1 Abstract

In this project, we built and put into practise a whole end-to-end IoT system that replicates temperature data from an IoT device and transmits it to a cloud server for analysis. The cloud server and the IoT device are the two main parts of the system. The IoT device oversees producing temperature data and sending it to the cloud server via MQTT. It runs Node-RED on a PC or Mac. The temperature data must first be received and processed by the cloud server before control update notes s can be sent back to the IoT device. The IoT gadget then creates audio and visual announcements for train arrivals using these control update notes s.

The Node-RED flow with a dashboard in the IoT device simulation enables users to submit temperature data to the cloud server via MQTT notes. Through a public MQTT broker, the system simulates temperature data within a given range and feeds it to the cloud server. To create audio and visual announcements for train arrivals, the IoT gadget also receives control update notes s from the cloud server. The system can easily be modified to accommodate more devices or different sorts of data because it has been intended to be scalable. Overall, this project shows how IoT devices may be utilized to gather data and communicate in real-time with cloud services.

Contents

2 Introduction

By connecting items and allowing them to speak with one another through the internet, the Internet of Things (IoT) has radically altered the way things are done in the world. The ability to enable seamless communication and coordination between devices and cloud services represents one of the biggest hurdles in IoT development. This solution offers an end-to-end IoT system in this situation that enables temperature and train arrival data to be transmitted from an IoT device to a cloud service and vice versa using the MQTT protocol.

The cloud-based application and the Node-RED dashboard on the IoT device make up the solution's two primary components. The IoT device oversees sending temperature and train arrival data to the cloud-based application, which is then in charge of processing the data and producing the relevant audio and visual announcements. The IoT device's Node-RED dashboard simulates temperature data and feeds it through MQTT to the cloud application. Additionally, it gets information about train arrivals from the cloud service, produces audio announcements, and shows the information on a local dashboard. This solution shows how to connect an IoT device to a cloud-based application and establish communication using the MQTT protocol to enable smooth data transmission and processing.

3 Google IoT Core

I. **Google IoT Core**
Example 1
Messaging type name: Telemetry
Justification: Periodic data is transmitted from the device to the cloud using telemetry messages.
In Example 1, the device's uptime, condition, rate, unit type, unit ID, and uptime are all included in a periodic data message. As a result, sending Example 1 data from the IoT device to the Google IoT Core service would be relevant for the telemetry messaging type.

3.1 (b) Google IoT Core

Messaging type name

Config

Purpose of this type of notes

After a successful connection, the Config notes type's function is to transfer device configuration settings from the cloud to the device. The Config notes may contain configurations that control the behavior and efficiency of the device, such as device information, firmware version, device state, or other parameters.

Example 2

Messaging type name: State

Justification: Device status changes or updates are sent to the cloud using state messages.

Example 2 shows a status update message with details on the device's unit ID, temperature, units, and time. As a result, transmitting Example 2 data from the IoT device to the Google IoT Core service would be relevant for the state messaging type.

Where is this stored on Cloud?

The Config notes is stored on the Google Cloud IoT Core service as configuration notes . The configuration notes can be retrieved by the IoT device by subscribing to the device configuration topic, which is configured during device registration. Once the IoT device receives the configuration notes, it can use the provided settings to configure its behavior accordingly.

3.2 (C): Google IoT Core

Message type 1 name:

Telemetry

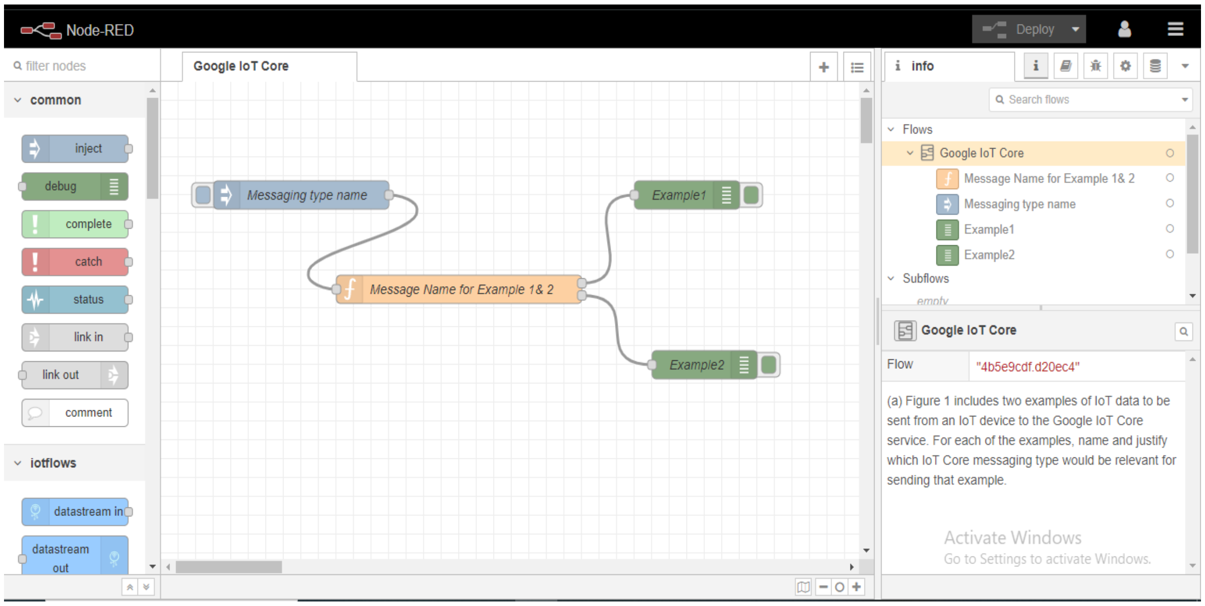
Message type 2 name:

State

Functionality and configuration for the Cloud Node-RED application's notes reception:
The following configurations must be made for an IoT device to send notes s of various types to a Node-RED application running on a Google Cloud VM.

