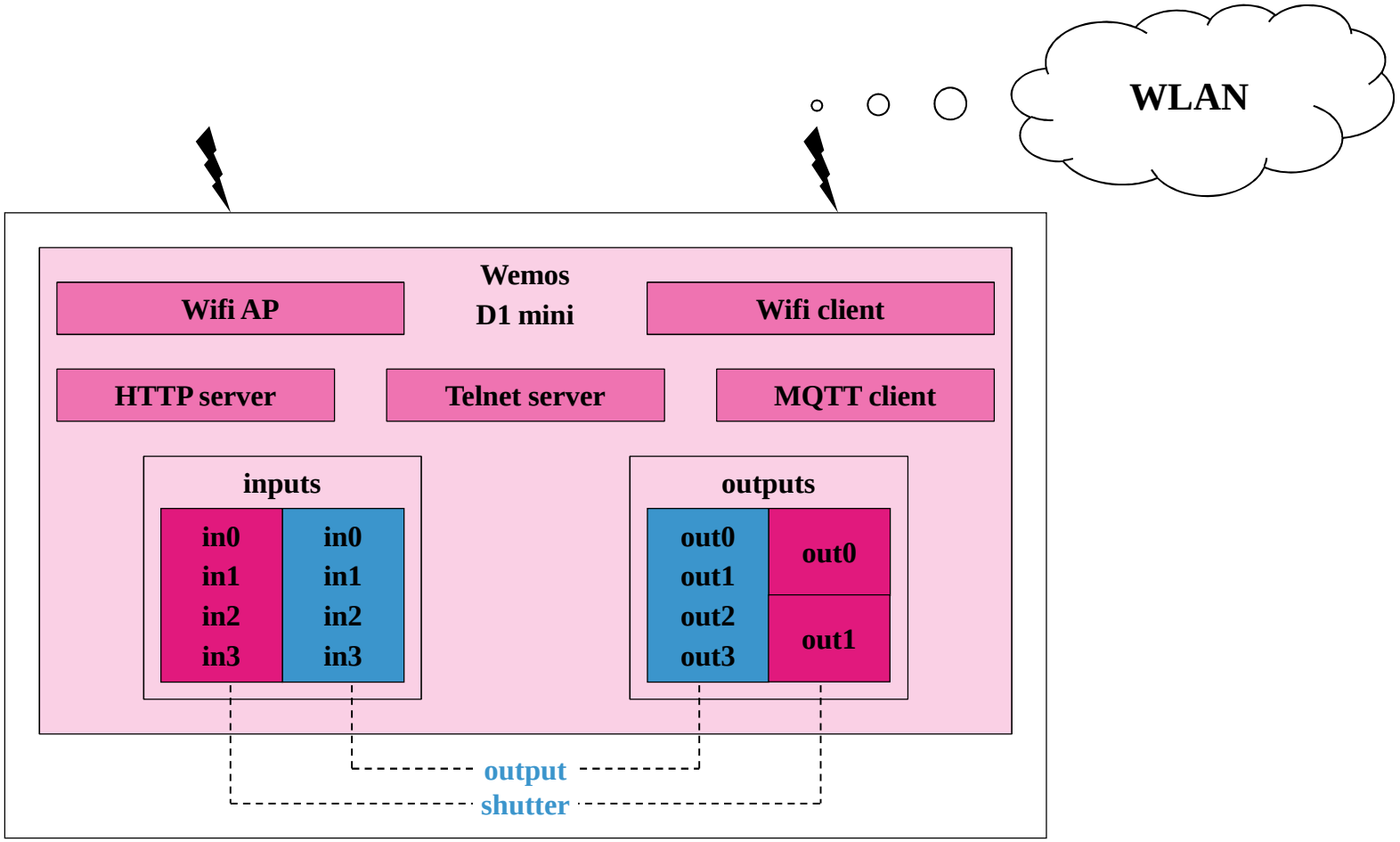
wiControl

Jo Simons (simonjo@telenet.be)

Project Target

- Build simple module to control lights and shutters
- Must be small enough to retrofit an existing 220V electrical system (unterputz)
- Able to work standalone
- Or work centralized but then in a wireless fashion
- Upgradable over-the-air
- Remote diagnostic and test functionality
- Secure when connected wireless
- Scalable software framework for different configurations
- Low cost

Module Structure (1/5)



Module Structure (2/5)

