Faculty of Information Technology

University of Moratuwa

2023

Final Evaluation Report (IN 1901)

Level 1

Miner Safety Helmet

Group 28

Index number	Name with initials
214086K	**T.A.S.I. Jayalath**
214023R	A.M.C.I. Athapaththu
214211R	W.A.T.M. Thilakarathna
214177P	A.S.F. Sahla
214007X	A.D.R.V. Abeysiri

Name of the supervisor: Mr.B.H.Sudantha
Dean/Senior lecturer
Faculty of Information Technology

Name of the supervisor: Ms.Nipuni Chandimali
Lecturer
Department of Information Technology

Signature of Supervisor:

1. .....
2. .....

Date of Submission:
30/09/2023

Table of Contents

1.0 Introduction 1

2.0 Literature Survey 2

3.0 Aim and Objectives 3

3.1 Aim 3

3.2 Objectives 3

4.0 Proposed Solution 4

5.0 Analysis and Design 5

5.1 Blog Diagram 5

5.2 3D View 6

5.3 Schematic & PCB diagrams 7

6.0 Testing and Implementation 10

7.0 Source Code 14

8.0 Resource Requirement 24

9.0 Estimated Cost 25

Appendix A 26

Individual Contribution to the Project 26

Appendix B 31

References 31

1.0 Introduction

Mining is the process of extracting minerals from the Earth, including from its seas. This activity is necessary to obtain materials that cannot be naturally or artificially cultivated. It constitutes a major economic endeavour, involving numerous countries and offering various opportunities across society, ultimately benefiting the nation.

Mining operations can have adverse effects, both during active mining activities and after a mine has ceased operations. Workers engaged in underground mining are exposed to safety and health risks due to the challenging working environment. Consequently, nations worldwide have enacted regulations to mitigate these risks, leading to increased efforts to develop safety equipment.

When creating safety equipment, the primary focus is on ensuring the well-being of underground miners and facilitating communication in subterranean environments. Underground miners often lack effective means of communication with their colleagues, which led to the development of a two-way audio transmission system for improved communication.

Furthermore, miners venturing deep underground remain unaware of the safety conditions within the environment. Therefore, this project involves monitoring temperature, as well as concentrations of CO and CH4 gases, to assess whether the working conditions are safe or hazardous. These proactive measures can save lives and enhance mining productivity.

2.0 Literature Survey

For miners' safety, people made many miner's safety helmets in the world. There are some similarities and differences with this product. There are some examples of miner safety helmets around the world.

2.1 Mining helmet MHBL-1 to 6

This mining helmet has a full or gradational brim to protect miners' ears and eyes from falling debris and objects. Face shields, wireless video cameras, or breathing apparatuses can be fastened to the camera.

Figure 2.1-1 – Mining helmet MHBL

2.2 Jannatec smart helmet

This helmet acts as a connected device hub for hard rock miners and facilitates the collection of a range of data from the user. This helmet combines radio frequency identification tagging, photography and video capability, and integration with biometric sensors. This helmet's camera can be used to record underground incidents and transmit them to the surface.

Figure 2.2-2 -Jannatec smart helmet

3.0 Aim and Objectives

3.1 Aim

Design a miner's safety helmet that can identify bad conditions and enables two-way audio transmission.

3.2 Objectives

i. Detecting harmful gases and temperature.

ii. Enable real-time communication among miners using audio transmission features.

iii. Provide an LCD display for monitoring and analyzing sensor data.

iv. Improve miner safety and work efficiency through the advanced features of the safety helmet.

4.0 Proposed Solution

- Advanced sensors detect methane, CO concentrations (MQ4, MQ7).
- DHT22 sensors measure temperature in the mine.
- Wireless transmission enables real-time monitoring, analysis.
- Wearable helmet design provides convenience, constant monitoring.
- NRF24 module enables audio transmission between miners.
- NRF24 module enables audio transmission between miners.
- LCD display shows real-time data on gas, temperature.

