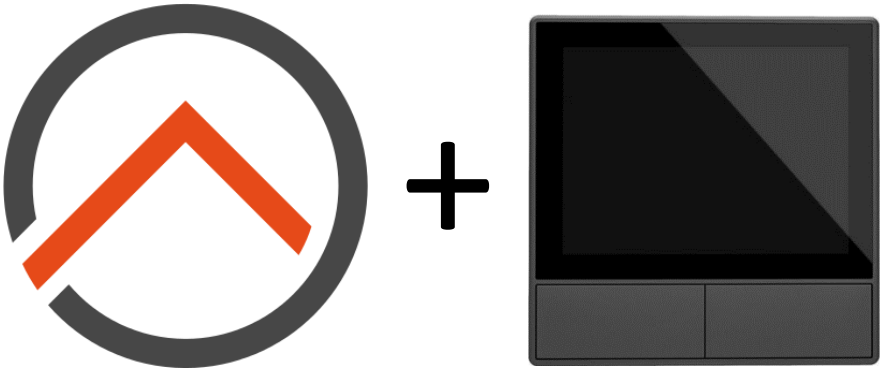
OpenHAB3 & NSPanel



Installation and configuration guide

Alf Pfeiffer, 2022-03-27

Table of content

1. Overview

This documentation describes the installation steps for how to flash a Sonoff NSPanel with Tasmota firmware and then to connect it to a OpenHAB3 system. The setup also assumes you would like to get weather information on the start panel.

I've read (and reread) all the posts on this topic on the OpenHAB forum and my finding is that most people – just like me – get stuck on 1. Get NSPanel to "talk" to OpenHAB and 2. Configuring the panels (screens) in NSPanel. For quick answer on 1, check picture in chapter 6.

Components used for the setup:

- A Windows PC to do the work on
- Raspberry Pi (minimum 3, recommended 4)
- A USB Serial Adapter
- Some cables to connect the USB serial adapter to the circuit board of the NSPanel.
- Sonoff NSPanel EU
- OpenHABian (v1.7.2), components needed:
    - Binding: **MQTT Binding**
    - Binding: **OpenWeatherMap Binding**
    - Add-on: **JSONpath Transformation**
    - Add-on: **RegEx Transformation**
    - Automation: **Groovy Scripting**
- Mosquitto MQTT broker (included in OpenHABian)
- Openweathermap cloud service

Disclaimer

Use this documentation at your own risk! The author assumes no responsibility of any mishaps resulting in your use of this documentation.

Acknowledgements

m-home (Mike) – For his initiative and appreciated efforts to bring NSPanel to OpenHAB

Blakadder – For creating a Tasmota firmware for NSPanel

Lewis Barclay – Especially this video which is the source for my flashing documentation (I actually suggest you use this for the flashing part and use my documentation only as a reference).

5. Post configuration of Tasmota on NSPanel

Post configuration of Tasmota on NSPanel after flashing to make it ready for integration with OpenHAB.

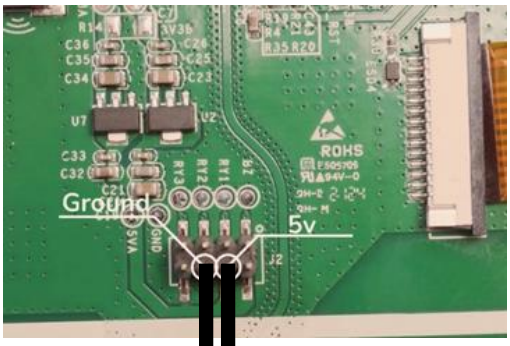
Post configuration steps after flashing.

Steps are:

1. Unplug the 3.3V power (disconnect USB from serial adapter)
2. On NSPanel: Plug 5V + GND on two bottom middle pins:
3. On USB Serial Adapter: Plug 5V + GND