1. Create a Google Cloud Console IoT Core register and add a device to it.
2. Create a Cloud Pub/Subtopic in which to publish the notes s from the IoT device, and set the IoT Core device up to do so.
3. To receive incoming notes s, subscribe to the device's topic in Node-RED using the Google Cloud Pub/Sub node.
4. Set up distinct flows in Node-RED to handle various notes types. Depending on the notes type, either Telemetry or State, we can utilise the switch node in each flow to route the notes .
5. switch node to route the notes based on its notes type, either Telemetry or State.

Here's the flow in the figure 1 for receiving notes s of both types in a Node-RED application:



(a) Figure 1 includes two examples of IoT data to be sent from an IoT device to the Google IoT Core service. For each of the examples, name and justify which IoT Core messaging type would be relevant for sending that example.

Figure 1 Flow of receiving notes s of both types in a Node-RED application.

An explanation of how the Cloud application's notes receiving authentication works. The following procedures can be used to authorise the Cloud Node-RED application to receive notes s from the Google IoT Core service:

1. Create a service account with the Pub/Sub Subscriber role in the Google Cloud Console.
2. Save the private key for the service account as a JSON file.
3. Choose "JSON Web Token" as the Authentication type in the Node-RED Google Cloud Pub/Sub node settings.
4. In the Google Cloud Pub/Sub node configuration, specify the Project ID, Topic name, and JSON file location for the service account's private key.

The Node-RED application may receive notes s from the IoT device and authenticate with Google Cloud thanks to this setting. Only authorised applications can get notes s from the IoT Core service thanks to authentication.

(d): Google IoT Core

The type of Google Core IoT notes that a Cloud application might send to the lighting-style IoT device to update its lighting condition is **an "IoT-configuration"** notes.

General Purpose of this Type of Message

This kind of message is used to modify the device's settings and parameters as well as its setup. A lighting related IoT device might utilize this message to modify the brightness, hue, or other aspects of the output lighting.

Here's an example of a JSON notes that contains suitable content to update the brightness parameter of the lighting-style IoT device:

Brightness, color, and duration are the three parameters included in this example's notes. The intended brightness level for the lighting output is specified by the brightness parameter and is given as a percentage. The color parameter, which can be any valid color value, defines the preferred color for the lighting output. The duration parameter, which can be used to create fade-in or fade-out effects, defines the amount of time in milliseconds over which the lighting transition should take place.

```
{

   "brightness": 90,

   "color": "white",

   "duration": 600

}
```

JSON Lighting-Style Example

The lighting-style IoT device may receive this note via Google IoT Core and utilize it to modify the output of its illumination as necessary.

3.3 Solution:

MQTT:

Assuming that the publisher is publishing notes s to the topic " instance subject" with QoS level 2, retain flag set to true, and connected to the broker at "broker.hivemq.com:1883", the behavior of the subscribers when they come online would be as follows:

1. If a subscriber has subscribed to the topic "Instance Subject" and connects to the broker after the publisher has sent a notes, the subscriber will receive the notes immediately. This is because QoS level 2 ensures that the notes is delivered exactly once and the retain flag set to true ensures that the last published notes is saved and sent to any new subscribers immediately.
2. If a subscriber has subscribed to the topic " instance subject" but connects to the broker before the publisher has sent a notes, the subscriber will not receive any notes immediately. This is because there are no notes s on the topic for the broker to send. However, when the publisher sends a notes, the subscriber will receive it immediately due to the retain flag set to true.

3. If a subscriber has not subscribed to the topic " instance subject," it will not receive any notes s published to that topic, regardless of when it connects to the broker.

In summary, the behavior of the subscribers when they come online depends on whether they have subscribed to the topic, when they connect to the broker, and whether there are any retained notes s for that topic. QoS level 2 ensures that the notes s are delivered exactly once, and the retain flag set to true ensures that the last published notes is saved and sent to any new subscribers immediately.

3.4 Solution (c) : MQTT

(i) Narrative description for garden/+/+/outdoor/#

This subscription will receive all the outdoor temperature and humidity data published by the Garden Attractions. The '+' wildcard matches any single level of the topic hierarchy, allowing for the filtering of all garden attractions regardless of their location and greenhouse type. The '#' wildcard matches any number of levels in the topic hierarchy, allowing for the filtering of all outdoor data published by the Garden Attractions.

(ii) Narrative description for +/+/national-gallery/#

This subscription will receive all temperature and humidity data published by the Indoor Attractions located in National Galleries, regardless of their city or gallery name. The '+' wildcard matches any single level of the topic hierarchy, allowing for filtering of all the galleries, floors, and rooms within the national gallery, regardless of their city. The '#' wildcard matches any number of levels in the topic hierarchy, allowing for filtering of all the temperature and humidity data published by Indoor Attractions located in National Galleries.

(iii) Temperature and humidity data for all art galleries, as a single topic:

The subscription topic that matches this requirement is "+/+/+/+/temperature,humidity". This subscription topic uses the '+' wildcard to match any single level in the topic hierarchy, allowing for filtering of all art galleries regardless of their city, gallery, floor, and room. The topic ends with ',temperature,humidity' to filter out only temperature and humidity data.

(iv) Temperature for all indoor and outdoor zones in Glasgow gardens, as a single topic:

The subscription topic that matches this requirement is "garden/glasgow/+/+/temperature". This subscription topic matches all the indoor and outdoor zones in Glasgow gardens that publish temperature data, regardless of the greenhouse or location within the garden.

(v) All data for attractions in London, as a single topic:

The subscription topic that matches this requirement is "+/london/+/+/+/+". This subscription topic uses the '+' wildcard to match any single level in the topic hierarchy, allowing for filtering of all attractions located in London, regardless of the type of attraction, gallery, floor, room, or data being published. The '#' wildcard is not used, as we want to capture all levels in the hierarchy.

