Big Data Platforms for Internet of Things Applications

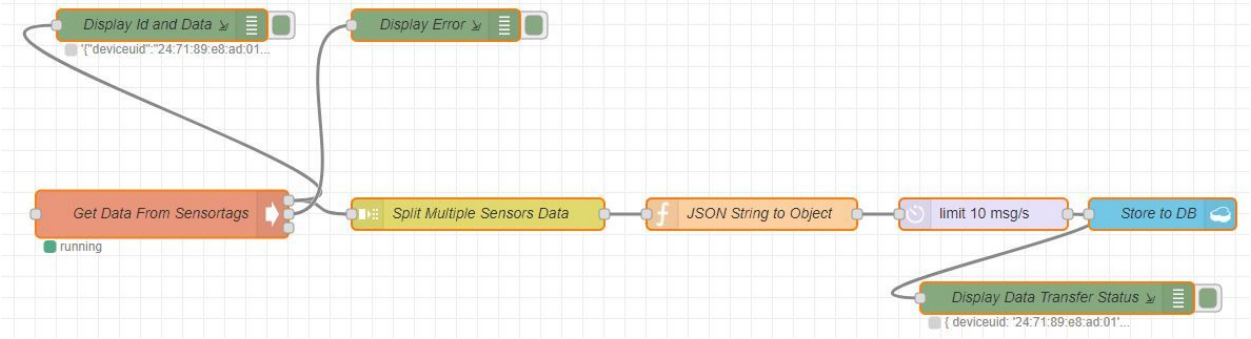
Project Purpose:

An Internet of Things application, a smart building for example, contains thousands of sensors and cameras. IoT data could be stored and processed in the cloud. The purpose of this project is to explore the architecture design of collecting and transferring data from sensors and cameras to IBM Cloud via Raspberry Pie. Also, it evaluates the optimal performance (frequency).

Architecture Overview:

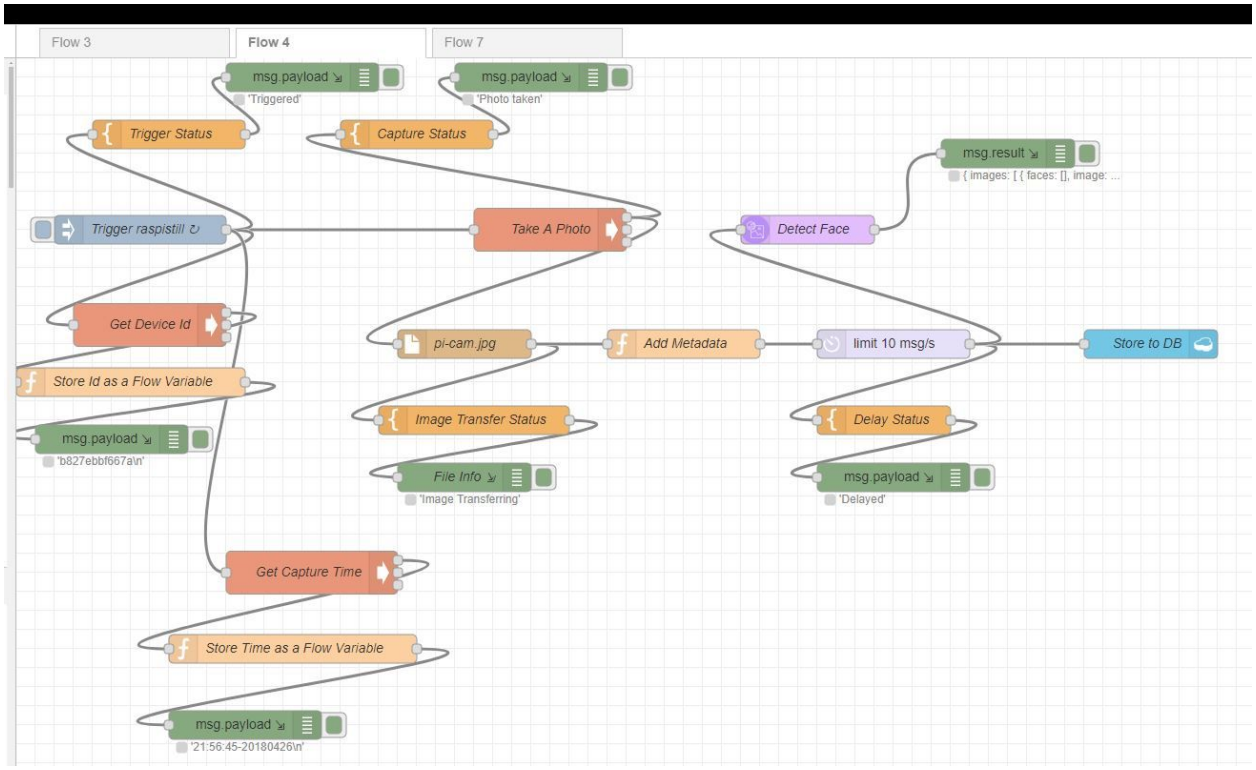
The data could be divided into two categories: sensor and image data.

