MQTT IOMODUL 16DI/16DO

PRODUCT DESCRIPTION

- Dimensions (L/W/H): 104 x 92 x 30 mm
- Power supply module 5V / 190mA (idle mode)
- 16 digital inputs, 12-24V based, galvanically isolated
- 16 digital outputs, 12-24V based, galvanically isolated, maximum switching load per output: 50 mA
- Ethernet RJ45 interface
- MQTT protocol (3.1.1, QoS 0, Port 1883) is used to set an output or inform about input changes
- Status inputs and outputs as well as status of modul is signaled by LED
- Easy parameter configuration (IP addresses, debounce time, …)
- No specialized outputs, so that voltages up to 230V can switched via coupling relay
- No built-in power, input and output connectors (spacing: 3,5 mm)

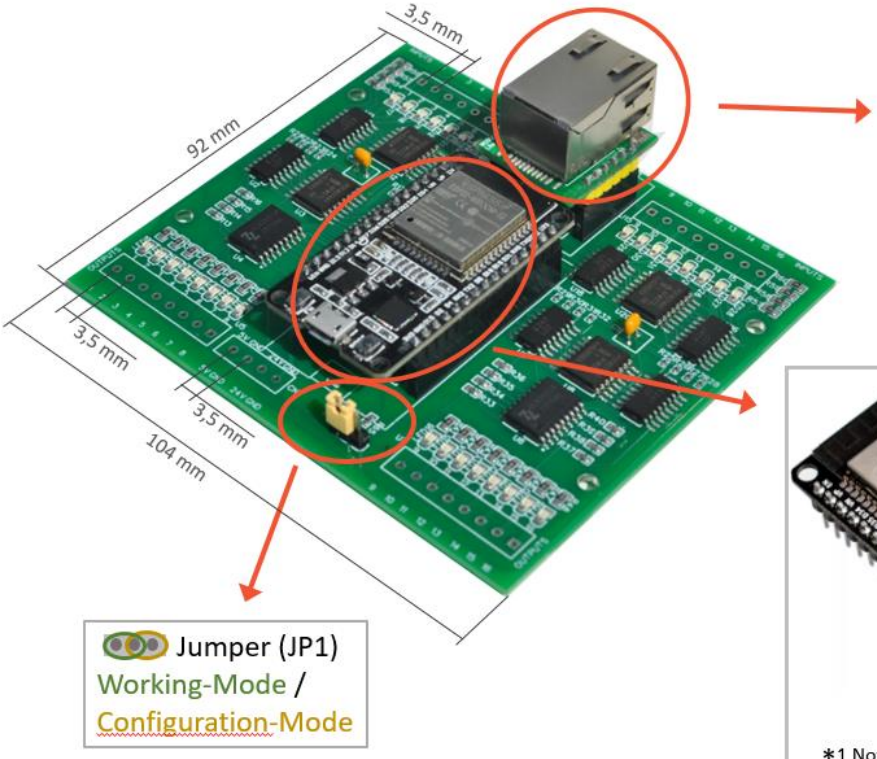


Figure 1: circuit board with jumper

Figure 3: Wiznet W5500

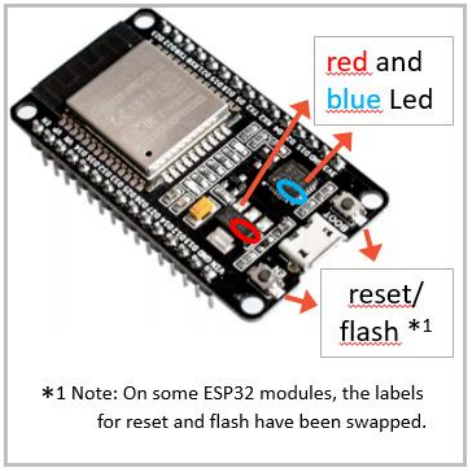


Figure 2: ESP32 module

HARDWARE

A stable 5V external power supply is required for operation. The 5V supply is either via socket (connector CN 1) on the circuit board or via the USB-UART interface of the ESP32. For reasons of stability, we recommend using the connector on circuit board. The simultaneous use of 5 V on the circuit board and USB-UART to supply the device can lead to damage.

External 12-24 V voltage power supply is required for operation of I/O ports.

5V ground line and 24V ground line are routed separately on the board, connection of both lines removes galvanic separation (see circuit diagram).

