REST Server

Student: Yi Qiang Ji Zhang
Professor: Dr. Jaume Figueras Jove
Aerospace Engineering
Polythecnical University of Catalonia

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

07 June 2021

1 Creating a database using SQLite

The following code intends to create a REST Server with SQLite and then read a write data from it.

```
# Create a database and add different tables

# Import the SenseHat object from the emulator
from sense_emu import SenseHat
import time # Time library
import datetime
import csv
import sqlite3

# Create an object
sense = SenseHat()
sense.clear()

# Create database
sqlite_file = "database.db"

# Connection
conn = sqlite3.connect(sqlite_file)

# Cursor for commands and accessing information
cur = conn.cursor()

# 0=real, 1=virtual=compound

# Insert new sensors to database file
# Command to create a SENSOR table
sql = "CREATE TABLE sensors (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, ...
    description TEXT, compound INTEGER)"
cur.execute(sql)

```

```
# Command to create a VARIABLES table
sql = "CREATE TABLE variables (id INTEGER PRIMARY KEY AUTOINCREMENT, sensor_id INTEGER, name TEXT... NOT NULL, description TEXT, units TEXT)"
cur.execute(sql)

# Command to create a MEASURES table
sql = "CREATE TABLE measures (id INTEGER PRIMARY KEY AUTOINCREMENT, variable_id INTEGER, measure ... TEXT, date TEXT)"
cur.execute(sql)

# Commit the changes
conn.commit()


# Insert new sensors into table
# 1. Pressure sensor
new_sensor = "INSERT INTO sensors (name, description, compound) VALUES ('Pressure', 'Pressure ... sensor', 0)"
cur.execute(new_sensor)

# 2. Humidity sensor
new_sensor = "INSERT INTO sensors (name, description, compound) VALUES ('Humidity', 'Humidity ... sensor', 0)"
cur.execute(new_sensor)

# 3. Temperature sensor
new_sensor = "INSERT INTO sensors (name, description, compound) VALUES ('Temperature', '... Temperature sensor', 0)"
cur.execute(new_sensor)

# 4. Magnetometer (Compass) sensor
new_sensor = "INSERT INTO sensors (name, description, compound) VALUES ('Magnetometer', '... Magnetometer (Compass) sensor', 0)"
cur.execute(new_sensor)

# 5. Accelerometer sensor
new_sensor = "INSERT INTO sensors (name, description, compound) VALUES ('Accelerometer', '... Accelerometer sensor', 0)"
cur.execute(new_sensor)

# 6. Gyroscope sensor
new_sensor = "INSERT INTO sensors (name, description, compound) VALUES ('Gyroscope', 'Gyroscope ... sensor', 0)"
cur.execute(new_sensor)

# 7. IMU sensor
new_sensor = "INSERT INTO sensors (name, description, compound) VALUES ('IMU', 'IMU sensor (... orientation processed by IMU)', 0)"
cur.execute(new_sensor)

# Commit the changes
conn.commit()


# Insert new variables into table
# Pressure variable from Pressure sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('1', 'Pressure','mbar')"
cur.execute(new_var)
```

```
79
80 # Temperature variable from Temperature sensor
81 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('1', 'Temperature','°C')"
82 cur.execute(new_var)
83
84 # Humidity variable from Humidity sensor
85 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('2', 'Humidity','%rH')"
86 cur.execute(new_var)
87
88 # Temperature variable from Humidity sensor
89 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('2', 'Temperature','°C')"
90 cur.execute(new_var)
91
92 # Temperature variable from Temperature sensor
93 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('3', 'Temperature','°C')"
94 cur.execute(new_var)
95
96 # Magnetometer variable from Magnetometer sensor
97 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('4', 'Magnetometer','% compass...
       ')"
98 cur.execute(new_var)
99
100 # Accelerometer (X,Y,Z) variables from Accelerometer sensor
101 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('5', 'X','g')"
102 cur.execute(new_var)
103 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('5', 'Y','g')"
104 cur.execute(new_var)
105 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('5', 'Z','g')"
106 cur.execute(new_var)
107
108 # Gyroscope (Pitch,Roll,Yaw) variables from Gyroscope sensor
109 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('6', 'Pitch','°')"
110 cur.execute(new_var)
111 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('6', 'Roll','°')"
112 cur.execute(new_var)
113 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('6', 'Yaw','°')"
114 cur.execute(new_var)
115
116 # IMU (Pitch,Roll,Yaw) variables from IMU sensor
117 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('7', 'Pitch','°')"
118 cur.execute(new_var)
119 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('7', 'Roll','°')"
120 cur.execute(new_var)
121 new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('7', 'Yaw','°')"
122 cur.execute(new_var)
123
124 # Commit the changes
125 conn.commit()
126
127
128 ## Get measures
129 # NUmber of measures
130 n = 5
131
132 for i in range(1,n):
133
134     # Save current beginning time
135     start = time.time()
```

```
136
137     # Read an write time
138     current_time = datetime.datetime.utcnow()
139
140     # Read pressure from pressure sensor
141     pressure = sense.get_pressure()