Cabling for post configuration

NSPanel Circuit board

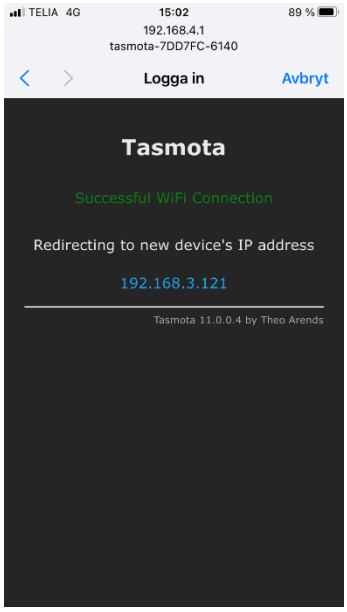


USB Serial Adapter

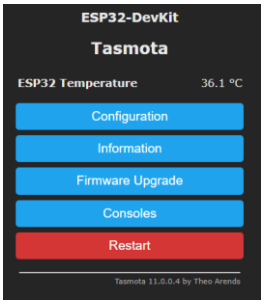
4. Power your USB serial adapter by plugging it in on your PC
5. A WiFi hotspot should now appear called e.g., “Tasmota7DD7FC-6140” (or something similar)
6. Connect to the WiFi hotspot (used iPhone for this, didn’t detect it on my PC..)

7. Put in WiFi SSID and password for you home WiFi and press “Save”:

and then this is shown:



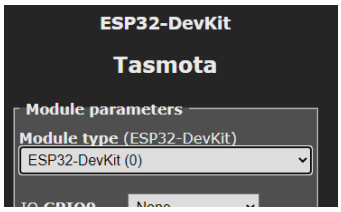
8. The NSPanel will now connect to your WiFi



9. Browse to the IP that is displayed (192.168.3.121):
10. Do some initial configuration:
    a. Select: Configuration
    b. Select: Configure Other
    c. Replace Template string with:

```
{"NAME":"NSPanel","GPIO":[0,0,0,0,3872,0,0,0,0,0,32,0,0,0,0,225,0,48
0,224,1,0,0,0,33,0,0,0,0,0,0,0,0,0,0,4736,0],"FLAG":0,"BASE":1,"CMND
":"ADCParam 2,11200,10000,3950 | Sleep 0 | BuzzerPWM 1"}
```

    d. Select: “Save” (Tasmota now reboots)
    e. The screen should now come alive!
2. One final change
    a. Select: Configuration
    b. Select: Configure Module
    c. Select: ESP32-DevKit (0):



    d. Select: “Save” (Tasmota now reboots)

At this stage you now have a running NSPanel that is ready to be integrated to OpenHAB 😊.

If you used the instruction from the Tasmota, this is also the cut-off point where you **use Mikes “nxpanel.be” file** instead of installing the “nspanel.be” file described in the Tasmota instruction.

6. Base setup of NSPanel-to-OpenHAB communication

This final step describes how the panel interface is adopted to work with OpenHAB. This is where the work from Mike comes into play. He has created a new “visual layout” of the panel (screen) which also supports several different panel types. The big advantage is that you with this change will be able to better adapt and extend the NSPanel to your home automation needs. I do not really understand how this actually “works”, just appreciate that it does and fits my purpose.

After the steps in this chapter, you will have:

- A new panel layout installed (Mikes)
- Base communication between NSPanel and OpenHAB setup established
- Customized the primary panel with on your OpenHAB items (temperature and weather)

Links and references

- Server/location hosting latest nxpanel.tft definition: Index of /nxpanel (proto.systems)
- Location of “nxpanel.be”, the panel definition file adapted for OpenHAB: ns-flash/berry at master · peepshow-21/ns-flash · GitHub

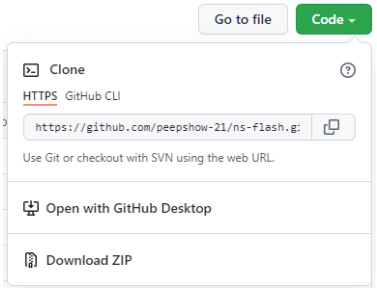
Preparations

Again, some preparations

Download an OpenHAB adopted “nxpanel.be”

Steps are:

- Download nxpanel.be from here: GitHub - peepshow-21/ns-flash
- Select: **Code**



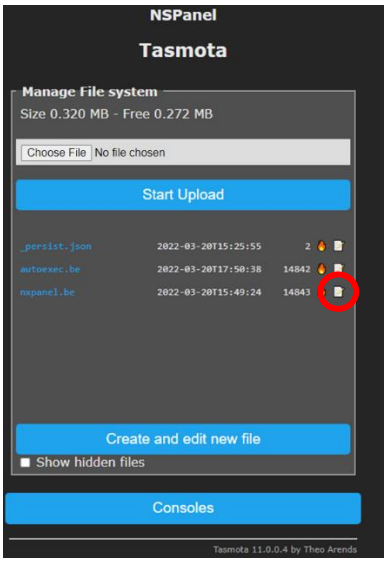
- Select: **Download ZIP**
- You will down download a file called “ns-flash-master.zip”
- Extract the file “ns-flash-master.zip\ns-flash-master\berry**nxpanel.be**” from this zip and put it in a directory. (there might be other ways to do this, but this is what I did...)

