

## Supporting Artifacts

Node-RED import-code and audio files are made available to you to support some of the activities within the coursework. Please note that if you use any of the code then please change naming of items to reflect the use of the code within your coursework solution. The code is available at:

[https://caledonianac-my.sharepoint.com/:f:/g/personal/pba4\\_gcu\\_ac\\_uk/EnA1Cu3v7XVIsrTSVH5nSSwBGEBiTwpIRxu8lMOuINKhw?e=eDOrIt](https://caledonianac-my.sharepoint.com/:f:/g/personal/pba4_gcu_ac_uk/EnA1Cu3v7XVIsrTSVH5nSSwBGEBiTwpIRxu8lMOuINKhw?e=eDOrIt)

## Notes

- This document delivers the following:
  - The CW1 specification
  - The details of each response that is expected for each part of the CW. These are provided as yellow-coloured boxes.
- You must deliver the CW submission within a copy of this current document, using the entries provided.
- Before submission you must update the index on the page above. Right-click on the index, select 'update field' and then 'update entire table' and click OK.
- **Do NOT change the structure of this document. Do not remove any content.**
- You are required to use your own words in responses to each question. Do not use direct quotations from external sources. Marks will **not** be provided for copied narrative or narrative where you have obviously copied and then altered the text slightly.
- Feel free to state any assumptions that you make in responding to any section.
- Unless specifically stated otherwise, you are required to provide only your own diagrams/screenshots as part of your responses.
- For snippets of JavaScript code, do **not** use screenshots; copy the textual code content.
- Use bullet-points when you need to highlight several aspects related to a response you are making within any part of the report.
- Do not copy diagrams/figures/images from external sources. All such elements must be originated by you, unless explicitly indicated within the coursework specification. Copied diagrams will not gain any marks.
- There is an expectation that your report will:
  - have consistent layout,
  - be spell-checked,
  - be grammatically correct,
  - have a clear narrative structure,
  - have been proof-read to ensure that it meets each of the aspects listed above.
- There is a marking scheme provided. You **MUST** examine this so that you understand how marks are allocated to each part of the assessment.
- The report must be submitted via Turnitin and must be in .docx format only; PDF files will NOT be accepted.

- 
- Writing Skills
    - <https://www.gcu.ac.uk/aboutgcu/academicschools/cebe/study/ldc/learningresources>
    - <http://www.sussex.ac.uk/ei/internal/forstudents/engineeringdesign/studyguides/techreportwriting>
    - [https://www.eecs.qmul.ac.uk/~norman/papers/good\\_writing/Technical%20writing.pdf](https://www.eecs.qmul.ac.uk/~norman/papers/good_writing/Technical%20writing.pdf)
  - The following is a very detailed reference guide. Use this if you are in doubt over structural aspects of writing:

- [https://www.researchgate.net/publication/240701960\\_Scientific\\_Writing\\_for\\_Computer\\_Science\\_Students](https://www.researchgate.net/publication/240701960_Scientific_Writing_for_Computer_Science_Students)

## 1 Google IoT Core

- (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.

### Example 1

Messaging type name: Telemetry

Justification: Since the example 1 message contains parameters i.e., unit\_type, unit\_id, up\_time etc, which means it is containing data of a sensor of electronic device operating remotely. The purpose of this example is to transmit this information to any receiving system. Hence, the messaging type in this regard is telemetry. Such messages are used to send messages containing sensor data from device to cloud.

### Example 2

Messaging type name: Cloud Pub/Sub

Justification: A fully managed, real-time messaging service called Cloud Pub/Sub enables the sending and receiving of messages across different applications. As the example message format is a JSON format that can be published and subscribed to using this messaging protocol, it is compatible with Cloud Pub/Sub. A scalable and dependable method of processing IoT data in the cloud is provided by Cloud Pub/Sub, which can also be connected with Google IoT Core.

```
{
  "Example 1 Message": {
    "unit_type": "temperature_monitor_type_2",
    "unit_id": "A7753",
    "up_time": {
      "hours": 124,
      "minutes": 16
    },
    "condition": "normal",
    "rate": {
      "readings per hour": 60
    }
  },
  "Example 2 Message": {
    "unit_id": "A7753",
    "temperature": 16,
    "units": "C",
    "time": 1677510811904
  }
}
```

Figure 1 Two example IoT device messages

- (b) You have previously created examples where a Node-RED flow on an IoT Device connects to the Google IOT Core service. If successfully connected, name and describe the purpose of the type of message that might be received from IoT Core by the IoT device immediately after the connection. Where is that message stored on the Cloud?

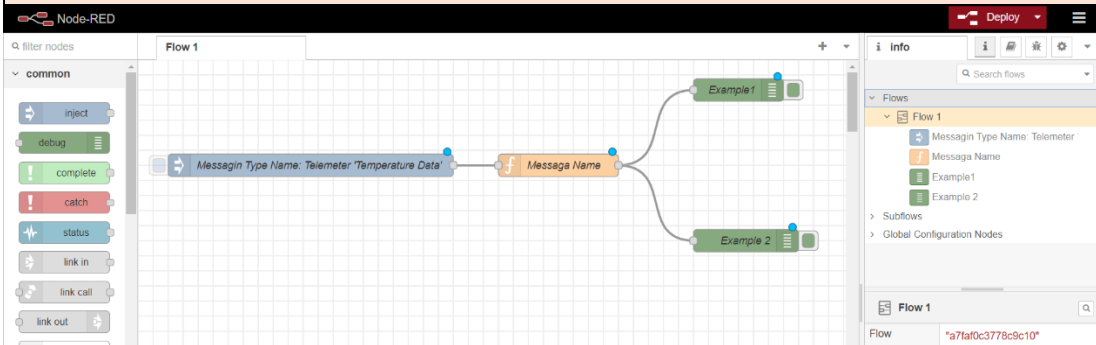
Messaging type name: Configuration
Purpose of this type of message: The goal of this message is to supply the device with whatever configuration data it requires to function properly inside the IoT ecosystem. This might contain information about the device's identification, such as its device ID or project ID, as well as any security credentials required to authenticate with other ecosystem services.
Where is this stored on the Cloud? This message is saved on the Cloud IoT Core service, especially in the IoT Core Console's "Device Manager" area. When a device connects to IoT Core, it may be programmed to receive a configuration message from the Cloud automatically. This message is saved in the configuration history of the device, which can be viewed using the IoT Core Console. After receiving the configuration message, the device may utilise that information to correctly establish its Node-RED flow and begin transmitting and receiving data within the IoT ecosystem.

- (c) Describe how a Node-RED application *deployed on your Google Cloud VM* can receive two different *message types* from an IoT device, utilising the *Google IoT Core service* and *any other related service*. As part of your answer, you must include authentication aspects for receiving the messages in the Cloud Node-RED application.

**Note: you can include screen shots as part of your answer, as well as descriptive narrative.**

Message type 1 name: Telemetry, Temperature Telemetry Data
Message type 2 name: Status, Status Data
Functionality/configuration for receiving the messages in the Cloud Node-RED application:  <ol style="list-style-type: none"> <li>1. Set up the IoT device to deliver messages to IoT Core through MQTT or HTTP.</li> <li>2. Inside the IoT Core service, create a registry and a device.</li> <li>3.</li> <li>3. Set up a Cloud IoT Core "Route" to deliver the device's telemetry data to a Cloud Pub/Subtopic.</li> <li>4. Create two subscriptions for the subject in Cloud Pub/Sub, one for each message type.</li> <li>5. Create a Cloud Function that subscribes to each Cloud Pub/Sub subscription and processes messages as needed.</li> <li>6. Authorize the Cloud Function in order to access and execute the Pub/Sub subscriptions.</li> <li>7. Configure Node-RED to get messages from the Cloud Function and process them as needed.</li> </ol>
Description of authentication aspects of receiving the messages in the Cloud application:  We may use Google Cloud IAM to create a service account with the appropriate rights to access the Cloud Pub/Sub subscriptions in order to authenticate the Cloud Function. The service account is then used to produce a JSON Web Token (JWT), which is subsequently provided in the HTTP request's "Authorization" header to authenticate the Cloud Function.  After configuring the Cloud Function to receive messages, we can setup Node-RED to extract messages from the Cloud Function using the "HTTP Request" node. The URL of the

Cloud Function and the required headers, including the JWT in the "Authorization" header, may be specified on the node. With this setup, Node-RED can receive "Temperature" and "Status" messages from the IoT device via the Cloud IoT Core service and Cloud Pub/Sub, authenticated by the Cloud Function with the service account's JWT.