- Makes use of ESP8266 module on Wemos D1 mini subassembly
  - Easy due to built-in USB interface and 5V-to-3V3 convertor
  - Doesn't require implicit boardspace due to it's leveraged mounting
  - Has Wifi client to connect to remote access point
    - Not required if you want to operate the module in standalone mode
    - Else configure SSID + security method + secret
  - Has Wifi AP accessible @ 192.168.4.1
    - Can be secured with password and/or turned off
    - If Wifi client is disabled or fails, the Wifi AP will be turned on at module boot
  - Configuration/setup can be done via HTTP
  - 9 GPIO's used

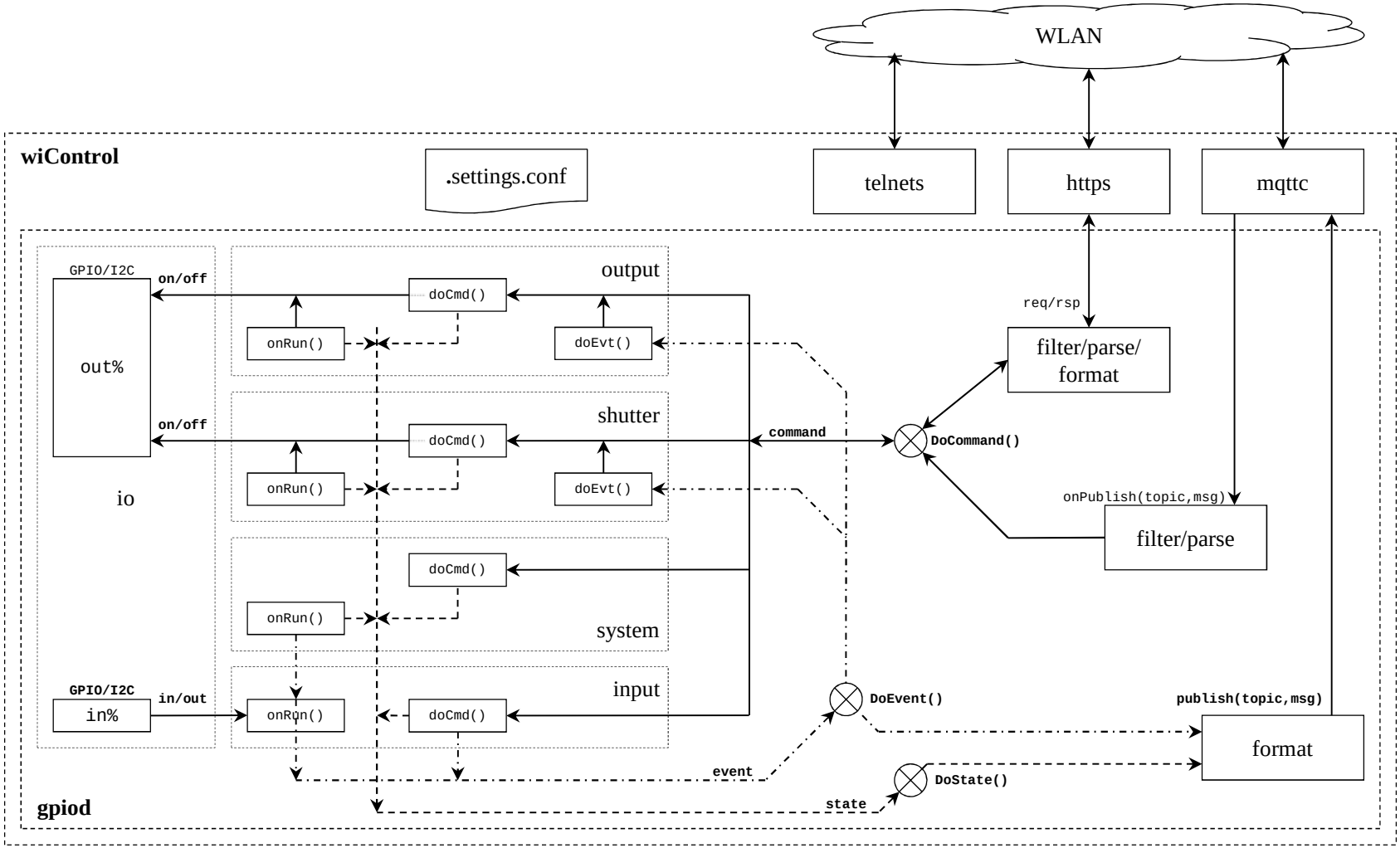
- Controllable objects included are:
  - 4 low-voltage inputs supporting normal-open push-buttons that connect to ground when pressed
  - 4 high-voltage (220V) SSR outputs upto 1A

- System LED
  - Helps to locate a module

- 3D printed protective cover
  - Against accidental touching 220V contacts
  - For mounting in installation box
  - For attaching Niko blind frontplate