142     # Read temp from pressure sensor
143     temp_p = sense.get_temperature_from_pressure()
144
145     # Read humidity from humidity sensor
146     humidity = sense.get_humidity()
147     # Read temp from humidity sensor
148     temp_h = sense.get_temperature_from_humidity()
149
150     # Read temp from temperature sensor
151     temp = sense.get_temperature()
152
153     # Read magnetometer (compass) data from magnetometer sensor
154     for i in range(0,10):
155         compass = sense.get_compass()
156
157     # Read accelerometer data from accelerometer sensor
158     for i in range(0,10):
159         accel_only = sense.get_accelerometer()
160     pitch_acc = accel_only["pitch"]
161     roll_acc = accel_only["roll"]
162     yaw_acc = accel_only["yaw"]
163
164     # Read gyroscope data from gyroscope sensor
165     for i in range(0,10):
166         gyro_only = sense.get_gyroscope()
167     pitch_gyro = gyro_only["pitch"]
168     roll_gyro = gyro_only["roll"]
169     yaw_gyro = gyro_only["yaw"]
170
171     # Read IMU data from IMU sensor (processed)
172     for i in range(0,10):
173         o = sense.get_orientation() # 'o' object is a dictionary
174     pitch_IMU = o["pitch"]
175     roll_IMU = o["roll"]
176     yaw_IMU = o["yaw"]
177
178     # Write Pressure measurement from Pressure sensor
179     query = "INSERT INTO measures (variable_id, measure, date) VALUES (1, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(pressure,current_time)
180     cur.execute(query)
181
182     # Write Temperature measurement from Pressure sensor
183     query = "INSERT INTO measures (variable_id, measure, date) VALUES (2, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(temp_p,current_time)
184     cur.execute(query)
185
186     # Write Humidity measurement from HUmidity sensor
187     query = "INSERT INTO measures (variable_id, measure, date) VALUES (3, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(humidity,current_time)
188     cur.execute(query)
189
190     # Write Temperature measurement from Humidity sensor
```

```
191     query = "INSERT INTO measures (variable_id, measure, date) VALUES (4, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(temp_h,current_time)
192     cur.execute(query)
193
194     # Write Temperature measurement from Temperature sensor
195     query = "INSERT INTO measures (variable_id, measure, date) VALUES (5, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(temp,current_time)
196     cur.execute(query)
197
198     # Write Magnetometer (compass) measurement from Magnetometer sensor
199     query = "INSERT INTO measures (variable_id, measure, date) VALUES (6, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(compass,current_time)
200     cur.execute(query)
201
202     # Write Accelerometer measurement from Accelerometer sensor
203     query = "INSERT INTO measures (variable_id, measure, date) VALUES (7, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(pitch_acc,current_time)
204     cur.execute(query)
205     query = "INSERT INTO measures (variable_id, measure, date) VALUES (8, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(roll_acc,current_time)
206     cur.execute(query)
207     query = "INSERT INTO measures (variable_id, measure, date) VALUES (9, {0}, '{1:%Y-%m-%d %H:%M...
        %S.%f}')".format(yaw_acc,current_time)
208     cur.execute(query)
209
210     # Write Gyroscope measurement from Gyroscope sensor
211     query = "INSERT INTO measures (variable_id, measure, date) VALUES (10, {0}, '{1:%Y-%m-%d %H:%...
        M%S.%f}')".format(pitch_gyro,current_time)
212     cur.execute(query)
213     query = "INSERT INTO measures (variable_id, measure, date) VALUES (11, {0}, '{1:%Y-%m-%d %H:%...
        M%S.%f}')".format(roll_gyro,current_time)
214     cur.execute(query)
215     query = "INSERT INTO measures (variable_id, measure, date) VALUES (12, {0}, '{1:%Y-%m-%d %H:%...
        M%S.%f}')".format(yaw_gyro,current_time)
216     cur.execute(query)
217
218     # Write IMU measurement from IMU (processed) sensor
219     query = "INSERT INTO measures (variable_id, measure, date) VALUES (13, {0}, '{1:%Y-%m-%d %H:%...
        M%S.%f}')".format(pitch_IMU,current_time)
220     cur.execute(query)
221     query = "INSERT INTO measures (variable_id, measure, date) VALUES (14, {0}, '{1:%Y-%m-%d %H:%...
        M%S.%f}')".format(roll_IMU,current_time)
222     cur.execute(query)
223     query = "INSERT INTO measures (variable_id, measure, date) VALUES (15, {0}, '{1:%Y-%m-%d %H:%...
        M%S.%f}')".format(yaw_IMU,current_time)
224     cur.execute(query)
225
226     # Save current end time
227     end = time.time()
228     elapsed_time = start - end
229
230     # Sample sample_frequency
231     sample_frequency = 1/elapsed_time
232
233     # Make measurements every second
234     time.sleep(1 - elapsed_time) # Sleep for 1 second taking into account the elapsed time
235
236     # Commit the changes
```

```
237     conn.commit()
238
239
240 # Commit the changes
241 conn.commit()
242
243 # Sleep for 5 seconds
244 time.sleep(5)
245
246 # Close connection
247 conn.close()
```