5.0 Analysis and Design

5.1 Blog Diagram

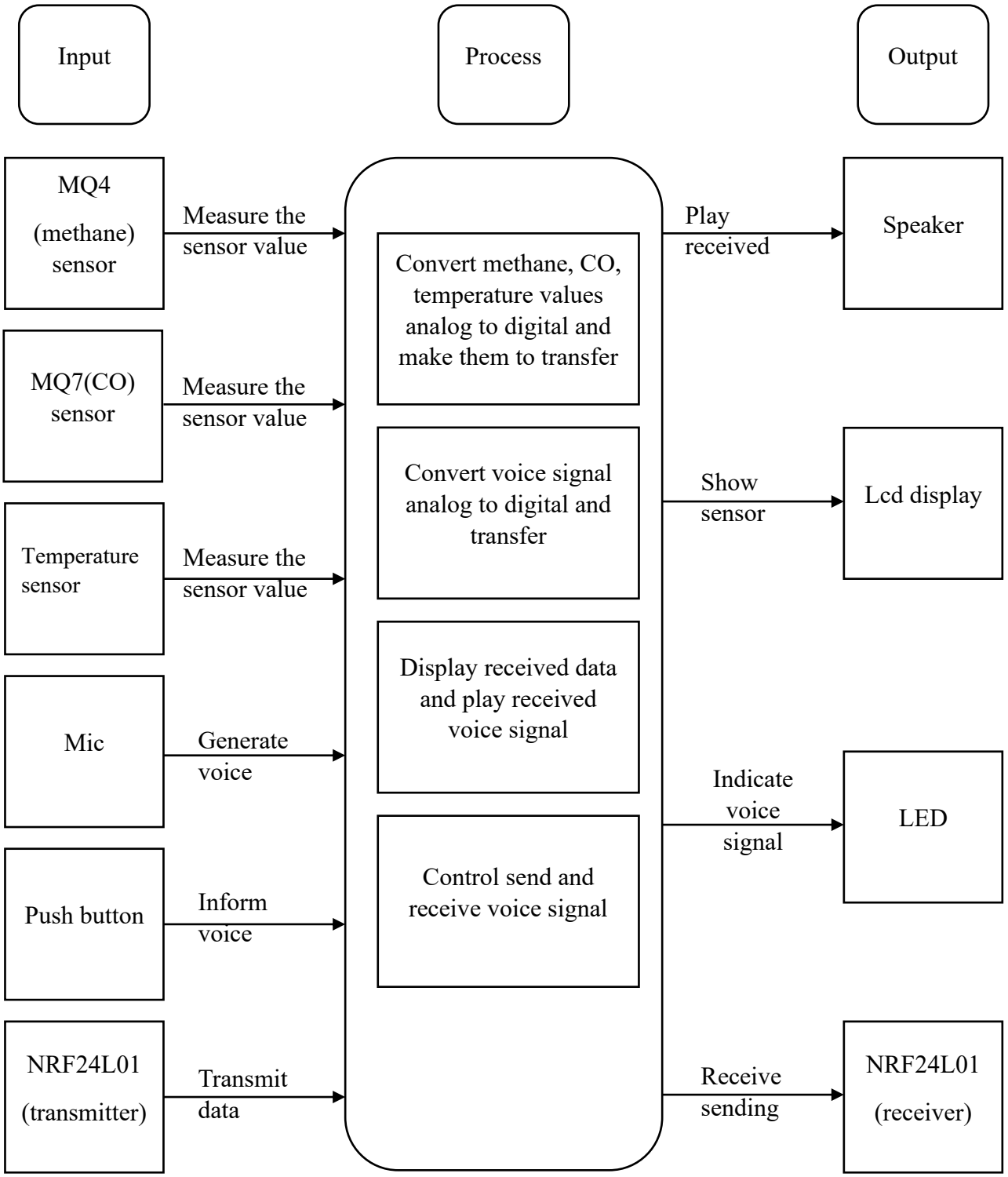

Figure 5.1-1 – Block diagram

5.2 3D View

Figure 5.2-1 - Transmitter

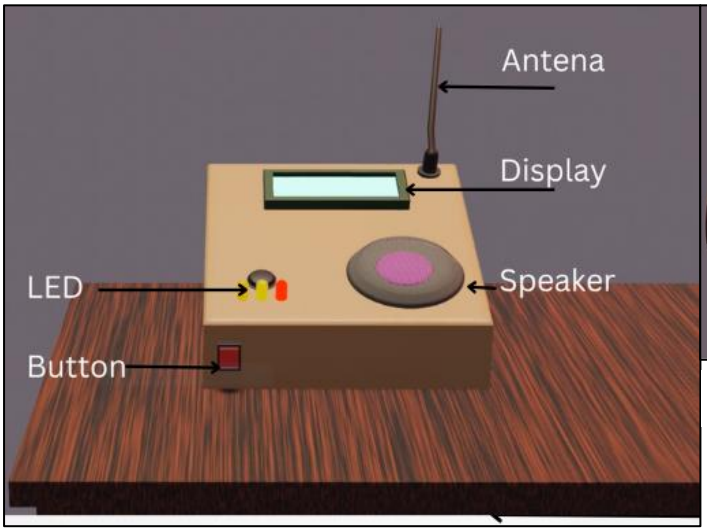

Figure 5.2-2 - Receiver

Figure 5.2-3 – Transmitter side 2

Figure 5.2-4 – Helmet with miner

Figure 5.2-5 – miner

5.3 Schematic & PCB diagrams

5.3.1 Transmitter

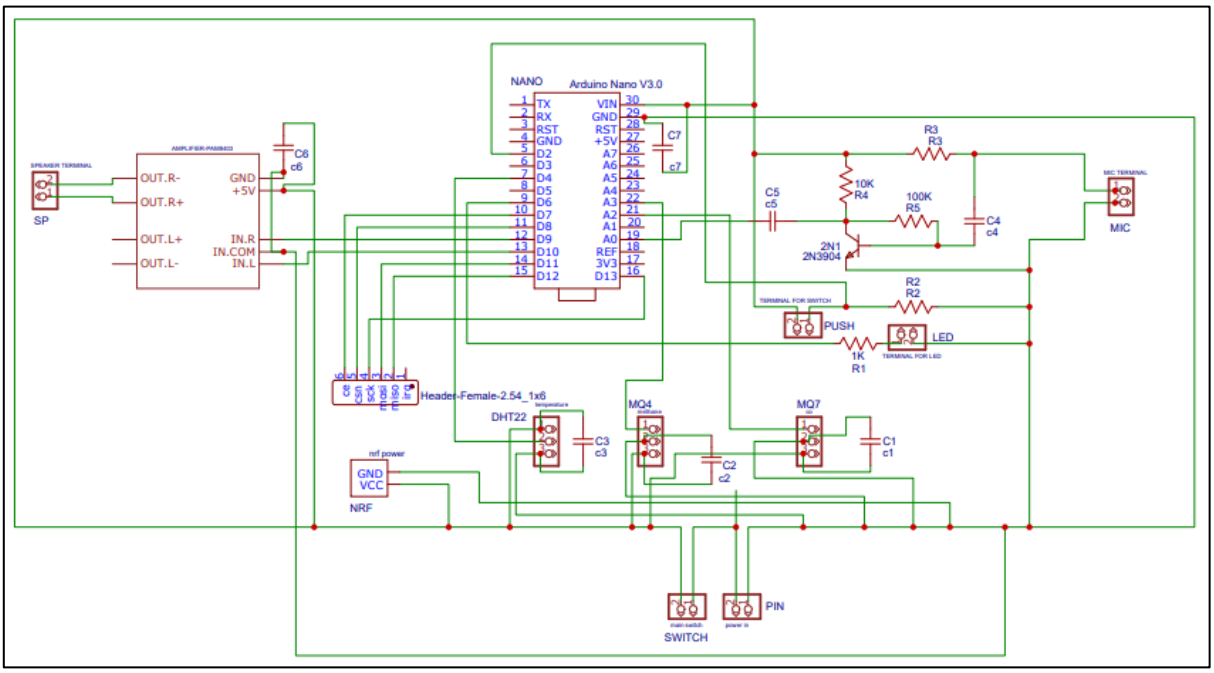

Figure 5.3.1-1 – Schematic of transmitter

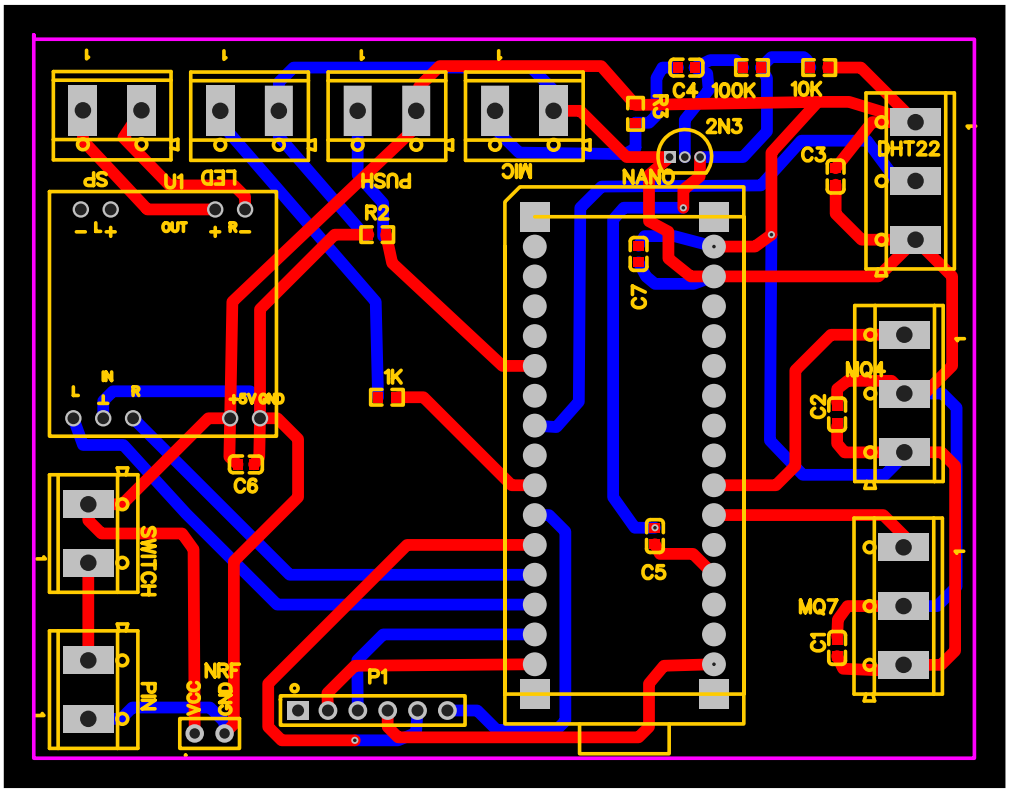

Figure 5.3.1-2 – PCB of transmitter

5.3.2 Receiver

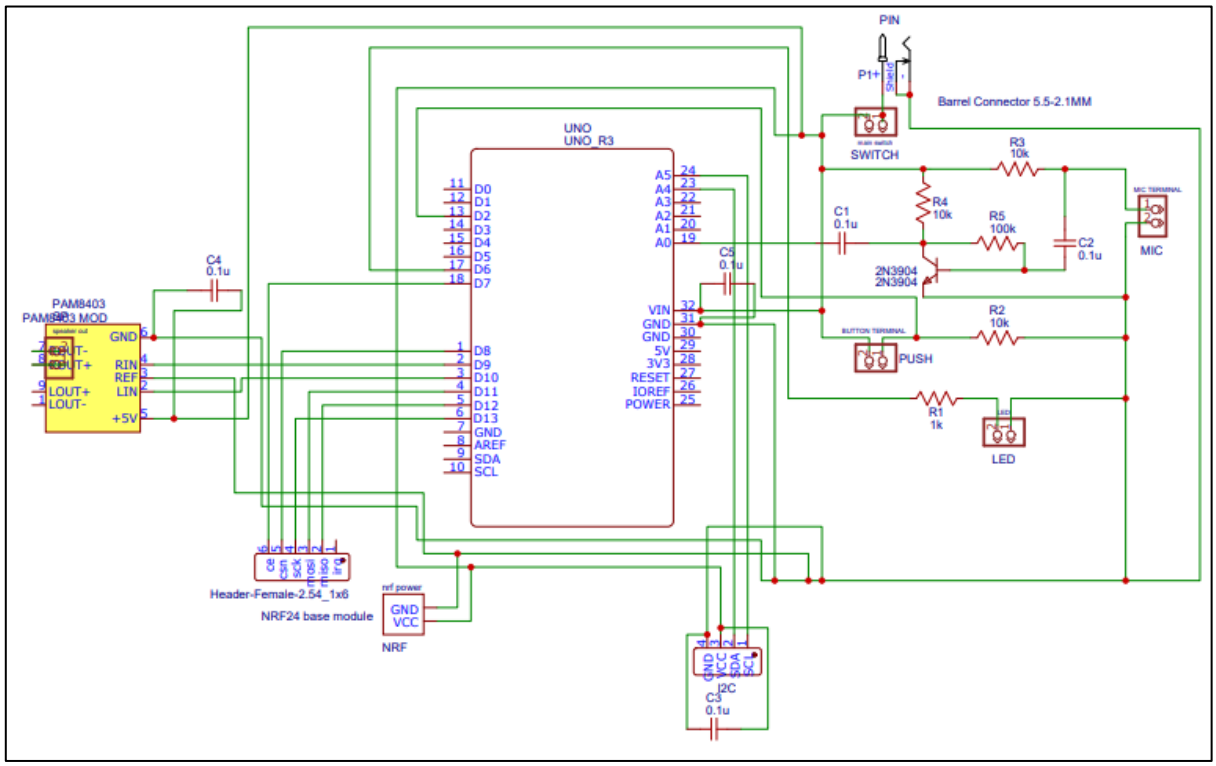

Figure 5.3.2-1 – Schematic of receiver

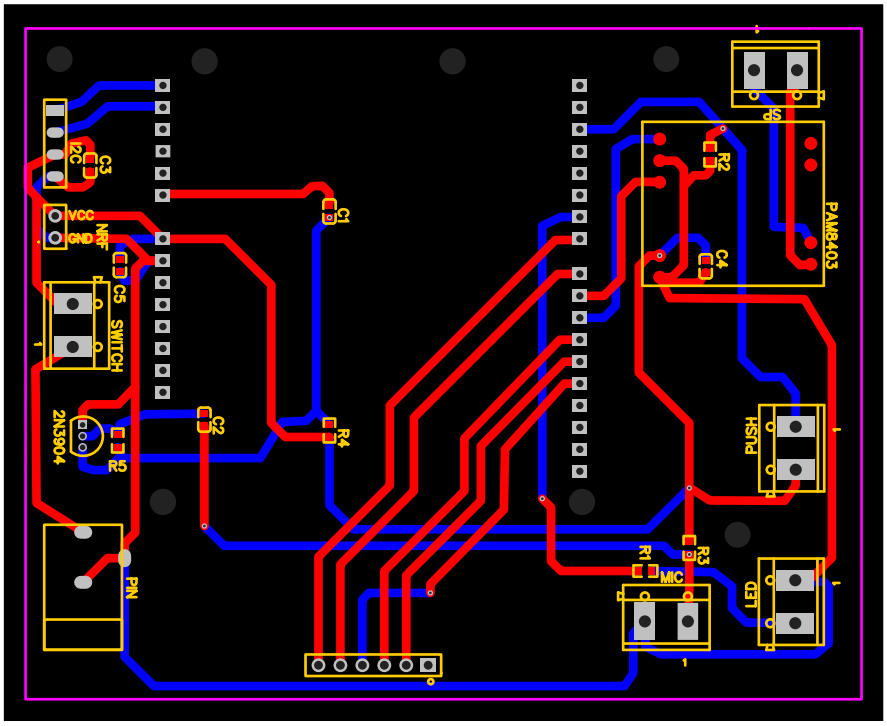

Figure 5.3.2-2 – PCB of receiver

5.3.3 Repeater

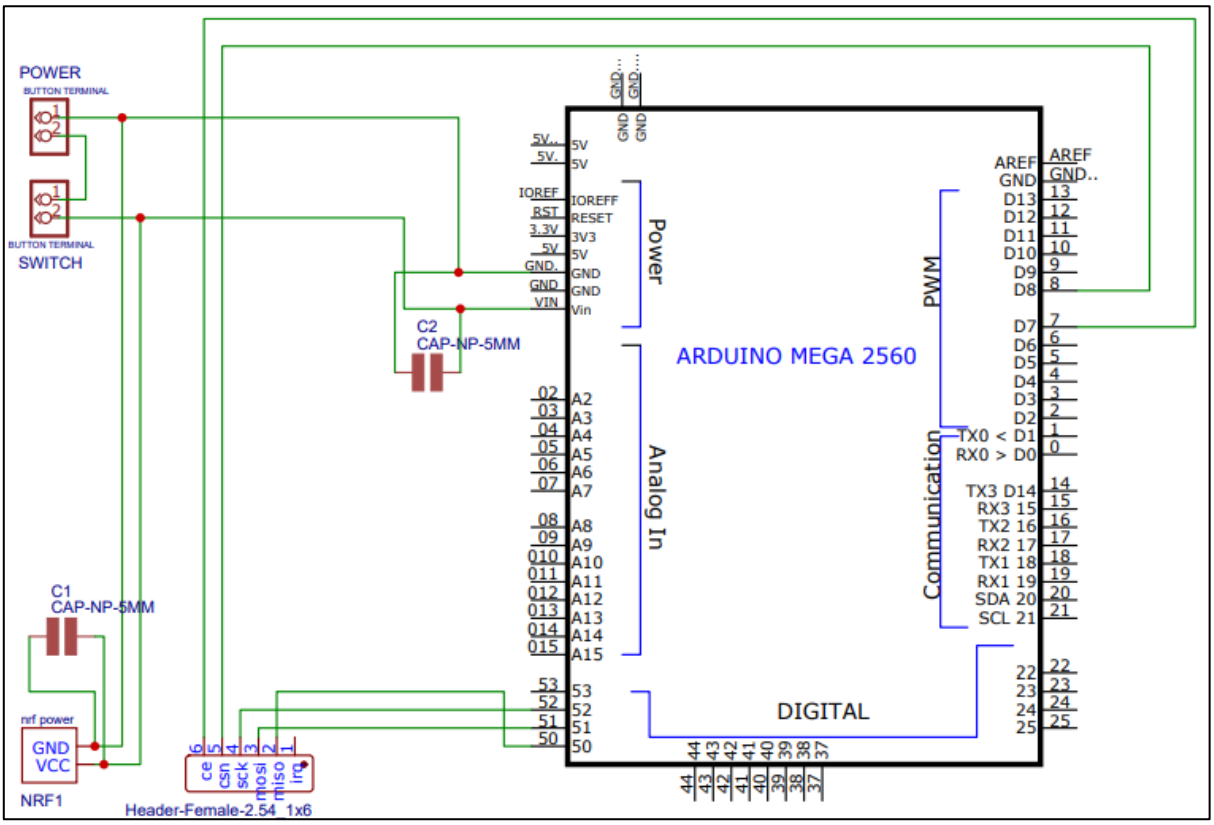

Figure 5.3.3-1 – Schematic of Repeater

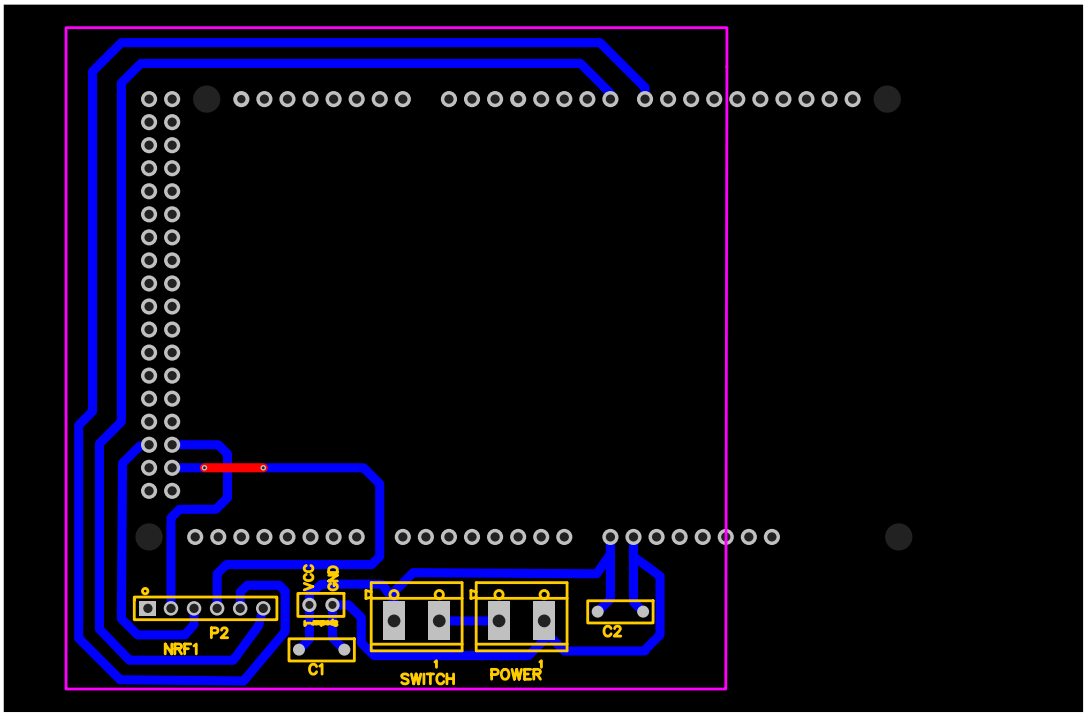

Figure 5.3.3-2 – PCB of Repeater

