Design & Development of Remote Oil Well Monitoring System for Oil and Gas Industry

Senior Design Project Report

Wah Engineering College

Group Members:

Saif Tariq	UW-19-MTS-BSc-003
Muhammad Faisal Bashir	UW-19-MTS-BSc-010

Supervisor Name:

Dr. Shaukat Ali

Department of Mechatronics Engineering

Wah Engineering College

University of Wah

2023

……………Design & Development of Remote Oil Well Monitoring System for Oil and Gas Industry…………… Mechatronics Engineering 2023

WEC

Design & Development of Remote Oil Well Monitoring System for Oil and Gas Industry

Senior Design Project Report

Submitted to the Department of Mechatronics Engineering
In partial fulfilment of the requirements
For the degree of
Bachelors of Science in Mechatronics Engineering
2023

Supervisor Name:

Dr. Shaukat Ali

Submitted By:

1. **Saif Tariq**

   UW-19-MTS-BSc-003

2. **Muhammad Faisal Bashir**

   UW-19-MTS-BSc-010

In the Name of Allah, the Most Beneficent the Most Merciful

Final Year Design Project Thesis Report

Project ID			**Number Of Members**	02

Title	Design & Development of Remote Oil Well Monitoring System for Oil and Gas Industry

Project Supervisor	**Dr. Shaukat Ali**	**Mechatronics Department**

Group Members		
Saif Tariq	UW-19-MTS-BSC-003	uw-19-mts-bsc-003@wecuw.edu.pk
M. Faisal Bashir	UW-19-MTS-BSC-010	uw-19-mts-bsc-010@wecuw.edu.pk

Supervisor Signature	**Chairperson MTS Signature**

CHECKLIST

- Number of pages attached with this form **Yes/No**
- I/We have enclosed the softcopy of this document along with the codes and scripts created by ourselves. **Yes/No**
- My/Our supervisor has attested the attached document. **Yes/No**
- I/We confirm to state that this project is free from any type of plagiarism and misuseof copyrighted material. **Yes/No**

UNDERTAKING

It is declared that the work entitled **"Design & Development of Remote Oil Well Monitoring System for Oil and Gas Industry "** presented in this report is an original piece of our own work, except where otherwise acknowledged in text and references. This work has not been submitted in any form for another degree or diploma at any university or other institution for tertiary education and shall not be submitted by us in future for obtaining any degree from this or any other University or Institution.

Group Members Signature
1-
2-

ACKNOWLEDGEMENT

All praises to Almighty Allah who has created this world of knowledge for us. We are grateful to Allah, who has given us strength, guidance and abilities to achieve this task. We would like to thanks our parents. With their eternal prayers we have done this job. May Allah give them reward for their efforts.

We would like to express our deepest appreciation to our project supervisor Dr. Shaukat Ali for his guideline and motivation. He provided valuable knowledge, time and supervision for completing the project work. A special gratitude to him, without his support this project would not have been possible.

ABSTRACT

There is a great chance that fire breakouts in industries like chemicals, petroleum, oil and gas would leads to extensive loss of property, damage, and most important, the loss of human life. It is essential to have a system in place that can maintain the area safe and alert the management as soon as an event occurs. These issues may be reduced using a remote monitoring system (IOT based).

Design and development of the remote oil well monitoring system utilizes stm-32 microcontroller, flow rate sensor, pressure, and temperature sensor, ESP32 & ESP82, and Firebase to visualization the data. Firstly, by utilizing the stm-32 microcontroller data is collected from the and flow rate sensors, pressure, temperature sensor. The designed system also uses a Peltier and fan cooling system to control the circuitry box temperature from high environmental temperature. And the collected data sent wirelessly to ESP32 controller/module, which transmit the data to cloud and visualization the data on Firebase for real-time and ESP82 for data storage on the cloud. The system addresses the limitations of old oil well monitoring systems and also improves the efficiency and safety of oil well. The system is designed to be efficient, scalable, and low-cost that makes it perfect for Industries like petroleum, chemicals, oil, and gas monitoring applications. The system doesn’t need any human association.

TABLE OF CONTENTS

ACKNOWLEDGEMENT vi

ABSTRACT vii

TABLE OF CONTENTS viii

LIST OF FIGURES xi

LIST OF TABLES xiii

Chapter 1 - Introduction 14

1.1 Problem Statement 14

1.2 Aims and Objectives 15

1.3 Overview 15

1.4 Scope of Study 17

1.5 Significance of Project 17

1.6 Technological Significance 17

1.7 Methodology 18

1.8 Background and Related work 19

Chapter 2 – Basic Description 20

2.1 Conceptual Approach and Design Objectives 20

2.2 Description of Prototype 21

2.4 Power Consumption of System 22

2.5 The Remote Oil and Gas Well Monitoring System 24

2.6 PSEUDOCODE: STM32 and ESP Module PSEUDOCOD 25

3.1 STM32F103C8 Microcontroller 26

3.1.1 Introduction 26

3.1.2 Specifications 26

3.1.3 Component Diagram 26

3.1.4 Software for STM32F103 27

3.2 ESP32 Module 28

3.2.1 Specifications.. 28

3.2.2 Component Diagram 29

3.3 ESP82 Module 29

3.3.1 Specifications.. 30

3.3.2 Component Diagram 30

3.4 16x4 Liquid Crystal Display (LCD) 30

3.5 Temperature Sensor (DS18b20)..31

3.5.1 Specifications...32

3.5.2 Component Diagram..32

3.6 Pressure Sensor (KH100C) 32

3.6.1 Specifications...32

3.6.2 Component Diagram..32

3.7 Flow Rate Sensor YF-S201..33

3.7.1 Specifications...33

3.7.2 Component Diagram..33

3.8 Solar Charge Controller..33

Component Diagram...34

3.9 Buck Converter..34

3.9.1 Specifications...34

3.9.2 Component Diagram..34

3.10 SD Card Module……………. ……………..…………..…………………………….....…………….35

3.10.1 Specifications…………….. …………..…………...……………………...…..……………..35

3.10.2 Component Diagram……………. ………….…..…………..………………….…..……………..35

3. 11 Cooling Fan 36

3.12 Thermoelectric cooling Peltier 36

Chapter 4 - Working Principle 38

4.1 What are Smart IOT Based Mentoring Systems? 38

4.2 How does Smart Mentoring Systems work? 39

4.3 Block Diagram 39

4.9 Flow Chart 42

Chapter 5 - Implementation and Results 43

5.1 Design and Software Integration... 43

5.2 Implementation 44

5.2.1 Programming the STM32F103C8 Microcontroller 44

5.2.2 Constructing the Circuit 45

5.2.3 Mechanical Design 45

5.3 Results.. 46

Chapter 6 - Conclusion and Future Recommendations.. 50

6.1 Conclusion.. 50

6.2 Future Recommendations... 50

References... 51

Appendix………………………………………………………………………...…………………....….55

LIST OF FIGURES

Figure 1.1 Proposed architecture of monitoring system.......16

Figure 2.1 CAD design of monitoring box and complete system.......21

Figure 2.2 System Architecture.......24

Figure 3.1 STM32F103C8.......26

Figure 3.2 STM32F103C8 Pin Description.......27

Figure 3.3 STM32 cube IDE Software.......27

Figure 3.4 Fritzing Software circuit example.......28

Figure 3.5 ESP32 Module.......29

Figure 3.6 ESP32 Module Pin Description.......29

Figure 3.7 ESP82 Module.......30

Figure 3.8 ESP82 Module Pin Description.......30

Figure 3.9 20 x 4 LCD Display.......31

Figure3.10 DS18b20 Temperature Sensor.......32

Figure 3.11 Pressure Sensor KH100C.......32

Figure 3.12 Flow rate Sensor YF-S201.......33

Figure 3.13 Solar Charge Controller.......34

Figure 3.14 Buck Converter.......34

Figure 3.15 SD card Module.......35

Figure 3.16 Cooling Fan.......36

Figure 3.17 Cooling Fan.......37

Figure 4.1 Smart oil and gas well monitoring system.......38

Figure 4.2 Flow chart of the system.......39

Figure 4.3 Mobile application through MIT App Inventor.......41

Figure 4.4 Block Diagram of Monitoring System.......42

Figure 4.5 Temperature control circuit of thermistor……………………..………………………...…..42

Figure 5.1 Electronics design in the Fritzing software…………………….........................……..43

Figure 5.2 Inside View of Monitoring Box with Electronics………………………………..…...…...44

Figure 5.3 Mechanical Design of the system…………………………....…...……………………..……46

Figure 5.4 Parameters Values on the LCD Display on Monitoring Box…………..……...……47

Figure 5.5 Tank with Sensors Mounted on it………………………………..………………………...…47

Figure 5.6 Complete Monitoring System in Running/Working Condition……………......……...48

Figure 5.7 Data Visualization on the Web (Firebase)………………………...……………………..…48

Figure 5.8 Data Visualization on the Phone (Application)……………………………..……...………59

LIST OF TABLES

Table 2.4.1 Voltage and Current Ratings of Components..22

Table 2.4.2 Power Ratings of Components..23

Chapter 1 - Introduction

1.1 Problem Statement

The Oil and Gas industry plays a crucial role in providing resources for manufacturing, construction and tech development. However, it also involves significant operational expenses and carries a high risk of asset safety and operational failures. Industries such as chemicals, petroleum, oil and gas are at high risk for fire outbreaks, which can lead to significant property loss, destruction, and most important, the loss of human life[1]. Having a reliable security system is crucial for maintaining a secure premises and notifying authorized individuals in the event of an incident. Monitoring and controlling oil field operations remotely in real-time is crucial for ensuring safety and efficiency, now more than ever. Oil field operations' conventional monitoring and control systems are usually centralized, which makes them susceptible to failure and inefficient[2].

The oil and gas industry is subject to strict regulations and requires significant capital investment. Numerous operations are carried out in harsh and remote environments. Although globally the demand of crude oil is high, oil and gas operators face challenges due to low oil prices and increasing regulatory constraints. As a result, they must be more innovatively advance in order to increase efficiency and reduce costs of Oil and gas. Reliable monitoring of wellheads is necessary to ensure they can operate for extended periods without shutting down. This also helps to increase the lifespan of wellhead equipment, ultimately preventing significant profit losses. Performing well head maintenance activities and physically inspecting every well can be challenging, as it requires sending employees to each location[1].

One solution to address these problems is to develop an IoT-based monitoring system for oil and gas wells. This system would involve sensors that gather precise data on pressure, temperature, and flow from the wells. After processing this data, it would be sent to the cloud for further analysis. Sensors are commonly installed on wellheads in architecture to provide essential process data and detect any abnormal events such as oil and gas theft, fire, and gas incidents[3]. This enables timely and reliable reporting of such incidents. Smart monitoring systems can optimize pumping operations, maintain pipes and wells, detect equipment failures based on well performance, minimize health and safety risks, and monitor and improve production performance while reducing costs.

1.2 Aims and Objectives

The aim of the project is to mainly focus on development of a remote oil well monitoring system to track the production rate, temperature and pressure remotely and inform Managers to make decisions timely. The objectives of remote oil well monitoring System are following:

- Designing of the conceptual model for remote monitoring system.
- Designing of the electronics system for remote monitoring system.
- Interfacing of Mechanical Part and Electronics Hardware
- Development of a data acquisition system for monitoring production parameters, temperature and pressure of the well
- Testing of the designed Monitoring System

1.3 Overview

Due to asset, process, and operational failures, the oil and gas industry regularly faces serious challenges at high operating costs. They also lack open communication and effective cooperation. Production facilities for oil and gas are quite numerous and have a broad dispersion. Manual data collection is thus challenging and time-consuming. Because the pumping unit runs constantly around the clock, manual data collection is severely limited. In short, conventional oil production facilities use antiquated information monitoring and management methods[4].

Some wells are opened at the bottom and closed at the peaks in order to achieve precise control and lower the cost of power. Early in the morning, frontline workers must shut the well, and also at this time, the personnel on duty may not be accessible and more prone to safety mishaps. Additionally, there is a delay in halting the well, which lengthens the time it takes to open an invalid well and uses more energy. The aforementioned issues significantly slowed well opening times and decreased oil recovery rates. Therefore, a mobile remote monitoring system was designed to ease the operation of frontline staff, decrease the number of well visits, immediately and efficiently monitor the conditions of oil wells, and increase the level of field management. On the other hand, an oil and gas corporation is exposed due to the sheer number of cyber threats[5]. Cyber-criminals utilize malware and viruses that are particularly designed to damage networks, control systems, or servers that contain incredibly important data like scientific findings, tedious processes, fresh oil reserves, and the chemistry of premium items. Therefore, industries are more willing to pay the ransom in order to retrieve lost data[6].

The ultimate objective is to better output by keeping an eye on and managing the oil and gas wells. In order to fulfil the required criteria and avoid the drawbacks, edge processing-based IoT solutions are becoming more necessary. Figure 1.1 depicts the basic idea of the suggested architecture, which provides prompt data gathering for remote conditional monitoring, the optimal control actions to minimize facility downtime, and machine-to-machine links for improved automation and control[7]. Despite the fact that a lot of work has been proposed for IoT-based systems where sensors collect sensors values of pressure, flow rate, and temperature from oil and gas wells and then transmit the processed data to the cloud.