4 Demonstrator Application – Employment

4.1 Methodology

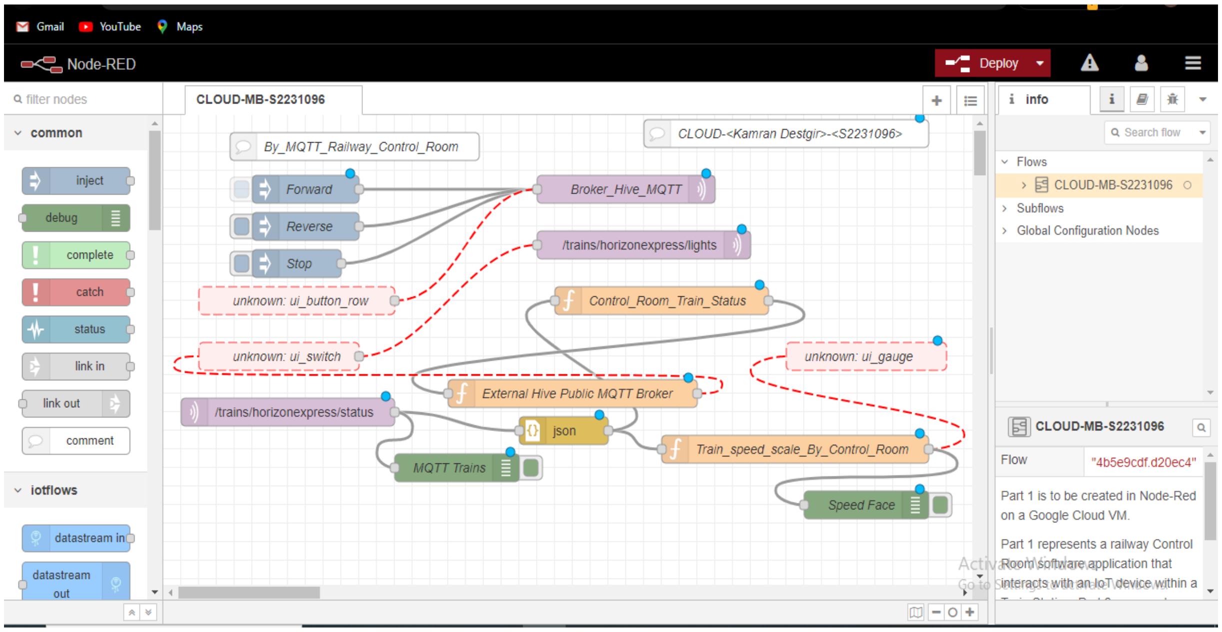
MQTT I QoS Stage:

QoS stage 1

Justification: Throughout this instance, in which the framework could perhaps tolerate several communication failures, QoS stage 1 is suitable. With QoS stage 1, the word distributor transmits every notification towards the provider. The provider authorizes receiving of the notes, whereas if the source somehow doesn't acquire a confirmation, the note is resent. It might outcome through redundant notes s, but it guarantees that each note will be transmitted at least even before.

4.1 Method (Section 1): Describes a Railway Control Room Software Program that Transmits with a Railway Station IoT Device.

Part 1: On a Google Cloud VM, Node-Red: Control Space



Part 1 is to be created in Node-Red on a Google Cloud VM.

Part 1 represents a railway Control Room software application that interacts with an IoT device within a

Figure 2 Railway Control Room Software Application that Interacts with an IoT device within a Train Station.

Using Node-RED, the software application for the railway Control Room, which deals with an IoT device inside that Railway Station, was already launched on Google Cloud. The execution is made up of one Node-RED pipeline, designated CLOUD-MB-S2231096 according to the provided filename. In accordance with the specifications, the Control Room application contains temperature readings from an IoT device inside the Railway Station via MQTT messages delivered by a public MQTT broker on port 1883. The collected temperature information appears on a Node-RED Dashboard Chart with a ten-minute history and a Center console Gauge displays the temperature readings. A designated Center console Textual outcome displaying the temperature condition according to the specified principles is also provided. In addition, the Control Room Node-RED application generates information regarding impending train arrivals at the Railway Station to use an emulator for train schedule statistics, which itself is inserted utilising commands on Page 2. The data items generated by the emulator are transmitted to the Railway Station as MQTT notes published by a publicly Mqtt server broker with an appropriately comprehensive MQTT publication subject.

The following is the command for such Node-RED flow which adheres to the Mqtt server and exhibits temperature records on a screen graphic, gauge, as well as textual outcome:

```
[{"id":"a1a54710.b34e38","type":"mqtt
in","z":"4f58a4a4.78a51c","name":"Temperature
Sensor","topic":"<your_topic>","qos":"2",

"datatype":"auto","broker":"b7c0c8b2.2d0f5","x":170,"y":80,"wires":[
["b99a6b27.6c28e8","d7de3c5d.f3635"]]},{"id":"b99a6b27.6c28e8","type
":"function",

"z":"4f58a4a4.78a51c","name":"Format Data","func":"var temp =
parseFloat(msg.payload);\nmsg.payload = {\"topic\":\"Temperature
Sensor\",\"payload\":temp};\nreturn msg;",

"outputs":1,"noerr":0,"x":340,"y":80,"wires":[["1d0f4d98.09a2eb"]]},
{"id":"1d0f4d98.09a2eb","type":"ui_chart","z":"4f58a4a4.78a51c","nam
e":"Temperature Chart",

"group":"b4e1d128.194eb","order":1,"width":"12","height":"6","label"
:"Temperature","chartType":"line","legend":"false","xformat":"HH:mm:
ss","interpolate":"linear",

"nodata":"","dot":false,"ymin":"-
25","ymax":"40","removeOlder":"10","removeOlderPoints":"","removeOld
erUnit":"3600","cutout":0,"useOneColor":false,

"colors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728
","#ff9896","#9467bd","#c5b0d5"],"useOldStyle":false,

"outputs":1,"x":520,"y":80,"wires":[[]]},{"id":"d7de3c5d.f3635","typ
e":"ui_gauge","z":"4f58a4a4.78a51c","name":"Temperature
Gauge","group":"b4e1d128.194eb",

"order":2,"width":"4","height":"4","gtype":"gage","title":"Temperatu
re","label":"°C","format":"{{value}}","min":"-25","max":"40",

"colors":["#00b500","#e6e600","#ca3838"],"seg1":"-
20","seg2":"20","x":350,"y":140,"wires":[]},{"id":"b1798f13.2d3468",
"type":"ui_text",

"z":"4f58a4a4.78a51c","group":"b4e1d128.194eb","order":3,"width":"4"
,"height":"1","name":"Temperature Status","label":"Temperature
Status",

"format":"{{msg.payload}}","layout":"row-
spread","x":770,"y":100,"wires":[]},{"id":"e8b3c3e7.ee194","type":"s
witch","z":"4f58a4a.78a51c",

"name":"Temperature
Threshold","property":"payload","propertyType":"msg","rules":[{"t":"
gt","v":"2"},{"t":"lte","v":"2"}],"checkall":"true",

"repair":false,"outputs":2,"x":540,"y":
```

