Golden helmet

IOT-Based Miner Safety System

Graduation Project submitted to the Faculty of Computers and Information in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science

Information Technology Department &
Computer Science Department

Mostafa Mahmoud Elsayed
Sohila Ahmed Mohamed
Shaimaa Mohamed Taha
Hamdy Mohamed Elbeltagy
Zeyad Amin Zahran
Mostafa Mohamed Mamoon
Rawda Essam Abd Elbary
Mostafa Essam Mostafa
Mahmoud Mamdouh Mohamed

Supervisor

Dr. Mohamed Handousa
Dr. Fatmaelzahraa Ahmed AbdelGhani

Abstract

The life of the miners matters.

Through the history of mines and working in them, we shared many dangerous stories that face people who work in mines, for example, explosions that caused the death of many people or people spending all their life disabled.

We have been thinking day and night about how we can help those people. Therefore, our project is classified into two main parts the first part is the helmet and bracelet, and the second part is the application.

We named our project the GOLDEN HELMET.

The helmet that a miner can wear on his head during work. It has many sensors connected to the application on the supervisor's mobile. So if there is any problem that faces the miner, the helmet will send a message to the supervisor as well as the helmet will vibrate. At the same time, the alarm system on the helmet will alarm the miner.

In addition, there will be a bracelet on the miner's arm. That bracelet will detect the miner's health problem and send a message to alarm the supervisor about this problem.

Finally, we want to tell you that we are trying our best to help the miner, keep him safe, and reduce the overall dangers.

Acknowledgment

It is a pleasure to acknowledge many people who knowingly and unwittingly helped us to complete our project. First, let us thank God for all the blessings, which carried us through all these years.

We express our gratitude to Dr. Mohamed Handousa and Dr. Fatmaa Elzahraa.
For their cooperation and encouragement during the completion of this project.

Also, we want to thank here as our project guide who has always stood by our side and guided, appreciated, and encouraged us to get into more ventures. Which proved to be of immense help in the successful completion of this project.

We extend our sincere gratitude to our teachers and guides who made an unforgettable contribution. We thank all the non-teaching staff of our faculty that were always ready to help in whatever way they could.

Table of Content

List of Figures

Chapter 1
Introduction

1.1 Overview

Mining is indispensable to goods, infrastructure, and services that enhance the quality of their lives. As a society, we enjoy the many advantages that industry-manufactured products provide us by processing these raw materials. Working on the earth presents many different security and health dangers. Frequently the underground environment is shaky or unpleasant. The safety of workers should always be a consideration in any form of mining. Mining operations prove to be a risky venture as far as the safety and health of workers are concerned. These risks are due to different techniques used for extracting various minerals. The deeper the mine, the greater the risk. These safety issues are of grave concern, especially in the case of coal industries. The safety of workers is necessary for any form of mining, whether it is coal or other minerals. In underground mining, there is a concern about the safety of the workers due to its highly changing environment. Thousands of miners die from mining accidents each year.

1.2 Problem Statement

- Fatal accidents in mines increased by 10.3 %.
- Many miners face Explosions that occur and lead to the death of dozens, perhaps hundreds of gases called fire dumps.
- Choking workers with carbon monoxide and other gases such as carbon dioxide and smoke.
- Methane is a flammable gas when 5% to 16% concentration is mixed. In deep mines under coal or pits under pressure.
- Carbon monoxide leads to red blood cells not transporting oxygen to tissues, leading to suffocation and damage to the brain and heart tissue. Symptoms include headache, dizziness, nausea, vomiting, and coma. Eventually, carbon monoxide leads to death.
- Not knowing the location of the miner workers. Because of the difficulty of detecting the entire mine and reaching the workers.
- Difficulty responding quickly to all this and reaching the worker easily.
- Not following enough for the worker's health, such as blood pressure, blood oxygen, and sugar in those atmospheres, leads to fainting and sometimes death.

1.3 Motivations

Egypt has witnessed development in the past years in the mining sector, as it contains 23 million mines. Despite this development, there is a problem, which is the health of miners, that we lose about 20% and more per day due to gas leaks or changes in temperatures, which affect the health of the worker. We can't ignore this percentage, so we thought about making a helmet that warns the miner if there is a problem and the supervisor to know the status of each miner to make the appropriate decision to save him.

1.4 Objectives

- In our project, an Intelligent System has been developed to assist the Miners. It consists of Various Sensors to detect and Monitor any Hazardous Events to which the miner has exposed any danger. The alarm device warns the miner and sends it to the supervisor warning through the mobile application.
- In our device. We use sensors that ensure the health and safety of miner workers through helmets and bracelets.
- We use an exclusive sensor to Predicting explosions, another sensor for Detection Gas Leakage (methane, carbon monoxide, carbon dioxide, smoke), and a sensor for Measuring Temperature and Humidity, and those sensors integrated into IoT helmet.
- To ensure the health and track vital rates. We use a band that detects the percentage of oxygen in the blood, heart rate, and body temperature.
- We use Alarm beeps and send a warning message through the application when a problem occurs to take quick action.

1.5 Contributions

- Ensure the health and safety of the workers through helmets and bracelets.
- Predicting explosions by an exclusive sensor.
- High Efficiency.
- Gas Leakage Detection.
- Measuring Temperature and Humidity.
- Tracking the vital rates of miners through the bracelet.
- Alarm beeps and sends a warning message when a problem occurs.

1.6 Project Outlines

This document consists of six chapters in addition to one appendix and references.

A brief description of the contents of each chapter in the following paragraphs:

Chapter 1: Introduction of our project.

Chapter 2: Literature Review provides the reader with an overview of the previous related work and the relation between our work and the relevant work.

Chapter 3: - System Analysis includes the analysis of the existing system, system requirements, user requirements, system architecture, and development methodology using the tools and languages used in our system.

Chapter 4: - System Design provides the system design, a class diagram, database design and interface design.

Chapter 5: - System Implementation shows the process of mapping design into implementation, sample application code, system testing, results of the investigation, and goals achieved.

Chapter 6: - Future work and conclusion.

Chapter 2
Related work

Preamble

During this chapter, we will mention all the competitor's projects, their project ideas, and the pros and cons of their projects.

Through the cons in their projects, we started looking for further solutions that help to decrease the percentage of problems that faces the miner during work.

2.1 Previous systems

2.1.1- Smart Helmet for Factory Workers:

Basic idea:

The Smart helmet for factory workers uses a sensor WGTP that detects temperature and humidity Through light and movement that should be between 25:28 and 31 is hot because the worker cannot notice that it is a high temperature. In this case, using another sensor to warn of a helmet collision, if the worker stops the helmet sensor, it is okay that the supervisor takes action. The helmet is connected to a smartphone by Bluetooth to send a message to the supervisor in danger.

Advantages:

1- Detect temperature and humidity
2- Gives a warning of a helmet collision

Problems:

1- Not giving everyone health care for everyone.
2- Not detect gas leakage.
3- Send a message by Bluetooth

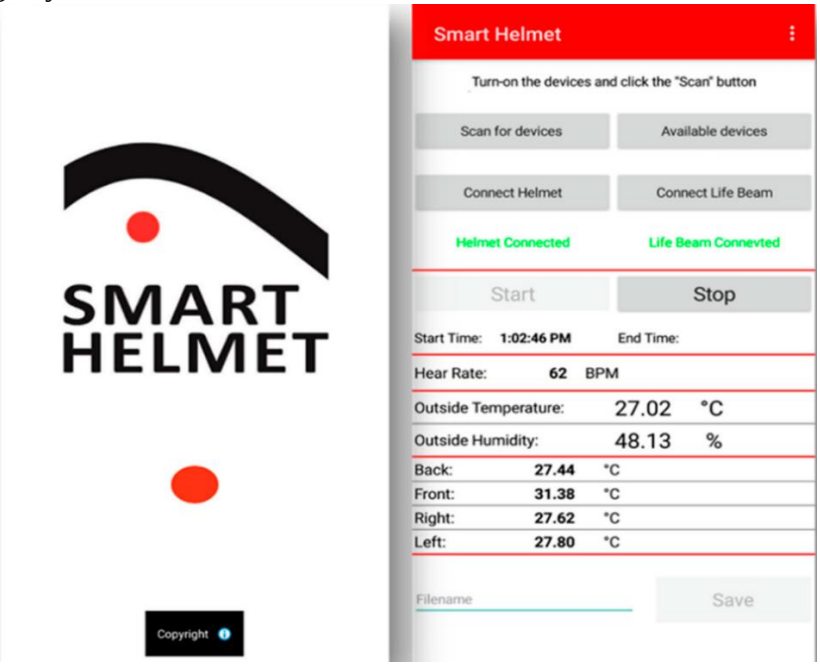


Figure 2-1 Smart Helmet for Factory Workers

2.1.2- Intelligent Helmet for Coal Miners

Basic idea:

Intelligent Helmet for Coal Miners and safety monitoring. Various parameters like Methane gas, Carbon monoxide gas, Temperature, and Humidity are monitored for the safety of coal miners. These sensors should be in the helmet of the coal miners. Whenever any parameter crosses a particular threshold value, a buzzer turns on to indicate the miner of the danger. This system performs the function of the miner's safety.

Advantages:

1- Used in various coal mines to detect dangerous environmental situations.
2- Provides an automatic safety system for miners and other workers or engineers entering the coal mine.

Problems:

1- Not send this data to a remote internet.
2- Not adding the module of the voice alarm system to indicate parameters crossing the threshold value.

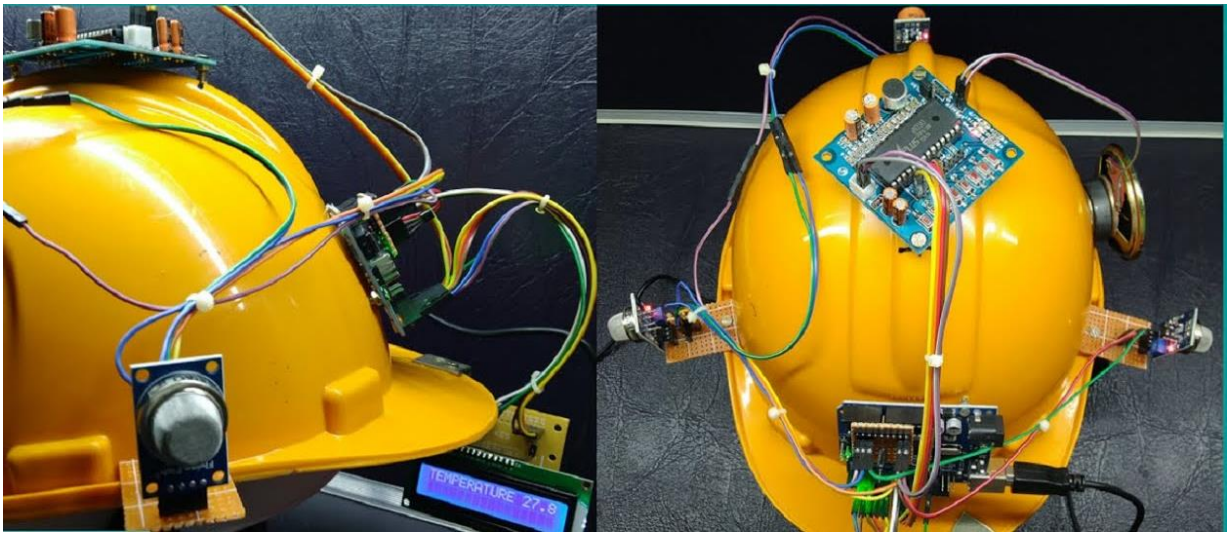


Figure 2-2 Intelligent Helmet for Coal Miners

2.1.3-Ava Doctor (Vital Signs Bracelet)

Basic idea:

AVA Doctor brings an automated method of collecting vital signs and essentially helping both patient and the doctors to address the issue at the earliest. AVA Doctor is a wearable bracelet that can be given to the patient by a hospital at an emergency point, check-in point, and/or for outpatient home treatment to collect vital signs such as blood pressure, and heartbeat/oxygen rates and instantly transfer these data over the internet to the cloud and then to the doctors, paramedic, nurses, and patient using mobile or web applications.

AVA Doctor allows you to measure:

- Heart Rate: Heart Beats per minute (bpm).
- Heart Oxygen: Oxygen in red blood cells (%).
- Blood Pressure: Pressure of circulating blood (mmHg)
- Steps: Daily Steps & Distance (km, miles)
- Calories: Daily calories spent (KCal).
- Sleeping Hours: Deep and Light Sleep (hours, min).

Advantages:

1- Waterproof: IP67.
2- Battery: polymer lithium battery.
3- Standby time: 10-15 days.

Problems:

1- Connected by Bluetooth.

Figure 2-3 Ava Doctor (Vital Signs Bracelet)

2.1.4- IOT Based Intelligent Helmet for Miners

Basic idea:

It's a smart helmet, and an Intelligent System has been developed to assist the Miners. It consists of Various Sensors to detect and Monitor any Hazardous Events and transmit the data to the Control Station to Alert the Miners. The Transmission is done using the Zigbee Protocol and The Monitoring is done on a Control Station.IOT Based Intelligent Helmet allows us to: The Intelligent Helmet is integrated with a variety of sensors including Temperature sensors, IR sensors, Smoke sensors, Methane Gas sensors, and Heart Rate Sensors. The Oxygen Sensor measures the level of Oxygen in the Underground Mining Environment. The Temperature Sensor Measures the Real-Time Temperature of the surroundings. The IR sensor is used to detect whether the miner is wearing a helmet or not. MQ2 smoke sensor detects the concentration of gases such as H2, LPG, CH4, CO, Alcohol, Smoke, or Propane. MQ4 Methane Gas Sensor.IOT Based Intelligent Helmet for Miners detects the Methane Concentration in the Underground Mines. A heart rate sensor is used to monitor the heart rate of miners and any drastic changes in the Heart Rate can be determined. A Panic button is placed in the helmet to trigger the Emergency Alert. Whenever any hazardous events happen in the underground area buzzer will be activated. Zigbee protocol is used to transmit the data from the sensor to the Receiving Station and the transmitted data is sent to the cloud.

Advantages:

1- Including a Temperature Sensor, Smoke Sensor, Heart Rate Sensor
2- Monitor the Biological rates of humans.
3- Locate the position.

Problems:

1- Connected by Bluetooth.
2- Unreliable Transmission. The Connection between the Control Room and the Miners is not stable and may cause a loss of Connection and the Whole system may NOT work.
3- Not anticipating danger as soon as possible

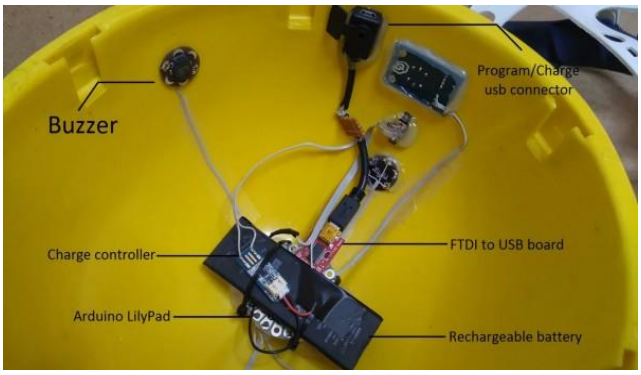


Figure 2-4 IOT Based Intelligent Helmet for Miners

2.1.5- Angel Helmet

Basic idea:

It's a smart helmet, invented by a Spanish company, which has been reinvented with the only aim: of saving lives. It contains detecting systems of motion, impact, and active and passive location, besides communication systems, health sensors, and atmospheric quality sensors.

Angel helmet allows us to:

- Analyze the quality of air and notice if it is harmful to the worker.
- Monitor Blood Tension, Temperature, Heart Rate, and Blood Oxygen.
- Locate the positions of miners.

Advantages:

1- Monitor the vital rates of humans.
2- Locate the position.

Problems:

1- Inaccurate knowledge of the type of danger coming.
2- Not anticipating danger as soon as possible.
3- Supervisor can't track miners ‘status from time to time.

Figure 2-5Angel Helmet

2.2 Competitive Analysis

CATEGORY	Golden helmet	Smart helmet for factory workers	intelligent helmet for coal miners	IOT based intelligent helmet for miners	Angel helmet	Ava doctor (vital sign bracelet)
Detect Smoking	✔	✘	✔	✔	✘	✘
Detect LPG	✔	✘	✘	✔	✘	✘
Detect (CO)	✔	✘	✔	✘	✘	✘
Detect CH_4	✔	✘	✔	✔	✘	✘
Detect Natural gas	✔	✘	✘	✘	✔	✘
Detect Butane	✔	✘	✘	✘	✘	✘
Detect Propane	✔	✘	✘	✔	✘	✘
Miner's location	✘	✘	✘	✔	✔	✘
Vibration (produce a tone)	✔	✘	✔	✔	✘	✘
Detect temperature and humidity	✔	✔	✔	✔	✘	✘
Percentage of oxygen in blood	✔	✘	✘	✘	✔	✔
Measure heart rate	✔	✘	✘	✔	✔	✔
Miner's temperature	✔	✘	✘	✘	✔	✔
Alert message in case of danger	✔	✘	✔	✘	✘	✘
Warning of gas leakage	✔	✔	✘	✘	✘	✘

2.3 Summary

- So, during our investigations, we saw the cons of the previous projects. We tried our best to develop a new idea with better perspectives.
- Firstly, we were able to track the vital rates of the miner through the bracelet as well as the gas percentage at the same time through the helmet.
- Secondly, if there was any threat the helmet and bracelet will send an alert to the supervisor and a vibration to the miner's helmet.

Chapter 3
System Analysis

Background:

This section studies the system analysis and design process. It is a process of collecting and interpreting facts and identifying the problems. System analysis is for knowing a system or its parts to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components work efficiently to accomplish their purpose. In addition, it is a process of observing systems for troubleshooting or development purposes. And systems analysis can include looking at the end-user implementation of a Software package or product. Objectives of the system analysis assist with planning systems where subsystems may have clashing destinations and enable comprehension of complicated structures. Likewise, System analysis with accomplishing bury similarity and solidarity of the subsystems. System analysis gives a favorable position and contrasts the subsystem's capacities and the complete system. Also, system analysis makes the software more flexible. This way, it can adapt to the changing business environment. It includes the functional and non-functional requirements and UML Diagrams. It is one of the significant stages in the growth of any system by detailing the needs that satisfy alterations in the future. Hope this makes you clear about every aspect of the system analysis, such as its definition, objectives, benefits, system analysis, design, and so on.

3.1 System Requirements:

System requirements are the configuration that a system must-have for a hardware or software application to run smoothly and efficiently. Failure to meet these requirements can result in installation problems or performance problems. System requirements are also known as minimum system requirements.

3.1.1 Functional Requirements:

Functional requirements are product features and focus on user requirements. Features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work.

Functional requirements of the system:

- Register User with Role:
  - Register and log-in with an admin or supervisor.
- Supervisor Actions:
  - View miners’ profiles.
  - View sensor readings.
  - Check the miner’s status via the app.
- Admin Actions:
  - The administrator adds the miners' data to the application, and it can be accessed or modified later by the application.
- Register Miner Data:
  - Save data about the name, age, health status, helmet ID and bracelet ID by admin.
- Miner Actions:
  - Wearing bracelet.

    - Wearing helmet
    - Asking for help if the helmet alert is on.
- Bracelet Functions:
    - The bracelet measures a miner's vital rates (Like: the Percentage of oxygen in the blood, Heart rate and Body temperature) and then sends them to the application.
    - Sending a warning message to the application if the miner is in an unstable state of health.
- Helmet Functions:
    - Detect gases.
    - Detect temperature and humidity.
    - Gives a warning of a helmet Collision or gas leakage.
    - Send a warning message to the application if the danger of toxic gases increases or if the miner is in an unstable state of health.
- Sensor configuration:
    - The sensors will be programmed by the embedded system at accurate rates and notifications will be sent to the app quickly so that we can detect toxic gases early.

