2023-24 - Internet of Things (MMI126818)

Coursework 2, 1st Diet.

[1 Introduction 3](#_Toc162476615)

[2 Q1: Node-RED Dashboard 4](#_Toc162476616)

[3 Q2: Google Maps on Node-RED Cloud Deployment 5](#_Toc162476617)

[4 Q3 Using a database to store and analyse the vehicle data (on Cloud) 5](#_Toc162476618)

[4.1 Storing data in the Firestore database 6](#_Toc162476619)

[4.2 Analysing the data 7](#_Toc162476620)

|  |  |
| --- | --- |
| Surname |  |
| First Name |  |
| Student ID |  |
| Caledonian Email Address |  |
| Date |  |

|  |  |  |
| --- | --- | --- |
| I declare that this coursework is an original piece of work that has been created only by me. The work is not based on previous coursework that has been submitted as part of another module or submitted to another academic programme of study.   |  | | --- | | **Enter your name below as confirmation of this declaration.** | | Name: YOUR NAME | |

**General Notes**

* This document delivers the following:
  + The CW2 specification
  + The details of each response that is expected for each part of the CW.
* 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 or JSON, 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>
  + <http://www.sussex.ac.uk/ei/internal/forstudents/engineeringdesign/studyguides/techreportwriting>
  + <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>

**CW2 Notes**

1. **Please note that all your browser-based screenshots must show the full-width URL bar of the browser window that contains the flow. Please use the Chrome browser. You must not run the Node-RED application in browser ‘incognito’ (anonymous) mode. The screenshot of any browser window that contains Cloud components of the coursework should show the icon for your Google Login.**
2. For any diagram or screenshot, you must provide a *Figure number* and *Caption* using the MS Word facilities.

# Introduction

The coursework uses simulated data from a moving delivery vehicle making several deliveries (the deliveries repeat in a loop within the simulation). The vehicle is refrigerated, and the current temperature (degrees C) is included in the vehicle data. Each delivery will have its own ‘delivery\_id’. The vehicle data is being published via MQTT, with the following configuration:

MQTT Broker: broker.hivemq.com

Port: 1883

Subscription Topic 1: pb54327xyCW-2-2024/dev-PC0099/telemetry

Publishing: **No publishing is required. You must only subscribe.**

Received data format example:

A screenshot of a computer code

Description automatically generated

Subscription Topic 2: pb54327xyCW-2-2024/dev-PC0099/state

Vehicle ‘Fault’ messages are published on the ‘state’ topic. There is a range of possible vehicle faults. For testing purposes these faults arrive approximately every minute.  
Publishing: **No publishing is required. You must only subscribe.**  
Received data format example:

A screenshot of a computer code

Description automatically generated

# Q1: Node-RED Dashboard

Create a new flow on your **Google Cloud deployment** that delivers a Node-RED dashboard. It must contain the following elements:

* Code to receive the messages, prepare the data and send relevant data to the dashboard.
* A chart showing the current vehicle refrigeration temperature.
  + This must be configured to show one-hour of data.
  + It should be configured with an appropriate temperature range on the chart Y-axis. You will need to explore the temperature range that is being published.
  + The chart should be clearly labelled, including temperature units.
* A ‘text’ output showing the current delivery\_id.
* It must be within a **new** Dashboard tab, not any existing one.
  + You should provide an appropriate tab name.
  + The ‘group’ that is used on the tab should have an appropriate name.

1. Provide a screenshot of the complete browser page for the dashboard tab, including at least 30 minutes of temperature data.

|  |
| --- |
| Screenshot of dashboard below |

1. Provide a screenshot copy of the complete Node-RED flow that implements the specification above.

|  |
| --- |
| Screenshot of flow below |

1. Provide a screenshot of the configuration of the chart node used in the flow.

|  |
| --- |
| Screenshot of chart configuration below |

1. Provide a copy of the code used for the Node-RED *function* that creates the correct data to be sent to the chart node.

Code (in text format)

|  |
| --- |
|  |

# Q2: Google Maps on Node-RED Cloud Deployment

Create a new flow that will connect to the Google Maps flow that was deployed to Cloud Node-RED within the relevant Laboratory session. **Your new flow should use a ‘link’ node** to connect to the Google Maps flow.   
  
The new flow should implement the following:

* Receives vehicle data via MQTT as described above.
* Places a marker on the map for the vehicle location.
* The marker has an InfoWindow that delivers two data-items with suitable labels:
  + Temperature
  + Delivery\_id

1. Provide a screenshot that shows the vehicle marker on the map with an active InfoWIndow.

|  |
| --- |
| Screenshot of map with marker and InfoWindow below |

1. Provide a copy of the code used for the Node-RED *function* that creates the object to be sent to the Google Maps flow.