6.0 Testing and Implementation

When we decide to design our project, we have two major parts: the cording part and the mechanical part. We divide the work among the group members and follow many tutorials to learn about Arduino, modules, and sensors. After studying, we began to implement the circuits.

After writing codes and designing circuits, we began to test our work. Using NRF24 modules, we tested two-way audio transmission between two computers. One NRE24 module is connected to one computer, and the other NRF24 module is connected to the other computer, and audio is sent between the two computers. We send messages and test whether they are received from another side.

We test the mq4, mq7, and DHT22 temperature sensor, get data, send that data via the NRF from one computer to another, and check that the data is displayed correctly.

We decide on the overall system as two circuits. One is a transmitter, and the other is a receiver. mq4, mq7, and DHT22 temperature sensors and NRF24L01 connect as transmitters, and we create a PCB for them. We decide the dimension of the PCB as suitable for include inside the helmet and put connectors as needed to get the sensor outside of the helmet. The receiver is designed with the NRF24L01 display module.

We design the power supply for the transmitter. First, we calculate the voltage of the transmitter using a mustimeter and decide what the suitable voltage is for the transmitter. We get the 3.7V battery and use the XL6009 buck converter to convert the voltage to 5V. We test the charging hours when the transmitter is working and use the TP4056 battery charging module to charge the battery. We decided to use a 5V, 1A power adapter for the receiver and repeater because of its low cost.

