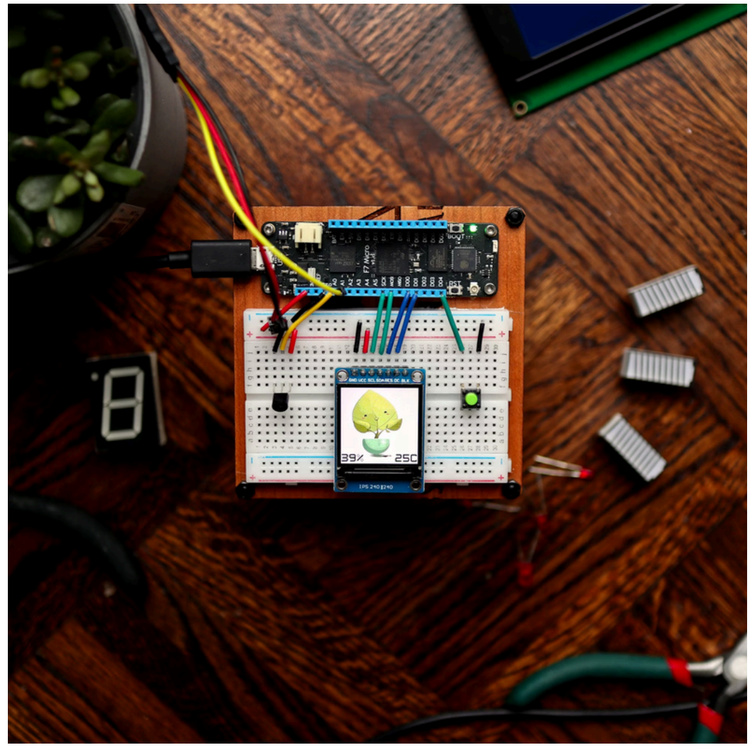
AquaWatch IoT



Presented by :

Pratik Anand (21BCT0014)
Advaita Sharma(21BCT0094)
Sparsh Teotia(21BCI0228)

Project Overview

This project aims to develop :

- Web-based mobile application for real-time temperature monitoring
- Designed for 100-hour boiling tests at remote locations
- Automates data collection and analysis process
- Enhances efficiency and reliability of insulator testing

The application will enable users to access and visualize temperature data in a convenient and accessible manner.

Problem Statement

The manual monitoring of the 100-hour boiling test requires continuous human presence, which is time-consuming and prone to errors. A web-based mobile application that displays real-time temperature data can automate the monitoring process and improve efficiency.

Literature Survey

- **Temperature Measurement and Sensors**: Research on various sensor types, calibration methods, and optimal placement.

- **Wireless Communication**: Studies on communication protocols and network reliability.
- Data Processing and Analysis: Research on data storage, visualization, and analysis techniques.

- **Mobile Application Development**: Studies on user interface design and development frameworks.

- **Real-Time Systems**: Research on real-time operating systems and synchronization mechanisms.

- **Industrial Automation**: Studies on automation technologies and integration with laboratory equipment.

Conclusion & Identification of Gaps

- The production of high-quality insulators is a critical component of various electrical systems, ensuring the safe and efficient transmission of power.

- One essential aspect of insulator quality assurance is the rigorous testing of their performance under extreme conditions, such as prolonged exposure to high temperatures and corrosive environments

Current method: Manual observation and recording.

Challenges:

- Time-consuming
- Labor-intensive
- Prone to human error

Motivation:

- Improve efficiency and accuracy
- Enable remote access to data
- Support research and development efforts

Project Objectives

Objective 1

Implement accurate real-time temperature monitoring (±1°C accuracy)

