Smart Senior Care

Sensor Network & Raspberry Pi & AWS Documentation

By Peijmon Kasravi

Setting up Vibration/Motion Sensor Node:

1. Install Arduino IDE
2. Install ESP8266 Driver
    a. Enter *http://arduino.esp8266.com/stable/package_esp8266com_index.json* into Additional Board Manager URLs field in the Arduino v1.6.4+ preferences.
    b. Next, use the Board manager to install the ESP8266 package.
    c. After the install process, you should see that esp8266 package is marked INSTALLED. Close the Boards Manager window once the install process has completed.
3. When you've restarted, select **NodeMCU 1.0 (ESP-12E Module)** from the Tools->Board dropdown
4. Set 80 MHz as the CPU frequency
5. Select 115200 baud for Upload Speed
6. Select the matching COM port for your USB cable. (Usually dev/cu.SLAB_USBtoUART)
    a. If COM port is not available install SiLabs CP2104 Driver from *https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers*
7. Install MQTT library
    a. Sketch->Include Libraries->Manage Libraries
    b. Install PubSubClient
8. Open /SmartSeniorCare/Arduino/sensorNode.ino
9. In line 5 enter GPIO pin being used
10. In Line 8 define sensorID desired (Syntax is <Vibration/Motion>_Sensor_<ID#>_<Location>)
11. In Line 13 change IP of Raspberry Pi if different IP is being used.
12. Flash program to ESP8266
13. Serial Monitor can be used to view debugging messages
14. Connect GND from sensor to GND on ESP8266, connect Vcc from sensor to 3.3V on ESP8266, connect Out from sensor to GPIO13 on ESP8266(unless otherwise defined in code).

Setting up Raspberry Pi

Defining Personal Area Network for sensors to publish data to

1. Connect Raspberry Pi to router using ethernet cable
2. In Terminal type, *sudo apt-get -y install hostapd dnsmasq*
3. Edit the dhcpcd file: *sudo nano /etc/dhcpcd.conf*
4. Scroll down, and at the bottom of the file, add: *denyinterfaces wlan0*
5. Uncomment the following lines:
    a. profile static_eth0
    b. static ip_address=192.168.1.23/24 (**Change this to 192.168.0.166/24**)
    c. static routers=192.168.1.1

   d. static domain_name_servers=192.168.1.1

6. Press CTRL+X, press Y, hit Enter
7. Open the interfaces file: *sudo nano /etc/network/interfaces*
8. At the bottom of that file, add the following:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.5.1
    netmask
255.255.255.0
    network 192.168.5.0
    broadcast
192.168.5.255
```

9. Press CTRL+X, press Y, hit Enter
10. Edit the *hostapd.conf* file: *sudo nano /etc/hostapd/hostapd.conf*

```
interface=wlan0
driver=nl80211
ssid=MyPiAP
hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=raspberry
rsn_pairwise=CCMP
```

11. Press CTRL+X, press Y, hit Enter
12. Open /etc/default/hostapd: *sudo nano /etc/default/hostapd*
13. Find the line #DAEMON_CONF="" and replace it with: *DAEMON_CONF="/etc/hostapd/hostapd.conf"*
14. Press CTRL+X, press Y, hit Enter
15. Edit dnsmasq: *sudo nano /etc/dnsmasq.conf*