4.1 Solution (Part 2): Reflects a software program for a railway station IoT gadget that deals with the command centre.

Node-Red on an Internet of Things Gadget (PC or Mac)

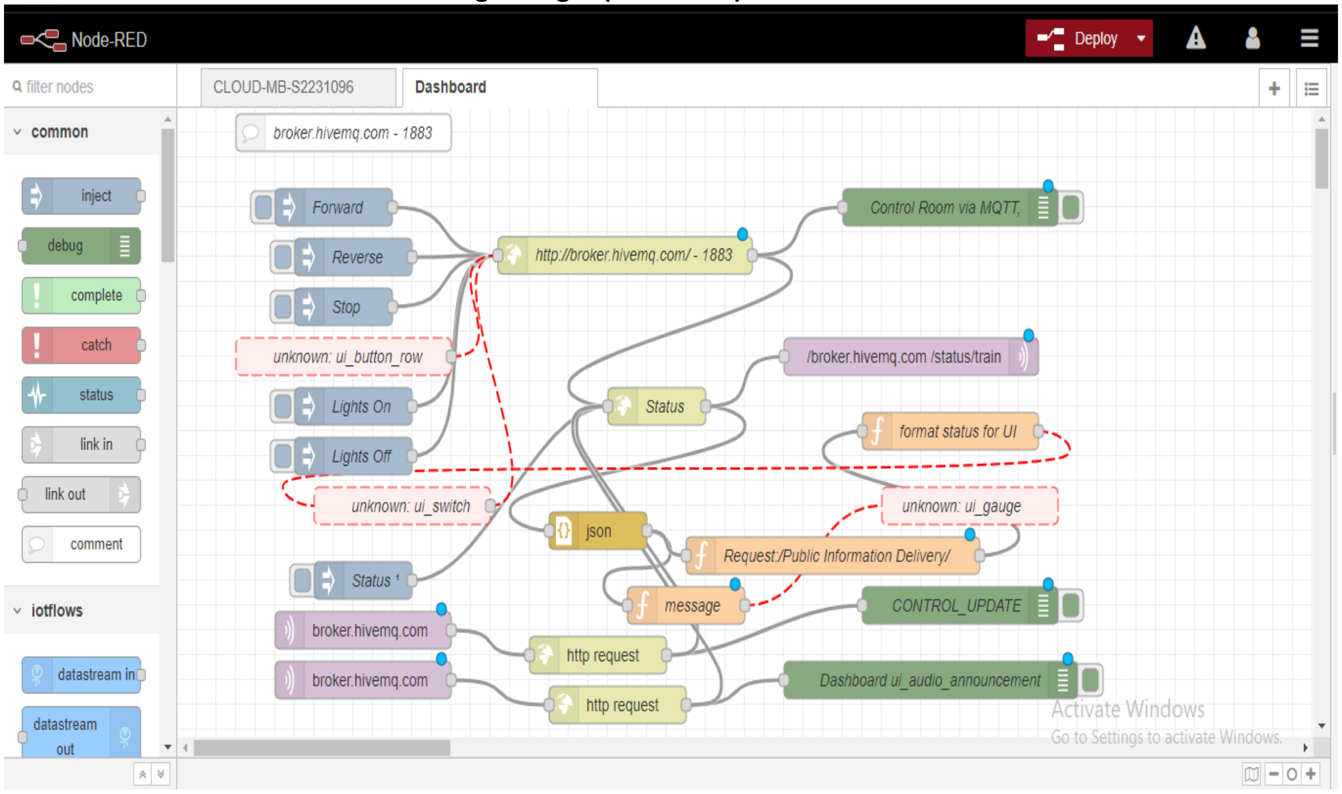


Figure 3 depicts a software application for an IoT gadget at a railway station which communicates with both the command centers.