- Sensor Data:
    - Once the flow starts, Node-Red runs sensortagcollector.py which scans SensorTags, reads temperature data, and directs data into a Node-Red flow.
    - Two function nodes split and convert the data (from Step 1) to a JSON object.
    - The converted object data goes through a delay node which limits 10 messages per second for writing data to Cloudant. As a free Cloudant Database plan maximumly allows 10 writes per second. If writes exceed the number, error raised. Untransferred data would be put in a queue.
    - The data then will be sent to and stored at Cloudant.



- Image Data:
    - Set a repeated time interval (e.g. every 5 seconds) in Node-Red Inject node for triggering the whole flow.
    - Once the flow triggered, it first gets metadata (e.g. get device id by issuing `service iot getdeviceid`) and adds the metadata to a flow variable so it could be used later.
    - Then it captures an image (e.g. raspistill -op 10 -t 10 -o /home/pi/camera/pi-cam.jpg -q 100). The image file would be stored in /home/pi/camera/pi-cam.jpg.

- A Node-Red File node reads the image file and sends data to next node.
- Embed metadata (e.g. Device Id) in the data.
- The image data goes through a delay node which limits 10 messages per second for writing data to Cloudant. As a free Cloudant Database plan maximumly allows 10 writes per second. If writes exceed the number, error raised. Untransferred messages would be put in a queue.
- The data then will be sent to and stored at Cloudant.
- In addition, the data goes through a Watson Visual Recognition node, which detects face. If face detected, the msg.result would display the age, face location, and gender with probability. If not, an empty array returned.



Reproduction Procedures:

Sensor Data:

- A Cloudant Database instance and database have been created.
- Install node-red-node-daemon into Node-Red (on Raspberry Pi).
- Install node-red-bluemix-nodes into Node-Red (on Raspberry Pi).
- Find the attached Sensor_Workflow.json and import the json content into a Node-Red flow.
- Copy and paste sensortagcollector.py to /home/pi.
- Find your SensorTags' uid and make a meaningful alias. For example, a SensorTag uid looks like A0:E6:F8:B6:A2:01 and a meaningful alias looks like TEL1001a.

