COLLEGE CODE:1105

COLLEGE NAME:GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY

DEPARTMENT:ELECTRONICS AND COMMUNICATION ENGINEERING

STUDENT NM-ID:6A777A26E4C04BDBD65419551663F813

ROLL NO:110523106014

DATE:4/05/2023

COMPLETED THE PROJECT AS

STRUCTURAL HEALTH MONITORING

TECHNOLOGY-PROJECT NAME-IBM-AI-EBPL

SUBMITTED BY,

Your Name and team member names

S. SRIMATHI

V. SANDHIYA

G. SUNDAR SRI HARI

K. RUBAN

L. NAGARAJ

Phase 5: **Project Demonstration** & **Documentation**

Title: **AI-Powered Health Structural Monitoring**

Abstract:

Structural Health Monitoring (SHM) is a rapidly advancing field focused on the real-time assessment of structural integrity in civil, aerospace, and mechanical systems. By integrating sensors, data acquisition systems, and analytical algorithms, SHM enables continuous monitoring, early damage detection, and performance evaluation of structures. This technology enhances safety, reduces maintenance costs, and extends the lifespan of critical infrastructure such as bridges, buildings, and aircraft. This paper presents an overview of SHM principles, sensor technologies, data interpretation methods, and recent advancements, highlighting its growing significance in ensuring structural reliability and resilience.

1.Project demonstration

Objective:

To demonstrate how sensor-based systems can detect structural changes or damage in real-time.

Components Required:

ESP32 or Arduino microcontroller

Vibration sensor (e.g., piezoelectric sensor or accelerometer like ADXL345)

Strain gauge or flex sensor

Buzzer/LED (for alerts)

Breadboard, jumper wires, and power supply

(Optional) Wi-Fi module for IoT-based monitoring (ESP32 has built-in Wi-Fi)

Working Principle:

1. Sensors are attached to a model structure (e.g., cardboard bridge or beam).
2. Strain/vibration sensors detect any physical stress or abnormal vibrations.
3. The microcontroller collects data from the sensors.
4. If the stress/vibration exceeds a predefined threshold, an alert is triggered via buzzer/LED.
5. (Optional) Data can be sent to a cloud dashboard for remote monitoring.

Demonstration Steps:

1. Introduce your structure model (e.g., a mini bridge).
2. Show how sensors are attached and what they monitor.
3. Apply light force or vibration to the structure and observe normal readings.
4. Apply excessive force to simulate damage — sensors will detect this.
5. Buzzer/LED goes off when critical thresholds are crossed.
6. (Optional) Show the live data on a serial monitor or IoT dashboard.

2.Project Documentation

Overview

Structural Health Monitoring (SHM) is a system designed to assess the condition of structures in real-time using sensors. This project demonstrates a low-cost, sensor-based SHM model that detects stress, strain, and vibrations in a miniature structure (like a bridge), providing early warnings of potential damage. It combines embedded systems and (optionally) IoT for alerting and data monitoring.

Project Details

Components:

ESP32/Arduino Uno microcontroller

Strain gauges, accelerometer, vibration sensors

Buzzer and LEDs for alerts

(Optional) IoT platform (e.g., Blynk/ThingSpeak)

Working Principle:

1. Sensors detect structural changes (bending, shaking, vibrations).
2. The microcontroller processes these signals.
3. If values exceed safety thresholds, visual/auditory alerts are triggered.
4. (Optional) Sensor data is sent to a web dashboard for remote monitoring.

Implementation Steps:

Build a scaled-down model structure.

Integrate sensors at stress-prone areas.

Write microcontroller code for sensor reading and threshold detection.

Test the model under normal and stressed conditions.

Outcome

Successfully monitored structural changes using live sensor data.

Triggered alerts in real time when stress/vibration levels crossed safe limits.

Demonstrated the potential for applying this model in real-world civil engineering and safety systems.

Opened scope for IoT-based data analysis, predictive maintenance, and structural risk assessment.

3.Feedback and Final Adjustments

Overview

After the initial implementation and demonstration of the SHM project, feedback was collected from mentors, peers, or evaluators. This feedback helped identify areas for improvement in terms of system accuracy, alert timing, and presentation clarity. Final adjustments were made to improve performance and reliability before project handover.

Steps Taken

1. **Feedback Collection**

Observers suggested improving sensor calibration for more accurate readings.

Alert mechanisms were recommended to be more responsive and distinguishable.

Suggestions included better data visualization and simplified system explanation.

2. **Technical Adjustments**

Re-calibrated sensors for more accurate threshold detection.

Added delay filters to avoid false alerts caused by brief signal spikes.

Improved code structure for stability and cleaner output.

3.Alert System Enhancements

Adjusted buzzer timing and LED behavior for better visibility and clarity.

Implemented multiple alert levels (e.g., warning vs. critical).