Interface Definition

INPUT 01-16 : Digital input channels

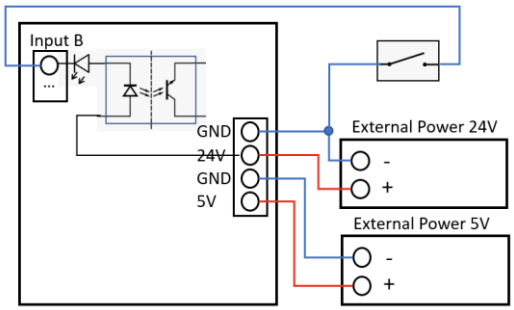
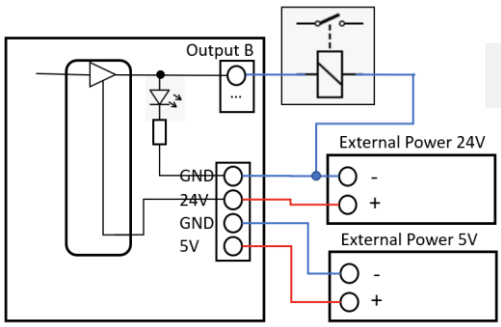


Figure 4: Circuit design for inputs

OUTPUT 01-16 : Digital output channels



Maximum output current 50mA!

Figure 5: Circuit design for outputs

5V : 5V power supply for operation
GND : ground line 5V

24V : 12-24V power supply for operation of I/O ports
GND : ground line 24V

CONFIGURATION-MODE

When jumper (JP1) is set to yellow circled slot (see figure 1) while ioModul is booting configuration-mode is set and the module starts as HTTP server. In this mode the ioModul always tries to establish a DHCP connection and the DHCP server assigns ioModul an IP address. When HTTP server started successfully the blue LED on ESP32 module (see figure 2) flashes every 100ms. If problems occurred during boot, the blue LED signals this by a different flashing code (see table blue LED indicator description).

Determine IP address of the ioModul and enter IP address (port 80) in a browser to configure settings.

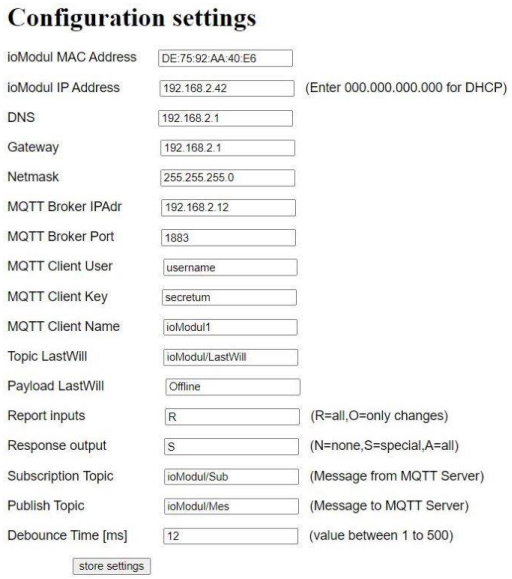


Figure 6: HTML page settings

Additional information:

- Spaces and special characters are not permitted. Exceptions: forward slashes (/) in topic fields, colons (:) in MAC address and periods (.) in IP address fields.
- Maximum path length: 60 characters
- If a non-compliant attribute value is found, data will not be saved.
- *"Subscription Topic":* message topic on which the module is waiting to react to incoming MQTT messages (e.g. switch output 12, provide log messages, etc.)
- *"Publish Topic":* ioModul sends notifications to MQTT broker with defined topic
- *"Topic or Payload LastWill":* if one of these fields is empty, the ioModul does not register "LastWill" message when connecting to broker.
- *"Report input":* set verbosity of inputs, possible values are: R= send notification on reboot, reconnection and changes; O=only on changes, send no notification on reboot or reconnection
- *"Response output":* set verbosity of output, possible values are: N=no response; S= Special, only output which has been written is returned; A= all outputs are returned
- *"Debounce time":* Value between 1 and 500. Time in milliseconds until module allows a new state change at input channels, status changes on a PIN generate a new MQTT message. Status changes within debounce time threshold are not passed through, new status is only forwarded after debounce time elapsed.

WORKING-MODE

When jumper (JP1) is set to green circled slot (see figure 1) while ioModul is booting working-mode is set and module works as MQTT client. The device runs with a fixed IP address or an IP address assigned by the DHCP server, depending on the configuration. If the ioModul is running most of the time, a fixed IP address is recommended. When MQTT client started successfully the blue LED on ESP32 module (see figure 2) flashes slowly (1000ms on and 400ms off). If problems occurred during boot the blue LED indicates this by a different flashing code (see table blue LED indicator description).