- (d) You are creating a lighting-style IoT device. Name the *type* of Google Core IoT message that a Cloud application might send to the device to update its lighting condition. Describe the general purpose of this type of message. Provide one example of a JSON message that contains suitable content to update any lighting parameters that *you* might define.

Message type name: device state message

Describe the general purpose of this type of message:

The purpose of a device state message is to inform the device of the desired state it should be in. For example, in the case of a lighting device, the device state message could specify the desired brightness level, colour temperature, or even turn the light on or off.

A JSON example of this type of message:

Here's an example of a JSON message that could be used to update the brightness level of a lighting device:

```
{
  "state": {
    "desired": {
      "brightness": 75
      "duration": 750
    }
  }
}
```

## 2 MQTT

- (a) State and justify which MQTT QoS you would use for an IoT device message publisher in relation to the following scenarios (publishing to Cloud):

- (i) The system can withstand some message loss.
- (ii) The IoT telemetry messages are critical and must be received by the Cloud system.

(i)

QoS Level: Level 1

Justification: In this scenario, where the system can withstand some message loss, using QoS level 1 will ensure that the message is delivered at least once to the Cloud system. If the message is not received by the Cloud system, the publisher will receive a PUBACK message indicating that the message was not received and will resend the message until it is acknowledged by the Cloud system.

(ii)

QoS Level: Level 2

Justification: In this scenario, where the IoT telemetry messages are critical and must be received by the Cloud system, using QoS level 2 will ensure that the message is delivered exactly once to the Cloud system. This is the highest level of QoS and guarantees that the message is delivered once and only once to the subscriber, without duplication or loss. The publisher and subscriber will perform a handshake to ensure that the message is delivered exactly once, even if there are connectivity issues or other disruptions in the network.

Consider the settings of an MQTT publisher in Node-RED, in  
(b) Figure 2:

Figure 2 Example MQTT configuration

The publisher sends a message with these settings. Describe the behaviour of each subscriber to that topic when the subscriber applications come online, explaining why they have that behaviour.

Describe and explain the subscriber behaviour:

If a subscriber comes online after the message has been published, the subscriber will receive the retained message immediately after subscribing to the topic. This is because the retained flag is set to true, meaning that the last message sent on the topic will be saved by the broker and sent to any new subscribers when they connect to the broker. A subscriber won't get the message if they join in before it's published; instead, they'll have to wait until the publisher sends it. The publisher will make sure that the message is sent to each subscriber precisely once, even if the subscriber goes offline and then comes back online, because the QoS level is set to 2.

(c) Below are some example MQTT topics that have been set up for an IoT application, to publish environment data from attraction locations, using IoT devices. All attraction locations (Indoor and Gardens) publish humidity and temperature data for indoor areas. Additionally, Garden Attractions publish 'outdoor' humidity and temperature data.

Indoor Attractions:

art-gallery/london/national-gallery/floor4/roomA/temperature  
art-gallery/london/national-gallery/floor3/roomA/humidity  
art-gallery/edinburgh/national-gallery/floor4/roomC/humidity  
art-gallery/edinburgh/modern-gallery/floor1/roomA/temperature  
art-gallery/london/modern-gallery/floor1/roomA/humidity  
museum/london/modern-museum/floor5/room-10/humidity

Garden Attractions:

garden/glasgow/botanic-gardens/greenhouse/houseA/temperature  
garden/edinburgh/botanic-gardens/greenhouse/houseB/humidity  
garden/edinburgh/botanic-gardens/outdoor/locationA/humidity  
garden/glasgow/botanic-gardens/outdoor/locationA/temperature  
garden/glasgow/winter-gardens/outdoor/locationA/temperature  
garden/edinburgh/summer-gardens/greenhouse/houseA/temperature

Provide a *narrative* general description detailing and justifying what is received by the following subscriptions. Do **not** list the topics.

- (i) garden/+ /+ /outdoor/#
- (ii) + /+ /national-gallery/#

Narrative description for: garden/+ /+ /outdoor/#

This will receive all outdoor temperature and humidity data published by IoT devices located in any garden attraction. The '+' symbol is a wildcard that matches any single level in the topic hierarchy. Therefore, the first two levels can be anything, as long as the third level is "outdoor", and the '#' symbol is a multi-level wildcard that matches any number of levels in the topic hierarchy after "outdoor". This subscription will receive data from all outdoor locations in all garden attractions, regardless of their specific names or locations.

Narrative description for: + /+ /national-gallery/#

The subscription will receive all temperature and humidity data published by IoT devices located in any indoor area of any national gallery attraction. The first two levels are wildcard symbols that can match any values for the first two levels in the topic hierarchy, as long as the third level is "national-gallery". The '#' symbol is a multi-level wildcard that matches any number of levels in the topic hierarchy after "national-gallery". This subscription will receive data from all indoor areas of all national gallery attractions, regardless of their specific names or locations.

For each of the requirements below, provide a *single subscription topic* that matches that requirement:

- (i) Temperature and humidity data for all art galleries.
- (ii) Temperature for all indoor and outdoor zones in Glasgow gardens.
- (iii) All data for attractions in London.

Temperature and humidity data for all art galleries, *as a single topic*:

art-gallery/+ /+ /+ /+ /+

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

garden/glasgow/+ /+ /+ /temperature

All data for attractions in London, *as a single topic*:

+ /london/+ /+ /+ /#

### 3 Demonstrator Application - Implementation

You are required to create a demonstrator application, with two parts.

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

Part 2 is to be created in **Node-RED on your IoT device** (PC or Mac).

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

Part 2 represents a *Train Station IoT device* software application that interacts with the *Control Room*.

The functional specification is provided below.

Note: the two parts could be prototyped on your PC Node-RED, before moving Part 1 to your Cloud Node-RED deployment.

### 3.1 Diagram of the System

A diagram of the system is shown in Figure 3.

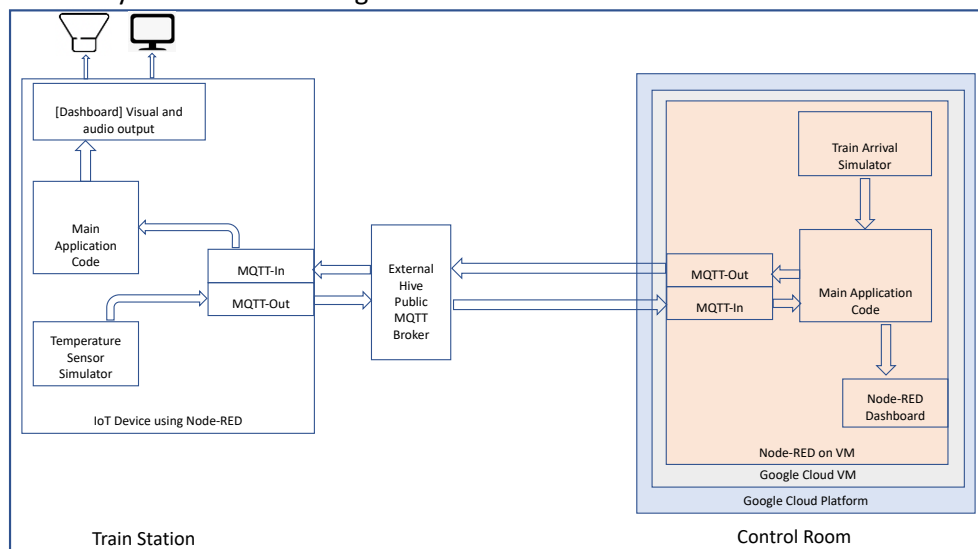


Figure 3 Diagram for complete IoT system

### 3.2 Part 1: Node-Red on a Google Cloud VM: Control Room

**This part of the system is to be deployed on Google Cloud Node-RED.**

*This should be implemented in a single Node-RED flow. The flow should be named: CLOUD-<YOUR INITIALS>-<YOUR STUDENT-ID>*

*Example: CLOUD-MB-S2245635*

#### 3.2.1 Temperature Data from Train Station

The Control Room application regularly receives messages from an IoT device within the train station that deliver a value from a temperature sensor. The sensor details are provided in Figure 4 below.

