1. General information

1.1. Description and characteristics

The world is evolving at a breakneck pace, and we are helping it do so. To support developers, innovators, enthusiasts, and others, we have created the Atom Black Edition product, and provide the opportunity to use our platform to develop new products and solutions in the world of automation or IoT. Our devices are designed with a modular architecture, giving you a possibility to choose any peripherals you need for your research or development.

We have developed two types of controllers in this series to suit any of your needs. We provide you with the ability to program under STM32 and ESP32 controllers, which meet the standards of stability and reliability in today's world.

This series can be used when writing your own software, as well as to use existing solutions popular in the world of automation and IoT (micropython, esphome.io, tasmota and others).

Our devices can be used by integrators, laboratories, startups, as well as educational institutions to train the next generation of engineers. Atom Black Edition series controllers are perfectly protected against overloads, erroneous connections and also have excellent protection in the form of the case, to prevent accidental contact with electrically conductive parts.
Atom Black Edition is based on the powerful and multifunctional module Espressif ESP32-PICO-D4.

Features of ESP32-PICO-D4:

- Based on ESP32, 32-bit Xtensa® dual-core LX6 (600 DMIPS)
- 32 Mbps Flash, 520 KB RAM, 448 KB ROM, 1 KB eFuse
- Standards 802.11 b / g / ni and Bluetooth v4.2 BR / EDR / BLE
- Placing all the necessary components inside the chip
- Interfaces GPIO, UART, SPI, SDIO, I2C, PWM, I2S, IR, ADC, DAC
- Built-in Hall and temperature sensors

In addition to the ESP32 module, the board has an I2C GPIO expansion module, a USB-UART converter and a power stabilizer.

This solution opens up the possibility of using Atom series controllers in already established or new automation systems as a multifunctional I / O unit with extensive integration capabilities. The most popular platform for implementing a "smart home", where Atom controllers can be used based on the Espressif ESP32-PICO-D4 module - Home Assistant with ESPHome control system.

ESPHome is a system for managing ESP8266 / ESP32-based devices with simple but powerful configuration files and remote control with home automation systems.

Detailed characteristics of ESP32-PICO-D4

1.2. Modifications

All models of Atom controllers with proper configuration files are suitable for use in ESPHome automation systems. The difference from the basic models is that the controller uses a board based on ESP-Pico-D4 instead of the CPU board STM32. There are also no CAN and RS-485 interfaces. Connectors in the appropriate positions are used to program the controller with a USB cable.

Modifications with the OLED screen which can be easily adjustable in a configuration file are available.

2. Getting started - firmware, settings.

To work in the automation system, the controller requires configuration, which is done by loading to it a configuration file in *.yaml format. The controller can be used both as a separate control device and as part of the Home Assistant automation system using the ESPHome integration software module. In both cases, the first upload of the configuration file should be done using the USB interface, all subsequent uploads are easier to do via OTA-update, using the home WiFi network.

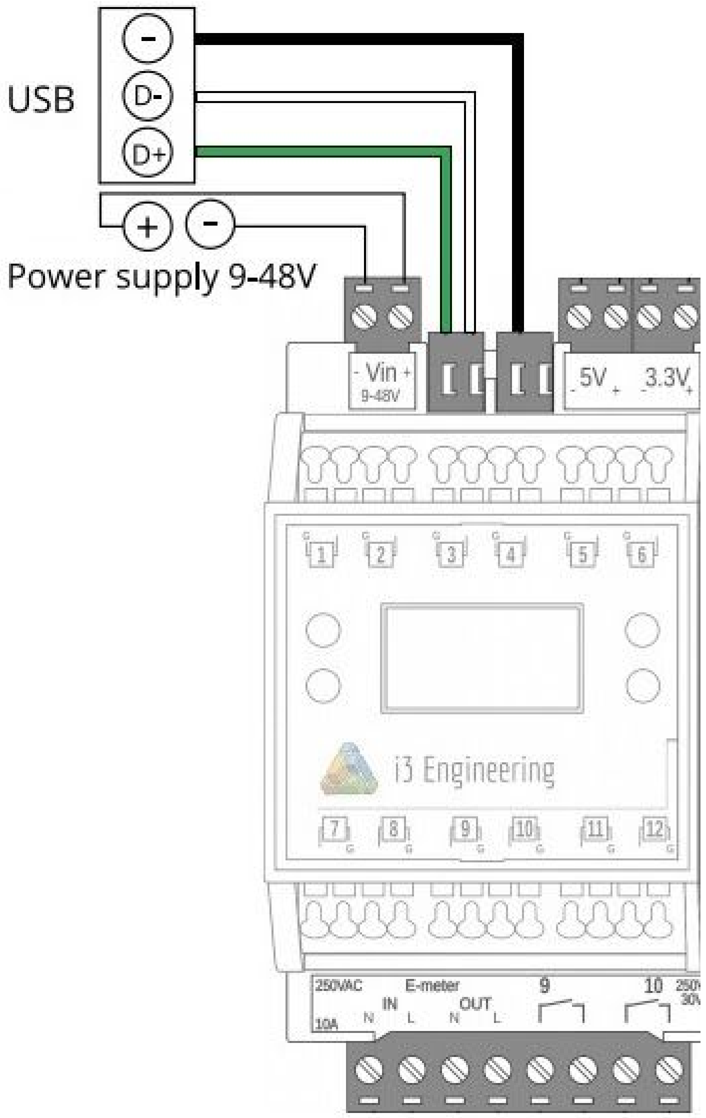
At the software level, the command line functionality is sufficient to load the configuration into the controller, and the built-in web server is sufficient for monitoring and control, which allows monitoring the status of inputs and outputs and information from sensors in digital format, and provides a simplified interface for managing connected devices.

