Work in Progress

Lighthouse^{58}

Internet of Things Learning Case

by rwblinn.de

30/09/2016

Projectname: LiHo58

Objectives

- To develop an IoT learning case
- To control objects containing devices
  - An object is for example a house or a group of houses
  - A device is for example a light, switch, servo motor, sensor (e.q. temperature, illuminance, contact, motion) or virtual (e.q. weather request).
  - Each device has a set of actions to control, e.g. light = on | off | state | blink(n)
- To experiment with hardware, sensors, software
- To build objects with devices
- To design a messaging concept
- To apply learning's from previous IoT experiments
- To explore, learn & share
- To have fun

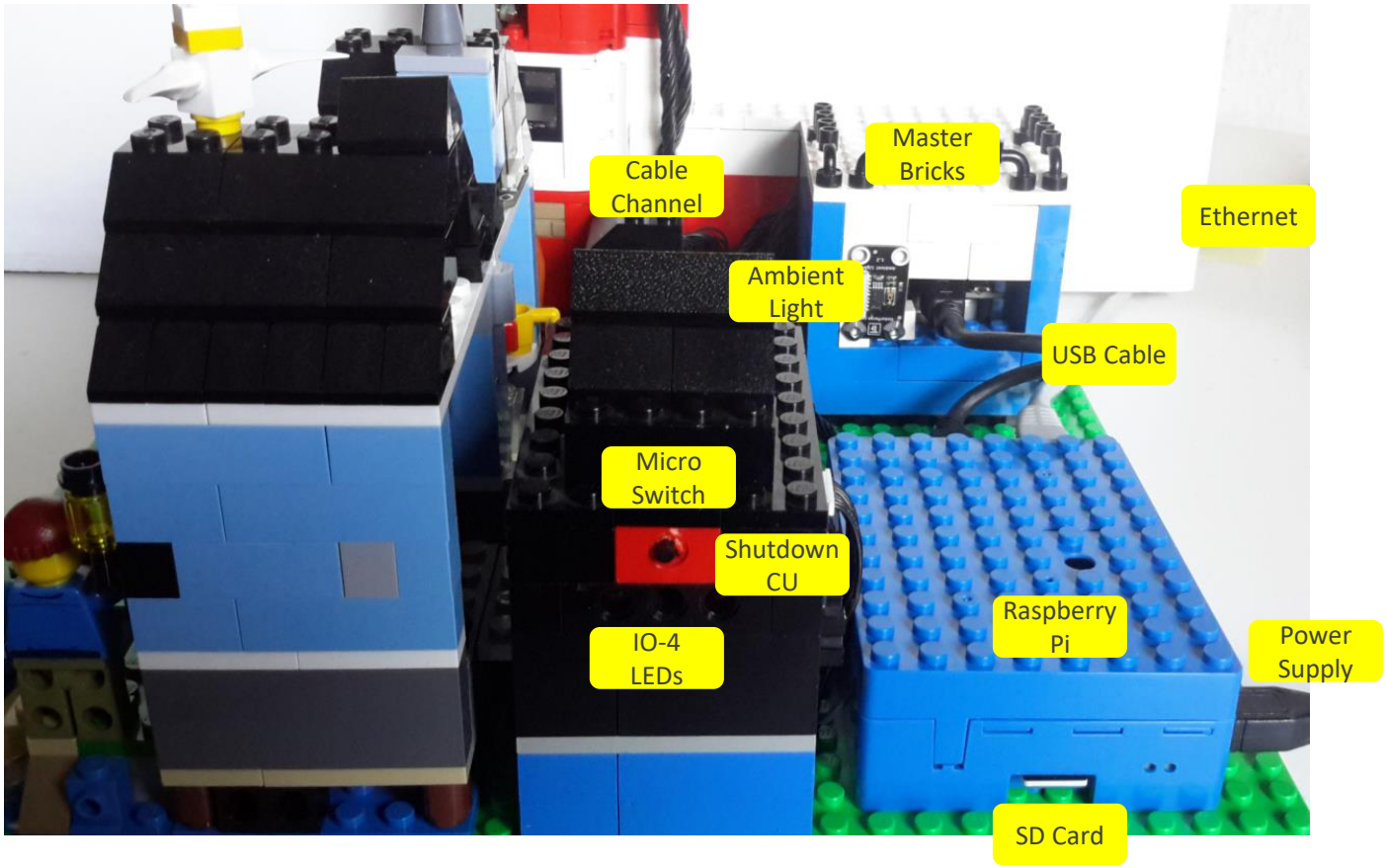
Solution Overview

It is planned to extend the solution after proof of concept.

Lighthouse[58] is based upon the LEGO® Lighthouse Point 31051 which has been extended:

- **Integrated are Objects** with devices, e.g. top light, weather display, motion detector, out-/indoor lights with dimmer, ambient light.
- **Browser & Windows Dashboard** to control the devices.
- **Weather Data** for the selected Lighthouse requested from Weatherunderground.
- **Hardware using** Raspberry Pi with TinkerForge Bricks & Bricklets.

Solution Overview – Control Unit



Resources

- **Hardware**
  - Central Control: Raspberry Pi
  - Device Control: TinkerForge Master Bricks
  - Devices: TinkerForge Bricklets
- **Software**
  - Central Control: Raspian Jessie with Node-RED , B4J Windows Client
  - Device Control: JavaScript, Python
- **Communication**
  - LAN (WiFi)
- **Messaging**
  - Message Queue Telemetry Transport (MQTT), Mosquitto message broker, TinkerForge Brick MQTT Proxy, JavaScript Object Notation (JSON)
- **Objects**
  - Build with LEGO

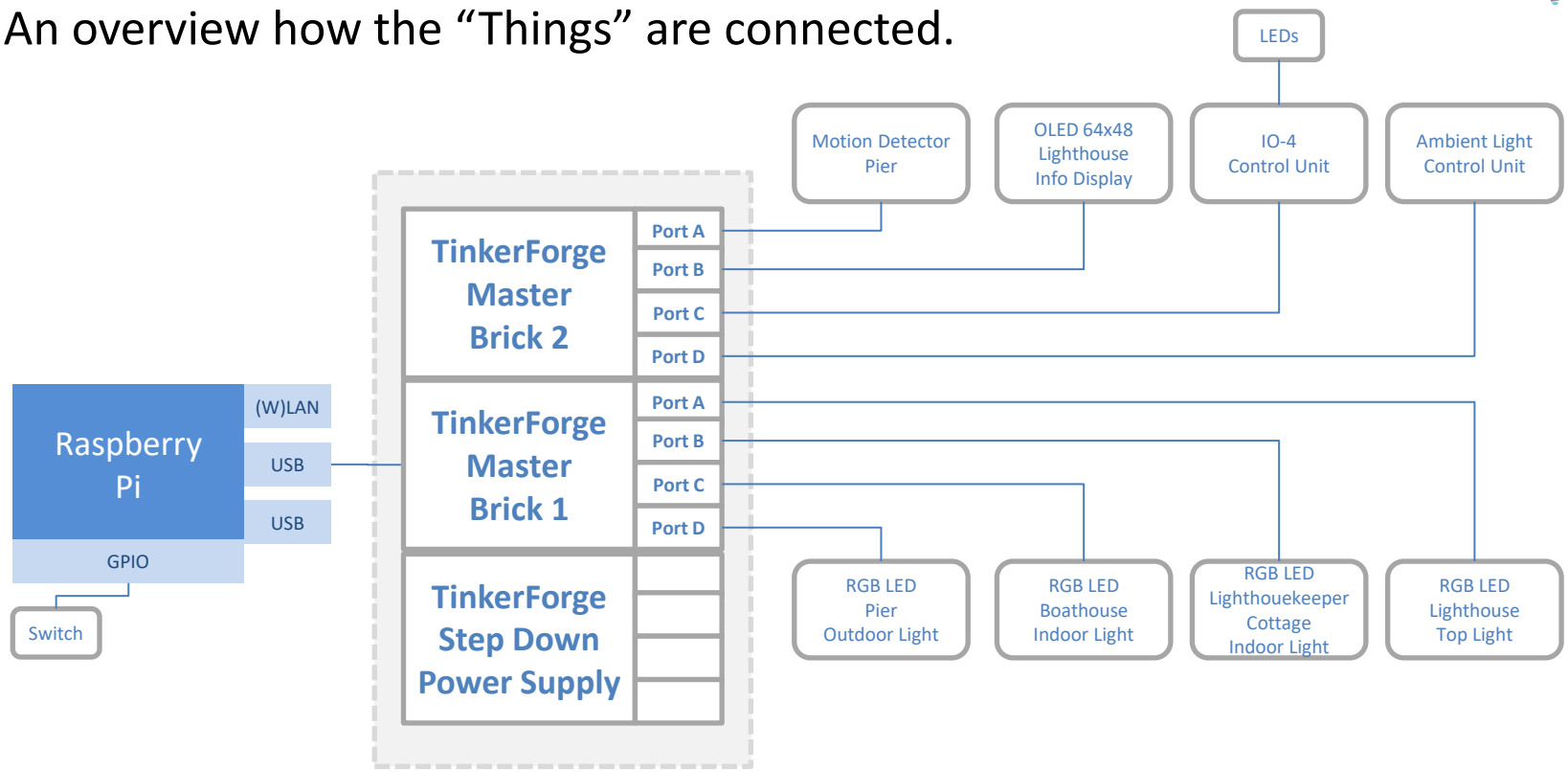
(LEGO® is a trademark and/or copyright of the LEGO® Group)