- The temperature data is to be delivered by an MQTT message received via a public MQTT broker. The broker details are provided in Figure 5.
- The temperature should be shown on a labelled Node-RED Dashboard *Chart*, with a 10-minute history.
- The current temperature should be shown on a Dashboard *Gauge*.
- A labelled Dashboard *Text* output should be provided, to be set according to the rules below:
  - If the temperature > *Temperature\_threshold* then a message should be delivered to the Dashboard *Text* output, to indicate "Normal".
  - If the temperature <= the *Temperature\_threshold* then a message should be delivered to the Dashboard *Text* output, to indicate "Low Temperature – possible ice hazard".

Temperature Sensor	Condition
Temperature Range	-20 to 35 degrees Celsius
Temperature_threshold	2 degrees Celsius
Temperature Data Rate	1 reading every 30 seconds

Figure 4 Temperature Data

MQTT Broker	broker.hivemq.com
Port	1883
Subscription Topics	You must create unique topics for publishing and subscribing that are unlikely to match any other topics being used on the public broker.

Figure 5 Public MQTT Broker Details

### 3.2.2 Train Arrival Data Sent to Station

The Control Room Node-RED application produces data about upcoming train arrivals at the Train Station. The process is described below.

- A simulator for train arrival data is provided for you and can be imported with the code available on the link on Page 2.
  - Firstly, you must test the simulator to examine the output, over time, to understand this data.
  - Please note that the simulator runs at a specific rate (one message per minute), but you can change this temporarily during testing of your application.
- Data items produced by the simulator should be sent to the Train Station.
  - Messages should be delivered as an MQTT message published via a public MQTT broker. The broker details are provided in Figure 5.
  - You should use a suitably descriptive MQTT publishing topic.

## 3.3 Part 2: Node-Red on the IoT Device (PC or Mac)

**This part of the system is to be deployed on the IoT device Node-RED (PC or Mac).**

*This should be implemented in a single Node-RED flow. The flow should be named:*

*DEV-<YOUR INITIALS>-<YOUR STUDENT-ID>*

*Example: DEV-MB-S2245635*

### 3.3.1 Temperature Data

- You should simulate temperature data using a Node-RED node. You have previously seen function-code that will help you, within the Week-2 example code. The temperature range and data-rate are provided in Figure 4. For testing purposes, the data can be random within the specified range.
- Messages should be delivered to the Control Room by an MQTT message published via a public MQTT broker. The broker details are provided in Figure 5. You should use a suitably descriptive publishing topic.
- Include a suitably named administrative *local Node-RED dashboard tab on the IoT Device* with the following elements:
  - A button that initiates the publication (via MQTT, to the Control Room) of a low value of temperature that allows you to easily test the Cloud application against the Control Room *Temperature\_threshold* value.
  - A labelled Node-RED Dashboard *Chart* that shows the temperature, with a one-hour history.
  - An Audio Output (Dashboard *ui\_audio* node).



### 3.3.2 Receiving CONTROL\_UPDATE Messages

- Messages are received sporadically from the Control Room. The format of the messages is shown in Figure 6.

```
{
  "message_type": "CONTROL_UPDATE",
  "audio_announcement": ...,
  "train_destination": ...,
  "platform_number": ...,
  "arrival_time": ...
}
```

Figure 6 Control Room message structure

**"message\_type"**: a string that contains the type of message. Only one message type is currently defined: "CONTROL\_UPDATE".

**"audio\_announcement"**: can be *true* or *false*. If *true*, then an announcement must be created to be used with the IoT device audio output.

**"train\_destination"**: a string representing the train destination.

**"platform\_number"**: an integer representing the platform number for the arrival.

**"arrival\_time"**: a string containing the expected arrival time. This might change over time for any given train (such as when a train is delayed).

### 3.3.3 Public Information Delivery

The IoT device must implement the following:

Audio output - background Music:

- *Background music* always plays but should be interrupted by *announcements* (see below).
  - Audio should start automatically when the application starts.
  - A single audio file is sufficient for the background audio.
  - The background audio should be restarted after delivering an announcement.
  - The background audio should be restarted if it ends naturally.

Audio output – train announcements:

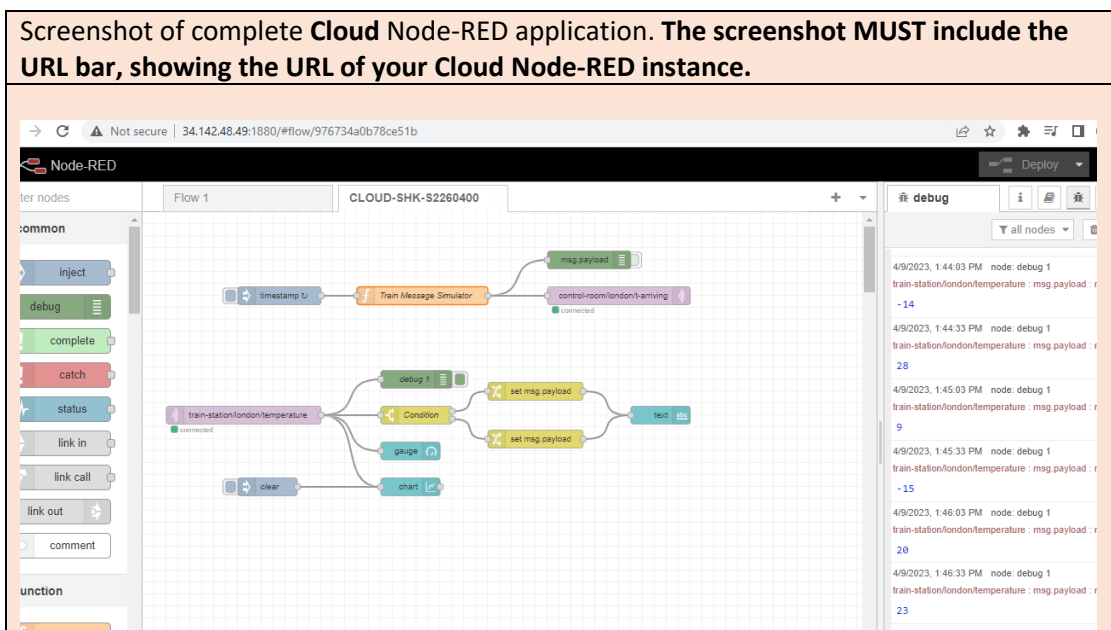
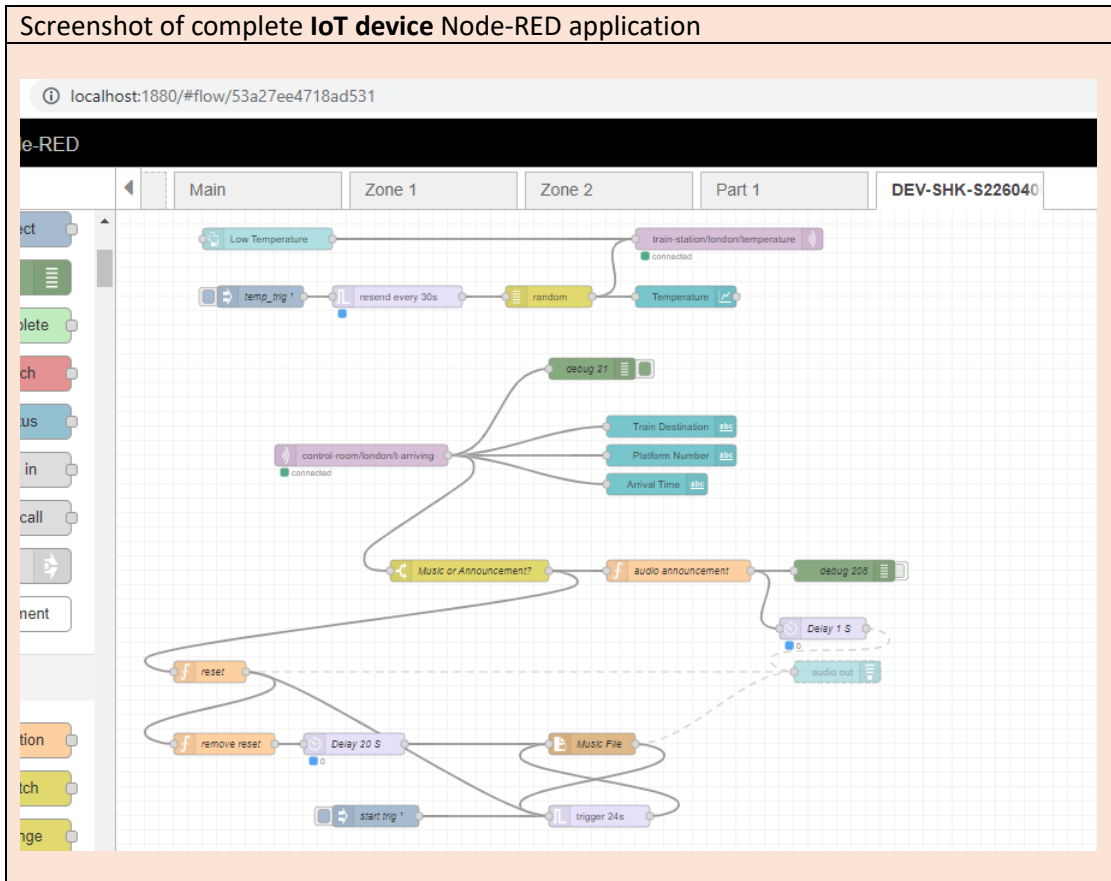
- When a CONTROL\_UPDATE message is received with "audio\_announcement" set to true, then an audio announcement should be created (via the Dashboard Audio Output).
  - The message content should be decided by you but should clearly convey all aspects of the train-related information within the message that has been received.
  - The message should interrupt the background music, by stopping it.
  - There should be a one-second delay, then the audio announcement should be delivered.
  - When the announcement is completed, then there should be a one-second delay, then the background music should start from the beginning.