Listing 1: Main Program

```
1  # Insert data to database
2
3  # Libraries
4  import sqlite3
5
6  # Create database
7  sqlite_file = "database.db"
8
9  # Connection
10 conn = sqlite3.connect(sqlite_file)
11
12 # Cursor for commands and accessing information
13 cur = conn.cursor()
14
15 # Insert data
16 sql = "INSERT INTO sensors (name, description, compound) VALUES ('Pressure', 'Pressure sensor',0)...
       "
17 cur.execute(sql)
18 sql = "INSERT INTO sensors (name, description, compound) VALUES ('Humidity', 'Humidity sensor',0)...
       "
19 cur.execute(sql)
20 sql = "INSERT INTO sensors (name, description, compound) VALUES ('Temperature', 'Temperature ...
       sensor',1)"
21 cur.execute(sql)
22
23 # Commit the changes
24 conn.commit()
25
26 # Close connection
27 conn.close()
```

Listing 2: Insert data to dababase

2 Creating a REST server using FLASK

Once the database is all set, the next step is to create a Web server so it can be accessible thought the browser (locally). First, let's create a database and write some measurements on it:

```
1 # Create a database and add different tables
2
3 # Import the SenseHat object from the emulator
```

```
from sense_emu import SenseHat
import time # Time library
import datetime
import sqlite3

# Create an object
sense = SenseHat()
sense.clear()

# Function to check if database is used by a process
def is_open(conn):
     try:
        conn.cursor()
        return True
     except Exception as ex:
        return False


# Open database
sqlite_file = "database.db"

# Connection
conn = sqlite3.connect(sqlite_file)

# If database is not being used (not opened)
if is_open(conn):

    # Cursor for commands and accessing information
    cur = conn.cursor()

    ## Get measures
    # NUmber of measures
    n = 3

    for i in range(1,n):

        # Save current beginning time
        start = time.time()

        # Read an write time
        current_time = datetime.datetime.utcnow()

        # Read pressure from pressure sensor
        pressure = sense.get_pressure()
        # Read temp from pressure sensor
        temp_p = sense.get_temperature_from_pressure()

        # Read humidity from humidity sensor
        humidity = sense.get_humidity()
        # Read temp from humidity sensor
        temp_h = sense.get_temperature_from_humidity()

        # Read temp from temperature sensor
        temp = sense.get_temperature()

        # Read magnetometer (compass) data from magnetometer sensor
        for i in range(0,10):
            compass = sense.get_compass()
```

```
62
63          # Read accelerometer data from accelerometer sensor
64          for i in range(0,10):
65              accel_only = sense.get_accelerometer()
66          pitch_acc = accel_only["pitch"]
67          roll_acc = accel_only["roll"]
68          yaw_acc = accel_only["yaw"]
69
70          # Read gyroscope data from gyroscope sensor
71          for i in range(0,10):
72              gyro_only = sense.get_gyroscope()
73          pitch_gyro = gyro_only["pitch"]
74          roll_gyro = gyro_only["roll"]
75          yaw_gyro = gyro_only["yaw"]
76
77          # Read IMU data from IMU sensor (processed)
78          for i in range(0,10):
79              o = sense.get_orientation() # 'o' object is a dictionary
80          pitch_IMU = o["pitch"]
81          roll_IMU = o["roll"]
82          yaw_IMU = o["yaw"]
83
84          # Write Pressure measurement from Pressure sensor
85          query = "INSERT INTO measures (variable_id, measure, date) VALUES (1, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(pressure,current_time)