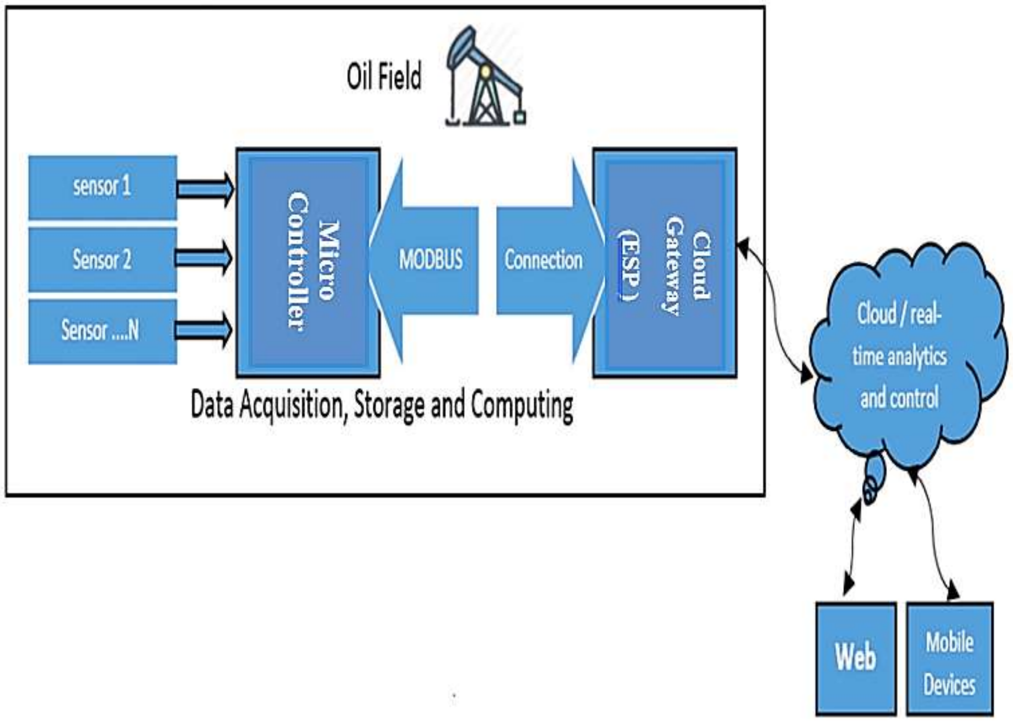


Figure 1.1 Proposed architecture of monitoring system

1.4 Scope of Study

The project is limited to an enclosed system (model house), making use of a fan and a heater with the circuit systems in it.

The scope of this project is;

- To design a monitoring system that fits for monitoring the oil and gas well.
- Real time data collection and analysis by implementing IOT based infrastructure.
- Secure communication system for data transmission to the cloud.
- Efficient organization and analysis of collected data by storing the data.

1.5 Significance of Project

This project value cuts across various fields, below are the advantages and application of this project.

- It can change traditional monitoring to remote monitoring and reduces cost of the oil well monitoring.
- It is easy for manager to get updated with the well's condition at any time through smart phone or PC.
- Well performance history will be put on the webserver and the monitoring unit can get to the data at whatever point required from any place.
- Easy to use the system at anytime and anywhere for professionals.
- The whole system can save a lot of time and money[5].

1.6 Technological Significance

The technological significance of this project is seeking for a prototype of a workable remote oil and gas well monitoring system to monitor the pressure, Temperature and Flow rate of an oil and gas well using a Pressure, Temperature, Flow rate sensor and a microcontroller, which could be used or enhanced by future developers[8].

1.7 Methodology

The methodology for the accomplishment of our project is as follows:

- ***Literature review:*** This is the first and basic part of our project execution methodology in which we analyze and gather relevant research, by studying research papers as well as general information that is already done.
- ***Conceptual Designing:*** We will be completing our conceptual designing part of the system by making block diagrams and flow charts of the monitoring system, a 3D model for the system and electronic circuits. Further, we will be moving towards testing of mechanical system.
- ***Designing and Testing of Electronics:*** We will be completing our electronics designing part by making electronic circuits for the system in Fritzing software and simulating and to making sure that the components are most efficient in every way for the best monitoring of the well. Further, we will be moving towards selection of components.
- ***Selection of components:*** The most important part is the selection of best and compatible components for our use to control the whole system. We will be selecting the components very carefully so that the system works at its full potential.
- ***Interfacing electronics and cloud programming of System:*** We will be interfacing the designed electronic circuits and programming the system for monitoring the required parameters of the well and transmitting the parameters to cloud.
- ***Fabrication of system:*** The final fabrication of the whole system would be done where the prototype would be ready for final experimentation and testing.
- ***Experimentation and testing***: We will be testing our final model and completed system that will qualify the working of all objectives defined.

1.8 Background and Related work

Employees used to need to go to the well site for essential diagnosis in the previous old traditional methods. The main issue with this strategy is that the employee has to be on field for the well monitoring for 24 hours[9]. The time to action in IoT, however, is measured in minutes, seconds, or microseconds. IoT equipment generates a lot of data and information. By recoding and gathering monitoring data, IOT-based monitoring services are becoming better and more affordable. Taking decisions on how the performance is going and what to do next is made easier by having vast amount of real-time data access o performance of well. Additionally, it permits the monitoring device to use the data collection's findings. In real time, it links people to things and things to people. These gadgets can autonomously collect data from equipment sensors and relay it to management without the need for human input[9].

According to ***McNeill et al. (2018),*** wellheads are subjected to high pressure, temperature and flow rates. Noted that employing monitoring systems on oil well's take more response time than IoT based monitoring system, make complexity of analysis of collected data[4]. ***Wang et al. (2018)*** summarized his research that developments in the field of monitoring of oil and gas wells, by using the typical monitoring systems scope and latest sensor-based technologies. Observed that latest sensor-based performance gave better results/output[10]. ***International Research Board 2015*** Executive Committee discussed the real-time monitoring (or smart monitoring). Their study concluded that there are financial returns using Smart systems[2].

A system that is based on wireless sensor networks and connected to the internet of things has been developed as a means of monitoring the productivity of wells. This system, which is designed to transmit and manage decisions about the wells performance ***(Pan Yi et al 2021)***[10]**.** An intelligent control system that is based on a sensor network is the best answer for the problem of monitoring oil and gas wells without resorting to human monitoring. This method has been presented as a solution. A wireless transmission sends data about the wells and tanks to a remote administrator, who may use this data to assess the overall health of a number of oil wells and oil storage tanks. The system makes use of level, temperature, and gas sensors in various configurations. ***(R. Barani et al in 2020)***[11]**.**

Chapter 2 – Basic Description

2.1 Conceptual Approach and Design Objectives

In order to improve oil and gas production, we plan to design a remote monitoring system for oil and gas wells. This system will allow for timely data collection from afar and offer advantages like decreased facility downtime and better automation and control through machine-to-machine connections. Due to the 24/7 nature of the pumping unit, this method is preferable to manual data collecting, which is labor-intensive and time-consuming. In order to optimize efficiency and production, its necessary to maintain oil and gas wells regularly, monitor equipment failures based on well performance, minimize risks to health and safety, and monitor and improve production performance at reduced costs, it is necessary to design an Internet of Things (IoT) based monitoring system for oil and gas well that utilizes sensors to obtain detailed data wells parameters and then send data to the cloud for processing[12].

Several objectives are derived from analyses of existing oil and gas well monitoring systems practices.

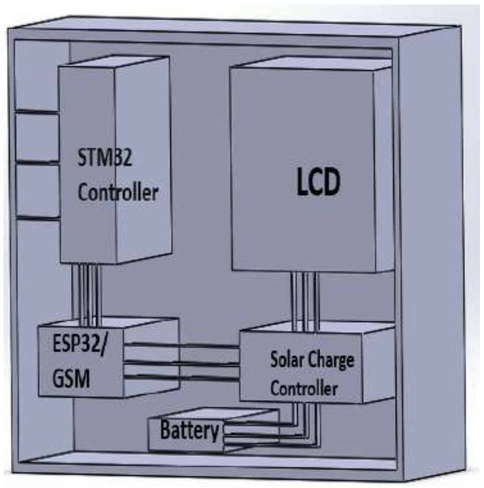
- **Predictive Maintenance:** Internet of Things (IoT) can collect real-time data and information from upstream industries to foresee potential breakdowns. Upstream, midstream, and downstream facilities may all benefit from predictive and preventative maintenance, which schedules work based on past results[13].
- **Health and Safety:** Accidents are expensive and a priority for health and safety. In the oil and gas industry, companies may reduce risk to their business and their workers by using IoT-enabled safety measures.
- **Remote Monitoring:** Remote monitoring of well to track pressure, temperature and flow rate of well remotely.
- **Data Management:** Oilfield, pipeline, and other energy site data management and analytics are critical to the O&G industry's performance. Improved uptime and recovery rates, more informed decision making, and more refining capacity are just some of the ways that collected data may be put to use (Lee, 2019)[8].

2.2 Description of Prototype

We built a prototype of oil and gas well monitoring system infrastructures in the oil and gas sector based on our conceptual approach. The prototype is made up of a smartphone application, a cloud server in the backend that stores data, and the following parts that are deployed in the site/Field:

- Connected pressure, temperature and flow rate sensor that provide pressure, temperature, and flow rate measurement data.
- A STM32F103C8 microcontroller to process the measurement data of sensor and for cloud transfer, send data to ESP module.
- A 16x4 LCD mounted on the monitoring box to display the measurement parameters.
- ESP32 module then transmit the measured data to cloud to visualize the data on a website and on the mobile application.
- ESP82 module is used to store the data in the form of google sheet for better visualization.
- Firebase an open-source software is used to visualize the data on the web.

2.3 CAD Design



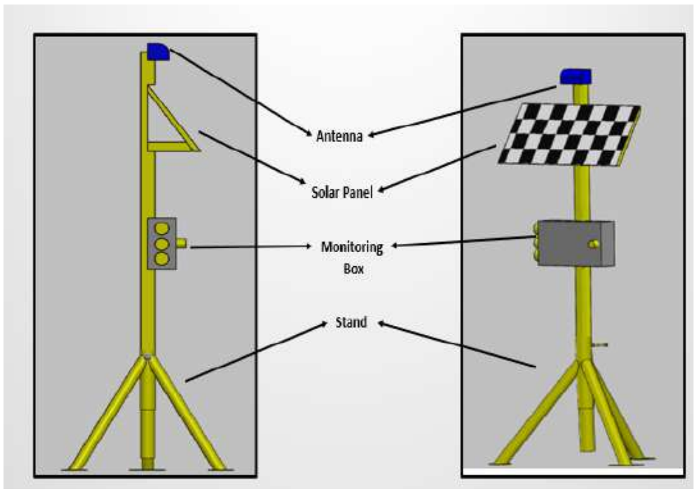


Figure 2.1 CAD design of monitoring box and complete system (solar panel is optional)

2.4 Power Consumption of System

2.4.1 Voltage and Current Ratings of components:

Sr. No.	**Components**	**Voltage and Current Ratings of components**
1	STM32 Microcontroller	**Voltage:** 1.8-3.6v **Current:**10-200mA
2	ESP32 Module	**Voltage:** 2.2-3.6v **Current:**280mA
3	ESP82 Module	**Voltage:** 2.2-3.6v **Current:**280mA
4	SD Card Module	**Voltage:** 3.3-5v **Current:**100-200mA
5	16x4 LCD	**Voltage:** 5v **Current:**45mA
6	Cooling Fan	**Voltage:** 12v **Current:**100-500mA
7	Peltier	**Voltage:** 12v **Current:**6A
8	Pressure Sensor (HK-100C)	**Voltage:** 5v **Current:**1-10mA
9	Temperature Sensor (DS18b20)	**Voltage:** 3.3-5v **Current:**1mA
10	Flow Rate Sensor (YF-S201)	**Voltage:** 5-24v **Current:**15mA

Table 2.4.1 Voltage and Current Ratings of Components

2.4.2 Power Consumption of components:

Sr. No.	Components	Power Consumption of components
1	STM32 Microcontroller	Power: 3.6v x 0.2A= 0.72W
2	ESP32 Module	Power: 3.6v x 0.28A=0.1W
3	ESP82 Module	Power: 3.6v x 0.28=0.1W
4	SD Card Module	Power: 5v x 0.2A=1W
5	16x4 LCD	Power: 5v x 0.045A=0.225W
6	Cooling Fan	Power: 12v x 0.5A=6W
7	Peltier	Power: 12v x 6A=72W
8	Pressure Sensor (HK-100C)	Power: 5v x 0.10A=0.05W
9	Temperature Sensor (18b20)	Power:5V x 0.001A=0.005W
10	Flow Rate Sensor (YF-S201)	Power: 5V x 0.015=0.075W

Table 2.4.2 Power Ratings of Components

Now by adding the power of all the components, we can calculate the total power consumption of the system:

Total Power Consumption = 0.072W + 0.1W + 1W + 0.225W + 6W + 0.05W + 0.005W + 0.075W+72W = **80.0372W**

Therefore, the combined power is **80.0372 watts.**

2.4.3 24h Energy consumption by the system:

Energy consumed by a system in 24 hours, By using the formula:

Energy (Wh) = Power (W) x Time (h)

Time = 24 hours

Power = 80.0372W

Energy = 80.0372W x 24 h = **1920.8928Wh**

Therefore, system will consume 1920.8928Wh of energy in 24h.