Code (in text format)

|  |
| --- |
|  |

1. Provide one example of an object that is sent to the Google Maps flow. Use a screenshot from the ‘Debug’ panel. You must fully expand the object and child-objects in the debug panel.

|  |
| --- |
| Screenshot from debug panel below |

# Q3 Using a database to store and analyse the vehicle data (on Cloud)

You must create a new flow on your Node-RED Cloud VM deployment and implement code according to the requirements below. You must store an extract of the data from every vehicle *telemetry* and *fault*message into documents in a Google Firestore database. The extraction should be implemented with suitable Node-RED *function* nodes, one function for *Telemetry* messages and another for *State* messages. The following configurations are required. Note that for the stored documents, the keys are specified below, and you must extract the values from the incoming MQTT messages.

In Question 4.2 you will be analysing the data that is stored in the database.

## Storing data in the Firestore database

*Telemetry* messages :

* Firestore collection name: ‘vehicle\_data’.
* Stored document fields (each has a *key* and a *value*):
  + The type of message. The key should be ‘msgType’.
  + The vehicle ID. The key should be ‘vehicle\_id’.
  + The temperature. The key should be vehicle\_storage\_temperature’.
  + The time in UTC Milliseconds. The key should be: ‘date\_time\_UTC\_milliseconds’. You must generate the value for this with suitable code.

*State* messages:

* Firestore collection name: ‘vehicle\_data’.
* Stored document fields:
  + The type of message. The key should be ‘fault’.
  + The vehicle ID. The key should be ‘vehicle\_id’.
  + The time in UTC Milliseconds. The key should be: “date\_time\_UTC\_milliseconds”.
  + The fault type: The key should be ‘vehicleFault’.

1. Provide a screenshot of the entire Node-RED flow. This includes code for storing documents in the database and code for extracting and analysing the data stored in the database.

|  |
| --- |
| Screenshot of flow below |

1. Provide two screenshots from your Firebase Console that illustrate documents stored in the Firestore database, one document based on *telemetry* messages and the other based on *state* messages.

|  |
| --- |
| Screenshot 1 (document based upon telemetry messages) below |

|  |
| --- |
| Screenshot 2 (document based upon state messages) below |

1. Provide a copy of the code used for the two Node-RED function nodes that you have created to organise the data to be stored in the Firestore database collection

Code (**in text format**) for storing Telemetry messages

|  |
| --- |
|  |

Code (**in text format**) for storing State messages

|  |
| --- |
|  |

## Analysing the data

The following questions are based upon (i) The design of Node-RED code to query the database, and (ii) The design of code to undertake further analysis of the returned query data.

Examine the following code:



The function code is:

msg.firestore = {

collection: 'vehicle-2024',

query: [ {where: ["msgType", "==", "fault"]},

{orderBy:["date\_time\_UTC\_milliseconds", "desc"]},

{limit:3}

]

}

return msg;

1. Provide a brief *non-technical description* of what the query accomplishes.

|  |
| --- |
|  |

1. Provide a *detailed description* of the design of *msg.firestore* object that forms the query.

|  |
| --- |
|  |

The following questions require you to develop and test code in a new Cloud Node-RED flow.

1. Implement code for a function that sets up a query that should deliver the most recent 10 documents that contain the vehicle temperature readings.

Code:

|  |
| --- |
|  |

1. Provide a screenshot including the Node-RED debug panel that demonstrates that you have retrieved up to 10 documents based on your function code for Part (vi). You should expand one of the retrieved document objects to demonstrate that this is the correct document type.

|  |
| --- |
| Screenshot below |

1. You are required to implement part of the system that delivers the highest vehicle temperature over the last 30 minutes. As part of the solution, you must use the Node-RED ‘statistics’ node that was employed in one of the laboratory exercises relating to database queries. This node comes from the module: ‘node-red-contrib-statistics’.

You must deliver the following:

* + - A screenshot of the flow.
    - The code for the query that retrieves 30 minutes of data.
    - The code to set-up the data/configuration for the Node-RED ‘statistics’ node.
    - Screenshot of a debug output that shows the highest temperature value.

|  |
| --- |
| Flow Screenshot below |

The query code for 30 minutes of temperature data.

|  |
| --- |
|  |

The code to set-up the data/configuration for the statistics node.

|  |
| --- |
|  |

|  |
| --- |
| Debug output screenshot of highest temperature value below |