AKHomeAutomation table of contents

AKHomeAutomation

AKHomeAutomation is Raspberry Pi based home automation / smart home / smart house solution based on common and cheap components, works realiably and I think is very convenient to use.

It allowins to control many things from any device in local network and also automate certain things (oh that hot coffe in the morning ready when you wake up)

My main goal for this home automation system was to create something that:

– would work inside of local network and be accessible from any device whether it is computer, phone or tablet for convenient use.

– not to be dependent in any way on anything outside of local network – there are no connections to cloud or some 3rd party services. It is all „inside" of local and independent from outside world. Internet is not needed here – just internal network will suffice.

– most important initial feature of this system was delayed auto disable. Devices that you configure into the system do not have to have that feature enabled but it definately turns out very practical in everyday usage – especially for lamp devices.

What it currently provides:

– web interface which allows to operate defined devices to anyone from your home network

– it allows control over radio operated devices/power switches

– it allows control over web operated devices/power switches

– provides way to schedule working time frames for devices by simply clicking in UI

– allows to define auto disable option with delay times after which devices would be turned off (energy saving for forgetting and salvation for lazy ones ;-))

– allows to define PIR sensors connected to Raspberry pi and basing on their signal can operate defined devices and/or trigger alarm/send alarm mails

Web interface looks like this. It is tablet and phone friendly.

Radio Switch 🗓   2h 0m

| Enable | Disable |

Coffee machine 🗓   2h 0m

| Enable | 00:03:59 (19:00) |

Humidfier Web Switch 🗓   30m

| Enable | Disable |

Living Room Lamp 🗓   30m

| Enable | 00:23:59 (19:20) |

TV Lamp 🗓   2h 0m

| Enable | Disable |

Room Lamp 🗓   2h 0m

| Enable | Disable |

Computer TV 🗓 Check availability

| Enable | Disable |

TV stack 🗓

| Enable | Disable |

All lights

| Enable | Disable |

For devices that can be be auto disabled after delay there is default auto off time defined in such device definition but user can also manually change delay when tapping on device name and using the following slider:
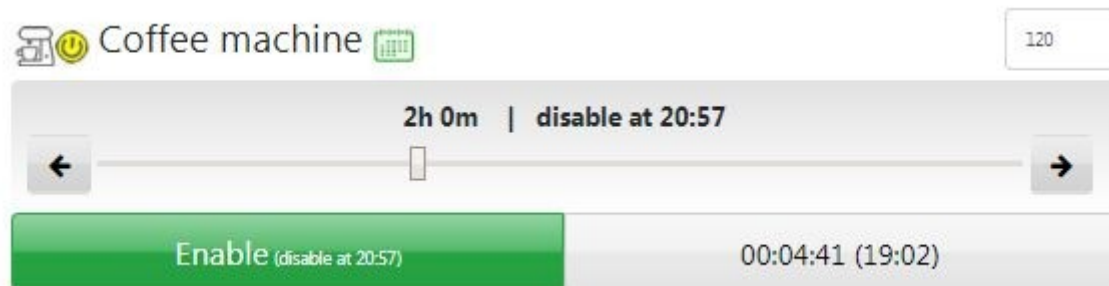
enabled device will then count down its life time on Off button

Schedulter timelines can be specified for the device by clicking on calendar icon and setting up its timelines:
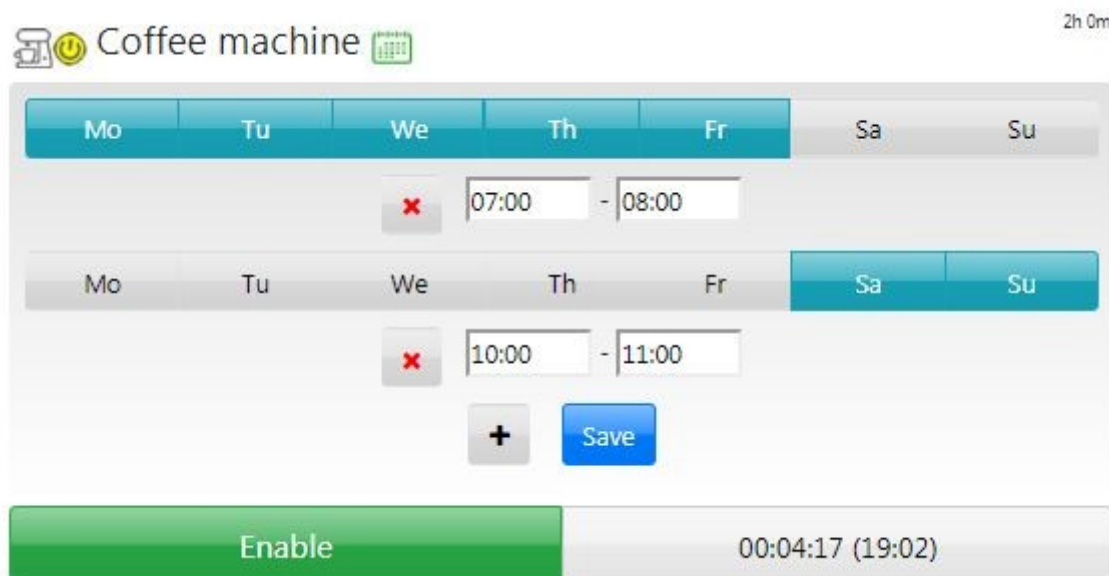
What Hardware is needed

1. Raspberry Pi with AC adapter/SD card/Wifi dongle. Because Raspberry Pi 3 has embedded wifi and is pretty fast I would recommend it but AKHomeAotomation should work on any Rasberry Pi. I have run it on 1/2/3 and had no problems on any. However R1 is already a bit on the slow side.