Objective 2

Develop a web-based mobile app for remote access and visualization

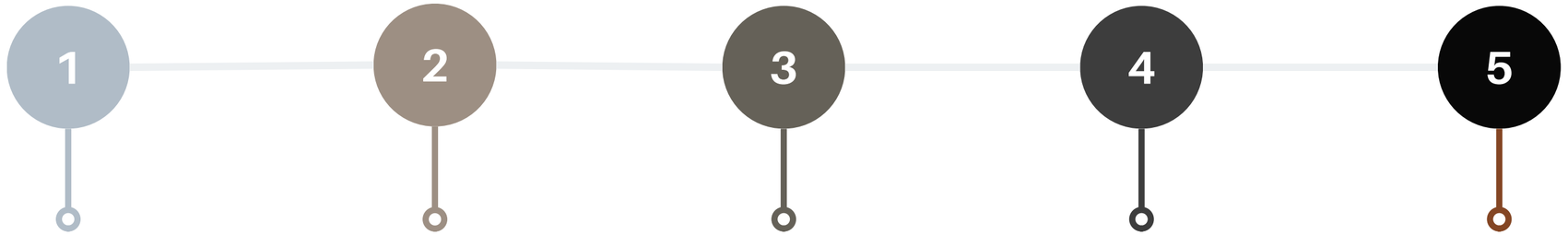
Objective 3

Reduce manual data recording errors by 90%

Objective 4

Increase overall testing efficiency by 20%

PROJECT TIMELINE



July 21 - Aug 04

Phase 1
Project Planning and Initialization

Aug 05 - Aug 18

Phase 2
Research and Requirements Gathering

Aug 19 - Sep 08

Phase 3
Design Phase

Sep 09 - Oct 13

Phase 4
Development Phase

Oct 14 - Nov 06

Phase 5
Testing and Finalization

Requirement Analysis

1. Temperature Monitoring :
   - The system must continuously measure the boiling water temperature using thermocouples.
   - It must record temperature data at regular intervals (e.g., every minute or hour) for the entire duration of the test.
2. Data Logging
   - The Raspberry Pi must process and log temperature data from the thermocouples.
   - NodeMCU should handle data transmission from the Raspberry Pi to the Thingspeak cloud.
3. Data Transmission
   - The system should transmit recorded data to the Thingspeak cloud in real-time via Wi-Fi, using NodeMCU.

Requirement Analysis

4. Real-time Graphical Representation
   - Thinkspeak cloud should display the recorded temperature data in graphical form, updated in real-time.
   - The graph must show the current temperature, historical data, and elapsed time during the test.
5. Mobile App Access
   - The system should provide remote access to the data via a mobile app, allowing users to monitor the temperature and elapsed time from anywhere.
6. Data Storage and Retrieval
   - After the test completion, users must be able to access and verify historical temperature data from Thingspeak.
7. Alerts and Notifications (Optional for prototype)
   - The system may trigger alerts or notifications via the mobile app if the temperature exceeds or drops below predefined thresholds.

System Architecture

Test Environment

- Test specimens in 5% NaCl solution
- Heating element
- Temperature sensor

Local Control Unit

- Microcontroller (Raspberry Pi)
- Wi-Fi module
- Local storage

Cloud Infrastructure

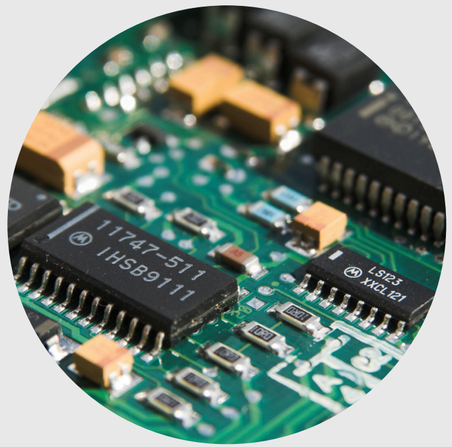
- IoT Gateway
- Cloud Database
- Web Server
- Authentication Server

User Interface

- Web Dashboard
- Mobile Application (Optional)

