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)
```

```

# Temperature variable from Temperature sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('1', 'Temperature','°C')"
cur.execute(new_var)

# Humidity variable from Humidity sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('2', 'Humidity','%rH')"
cur.execute(new_var)

# Temperature variable from Humidity sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('2', 'Temperature','°C')"
cur.execute(new_var)

# Temperature variable from Temperature sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('3', 'Temperature','°C')"
cur.execute(new_var)

# Magnetometer variable from Magnetometer sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('4', 'Magnetometer','% compass...
    ')"
cur.execute(new_var)

# Accelerometer (X,Y,Z) variables from Accelerometer sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('5', 'X','g')"
cur.execute(new_var)
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('5', 'Y','g')"
cur.execute(new_var)
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('5', 'Z','g')"
cur.execute(new_var)

# Gyroscope (Pitch,Roll,Yaw) variables from Gyroscope sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('6', 'Pitch','°')"
cur.execute(new_var)
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('6', 'Roll','°')"
cur.execute(new_var)
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('6', 'Yaw','°')"
cur.execute(new_var)

# IMU (Pitch,Roll,Yaw) variables from IMU sensor
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('7', 'Pitch','°')"
cur.execute(new_var)
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('7', 'Roll','°')"
cur.execute(new_var)
new_var = "INSERT INTO variables (sensor_id, name, units) VALUES ('7', 'Yaw','°')"
cur.execute(new_var)

# Commit the changes
conn.commit()


## Get measures
# NUmber of measures
n = 5

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()

    # Read accelerometer data from accelerometer sensor
    for i in range(0,10):
        accel_only = sense.get_accelerometer()
    pitch_acc = accel_only["pitch"]
    roll_acc = accel_only["roll"]
    yaw_acc = accel_only["yaw"]

    # Read gyroscope data from gyroscope sensor
    for i in range(0,10):
        gyro_only = sense.get_gyroscope()
    pitch_gyro = gyro_only["pitch"]
    roll_gyro = gyro_only["roll"]
    yaw_gyro = gyro_only["yaw"]

    # Read IMU data from IMU sensor (processed)
    for i in range(0,10):
        o = sense.get_orientation() # 'o' object is a dictionary
    pitch_IMU = o["pitch"]
    roll_IMU = o["roll"]
    yaw_IMU = o["yaw"]

    # Write Pressure measurement from Pressure sensor
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (1, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".format(pressure,current_time)
    cur.execute(query)

    # Write Temperature measurement from Pressure sensor
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (2, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".format(temp_p,current_time)
    cur.execute(query)

    # Write Humidity measurement from HUmidity sensor
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (3, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".format(humidity,current_time)
    cur.execute(query)

    # Write Temperature measurement from Humidity sensor
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (4, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".format(temp_h,current_time)
    cur.execute(query)

    # Write Temperature measurement from Temperature sensor
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (5, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".format(temp,current_time)
    cur.execute(query)

    # Write Magnetometer (compass) measurement from Magnetometer sensor
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (6, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".format(compass,current_time)
    cur.execute(query)

    # Write Accelerometer measurement from Accelerometer sensor
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (7, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".format(pitch_acc,current_time)
    cur.execute(query)
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (8, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".format(roll_acc,current_time)
    cur.execute(query)
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (9, {0}, '{1:%Y-%m-%d %H:%M...
    %S.%f}')".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-%d %H:%...
    M%S.%f}')".format(pitch_gyro,current_time)
    cur.execute(query)
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (11, {0}, '{1:%Y-%m-%d %H:%...
    M%S.%f}')".format(roll_gyro,current_time)
    cur.execute(query)
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (12, {0}, '{1:%Y-%m-%d %H:%...
    M%S.%f}')".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-%d %H:%...
    M%S.%f}')".format(pitch_IMU,current_time)
    cur.execute(query)
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (14, {0}, '{1:%Y-%m-%d %H:%...
    M%S.%f}')".format(roll_IMU,current_time)
    cur.execute(query)
    query = "INSERT INTO measures (variable_id, measure, date) VALUES (15, {0}, '{1:%Y-%m-%d %H:%...
    M%S.%f}')".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