Visual output – train announcements:

- Include a suitably named *local Node-RED dashboard tab on the IoT Device* (note: this is the second tab you need on the IoT device) with the following content:
  - A *title* that has the following text:
    - "Next Arrival"

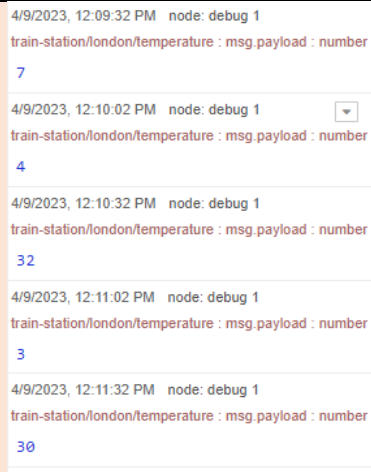
- Content from the most recent arrival message:
  - When a CONTROL\_UPDATE message is received then the following data must be displayed:
    - The train *destination* on one line, below the *title*.
    - The *platform number* on one line, below the *destination*.
    - The *arrival time* on one line, below the *platform number*.

### 3.4 Demonstration Application Deliverables



The topic used for publishing temperature data from IoT device to Cloud  
train-station/london/temperature

Provide a Node-RED debug example of temperature messages being received by the Cloud application (screenshot).

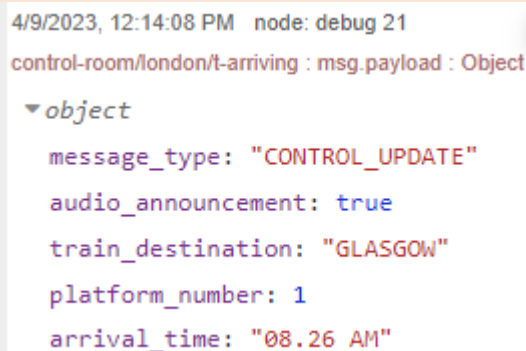


A screenshot of the Node-RED debug console showing five messages. Each message has a timestamp, 'node: debug 1', and a topic 'train-station/london/temperature : msg.payload : number'. The payloads are the numbers 7, 4, 32, 3, and 30.

Timestamp	Node	Topic	Payload
4/9/2023, 12:09:32 PM	node: debug 1	train-station/london/temperature : msg.payload : number	7
4/9/2023, 12:10:02 PM	node: debug 1	train-station/london/temperature : msg.payload : number	4
4/9/2023, 12:10:32 PM	node: debug 1	train-station/london/temperature : msg.payload : number	32
4/9/2023, 12:11:02 PM	node: debug 1	train-station/london/temperature : msg.payload : number	3
4/9/2023, 12:11:32 PM	node: debug 1	train-station/london/temperature : msg.payload : number	30

The topic used for publishing train arrival data from Cloud to IoT device  
control-room/london/t-arriving

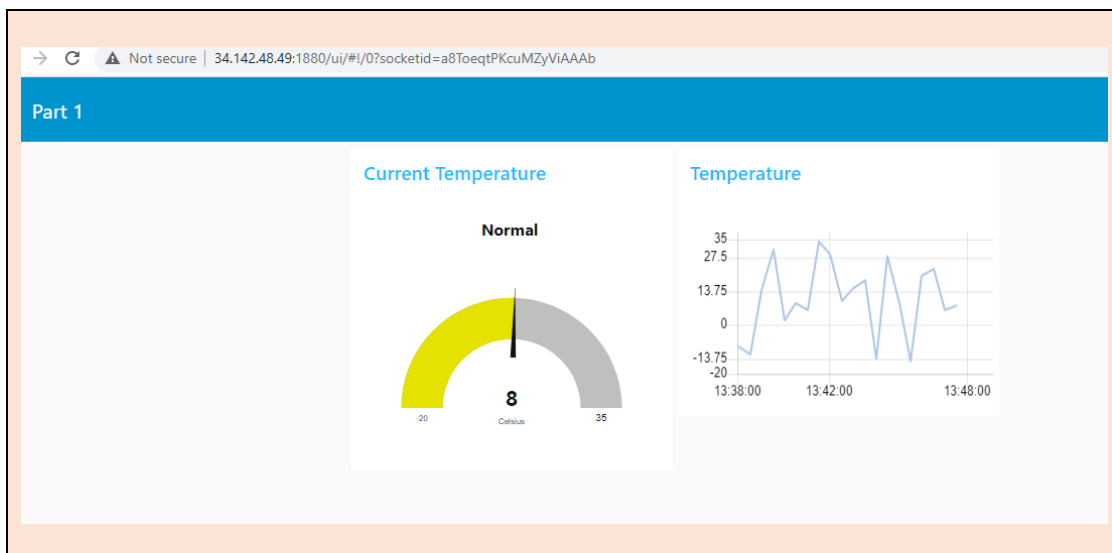
Provide a Node-RED debug example of train arrival messages being received by the IoT device application (screenshot).