Incomparably more convenient and functional is the solution based on Home Assistant which, in addition to advanced graphical interface for control and management, allows you to integrate the controller into a home automation system, provide data exchange between devices, coordinate the system as a whole.

2.1. Options for connecting the device to upload the firmware

Firmware upload via USB

1. Download and install the built-in USB-UART adapter driver.
2. Connect the USB cable to the controller according to the diagram



3. Turn on power to the device.
4. Connect the USB cable to the computer. When using a controller in a Home Assistant-based system - to the system control unit (eg Raspberry Pi USB port)
5. Start the firmware upload process from the command line or from the Home Assistant according to the appropriate instructions.
6. Create the basic configuration of the controller according to the wizard, type of controller - "pico32".
7. Check that the controller is connected to the WiFi network.

OTA update

All the following settings of the controller connected to the home WiFi network are convenient to do using OTA. If the USB cable is disconnected from the computer, no additional action is required - the WiFi connection and the firmware upload process will take place automatically, according to the entered commands.

2.2. Software tools to upload the firmware to the device

Firmware upload through the command line

In the simplest version of the controller settings, it is enough to use the command line to communicate with the processor, and any text editor (Notepad or Notepad ++) to edit the configuration file. The full English-language instruction is available at:

https://esphome.io/guides/getting_started_command_line.html

Firmware upload in the Home Assistant system

In case of integrating the controller into an already created and configured Home Assistant system, you should use the capabilities and functionality of the software integration module ESPHome. Detailed process instructions are described at the following links:

https://www.home-assistant.io/integrations/esphome/

https://esphome.io/guides/getting_started_hassio.html

3. Creating a configuration file

3.1. Connection web-server

```
web_server:
  port: 80
```

3.2. Prefill sntp-servers

```
time:
  - platform: sntp
    id: sntp_time
    servers:
      - 0.pool.ntp.org
      - 1.pool.ntp.org
      - 2.pool.ntp.org
```

3.3. If you are using an OLED display:

3.3.1. Initialize the I2C bus

```
i2c:
  sda: 21
  scl: 19
  scan: True
```

3.3.2. Install fonts. To do this, upload the desired font files in *.ttf format to the user's root folder. Next, add them in the configuration file.

```
font:
  - file: 'arial.ttf'
    id: font1
    size: 16
  - file: 'arial.ttf'
    id: font2
    size: 35
  - file: 'arial.ttf'
    id: font3
    size: 14
```

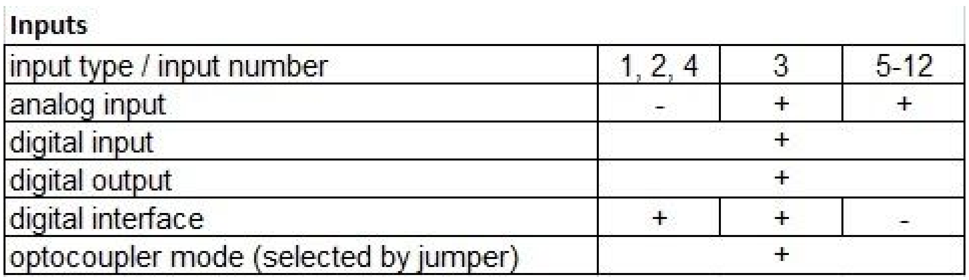
3.3.3. Connect the display and create an image design. An example of display settings with the display of the manufacturer's name of the controller, the current time and readings of 2 temperature sensors.

```
display:
  - platform: ssd1306_i2c
    model: "SSD1306 128x64"
    address: 0x3C
    update_interval: 3s
    lambda: |-
      it.printf(64, 0, id(font1), TextAlign::TOP_CENTER, "i3 Engineering");
      it.strftime(0, 55, id(font2), TextAlign::BASELINE_LEFT ,"%H:%M", id(sntp_time).now());
      if (id(temp1).has_state()) {
        it.printf(127, 23, id(font3), TextAlign::TOP_RIGHT ,"%.1f°", id(temp1).state);
        }
      if (id(temp2).has_state()) {
        it.printf(127, 60, id(font3), TextAlign::BASELINE_RIGHT ,"%.1f°", id(temp2).state);
        }
```

3.4. Inputs settings

The entire list of equipment available for connection to the controller inputs, with the corresponding settings in the configuration file is presented on the ESPHome project page.

