A Real-time Remote Monitoring Web Based Application Using Telemetry Technology

May 8, 2017/ by Zhengpu Chen

Table of Contents

[1 Introduction 2](#_Toc482302344)

[2 Hardware Requirements 2](#_Toc482302345)

[3 Hardware Configuration 3](#_Toc482302346)

[4 Software Requirements 4](#_Toc482302347)

[5 Yoctopuce Module Testing and Configuration 5](#_Toc482302348)

[6 Web Application Development 6](#_Toc482302349)

[6.1 Server Setup 6](#_Toc482302350)

[6.2 Flask Application Setup 7](#_Toc482302351)

[6.3 Data Collection 7](#_Toc482302352)

[6.4 Website Header 7](#_Toc482302353)

[6.5 Index Page 8](#_Toc482302354)

[6.6 Registration System 8](#_Toc482302355)

[6.7 Decorators 12](#_Toc482302356)

[6.8 Line Chart Plot 13](#_Toc482302357)

[6.9 Mapping Data with Geolocation 16](#_Toc482302358)

[6.10 Raw data Display and Download 18](#_Toc482302359)

[6.11 Error Handler 20](#_Toc482302360)

[7 Future Work 20](#_Toc482302361)

# Introduction

Yoctocpuce (<http://www.yoctopuce.com/>) products intend to enable developers to create simple systems to automate daily tasks or implement innovation ideas. There are several advantages of Yoctopuce products comparing to its counterparts. The products are designed to be easily and quickly implement. The products are essentially USB peripherals, which are easy to connect with computers or other devices with USB connection available without the using of a soldered connection. The products are small and have a low energy cost, which suits well with automation and remote control.

A supporting web base application is developed for the Yoctopuce devices to realize data collecting, graphing and mapping functions. The main purpose of developing the application is to produce a competitive alternative for the commercially available web tools. The designed application provides researchers flexibility collecting and analyzing the data and further lower the cost of the telemetry system. The application being developed is a functional application. This means that the prototype mainly focuses on realizing the core functions, instead of developing a fully-fledged application with a well-polished user interface.

# Hardware Requirements

In this project, a YoctoHub-Wireless-g module is used to enable the Internet connection. The YoctoHub-Wireless-g is a wireless-enabled module that can host three Yoctopuce modules to access them remotely through a Wi-Fi network (802.11g). Through the network, the device posts data on a web server using the HTTP callback. It can be powered either by a USB A to Micro-B cable and a regular phone charger, or a 5V battery pack. A built-in clock timer enables to put the device into low power deep sleep mode and wake up automatically at scheduled time. The power consumption may be reduced to 15 µA while sleeping, which is crucial for projects using solar panels or battery as power supply method. Besides, the device is provided with an articulated antenna (9cm, with RP-SMA connector) and a connection cable (uFL to RP-SMA), which make it possible to a further access point.

Up to three Yoctopuce modules, like Yocto-GPS, Yocto-Meteo, and Yocto-Light-V3, can be connected to a YoctoHub-Wireless-g using USB Micro-B to Micro-B cables. More information about Yoctopuce modules can be found on the company’s website: <http://www.yoctopuce.com/>. YoctoHub-Wireless-g is able to collect the data measured by the modules, and send the data to the cloud. The system is scalable by simply includes more Yoctopuce modules. YoctoHub-Shield extension can be used to connect additional modules with the network hub, if the system has more than three modules, as long as the total power consumption of the modules is less than 2A.

For testing purpose, a Yocto-GPS, a Yocto-Meteo, and a Yocto-Light-V3 are used when building the web application. The plots listed in the following sections use the data collected by these three sensors. The modules used in the project are shown in Figure 2.1.

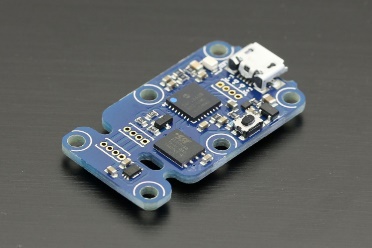
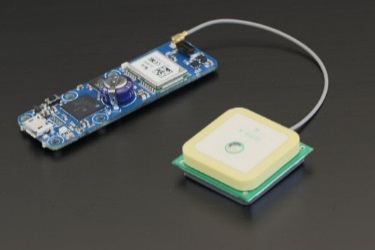
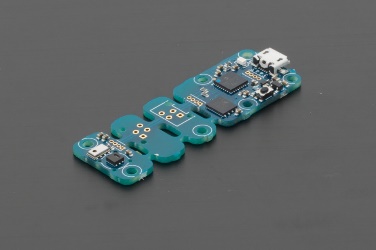


Figure 2.1: Yoctopuce Modules (From Left to Right: YoctoHub-Wireless-g, Yocto-Meteo, Yocto-GPS, and Yocto-Light-V3)

Based on the hardware selection described above, Table 2.1 is created as a summarization. The Hardware ID, description, function of each component are listed in the table.

Table 2.1: Summarized Description of the Hardware

|  |  |  |
| --- | --- | --- |
| Summarized Description of the Hardware | |  |
| Hardware ID | Description | Function |
| YoctoHub-Wireless-g | Wireless-enabled module | Host Yoctopuce modules to access them remotely through a Wi-Fi network |
| Yocto-Meteo | USB humidity, pressure, and temperature sensor | Record humidity, pressure, and temperature |
| Yocto-Light-V3 | USB ambient light sensor | Record light intensity |
| Yocto-GPS | 32 channels receiver for GNSS signals | Record latitude, longitude, altitude, and speed |
| USBA0100 | USB Cable - A to Micro B | Provide connection between sensor hub and power source |
| OTGBB100 | USB Cable - MicroB to MicroB | Provide connection between sensor hub and sensors |

