

MineGuard 2040

(smart mining)

Contents

Chapter 1: Introduction	3
Domain:.....	3
Sub-domain	3
Project title	3
Background of study	3
Identified problems:.....	4
Proposed solutions:.....	5
Aim:.....	5
Objectives:	6
Chapter 2:Draw a Flow Chart/process flow diagram :	Error! Bookmark not defined.
List of IoT Components:.....	Error! Bookmark not defined.
Circuit / Schematic Diagram	Error! Bookmark not defined.
Chapter 3: Implementation	18
Chapter 4: Results and findings	35
Results.....	35
Recommendations	Error! Bookmark not defined.
Conclusions	36
Chapter 5: References & Appendices	37
Referencing is required:	Error! Bookmark not defined.
Questionnaire (if applicable)	Error! Bookmark not defined.

Chapter 1: Introduction

Domain:

Government (Mining)

Sub-domain:

Safety

Project title

MineGuard 2040

Background of study

The selected domain for this project is the government, focusing on safety in Oman's mining industry. As a key sector supporting Oman's economic diversification under Vision 2040, mining presents significant risks to workers, equipment, and the environment. The Omani government is crucial in enforcing safety regulations to protect workers' well-being and minimize environmental impact.

Existing methods and technologies for mining safety, such as IoT-based monitoring systems, have made progress but face limitations. The Omani government enforces safety regulations to protect workers and minimize environmental impact. Existing solutions, like IoT-based smart helmets and environmental monitors, address issues like gas detection and temperature monitoring but lack comprehensive features such as accurate fatigue monitoring, machine health tracking, and dust pollution control. Additionally, many rely on less accurate sensors, limiting their effectiveness. These gaps underscore the need for a more advanced and integrated solution like MineGuard 2040 to strengthen mining safety in Oman. The following is the background study carried out to understand the context and challenges of mining safety.

Worker safety in mining is a critical concern due to exposure to toxic gases, physical fatigue, and high temperatures. Existing systems, such as the one suggested by Suriyakrishnaan et al. (2021)[1], include collision detection and gas sensors for detecting toxic gases released during mining. Another solution by V. Parkavi.(2023)[2], measures heart rate and SpO2 levels using infrared sensors.

Monitoring the health of mining machinery is vital to ensure operational efficiency and prevent accidents. Artificial intelligence (AI) is increasingly being utilized for vibration detection in predictive maintenance systems, as explored in the article "*Vibration Analysis for IoT Enabled Predictive Maintenance*" [3], providing valuable insights into machine performance.

Dust pollution is a significant issue in mining, affecting both workers and local communities. Despite its impact, the majority of the existing systems lack a dust monitoring system. For example, the system suggested by Suriyakrishnaan et al. (2021)[1] does not address dust pollution; instead, it concentrates on gas detection and collision prevention. However, a recent 2024 scientific report[4] has studied dust pollution in ball clay surface mines, highlighting the challenges of dust generation in mining environments, particularly in dry and dusty conditions.

Identified problems:

Mining is a key sector in Oman's economy, but it faces challenges like harsh working conditions, dust pollution, and toxic gases, impacting worker safety and environmental sustainability. The following section outlines the major issues identified through the background study, emphasizing the need for innovative solutions like MineGuard 2040.

Problem 1: Worker Safety

In Oman, miners are subjected to hazardous conditions like as extreme temperatures, physical exhaustion, and toxic gases (such as carbon monoxide and methane). These circumstances place miners at serious risk and are frequently found while mining non-coal minerals, including copper, gypsum, and limestone. Existing safety systems often fail to monitor worker fatigue accurately, relying on less reliable IR sensors. This can lead to undetected health issues and accidents, putting workers' lives at risk.

Problem 2: Machine Health

Due to harsh weather and working conditions, mining machinery in Oman experiences overloading and overheating issues. While current systems use AI-based vibration analysis for fault detection, they do not monitor overloading or overheating. This gap leads to frequent breakdowns, increased downtime, and higher maintenance costs for the mining machines.

Problem 3: Environmental Monitoring

In Oman, dust monitoring is essential because airborne particulate matter in mining sites poses respiratory health risks to residents and workers. Oman's desert climate makes this worse. The scientific report from the background study is not particular to the many mining industries in Oman, such as copper, gypsum, or other minerals; rather, it focuses on dust monitoring in ball clay mining.

Proposed solutions:

Based on the challenges identified in the background section, the following solutions are proposed to enhance machine health, workers' safety, and environmental monitoring. In order to increase mining operations' efficiency and safety, these solutions concentrate on automatic alerts, real-time data collection, and preventive measures.

Solution to Problem 1: Worker Safety

Wearable devices, like a smart helmet or wristband integrated with a pulse sensor instead of an IR sensor to monitor heart rate to detect fatigue in real time, also include a gas sensor for toxic gas monitoring. If high fatigue or gas level is detected, an LED indicator and a buzzer will be turned on to alert the worker.

Solution to Problem 2: Machine Health

Integrating load and temperature sensors into mining machinery to avoid overloading and overheating, respectively. If the load or temperature exceeds set limits, a buzzer or LED triggers an alert, helping prevent damage and ensuring safe operation.

Solution to Problem 3: Dust Pollution

Install dust sensors (such as the GP2Y1010AU0F) to measure the dust level in the air in real time. If dust levels exceed safe limits, the system will trigger alarms or activate dust suppression systems (e.g., water pump), protecting workers and local communities.

Aim:

MineGuard 2040 is an IoT-based system that enhances mining safety and efficiency by monitoring workers' health, machine performance, and environmental conditions, aligning with Oman Vision 2040 with a focus on sustainability and smart technology.

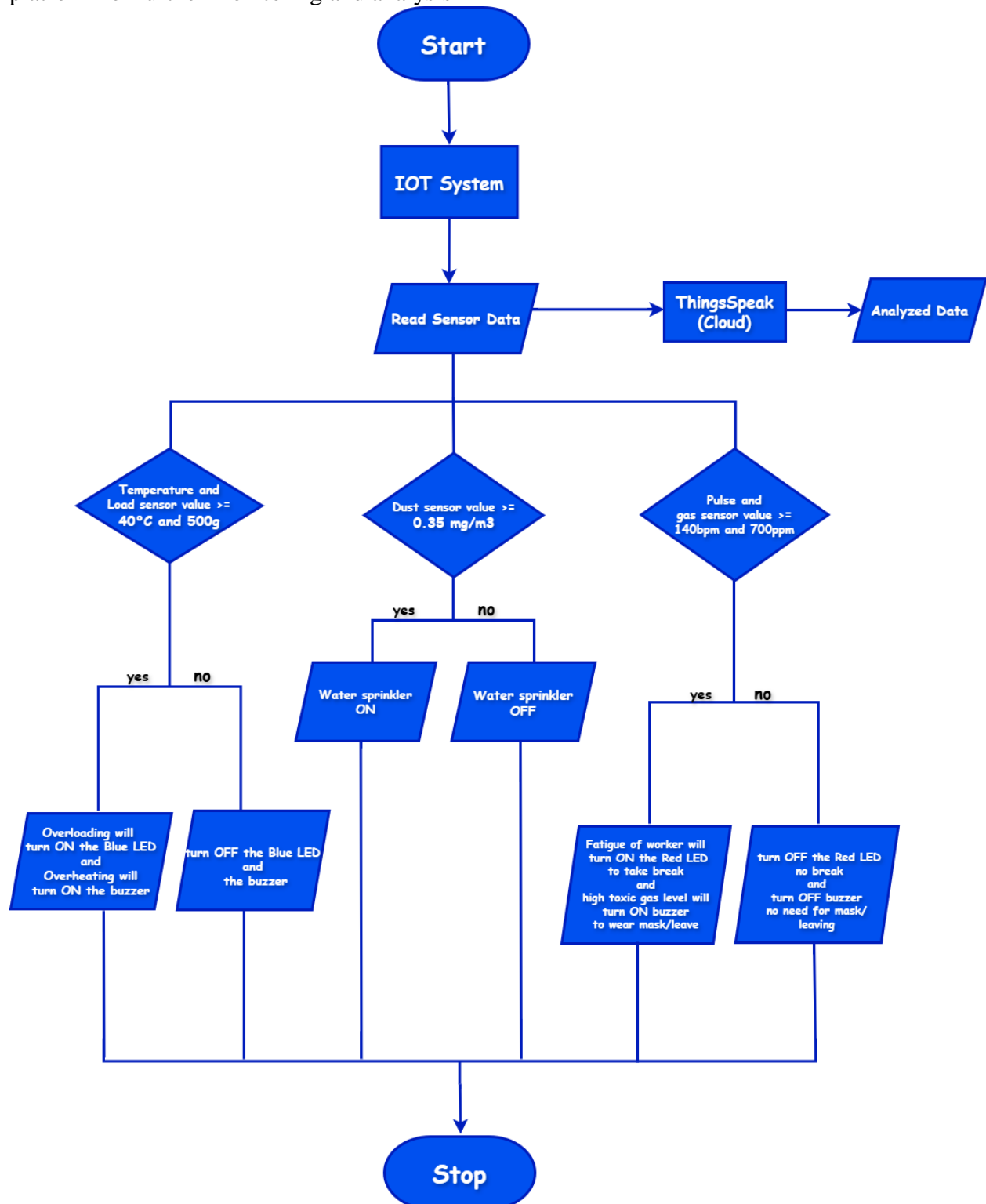
Objectives:

The primary objective of *MineGuard 2040* is to enhance miners' safety, monitor environmental conditions, and machine health using IoT-based solutions. To achieve this:

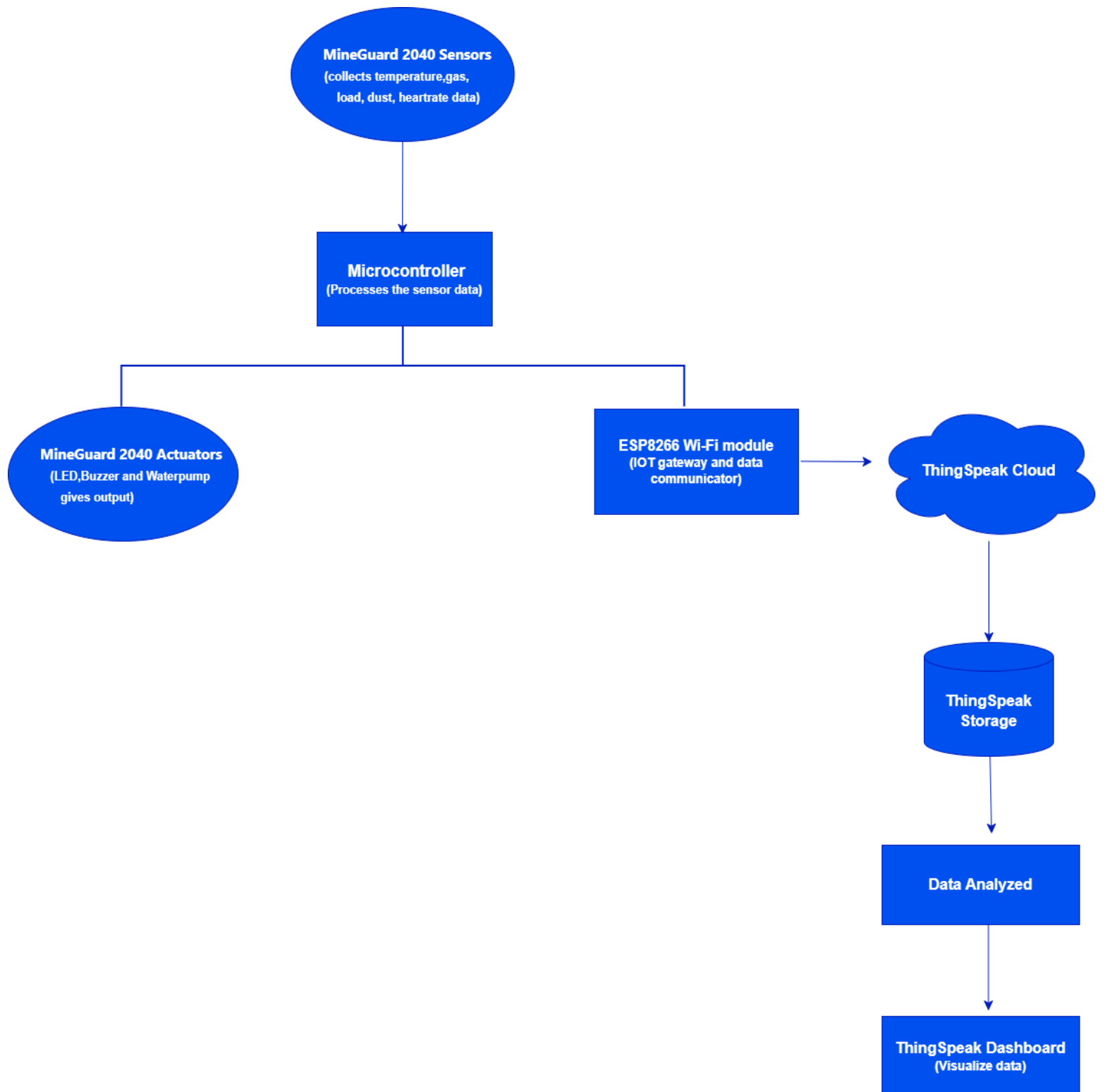
- A pulse sensor monitors workers' heart rates and detects signs of fatigue.
- An MQ-7 gas sensor detects carbon monoxide levels to ensure air quality and safety from toxic gases.
- A load cell measures the weight carried by machines, helping to prevent overloading of machines.
- A temperature sensor (DHT11) monitors the machine temperature to prevent overheating.
- A dust sensor (GP2Y1010AU0F) evaluates air quality and activates a water pump via relay if dust concentration exceeds safe limits
- Connecting the designed prototype with cloud platforms, such as ThingSpeak, for transmitting data, allowing for monitoring and further analysis.
- Testing and Documentation- Test the prototype designed and document the results.

Flow chart

The following flowchart illustrates the process of the MineGuard 2040 system, showing how data is collected from different sensors, processed, and transmitted to the ThingSpeak cloud platform for further monitoring and analysis.



Fig_1: Flow chart 1



Fig_2: Flow Chart 2

Process Flow Diagram:

The Process Flow diagram below outlines the processes of MineGuard 2040, from the collection and processing of sensor data to the activation of actuators for output.

Machine Health Monitoring Flow Process

Overheating

The DHT11 sensor installed in the mining equipment will continuously monitor its temperature. If the temperature exceeds the predefined limit, the system will trigger an alert using a buzzer to notify workers of the overheating condition, allowing them to take necessary actions.



Fig_3: Overheating process flow

Overloading

The load cell will measure the weight or force applied on the mining equipment, and the HX711 amplifier will process the signal for accurate readings. If the detected load exceeds the predefined safety limit, the system will trigger an alert using an LED to warn workers of the overload condition, allowing them to unload the machine to prevent damage or failure.



Fig_4: Overloading process flow

Workers' Health Monitoring Flow Process

Fatigue detection

For fatigue detection, a pulse sensor continuously monitors the worker's heart rate. The sensor detects abnormal heart rate, which can indicate signs of fatigue, notifying the workers using an LED and displaying it through an LCD to take a break.



Fig_5: Fatigue detection process flow

Toxic gas detection:

For toxic gas detection, the MQ2 sensor continuously monitors the environment for gases like methane and carbon monoxide. If the concentration exceeds a safety threshold, the sensor triggers an alarm warning the worker to wear a gas mask. Since exposure to high levels of these gases can cause respiratory problems, lung damage, headaches, and poisoning.



Fig_6: Toxic gas process flow

Environment Monitoring Flow Process

Dust detection

The Sharp GP2Y1010AU0F dust sensor detects dust concentration in the air. If the dust concentration exceeds a safe level, the water pump will be activated to release water, settling the dust particles and improving air quality to minimize respiratory problems for locals and workers.


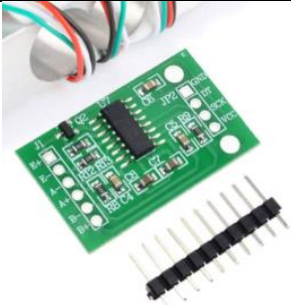





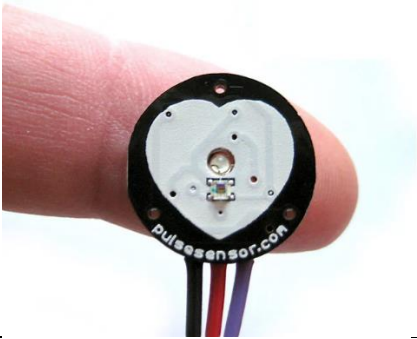



Fig_7: Dust detection process flow




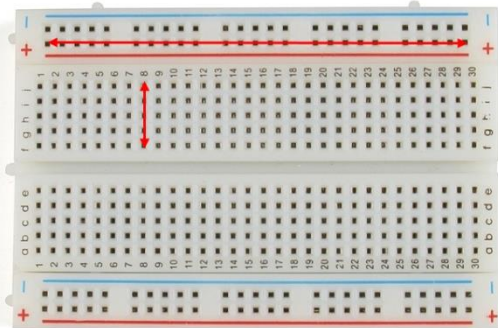
List of IoT components:

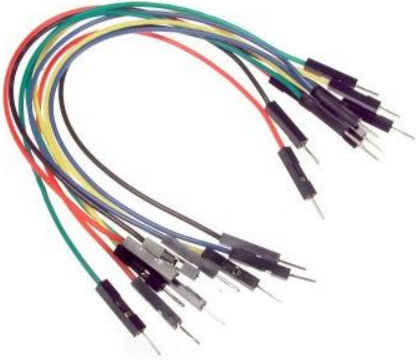
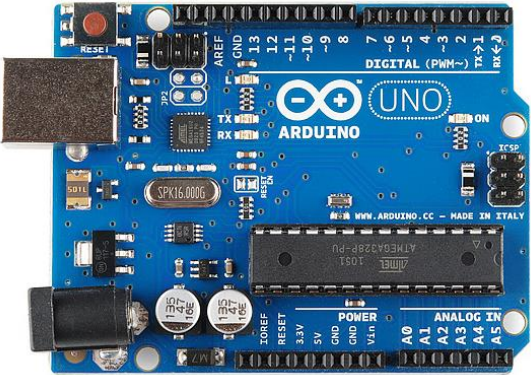

Below are the hardware and software components required for the MineGuard 2040 project, for efficient monitoring, processing, and data transmission within the mining environment to improve mine safety.



