Multi-Sensor Data Acquisition System

User Manual

April 3rd, 2020

Developer: Xiaomin Heil (University of Louisville)

Client: Dr. Daniel Sierrasosa (University of Louisville)

Table of Contents

Introduction

Multi-Sensor data Acquisition System has two parts: hardware—Multi-Sensor Data Collection Device and software—Multi-Sensor Web Server. Multi-Sensor Data Collection Device consists of four sensors, one distance sensor, three gas sensors which are MQ3, MQ5, MQ8.

Multi-Sensor Web Server is a User Interface (UI) application working in Linux environment, written in Python 3, to collect data in real time and allow users to view data from a website. Multi-Sensor data Acquisition System will help researchers in Innovative & Emerging Technologies Lab for different projects, since data that has been stored can be queried easily.

System Description

Below is the diagram for Multi-Sensor Data Collection Device and its Web Server:

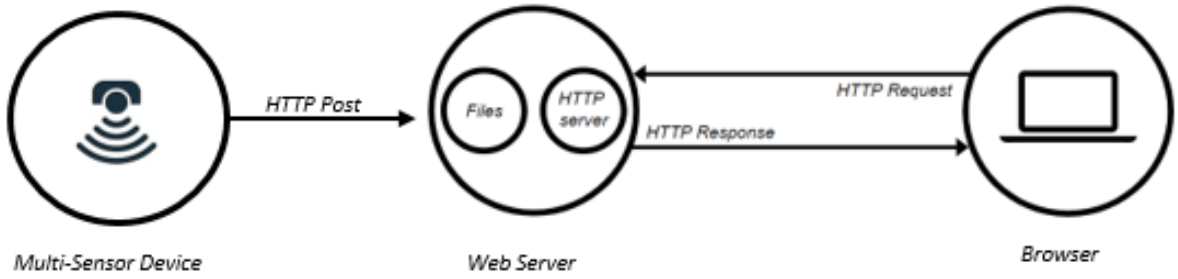

Figure 1. System Diagram

a. Overview

- **Sensors:** There are four sensors used in this system, three of them are used for different gases and the fourth is a distance sensor.
- **Arduino**: An Arduino controller is used as a secondary controller that acts as the interface with the Raspberry Pi and the sensors.
- **Raspberry Pi:** The Raspberry Pi is the main controller for the system. We use two Raspberry Pi's, one at the client side which acts as the main controller and one as the server.

Figure 2. Raspberry Pi overview

- **Operating System:** We use Raspbian Buster as the OS in this project as that is the preferred OS for Raspberry Pi systems as indicated by the Raspberry Pi foundation. Raspbian is a 32-bit Debian Linux distribution, in our work we use version 10 (Buster).

- **Language**:
    - **Python**. Python is a general-purpose programming language used widely in the area of Bioinformatics. Installing Python on any system is easy and the syntax of the language is simple to learn. Python applications can be run on any popular computer operating system. This allows most Python applications to be developed once and then spread to any system that wishes to execute it.
    - **C:** We use C for the Arduino to program the reading of the data from the sensors.

b. Hardware

The schematic for the system is shown in the figure below.