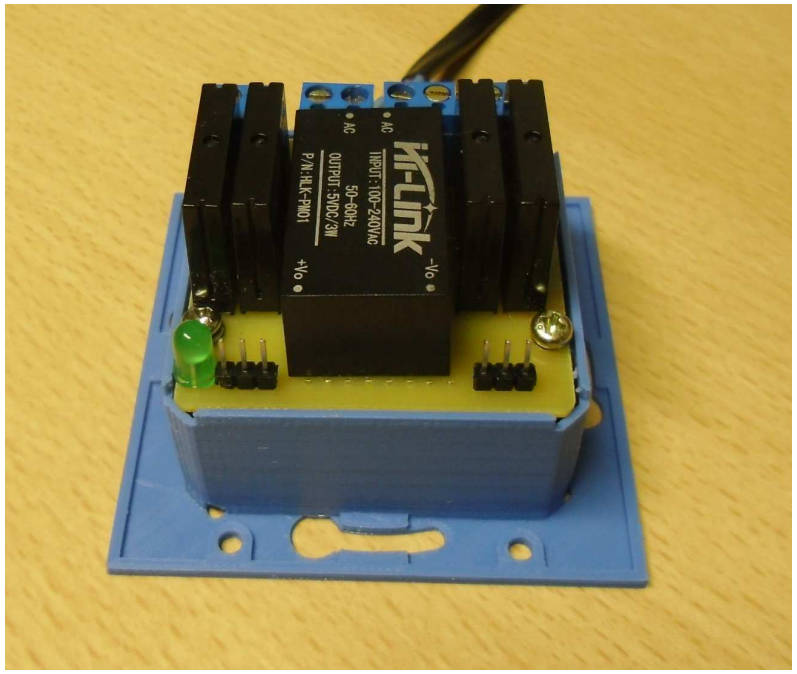
Module Structure (3/5)

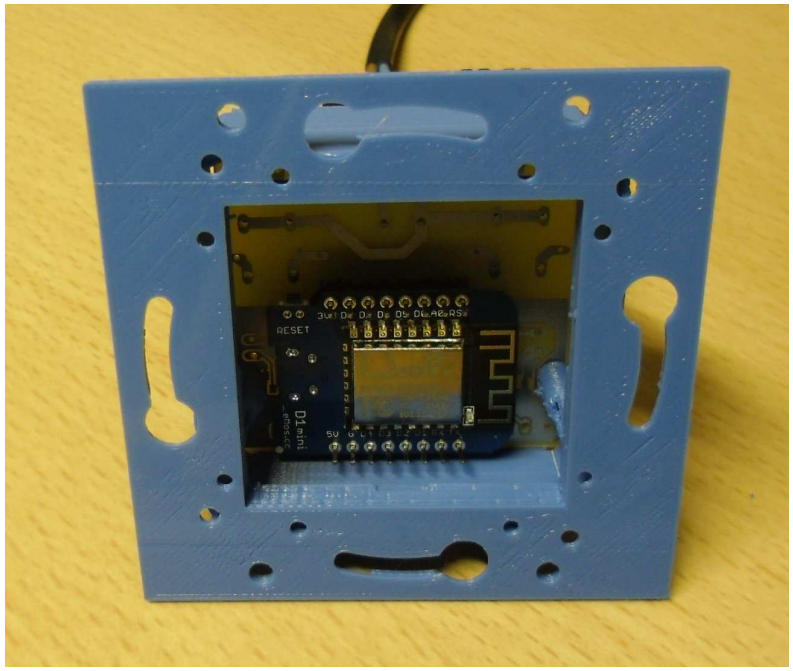
- Functional Block Diagram



Module Structure (4/5)