3.1.2 Non-Functional Requirements:

Non-functional requirements are product properties and focus on user expectations. Nonfunctional requirements define system behavior, features, and general characteristics that affect the user experience. Therefore, it helps in the ease of use of the system, and here lies its importance. They are contrasted with functional requirements that define specific behavior or functions. The following points clarify them:

- Portability
  Systems running on one platform can easily be converted to running on another platform.
- Reliability
  The ability of the system to behave consistently in a user-acceptable manner when operating within the environment for which the system was intended.
- Availability
  The system should be always available.
- Maintainability
  A commercial database is used for maintaining the database and the application server takes care of the site.
- Security
  Secure access to confidential data (supervisor, miner information).
- User friendly
  The system should be easily used by the supervisor.
- Performance
  Performance should be fast.
- Efficient
  The system should be efficient so that it won't get hung if heavy traffic of data is placed.
- Safety

Data in the database of the system should not lose or damage.

- Privacy

  The Personal data of the system should not disclose to anyone.

3.2 UML Diagrams

3.2.1 Architecture design

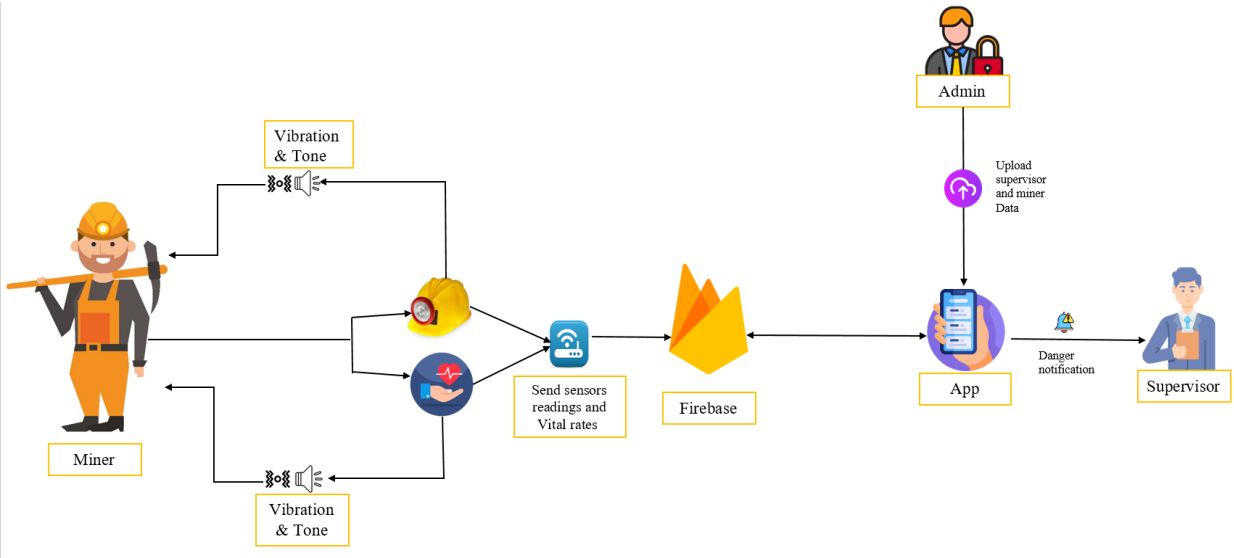


Figure 3-1 Architecture design

3.2.2 Use Case Diagram:

A use case diagram is a dynamic or behavior diagram in UML. That is a set of actions, services, and functions that the system needs to perform. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

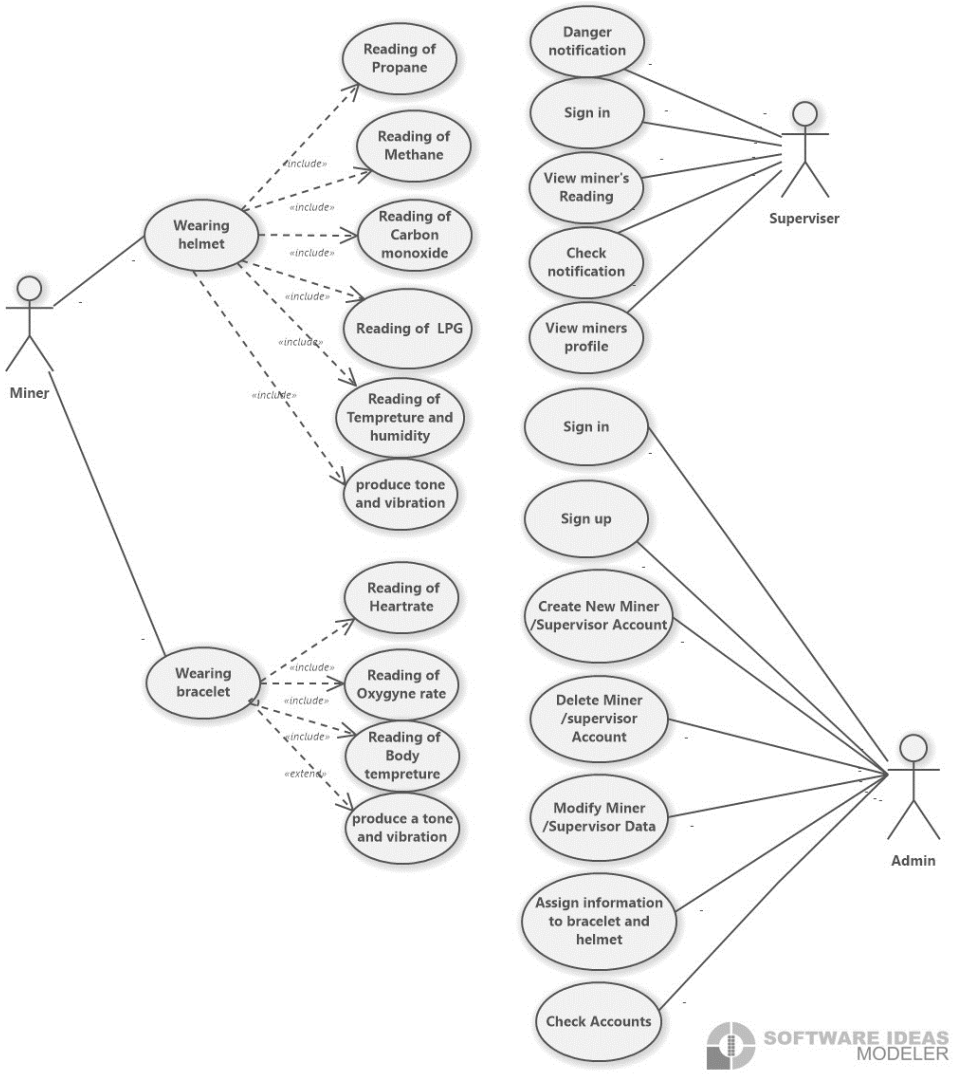


Figure 3-2 Use Case

The application use case diagram represents the total activities in the application. The user interacts with the application. We have three actors interacting with the application (Miner, Supervisor, and Admin).

The activities in applications are “sign up and check Accounts”. These actions are needed to log in to the application. Admin and Supervisor can do this, and their other activities assets in applications (Modify miners’ profile, View miner’s profile, and Check notifications). These features or activities can be accessed by the Admin and Supervisor (Miner can’t use the app).

<u>Miner use case Diagram:</u>

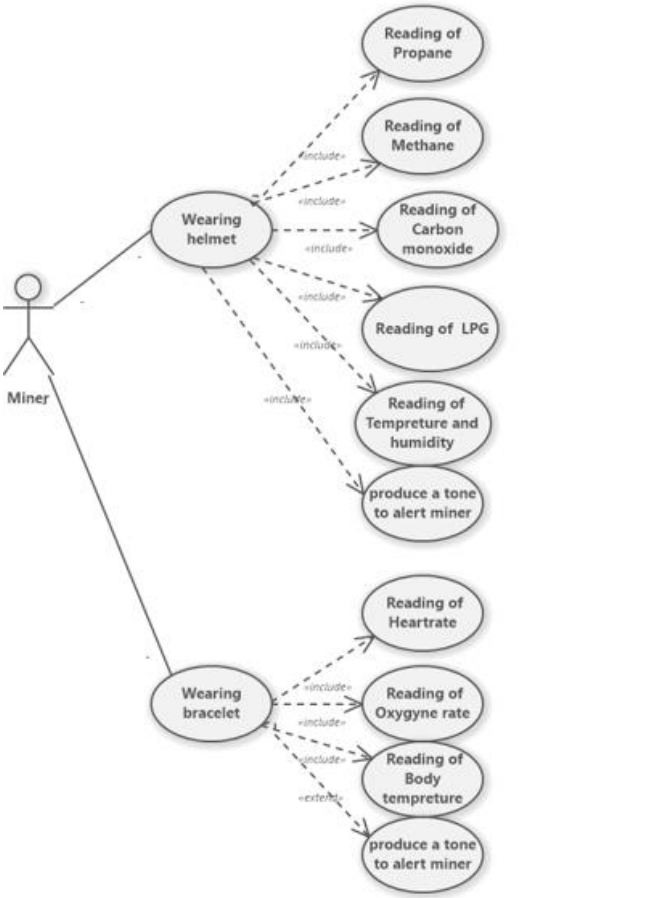


Figure 3-3 Miner use case Diagram

Miner use case diagram which represents the activities the miner can do in the system which are the miner wears helmet and bracelet which contain the following sensors: - MQ2, MQ5, MQ7, MQ9, DHT22, MAX30102 which can detect the surrounding gases and vital rates of each miner. (Produce a tone and vibration to alert miner) if there is a problem the buzzer produces a tone to alert miner.

Supervisor use case diagram:

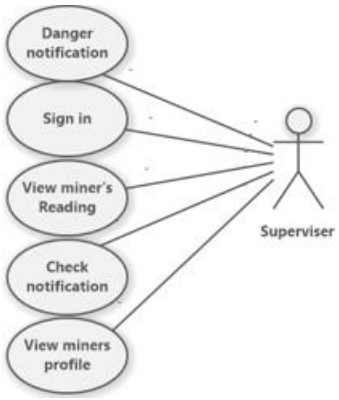


Figure 3-4 Supervisor use case Diagram

Supervisor use case diagram which represents the activities the supervisor can do in the application which are: -

(Sign in) used to enter the application, (view miners profile) which is a supervisor can view miners profile, (view miner's reading) supervisor can view reading of sensors and check vital rates of every miner, (check notification) a pop-up message generated by an application to notify the supervisor about the situation of every miner, (danger notification) a pop-up message generated by an application to notify the supervisor about the danger.

Admin use case diagram:

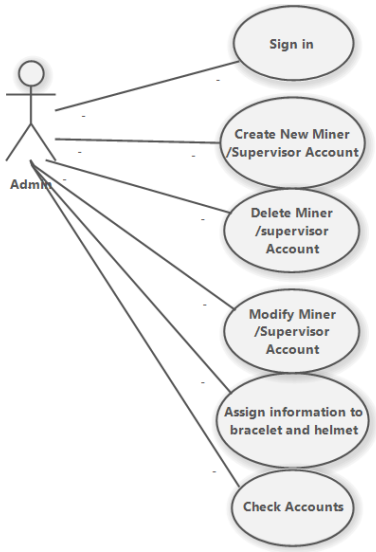


Figure 3-5 Admin use case Diagram

Admin use case diagram which represents the activities the admin can do in the application which are: -

(Sign in) used to enter the application, (create new miner/supervisor account) to add new miner/supervisor account, (delete miner/supervisor account) to remove miner/supervisor account, (assign information to bracelet and helmet) to assign information like ID or name to bracelet and helmet, (check accounts) which is an admin can view miners/supervisor profile.

3.2.3 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. Sequence diagram components:

• Actor: in a UML diagram represents a type of role where it interacts with the system and its objects

• Messages: communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline.

• Lifeline: is a named element which depicts an individual participant in a sequence diagram.