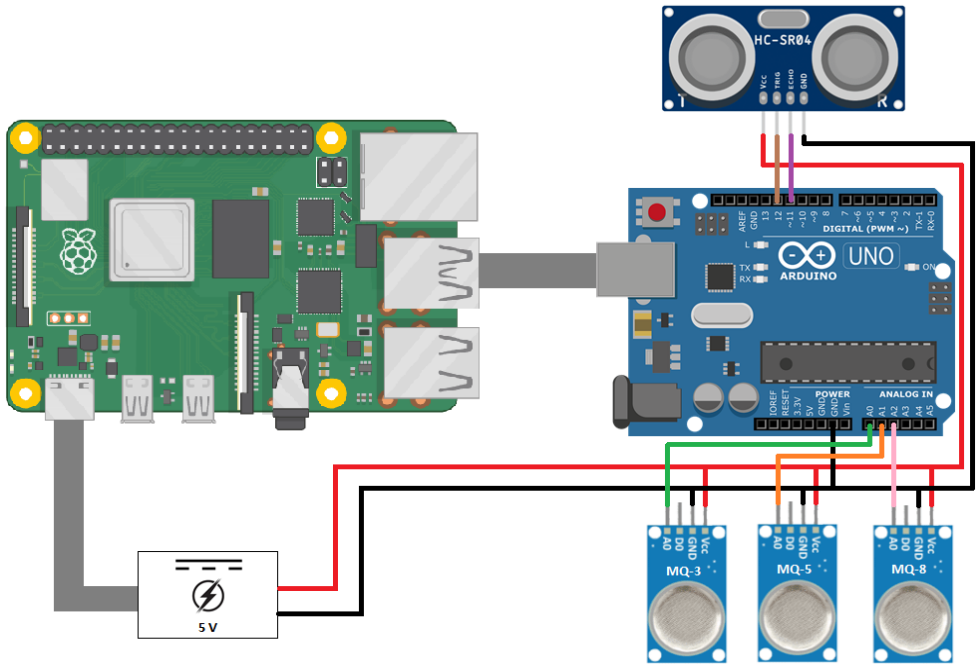

Figure 3. System Schematic

1) **Sensors:** There are four sensors used in this system, three of them are gas sensors and are used for the detection of different gases and the fourth is an ultrasonic distance sensor. We discuss the sensors used in this section.

i. MQx Gas Sensors

Three gas sensors from the MQx line of gas sensors have been used in this project. These are the MQ8, MQ5 and MQ3 sensors. The MQ8 sensor is used to detect hydrogen, even though it is sensitive to other gases such as CO, alcohol vapors. The MQ5 is sensitive to multiple gases and can be used to detect H2, LPG, CH4, CO, and Alcohol; and is generally used to detect leakage of inflammable gases. Lastly, the MQ3 sensor is primarily used as an alcohol detection sensor as it has high sensitivity to it. It also is slightly sensitive to Benzine.

All these sensors are analog semiconductor sensors that have a similar construction and consist of a heater coil, ceramic tube, an electrode and a gas sensing layer made up of Tin Oxide (SnO_2). The heater is important to create the conditions necessary for using the sensor.

The conductivity of the sensing layer is low in clean air and increases as the sensing element is exposed to the relevant gas. This change of resistance is translated into a voltage by means of a voltage divider formed by the heater+filament resistance and the load resistance. This can be used to determine the change in gas concentration or the gas concentration absolutely after calibration.

ii. Ultrasonic Distance Sensor

The HC-SR04 ultrasonic sensor uses ultrasonic ranging to detect distance between itself and an object up to a range of 40cm. This sensor provides a convenient way to detect obstacles for embedded projects due to its ease of use and low size. It consists of a transmitter, a receiver, pulses are sent through the transmitter and its echo from an object is detected through the receiver. The time difference between the transmission and reception of the pulses is used to calculate the distance of the sensor from the object producing the echo.

2) **Arduino**: An Arduino controller is used as a secondary controller that acts as the interface with the Raspberry Pi and the sensors. The Arduino board used in this project is the Arduino Uno board. It consists of an Atmega328 microcontroller having 32KB of flash memory, six analog inputs and fourteen digital port pins. Consisting of a 10-bit successive approximation Analog to Digital Converter, the Arduino can be used to read off analog voltages using its analog pins.

   The Arduino, being an open source platform provides for a host of resources to interface and use different sensors, both analog and digital to develop embedded systems for a variety of applications, therefore, the Arduino is well suited for use as the directly connected processor to sensors when developing measurement systems.

3) **Raspberry Pi:** The Raspberry Pi is the main controller for the system. We use two Raspberry Pi's, one at the client side which acts as the main controller and one as the server. The Raspberry Pi is a self-contained computer consisting of an ARM Cortex A processor, RAM, a host of peripheral interfaces (USB, Ethernet, Wi-Fi, Bluetooth, HDMI, Audio) and general-purpose Input/output ports. The Raspberry runs a Linux distribution. In our system, the Raspberry Pi is used to read off data from the Arduino and post it to the server.

C. Software

Below is the flowchart showing the logic of the Server and its interaction with Client.

