IoT Motion-Door Sensor System Design

I would like to share with you my system design to monitor motion and door opening. I want to start by saying that this is not a tutorial on hardware design, it is a complete user friendly system design description. I will talk about the hardware aspect, share the schematic and firmware of the hardware “Internet of Things” (IoT) device, but not really get too deep into the hardware design or construction. The main focus of this paper is the system architecture and how the IoT device interacts with the Internet, the cloud, the server, mobile devices, and of course the user.

I will also have to give the usual disclaimer that I will not likely be able to help persons wanting to duplicate this in whole or part much individual attention. You'll need to have a decent grasp of box construction, electronics and programming to duplicate this project. In particular one of the most challenging things you will need to figure out the server file and folder structure and possibly modify the code provided to fit your folder structure. If you don't have all these skills then have fun learning them, as I am not going to be able to teach it all to you. Basically, you are going to be pretty much on your own. I will be updating this web site from time to time with answers to frequently asked questions as time goes on. So if you don't get a response to a question, check back from time to time. If a lot of people are asking the same thing, I may address it here. You may use or distribute this software freely as long as you allow my credits to remain noted as you see them in the headers of the firmware and scripts and on all copies of this paper.

Don't feel you need to copy every part or software exactly. Feel free to innovate, experiment, and substitute. Lots of things can be changed or modified to end up with a similar design. There must be dozens of different ways to implement the electronics alone. You could use an Arduino, a Raspberry Pi, or other micro-controller, instead of the WiFi enabled ESP-8266 development board I used to build the hardware IoT device. This is just the way I did it. I hope it sparks your imagination and ambition.

I have attempted to show the very highest level architecture in the diagram Figure 1.

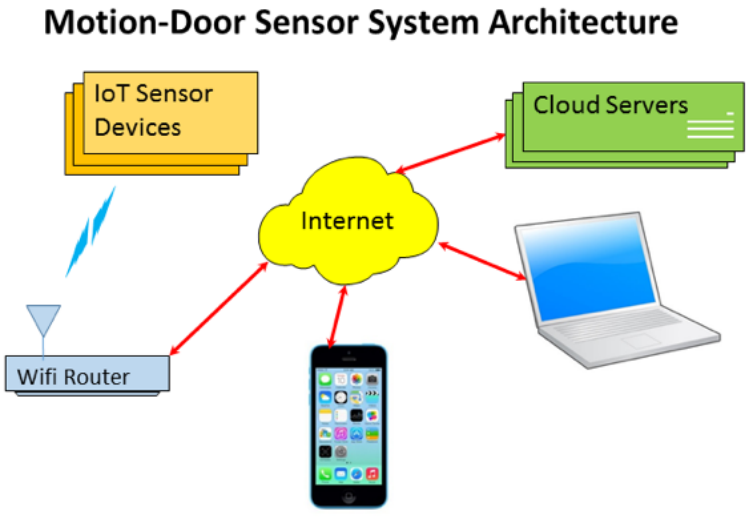

Figure 1 - Motion-Door Architecture

System Architecture

Shown in Figure 1 you will find the IoT hardware device(s) that talks to the WiFi router, so these devices can placed in any location that can communicate with the WiFi allowing great flexibility in placement. You may have many IoT devices connected to this system, the only real number limit is the number of WiFi connections allowed by your router(s). The connected IoT device then sends its status to the router which in turn communicates with the cloud server via the internet. The server then responds to the IoT device, collects, logs, and acts on the sensor data, allows transmission of SMS text alerts, production of analytics, charts, trends, and statistics. Access to the server is done via mobile or PC webpage interaction. The system also includes three IOS applications that run on iPhones and iPads for system control and configuration.

Software included in the system:

- Firmware in the IoT device
- Server scripts written in PHP
- Configuration files
- Webpage HTML

IOS source code will not be included, but the applications are available free at the Apple App Store. Graphing, plotting and trending code are also not a subject of this paper. As with any system there are still a few things that must be configured manually via a text editor of some configuration files on the server. At some point I may write new apps to deal with some of these.