# Hardware Configuration

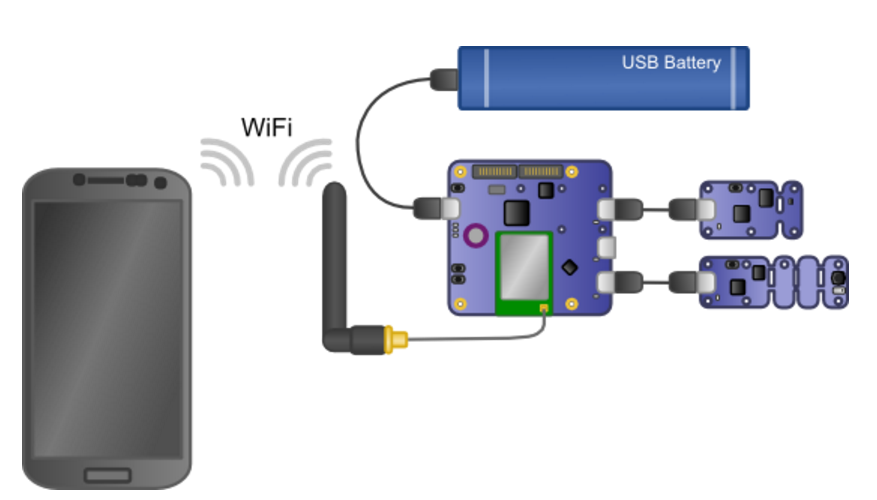
One possible hardware configuration is depicted in Figure 2.2 (<http://www.yoctopuce.com/EN/article/how-to-use-yoctopuce-modules-in-the-middle-of-nowhere>). Sensors, sensor hub and power source are connected by proper USB cables. The sensors conduct the measurement periodically and send data to the YoctoHub-Wireless module, which commonly connects to a Wi-Fi hotspot created by the smartphone. The smartphone enables the wireless communication between the sensor station and the web server through GPRS/3G/4G depending on its own setting. The YoctoHub-Wireless module and smartphone can be replaced by a YoctoHub-GSM-3G module, which can connect to 3G GSM cellular network.

Figure 2.2: Hardware Configuration

# Software Requirements

VirtualHub (<http://www.yoctopuce.com/EN/products/not-published/virtualhub>), developed by Yoctopuce Module manufacturer, is a software to manage USB modules. The software helps developers to conduct the configuration and testing for USB modules. Data transmitted by the YoctoHub-Wireless module to the web server are saved in a TinyDB database (<https://tinydb.readthedocs.io/en/latest/>). An interactive web application for data retrieval, analysis, manipulation, and visualization is developed using Flask (a Python microframework, <http://flask.pocoo.org/>) and Bokeh (a Python interactive visualization library, <http://bokeh.pydata.org/en/latest/>) on the web server. The application is the host on an Amazon Elastic Compute Cloud (Amazon EC2) server (<https://aws.amazon.com/>). The service is chosen for its low setup price, high-security rate, resizable compute capacity and good reputation. To enable the interaction with the server, a secure shell (SSH) client and a remote file managing client are required. For this project, PuTTY (<http://www.putty.org/>) is chosen for SSH client and WinSCP (<https://winscp.net/eng/download.php>) is chosen for file management. The web page content is produced through the use of HTML 5, Bootstrap (<https://getbootstrap.com/>) and JavaScript. Leaflet, an open-source JavaScript library, helps the development of mapping geo data. A registration system is built based on MySQL database (<https://www.mysql.com/>). The web application provides registrated users a handy tool to easily monitor the sensor measurements. The users get the service they wanted via sending requests to the web application. The web application returns information back instantly upon the requests. A summarization of the softwares and libraries requirements is listed in Figure 3.1.

Figure 3.1: Software and libraries Requirement

# Yoctopuce Module Testing and Configuration

Yoctopuce Module can be tested and configured using VirtualHub. A detailed documentation: *Configuring your Yoctopuce modules with the VirtualHub* (<http://www.yoctopuce.com/EN/article/configuring-your-yoctopuce-modules-with-the-virtualhub>) is provided by Yoctopuce. The documentation describes a step by step process of configuring Yoctopuce sensor module.

The process of configuring a sensor hub is slightly different from configuring a sensor. A general method can be found in the documentation introduced the sensor hub ([http://www.yoctopuce.com/EN/products/‌yoctohub-wireles‌‌‌‌‍‍‍‑‌s-g/doc/YHUBWLN3.usermanual.html#CHAP3](http://www.yoctopuce.com/EN/products/‌yoctohub-wireles‌‌‌‌‍‍‍‌s-g/doc/YHUBWLN3.usermanual.html#CHAP3)). The configuration of the sensor hub’s outgoing callbacks used in this project are described below.

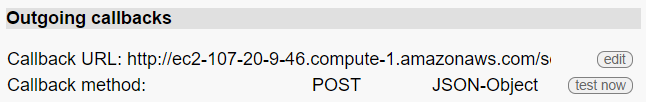
First, run the VirtualHub on the computer and connect it to the YoctoHub-Wireless-g through USB port. Launch the URL (usually it is <http://127.0.0.1:4444>) of the VirtualHub and click on the configure button corresponding to the YoctoHub-Wireless-g. After configuring the “Wake-up Scheduler” and “Network Configuration” section, click on the “edit” button on the left side of “Callback URL” in the “Outgoing Callbacks” section as shown in Figure 5.1.

Figure 5.1: Outgoing Callbacks Section

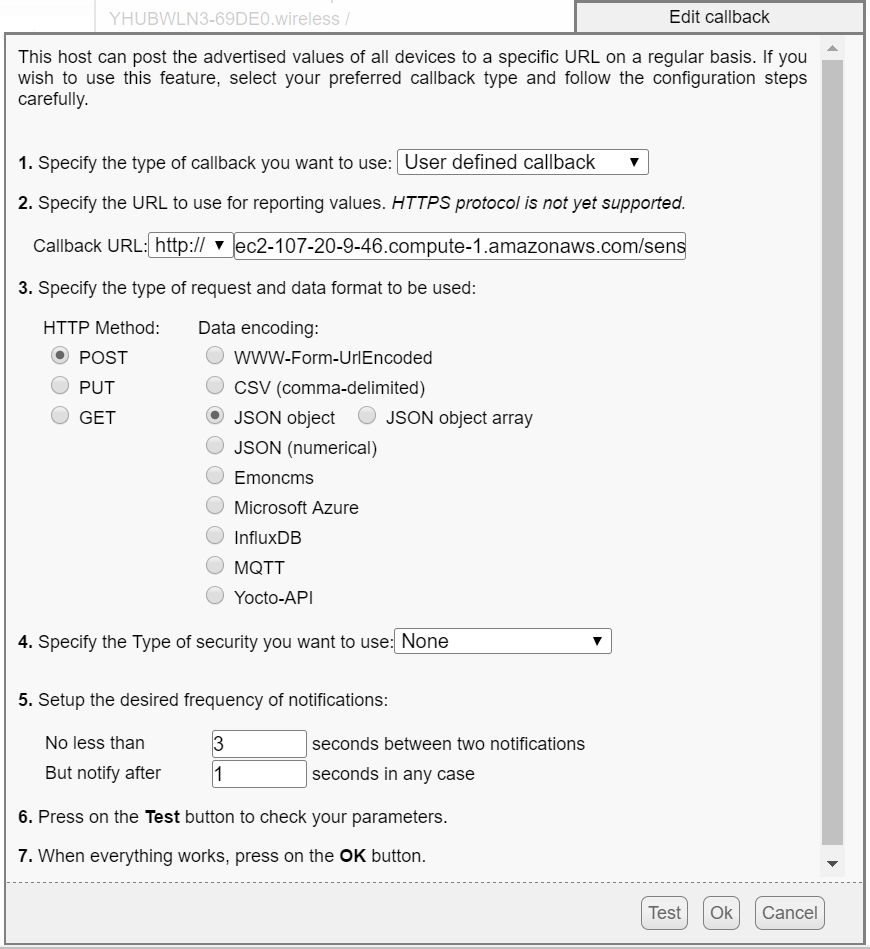
 Then, a window named “Edit Callback” will pop up as shown in Figure 5.2. Select the type of callback as “User Defined callback”. Type in the URL which will receive the transmitted data. Specify the type of the request as “POST” and data format as “JSON object”. Setup the frequency of the notifications as the project desired. After that, press on the “Test” button to check your parameters. Finally, press on the “OK” button to confirm the setup.