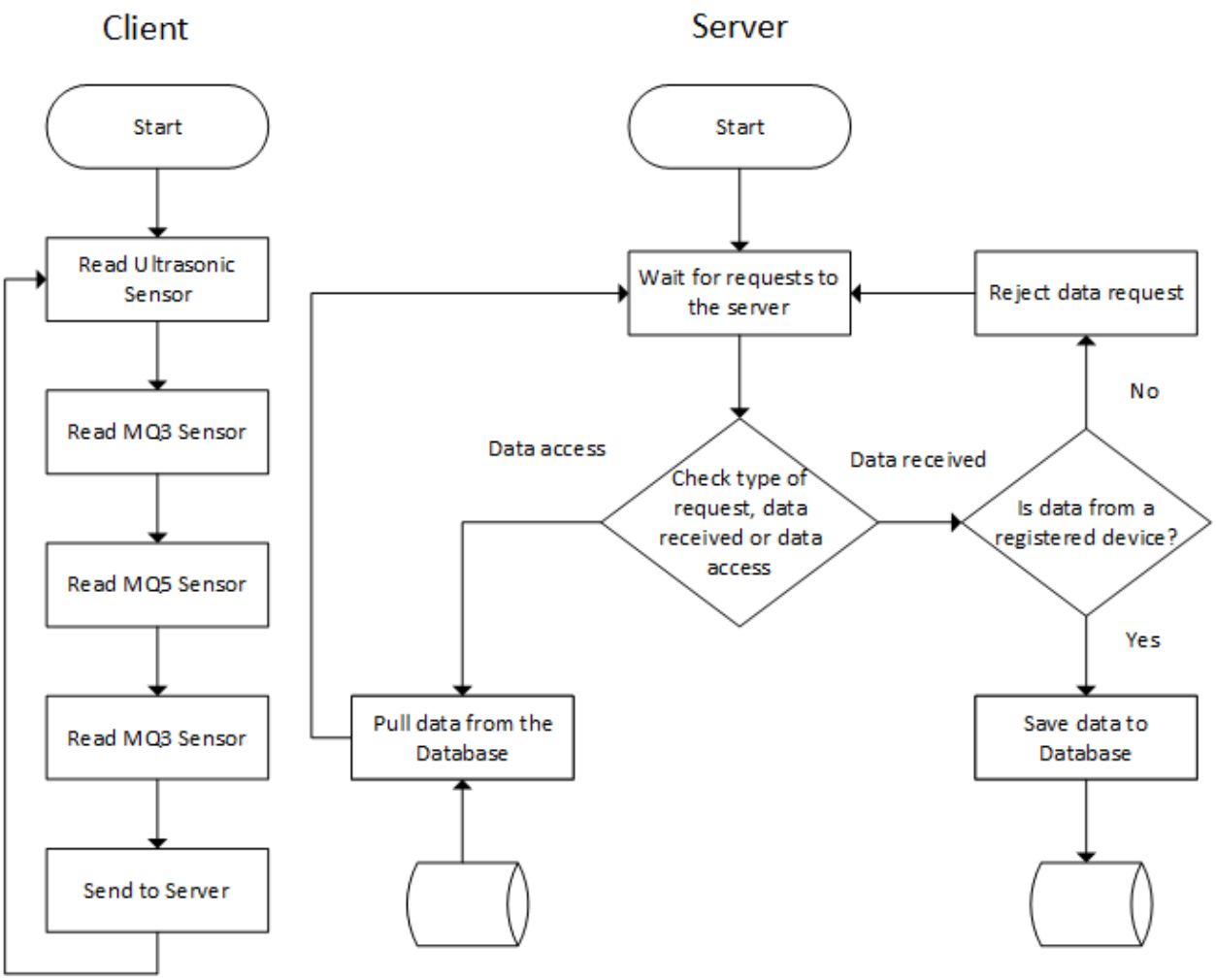

Figure 4. Software implementation

1) Python library

First, the requirements for the UI application will be listed. These are all Python packages, some come installed with base Python and others can be installed using a Python package manager:

> *Sys, subprocess, datetime, logging, Flask, render_template, request, abort, jsonify and sqlite3.*

Most of these packages are used in their basic use cases. The only package used in a custom fashion is Flask which is used to design website routing.