```
interface=wlan0
listen-address=192.168.5.1
bind-interfaces
server=8.8.8.8
domain-needed
bogus-priv
dhcp-range=192.168.5.100,192.168.5.200,24h
```

16. Press CTRL+X, press Y, hit Enter

17. Reboot Pi: *sudo reboot*
18. Now when turning on sensor nodes, using serial monitor you can see that they successfully connect to the Pi's network

Setting up MQTT on Pi

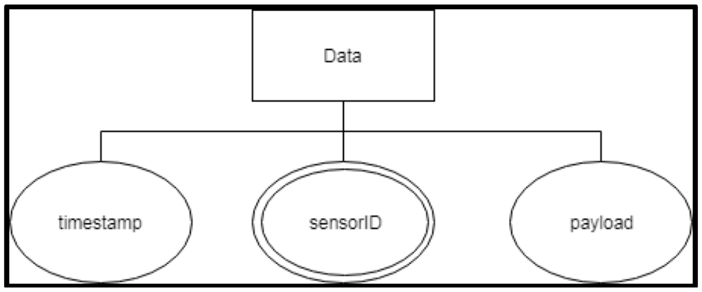
1. Install mosquitto and then the mosquitto-clients packages:
   i. *sudo apt-get install mosquitto -y*
   ii. *sudo apt-get install mosquitto-clients -y*
2. Configure the broker
   i. *sudo nano /etc/mosquitto/mosquitto.conf*
3. Delete the line: *include_dir /etc/mosquitto/conf.d*
4. Add the following lines to the bottom of the file: *listener 1883*
5. Press CTRL+X, press Y, hit Enter
6. Reboot Pi: *sudo reboot*

Installing dependencies on Pi

1. Enter in Terminal:
   i. *sudo apt-get update*
   ii. *sudo apt-get upgrade*
   iii. *sudo apt-get install python-setuptools python-dev build-essential*
   iv. *sudo apt-get install python-pip*
   v. *pip install paho-mqtt*
   vi. *pip install pytz*
   vii. *sudo apt-get install libmariadbclient18 libpq5 libavcodec57 libavformat57 libavutil55 libswscale4*
   viii. *sudo apt-get install motion -y*
   ix. *sudo apt-get install miniupnpc*

Setting up AWS

1. Create an AWS account
   i. Download AWS IoT Python SDK and required certificates
   ii. Copy /AWS/basicPubSub.py to /aws-iot-device-sdk-python/samples/basicPubSub/
   iii. Copy /AWS/subtest.py to /aws-iot-device-sdk-python/samples/basicPubSub/
2. Create table in DynamoDB following database diagram



3. Create Lambda functions using supplied code in /Python/lambdaFunctions.py
4. Create two topics in AWS SNS: *noMotion* and *vibration*

5. To subscribe new phone numbers to notification system, check the box next to *noMotion* or *vibration*. Click Actions and select Subscribe to topic.
6. Select SMS for protocol (or email if desired)
7. Enter phone number or email address
8. Click Create subscription
9. Repeat for remaining topic

Setting up Camera Livestream on Pi

1. Connect camera(s) to an available USB port on Raspberry Pi
2. Configure Pi video stream settings for your camera
3. In one Terminal enter *v4l2-ctl -V*
4. Open a second Terminal and edit Motion's configuration file: *sudo nano /etc/motion/motion.conf*
5. The following lines have to be adjusted (the variable can be searched with CTRL + W, the bold values have been changed):

```
# Start in daemon (background) mode and release terminal (default:
off)
daemon on
...
# Restrict stream connections to localhost only (default: on)
stream_localhost off
...
# Target base directory for pictures and films
# Recommended to use absolute path. (Default: current working
directory)
target_dir /home/pi/Monitor
```

6. The following lines also need to be changed according to the camera's output values seen in first Terminal window

```
v4l2_palette 15    #Video format setting
...
# Image width (pixels). Valid range: Camera dependent, default: 352
width 640

# Image height (pixels). Valid range: Camera dependent, default:
288
height 480

# Maximum number of frames to be captured per second.
# Valid range: 2-100. Default: 100 (almost no limit).
framerate 10
```

7. Press CTRL+X, press Y, hit Enter
8. Now we have to activate the daemon
   i. *sudo nano /etc/default/motion*
   ii. Replace no with yes: *start_motion_daemon=yes*
   iii. Press CTRL+X, press Y, hit Enter
9. Create the folder to store captured frames
   i. *mkdir /home/pi/Monitor*
   ii. *sudo chgrp motion /home/pi/Monitor*

        iii. *chmod g+rwx /home/pi/Monitor*

10. Forward the port to the outside world
    i. *upnpc -r 8081 tcp*
    ii. After running this command an external IP address will be given and this must be entered into the Android code for viewing

Running the System

1. Ensure all sensor nodes are connected to power
2. Ensure Raspberry Pi is connected to power and router
3. In Terminal enter: *sudo service motion restart*   This will begin the camera streaming service
4. Run by entering *./start.sh*

- Now the Raspberry Pi will begin listening for messages sent by the sensor nodes. When received will upload data to AWS IoT. The IoT rules will save the data in the DynamoDB table and determine when an alert is received and send out an SMS notification. The camera port(s) will also be open for streaming upon request.

Directory Description

- SmartSeniorCare -> Parent Folder
- SmartSeniorCare/Arduino/sensorNode.ino -> Arduino Code for sensor nodes
- SmartSeniorCare/Python/basicPubSub.py -> Our modified AWS IoT Python script. This file needs to be copied into /aws-iot-device-sdk-python/samples/basicPubSub/
- SmartSeniorCare/Python/subtest.py -> Our Python script for receiving MQTT messages from sensor nodes to be accessed by basicPubSub.py script. This file needs to be copied into /aws-iot-device-sdk-python/samples/basicPubSub/
- SmartSeniorCare/Python/piSubTest.py -> Debugging script used to determine if Raspberry Pi is properly receiving messages from sensor nodes
- SmartSeniorCare/Python/lambdaFunctions.py -> Code for Lambda functions used. This code needs to be loaded into two separate Lambda functions. The Lambda functions must be triggered by AWS IoT and have access permissions for SNS and DynamoDB.
- SmartSeniorCare/AWS/AWS_IoT_Rules.txt -> This file contains the SQL statements used for the respective IoT rules in the project that trigger the Lambda functions
- SmartSeniorCare/AWS/start.sh -> This is the startup script that starts the show