İTÜ

Electrical-Electronical Faculty

Control and Automation Engineering

Microconroller Based Systems

Season Project

Subject:Home Automation

Students:

Abdulbaki Aybakan-040120523

2015-2016

Project Overview

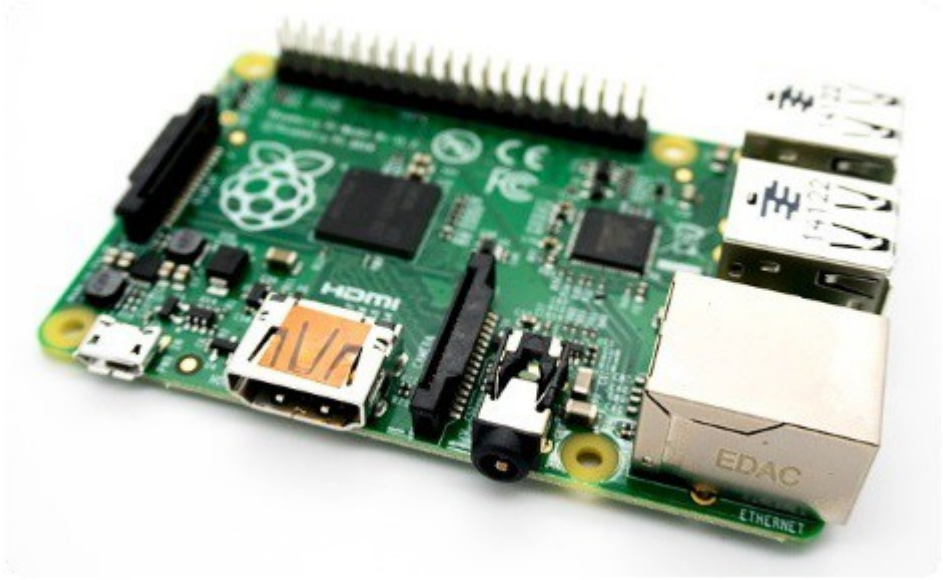
We are going to set up a kind of Home Automation system with help of Arduino and Rasperry Pi. The steps are like below;
1-A single Arduino with a temperature and humidity sensor,
2-Which sends the data of the sensors wirelessly to a Rasperry Pi
3-Which will fetch the message,
4-Publish it to an MQTT channel,
5-Where the OpenHAB MQTT binding will pick it up,
6-So it can be displayed on the web browser

Components

- Rasperry Pi Starter Kit
- Arduino Nano
- Ethernet cable
- Temperature and Humidity Sensor
- 2 nRF24L01+
- Set of Dupont wires

Steps

1-Rasperry Pi



Rasperry Pi is a mini-computer.

First, we need to have some components to control and use it just like a fully functional computer;

- plug a mouse and keyboard into the USB ports,
- attach a monitor or TV to the HDMI port,
- insert a micro SD card with Raspbian,

Raspian is the operating system which is used with Raperry Pi.

Internet access can be via a Wifi dongle or an ethernet cable.

- Internet Access
If we connect Rasperry Pi to the router via the ethernet port, we dont need any setup but in order to use Wifi module, we need to install driver for that module.

For example, if we use Tp-link WN-722N Wifi module, we use the following Linux commands,

```
sudo apt-get update
sudo apt-get dist-upgrade
apt-get install gcc build-essential linux-headers-generic linux-headers-`uname -r`
tar xvfz backports-3.16-1.tar.gz
cd backports-3.16-1
sudo make clean
sudo make
sudo make defconfig-ath9k
sudo make
sudo make install
```

We also want your router to assign a fixed IP address to the Pi so we can always find it. There are two configuration files we need to edit for this. First:

```
sudo nano /etc/network/interfaces
```

The example below is for a network that uses the 192.168.1.x range. In the example, the Pi will have address 192.168.1.80 and the router is at 192.168.1.1. If your local network uses a different IP range, make sure to follow that instead.

```
auto lo
iface lo inet loopback
allow-hotplug wlan0
auto wlan0

iface wlan0 inet static
address 192.168.1.80
netmask 255.255.255.0
gateway 192.168.1.1
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

When you're done modifying the contents of the file you press CTRL+X to exit. The program will ask if you want to save so you press Y for Yes and Enter to confirm the file name.

Now in the second configuration file you enter your :

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

This file might be empty. Add the text below and replace "wifiname" and "wifipassword" with your network's details.