3. **Presentation Refinements**

Clarified demonstration steps for smoother delivery.

Added real-world examples to relate the project to practical applications.

4. **Optional IoT Adjustments**

Improved data update intervals on IoT dashboard.

Optimized Wi-Fi handling for better connectivity.

Outcome

Improved Accuracy: More consistent sensor readings and fewer false alerts.

Better Usability: Clearer alerts and cleaner data display made the system easier to interpret.

Professional Finish: Polished presentation, well-documented logic, and structured final report.

Readiness for Deployment: Project ready for handover or further development.

4.Final Project and Report Submission

1. **Overview**

The Structural Health Monitoring (SHM) project culminates in a final submission that includes a fully functional prototype and a comprehensive written report. The aim is to present a complete record of the project lifecycle—from problem identification to solution development and testing—alongside clear results and future improvement possibilities.

2. **Report Section**

1. **Introduction**

Importance of SHM

Project background and motivation

2. **Objectives**

Key goals of the system (e.g., real-time detection, alerting, data logging)

3. **System Design & Components**

List and explanation of hardware and software used

Block diagram and circuit diagrams

4. **Methodology**

Step-by-step explanation of implementation

Sensor integration, coding, testing process

5. **Results**

Summary of observations during testing

6. **Final Adjustments**

Feedback received

Calibrations and improvements made before finalization

7. **Conclusion**

Key takeaways from the project

Effectiveness of the system

8. **Future Scope**

Potential improvements (e.g., machine learning, GSM, solar integration)

9. **References**

Books, articles, and sources referred to

10. **Appendix**

Full code, schematics, datasheets

3. **Outcomes**

Working Prototype: Successfully demonstrated stress/vibration detection and alerts

Clear Documentation: Well-organized, detailed final report

Improved Design: Adjustments made based on feedback enhanced performance

Presentation Ready: Project is complete and suitable for evaluation or future use

5.Project Handover And Future works

Objective: Ensure a smooth transition of the completed project to the relevant stakeholders, such as mentors, clients, or institution.

Steps for Handover:

1. **Hardware Handover**:

Deliver the fully assembled SHM prototype.

Provide a written setup guide for re-calibration or testing.

Include a list of components and supplies.

2. **Documentation Handover**:

Final Report (Printed/PDF).

Circuit diagrams and schematics.

Source code (USB drive or GitHub repository).

Presentation slides and supporting materials.

3. **Guidelines for Use**:

Explain how to operate the system.

Describe troubleshooting tips for common issues.

Outline maintenance procedures (if needed).

4. **Formal Sign-Off**:

Obtain acknowledgment from the mentor/stakeholder that the project meets the requirements.

Document any remaining issues or considerations.

The project can be extended for greater functionality and real-world deployment:

Possible Enhancements:

1. **IoT Expansion**:

Integrate GSM or LoRa for remote monitoring in areas without Wi-Fi.

Implement cloud-based analytics for predictive maintenance.

2. **Advanced Data Analytics**:

Use machine learning models to predict failures and structural weaknesses.

Integrate historical data to refine alert thresholds.

3. **Energy Efficiency**:

Introduce solar-powered sensors for off-grid applications.

4. **Scalability**:

Scale the system for larger and more complex structures like bridges or high-rise buildings.

5. **Robustness**:

Weatherproofing the device for outdoor deployments.

Using industrial-grade sensors for high-accuracy applications.

Source code:

```
#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_ADXL345_U.h>

#include <WiFi.h>


// Pin Definitions

#define BUZZER_PIN 12

#define LED_PIN 13


// Wi-Fi Credentials (for IoT integration)

Const char* ssid = “your_SSID”;

Const char* password = “your_PASSWORD”;
```

```
// Initialize Accelerometer (ADXL345)
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

// Define Thresholds
Const float vibrationThreshold = 2.5;  // Threshold for vibration (Adjust as needed)
Const int strainThreshold = 1000;     // Arbitrary threshold for strain (for demo)

// Variables
Int strainReading = 0;  // Placeholder for strain sensor reading
Float vibrationReading = 0;  // Placeholder for accelerometer vibration reading

Void setup() {
  Serial.begin(115200);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);

  // Set up Wi-Fi (optional, for IoT integration)
  WiFi.begin(ssid, password);
  While (WiFi.status() != WL_CONNECTED) {
    Delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to Wi-Fi!");

  // Initialize Accelerometer
  If (!accel.begin()) {
```

```
   Serial.println("Couldn't find the sensor.");
   While (1);
  }
  Serial.println("Accelerometer initialized.");
}

Void loop() {
  // Reading the strain (simulated here as a random value)
  strainReading = analogRead(A0);  // Assuming strain sensor is connected to analog pin A0
  Serial.print("Strain Reading: ");
  Serial.println(strainReading);

  // Reading the accelerometer values
  Sensors_event_t event;
  Accel.getEvent(&event);
  vibrationReading = event.acceleration.x * event.acceleration.x +
                event.acceleration.y * event.acceleration.y +
                event.acceleration.z * event.acceleration.z;
  vibrationReading = sqrt(vibrationReading);

  Serial.print("Vibration Magnitude: ");
  Serial.println(vibrationReading);

  // Threshold checks and alert triggers
  If (strainReading > strainThreshold) {
   Serial.println("Strain threshold exceeded!");
```

```
  digitalWrite(LED_PIN, HIGH);  // Turn on LED for alert
  digitalWrite(BUZZER_PIN, HIGH);  // Activate buzzer
  delay(1000);  // Keep alert active for 1 second
  digitalWrite(BUZZER_PIN, LOW);  // Deactivate buzzer
  digitalWrite(LED_PIN, LOW);  // Turn off LED
 }

 If (vibrationReading > vibrationThreshold) {
  Serial.println("Vibration threshold exceeded!");
  digitalWrite(LED_PIN, HIGH);  // Turn on LED for alert
  digitalWrite(BUZZER_PIN, HIGH);  // Activate buzzer
  delay(1000);  // Keep alert active for 1 second
  digitalWrite(BUZZER_PIN, LOW);  // Deactivate buzzer
  digitalWrite(LED_PIN, LOW);  // Turn off LED
 }

 Delay(500);  // Delay between readings
}
```