Figure 5.2: Edit Callback

# Web Application Development

This section explains the scripts of the web application following the development process. Example layout of each functions and corresponding section of code in the flaskapp,py file are provided to help the understanding. The tutorials of the tools and libraries applied in the application are also given. The codes of the HTML template files are not included in this document. However, the template(s) used for each function are indicated in the parentheses after the first mention of the function.

## Server Setup

A step by step instructions for running a Flask application on an EC2 instance can be found in the following guide: <http://www.datasciencebytes.com/bytes/2015/02/24/running-a-flask-app-on-aws-ec2/>. The following links provide the instructions on connecting an Amazon EC2 instance with PuTTY and WinSCP: <https://www.youtube.com/watch?v=mJaHARCfcA0>, <https://www.youtube.com/watch?v=nSX4GjnmGlU>. A general tutorial for using PuTTY and WinSCP and setting up a web server is provided by pythonprogramming.net: <https://pythonprogramming.net/basic-flask-website-tutorial/>.

## Flask Application Setup

A tutorial for Flask is provided on pythonprogramming.net: <https://pythonprogramming.net/practical-flask-introduction/>. For this project, an initialization python file named “flaskapp.py” is created to store routes and corresponding functions. The libraries used to develop the application are listed in requiements.txt file.

## Data Collection

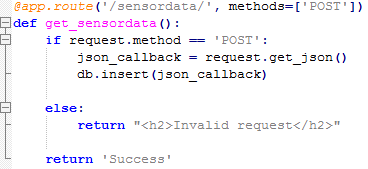
The first function “get\_sensordata” the application realized is data collection. For this function, the application saves the callback data in the format of JSON object in a TinyDB database. The code is shown in Figure 6.2.1. Test the outgoing callback in VirtualHub. A “Success” should show in the popup window.

Figure 6.2.1 Data Collection Code in flaskapp.py

## Website Header

A website header (templates/header.html) runs across the top of the page and appears on every page of the web application. By clicking on the Purdue loge, the user will be directed to the index page (templates/homepage.html). The left side of the header indicates the login status. Before login, the header will show login and sign up. By clicking on the corresponding icon, users will be directed to login page (templates/login.html) and sign up page (templates/register.html). After login, the login and sign up icon will be replaced by the username and logout icon. When users click on the username icon, they will be directed to user information page (templates/user\_info.html). The user will be logged out, if they click on logout icon.

Figure 6.4.1: Header Showing in Web Browser (Top: Before log in; Bottom: After log in)



## Index Page

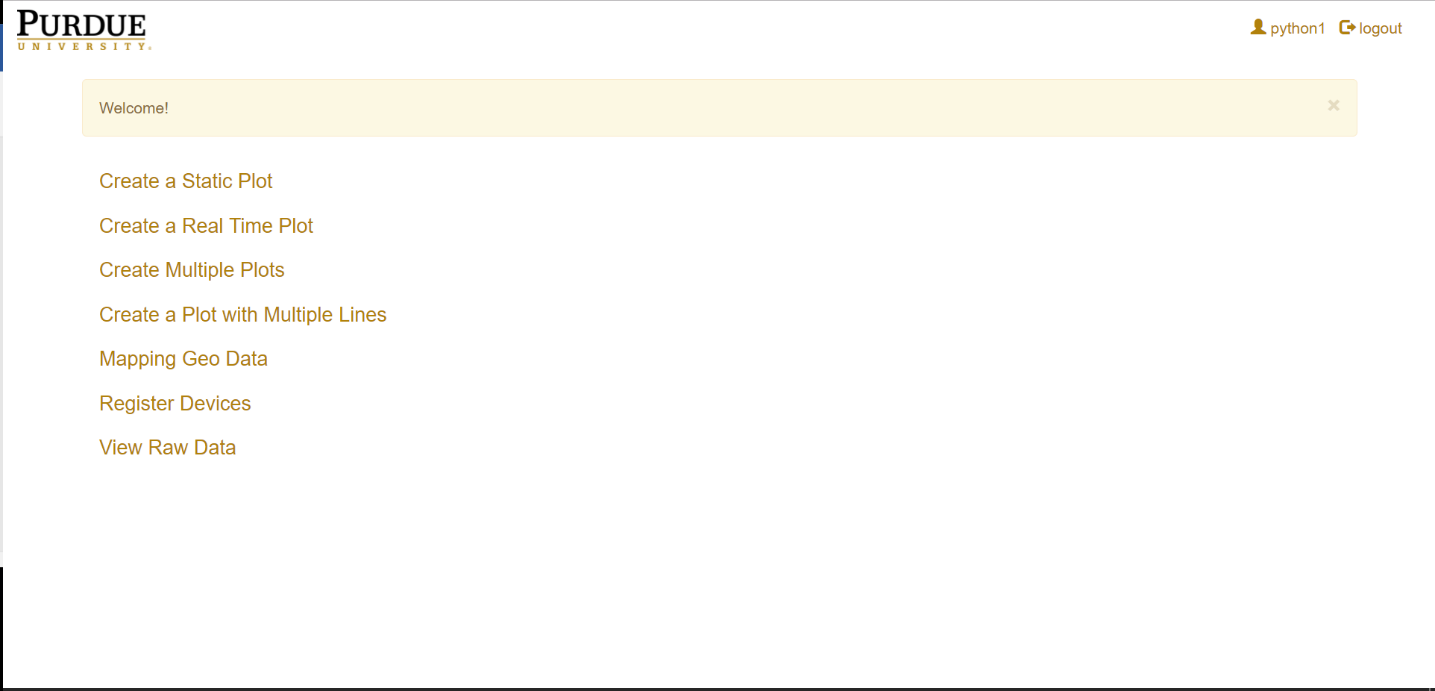
An index page (Figure 6.4.1) is developed to provide a navigation to different functions of the application. “homepage” function (Figure 6.4.2) in the flaskapp.py file format the index page. Instead of referencing hard URLs, the program reference elements within a dictionary stored in content\_management.py file (Figure 6.4.3). In this dynamic way, the process of adding, removing, editing function links will be simplified.