1. **User login:**

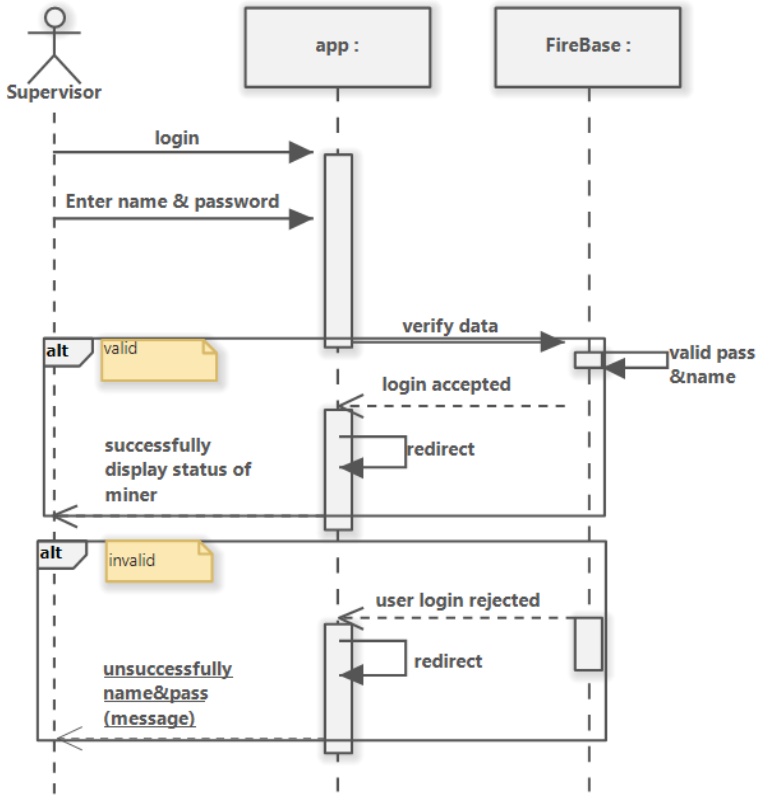


Figure 3-6 User login sequence Diagram

2. Helmet cycle

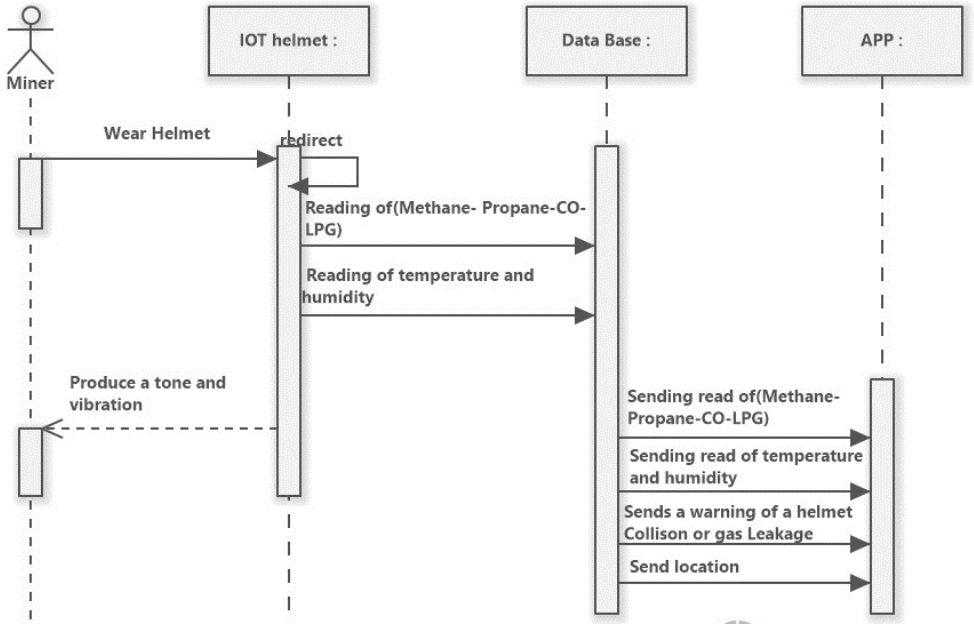


Figure 3-7 Helmet cycle sequence Diagram

3. Bracelet cycle

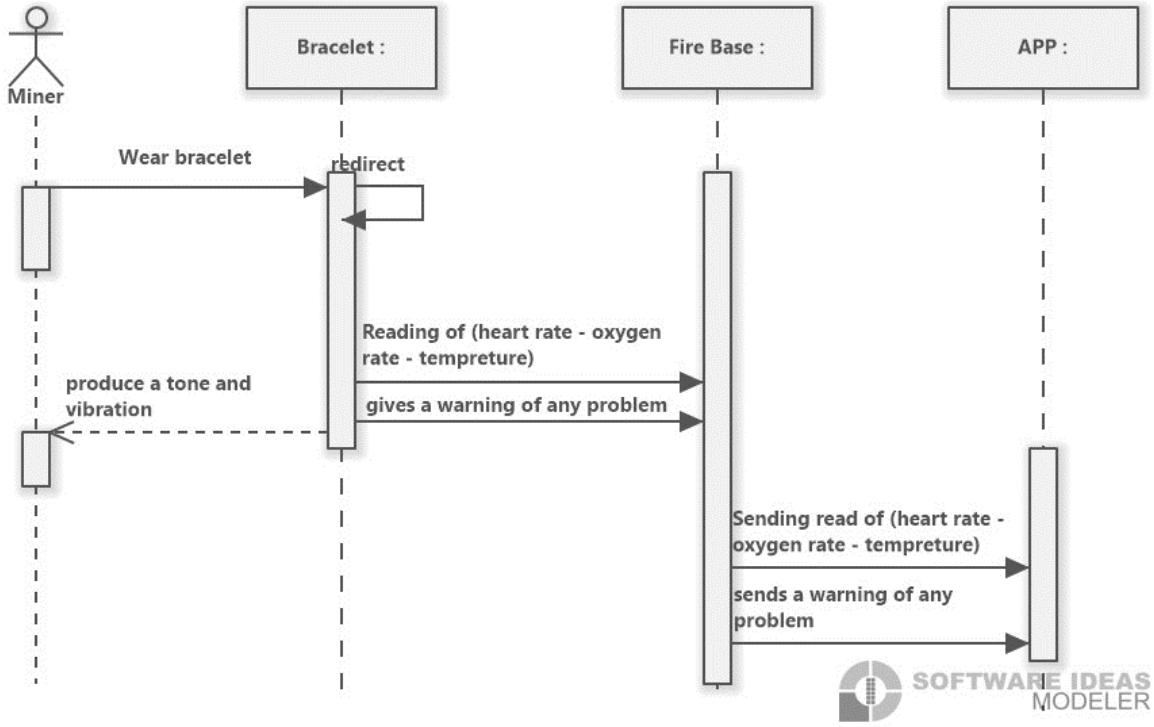


Figure 3-8 Bracelet cycle sequence Diagram

4. Sensor cycle

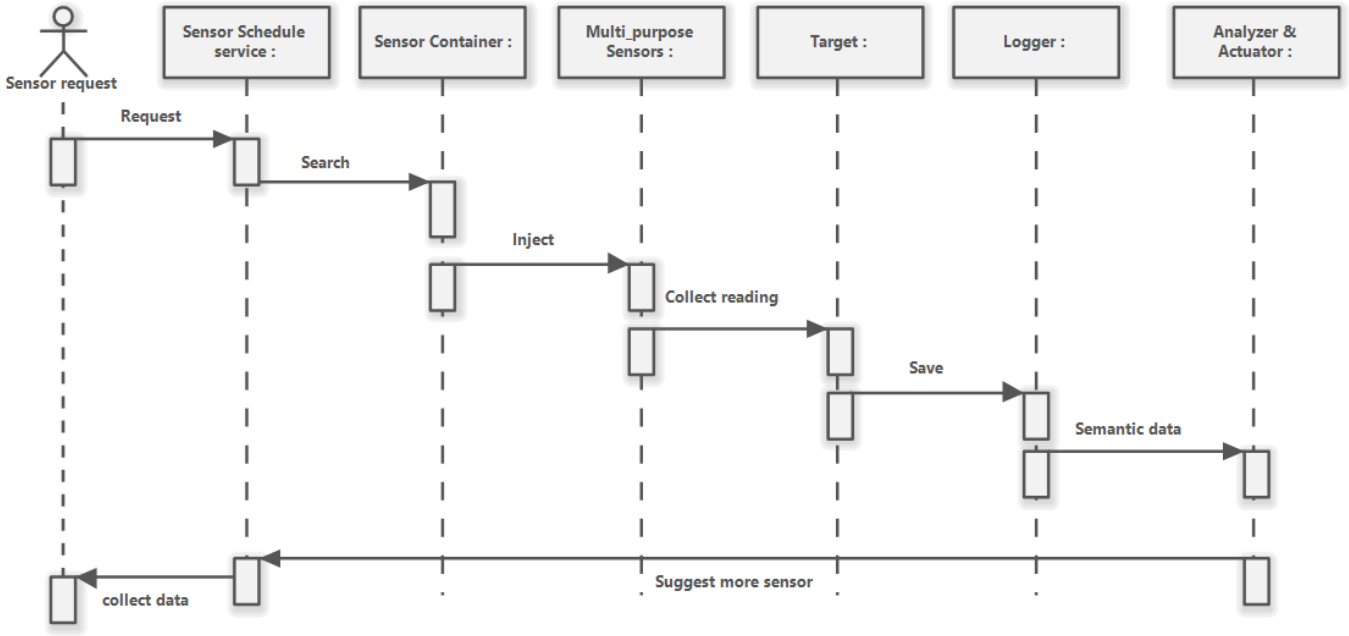


Figure 3-9 Sensor cycle sequence Diagram

5. Edit information of miner or supervisor

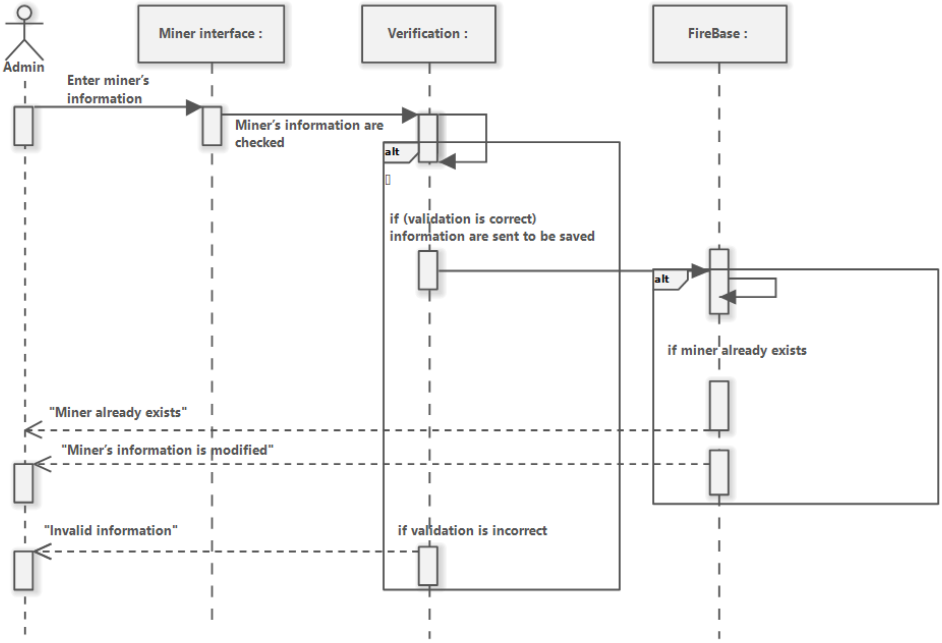


Figure 3-10 Edit information of miner or supervisor sequence Diagram

6. Delete miner information

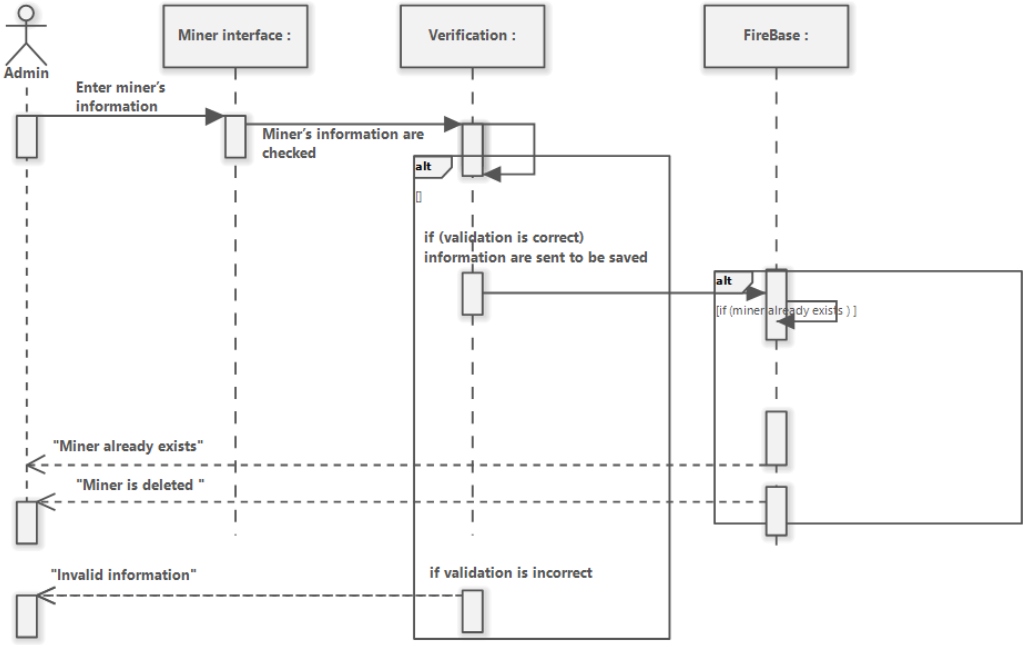


Figure 3-11 Delete miner information sequence Diagram

7. Add miner to the system

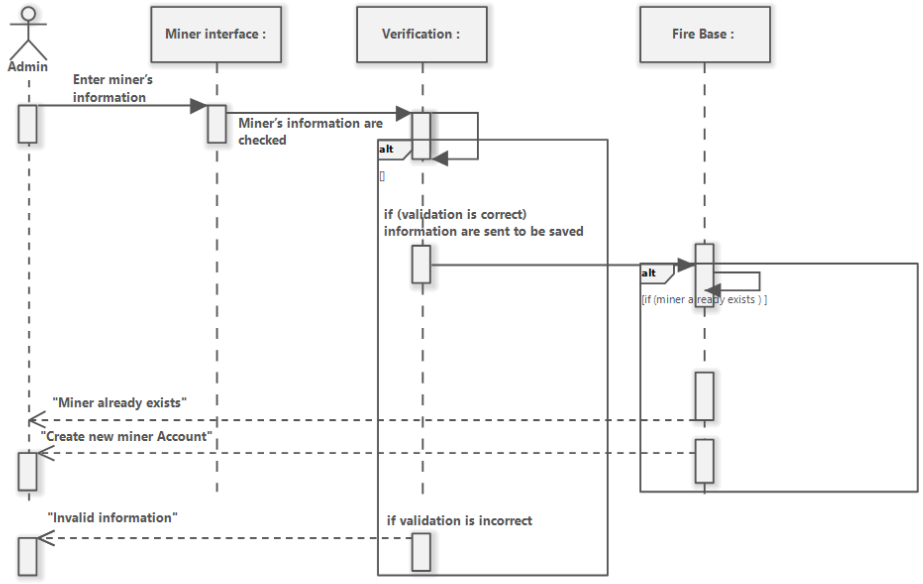


Figure 3-12 Add miner to the system sequence Diagram

3.3 business model

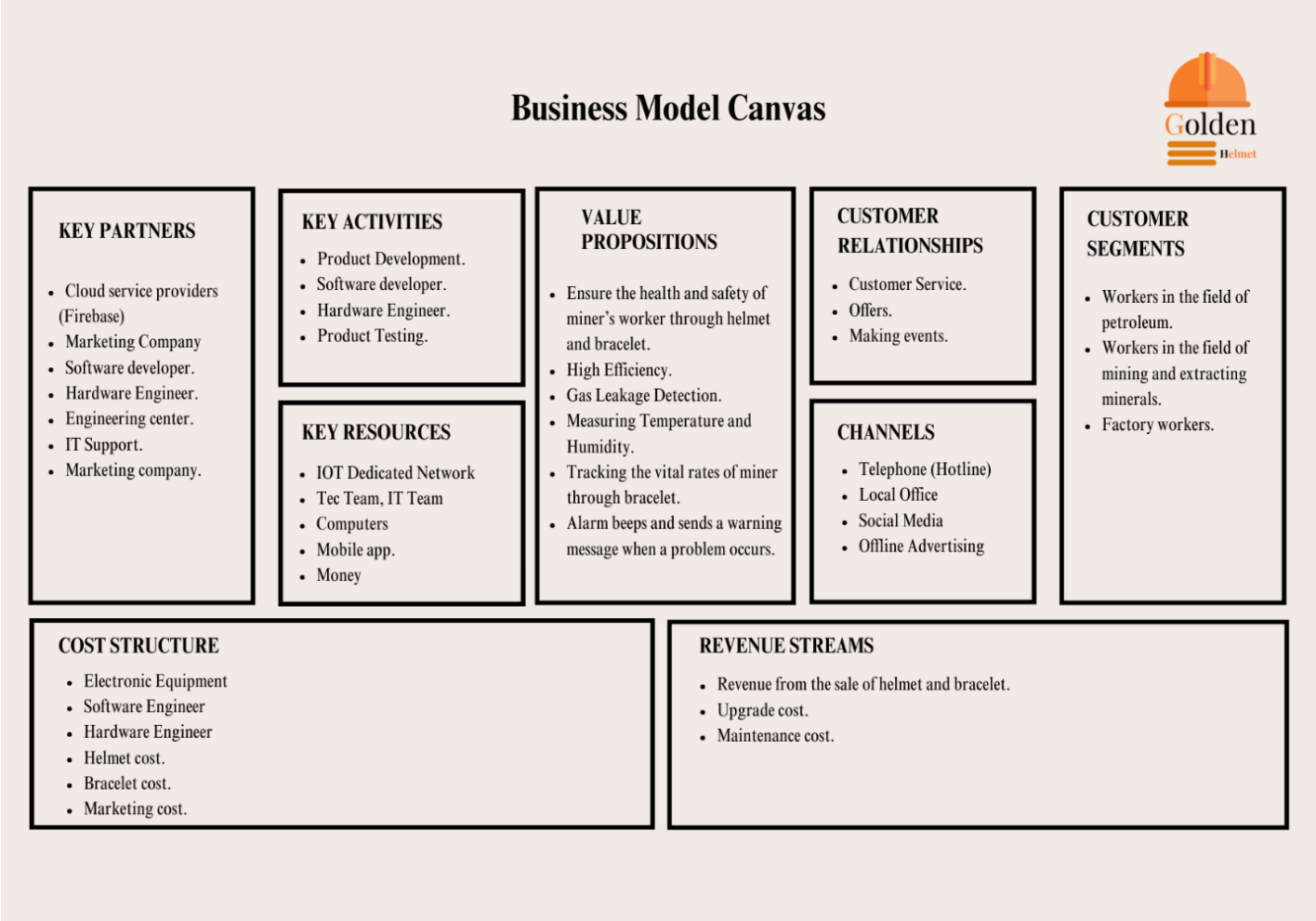


Figure 3-13 Business model

Chapter 4
System Design

Preamble

User interface (UI) is a part of Human-Computer Interaction (HCI). User interface (UI) focuses on the style and look of the project, it should have all communication means between users and the information system, windows can contain tools that make the interface more flexible such as data options, processing options, or both.

User interface elements:

User interface elements include but are not limited to:

- Input controls
- Navigational components
- Informational components
- Containers

In this chapter, we are going to present Golden Helmet screens and their detailed description.

4.2 Used Tools

Figma is a web-based design and prototyping tool that allows users to create and collaborate on digital designs for websites, apps, and other digital products. Figma's key features include a user-friendly interface, real-time collaboration, and the ability to create and share design components for consistent design across multiple pages or projects. With Figma, you can create wireframes, mockups, and high-fidelity designs, and easily share your work with team members or stakeholders for feedback and input. Figma is popular among designers because it is easy to use, flexible, and accessible from anywhere with an internet connection.

4.3 Interface Design

Splash Screen:

Figure 4-1 Splash Screen

(Also known as launch screen) provide a simple initial experience while your mobile app loads. They set the stage for your application, while allowing time for the app engine to load and your app to initialize.

Welcome screens:

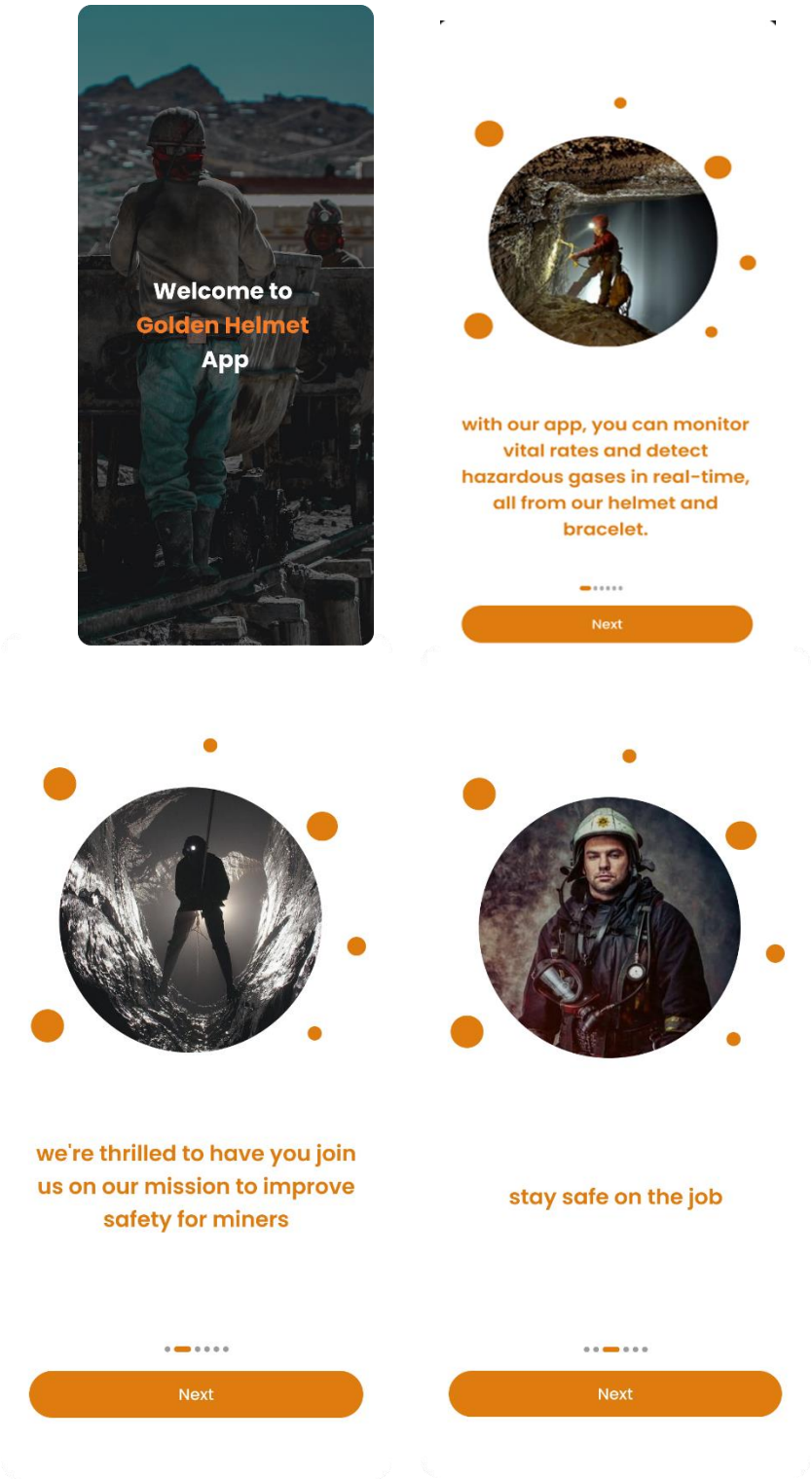


Figure 4-2 Welcome screens

Log-in as (supervisor or admin):

Figure 4-3 Sign-in as (supervisor or admin)

On this screen, you will specify if you will register as Supervisor or Admin. If you click on Supervisor, it will take you to the supervisor log in screen. If you click on Admin, it will take you to the Admin log in screen.

Sign-in Screens:

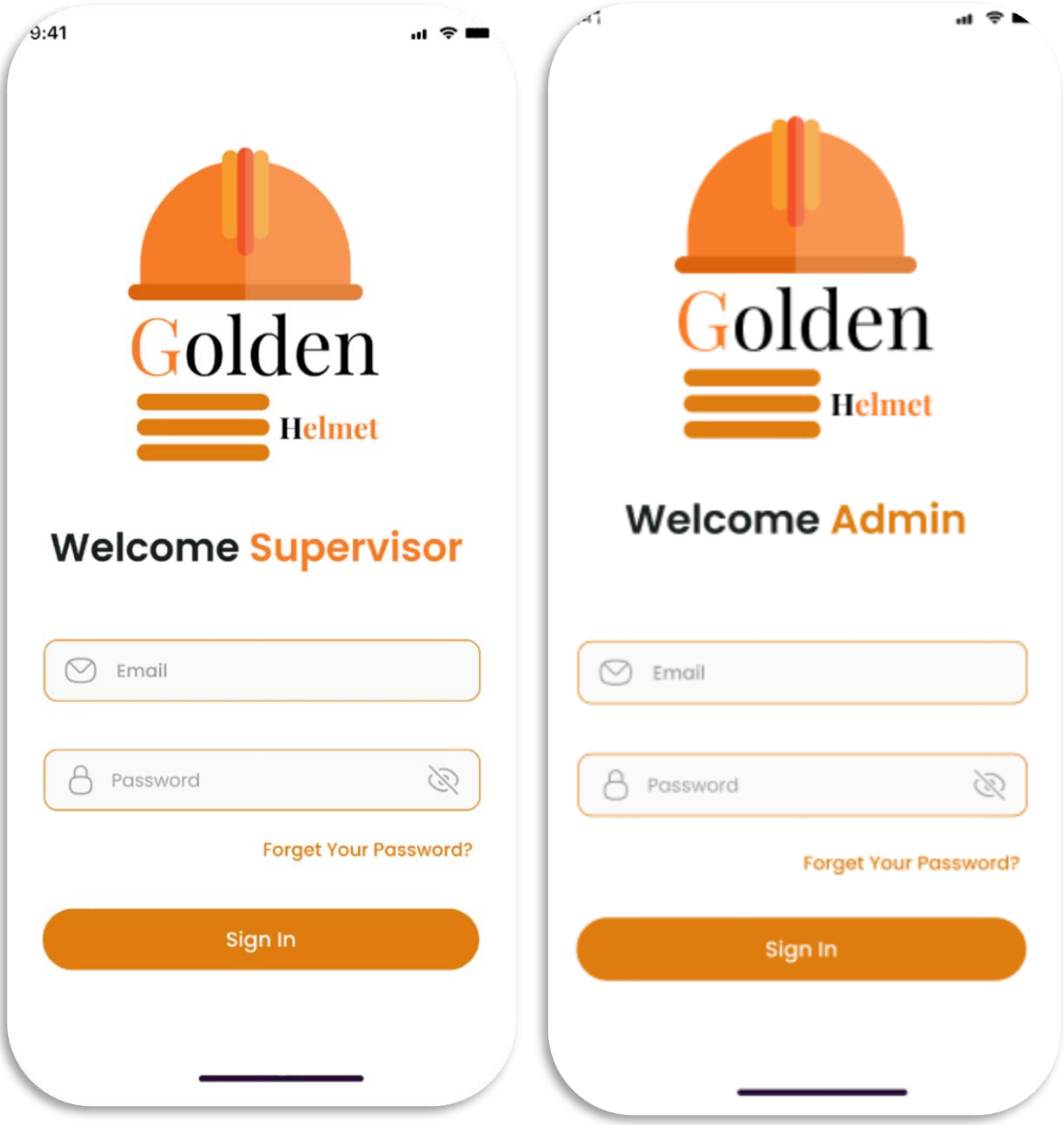


Figure 4-4 Supervisor or admin sign-in Screens

Sign-in screen (as supervisor and admin):

Supervisor or Admin enter his email and password to sign in

Miners Profile Screen:

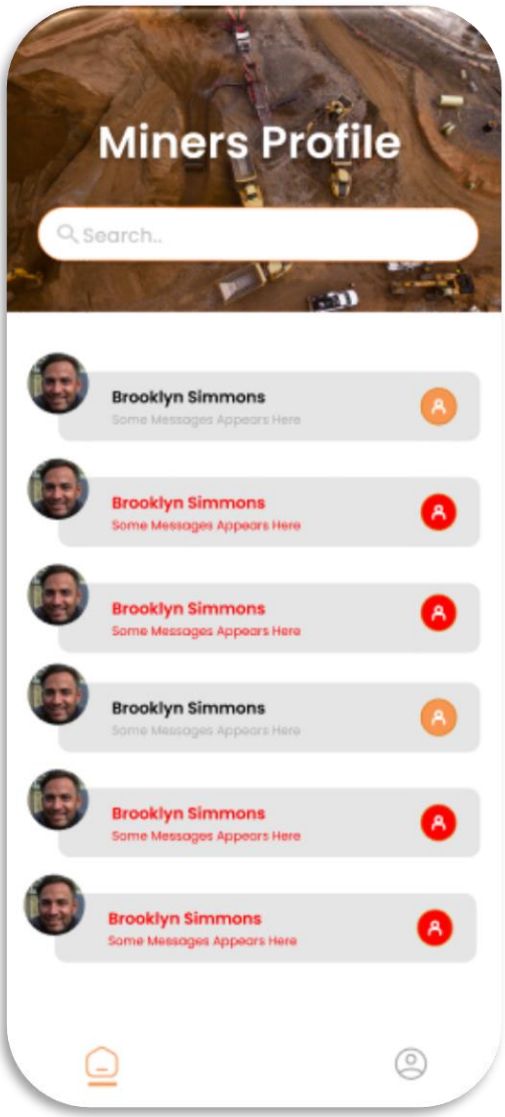


Figure 4- 5 Miners Profile Screen

Show a list of miners, and show if anyone in a danger, if supervisor click on a miner's name or profile picture, will take him to system control panel screen.