86          cur.execute(query)
87
88          # Write Temperature measurement from Pressure sensor
89          query = "INSERT INTO measures (variable_id, measure, date) VALUES (2, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(temp_p,current_time)
90          cur.execute(query)
91
92          # Write Humidity measurement from HUmidity sensor
93          query = "INSERT INTO measures (variable_id, measure, date) VALUES (3, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(humidity,current_time)
94          cur.execute(query)
95
96          # Write Temperature measurement from Humidity sensor
97          query = "INSERT INTO measures (variable_id, measure, date) VALUES (4, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(temp_h,current_time)
98          cur.execute(query)
99
100         # Write Temperature measurement from Temperature sensor
101         query = "INSERT INTO measures (variable_id, measure, date) VALUES (5, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(temp,current_time)
102         cur.execute(query)
103
104         # Write Magnetometer (compass) measurement from Magnetometer sensor
105         query = "INSERT INTO measures (variable_id, measure, date) VALUES (6, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(compass,current_time)
106         cur.execute(query)
107
108         # Write Accelerometer measurement from Accelerometer sensor
109         query = "INSERT INTO measures (variable_id, measure, date) VALUES (7, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(pitch_acc,current_time)
110         cur.execute(query)
111         query = "INSERT INTO measures (variable_id, measure, date) VALUES (8, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(roll_acc,current_time)
```

```
        cur.execute(query)
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (9, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(yaw_acc,current_time)
        cur.execute(query)

        # Write Gyroscope measurement from Gyroscope sensor
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (10, {0}, '{1:%Y-%m-%dT...
    %H:%M%S.%fZ}')".format(pitch_gyro,current_time)
        cur.execute(query)
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (11, {0}, '{1:%Y-%m-%dT...
    %H:%M%S.%fZ}')".format(roll_gyro,current_time)
        cur.execute(query)
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (12, {0}, '{1:%Y-%m-%dT...
    %H:%M%S.%fZ}')".format(yaw_gyro,current_time)
        cur.execute(query)

        # Write IMU measurement from IMU (processed) sensor
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (13, {0}, '{1:%Y-%m-%dT...
    %H:%M%S.%fZ}')".format(pitch_IMU,current_time)
        cur.execute(query)
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (14, {0}, '{1:%Y-%m-%dT...
    %H:%M%S.%fZ}')".format(roll_IMU,current_time)
        cur.execute(query)
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (15, {0}, '{1:%Y-%m-%dT...
    %H:%M%S.%fZ}')".format(yaw_IMU,current_time)
        cur.execute(query)

        # Save current end time
        end = time.time()
        elapsed_time = start - end

        # Sample sample_frequency
        sample_frequency = 1/elapsed_time

        # Make measurements every second
        time.sleep(1 - elapsed_time) # Sleep for 1 second taking into account the elapsed time

        # Sleep for 5 seconds
        time.sleep(5)

        # Commit the changes
        conn.commit()


    # Commit the changes
    conn.commit()

    # Close connection
    conn.close()

else:
    print("Database is currently being used")
```