2.5 The Remote Oil and Gas Well Monitoring System

The remote oil and gas well monitoring system comprises of three (3) main subsystems: Sensing unit, Processing unit and Cloud transmission Unit.

- ***Sensing unit:*** The sensing unit comprises of three sensors, pressure, temperature and flowrate sensor that measures the pressure, temperature and flowrate of well in the real time and continuously[14].
- ***Processing Unit:*** The processing unit comprises of a STM32F103C8 microcontroller and act as the brain of the monitoring system. Microcontroller process the sensors data values and display on the LCD and also control the temperature of the monitoring box by using a fan, Peltier, and a thermistor[15].
- ***Cloud Transmission unit:*** Comprises of ESP32 and 82 module, STM32 microcontroller send the processed data of sensors values to the ESP32 module and ESP32 transmit the data to the firebase server and ESP82 store data in the google sheet.

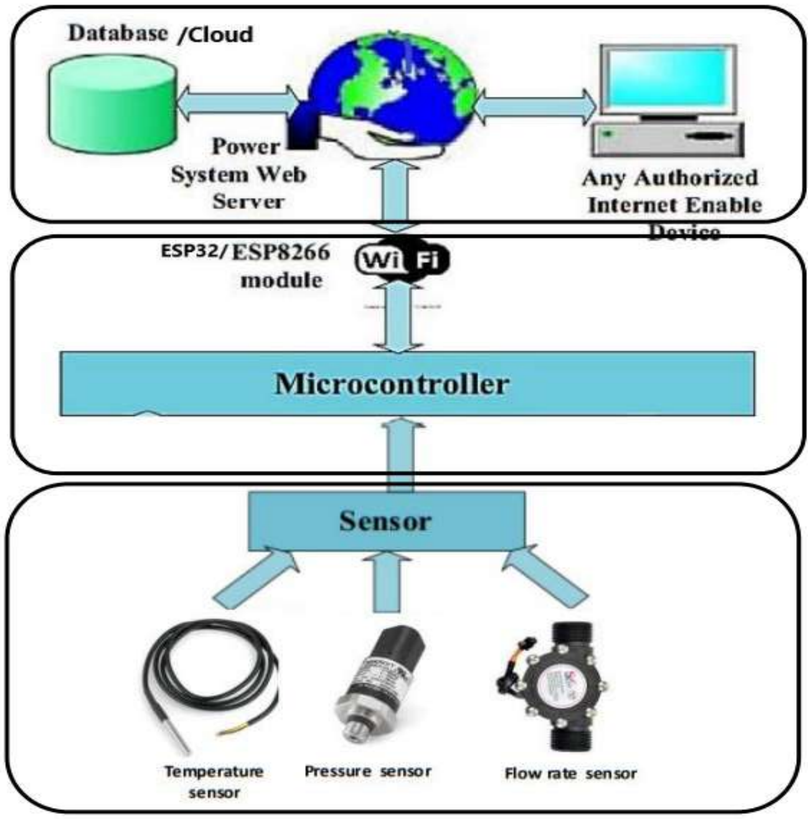


Figure 2.2 System Architecture

2.6 PSEUDOCODE: STM32 and ESP Module PSEUDOCOD

```
Initialize ports and variables
Read Temperature from the sensor
Read Pressure from the sensor
Read Flowrate from the sensor
Display temperature, pressure and flowrate on the LCD
Measure the environment temperature (actual temperature)
If Temperature Reference is less than actual Temperature, Switch ON fan and Peltier
If Temperature Reference is greater than actual Temperature, Switch OFF the Fan and Peltier
Send data values (pressure, temperature and flowrate) to the ESP32/82 Module
ESP32 send the data values to the cloud server
ESP82 get the data values and store the data in the form of google sheet
```

The C/C++ programming language was used to write the code for the microcontroller, and the STM32cube IDE was used to build the code. STM32CubeIDE is a powerful C/C++ programming platform to program STM32 controllers, its features include debugging, code creation and compilation. Then the code file created by the software uploaded in electronics hardware to test/check that it works or not.

.

Chapter 3 - Components

Electronics Components

3.1 STM32F103C8 Microcontroller

3.1.1 Introduction

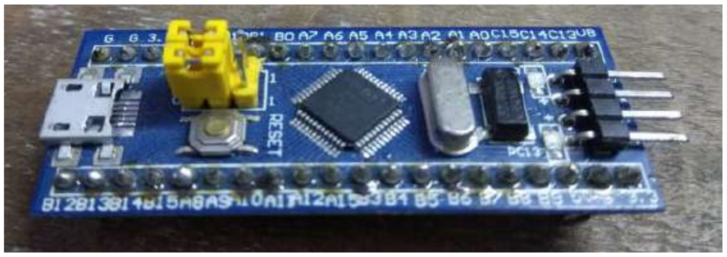
An electrical component that is a member of the microcomputer family is referred to as a microcontroller. It performs the function of the project's brain. The question of how to choose the microcontroller has arisen, and the answer will be determined by the project's goals and tasks, in addition to its memory and processing speed. In the course of this research, we make use of a variety of sensors and other components that assess the state of health of the patient. A micro-controller is the one responsible for making all of these decisions. In order to fully automate the project, which does not need any involvement from a person, microcontrollers are used.

The purpose of the STM32 controller in this monitoring system is to act as the central processing unit, responsible for collecting data from the pressure, temperature, and flow rate sensors. It processes and manages the sensor data, enabling real-time analysis and control.

3.1.2 Specifications

- 3.6 V power supply
- 4-16 MHz oscillator
- USARTs interface
- USB 2.O interface
- I2C interfaces

3.1.3 Component Diagram



. Figure 3.1 STM32F103C8

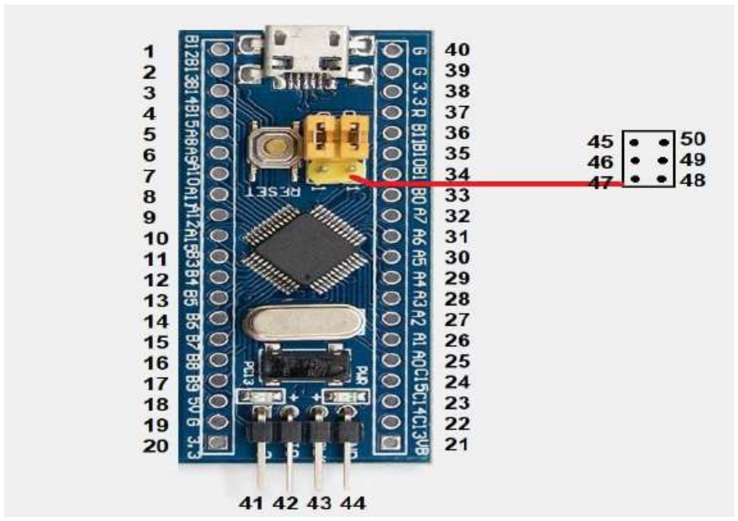


Figure 3.2 STM32F103C8 Pin Description

3.1.4 Software for STM32F103

STM32CubeIDE is a powerful C/C++ programming platform for STM32 microcontrollers and microprocessors. Its features include peripheral setup, code creation, code compilation, and debugging capabilities. The Eclipse / CDT framework, the GCC toolchain, and the GDB debugger are used throughout the development process. And this IDE is used for the STM32 controller Programming.

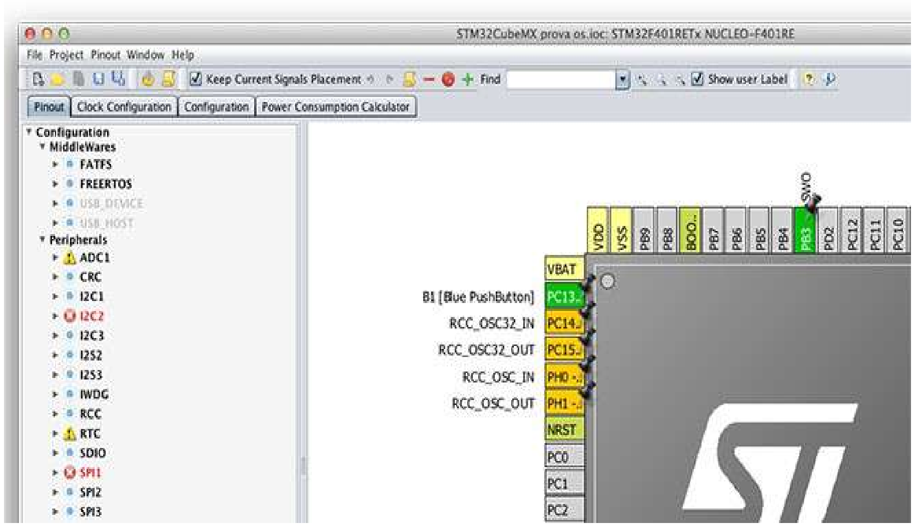


Figure 3.3 STM32 cube IDE Software

3.1.5 Fritzing Software

The development of amateur or hobby CAD software for the design of electrical hardware is the focus of the Fritzing open-source project. This software is designed to make it possible for designers and artists to construct more permanent circuits from prototypes. The University of Applied Sciences in Potsdam was the institution responsible for its creation. The source code for Fritzing is hosted on GitHub, and although the GPL does allow for a fee to be charged for the binaries, the program itself is free to use and is licensed by a later version of the GNU Public License (GPL). Fritzing is used for making circuit of the monitoring system.

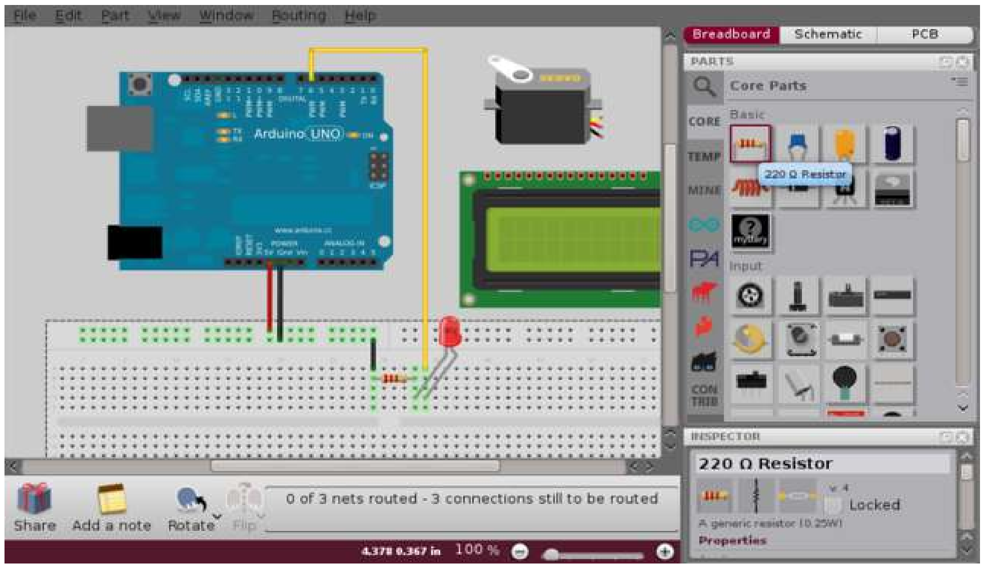


Figure 3.4 Fritzing Software circuit example

3.2 ESP32 Module

The ESP32 is a system-on-a-chip (SoC) microcontroller family that combines cheap price, low power consumption, and built-in Wi-Fi and dual-mode Bluetooth. Built within each ESP32 device are antenna switches, an RF balun, a power amplifier, a low-noise receive amplifier, filters, and power-management modules. The ESP32 module in this monitoring system serves as a communication gateway between the local microcontroller (STM32) and the cloud platform (Firebase). It enables wireless data transmission over Wi-Fi.

3.2.1 Specifications

- Dual Core 32bit micro-processor
- Bluetooth v4.2

- Two 8-bit DACs
- Internal Temperature Sensor
- Support up to 16MB flash.
- Built in USB-to-UART Bridge.
- Wi-Fi: 802.1

3.2.2 Component Diagram

Figure 3.5 ESP32 Module

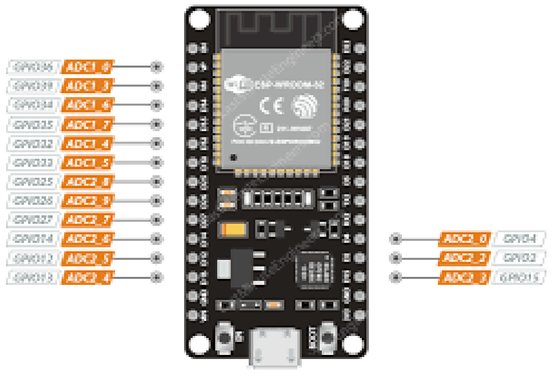


Figure 3.6 ESP32 Module Pin Description

3.3 ESP82 Module

ESP8266 module, microcontrollers are able to connect to Wi-Fi networks operating at 2.4 GHz. Either it may be used as a self-sufficient MCU by running an RTOS-based software development kit or it can be used with ESP-AT firmware to give Wi-Fi connection to external host MCUs. Both of these uses are possible. The ESP32 module in this monitoring system serves as a communication gateway, facilitating wireless data transmission from the STM32 microcontroller to the cloud (Firebase). It enables seamless connectivity and real-time updates, ensuring that the sensor data is efficiently delivered for remote monitoring.

