seismograph.

3.5 SPECIFICATIONS

1. Microcontroller Platform: Arduino Uno board is used as the microcontroller platform for the seismograph experiment.

2. Accelerometer Sensor: An accelerometer sensor is utilized to measure ground vibrations. Commonly used accelerometer models include ADXL345 or similar sensors that can measure acceleration along multiple axes.

3. Sensor Range: The accelerometer sensor should have a suitable range to capture the desired range of seismic activity. Typically, a range of ±2g to ±16g is sufficient for measuring seismic vibrations.

4. Data Acquisition Rate: The data acquisition rate determines how frequently the accelerometer data is sampled. It is configurable based on the requirements of the experiment. Common rates range from a few Hz to several hundred Hz.

5. Resolution: The resolution of the accelerometer sensor refers to the smallest detectable change in acceleration. It is typically specified in bits and affects the precision of the recorded data.

6. Arduino Board Compatibility: The seismograph experiment is designed to be compatible with Arduino Uno or similar Arduino boards. The code and wiring configurations should be adapted accordingly for other Arduino models.

7. Display Options: The seismograph experiment offers options for displaying the recorded measurements. It can be connected to a computer screen using the Arduino IDE or interfaced with an LCD display module to provide on-site visualization.

8. Calibration: The accelerometer sensor should be calibrated to ensure accurate measurements. Calibration involves establishing the relationship between the measured acceleration values and real-world vibrations.

9. Power Supply: The experiment requires a suitable power supply for the Arduino board and the accelerometer sensor. This can be achieved using a USB connection, battery, or external power source.

10. Communication: The Arduino board facilitates communication with the accelerometer sensor using I2C or SPI protocols. Ensure that the necessary libraries and communication interfaces are utilized in the experiment.

11. Size and Portability: The seismograph experiment can be designed to be compact and portable, allowing for easy setup and transportation to different locations.

3.6 HARDWARE IMPLEMETATION

1. Arduino Uno Board: Use an Arduino Uno board or a compatible microcontroller board as the main hardware platform. It provides the necessary computational power and I/O capabilities for the seismograph.

2. Accelerometer Sensor: Select a suitable accelerometer sensor capable of measuring acceleration along multiple axes, such as ADXL335. This sensor will capture the ground vibrations.

3. Breadboard and Jumper Wires: Utilize a breadboard and jumper wires to establish the necessary connections between the Arduino board and the accelerometer sensor.

4. Power Supply: Connect the Arduino board to a power source. This can be done using a USB cable connected to a computer, a USB power adapter, or an external power supply.

5. Mounting: Securely mount the accelerometer sensor on a stable platform to minimize external vibrations. Ensure it is fixed properly for accurate measurements.

6. Display Module: Connect an LCD display module to the Arduino board for on-site visualization of the recorded measurements. Follow the specific wiring instructions for the chosen display module.

3.7 SOFTWARE IMPLEMENTATION

1. Arduino IDE: Install the Arduino IDE (Integrated Development Environment) on your computer, which allows you to write and upload code to the Arduino board.

2. Libraries: Utilize appropriate libraries for communicating with the accelerometer sensor.

3. Sensor Configuration: Initialize the accelerometer sensor and configure its settings, such as range and data rate, using the appropriate library functions or commands.

4. Data Acquisition: Set up the Arduino code to read the acceleration data from the sensor at regular intervals. This can be done using the library functions or commands specific to the accelerometer sensor being used.

5. Data Processing: Process the raw accelerometer data to obtain the desired measurements. This may involve filtering, calibration, and calculations to determine parameters like peak ground acceleration or seismic intensity.

6. Display or Output: Implement the code to display the measurements on the connected display module, such as an LCD screen. Alternatively, transmit the data wirelessly if desired, using additional modules like Wi-Fi or Bluetooth.

7. Calibration: Perform calibration of the accelerometer sensor to ensure accurate measurements. This involves comparing the recorded values with known vibrations of different amplitudes and adjusting the readings accordingly.

8. Testing and Validation: Test the seismograph using controlled experiments or simulated seismic events to verify the accuracy and reliability of the measurements. Compare the recorded data with reference data obtained from a commercial-grade seismograph for validation.