You are now ready to replace the panel definition file.

Installation and configuration

Next step is to install the new interface of NSPanel. Instead of using the “nspanel.be” file according to the Tasmota installation instruction, use the “nxpanel.be” file (see "Download an OpenHAB adopted “nxpanel.be”).

1. Browse to the IP-address of your NSPanel
2. The Tasmota web interface is now shown
3. Select: **Consoles**
4. Select: **Manage File System**
5. Select: **Choose File**
6. Browse to where you stored the file and Select: **nxpanel.be**

7. Select: **edit-icon** for nxpanel.be 

8. Rename the file to: **autoexec.be** 

9. Select: **Save**
10. Select: **Consoles**
11. Select: **Main menu**
12. Select: **Restart**
13. Select: **Consoles**
14. Select: **Console**
15. Type: **InstallNxPanel**
    a. NSPanel now starts flashing the “nxpanel-latest.tft” downloaded from this site: Index of /nxpanel (proto.systems), screen looks like this: 

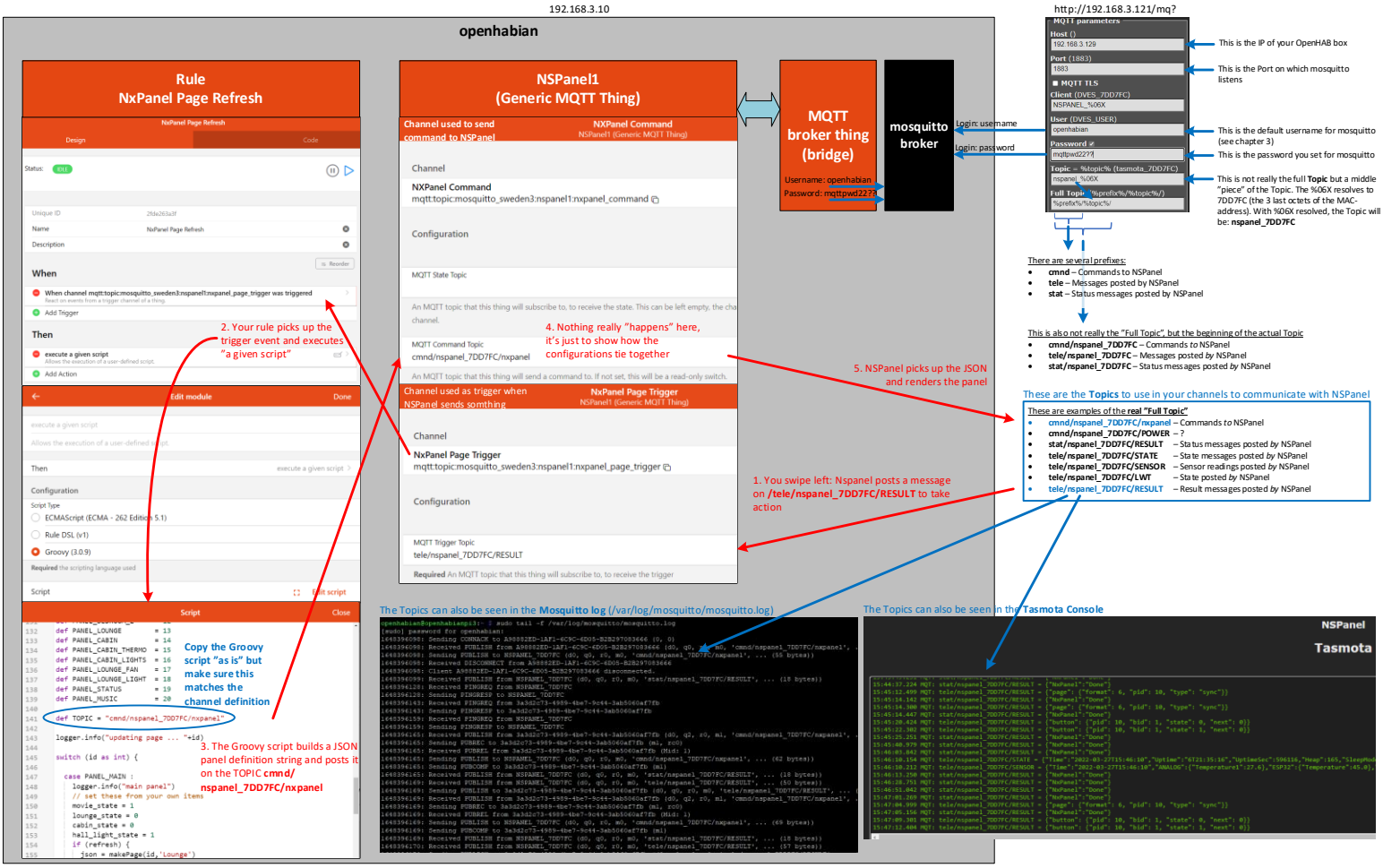
b. After restart, the panel now looks like this:

c. **This is a good place to be!** It's now time to connect the NSPanel to OpenHAB.

Connecting NSPanel with OpenHAB

To facilitate the understanding how this is all connected see picture below. Details of how to configure this will follow in the next sections. Legend:

- **Blue**: Configuration stuff
- **Red**: Execution flow



Enable logging!

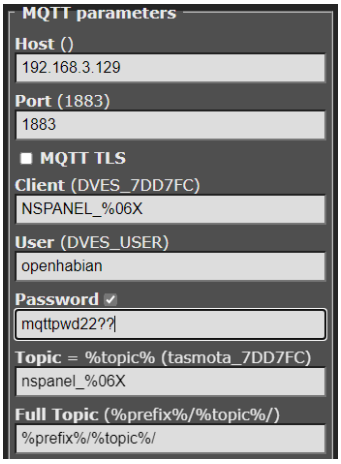
Before you begin configuring the connection, I suggest you prepare logging so you can monitor what's happening. Three logs of interest are:

- The normal **OpenHAB log** (frontail) available here: http://<your-openhab-IP>:9001
- The **mosquitto broker log** will full logging enabled (see end of section "**Fel! Hittar inte referenskälla.**" in Chapter **Fel! Hittar inte referenskälla.**). To look at the log:
    - Log on your openhab with putty (or any other ssh client)
    - Run the command: **sudo tail -f /var/log/mosquitto/mosquitto.log**
- The NSPanel Console log available here: http://<your-NSPanel-IP/cs?
    - Select: **Consoles**
    - Select: **Console**
    - Enter command: **weblog 4**
    - This turns on extended logging (to reset to normal, enter command: **weblog 2**)

Configure MQTT in NSPanel

This is where we configure the MQTT settings to start talking to the mosquito broker in OpenHAB.

1. Browse to the IP-address of your NSPanel
2. The Tasmota web interface is now shown
3. Select: **Configuration**
4. Select: **Configure MQTT**
5. Enter Host: **<IP of your OpenHAB>**
6. Enter Client: **NSPANEL_%06X** (don't actually think this is used somewhere)
7. Enter User: **openhabian** (the default user for Mosquitto)
8. Tick the box to the left of Password
9. Enter Password: **mqttpwd22??** (Must match the one you entered when installing Mosquitto)
10. Enter the Topic: **nspanel_%06X** (you can use anything, just make sure this matches everywhere)
11. When done the screen looks something like this:



12. Select: **Save**
13. After reboot, entries should now start to show in /var/log/mosquitto/mosquitto.log
14. This is good. Your NSPanel has successfully logged into your mosquito broker. But nothing will happen as no one is listening in OpenHAB yet…

Configuring MQTT in OpenHAB

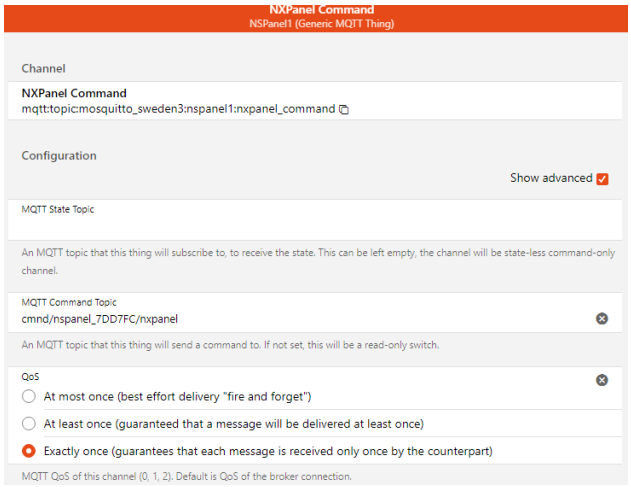
This is where we configure the MQTT settings in OpenHAB to be able to 1. Send commands to NSPanel and 2. To listen what NSPanel is posting to us. We will also create a rule that uses the template Groovy script from Mike just to get us started on getting our custom panels in place to confirm communication back and forth is working.