2) Flask API

A Flask web server was created from the Flask module in python script (app.py) representing the web application.

```
app = FlaskAPI(__name__)
```

Add pages to the website by adding more routes.

```
@app.route("/", methods=['GET', 'POST'])

def main():
    return render_template('main.html')
```

The task of designing a web service or API that adheres to the REST guidelines then becomes an exercise in identifying the resources that will be exposed and how they will be affected by the different request methods. The REST architecture was originally designed to fit the HTTP protocol that the world wide web uses.

Tasks resource of Multi-Sensor Web App will use HTTP methods as follows:

Table 1. HTTP methods

HTTP Method	URI	Action
GET	http://[hostname]/task	Retrieve list of tasks
POST	http://[hostname]/task	Create a new task

A POST request on the user's resource would represent a new user registering for the service. A GET request would return user information back to the client.

3) Static files and templates

The basic web application is functional but minimal, so in order to make the website look nicer, we must add HTML and CSS. The Flask Framework looks for HTML files in a folder called templates; and in static folder, Flask Framework stores CSS, JavaScript, images and other necessary files. The structure of the folders can be seen from Figure 7.

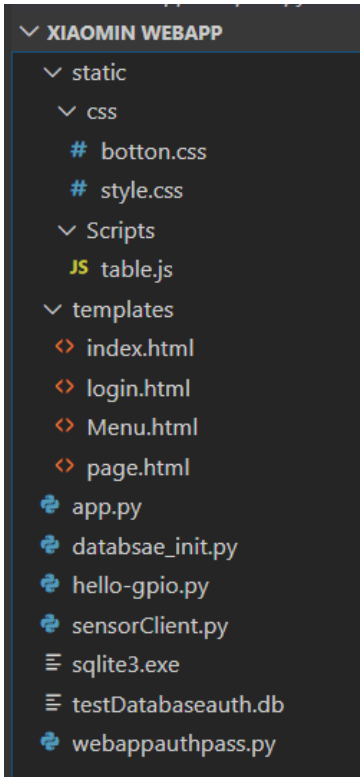

Figure 7. Actual directory layout

Images, CSS files, JavaScript files, etc. are stored in static folder next to the main application file (app.py) and in the HTML skeleton you refer to the files as /static/…. I put the css file in a subdirectory called css and the JavaScript in a subdirectory called Scripts.

Flask API utilizes blueprints for managing browsable pages' template path. I put templates for every page under the folder templates.

4) Bootstrap

In this project I used Bootstrap as my toolkit for developing with HTML, CSS, and JS. In detail, in order to load CSS, I included stylesheet listed below:

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstr
ap.min.css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
```

And many of the components require the use of JavaScript to function. Specifically, requires jQuery and Popper.js. Thus, I included following scripts:

```
<script src="https://code.jquery.com/jquery-1.9.1.min.js"
integrity="sha384-
```

```
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
<script
src="//cdn.datatables.net/1.10.12/js/jquery.dataTables.min.js"></scr
ipt>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.m
in.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap
.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
```

With Bootstrap, is very easy add filtering feature on detail page. For each table row, in related html file, add feature:

```
<th onclick="sortTable(1)" style="cursor:crosshair">{{ key }}</th>
```

5) Client

The post request sent by client needs four fields to be filled correctly. In the headers field, set the Content-Type to `application/json; charset=utf-8`, set the accept type to `text/plain`.

The root URL to access this service:

`http://[hostname]/sensors/api/endpoint/`

The authentication field needs the device ID and according password pre-defined by master user.

Finally, the data field will be streaming data from Multi-Sensor Device.

Appendices

Client Contact Information:

Dr. Daniel Sierrasosa
Email: d.sierrasosa@louisville.edu
Office: Duthie Center for Engineering (Room 237) University of Louisville

Developer Contact Information:

Xiaomin Heil
Email: x0zhan35@louisville.edu

Source Code:

The source code for Multi-Sensor Web Server can be found on GitHub at the following repository: https://github.com/xiaomin0724/MultiSensor.git

This repository is private as the source code need not be publicly available, to gain access to the repository please contact Dr. Sierrasosa with the contact information located above.