3.3.1 Specifications

- 32-bit RISC CPU
- 3.3V Operating Voltage
- 4 MB Flash Memory
- 80 MHz Clock Speed
- 16 Digital I/O Pins

3.3.2 Component Diagram

Figure 3.7 ESP82 Module

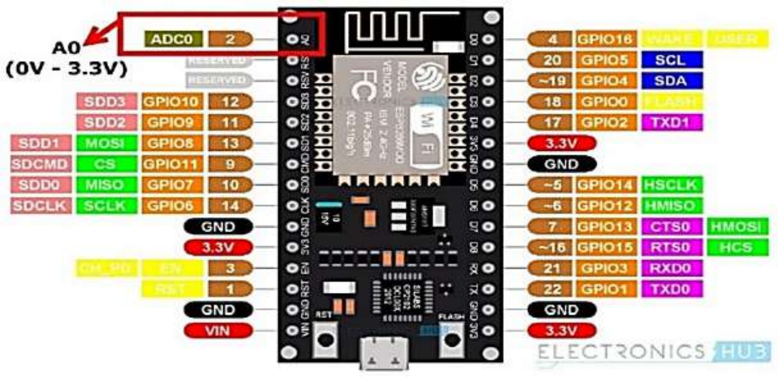


Figure 3.8 ESP82 Module Pin Description

3.4 16x4 Liquid Crystal Display (LCD)

A liquid crystal display (LCD) is a kind of electrical modulated optical device that employs liquid crystals and polarizers to create an image on a flat screen. Characters may be seen on the character display. The graphical LCD displays text and graphics that the user specifies. The purpose of the 16x4 Liquid Crystal Display (LCD) in this monitoring system is to provide a local interface for on-site personnel to access real-time well data directly from the microcontroller.

3.4.1 Specifications

- 146 x 62.5 x 14 mm Module Size
- 23 x 42.66 mm Viewing area1
- 4.85 x 9.15 mm Character size
- 0.98 x 1.15 mm Dot size
- 8bit parallel interface
- -20 to 70 ^{o}C Operating Temperature

3.4.2 Component Diagram

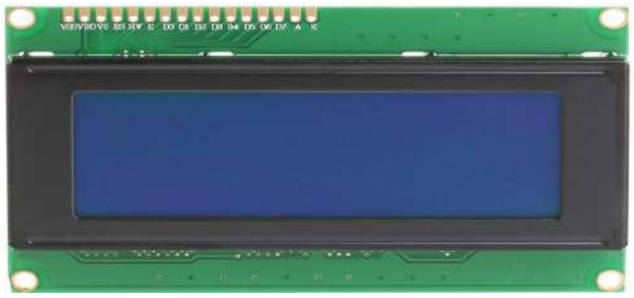


Figure 3.9 16 x 4 LCD Display

3.5 Temperature Sensor (DS18b20)

The DS18B20 is one kind of temperature sensor, and the values of temperature that it provides may range from 9 bits to 12 bits. These numbers indicate the temperature of the specific equipment being discussed. This sensor is capable of communicating with an internal microprocessor via the use of a one-wire bus protocol, which only requires the usage of a single data line for transmission. The purpose of the Temperature Sensor (DS18B20) in this monitoring system is to measure and provide accurate temperature data of the oil and gas well.

3.5.1 Specifications

- 3V to 5V Operating voltage
- -55°C to +125°C Temperature Range
- Digital Temperature Sensor
- Communicates through one Wire

3.5.2 Component Diagram

Figure 3.10 DS18b20 Temperature Sensor

3.6 Pressure Sensor (KH100C)

A pressure sensor is a device for pressure measurement of gases or liquids. The force that is necessary to prevent a fluid from expanding is the force that is referred to as the pressure. The function of a pressure sensor is typically that of a transducer; it produces a signal that is proportional to the pressure that is being applied. A signal of this kind is considered to be electrical for the purposes of this article. The purpose of the Pressure Sensor (KH100C) in this monitoring system is to measure and monitor the pressure of the oil and gas well.

3.6.1 Specifications

- 5v Working Voltage
- <=10 mA Working Current
- - 3.0 %FSO Temperature Range Error
- <=2.0 MS Response Time
- 1.5 MPa The Biggest Pressure
- - 1.0 %FSO Measuring Error

3.6.2 Component Diagram

Figure 3.11 Pressure Sensor KH100C

3.7 Flow Rate Sensor YF-S201

The YF-S201 is a water flow measuring sensor that has a quality sealing property of a very high standard. The flow rate range is from 1 to 30 litres per minute, and it operates using the Hall effect concept. The module has three pins, which are labelled Power, Ground, and the Analogue output respectively. The YF-S201 has a very low current consumption rate and can function with an operating pressure of up to 1.75 MPa. The purpose of the Flow Rate Sensor YF-S201 in this monitoring system is to measure and monitor the rate of fluid flow (e.g., oil or gas).

3.7.1 Specifications

- 4.5V-18V Operating Voltage
- 15mA at 5V Maximum current draw
- ±10% Accuracy
- 2.0 MPa Maximum water pressure
- 1 to 30 Liters/Minute Working Flow rate

3.7.2 Component Diagram

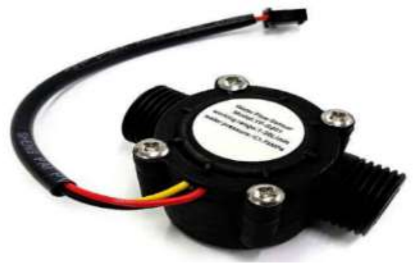


Figure 3.12 Flow rate Sensor YF-S201

3.8 Solar Charge Controller

A solar system's solar charge controller has the ability to automatically regulate the functioning of the solar panels and batteries in the system. This is done to safeguard the lifetime of the battery. If you do not have a solar charge controller, your battery will not be protected. The purpose of the Solar Charge Controller in this monitoring system is to regulate and control the charging of the solar panels' battery or energy storage unit.

3.8.1 Component Diagram

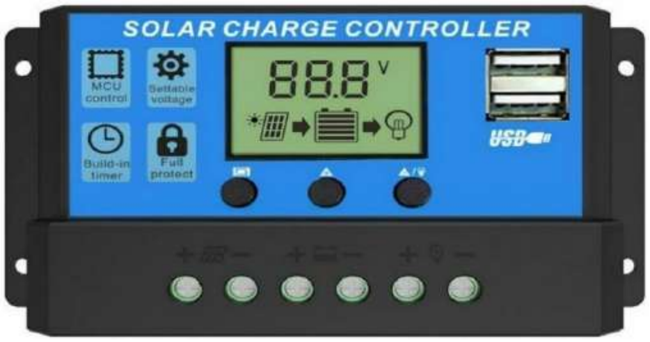


Figure 3.13 Solar Charge Controller

3.9 Buck Converter

A step-down (buck) switching regulator DC-DC Buck Converter is used to Step Down Power Supply to a acceptable level. The LM2596 Buck is able to drive a load of 3 A and has outstanding line and load control. The purpose of the buck converter in this monitoring system is to efficiently step down the voltage from a higher source (e.g., battery or power supply) to a lower voltage level required to power the sensors, microcontroller, and other low-voltage components.

3.9.1 Specifications

- 1.25V-35V Output voltage
- 65KHz Switching frequency
- -45 - +85Working temperature
- 3A Output current

3.9.2 Component Diagram

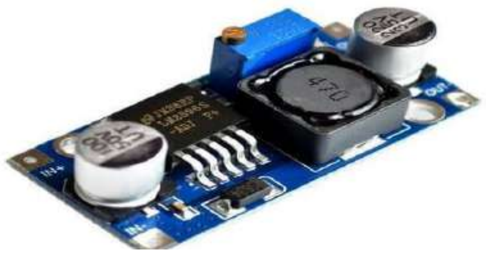


Figure 3.14 Buck Converter

3.10 SD Card Module

A microcontroller may read and write data to an SD card using a breakout board called an SD Card Module. This device is used for SD card operations. It is possible to use the board with microcontroller-based systems. You may easily plug a conventional SD card into the board; however, if you need to utilize microSD cards, you will need to make use of an adapter. The purpose of the SD Card Module in this monitoring system is to provide a local data storage solution. It allows the system to log and save sensor data directly to an SD card, ensuring data continuity and enabling offline data access when the internet connection is unavailable.

3.10.1 Specifications

- 3.3V-5V supply Voltage
- 0.2-200mA Current
- Supports Micro SD up to 2GB
- Pinouts GND, VCC, , SCK, CS ,MISO, MOSI for SPI interface

3.10.2 Component Diagram

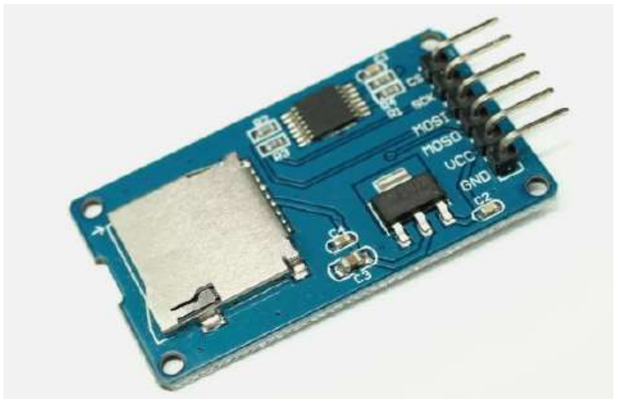


Figure 3.15 SD card Module

Mechanical Components

3. 11 Cooling Fan

The cooling fan maintain a consistently low operating temperature by cooling individual components by circulating air over a heatsink. It is essential that your computer be equipped with a cooling fan in order to remove excess heat and ensure that it remains in a cool condition at all times. This will prevent the components from being damaged in any way.

3.11.1 Specifications

- 12V Rated Voltage
- 6500 ±10%rpm Rated Speed
- 15±10%dB Noise
- 0.02 Amp Rated Current

3.11.2 Component Diagram

Figure 3.16 Cooling Fan

3.12 Thermoelectric cooling Peltier

The Peltier module thermoelectric module is a kind of thermal control module that may either warm or cool the surrounding air. It is possible to modify the surface temperature of the module and retain it at the desired temperature by feeding an electric current through the module. The "Peltier effect" is the name given to the phenomenon that occurs when electrons move in one element and positive holes move in the other element as a result of the passage of electricity through the module. Because of this, it is possible for one side of the substrate to absorb heat while the other side radiates heat; hence, the hot and cold sides of the substrate may be swapped depending on the direction in which the current flows. The purpose of Thermoelectric cooling Peltier in this monitoring system is to maintain optimal operating temperatures for the electronic components, especially the sensors and microcontrollers.

3.12.1 Specifications

- 10A Max Current
- 16V Max Voltage
- >70°C Difference Range
- 101.1W Max Refrigerating Power

3.11.2 Component Diagram

Figure 3.17 Cooling Fan

Chapter 4 - Working Principle

4.1 What are Smart IOT Based Mentoring Systems?

SMART stands for "Self-Monitoring, Analysis, and Reporting Technology," which is something that we have already shared with you. However, what precisely is it that makes a person smart?

The analysis of dynamic systems and the processing of billions of events and alerts data packets are both part of the Smart IoT monitoring process. Monitoring that is based on the Internet of Things also makes it possible to close the gap between devices and people by collecting and analysing a wide variety of IoT data at web scale across all linked devices. You should also bridge performance gaps by optimising performance across numerous apps, APIs, networks, and protocols, and you should also get actionable insights to enhance operational experience, correct issues, and maximise efficiency[16].

In the Internet of Things-based oil and gas well monitoring system, sensors collect precise data (such as pressure, temperature, and flow) from oil and gas wells, and then, after processing the data, transmit it to the cloud so that it may be visualised on the web or in an application[17].