In short, we will configure:

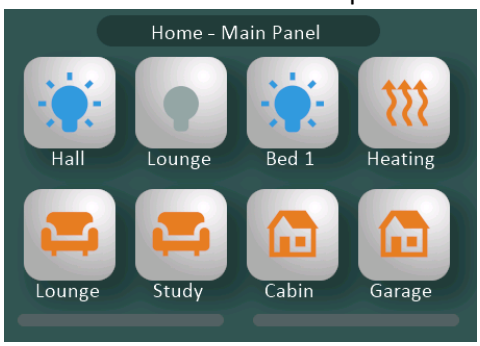
- A **Generic MQTT thing** representing our NSPanel
- Two **channels** for the above thing, one for receiving messages and one for sending commands to the NSPanel.
- One **rule** that triggers on received messages from NSPanel and sends commands back

Steps are:

- Log on as admin in the OpenHAB web interface.
- Select: **Settings**
- To create the MQTT Thing for NSPanel
    - Select: **Things** and press ”+”
    - Select: **MQTT Binding**
    - Select: **Generic MQTT Thing**
    - Enter a Label: **NSPanel1 (Generic MQTT Thing)**
    - Select Bridge: **MQTT Broker**
    - Select: **Save** (top right corner)
- To create the two channels for the above thing, we start with the command channel:
    - On the Things Menue, Select: **NSPanel1 (Generic MQTT Thing)**
    - Select: **Channels** (top middle)
    - Select: **Add Channel**
    - As Channel Identifier; Enter: **nxpanel_command**
    - As Label; Enter: **NXPanel Command**
    - Select: **Text Value**
    - Tick: **Show Advanced**
    - As MQTT Command Topic; Enter: **cmnd/nspanel_7DD7FC/nxpanel**
    - As QoS; Enter: **Exactly Once**
    - Select: **Save** (at the bottom)
    - After creation, the channel should look like this:



    - To create the trigger channel for **NSPanel1 (Generic MQTT Thing)**:
    - Select: **Channels** (top middle)
    - Select: **Add Channel**
    - As Channel Identifier; Enter: **nxpanel_trigger**
    - As Label; Enter: **NXPanel Trigger**
    - Select: **Trigger**
    - Tick: **Show Advanced**
    - As MQTT Command Topic; Enter: **tele/nspanel_7DD7FC/RESULT**

    - As QoS; Enter: **Exactly Once**
    - Select: **Save** (at the bottom)
- Only the rule left to configure
    - Select: **Settings**
    - Select: **Rules** and press ”+”
    - As Name Enter: **NXPanel Trigger Rule**
    - Select: **Add Trigger**
    - Select: **Thing Event**
    - Select: **NSPanel1 (Generic MQTT Thing)**
    - Select: **A trigger channel fired**
    - Select: **Done** (top right)
    - Select: **Add Action**
    - Select: **Run Script**
    - Select: **Groovy** (remember to have installed the Groovy Automation)
    - Cut and Paste Mikes default Grovy script, you can either pick it from section “Mikes Groovy script” below or from the community post
    - **Important!** After adding the script code: **Go to line 141** (the one that says “def TOPIC = "cmnd/nxpanel/nxpanel") and replace the Topic with “**cmnd/nspanel_7DD7FC/nxpanel”**. If you don’t, the script will post the response on the wrong Topic.
    - Select: **Save (Ctrl-S)** (top right corner).
- **Done!** Swipe left on Your NSPanel and Mikes test panel should now be displayed, the first panel looks like this: 

If this does not work, check your logs that the “topics” are all correctly matched. You will most probably have another topic compared with the one above as this is based on the MAC address on my NSPanel.