Hardware IoT Device

The electronic hardware is quite simple, it is made up of 3 modules, ESP8266 controller, PIR sensor, and Power pack. The device has one push button and one multi-color LED for status. The push button allows the user to load the Wifi credential, server URL, and user ID into the device via iPhone app “IoT Loader”. When the button is held in when powered up, the IoT device enters the “load” mode and is listening for the credentials that will be sent directly to it by the IOS app. The IOS device connects to the Wifi hotspot presented by the IoT device and directly connects to it to load the credentials. This is required only once unless the credentials must be changed as the credentials are held in flash memory on the IoT device. You will also have to configure the door contact switch external to this device for door open and close notifications. Figure 2 shows the schematic of the device.

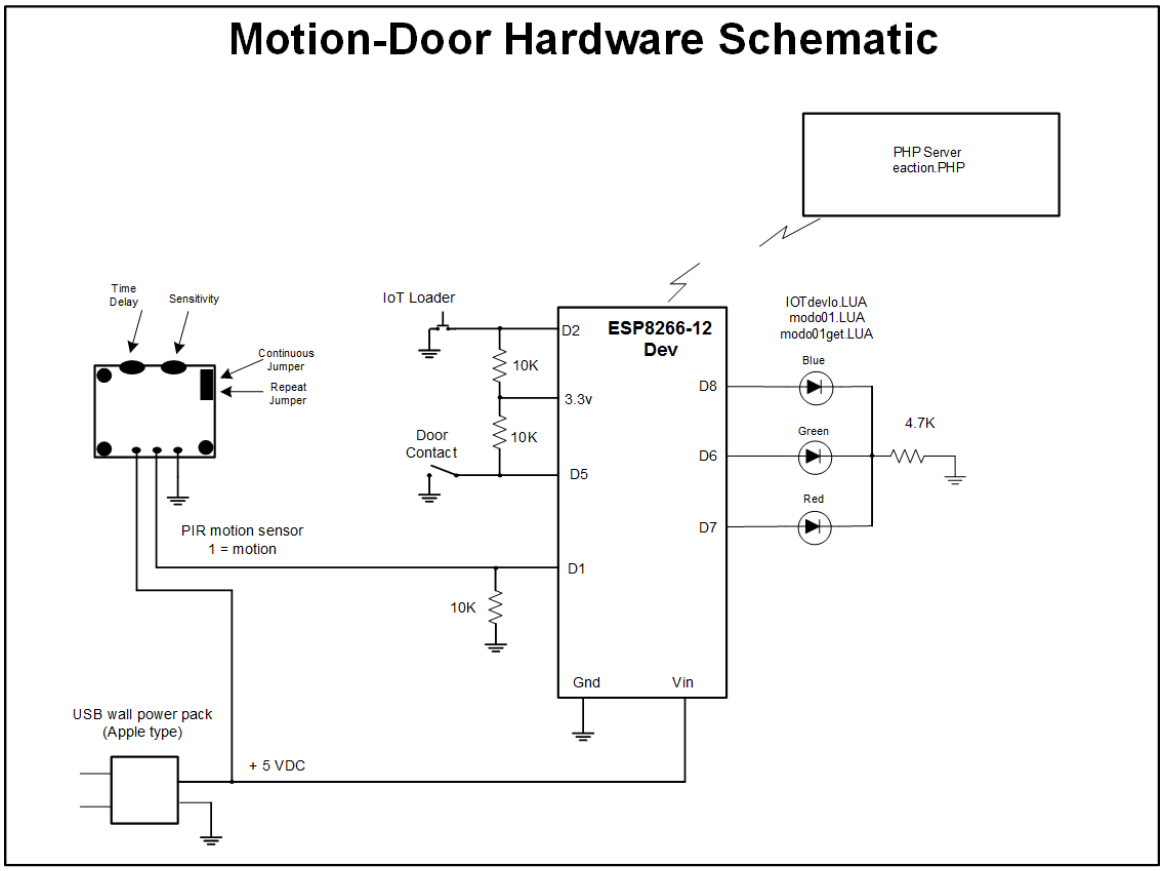

Figure 2 - IoT Sensor for Motion and Door

The LED is multi-color. Red illumination indicates that there is an error in the loaded credentials, Green illumination indicates that motion has been detected. And last Blue indicates that a door switch has opened or closed. When the LED flashes 3 times it indicates that the server has acknowledge the event and logged and acted on it accordingly.