2. If you are going to operate radio controlled power switches:

   1. You will need radio transmitter that can be connected to Raspberry PI. I use **XY-FST, XD-FST** radio transmitters.

   2. Radio signal receiver is optional and is needed if you dont know radio code that should be send in order to put it into the AKHomeAutomation. In such case you read radio signal, that operates your radio controlled switch, with radio receiver connected to your Raspberry Pi and then put the code into the AKHomeAutomation. I use **XY-MK-5V, XD-RF-5v** radio receivers.
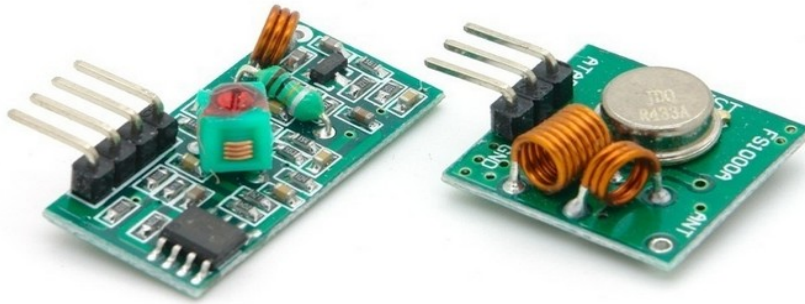
3. Note that above transmitters should suit frequency of devices that you want to handle. Above devices would handle 315Mhz, 330 Mhz and 433 Mhz frequencies. I have only worked with them for 433Mhz switches.

4. Below I puth url to youtube playlist that will contain any videos regarding AKHomeAutomation that I create. It will include vids describing what these devices are, how to connect them and how to use them with some radio controlled power switches.

   1. https://www.youtube.com/watch?v=C19ARWDYR3c&list=PLjd2MVjW6mhFygrvXyVcdNoq6pHK8MdUW

3. AKHomeAutomation also allows to define devices that will send web requests for ENABLE and DISABLE actions. These web requests operate as GET requests and can contain anything you define for ENABLE or DISABLE action. This actually presents many possibilities of use:

   1. For example you can operate **SONOFF** devices which are basically wifi operated powerswitches. SONOFF devices allow to upload flash them with your own firmware (lua scripts) which basically allows to to define how they are suppose to operate. I will create video describing SONOFF devices use, how to change their firmware and also provide script than can be used to connect SONOFF devices to AKHomeAutomation. Below image of **basic model** but it would probably work on all models using ESP8266 chip. I have also used **TH10** and **TH16** models and they work great.
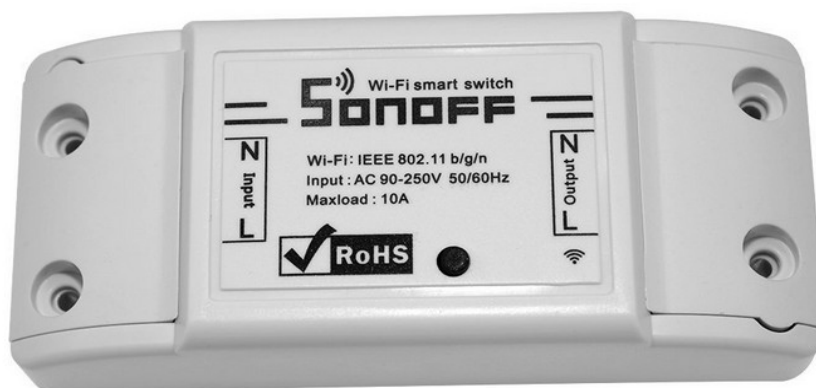
2. You can send requests to any device on your network. Requests can be different for ENABLE OR DISABLE action just the way you define them in AKHomeAutomation configuration.

4. AKHomeAUtomation allows to connect and manage **HC SR501** PIR detector which can be configured to enable other devices in the system or to trigger alarms.

AKHomeAutomation deployment in big short

I am not a doc guy so writing spesc and istructions is not easy for me. Believe me that I did put some effort into describing what should be done below and if this is still not clear please contact me at office@webproject.waw.pl . I will try to help if I can.

In big short please follow these chapters to make it all work. Enjoy :-)

1. Rasberry Pi Raspbian installation and setup
2. AKHomeAutomation files deployment
3. Radio Controlled Switches setup
4. AKHomeAutomation configuration

Rasberry Pi Raspbian installation and setup

This chapter is about setting up Rasberry Pi to be able to work with AKHomeAutomation system. If you see point here that you already have installed just skip it and go to the next.