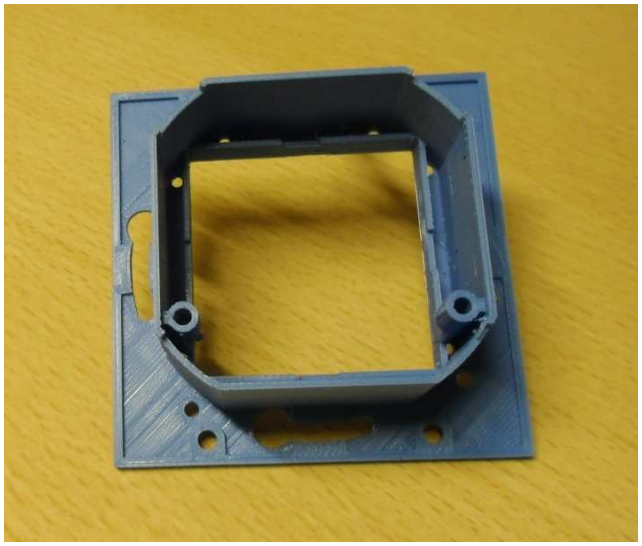
- PCB edition 01

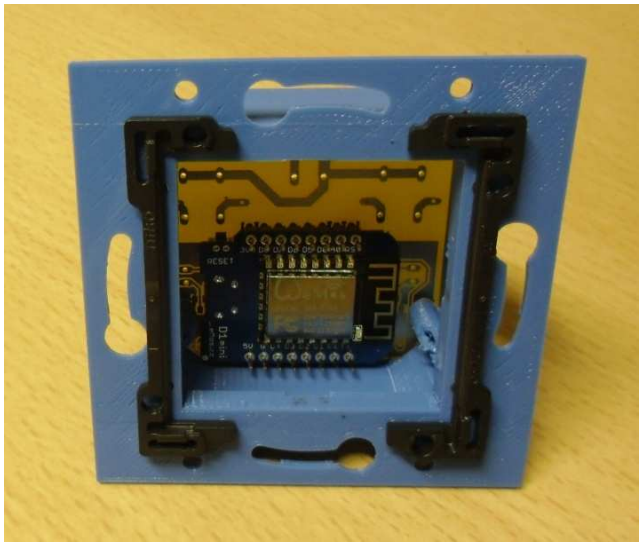


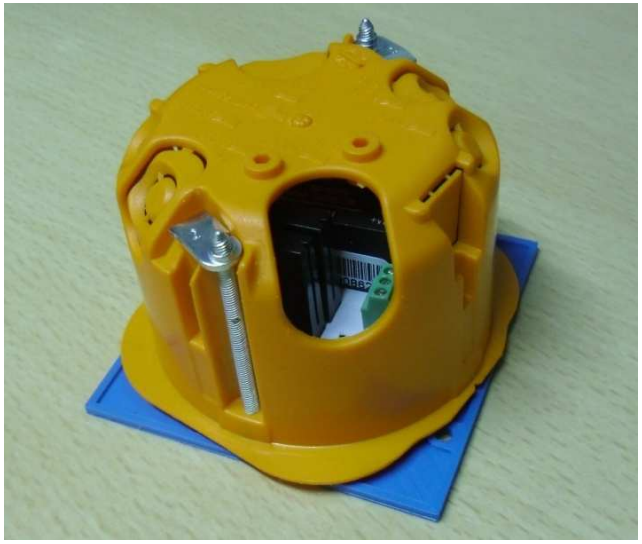


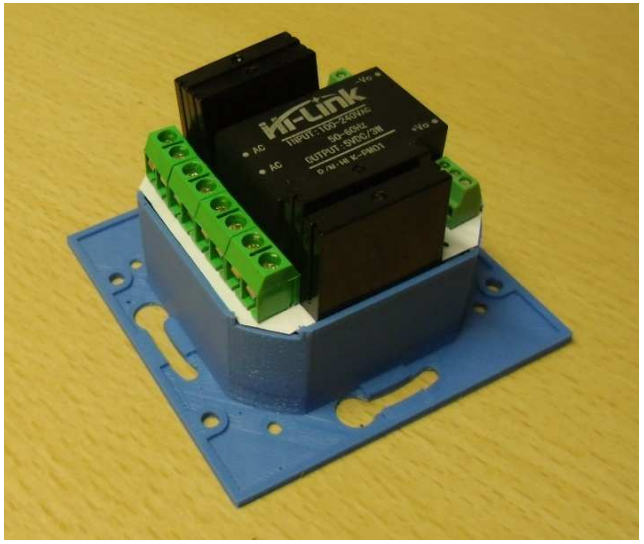
Module Structure (5/5)

- PCB edition 02









Features (1/3)

- Supported emulations:
  - Output
    - drives 4 individual outputs
    - dedicated set of commands: off, on, toggle, ontimed, …
  - Shutter
    - combines 2x2 outputs to drive 2 shutter/jalousie
    - dedicated set of commands: stop, up, down, …

- Supported modes:
  - Local
    - inputs control outputs locally
    - requires no extra infrastructure to operate
    - internal logic links input events to output commands
  - Remote (via MQTT)
    - connects to MQTT broker, so extra infrastructure is required
    - allows remote operation via MQTT protocol
  - Both
    - combines standalone and MQTT modes

- Safe mode:
  - If MQTT mode enabled and connectivity drops >1min, goto predefined safe state on the outputs
  - To be defined