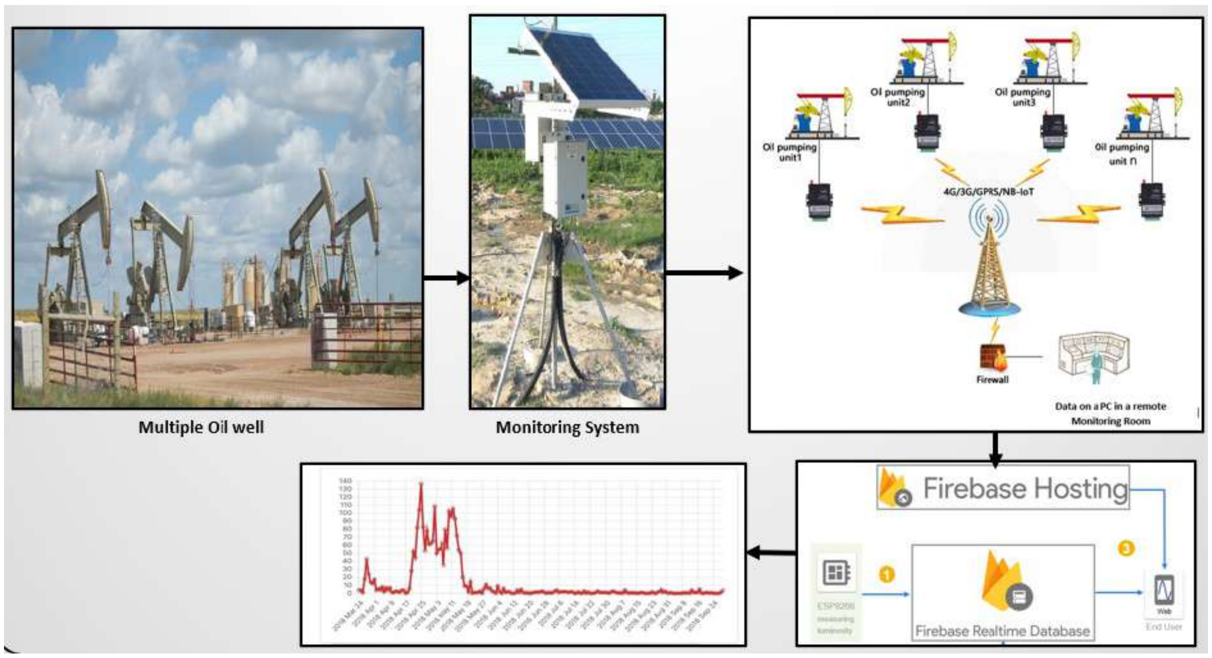


Figure 4.1 Smart oil and gas well monitoring system

4.2 How does Smart Mentoring Systems work?

Web-enabled smart devices with embedded systems like CPUs, sensors, and communication gear make up a smart IoT ecosystem that can gather, transmit, and act on data from its surrounding environment. Through a connection to an IoT gateway or other edge device, data collected by the sensors may be displayed on a LCD and either transferred to cloud servers for remotely analysis or analyzed locally. Some of these devices/controllers are able to communicate to one another and take action based on the measured parameters data of the well. Most of the work is done automatically by the devices, but humans still need to engage with them in some way, for example to set them up, provide instructions, or retrieve the data[18].

4.3 Block Diagram

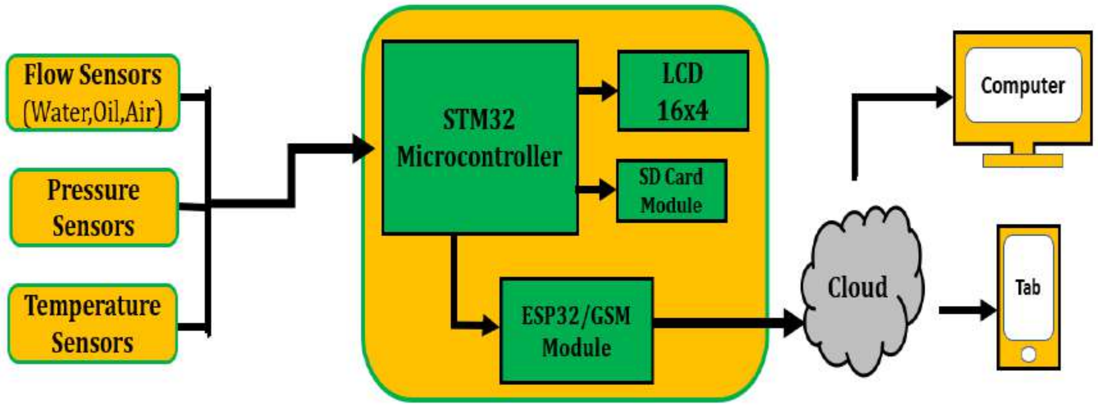


Figure 4.3 Block Diagram of Monitoring System

4.4 Sensing Layer

Sensing devices, which mediate between the real and the virtual, make up the foundation of the system. Various meters, gauges, sensors may read parameters like temperature, humidity, etc. on the physical layer, which is referred to as "perception." These gadgets are placed at the system's terminals in order to get the raw data that represents the necessary parameters.

The oil well monitoring system's sensing unit consists of a pressure sensor, a temperature sensor, and a flowrate sensor, all of which are used to constantly and in real time gauge the well's pressure, temperature, and flowrate[19].

4.5 Processing Layer

In the Processing layer, the microcontroller receives the sensor data, processes it, and then either sends the data elsewhere or produces signals for actuators. Communication interfaces including I2C, SPI, UART, and analogue signals are used by sensors to provide data to a microcontroller. Data values from the sensors' parameters (Pressure, Temperature, and Flowrate) are sent from the processing and the LCD display to the microcontroller in the monitoring system. The data from the sensors is sent to the cloud via the microcontroller through the ESP module[20].

Using a fan, Peltier, a thermistor, and a transistor, the microcontroller regulates the environmental temperature of the monitoring box. When the temperature drops, the thermistor's resistance goes up, and when the temperature rises, the thermistor's resistance goes down. The thermistor's resistance is 10kohms when it's at room temperature. A transistor links the fan and Peltier to their respective circuits. The fan will now activate when the temperature inside the box rises and the resistance drops. As the temperature drops, the fan will stop running.

4.6 Network Layer

When it comes to the design of IoT systems, the network layer is in charge of the necessary communication between devices, networks, and cloud-based services. Gateways that translate signals across protocols are another viable option for realising this concept. The gateways in an IoT system may encrypt and decrypt data in transit. The network connectivity supports the transfer of measured parameters data from the gateway to the Cloud server, much as the sensor to gateway network system does. It's possible that this network might cover a large geographical region, enabling for data to be sent to far-flung areas. Ethernet, Wi-Fi, and cellular are the common protocols for a network connectivity[22].

4.7 Application layer:

Applications used by end users make use of data stored in the cloud at the application layer. Users may see the gathered data on their screens to keep tabs on the metric of interest to them. The data may be presented in a variety of graphical and numerical formats, allowing the user to easily extract meaningful insights and make well-informed choices[12]. Users and administrators may access reports and data analyses generated by the cloud through websites.

IoT platform with individualized capabilities such as real-time alerts, analytics, and remote monitoring. And we built the monitoring app using MIT's App Inventor mobile app development platform[23].

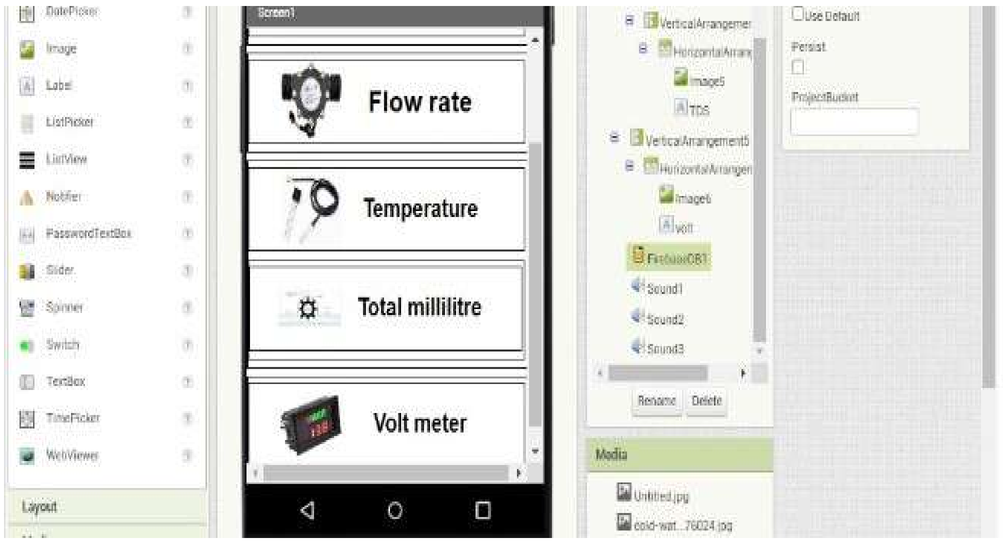


Figure 4.3 Mobile application through MIT App Inventor

4.8 Temperature Control of Monitoring Box

Other aspect of the switching unit is the base biasing of the transistor. And on the other hand, to control the monitoring box temperature a fan and Peltier is used along with a thermistor, and transistor.

Thermistors are a kind of temperature-dependent variable thermal resistor that are made of semiconductor material. Their resistance varies depending on the temperature. The typical temperature range for a thermistor's operation is between 55 and +150 degrees Celsius; however, certain glass body thermistors have a maximum functioning temperature of +300 degrees Celsius.

A 10k thermistor and a 10k resistor are linked to one another in order to construct a voltage divider circuit in each of the circuits. The NTC thermistor is what was utilized for this project. This kind of thermistor has a resistance that goes up when the temperature goes down, and it goes down when the temperature goes up. The thermistor has a resistance of 10 kohms when it is at normal temperature. The transistor is linked to the fan and the Peltier device. Now, as the temperature of the box continues to rise, the resistance will continue to drop, which will ultimately result in the fan turning on. The fan will be turned off at the point when the temperature falls below the set point.

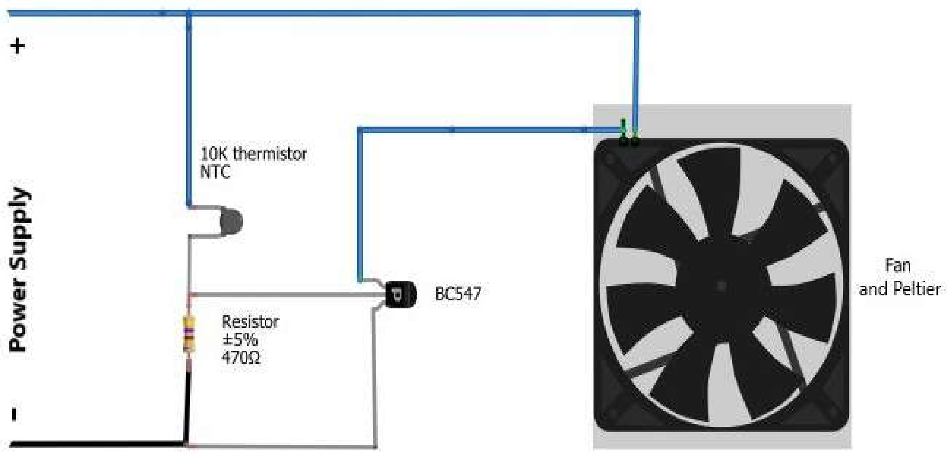


Figure 4.4 Temperature control circuit of thermistor

4.9 Flow Chart

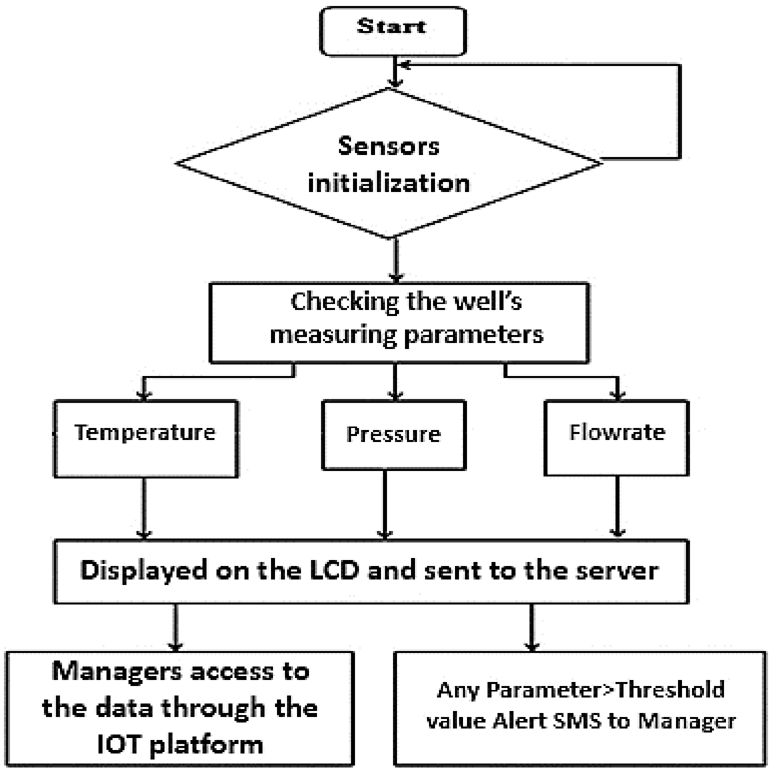

Start
Sensors initialization
Checking the well's measuring parameters
Temperature
Pressure
Flowrate
Displayed on the LCD and sent to the server
Managers access to the data through the IOT platform
Any Parameter>Threshold value Alert SMS to Manager

Figure 1.2 Flow chart of the system

Chapter 5 - Implementation and Results

5.1 Design and Software Integration

We have designed our system electronics and tested (Remote oil and gas well monitoring system) via using Fritzing and STM32 cube IDE software. It is seen that the calculated results (Power Consumption) agreed with the actual results. In figure, we show the electronic circuitry of the monitoring system. The STM32F103C8 microcontroller reads the temperature Pressure and flow rate of the well every 2s. Then the microcontroller after processing the sensors measurement data, display the data on a 16x4 LCD, mounted on the door of the monitoring box for on spot monitoring.