Raspbian System installation

In big short. Those who did this before even once will know whats what and will do this crazy fast. For those who did not – use below points as general hints. Use instructions and all helpul info from https://www.raspberrypi.org/learning/software-guide/ . This point can be skipped if you already have Raspberry Pi with system on it.

1. Download Noobs image from https://www.raspberrypi.org/downloads/noobs/
2. Format SD card according to instructions from raspberry pi site.
3. Copy extracted system files directly to SD card
4. Run Raspberry Pi and let it install system
5. after instllation run to have latest version of your system:

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt-get dist-upgrade

Setup Machine name

1. `sudo nano /etc/hosts`

Leave all of the entries alone except for the very last entry labeled `127.0.1.1` with the hostname "`raspberrypi`". This is the only line you want to edit. Replace "raspberrypi" with whatever hostname you desire.

1. `sudo nano /etc/hostname`
2. sudo reboot

Setup password

Sudo raspi-config

```
1 Change User Password          Change password for the current u
2 Hostname                      Set the visible name for this Pi
```

Setup Wifi

sudo nano /etc/network/interfaces

```
GNU nano 2.2.6          File: /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

```
  GNU nano 2.7.4    File: /etc/wpa_supplicant/wpa_supplicant.conf    Modified

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
ap_scan=1
update_config=1
network={
        ssid="MyNetworkName"
        psk="Supersecretpassword
        key_mgmt=WPA-PSK
        scan_ssid=1
}
```

Setup Static IP

Sudo nano /etc/dhcpcd.conf

```
# Example static IP configuration:
interface wlan0
static ip_address=192.168.0.99/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8
```

Enable FTP

1. sudo raspi-config

```
  4 Localisation Options         Set up language and regional sett
  5 Interfacing Options          Configure connections to peripher
  6 Overclock                    Configure overclocking for your P

  P2 SSH                         Enable/Disable remote command lin
  P3 VNC                         Enable/Disable graphical remote a
```

```
Would you like the SSH server to be enabled?




                  <Yes>                         <No>
```

2. sudo apt-get install proftpd

   1. choose standalone
   2. [optional config changes] sudo nano /etc/proftpd/proftpd.conf

3. `sudo service proftpd restart`

Enable VNC

3. sudo raspi-config

```
 4 Localisation Options          Set up language and regional sett
 5 Interfacing Options           Configure connections to peripher
 6 Overclock                     Configure overclocking for your P
```

```
 P2 SSH                          Enable/Disable remote command lin
 P3 VNC                          Enable/Disable graphical remote a
 P4 SPI                          Enable/Disable automatic loading
```

```
Would you like the VNC Server to be enabled?




                <Yes>                    <No>
```
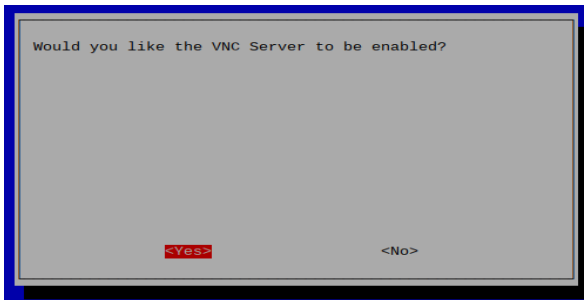
Now using VNC client you can connect to your Rasberry Pi using address yourstaticip:0, login: pi, password is you password that you set in „Setup password" chapter.

[Optional] Install wakeonlan

If you want to have option that would allow you to enable computers in your net via AKHomeAutomation → wakeonlan should be installed.

1. sudo apt-get install wakeonlan

Install and activate smbus and i2c

1. sudo apt-get install python-smbus
2. sudo raspi-config

```
 P4 SPI                          Enable/Disable automatic loading
 P5 I2C                          Enable/Disable automatic loading
 4 Localisation Options          Set up language and regional sett
 5 Interfacing Options           Configure connections to peripher
 6 Overclock                     Configure overclocking for your P
```

Would you like the ARM I2C interface to be enabled?

<Yes>   <No>

Install and configure Apache and PHP
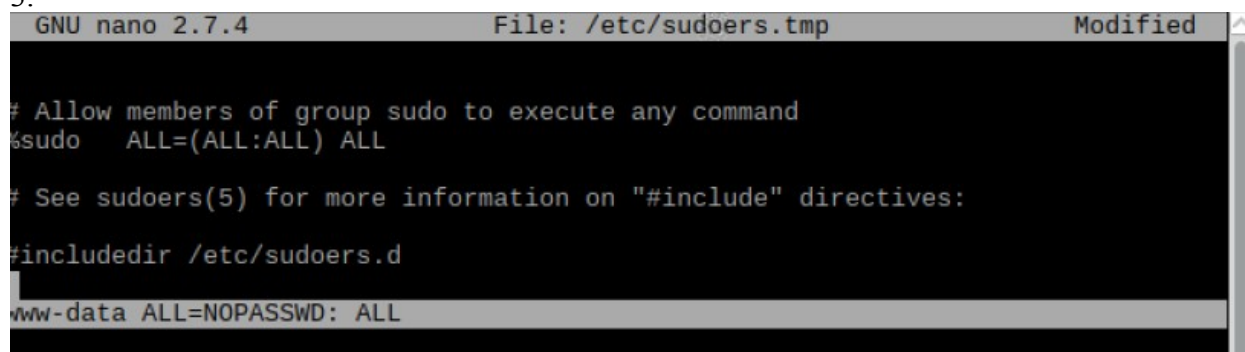
1. sudo apt-get install apache2 -y
2. sudo apt-get install php5 libapache2-mod-php5 -y
3. sudo apt-get install php5-curl
4. optional if you want to change apache server root directory
   1. sudo nano /etc/apache2/sites-enabled/000-default.conf
5. sudo /etc/init.d/apache2 reload

If you left default apache root path which is /var/www/html use below command to add pi user rights to this ridectory so you will be able to copy data there

1. sudo chown -R pi /var/www/html

We also give apache proper rights so it can run shell commands which will be needed when running AKHomeAutomation scripts

1. sudo visudo
2. add marked line to the ond of the file (www-data ALL=NOPASSWD: ALL)
3.