Hardware Components:

<i>S.No.</i>	<i>Name of the Component</i>	<i>Picture</i>	<i>Description</i>
1.	Load Cell Sensor		Measures the load on mining equipment.
2.	Hx711		Used with the load cell to amplify the load cell signal.
3.	Submersible Water Pump		Activated via relay module when dust levels are too high to release water.
4.	Water Pipe		It is connected to the water pump to release water automatically when the dust concentration is high to prevent the dust.



5.	DHT11		Measures temperature level in the mining environment for detecting overheating of mining equipment.
6.	Pulse Sensor		Measures the worker's heart rate for fatigue detection.
7.	Sharp GP2Y1010AU0F Dust Sensor		Detects dust concentration in the mining areas.
8.	MQ-7 Gas Sensor		Detect toxic gases in the mining area.
9.	Buzzer		Provides an audio alert when the gas and temperature levels exceed safe limits.

10.	Relay Module (water pump control)		Acts as a switch to turn the water pump ON/OFF.
11.	LED		Works as a warning when overloading of machine and fatigue of worker detected.
12.	Battery (power supply for water pump)		Provides independent power to the water to avoid overloading the Arduino.
13.	Breadboard		To connect multiple components.

14.	Jumper wires		To connect sensors and actuators to the NodeMCU and Arduino
15.	Arduino Uno		Used for processing and powering the dust and load sensor, as it requires a minimum of 5V.
16.	NodeMCU		Used for processing data from other sensors and wireless communication to send data to a cloud platform.

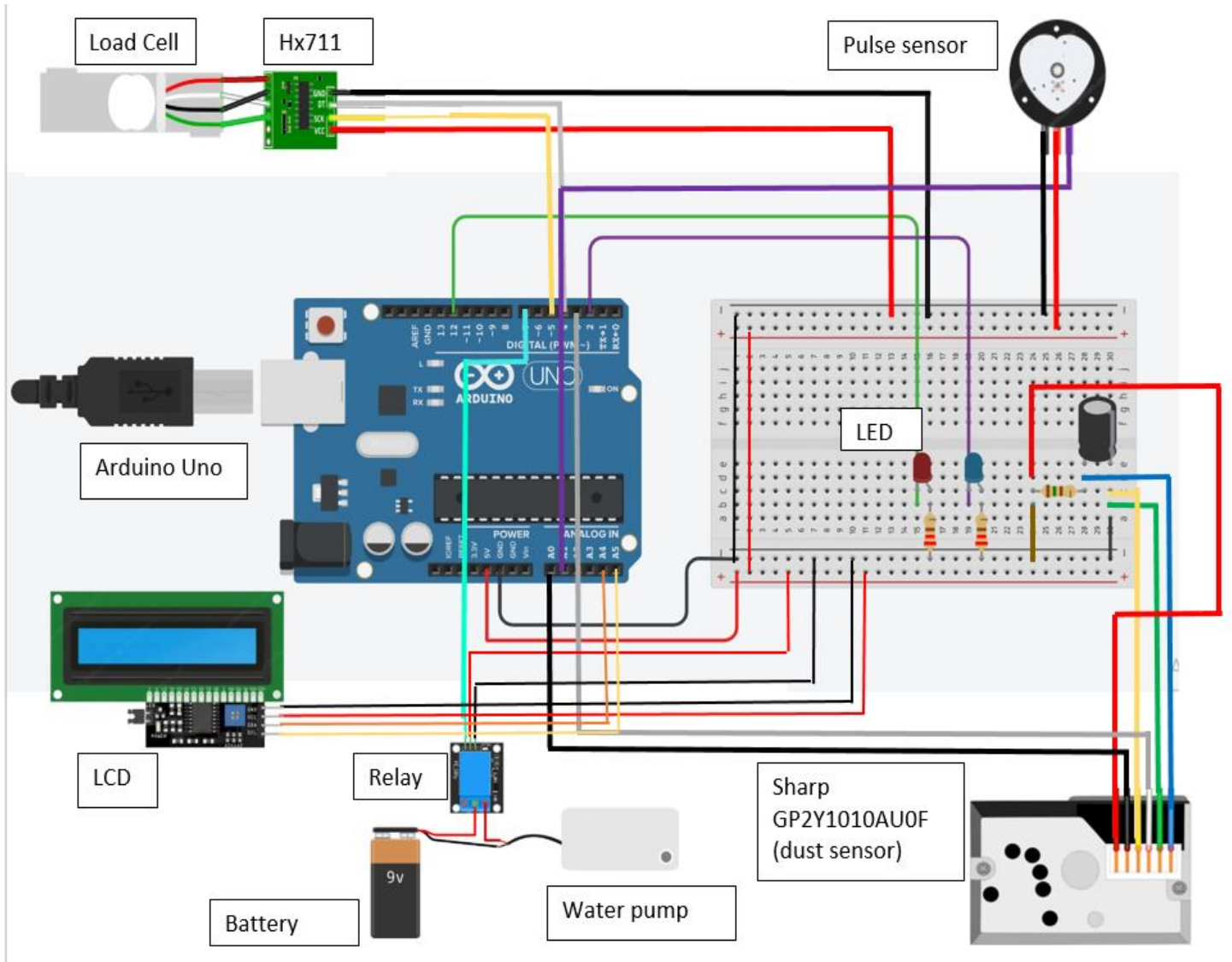
17.	LCD(Liquid Crystal Display)		It is used to display the sensor data.
18.	Resistors		Resistors are electrical components that limit the flow of current to certain components in the circuit.

Software components:

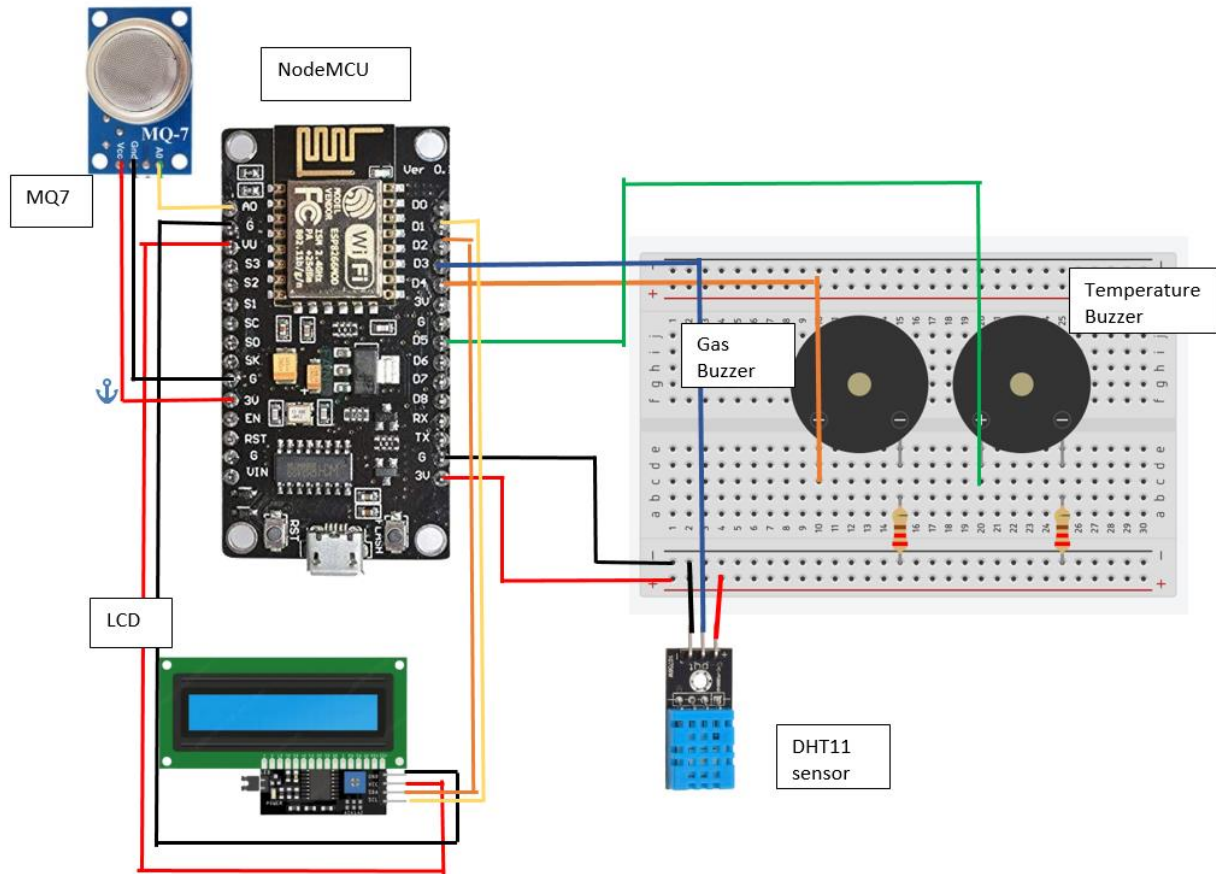
<i>S.No.</i>	<i>Name of the Component</i>	<i>Picture</i>	<i>Description</i>
1.	Arduino IDE		The Arduino IDE will be used for programming and uploading code to the Arduino board and NodeMCU.
2.	ThingSpeak		The sensor data will be transmitted to ThingSpeak(Cloud) for monitoring and further analysis..

