Mobile App

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 Create REST server

In the following report, it is shown how to make queries from a mobile app using cordova.

The intention is to have the raspberry running and at the same time, use the mobile interface to make queries.

```python

# Import Flask
from flask import Flask, redirect
from flask_cors import cross_origin
from sense_emu import SenseHat
import datetime
from flask import jsonify
from flask import request

sense = SenseHat()

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

# Route of the app is the main route of the web server
@app.route('/')
@cross_origin()
# Function
def index():
    message = "Raspberry PI ICT REST Server"
    return message


# Create a new sensors route
# (http://127.0.0.1:5000/sensors?origin={temperature,pressure,humidity,accelerometer,gyroscope,...
    magnetometer,imu})
@app.route('/sensors')
@cross_origin()
```

```python
# Function to show all available sensors
def sensors():
    origin = request.args.get('origin')
    if origin is None:
        return "Select sensor: /sensors?origin={temperature,pressure,humidity,accelerometer,...
    gyroscope,magnetometer,imu}"
    else:
        if origin == 'temperature':
            return redirect('/sensors/temperature')
        elif origin == 'pressure':
            return redirect('/sensors/temperature')
        elif origin == 'humidity':
            return redirect('/sensors/humidity')
        elif origin == 'acceloremeter':
            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')
@cross_origin()
# 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')
@cross_origin()
# 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')
@cross_origin()
# 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')
@cross_origin()
# 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')
@cross_origin()
# 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')
@cross_origin()
# 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')
@cross_origin()
# 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()
    # 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')
@cross_origin()
# 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')
@cross_origin()
# 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')
@cross_origin()
# Request history
def temp_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)

# Temperature from Pressure Sensors history
@app.route('/sensors/temperature/presssure/history')
@cross_origin()
# Request history
def temp_p_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)

# Temperature from Humidity sensor history
@app.route('/sensors/temperature/humidity/history')
@cross_origin()
# Request history
def temp_h_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)

# Pressure history
@app.route('/sensors/pressure/history')
@cross_origin()
# Request history
def pressure_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)

# HUmidity history
@app.route('/sensors/humidity/history')
@cross_origin()
# Request history
def humidity_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)

# Compass history
@app.route('/sensors/compass/history')
@cross_origin()
# Request history
def compass_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)

# Accelerometer history
@app.route('/sensors/accelerometer/history')
@cross_origin()
# Request history
def accelerometer_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)

# Gyroscope history
@app.route('/sensors/accelerometer/history')
```

```
@cross_origin()
# Request history
def gyroscope_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)

# IMU history
@app.route('/sensors/imu/history')
@cross_origin()
# Request history
def imu_history():
    from_date = request.args.get('from')
    to_date = request.args.get('to')
    return "From {0} to {1}".format(from_date,to_date)



# Debug
if __name__ =='__main__':
    app.run(debug=True,host='0.0.0.0')
```

Listing 1: Create and write to database

Once this has been performed, we shall go and develop the mobile interface.

The main problem found was the security reason. For security standards, the app could not connect to the REST server due to security standards.

References

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