Figure 6.5.1: Index Page Showing in Web Browser

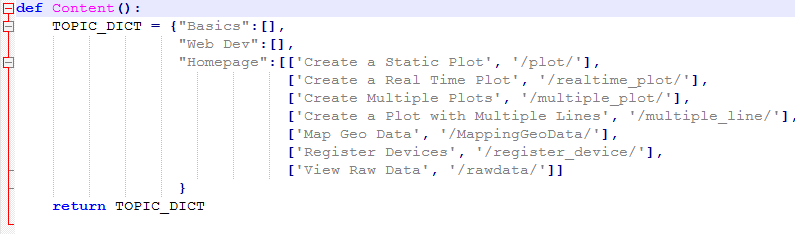
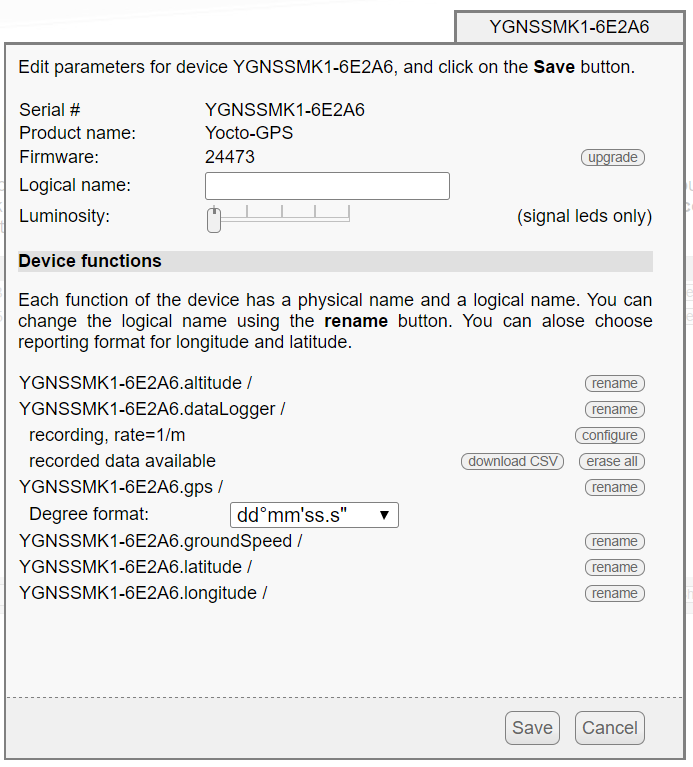
*Figure 6.5.2: Index Page Code in flaskapp.py*

Figure 6.5.3: content\_management.py Code

## Registration System

Before the usage of the application, registration is required for new users of application. The purpose of including a registration system is to save the device information for users. After the registration, an account is created for the user, and the users can to login to the account. A logged in user is able to register devices. To register a device, the serial number of the device need to be first acquired through VirtualHub (Figure 6.5.1) or Yoctopuce API. By clicking on “Register Devices” in the index page, users will be directed to the register device form. The registration form (templates/register.html), the login form (templates/login.html), the register device form (templates/register\_device.html) are shown in Figure 6.5.2. Plot functions of the application will not be activated if no device is registered. The user and device information is stored in a MySQL database, the information can be retrieved and displayed by clicking on the user account icon located at the upper right of the web page. A sample user information page is shown in Figure 6.6.3.



*Figure 6.6.1: Acquire Device Serial Number in VirtualHub*

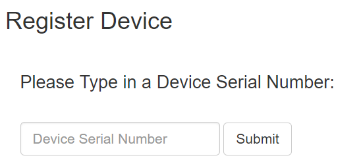
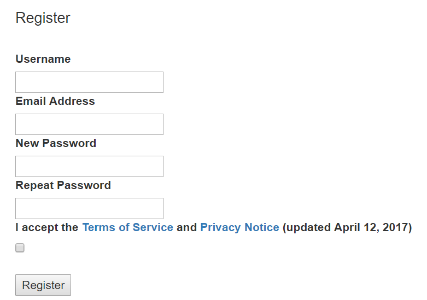
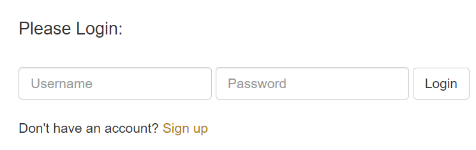


Figure 6.6.2: Registration and Login Forms (From Left to Right: Registration Form, Login Form, and Register Device Form)

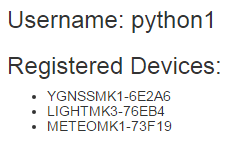


Figure 6.6.3: Sample User Information Page

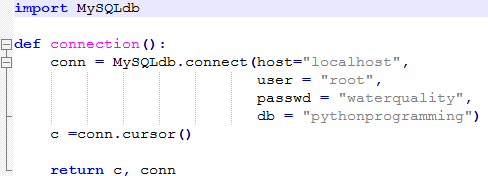
 A MySQL database is set up to store the user information. The method of creating database and table are introduced in: <https://pythonprogramming.net/mysql-database-with-flask-tutorial/>. Figure 6.6.4 shows the code for MySQL database connection.

Figure 6.6.4: Code for MySQL Database Connection

Figure 6.6.5, 6.6.6, 6.6.7, 6.6.8, and 6.6.9 show the corresponding code in flaskapp.py of registration, login, device registration, logout, and user information display functions.