Functionality

- Lighthouse
  - Top Light Flashing based on the properties of the selected Lighthouse.
  - Information Display showing a clock & weather information (temperature, airpressure, beaufort, wind direction) measured at the nearest lighthouse lat/lon position.
- Pier
  - Motion Detector with Dashboard indicator Start & Stop time.
  - Outdoor light switch, dimmer and red, green or blue color setting.
- Lightkeeper Cottage
  - Room light switch, dimmer and red, green or blue color setting.
- Boathouse
  - Room light switch, dimmer and red, green or blue color setting.
- Control Unit
  - Ambient Light measuring illuminance (in Lux).
  - Shutdown the Control Unit via Dashboard or Micro Switch.
  - Status indicators (Trend, Alarm) for CPU °C, Disc Space Used, Memory Used
- Weather Underground data every hour for the Lighthouse position and display on the Dashboard.

Connecting the “Things”

An overview how the “Things” are connected.



Central Control

Device Control

Devices [TinkerForge Bricklets]

Browser Access: RPi-IP_Address:1880 (development) & /ui (dashboard)

Objects & Devices

Lighthouse	Pier	Lightkeeper Cottage	Boathouse	Control Unit
Top Light RGB LED UID: zJW Position: A	Outdoor Light RGB LED UID: zPN Position: D	Indoor Light RGB LED UID: zNx Position: B	Indoor Light RGB LED UID: zMT Position: C	Control RGB LEDs Master Brick 1 UID:6JKVjS Position: 0
Information Display OLED 64x48 UID: x6y Position: B	Detect Motion Motion Detector UID: wi9 Position: A			Control Various Master Brick 2 UID:6e77gm Position: 1
				Illuminance Ambient Light UID: mdh Position: D
				IO-Control IO-4 UID: gVf Position: C

List of TinkerForge Devices

Listed using the TinkerForge Brick Viewer

Name	UID	Positior	FW Version
Master Brick 2.1	6JKVjS	0	2.4.1
RGB LED Bricklet	zJW	A	2.0.1
RGB LED Bricklet	zNx	B	2.0.1
RGB LED Bricklet	zMT	C	2.0.1
RGB LED Bricklet	zPN	D	2.0.1
Master Brick 2.1	6e77gm	1	2.4.1
Motion Detector Bricklet	wi9	A	2.0.0
OLED 64x48 Bricklet	x6y	B	2.0.0
IO-4 Bricklet	gVf	C	2.0.1
Ambient Light Bricklet	mdh	D	2.0.3

Legend

Find more about TinkerForge here.

Table Information:

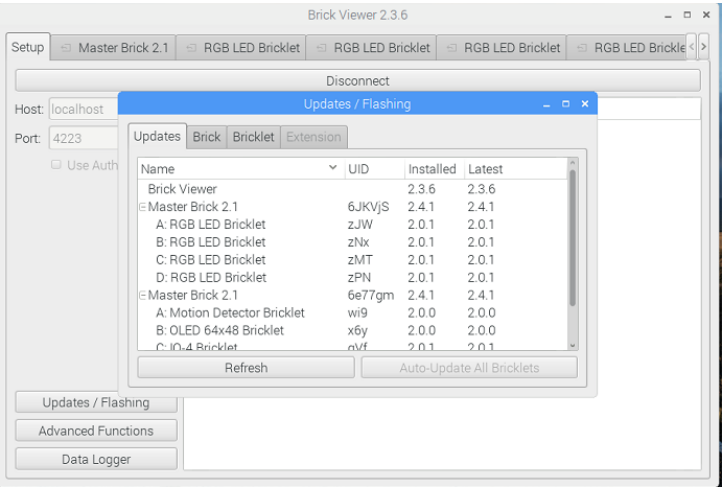
- Function
- TinkerForge Brick or Bricklet
- Unique UID
- Connected Position Master Brick

TinkerForge

TinkerForge is an open source hardware platform of stackable microcontroller building blocks (Bricks) used to control different modules (Bricklets).
The hardware can be controlled by external programs written in various languages, like C, Delphi, Java etc.

Lighthouse58 makes use of the Brick MQTT Proxy to control the Master Bricks with its connected Bricklets. Lookup for Brick MQTT Proxy updates here.
Hint: The Python source code is also useful to explore topic getters/setters.

The TinkerForge Brick Viewer is used to check & update the firmware of the Master Bricks & Bricklets.

