Project 3: HTTP Client for Internet of Things Monitoring & Control

Objective:

In this project you will build on your experience in Project 1 (using Python socket programming) and Labs 7-9 (using an ESP8266-based Internet of Things (IoT) kit). You will develop an HTTP web client that accesses an ESP8266 web server connected to IoT hardware. Your program will allow a user running the HTTP client to monitor and control that hardware. You will demonstrate monitoring and control (M&C) using your program and the IoT kit you were issued.

Your primary responsibility is for the HTTP web client. However, you are responsible for all facets of the M&C system, including the hardware connections and the web server code. You are free to modify the hardware and code in any way you choose, but you do not have to.

Software:

Python socket library

Arduino IDE and supporting libraries (ESP8266 board support, DHT sensor driver, Adafruit Unified Sensor framework)

Project details:

Due date:

The project is due at 10 AM on 12/09/19, by which point all deliverables should be uploaded to myGCC. The M&C system will be demonstrated to the instructor and/or graders at a time around this deadline, and no later than 4 PM on 12/11/19.

Expected deliverables:

- One Python program called HTTPIoTClient.py
- One ESP8266 program called HTTPIoTServer.ino (only if you choose to modify the web server code provided)
- One submission form

Functional behavior:

The behavior is to monitor and control hardware connected to the ESP8266 web server using the HTTP web client.

<u>ESP8266 web server</u>

The web server is your NodeMCU board based on the ESP8266 microcontroller, which should run the web server used for Lab 9.[1] See Lab 9 for hardware connections, pin assignments, and the HTTP message "API" used by the web server.

IoT kit hardware

You may choose to use any of the IoT kit hardware you like, as long as it contains one or more actuators and two or more sensors. The DHT sensor must be one of the sensors.

Your kit hardware should be chosen to simulate an IoT use scenario. You may choose your own, or you may use one of the options below.

- Home security system:
  - Proximity sensor indicating if doors & windows closed
  - Temperature sensor indicating if home is on fire or freezing cold
  - White LED as home's exterior lights
  - Red LED and buzzer as home's burglar alarm
- Factory control system:
  - Light sensor indicating if space is occupied
  - Temperature and humidity sensors indicating if heat index is too low or high for workers
  - Green and red LEDs as production status indicators
- Hospital patient monitoring system:
  - Proximity sensor indicating if patient bed is in use
  - Touch sensor as patient's button to request emergency assistance
  - Temperature sensor indicates patient body temperature
  - Red LED and buzzer as patient emergency alarms

HTTP client behavior

When the HTTP client starts, it should follow this sequence:

1. Ask the user for an IP address for the web server
2. Immediately attempt to connect one time
3. Once connected, list the valid actions available to the user, a provide a prompt for the user to select an action.

The client should then provide support for the actions listed in Table 1 below.

[1] This web server code was updated after Lab 9 was run in class, and should be re-downloaded for use in Project 3.

Table 1. Valid HTTP client commands and desired responses

Requested action	**Information provided to user**	**HTTP message to server**
List hardware	Hardware available, including pin connections	[none]
Query specified sensor value	Current value of specific sensor	GET /sensors/…
Query all sensor values	Current value of all sensors	GET /sensors
Set actuator value	Confirmation of actuator being set	PUT /led/… or PUT /buzz/…
Set alarm condition	Confirmation of alarm condition; current value of alarmed sensor	GET /sensors/…
Start logging sensor data	Confirmation of logging start; current value of sensor being logged	GET /sensors/…
Stop logging sensor data	Confirmation of logging stop; current value of sensor being logged	GET /sensors/…

The system should also provide three advanced features:

- A password should be requested the first time a user tries to set an actuator value
- When an alarm condition is active, the system should query the sensors on its own every 15 seconds
- An email or text should be sent when an alarm condition is detected

Project Policies:

- This project is a group project. Every student needs to work as a part of a 2-person team. For this project, you will be assigned a teammate randomly.
- This system will be demonstrated in person to the instructor and/or graders. Teams will sign up for a 10-minute demonstration time slot between 12/02/19 and 12/11/19. Retries will be considered, but may not be logistically feasible, so each team needs to plan on getting only one chance for this demonstration.
- Written assignments must be submitted electronically via myGCC. Be sure to upload your files correctly the first time.
- 20% of the grade will be weighed with the peer evaluation. Students are expected to turn in the peer evaluation form posted on myGCC. If a student works on his or her own without a team, then 20% of the grade will be deducted from the project grade.

Grading Rubric:

Grades will be assigned based on:

- System functionality as demonstrated to the instructor and/or graders (60%)
    - The client connects to the web server: 10%
    - The system allows the user to query sensors and control actuators as in Table 1: 10%
    - The client implements an alarm system when requested by the user, as described above: 20%

  - The client implements data logging when requested by the user, as described above: 20%
- Project deliverable submissions (40%)
  - Program is properly organized and commented: 10%
  - Project submission form – peer evaluation: 20%
  - Project submission form – screenshots demonstrating project completion: 10%

Academic Integrity Policy:

- Each team is expected to work on its own. Members of each team can work together, discuss ideas, look at each other's code, and share files among each other.
- Students belonging to different teams should not share code that directly bears on this project, or look at each other's code for this project. Any instances of this will be considered a violation of the academic integrity policy of this course and will be reported to the SFRC committee.
- Use or possession of past solutions and similar solutions from online resources is strictly prohibited and is considered a violation of the academic integrity of this course.
- You may use online resources to look up how to use a function, but you may not copy code from online resources. Any copied code (whether cited or not) is considered a violation of the academic integrity policy of this course.

Last modified: 11/18/19 lkr