After a successful MQTT connection is established, the following messages can be sent to the ioModul and the module will respond as described below.

a) Request with "…/**State**" and Payload "**0**" to ioModul => ioModul sends "Online" message in response.

	TOPIC	PAYLOAD
REQUEST	<Subscription Topic>**/State**	**0**
RESPONSE	<Message Topic>/State	Online

Example with default topics:

	TOPIC	*PAYLOAD*
REQUEST	*ioModul/Sub/State*	*0*
RESPONSE	*ioModul/Pub/State*	*Online*

b) Request with "…/**State**" and Payload "**1**" to ioModul => ioModul sends state of all inputs as JSON string in response.

	TOPIC	PAYLOAD
REQUEST	<Subscription Topic>**/State**	**1**
RESPONSE	<Message Topic>/State	{"Inputs": {"01": "<ON or OFF>", ... "16": "<ON/OFF>"}}

Example with default topics:

	TOPIC	*PAYLOAD*
REQUEST	*ioModul/Sub/State*	*1*
RESPONSE	*ioModul/Pub/State*	*{"Inputs": {"01": "ON", … "16": "OFF"}}*

c) Request with "…/**State**" and Payload "**2**" to ioModul => ioModul sends state of all outputs as JSON string in response.

	TOPIC	PAYLOAD
REQUEST	<Subscription Topic>**/State**	**2**
RESPONSE	<Message Topic>/State	{"Outputs": {"01": "<ON or OFF>", ... "16": "<ON/OFF>"}}

Example with default topics:

	TOPIC	*PAYLOAD*
REQUEST	*ioModul/Sub/State*	*2*
RESPONSE	*ioModul/Pub/State*	*{"Outputs": {"01": "ON", … "16": "OFF"}}*

When an **external 12-24 V power supply** is connected to the ioModul (in addition to the existing MQTT connection) input and output ports are fully operational.

d) For switching an **output** send the following MQTT message to the ioModul. The module responds depending on the value "Response output" (see configuration settings).

	TOPIC	PAYLOAD
REQUEST	<Subscription Topic>/**OUT**<**xy**>	**<ON** or **OFF>**
RESPONSE	<Message Topic>/OUT<xy>	<ON or OFF>

Example with default topics (Response output A)

	TOPIC	*PAYLOAD*
REQUEST	*ioModul/Sub/OUT01*	*ON*
RESPONSE	*ioModul/Pub/OUT01*	*ON*

e) If an **input** is switched (i.e. is pulled to ground or is removed from ground) the module sends the following MQTT message to the MQTT broker.

	TOPIC	PAYLOAD
MESSAGE	<Message Topic>/**IN**<**xy**>	**ON or OFF**

Example with default topics for input 1:

	TOPIC	*PAYLOAD*
MESSAGE	*ioModul/Pub/IN01*	*ON*

Special functions for monitoring ongoing operations.

f) Request with "…/**State**" and Payload "**3**" to ioModul => ioModul sends version information as JSON string in response.

	TOPIC	PAYLOAD
REQUEST	<Subscription Topic>**/State**	**3**
RESPONSE	<Message Topic>/State	{"Versioninfo": {"Version": …} }

g) Request with "…/**State**" and Payload "**4**" to ioModul => ioModul sends runtime informations in response.

	TOPIC	PAYLOAD
REQUEST	<Subscription Topic>**/State**	**4**
RESPONSE	<Message Topic>**/State**	{"RTI": {"maxLoopTime": <xyzv>, … } }

Example with default topics:

	TOPIC	*PAYLOAD*
REQUEST	*ioModul/Sub/State*	*0*
RESPONSE	*ioModul/Pub/State*	*{"RTI": {"maxLoopTime": 51, "minLoopTime": 4711, "timeAverage": 66, "timeCurrent": 67, "loopsOverHun": 5, "loops": 483376, "ipAdr": "192.168.2.12", "reconCause": "reboot"}}*

Structure of runtime informations:
All tasks are sequentially processed. When module returns back to first task a loop is completed. Required time for loop completion varies, therefore the following information is collected (in addition to IP Address and reconnect cause):