```
  GNU nano 2.7.4              File: /etc/sudoers.tmp                  Modified

 # Allow members of group sudo to execute any command
 %sudo   ALL=(ALL:ALL) ALL

 # See sudoers(5) for more information on "#include" directives:

 #includedir /etc/sudoers.d

 www-data ALL=NOPASSWD: ALL
```

   sudo gpasswd -a pi www-data
   1. this will add pi user to www-data group

AKHomeAutomation files deployment

Deploy files to Raspberry pi

Copy whole folder to Apache root folder /var/www/html

```
⊞ 📁 tmp
⊟ 📁 www
    ⊟ 📁 html
        ⊟ 📁 home
            ⊞ 📁 executables
            ⊞ 📁 graphics
            ⊞ 📁 inc
```
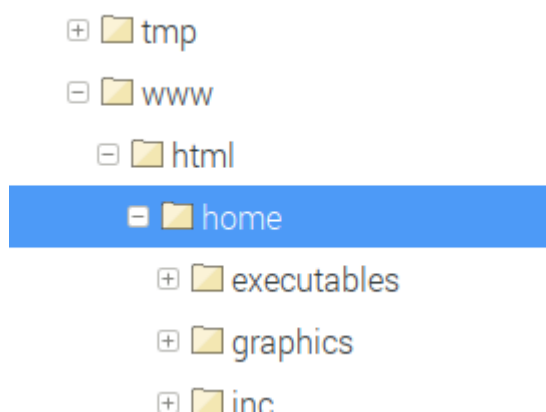
Modify settings file

Open /var/www/html/home/executables/settings.json

Change nodes visible below to match your server address and your paths

```
settings.json ✕
1  ⊟{
2      "webServerAddress":"http://192.168.0.99/home",
3      "satelliteServerAddresses":[],
4      "mainPath":"/var/www/html/home/",
5      "delayfilesPath":"/var/www/html/home/executables/delayfiles/",
6      "regularactionfilesPath":"/var/www/html/home/executables/regularactionfiles/",
7      "codeSendPath":"/var/www/html/home/RadioDevices/codesend",
8      "conradCodeSendPath":"/var/www/html/home/RadioDevices/codesendConrad",
```

Apply permissions to folders

Now we have to change ownership and permission levels to directories so scripts and apache can use them

1. sudo chown -R pi.www-data /var/www/html/home
2. sudo chown -R www-data.www-data /var/www/html/home/executables
3. sudo chmod -R 4755 /var/www/html/home

Compile signal sending files

Go to directory
– cd /var/www/html/home/RadioDevices

and perform „make" operation to recompile script files that will be used for transmitting signals. The results should look somehow like this. You will need to have wiringPi installed on your Raspberry pi for this compilation to succeed. If you do not have wiringPi you should install it by following these steps: http://wiringpi.com/download-and-install/

