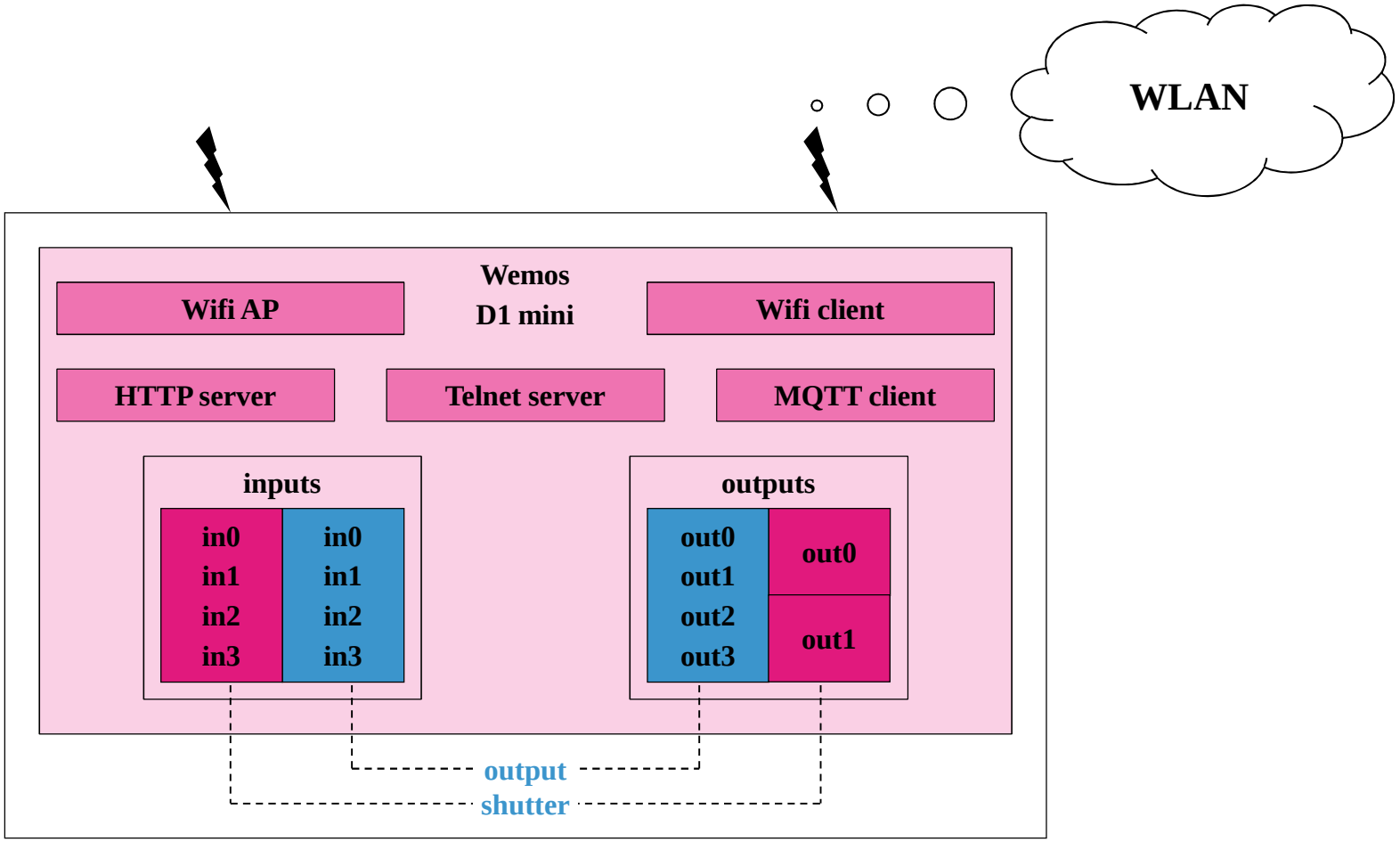
wiControl

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 (1/4)



Module Structure (2/4)