https://esphome.io/index.html

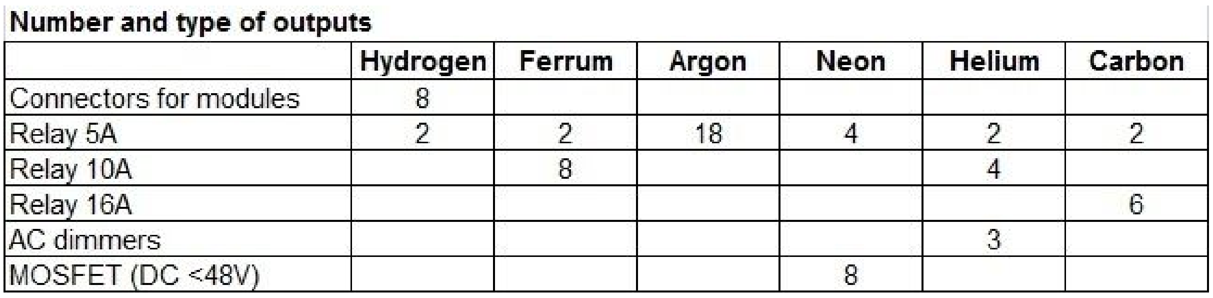


Inputs

input type / input number	1, 2, 4	3	5-12
analog input	-	+	+
digital input		+	
digital output		+	
digital interface	+	+	-
optocoupler mode (selected by jumper)		+	

3.5. Outputs settings

Actuating devices that can be connected to the outputs of the controller, and their number, can be selected depending on the model of the controller:

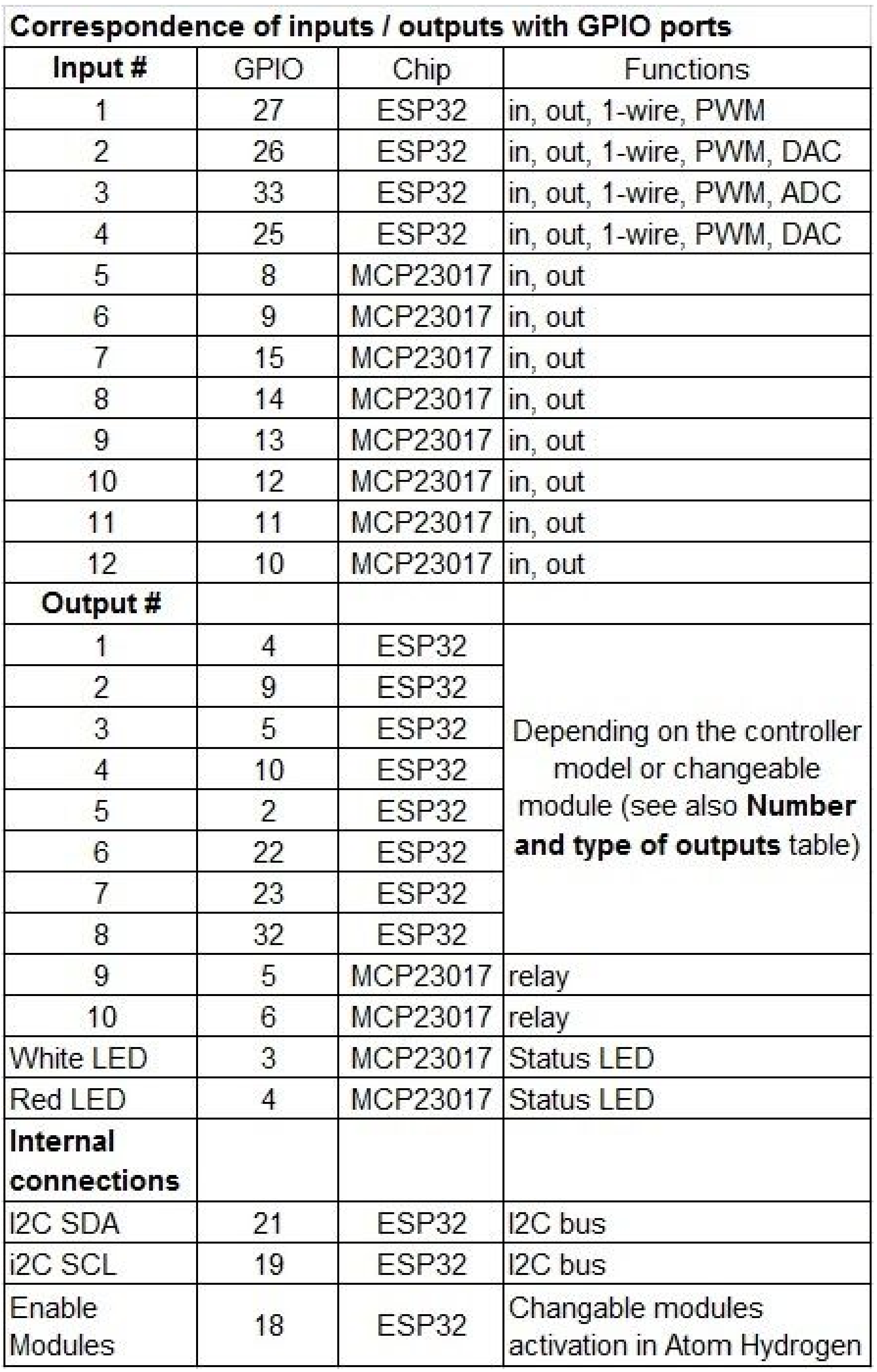


Number and type of outputs

	Hydrogen	Ferrum	Argon	Neon	Helium	Carbon
Connectors for modules	8					
Relay 5A	2	2	18	4	2	2
Relay 10A		8			4	
Relay 16A						6
AC dimmers					3	
MOSFET (DC <48V)				8		

The following interchangeable modules are available for the Atom Hydrogen Black Edition controller: relay 5A, DC MOSFET 48V 5A, 0-10V PWM.