System control panel Screen:

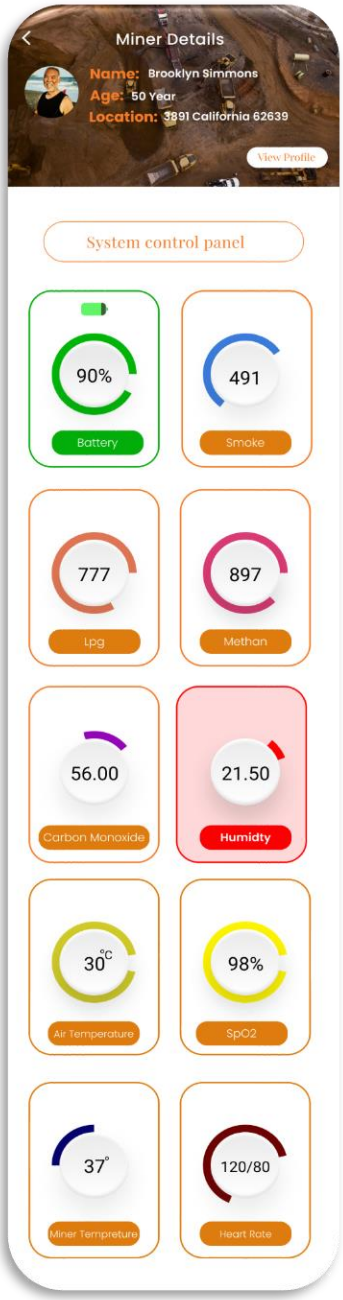


Figure 4-6 System control panel Screen

In this screen, the supervisor will track vital rates of miner, temperature, humidity and the percentage of gases in the air.

View profile Screen:

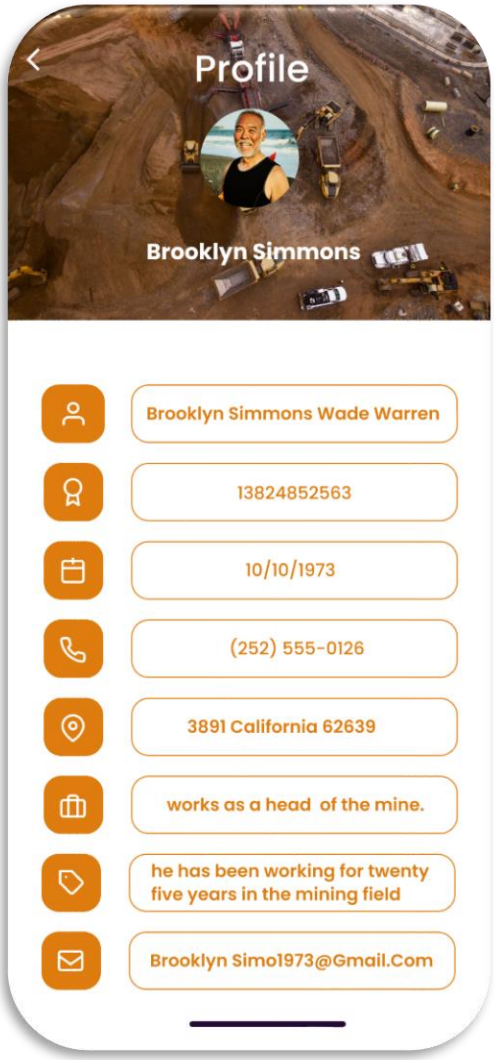


Figure 4-7 View profile Screen

If the supervisor clicks on the view profile button in the system control panel screen, it will take him directly to the profile screen. This page show more information about the miner such as name, id, birthdate, email.

Password Screens:

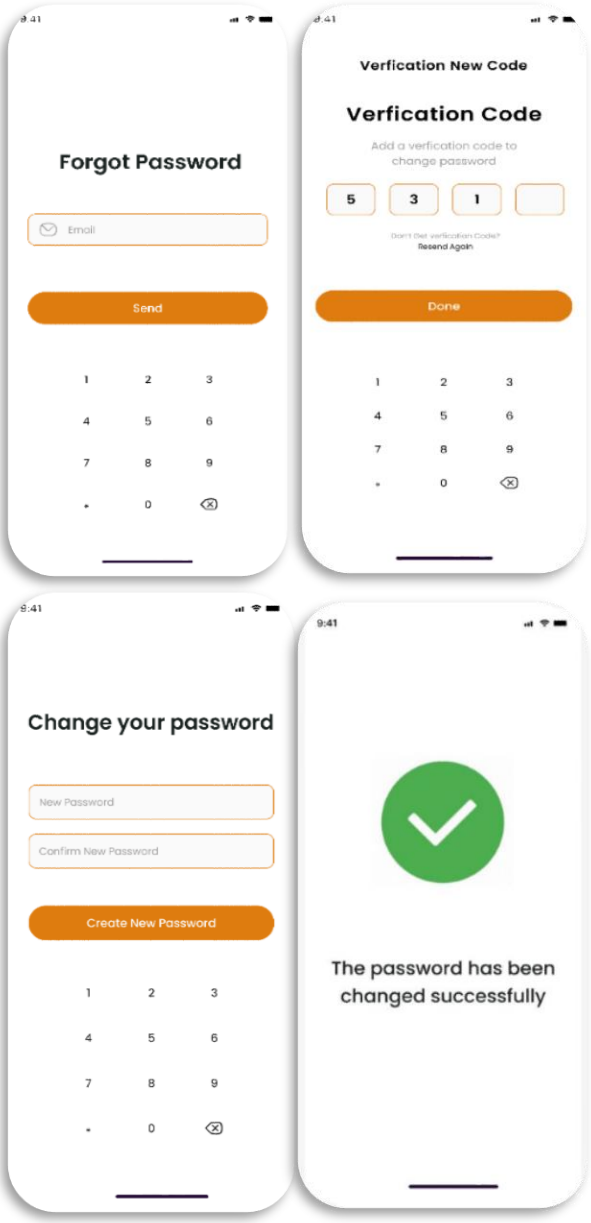


Figure 4-8 forget password screens

If the supervisor or admin clicks on the "forgot password" option, they will be directed to the first screen, where they can reset their password by entering their email. After entering their email, they will be directed to a screen where they will receive a verification code via email. Once they enter the correct verification code, they will be taken to a screen where they can change their password by entering a new one and confirming it. Finally, upon successfully changing their password, a confirmation screen will be displayed.

Admin control panel Screen

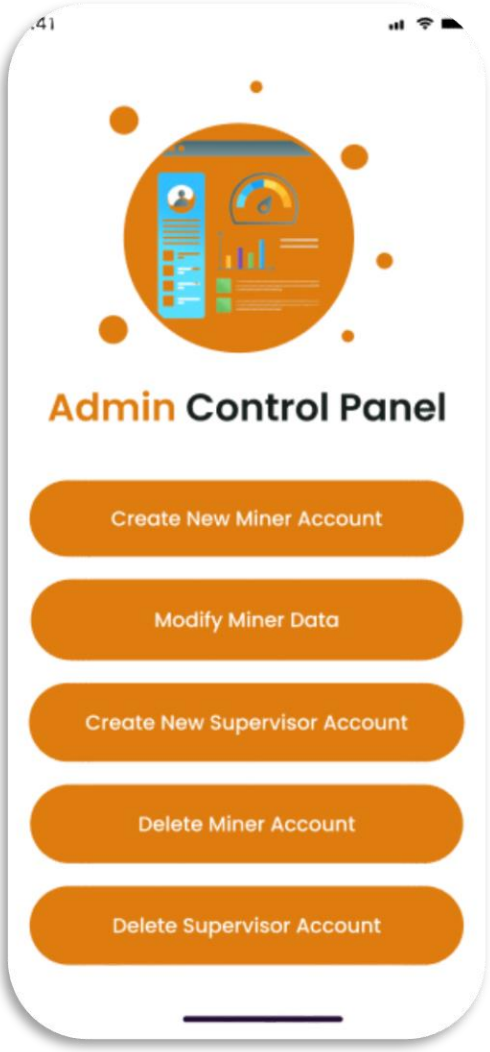


Figure 4-9 Admin control panel Screen

When you click to sign in as admin in figure 4.3 will take you to this screen. This page makes admin able to create new miner account, modify miner data, delete miner account or create new supervisor account.

Create new miner account Screen

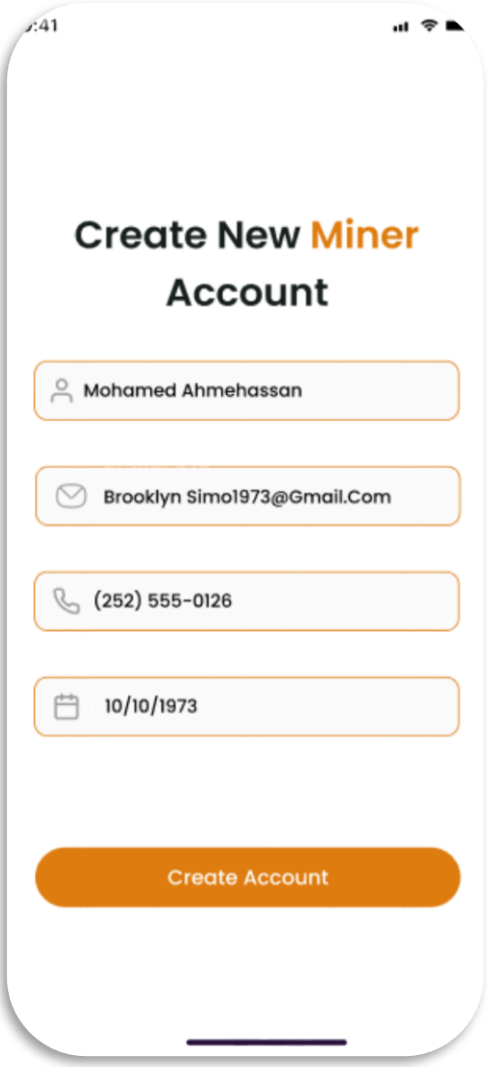


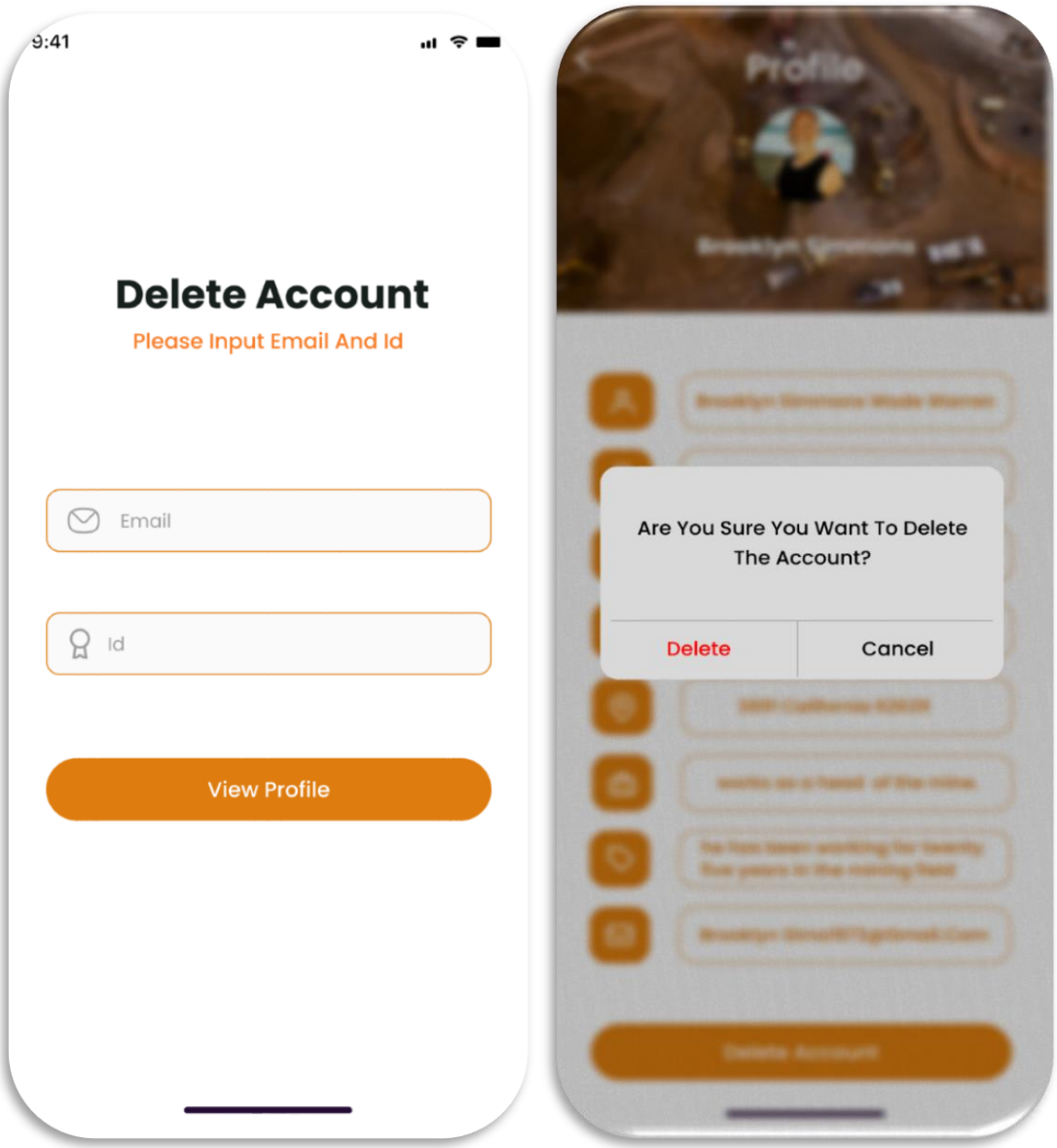
Figure 4-10 Create new miner account Screen

If admin click on create new miner account button in admin control panel will take him to this screen. Admin add new miner account by entering his name, email, phone and age.

Delete miner account Screen

Figure 4-3 Delete miner account Screen

If admin clicks on “Delete Miner Account” button in admin control panel, it will take him to this



screen. Admin can remove miner account by entering email and id. If admin click on delete button the account will be removed successfully.

Modify miner data screen:

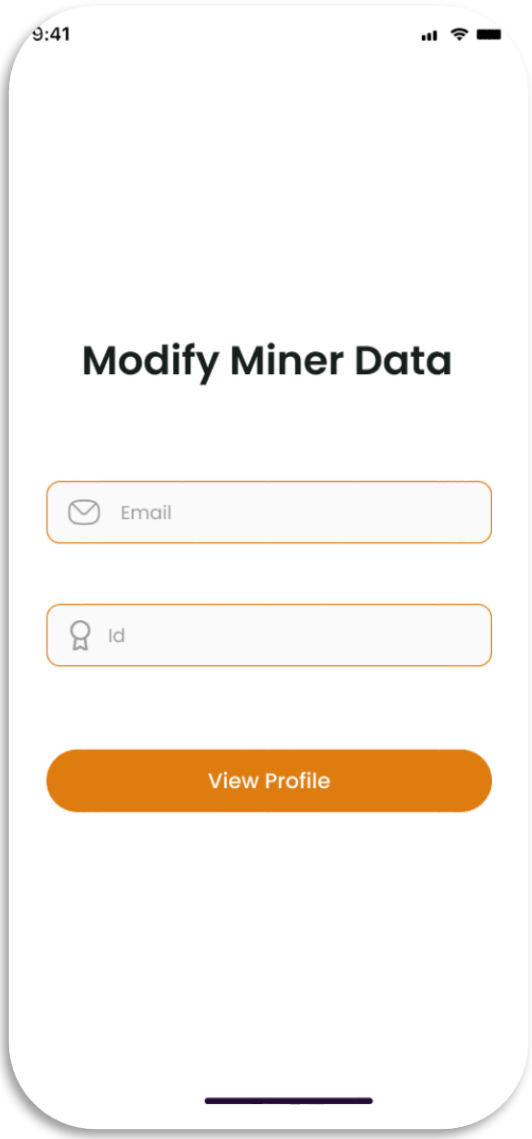


Figure 4-12 Modify miner data screen

If he clicks on “modify miner data account” button in admin control panel, it will take him to this screen. In this screen admin will put miner’s email and password to edit his data.