Circuit Diagram

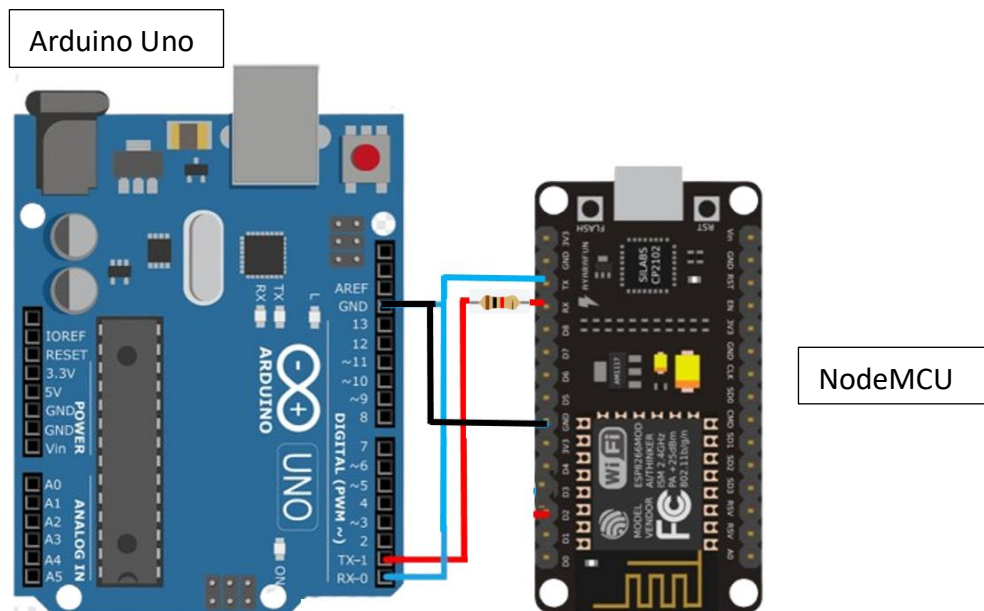
The circuit diagram highlights the connections between various components, including sensors (pulse, gas, load, temperature, dust), actuators (LED, buzzer, water pump), and the microcontroller.



Fig_8: Circuit Diagram



Fig_9: Circuit Diagram



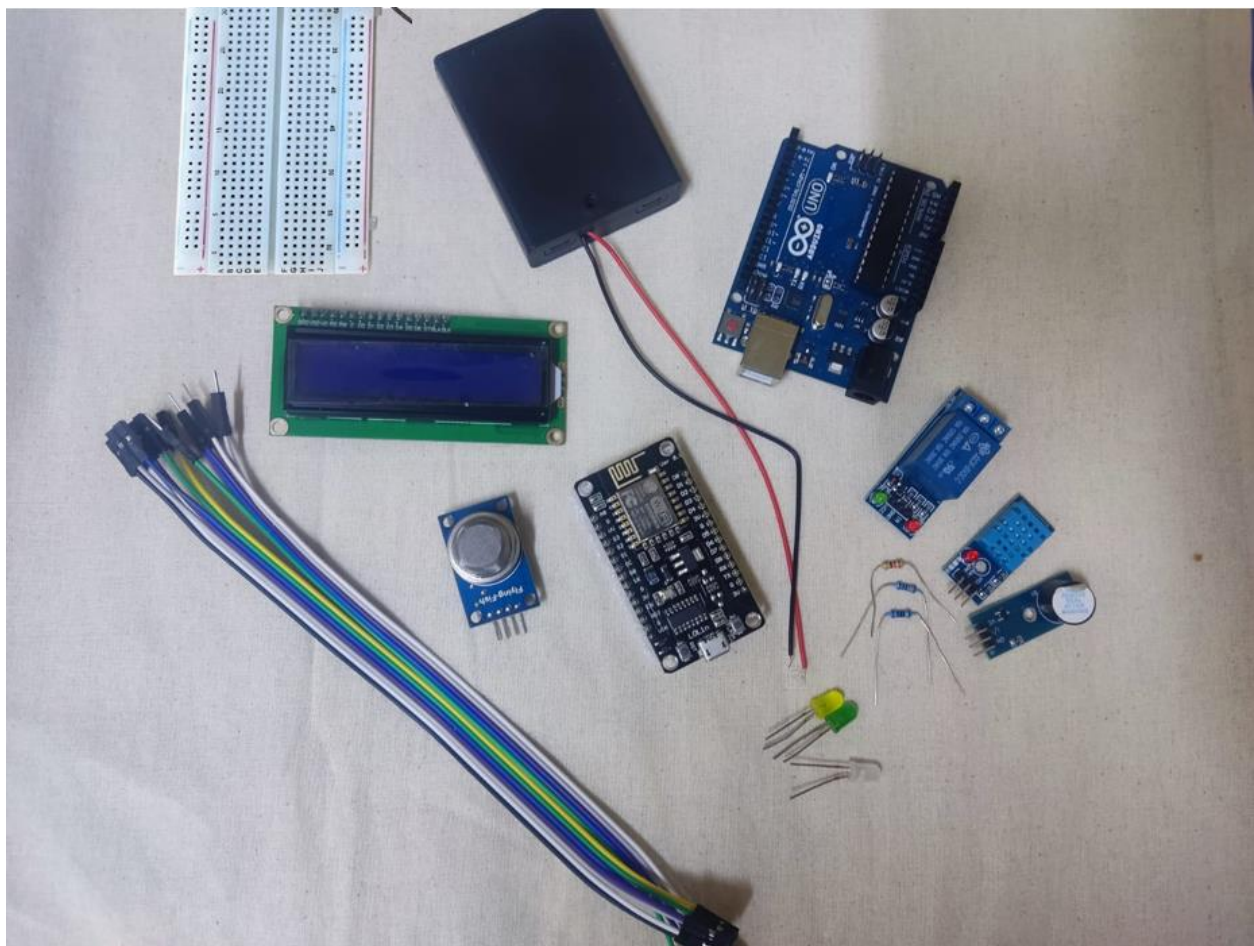
Fig_10: Circuit Diagram

Chapter 3: Implementation

This section outlines the step-by-step implementation of the MineGuard 2040 prototype. We aimed to connect all hardware components correctly, sensor calibration, write the code, upload it to the Thingspeak cloud, and make everything work together to detect and monitor real-time mining safety hazards.

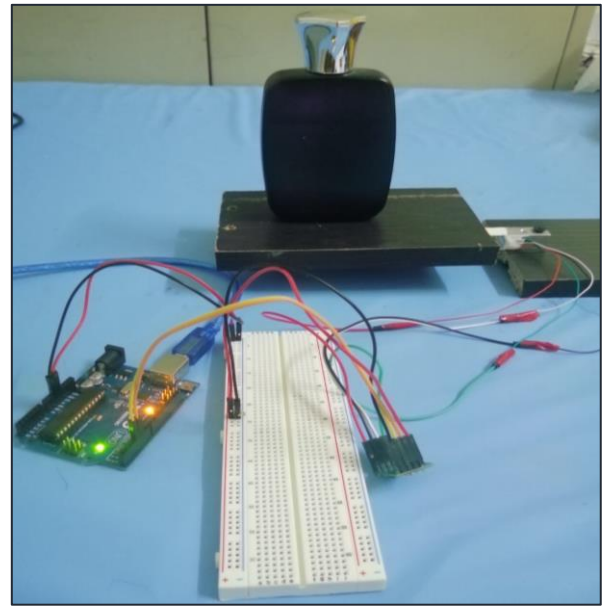
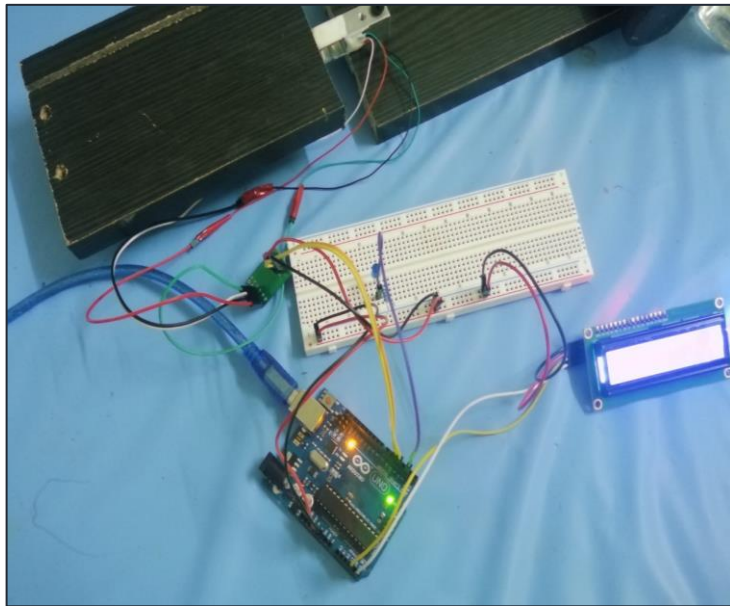
Step 1: Collecting the raw materials

We started by gathering all the components we had listed earlier, such as the gas sensor, dust sensor, pulse sensor, temperature sensor, load sensor, LCD, buzzers, water pump, and of course, the Arduino Uno and NodeMCU, during the first week of prototype implementation.