Important! In the I / O configuration, the GPIO port numbering of the ESP32-PICO-D4 chip or the MCP23017 port expansion chip is specified, not the numbering of the physical pins of the chips or the numbering of the inputs / outputs of the controller. The table of correspondence between GPIO ports and inputs / outputs of the controller is given in the table below:



Correspondence of inputs / outputs with GPIO ports

Input #	GPIO	Chip	Functions
1	27	ESP32	in, out, 1-wire, PWM
2	26	ESP32	in, out, 1-wire, PWM, DAC
3	33	ESP32	in, out, 1-wire, PWM, ADC
4	25	ESP32	in, out, 1-wire, PWM, DAC
5	8	MCP23017	in, out
6	9	MCP23017	in, out
7	15	MCP23017	in, out
8	14	MCP23017	in, out
9	13	MCP23017	in, out
10	12	MCP23017	in, out
11	11	MCP23017	in, out
12	10	MCP23017	in, out
Output #			
1	4	ESP32	Depending on the controller model or changeable module (see also **Number and type of outputs** table)
2	9	ESP32	
3	5	ESP32	
4	10	ESP32	
5	2	ESP32	
6	22	ESP32	
7	23	ESP32	
8	32	ESP32	
9	5	MCP23017	relay
10	6	MCP23017	relay
White LED	3	MCP23017	Status LED
Red LED	4	MCP23017	Status LED
Internal connections			
I2C SDA	21	ESP32	I2C bus
i2C SCL	19	ESP32	I2C bus
Enable Modules	18	ESP32	Changable modules activation in Atom Hydrogen

3.6. Examples of appointments:

GPIO27 ESP32 port:

```
binary_sensor:
  - platform: gpio
    name: "IN1"
    pin: 27
```

GPIO8 MCP23017 port:

```
binary_sensor:
  - platform: gpio
    name: "IN5"
    pin:
      mcp23017: mcp23017_hub
      number: 8
      mode: INPUT_PULLUP
      inverted: False
```

4. Connecting the equipment

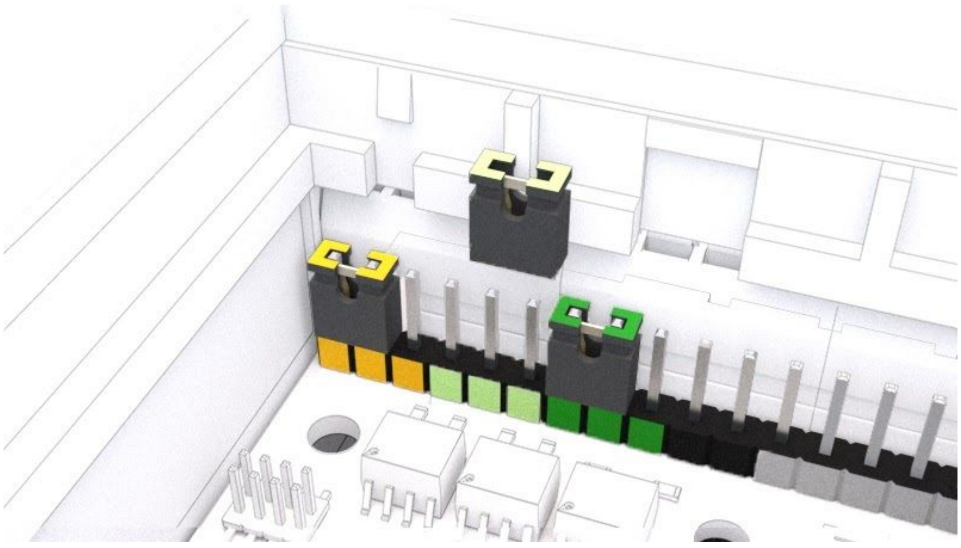
4.1. Connecting inputs.

Connect sensors, switches and other devices to the inputs of the controller. Inputs of the controller connectors are with spring-loaded contacts, which provide convenience and reliability of connection of conductors with a cross section up to 1.5 mm2.