A fan, Peltier, a thermistor, and a transistor are used to regulate the temperature of the monitoring box in an extreme setting. The 10k Resistor and thermistor together make a voltage divider circuit. The thermistor's resistance is 10kohms when it's at room temperature. A transistor links the fan and Peltier to their respective circuits. Now, when the inside temperature of the box rises, the resistance will fall, and the fan and pettier will activate. When the temperature drops below a certain point, the fan and pettier will automatically shut off.

STM32 microcontroller send the processed data of sensors values to the ESP32 module and ESP32 transmit the data to the firebase cloud server and ESP82 module store data in the google sheet. The data can be accessed at any time and place through an internet supported device.

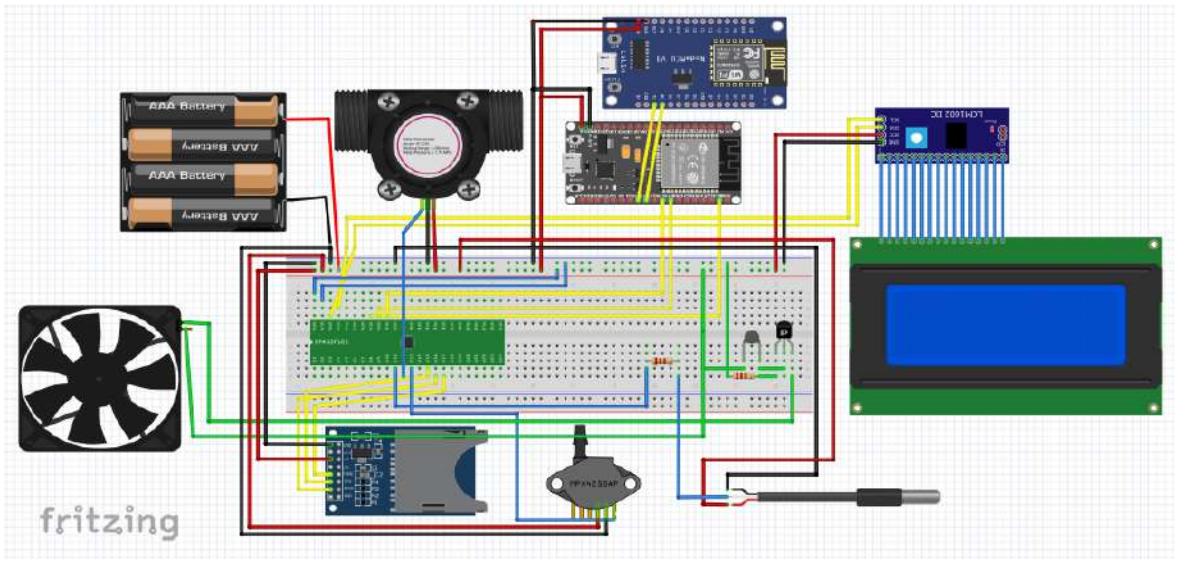


Figure 5.1 electronics design in the Fritzing software

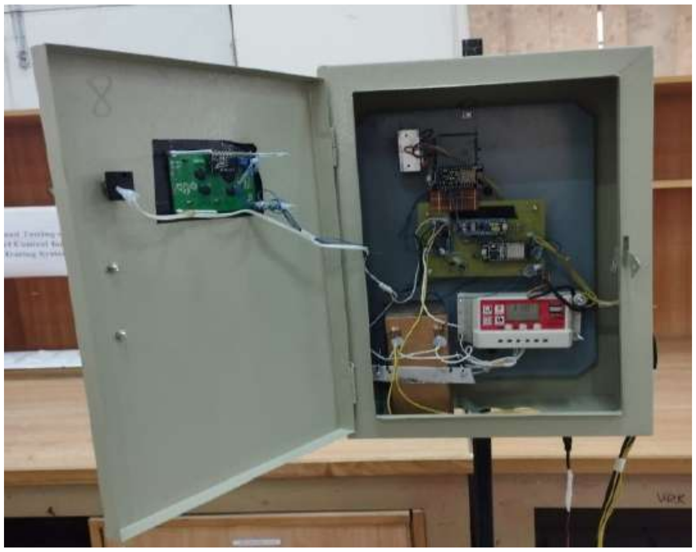


Figure 5.2 Inside View of Monitoring Box with Electronics

5.2 Implementation

At this point of the project, all the findings that has been gathered for the problem statement and analysis are carried out and used to generate codes that would control the working process of the system which is fully implemented. The first step was to design the circuit using the Fritzing software to ensure that the components are in right place when placed on the PCBoard. The design layout is simply a guide that aided in the arrangement of the electrical components on the project circuit board, it contains information such as the voltage and resistance rating for several electrical components (resistors, transistors etc.) used for the design of the circuit, it also provides information on how to fully connect the circuit by showing how every port and pin of the components should be connected to one another on the project circuit board. The next stage in the implementation is programming the microcontroller.

5.2.1 Programming the STM32F103C8 Microcontroller

After the code is fully developed and tested with flow code and the knowledge of the design layout is fully understood, the next step is uploading the code to the STM32F103C8 microcontroller. The STM32 cube IDE code is compiled to a. Hex format, the code is converted because the microcontroller can only interpret and execute commands in a. Hex format. The .Hex file is uploaded into the STM32 with the use of a debugger.

5.2.2 Constructing the Circuit

- This first step is mounting the STM32F103C8 microcontroller on the project circuit board and solder it against the board carefully.
- When mounting components parts on the project circuit board, the circuit layout diagram is used as guide because it contains all the necessary information about the components and how they are connected to one another.
- The resistors and diodes are mounted vertically on the board because of space constraints.
- The LCD is installed on the project circuit board and calibrated with a variable resistor to the appropriate voltage required. The LCD is connected to the project circuit board with cables connected to the appropriate port with respect to the STM32F103C8 microcontroller and use I2C protocol.
- The next step is mounting the transistors on the project circuit board connected to the microcontroller and then solder against the board firmly and carefully.
- The temperature sensor is connected to the proper PA0 port on the STM32F103C8 microcontroller. And the Pressure sensor is connected to the proper PA7 port on the STM32F103C8 microcontroller. Lastly the Flow rate sensor is connected to the proper PA14 port on the STM32F103C8 microcontroller.
- The SD card module is mounted on the board and cables connected to the appropriate port with respect to the STM32F103C8 microcontroller and uses SPI protocol.
- A buck converter is mounted on the project circuit board to control the flow of electric current.
- Every other component is mounted appropriately on the project circuit board and properly connected to their respective terminals

5.2.3 Mechanical Design

In the mechanical design, the model of the project was constructed firstly in the software and then build the hardware, which include a monitoring box, uses a electrical distribution panel which have all the electronics, the box prevent the electronics from the environmental damages such as harsh environmental temperature and any kind of weather storm. A stand to hold the monitoring box at any place. A fan is mounted on one side of the box, a pettier on the inside back wall of the box.

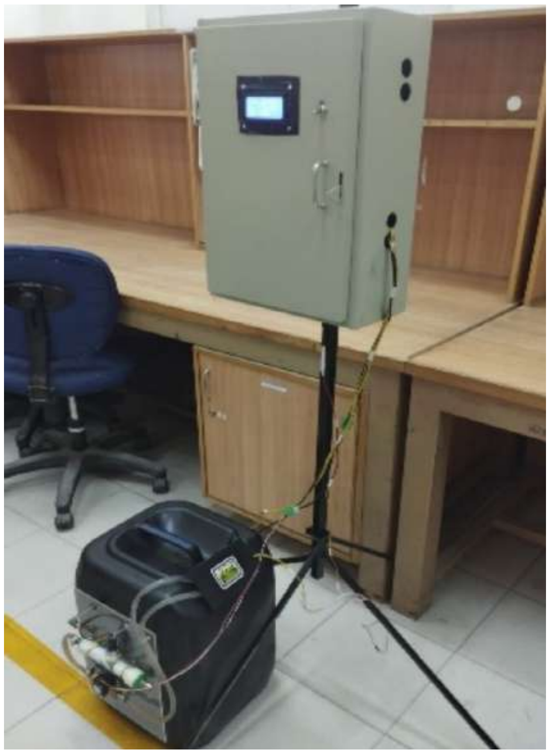


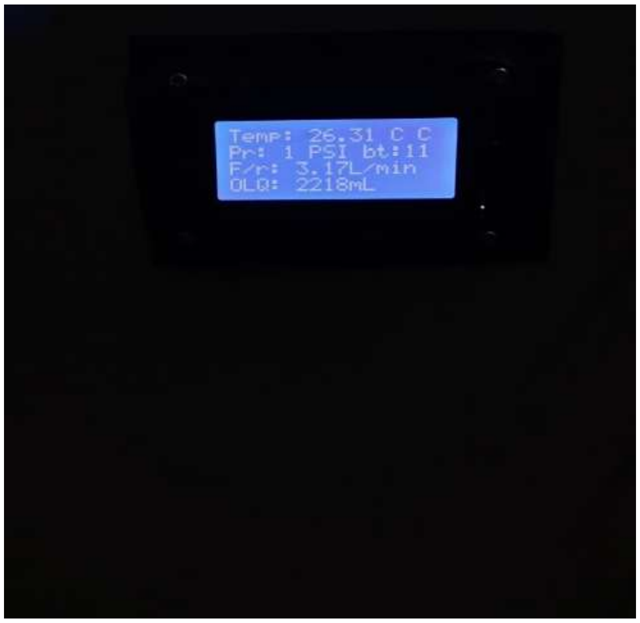
Figure 5.3 Mechanical Design of the system

5.3 Results

This study presents a novel prototype of a Remote oil well monitoring system that was developed to aid the oil and gas industry in its efforts to keep tabs on and better manage its oil and gas wells in order to boost output by means of, among other things, the remote collection of data in real time, the lessening of facility downtime, and the establishment of connections between machines that allow for greater automation and control. This is preferable to the laborious and time-consuming process of manually collecting data, which would be necessary due to the pumping unit's constant operation.

In order to achieve the goals, set out at the start of this project, an Internet of Things based oil and gas well monitoring system for an oil and gas well has been designed and built, using an STM32 microcontroller, an ESP module to send the measured parameter data to the cloud, and Fire base for user interaction.

In conclusion, we designed extremely effective and safe tools for remote monitoring and data collecting at an oil and gas well that are also simple to use.



```
Temp: 26.31 C C
Pr: 1 PSI bt:11
F/r: 3.17L/min
OLQ: 2218mL
```