- *"maxLoopTime"*: Longest time the module needed for loop completion; resetted with every request (state 0)
- *"minLoopTime"*: Shortest time the module required for loop completion; resetted with every request (state 0)
- *"timeAverage":* average loop time
- *"timeCurrent":* current loop time
- *"loops" and "loopsOverHun":* ioModul counts amount of loops that needed more than 100us, based on the value "loops"; this counter is resetted after 1000000 loops.
- *"ipAdr": IP address of the ioModul*
- *"reconCause": Reason why it was necessary to reconnect (reboot, Ethernet was interrupted, MQTT was interrupted, root cause not defined)*

Times are in microseconds.

h) Request with "…/**State**" and Payload "**5**" to ioModul => ioModul sends last 16 logging messages and microsecond in multiple responses.

	TOPIC	PAYLOAD
REQUEST	<Subscription Topic>**/State**	***5***
RESPONSE	<Message Topic>**/State**	{"Log": {"01": "<micros>: <log01>"} }
RESPONSE	…	…
RESPONSE	<Message Topic>**/State**	{"Log": {"16": "<micros>: <log16>"} }

i) ioModule can be set to send a MQTT message if loop completion takes longer than a defined number of microseconds by sending the following message. Request with **"…/State"** and payload **4-digits** (number of microseconds between 1000 and 9999) defines the threshold. This functionality is not activated by default but must be explicitly activated with every restart.

	TOPIC	PAYLOAD
REQUEST	<Subscription Topic>**/State**	**<vxyz>**

Message if loop completion takes longer.

	TOPIC	PAYLOAD
MESSAGE	<Message Topic>/State	{"Timeloop": <vxyz>}

Example for reporting all loops taking more than 2000 microseconds:

	TOPIC	*PAYLOAD*
REQUEST	*ioModul/Sub/State*	*2000*

MQTT message example when IoModul requires 3512 microseconds for loop completion.

	TOPIC	*PAYLOAD*
MESSAGE	*ioModul/Pub/State*	*{"Timeloop": 3512}*

RUNNING IOMODUL

The following instructions explain how to setup ioModule for first usage.

It is assumed that a DHCP server and a MQTT broker is already installed in your network and can be accessed using it's IP address and port. Lots of tutorials about MQTT message broker setup are already available online, Mosquitto is the recommended MQTT broker here.

For testing purposes it is also very helpful to have another MQTT client already and working (i.e. it can send and subscribe to messages, consider using mqtt.fx or Chrome MQTTLens as MQTT test client). Then subscribe to topic "ioModul/#" on the MQTT client so that you can follow all messages to and from ioModul.

Before starting make sure you have flashed the ESP32 module with the firmware (if not see Flash ESP32) and that ESP Module and Wiznet W5500 are installed in the correct orientation on the board (figure 1). Ethernet cable must be plugged into the W5500 and DHCP server must be running. Set jumper (JP1) to yellow circled slot (see figure 1), the Configuration-Mode.

Connect a stable 5V external power supply to the board. The red LED of the ESP32 (see figure 2) lights up when the module is powered on. The blue LED of the ESP32 must start to light up. Please wait a short time until the final state is reached. The LED indicates operating mode. Slow flashing (2000ms on/ 400ms off) indicates operation in Working-Mode, fast flashing (100ms on/ 100ms off) operation in Configuration-Mode. Other flashing indicates a problem (see table Indicator blue LED description).

When module is in Configuration-Mode, the DHCP server has assigned an IP address to the device. Use a network scanner (for example SoftPerfect Network Scanner or something similar) to determine its IP address. Enter the determined IP address in a browser.

Please change only its IP address, DNS, gateway, netmask and MQTT IP address in configuration settings in first step. If MQTT Broker is configured to require client authentication before a connection is allowed, enter a valid username (MQTT user) and password (MQTT key). Leaving remaining fields to default settings will make further steps easier. Settings can be changed at any time if required. Save your settings.

Set the jumper to Working-Mode (see figure 1) and reboot ioModul by pressing the reset button on ESP32. ioModul should now start in Working-Mode with LEDs flashing slowly.

Send the following MQTT message to the server using a MQTT client:

Topic: ioModul/Sub/State *Payload: 0*

The ioModule should send a message back to MQTT broker which distributes the message to all subscribers:

Topic: ioModul/Pub/State *Payload: Online*

An external power supply (12-24 V) must be connected to the ioModul for testing inputs and outputs. For details see figure 4 and 5.

Send the following MQTT message to the server using a MQTT client:

Topic: ioModul/Sub/OUT01 *Payload: ON*

Output LED01 should light up.