Profile screen

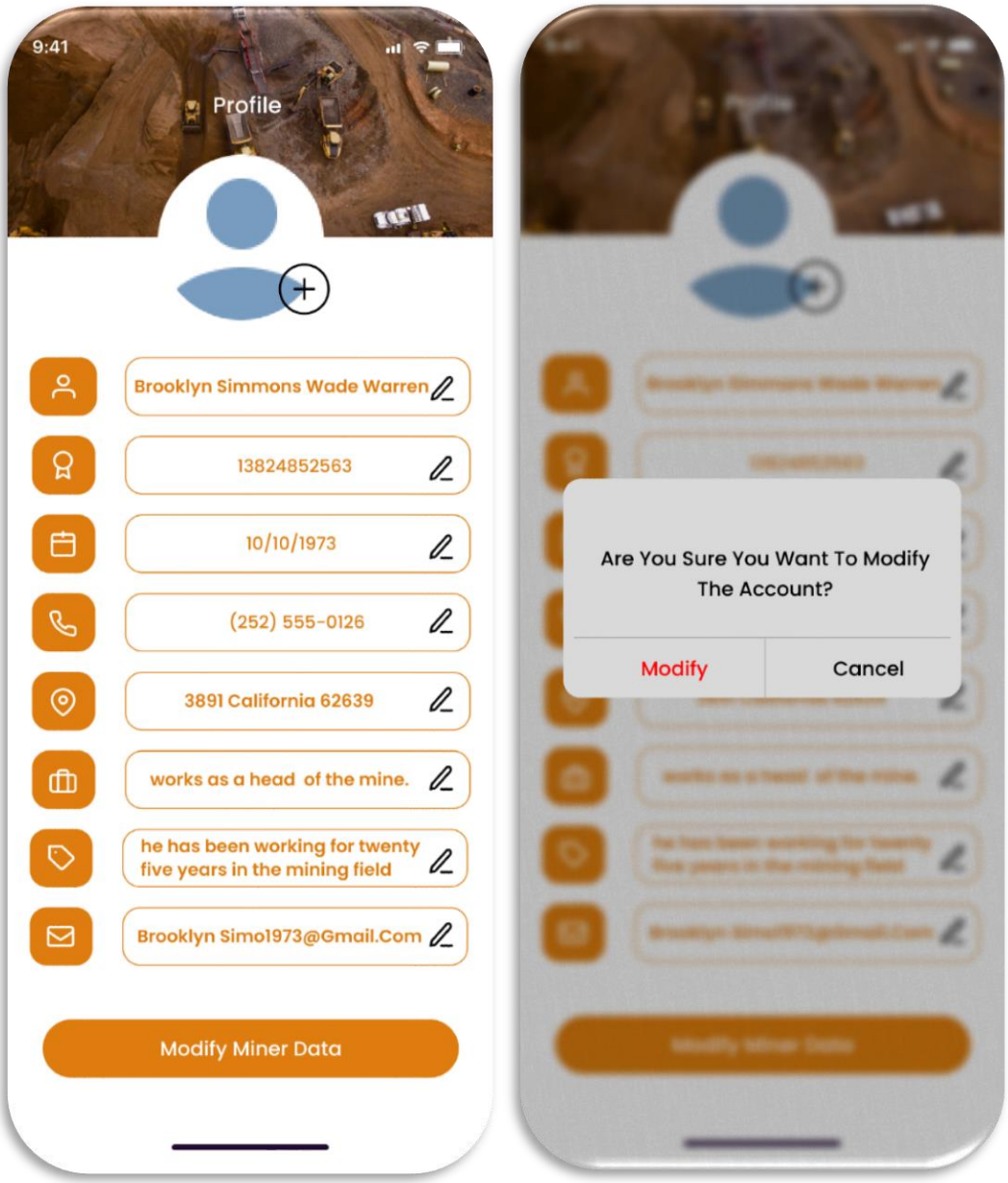


Figure 4-4 Profile screen

By clicking on view profile, it will take him to this screen .In this screen admin can edit miner's data. If he click on modify he confirm it.

Create new supervisor account Screen:

.

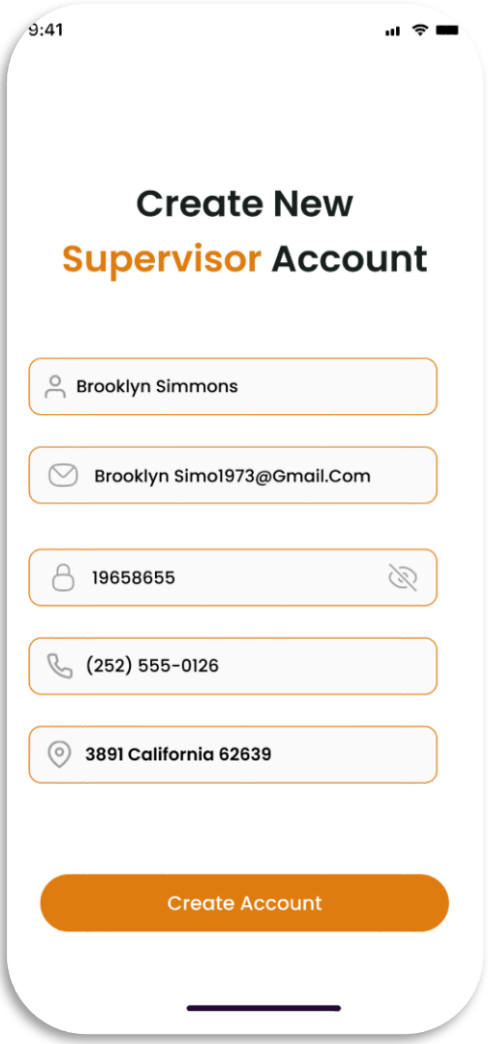


Figure 4-5 Create new supervisor account Screen

If he clicks on “create new supervisor account” button in admin control panel, it will take him to this screen Admin can create new supervisor account by entering information such as his name, email, and phone.

Chapter 5
System Implementation

5.1 Introduction

In previous chapters, we have demonstrated the potential requirements and limitations of our system. In this chapter, we provide an implementation tools, and provide a sample of the most important codes we used to build our prototype, and application the implementation process is divided into three parts which are hardware, firebase and mobile application.

5.2 Hardware

Consisting of a helmet and a bracelet, our hardware employs an Arduino Nano and an ESP8266 Wi-Fi module to send sensor data to Firebase in real-time.

5.2.1 Arduino nano:

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It has similar functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack and works with a Mini-B USB cable instead of a standard one. The Arduino Nano has several facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX).

Specifications:

- Microcontroller: ATmega328P – 8-bit AVR family microcontroller
- Operating Voltage:5V
- Recommended Input Voltage for Vin pin:7-12V
- Analog Input Pins:6 (A0 – A5)
- Digital I/O Pins:14 (Out of which 6 provide PWM output)
- DC Current on I/O Pins:40 mA
- DC Current on 3.3V Pin:50 mA
- Flash Memory :32 KB (2 KB is used for Boot loader)
- SRAM:2 KB
- EEPROM:1 KB
- Frequency (Clock Speed):16 MHz
- Communication: SPI, USART

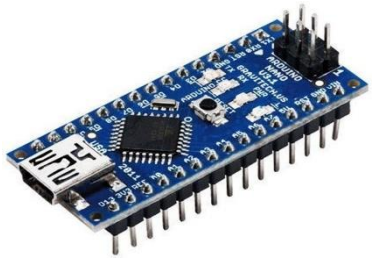


Figure 5-1 Arduino nano

5.2.2 ESP8266 Wi-Fi Module:

The ESP8266 is a low-cost Wi-Fi module that is widely used in a variety of projects and devices. These specifications make the ESP8266 a highly capable Wi-Fi module that can be used in a wide range of applications. It supports multiple Wi-Fi modes, including station, softAP, and station+softAP, which makes it suitable for both connecting to existing Wi-Fi networks and creating new Wi-Fi networks. The module also includes built-in TCP/IP protocol stack, which simplifies network communication and software development. ESP8266 module has a 32-bit microcontroller, 17 GPIO pins, and support for I2C, SPI, and UART interfaces, which makes it suitable for controlling various devices and sensors. It also has 4MB or 8MB of flash memory, which provides ample storage space for firmware and data.

Technical Specifications:

- Processor: Ten silica L106 32-bit microcontroller.
- Operating voltage: 3.0V to 3.6V.
- Wi-Fi protocol: 802.11 b/g/n.
- Wi-Fi modes: Station, SoftAP, and Station+SoftAP.
- Data rates: 110-460800 bps.
- Security protocols: WPA/WPA2, WEP, TKIP, AES.
- TCP/IP stack: Built-in TCP/IP protocol stack.
- GPIO pins: 17 GPIO pins, including GPIO16 used for wake-up from deep sleep mode.
- Analog-to-digital converter (ADC): 1 input channel with 10-bit resolution.
- I2C, SPI, and UART interfaces.
- Flash memory: 4MB or 8MB.
- Operating temperature range: -40°C to 125°C.

How it works:

The ESP8266 Wi-Fi module works by connecting to a Wi-Fi network and exchanging data with other devices over the network. It can operate in three different modes: Station mode, SoftAP mode, and Station+SoftAP mode.

In Station mode, the ESP8266 connects to an existing Wi-Fi network as a client. It can then send and receive data over the network using TCP/IP or UDP protocols. This mode is commonly used to add Wi-Fi connectivity to devices that do not have built-in Wi-Fi capabilities, such as microcontrollers like the Arduino.

In SoftAP mode, the ESP8266 acts as an access point and creates its own Wi-Fi network. Other devices can then connect to this network and exchange data with the ESP8266. This mode is useful for creating Wi-Fi hotspots or for creating ad-hoc networks between devices.

In Station+SoftAP mode, the ESP8266 can act as both a client and an access point simultaneously. This allows it to connect to an existing Wi-Fi network while also creating its own network for other devices to connect to.

The ESP8266 Wi-Fi module communicates with other devices over a variety of interfaces, including UART, SPI, and I2C. It can be programmed using standard programming languages such as C and C++, as well as using dedicated firmware such as the Arduino IDE or the NodeMCU Lua firmware.

The ESP8266 module also includes a built-in TCP/IP protocol stack, which simplifies network communication and software development. It supports multiple security protocols, including WPA/WPA2, WEP, TKIP, and AES, which helps to ensure the security of data transmitted over the network.

ESP8266 is a powerful and versatile Wi-Fi module that can add wireless connectivity to a wide range of projects and devices. Its affordable price, ease of use, and compatibility with popular programming languages and firmware make it a popular choice for hobbyists, makers, and developers.

Figure 5-2 ESP8266 Wi-Fi Module

5.2.3 Helmet Sensors:

MQ2

Works on 5V DC and draws around 800mW. It can detect LPG, Smoke, Alcohol, Propane,

Hydrogen, Methane and Carbon Monoxide concentrations anywhere from 200 to 10000ppm.

Technical Specifications:

- Operating voltage: 5V.
- Load resistance: 20 KΩ.
- Heater resistance: 33Ω ± 5%.
- Heating consumption: <800mw.
- Sensing Resistance: 10 KΩ – 60 KΩ.
- Concentration Range: 200 – 10000ppm.
- Preheat Time: Over 24 hours.

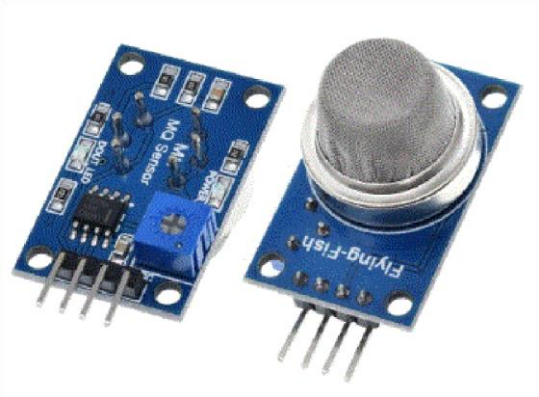


Figure 5-3 MQ2 sensor

MQ5

The sensitive material of MQ-5 gas sensor is SnO2, which with lower conductivity in clean air. When the target combustible gas exists, the sensors conductivity is higher along with the gas concentration rising. The sensor converts change of conductivity to correspond output signal of gas concentration.

MQ-5 gas sensor has high sensitivity to Methane, Propane and Butane, ch4 and natural gas.

Technical Specifications:

- Operating Voltage (VDC):5**.**
- Operating Temperature (°C): -20 to +40.
- Length (mm):32.
- Width (mm):20.
- Height (mm):22.

- Weight (gm): 7.

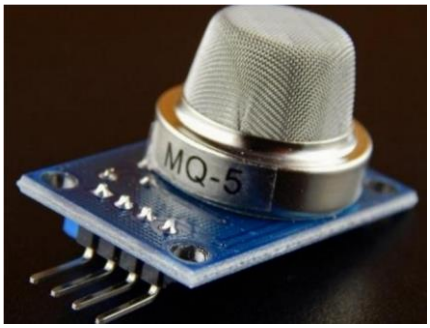


Figure 5-4 MQ5 sensor

MQ7

The MQ7 is a simple-to-use Carbon Monoxide (CO) sensor suitable for sensing CO concentrations in the air. It can detect CO-gas concentrations anywhere from 20 to 2000ppm.

Technical Specifications:

- Operating voltage (v): DC 5.
- Range: 10 ~ 1000 ppm.
- Characteristic gas: 100 ppm CO.
- Sensitivity: ≥ 3%.
- Return time: ≤ 30 sec.
- Heating resistance: ± 31 Ω.
- Heating current: ≤ 180 mA.
- Heating voltage: 5.0V ± 2V / 1.5 ± 1V.
- Heating power approx.: 350 mW.
- Ambient temperature (°C): -20 ~ + 50.
- Humidity: ≤ 95% RH.

Figure 5-5 MQ7 sensor

MQ9

MQ-9 gas sensor has high sensitivity to Carbon Monoxide, Methane and LPG. The sensor could be used to detect different gases containing CO and combustible gases, it is with low cost and suitable for different application.

Technical Specifications:

- Operating Voltage (VDC): 5.
- Current Consumption (mA): 150.
- Do output: TTL digital 0 and 1 (0.1 and 5V).
- A0 output: 0.1-0.3V (relatively clean).
- Length (mm): 32.

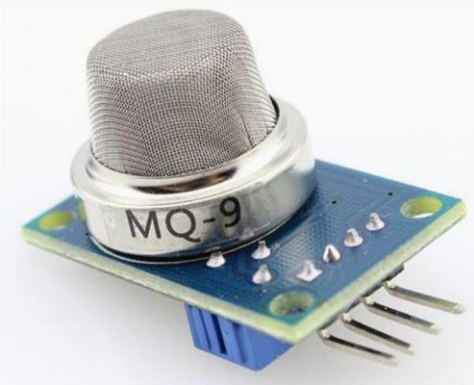


Figure 5-6 MQ9 sensor

How MQ Sensors work:

The MQ (standing for "Metal Oxide") gas sensors are a type of semiconductor gas sensor that detects various gases in the air by measuring changes in the electrical resistance of its metal oxide semiconductor material when it encounters gases.

The MQ sensors are typically composed of a sensing element, a heater element, and a circuit board that processes the sensor output. The sensing element is made of a metal oxide semiconductor material that is sensitive to certain gases, and it is heated by the heater element to a high temperature (usually between 200-400°C) to enable it to react with the gases in the air.

When the sensing element encounters a gas, the gas molecules are adsorbed onto the surface of the semiconductor material, causing a change in its electrical resistance. This change in resistance is then amplified and processed by the circuit board, which converts it into a measurable output signal.

Different gases have different chemical properties and thus affect the sensor's electrical resistance in different ways. MQ sensors are available that are designed to detect a wide range of gases such as carbon monoxide, methane, propane, alcohol, and many others.

The output signal from the MQ sensor is usually an analog voltage signal that can be measured by a microcontroller or other electronic device. The voltage output is proportional to the concentration of the gas being detected. By measuring the voltage output of the sensor, it is possible to determine the concentration of the gas in the air.

DHT11

The DHT11 is a commonly used Temperature and humidity sensor that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

Technical Specifications:

- Operating Voltage: 3.5V to 5.5V.
- Operating current: 0.3mA (measuring) 60uA (standby).
- Output: Serial data.
- Temperature Range: 0°C to 50°C.
- Humidity Range: 20% to 90%.
- Resolution: Temperature and Humidity both are 16-bit.
- Accuracy: ±1°C and ±1%.

Figure 5-7 DHT11 sensor

DC 3-24V Active Electronic Buzzer:

In simplest terms, a piezo buzzer is a type of electronic device that's used to produce a tone, alarm, or sound. It's lightweight with a simple construction, and it's typically a low-cost product.

Technical Specification:

- Alarm Diameter: ~30mm.
- Alarm Height: 15mm.
- 2 Mounting Holes distance: 40mm.

- 2 Wires length: 105mm / 4.13″.
- Rated Voltage: 12V.
- Operating Voltage: 3-24V.
- Rated Current (MAX): 20mA.
- Min Sound Output at 10 cm: 95 DB.
- Resonant Frequency: 3100±500.
- Operating Temperature: -20～+80℃.
- Continuous Sound.
- Mounting Holes.

Figure 5-8 DC 3-24V Active Electronic Buzzer

How it works:
A buzzer is an electronic device that generates a continuous or intermittent sound when it is activated. When powered with 12V, a buzzer can produce a louder sound than a buzzer powered with lower voltage.
A buzzer typically consists of a piezoelectric ceramic element, a driver circuit, and a housing. When a voltage is applied to the piezoelectric element, it vibrates at a high frequency, producing an audible sound. The driver circuit is used to generate the voltage signal that drives the piezoelectric element at the desired frequency and intensity.
The piezoelectric element in the buzzer is made of a special type of ceramic material that generates an electric signal when it is subjected to mechanical stress. When an alternating voltage is applied to the piezoelectric element, it causes it to expand and contract rapidly, which creates a corresponding mechanical vibration in the surrounding air. This vibration produces a sound wave that is heard as a buzzer sound.
Overall, a buzzer powered with 12V works by converting an electrical signal into a mechanical vibration, which produces an audible sound that can be used for various applications such as alarms, notifications, and sound effects.

Mini Vibration motor DC 3-6V:

Features Very small electric consumption, High efficiency.

Technical Specification:

- Motor model: 1220 motor

- Motor size: 12*20mm
- Weight: 15g
- Suitable voltage: 3-6V

How it works:

A micro mini vibration motor is a small DC motor that is designed to vibrate when it is powered on. These motors are commonly used in electronic devices such as mobile phones, game controllers, and wearable devices to provide haptic feedback. The micro mini vibration motor typically consists of a cylindrical housing, a motor with an eccentric weight attached to its rotor shaft, and two electrical contacts for providing power to the motor. When power is applied to the motor, the eccentric weight rotates with the shaft, causing the motor and the housing to vibrate.

The motor operates on the principle of electromagnetic induction, where a magnetic field is generated when an electric current flow through a coil of wire. In a DC motor, this magnetic field interacts with a permanent magnet located inside the motor, causing the motor to rotate. The eccentric weight attached to the rotor shaft creates an unbalanced force, which causes the motor and the housing to vibrate when the motor rotates.

The micro mini vibration motor is typically powered by a DC voltage of 3 volts, and the vibration intensity can be controlled by adjusting the voltage level. The higher the voltage, the faster the motor rotates and the more intense the vibration.

Figure 5-9 Mini Vibration motor DC 3-6V

Steps for making helmet

1- Connect the MQ2, MQ5, MQ7, and MQ9 gas sensors: Each of these sensors has four pins - VCC, GND, AOUT, and DOUT. Connect the VCC pins of all the sensors to the 5V pin of the Arduino Nano board, and connect the GND pins of all the sensors to the GND pin of the Arduino Nano board. Connect the AOUT pins of the sensors to analog input pins on the Arduino board, and connect the DOUT pins of the sensors to digital input pins on the Arduino board.

2- Connect the DHT11 sensor: The DHT11 sensor has three pins - VCC, GND, and DATA. Connect the VCC pin of the DHT11 to the 5V pin of the Arduino Nano board, and connect the GND pin of the DHT11 to the GND pin of the Arduino Nano board. Connect the DATA pin of the DHT11 to a digital input/output pin on the Arduino board.
3- Connect the mini vibration motor: The mini vibration motor has two pins - VCC and GND. Connect the VCC pin of the motor to a digital output pin on the Arduino board, and connect the GND pin of the motor to the GND pin of the Arduino board.
4- Connect the buzzer: The buzzer also has two pins - VCC and GND. Connect the VCC pin of the buzzer to a digital output pin on the Arduino board, and connect the GND pin of the buzzer to the GND pin of the Arduino board.
5- Connect the ESP8266 module: Connect the ESP8266 module to the Arduino Nano board using a serial communication interface such as UART. You will need to connect the TX pin of the ESP8266 module to the RX pin of the Arduino Nano board, and the RX pin of the ESP8266 module to the TX pin of the Arduino Nano board. You will also need to connect the VCC and GND pins of the ESP8266 module to the 3.3V and GND pins of the Arduino Nano board, respectively.

5.2.4 Bracelet Sensors:

Max30102

The module features the MAX30102 – a modern (the successor to the MAX30100), integrated pulse oximeter and heart rate sensor IC, from Analog Devices. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry (SpO2) and heart rate (HR) signals.