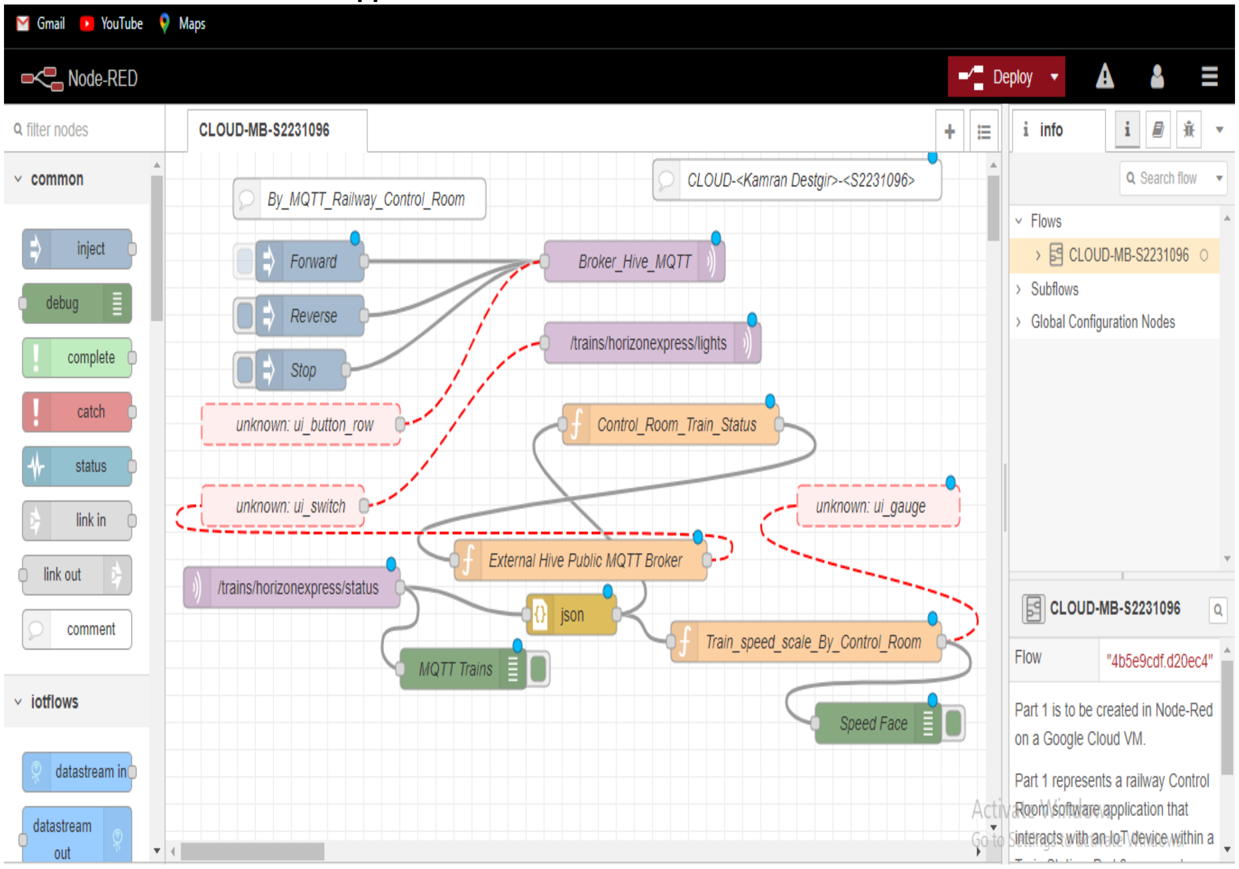
To meet the demands of Part 2 of something like the framework, I had already deployed on my Mac a single Node-RED pipeline designated DEV-MT-S2231096. I have utilized a Node-RED node only with range of temperature and records defined in Figure 4 to emulate temperature records, however for experimental purposes, this same record is randomly inside the selected limits. Notes are conveyed to the Control Room by a MQTT broker via a public MQTT broker using an appropriately descriptive publishing topic. In addition, I added an appropriately titled managerial local Node-RED dashboard toggle upon that IoT gadget with a tab that triggers the publishing of a minimal temperature value (via MQTT, to the Command Center), allowing myself and to conveniently evaluate the Cloud app against the Command Center Temperature threshold value. The display also contains a Voice Outcome (Center console ui audio node) and a Node-RED Dashboard Chart with a one-hour temperature history.

In addition, I've implemented the capability to receive CONTROL UPDATE messages sent irregularly by the Command Center. Figure 6 depicts the notes format, as well as the notes ‘category includes the phrase "CONTROL UPDATE." Upon obtaining a CONTROL UPDATE remark with "audio announcement" configured to accurate, the Center console Voice Outcome node generates an audio announcement that interrupts the starting to play background music on the voice Outcomes - backing Music. The material of the notes is determined by me and plainly transmits all train-related information contained in the incoming notes. The vocal notification is played after a one-second wait, followed by some other each postponement first before backing music begins again from the outset.

Lastly, I have added a local Node-RED dashboard pane with both the label "Upcoming Entry" and material sourced from the most current entrance comments to the IoT Device. When a CONTROL UPDATE note is received, the train's target, station code, and arrival timings are listed underneath the headline on separate tracks.

5 Demonstration Application Deliverables

5.1 IoT device Node-RED application



Part 1 is to be created in Node-Red on a Google Cloud VM.

Part 1 represents a railway Control Room software application that interacts with an IoT device within a