Connect input 01 to ground. The Input LED01 should light up and the ioModul should send the following message:

Topic: ioModul/Pub/IN01 *Payload: ON*

RESPONSE TIMES

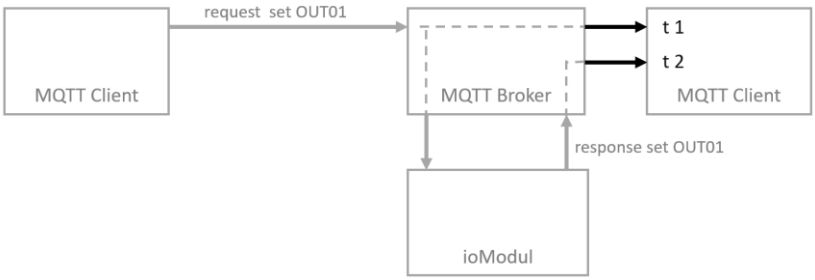
Switching times depend on system landscape and current ioModule load including network and MQTT broker also latency of all components involved (i.e. ethernet latency, computer hardware and operating system latency) is an influencing factor. Realtime ability is also not given as recipient does not acknowledge message receipt and message sender does not store and re-transmit messages.

Reference times in this document were determined in a 1000 Mbit/s LAN using Mosquitto server on a Raspberry PI 3 with fixed IP address. Times were measured while CPU was idle.

Response times listed here should provide an order of magnitude that is usually reached. Individual temporal outliers are possible (due to e.g. ethernet instability). Please note that time compliance therefore cannot be guaranteed.

- A "normal" loop pass takes less than 100us. However, the MQTT client updates its status on the MQTT broker (KeepAlive) before an agreed time period elapses. In above setup, this happens about every 100.000 passes loop time then increases to about 1,5ms. If MQTT session is reconnected or ethernet connection is unstable the time for loop completion increases.

- Time (t2-t1) to set an output from send a MQTT message to the ioModul to receiving the response that the output has been set: about 45ms



- Time (t2-t1) to set an output from send a MQTT message to the ioModul to receiving the message that an electrically connected input has been changed: about 45ms