Technical Specifications:

- Power supply voltage range: 1.8V to 5.5V.
- Operating current: 1.8mA.
- LED peak current: 50mA.
- LED current range: 0 to 50mA.
- Photodiode range: 0.4nA to 16mA.
- Sampling rate: up to 3200 samples per second (SPS).
- Integrated 16-bit ADC.
- I2C interface with programmable interrupt and FIFO.
- Algorithm for heart rate and SpO2 calculation.
- Operating temperature range: -40°C to 85°C.
- Package size: 5.6mm x 3.3mm x 1.55mm.

How MAX 30102 works:

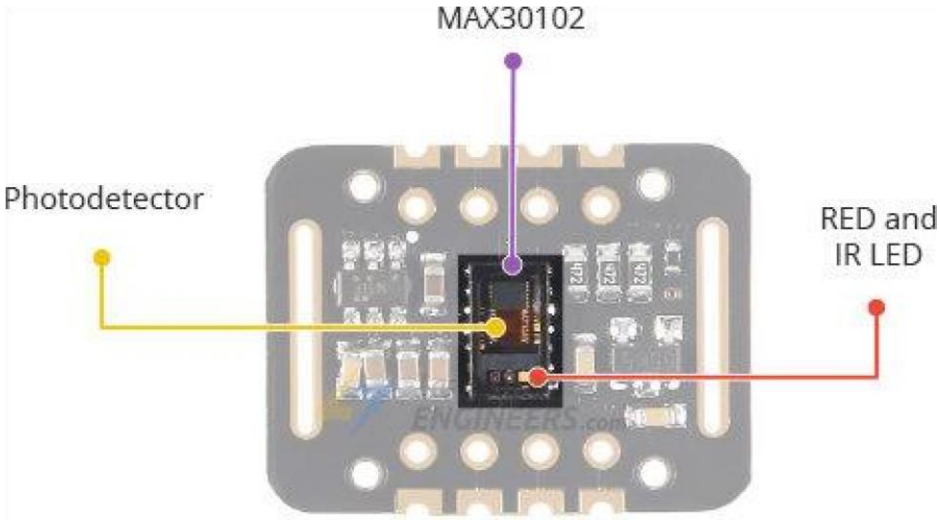


Figure 5-10 LEDs of ESP32

Behind the window on one side, the MAX30102 has two LEDs – a RED and an IR LED. On the other side is a very sensitive photodetector. The idea is that you shine a single LED at a time, detecting the amount of light shining back at the detector, and based on the signature, you can measure blood oxygen level and heart rate.

Power Requirement:

The MAX30102 chip requires two different supply voltages: 1.8V for the IC and 3.3V for the RED and IR LEDs. So, the module comes with 3.3V and 1.8V regulators.

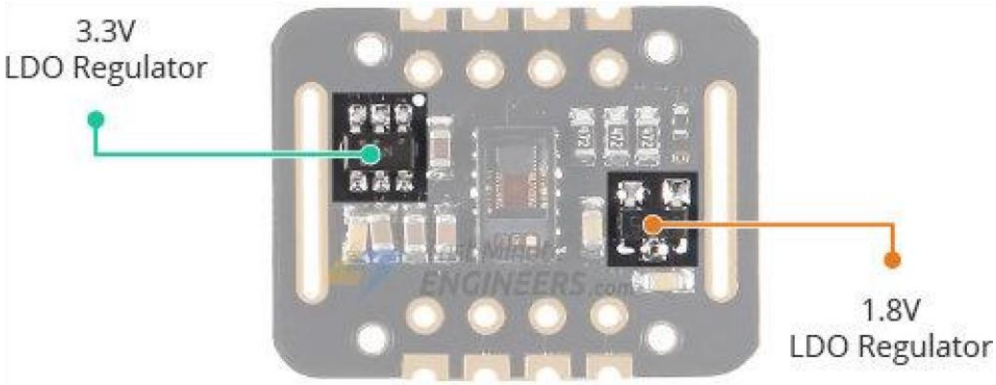


Figure 5-11MAX30102 Regulators

On the back of the PCB, you'll find a solder jumper that can be used to select between 3.3V and

1.8V logic level. By default, 3.3V logic level is selected which is compatible with logic levels for Arduino. But you can also select 1.8V logic level as per your requirement. This allows you to connect the module to any microcontroller with 5V, 3.3V, even 1.8V level I/O.

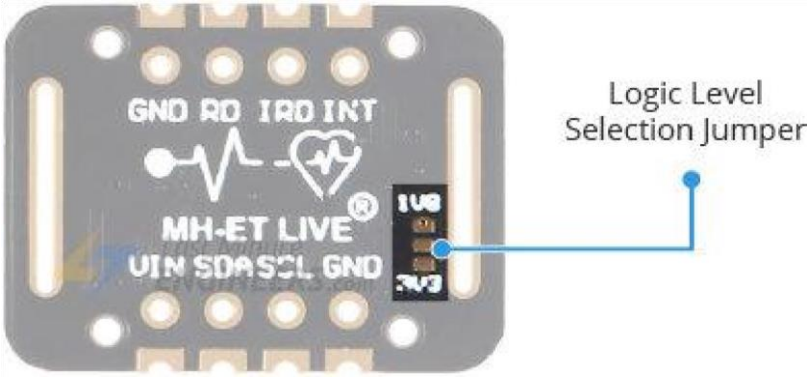


Figure 5-12 Solder jumper

One of the most important features of the MAX30102 is its low power consumption: the MAX30102 consumes less than 600µA during measurement. Also, it is possible to put the MAX30102 in standby mode, where it consumes only 0.7µA. This low power consumption allows implementation in battery-powered devices such as handsets, wearable, or smart watches.

On-Chip Temperature Sensor

The MAX30102 has an on-chip temperature sensor that can be used to compensate for the changes in the environment and to calibrate the measurements.

This is a reasonably precise temperature sensor that measures the 'die temperature' in the range of -

40˚C to +85˚C with an accuracy of ±1˚C

How MAX30102 Pulse Oximeter and Heart Rate Sensor Works:

The MAX30102, or any optical pulse oximeter and heart-rate sensor for that matter, consists of a pair of high-intensity LEDs (RED and IR, both of different wavelengths) and a photodetector. The wavelengths of these LEDs are 660nm and 880nm, respectively.

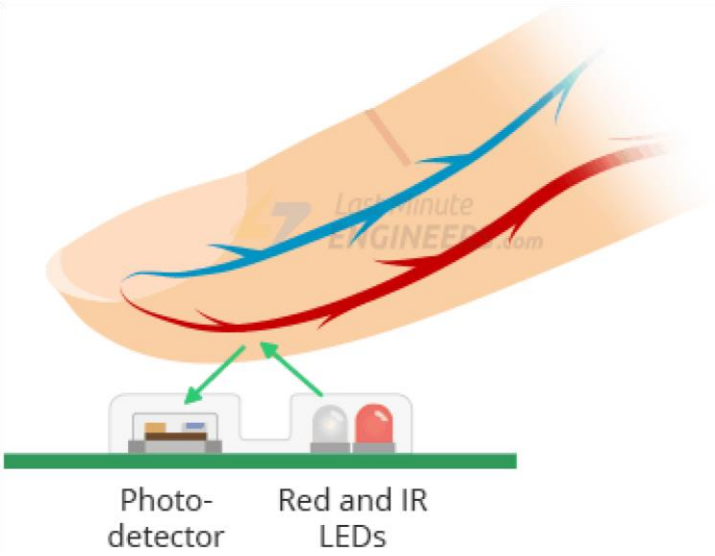


Figure 5-13 measuring about of reflected light

The MAX30102 works by shining both lights onto the finger or earlobe (or essentially anywhere

Where the skin isn't too thick, so both lights can easily penetrate the tissue) and measuring the amount of reflected light using a photodetector. This method of pulse detection through light is called Photoplethysmogram.

The working of MAX30102 can be divided into two parts: Heart Rate Measurement and Pulse Oximetry (measuring the oxygen level of the blood).

Heart Rate Measurement

The oxygenated hemoglobin (HbO2) in the arterial blood has the characteristic of absorbing IR light. The redder the blood (the higher the hemoglobin), the more IR light is absorbed. As the blood is pumped through the finger with each heartbeat, the amount of reflected light changes, creating a changing waveform at the output of the photodetector. As you continue to shine light and take photodetector readings, you quickly start to get a heartbeat (HR) pulse reading.

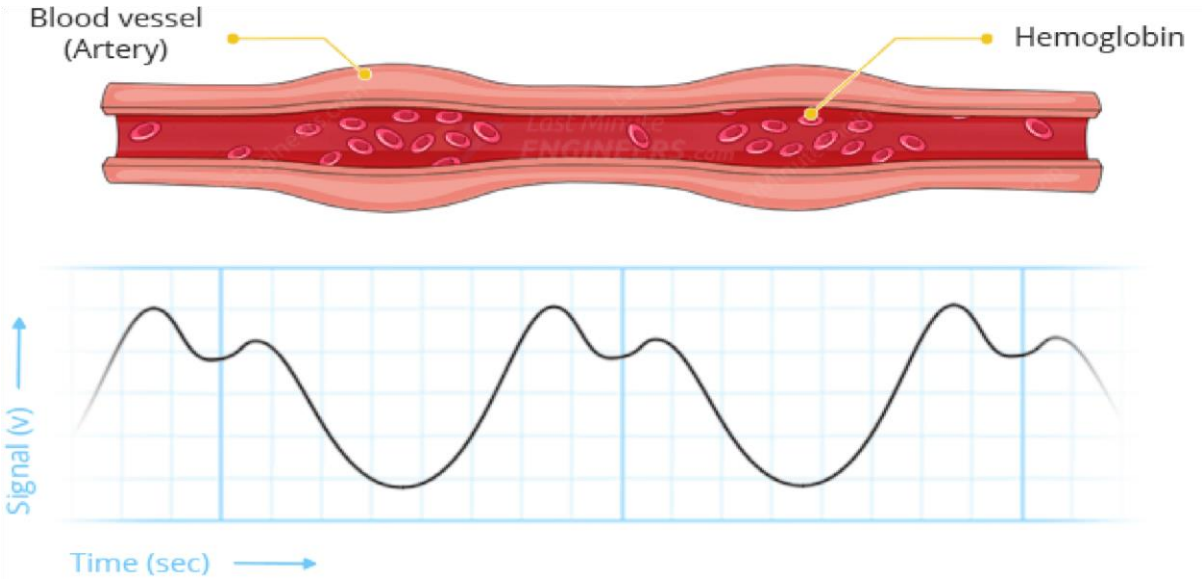


Figure 5-14 Heart Rate Measurement

Pulse Oximetry:

Pulse oximetry is based on the principle that the amount of RED and IR light absorbed varies depending on the amount of oxygen in your blood. The following graph is the absorption spectrum of oxygenated hemoglobin (HbO2) and deoxygenated hemoglobin (Hb).

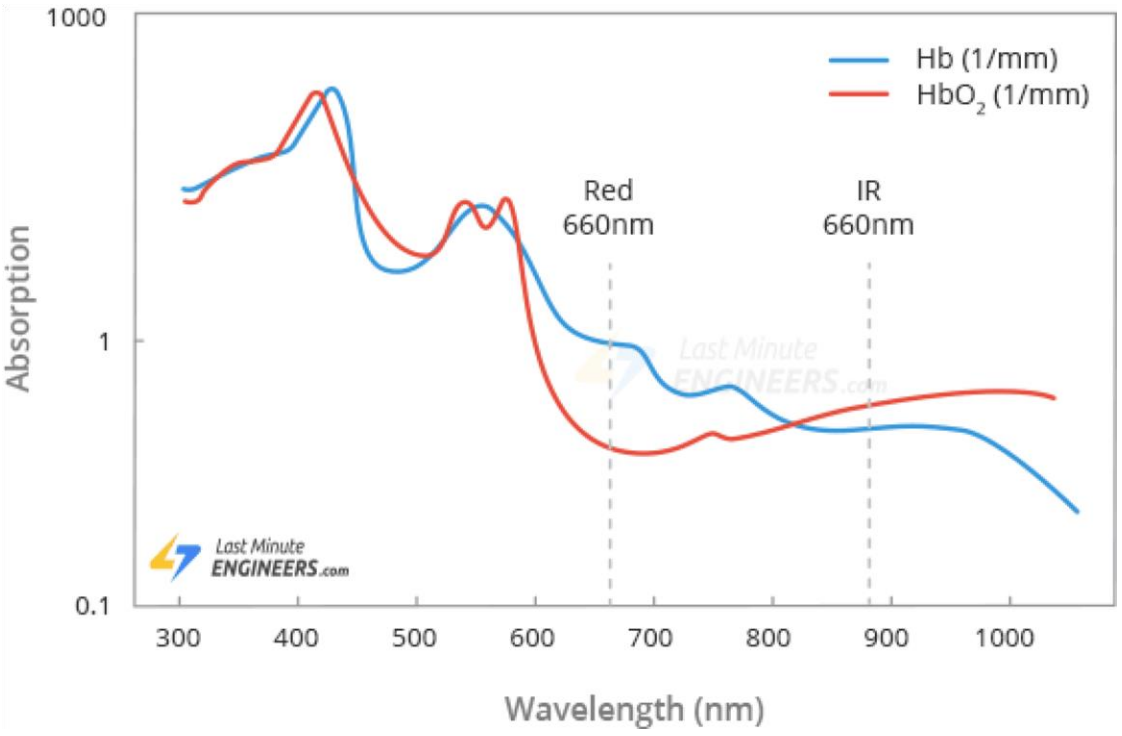


Figure 5-15 Absorption spectrum of (HbO2) and (Hb)

As you can see from the graph, deoxygenated blood absorbs more RED light (660nm), while oxygenated blood absorbs more IR light (880nm). By measuring the ratio of IR and RED light received by the photodetector, the oxygen level (SpO2) in the blood is calculated.

MAX30102 Module Pin out

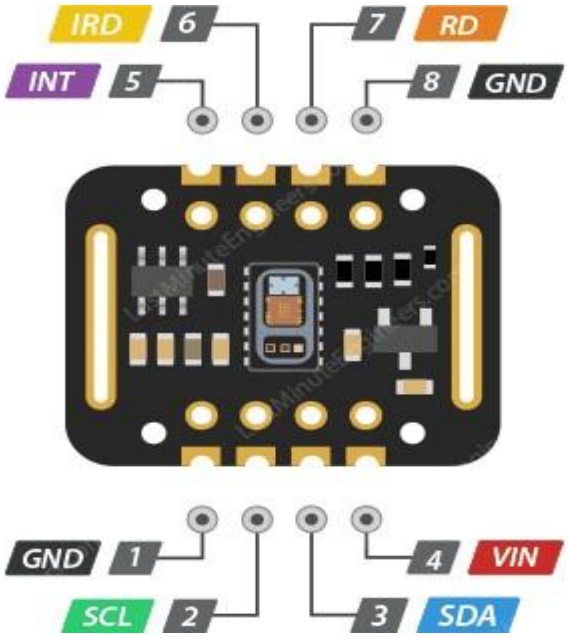


Figure 5-16 MAX30102 pins

The MAX30102 module brings out the following connections.

VIN: is the power pin. You can connect it to 3.3V or 5V output from your Arduino.

SCL: is the I2C clock pin, connect to your Arduino's I2C clock line.

SDA: is the I2C data pin, connect to your Arduino's I2C data line.

INT: The MAX30102 can be programmed to generate an interrupt for each pulse. This line is open drain, so it is pulled HIGH by the onboard resistor. When an interrupt occurs the INT pin goes LOW and stays LOW until the interrupt is cleared.

IRD: The MAX30102 integrates an LED driver to drive LED pulses for SpO2 and HR measurements. Use this if you want to drive the IR LED yourself, otherwise leave it unconnected.

RD: pin is like the IRD pin but is used to drive the Red LED. If you don't want to drive the red LED yourself, leave it unconnected.

GND: is the ground.

Temperature waterproof DS18B20

This is a pre-wired and waterproofed (with heat shrink) version of a 1 Wire DS18B20 sensor. Handy for when you need to measure something far away, or in wet conditions. While the sensor is good up to 125°C the cable is jacketed in PVC so we suggest keeping it under 100°C. Because they are digital, you don't get any signal degradation even over long distances! These 1-wire digital temperature sensors are fairly precise (±0.5°C over much of the range) and can give up to 12 bits of precision from the onboard digital-to-analog converter. They work great with any microcontroller using a single digital pin, and you can even connect multiple ones to the same pin, each one has a unique 64-bit ID burned in at the factory to differentiate them. Usable with 3.0-5.0V systems.

Technical Specifications:

- Temperature range: -55°C to +125°C (-67°F to +257°F).
- Accuracy: ±0.5°C from -10°C to +85°C.
- Resolution: 9 to 12 bits (configurable).
- Operating voltage: 3V to 5.5V.
- Operating current: 1.5mA (max).
- Waterproof rating: IP67 (fully waterproof and submersible).
- Cable length: 1 meter (3.3 feet) or customized lengths.
- Interface: 1-wire digital interface.

How it works:

The DS18B20 waterproof temperature sensor uses a 1-wire digital interface to communicate with a microcontroller or other digital device. The 1-wire interface requires only a single data line and a ground connection, simplifying the wiring and reducing the number of pins needed on the microcontroller.

To read the temperature from the DS18B20 sensor, the microcontroller sends a command to the sensor over the 1-wire interface. The sensor responds by sending the temperature data back to the microcontroller. The DS18B20 can operate in several modes, including continuous temperature measurement mode, where it continuously measures the temperature and updates the temperature readings.

The DS18B20 sensor uses a digital temperature sensor chip that converts the temperature into a digital signal. The sensor chip uses a temperature-to-digital conversion technique that provides

high accuracy and resolution. The sensor also includes a scratchpad memory that can be used to store configuration data or other information.

The waterproof version of the DS18B20 sensor includes a waterproof casing that protects the sensor and wiring from moisture and water. The casing is typically made of plastic or stainless steel and includes a cable that connects the sensor to the microcontroller. The cable is also waterproof and can be customized to different lengths depending on the application.

Figure 5-17 Temperature waterproof DS18B20

Buzzer 12V

In simplest terms, a piezo buzzer is a type of electronic device that's used to produce a tone, alarm, or sound. It's lightweight with a simple construction, and it's typically a low-cost product.

Technical Specifications:

• Rated Frequency: 4,100Hz.
• Operating Voltage: 12 Vdc.
• Current Consumption: 7mA @ 12Vdc.
• Sound Pressure Level(30cm): 70dB @ 12Vdc.
• Dimensions: 13mm Diameter, 7mm High, Pin Spacing – 7mm.

Figure 5-18 Buzzer 12V

Micro Mini 3V

Technical Specifications:

- Product Name: DC Vibration Motor.
- Rated Voltage: DC 3V.
- Voltage Usage Range: 2.5-4V DC
- Rated Current: 70mA Max.
- Start Current: 901mA Max.
- Start Voltage: 2.3V DC.
- Insulation Resistance: 10MΩ Min.
- Speed: 12000±2500RPM/Min.
- Diameter: 1cm/0.39″.
- Quantity: 1 Pcs.

How it works:

Same as Mini Vibration motor DC 3-6V.

Figure 5-19 Micro Mini 3V

Steps for making bracelet:

Connect the MAX30102 sensor to the Arduino Nano board:

Connect the VIN pin of the sensor to the 3.3V pin of the Arduino Nano board.

Connect the GND pin of the sensor to the GND pin of the Arduino Nano board.

Connect the SCL pin of the sensor to the A5 pin of the Arduino Nano board.

Connect the SDA pin of the sensor to the A4 pin of the Arduino Nano board.

1- Connect the DS18B20 temperature sensor to the Arduino Nano board:

Connect the VCC pin of the sensor to the 5V pin of the Arduino Nano board.

Connect the GND pin of the sensor to the GND pin of the Arduino Nano board.

Connect the DATA pin of the sensor to the D2 pin of the Arduino Nano board.