Parts list:

- ESP8266-12 development board
- PIR motion sensor
- USB power pack (Apple type)
- Momentary push button
- Tri-color LED
- 3 each 10K resistors
- 1 each 4.7K resistor
- Construction box
- External door contact switch (usually magnet reed)

Figure 3 shows the completed IoT device and its major components. Note that the device plugs directly in to a wall outlet which allows for convenient placement of the devices. To hook up the door contact switch you must run a two conductor small gauge wire (24 awg is what I used) from the box to the switch on the door.

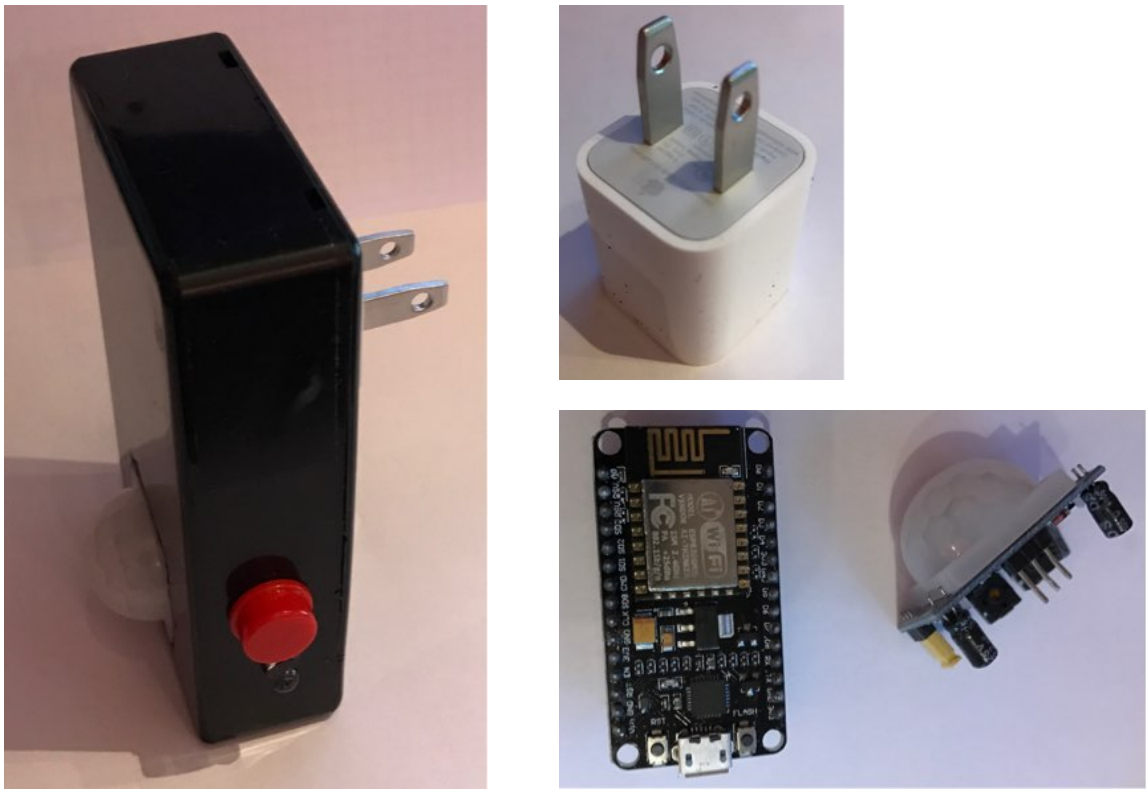

Figure 3 - Completed IoT device and major components

Software-Firmware Architecture

System software and firmware is the key to the system operation. Figure 4 shows the hardware interaction with the server and user control functions. The interface is made up of three compartmentalized components, top left are the hardware devices that communicated via WiFi, top right is the cloud server file structure, and bottom are the IOS user control function applications. Credentials telling the hardware IoT device the server URL and user identification information are loaded via the IOS app IoT-Loader via local Wifi. The IoT device can then send status and alerts to server for any configured action and data logging. The IoT-Alert allows the user to enable or disable alerts being SMS texted. IoT-Alert is particularly useful when you know events will occur such as occupancy of area and you don't want to receive text alerts all day long. Software is described and can be downloaded below (zip file), but note that there are some proprietary items and functions that will not be provided noted in commented areas of code. The system should however function without these modules with some missing features.