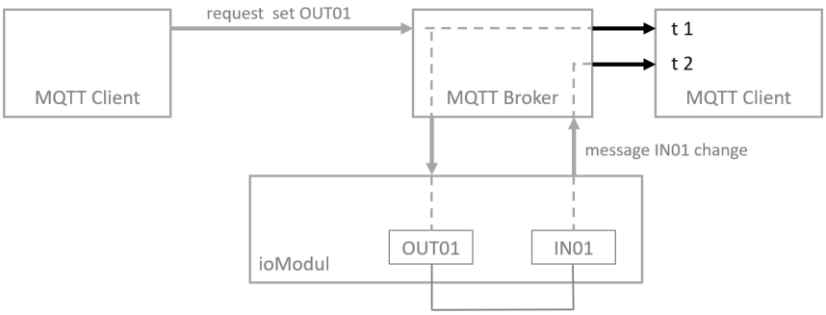
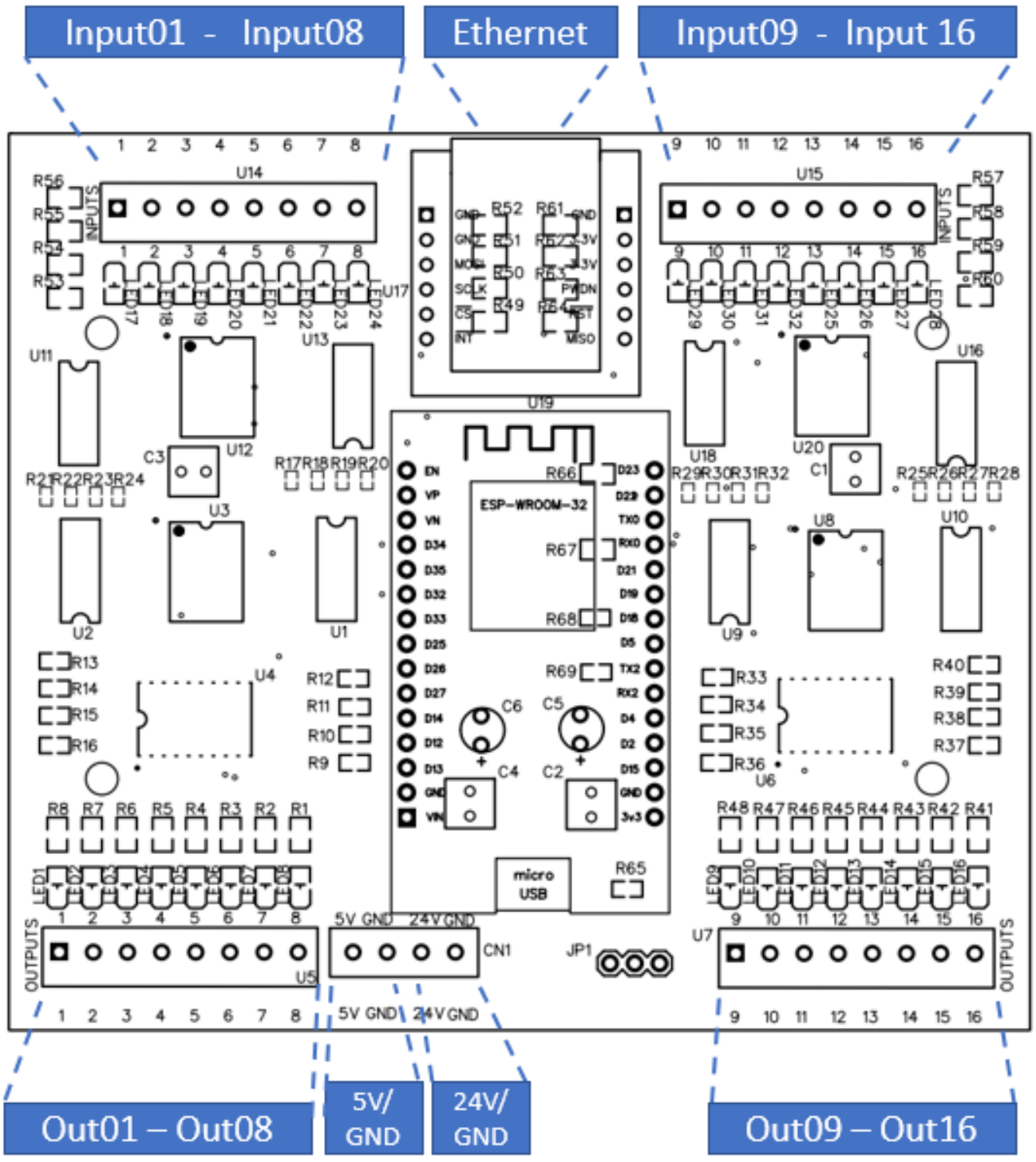