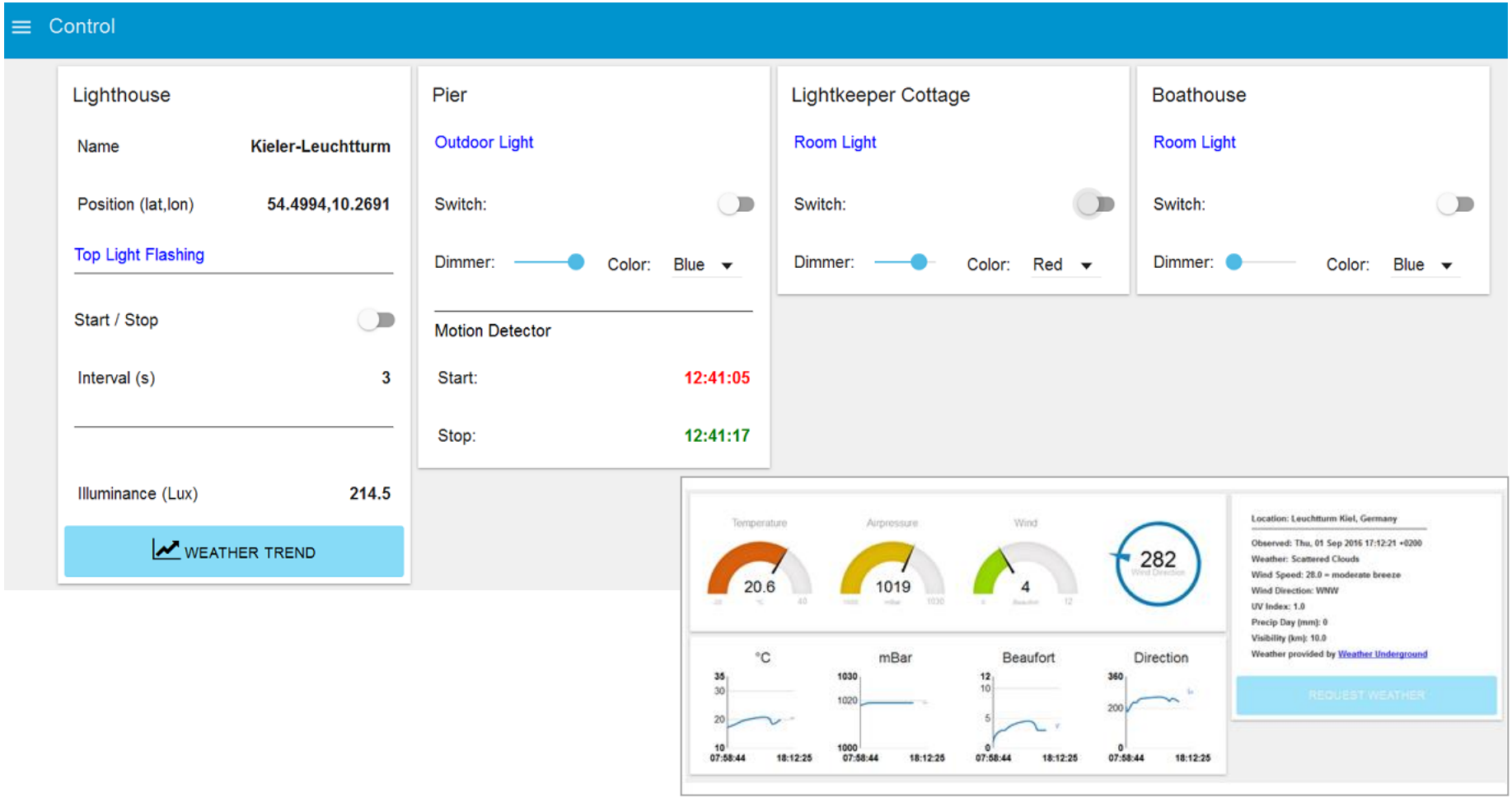


Name	UID	Positior	FW Version
Master Brick 2.1	6JKVjS	0	2.4.1
RGB LED Bricklet	zJW	A	2.0.1
RGB LED Bricklet	zNx	B	2.0.1
RGB LED Bricklet	zMT	C	2.0.1
RGB LED Bricklet	zPN	D	2.0.1
Master Brick 2.1	6e77gm	1	2.4.1
Motion Detector Bricklet	wi9	A	2.0.0
OLED 64x48 Bricklet	x6y	B	2.0.0
IO-4 Bricklet	gVf	C	2.0.1
Ambient Light Bricklet	mdh	D	2.0.3

Lighthouse58 Browser Dashboard solution.

NODE-RED

Node-RED – Browser Dashboard

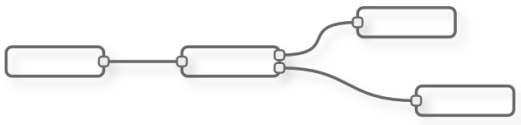


Lighthouse58 Dashboard running in a Browser

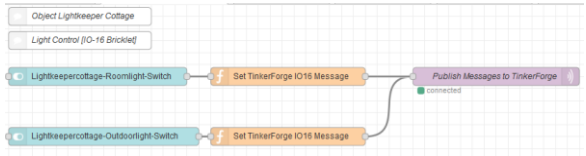
Node-RED Introduction

What

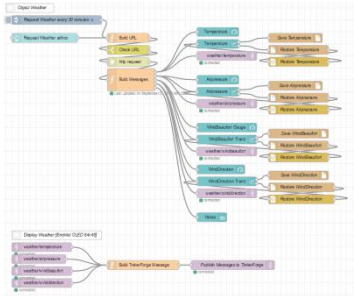
Node-RED, an open source visual tool for wiring the Internet of Things created by IBM Emerging Technologies .
As of the November 2015 version of the Raspberry Pi OS Raspbian Jessie, Node-RED is preinstalled.



Simple Flow Lightkeeper Cottage



Complex Flow Weather



How

As Node-RED is core for developing **Lighthouse**[58], the documentation is structured by the Flows defined.

Notes

- The intention of this show case is not to explain Node-RED. Recommend to visit the Node-RED homepage to learn more.
- The author is a
  - Node-RED NewBie – the solutions are evolving whilst learning.
  - Chemical Engineer used to work with flow charts for plants – means Node-RED fits very well in my mental flow model = instead of fluids, messages are flowing ☺

Node-RED AddOns

Following modules are used, in addition to the Node-RED core installation.

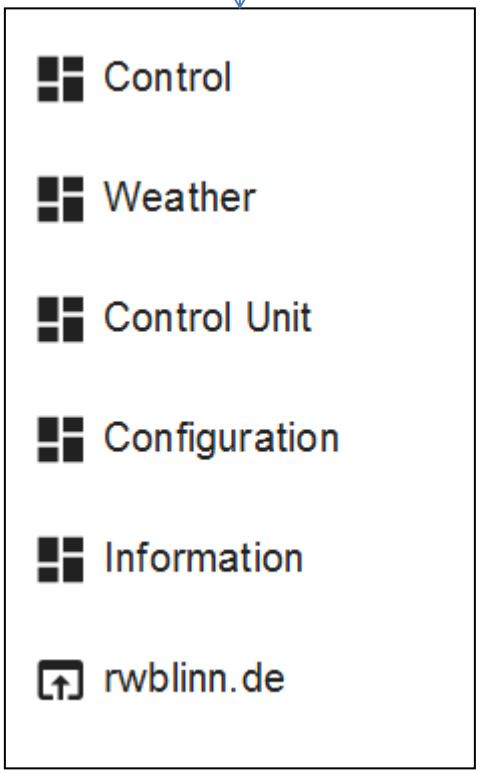
AddOn	Reference	Install	Notes
Dashboard	https://github.com/node-red/node-red-dashboard	npm install node-red-dashboard	
Beaufort	https://www.npmjs.com/package/beaufort	npm install beaufort	beaufort scale convertor for node.js.
		Prior installation ensure to change to the Node-RED folder, e.g. cd /home/pi/.node-red	

Node-RED Flows

An overview of the Flows defined = one Flow per Object.

The list is build according the Dashboard Menu

- Control
  - Lighthouse
  - Pier
  - Lightkeeper Cottage
  - Boathouse
- Weather
- Control Unit
- Configuration
- Information



Node-RED Flow Lighthouse – Toplight Flashing

Functionality

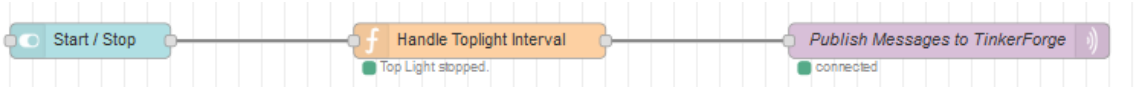
- Control flashing of the top light at regular intervalls set by the selected lighthouse.

Solution

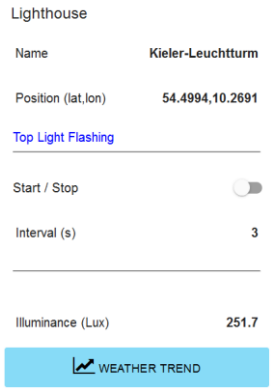
- TinkerForge RGB LED bricklet integrated in the top light (replaces the original Lego Light Brick).
- Dashboard numeric input to configure the flash interval.
- Dashboard ui_switch to trigger flashing.
- Flashing handled by Javascript functions setInterval and clearInterval.