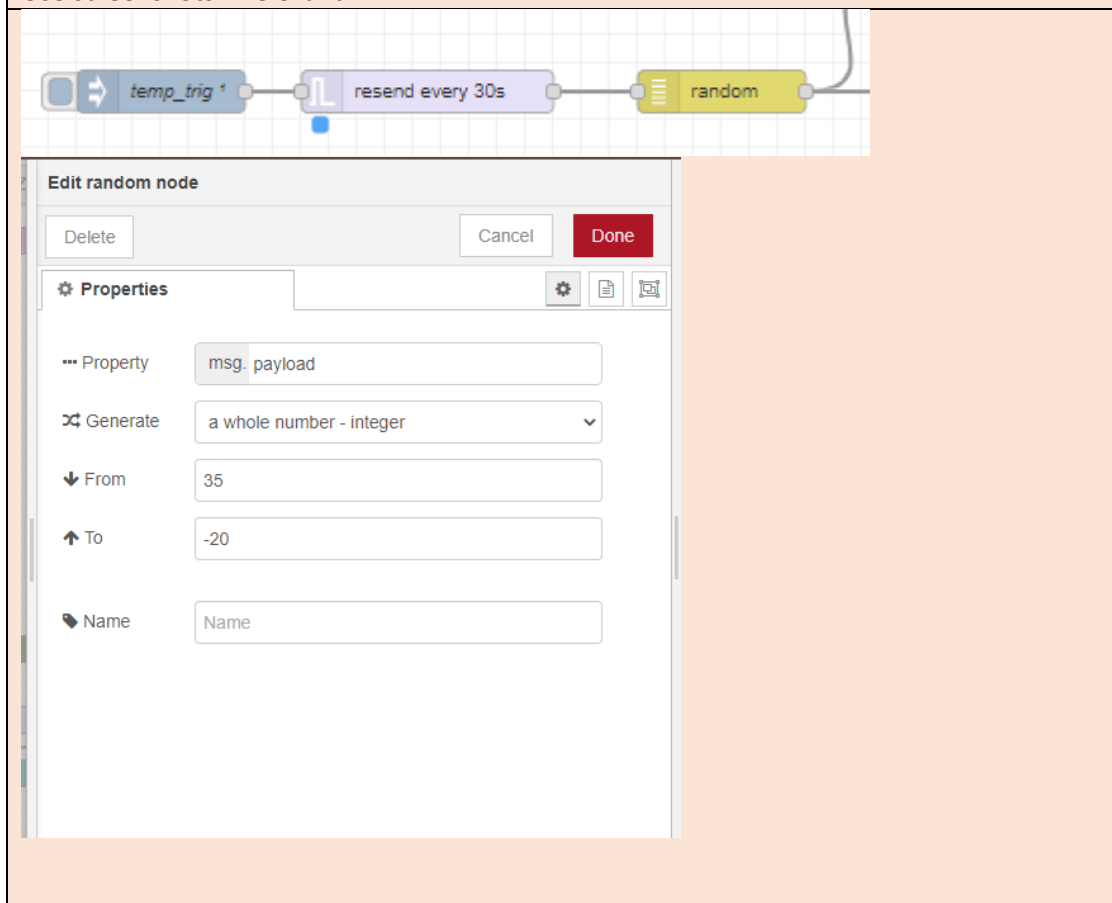
A screenshot of the Node-RED debug console showing a message with a timestamp of 4/9/2023, 12:14:08 PM, node: debug 21, and topic 'control-room/london/t-arriving : msg.payload : Object'. The payload is an object with the following properties: message\_type: 'CONTROL\_UPDATE', audio\_announcement: true, train\_destination: 'GLASGOW', platform\_number: 1, and arrival\_time: '08.26 AM'.

```
4/9/2023, 12:14:08 PM node: debug 21
control-room/london/t-arriving : msg.payload : Object
▼ object
  message_type: "CONTROL_UPDATE"
  audio_announcement: true
  train_destination: "GLASGOW"
  platform_number: 1
  arrival_time: "08.26 AM"
```

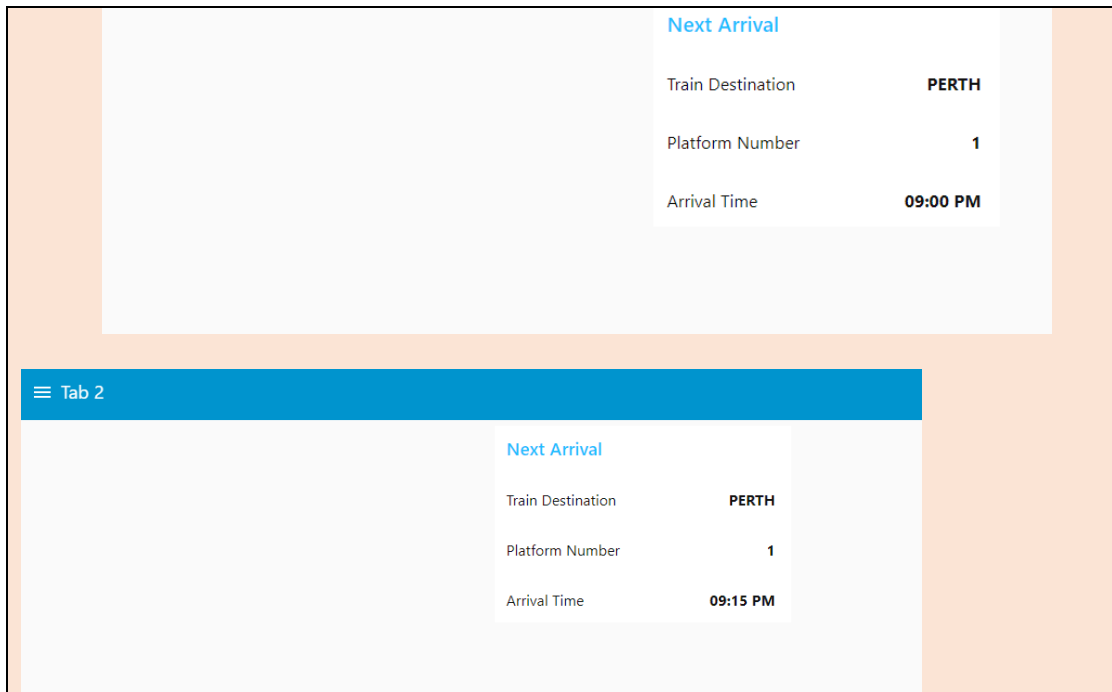
Screenshot of the complete Cloud Node-RED application dashboard (including temperature shown over a 10 minute period). **The screenshot MUST include the URL bar, showing the URL of your Cloud Node-RED instance.**



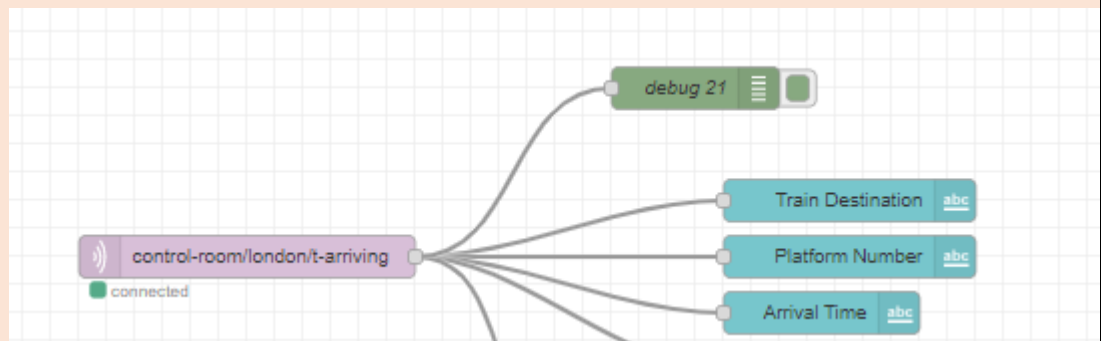
The code to generate the temperature on the IoT device. Provide the contents of any function node. Provide the configuration from any other node used (if any). Use screenshots if relevant.



Two Screenshots of the IoT device arrivals dashboard, from consecutive train arrival messages.



A description of your code implementation for delivering output to the IoT device train arrivals dashboard. This should start from the *train arrival messages* through to the output of information to the dashboard. **Use screenshots and narrative for the description.**



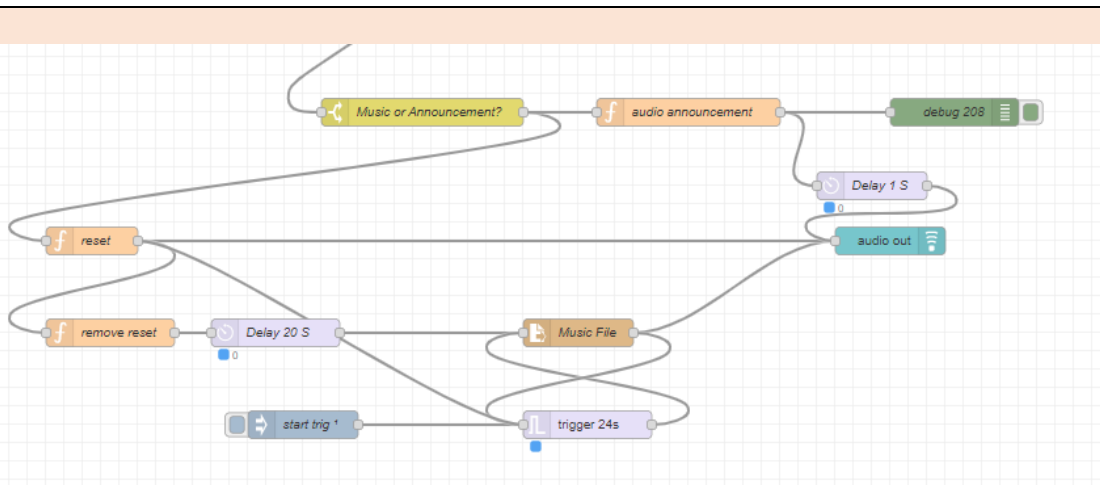
For train arrival dashboard, I used 3 ui-text output nodes. Output from the mqtt in node is json object which contain train\_destination, platform\_number, arrival\_time. So, inside the text output node, value format is set to corresponding key of the json object. (Ex : `{{msg.payload.platform_number}}`)

A description of your code implementation for the *two audio outputs* for the IoT device application. This should start from the *train arrival messages* and should cover *background music* and *train announcements*.