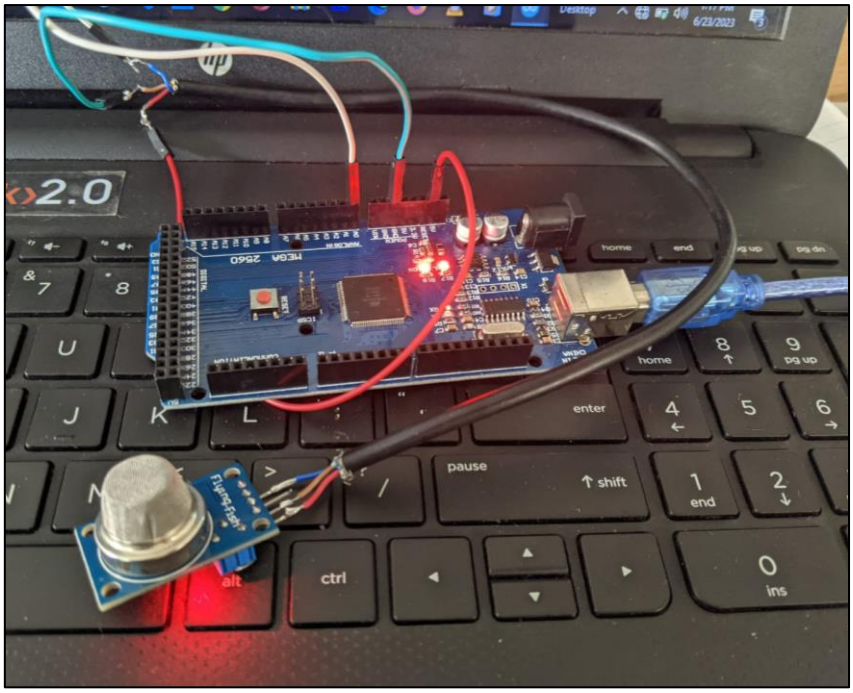

Figure 6.0-1 – Testing MQ4 Sensor

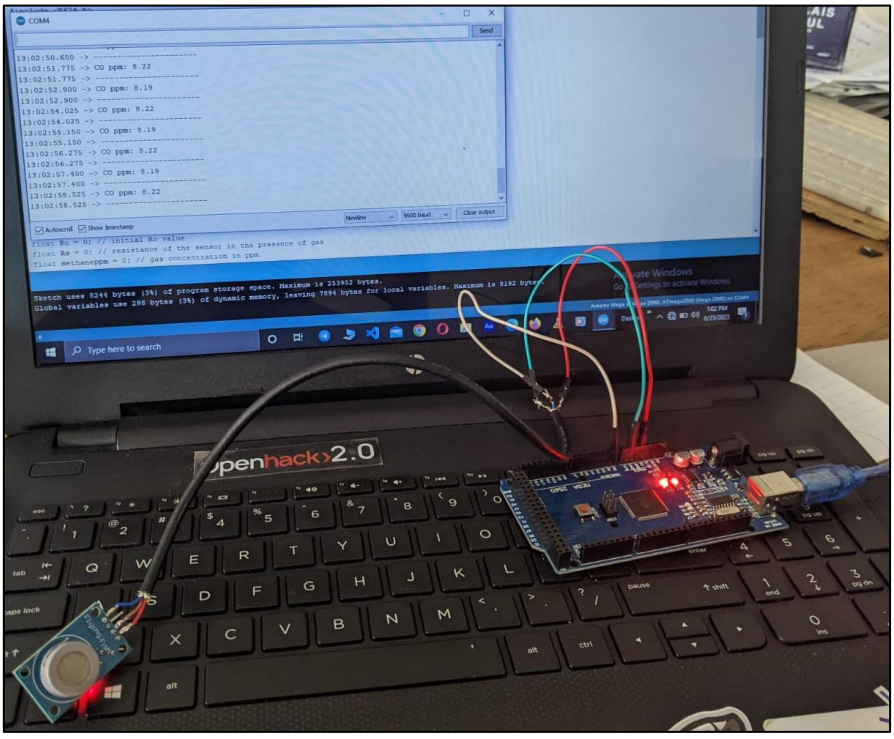

Figure 6.0-2 – Testing MQ7 Sensor

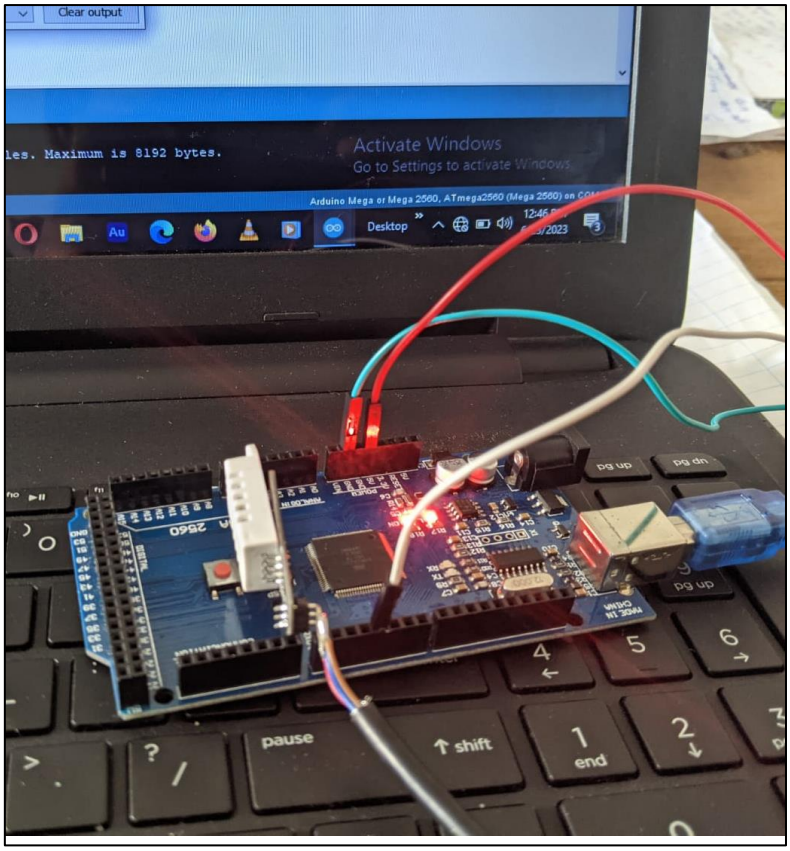

Figure 6.0-3 – Testing DHT22 Sensor

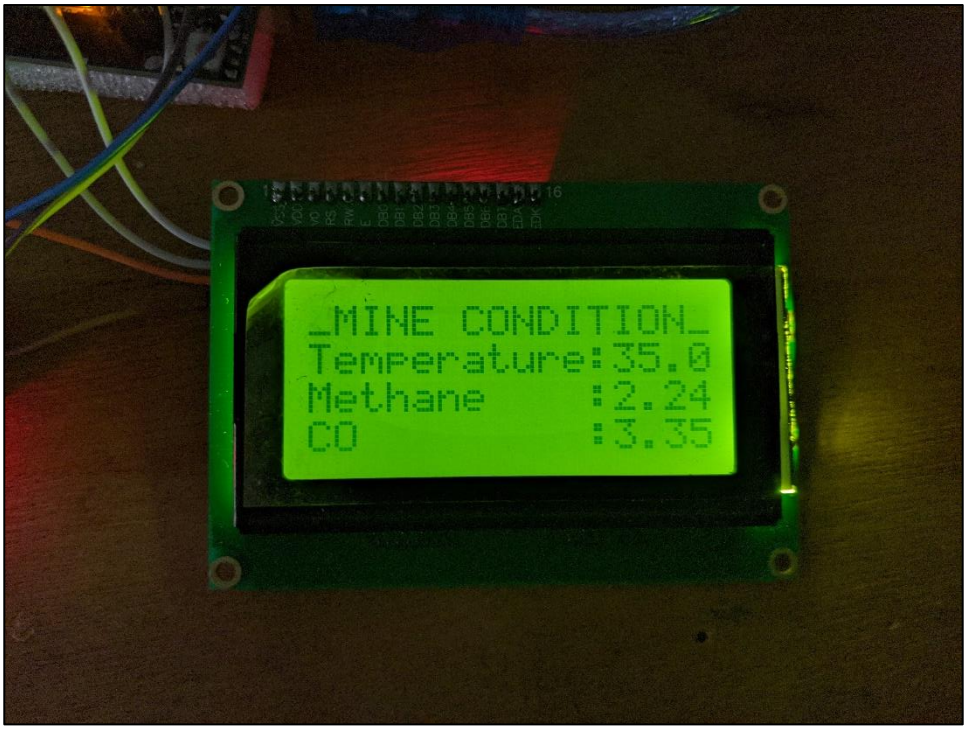

Figure 6.0-4 – Testing Lcd Display & I2c module

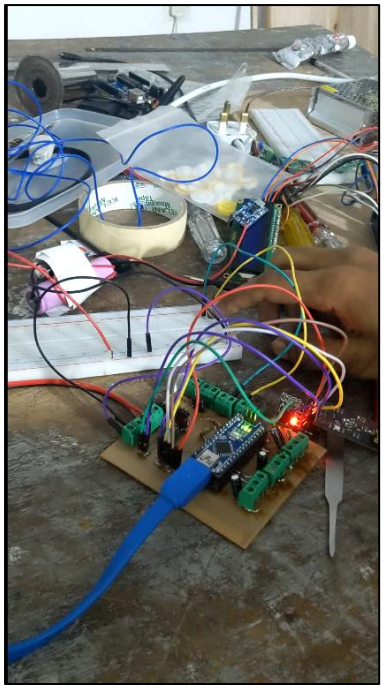

Figure 6.0-5 – Testing Battery pack

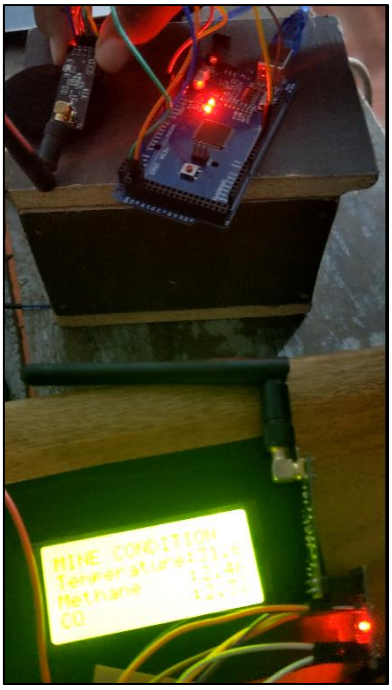

Figure 6.0-6 – Testing Nrf24l01

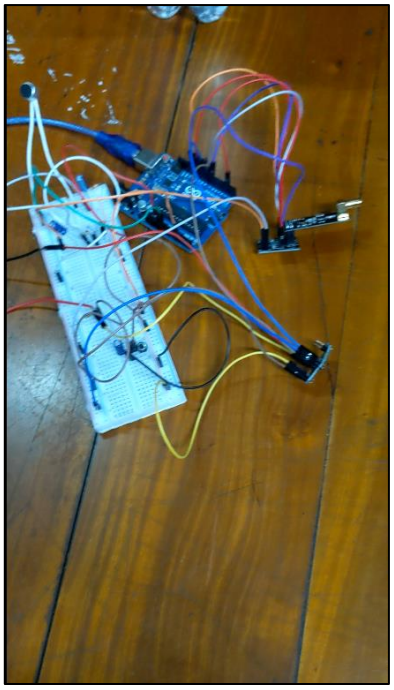

Figure 6.0-7 – Testing nrf24l01 audio transmission

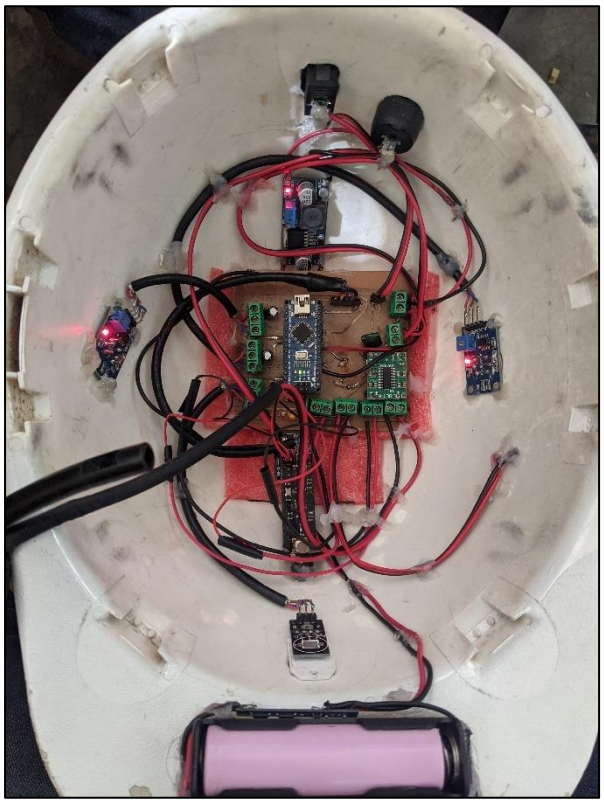

Figure 6.0-8 – Testing transmitter

Figure 6.0-9 – Testing receiver

7.0 Source Code

7.1 Transmitter

```
#include <RF24.h>
#include <SPI.h>
#include <RF24Audio.h>
#include <ezButton.h>
#include "printf.h" // General includes for radio and audio lib
////////////////////////////
#include <nRF24L01.h>
#include <DHT.h>
#include "MQ7.h"
////////////////////////////

ezButton toggleSwitch(5);

RF24 radio(7, 8); // Set radio up using pins 7 (CE) 8 (CS)
RF24Audio rfAudio(radio, 1);

const unsigned long duration = 5000; // Duration in milliseconds (5 seconds)
unsigned long startTime;
void (*resetFunc)(void) = 0;

const int buttonPin = 2; // Replace with the actual pin number of the button
bool buttonPressed = false;
bool audioInitialized = false;

const int switchrepeater = 5;

////////////////////////////////////////


// Setup the DHT22 sensor
#define DHTPIN 4 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22 // DHT22 (AM2302)
DHT dht(DHTPIN, DHTTYPE);

// Setup the MQ7 CO sensor
const int mq7Pin = A1;
float m = -0.6527;       // Slope
float b = 1.30;          // Y-Intercept
float R0Co = 21.91;        // Sensor Resistance in fresh air from previous
code
```

```
//////////////////////////setup the ch4 senor
const int MQ4Sensor = A2;
float R0 = 10.0;

////////////////////////////////////

// Struct to hold the temperature and CO values
struct SensorData {
  float temperature;
  float co_ppm;
  float other_value;
};



/////////////////////////////////////////


void setup() {
  // Initialization code
  Serial.begin(9600);
  radio.begin();
  pinMode(LED_BUILTIN, OUTPUT);
  startTime = millis(); // Set the start time
  radio.openWritingPipe(0xF0F0F0F0E1LL); // Address to send to (6-byte
alphanumeric)
  pinMode(switchrepeater, INPUT_PULLUP);
  toggleSwitch.setDebounceTime(50);

  // Setup the DHT22 sensor
  dht.begin();
  pinMode(MQ4Sensor, INPUT);
  pinMode(mq7Pin, INPUT);

}



void sendMessage() {

  float methaneppm = mq4Value(); // calculate the gas concentration in ppm