Implementation

Hardware



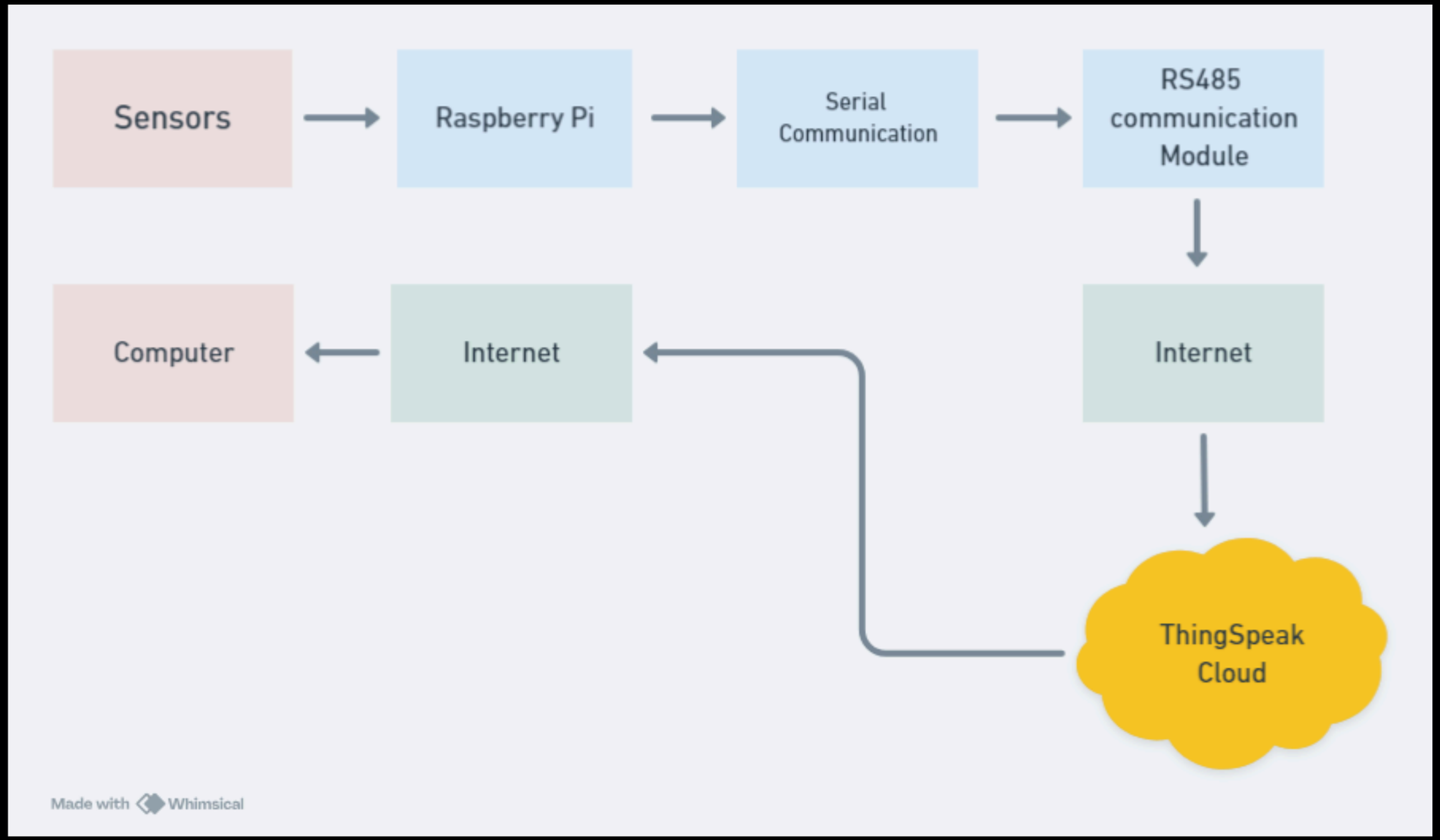
- Processor: Raspberry Pi 4 Model B (Quad-core Cortex-A72 64-bit SoC @ 1.5GHz)
- Memory: 2GB LPDDR4 SDRAM
- Storage: 16GB or 32GB microSD card
- Temperature Sensors: Thermocouples for high-temperature measurements / DS18B20 Digital Thermometer
- Networking: Built-in Wi-Fi module in Raspberry Pi / Node MCU Module