```
pi@sznapsarian1:/var/www/html/home/RadioDevices $ make
g++    -c -o RCSwitch.o RCSwitch.cpp
g++    RCSwitch.o send.o -o send -lwiringPi
g++    RCSwitch.o codesend.o -o codesend -lwiringPi
g++    RCSwitch.o codesendConrad.o -o codesendConrad -lwiringPi
g++    RCSwitch.o paringConrad.o -o paringConrad -lwiringPi
g++    RCSwitch.o receive.o -o receive -lwiringPi
g++    RCSwitch.o RFSniffer.o -o RFSniffer -lwiringPi
pi@sznapsarian1:/var/www/html/home/RadioDevices $
```
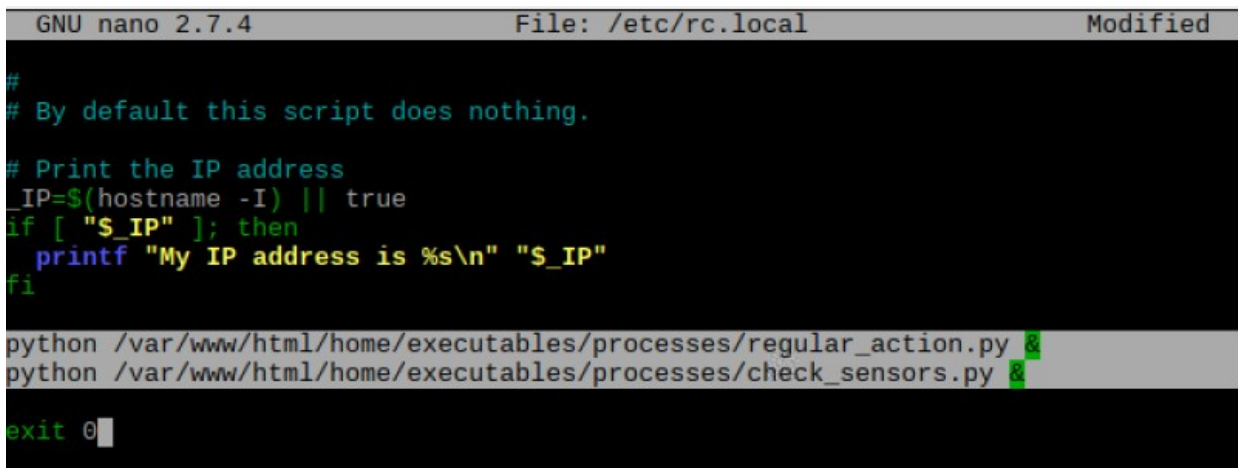
Setup startup scripts

There are two scripts in AKHomeAutomation that start on Raspberry Pi startup and perform tasks in background. These scripts take care of scheduler and sensors logic. To make them work perform the following steps. If you do not want to use scheduler tasks or sensors then just skip this point.

1. sudo nano /etc/rc.local

2. at the end of file (before „exit 0") add two lines marked on the screen. It is important that each of these lines ends with „ &"

    1. python /var/www/html/home/executables/processes/regular_action.py &

    2. python /var/www/html/home/executables/processes/check_sensors.py &

```
  GNU nano 2.7.4              File: /etc/rc.local                 Modified

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi


python /var/www/html/home/executables/processes/regular_action.py &
python /var/www/html/home/executables/processes/check_sensors.py &

exit 0
```

Radio Controlled Switches setup

At this point I assume that you deployed AKHomeAutomation files onto Raspberry Pi and also recompiled RFSniffer libraries as described in „Compile signal sending files" chapter and applied proper permissions to folders as described in „Apply permissions to folders" chapter.

How to set up radio controlled power switches I describe in vids that are on playlist below:

 – https://www.youtube.com/watch?
   v=C19ARWDYR3c&list=PLjd2MVjW6mhFygrvXyVcdNoq6pHK8MdUW

To proceed with reading your powerswitches signals and testing them using RFSniffer in chapters below you should have your radio transmitter (example: XY-FST) and radio receiver (example: XY-MK-5V) conneted to Raspberry Pi in a way shown in vids above.

In case of AKHomeAutomation system transmitter data pin should be connected to pin 11 (GPIO17) and receiver data pin should be connected to pin 13 (GPIO27).

Reading Radio controlled power switch signal using RFSniffer

Prepare your radio controlled power switch and remote that controls it.

Go to directory
- cd /var/www/html/home/RadioDevices

Type:
- sudo ./RFSniffer

This will initiate RFSniffer which will constantly read radio signals. Place your remote radio transmitter near radio receiver connected to Raspberry Pi (example: XY-MK-5V). Slowly and with some time gaps push buttons responsible for actions on your radio controlled power switch and observer what RFSniffer outputs. Push each button a couple of times to ensure that RFSniffer returns repeatadly same values for same button to eliminate any signal distortions. Write down codes returned by RFSniffer

The output should look somehow like this (in given example one button was pressed repeatedly. Since value 263505 repeats we will write it down as value that represents signal generated by this button):

```
pi@sznapsarian3:/var/www/html/home/RadioDevices $ sudo ./RFSniffer
Received 263505
Received pulse 190
Received 1070353
Received pulse 190
Received 263505
Received pulse 190
Received 263505
Received pulse 190
Received 267345
Received pulse 189
```

Manually testing radio signal using RFSniffer

Lets take code read in chapter before which is 263505 and test if this will work on our radio controlled powerswitch. Connect power switch to mains.

Go to directory
- cd /var/www/html/home/RadioDevices

Type:
1. sudo ./codesend 263505 (where 263505 is code enabling/disabling your device that you identified in previous chapter)

Check if this enabled radio controlled powerswitch. If it did you are good to go!

The ouptput should look somehow like this:

```
pi@sznapsarian3:/var/www/html/home/RadioDevices $ sudo ./codesend 263505
sending code[263505]
```

If it did not that is not the end of the world since the pulse signal that your device use may be different than codesend file is actually using.

For example AKHomeAutomation system uses different file for radio controlled powerswitches manufactured by Conrad company. You can check out contents of this file in /var/www/html/home/RadioDevices/codesendConrad.cpp. See that different pulse is set there which is suitable for Conrad devices (701). It might be that your devices need also different pulse. To find out simplies would be to google it and with luck find proper setting for RFSniffer. Then overwrite codesend.cpp (previously backing it up) or creating new file, then compile it using make and try again. Good luck!

Web Controller Switches Setup

AKHomeAutomation can also be configured with web controller switches. That means that unlike Radio Controlled switches described above, system will send web requests on disable and enable actions and web device receiving these requests will act accordingly. Many web devices or even services can be used and paired with AKHomeAutomation because as you will see below in node usage description this system is actually quite elastic regarding what you use.

If you decide to map something that is present on your network into AKHomeAutomation system you just need to defice this „something" in AKHomeAUtomation system by introducing its node in nodes.js file. Then provide its „address" property which will define what the address is. And then provide „On" and „Off" properties which define what arguments will be present in „on" and „off" requests going from AKHomeAUtomation system into this particular device.

For example if you configure device with data address: 192.1.1.1, codeOn: [„pin=1"], codeOff: „pin=0" then on „Enable" action AKHomeAUtomation will send the following request „http://192.1.1.1/?pin=1" and on „Disable" action AKHomeAUtomation will send the following request „http://192.1.1.1/?pin=0". So regarding web devices that you can pair with AKHomeAutomation it can be anything .. device, service.. you name it.

Web Controlled Electric switches

If you are thinking about web controlled electric switches there is also very nice solution. On the market you can get relatively cheap and very realiable **Sonoff TH01** and **TH10**, **TH16** web controlled electric switches. These switches are based on **ESP8266** wifi module. By default these switches come with Sonoff software which operates in the cloud and allows you to toggle these devices from your home or perhaps pair it with other systems.

However because it uses ESP8266 component which is well known among automation enthusiasts these devices can be easily flashed which allows you to get rid of original software and install your own .lua scripts which do exactly what you want them to do.

On my GitHub repository **https://github.com/Sznapsollo/AK_Sonoff_nodeMCU_LUA_automation** I uploaded .lua scripts that I use to pair Sonoff devices with AKHomeAutomation system. These scripts were defined in such way which allows AKHomeAutomation system to also know what is the current state of each device which is an advantage over radio controlled devices where you cannot do this.

AK_Sonoff_nodeMCU_LUA_automation repository also contains description of how to flash Sonoff devices, where to get necessary tools from and also link to Youtube video showcasing the whole process.

For more information how to define web controlled device in AKHomeAUtomationSystem please also check content in next chapter.

How to set up Sonoff web controlled power switches I describe in vids that are on playlist below:

- https://youtu.be/AlX1ZiVodwY
- https://www.youtube.com/watch?v=C19ARWDYR3c&list=PLjd2MVjW6mhFygrvXyVcdNoq6pHK8MdUW

AKHomeAutomation configuration

Settings.json file setup

Settings.json file is placed in the following location:

– **cd /var/www/html/home/executables/settings.json**

It contains basic settings of AKHomeAUtomation.

```json
{
    "webServerAddress":"http://88.88.88.88/home",
    "satelliteServerAddresses":[],
    "mainPath":"/var/www/html/home/",
    "delayfilesPath":"/var/www/html/home/executables/delayfiles/",
    "regularactionfilesPath":"/var/www/html/home/executables/regularactionfiles/",
    "sensorsettingsfilesPath":"/var/www/html/home/executables/sensorsettingsfiles/",
    "codeSendPath":"/var/www/html/home/RadioDevices/codesend",
    "conradCodeSendPath":"/var/www/html/home/RadioDevices/codesendConrad",
    "sensorAlarmMail":{
        "enableMailingOnAlarm":true,
        "fromaddr":"------@gmail.com",
        "toaddr":"------@gmail.com",
        "password":"------",
        "smtpServerAddress":"smtp.gmail.com",
        "smtpServerPort":587
    },
    "saveDailyLogsToFile": true,
    "saveDailySensorLogsToFile": true,
    "translations":[
        {
            "code":"homepage",
            "description":"Home"
        },
```

Settings.json nodes

– **webServerAddress** – this node contains web path to AKHomeAutomation which should be called from background running services (regular_actions, check_sensors) which were mentioned in „Setup startup scripts"

– **satelliteServerAddresses** – AKHomeAutomation allows to pass signals to also other machines on your network, which for example could be situated in different areas of your flat to increase signal range. This node would contain list of web paths to AKHomeAutomation on other machines. Please be advised that these other instances should not have satelliteServerAddresses containing initially calling machine as that would lead to some loops ;-). Initially this node is empty. If it contained some values it would look somehow like this:

– "**satelliteServerAddresses**":[
"http://x.x.x.1/home", "http://x.x.x.2/home", "http://x.x.x.3/home"
],

– **mainPath** – this node defines where AKHomeAutomation is deployed on local machine

– **delayfilesPath** – AKHomeAutomation allows for delayed devices disabling. The delay is actually managed by files which are created in folder defined in this node.

– **regularactionfilesPath** – AKHomeAutomation allows for setting up scheduler actions per each of your devices defining during which days and time they should work. User sets this up from webpage by clicking on calendar icon next to each device. Files that contain configuration set up by user will be stored and managed in folder defined by regularactionfilesPath setting.

- **sensorsettingsfilesPath** – if there are any sensors connected to AKHomeAutomation system, their initial setup is defined in nodes.js configuration file. However use can also change this setup for each sensor from UI. If such customization is performed then config files containing new settings will be generated and contained within directory defined in this setting.

- **codeSendPath** – this node defines where RFSniffer libraries codesend file responsible for transmitting radio devices is placed. This file is used for radio devices that will have sending option defined as "sendOption": 0

- **conradCodeSendPatht** - his node defines where RFSniffer libraries codesendConrad file responsible for transmitting radio devices is placed. This file is used for radio devices that will have sending option defined as "sendOption": 1. Please have alook at vids mentioned in Radio Controlled Switches setup for more ino on this

- **sensorAlarmMail** – this option will be used if you have some sensors defined in nodes.json and will want to use alarm option and sending mails on alarm.

  - **enableMailingOnAlarm** – if enabled will attempt to send mail on alarm event

  - **fromaddr** – source mail account address

  - **toaddr** – where the alarm mail should be send

  - **password** – password to source mail account

  - **smtpServerAddress** – smtp server of source mail account

  - **smtpServerPort** – port of smtp mail account

- **saveDailyLogsToFile** – if set to true it will create daily log file of all enable actions. This file will be places in executables/logs/actions folder and will be titled actions_YYYYMMDD.json (where YYYYMMDD will correspond to day date)

- **saveDailySensorLogsToFile** – if set to true it will create daily log file of all sensor enable actions. This file will be places in executables/logs/sensors folder and will be titled sensors_YYYYMMDD.json (where YYYYMMDD will correspond to day date)

- **translations** - this node contains strings that will be presented in webpage. They can be changed/translated here

Nodes.json file setup

Settings.json file is placed in the following location:
- cd /var/www/html/home/executables/nodes.json

It contains list of devices that will be used in AKHomeAUtomation.
It also contains list of sensors that could be connected to AKHomeAUtomation that would work if check_sensors.py is running. (see „Setup startup scripts" how to enable it)

Sensors

If you have connected HC SR501 PIR detector to your Raspberry PI and enabled check_sensros.py script to be run on system start (see „Setup startup scripts" how to enable it) then you can define sensors in node.js defining what devices should they trigger, what alarma at what timeframes etc.

sensros node is a list of sensor json objects which will be described below. We will use the following example to describe all properties.