    # Commit the changes
    conn.commit()


# Commit the changes
conn.commit()

# Sleep for 5 seconds
time.sleep(5)

# Close connection
conn.close()
```

Listing 1: Main Program

```
# Insert data to database

# Libraries
import sqlite3

# Create database
sqlite_file = "database.db"

# Connection
conn = sqlite3.connect(sqlite_file)

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

# Insert data
sql = "INSERT INTO sensors (name, description, compound) VALUES ('Pressure', 'Pressure sensor',0)...
    "
cur.execute(sql)
sql = "INSERT INTO sensors (name, description, compound) VALUES ('Humidity', 'Humidity sensor',0)...
    "
cur.execute(sql)
sql = "INSERT INTO sensors (name, description, compound) VALUES ('Temperature', 'Temperature ...
    sensor',1)"
cur.execute(sql)

# Commit the changes
conn.commit()

# Close connection
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:

```
# 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 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()

        # Read accelerometer data from accelerometer sensor
        for i in range(0,10):
            accel_only = sense.get_accelerometer()
        pitch_acc = accel_only["pitch"]
        roll_acc = accel_only["roll"]
        yaw_acc = accel_only["yaw"]

        # Read gyroscope data from gyroscope sensor
        for i in range(0,10):
            gyro_only = sense.get_gyroscope()
        pitch_gyro = gyro_only["pitch"]
        roll_gyro = gyro_only["roll"]
        yaw_gyro = gyro_only["yaw"]

        # Read IMU data from IMU sensor (processed)
        for i in range(0,10):
            o = sense.get_orientation() # 'o' object is a dictionary
        pitch_IMU = o["pitch"]
        roll_IMU = o["roll"]
        yaw_IMU = o["yaw"]

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

        # Write Temperature measurement from Pressure sensor
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (2, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(temp_p,current_time)
        cur.execute(query)

        # Write Humidity measurement from HUmidity sensor
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (3, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(humidity,current_time)
        cur.execute(query)

        # Write Temperature measurement from Humidity sensor
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (4, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(temp_h,current_time)
        cur.execute(query)

        # Write Temperature measurement from Temperature sensor
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (5, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(temp,current_time)
        cur.execute(query)

        # Write Magnetometer (compass) measurement from Magnetometer sensor
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (6, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(compass,current_time)
        cur.execute(query)

        # Write Accelerometer measurement from Accelerometer sensor
        query = "INSERT INTO measures (variable_id, measure, date) VALUES (7, {0}, '{1:%Y-%m-%dT%...
    H:%M%S.%fZ}')".format(pitch_acc,current_time)
        cur.execute(query)
        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
```

```python
from flask import Flask, redirect
from sense_emu import SenseHat
import datetime
from flask import jsonify
from flask import request
import sqlite3

sense = SenseHat()

# Store name of the program
app = Flask(__name__)

# Route of the app is the main route of the web server


@app.route('/')
# Function
def index():
    message = "Raspberry PI ICT REST Server"
    return message

# 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)


# Create a new sensors route
# (http://127.0.0.1:5000/sensors?origin={temperature,pressure,humidity,accelerometer,gyroscope,...
    magnetometer,imu})
@app.route('/sensors')
# Function to show all available sensors
def sensors():
    origin = request.args.get('origin')
    if origin is None:
        # Create a dictionary
        data = dict()
        # Save variables
        data['00_message'] = "Sensors:"
        data['temperature'] = "Temperature"
        data['pressure'] = "Pressure"
        data['humidity'] = "Humidity"
        data['accelerometer'] = "Accelerometer"
        data['gyroscope'] = "Gyroscope"
        data['magnetometer'] = "Magnetometer"
        data['imu'] = "IMU"
        return jsonify(data)

    else:
        if origin == 'temperature':
            return redirect('/sensors/temperature')
        elif origin == 'pressure':
            return redirect('/sensors/temperature')
        elif origin == 'humidity':
            return redirect('/sensors/humidity')
        elif origin == 'accelorometer':
            return redirect('sensors/accelerometer')
        elif origin == 'gyroscope':