Implementation

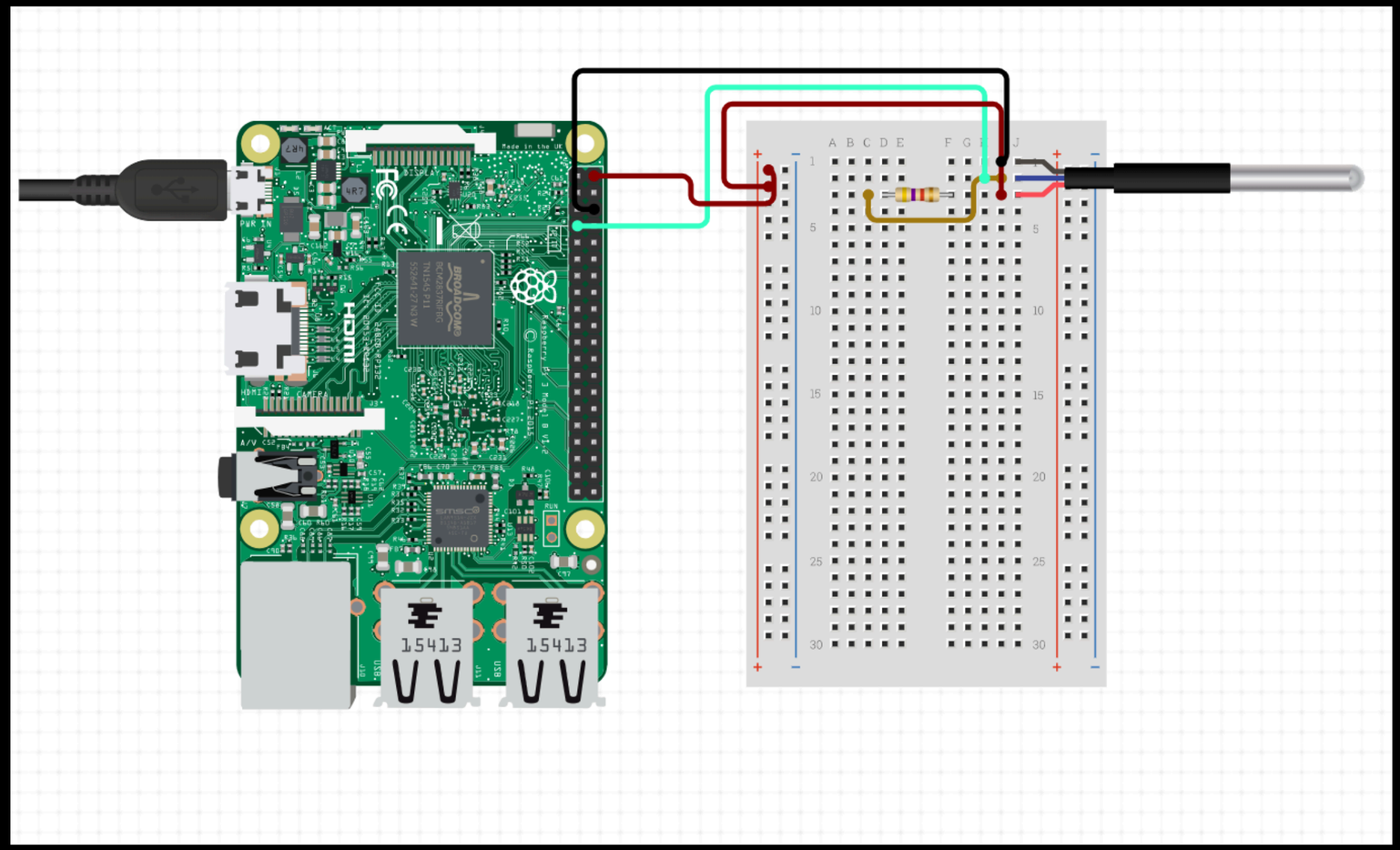
Software

- Operating System: Raspberry Pi OS (formerly Raspbian)
- Programming Languages:
  - Python: For sensor data collection and cloud communication
  - JavaScript/HTML/CSS: For web interface development
- Development Environment:
  - Visual Studio Code: IDE for Python script development
  - ThingSpeak API: For data integration and visualization
- Libraries and Frameworks:
  - PySerial: For interfacing with thermocouple sensors
  - Requests/HTTP: For sending data to ThingSpeak cloud
  - Flask or Django: For custom web interface (if needed)
- Cloud Infrastructure: ThingSpeak Cloud: For data storage, visualization, and API services
- Security: SSL/TLS for secure communication, API Key Authentication for data transmission
- User Interfaces: Mobile App: Cross-platform app for real-time monitoring and alerts
- Web Dashboard: Comprehensive interface for detailed data analysis and system management