Prototype RGB LED in the toplight and OLED display at level 2

Flow



Dashboard



Node-RED Flow Lighthouse – Information Display

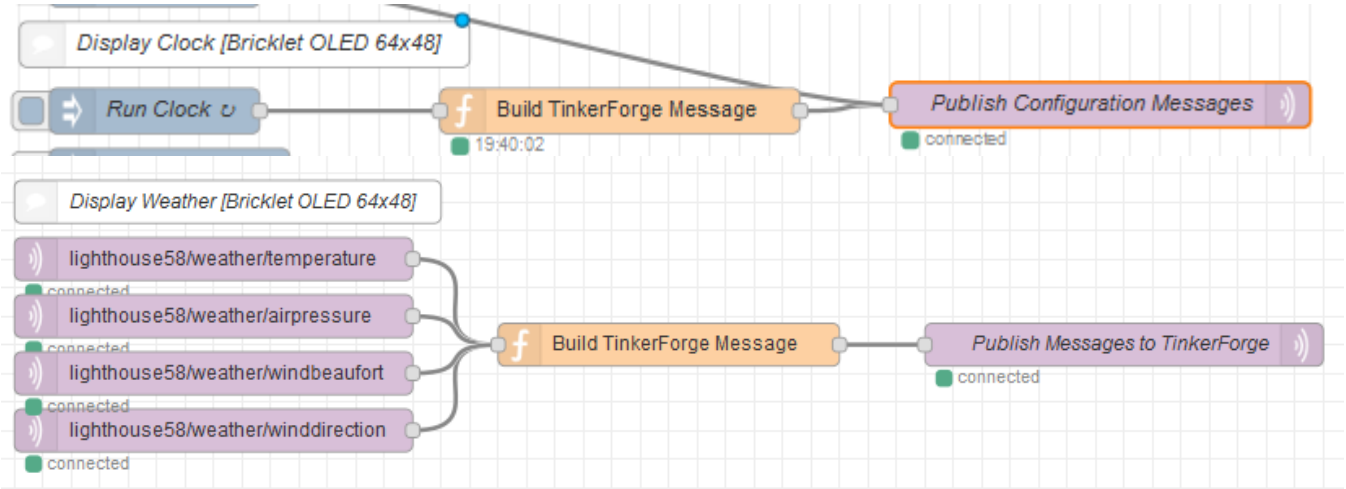
Functionality

- Clock hh:mm:ss.
- Weather information temperature, airpressure, beaufort, wind direction.

Solution

- TinkerForge OLED 64x48 bricklet integrated in the 2nd level of the lighthouse (6 lines with max 13 characters per line).
- Weather Underground API to request weather information for the lighthouse lat & lon position.

Flows to display the clock at OLED display line 0 and weather information at lines 2 - 5



Prototype
OLED display at level 2

Node-RED Flow Lighthouse – Illuminance

Functionality

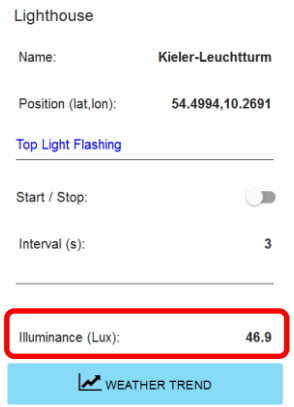
- Measure & display the Illuminance.
- Switch on the Pier Outdoor Light if Illuminance < Darkness Level

Solution

- TinkerForge Ambient Light bricklet integrated in Control Unit.
- Measure the Illuminance once per minute
- Sent the Illuminance to the Dashboard.
- Switch Pier Outdoor Light depending settings.

Flow

Dashboard



Node-RED Flow Pier

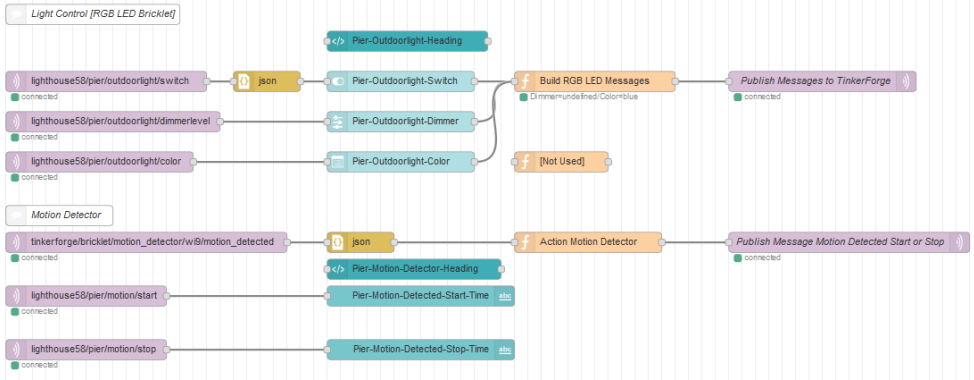
Functionality

- Control the pier outdoor light: switch on / off, set dim level, set light color red, green or blue.
- Detect motion, e.g. if a ship enters the pier take action: notify via email and set pier outdoor light on with red color [PLANNED].

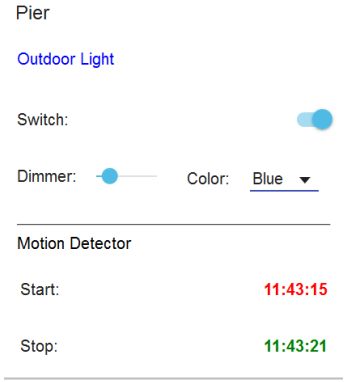
Solution

- TinkerForge RGB LED bricklet integrated in the pier.
- TinkerForge Motion Detector bricklet integrated in the pier.

Flow



Dashboard



Node-RED Flow Lightkeeper Cottage

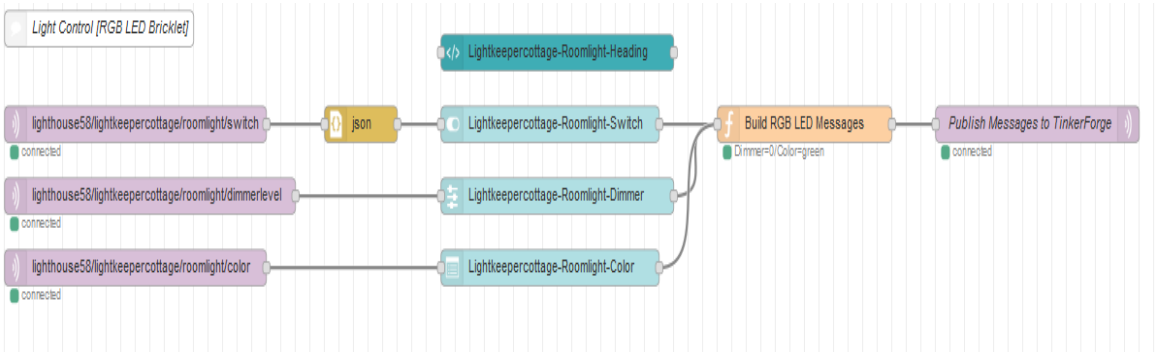
Functionality

- Control the indoor light: switch on / off, set dim level, set light color red, green or blue.

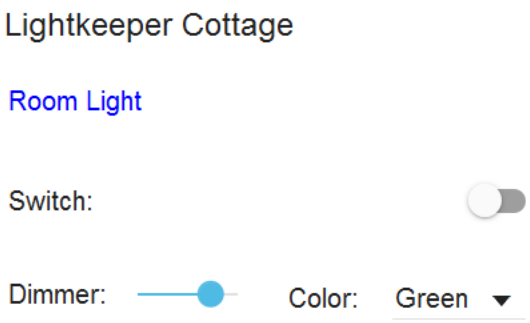
Solution

- TinkerForge RGB LED bricklet integrated in the cottage.

Flow



Dashboard



Node-RED Flow Boathouse

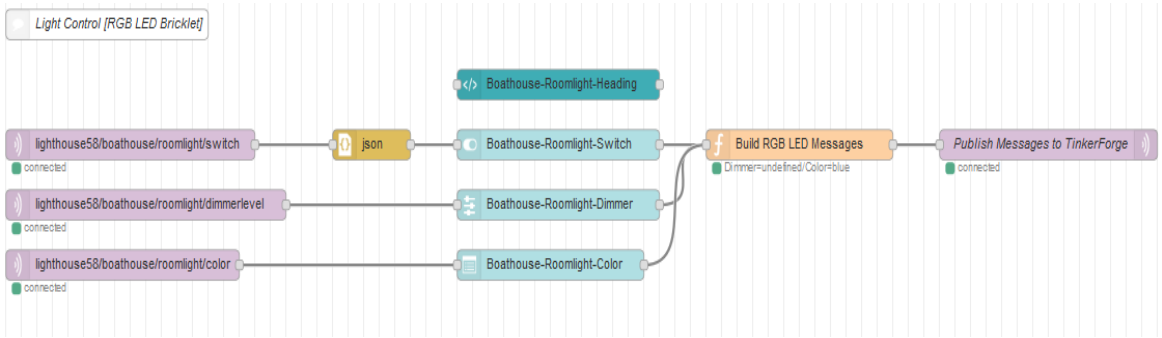
Functionality

- Control the indoor Light: switch on / off, set dim level, set light color red, green or blue.