  ////////////////////////


  // Read the temperature data from the DHT22 sensor
  float temperature = dht.readTemperature();

  // Read the CO data from the MQ7 CO sensor
  float co_ppm =readCOConcentration();
```

```
  // Create a struct to hold the temperature and CO values
  SensorData data;
  data.temperature = temperature;
  data.co_ppm = co_ppm;
  data.other_value = methaneppm;

  //   Send the temperature and CO data over the radio module
  radio.write(&data, sizeof(data));

}

//MQ7function

float readCOConcentration() {
  float sensor_volt;    // Define a variable for sensor voltage
  float RS_gas;         // Define a variable for sensor resistance
  float ratio;          // Define variable for ratio
  int sensorValue = analogRead(mq7Pin); // Read analog values of the sensor

  sensor_volt = sensorValue * (5.0 / 1023.0); // Convert analog values to
voltage

  RS_gas = ((5.0 * 10.0) / sensor_volt) - 10.0; // Get value of RS in a gas


  ratio = RS_gas / R0Co; // Get ratio RS_gas/RS_air


  float ppm_log = (log10(ratio) - b) / m; // Get ppm value in linear scale
according to the ratio
  float ppm = pow(10, ppm_log); // Convert ppm value to log scale

  return ppm;
}

//mq4 function
float mq4Value()
{
  float sensor_volt;
  float RS_gas;  // Get value of RS in a GAS
  float ratio;   // Get ratio RS_gas/RS_air

  sensor_volt = analogRead(MQ4Sensor);
  sensor_volt = sensor_volt / 1024 * 5.0;

  RS_gas = (5.0 - sensor_volt) / sensor_volt; // Calculate RS in GAS
  ratio = RS_gas / R0;                        // ratio = RS/R0
```

```
  float ppm = pow(ratio, -1.85);
  return ppm;

}

////////////////////////////////
void audioTransmission()
{
  if (digitalRead(buttonPin) == HIGH) {
    // Button is pressed, start transmitting audio
    if (!buttonPressed) {
      initializeAudio();
      rfAudio.transmit();
      buttonPressed = true;
    }
  } else {
    // Button is not pressed, start receiving audio
    if (buttonPressed) {
      initializeAudio();
      rfAudio.receive();
      buttonPressed = false;
    }
  }

}

void initializeAudio() {
  if (!audioInitialized) {
    rfAudio.begin();
    rfAudio.setVolume(7);
    audioInitialized = true;
  }
}

void loop() {
  toggleSwitch.loop();
  int state = toggleSwitch.getState();
  if (state == LOW)
  {

    sendMessage();


  }
  else
  {
    if (millis() - startTime < 15000) {
      digitalWrite(LED_BUILTIN, LOW);
```

```
      sendMessage();
    } else if (millis() - startTime < 180000) {
      digitalWrite(LED_BUILTIN, HIGH);
      audioTransmission();
    } else {
      resetFunc();
    }

  }
}
```