```json
"sensors": [{
    "id": "pir_ben_dover",
    "header": "Pir Label",
    "pin": 32,
    "rebound": 20,
    "timeUnits":[
        {"timeStart":"00:00","timeEnd":"08:00","daysOfWeek":",0,1,2,3,4"},
        {"timeStart":"15:30","timeEnd":"23:59","daysOfWeek":",0,1,2,3,4"},
        {"timeStart":"00:00","timeEnd":"23:59","daysOfWeek":",5,6"}
    ],
    "alarmTimeUnits":[
        {"timeStart":"09:00","timeEnd":"15:00","daysOfWeek":",1,2,4"}
    ],
    "on": [
    {
        "id": "roomlamp",
        "delay": 600,
        "rebound": 0,
        "dependencyMethod": "checkLuminosity",
        "dependencyOperation": "lwr",
        "dependencyValue": 25
    }],
    "onAlarm": [
    {
        "id": "testswitch1",
        "delay": 60
    }]
}],
```

- **id** – unique identifier for each sensor provided in this listopad

- **header** – User friendly label for given sensor which will be used to present sensor data from UI.

- **pin** – pin to which sensor data output has been connected to Raspberry Pi

- **rebound** – in seconds value that determines when next signal will be analyzed after last valid signal

- **timeUnits** – list of settings specifying when sensor should be active to operate devices

    - **timeStart** – time the sensors starts being active

    - **timeEnd** – time sensor stops being active

    - **daysOfWeek** – days during sensor should be active and consider timeStart and timeEnd defined in same object.

- **alarmTimeUnits** – list of settings specifying when sensor should be active to operate trigger alarm

    - **time** settings are done same as in case of timeUnits

    - if sensor will trigger during timeFrame defined by this setting it can send mail if it has been enabled ins ettings.json

- **on** – list of devices that should be triggered when sensor detects something withing active „timeUnits"

    - **id** – device identifier that corresponds to existing entry in nodes.json->nodes collection

    - **delay** – delay after which device should be disabled. Can be negative in which case no delay would be applied. In case of 0 default delay for this device would be applied (from

mdevice configuration definition)

- **dependencyMethod** – this is optional and can be removed. It supports checkLuminosity methow which would read light levels if light sensor would be connected to Raspberry Pi pin #39

- **dependencyOperation** – this is optional and can be removed. Corresponds to dependencyMethod. Can have values „lwr", „grtr" which define logical condition between dependencyMethod and dependencyValue.

- **dependencyValue** – this is optional and can be removed. Contains value which is compared to dependencyMethod.

Nodes (devices)

Contains the list of devices that can be operated through AKHomeAUtomation.

There can be few device types defined depending on device sendOption properties as described in a moment. There can also be device defined which will enable/disable whole groups of devices.

Communication witch device is determined by **sendOption** property which can have the following values:

- 0 - this is radio controlled device. Signal will be send using file defined in settings.json → codeSendPath setting

- 1 - this is radio controlled device but controlled by different signal pulse length than 0. Signal will be send using file defined in settings.json → conradCodeSendPath setting

- 2 – this is web operated device. Such device should have „address" property defined which defines address of device. CodeOn and codeOff for such defvice should contain GET request parameters that will be included in request url

- 3 – this is wakeonlan device type that will allow to send wakeonlan signals to specified device by this node.

Now Lets see some examples of different device types and below I will exmplain all properties that we can assign to devices and what they do.

Example radio controlled device