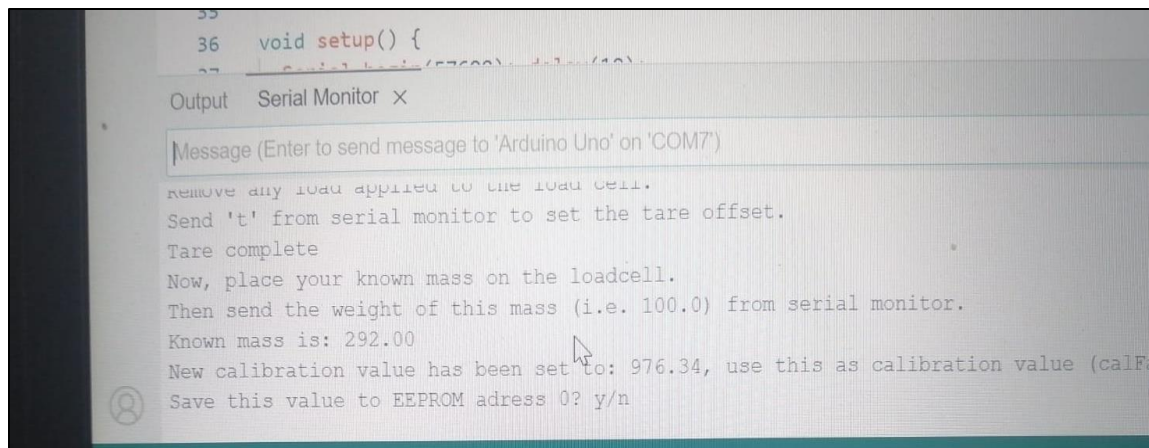


Step 2: Setting up the load sensor

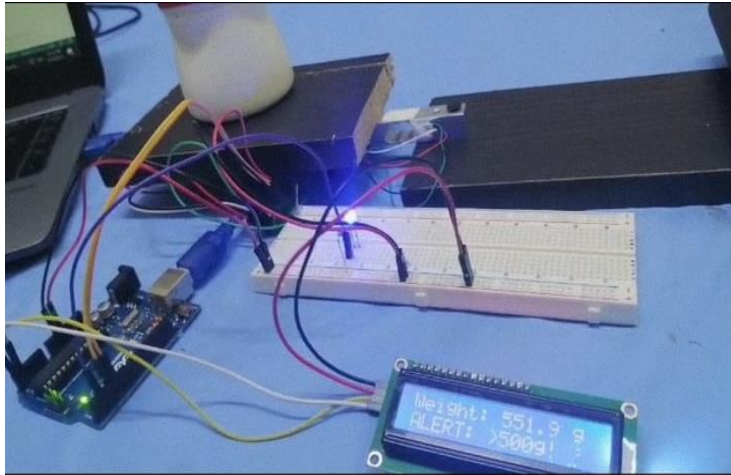
We began by connecting the load cell to the HX711 amplifier module. The HX711 was then wired to the Arduino Uno as follows: DT to D2, SCK to D3, VCC to 5V, and GND to GND.



For calibration, we uploaded a default calibration sketch of the HX711_ADC library to the Arduino and placed known weights of 292g on the load cell. By adjusting the calibration factor and observing the readings through the serial monitor, we successfully calibrated the system and determined the calibration value to be **976**.



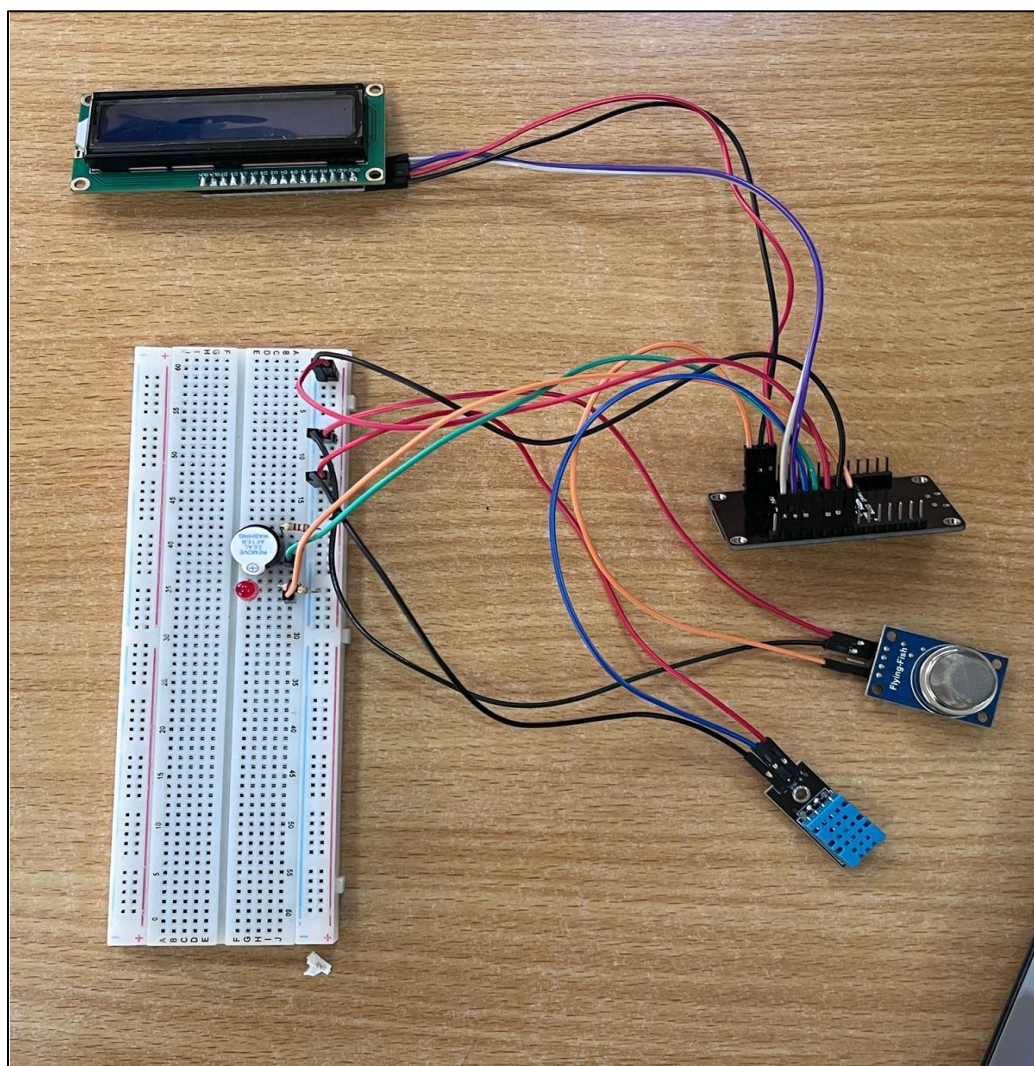
The load sensor sends weight data to the HX711 amplifier, which amplifies the signal and passes it to the **Arduino**. If the measured weight exceeds **500 grams**, a blue LED lights up, and the LCD displays an alert message "**ALERT: >500g**" to indicate an overload, signaling the miner to unload the machine to avoid potential damage.



Step 3 : Setting up MQ-7 gas and DHT11 temperature sensor

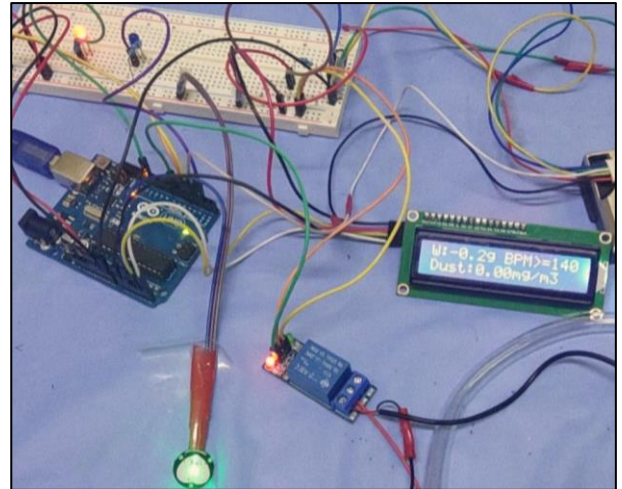
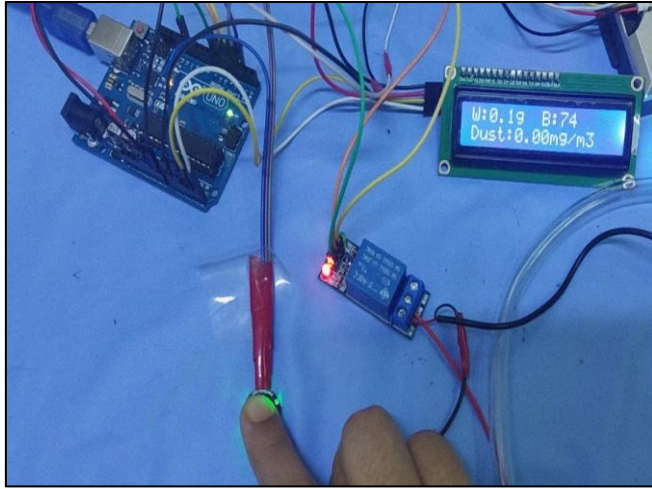
The MQ-7 gas sensor and DHT11 temperature sensor were both connected to the 5V and GND pins of the NodeMCU. The MQ-7 sensor's analog output was connected to an analog pin (A0), while the DHT11 data pin was connected to digital pin D3. The sensor readings for carbon monoxide (CO) concentration and temperature are continuously displayed on the LCD. If the CO level exceeds 700 ppm or the temperature rises above 40°C, the buzzer sounds to alert the miner.