- 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

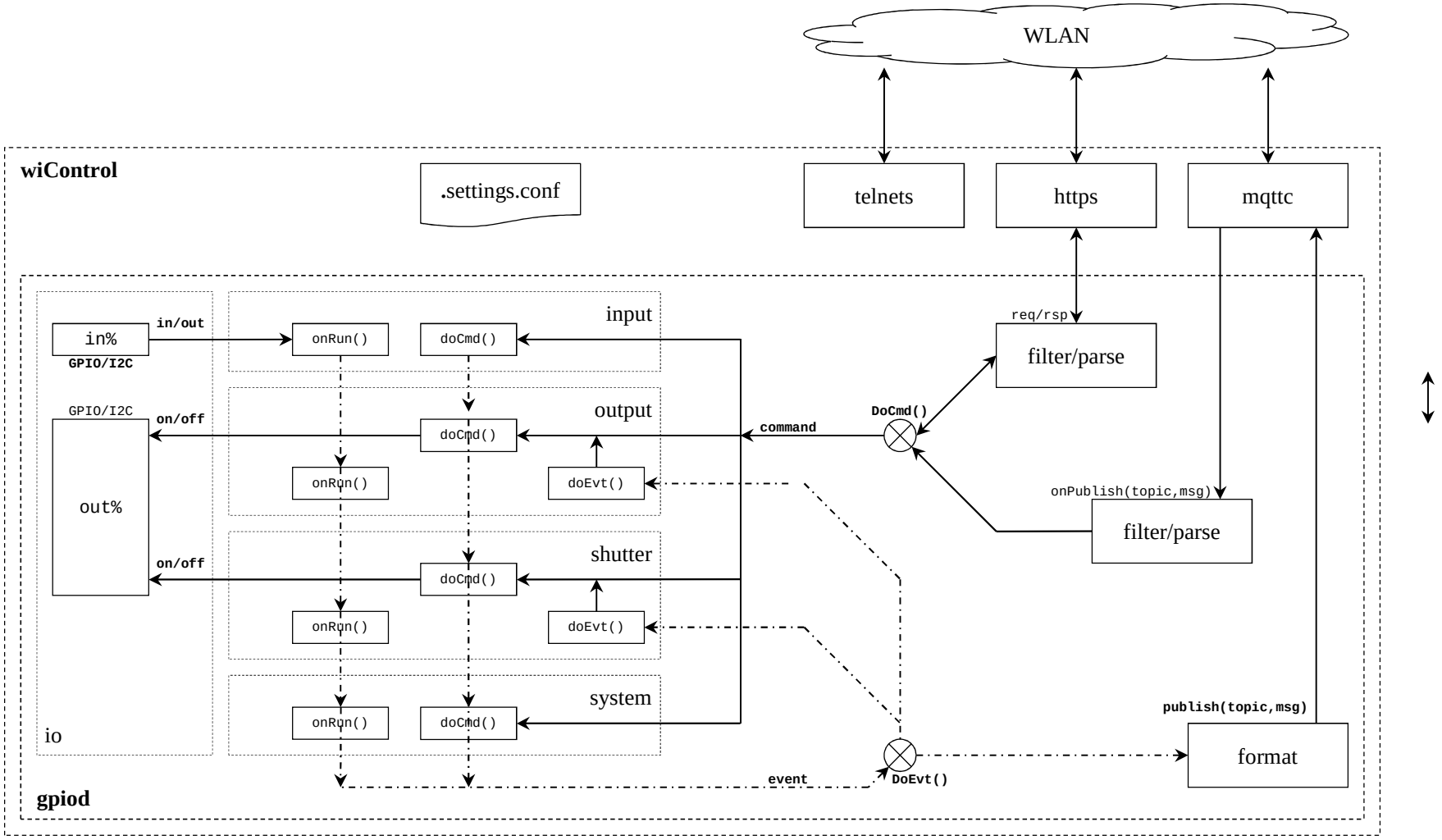
- Controllable objects included are:
  - 4 low-voltage inputs supporting normal-open push-buttons that connect to ground when pressed
  - 4 high-voltage (220V) SSR outputs upto 1A

- System LED
  - Helps to locate a module

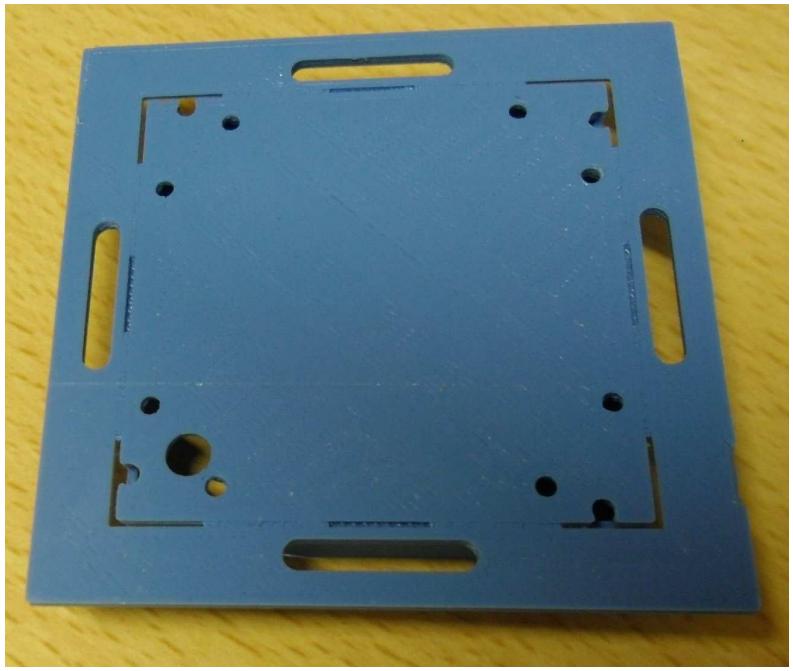
- 3D printed protective cover
  - Against accidental touching 220V contacts
  - For mounting in installation box
  - For attaching Niko blind frontplate