7.2 Receiver

```
#include <RF24.h>
#include <SPI.h>
#include <RF24Audio.h>
#include "printf.h" // General includes for radio and audio lib
#include <ezButton.h>
#include <nRF24L01.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

ezButton toggleSwitch(5);

RF24 radio(7, 8); // Set radio up using pins 7 (CE) 8 (CS)
RF24Audio rfAudio(radio, 1);

const int spin1 = 9;
const int spin2 = 10;


const unsigned long duration = 5000; // Duration in milliseconds (5 seconds)
unsigned long startTime;
void(* resetFunc) (void) = 0;

const int buttonPin = 2; // Replace with the actual pin number of the button
bool buttonPressed = false;
bool audioInitialized = false;

const int switchrepeaterstate = 5;

LiquidCrystal_I2C lcd(0x27, 16, 4); // Set the I2C address and dimensions of
your display
```

```
// Define the struct to hold temperature and CO values
struct SensorData {
  float temperature;
  float CO_ppm;
  float other_value;
};

float tempAudio;
float CoAudio;
float CH4Audio;

SensorData sensorData; // Create an instance of the struct

void setup() {

  Serial.begin(9600);
  radio.begin();
  pinMode(LED_BUILTIN, OUTPUT);
  startTime = millis(); // Set the start time
  pinMode(switchrepeaterstate, INPUT_PULLUP);
  toggleSwitch.setDebounceTime(50);
  lcdsetup();

  // digitalWrite(5,HIGH);
}

void lcdsetup() {
  lcd.begin(); // Initialize the LCD
  lcd.backlight(); // Turn on the backlight
  lcd.setCursor(0, 1); // Set the cursor to the beginning of the first line
  lcd.print("____MINOR_____"); // Print a message
  lcd.setCursor(-4, 2); // Set the cursor to the beginning of the first line
  lcd.print("__SAFETY HELMET_");
  delay(1000);
  lcd.clear();
}

void lcdloop(float temp, float meth, double co) {
  // Your code logic here
  lcd.setCursor(0, 0); // Set the cursor to the beginning of the second line
  lcd.print("MINE CONDITION");

  lcd.setCursor(0, 1); // Set the cursor to the beginning of the second line
  lcd.print("Temp(c)    :");
  lcd.print(temp);
  lcd.setCursor(-4, 2); // Set the cursor to the beginning of the second line
  lcd.print("Methan(ppm):");
  lcd.print(meth);
```

```
  lcd.setCursor(-4, 3); // Set the cursor to the beginning of the second line
  lcd.print("CO(ppm)    :");
  lcd.print(co);

  //delay(1000); // Delay for 1 second
}

void lcdLoopAudio()
{


  lcd.clear();
  lcd.setCursor(2, 0); // Set the cursor to the beginning of the second line
  lcd.print("Audio->");
  lcd.print(119-(millis()/1000));
  lcd.print(" Sec");
  lcd.setCursor(0, 1); // Set the cursor to the beginning of the second line
  lcd.print("Temp(c)    :");
  lcd.print(tempAudio);
  lcd.setCursor(-4, 2); // Set the cursor to the beginning of the second line
  lcd.print("Methan(ppm):");
  lcd.print(CH4Audio);
  lcd.setCursor(-4, 3); // Set the cursor to the beginning of the second line
  lcd.print("CO(ppm)    :");
  lcd.print(CoAudio);
  delay(1000);

}


void receiveMessage()
{
  radio.openReadingPipe(1, 0xF0F0F0F0E1LL); // Address to receive from (6-byte
alphanumeric)
  radio.startListening();
  if (radio.available()) {


    radio.read(&sensorData, sizeof(sensorData));
    lcdloop(sensorData.temperature, sensorData.other_value,
sensorData.CO_ppm);
    tempAudio=sensorData.temperature;
    CoAudio=sensorData.CO_ppm;
    CH4Audio=sensorData.other_value;
  }



}
```

```
void audioTransmission()
{
  if (digitalRead(buttonPin) == HIGH) {
    // Button is pressed, start transmitting audio
    if (!buttonPressed) {
      initializeAudio();
      rfAudio.transmit();
      buttonPressed = true;
    }
  } else {
    // Button is not pressed, start receiving audio
    if (buttonPressed) {
      initializeAudio();
      rfAudio.receive();
      buttonPressed = false;
    }
  }

}


void initializeAudio() {
  if (!audioInitialized) {
    rfAudio.begin();
    rfAudio.setVolume(7);
    audioInitialized = true;
  }
}

void receiverMessageWithRepeater()
{
  radio.openReadingPipe(1, 0xF0F0F0F0D2LL); // Address to receive from (6-byte
alphanumeric)
  radio.startListening();
  if (radio.available()) {
    char text[32] = ""; // Assuming the message won't exceed 32 characters
    radio.read(&sensorData, sizeof(sensorData));
    lcdloop(sensorData.temperature, sensorData.other_value,
sensorData.CO_ppm);
  }

}
```

```
void loop() {
  toggleSwitch.loop();
  int state = toggleSwitch.getState();

  if (state == LOW)
  {
    // Serial.println("Inside repeater");
    receiverMessageWithRepeater();
  }
  else
  {
    if (millis() - startTime < 10000) {
      digitalWrite(spin1, LOW);
      digitalWrite(spin2, LOW);

      digitalWrite(LED_BUILTIN, LOW);
      receiveMessage();

    }
    else if (millis() - startTime < 120000) {
      digitalWrite(LED_BUILTIN, HIGH);
      lcdLoopAudio();
      audioTransmission();
    } else {
      resetFunc();

    }


  }


}
```

7.3 Repeater

```
#include <SPI.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN pins

struct SensorData {
  float temperature;
  float CO_ppm;
  float other_value;
};
```

```
SensorData sensorData; // Create an instance of the struct

void setup() {
  radio.begin();
  radio.setPALevel(RF24_PA_HIGH); // Set power level
  radio.openReadingPipe(1, 0xF0F0F0F0E1LL); // Address to receive from (6-byte
alphanumeric)
  radio.openWritingPipe(0xF0F0F0F0D2LL); // Address to transmit to (6-byte
alphanumeric)
  radio.startListening();
}

void loop() {
  if (radio.available()) {

    radio.read(&sensorData, sizeof(sensorData));

    // Process the received data if needed

    // Forward the received data to the receiver
    radio.stopListening();
    radio.write(&sensorData, sizeof(sensorData));
    radio.startListening();
  }
}
```

8.0 Resource Requirement

Software

- Arduino IDE - Codes
- Blender - 3D design
- EasyEDA - PCB design

Hardware

- MQ4 sensor
- MQ7 sensor
- DHT22 sensor
- NRF24L01
- Arduino nano
- Arduino uno
- Mics
- 16x4 Lcd display
- Speakers

9.0 Estimated Cost

Item	Quantity	Unit price (Rs)	Price (Rs)
Arduino nano	1	2530	2530
Arduino uno	1	1500	1500
NRF 24L01	3	860	2580
MQ4 sensor	1	540	540
MQ7 sensor	1	550	550
DHT22 temperature sensor	1	800	800
16*4 Lcd display	1	1200	1200
NRF 24L01 Base module	3	275	825
I2C Adapter Board	1	250	250
PAM8403 stereo audio amplifier	2	130	260
8 Ohm 0.5W Speaker	2	100	200
Condenser mic	2	40	80
LM2596 buck converter	1	320	640
5V SMPC power adapter	2	550	1100
3.7V 18650 battery	2	1020	2040
Other			5000
Total			**19875**

Table 9.0-1 – Estimated cost

Appendix A

Individual Contribution to the Project

T. A. S. I. Jayalath (214086K)

As the leader of the group, I divided the work among group members. And I checked the progress they have reached in each area that they were assigned to. According to that the part that I was assigned to was build a wireless communication system utilizing the NRF24 module. The purpose was to enable sensor value transmission and reception between the helmet and a central monitoring device, assuring the safety of miners in a variety of environmental conditions.