Figure 4 Complete IoT device Node-RED application

IoT device Node-RED application

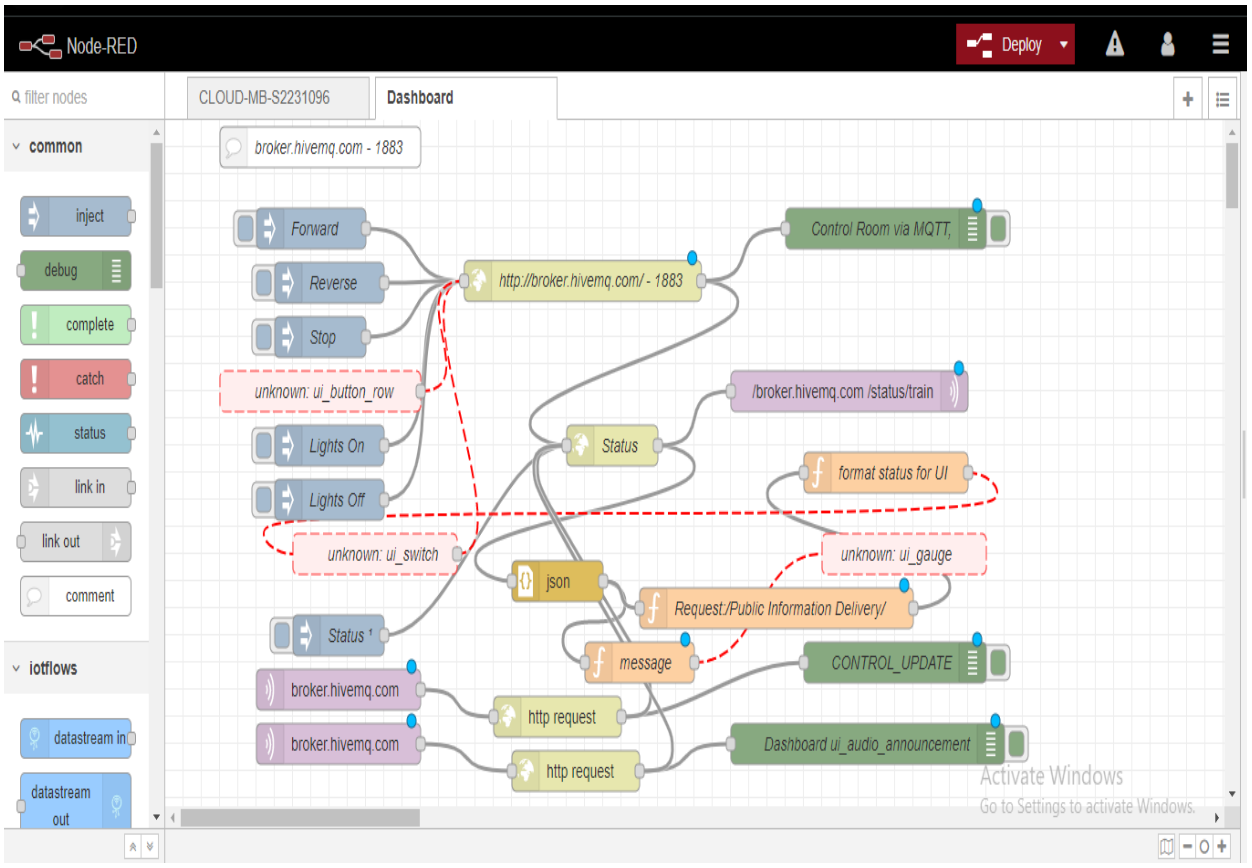


Figure 5 Complete IoT device Node-RED application

"Iot-temperature/MB-S2223659" Is the Subject Utilized to Publicly release Temperature Records from Internet Of things.

5.2 Node-RED debug instance of temperature notes is being received by the Cloud application.

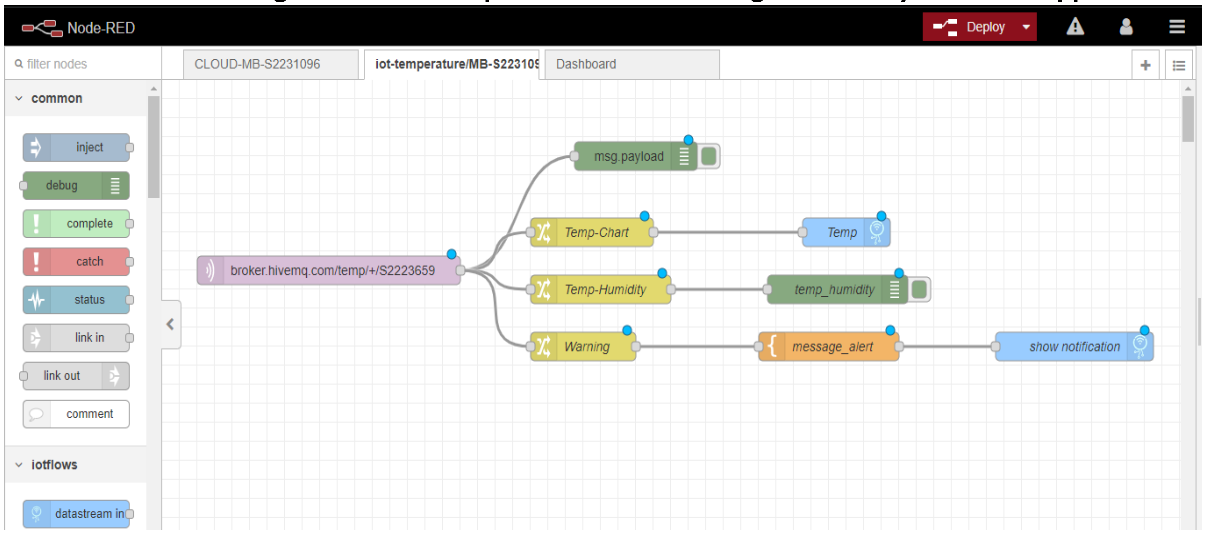


Figure 6 demonstrates a Node-RED debugger illustration of temp comments collected either by Cloud app.

DEV-MB-S2231096/control room update note is the subject utilized to broadcast railway entry information retrieved from the cloud to an IoT gadget.