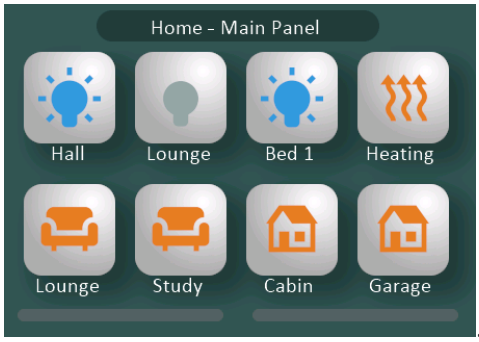
How it works

This is about what is sent between OpenHaB and NSPanel and assumes you have a working connection.

When you swipe left, the NSPanel posts the JSON in blue to OpenHAB.

```
2022-03-27 21:02:48.967 [INFO ] [openhab.event.ChannelTriggeredEvent ] -
mqtt:topic:mosquitto_sweden3:nspanel1:nxpanel_page_trigger triggered {"page": {"format": 6, "pid": 10,
"type": "sync"}}
2022-03-27 21:02:48.974 [INFO ] [org.openhab.core.automation.nspanel ] - Demo page rules called
2022-03-27 21:02:48.978 [INFO ] [org.openhab.core.automation.nspanel ] - updating page ... 10
2022-03-27 21:02:48.981 [INFO ] [org.openhab.core.automation.nspanel ] - main panel
2022-03-27 21:02:48.988 [INFO ] [org.openhab.core.automation.nspanel ] - rule done
```

The rule you created is triggered and the action for the rule is to run the Groovy script.

The piece: `"pid": 10` (pid = Panel ID) tells the script to render the panel with ID 10. This panel looks like this: 

.

The script posts the following JSON in response to NSPanel which renders the panel:


```
{"refresh":{"pid":10,"name":"Lounge",6buttons:[{"bid":1,"label":"Movie","type":1,"state":1,"ic
on":1},{"bid":2,"label":"Lounge","type":1,"state":0,"icon":1},{"bid":3,"label":"Hall","type":2
,"icon":6},{"bid":4,"label":"Bedroom","type":10,"next":11,"state":5,"icon":5},{"bid":5,"label"
:"Temp","type":10,"next":15,"state":9,"icon":9},{"bid":6,"label":"Light","type":3,"next":18,"s
tate":1,"icon":2},{"bid":7,"label":"Dimmer","type":4,"next":16,"state":0,"icon":3},{"bid":8,"l
abel":"Status","type":10,"next":19,"state":15,"icon":16}]}}
```


Or if formatted in a bit more readable form:


```
{
   "refresh":{
      "pid":10,
      "name":"Lounge",
      "6buttons":[
         {
            "bid":1,
            "label":"Movie",
            "type":1,
            "state":1,
            "icon":1
         },
         {
            "bid":2,
            "label":"Lounge",
            "type":1,
            "state":0,
            "icon":1
         },
         {
            "bid":3,
            "label":"Hall",
            "type":2,
            "icon":6
         },
         {
            "bid":4,
            "label":"Bedroom",
            "type":10,
            "next":11,
            "state":5,
            "icon":5
         },
         {
            "bid":5,
            "label":"Temp",
            "type":10,
            "next":15,
            "state":9,
            "icon":9
         },
         {
            "bid":6,
            "label":"Light",
```

```
            "type":3,
            "next":18,
            "state":1,
            "icon":2
         },
         {
            "bid":7,
            "label":"Dimmer",
            "type":4,
            "next":16,
            "state":0,
            "icon":3
         },
         {
            "bid":8,
            "label":"Status",
            "type":10,
            "next":19,
            "state":15,
            "icon":16
         }
      ]
   }
}
```