Listing 3: Create and write to database

```

# Import Flask
```

```
3  from flask import Flask, redirect
4  from sense_emu import SenseHat
5  import datetime
6  from flask import jsonify
7  from flask import request
8  import sqlite3
9
10 sense = SenseHat()
11
12 # Store name of the program
13 app = Flask(__name__)
14
15 # Route of the app is the main route of the web server
16
17
18 @app.route('/')
19 # Function
20 def index():
21     message = "Raspberry PI ICT REST Server"
22     return message
23
24 # Function to check if database is used by a process
25
26
27 def is_open(conn):
28     try:
29         conn.cursor()
30         return True
31     except Exception as ex:
32         return False
33
34
35 # Open database
36 sqlite_file = "database.db"
37
38 # Connection
39 conn = sqlite3.connect(sqlite_file)
40
41
42 # Create a new sensors route
43 # (http://127.0.0.1:5000/sensors?origin={temperature,pressure,humidity,accelerometer,gyroscope,...
       magnetometer,imu})
44 @app.route('/sensors')
45 # Function to show all available sensors
46 def sensors():
47     origin = request.args.get('origin')
48     if origin is None:
49         # Create a dictionary
50         data = dict()
51         # Save variables
52         data['00_message'] = "Sensors:"
53         data['temperature'] = "Temperature"
54         data['pressure'] = "Pressure"
55         data['humidity'] = "Humidity"
56         data['accelerometer'] = "Accelerometer"
57         data['gyroscope'] = "Gyroscope"
58         data['magnetometer'] = "Magnetometer"
59         data['imu'] = "IMU"
```

```
60          return jsonify(data)
61
62      else:
63          if origin == 'temperature':
64              return redirect('/sensors/temperature')
65          elif origin == 'pressure':
66              return redirect('/sensors/temperature')
67          elif origin == 'humidity':
68              return redirect('/sensors/humidity')
69          elif origin == 'acceloremeter':
70              return redirect('sensors/accelerometer')
71          elif origin == 'gyroscope':
72              return redirect('sensors/gyroscope')
73          elif origin == 'magnetometer':
74              return redirect('sensors/magnetometer')
75          elif origin == 'imu':
76              return redirect('sensors/imu')
77
78
79  # Sensors
80  # Create a new temperature route
81  @app.route('/sensors/temperature')
82  # Function to get temperature from temperature sensor
83  def temp():
84      temp = sense.get_temperature()
85      # Create a dictionary
86      data = dict()
87      # Save variables
88      data['temp'] = temp
89      data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
90          datetime.datetime.utcnow())
91      return jsonify(data)
92
93
94  @app.route('/sensors/temperature/pressure')
95  # Function to get temperature from pressure sensor
96  def temp_pressure():
97      temp_p = sense.get_temperature_from_pressure()
98      # Create a dictionary
99      data = dict()
100     # Save variables
101     data['temp_p'] = temp_p
102     data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
103         datetime.datetime.utcnow())
104     return jsonify(data)
105
106
107 @app.route('/sensors/temperature/humidity')
108 # Function to get temperature from humidity sensor
109 def temp_humidity():
110     temp_h = sense.get_temperature_from_humidity()
111     # Create a dictionary
112     data = dict()
113     # Save variables
114     data['temp_h'] = temp_h
115     data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
116         datetime.datetime.utcnow())
117     return jsonify(data)
```

```
118
119
120 # Pressure route
121 @app.route('/sensors/pressure')
122 # Function to get pressure from presure sensor
123 def pressure():
124     pressure = sense.get_pressure()
125     # Create a dictionary
126     data = dict()
127     # Save variables
128     data['pressure'] = pressure
129     data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
130         datetime.datetime.utcnow())
131     return jsonify(data)
132
133 # Humidity route
134
135
136 @app.route('/sensors/humidity')
137 # Function to get pressure from presure sensor
138 def humidty():
139     humidity = sense.get_humidity()
140     # Create a dictionary
141     data = dict()
142     # Save variables
143     data['humidity'] = humidity