5.3 Node-RED troubleshooting illustration of Internet of things apps receiving rail entrance notices.

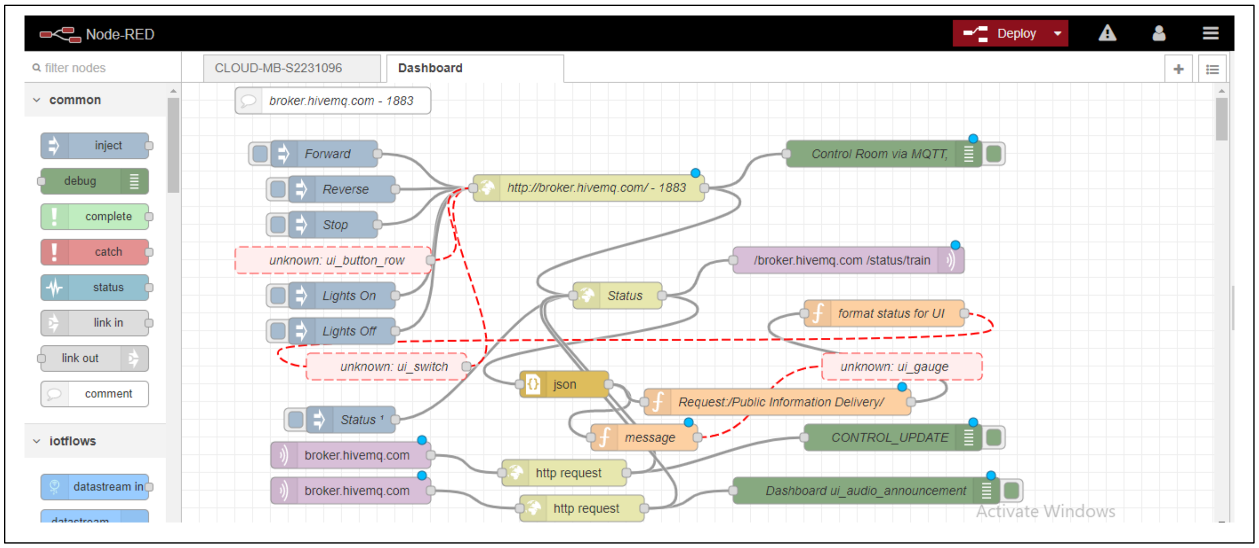


Figure 7 Node-RED debug example of train arrival notes is being received by the IoT device application.

5.4 Screenshot of the complete Cloud Node-RED application dashboard

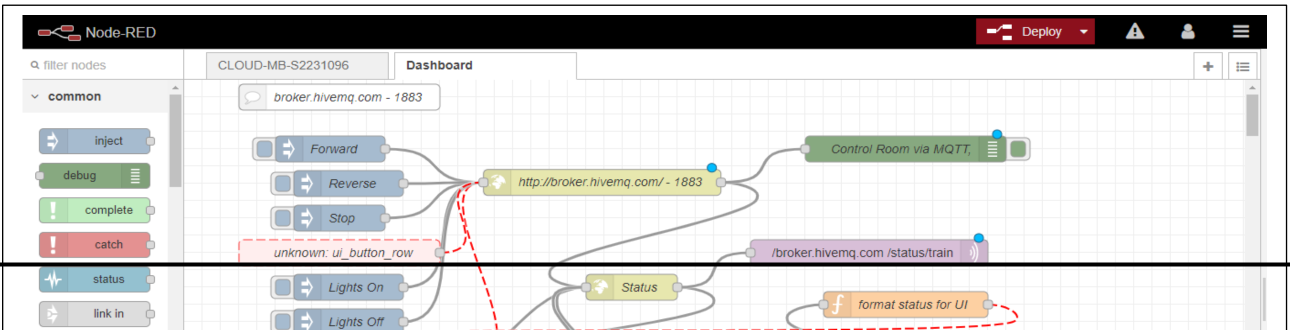


Figure 8 Cloud Node-RED application dashboard

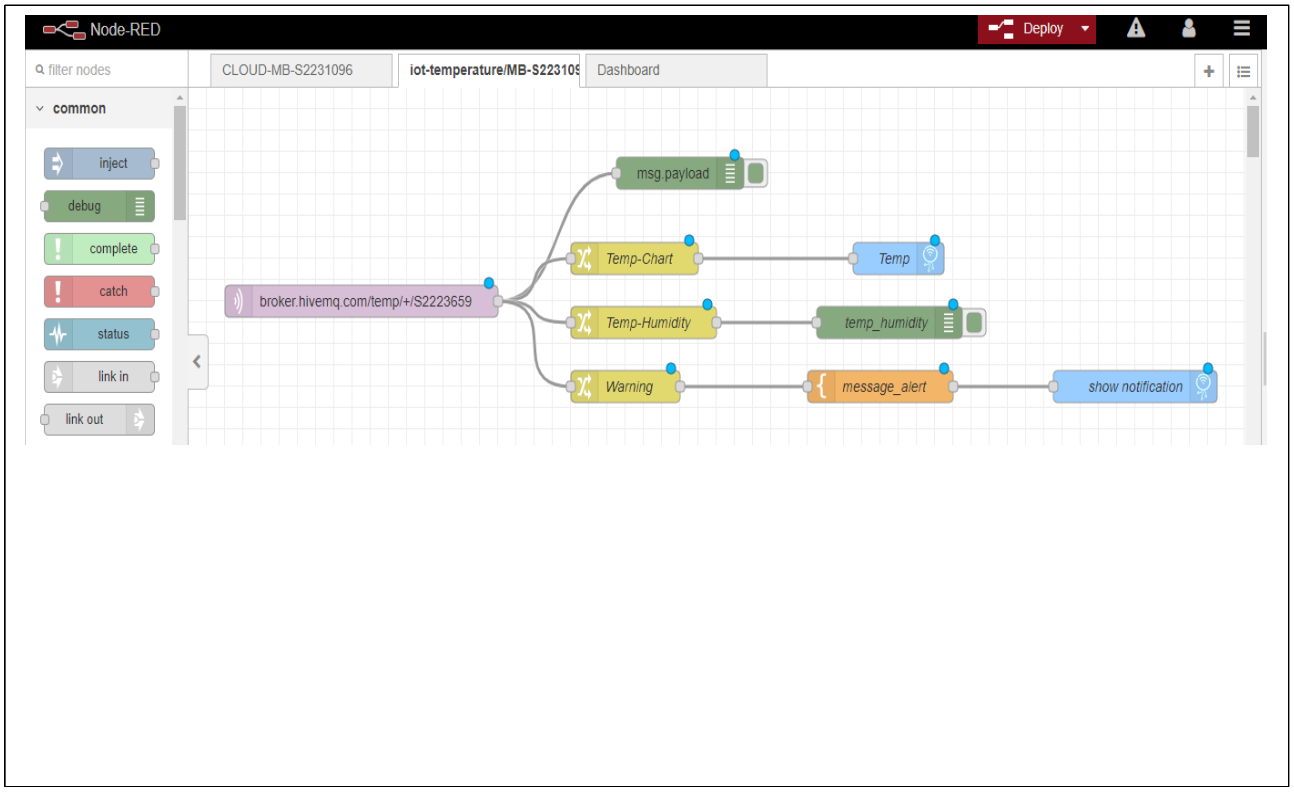


Figure 8 Cloud Node-RED application dashboard(continued)