Module Structure (3/4)

- Functional Block Diagram



Module Structure (4/4)



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:
  - Standalone
    - inputs control outputs locally
    - requires no extra infrastructure to operate
    - emulator has fixed logic onboard to link input events to output commands
  - MQTT
    - connects to MQTT broker, so extra infrastructure is required
    - allows remote operation via MQTT protocol
  - Both
    - combines standalone and MQTT modes

- Safe mode:
  - 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
  - Also used for automatic testing
  - Once connected to broker
    - the module will subscribe to command topic <node-id>/cmd/#
    - the module will publish it's topology and version <node-id>/boo/hw=<topo>, <node-id>/boo/<sw>=<version>
  - In general
    - incoming publish messages will be handled by emulator <node-id>/cmd/<object>=<cmd> *[.<parm>]
    - object generated events are published to broker <node-id>/evt/<object>=<event>
    - object status requests are published to broker <node-id>/sta/<object>=<state>In general
  - Event/status formats
    - can be textual (default) i.e. myHouse/3/0/out0=on
    - can be numerical i.e. myHouse/3/0/out0=1

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
  - 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)

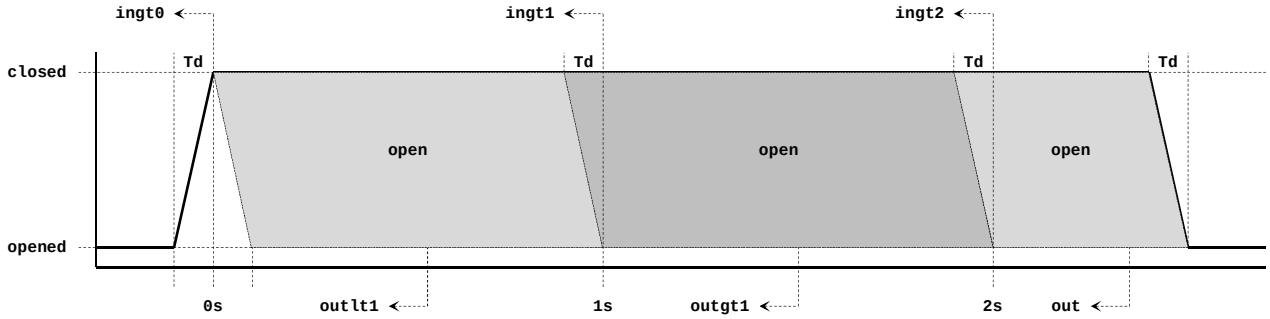
- Objects accept commands
  - That instructs it to do something and update it's state
  - MQTT <node-id>/cmd/<object>=<cmd> *[.<parm>] *[; <cmd> *[.<parm>]]
  - HTTP http://<ip-address>/ats?ccmd=<object>.<cmd> *[.<parm>] *[; <object>.<cmd> *[.<parm>]]
    - returns current state in numerical terse format,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

- When an object changes state
  - It will raise an event which will be distributed depending on configuration
  - To output objects as per emulation
  - To MQTT <node-id>/evt/<object>=<event>
  - Not to HTTP !!!