Solution

- TinkerForge RGB LED bricklet integrated in the boathouse.

Flow



Dashboard

Node-RED Flow Weather

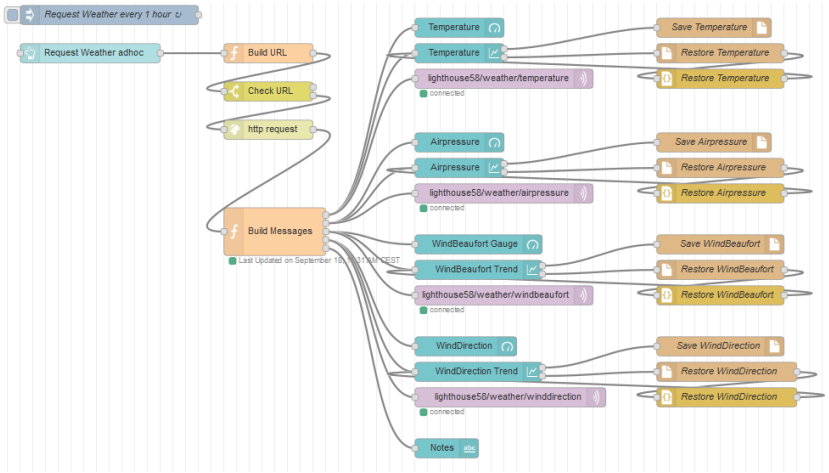
Functionality

On the lighthouse information display & the weather dashboard show, at regular interval, weather information - temperature (C), airpressure (mBar), beaufort (Bft), wind direction (deg) - from the lat / lon position.

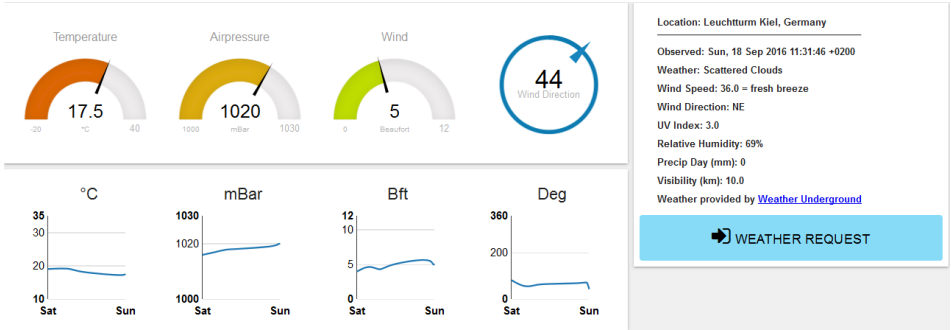
Solution

- Get Data: request from Weatherunderground every hour weather information. This requires to be signed up to obtain an API key. This key is used to place weather information requests using the lat / lon position.

Flow



Dashboard

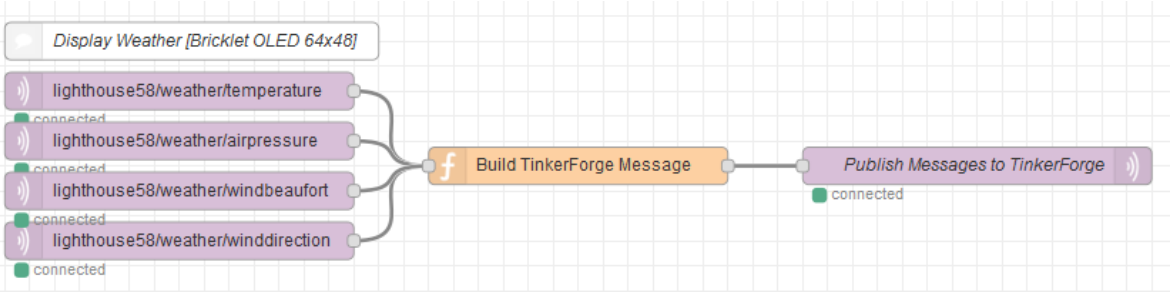


Node-RED Flow Weather - Messages

MQTT Weather Topics published

- lighthouse58/weather/temperature
- lighthouse58/weather/airpressure
- lighthouse58/weather/windbeaufort
- lighthouse58/weather/winddirection

The payload is used to display on the lighthouse information display (described here).



The Function Node "Build TinkerForge Message" builds & publishes a line for each of the weather topics displayed on the OLED 64x48 display lines:

1. Clock,
2. Empty Line
3. Temperature
4. Airpressure
5. Beaufort
6. Wind Direction

Node-RED Flow Control Unit - Shutdown

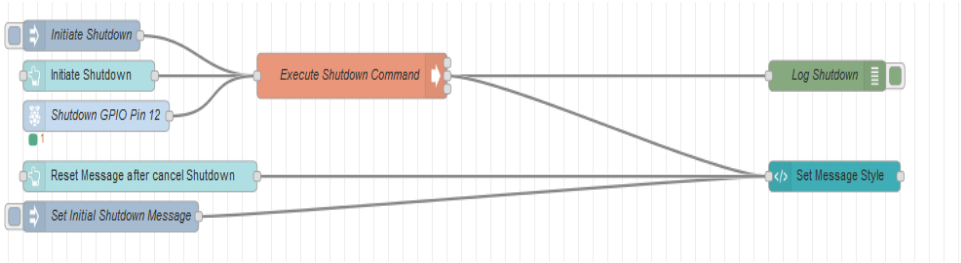
Functionality

- Shutdown the Control Unit Raspberry Pi with TickerForge Bricks.
- Initiated via Dashboard or Micro Switch placed at the side of the Unit.

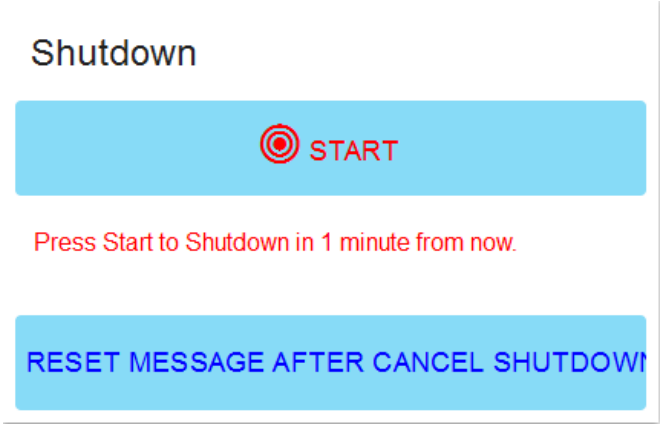
Solution

- Run Exec command “sudo shutdown -h” from “Exec Node”.

Flow



Dashboard



Node-RED Flow Control Unit - Status

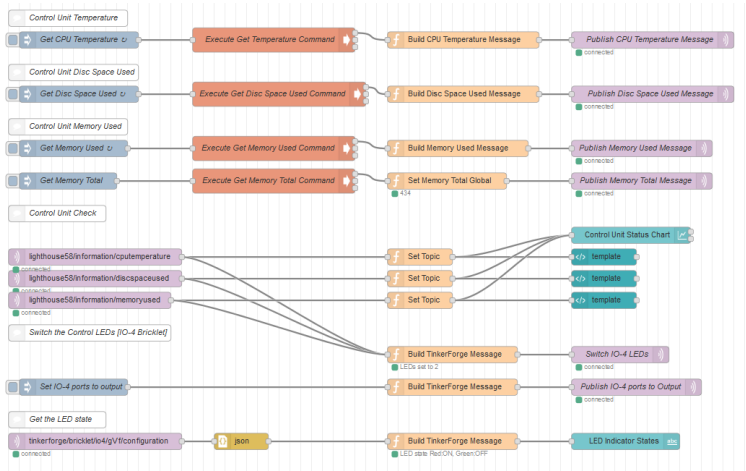
Functionality

- Show status with trend and alarm (Red LED) for CPU °C, Disc Space Used, Memory Used.