Program:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <WiFi.h>

// Pin Definitions
#define BUZZER_PIN 12
```

```
#define LED_PIN 13

// Wi-Fi Credentials (for IoT integration)
Const char* ssid = “your_SSID”;
Const char* password = “your_PASSWORD”;

// Initialize Accelerometer (ADXL345)
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

// Define Thresholds
Const float vibrationThreshold = 2.5;  // Threshold for vibration (Adjust as needed)
Const int strainThreshold = 1000;     // Arbitrary threshold for strain (for demo)

// Variables
Int strainReading = 0;  // Placeholder for strain sensor reading
Float vibrationReading = 0;  // Placeholder for accelerometer vibration reading

Void setup() {
  Serial.begin(115200);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);

  // Set up Wi-Fi (optional, for IoT integration)
  WiFi.begin(ssid, password);
  While (WiFi.status() != WL_CONNECTED) {
    Delay(1000);
```

```
  Serial.println("Connecting to WiFi...");
 }
 Serial.println("Connected to Wi-Fi!");

 // Initialize Accelerometer
 If (!accel.begin()) {
  Serial.println("Couldn't find the sensor.");
  While (1);
 }
 Serial.println("Accelerometer initialized.");
}

Void loop() {
 // Reading the strain (simulated here as a random value)
 strainReading = analogRead(A0);  // Assuming strain sensor is connected to analog pin A0
 Serial.print("Strain Reading: ");
 Serial.println(strainReading);

 // Reading the accelerometer values
 Sensors_event_t event;
 Accel.getEvent(&event);
 vibrationReading = event.acceleration.x * event.acceleration.x +
            event.acceleration.y * event.acceleration.y +
            event.acceleration.z * event.acceleration.z;
 vibrationReading = sqrt(vibrationReading);
```

```
 Serial.print("Vibration Magnitude: ");
 Serial.println(vibrationReading);

 // Threshold checks and alert triggers
 If (strainReading > strainThreshold) {
  Serial.println("Strain threshold exceeded!");
  digitalWrite(LED_PIN, HIGH);  // Turn on LED for alert
  digitalWrite(BUZZER_PIN, HIGH);  // Activate buzzer
  delay(1000);  // Keep alert active for 1 second
  digitalWrite(BUZZER_PIN, LOW);  // Deactivate buzzer
  digitalWrite(LED_PIN, LOW);  // Turn off LED
 }

 If (vibrationReading > vibrationThreshold) {
  Serial.println("Vibration threshold exceeded!");
  digitalWrite(LED_PIN, HIGH);  // Turn on LED for alert
  digitalWrite(BUZZER_PIN, HIGH);  // Activate buzzer
  delay(1000);  // Keep alert active for 1 second
  digitalWrite(BUZZER_PIN, LOW);  // Deactivate buzzer
  digitalWrite(LED_PIN, LOW);  // Turn off LED
 }

 Delay(500);  // Delay between readings
}
```

Output:

```
Connecting to WiFi...
```

Connected to Wi-Fi!

Accelerometer initialized.

Strain Reading: 475

Vibration Magnitude: 1.63

Strain Reading: 1120

Vibration Magnitude: 1.78

Strain threshold exceeded!

Strain Reading: 389

Vibration Magnitude: 3.05

Vibration threshold exceeded!

Strain Reading: 1230

Vibration Magnitude: 3.42

Strain threshold exceeded!

Vibration threshold exceeded!