The NRF24 module was essential in establishing a dependable and effective wireless connection. It operates at 2.4GHz and uses the popular nRF24L01+ chip, which has low power consumption and an outstanding range. We were able to establish a smooth connection between the helmet and the central unit by using its capabilities.

We accomplished this by incorporating the NRF24 module into the safety helmet and outfitting it with sensors such as MQ7, MQ4, and DHT22. These sensors gathered information about environmental circumstances. The NRF24 module served as a communication bridge, wirelessly sending sensor information to the central monitoring unit. We had another NRF24 module coupled with an Arduino on the central unit side, which received sensor data from the helmet. We developed a receiver program to evaluate the received data and assess any potential safety issues. For example, if the helmet detected a sudden impact or excessive temperatures, the central unit would sound to the relevant authorities.

Overall, my contribution was to integrate the NRF24 module in order to provide flawless wireless communication between the safety helmet and the central monitoring unit. We were able to improve safety measures for minors by continually monitoring their vital sensor values in real-time, providing rapid responses to potential threats, and safeguarding their well-being in a variety of contexts by employing this technology.

A. M. C. I. Athapathu (214023R)

My contribution to the project was focused on implementing a 16x4 LCD display and an I2C module to provide real-time information and alerts in a control unit placed outside the helmet. This setup aimed to enhance the safety features of the helmet by enabling easy monitoring and feedback for the wearer.

To achieve this, I integrated a 16x4 LCD display into a control unit box separate from the helmet. The control unit contained the necessary circuitry and Arduino, to interface with the LCD display and communicate with the helmet's main control board. I connected the I2C module to the control unit, which facilitated the communication between the display and the microcontroller via the I2C protocol.

After setting up the hardware, I proceeded to code the functionality of the display. Utilizing appropriate libraries and code snippets, I programmed the Arduino to send relevant information to the LCD display. This information includes mainly environmental conditions such as temperature, humidity, CO, and Methane concentration by the helmet's integrated sensors.

By incorporating the 16x4 LCD display and the I2C module into the control unit, we provided a clear and concise interface for monitoring and displaying crucial data related to the minor's safety. The display's real-time information allowed the wearer to stay informed about their surroundings and vital statistics, ensuring a heightened level of awareness in potentially hazardous situations. This contribution significantly enhanced the safety capabilities of the Minor Safety Helmet project, empowering minors and their guardians with valuable information to prevent accidents and respond effectively to any potential risks.

W. A. T. M. Thilakarathna (214211R)

My contribution to the project mainly focused on designing and implementing the power and charging system, Audio transmission and the PCB (Printed Circuit Board) layout for the helmet's electronics. These aspects were crucial in ensuring the reliable and efficient operation of the safety helmet.

For the power and charging system, I carefully selected appropriate components and integrated them into the helmet's design. This involved choosing a suitable rechargeable battery with the necessary capacity and voltage specifications to power the helmet's sensors, communication modules, and other electronic components.

To transmit audio signals using the NRF24L01 module, we employ a specialized library known as 'rf24audio.h.' This library provides essential functions enabling the transmission of audio signals in a half-duplex mode, allowing for efficient bidirectional communication. As part of our setup, we integrate the PAM8403 audio amplifier module to enhance the audio output quality significantly.

In terms of PCB design, I worked on creating an optimized layout that accommodated all the necessary electronic components and provided efficient electrical connections. I carefully considered the placement of components to minimize signal interference and maximize the helmet's performance. Through meticulous attention to detail, I ensured that the PCB design followed industry best practices for reliability, signal integrity, and manufacturability. This included proper ground planes, routing of traces, and consideration of any specific requirements for high-speed or sensitive signals.

Overall, my contributions to the Minor Safety Helmet project focused on the power, Audio transmission and PCB design aspects. These contributions played a vital role in the overall functionality and reliability of the safety helmet, enhancing its safety features and usability for the target user.

A. S. F. Sahla (214177P)

My contribution to the project involved studying and calibrating the MQ-7 sensor to accurately measure carbon monoxide (CO) concentration, as well as implementing the DHT22 sensor to collect temperature and humidity data. These contributions played a critical role in enhancing the safety features of the helmet by providing real-time environmental information.

To begin, I extensively researched and studied the MQ-7 sensor, understanding its working principle and specifications. I performed calibration experiments to ensure precise and accurate measurements of CO concentration. This involved exposing the sensor to known CO concentrations and adjusting the calibration factors accordingly. By calibrating the sensor, we could obtain reliable CO readings that accurately reflected the level of this potentially harmful gas in the surrounding environment.

In addition to the MQ-7 sensor, I collaborated with another team member to study and implement the DHT22 sensor. We researched the sensor's capabilities and determined the appropriate methods to acquire temperature and humidity readings. Working together, we designed and coded a program to interface with the DHT22 sensor and retrieve the necessary data.

By integrating the calibrated MQ-7 sensor and the implemented DHT22 sensor into the safety helmet, we provided valuable environmental data to the wearer. Real-time CO concentration readings enabled the early detection of potential gas leaks or hazardous environments, while temperature and humidity data offered insights into comfort and environmental conditions. These contributions significantly enhanced the safety measures of the Minor Safety Helmet project, empowering minors to make informed decisions and take necessary precautions in various situations.

A. D. R. V. Abeysiri (214007X)

In the project, my contribution focused on studying and calibrating the MQ-4 sensor to accurately measure methane concentration. Additionally, I collaborated with another team member to study and implement the DHT22 sensor, enabling the collection of temperature data. These contributions were crucial in enhancing the safety features of the helmet by providing real-time environmental information.

To begin, I conducted extensive research and familiarized myself with the MQ-4 sensor's specifications and working principles. I studied its response characteristics to methane gas and conducted calibration experiments to ensure precise measurements. By exposing the sensor to known concentrations of methane and adjusting the calibration factors, we achieved accurate and reliable readings of methane concentration. This calibration process was essential to ensure that the sensor provided accurate and actionable data in potentially hazardous environments.

In parallel, I collaborated with a teammate to study and implement the DHT22 sensor. We explored the sensor's capabilities and collectively designed a program to interface with the DHT22 sensor and retrieve temperature data. By combining our efforts, we coded the necessary functions to ensure reliable and accurate readings. The DHT22 sensor enables the helmet to monitor ambient temperature, providing valuable insights into the wearer's comfort and environmental conditions.

By integrating the calibrated MQ-4 sensor for methane detection and the implemented DHT22 sensor for temperature monitoring, our combined efforts significantly enhanced the safety capabilities of the Minor Safety Helmet project. The real-time methane concentration readings enabled early detection of potential gas leaks or hazardous environments, ensuring the wearer's safety. Additionally, the temperature and humidity data provided valuable information for comfort and environmental awareness. These contributions empowered minors to make informed decisions, take necessary precautions, and navigate potentially hazardous situations with greater confidence.

Appendix B

References

1) *https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/#how-does-the-nrf24l01-module-work*
2) *https://www.roboticskanti.com/post/wifi-walkie-talkie*
3) *https://www.electronicwings.com/*
4) *https://scionelectronics.com/*
5) *https://www.nordicsemi.com/products/nrf24-series*
6) *https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/*
7) *https://github.com/nRF24/RF24Audio*
8) *https://docs.arduino.cc/hardware/uno-rev3*