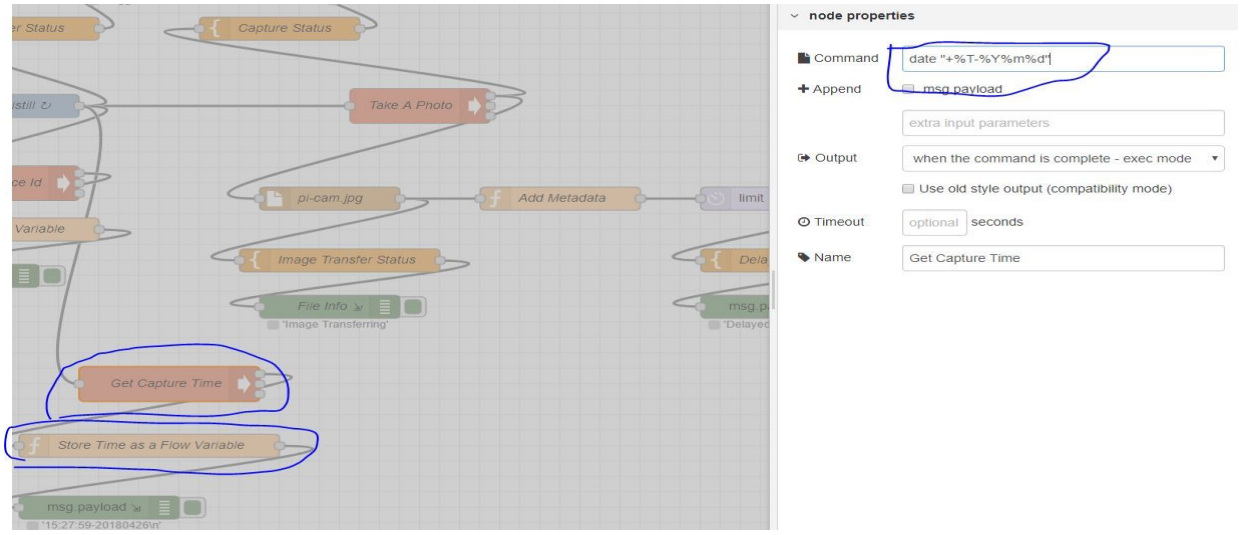
- In the first node, provide parameters.
    - For example, python -u /home/pi/sensortagcollector.py -s IRtemperature -d 24:71:89:e8:ad:01=TEL1001 54:6c:0e:52:cc:88=TEL1002
        - where -u for script location, -s for getting temperature data every 5 seconds (-f for 1 second, -m for 3 seconds), -d and = for SensorTags' uid and a meaning alias.
        - -d 24:71:89:e8:ad:01=TEL1001 54:6c:0e:52:cc:88=TEL1002 is omittable. If a user doesn't provide -d parameters, the device name would be ST-1, ST-2 and so on.
    - The retrieved temperature data would be put into an array (e.g. [25.3125,16.875]). The first element means ambient temperature, the second element means IR temperature.
- In cloudant node, replace the server credentials and database name to yours.
- Finally click Deploy button to start.

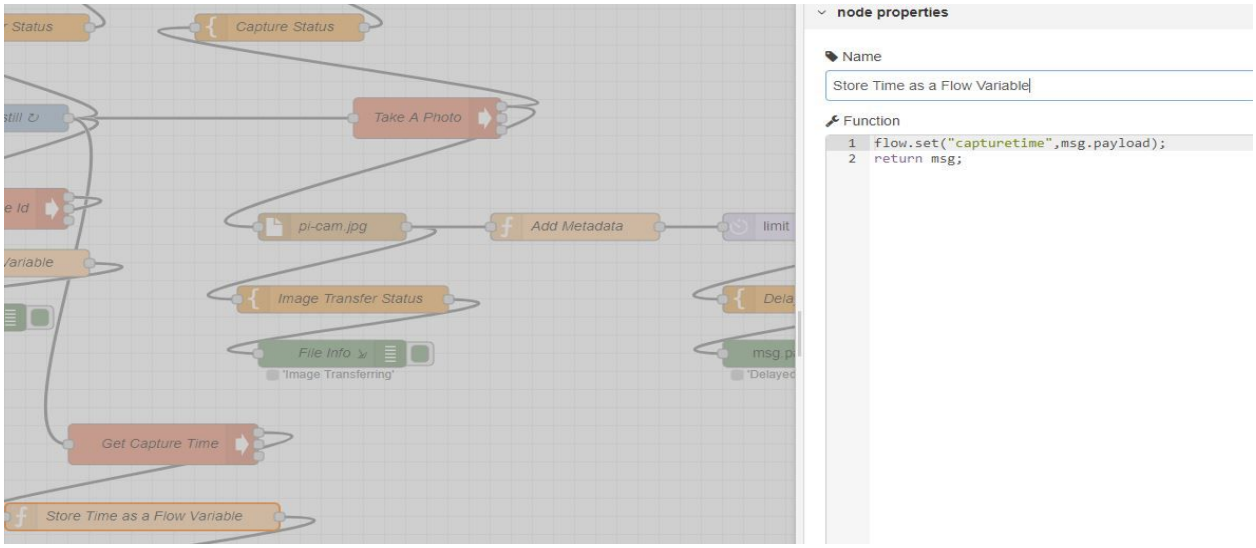
Image Data:

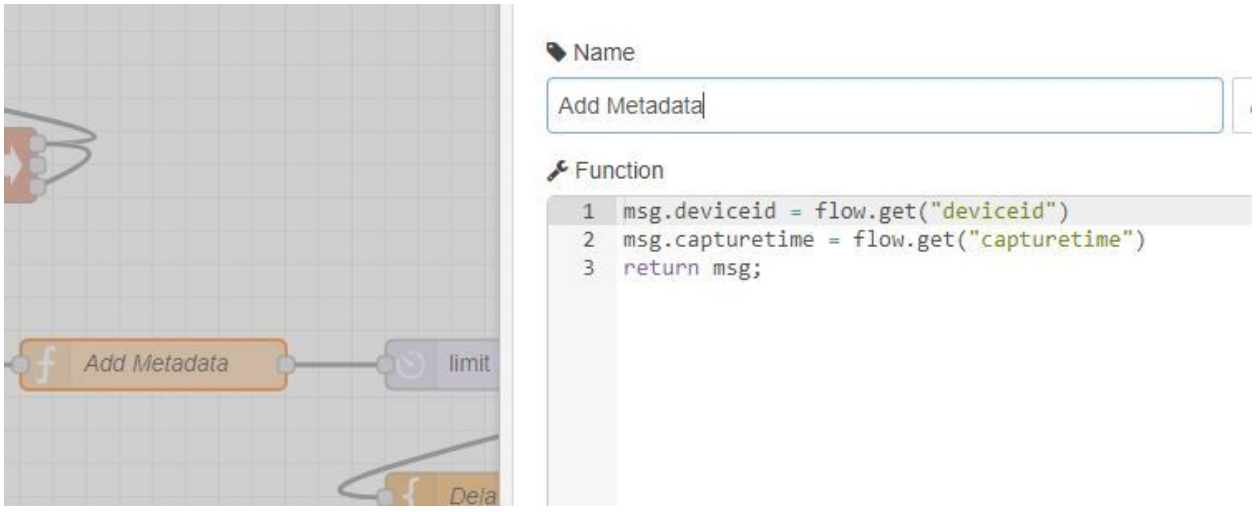
- A Cloudant Database instance and database have been created.
- Raspberry Pi Camera has been setup.
- Install node-red-node-cf-cloundant into Node-Red (on Raspberry Pi).
- Find the attached Image_Workflow.json and import the json content into a Node-Red flow.
- Adjust the first node (Inject node) for your prefered time interval.
- In cloudant node, replace the server credentials and database name to yours.
- Finally click Deploy button to start.

Metadata Integration:

- At the beginning of a flow, Node-Red runs shell commands and stores the output as flow variables, which are accessible by the whole flow.
- Later in Add Metadata node, it declares new variables and assigns values.