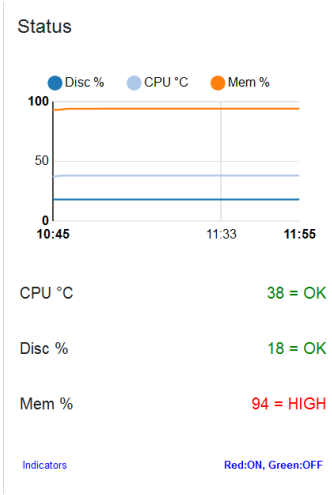
Solution

- Run Exec commands from “Exec Nodes” for CPU °C, Disc Space Used, Memory Used
  - cat /sys/class/thermal/thermal_zone0/temp
  - df -H | grep 'root' | awk '{ print $5 }‘
  - free -o -h | head -n2 | tail -n1 | awk '{print $3}' | tr -d [=M=]

Flow



Dashboard



Node-RED Flow Control Unit – Status Indicators

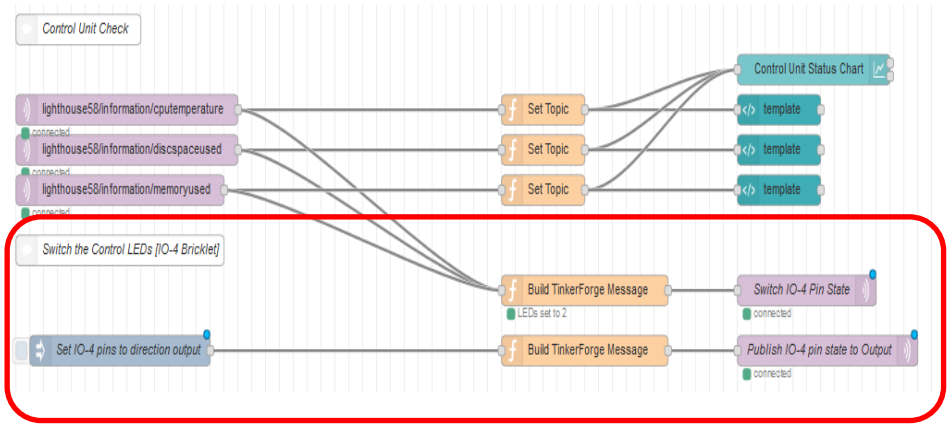
Functionality

- Show Control Unit status indicator LED Green or Red.

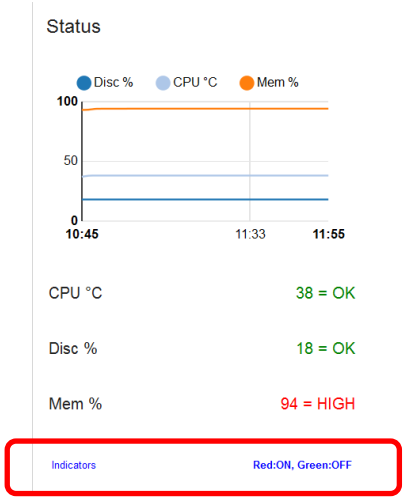
Solution

- LEDs Green & Red connected to a TinkerForge IO-4 Bricklet which is build into the Control Unit

Flow



Dashboard



Node-RED Flow Information

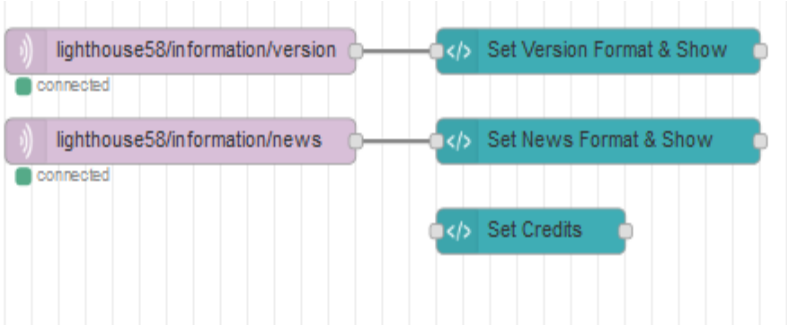
Functionality

- Show information News, About and Credits.

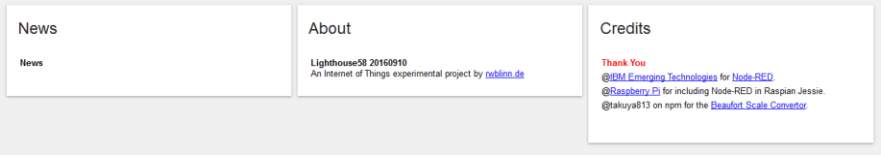
Solution

- Three Groups with ui_text widgets.

Flow

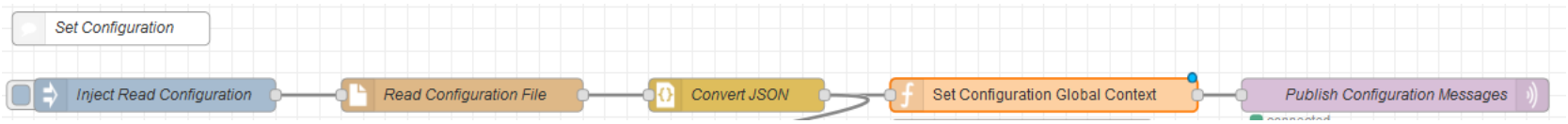


Dashboard



Node-RED Flow Configuration

Global context variables are used across all nodes. These are stored in a configuration file (/home/pi/lighthouse58/lighthouse58.json) in JSON format and read at the start of the configuration flow.



The function “Set Configuration Global Context” sets and publishes the global configuration object, lighthouse name and lighthouse58 version. Configuration format,content are subject to change while evolving the project.


```
{
"lighthouses" : [
   {"name":"Kieler-Leuchtturm", "position":"54.4994,10.2691", "interval":{"on":"3", "off":"3"}, "color":{"r":255, "g":255, "b":255} },
   {"name":"Neuwerk", "position":"53.92,8.48", "interval":{"on":"3", "off":"3"}, "color":{"r":255, "g":255, "b":255} },
   {"name":"Neuland", "position":"54.356974,10.604179", "interval":{"on":"5", "off":"5"}, "color":{"r":0, "g":255, "b":0} }
  ],
"lighthouseselected" : "Kieler-Leuchtturm",
"bricklets" : [
   {"name":"IO16-1", "uid":"gkM", "description":"LED Light Control" },
   {"name":"OLED-1", "uid":"x6y", "description":"Lighthouse Display" },
   {"name":"RGB-1", "uid":"zJW", "description":"Lighthouse Toplight" }
  ],
"wundergroundapikey" : “**************",
"version" : "20160903",
"copyright" : "by rwblinn.de"
}
```

Node-RED Flow Configuration - Global Context

The content of the configuration file is stored in the Global Context “config”:

```
global.set("config", msg.payload);
```

To access the members of the configuration object, use

```
global.get("config").member
```

Examples:

- “LiHo58 Version:" + global.get("config").version
- Publish MQTT Message lighthouse/name:
  var msgname = {topic : "lighthouse/name", payload : global.get("config").lighthouseselected };
  node.send(msgname);
- Log the Weather Underground API Key:
  node.log("Weather Underground API Key:" + global.get("config").wundergroundapikey);
- Build ui_dropdown list containing Lighthouses:

```
msg.options = LightHousesArray(obj.lighthouses);
LightHousesArray(arr) {
  var lharr = [];
  var i;
  for(i = 0; i < arr.length; i++) {
    lharr.push(arr[i].name);
   }
return lharr;
}
```


```
{
"lighthouses" : [
   {"name":"Kieler-Leuchtturm", "position":"54.4994,10.2691", "interval":{"on":"3", "off":"3"}, "color":{"r":255, "g":255, "b":255} },
   {"name":"Neuwerk", "position":"53.92,8.48", "interval":{"on":"3", "off":"3"}, "color":{"r":255, "g":255, "b":255} },
   {"name":"Neuland", "position":"54.356974,10.604179", "interval":{"on":"5", "off":"5"}, "color":{"r":0, "g":255, "b":0} }
 ],
"lighthouseselected" : "Kieler-Leuchtturm",
"bricklets" : [
   {"name":"IO16-1", "uid":"gkM", "description":"LED Light Control" },
   {"name":"OLED-1", "uid":"x6y", "description":"Lighthouse Display" },
   {"name":"RGB-1", "uid":"zJW", "description":"Lighthouse Toplight" }
 ],
"wundergroundapikey" : “**********”,
"version" : "20160903",
"copyright" : "by rwblinn.de"
}
```

Node-RED Flow Configuration - Lighthouses

The lighthouses are defined in the configuration file in JSON format.
An array holds the list of lighthouses with properties:

Name	Position	Interval	Color	Info
Kieler-Leuchtturm	54.4994,10.2691	"on":"3", "off":"3"	"r":255, "g":255, "b":255	Info
Neuwerk	53.92,8.48	"on":"3", "off":"3"	"r":255, "g":255, "b":255	Info
Neuland	54.356974,10.604179	"on":“5", "off":“5"	"r":0, "g":255, "b":0	Info
Reference	Lat, Lon. To convert a Lighthouse address to Lat, Lon: Address to Lat & Lon	The inervals are placeholders for now. TODO: Set vale according lighthouse definition	The colors are placeholders for now. TODO: Set vale according lighthouse definition	

Node-RED Flow Configuration – Lighthouse Selection

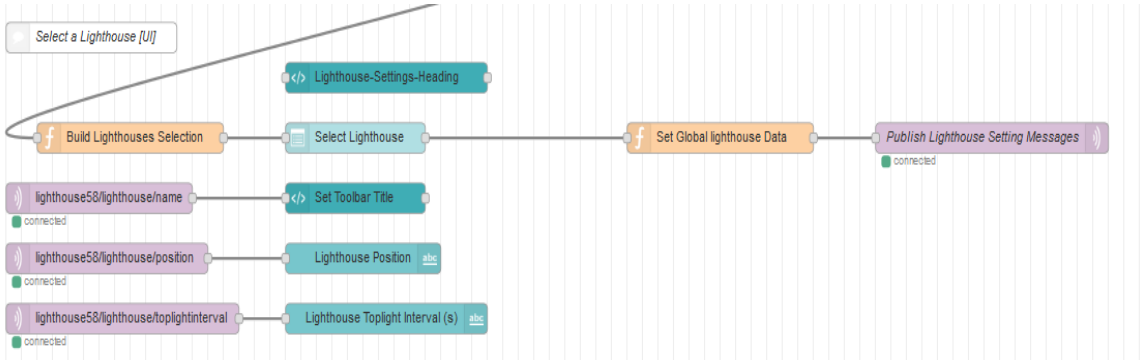
Functionality

Select a Lighthouse with properties.

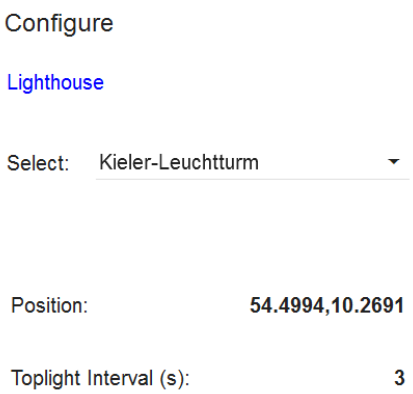
Solution

A select box is populated from the Configuration file.
When selecting, the properties are set.

Flow



Dashboard



Node-RED Flow Configuration – Outdoor Light Darkness

Functionality

Set the option of turning on the Pier Outdoor Light when Illuminance below certain Darkness Level.

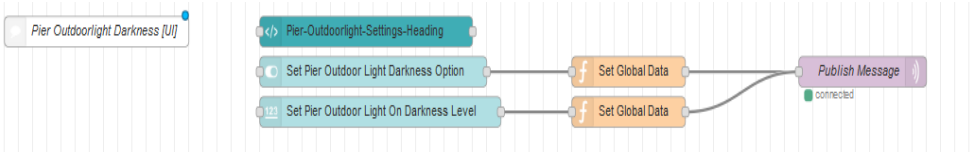
Solution

Switch to set the option Light On Darkness.

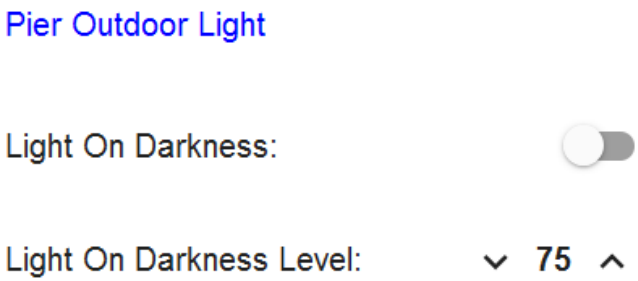
Numeric input field with range 0 – 200 to set the Darkness Level.

When Illuminance < Darkness Level and Switch is On, the Pier Outdoor Light is turned on (Note that the Dimmer Level must > 0).

Flow



Dashboard



Node-RED MQTT Messages Published /1

MQTT messages published, which can be subscribed by any MQTT client.

Description	Topic
Lighthouse	• lighthouse58/lighthouse/name • lighthouse58/lighthouse/position • lighthouse58/lighthouse/toplightinterval • lighthouse58/lighthouse/toplightstatus
Pier	• lighthouse58/pier/outdoorlight/switch/set • lighthouse58/pier/outdoorlight/switch • lighthouse58/pier/outdoorlight/darknessswitch • lighthouse58/pier/outdoorlight/darknessswitchlevel • lighthouse58/pier/outdoorlight/dimmerlevel/set • lighthouse58/pier/outdoorlight/dimmerlevel • lighthouse58/pier/outdoorlight/color/set • lighthouse58/pier/outdoorlight/color • lighthouse58/pier/motion/start • lighthouse58/pier/motion/stop
Lightkeeper Cottage	• lighthouse58/lightkeepercottage/roomlight/switch/set • lighthouse58/lightkeepercottage/roomlight/switch • lighthouse58/lightkeepercottage/roomlight/dimmerlevel/set • lighthouse58/lightkeepercottage/roomlight/dimmerlevel • lighthouse58/lightkeepercottage/roomlight/color/set • lighthouse58/lightkeepercottage/roomlight/color