**Use screenshots and narrative for the description.**

You should include the following items, as well as any other description that justifies your design:

- A screenshot of the subset of nodes used for the audio section of your flow.
- The contents of any *function node* used.
- The configuration of any of the (non-function) nodes utilised.
- A debug output that shows the audio announcement text.



Background music is playing continuously until announcement is available. Start trig is trigger the trigger node and start then read it music file and sent it to the audio out. Trigger waiting time is set to length of background music(24 sec) so that it will retriggered after end of music. When arriving message received, the Music or Announcement switch node check whether it's has announcement. If so, function node create the announcement and reset node send reset message to audio out and trigger. Then announcement is sent to audio out with 1 sec delay. Again trigger music file after 20 sec(length of announcement) without reset message, So that music will start again.

Audio announcement function code:

```
// Construct the announcement message using the train-related
information
var announcement = "Attention all passengers, we would like to
notify you that the train scheduled to arrive at platform " +
msg.payload.platform_number + " is the " +
msg.payload.train_destination + " train It is estimated to arrive at
" + msg.payload.arrival_time + " Please make sure that you have
collected all of your belongings We appreciate your choice to travel
with us and thank you for your cooperation";
msg.payload = announcement;
msg.topic = "announcement";
return msg;
```

reset function code:

```
msg.reset = true
return msg;
```

remove reset function code:

```
delete msg.reset;
return msg;
```

debug output of announcement:

```
announcement : msg.payload : string[305]
```

```
"Attention all passengers, we would like to  
notify you that the train scheduled to arrive  
at platform 2 is the ABERDEEN train It is  
estimated to arrive at 02:41 PM Please make  
sure that you have collected all of your  
belongings We appreciate your choice to  
travel with us and thank you for your  
cooperation"
```

The complete export of your Node-RED **device** flow, Courier font, font size 8

```
[  
  {  
    "id": "53a27ee4718ad531",  
    "type": "tab",  
    "label": "DEV-SHK-S2260400",  
    "disabled": false,  
    "info": "",  
    "env": []  
  },  
  {  
    "id": "db67869d7fbcc03b",  
    "type": "ui_text",  
    "z": "53a27ee4718ad531",  
    "group": "0c4c08e0ce8ef40c",  
    "order": 1,  
    "width": 0,  
    "height": 0,  
    "name": "",  
    "label": "Train Destination",  
    "format": "{{msg.payload.train_destination}}",  
    "layout": "row-spread",  
    "className": "",  
    "x": 770,  
    "y": 300,  
    "wires": []  
  },  
  {  
    "id": "3d5183b49f93bfb7",  
    "type": "debug",  
    "z": "53a27ee4718ad531",  
    "name": "debug 21",  
    "active": true,  
    "tosidebar": true,  
    "console": false,  
    "tostatus": false,  
    "complete": "payload",  
    "targetType": "msg",  
    "statusVal": "",  
    "statusType": "auto",  
    "x": 660,  
    "y": 220,  
    "wires": []  
  },  
  {  
    "id": "3b4b95ec69987233",  
    "type": "ui_text",  
    "z": "53a27ee4718ad531",
```

```

        "group": "0c4c08e0ce8ef40c",
        "order": 2,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Platform Number",
        "format": "{{msg.payload.platform_number}}",
        "layout": "row-spread",
        "className": "",
        "x": 770,
        "y": 340,
        "wires": []
    },
    {
        "id": "d94e1f1996c6e648",
        "type": "ui_text",
        "z": "53a27ee4718ad531",
        "group": "0c4c08e0ce8ef40c",
        "order": 3,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Arrival Time",
        "format": "{{msg.payload.arrival_time}}",
        "layout": "row-spread",
        "className": "",
        "x": 750,
        "y": 380,
        "wires": []
    },
    {
        "id": "e4404045fd53496e",
        "type": "function",
        "z": "53a27ee4718ad531",
        "name": "audio announcement",
        "func": "// Construct the announcement message using the
train-related information\n    var announcement = \"Attention all
passengers, we would like to notify you that the train scheduled to
arrive at platform \" + msg.payload.platform_number + \" is the \"
+ msg.payload.train_destination + \" train It is estimated to
arrive at \" + msg.payload.arrival_time + \" Please make sure that
you have collected all of your belongings We appreciate your choice
to travel with us and thank you for your cooperation\";\n\n
msg.payload = announcement;\n    msg.topic = \"announcement\";\n
return msg;\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 780,
        "y": 500,
        "wires": [
            [
                "f0accbeb6c139838",
                "eb84412f8b650c5c"
            ]
        ]
    },
    {
        "id": "6d041a2984c6d2b0",
        "type": "file in",
        "z": "53a27ee4718ad531",

```



```

        "name": "Music File",
        "filename":
"C:\\\\Users\\\\Sahan\\\\Downloads\\\\station_audio_short.wav",
        "filenameType": "str",
        "format": "",
        "chunk": false,
        "sendError": false,
        "encoding": "none",
        "allProps": false,
        "x": 660,
        "y": 740,
        "wires": [
            [
                "f3a479b68b56e4de",
                "3f6f00af2d457b74"
            ]
        ]
    },
    {
        "id": "f3a479b68b56e4de",
        "type": "ui_audio",
        "z": "53a27ee4718ad531",
        "d": true,
        "name": "",
        "group": "0c4c08e0ce8ef40c",
        "voice": "",
        "always": true,
        "x": 1000,
        "y": 640,
        "wires": []
    },
    {
        "id": "4a334193066db11c",
        "type": "inject",
        "z": "53a27ee4718ad531",
        "name": "start trig",
        "props": [
            {
                "p": "payload"
            }
        ],
        "repeat": "",
        "crontab": "",
        "once": true,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 360,
        "y": 840,
        "wires": [
            [
                "3f6f00af2d457b74"
            ]
        ]
    },
    {
        "id": "3ba164f491a401f8",
        "type": "switch",
        "z": "53a27ee4718ad531",
        "name": "Music or Announcement?",
        "property": "payload.audio_announcement",
        "propertyType": "msg",

```

```

    "rules": [
      {
        "t": "true"
      }
    ],
    "checkall": "true",
    "repair": false,
    "outputs": 1,
    "x": 490,
    "y": 500,
    "wires": [
      [
        "9612a3e74e02d836",
        "e4404045fd53496e"
      ]
    ]
  },
  {
    "id": "9612a3e74e02d836",
    "type": "function",
    "z": "53a27ee4718ad531",
    "name": "reset",
    "func": "msg.reset = true\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 130,
    "y": 640,
    "wires": [
      [
        "f3a479b68b56e4de",
        "38ffaddb14bca652",
        "3f6f00af2d457b74"
      ]
    ]
  },
  {
    "id": "f0accbeb6c139838",
    "type": "delay",
    "z": "53a27ee4718ad531",
    "name": "Delay 1 S",
    "pauseType": "delayv",
    "timeout": "1",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "1",
    "randomLast": "5",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 980,
    "y": 580,
    "wires": [
      [
        "f3a479b68b56e4de"
      ]
    ]
  },
},

```

```

{
  "id": "3f6f00af2d457b74",
  "type": "trigger",
  "z": "53a27ee4718ad531",
  "name": "",
  "op1": "1",
  "op2": "0",
  "op1type": "str",
  "op2type": "num",
  "duration": "24",
  "extend": false,
  "overrideDelay": false,
  "units": "s",
  "reset": "",
  "bytopic": "all",
  "topic": "topic",
  "outputs": 1,
  "x": 670,
  "y": 840,
  "wires": [
    [
      "6d041a2984c6d2b0"
    ]
  ]
},
{
  "id": "b587e37afe721e94",
  "type": "delay",
  "z": "53a27ee4718ad531",
  "name": "Delay 20 S",
  "pauseType": "delayv",
  "timeout": "20",
  "timeoutUnits": "seconds",
  "rate": "1",
  "nbRateUnits": "1",
  "rateUnits": "second",
  "randomFirst": "1",
  "randomLast": "5",
  "randomUnits": "seconds",
  "drop": false,
  "allowrate": false,
  "outputs": 1,
  "x": 330,
  "y": 740,
  "wires": [
    [
      "6d041a2984c6d2b0"
    ]
  ]
},
{
  "id": "38ffadddb14bca652",
  "type": "function",
  "z": "53a27ee4718ad531",
  "name": "remove reset",
  "func": "delete msg.reset;\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "initialize": "",
  "finalize": "",
  "libs": [],
  "x": 150,
  "y": 740,