Firmware modules in the IoT device:
init.LUA - Boot start routine upon reset or power up
modo01.LUA/LC - Compiled main control routine
modo01get.LUA/LC - Compiled Wifi control module
IOTDevlo.LUA/LC - Compiled credential loader

Server side software modules for hardware interface:
aconfig.TXT - contains configuration information
acon.PHP - function called by the IoT-Alert application
acon.TXT - contains the Alert on/off switch settings
alabels.TXT - contains the labels displayed on the IoT-Alert app screen
eaction.PHP - function called by the IoT hardware device upon action required.
SN123456elog.TXT - contains the historical log of events for the IoT device
SN123456lX.TXT - contains the last time of the X event

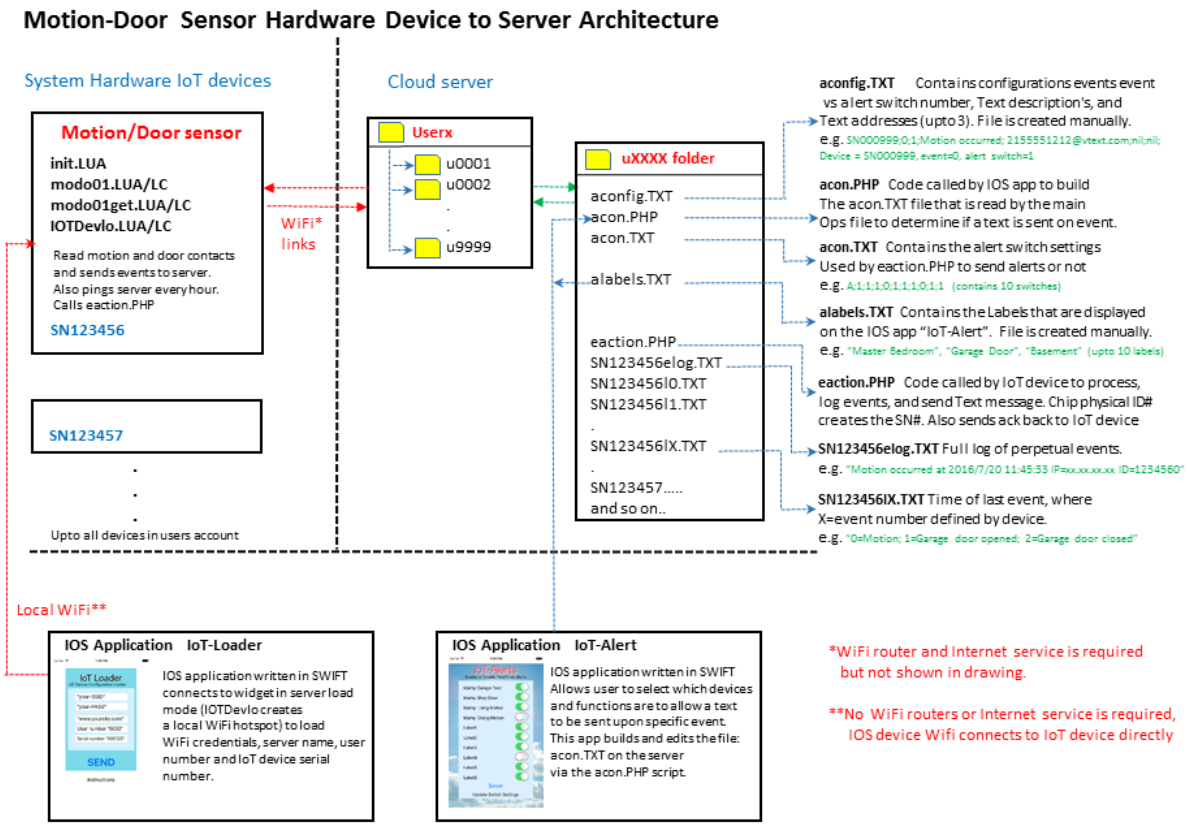

Figure 4 - Hardware to Server interface

Now that the hardware can talk to the server, we need to move on to the users interface for plotting and data analysis as shown in Figure 5. In the figure you can see there are only two main sections; the cloud server which is the same as discussed in figure 4, and the IoT-Pass IOS application. The IoT-Pass allows the user to change his PIN for access to the webpages that construct and display his data. Live examples of data display can be seen at http://www.remgate.com/IOT/examples.php. All activity to and from the user via webpage or IoT-Pass are encrypted SSL. The user logs into his server webpage for credential validation and access to his data and plots.