Features (2/3)

- MQTT Client
  - Not possible if there is no Wifi connection
  - Not required if you want to operate the module in standalone mode
  - Else configure broker IP address/port, optional username/password, optional <node-id> override
    - default <node-id> is module chipid, i.e. esp12345
    - structured <node-id> can be <location>/<floor>/<number>
  - Requires an MQTT broker to be operational (i.e. Mosquitto on Synology NAS)
  - Uses a request only model
  - Is low on memory overhead for the system (in the order of bytes)
  - Ideal for automatic testing
  - Notational convention of MQTT request <topic> [=<msg>]

  - Once connected to broker
    - the module will subscribe to command topics <node-id>/cmd/#
    - the module will publish it's presence <node-id>/evt/boot=<sw-version>;<hw-topology>

  - In general
    - incoming command messages will be executed <node-id>/cmd/<object>=<cmd> *[.<parm>]
    - <node-id>/cmd/<cmd>= [<parm> *[.<parm>]]
    - object events are published to broker <node-id>/evt/<object>=<event>
    - <node-id>/evt/<event>=<detail>
    - object status is published to broker <node-id>/sta/<object>=<state>

Features (3/3)

- HTTP Server
  - Not possible if there is no Wifi connection
  - Uses a request-response model
  - Has more overhead on memory for the system, can drain the system (in the order of hundreds of bytes)
  - Main task is to configure the module with a browser
  - Also allows to send commands to the controllable objects using structured URL
    - URL: http://<ip-address>/ats?ccmd=[<object>.]<cmd> *[.<parm>] [; <object>.<cmd> *[.<parm>]]
    - command results are sent in HTTP response in terse mode
    - also used for automatic testing
- Telnet Server
  - Not possible if there is no Wifi connection
  - Mainly used for remote diagnostic/support

Terminology

- The module contains controllable objects
  - in0..in3
  - out0..out3 (or out0..out1 for shutter emulation)
  - system objects (loglevel, emul, mode, lock, disable)

- Objects accept commands
  - That instructs it to do something and update it's state
  - MQTT <node-id>/cmd/<object>=<cmd> *[.<parm>] *[; <cmd> *[.<parm>]]
  - HTTP http://<ip-address>/ats?ccmd=<object>.<cmd> *[.<parm>] *[; <object>.<cmd> *[.<parm>]]
    - returns current state in numerical terse format, i.e. '0', '1', '2'
    - response of multiple commands will be concatinated, i.e. '0;1'

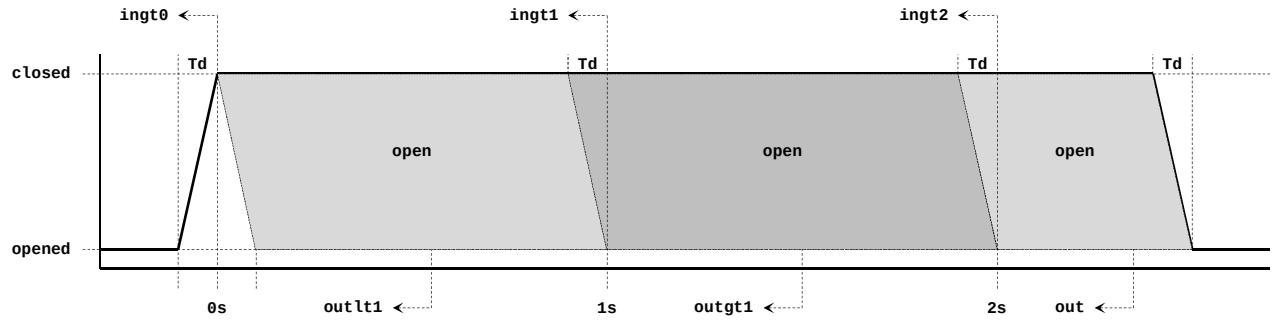
- An input object reports an event when detecting an external change
  - It will raise an event which will be distributed depending on configuration and emulation
  - To MQTT <node-id>/evt/<object>=<event>
  - Not to HTTP !!!