In actual mining conditions, temperatures can reach up to 80°C, and CO concentrations above 400–800 ppm are considered dangerous, with levels above 1200 ppm being highly hazardous. This system provides early warnings to miners to help prevent exposure to toxic gases.



Step 4: Setting up the pulse sensor

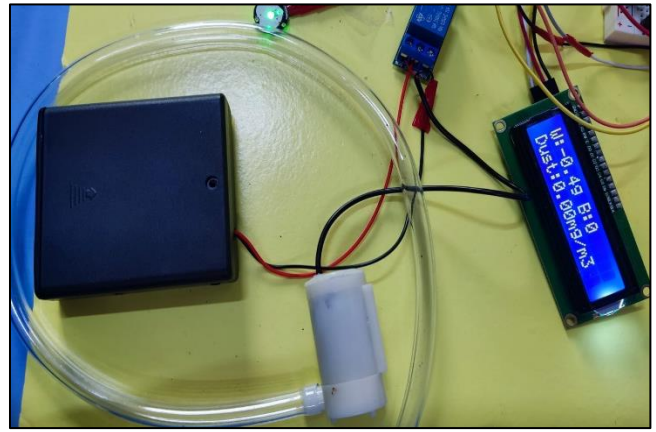
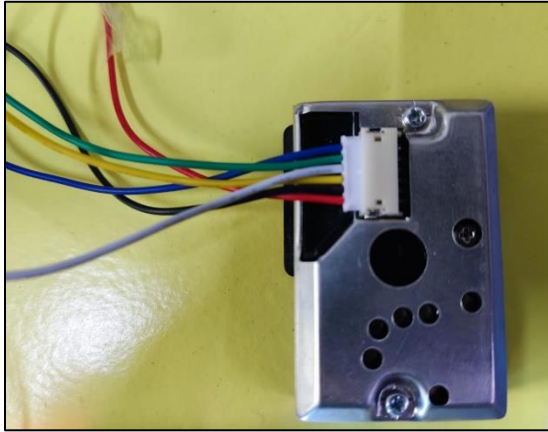
The pulse sensor is connected to the Arduino's analog pin A1, with VCC and GND properly connected. It measures the heart rate and displays the BPM (beats per minute) readings on the LCD. If the BPM exceeds 140, a red LED lights up, indicating possible fatigue signalling the miners to rest.



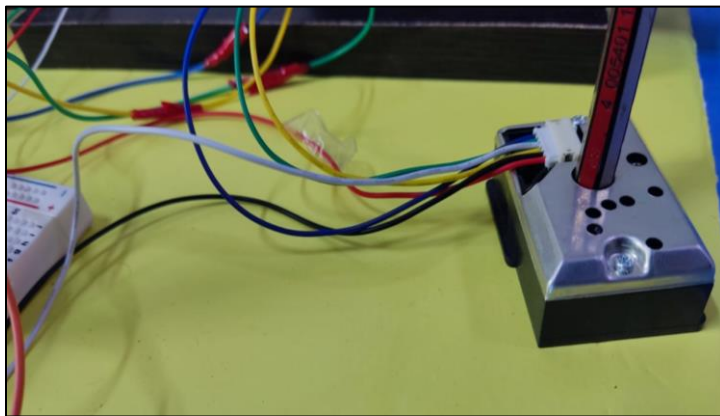
Step 5: Setting up the Sharp GP2Y1010AU0F Optical Dust Sensor

The dust sensor is connected to the Arduino through analog pin A0 to measure dust concentration in the air. A capacitor is used to stabilize the sensor's signal output for more accurate readings. If the dust concentration exceeds **0.35 mg/m³**, the Arduino sends a signal through digital pin 7 to activate a relay, which powers a water pump connected to a 6V battery. The water is used to settle airborne dust and reduce exposure. The average dust concentration is continuously displayed on the LCD; if the value is high, the display shows "**High Dust**" along with the current reading.

In a study conducted at a [Zambian copper mine \[5\]](#), respirable dust exposure ranged between 0.02 mg/m³ and 18.0 mg/m³, with a geometric mean of 0.44 mg/m³, highlighting the importance of early dust control. This system helps detect and respond when dust levels become hazardous.

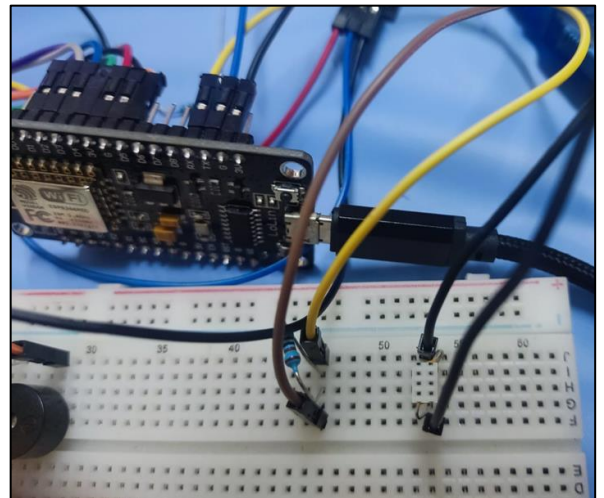
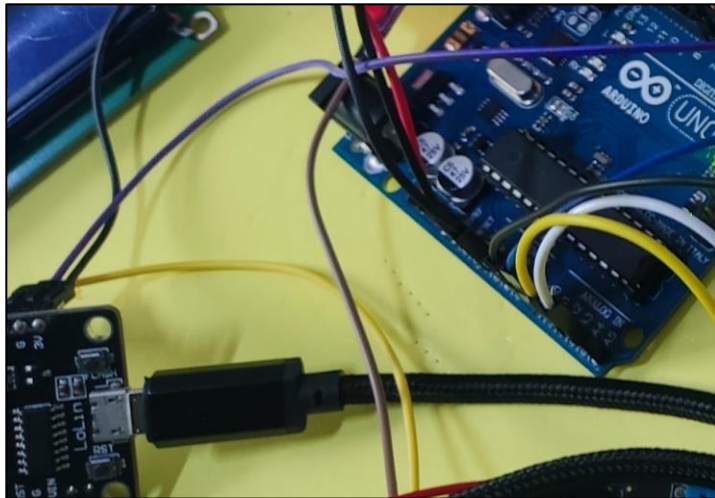


To simulate high dust levels during testing, we inserted a pencil into the dust sensor. Since the sensor operates using an optical method—detecting particles that scatter a light beam—the pencil blocks the light path, which the sensor interprets as a high dust concentration. This allowed us to verify the system's response to high dust levels and ensure the relay, water pump, and LCD alert functions were working correctly.



Step 5: Connecting to the Thingspeak cloud

To enable cloud connectivity, the NodeMCU is used to upload all sensor data to ThingSpeak. The gas (MQ-7) and temperature (DHT11) sensors are connected directly to the NodeMCU. Meanwhile, the Arduino Uno handles data from the dust sensor, pulse sensor, and load cell, and sends it to the NodeMCU. This is done by connecting the TX pin of the Arduino to the RX pin of the NodeMCU to transmit the data. A 1000-ohm resistor is used in series to reduce the voltage for safe input to the NodeMCU as it works with 3.3V. All the data is uploaded to the same ThingSpeak channel using a shared API key, allowing real-time monitoring of environmental and health parameters on a single platform.