- A status command will report an objects state regardless of change
  - 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 concatinated, i.e. '0;1'

Inputs

- Input changes are validated with an adjustable decounce period (Td), default=150ms
- Validated changes are fed into a time based state machine to generate events



- Supported events
  - ingt0 (2) input was initially closed
  - outlt1 (3) input was opened after less than 1 second
  - ingt1 (4) input was closed for more than 1 second
  - outgt1 (5) input was opened after more than 1 but less than 2 seconds
  - ingt2 (6) input was closed for more than 2 seconds
  - out (7) input was opened after more than 2 seconds

- Supported commands
  - status return current state of input, 0=out|1=in
  - 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

Outputs – Output Emulation

- Supported commands
  - status — return current status of output, 0=off|1=on
  - on — turn output on
  - off — turn output off
  - onlocked — turn output on and lock it
  - offlocked — turn output off and lock it
  - toggle — toggle output between off and on
  - unlock — unlock output, all commands will be executed after this
  - ondelayed.<delay> — turn output on after <delay> seconds
  - offdelayed.<delay> — turn output off after <delay> seconds
  - ontimed.<run> — turn output on during <run> seconds, then back off
  - offtimed.<run> — turn output off during <run> seconds, then back on
  - toggledelayed.<delay> — toggle output after <delay> seconds, lock it till command done
  - toggletimed.<run> — toggle output for <run> seconds, lock it till command done
  - lock — lock output, no command exept unlock will be executed
  - locktimed.<run> — lock output for <run> seconds
  - timeset.<run> — set running time again to <run> seconds
  - timeadd.<run> — add <run> seconds to running time
  - timeabort — abort running time without unlocking or final action, will leave output as-is !!!
  - blink — alternate output between on and off with a 1s cadence, all outputs are synched
  - blinktimed.<run> — blink output for <run> seconds
- Supported events
  - off (0) — output was turned off
  - on (1) — output was turned on
- Parameters
  - delay,run — 1-3600 seconds

Outputs – Shutter Emulation (1/2)

- Uses the concept of priority level (0=highest, 5=lowest) and lock (0=no-lock, 1=lock)
  - 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

- Supported commands
  - status return current status of shutter, 0=stop|1=up|2=down
  - stop.<level> stop shutter
  - 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
  - up.<level>.<lock>.<run> move shutter up for given run time
  - down.<level>.<lock>.<run> move shutter down for given run time
  - tipup.<level>.<lock>.<run> short move up to tip the lamello's on a jalousie
  - tipdown.<level>.<lock>.<run> 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
  - delayedup.<level>.<lock>.<delay>.<run> move shutter up for given run time after delay
  - delayeddown.<level>.<lock>.<delay>.<run> move shutter down for given run time after delay
  - tipdelayedup.<level>.<lock>.<delay>.<run>.<tip> move shutter up for given run time after delay, do reverse tip move
  - tipdelayeddown.<level>.<lock>.<delay>.<run>.<tip> move shutter down for given run time after delay, do reverse tip move

Outputs – Shutter Emulation

- Supported Events
  - upon (1) — shutter started moving up
  - downon (2) — shutter started moving down
  - upoff (3) — shutter stopped moving up
  - downoff (4) — shutter stopped moving down

- Parameters
  - <level> — 0 (highest) – 5 (lowest)
  - <lock> — 0 (no lock) – 1 (lock)
  - <delay> — 0-3600 seconds
  - <run> — 1-3600 seconds
  - <tip> — 0-3600 1/10th seconds
  - <mask> — bit per level, 0=0x01, 1=0x02, 2=0x04, 3=0x08, 4=0x10, 5=0x20

System

- Supported commands
  - ping — will blink the LED for about 10 seconds
  - version — get current sw version
  - memory — get current memory statistics
  - loglevel [.<level>.ack] — get or set loglevel where level can be decimal or hexadecimal
  - emul [.<emul>.ack] — get or set emulation, 1=output|2=shutter
  - mode [.<mode>.ack] — get or set mode, 1=standalone|2=MQTT|3=both
  - efmt [.<format>.ack] — get or set MQTT event formatting, 1=numerical|2=textual
  - lock [.<lock>.ack] — get or set lock state of system commands, when locked only get commands and ping will work
  - restart.ack — restart module

- Supported events
  - version — 4.x.y.z
  - memory — %d
  - loglevel — %d or 0x%08X
  - emul — 1-2 or output|shutter
  - mode — 1-3 or %s
  - efmt — 1 or textual
  - lock — 0-1 or off|on

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

- Emulation=shutter
  - in0.ingt0/in1.ingt0 -> out0.stop
  - in0.ingt1 -> out0.up. <default-time>
  - in1.ingt1 -> out0.down. <default-time>
  - in2.ingt0/in3.ingt0 -> out1.stop
  - in2.ingt1 -> out1.up. <default-time>
  - in3.ingt1 -> out1.down.<default-time>