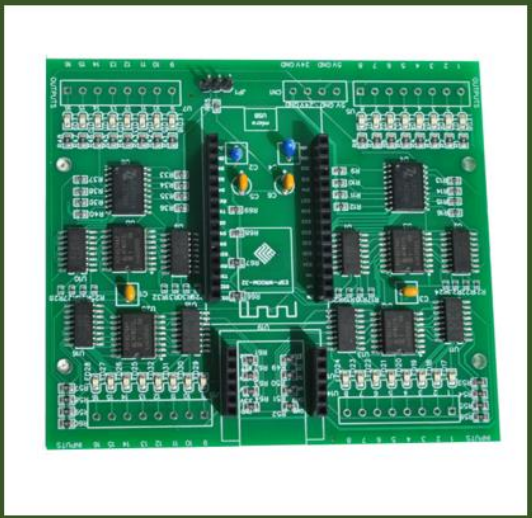
TABLE BLUE LED INDICATOR DESCRIPTION

Status	LED	Class	Description
WORK_MODE	2000ms(on) 400ms (off)	normal	ioModul operates in Working-Mode as MQTT-Client and sends and receives MQTT messages.
CONF_MODE	100ms (on) 100ms (off)	normal	ioModul operates in Configuration-Mode as HTTP-Server. Start configuration.
FILE_ERROR	1x400ms pause (1500)	error	Problems with file operations, if necessary, change ESP32.
ETH_ CONNECT	2x400ms pause (1500)	problem	ioModul fails with DHCP connection; please check ethernet network connection.
MQ_CONNECT	3x400ms pause (1500)	problem	ioModul is unable to connect to MQTT broker. Please check broker availability.
ETH_CABLE	4x400ms pause (1500)	problem	Please check physical ethernet connection.
W5500_ ERROR	5x400ms pause (1500)	error	Problems with ethernet shield, if necessary, change wiznet W5500.
CDATA_ ERROR	6x400ms pause (1500)	problem	Problems with configuration data. Please check data in Configuration-Mode.
MAC_ERROR	1200, 1x400ms pause (1500)	problem	MAC-Address not available, default address is used. Please change MAC-Address.
MQIP_ERROR	1200, 2x400ms pause (1500)	problem	Problems with IP address or port of the MQTT broker. Please check configuration data.
MOIP_ERROR	1200, 3x400ms pause (1500)	problem	Problems with IP address of the ioModul. Please check configuration data.
DGN_ERROR	1200, 4x400ms pause (1500)	problem	Problems with DNS, gateway or netmask. Please check configuration data.
PCF_ERROR	1200, 5x400ms pause (1500)	error	Problems with I2C devices, if necessary, change circuit board
	1200, 7x400ms pause (1500)	error	system error This error should never occur ☹.

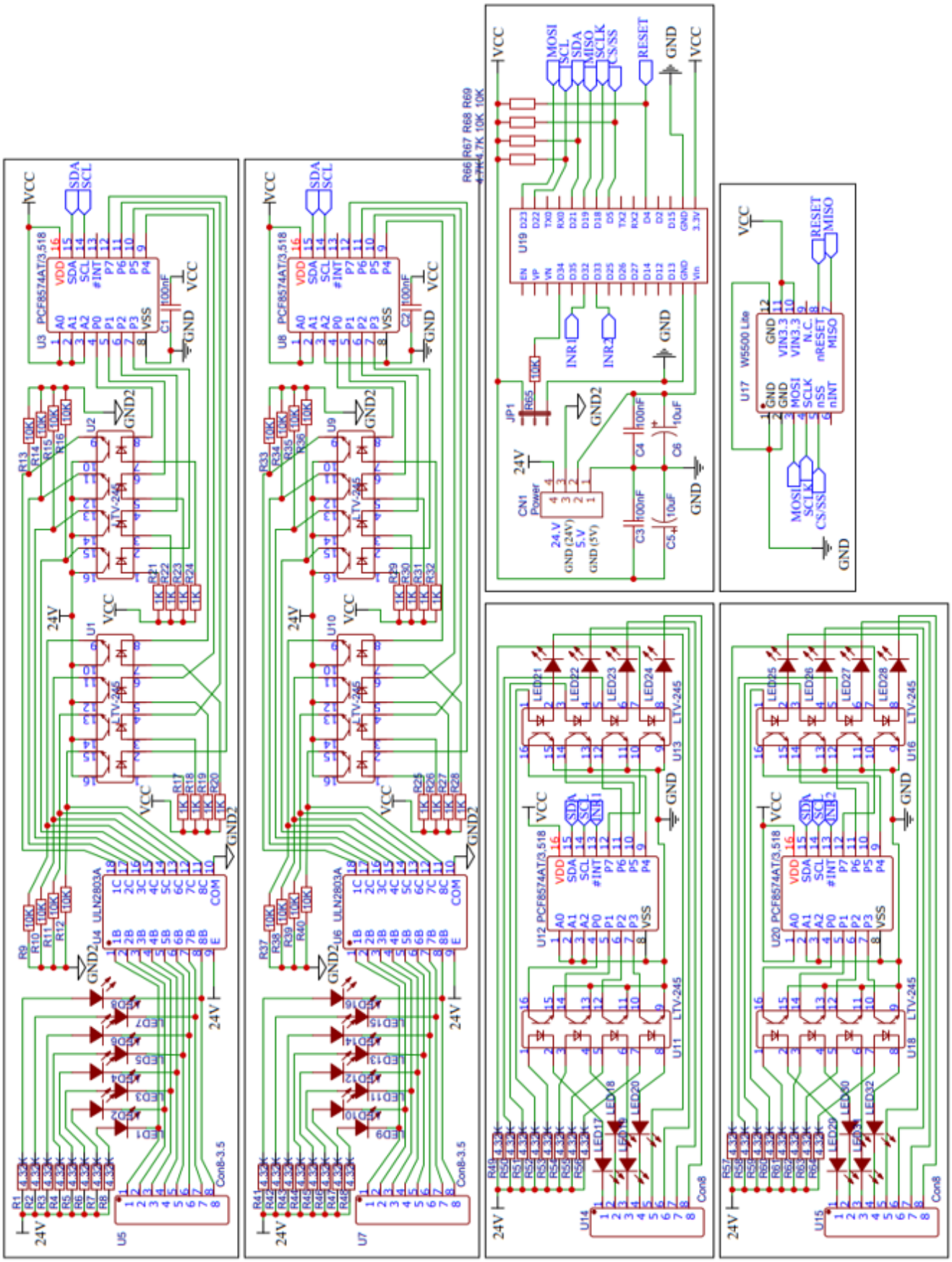
BOARD


Input01 - Input08
Ethernet
Input09 - Input 16
ESP-WROOM-32
micro USB
5V GND
24V GND
CN1
JP1
OUTPUTS
Out01 – Out08
5V/ GND
24V/ GND
Out09 – Out16