- A command will report an objects state when changed, but always for a status command
  - To MQTT <node-id>/sta/<object>=<state>
  - To HTTP http://<ip-address>/ats?ccmd=<object>.status
    - returns current state in numerical terse format, i.e. '0', '1', '2'
    - response of multiple commands will be concatenated, i.e. '0;1'

Inputs (1/2)

- Input objects: in0..in3

- Input changes are validated with an adjustable decounce period (Td), default=100ms
- Validated changes are fed into a time based state machine to generate events



- Supported MQTT event messages
  - <node-id>/evt/in%=2 input was initially closed (ingt0)
  - <node-id>/evt/in%=3 input was opened after less than 1 second (outlt1)
  - <node-id>/evt/in%=4 input was closed for more than 1 second (ingt1)
  - <node-id>/evt/in%=5 input was opened after more than 1 but less than 2 seconds (outgt1)
  - <node-id>/evt/in%=6 input was closed for more than 2 seconds (ingt2)
  - <node-id>/evt/in%=7 input was opened after more than 2 seconds (out)

Inputs (2/2)

- Supported commands (HTTP and MQTT)
  - [status] — get current state of input
  - ingt0 — simulate input closed initially
  - outlt1 — simulate input opened after less than 1 second
  - ingt1 — simulate input closed for more than 1 second
  - outgt1 — simulate input opened after more than 1 but less than 2 seconds
  - ingt2 — simulate input closed for more than 2 seconds
  - out — simulate input opened after more than 2 seconds
  - debounce [.<time>] — get or set debounce time (Td) in ms, default=100
- Parameters
  - time — 50-3600000 ms
- Supported MQTT state messages
  - <node-id>/sta/in%<state> — current input state, 0=out,1=ingt0,2=ingt1,3=ingt2
  - <node-id>/sta/in%/debounce=<time> — current debounce time in ms

Outputs – Output Emulation (1/2)

- Output objects: out0..out3

- Uses a protective lock
  - When <lock> = 1, the output object will be locked during the command, no other command except unlock can be issued
- Uses combination of <delay> and <run> timers
  - When <delay> = 0, no delay will be inserted before executing a command, otherwise <delay> seconds are waited
  - When <run> = 0, the command will last forever, otherwise only for <run> seconds

- Supported commands (HTTP and MQTT)
  - [status] get current output state, 0=off, 1=on
  - off.<lock>.<delay>.<run> turn output off
  - on.<lock>.<delay>.<run> turn output on
  - toggle.<lock>.<delay>.<run> toggle output between off and on
  - blink.<lock>.<run> alternate output between on and off with a 1 second cadence, all outputs are synched
  - lock [.<lock>] get or set lock state, when locked only unlock can be executed
  - defrun [.<def-run>] get or set default run time for local standalone operation, default=0

- Examples
  - on.0.0.0 turn output on forever, not locked
  - toggle.0.5.0 toggle output after 5 seconds, not locked
  - toggle.0.5.10 toggle output after 5 seconds, and again after another 10 seconds, not locked
  - blink.1.0 blink output forever, locked

Outputs – Output Emulation (2/2)

- Parameters
  - lock — 0 (unlock) – 1 (lock)
  - delay — 0-3600 seconds, 0=no delay
  - run — 0-3600 seconds, 0=infinite
  - def-run — 0-3600 seconds, 0=infinite

- Supported MQTT state messages
  - <node-id>/sta/out%=<state> — current output state, 0=off, 1=on
  - <node-id>/sta/out%/lock=lock> — current lock state, 0=unlocked, 1=locked
  - <node-id>/sta/out%/defrun=<def-run> — current default run time in seconds for local standalone operation

Outputs – Shutter Emulation (1/2)

- Shutter objects: out0..out1

- Uses the concept of priority level (0=highest, 5=lowest) and lock (0=no-lock, 1=lock)
  - A command of a given level is rejected when that level is locked
  - When a command of a given level sets the lock, any lower level commands are refused
  - Useful for protecting extending sunblinds against strong winds or rain with a sensor locking the blinds in prio 0 or 1
  - Mask versus level: a priority mask has a bit per level, level 0=0x01, 1=0x02, 2=0x04, 3=0x08, 4=0x10, 5=0x20

- Implied slow start to protect outputs and shutter motors

- Supported commands (HTTP and MQTT)

– [status]	get current status of shutter, 0=stop|1=up|2=down
– stop.<level>	stop shutter
– up.<level>.<lock>.<delay>.<run>.<tip0>	move shutter up for given run time, optional delay and tip
– down.<level>.<lock>.<delay>.<run>.<tip0>	move shutter down for given run time, optional delay and tip
– toggleup.<level>.<lock>.<run>	toggle between stop and up for given run time
– toggledown.<level>.<lock>.<run>	toggle between stop and down for given run time
– tipup.<level>.<lock>.<tip1>	short move up to tip the lamello's on a jalousie
– tipdown.<level>.<lock>.<tip1>	short move down to tip the lamello's on a jalousie
– priolock.<mask>	lock given levels, all non-locked levels remain operational
– priounlock.<mask>	unlock given levels
– prioset.<mask>	set given levels, only set level and higher levels remain operational
– prioreset.<mask>	reset given levels
– defrun [.<def-run>]	get or set default run time for standalone operation, default=30