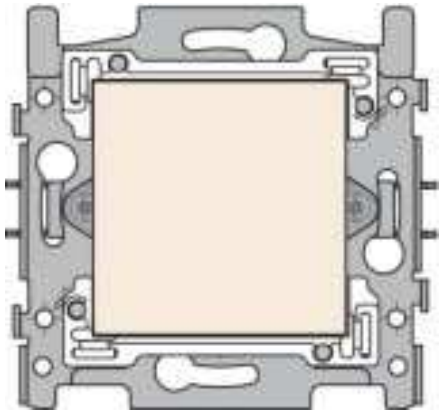
Outstanding

- Scaling of time parameters: seconds, 1/10 second, milliseconds… -> done
- Timer limitation: wrap-around after approximately 4200 seconds -> done
- Additional timer or clock objects
- Configuration commands
- Compact mode
  - reduce required pushbuttons
  - i.e. outlt1 -> out0.toggle, ingt2 -> out1.toggle
- Variants
  - DIN rail module with 8-16 outputs…
- Telnet command interface
- Shrink PCB to fit in Spelsberg in-wall box -> done
- Integrate with Niko quadruple pushbutton or blind frame -> done
- 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)

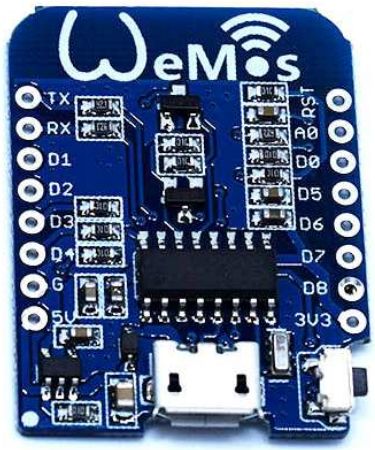
Niko Blindplate



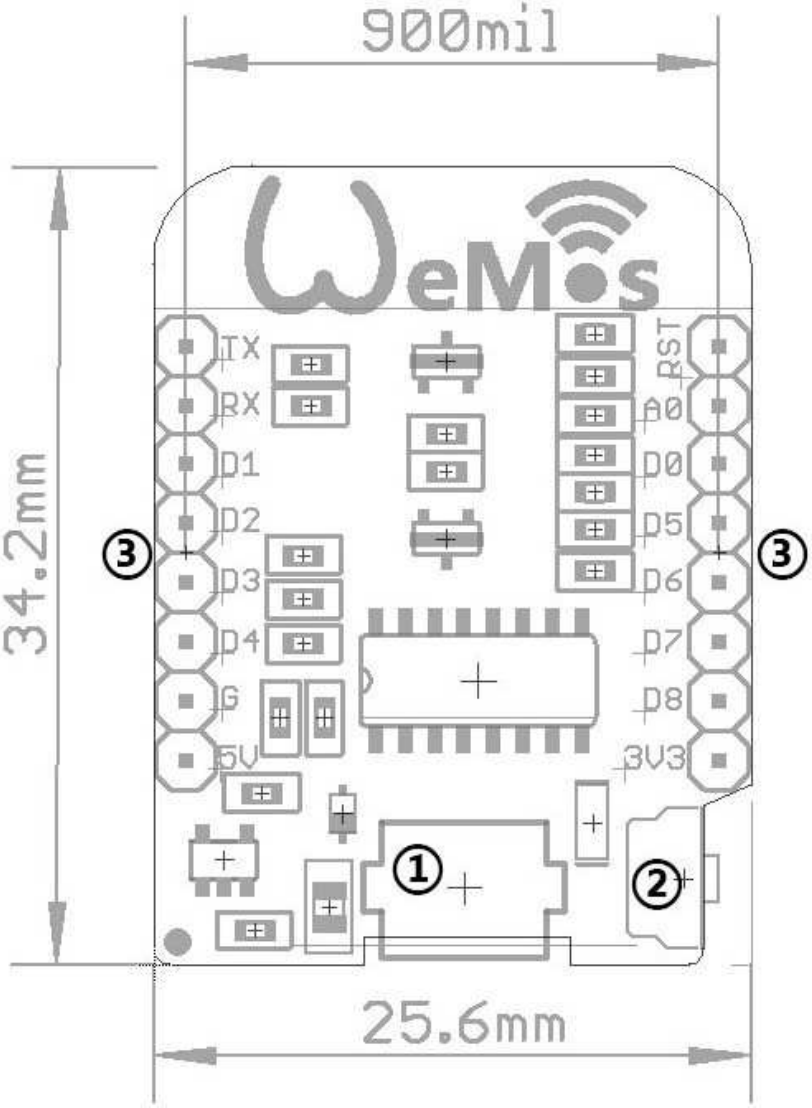
Wemos D1 Mini (1/2)

- ESP8266 based

TOP



BOTTOM



Wemos D1 Mini (2/2)