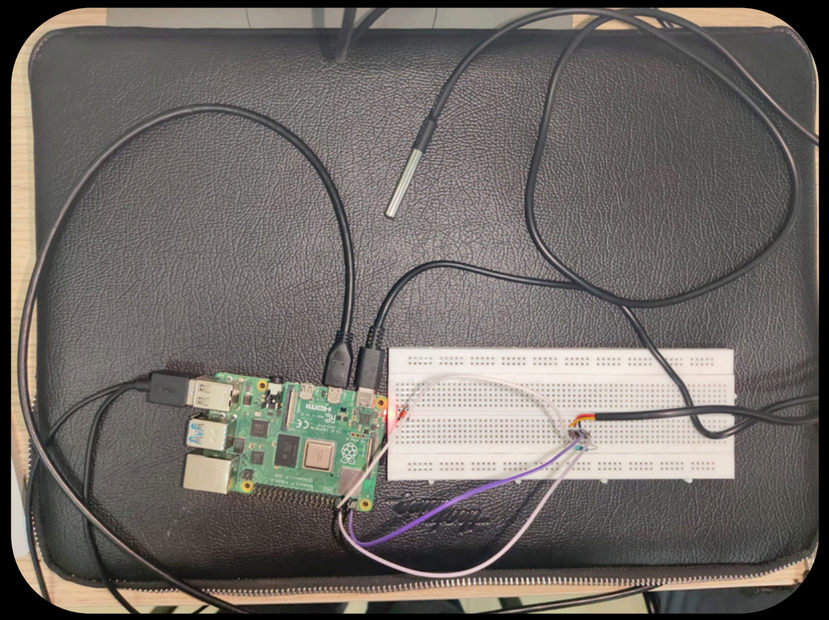
Block Diagram

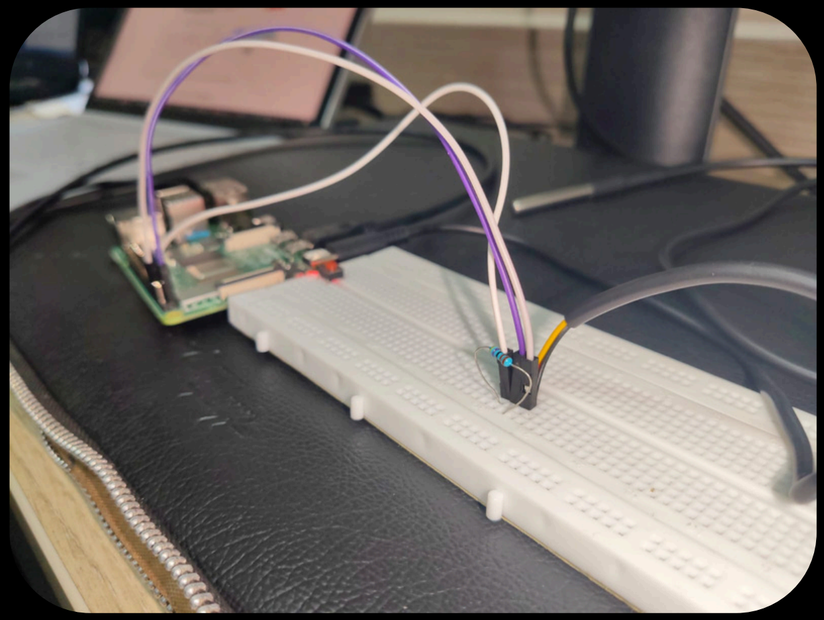


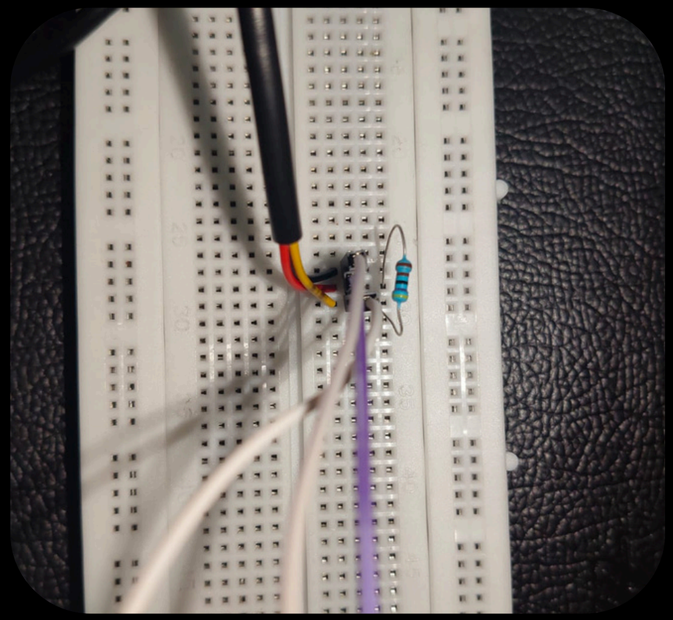
Circuit Diagram



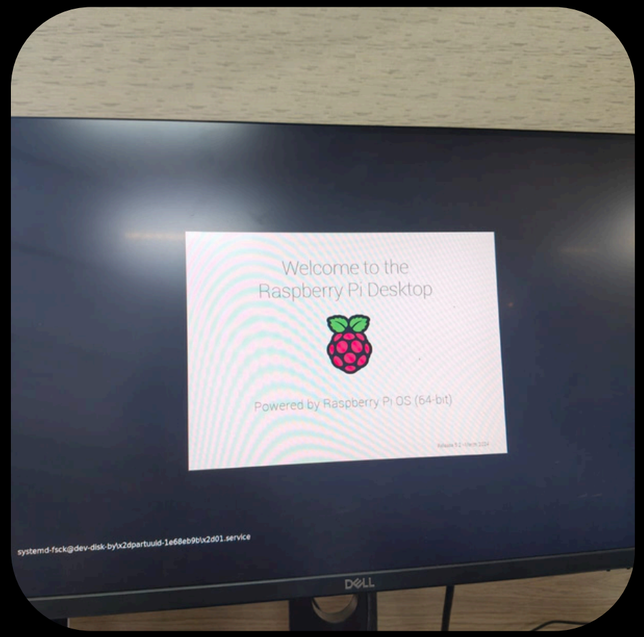
Implementation of Circuit Diagram

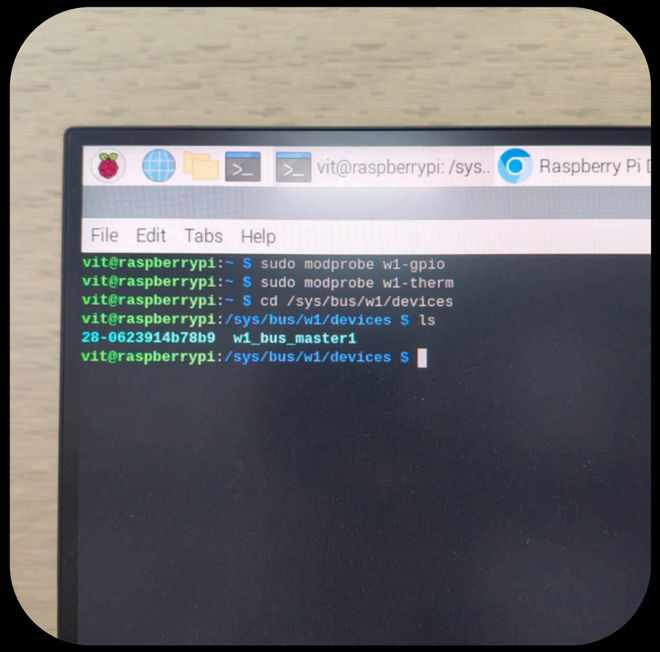


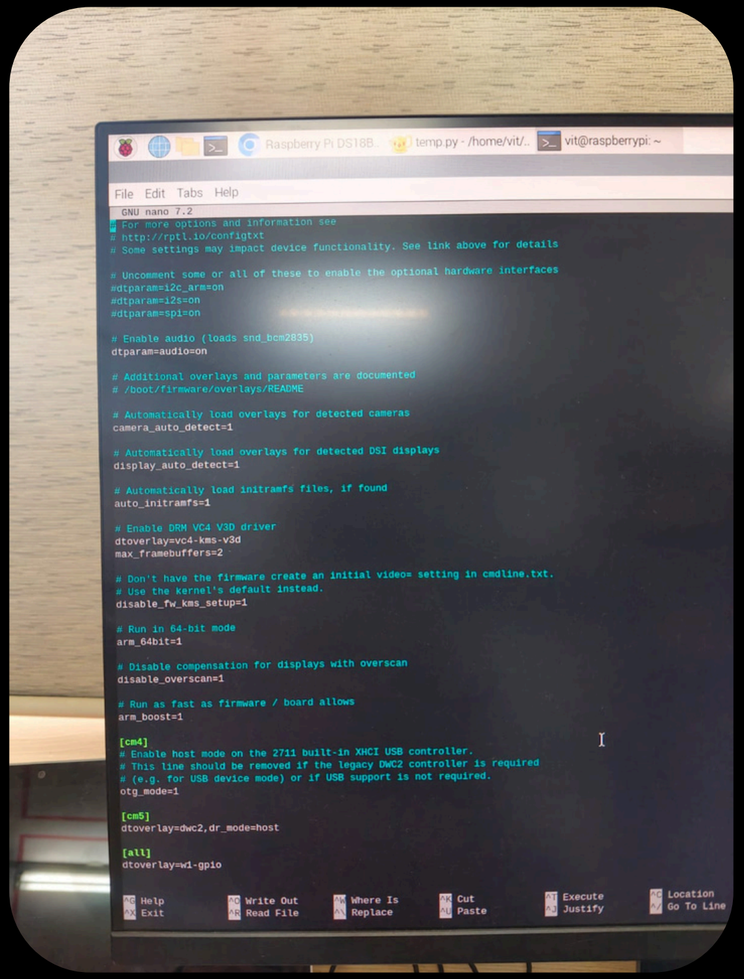




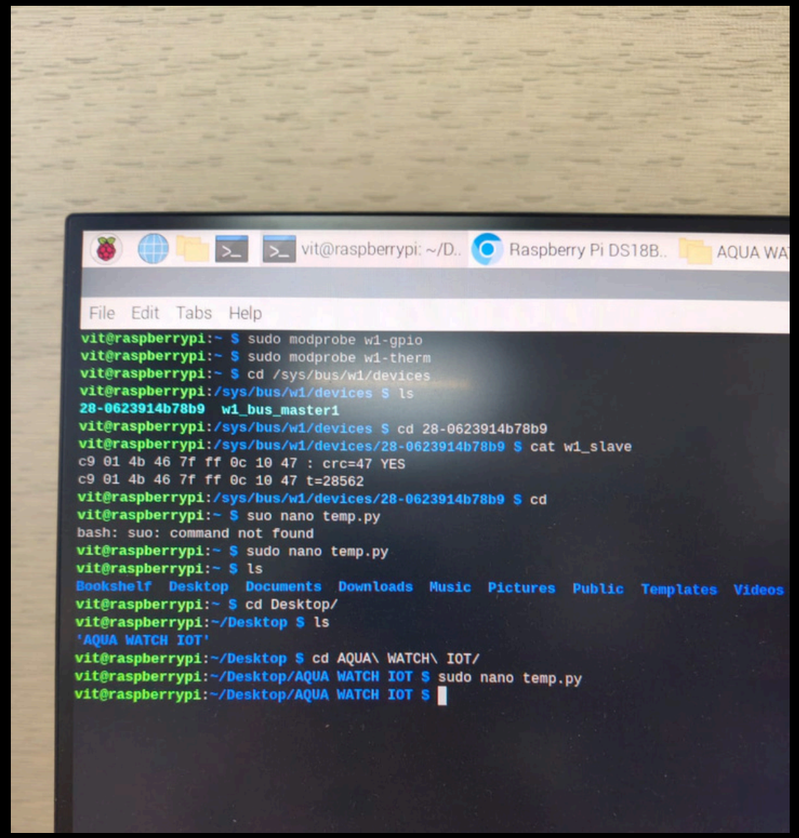