Arduino code

```
#include <HX711.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <PulseSensorPlayground.h>
#include <GP2YDustSensor.h>

// ===== Load Cell & Pulse Sensor Setup (Original Code) =====
// HX711 Pins
#define LOADCELL_DOUT_PIN 4
#define LOADCELL_SCK_PIN 5

// LCD Configuration (I2C Address: 0x27, 16x2)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// LED Pin (for load cell alert)
#define LED_PIN 2

// Calibration Factor (Adjust based on known weight)
float calibration_factor = 976.0;

HX711 scale;

// PulseSensor Setup
const int PulseWire = A1;      // PulseSensor PURPLE wire on Analog Pin 0
const int PulseLED = LED_BUILTIN; // Built-in LED (usually pin 13)
const int BPM_LED = 12;       // External LED on Pin 12 (for BPM > 140)
int Threshold = 519;          // Adjust if needed (default: 520)

PulseSensorPlayground pulseSensor; // PulseSensor object

// ===== Dust Sensor Setup (Original Code) =====
const uint8_t SHARP_LED_PIN = 3; // LED control pin
const uint8_t SHARP_VO_PIN = A0; // Analog pin for sensor output
const uint8_t RELAY_PIN = 7;     // Relay control pin

GP2YDustSensor dustSensor(GP2YDustSensorType::GP2Y1014AU0F, SHARP_LED_PIN,
SHARP_VO_PIN, 10);

// ===== Setup Function =====
void setup() {
    Serial.begin(115200);

    // Initialize LCD
    lcd.init();
    lcd.backlight();
    lcd.clear();
```

```

lcd.setCursor(0, 0);
lcd.print("Welcome to");
lcd.setCursor(0, 1);
lcd.print("\\"MineGuard 2040\\"");
delay(1000);
lcd.clear(); // Initialize LED (for load cell)
pinMode(LED_PIN, OUTPUT);
digitalWrite(LED_PIN, LOW);

// Initialize Load Cell
scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
scale.set_scale(calibration_factor);

// Tare the scale (remove any initial offset)
lcd.setCursor(0, 1);
lcd.print("Remove Weight...");
delay(2000); // Wait 2 sec to ensure no weight is placed
scale.tare(); // Reset to zero
lcd.clear();
lcd.setCursor(0, 1);
lcd.print("Ready to Measure");
delay(1000);
lcd.clear();

// Initialize PulseSensor
pulseSensor.analogInput(PulseWire);
pulseSensor.blinkOnPulse(PulseLED); // Blink built-in LED with heartbeat
pulseSensor.setThreshold(Threshold);
pinMode(BPM_LED, OUTPUT); // LED for high BPM
digitalWrite(BPM_LED, LOW); // Start with LED off

if (pulseSensor.begin()) {

}

// Initialize Dust Sensor (Original Code)
dustSensor.setBaseline(0.95); // Adjust based on your "no dust" readings
dustSensor.begin();

pinMode(RELAY_PIN, OUTPUT);
digitalWrite(RELAY_PIN, HIGH); // Make sure relay is OFF initially
}

// ===== Main Loop =====
void loop() {
  // --- Load Cell Section (UNCHANGED) ---
  float weight = scale.get_units(3); // Read weight (average of 3 readings)

```

```

lcd.setCursor(0, 0);
lcd.print("W:");
lcd.print(weight, 1);
lcd.print("g ");

if (weight > 500.0) {
    digitalWrite(LED_PIN, HIGH); // Turn on LED
    lcd.setCursor(0, 1);
    lcd.print("ALERT: >500g!");
} else {
    digitalWrite(LED_PIN, LOW); // Turn off LED
    // Don't clear this line completely as we might display other info
}

static int myBPM = 0; // Declare outside the if block and retain last value
// --- Pulse Sensor Section (UNCHANGED) ---
myBPM = pulseSensor.getBeatsPerMinute();

// Control BPM LED (Pin 12)
if (myBPM > 140) {
    digitalWrite(BPM_LED, HIGH);
    lcd.setCursor(8, 0);
    lcd.print("REST ");
} else {
    digitalWrite(BPM_LED, LOW);
    lcd.setCursor(8, 0);
    lcd.print("B:");
    lcd.print(myBPM);
    lcd.print(" "); // Clear remaining chars
}

// --- Dust Sensor Section (UNCHANGED) ---
float dust_ugm3 = dustSensor.getDustDensity();
float average_ugm3 = dustSensor.getRunningAverage();

float dust_mgm3 = dust_ugm3 / 1000.0; // Convert to mg/m³
float average_mgm3 = average_ugm3 / 1000.0; // Convert to mg/m³

// Check and control the relay
if (dust_mgm3 > 0.35) {
    digitalWrite(RELAY_PIN, LOW); // Turn ON the relay (pump ON)
} else {
    digitalWrite(RELAY_PIN, HIGH); // Turn OFF the relay (pump OFF)
}

// Display dust info on LCD second line when not showing weight alert
if (weight <= 500.0) {
    lcd.setCursor(0, 1);

```

```

    if (dust_mgm3 > 0.35) {
        lcd.print("High dust:");
        lcd.print(average_mgm3, 2); // Pad with spaces to clear old text
    } else {
        lcd.print("Dust:");
        lcd.print(average_mgm3, 2);
        lcd.print("mg/m3   "); // Clear remaining characters
    }
}

// Serial data
Serial.print(myBPM);
Serial.print(",");
Serial.print(weight, 1);
Serial.print(",");
Serial.print(average_mgm3, 2);
Serial.println();

delay(2000); // Maintain original timing from pulse/load cell code
}

```

NodeMCU code

```
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <ESP8266WiFi.h>

//thingspeak connection
String apiKey = "thingspeak_apikey";    // Enter your Write API key from
ThingSpeak

const char *ssid = "ssid";    // replace with your wifi ssid and wpa2 key
const char *pass = "wifi_paasword";
const char* server = "api.thingspeak.com";

// Define pins
#define DHTPIN D3    // DHT11 data pin connected to NodeMCU D3
#define MQ7PIN A0    // MQ7 analog output connected to NodeMCU A0
#define TEMP_BUZZ D4    // LED for temperature warning
#define GAS_BUZZ D5    // LED for gas warning

// Define thresholds
#define TEMP_THRESHOLD 40.0    // Temperature threshold in Celsius
#define GAS_THRESHOLD 700    // Gas concentration threshold (adjust based on
calibration)

// Initialize DHT sensor
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

WiFiClient client;

void setup() {
    // Initialize serial communication
    Serial.begin(115200);
    // Turn off BUZZER INITIALLY initially

    // Initialize the LCD
    lcd.init();
    lcd.backlight();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Initializing...");
```

```

// Initialize DHT sensor
dht.begin();

// Set pin modes
pinMode(TEMP_BUZZ, OUTPUT);
pinMode(GAS_BUZZ, OUTPUT);
digitalWrite(TEMP_BUZZ, LOW);
digitalWrite(GAS_BUZZ, LOW);


//connecting to wifi
Serial.println("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, pass);

while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

delay(2000); // Give sensors time to initialize
lcd.clear();
}

void loop() {
    float heartRate = 0.0;
    float weight = 0.0;
    float dust = 0.0;

    if (Serial.available()) {
        String incomingData = Serial.readStringUntil('\n');
        incomingData.trim(); // Remove \r or whitespace

        Serial.println("From Arduino: " + incomingData);

        int firstComma = incomingData.indexOf(',');
        int secondComma = incomingData.indexOf(',', firstComma + 1);

        if (firstComma > 0 && secondComma > firstComma) {
            String hrStr = incomingData.substring(0, firstComma);
            String wtStr = incomingData.substring(firstComma + 1, secondComma);
            String dustStr = incomingData.substring(secondComma + 1);

            heartRate = hrStr.toFloat();

```

```

        weight = wtStr.toFloat();
        dust = dustStr.toFloat();
    }
}

// Read sensors
float temperature = dht.readTemperature();
int gasValue = analogRead(MQ7PIN);
int gasPPM = map(gasValue, 0, 1023, 0, 1000);

// Sensor error check
if (isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("DHT Error");
    delay(2000);
    return;
}

// LCD Display
lcd.clear();
lcd.setCursor(0, 0);
if (temperature > TEMP_THRESHOLD) {
    lcd.print("Overheating");
} else if (gasPPM > GAS_THRESHOLD) {
    lcd.print("Toxic gas");
    lcd.setCursor(0, 1);
    lcd.print("detected");
} else {
    lcd.print("Temp: ");
    lcd.print(temperature, 1);
    lcd.print(" C");

    lcd.setCursor(0, 1);
    lcd.print("Gas: ");
    lcd.print(gasPPM);
    lcd.print(" ppm");
}

// ThingSpeak Upload
if (client.connect(server, 80)) {
    String postStr = apiKey;
    postStr += "&field1=" + String(temperature);
    postStr += "&field2=" + String(weight);
    postStr += "&field3=" + String(gasPPM);
}