Server webpage login software:
ulogin.PHP - credential validator and credential entry window
uloginstyle.CSS - Style page for the login window
syscon.TXT - contains the user validation information (also modified by IoT-Pass)
pwtest.PHP - called by IoT-Pass to change PIN
trysco.TXT - contains the number of failed password attempts (if 10 tries you are locked out)

Once the user is validated he proceeds on to the server display area.

Webpage display software:
ustart.PHP - displays the HTML page of prebuilt user displays to select from (see example link above)

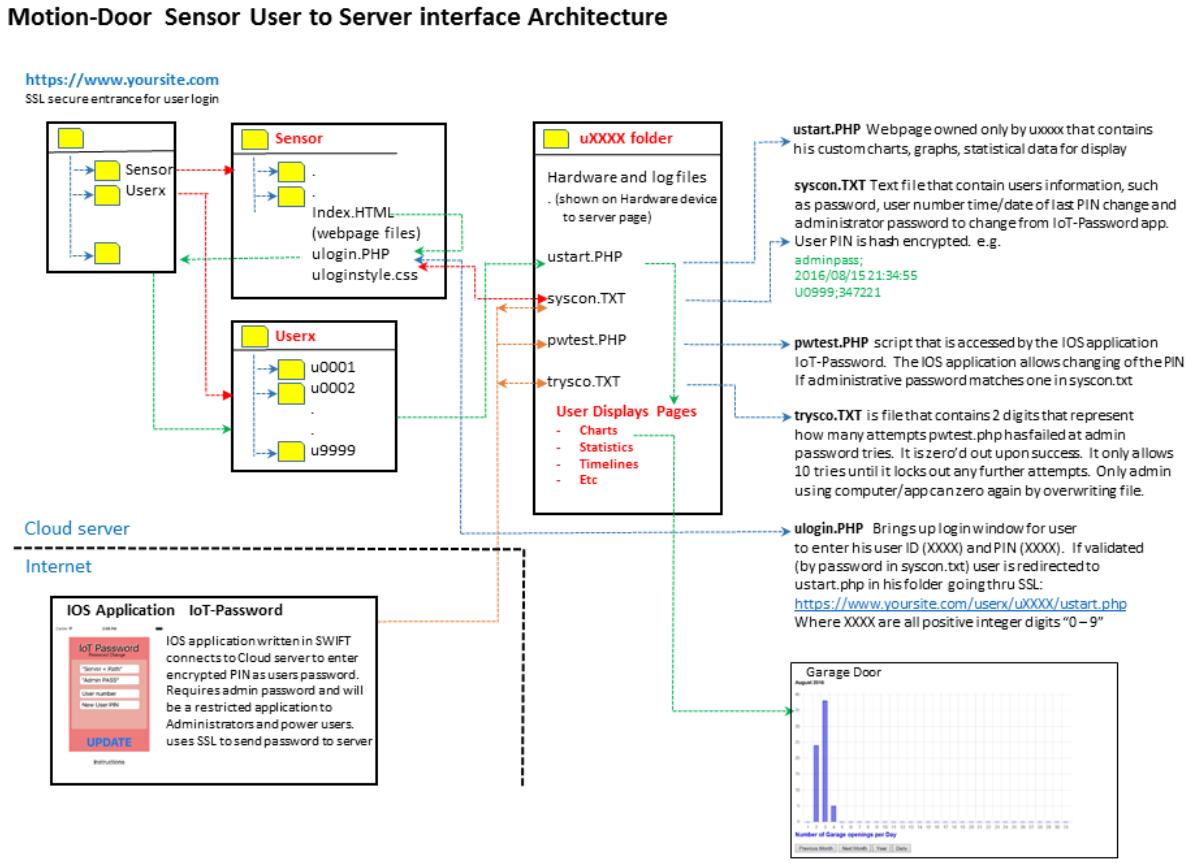

Figure 5 - User to system interface

IOS Applications

Three IOS applications are used for this system. These apps allow for the loading of the IoT device credentials “IoT-Loader”, the changing and entry of your system password for access to the secure website for charts, graphs and statistics “IoT-Pass”, and finally “IoT-Alerts” which allow the user to turn on and off SMS text alerts when an event occurs. These apps are shown in figure 6 and are available at the Apple App Store (search Dave Massey to quickly find them).