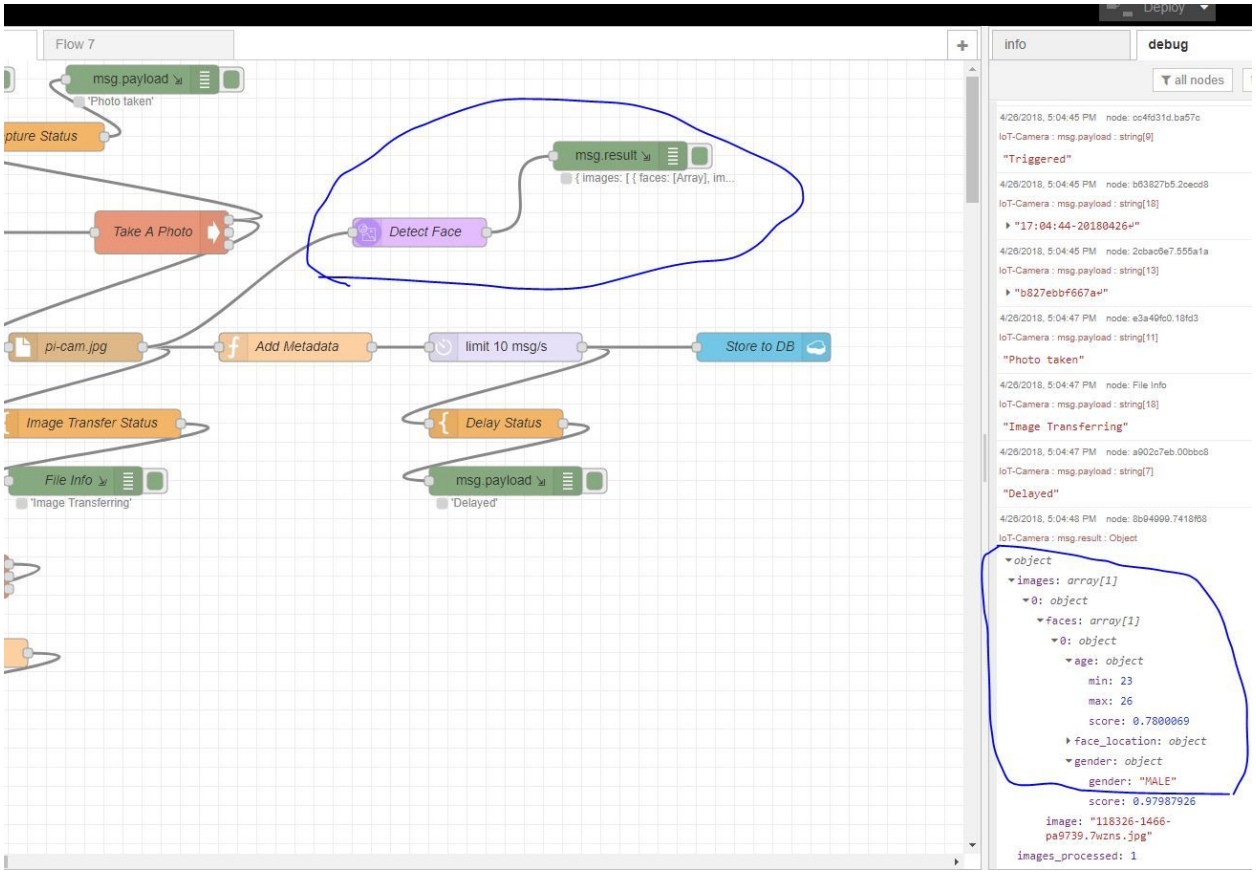




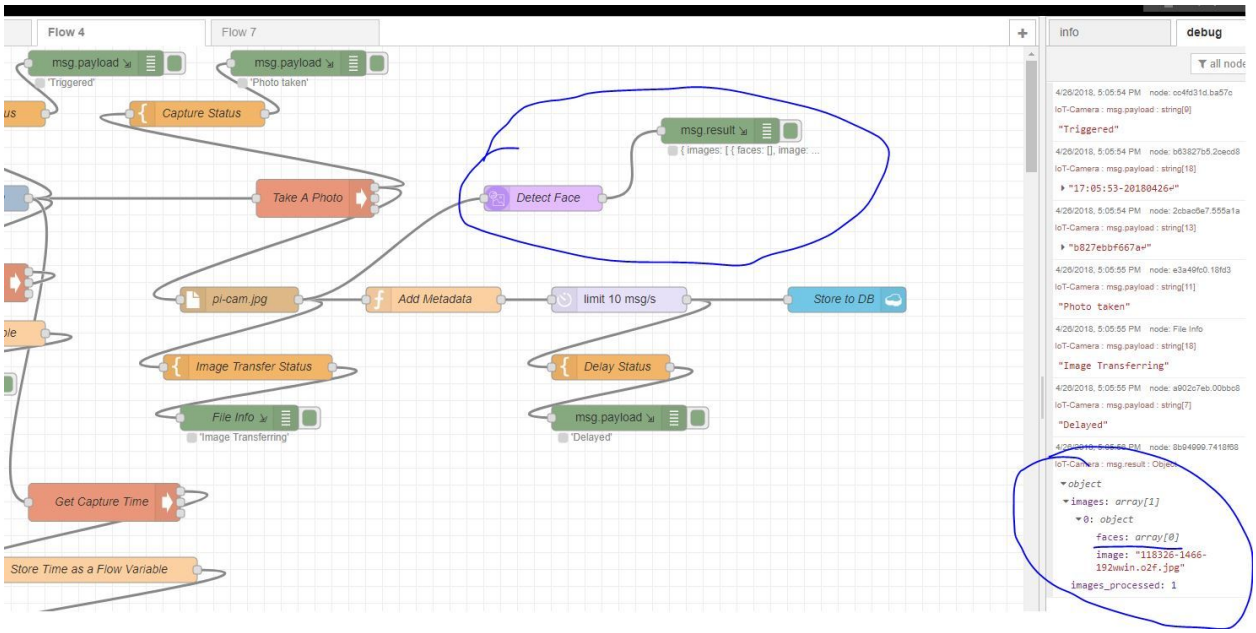


Face Detection:

If face detected, the msg.result would display the age, face location, and gender with probability.



If not, an empty array returned.



Performance Results:

SensorTag Data:

- For a single SensorTag, the maximum frequency of getting and sending data to Cloud is 0.1 second.

Image Data (by using raspistill command):

- The frequency of capturing and transferring an image to Cloudant DB depends on the image size, output data type, and network speed.

- Briefly,
    - Image Size <= 250KB
        - Frequency = 1 second
    - 250KB < Image Size <= 1MB
        - Frequency = 2 seconds
    - Image Size = 2MB
        - Frequency = 10 seconds
    - Image Size = 3MB
        - Frequency = 20 seconds
    - Image Size = 5MB
        - Frequency = 30 seconds
- Specifically,
    - Frequency depends on file size and output data type:
        - Image size <= 250 KB && Output Data Type == A single buffer object:
            - Maximum Frequency == 1s
        - 250 KB <= Image size < 1 MB:
            - If the output data type is “A single buffer object” && Frequency == 2s:
                - Failed to transfer.
                    - Error 413 raised. Request entity too large. The maximum request body size for an API request sent to Cloudant NoSQL DB on IBM Bluemix is 1 MB.
                    - Even though the actual image size is less than 1MB, the body size (converted value) exceeds the maximum.
            - If the output data type is “A stream of buffers” && Frequency == 2s:
                - Succeeded to transfer.

                - The request body would be separated by 4 or more messages (documents) that write to Cloudant.
        - Image >= 1MB:
            - Output Data Type: A stream of buffers.
                - Frequency == 2s
                    - Failed to transfer.
                        - Error 429 raised. The user has sent too many requests in a given amount of time.
                        - The number of separated messages (documents) write to Cloudant is larger than 10 per second.
                        - A free Cloudant plan allows 10 messages to transfer per second.
            - Limit the messages transfer rate: maximumly allows 10 documents to be transferred per second.
                - The frequency then depends on the image size.
                    - 2M - 10 second.
                    - 3M - 20 second.
                    - 5M - 30 second.

- Frequency depends on network speed:
    - The experiment above conducted at midnight around 1 AM, which gave the optimal result.
    - If conducting the experiment around 7 PM (busy hour), it requires more time for data transfer and there is a higher possibility of failed data transfer (raising an error 500, invalid information or lost of packets).