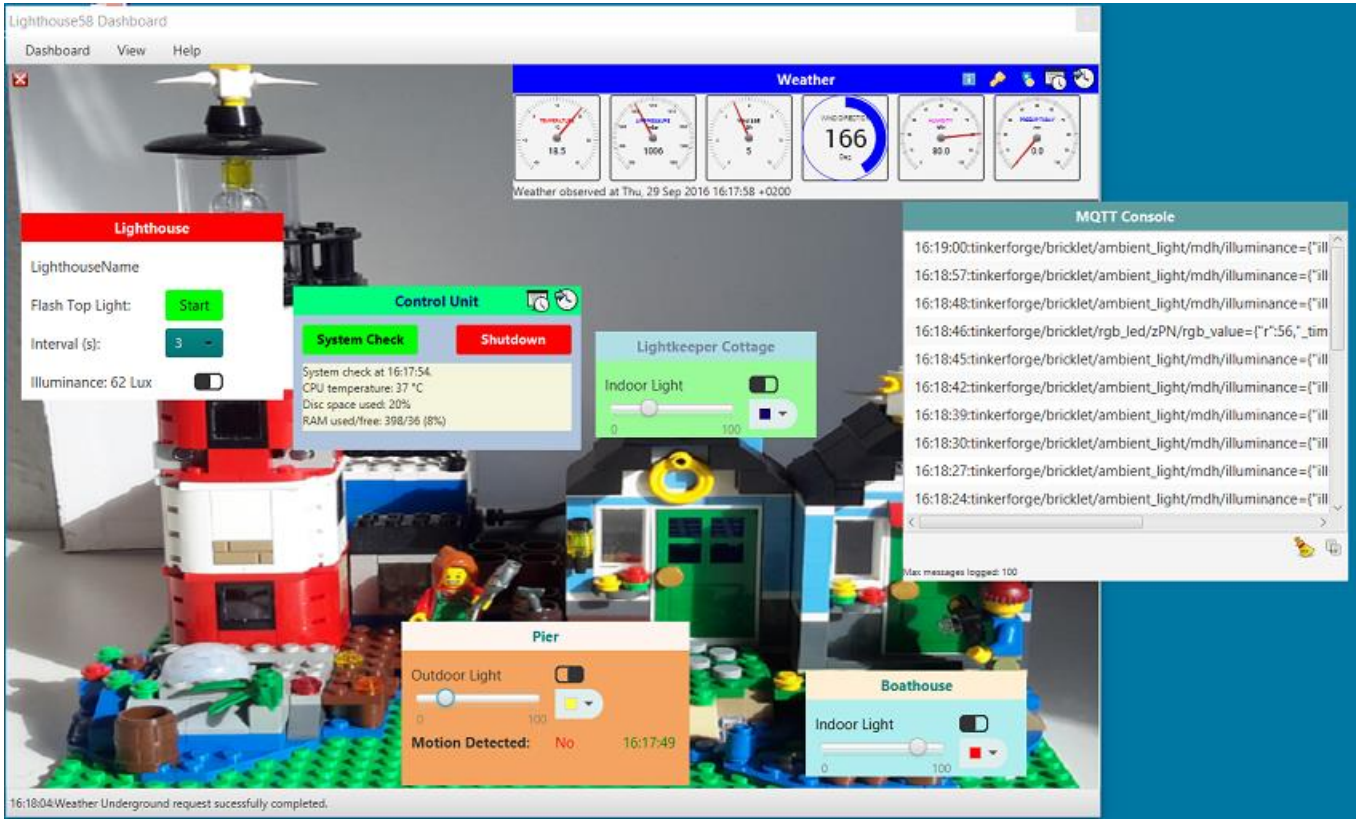
Node-RED MQTT Messages Published /2

Description	Topic
Boathouse	• lighthouse58/boathouse/roomlight/switch/set • lighthouse58/boathouse/roomlight/switch • lighthouse58/boathouse/roomlight/dimmerlevel/set • lighthouse58/boathouse/roomlight/dimmerlevel • lighthouse58/boathouse/roomlight/color/set • lighthouse58/boathouse/roomlight/color
Weather	• lighthouse58/weather/temperature • lighthouse58/weather/airpressure • lighthouse58/weather/windbeaufort • lighthouse58/weather/winddirection
Control Unit	• lighthouse58/information/cputemperature • lighthouse58/information/discspaceused • lighthouse58/information/memoryused • lighthouse58/information/memorytotal
Configuration	• lighthouse58/lighthouse/config
Information	• lighthouse58/information/version • lighthouse58/information/news

Lighthouse58 Windows Dashboard solution.

B4J

B4J – Windows Dashboard



Lighthouse58 Dashboard application running on Windows 10

B4J - Introduction

What

B4J is a 100% free development tool for desktop, server and IoT solutions created by Anywhere Software.

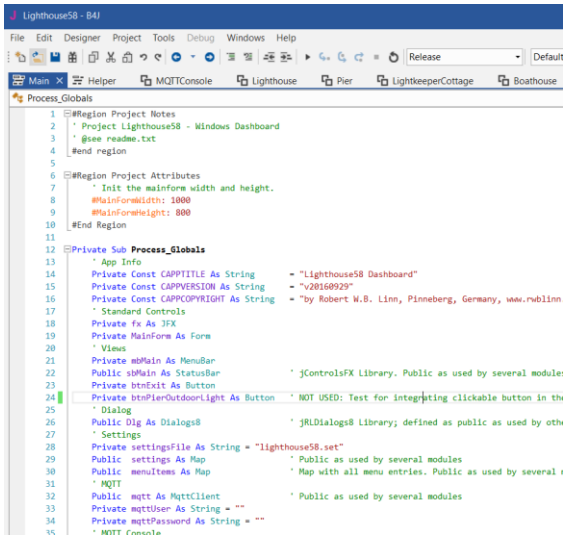
How

The B4J solution messaging is based on MQTT using TinkerForge messages only. Each object has its own Class module. The run the solution, Java 8+ must be installed on the Windows client.

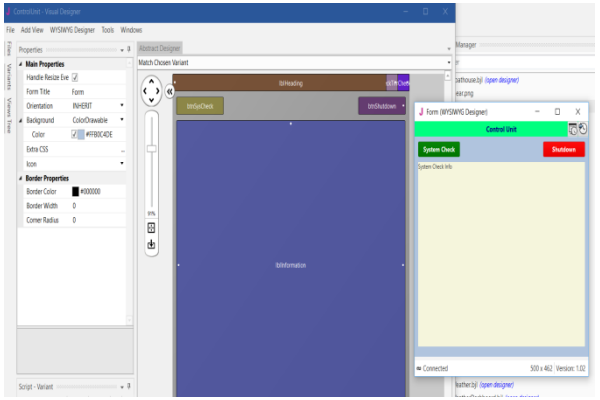
Notes

- The intention of this show case is not to explain B4J. Recommend to visit the B4J homepage to learn more.
- The author is a B4J enthusiast and ever learning this great tool.

B4J IDE snippet



B4J Visual Designer snippet



B4J – Messaging Concept

Messaging based on MQTT using TinkerForge messages. No linkage with Node-RED, means the B4J solution runs without Node-RED.

Windows

Dashboard

B4J MQTT Client

Subscribes & Publishes TinkerForge MQTT Messages

Raspberry Pi

Mosquitto / paho-mqtt

TinkerForge Brick MQTT Proxy

Tinkerforge TCP/IP Protocol (TFP) and MQTT translator

TinkerForge Brick Deamon

TinkerForge Master Bricks

TinkerForge Bricklets

B4J – Modules Concept

Each Object, like the Lighthouse, Pier, Lighthouse Cottage, Boathouse, Control Unit, MQTT Console:

- is defined as a single Class Module (e.g. pier.b4a)
- has a movable non-modal form which layout is created with the B4J Visual Designer (e.g. pier.bjl)
- the titlebar contains clickable icon buttons (e.g. Timer.png)
- Settings are maintained within the Class but manained by the Main Class
- In addition
    - A helper class is used for common tasks.
    - The TFControl class manages MQTT communication.

The B4J source code can be downloaded here.

Concepts

Node-RED Messaging Concept

- **Messaging** is based upon MQTT.
- **Topic: Object/Device/Action**
  An object has one or more devices. A device has one or more actions.
- **Payload: Set the action value**
  The value is depending on the device.
- E.g.
  Object: lighthouse, pier
  Device: outdoorlight, toplight
  Action: on, off, state, blink(n), flashinterval, illuminance

Node-RED Messaging Concept - Examples

- **Switch boathouse(=object) roomlight (=device) on (=action)**
  (action values are on or off)
  Topic: lighthouse58/boathouse/roomlight/switch/set
  Payload: on
- **Dim the boathouse roomlight**
  (action value between 0 – 255 which is assigned to one color)
  Topic: lighthouse58/boathouse/roomlight/dimmerlevel/set
  Payload: 99
  Note: To obtain the dimmerlevel, use the topic:
  lighthouse58/boathouse/roomlight/dimmerlevel
- **Set lighthouse toplight flash interval in secs**
  (action value between 0 – 60)
  Topic: lighthouse58/lighthouse/toplightinterval
  Payload: 10
- Note: This is done from the configuration