```
network={

ssid="wifiname"

psk="wifipassword"

key_mgmt=WPA-PSK

}
```

If your wifi network is more complicated, you can find online guides by googling "wpa_supplicant raspberry pi". When you're done, again exit with CTRL+X, Y, Enter.

2- Arduino

First, we need to include some libraries to the code.

For talking to the DHT sensors, we need to include dht.h library as below;

```
#include <dht.h>
```

Then we add a 10k resistor between sensor and arduino nano and connect to the digital-2 pin.

#define DHTPIN 2

And we add the RF24 and RF24Network libraries to the Arduino Code.

The code for Arduino Nano is

```
#include <RF24Network.h>
#include <RF24.h>
#include <SPI.h>
#include <DHT.h>

// The DHT data line is connected to pin 2 on the Arduino
#define DHTPIN 2

// Leave as is if you're using the DHT22. Change if not.
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

// Radio with CE & CSN connected to 7 & 8
RF24 radio(7, 8);
RF24Network network(radio);

// Constants that identify this node and the node to send data to
const uint16_t this_node = 1;
const uint16_t parent_node = 0;

// Time between packets (in ms)
const unsigned long interval = 2000;

// Structure of our message
struct message_t {
  float temperature;
  float humidity;
};
message_t message;

// The network header initialized for this node
RF24NetworkHeader header(parent_node);

void setup(void)
{
       // Initialize all radio related modules
       SPI.begin();
       radio.begin();
       delay(5);
       network.begin(90, this_node);

       // Initialize the DHT library
       dht.begin();

       // Set up the Serial Monitor
       Serial.begin(9600);
}

void loop() {

       // Update network data
       network.update();

       // Read humidity (percent)
```

```
	float h = dht.readHumidity();
	// Read temperature as Celsius
	float t = dht.readTemperature();
	// Read temperature as Fahrenheit
	float f = dht.readTemperature(true);

	// Construct the message we'll send (replace t with f if you're sending
Fahrenheit)
	message = (message_t){ t, h };

	// Headers will always be Temperature for this node
	// We set it again each loop iteration because fragmentation of the messages might
change this between loops
	header.type = 't';

	// Writing the message to the network means sending it
	if (network.write(header, &message, sizeof(message))) {
		Serial.print("Message sent\n");
	} else {
		Serial.print("Could not send message\n");
	}

	// Wait a bit before we start over again
	delay(interval);
}
```

3- Mosquitto

First of all, Mosquitto is an open-source message broker, which means it lets different programs to exchange data in a way that they can all understand.

Information is exchanged via channels, where you publish information or subscribe to to receive information. Each sensor needs a channel for this use.

The Raspberry Pi, receiving information from various places, puts that information on the right channel so home automation software like **OpenHAB** can pick it up.

Secondly, we install Mosquitto to the Rasperry Pi as below;

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key

sudo apt-key add mosquitto-repo.gpg.key

rm mosquitto-repo.gpg.key

cd /etc/apt/sources.list.d/

sudo wget http://repo.mosquitto.org/debian/mosquitto-wheezy.list

sudo apt-get update

sudo apt-get install mosquitto mosquitto-clients
```

4- RF24

There are 8 pins on the nRF24L01 wireless module. But we dont need to use IRQ pin and you can see the ground pin is surrounded by white line. You can look at the pins on the website referred on references.

