## Introduction of project and report

The Climate Monitoring System tracks climate and humidity conditions and alerts users if an anomalous temperature is recorded. The system uses MQTT protocol to publish/subscribe information over a TCP/IP connection to the internet. The device consists of the ESP 32 hardware and the web based Graphical User Interface based on Node-Red flows.

**Device Functionalities:**

* Alarm Mode
* Notification System
* Audio Notification System
* Task Manager for CPU Usage
* Sleep Mode for reduced power consumption
* Measuring and displaying humidity, temperature, and heat index levels

The device can be operated from the hardware component (ESP 32 device) as well as the software component (user interface on node red).

**Alarm Mode:**

The regular mode blinks the LED (the red one) when anomalous climate conditions are detected. In the alarm mode however, a buzzer goes off in conjunction with the LED light.

**Notification System:**

The notification system is on the user interface (node red). Essentially, it notifies the user of new changes to the system. For this device, the user is notified when the ESP is sent to sleep, when it is awoken, when the alarm state is changed, and when anomalous temperatures are detected. While the buzzer notifies the user of anomalous temperatures via sound, the notification system alerts the user via text message. This addition is useful for users that need assisted technology.

**Audio Notification System:**

Asides the buzzer that is activated in alarm mode, the user interface has an audio component that reads out notification messages.

**Task Manager for CPU Usage**

The task manager is under “Demo” in the navigation bar on node red. The task manager essentially shows the CPU usage in terms of percentage and shows a chart detailing the usage over a time interval.

**Sleep Mode**

The sleep mode is activated from the user interface by clicking on the “sleep” button. The sleep mode turns of certain components of the ESP 32 hardware device. Effectively, the device consumes less power than it does when its active.

**Measuring and Displaying humidity, temperature and heat index levels:**

The system comes with a temperature and humidity sensor that detects the temperature and humidity of the surrounding environment, reports it to the ESP 32 device, which, in turn, displays it on the user interface.

## a full system design using a data flow graph (a block diagram of the system showing the flow of information. Arrows point from source to destination. Rectangles represent hardware components. Ovals represent software modules)

BUZZER

LEDs (2)

BREADBOARD

SENSOR

ESP 32 Device

## a schematic diagram of your system hardware implementation

Diagram, schematic

Description automatically generated

## A breadboard view of your system hardware implementation (use a fritzing diagram and add IMAGES OPTIONALLY)

Diagram

Description automatically generated

## system software implementation

Refer to “flows.json” or “finalProjectFlow.txt” in the repository.

Both files contain the code used to generate the front end of the IoT device.

To test it with node red, navigate to import (under the triple bars to the top left on node-red) and either import json file from local PC or paste the code from the .txt file into the space provided.

## C/C++ source Code

See main.cpp under src

## Materials, Devices and Instruments used in the project

* LEDs
* Resistors
* F.M Cables
* Bread board
* ESP-32 device
* Jumper Wires
* Piezo Speaker
* Humidity and Temperature sensor
* Node-Red Graphical User Interface

## Project Methods and Procedure

The methods are explained in-depth in the README file in the C++ code.

## Trouble in the Project and how you solved it

I had a few headaches trying to figure out how to subscribe and publish data using the MQTT protocol but my biggest headache was by far figuring out how to use node red. The problems I had and my solutions are detailed below.

GUI Issues:

I had a couple of issues directing input from the client to various nodes on the user interface. I have multiple nodes on my interface and I wanted to direct inputs to specific nodes but everything that was published to the MQTT port was reflected on all the nodes.

I solved this problem by using a switch node to direct input to other nodes based on different conditions. A condition like “if string” to separate messages (assume string) from climate levels (assume number) did not do much good however because all data that is sent to the MQTT port is of type string.

I found, after much frustration, that being specific with the condition is the key to directing the right input into the correct node. For example, I send warning messages when the climate levels recorded are abnormal. To direct these messages to the appropriate node, I use a condition like “if contains ‘WARNING’ “, then send to specific node.

For the humidity, temperature and heat index levels, I concatenated a “key” to the string that was to be published based on the type.

newsI = newsI + "H";

newsh = newsh + "%";

newst = newst + "°C";

“H” for heat index, “%” for temperature, and “℃” for temperature. Then I used a “change” node to direct input. Essentially, the node collects an input and changes it, either by setting a different value or deleting it etc.

Graphical user interface, text, application, email

Description automatically generated

My reasoning was that by leaving the replace field empty, it will be replaced with an empty string and it worked. So, if the switch node has a condition, “if contains %” and is connected to a change node which searches for “%” and replaces it with “ ”, whatever is sent to the final node is the original value without the concatenated key.

## Results and Conclusion

The IoT device is equipped with the following functionalities:

* Alarm Mode
* Notification System
* Audio Notification System
* Task Manager for CPU Usage
* Sleep Mode for reduced power consumption
* Measuring and displaying humidity, temperature and heat index levels

All functionalities (excluding the buzzer in alarm mode) have be tested extensively and were found to work in good condition.

On a personal note:

I think this was a very informative lab. It was an eye opener to how software and hardware work in unison – something I have keen interest in. I thoroughly enjoyed working on it (especially with node red).