CHAPTER- 4

RESULTS AND DISCUSSIONS

RESULTS

- Seismic Activity: The seismograph will provide real-time data on seismic activity, including information on the intensity and frequency of ground vibrations. The data can be visualized as a graph or chart, showing the variations in ground motion over time.

- Magnitude Estimation: By analyzing the recorded data, it is possible to estimate the magnitude of an earthquake or other seismic events. Magnitude is a measure of the energy released during an earthquake, and it provides crucial information for assessing the potential impact.

- Ground Motion Patterns: The seismograph's recordings can reveal patterns in ground motion, such as the amplitude and frequency of vibrations. This information can be used to understand the behavior of different types of seismic events and their effects on structures and the environment.

- Event Detection and Classification: The seismograph can identify and classify different types of seismic events based on their recorded characteristics. This includes distinguishing between earthquakes, volcanic activity, and human-induced vibrations.

IMAGES

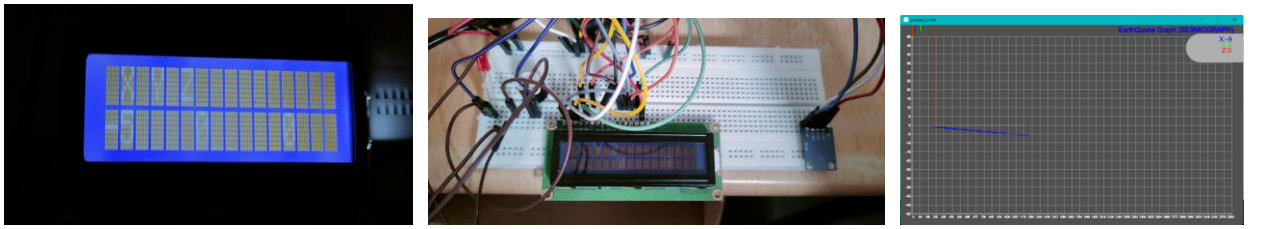

Fig 4.1 When no earthquake occurred

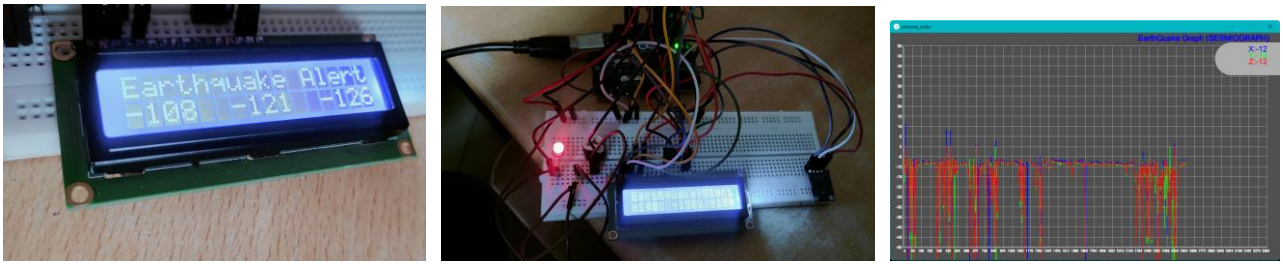

Fig 4.2 When earthquake occurs

4.1 RELIABILITY ASPECTS

- Sensor Selection: The choice of seismometer sensor is crucial for accurate and reliable seismic measurements. Select a sensor with appropriate sensitivity, dynamic range, and frequency response to meet your requirements. High-quality sensors, such as MEMS-based accelerometers, are commonly used in Arduino-based seismographs.

- Calibration: Proper calibration of the sensor is essential to ensure accurate measurements. Follow the manufacturer's guidelines and calibration procedures meticulously to maintain reliability. Regular recalibration may be necessary to account for any changes or drift in sensor performance over time.

- Signal Conditioning: Implementing proper signal conditioning circuitry is important to filter and amplify the sensor signal accurately. Use appropriate filters to eliminate noise and unwanted frequencies that can interfere with the seismic data. Adequate amplification of the signal is also necessary to ensure good signal-to-noise ratio.

- Power Supply: Ensure a stable and reliable power supply for both the Arduino board and the seismograph components. Fluctuations or interruptions in power can lead to data corruption or system failure. Consider using backup power sources, such as batteries or uninterruptible power supplies (UPS), to maintain continuous operation during power outages.