            return redirect('sensors/gyroscope')
        elif origin == 'magnetometer':
            return redirect('sensors/magnetometer')
        elif origin == 'imu':
            return redirect('sensors/imu')


# Sensors
# Create a new temperature route
@app.route('/sensors/temperature')
# Function to get temperature from temperature sensor
def temp():
    temp = sense.get_temperature()
    # Create a dictionary
    data = dict()
    # Save variables
    data['temp'] = temp
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)


@app.route('/sensors/temperature/pressure')
# Function to get temperature from pressure sensor
def temp_pressure():
    temp_p = sense.get_temperature_from_pressure()
    # Create a dictionary
    data = dict()
    # Save variables
    data['temp_p'] = temp_p
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)


@app.route('/sensors/temperature/humidity')
# Function to get temperature from humidity sensor
def temp_humidity():
    temp_h = sense.get_temperature_from_humidity()
    # Create a dictionary
    data = dict()
    # Save variables
    data['temp_h'] = temp_h
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)
```

```


# Pressure route
@app.route('/sensors/pressure')
# Function to get pressure from presure sensor
def pressure():
    pressure = sense.get_pressure()
    # Create a dictionary
    data = dict()
    # Save variables
    data['pressure'] = pressure
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)

# Humidity route


@app.route('/sensors/humidity')
# Function to get pressure from presure sensor
def humidty():
    humidity = sense.get_humidity()
    # Create a dictionary
    data = dict()
    # Save variables
    data['humidity'] = humidity
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)

# Compass route


@app.route('/sensors/compass')
# Function to get magnetometer (compass) from magnetometer sensor
def compass():
    compass = sense.get_compass()
    # Create a dictionary
    data = dict()
    # Save variables
    data['compass'] = compass
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)


# Accelerometer route
@app.route('/sensors/accelerometer')
# Function to get accelerometer from accelerometer sensor
def accelerometer():
    # Read accelerometer data from accelerometer sensor
    for i in range(0, 10):
        accel_only = sense.get_accelerometer()
    pitch_acc = accel_only["pitch"]
    roll_acc = accel_only["roll"]
    yaw_acc = accel_only["yaw"]
    # Create a dictionary
    data = dict()
```

```python
    # Save variables
    data['pitch_acc'] = pitch_acc
    data['roll_acc'] = roll_acc
    data['yaw_acc'] = yaw_acc
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)

# Gyroscope route


@app.route('/gyroscope')
# Function to get gyroscope from gyroscope sensor
def gyroscope():
    # Read gyroscope data from gyroscope sensor
    for i in range(0, 10):
        gyro_only = sense.get_gyroscope()
    pitch_gyro = gyro_only["pitch"]
    roll_gyro = gyro_only["roll"]
    yaw_gyro = gyro_only["yaw"]
    # Create a dictionary
    data = dict()
    # Save variables
    data['pitch_gyro'] = pitch_gyro
    data['roll_gyro'] = roll_gyro
    data['yaw_gyro'] = yaw_gyro
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)


# IMU route
@app.route('/sensors/imu')
# Function to get IMU from IMU sensor (processed)
def imu():
    # Read IMU data from IMU sensor (processed)
    for i in range(0, 10):
        o = sense.get_orientation()  # 'o' object is a dictionary
    pitch_IMU = o["pitch"]
    roll_IMU = o["roll"]
    yaw_IMU = o["yaw"]
    # Create a dictionary
    data = dict()
    # Save variables
    data['pitch_IMU'] = pitch_IMU
    data['roll_IMU'] = roll_IMU
    data['yaw_IMU'] = yaw_IMU
    data['time_stamp'] = "{0:%Y-%m-%dT%H:%M:%S.%fZ}".format(
        datetime.datetime.utcnow())
    return jsonify(data)


# History Requests
# http://127.0.0.1:5000/sensors/temperature/history?from=2021-05-11&to=2021-05-12

# Temperature history
@app.route('/sensors/temperature/history')
# Request history
```

```
def temp_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')

    from_date_complete = from_date + "T00:00:00"
    to_date_complete = to_date + "T23:59:59"

    # 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"

    query = "SELECT * FROM Temperature_sensor WHERE date > from_date_complete AND date < ...
    to_date_complete"

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

        cur.execute(query)
        rows = cur.fetchall()
        print("Measures in database")
        print(rows)
    else:
        print("Database is currently being used")
        # WRITE ERROR MSG

    return "From {0} to {1}".format(from_date, to_date)


# Debug
if __name__ == '__main__':
    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.