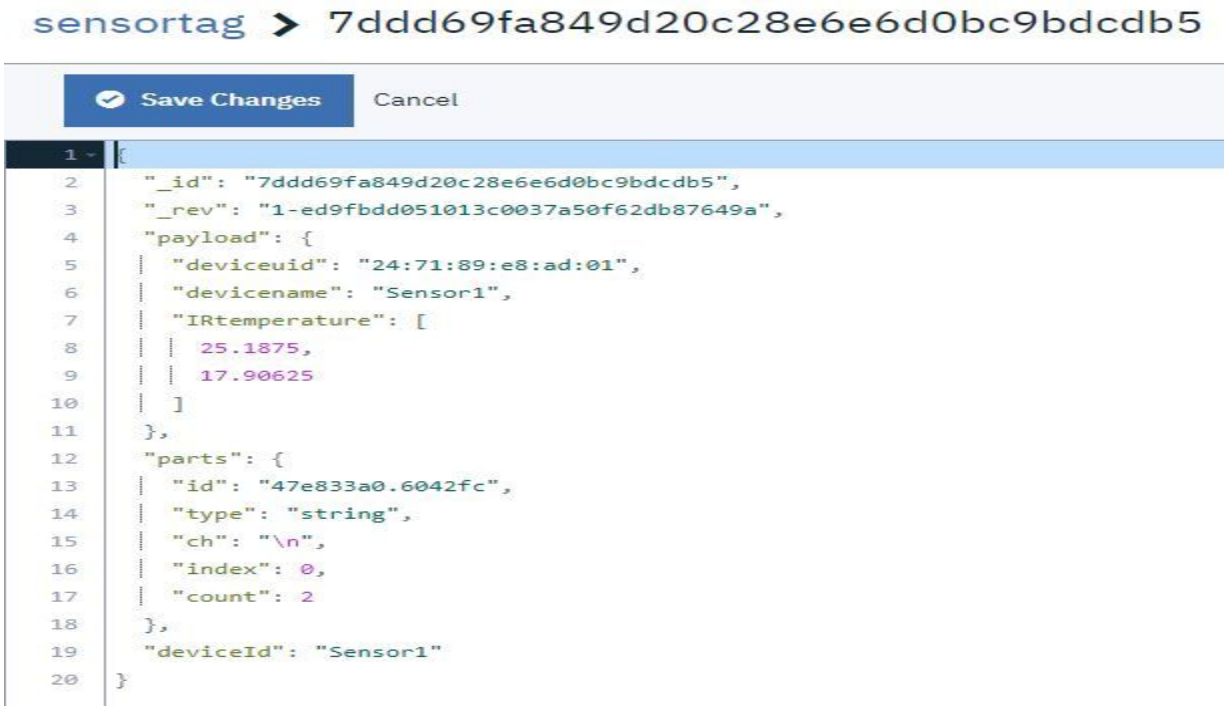
Limitations:

A free Cloudant Database plan maximumly allows 10 writes (data transfer) per second.
It limits:

- Sensor Data:
    - If one SensorTag data needs to be transferred, the frequency is 0.1 second.
    - If two SensorTags' data need to be transferred, the frequency is 0.2 second.
    - If ten SensorTags' data need to be transferred, the frequency is 1 second.
- Image Data:
    - When the image size is large enough, the data which needs to be transferred would be separated into parts.
    - If the image size is larger than 1MB, the data would be separated into more than 10 documents, which means it exceeds 10 writes per second and raise an error.
    - Consequently, limitation of message writes per second required and it increases the frequency.

Appendix:

Sample SensorTag Data:



Sample Image Data:


```
{
  "_id": "05ce3b3ee95b88f0e149eee501b379d2",
  "_rev": "1-707e0c1087cda1b9a5942a29c209a384",
  "payload": {
    "type": "Buffer",
    "data": [
      210,
      89,
      210,
      85,
      75,
      57,
      37,
      185,
      142,
      226,
      48,
      247,
      51,
      75,
      42,
      228,
      199,
      109,
      108,
      209,
      0,
      155,
      0,
      82,
      49,
      180,
      153,
      102,
      9,
      20,
      141,
```


```
65525            121,
65526            197,
65527            38,
65528            8,
65529            20,
65530            0,
65531            191,
65532            94,
65533            148,
65534            163,
65535            36,
65536            243,
65537            214,
65538            154,
65539            48,
65540            125,
65541            105,
65542            73
65543          ]
65544        },
65545        "topic": "IoT-Camera",
65546        "filename": "/home/pi/camera/pi-cam.jpg",
65547        "parts": {
65548          "index": 4,
65549          "ch": "",
65550          "type": "buffer",
65551          "id": "97dc220.381eee"
65552        },
65553        "deviceid": "b827ebbf667a\n",
65554        "capturetime": "15:27:59-20180426\n"
65555    }
```

Reference:

https://developer.ibm.com/recipes/tutorials/connecting-multiple-ti-sensortags-to-iot-platfom-using-a-raspberry-pi-gateway/

https://developer.ibm.com/recipes/tutorials/rpi-cam-image-analysis-using-visual-recognition-method/

https://www.raspberrypi.org/app/uploads/2013/07/RaspiCam-Documentation.pdf/

https://www.ibm.com/cloud/cloudant/pricing/