Update values on first screen

<still working on this>

Mikes Groovy script

```
import org.slf4j.LoggerFactory

def PAGE_HOME            = 1
def PAGE_2_BUTTON        = 2
def PAGE_3_BUTTON        = 3
def PAGE_4_BUTTON        = 4
def PAGE_6_BUTTON        = 5
def PAGE_8_BUTTON        = 6
def PAGE_DIMMER          = 7
def PAGE_DIMMER_COLOR    = 8
def PAGE_THERMOSTAT      = 9
def PAGE_ALERT_1         = 10
def PAGE_ALERT_2         = 11
def PAGE_ALARM           = 12
def PAGE_MEDIA           = 13
def PAGE_PLAYLIST        = 14
def PAGE_STATUS          = 15

def BUTTON_UNUSED        = 0
def BUTTON_TOGGLE        = 1
def BUTTON_PUSH          = 2
def BUTTON_DIMMER        = 3
def BUTTON_DIMMER_COLOR  = 4
def BUTTON_PAGE          = 10

def ICON_BLANK           = 0
def ICON_BULB            = 1
def ICON_DIMMER          = 2
def ICON_DIMMER_COLOR    = 3
def ICON_VACUUM          = 4
def ICON_BED             = 5
def ICON_HOUSE           = 6
def ICON_SOFA            = 7
def ICON_BELL            = 8
def ICON_HEAT            = 9
def ICON_CURTAINS        = 10
def ICON_MUSIC           = 11
def ICON_BINARY          = 12
def ICON_FAN             = 13
```

```
def ICON_SWITCH          = 14
def ICON_TALK            = 15
def ICON_INFO            = 16

def NONE                 = 0

def logger = LoggerFactory.getLogger("org.openhab.core.automation.nspanel")

def mqtt = actions.get("mqtt","mqtt:broker:mqtt_broker")

def str = event.getEvent()

logger.info("Demo page rules called")

if (str.indexOf('{"page":')!=0) {
  return
}

/*
 * Utility functions - start
 */

def makeButton(bid,label,type,icon=null,state=null,next=null) {
  var str = ""<<((bid==1)?"":",")
  str<<'{"bid":'<<bid<<',"label":"'<<label<<'","type":'<<type
  if (next!=null) {
    str<<',"next":'<<next
  }
  if (state!=null) {
    str<<',"state":'<<state
  }
  if (icon!=null) {
    str<<',"icon":'<<icon
  }
  str<<'}'
  return str
}

def makePage(pid,name) {
  var str = new StringBuilder('{"refresh":')
  str<<'{"pid":'<<pid<<',"name":"'<<name<<'",'
  return str
}

def makeEmptySync(pid) {
  var str = new StringBuilder('{"sync":')
  str<<'{"pid":'<<pid<<'}}'
  return str
}

def makeEmptyRefresh(pid) {
  var str = new StringBuilder('{"refresh":')
  str<<'{"pid":'<<pid<<'}}'
  return str
}

def makeSyncButtonStart(pid,bid,state) {
  var str = new StringBuilder('{"sync":')
  str<<'{"pid":'<<pid
  str<<',buttons:[{"bid":'<<bid<<',"state":'<<state<<'}'
  return str
}

def addSyncButton(bid,state) {
  var str = ',{"bid":'<<bid<<',"state":'<<state<<'}'
  return str
}

/*
 * Utility functions - end
 */
```

```
/*
 * Get data from the page message
 * (would be good to use JsonSluper here but currently can't access)
 */

var i = str.indexOf("\"pid\"")
var i2 = str.indexOf(",",i+7)
var id = str.substring(i+7,i2)
i = str.indexOf("\"format\"")
i2 = str.indexOf(",",i+10)
var format = str.substring(i+10,i2)

// check if a full refresh or just a status update
var refresh = str.indexOf("refresh")>0

var json

def PANEL_MAIN          = 10
def PANEL_BEDROOM_1     = 11
def PANEL_BEDROOM_2     = 12
def PANEL_LOUNGE        = 13
def PANEL_CABIN         = 14
def PANEL_CABIN_THERMO  = 15
def PANEL_CABIN_LIGHTS  = 16
def PANEL_LOUNGE_FAN    = 17
def PANEL_LOUNGE_LIGHT  = 18
def PANEL_STATUS        = 19
def PANEL_MUSIC         = 20

def TOPIC = "cmnd/nspanel/nxpanel"

logger.info("updating page ... "+id)

switch (id as int) {

  case PANEL_MAIN :
    logger.info("main panel")
    // set these from your own items
    movie_state = 1
    lounge_state = 0
    cabin_state = 0
    hall_light_state = 1
    if (refresh) {
      json = makePage(id,'Lounge')
      json<<format<<'buttons:['