144     data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
145         datetime.datetime.utcnow())
146     return jsonify(data)
147
148 # Compass route
149
150
151 @app.route('/sensors/compass')
152 # Function to get magnetometer (compass) from magnetometer sensor
153 def compass():
154     compass = sense.get_compass()
155     # Create a dictionary
156     data = dict()
157     # Save variables
158     data['compass'] = compass
159     data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
160         datetime.datetime.utcnow())
161     return jsonify(data)
162
163
164 # Accelerometer route
165 @app.route('/sensors/accelerometer')
166 # Function to get accelerometer from accelerometer sensor
167 def accelerometer():
168     # Read accelerometer data from accelerometer sensor
169     for i in range(0, 10):
170         accel_only = sense.get_accelerometer()
171     pitch_acc = accel_only["pitch"]
172     roll_acc = accel_only["roll"]
173     yaw_acc = accel_only["yaw"]
174     # Create a dictionary
175     data = dict()
```

```
176     # Save variables
177     data['pitch_acc'] = pitch_acc
178     data['roll_acc'] = roll_acc
179     data['yaw_acc'] = yaw_acc
180     data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
181         datetime.datetime.utcnow())
182     return jsonify(data)
183 
184 # Gyroscope route
185 
186 
187 @app.route('/gyroscope')
188 # Function to get gyroscope from gyroscope sensor
189 def gyroscope():
190     # Read gyroscope data from gyroscope sensor
191     for i in range(0, 10):
192         gyro_only = sense.get_gyroscope()
193     pitch_gyro = gyro_only["pitch"]
194     roll_gyro = gyro_only["roll"]
195     yaw_gyro = gyro_only["yaw"]
196     # Create a dictionary
197     data = dict()
198     # Save variables
199     data['pitch_gyro'] = pitch_gyro
200     data['roll_gyro'] = roll_gyro
201     data['yaw_gyro'] = yaw_gyro
202     data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
203         datetime.datetime.utcnow())
204     return jsonify(data)
205 
206 
207 # IMU route
208 @app.route('/sensors/imu')
209 # Function to get IMU from IMU sensor (processed)
210 def imu():
211     # Read IMU data from IMU sensor (processed)
212     for i in range(0, 10):
213         o = sense.get_orientation()  # 'o' object is a dictionary
214     pitch_IMU = o["pitch"]
215     roll_IMU = o["roll"]
216     yaw_IMU = o["yaw"]
217     # Create a dictionary
218     data = dict()
219     # Save variables
220     data['pitch_IMU'] = pitch_IMU
221     data['roll_IMU'] = roll_IMU
222     data['yaw_IMU'] = yaw_IMU
223     data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
224         datetime.datetime.utcnow())
225     return jsonify(data)
226 
227 
228 # History Requests
229 # http://127.0.0.1:5000/sensors/temperature/history?from=2021-05-11&to=2021-05-12
230 
231 # Temperature history
232 @app.route('/sensors/temperature/history')
233 # Request history
```

```
234 def temp_history():
235     from_date = request.args.get('from')
236     to_date = request.args.get('to')
237 
238     from_date_complete = from_date + "T00:00:00"
239     to_date_complete = to_date + "T23:59:59"
240 
241     # query = "SELECT sensors.name, variables.name, measures.measure, max(measures.date), ...
        variables.units FROM sensors, variables, measures WHERE sensors.id = variables.sensor_id AND ...
        variables.id = measures.variable_id GROUP BY variables.id"
242 
243     query = "SELECT * FROM Temperature_sensor WHERE date > from_date_complete AND date < ...
        to_date_complete"
244 
245     if is_open(conn):
246         # Cursor for commands and accessing information
247         cur = conn.cursor()
248 
249         cur.execute(query)
250         rows = cur.fetchall()
251         print("Measures in database")
252         print(rows)
253     else:
254         print("Database is currently being used")
255         # WRITE ERROR MSG
256 
257     return "From {0} to {1}".format(from_date, to_date)
258 
259 
260 # Debug
261 if __name__ == '__main__':
262     app.run(debug=True)
```

Listing 4: REST Server

The main problem encountered here is to try to retrieve the date from the url since it must be formatted as UTC Time.

References

[1] Python Hosted. *Sense HAT API Reference*. 2021. URL: https://pythonhosted.org/sense-hat/api/#imu-sensor.