Outputs – Shutter Emulation (2/2)

- Parameters
  - <level> 0 (highest) – 5 (lowest)
  - <lock> 0 (unlock) – 1 (lock)
  - <delay> 0-3600 seconds
  - <run>, <def-run> 1-3600 seconds
  - <tip0> 0-3600 1/10th seconds
  - <tip1> 1-3600 1/10th seconds
  - <mask> bit per level, 0=0x01, 1=0x02, 2=0x04, 3=0x08, 4=0x10, 5=0x20

- Supported MQTT state messages
  - <node-id>/sta/out%=<state> current state of shutter, 0=stopped,1=moving up,2=moving down
  - <node-id>/sta/out%/defrun=<def-run> current default run time for standalone operation

System

- Configuration command locking
  - When the lock is disabled, all configurations commands will work
  - When the lock is enabled, configuration cannot be changed (except lock), this to avoid accidental changes

- Operational command disable
  - When disabled, all non-system commands are blocked
  - When enabled, all commands can be used freely, limited by above lock mechanism

- Supported commands (HTTP or MQTT)

• ping	will blink the LED for about 10 seconds
• version	get current sw-version and hw-topology
• memory	get current memory statistics
• loglevel [=<level>]	get or set loglevel where level can be decimal or hexadecimal
• emul [=<1..2>]	get or set emulation, 1=output,2=shutter
• mode [=<1..3>]	get or set mode, 1=local,2=remote (via MQTT),3=both (*)
• lock [=<0..1>]	get/set configuration commands lock state
• disable [=<0..1>]	get/set operational commands disable state
• restart=ack	restart module
• save=ack	save (modified) configuration data to NVRAM

  - (*) lockout prevention

System

- Supported MQTT event messages
  - <node-id>/evt/pong=<ip-addr> — response to ping, includes module ip-address
  - <node-id>/evt/boot=<version>;<topo> — sw version and hw topology sent after boot, i.e. ‘4.0.0.0;4I4O/1’
  - <node-id>/evt/restart — sent when module executes restart command
  - <node-id>/evt/save — sent when configuration data was saved to NVRAM

- Supported MQTT state messages
  - <node-id>/sta/version=<version>;<topo> — sw version and hw topology, i.e. ‘4.0.0.0;4I4O/1’
  - <node-id>/sta/memory=<bytes> — current free memory in decimal format
  - <node-id>/sta/loglevel=<level> — current loglevel in decimal format
  - <node-id>/sta/emul=<1..2> — current emulation
  - <node-id>/sta/mode=<1..3> — current mode
  - <node-id>/sta/lock=<0..1> — current lock state
  - <node-id>/sta/disable=<0..1> — current disable state

Standalone Mode

- Allows module to work without further infrastructure by using input events to control outputs

- Emulation=output
  - in%.outlt1 -> out%.toggle
  - in%.ingt2 -> out%.blink.<def-run>

- Emulation=shutter
  - in0.ingt0/in1.ingt0 -> out0.stop.0
  - in0.ingt1 -> out0.up.0.0.0.<def-run>.0
  - in1.ingt1 -> out0.down.0.0.0.<def-run>.0
  - in2.ingt0/in3.ingt0 -> out1.stop.0
  - in2.ingt1 -> out1.up.0.0.0.<def-run>.0
  - in3.ingt1 -> out1.down.0.0.0.<def-run>.0