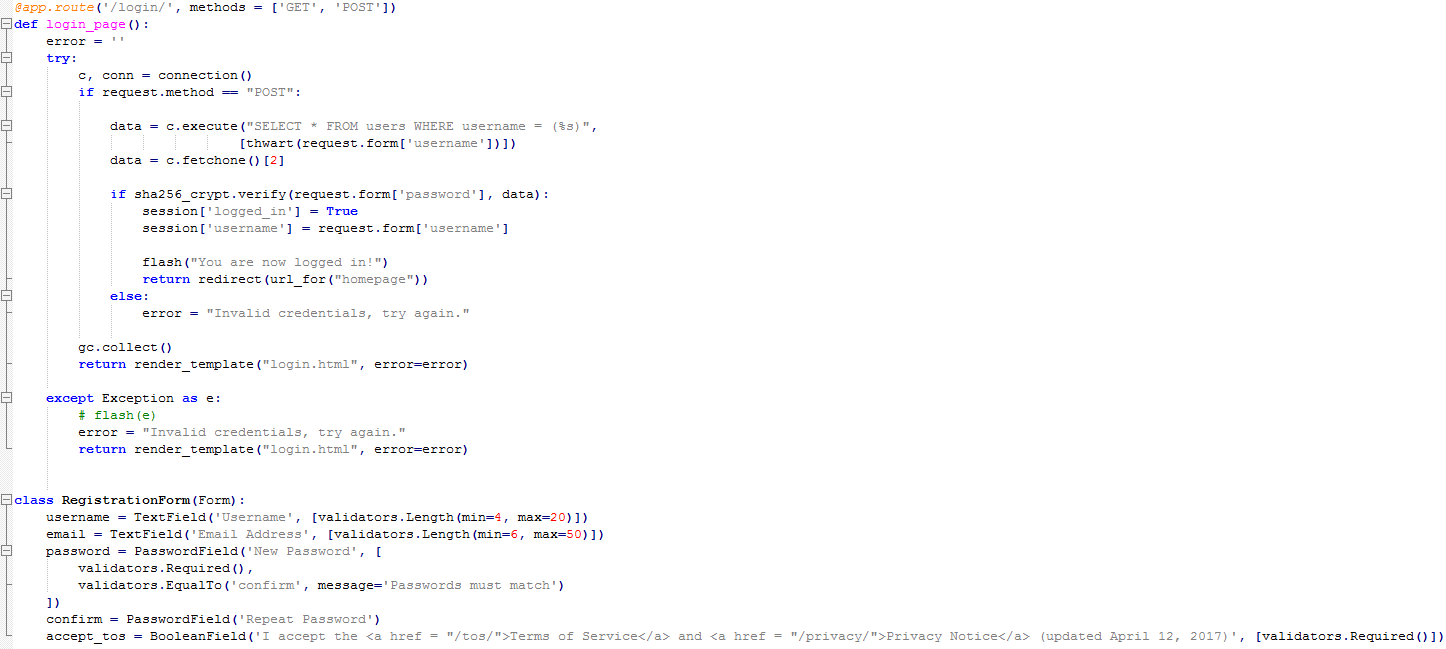
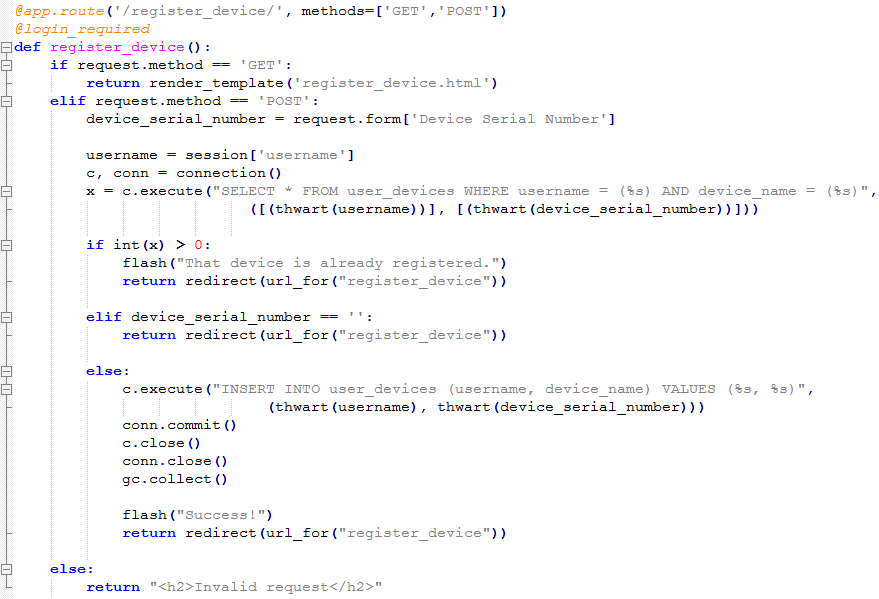
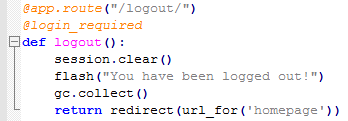
Figure 6.6.5: Registration Function in flaskapp.py

Figure 6.6.6: Login Function in flaskapp.py



Figure 6.6.7: Device Registration Function in flaskapp.py

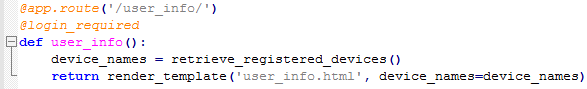
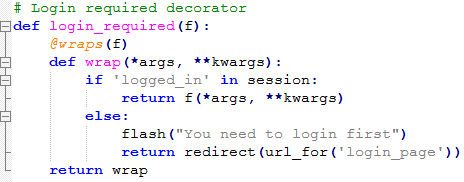
 Figure 6.6.8: Logout Function in flaskapp.py

Figure 6.6.9: User Information Display Function in flaskapp.py

## Decorators

As mentioned above, login and device registration are required before the use of the application. Decorators are created to realize the login and device registration requirements. New users (who have not logged in or registered any device) will be redirected to login and device registration page if they try to use any of the function. The code of decorators in flaskapp.py are provided in Figure 6.7.1 and 6.7.2.