2- Connect the microcontroller to the Arduino Nano board:

Connect the VCC pin of the microcontroller to the 5V pin of the Arduino Nano board.

Connect the GND pin of the microcontroller to the GND pin of the Arduino Nano board.

Connect the OUT pin of the microcontroller to the D3 pin of the Arduino Nano board.

3- Connect the buzzer to the Arduino Nano board:

Connect the positive pin of the buzzer to the D7 pin of the Arduino Nano board.

Connect the negative pin of the buzzer to the GND pin of the Arduino Nano board.

4- Connect the ESP8266 Wi-Fi module to the Arduino Nano board:

Connect the TX pin of the ESP8266 module to the RX pin (D0) of the Arduino Nano board.

Connect the RX pin of the ESP8266 module to the TX pin (D1) of the Arduino Nano board.

Connect the CH_PD pin of the ESP8266 module to the 3.3V pin of the Arduino Nano board.

Connect the VCC pin of the ESP8266 module to the 3.3V pin of the Arduino Nano board.

Connect the GND pin of the ESP8266 module to the GND pin of the Arduino Nano board.

5.3 Firebase

Firebase is a cloud-based platform that allows developers to create and manage mobile and web applications. It provides various tools and services such as real-time database, authentication, hosting, storage, and analytics that help developers build and deploy their applications quickly and easily.

Firebase's real-time database is one of its most popular features. It is a No SQL database that allows developers to store and sync data in real-time across devices and platforms. This enables applications to update data instantly and provide a seamless user experience.

Firebase also provides various authentication methods such as email and password, social media, and phone number authentication, helping developers to secure their applications and user data.

Firebase also offers hosting services that enable developers to deploy and host their applications on Firebase servers, making it easier to manage and scale their applications.

Overall, Firebase provides a comprehensive set of tools and services that can help developers build, manage, and scale their applications quickly and easily.

Why we use firebase for our project?

Firebase can be a great choice for our project for several reasons, depending on our project requirements. Here are some of the common reasons why we choose Firebase:

1- Real-time Database: Firebase's real-time database allows you to store and sync data in real-time across devices and platforms. This enables you to provide a seamless and responsive user experience.
2- Authentication: Firebase provides various authentication methods such as email and password, social media, and phone number authentication, helping developers to secure their applications and user data.
3- Hosting: Firebase hosting services make it easier to deploy and host your application on Firebase servers, making it easier to manage and scale your application.
4- Easy to integrate: Firebase is easy to integrate with various platforms and programming languages such as Android, IOS, JavaScript, and Unity.
5- Analytics: Firebase provides powerful analytics tools that give your insight into your application's usage and performance, allowing you to optimize your application and user experience.
6- Cloud Functions: Firebase Cloud Functions allow you to run server-side code without setting up a server. This enables you to perform complex backend tasks and integrate with third-party services easily.
7- Scalability: Firebase is designed to be highly scalable, allowing you to easily handle large amounts of data and traffic as your application grows.
8- Cost-effective: Firebase offers a free plan with generous usage limits, making it a cost-effective solution for small to medium-sized projects.

overall, Firebase provides a comprehensive set of tools and services that can help you build, manage, and scale your application quickly and easily, making it a popular choice among developers.

For miners and supervisors' data, we use Cloud Fire store, which is a NoSQL document database that allows us to store and sync data between our app and the cloud. We create a collection for miners and supervisors and store their data as documents in the collection. This data can be easily queried and updated in real-time.

For sensors readings, we use Firebase's real-time database, which allows us to store and sync data in real-time across devices and platforms. We create a node for each sensor and store the sensor's readings as child nodes. This enables you to update the sensor data in real-time and provides a seamless and responsive user experience.

For authentication, Firebase provides various authentication methods such as email and password, social media, and phone number authentication. We use Firebase Authentication to authenticate users and provide access to different parts of your application based on their role. We create roles for sign-in, sign up admin, and supervisor and provide different levels of access based on their role.

5.3.1 Receive sensors readings by using real time database

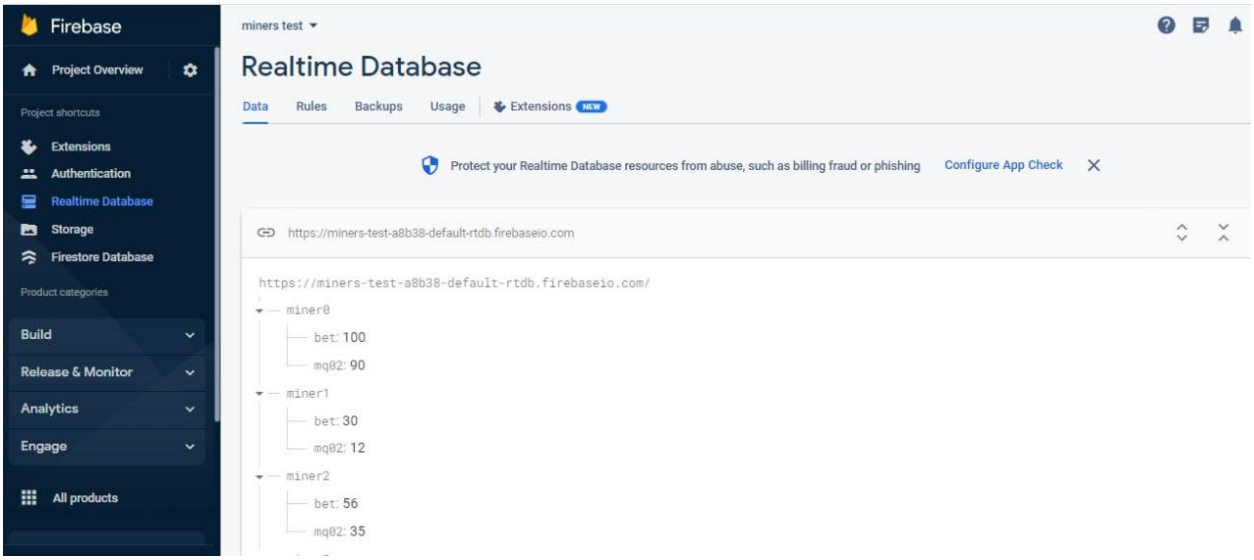


Figure 5-20 Receive sensors readings by using real time database

5.3.2 Supervisors data on cloud Firestore

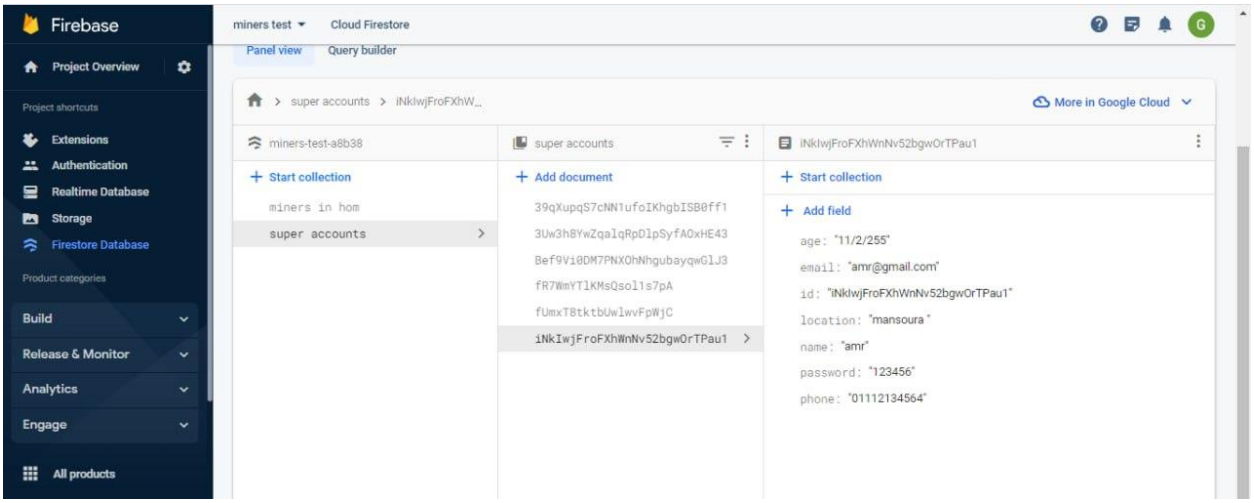


Figure 5-21 Supervisors data on cloud Firestore

5.3.3 Supervisors that already sign up (Authentication)

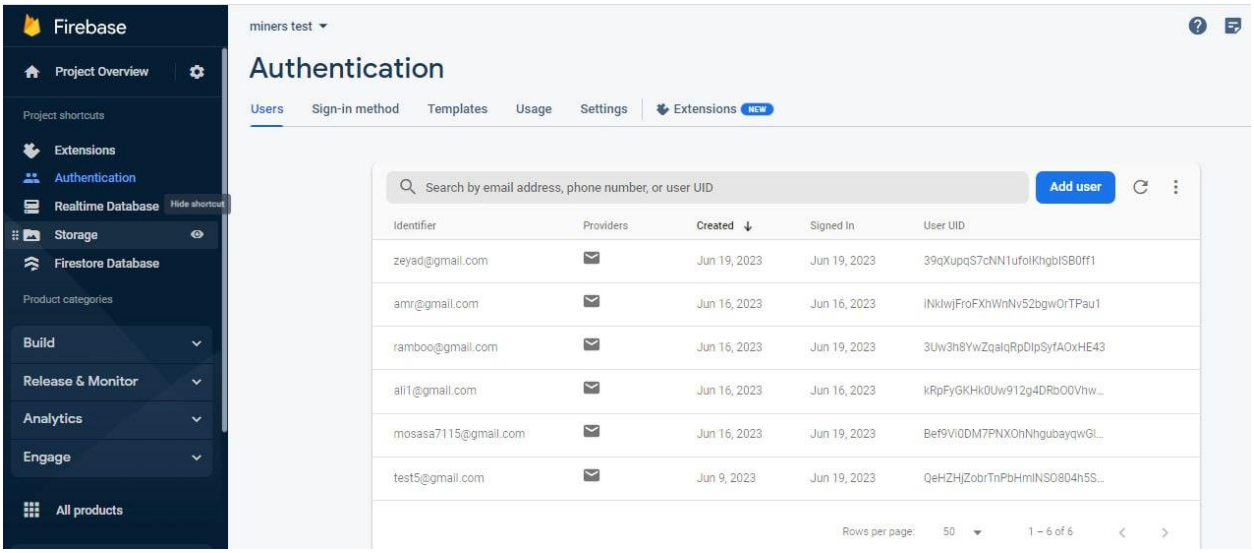


Figure 5-22 supervisors that already sign up (Authentication)

5.3.4 Miners details

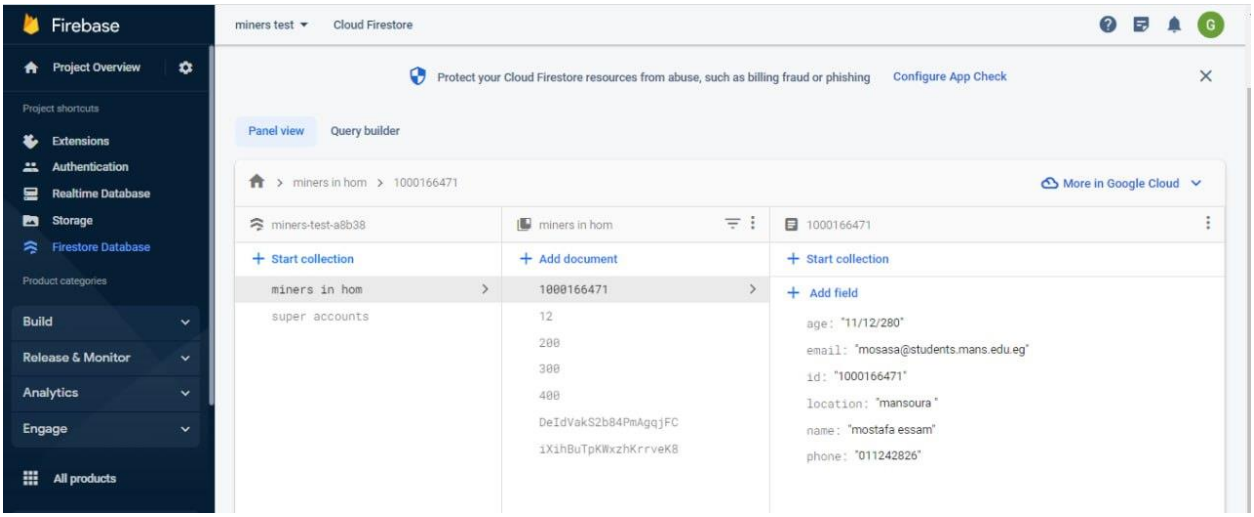


Figure 5-23 Miners details

5.4 Tools

5.4.1 Android Studio

Android Studio is an integrated development environment (IDE) for developing Android applications. It is the official IDE for Android app development, providing features such as code editing, debugging, testing, and deployment. Android Studio is based on IntelliJ IDEA and includes a comprehensive set of tools and resources to help developers build high-quality and reliable Android applications. With Android Studio, developers can create and manage Android projects, design user interfaces, write code, and test their applications on various virtual and physical devices. Overall, Android Studio is a powerful and user-friendly tool for Android app development.

5.4.2 Arduino IDE

Arduino IDE is an open-source software development environment used to program Arduino boards. It is a cross-platform application available for Windows, Mac, and Linux systems. The Arduino IDE provides a simple and easy-to-use interface for writing, compiling, and uploading code to an Arduino board. It also includes a text editor with syntax highlighting, a serial monitor to display data from the board, and a library manager to easily add third-party libraries to your project. With the Arduino IDE, you can write code in the C++ programming language and upload it to an Arduino board to control various electronic components such as sensors, motors, and displays. Overall, the Arduino IDE is a powerful tool for prototyping and developing electronic projects with an Arduino board.

5.4.3 C:

Procedural programming language with a static system that has the functionality of structured programming, recursion, and lexical variable scoping. C was created with constructs that transfer well to common hardware instructions. It has a long history of use in programs that were previously written in assembly language. a machine-independent programming language that is mainly used to create many types of applications and operating systems such as Windows, and other complicated programs such as the Oracle database, Git, Python interpreter, and games and is considered a programming foundation in the process of learning any other programming language. Operating systems and diverse application software for computer architectures ranging from supercomputers to PLCs and embedded systems are examples of such applications.

5.4.4 Dart:

Client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks. Languages are defined by their technical envelope — the choices made during development that shape the capabilities and strengths of a

language. Dart is designed for a technical envelope that is particularly suited to client development, prioritizing both development (sub second stateful hot reload) and high-quality production experiences across a wide variety of compilation targets (web, mobile, and 72 desktop). Dart also forms the foundation of Flutter. Dart provides the language and runtimes that power Flutter apps, but Dart also supports many cores developer tasks like formatting, analyzing, and testing code.

5.4.5 Flutter:

Google's free, open-source software development kit (SDK) for cross-platform mobile application development. Using a single platform agnostic codebase, Flutter helps developers build high-performance, scalable applications with attractive and functional user interfaces for Android or IOS. Flutter relies on a library of pre-made widgets that make it simple for even people with limited programming or development experience to launch their own mobile applications quickly. Created by Google in 2015 and officially launched in 2018, Flutter has quickly become the toolkit of choice developers. According to Statistics, Flutter has recently surpassed React Native to become the number one mobile app development framework.

Chapter 6
Conclusion & Futures work

Preamble

In this chapter, we will summarize our research effort and the challenges that faced us in finding a suitable solution. Finally, we will discuss some features that we will add and improve in the future.

6.1 Research Effort

We have been working tirelessly to develop a solution that is both effective and user-friendly to address the critical safety concerns facing miners. Our team has developed a Smart helmet and bracelet that can detect gas leaks, monitor vital rates, and track the location of miners in real-time. By providing this data to supervisors, we can ensure that miners are safe and that any potential issues are delivered immediately. Our solution is a significant advancement toward ensuring the safety and well-being of miners in the mining industry. With our smart helmet and bracelet, miners can work freely, knowing their health and safety are being monitored 24/7. We are proud of developing a solution that will be effective, comfortable, and easy to use, ensuring that miners can focus on their work without distraction. Our efforts are a testament to our commitment to improving safety standards in the mining industry and to save lives.

6.2 Challenges

The main technical challenges that we faced are the following:

1. Sensor Accuracy: To ensure the accuracy of the sensors used in the smart helmet and bracelet, we calibrated the sensors and conducted extensive testing to ensure that they were precise and reliable. We also incorporated redundant sensors to ensure that any potential inaccuracies could be detected and corrected.
2. Battery Life: To optimize battery life, we used low-power sensors and optimized the code to minimize power consumption. We also incorporated a rechargeable battery and designed the devices to be easily replaceable to ensure continuous operation.
3. Comfort and Usability: We conducted extensive user testing and incorporated feedback to ensure the Smart helmet and bracelet were comfortable and easy to use. We also designed the devices to be lightweight, adjustable, and compatible with safety gear to ensure a comfortable fit for all users.

6.3 Future Work

Here is not the end of the road, it still needs more research and development, and this is what we will continue to do to reach the best possible efficiency

1. Wireless communication: Consider adding wireless communication capabilities to your product. Enable miners to communicate with each other in case of an emergency. Help to improve safety and reduce response times in case of an incident.
2. Data analytics: Consider using data analytics to analyze the data collected by the helmet and bracelet. Help identify trends and patterns to improve safety and prevent incidents.
3. Machine learning: Consider using machine learning algorithms to improve the accuracy of our product. By analyzing large amounts of data, machine learning algorithms can learn to detect patterns and predict future events such as explosions.
4. Voice communication: The helmet can be equipped with a microphone and speaker to enable voice communication between miners. Useful in an emergency or to coordinate work activities.
5. SOS button: This could be used to send an emergency alert if the user is in danger.

6.4 Conclusion

To sum up, everything discussed in the previous chapters, our main aim was to develop a solution that would help improve safety standards in the mining industry and save lives. We achieved this by designing a Smart helmet and bracelet that could detect gas leaks and monitor the vital rates of miners in real-time. The data collected by these devices is transmitted in real-time to supervisors, enabling them to take immediate action in case of any potential issues. Despite the challenges we faced, such as sensor accuracy, battery life, comfort and usability, data transmission and storage, and integration with existing systems, we developed solutions that are accurate, reliable, and user-friendly. We believe that our solution is a significant step towards ensuring the safety and well-being of miners in the mining industry. By providing real-time data and alerts to supervisors, we can ensure that miners are safe and that any potential issues are delivered immediately. Our efforts are a testament to our commitment to improving safety standards in the mining industry and to saving lives. Looking towards the future, we intend to continue to improve and refine our solution, incorporating new technologies and features to further enhance safety standards in the mining industry. Our main goal is to make the lives of miners easier and safer, and we are committed to achieving this goal through ongoing innovation and development.

Appendix

1. ESP 8266 Wi-Fi module

Next code initializes the Wi-Fi connection by declaring the network name and password, connects to the network, and prints the SSID being connected to.