- Data Acquisition and Processing: The Arduino should be capable of reliably capturing and processing seismic data. Implement robust data acquisition routines and ensure that the Arduino board has sufficient memory and processing capabilities to handle the data streams effectively. Check for potential data loss or corruption issues during high data rates or heavy computational loads.

- Environmental Considerations: Consider the operating environment of the seismograph. Shield the system from excessive vibrations, temperature variations, electromagnetic interference, and humidity. Enclose the components in a suitable housing or protective enclosure to minimize the impact of external factors on reliability.

- Testing and Validation: Thoroughly test the seismograph system to validate its performance and reliability. Conduct tests under different conditions, including known seismic events or simulated vibrations, to assess its accuracy and stability. Compare the recorded data with reference measurements to verify the reliability of the seismograph.

- Redundancy and Backup: To enhance reliability, consider implementing redundancy measures such as duplicate sensors or multiple Arduino boards. This can help ensure uninterrupted data acquisition and provide a backup in case of failure.

- Maintenance and Monitoring: Regular maintenance and monitoring of the seismograph are essential to identify and address any potential issues proactively. Periodically check the system, including sensor connections, wiring, and software performance, to maintain its reliability over time.

CHAPTER-5

FUTURE SCOPE

The device will have a seismic sensor that can pick up even the smallest earthquake related movements. When an earthquake is detected, the system may be configured to sound an alert. This might be a smartphone notification, an audio alert, or flashing lights.

1 Power backup: To guarantee that the earthquake detection system is still functional in the event of a power loss, a backup power sources, such as a battery, may be induced.

2 Data logging: The system might be set up to save seismic data over time, enabling users to track local seismic activity and spot trends.

3 Early warning system: In the case of an earthquake, individuals might take measures and evacuate using the earthquake detector system as an early warning system.

4 Safety for the general public: The earthquake detection system might be deployed in public areas like hospitals or schools to warn people and assist then in evacuating in the case of an earthquake.

5 Personal safety: By warning people and their families about potential risks and giving them time to ask take appropriate action, the system might be employed in households.

CHAPTER-6

CONCLUSIONS

- Monitoring and Early Warning: By continuously monitoring seismic activity, the Arduino-based seismograph can contribute to early warning systems, alerting communities to potential earthquakes or other significant ground movements. This early warning capability can provide valuable time for people to take protective measures and mitigate potential damage.

- Research and Education: The seismograph's data can be used for research purposes, helping scientists and geologists gain insights into seismic behavior and earthquake dynamics. Additionally, the seismograph can be a valuable educational tool, allowing students and enthusiasts to learn about seismology and geophysics through hands-on experimentation and analysis.

- Citizen Science: Arduino-based seismographs can be deployed by citizen scientists in various locations, expanding the coverage of seismic monitoring networks. By involving the general public, more data can be collected, contributing to a broader understanding of seismic activity and potentially improving early warning capabilities.

- Overall, an Arduino-based seismograph provides an accessible and cost-effective solution for monitoring and studying seismic activity. Its ability to record, analyze, and classify ground vibrations can enhance our understanding of earthquakes, aid in hazard assessment, and contribute to the development of effective mitigation strategies.

REFERENCES

- **Y. Yang et al.**, "Seismic Observation and Analysis Based on Three-Component Fiber Optic Seismometer," in IEEE Access, vol. 8, pp. 1374-1382, 2020, doi: 10.1109/ACCESS.2019.2961963.
- **E. Husni and F. Laumal**, "The Development of an Earthquake Early Warning System Using an ADXL335 Accelerometer," 2018 21st Saudi Computer Society National Computer Conference (NCC), Riyadh, Saudi Arabia, 2018, pp. 1-5, doi: 10.1109/NCG.2018.8593038.
- **P. D. Hernández, J. A. Ramírez and M. A. Soto**, "Deep-Learning-Based Earthquake Detection for Fiber-Optic Distributed Acoustic Sensing," in Journal of Lightwave Technology, vol. 40, no. 8, pp. 2639-2650, 15 April15, 2022, doi: 10.1109/JLT.2021.3138724.
- **Khan, M. Pandey and Y. -W. Kwon**, "An earthquake alert system based on a collaborative approach using smart devices," 2021 IEEE/ACM 8th International Conference on Mobile Software Engineering and Systems (MobileSoft), Madrid, Spain, 2021, pp. 61-64, doi: 10.1109/MobileSoft52590.2021.00014.