      json<<makeButton(1,"Movie",BUTTON_TOGGLE,ICON_BULB,movie_state)
      json<<makeButton(2,"Lounge",BUTTON_TOGGLE,ICON_BULB,lounge_state)
      json<<makeButton(3,"Hall",BUTTON_PUSH,ICON_HOUSE)
      json<<makeButton(4,"Bedroom",BUTTON_PAGE,ICON_BED,PAGE_6_BUTTON,PANEL_BEDROOM_1)
      json<<makeButton(5,"Temp",BUTTON_PAGE,ICON_HEAT,PAGE_THERMOSTAT,PANEL_CABIN_THERMO)

json<<makeButton(6,"Light",BUTTON_DIMMER,ICON_DIMMER,hall_light_state,PANEL_LOUNGE_LIGHT)

json<<makeButton(7,"Dimmer",BUTTON_DIMMER_COLOR,ICON_DIMMER_COLOR,cabin_state,PANEL_CABIN_LIGH
TS)
      json<<makeButton(8,"Status",BUTTON_PAGE,ICON_INFO,PAGE_STATUS,PANEL_STATUS)
      json<<"]}}"
    } else {
      json = makeSyncButtonStart(id,1,movie_state)
      json<<addSyncButton(2,lounge_state)
      json<<"]}}"
    }
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_BEDROOM_1 :
    // set these from your own items
    fan_state = 1
    if (refresh) {
      json = makePage(id,'Bedroom 1')
      json<<format<<'buttons:['
```

```
      json<<makeButton(1,"A",BUTTON_PUSH,ICON_HOUSE)
      json<<makeButton(2,"Fan",BUTTON_DIMMER,ICON_FAN,fan_state,PANEL_LOUNGE_FAN)
      json<<makeButton(3,"C",BUTTON_PUSH,ICON_SOFA)
      json<<makeButton(4,"Music",BUTTON_PAGE,ICON_MUSIC,PAGE_MEDIA,PANEL_MUSIC)
      json<<makeButton(5,"D",BUTTON_PUSH,ICON_TALK)
      json<<makeButton(6,"Alarm",BUTTON_PAGE,ICON_BELL,PAGE_ALARM,NONE)
      json<<"]}}"
    } else {
      json = makeEmptySync(id)
    }
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_BEDROOM_2 :
    json = makeEmptySync(id)
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_LOUNGE :
    json = makeEmptySync(id)
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_CABIN :
    json = makePage(id,'Cabin')
    json<<"}}"
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_CABIN_THERMO :
    // set these from your own items
    var heater = 1
    var auto = 0
    var temp = 15
    var set = 21
    json = makePage(id,'Cabin')
    json<<format<<',"therm":{'
    json<<'"set":'<<set<<',"temp":'<<temp<<',"heat":'<<heater<<',"state":'<<auto<<'"'
    json<<"}}"
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_CABIN_LIGHTS :
    json = makePage(id,'Cabin Lights')
    json<<'"power":'<<ON<<',"hsbcolor":'<<'"10,100,50"'
    json<<"}}"
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_LOUNGE_FAN :
    // set these from your own items
    fan_state = ON
    fan_setting = 3
    json = makePage(id,'Lounge Fan')

json<<'"power":'<<fan_state<<',"min":'<<1<<',"max":'<<4<<',"icon":'<<ICON_FAN<<',"dimmer":'<<f
an_setting
    json<<"}}"
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_LOUNGE_LIGHT :
    json = makePage(id,'Lounge Light')
    json<<'"power":'<<ON<<',"dimmer":'<<30
    json<<"}}"
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  case PANEL_STATUS :
    json = makePage(id,'System Status')
    json<<'"status":['
    json<<'{"id":'<<1<<',"text":'<<'"Gate":'<<',"value":'<<'"Open"'<<',"color":'<<2<<'}'
    json<<','
    json<<'{"id":'<<2<<',"text":'<<'"Window":'<<',"value":'<<',"Shut"'<<',"color":'<<3<<'}'
    json<<','
    json<<'{"id":'<<5<<',"text":'<<'"Room Temp":'<<',"value":'<<',"20°C"'<<'}'
    json<<']}}'
    mqtt.publishMQTT(TOPIC, json.toString())
    break
```

```
  case PANEL_MUSIC :
    json = makePage(id,'Sonos Player')
    // set these from your own items
    json<<'"artist":'<<'"New Order"'<<',"album":'<<'"Movement"'<<',"track":'<<'"Power
Play"'<<',"volume":'<<70
    json<<"}}"
    mqtt.publishMQTT(TOPIC, json.toString())
    break
  default :
    logger.info("unknown page!")
    break

}

logger.info("rule done")
```

7. Custom panel configuration

<still working on this>