Arduino Nano

wifi.ino Functions.ino

```ino
#include <ESP8266WiFi.h>
#include <ESP8266Firebase.h>

#define _SSID "Minermate"        // Wifi USER NAME
#define _PASSWORD  "minermate2023"    // WiFi Password
#define FIREBASE_ID "miners-test-a8b38-default-rtdb"  // Firebase ID

Firebase firebase(FIREBASE_ID);
void wifiInit(){

  WiFi.mode(WIFI_STA);  // Set the WiFi mode to Station mode (client mode)
  WiFi.disconnect();     // Disconnect from any previously connected WiFi network
  delay(1000);

  Serial.println();
  Serial.println();
  Serial.print("Connecting to: ");
  Serial.println(_SSID);  // Print the SSID (network name) being connected to

  WiFi.begin(_SSID, _PASSWORD);  // Connect to the WiFi network using the provided SSID and password

  while (WiFi.status() != WL_CONNECTED) {  // Wait until connected to the WiFi network
    delay(500);
    Serial.print("-");
  }

  Serial.println("");
  Serial.println("WiFi Connected");

  Serial.print("Use this URL to connect: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());  // Print the local IP address of the connected WiFi network
  Serial.println("/");

}

String MSG;
void serialRecieve(){

  MSG = "";
  while(Serial.available()){
    char c = Serial.read();
    MSG = MSG + c;  // Append the received character to the MSG string
    if(!Serial.available())
      delay(20);  // Delay to allow more characters to be received
  }

  return;
}
// Mostafa:5.5;
void sendFirebase(){
  String key = "", tmp = "";
  for(int i = 0 ; i < MSG.length() ; ++i){
    if(MSG[i] == ':'){  // If a colon is encountered, store the key
      key = tmp;
      tmp = "";
    }else if(MSG[i] == ';'){  // If a semicolon is encountered, convert the value and send to Firebase
      firebase.setFloat("miner0/" + key, atof(tmp.c_str()));
      tmp = "";
    }else{
      tmp += MSG[i];  // Append the character to the temporary value string
    }
  }
}
```

2. Arduino (helmet)

This code reads data from gas sensors and a DHT11 temperature and humidity sensor, constructs a message with the sensor values, and sends it through serial communication. It also includes code to activate a buzzer and vibration motor if certain gas levels are detected.

helmet.ino | sensor_handler.ino

```ino
#include <DHT.h>  // Include the DHT library
#define Sensor_Num 2  // Define the number of sensors
#define DHT_pin 3  // Pin number for DHT sensor
#define DHTTYPE DHT11  // DHT sensor type
#define Buzzer_pin    8
#define Vibration_pin 9
DHT dhtSensor(DHT_pin, DHTTYPE);  // Create a DHT object
void setup() {
  Serial.begin(9600);  // Initialize the serial communication at a baud rate of 9600
  // Add a sensor with pin A0 and name "MQ2"
  addSensor(A0, "MQ9");
  // Add a sensor with pin A1 and name "MQ5"
  addSensor(A1, "MQ5");
  // Add a sensor with pin A2 and name "MQ7"
  addSensor(A2, "MQ7");
  // Add a sensor with pin A3 and name "MQ9"
  addSensor(A3, "MQ9");
  pinMode(Buzzer_pin, OUTPUT);  // Set Buzzer pin as OUTPUT
  pinMode(Vibration_pin, OUTPUT);  // Set Vibration pin as OUTPUT
  dhtSensor.begin();  // Initialize the DHT sensor
}
long time2 = 0;
void loop() {
  if(millis() - time2 > 2000){
    Check();
  }
  constructMSG();  // Construct the message with sensor values
  sendMSG();  // Send the message through serial communication
}
```

This code defines arrays and variables to store sensor information, including pins, names, values, and thresholds. It also includes functions to add sensors, read their values, and construct a message with the sensor values. The addSensor() function adds a sensor to the list with a pin and name. The constructMSG() function reads the sensor values, calculates their average, and updates the stored values. If a significant change in value is detected, it marks the sensor as needing an update and sets a flag indicating that new data is available.

elmet.ino | sensor_handler.ino

```
float multipliFactor[] = {5.0 / 1023.0, 5.0 / 1023.0, 5.0 / 1023.0, 5.0 / 1023.0};  // Array of multiplication factors for sensor readings
float threshold[] = {500, 500, 500, 500, 37};
String Sensor_Names[Sensor_Num];  // Array to store sensor names
int Sensor_Pins[Sensor_Num] = {};  // Array to store sensor pins
float Sensor_Values[Sensor_Num] = {};  // Array to store sensor values
bool needUpdate[Sensor_Num + 2] = {};  // Array to track which sensors need updates
float Humidity, Temperature;  // Variables to store humidity and temperature readings
int last_pin = 0;  // Index of the last sensor pin used

// Function to add a sensor with a pin and name
void addSensor(int pin, String Name){
  Sensor_Names[last_pin] = Name;  // Assign the name to the corresponding index in Sensor_Names array
  Sensor_Pins[last_pin++] = pin;  // Assign the pin to the corresponding index in Sensor_Pins array
  pinMode(pin, INPUT);  // Set the sensor pin as INPUT
}

String MSG;
bool newData;

// Function to construct the message with sensor values
void constructMSG(){
  MSG = "";

  // Loop through the sensors and read their values
  for(int i = 0 ; i < last_pin ; ++i){
    float reading = 0;

    // Perform multiple readings and calculate the average value
    for(int j = 0 ; j < 1 ; ++j){
      reading += analogRead(Sensor_Pins[i]) * multipliFactor[i];  // Read the analog value from the sensor pin and multiply by the corresponding factor
    }

    reading /= 1.0;

    // Check if the reading has changed significantly
    if(abs(reading - Sensor_Values[i]) > 1e-3){
      needUpdate[i] = true;  // Mark the sensor as needing an update
      newData = true;  // Mark that new data is available
      Sensor_Values[i] = reading;  // Update the stored sensor value
    }
  }

  float newHumidity = 0;
  float newTemperature = 0;

  // Perform multiple readings of humidity and temperature and calculate the average values
  for(int j = 0 ; j < 500 ; ++j){
    newHumidity += dhtSensor.readHumidity();  // Read the humidity value from the DHT sensor
    newTemperature += dhtSensor.readTemperature();  // Read the temperature value from the DHT sensor
  }

  newHumidity /= 500.0;
  newTemperature /= 500.0;

  // Check if the temperature has changed significantly
  if(abs(Temperature - newTemperature) > 1e-3){
    newData = true;  // Mark that new data is available
    needUpdate[last_pin] = 1;  // Mark the temperature as needing an update
    Temperature = newTemperature;  // Update the stored temperature value
  }

  // Check if the humidity has changed significantly
  if(abs(Humidity - newHumidity) > 1e-3){
    newData = true;  // Mark that new data is available
    needUpdate[last_pin + 1] = 1;  // Mark the humidity as needing an update
    Humidity = newHumidity;  // Update the stored humidity value
  }
}

unsigned long time;
bool tr;
// Function to send the constructed message
```

This code defines a function to send a constructed message with the sensor values through serial communication. It includes a timer to execute the code block every 10 seconds. The function checks if new data is available and loops through the sensors to add their values to the message string. If a significant change in value is detected, it adds the sensor name and value to the message string and resets the update flag. It then prints the message through serial communication and resets the new data flag.

```
void sendMSG(){
  if (millis() - time > 1000){  // Execute the code block every 10 seconds
    time = millis();  // Update the time variable
     tr = !tr;

    if(newData){  // Check if new data is available
      MSG = "";

      // Loop through the sensors and add their values to the message string
      for(int i = 0 ; i < last_pin ; ++i){
        if(needUpdate[i]){  // Check if the sensor needs an update
          MSG += Sensor_Names[i] + ":";  // Add the sensor name to the message string
          MSG.concat(Sensor_Values[i] + tr);  // Add the sensor value to the message string
          MSG += ";";
          needUpdate[i] = 0;  // Reset the update flag
        }
      }

      if(needUpdate[last_pin]){  // Check if the temperature needs an update
        MSG += "dht11/temperature:";  // Add the temperature label to the message string
        MSG.concat(Temperature);  // Add the temperature value to the message string
        MSG += ";";
        needUpdate[last_pin] = 0;  // Reset the update flag
      }

      if(needUpdate[last_pin + 1]){  // Check if the humidity needs an update
        MSG += "dht11/humidity:";  // Add the humidity label to the message string
        MSG.concat(Humidity);  // Add the humidity value to the message string
        MSG += ";";
        needUpdate[last_pin + 1] = 0;  // Reset the update flag
      }

      Serial.println(MSG);  // Print the constructed message
      newData = false;  // Reset the new data flag
    }
  }
```

This code defines functions to turn on and off an alarm through a buzzer and vibration motor.

```
}
void alarmON(){
  digitalWrite(Buzzer_pin, 1);
  digitalWrite(Vibration_pin, 1);
  delay(10);
}
void alarmOFF(){
  digitalWrite(Buzzer_pin, 0);
  digitalWrite(Vibration_pin, 0);
  delay(10);
}
int err_cnt = 0;
bool isError(){
  constructMSG();
  for(int i = 0 ; i < last_pin ; ++i){
    if(Sensor_Values[i] > threshold[i]){
      return 1;
    }
  }
  if(Temperature > threshold[last_pin]){
    return 1;
  }
  return 0;
}
void Check(){
  err_cnt += isError();
  if(err_cnt >= 3){

    do{
      alarmON();
      delay(3000);
      alarmOFF();
    }while(isError());
    err_cnt = 0;
  }else if(err_cnt == 2){
    do{
      alarmON();
      delay(2000);
      alarmOFF();
    }while(isError());
  }else if(err_cnt == 1){
    do{
      alarmON();
      delay(500);
      alarmOFF();
    }while(isError());
  }
}
```

3. Arduino (bracelet)

This code initializes the I2C and serial communication, adds a sensor with a pin and name, and configures the MAX30100 sensor. It also includes a loop that constructs a message with sensor values and sends it through serial communication.

```
#include <Wire.h>

#define MAX30100_ADDRESS 0x57

#define MAX30100_REG_MODE_CONFIG 0x06
#define MAX30100_REG_SPO2_CONFIG 0x07
#define MAX30100_REG_LED_CONFIG 0x09
#define MAX30100_REG_FIFO_DATA 0x05
#define MAX30100_REG_FIFO_CONFIG 0x08
#define Buzzer_pin    8
#define Vibration_pin 9
#define Sensor_Num 1  // Define the number of sensors

void setup() {
  Wire.begin();  // Initialize I2C communication
  Serial.begin(9600);  // Initialize serial communication
  // Add a sensor with pin A0 and name "Miner temp"
  addSensor(A0, "Miner temp");
  configureMAX30100();
  pinMode(5, OUTPUT);
}
long time2 = 0;
void loop() {
  if(millis() - time2 > 2000){
    Check();
  }
  constructMSG();  // Construct the message with sensor values
  sendMSG();  // Send the message through serial communication
}
```

This code includes functions to configure and read from the MAX30102 pulse oximeter sensor, as well as calculate heart rate and oxygen saturation based on the sensor data using the Maxim Heart Rate and Oxygen Saturation Algorithm.

```
void configureMAX30100() {
  // Configure mode and sample rate
  writeRegister(MAX30100_REG_MODE_CONFIG, 0x03);
  // Configure LED current and pulse width
  writeRegister(MAX30100_REG_LED_CONFIG, 0x1F);
  // Configure SpO2 ADC range and sample rate
  writeRegister(MAX30100_REG_SPO2_CONFIG, 0x07);
  // Configure FIFO settings
  writeRegister(MAX30100_REG_FIFO_CONFIG, 0x0F);
}
byte readRegister(byte regAddress) {
  Wire.beginTransmission(MAX30100_ADDRESS);
  Wire.write(regAddress);
  Wire.endTransmission(false);
  Wire.requestFrom(MAX30100_ADDRESS, 1);
  return Wire.read();
}
void writeRegister(byte regAddress, byte regValue) {
  Wire.beginTransmission(MAX30100_ADDRESS);
  Wire.write(regAddress);
  Wire.write(regValue);
  Wire.endTransmission();
}
int calculateHeartRate(byte redValue, byte irValue) {
  // Perform heart rate calculation based on sensor data
  // Using Maxim Heart Rate and Oxygen Saturation Algorithm (MASIMO)
  float redAC = redValue - 25.0;  // Red LED AC component
  float irAC = irValue - 25.0;    // IR LED AC component
  float redDC = redValue + 25.0;  // Red LED DC component
  float irDC = irValue + 25.0;    // IR LED DC component
  float ratio = (redAC / redDC) / (irAC / irDC);
  // Heart rate calculation
  int heartRate = 60 * (1 / (1 + ratio));
  return heartRate;
}

int calculateOxygenSaturation(byte redValue, byte irValue) {
  // Perform oxygen saturation calculation based on sensor data
  // Using Maxim Heart Rate and Oxygen Saturation Algorithm (MASIMO)
  float redAC = redValue - 25.0;  // Red LED AC component
  float irAC = irValue - 25.0;    // IR LED AC component
  float redDC = redValue + 25.0;  // Red LED DC component
  float irDC = irValue + 25.0;    // IR LED DC component
  float ratio = (redAC / redDC) / (irAC / irDC);
  // Oxygen saturation calculation
  int oxygenSaturation = 100 - 25 * ratio;
  return oxygenSaturation;
}
```

4. Mobile application

4.1 packages we used in our application

```
flutter_lints: ^2.0.1
percent_indicator: ^4.2.3
curved_navigation_bar: ^1.0.3
flutter_bloc: ^8.1.2
bloc: ^8.1.1
quickalert: ^1.0.1
iconly: ^1.0.1
get: ^4.6.5
flutter_screenutil: ^5.8.2
firebase_core: ^2.13.0
firebase_auth: ^4.6.1
fluttertoast: ^8.2.2
cloud_firestore: ^4.8.0
firebase_database: ^10.2.3
simple_circular_progress_bar: ^1.0.2
```

4.2 Sign in (linked with firebase)

```
Future<void> signIn({required email, required pass}) async {
  try {
    await FirebaseAuth.instance
        .signInWithEmailAndPassword(
            email: email.text.trim(), password: pass.text.trim())
        .then(
          (value) => navigateTo(context, HomeScreen()),
        );
    errorMessage(text: 'You are login successful', state: ToastState.success);
  } on FirebaseAuthException catch (e) {
    errorMessage(text: e.toString(), state: ToastState.error);
    if (kDebugMode) {
      print(e);
    }
  }
}
```

4.3 Sign up (linked with firebase)

```
Future<void> signup({required email, required pass}) async {
  try {
    await FirebaseAuth.instance
        .createUserWithEmailAndPassword(
            email: email.text.trim(), password: pass.text.trim())
        .then((value) => showAlert(QuickAlertType.success));
  } on FirebaseAuthException catch (e) {
    if (kDebugMode) {
      print(e);
    }
  }
}
```

4.4 Save miner data in firebase

```
postDetailsToFireStore() async {
  FirebaseFirestore firebaseFirestore = FirebaseFirestore.instance;

  MinerModel minerModel = MinerModel();

  minerModel.email = emailController.text;
  minerModel.uid = idController.text;
  minerModel.name = nameController.text;
  minerModel.phone = phoneController.text;
  minerModel.age = ageController.text;
  minerModel.location = locationController.text;

  await firebaseFirestore
      .collection("miners in hom")
      .doc(idController.text)
      .set(minerModel.toFirebase());
}
```

4.5 Miner model data

```
factory MinerModel.fromFirebase(map) {
  return MinerModel(
    name: map['name'],
    email: map['email'],
    age: map['age'],
    phone: map['phone'],
    location: map['location'],
    uid: map['id'],

  );
```

4.6 Supervisor model data

```dart
class UsersModel {
  String? name;
  String? email;
  String? age;
  String? phone;
  String? location;
  String? uid;
  String? password;

  UsersModel({
    this.name,
    this.email,
    this.age,
    this.phone,
    this.location,
    this.uid,
    this.password
  });

  Map<String, dynamic> toFirebase() {
    return {
      "name": name,
      "email": email,
      "age": age,
      "phone": phone,
      "location":location,
```

4.7 Receive sensors readings from firebase

```
class _SensorsPageState extends State<SensorsPage> {
  String battery = '0';
  String mQ02 = '0';
  double? valueMq02;
  double? batteryValue;
  var firebase = FirebaseDatabase.instance.ref();

  @override
  Widget build(BuildContext context) {
    DatabaseReference reference1 = firebase.child('miner$currentIndex/bet');
    DatabaseReference reference2 = firebase.child('miner$currentIndex/mq02');
    //listen to firebase realtime database value
    reference1.onValue.listen((event) {
      setState(() {
        battery = event.snapshot.value.toString();
        // batteryValue = double.parse(battery);
      });
    });
    reference2.onValue.listen((event) {
      setState(() {
        mQ02 = event.snapshot.value.toString();
        valueMq02 = double.parse(mQ02);
      });
    });
```

4.8 home screen

```
class _MinersProfileState extends State<MinersProfile> {
  MinerModel minerModel = MinerModel();

  Future getDocIds() async {
    docIds = [];
    setState(() {});
    await FirebaseFirestore.instance
        .collection('miners in hom')
        .get()
        .then((snapshot) => snapshot.docs.forEach((document) {
              docIds.add(MinerModel.fromFirebase(document.data(), document.id));
              // log(document.data().toString());
              // log(MinerModel.fromFirebase(document.data(), document.id)
              //     .toFirebase()
              //     .toString());
            }));
    log(docIds.length.toString(), name: 'len');
  }
```

4.9 system control panel

```
SensorItem(
  sensorName: 'LPG',
  valueNotifier: valueMq02!,
  circularProgressText: valueMq02.toString(),
), // SensorItem
SensorItem(
  sensorName: 'methane',
  valueNotifier: valueMq02!,
  circularProgressText: valueMq02.toString(),
), // SensorItem
SensorItem(
  sensorName: 'carbon monoxide',
  valueNotifier: valueMq02!,
  circularProgressText: valueMq02.toString(),
), // SensorItem
SensorItem(
  sensorName: 'temperature',
  valueNotifier: valueMq02!,
  circularProgressText: valueMq02.toString(),
), // SensorItem
SensorItem(
  sensorName: 'humidity',
  valueNotifier: valueMq02!,
  circularProgressText: valueMq02.toString(),
), // SensorItem
```

```
SensorItem(
    image: 'img/Battery.png',
    sensorName: 'Battery',
    valueNotifier: valueBa!,
    circularProgressText: valueBa.toStri
    isImage: true), // SensorItem

SensorItem(
  isImage: false,
  sensorName: 'Smoke',
  valueNotifier: valueMq02!,
  circularProgressText: valueMq02.toStri
), // SensorItem
```

References:

1- https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/
2- https://store.fut-electronics.com/products/gas-sensor-module-mq5-analog-digital
3- https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf
4- https://wiki.seeedstudio.com/Grove-Gas_Sensor-MQ9/
5- https://www.electronicwings.com/sensors-modules/dht11
6- https://firebase.google.com/learn/pathways/firebase-flutter
7- https://www.alldatasheet.com/view.jsp?Searchword=Max30102%20datasheet&gclid=CjwKCAjw-b-kBhB-EiwA4fvKrP5iwzimoSk7mzby3vKzHKrh6vfhj0d_xWY_LupoV6UbdxyonKc0mxoCTEAQAvD_BwE
8- https://www.quick-teck.co.uk/Management/EEUploadFile/1420644897.pdf
9- https://www.alldatasheet.com/view.jsp?Searchword=Arduino%20mega%20datasheet&gclid=CjwKCAjw-b-kBhB-EiwA4fvKrLt3U4aIw2_VA1xrWKSU6V_sDX10eFk8mlnCRDxdFo7ESlZP70D7CBoCy_sQAvD_BwE
10- https://www.researchgate.net/publication/368841431_An_Approach_to_Design_a_Smart_Helmet_for_Worker_Safety_in_Mining_Industry_Using_Kansei_Engineering
11- https://www.irjet.net/archives/V7/i3/IRJET-V7I3864.pdf
12- http://www.angelhelmet.com/index_eng.html
13- https://www.zora.uzh.ch/id/eprint/194414/1/AICAS2020_shu.pdf
14- https://www.avadoctor.com/
15- https://www.sciencedirect.com/science/article/abs/pii/S0263224120306400
16- https://www.dynaway.com/blog/6-challenges-that-mining-industry-is-facing-now#:~:text=Environmental%20and%20climate%20change%20pressures,long%2Dterm%20sustainability%20of%20mining.
17- https://www2.deloitte.com/uk/en/pages/energy-and-resources/articles/key-issues-facing-mining-sector.html
18- https://everymind.imgix.net/assets/Uploads/Partners-in-Mining-Fact-sheet_Mining-and-mental-health.pdf
19- https://www.wa.gov.au/system/files/2022-11/MARS_LMS_Preliminary_Report_2A.pdf
20- smart helmet for air quality and hazardous event detection for the mining industry Article.
21- Navigating a decade of challenges: Five winning initiatives for mining CEOs Article.