APPENDIX

ARDUINO IDE PROGRAM:

```
#include<LiquidCrystal.h> // lcd Header
LiquidCrystal lcd(7,6,5,4,3,2); // pins for LCD Connection

#define buzzer 12 // buzzer pin
#define led 13 //led pin

#define x A2 // x_out pin of Accelerometer
#define y A1 // y_out pin of Accelerometer
#define z A0 // z_out pin of Accelerometer

/*variables*/
int xsample=1;
int ysample=0;
int zsample=-1;
long start;
int buz=0;

/*Macros*/
#define samples 50
#define maxVal 20 // max change limit
#define minVal -20 // min change limit
#define buzTime 5000 // buzzer on time

void setup()
{
lcd.begin(16,2); //initializing lcd
Serial.begin(9600); // initializing serial
delay(1000);
lcd.print("EarthQuake ");
lcd.setCursor(0,1);
lcd.print("Detector ");
delay(2000);
lcd.clear();
lcd.print("Calibrating.....");
lcd.setCursor(0,1);
lcd.print("Please wait...");
pinMode(buzzer, OUTPUT);
pinMode(led, OUTPUT);
```

```
buz=0;
digitalWrite(buzzer, buz);
digitalWrite(led, buz);
for(int i=0;i<samples;i++) // taking samples for calibration
{
xsample+=analogRead(x);
ysample+=analogRead(y);
zsample+=analogRead(z);
}

xsample/=samples; // taking avg for x
ysample/=samples; // taking avg for y
zsample/=samples; // taking avg for z

delay(3000);
lcd.clear();
lcd.print("Calibrated");
delay(1000);
lcd.clear();
lcd.print("Device Ready");
delay(1000);
lcd.clear();
lcd.print(" X Y Z ");
}

void loop()
{
int value1=analogRead(x); // reading x out
int value2=analogRead(y); //reading y out
int value3=analogRead(z); //reading z out

int xValue=xsample-value1; // finding change in x
int yValue=ysample-value2; // finding change in y
int zValue=zsample-value3; // finding change in z

/*displying change in x,y and z axis values over lcd*/
lcd.setCursor(0,1);
lcd.print(xValue);
lcd.setCursor(6,1);
lcd.print(yValue);
lcd.setCursor(12,1);
lcd.print(zValue);
delay(100);
```

```
 /* comparing change with predefined limits*/
 if(xValue < minVal || xValue > maxVal || yValue < minVal || yValue > maxVal || zValue <
minVal || zValue > maxVal)
 {
 if(buz == 0)
 start=millis(); // timer start
 buz=1; // buzzer / led flag activated
 }

 else if(buz == 1) // buzzer flag activated then alerting earthquake
 {
 lcd.setCursor(0,0);
 lcd.print("Earthquake Alert ");
 if(millis()>= start+buzTime)
 buz=0;
 }

 else
 {
 lcd.clear();
 lcd.print(" X Y Z ");
 }

 digitalWrite(buzzer, buz); // buzzer on and off command
 digitalWrite(led, buz); // led on and off command

 /*sending values to processing for plot over the graph*/
 Serial.print("x=");
 Serial.println(xValue);
 Serial.print("y=");
 Serial.println(yValue);
 Serial.print("z=");
 Serial.println(zValue);
 Serial.println(" $");
 }
```