```

```

        "wires": [
            [
                "b587e37afe721e94"
            ]
        ]
    },
    {
        "id": "f0b3b92ab4c242d8",
        "type": "mqtt out",
        "z": "53a27ee4718ad531",
        "name": "",
        "topic": "train-station/london/temperature",
        "qos": "",
        "retain": "",
        "respTopic": "",
        "contentType": "",
        "userProps": "",
        "correl": "",
        "expiry": "",
        "broker": "c8fb3fa6c32e1abd",
        "x": 850,
        "y": 40,
        "wires": []
    },
    {
        "id": "5137956e0f57199c",
        "type": "inject",
        "z": "53a27ee4718ad531",
        "name": "temp_trig",
        "props": [
            {
                "p": "payload"
            }
        ],
        "repeat": "",
        "crontab": "",
        "once": true,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 200,
        "y": 120,
        "wires": [
            [
                "6f3125ebd893521c"
            ]
        ]
    },
    {
        "id": "6f3125ebd893521c",
        "type": "trigger",
        "z": "53a27ee4718ad531",
        "name": "",
        "op1": "1",
        "op2": "0",
        "op1type": "str",
        "op2type": "str",
        "duration": "-30",
        "extend": false,
        "overrideDelay": false,
        "units": "s",
        "reset": "",
    }

```

```

        "bytopic": "all",
        "topic": "topic",
        "outputs": 1,
        "x": 390,
        "y": 120,
        "wires": [
            [
                "3174f3a4a8a330c7"
            ]
        ]
    },
    {
        "id": "3174f3a4a8a330c7",
        "type": "random",
        "z": "53a27ee4718ad531",
        "name": "",
        "low": "35",
        "high": "-20",
        "inte": "true",
        "property": "payload",
        "x": 600,
        "y": 120,
        "wires": [
            [
                "f0b3b92ab4c242d8",
                "06359a0cfc15945e"
            ]
        ]
    },
    {
        "id": "cac78d881c80dd68",
        "type": "mqtt in",
        "z": "53a27ee4718ad531",
        "name": "",
        "topic": "control-room/london/t-arriving",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "c8fb3fa6c32e1abd",
        "nl": false,
        "rap": true,
        "rh": 0,
        "inputs": 0,
        "x": 340,
        "y": 340,
        "wires": [
            [
                "db67869d7fbcc03b",
                "3b4b95ec69987233",
                "d94e1f1996c6e648",
                "3ba164f491a401f8",
                "3d5183b49f93bfb7"
            ]
        ]
    },
    {
        "id": "49c79c6c9e4fe1d2",
        "type": "ui_button",
        "z": "53a27ee4718ad531",
        "name": "",
        "group": "0430f910110b83ca",
        "order": 0,
        "width": 0,
        "height": 0,
    }

```

```

    "passthru": false,
    "label": "Low Temperature",
    "tooltip": "",
    "color": "",
    "bgcolor": "",
    "className": "",
    "icon": "",
    "payload": "-15",
    "payloadType": "num",
    "topic": "topic",
    "topicType": "msg",
    "x": 210,
    "y": 40,
    "wires": [
      [
        "f0b3b92ab4c242d8"
      ]
    ]
  },
  {
    "id": "06359a0cfc15945e",
    "type": "ui_chart",
    "z": "53a27ee4718ad531",
    "name": "",
    "group": "0430f910110b83ca",
    "order": 1,
    "width": 0,
    "height": 0,
    "label": "Temperature",
    "chartType": "line",
    "legend": "false",
    "xformat": "HH:mm:ss",
    "interpolate": "linear",
    "nodata": "",
    "dot": false,
    "ymin": "",
    "ymax": "",
    "removeOlder": 1,
    "removeOlderPoints": "",
    "removeOlderUnit": "3600",
    "cutout": 0,
    "useOneColor": false,
    "useUTC": false,
    "colors": [
      "#1f77b4",
      "#aec7e8",
      "#ff7f0e",
      "#2ca02c",
      "#98df8a",
      "#d62728",
      "#ff9896",
      "#9467bd",
      "#c5b0d5"
    ],
    "outputs": 1,
    "useDifferentColor": false,
    "className": "",
    "x": 790,
    "y": 120,
    "wires": [
      []
    ]
  },
},

```

```

{
  "id": "eb84412f8b650c5c",
  "type": "debug",
  "z": "53a27ee4718ad531",
  "name": "debug 208",
  "active": false,
  "tosidebar": true,
  "console": false,
  "tostatus": false,
  "complete": "false",
  "statusVal": "",
  "statusType": "auto",
  "x": 1010,
  "y": 500,
  "wires": []
},
{
  "id": "0c4c08e0ce8ef40c",
  "type": "ui_group",
  "name": "Next Arrival",
  "tab": "7f9810e2deb090d4",
  "order": 2,
  "disp": true,
  "width": "6",
  "collapse": false,
  "className": ""
},
{
  "id": "c8fb3fa6c32e1abd",
  "type": "mqtt-broker",
  "name": "",
  "broker": "broker.hivemq.com",
  "port": "1883",
  "clientid": "",
  "autoConnect": true,
  "usetls": false,
  "protocolVersion": "4",
  "keepalive": "60",
  "cleansession": true,
  "birthTopic": "",
  "birthQos": "0",
  "birthPayload": "",
  "birthMsg": {},
  "closeTopic": "",
  "closeQos": "0",
  "closePayload": "",
  "closeMsg": {},
  "willTopic": "",
  "willQos": "0",
  "willPayload": "",
  "willMsg": {},
  "userProps": "",
  "sessionExpiry": ""
},
{
  "id": "0430f910110b83ca",
  "type": "ui_group",
  "name": "Temperature",
  "tab": "a920e6eb2a34f5af",
  "order": 1,
  "disp": true,
  "width": "6",
  "collapse": false,

```

```

        "className": ""
    },
    {
        "id": "7f9810e2deb090d4",
        "type": "ui_tab",
        "name": "Tab 2",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    },
    {
        "id": "a920e6eb2a34f5af",
        "type": "ui_tab",
        "name": "Tab 1",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]

```

The complete export of your **Cloud** Node-RED flow, Courier font, font size 8

```

[
  {
    "id": "976734a0b78ce51b",
    "type": "tab",
    "label": "CLOUD-SHK-S2260400",
    "disabled": false,
    "info": "",
    "env": []
  },
  {
    "id": "05c8636f12e29422",
    "type": "mqtt in",
    "z": "976734a0b78ce51b",
    "name": "",
    "topic": "train-station/london/temperature",
    "qos": "2",
    "datatype": "auto-detect",
    "broker": "c8fb3fa6c32e1abd",
    "nl": false,
    "rap": true,
    "rh": 0,
    "inputs": 0,
    "x": 170,
    "y": 340,
    "wires": [
      [
        "02ce309e6692cd8d",
        "d087d462e9190a31",
        "af85657fe51b9389",
        "140f6ca7db7c5cb7"
      ]
    ]
  },
  {
    "id": "e50af671577d18df",
    "type": "inject",
    "z": "976734a0b78ce51b",
    "name": "clear",

```



```

      "props": [
        {
          "p": "payload"
        }
      ],
      "repeat": "",
      "crontab": "",
      "once": false,
      "onceDelay": 0.1,
      "topic": "",
      "payload": "[]",
      "payloadType": "json",
      "x": 210,
      "y": 460,
      "wires": [
        [
          "02ce309e6692cd8d"
        ]
      ]
    },
    {
      "id": "af85657fe51b9389",
      "type": "switch",
      "z": "976734a0b78ce51b",
      "name": "Condition",
      "property": "payload",
      "propertyType": "msg",
      "rules": [
        {
          "t": "gt",
          "v": "2",
          "vt": "num"
        },
        {
          "t": "lte",
          "v": "2",
          "vt": "num"
        }
      ],
      "checkall": "true",
      "repair": false,
      "outputs": 2,
      "x": 460,
      "y": 340,
      "wires": [
        [
          "e1509aeac85199d7"
        ],
        [
          "3c72e8116a336909"
        ]
      ]
    },
    {
      "id": "e1509aeac85199d7",
      "type": "change",
      "z": "976734a0b78ce51b",
      "name": "",
      "rules": [
        {
          "t": "set",
          "p": "payload",
          "pt": "msg",