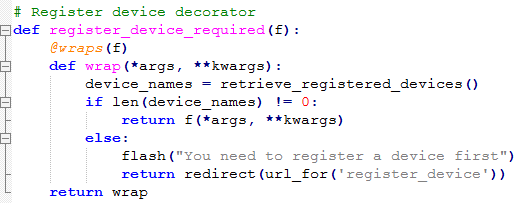
Figure 6.7.1: Login Required Decorator in flaskapp.py

Figure 6.7.2: Register Devices Required Decorator in flaskapp.py

## Line Chart Plot

With at least one device registered on the account, the user can visualize historical data in creating static plots or streaming real-time data in a real-time plot. Historical measurements for a single parameter can be graphed in one plot (Figure 6.8.1) as well as multiple plots (Figure 6.8.2). Multiple plots function provides user conveniently to make comparisons between different parameters’ data. Data collecting from different sensors can also be created on one plot. The function can be used when comparing same types of parameters.

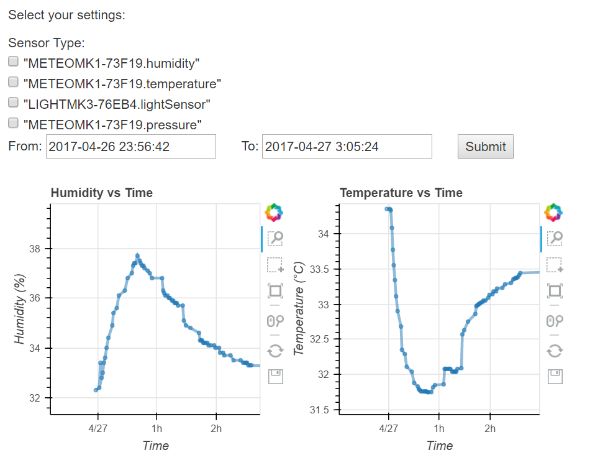
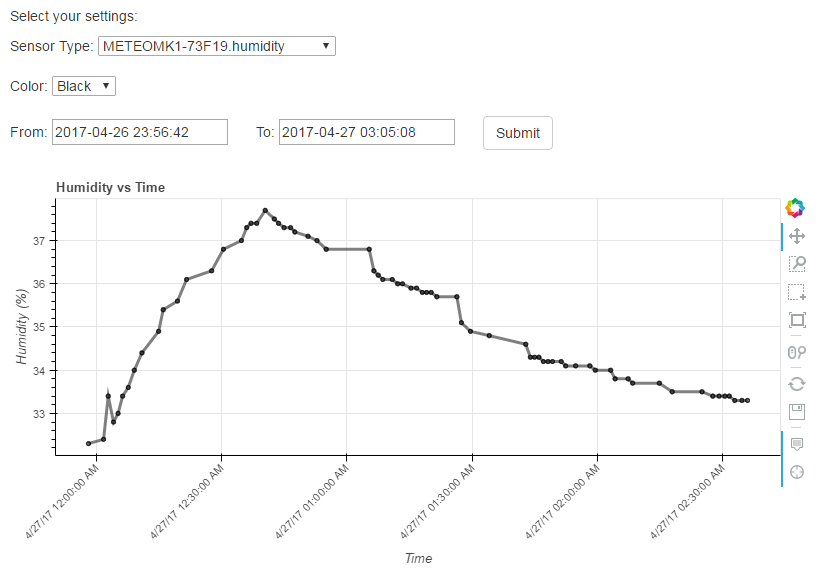


Figure 6.8.1: An Example of a Static Single Plot Figure 6.8.2: An Example of a Static Multiple Plots

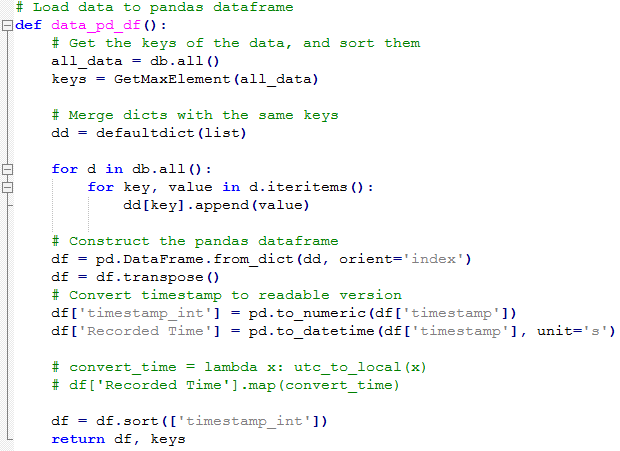
 Before generating plots, the data need to be retrieved from TinyDB database, and processed to a desired form. First the data is loaded to a pandas dataframe using data\_pd\_df function (Figure 6.8.3). The device serial names registered by the user are retrieved from MySQL database via retrieve\_registered\_devices function (Figure 6.8.4). Then, desired keys and dataframe columns are sorted out based on the device serial names. Undesired columns and keys, like datalogger, hubPort, and realTimeClock, are dropped using drop\_columns and drop\_keys functions (Figure 6.8.5).

Figure 6.8.3: Load Data to a Pandas Dataframe

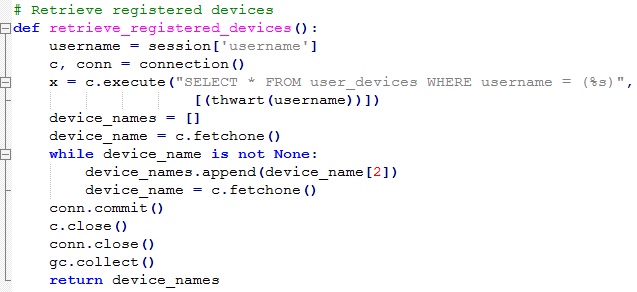
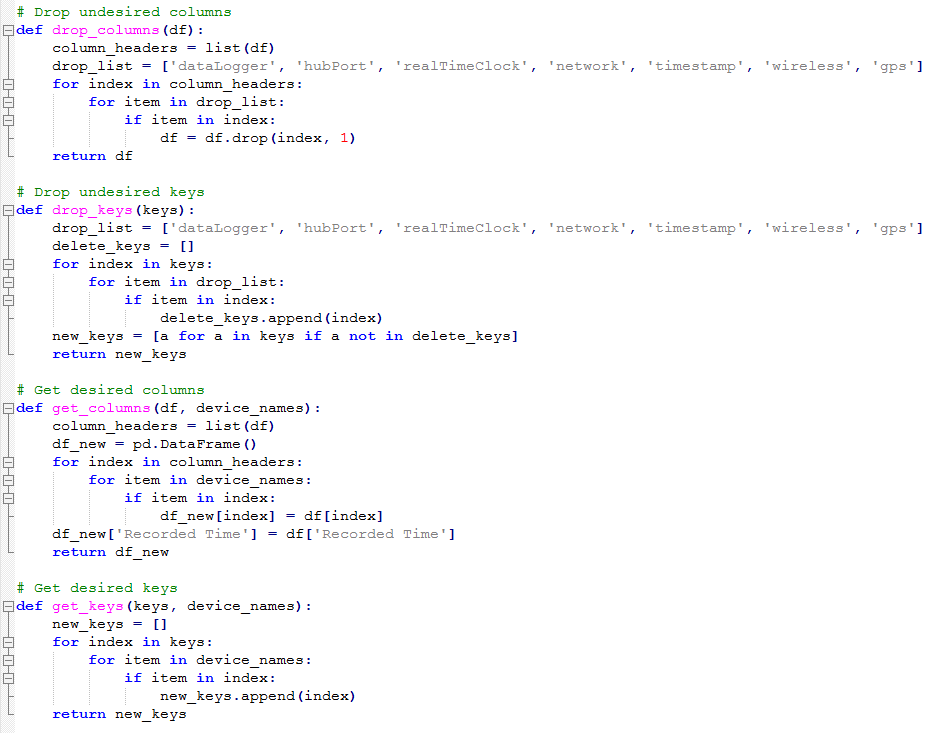
Figure 6.8.4: Retrieve Registered Device Serial Names