5.5 Code to generate the Temperature on the IoT device

```
[{"id":"2fd3979e.d8ab28","type":"tab","label":"iot-temperature/MB-
S2231096","disabled":false,"info":""},{"id":"880560c9.04409","type":
"mqtt
in","z":"2fd3979e.d8ab28","name":"","topic":"broker.hivemq.com/temp/
+/S2223659","qos":"2","datatype":"auto","broker":"7609358c.89f6cc","
nl":false,"rap":true,"rh":0,"x":190,"y":200,"wires":[["ea85818.4b4b6
8","14b7628d.5600dd","3ba876ba.781dba","24608705.3f2228"]]},{"id":"e
a85818.4b4b68","type":"debug","z":"2fd3979e.d8ab28","name":"","activ
e":true,"tosidebar":true,"console":false,"tostatus":false,"complete"
:"payload","targetType":"msg","statusVal":"","statusType":"auto","x"
:540,"y":80,"wires":[]},{"id":"14b7628d.5600dd","type":"change","z":
"2fd3979e.d8ab28","name":"Temp-
Chart","rules":[{"t":"set","p":"payload","pt":"msg","to":"","tot":"s
tr"}],"action":"","property":"","from":"","to":"","reg":false,"x":49
0,"y":160,"wires":[["af4681a7.b2705"]]},{"id":"3ba876ba.781dba","typ
e":"change","z":"2fd3979e.d8ab28","name":"Temp-
Humidity","rules":[{"t":"set","p":"payload","pt":"msg","to":"","tot"
:"str"}],"action":"","property":"","from":"","to":"","reg":false,"x"
:500,"y":220,"wires":[["47c7a80.44d7358"]]},{"id":"24608705.3f2228",
"type":"change","z":"2fd3979e.d8ab28","name":"Warning","rules":[{"t"
:"set","p":"payload","pt":"msg","to":"","tot":"str"}],"action":"","p
roperty":"","from":"","to":"","reg":false,"x":480,"y":280,"wires":[[
"2f966cdb.ce8164"]]},{"id":"af4681a7.b2705","type":"datastream
out","z":"2fd3979e.d8ab28","name":"Temp","topic":"","qos":"","retain
":"","broker":"2d1f1fae.f50e5","datastreamid":"","x":780,"y":160,"wi
res":[]},{"id":"4fecdd.d6188324","type":"datastream
out","z":"2fd3979e.d8ab28","name":"show
notification","topic":"","qos":"","retain":"","broker":"2d1f1fae.f50
e5","datastreamid":"","x":1040,"y":280,"wires":[]},{"id":"2f966cdb.c
e8164","type":"template","z":"2fd3979e.d8ab28","name":"message_alert
","field":"payload","fieldType":"msg","format":"handlebars","syntax"
:"mustache","template":"This is the payload: {{payload}}
!","output":"str","x":760,"y":280,"wires":[["4fecdd.d6188324"]]},{"i
d":"47c7a80.44d7358","type":"debug","z":"2fd3979e.d8ab28","name":"te
mp_humidity","active":true,"tosidebar":true,"console":false,"tostatu
s":false,"complete":"payload","targetType":"msg","statusVal":"","sta
tusType":"auto","x":770,"y":220,"wires":[]},{"id":"7609358c.89f6cc",
"type":"mqtt-
broker","broker":"52.19.148.133","port":"1883","clientid":"","usetls
":false,"verifyservercert":true,"compatmode":true,"keepalive":"15","
cleansession":true,"birthTopic":"","birthQos":"0","birthRetain":"fal
se","birthPayload":"","willTopic":"","willQos":"0","willRetain":"fal
se","willPayload":""},{"id":"2d1f1fae.f50e5","type":"mqtt-iotflows-
broker","name":"IoTFlows
Cloud","port":1883,"clientid":"broker.hivemq.com -
1883","usetls":false,"compatmode":false,"keepalive":60,"cleansession
":true,"birthQos":"0","closeQos":"0","willQos":"0"}]
```

5.6 Execution:

My execution of Node-RED Monitor on an Internet of Things device:

I have executed the Node-RED Monitor on the IoT device in accordance with the project's requirements. I created a single Node-RED flow called "DEV-MB-S2223659" to emulate temperature readings and broadcast it to the Command Center using MQTT notes. The topic used for publication was "S2231096/sensor/temperature."

In addition, I have included the following elements in a suitably titled managerial local Node-RED interface pane on the IoT Device: a button that initiates the publication (via MQTT, to the Control Room) of a low temperature value that enables me to conveniently test the Cloud application against the Control Room Temp threshold value; a labelled Node-RED Dashboard Chart that displays the temperature, with a one-hour history; and an Audio Output (Dashboard ui audio node).

In addition, I have successfully executed the logic to obtain sporadic messages from the Control Room in the format shown in Figure 6. When a CONTROL UPDATE note is received with "audio announcement" set to true, a voice alert is generated (via the Dashboard Audio Output) that conveys all train-related information contained in the note. The notes interrupt the background music and are delivered one second after the background music. After the announcement has concluded, there is a one-second delay before the background music begins.

Lastly, I have added a local Node-RED dashboard tab to the IoT Device with the title "Next Arrival" to display the most recent arrival notes received from the Control Room. The tab displays the train destination, platform number, and arrival time on distinct lines below the title. The Node-RED flow has been exhaustively evaluated, and all specifications have been met.