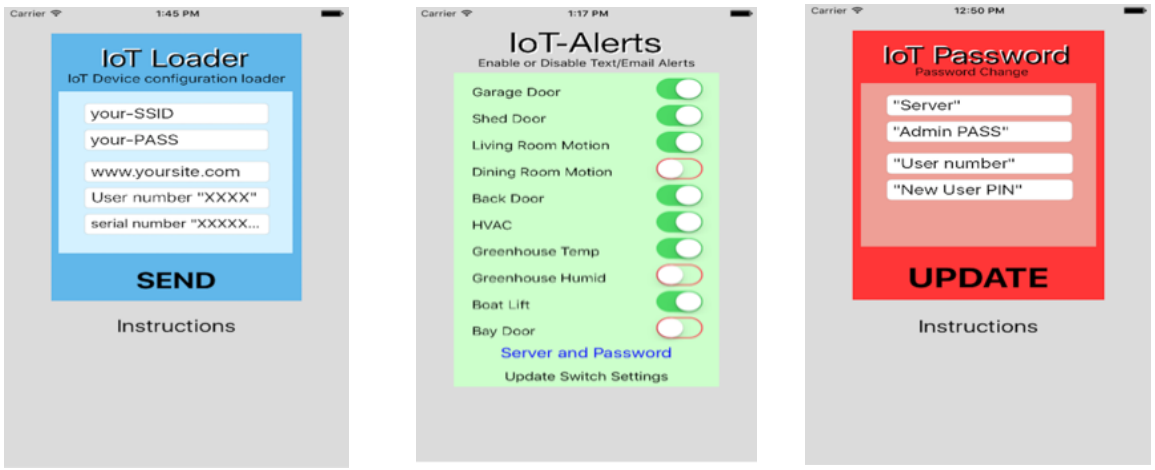

Figure 6 - IOS applications for system interface

Security

System security is of the utmost importance in the electronically connected world we live in. Most but not all of this system is encrypted, there is further work to be done. I will describe what security is implemented by module.

IoT-Loader application - No security has been implemented, now this is because the hardware IoT device ESP8266 has yet to produce a viable SSL function that can be implemented in a device with very limited resources. Noting no security here is probably very low risk of hacking or data interception. This is because the IoT-Loader communicates only locally with the hardware device only to load the WiFi credentials into the IoT hardware device. This normally is only done one time at setup, so the odds of this being intercepted are extremely low and no one could alter the IoT device unless it is booted up holding in the push button.

IoT Hardware Device - The IoT hardware device communicates with the WiFi router using what ever encryption method the router requires (WPA, etc). No SSL encryption is initiated from the hardware device and thus no protection of interception once data leaves the router to the Internet service provider. However a hacker would have to have knowledge of the data protocol and a very long time of monitoring in order figure out what the device is saying to the server and there is no opportunity for the hardware device to be altered or overtaken with a virus as there is no path from the server to the device. A physical wiretap, cooperation of, or a breach into the

service provider would also have to be implemented to see the data. Risk is low but not nonexistent.

IoT-Pass and IoT-Alerts - Both applications only use SSL to communicated with the server as https:, so risk of interception is extremely low.

User interaction with webpage - The users interaction to his webpage to view his data and charts is all by SSL (https:) including his credential validation, so risk of interception or monitoring is extremely low.

Security Summation - Security is not yet perfect, but does have some important SSL connections and once the community develops an SSL function for the ESP8266 it will be easy to lock down all the associated functions and applications to completely secure the system. An alternate method of course would be to choose a different hardware processor that has such an SSL function. The insecure functions today can be looked at to be “Security by Obscurity”, not what I would use for my financial activities, but low risk here.