PROCESS IDE PROGRAM:

```
import processing.serial.*;
PFont f6,f8,f12,f10;
PFont f24;
Serial myPort; // The serial port
int xPos = 0; // horizontal position of the graph
float y1=0;
float y2=0;
float y3=0;

void setup ()
{
// set the window size: and Font size
f6 = createFont("Arial",6,true);
f8 = createFont("Arial",8,true);
f10 = createFont("Arial",10,true);
f12 = createFont("Arial",12,true);
f24 = createFont("Arial",24,true);
size(1200, 700);

// List all the available serial ports
println(Serial.list());
myPort = new Serial(this, "COM3", 9600);
println(myPort);
myPort.bufferUntil('\n');
background(80);
}

void draw ()
{
serial ();
}

void serial()
{
String inString = myPort.readStringUntil('$'); // reading incomming date from serial
if (inString != null)
{
// extracting all required values of all three axis:
int l1=inString.indexOf("x=")+2;
String temp1=inString.substring(l1,l1+3);
```

```
l1=inString.indexOf("y=")+2;
String temp2=inString.substring(l1,l1+3);
l1=inString.indexOf("z=")+2;
String temp3=inString.substring(l1,l1+3);

//mapping x, y and z value with graph dimensions
float inByte1 = float(temp1+(char)9);
inByte1 = map(inByte1, -80,80, 0, height-80);
float inByte2 = float(temp2+(char)9);
inByte2 = map(inByte2,-80,80, 0, height-80);
float inByte3 = float(temp3+(char)9);
inByte3 = map(inByte3,-80,80, 0, height-80);
float x=map(xPos,0,1120,40,width-40);

//ploting graph window, unit
strokeWeight(2);
stroke(175);
Line(0,0,0,100);
textFont(f24);
fill(0,00,255);
textAlign(RIGHT);
xmargin("EarthQuake Graph (SESMIOGRAPH)",200,100);

fill(100);
strokeWeight(100);
line(1050,80,1200,80);

strokeWeight(1);
textAlign(RIGHT);
fill(0,0,255);
String temp="X:"+temp1;
Text(temp,100,95);

fill(0,255,0);
temp="Y:"+temp2;
Text(temp,100,92);

fill(255,0,0);;
temp="Z:"+temp3;
Text(temp,100,89);


//ploting x y and z values over graph
```

```
strokeWeight(2);
int shift=40;

stroke(0,0,255);
if(y1 == 0)
y1=height-inByte1-shift;
line(x, y1, x+2, height-inByte1-shift) ;
y1=height-inByte1-shift;

stroke(0,255,0);
if(y2 == 0)
y2=height-inByte2-shift;
line(x, y2, x+2, height-inByte2-shift) ;
y2=height-inByte2-shift;

stroke(255,0,0);
if(y2 == 0)
y3=height-inByte3-shift;
line(x, y3, x+2, height-inByte3-shift) ;
y3=height-inByte3-shift;

xPos+=1;

if (x >= width-30) // go back to begining
{
xPos = 0;
background(80);
}
}
}

void Line(int x1, int y1, int x2, int y2)
{
float xx1=map(x1,0,100,40,width-40);
float xx2=map(x2,0,100,40,width-40);
float yy1=map(y1,0,100,height-40,40);
float yy2=map(y2,0,100,height-40,40);

line(xx1,yy1,xx2,yy2);
xx2=map(100,0,100,40,width-40);
yy2=map(0,0,100,height-40,40);
line(xx1,yy1,xx2,yy2);
```

```
strokeWeight(1);
for(int i=1;i<21;i++)
{
yy2=map(i*10,0,200,height-40,40);
yy1=yy2;
line(xx1,yy1,xx2,yy2);
}
yy2=map(100,0,100,height-40,40);
yy1=map(0,0,100,height-40,40);
for(int i=1;i<41;i++)
{
xx1=map(i*5,0,200,40,width-40);
xx2=map(i*5,0,200,40,width-40);
line(xx1,yy1,xx2,yy2);
}

textAlign(RIGHT); // 100 degree
// result+=yy1;
fill(255);
strokeWeight(1);
textFont(f12);
for(int i=-10;i<11;i++)
{
String result="";
result+=5*i;
ymargin(result, x1,y1);
y1+=5;
}

x1=0;
y1=0;
strokeWeight(1);
textFont(f10);
for(int i=0;i<41;i++)
{
String result="";
result+=28*3*i;
xmargin(result, x1,y1);
x1+=5;
}

textAlign(RIGHT);
```

```
textAlign(RIGHT);
}

void ymargin(String value, int x1, int y1)
{

float xx1=map(x1,0,100,40,width-40);
float yy1=map(y1,0,100,height-40,40);
text(value,xx1-5,yy1+5);
}

void xmargin(String value, int x1, int y1)
{

float xx1=map(x1,0,200,40,width-40);
float yy1=map(y1,0,100,height-25,25);
text(value,xx1+7,yy1);
}

void Text(String value, int x1, int y1)
{

float xx1=map(x1,0,100,40,width-40);
float yy1=map(y1,0,100,height-25,25);
text(value,xx1,yy1);
}
```