Connections with Arduino is as below.

nRF24L01+	Arduino (Uno, Nano)
GND	GND
VCC	3.3V
CE	D7
CSN	D8
SCK	D13
MOSI	D11
MISO	D12
IRQ	-

And the Connections with Rasperry Pi is as below.

nRF24L01+	Raspberry Pi (B/B+)
GND	GND
VCC	3.3V (17)
CE	GPIO22 (15)
CSN	GPIO8 (24)
SCK	SCKL (23)
MOSI	MOSI (19)
MISO	MISO (21)
IRQ	-

And now we install RF24 Libraries to the Rasperry Pi with the code below;

```
wget http://tmrh20.github.io/RF24Installer/RPi/install.sh chmod +x
install.sh ./install.sh
```

- For Rasperry Pi to take the data

We need to download and run hareceiver code in order to get the information from Arduino via RF24 module. The code we use on the command line is as below;

```
cd ~/rf24libs
```

```
wget http://www.homeautomationforgeeks.com/code/hareceiver.cpp wget
http://www.homeautomationforgeeks.com/code/Makefile
```

```
make
```

```
sudo ./hareceiver
```

And now you'll see a printout of some radio details, followed by a dot every 2 seconds while the Pi patiently waits for data until the Arduino starts sending. If you want to change anything on the code , you can make changes on hareceiver.cpp file and again do "make".

5- OPENHAB

OpenHAB is an open-source software for integrating different home automation systems and technologies into one single solution that allows over-arching automation rules and that offers uniform user interfaces.

- Installing OpenHAB

We will install the OpenHAB in the Desktop folder.

```
sudo mkdir openhab
```

```
cd openhab
```

```
sudo wget https://bintray.com/artifact/download/openhab/bin/distribution-
1.7.0-runtime.zip
```

```
sudo unzip distribution-1.7.0-runtime.zip
```

```
sudo rm distribution-1.7.0-runtime.zip
```

- Configuring MQTT Binding

In order to make the connection between Mosquitto and Rasperry Pi, we need to make some changes on the configuration file

```
sudo cp /opt/openhab/configurations/openhab_default.cfg
/opt/openhab/configurations/openhab.cfg
```

```
sudo nano /opt/openhab/configurations/openhab.cfg
```

Scroll all the way down to the MQTT Transport section (under Transport Configurations). If you see the MQTT Persistence section, keep going. It’s further down still. Find the line with <broker>.url and <broker>.retain and activate them by removing the # sign at the start. And change them with

```
mqtt:mymosquitto.url=tcp://localhost:1883
```

```
mqtt:mymosquitto.retain=true
```

- Setting up Default Items and Sitemap

First, we configure the items file. Open the items file as below;

```
sudo nano /opt/openhab/configurations/items/default.items
```

We'll create a main group "All" with just a single "Ground Floor" and on that floor a single "Living Room":

```
-Group All

-Group gGroundFloor (All)

-Group GF_Living "Living Room" <video> (gGroundFloor)

-Number TestTemperature "Temperature [%.1f F]" <temperature> (GF_Living)
{mqtt="<[mymosquitto:home/temperature:state:default]"}
```

Secondly, we configure the sitemap.Open the sitemap file as below;

```
sudo nano /opt/openhab/configurations/sitemaps/default.sitemap
```

And we add Temperature and Humidity sections.

- Final Step: Start OpenHAB

We start the OpenHAB with this command;

cd Desktop/openhab

sudo ./start.sh

After we started it, we look at the http://localhost:8080/openhab.app address on the web browser.

Or if you want to access from a different computer, you should write the IP address of the Rasperry Pi instead of "localhost".

REFERENCES:

- http://www.homeautomationforgeeks.com
- https://github.com/maniacbug/RF24
- http://playground.arduino.cc/Main/DHTLib
- http://tmrh20.github.io/RF24/
- http://www.openhab.org/