```



```

    postStr += "&field4=" + String(heartRate);
    postStr += "&field5=" + String(dust);
    postStr += "\r\n\r\n";

    client.println("POST /update HTTP/1.1");
    client.println("Host: api.thingspeak.com");
    client.println("Connection: close");
    client.println("X-THINGSPEAKAPIKEY: " + apiKey);
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.println("Content-Length: " + String(postStr.length()));
    client.println();
    client.print(postStr);

    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.print("°C, Gas: ");
    Serial.print(gasValue);
    Serial.println(". Sent to ThingSpeak.");
}
client.stop();

// Control LED and buzzer based on thresholds
if (temperature > TEMP_THRESHOLD) {
    digitalWrite(TEMP_BUZZ, HIGH);
} else {
    digitalWrite(TEMP_BUZZ, LOW);
}

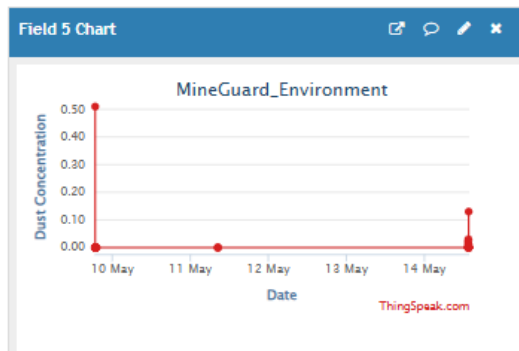
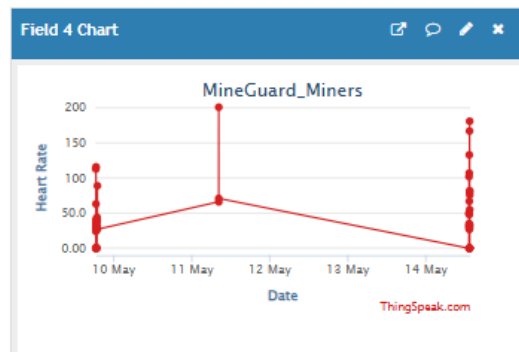
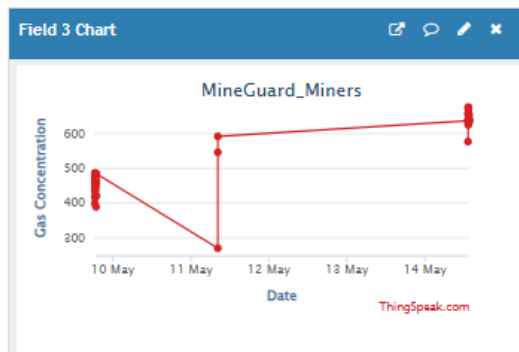
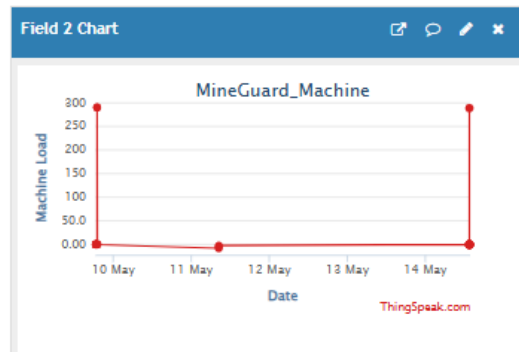
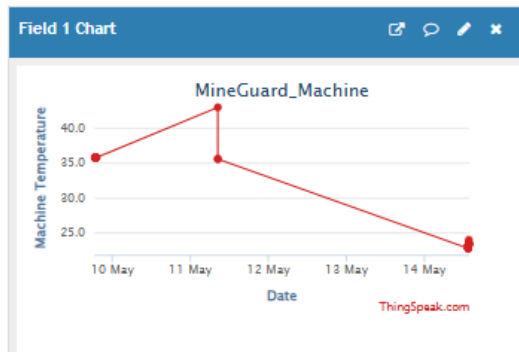
if (gasPPM > GAS_THRESHOLD) {
    digitalWrite(GAS_BUZZ, HIGH);
} else {
    digitalWrite(GAS_BUZZ, LOW);
}

Serial.print("Heart Rate: ");
Serial.print(heartRate);
Serial.print(" bpm | Weight: ");
Serial.print(weight);
Serial.print(" kg | Dust: ");
Serial.print(dust);
Serial.println(" mg/m3");

delay(2000); //
}

```

MineGuard_2040 sensors data



Chapter 4: Results and findings

This chapter outlines the key achievements of MineGuard 2040, highlights the successful components, discusses the challenges encountered, and suggests areas for future improvement.

Results:

All sensors and actuators in the MineGuard 2040 system functioned as intended during testing. The load cell accurately measured weight and triggered a blue LED and LCD alert when the load exceeded 500g. The dust sensor monitored air quality and activated a relay to power the water pump when dust concentration crossed the threshold of 0.35 mg/m^3 , with readings displayed on the LCD. The MQ-7 gas sensor and DHT11 temperature sensor continuously monitored carbon monoxide levels and temperature respectively, triggering a buzzer when CO levels exceeded 700 ppm or temperature surpassed 40°C . The pulse sensor was connected to measure heart rate, illuminating a red LED and displaying BPM on the LCD when the rate exceeded 140 bpm.

The sensor data were successfully uploaded to the ThingSpeak cloud platform via the NodeMCU, enabling real-time remote monitoring.

Limitations

- The pulse sensor (pulse amped) produced inconsistent readings, making it unreliable for precise fatigue monitoring.
- The dust sensor (GP2Y1010AU0F) was not calibrated to industrial standards, reducing the accuracy of measurements. A more accurate dust sensor is recommended for better performance in mining environments.
- The MQ-7 sensor only detects carbon monoxide and cannot identify other harmful gases found in mines.
- Both the temperature and gas sensors lacked proper calibration, which may affect data accuracy.

Recommendations

- Replace the current pulse sensor with a more accurate model like MAX30100 or MAX30102.
- Calibrate the dust sensor based on standardized dust concentration benchmarks.
- Upgrade to multi-gas detection modules to monitor a broader range of toxic gases.
- Apply proper calibration techniques for all environmental sensors to improve precision.

Challenges We Faced

- The pulse sensor proved challenging — despite multiple coding attempts and tests, it failed to provide consistent heart rate readings, making it unreliable for fatigue monitoring.
- The wiring became quite complex due to the number of components. Even small movements could loosen connections, which occasionally affected sensor performance.
- Integrating the dust sensor and water pump was difficult at times, especially with limited resources, which complicated proper wiring and setup.
- Calibrating the load sensor.

What We Learned

As a diploma project, we learned to program multiple IoT sensors, use NodeMCU, upload data to ThingSpeak, and enable communication between Arduino and NodeMCU. We also understood the importance of calibrating sensors for accurate readings.

Future Work

- Implement smart water management for dust suppression, especially important in regions like Oman, where freshwater is limited. This can be based on real-time dust and environmental data.
- Integrate temperature and heart rate sensors to enhance the detection of heat stress in workers, allowing for timely alerts and safety measures.
- Incorporate AI algorithms to predict equipment failures and potential safety risks, improving preventive maintenance and safety planning.
- Explore the use of laser-based dust sensors for more precise and reliable particulate concentration measurement, beyond what optical sensors like GP2Y1010AU0F can provide.

Conclusions

MineGuard 2040 successfully achieved its objectives by integrating multiple sensors to monitor critical mining parameters. The load cell reliably detected overloads above 500g, triggering alerts as designed. The dust sensor monitored air quality and activated the water pump when dust exceeded 0.35 mg/m^3 , though improved calibration is needed for accuracy. The MQ-7 gas sensor effectively tracked CO levels above 700 ppm, and the temperature sensor alerted at 40°C . While the pulse sensor struggled with accuracy, the overall system demonstrated real-time data upload to the cloud via NodeMCU. These results support Oman's Vision 2040 by enhancing mining safety and environmental monitoring, aligning with national priorities for sustainable industrial growth.

References & Appendices

- [1] Suriyakrishnaan, K., Arun Gandhi, R., Babu, R., Sakthivel, S., & Saurabh Dev. (2021). Smart Safety Helmet in Coal Mining Using Arduino. *Turkish Journal of Computer and Mathematics Education*, 12(11), 5481–5486.
- [2] V. Meenakshi, S. Radhika and V. V. Kaveri, "IoT Based Coal Mine Safety and health Monitoring System," 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2023, pp. 1-4, doi: 10.1109/ACCAI58221.2023.10200785. keywords: {Temperature sensors;Temperature measurement;Heart rate;Gases;Employment;Coal mining;Sensors;Coal Mining;Internet of Things (IoT);EDA & IR;HBR Sensor;Zig Bee}
- [3] D. Jung, Z. Zhang and M. Winslett, "Vibration Analysis for IoT Enabled Predictive Maintenance," 2017 IEEE 33rd International Conference on Data Engineering (ICDE), San Diego, CA, USA, 2017, pp. 1271-1282, doi: 10.1109/ICDE.2017.170. keywords: {Vibrations;Micromechanical devices;Monitoring;Reliability;Servers;Vibration measurement;Data collection}
- [4] Tripathi, A.K., Aruna, M., Parida, S. *et al.* Integrated smart dust monitoring and prediction system for surface mine sites using IoT and machine learning techniques. *Sci Rep* **14**, 7587 (2024). <https://doi.org/10.1038/s41598-024-58021-x>
- [5] Sifanu M, Kalebaila KK, Hayumbu P, Nabiwa L, Linde SJL. Analysis of respirable dust exposure data collected at a Zambian copper mine between 2017 and 2022. *Front Public Health*. 2024 Jan 31;11:1288485. doi: 10.3389/fpubh.2023.1288485. PMID: 38356653; PMCID: PMC10864506.
- [6] Florita et al. (2025a) Arduino with load cell and HX711 amplifier (Digital Scale), Random Nerd Tutorials. Available at: <https://randomnerdtutorials.com/arduino-load-cell-hx711/#:~:text=You%20wire%20the%20load%20cell,to%20connect%20to%20the%20Arduino.> (Accessed: 15 May 2025).
- [7] Sun Robotronics. (2021, April 28). PM2.5 Air Quality/Dust Sensor & Arduino Interfacing tutorial with GP2Y1010 || PM1.0, PM 2.5, PM 10 [Video]. YouTube. <https://www.youtube.com/watch?v=bRCVtVuYjRI>
- [8] Santos, S., & Santos, S. (2025, January 25). Arduino with Load Cell and HX711 Amplifier (Digital Scale) | Random Nerd Tutorials. Random Nerd Tutorials. <https://randomnerdtutorials.com/arduino-load-cell-hx711/#:~:text=You%20wire%20the%20load%20cell,to%20connect%20to%20the%20Arduino.>
- [9] TechForFun. (2023, March 16). E.P:-37 | Send DHT11 Data to ThingSpeak from Arduino Uno via NodeMCU | Techforfun [Video]. YouTube. <https://www.youtube.com/watch?v=ge6NZAhZNjI>