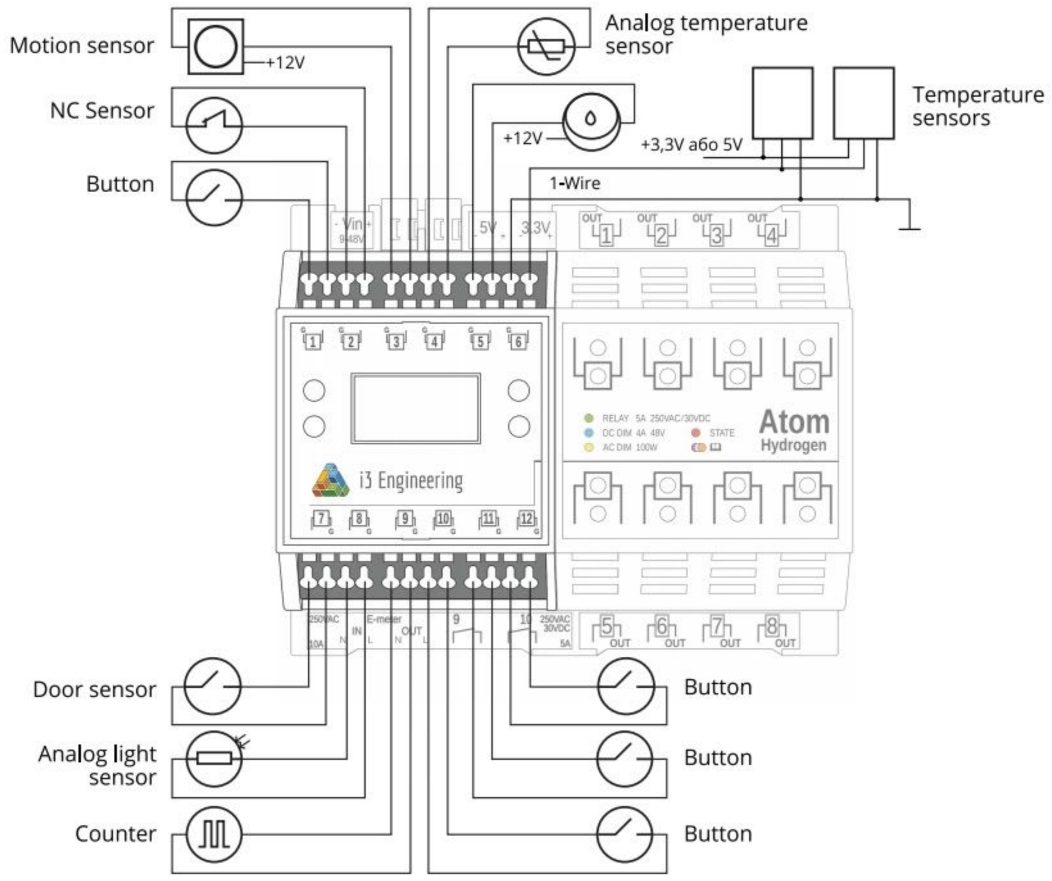
Important: the inputs are numbered and each input has a "common" output marking (marked with the letter "G"), which is connected to the "minus" of the power supply.

3.3V and 5V controller outputs are provided for power supply of sensors. If the system uses sensors or actuators with a different operating voltage, use additional power supplies or the power supply that powers the controller.

By default, all inputs with jumpers are configured in "Optocoupler" ("OP") mode. This is necessary if switches, discrete sensors or other devices are connected to the controller, which due to the specifics of operation, large length of the connection cable, other potentially dangerous factors can generate voltage values exceeding 3.3V at the input of the controller. If analog sensors or a bus with 1-wire devices are connected to the input, the input will be used in digital output mode, the inputs must be reconfigured to direct signal mode. To do this, use a flat-blade screwdriver to remove the front panel of the controller, remove the input display board, and switch the required jumpers on the input board to the “Dir” position.



Example of connecting inputs:

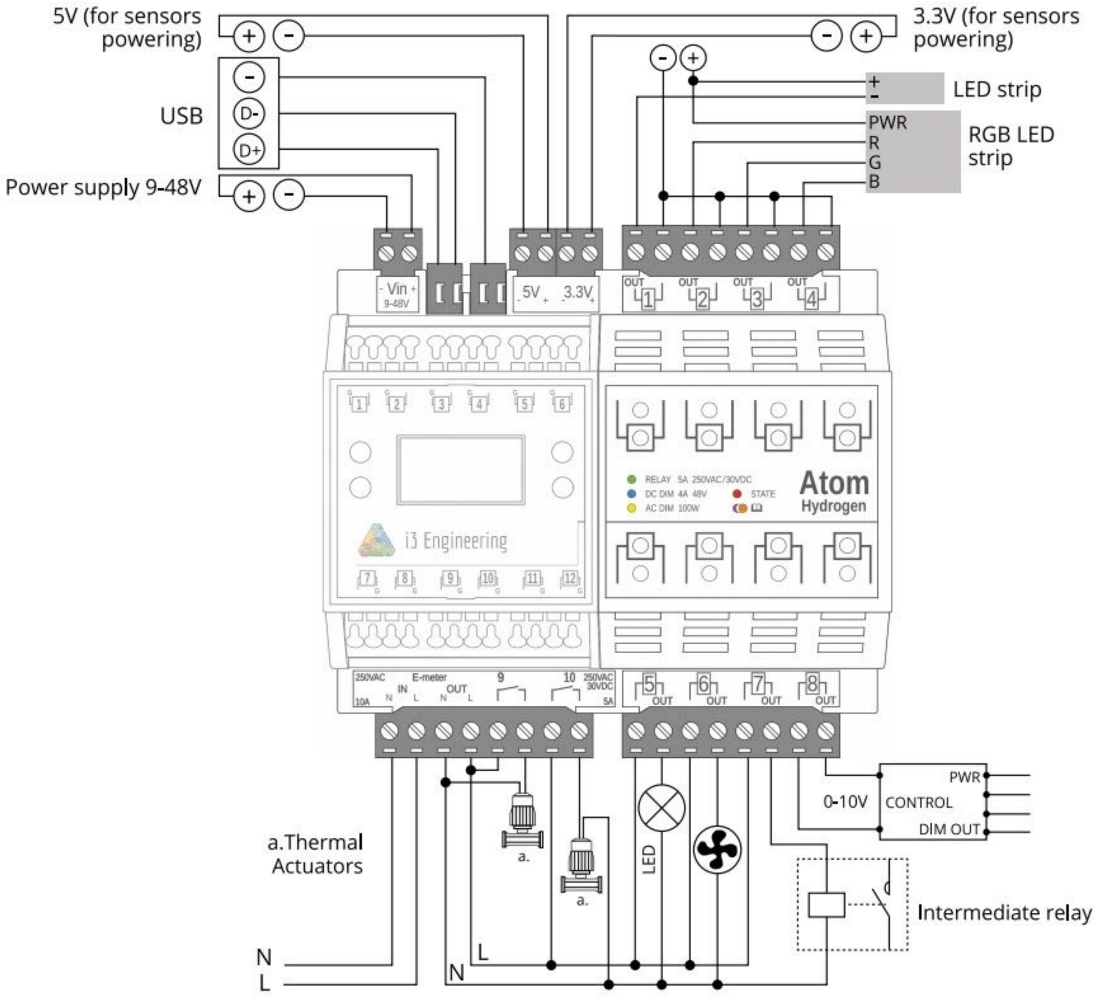


4.2. Connecting outputs

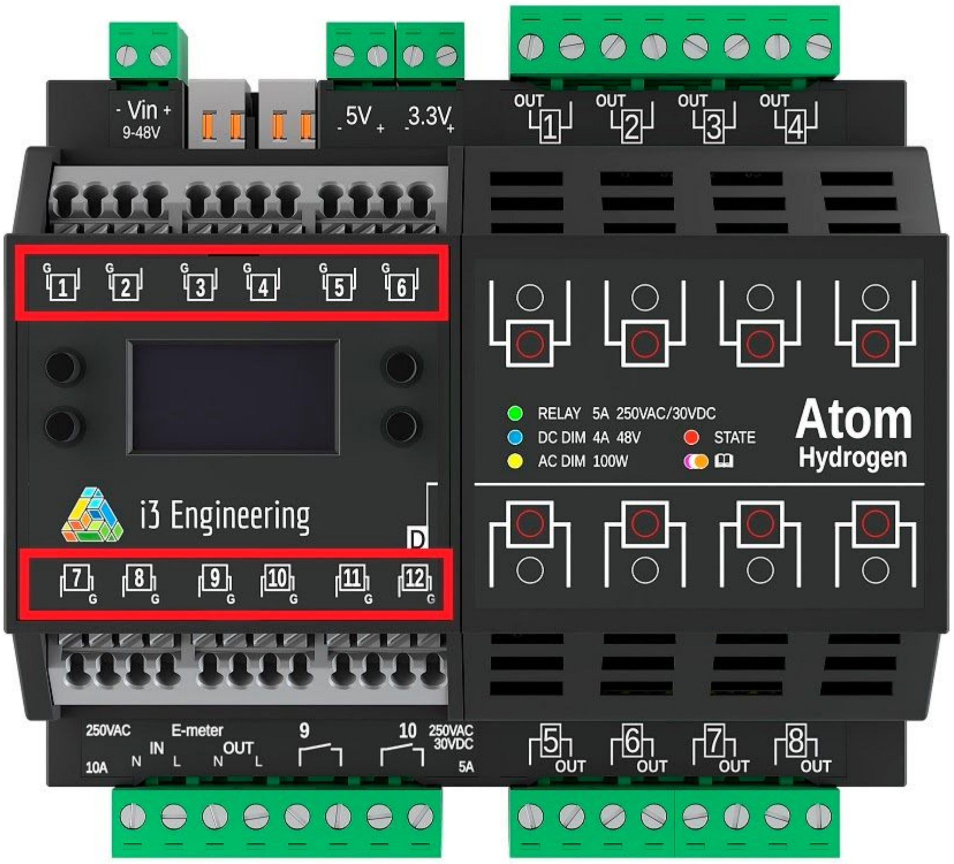
Connect the 220V phase and neutral conductors to the "L" and "N" connectors.

Connect the devices controlled by the controller to the outputs. Each output is implemented in the form of a 2-pin connector with screw terminals. One of the contacts is labeled "OUT". The load is connected according to the type of output.

Example of connecting outputs:

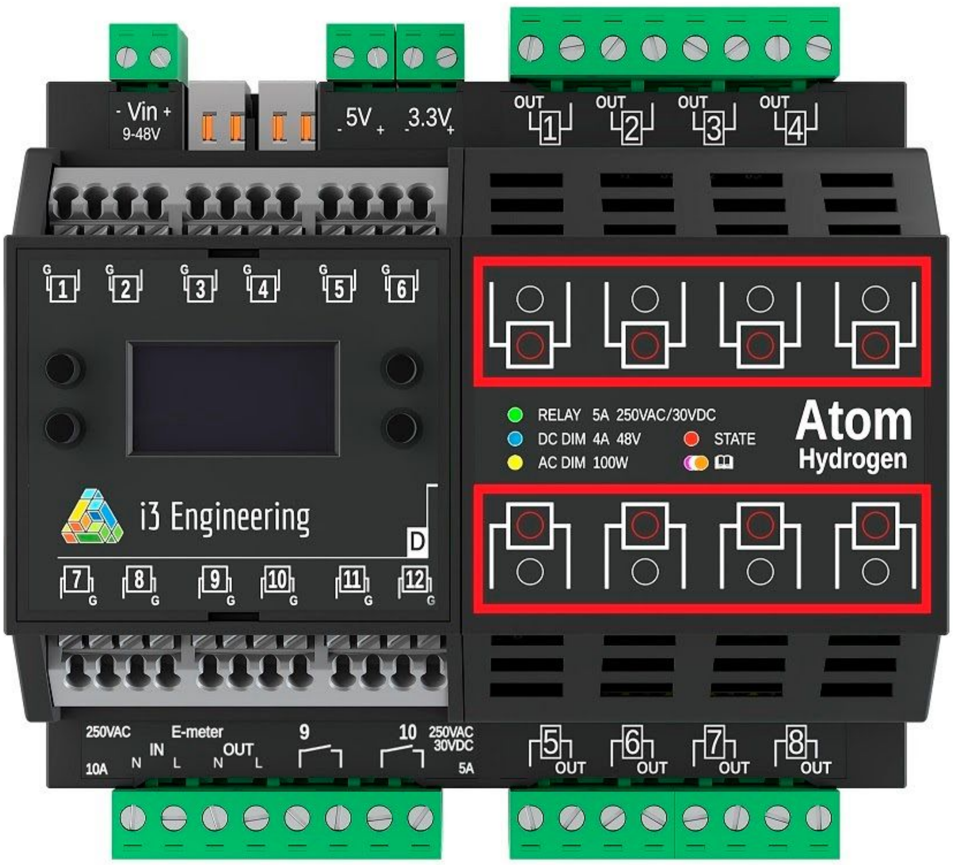


After connecting all devices, you should check their operation. It is convenient to use input and output indicators for this.



Input indicators - show the status of each input;

OLED display - for monitoring and adjustment, display of system parameters of the device, information from sensors.



Output status indicator - red LED that displays the active output status of the controller;

Output type indicators - the color of its glow corresponds to the output type:

- Yellow - Triac or AC MOSFET
- Blue - DC MOSFET
- Green - Relay

5. Automation configuration

The purpose of any automation system is, obviously, the ability to configure all devices to work according to certain rules and algorithms. The simplest option is to turn on the lamp when you press the switch. An example of such an algorithm is given below:

```
switch:
 - platform: gpio
   pin: GPIO3
   name: "Лампа у вітальні"
   id: Lamp1

binary_sensor:
  - platform: gpio
    pin: GPIO4
    name: "Вимикач у вітальні"
    on_press:
      then:
        - switch.toggle: Lamp1
```

In this case, a switch called “Living Room Switch” is connected to the GPIO3 output of the ESP card, and the “Living Room Lamp” control relay connected to GPIO4 changes its state (on or off) each time the switch is pressed.

Important! To properly configure automation, you do not need to re-describe the devices connected to the controller, as it is assumed that this procedure has already been completed. You only need the switch configuration, which looked like

```
binary_sensor:
  - platform: gpio
    pin: GPIO4
    name: "Вимикач у вітальні"
```

add code lines that are responsible for the lamp's response to a click:

```
    on_press:
      then:
        - switch.toggle: Lamp1
```

In the ESPHpme system it is possible to configure a lot of similar algorithms (automations) using different devices, different logic of their work, using groups of devices and a large number of options for relationships between devices.

Examples of automation implementation are described here:

https://esphome.io/guides/automations.html

6. Additional information

6.1. Examples of configuration files for controllers

```
esphome:
  name: atom_esphome
  platform: ESP32
  board: pico32

wifi:
  ssid: "your_wifi_ssid"
  password: "password"

  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:
    ssid: "Atom Esphome Fallback Hotspot"
    password: "Hwk4GqeoIwD5"
```

```
captive_portal:

# Enable logging
logger:

web_server:
  port: 80

# Enable Home Assistant API
api:
  password: "password"

ota:
  password: "password"

i2c:
  sda: 21
  scl: 19
  scan: True

time:
  - platform: sntp
    id: sntp_time
    servers:
      - 0.pool.ntp.org
      - 1.pool.ntp.org
      - 2.pool.ntp.org

font:
  - file: 'arial.ttf'
    id: font1
    size: 16
  - file: 'arial.ttf'
    id: font2
    size: 35
  - file: 'arial.ttf'
    id: font3
    size: 14

dallas:
  - pin: 25

sensor:
  - platform: dallas
    address: 0x603C01B6076BC628
    name: "Температура 1"
    id: temp1
    force_update: true

  - platform: dallas
    address: 0xFA3C01B607A16528
    name: "Температура 2"
    id: temp2
    force_update: true


  - platform: adc
    pin: 33
    name: "ADC33"
    attenuation: 11db
    update_interval: 1s


output:
  - platform: ac_dimmer
    id: dimmer1
    gate_pin: 2
    init_with_half_cycle: true
    method: leading pulse
    zero_cross_pin:
      number: 34
      mode: INPUT_PULLUP
      inverted: false

  - platform: ledc
    pin: 5
    id: gpio_5

# Example usage in a light
light:
  - platform: monochromatic
    output: gpio_5
    name: "LED strip"

  - platform: monochromatic
    output: dimmer1
    name: Dimmerized Light


  - platform: fastled_clockless
    chipset: WS2811
    pin: 27
    num_leds: 8
    rgb_order: RGB
    name: "FastLED WS2811 Light"
    effects:
      - addressable_rainbow:
      - addressable_rainbow:
          name: Rainbow Effect With Custom Values
          speed: 10
          width: 50
      - random:
          name: "My Fast Random Effect"
          transition_length: 4s
          update_interval: 5s
      - addressable_color_wipe:
      - addressable_color_wipe:
          name: Color Wipe Effect With Custom Values
          colors:
            - red: 100%
              green: 100%
              blue: 100%
              num_leds: 1
            - red: 0%
              green: 0%
              blue: 0%
              num_leds: 1
          add_led_interval: 100ms
```

```
        reverse: False
      - addressable_scan:
      - addressable_scan:
          name: Scan Effect With Custom Values
          move_interval: 100ms
          scan_width: 1
      - addressable_twinkle:
      - addressable_twinkle:
          name: Twinkle Effect With Custom Values
          twinkle_probability: 5%
          progress_interval: 4ms
      - addressable_fireworks:
      - addressable_fireworks:
          name: Fireworks Effect With Custom Values
          update_interval: 32ms
          spark_probability: 10%
          use_random_color: false
          fade_out_rate: 120

mcp23017:
  - id: 'mcp23017_hub'
    address: 32

binary_sensor:

#  - platform: gpio
#    name: "IN1"
#    pin: 27

  - platform: gpio
    name: "IN2"
    pin: 26

#  - platform: gpio
#    name: "IN3"
#    pin: 33

#  - platform: gpio
#    name: "IN4"
#    pin: 25

  - platform: gpio
    name: "IN5"
    pin:
      mcp23017: mcp23017_hub
      number: 8
      mode: INPUT_PULLUP
      inverted: False

  - platform: gpio
    name: "IN6"
    pin:
      mcp23017: mcp23017_hub
      number: 9
      mode: INPUT_PULLUP
      inverted: False

  - platform: gpio
    name: "IN7"
    pin:
      mcp23017: mcp23017_hub
      number: 15
      mode: INPUT_PULLUP
      inverted: False

  - platform: gpio
    name: "IN8"
    pin:
      mcp23017: mcp23017_hub
      number: 14
      mode: INPUT_PULLUP
      inverted: False

  - platform: gpio
    name: "IN9"
    pin:
      mcp23017: mcp23017_hub
      number: 13
      mode: INPUT_PULLUP
      inverted: False

  - platform: gpio
    name: "IN10"
    pin:
      mcp23017: mcp23017_hub
      number: 12
      mode: INPUT_PULLUP
      inverted: False

  - platform: gpio
    name: "IN11"
    pin:
      mcp23017: mcp23017_hub
      number: 11
      mode: INPUT_PULLUP
      inverted: False

  - platform: gpio
    name: "IN12"
    pin:
      mcp23017: mcp23017_hub
      number: 10
      mode: INPUT_PULLUP
      inverted: False

switch:
  - platform: gpio
    name: "White LED"
    pin:
      mcp23017: mcp23017_hub
      number: 3
      mode: OUTPUT
      inverted: False

  - platform: gpio
    name: "Red LED"
    pin:
      mcp23017: mcp23017_hub
      number: 4
      mode: OUTPUT
      inverted: False

  - platform: gpio
    pin: 18
    name: "Enable modules"

  - platform: gpio
    pin: 4
    name: "OUT1"

  - platform: gpio
    pin: 9
    name: "OUT2"

#  - platform: gpio
#    pin: 5
#    name: "OUT3"

  - platform: gpio
    pin: 10
    name: "OUT4"

#  - platform: gpio
#    pin: 2
#    name: "OUT5"

  - platform: gpio
    pin: 22
    name: "OUT6"

  - platform: gpio
    pin: 23
    name: "OUT7"

  - platform: gpio
    pin: 32
    name: "OUT8"

  - platform: gpio
    name: "OUT9"
    pin:
      mcp23017: mcp23017_hub
      number: 5
      mode: OUTPUT
      inverted: False

  - platform: gpio
    name: "OUT10"
    pin:
      mcp23017: mcp23017_hub
      number: 6
      mode: OUTPUT
      inverted: False

display:
  - platform: ssd1306_i2c
    model: "SSD1306 128x64"
```

```
address: 0x3C
update_interval: 3s
lambda: |-
  it.printf(64, 0, id(font1), TextAlign::TOP_CENTER, "i3 Engineering");
  it.strftime(0, 55, id(font2), TextAlign::BASELINE_LEFT ,"%H:%M", id(sntp_time).now());
  if (id(temp1).has_state()) {
    it.printf(127, 23, id(font3), TextAlign::TOP_RIGHT ,"%.1f°", id(temp1).state);
    }
  if (id(temp2).has_state()) {
    it.printf(127, 60, id(font3), TextAlign::BASELINE_RIGHT ,"%.1f°", id(temp2).state);
    }
```

6.2. Expansion of functionality

At present, most of the functionality of the Atom series controllers is implemented for the ESPHome system.

Planning to implement the following functionality:

- Using Triac and AC MOSFET interchangeable modules
- Control of current consumption at the outputs equipped with Hall sensors,
- Activate control of the controller using the buttons on the front panel of the controller
- Use of controllers in similar automation systems that support ESP32 - Tasmota and 1mSmart.

6.3. Examples of implementations of various functions

https://esphome.io/guides/diy.html