```

```

        "to": "Normal",
        "tot": "str"
    }
],
"action": "",
"property": "",
"from": "",
"to": "",
"reg": false,
"x": 660,
"y": 300,
"wires": [
    [
        "7bfd95af1955fbda"
    ]
]
},
{
    "id": "3c72e8116a336909",
    "type": "change",
    "z": "976734a0b78ce51b",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "payload",
            "pt": "msg",
            "to": "Low Temperature - possible ice hazard",
            "tot": "str"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 660,
    "y": 380,
    "wires": [
        [
            "7bfd95af1955fbda"
        ]
    ]
},
{
    "id": "7bfd95af1955fbda",
    "type": "ui_text",
    "z": "976734a0b78ce51b",
    "group": "de7d4276054a38b5",
    "order": 1,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "",
    "format": "{{msg.payload}}",
    "layout": "col-center",
    "className": "",
    "x": 870,
    "y": 340,
    "wires": []
},
{
    "id": "d087d462e9190a31",

```

```

    "type": "ui_gauge",
    "z": "976734a0b78ce51b",
    "name": "",
    "group": "de7d4276054a38b5",
    "order": 2,
    "width": 0,
    "height": 0,
    "gtype": "gage",
    "title": "",
    "label": "Celsius",
    "format": "{{value}}",
    "min": "-20",
    "max": "35",
    "colors": [
        "#00b500",
        "#e6e600",
        "#ca3838"
    ],
    "seg1": "",
    "seg2": "",
    "diff": false,
    "className": "",
    "x": 450,
    "y": 400,
    "wires": []
  },
  {
    "id": "02ce309e6692cd8d",
    "type": "ui_chart",
    "z": "976734a0b78ce51b",
    "name": "",
    "group": "f52df5059b8a6c5e",
    "order": 1,
    "width": 0,
    "height": 0,
    "label": "",
    "chartType": "line",
    "legend": "false",
    "xformat": "HH:mm:ss",
    "interpolate": "linear",
    "nodata": "",
    "dot": false,
    "ymin": "-20",
    "ymax": "35",
    "removeOlder": "10",
    "removeOlderPoints": "",
    "removeOlderUnit": "60",
    "cutout": 0,
    "useOneColor": false,
    "useUTC": false,
    "colors": [
        "#1f77b4",
        "#aec7e8",
        "#ff7f0e",
        "#2ca02c",
        "#98df8a",
        "#d62728",
        "#ff9896",
        "#9467bd",
        "#c5b0d5"
    ],
    "outputs": 1,
    "useDifferentColor": false,

```

```

        "className": "",
        "x": 450,
        "y": 460,
        "wires": [
            []
        ]
    },
    {
        "id": "727990445b5d0e85",
        "type": "function",
        "z": "976734a0b78ce51b",
        "name": "Train Message Simulator ",
        "func": "\n\nvar messages = [\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": false,\n        \"train_destination\": \"GLASGOW\",\n        \"platform_number\": 1,\n        \"arrival_time\": \"08:24 AM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": false,\n        \"train_destination\": \"GLASGOW\",\n        \"platform_number\": 1,\n        \"arrival_time\": \"08:26 AM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": true,\n        \"train_destination\": \"GLASGOW\",\n        \"platform_number\": 1,\n        \"arrival_time\": \"08:26 AM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": false,\n        \"train_destination\": \"EDINBURGH\",\n        \"platform_number\": 2,\n        \"arrival_time\": \"08:30 AM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": false,\n        \"train_destination\": \"EDINBURGH\",\n        \"platform_number\": 2,\n        \"arrival_time\": \"08:30 AM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": true,\n        \"train_destination\": \"EDINBURGH\",\n        \"platform_number\": 2,\n        \"arrival_time\": \"08:31 AM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": false,\n        \"train_destination\": \"ABERDEEN\",\n        \"platform_number\": 2,\n        \"arrival_time\": \"08:40 AM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": false,\n        \"train_destination\": \"ABERDEEN\",\n        \"platform_number\": 2,\n        \"arrival_time\": \"08:41 AM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": true,\n        \"train_destination\": \"ABERDEEN\",\n        \"platform_number\": 2,\n        \"arrival_time\": \"02:41 PM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": false,\n        \"train_destination\": \"PERTH\",\n        \"platform_number\": 1,\n        \"arrival_time\": \"09:00 PM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": false,\n        \"train_destination\": \"PERTH\",\n        \"platform_number\": 1,\n        \"arrival_time\": \"09:15 PM\",\n    },\n    {\n        \"message_type\": \"CONTROL_UPDATE\",\n        \"audio_announcement\": true,\n        \"train_destination\": \"PERTH\",\n        \"platform_number\": 1,\n        \"arrival_time\": \"09:25 PM\",\n    }\n]\n\nvar counter = context.get(\"messageCounter\") || 0 ;\nvar arraySize = messages.length - 1;\n\nmsg.payload = messages[counter++];\n\nif

```

```

(counter > arraySize){\n    counter = 0;\n
}\ncontext.set(\n"messageCounter\n", counter)\n\nreturn msg;\n",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 470,
    "y": 140,
    "wires": [
        [
            "61629d03effa7a4f",
            "21ce56b7bf3adbd0"
        ]
    ]
},
{
    "id": "61629d03effa7a4f",
    "type": "debug",
    "z": "976734a0b78ce51b",
    "name": "",
    "active": false,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "statusVal": "",
    "statusType": "auto",
    "x": 750,
    "y": 80,
    "wires": []
},
{
    "id": "c2f55f8e9aa153fc",
    "type": "inject",
    "z": "976734a0b78ce51b",
    "name": "",
    "props": [
        {
            "p": "payload"
        },
        {
            "p": "topic",
            "vt": "str"
        }
    ],
    "repeat": "60",
    "crontab": "",
    "once": true,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
    "x": 230,
    "y": 140,
    "wires": [
        [
            "727990445b5d0e85"
        ]
    ]
},
{
    "id": "21ce56b7bf3adbd0",

```

```

        "type": "mqtt out",
        "z": "976734a0b78ce51b",
        "name": "",
        "topic": "control-room/london/t-arriving",
        "qos": "",
        "retain": "",
        "respTopic": "",
        "contentType": "",
        "userProps": "",
        "correl": "",
        "expiry": "",
        "broker": "c8fb3fa6c32e1abd",
        "x": 800,
        "y": 140,
        "wires": []
    },
    {
        "id": "140f6ca7db7c5cb7",
        "type": "debug",
        "z": "976734a0b78ce51b",
        "name": "debug 1",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "false",
        "statusVal": "",
        "statusType": "auto",
        "x": 460,
        "y": 280,
        "wires": []
    },
    {
        "id": "c8fb3fa6c32e1abd",
        "type": "mqtt-broker",
        "name": "",
        "broker": "broker.hivemq.com",
        "port": "1883",
        "clientId": "",
        "autoConnect": true,
        "usetls": false,
        "protocolVersion": "4",
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "birthMsg": {},
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "closeMsg": {},
        "willTopic": "",
        "willQos": "0",
        "willPayload": "",
        "willMsg": {},
        "userProps": "",
        "sessionExpiry": ""
    },
    {
        "id": "de7d4276054a38b5",
        "type": "ui_group",
        "name": "Current Temperature",

```

```
    "tab": "1b5666d2179fd364",
    "order": 1,
    "disp": true,
    "width": "6",
    "collapse": false,
    "className": ""
  },
  {
    "id": "f52df5059b8a6c5e",
    "type": "ui_group",
    "name": "Temperature",
    "tab": "1b5666d2179fd364",
    "order": 2,
    "disp": true,
    "width": "6",
    "collapse": false,
    "className": ""
  },
  {
    "id": "1b5666d2179fd364",
    "type": "ui_tab",
    "name": "Part 1",
    "icon": "dashboard",
    "disabled": false,
    "hidden": false
  }
]
```