PHOTOS



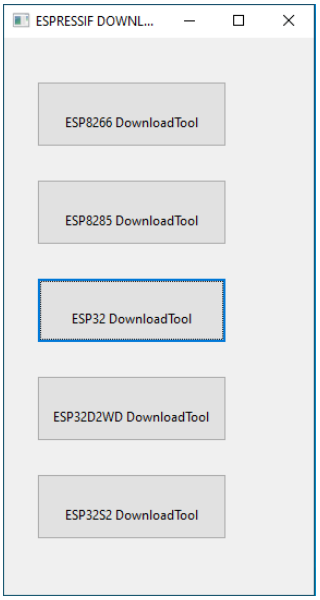
CIRCUIT


VCC
SDA
SCL
PCF8574AT/3.518
ULN2803A
LTV-245
GND
GND2
24V
MOSI
SCL
SDA
MISO
SCLK
CS/SS
RESET
INR1
INR2
W5500 Lite
Power
24.V
GND (24V)
5.V
GND (5V)
Con8
Con8-3.5

FLASH ESP32

The following steps describe how to flash your ESP32 by uploading the BIN files via the USB-to-UART adapter.

PLEASE NOTE THAT NO EXTERNAL 5 VOLT POWER SUPPLY MAY BE CONNECTED DURING THIS PROCESS!!!



1.) Download and store ioModul binary files from http://xmmqtt.de on your computer

2.) Download and unzip the “Flash Download Tools (ESP8266 & ESP32 & ESP32-S2)” from https://www.espressif.com/.

3.) Start the flash_download_tool_3.8.5.exe

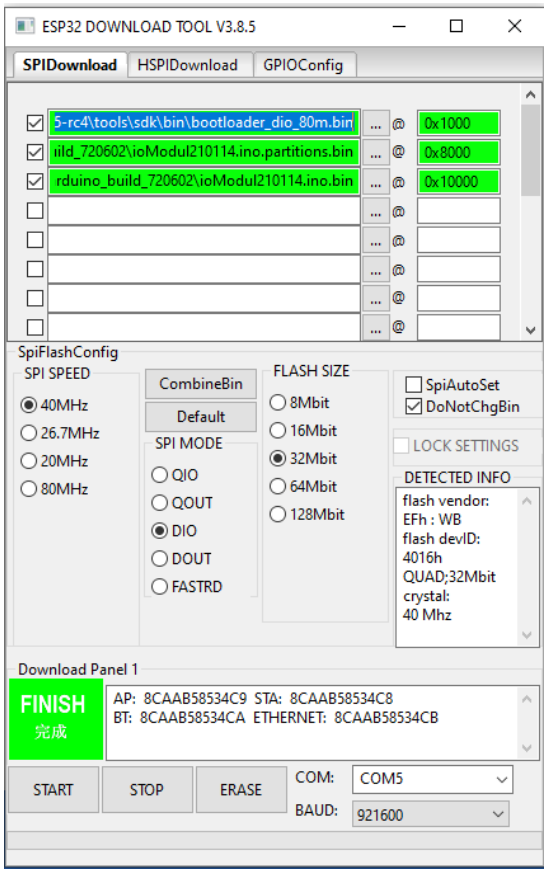
4.) Press “Developer Mode” and then “ESP32 Download Tool”

5.) Check that the ESP32 is connected to the Computer and select the COM-Port

6.) Press “START”. The synchronization with the ESP32 starts. If data was successfully read by the ESP, it is displayed in the lower area (green status field shows “FINISH”).

7.) Enter the path information to bin files followed by Flash parameter (data marked in green in upper area)

8.) Press “START” again.



If the download to ESP32 does not start automatically switch the ESP32 to flash mode. After successful completion green status field shows “FINISH”.

ESP32 is now successfully flashed.