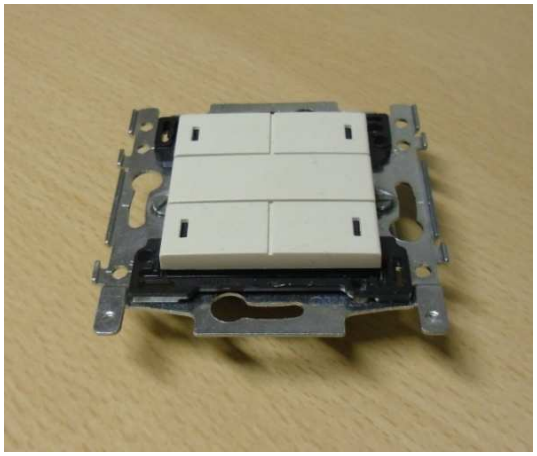
Outstanding

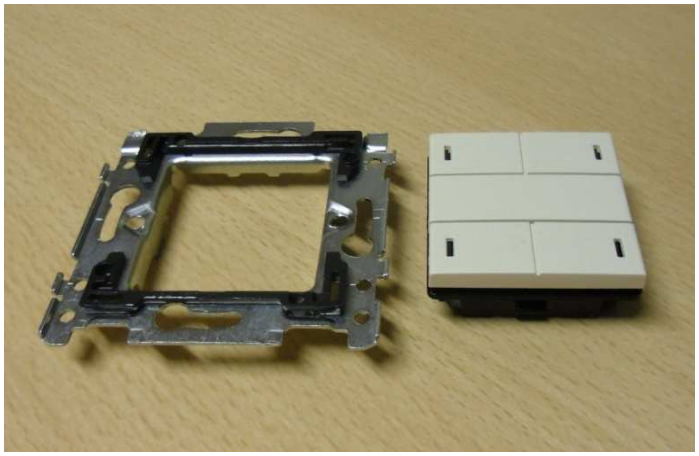
- Additional timer or clock objects
- Compact mode
  - reduce required pushbuttons
  - i.e. outlt1 -> out0.toggle, ingt2 -> out1.toggle
- Variants
  - 2 channel version with real relays and option for Wifi/RF interface
  - DIN rail module with 8-16 outputs…
- Telnet command interface
- Enhance shutter logic
  - introduce clocks
  - couple to internet dawn/sunset markers
- Check overall handling of Wifi / broker connection vs mode

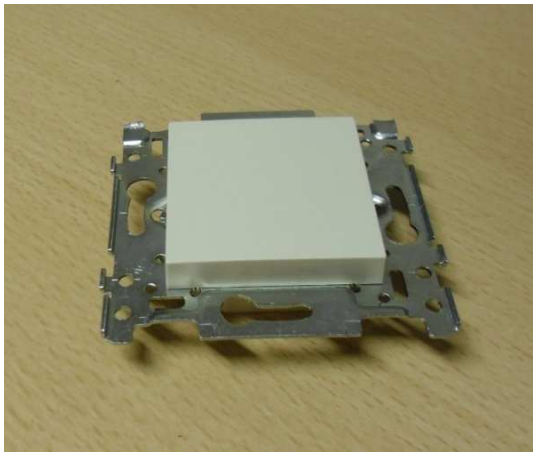
Backup slides

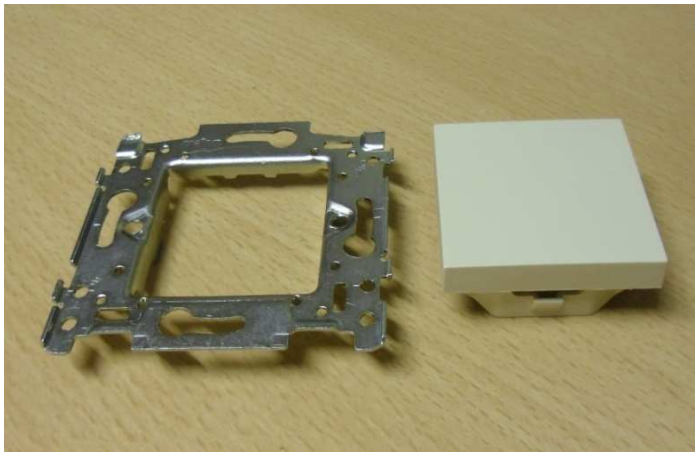
Jo Simons (icontrol@telenet.be)

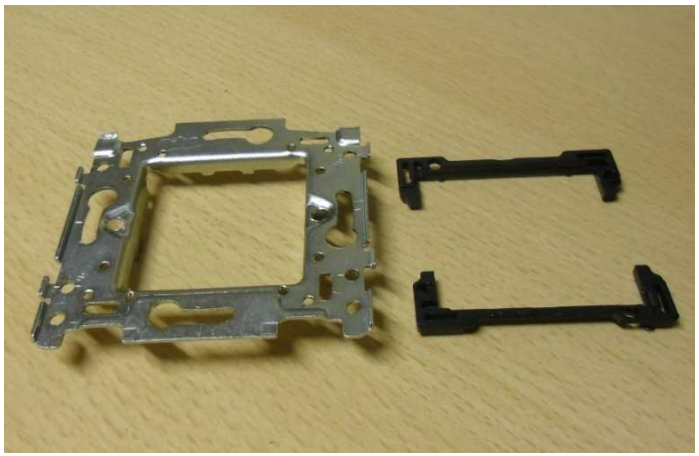
Niko Integration: Disassembly Blindplate/4-Fold Button











Niko Integration: Assembly Blindplate/4-Fold Button