```json
{
    "name": "kitchenlights",
    "image": "kitchenlights.jpg",
    "category": "general",
    "header": "Kitchen lights",
    "regularActions" : true,
    "codeOn": [
        44335
    ],
    "codeOff": 44365,
    "delay": 7200,
    "sendOption": 0,
    "enableOn": true,
    "enableOff": true
},
```

Example radio controlled device (alternative sendOption)

```json
{
    "name": "coffee",
    "image": "coffee.jpg",
    "category": "general",
    "header": "Coffee machine",
    "regularActions" : true,
    "codeOn": [
        966491648
    ],
    "codeOff": 899382784,
    "delay": 7200,
    "sendOption": 1,
    "enableOn": true,
    "enableOff": true,
    "notifySatellites": false
},
```

Example Web controlled device

```json
{
    "name": "humidfier",
    "category": "general",
    "header": "Humidifier",
    "regularActions" : true,
    "address": "http://192.168.20.121",
    "codeOn": [
        "pin=ON1"
    ],
    "codeOff": "pin=OFF1",
    "delay": 1800,
    "sendOption": 2,
    "enableOn": true,
    "enableOff": true,
    "codeDev": "---------"
},
```

Example group device

```json
{
    "name": "alllights",
    "image": "alllights.jpg",
    "category": "general",
    "header": "All lights",
    "questionOn": "Really? All lights?",
    "itemIDs": [
        "tvlamp",
        "roomlamp",
        "roomlampsecond",
        "kitchenlights",
        "lavalamp"
    ],
    "enableOn": true,
    "enableOff": true
},
```

Example confirm device

```
{
    "name": "tvstack",
    "category": "general",
    "header": "TV stack",
    "regularActions" : true,
    "questionOff": "Are you sure you want to turn it all off?",
    "codeOn": [
        12312
    ],
    "codeOff": 2612312313508,
    "delay": 0,
    "sendOption": 0,
    "enableOn": true,
    "enableOff": true,
    "codeDev": "-----"
}
```

Example wakeonlan device

```
{
    "name": "computertv",
    "image": "7.jpg",
    "category": "general",
    "header": "Computer TV",
    "regularActions" : true,
    "codeOn": [
        "44:8A:88:98:81:C7"
    ],
    "codeOff": "",
    "sendOption": 3,
    "enableOn": true,
    "enableOff": false,
    "codeDev": "--------"
},
```

Devices properties examplained

Ok lets go one by one property

– **name** – unique name for each device

– **image** – optional property. It holds image name that would be displayed next to device in UI. These images are taken from home\graphics\icons\ folder. There are some predefined icons there. If this property is not provided – no image would be displayed.

– **category** – web UI contains main page and also „Advanced" tab. This property when set to „general" will display device on main page and if set as „advanced" will display it in advanced tab.

– **header** – text that will be displayed for this device in web UI

- **questionOn** – optional property. Confirmation question that will be displayed in case of hitting enable button. If this property has value, the enable button will have ble color.

- **questionOff** – optional property. Confirmation question that will be displayed in case of hitting enable button. If this property has value, the disable button will have ble color..

- **address** – web address which is used for sendOption: 3 devices

- **codeOn** – collection of enabling codes for given device.

- **codeOff** – disabling code for given device

- **delay** – optional property. Default delay for given device after which delay device would be disabled (unless user sets different delay from UI). Value in seconds. Value -1 means there will be no delay applied. If this property is not set, the delay slider options that shop up when clicking on device header will no longer be active.

- **regularActions** - optional property. When set to „true", it will display calenar icon which is used for setting regular actions for particular device.

- **sendOption** – communication with device. Possible values 0/1/2/3 – described at the beginning ofthis section.

- **enableOn** – if true then Enable button for this device will be displayed in web UI

- **enableOff** – if true then Disable button for this device will be displayed in web UI

- **codeDev** – purely optional - just some comments for internal use of configurator

- **itemIDs** – list of device names that will be used for group device. Devices by these names should exist/be configured in the system.

Troubleshooting and exceptions

Sometimes there are problems. Here are some hints how to narrow down the reason for them.

In case there are some exceptions thrown during operation of python scripts log folder will be created containg exception files grouped by days (for one day there will be one exception file containing all excepons from that day).

Log folder will be situated in:

- **/var/www/html/home/logs**

If system does not seem to work but there are no exceptions in log file we can also try to debug server request response from within the javascript.

To do so open chrome web inspector and put brakepoint at place marked on screen below – then click on disable/enable on any device and check what response message contains – if there is some problem with apache or php this message could contain some hints to it.