Raspberry pi Interfaced with DS18B20 Temperature Sensor

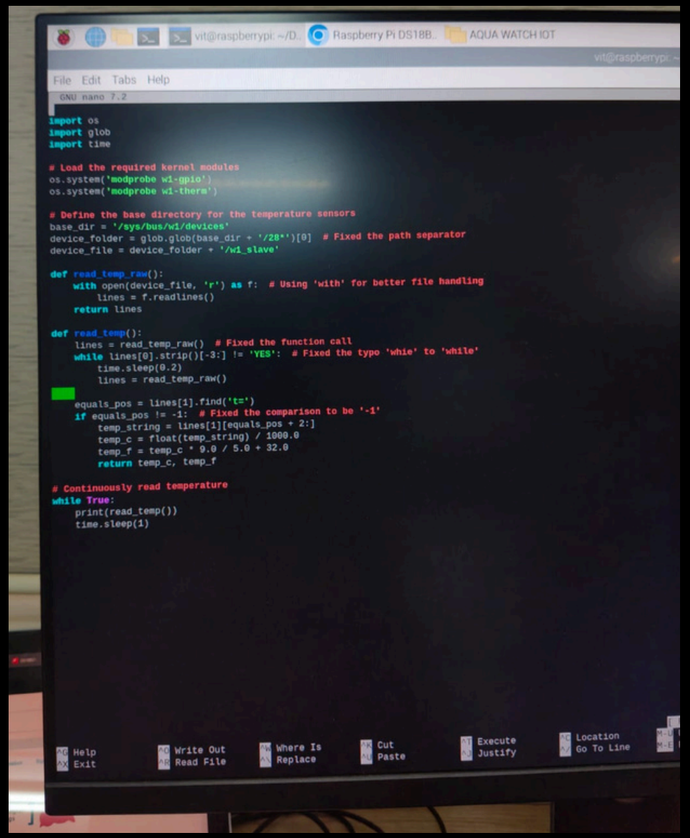






Enable the One-Wire interface on the Raspberry OS





Displaying the raw temperature reading output by the sensor

Code for output of temperature readings in Fahrenheit and Celsius to SSH terminal:

```
import os
import glob
import time

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines
```

```
def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f

while True:
        print(read_temp())
        time.sleep(1)
```

Expected Outcomes

- Improved accuracy in temperature monitoring
- Reduced workload for researchers and technicians
- Enhanced data analysis capabilities
- Increased flexibility through remote access
- Potential for scalability and future enhancements

Future Enhancements

1

Remote Control

Add capability to start/stop heating element remotely

2

Multiple Parameters

Include sensors for pH, conductivity, etc.

Thank you!

Contributions:

Pratik Anand (21BCT0014) : Hardware + Project Timeline
Advaita Sharma(21BCT0094) : Software + Research
Sparsh Teotia(21BCI0228) : Design Approaches + Software