Figure 6.8.5: Drop and Get Desired Keys and Columns

The plot can be made after the preprocessing of the data. The detailed codes of static plot (templates/plot.html), multiple plots (templates/multiple\_plot.html), plot with multiple lines (templates/multiple\_line.html) and real-time plot (templates/create\_realtime\_plot.html and templates/realtime\_plot.html) can be checked in flaskapp.py file under corresponding functions: plot, multiple\_plot, multiple\_line, realtime\_plot, and update\_plot.

## Mapping Data with Geolocation

By using the Yocto-GPS with other modules, data can be mapped with geolocation. A JavaScript library — Leaflet are used to create the interactive maps. Before using this function, a GPS module need to be registered. Yocto-GPS provides each data package callback with a position information. Markers are placed on the map where data being measured. By clicking on the makers, the time when the data is measured and the measurements of corresponding sensor modules are shown in the popup window. An example of mapping data with geolocation (templates/ MappingGeoDataJS) is shown in Figure 6.9.1, and the corresponding code is given in Figure 6.9.2.

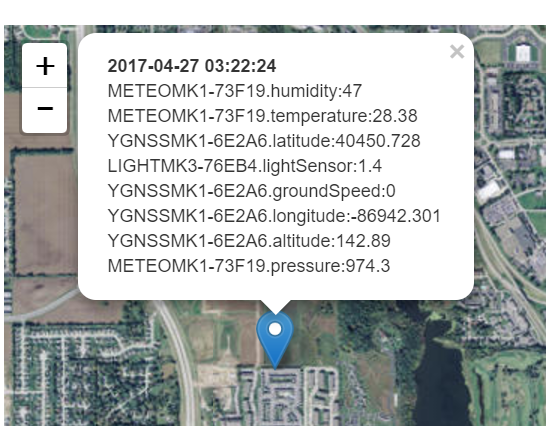


Figure 6.9.1: An Example of Mapping Data with Geolocation

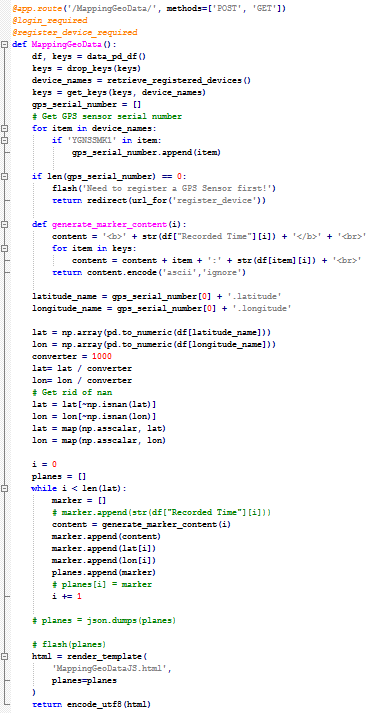


Figure 6.9.2: Code of Mapping Data with Geolocation in flaskapp.py

## Raw data Display and Download

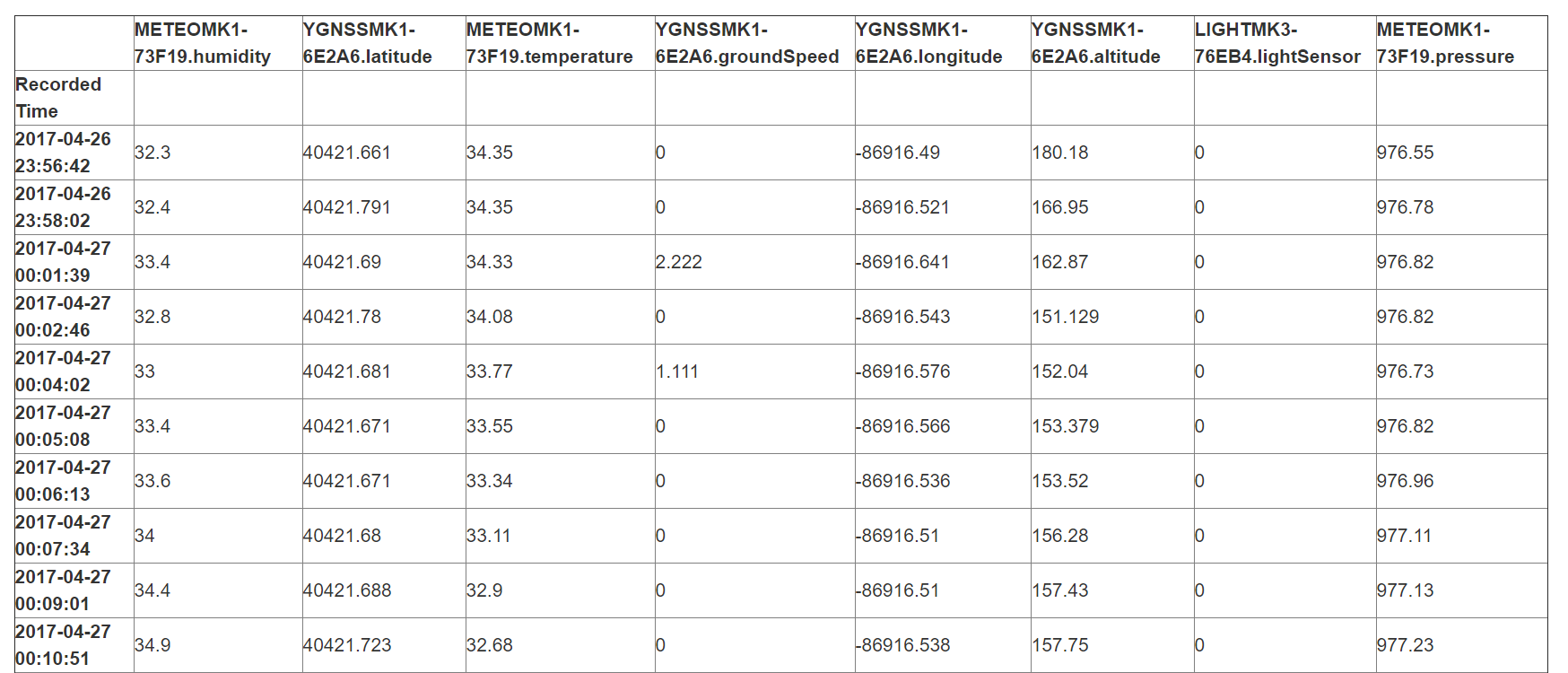
The raw data transmitting from the sensor hub to the web application is stored in a TinyDB database. The raw data can be displayed on a web page as a table, and downloaded as a CSV file for future data sharing and analysis. An example of a raw data table is shown in Figure 6.10.1, and the corresponding codes is shown in Figure 6.10.2 and 6.10.3.