Figure 5.4 Parameters Values on the LCD Display on Monitoring Box

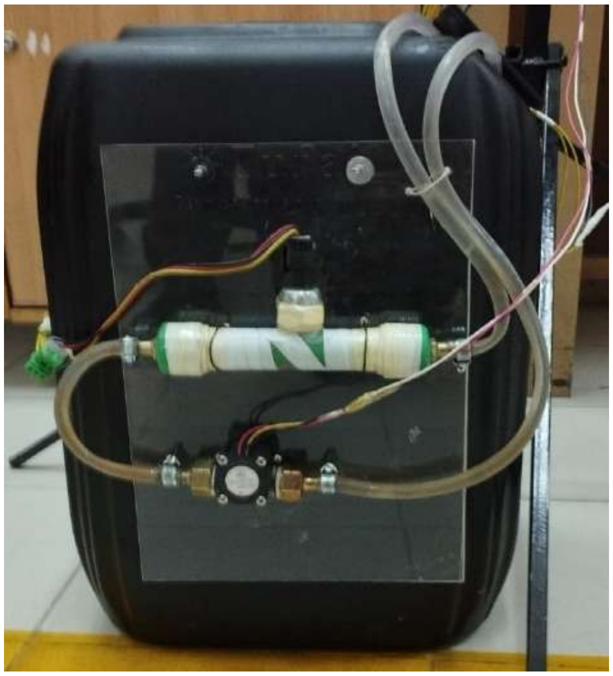


Figure 5.5 Tank with Sensors Mounted on it

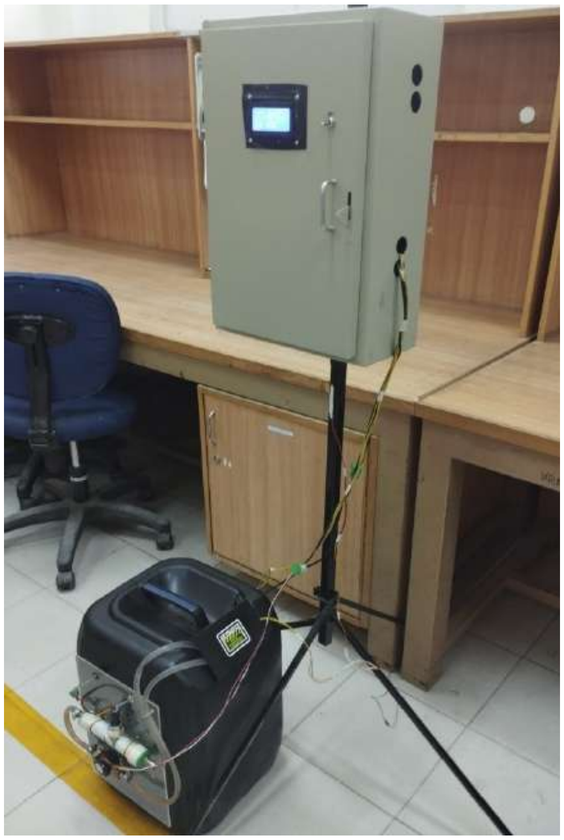
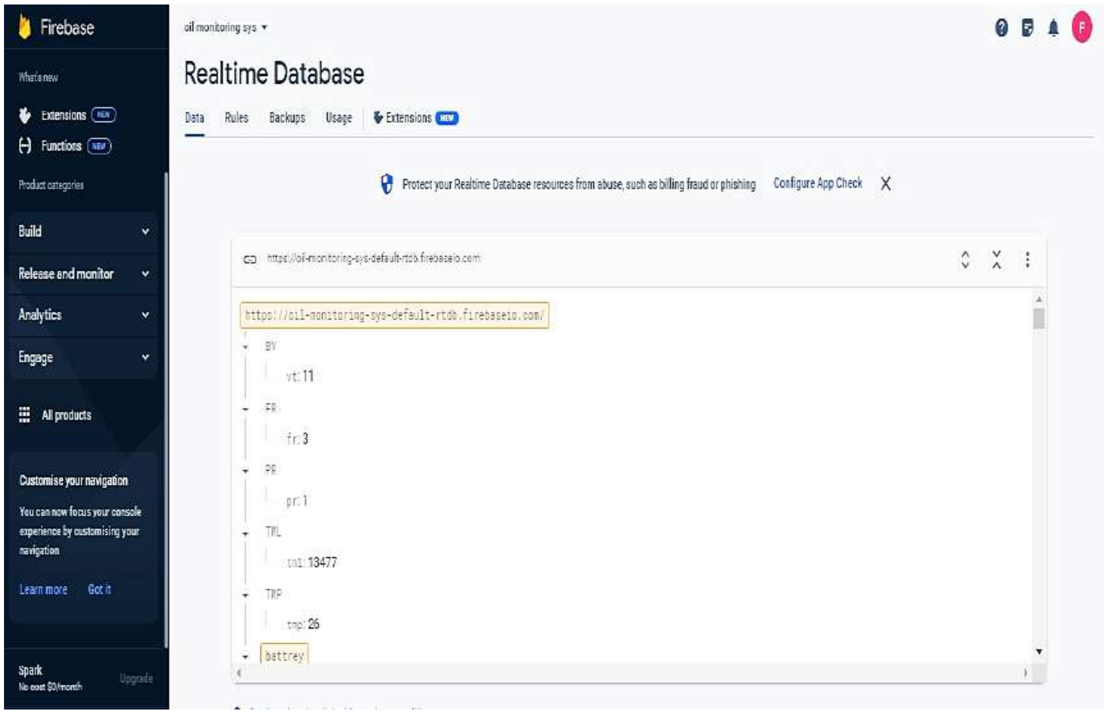
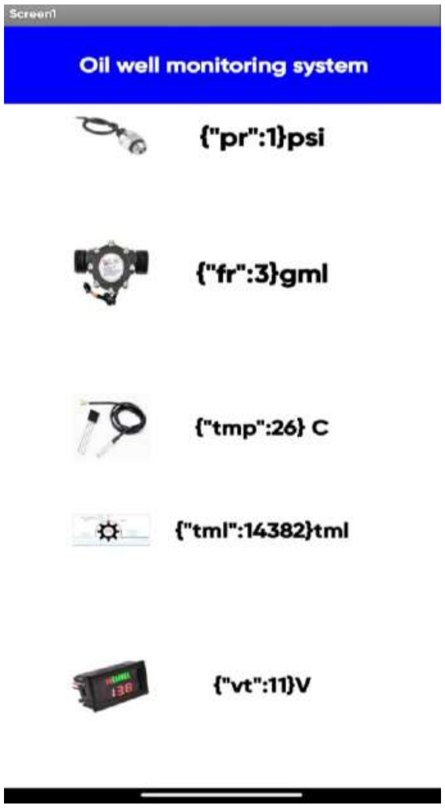


Figure 5.6 Complete Monitoring System in Running/Working Condition



```
https://oil-monitoring-sys-default-rtdb.firebaseio.com
https://oil-monitoring-sys-default-rtdb.firebaseio.com/
- BV
    vt: 11
- FR
    fr: 3
- PR
    pr: 1
- TML
    tml: 13477
- TMP
    tmp: 26
- battrey
```

Figure 5.7 Data Visualization on the Web (Firebase)


Screen1

Oil well monitoring system

{"pr":1}psi

{"fr":3}gml

{"tmp":26} C

{"tml":14382}tml

{"vt":11}V

Figure 5.8 Data Visualization on the Phone (Application)

Chapter 6 - Conclusion and Future Recommendations

6.1 Conclusion

To run efficiently, the oil and gas industry requires specialized knowledge, innovative system and constant oversight. Currently, IoT is our best bet for fixing this issue. With the advent of IoT technology, a system has been developed to keep an eye on oil and gas wells from afar in order to spot potential dangers and shut them down before they happen. There is a way to ensure the safety of employees and the well site by lowering the risk of explosions and fires and providing early alerts when they occur.

The Project gives us the following benefits:

- Increase workplace safety and ensure environmental protection.
- Increase operational efficiency by providing Realtime data.
- Equipment uptime and durability, improvement with the help of remote monitoring and data gathering.
- Reduced expenditures as a result of more remote automated monitoring.

6.2 Future Recommendations

For further advancement and improvement on this project, the following measures can be carried out:

- Integration of technologies like Machine Learning and Artificial Intelligence to expand the system ability to analyze the well's performance and predictability.
- Integration with data analytics platforms for better data visualization and analysis and optimization and management.
- Integration with satellite imagery to identify any type of leakages and fire on the well site through satellite-based images.

References

[1] H. Ramzey, M. Badawy, M. Elhosseini, and A. A. Elbaset, "I2OT-EC: A Framework for Smart Real-Time Monitoring and Controlling Crude Oil Production Exploiting IIOT and Edge Computing," *Energies (Basel)*, vol. 16, no. 4, Feb. 2023, doi: 10.3390/en16042023.

[2] W. Jinhuan, X. Zehui, and L. Chunxiang, "Research on the Intelligent Monitoring System of Oil Wells Based on Internet of Things," 2019.

[3] J. Davis, T. Edgar, J. Porter, J. Bernaden, and M. Sarli, "Smart manufacturing, manufacturing intelligence and demand-dynamic performance," *Comput Chem Eng*, vol. 47, pp. 145–156, Dec. 2012, doi: 10.1016/j.compchemeng.2012.06.037.

[4] Lembaga Ilmu Pengetahuan Indonesia, Institute of Electrical and Electronics Engineers. Indonesia Section, and Institute of Electrical and Electronics Engineers, *2017 ICRAMET proceeding : 2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET) : Jakarta, Indonesia, October 23-24, 2017*.

[5] L. Song, J. Zhang, C. Hao, H. Qi, B. Tian, and W. Li, "The Development and Application of Remote Monitoring System for Oil Well Working Condition with Mobile Phone Module," in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Jul. 2020. doi: 10.1088/1742-6596/1575/1/012050.

[6] N. Jan, A. Nasir, M. S. Alhilal, S. U. Khan, D. Pamucar, and A. Alothaim, "Investigation of cyber-security and cyber-crimes in oil and gas sectors using the innovative structures of complex intuitionistic fuzzy relations," *Entropy*, vol. 23, no. 9, Sep. 2021, doi: 10.3390/e23091112.

[7] M. Thibaud, H. Chi, W. Zhou, and S. Piramuthu, "Internet of Things (IoT) in high-risk Environment, Health and Safety (EHS) industries: A comprehensive review," *Decis Support Syst*, vol. 108, pp. 79–95, Apr. 2018, doi: 10.1016/j.dss.2018.02.005.

[8] F. Molaei, E. Rahimi, H. Siavoshi, S. Ghaychi Afrouz, and V. Tenorio, "A Comprehensive Review on Internet of Things (IoT) and its Implications in the Mining

Industry A Com-prehensive Review on Internet of Things (IoT) and its Implications in the Mining A Comprehensive Review on Internet of Things (IoT) and its Implications in the Mining Industry," *Industry. American Journal of Engineering and Applied Sciences*, vol. 2020, no. 3, pp. 499–515, 2020, doi: 10.3844/ajeassp.2020.499.515ï.

[9] B. K. Tripathy and J. Anuradha, "Internet of Things(IoT): Technologies, Applications, Challenges and Solutions Database Anonymisation techniques View project Big Data Analytics View project," 2017. [Online]. Available: https://www.researchgate.net/publication/320445442

[10] "A. Operation Level in the Oil and Gas Industry EEE 4 th International Conference on Knowledge-Based Engineering and Innovation (KBEI) I," 2017.

[11] G. Vinobala, M. Piramu, D. K. Rubasoundar, and P. G. Scholar, "MONITORING OF INDUSTRIAL ELECTRICAL EQUIPMENT USING IOT," 2021. [Online]. Available: www.ijcrt.org

[12] M. Babar and M. Sohail Khan, "ScalEdge: A framework for scalable edge computing in Internet of things–based smart systems," *Int J Distrib Sens Netw*, vol. 17, no. 7, 2021, doi: 10.1177/15501477211035332.

[13] S. Priyadarshy, "31 IoT REVOLUTION IN OIL AND GAS INDUSTRY," 2017. [Online]. Available: www.wiley.com/go/Geng/iot_data_analytics_handbook/

[14] Prof. Sathish and Dr. S. Smys, "A Survey on Internet of Things (IoT) based Smart Systems," *Journal of ISMAC*, vol. 2, no. 4, pp. 181–189, Sep. 2020, doi: 10.36548/jismac.2020.4.001.

[15] S. Purohit, A. K. Jain, and S. Purohit, "Evaluation Of The Efficacy Of Iot Deployment On Petro-Retail Operations," 2021. [Online]. Available: https://www2.deloitte.com/us/en/pages/consulting/articles/iot-digital-oil-and-gas.html

[16] Global IT Research Institute, IEEE Communications Society, and Institute of Electrical and Electronics Engineers, *The 19th International Conference on Advanced Communications Technology : "Opening Era of Smart Society" : ICACT 2017 : Phoenix Park, Pyeongchang, Korea (South) : Feb. 19-22, 2017 : proceeding & journal.*

[17] B. Shakerighadi, A. Anvari-Moghaddam, J. C. Vasquez, and J. M. Guerrero, "Internet of things for modern energy systems: State-of-the-art, challenges, and open issues," *Energies (Basel)*, vol. 11, no. 5, 2018, doi: 10.3390/en11051252.

[18] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing Smart Factory of Industrie 4.0: An Outlook," *Int J Distrib Sens Netw*, vol. 2016, 2016, doi: 10.1155/2016/3159805.

[19] Y. Zuo and Z. Qi, "A Blockchain-Based IoT Framework for Oil Field Remote Monitoring and Control," *IEEE Access*, vol. 10, pp. 2497–2514, 2022, doi: 10.1109/ACCESS.2021.3139582.

[20] T. R. Wanasinghe, R. G. Gosine, L. A. James, G. K. I. Mann, O. De Silva, and P. J. Warrian, "The Internet of Things in the Oil and Gas Industry: A Systematic Review," *IEEE Internet Things J*, vol. 7, no. 9, pp. 8654–8673, Sep. 2020, doi: 10.1109/JIOT.2020.2995617.

[21] Z. Wu, K. Qiu, and J. Zhang, "A smart microcontroller architecture for the internet of things," *Sensors (Switzerland)*, vol. 20, no. 7, Apr. 2020, doi: 10.3390/s20071821.

[22] O. Bhat, P. Gokhale, and S. Bhat, "Introduction to IOT," *International Advanced Research Journal in Science, Engineering and Technology ISO*, vol. 3297, no. 1, 2007, doi: 10.17148/IARJSET.2018.517.

[23] https://www.circuitsgallery.com/automatic-temperature-controlled-fan-circuit/

[24] https://www.broadcom.com/info/aiops/iot

[25] Muhammad Sohail Khan, Babar, and Mohammad. A framework for scalable edge computing in smart systems based on the Internet of Things is called ScalEdge. 15501477211035332, International Journal of Distributed Sensor Networks 17.7 (2021).

[26] Santanu Purohit. "Evaluation of IoT deployment's effectiveness on Petro-Retail Operations." TURCOMAT 12.4 (2021): 356-363 Turkish Journal of Computer and Mathematics Education.

Appendix

Code for the ESP32 Module

```
// include library to read and write from flash memory

#include <EEPROM.h>

// define the number of bytes you want to access

#define EEPROM_SIZE 1

#include <ArduinoJson.h>

int analogInput = 32;

float vout = 0.0;

int vin = 0.0;

float R1 = 10000.0; // resistance of R1 (100K) -see text!

float R2 = 1000.0; // resistance of R2 (10K) - see text!

int value = 0;

#include <WiFi.h>

#include <FirebaseESP32.h>

#define FIREBASE_HOST "oil-monitoring-sys-default-rtdb.firebaseio.com"

#define FIREBASE_AUTH "pliZZOuKDfX3EEBXOqqjBTRlmorzV2sAnPMPKyPY"

#define WIFI_SSID "nlink"

#define WIFI_PASSWORD "11223344"

FirebaseData firebaseData;

FirebaseJson jsonA;

int pr = 0;
```

```
FirebaseJson jsonB;

int fr = 0;

FirebaseJson jsonC;

int tmp = 0;

FirebaseJson jsonD;

int tml = 0;

FirebaseJson jsonE;

int bv = 0;

const float  OffSet = 0.254 ; // adjust this value

float avgV = 0.0;

float V = 0;

float P;

#include <LiquidCrystal_I2C.h>

int lcdColumns = 16;

int lcdRows = 4;

LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

/**************************************************/

#include <OneWire.h>

#include <DallasTemperature.h>

#define SENSOR_PIN  15

OneWire oneWire(SENSOR_PIN);

DallasTemperature DS18B20(&oneWire);

float tempC; // temperature in Celsius
float tempF; // temperature in Fahrenheit

/**************************************************/

#define SENSOR 27

long currentMillis = 0;

long previousMillis = 0;

int interval = 2000;

boolean ledState = LOW;

float calibrationFactor = 4.5;

volatile byte pulseCount;

byte pulse1Sec = 0;

float flowRate;

unsigned int flowMilliLitres;

unsigned long totalMilliLitres;

void IRAM_ATTR pulseCounter()

{

  pulseCount++;

}

void setup() {

  Serial.begin(9600); // initialize serial

 EEPROM.begin(EEPROM_SIZE);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("Connecting to Wi-Fi");

  while (WiFi.status() != WL_CONNECTED)
```

```
{
  Serial.print(".");
  delay(300);
}
Serial.println();
Serial.print("Connected with IP: ");
Serial.println(WiFi.localIP());
Serial.println();
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
Firebase.reconnectWiFi(true);
//Set database read timeout to 1 minute (max 15 minutes)
Firebase.setReadTimeout(firebaseData, 1000 * 60);
//tiny, small, medium, large and unlimited.
//Size and its write timeout e.g. tiny (1s), small (10s), medium (30s) and large (60s).
Firebase.setwriteSizeLimit(firebaseData, "tiny");
Serial.println("------------------------------------");
Serial.println("Connected...");
DS18B20.begin();    // initialize the DS18B20 sensor
lcd.init();
// turn on LCD backlight
lcd.backlight();
Serial.println("/** OIL pressure sensor demo **/");
pinMode(SENSOR, INPUT_PULLUP);
```

```
  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  previousMillis = 0;
  attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
totalMilliLitres = EEPROM.read(0);
}
int p;
// Function that gets current epoch time
unsigned long getTime() {
  time_t now;
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    //Serial.println("Failed to obtain time");
    return(0);
  }
  time(&now);
  return now;
}
void loop() {
  /*****************************************************************/
  DS18B20.requestTemperatures();      // send the command to get temperatures
```

```
tempC = DS18B20.getTempCByIndex(0);  // read temperature in °C
tempF = tempC * 9 / 5 + 32; // convert °C to °F
/*********************          Calculate          water          pressure
*******************************************/
// read the value at analog input
value = analogRead(analogInput);
vout = (value * 3.9) / 4096.0; // see text
vin = vout / (R2 / (R1 + R2));
if (vin < 0.09) {
  vin = 0.0; //statement to quash undesired reading !
}
/*******************************************************/
currentMillis = millis();
if (currentMillis - previousMillis > interval) {
  pulse1Sec = pulseCount;
  pulseCount = 0;
  flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor;
  previousMillis = millis();
  int pp = flowRate;
  // Divide the flow rate in litres/minute by 60 to determine how many litres have
  // passed through the sensor in this 1 second interval, then multiply by 1000 to
  // convert to millilitres.
  flowMilliLitres = (flowRate / 60) * 1000;
  // Add the millilitres passed in this second to the cumulative total
  totalMilliLitres += flowMilliLitres;
  EEPROM.write(0, totalMilliLitres);
  EEPROM.commit();delay(10);
  if (pp > 0)
  {
p = map(pp, 0, 21, 0, 8);
  }
  if (pp == 0)
  {
  p = 0;
  }
  // Print the flow rate for this second in litres / minute
 //  Serial.print("Flow rate: ");
 //  Serial.print(int(flowRate));  // Print the integer part of the variable
 //  Serial.print("L/min");
 //  Serial.print("\t");       // Print tab space
  // Print the cumulative total of litres flowed since starting
 //  Serial.print("Output Liquid Quantity: ");
 //  Serial.print(totalMilliLitres);
 //  Serial.print("mL / ");
 //  Serial.print(totalMilliLitres / 1000);
```

```
  // Serial.println("L");  }

// Serial.print(" Pressure:");

 //Serial.print(p);

// Serial.print(" PSI");

// Serial.println("//********************************//");

// Serial.print("Temperature: ");

// Serial.print(tempC);    // print the temperature in °C

 //Serial.print("°C");

 //Serial.print("  ~  ");  // separator between °C and °F

// Serial.print(tempF);    // print the temperature in °F

//  Serial.print("°F");

// Serial.println("//********************************//");

 lcd.setCursor(10, 1);

 lcd.print("bt:");

 lcd.print(vin);

 lcd.print(" ");

 lcd.setCursor(0, 0);

 lcd.print("Temp: ");

 lcd.print(tempC);

 lcd.print(" ");

 lcd.print("C");


 lcd.setCursor(0, 1);
lcd.print("Pr: ");
lcd.print(p);
lcd.print(" PSI");
lcd.print(" ");
lcd.setCursor(-4, 2);
lcd.print("F/r: ");
lcd.print(flowRate);
lcd.print("L/min");
lcd.print(" ");
lcd.setCursor(-4, 3);
lcd.print("OLQ: ");//Output Liquid Quantity:
lcd.print(totalMilliLitres);
lcd.print("mL");
lcd.print(" ");
/*Firebase.setFloat(firebaseData, "/TP ", tempC);
  Firebase.setFloat(firebaseData, "/FR ", flowRate);
  Firebase.setFloat(firebaseData, "/ml ", totalMilliLitres);
  Firebase.setFloat(firebaseData, "/VT ",vin );
  Firebase.setFloat(firebaseData, "/PR ", p);*/
pr = p;
fr = flowRate;
tml = totalMilliLitres;
tmp = tempC;
```

```
  bv = vin;
  jsonA.set("/pr", pr);
  Firebase.updateNode(firebaseData, "/PR", jsonA);
  jsonB.set("/fr", fr);
  Firebase.updateNode(firebaseData, "/FR", jsonB);
  jsonC.set("/tmp", tmp);
  Firebase.updateNode(firebaseData, "/TMP", jsonC);
  jsonD.set("/tml", tml);
  Firebase.updateNode(firebaseData, "/TML", jsonD);
  jsonE.set("/vt", bv);
  Firebase.updateNode(firebaseData, "/BV", jsonE);
/***********************************************************/
  StaticJsonBuffer<1000> jsonBuffer;
  JsonObject& root = jsonBuffer.createObject();
  root["data1"] =  pr ;// in
  root["data2"] = fr ;  // out
  root["data3"] = tmp ;// fire
  root["data4"] = bv  ;// gas
  root.printTo(Serial);
  delay(100);
}
```

Code for the stm32f103c8 microcontroller:

```
#include "stm32f1xx.h"

#include "stdio.h"

#include "lcd.h"

#include "ds18b20.h"

#include "hk100c.h"

#include "flowrate.h"

#include "esp32.h"

#include "esp8266.h"

// Define your pin connections

#define DS18B20_PIN       GPIO_PIN_0

#define DS18B20_PORT      GPIOA

#define HK100C_PIN        GPIO_PIN_1

#define HK100C_PORT       GPIOA

#define FLOWRATE_PIN      GPIO_PIN_2

#define FLOWRATE_PORT     GPIOA

#define THERMISTOR_PIN    GPIO_PIN_3

#define THERMISTOR_PORT   GPIOA

#define LCD_RS_PIN        GPIO_PIN_4

#define LCD_RS_PORT       GPIOA

#define LCD_RW_PIN        GPIO_PIN_5

#define LCD_RW_PORT       GPIOA

#define LCD_EN_PIN        GPIO_PIN_6
```

```
#define LCD_EN_PORT     GPIOA
#define LCD_D4_PIN      GPIO_PIN_7
#define LCD_D4_PORT     GPIOA
#define LCD_D5_PIN      GPIO_PIN_8
#define LCD_D5_PORT     GPIOA
#define LCD_D6_PIN      GPIO_PIN_9
#define LCD_D6_PORT     GPIOA
#define LCD_D7_PIN      GPIO_PIN_10
#define LCD_D7_PORT     GPIOA
// Define your ESP32 and ESP8266 connections
#define ESP32_UART      USART1
#define ESP8266_UART    USART2
// Function prototypes
void configureGPIO();
void configureLCD();
void readSensors(float* temperature, float* pressure, float* flowRate, float* thermistor);
void sendDataToESP32(float temperature, float pressure, float flowRate, float thermistor);
void sendDataToESP8266(float temperature, float pressure, float flowRate, float thermistor);
int main(void) {
   float temperature, pressure, flowRate, thermistor;
   // Configure GPIO pins
   configureGPIO();
   // Initialize LCD
```

```
 configureLCD();
 LCD_Init();
 LCD_Clear();
 // Initialize sensors
 DS18B20_Init(DS18B20_PIN, DS18B20_PORT);
 HK100C_Init(HK100C_PIN, HK100C_PORT);
 FlowRate_Init(FLOWRATE_PIN, FLOWRATE_PORT);
 // Initialize ESP32 and ESP8266
 ESP32_Init(ESP32_UART);
 ESP8266_Init(ESP8266_UART);
while (1)
{   // Read temperature from DS18B20 sensor
 temperature = DS18B20_ReadTemperature(DS18B20_GPIO_PORT, DS18B20_PIN);
 // Convert temperature to string
 sprintf(temperature_string, "%.2f", temperature);
 // Display temperature on LCD
 LCD_Clear();
 LCD_SetCursor(0, 0);
 LCD_Print("Temperature:");
 LCD_SetCursor(0, 1);
 LCD_Print(temperature_string);
 // Send temperature data to ESP32 and ESP8266 modules
 char uart_data[20];
```

```
  sprintf(uart_data, "TEMP:%s\r\n", temperature_string);
  HAL_UART_Transmit(&huart1,          (uint8_t*)uart_data,          strlen(uart_data),
HAL_MAX_DELAY);
 // Read pressure from HK100C sensor
  pressure = ReadPressure();
  // Convert pressure to string
  sprintf(pressure_string, "%.2f", pressure);
  // Display pressure on LCD
  LCD_Clear();
  LCD_SetCursor(0, 0);
  LCD_Print("Pressure:");
  LCD_SetCursor(0, 1);
  LCD_Print(pressure_string);
  // Send pressure data to ESP32 and ESP8266 modules
  char uart_data[20];
  sprintf(uart_data, "PRESS:%s\r\n", pressure_string);
  TransmitData(uart_data);
  // Delay before the next reading
  HAL_Delay(1000);
 // Convert flow rate to string
  sprintf(flowRateString, "%.2f", flowRate);
  // Display flow rate on LCD
  LCD_Clear();
```

```
    LCD_SetCursor(0, 0);
    LCD_Print("Flow Rate: ");
    LCD_SetCursor(0, 1);
    LCD_Print(flowRateString);
    // Send flow rate data to ESP32 and ESP8266 modules
    char uartData[20];
    sprintf(uartData, "FLOW:%s\r\n", flowRateString);
    UART_SendData(uartData);
    // Delay for a period
    HAL_Delay(1000);  } }
void SystemClock_Config(void)
{   RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  __HAL_RCC_PWR_CLK_ENABLE();
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
Error_Handler();   }
```

Code for the ESP82 Module

```
#include <ArduinoJson.h>
int data1;
int data2;
int data3;
int data4;
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
// Set these to run example.
#define FIREBASE_HOST "oil-monitoring-sys-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "pliZZOuKDfX3EEBXOqqjBTRlmorzV2sAnPMPKyPY"
#define WIFI_SSID "nlink"
#define WIFI_PASSWORD "11223344"
void setup() {   // put your setup code here, to run once:
  Serial.begin(9600);
  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);   }
  Serial.println();
  Serial.print("connected: ");
```

```
 Serial.println(WiFi.localIP());
 Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);  }
void loop() {   // put your main code here, to run repeatedly:
 StaticJsonBuffer<1000> jsonBuffer;
 JsonObject& root = jsonBuffer.parseObject(Serial);
 if (root == JsonObject::invalid())
  return;
 Serial.println("JSON received and parsed");
 root.prettyPrintTo(Serial);
 Serial.println("--------------------xxxxx--------------------");
 delay(2);
 data1 = root["data1"];
 data2 = root["data2"];
 data3 = root["data3"];
 data4 = root["data4"];
 Firebase.pushInt("pressure", data1);
 Firebase.pushInt("flow Rate", data2);
 Firebase.pushInt("temp", data3);
 Firebase.pushInt("battrey", data4);
 if (Firebase.failed()) {   Serial.print("pushing /logs failed:");
  Serial.println(Firebase.error());
  return;   }  }
```