Figure 6.10.1: An Example of a Raw Data Table

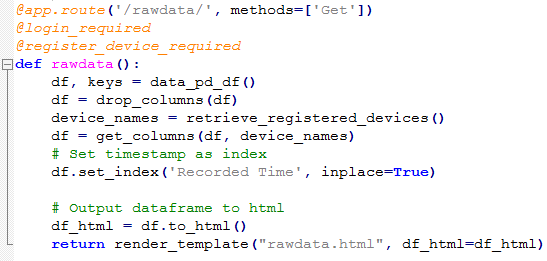
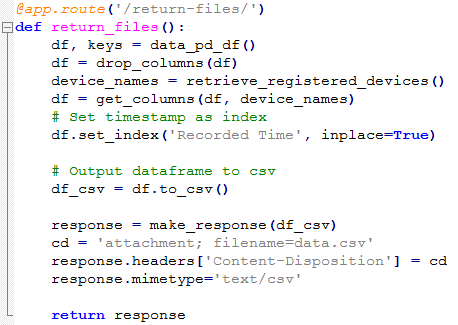
Figure 6.10.2: Raw Data Display Function in flaskapp.py

Figure 6.10.3: Raw Data Download Function in flaskapp.py

## Error Handler

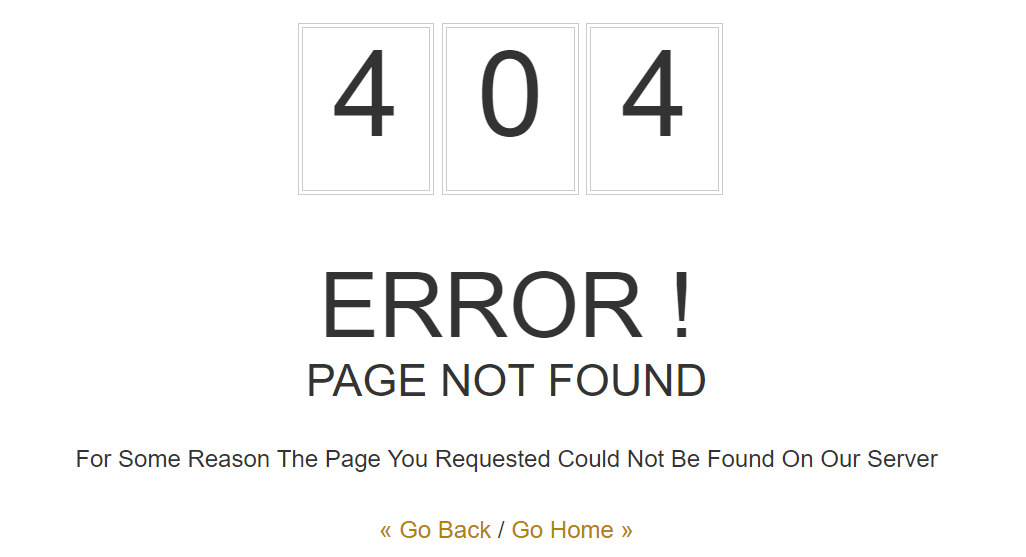
A 404 page (templates/404.html) is created to handle “the page not found” error. A sample 404 page and the corresponding code in flaskapp.py are shown in Figure 6.11.1 and 6.11.2

Figure 6.11.1: 404 Error Page

Figure 6.11.2: Error Handler Function in flaskapp.py

# Future Work

This project initials the development of the web-based application for the telemetry system, further improvements can be made for the application. As a functional application, the user interface can be refined. First, more functions can be added to the user registration system. Detailed user information, like registration time, education background and purpose of using the application, can be added to the user information page. A web page showing the working condition of the sensor, like the strength of the wireless connection, on or off of the data logger, sensor name connected to each port of the sensor hub, can be developed. Most of the Yoctopuce sensor modules have a built-in data logger. The data logger is capable of storing ongoing data automatically, without requiring a connection to a computer or network (Yoctopuce, 2016). By properly configuring the sensor and modifying the existed script, the sensor hub should be able to send the measurements stored in the data logger to the cloud. With such function, the system will be more competent working in the rural area where the wireless or cellular network is poor. More data visualization method can be developed, like dashboard and meter. Different methods of realizing multiple lines plot, multiple graph plots, and real-time plot may be implemented to optimize the performance of the application. Data sharing functions through email, social media or text message may be added. Sending notification is another function worth implementing. For example, when the value of certain parameter testing by the sensor is over the preset warning line, notification should be sent to user promptly. Besides, data analysis may be another area to probe into. The application may be able to calculate the statistical parameters, like mean, variance and median or even make predictions on the basis of historical data. Knowing the basic principle of web application development, specific functions can be developed based on the usage of the system.