Logicbus

¡ Tu Sitio de Automatización !

Tutorials Manual

Controllino

Contenido

1-WIRE TEMPERATURE SENSOR (DS18B20)

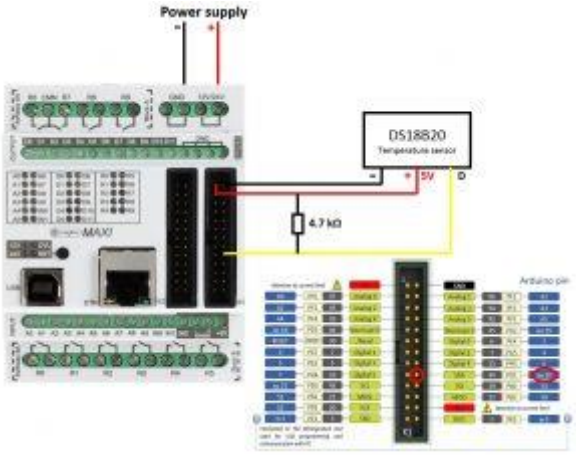
This example shows you how to connect 1-wire temperature sensor to the Controllino device and read the temperature or address of the sensor on the bus. **DS18B20** is 1-Wire digital temperature sensor from Maxim IC. Reports degrees in Celsius or Fahrenheit with 9 to 12-bit precision, from –55°C to +125°C (–67°F to +257°F)±0.5°C. Each sensor has a unique 64-Bit Serial number etched into it – allows for a huge number of sensors to be used on one data bus.

IMPORTANT INFORMATION!
Please, select proper target board in **Tools->Board->Controllino MINI/MAXI/MEGA** before Upload to your CONTROLLINO. (Please, refer to https://github.com/CONTROLLINO-PLC/CONTROLLINO_Library if you do not see the CONTROLLINOs in the Arduino IDE menu **Tools->Board**.)

Hardware Required

- Controllino MINI/MAXI/MEGA
- 12/24V DC Power supply
- DS18B20 – 1-wire temperature sensor
- 4.7 kΩ resistor

Circuit



When connecting your sensor you can choose the pin that you want to connect it to.

In this case the data pin of the sensor is connected to the SDA communication pin, but it can also be connected on any digital, relay output (5V pin), or communication pin. The best way is to choose the pin that is free and not used.

Note*

Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

Code

To get the readings from the temperature sensor you need to instal OneWire library.

Open **Sketch->Include Library->Manage Libraries…**

In this example we use OneWire 2.3.3 library.

After installing the library you have to open **File->Examples->OneWire->DS18x20_Temperature** and run the example. The only thing you have to change in the code is the data pin that you are using in line:

***OneWire ds(20);**

```
#include <OneWire.h>
```

```
// OneWire DS18S20, DS18B20, DS1822 Temperature Example
//
// https://www.pjrc.com/teensy/td_libs_OneWire.html
//
// The DallasTemperature library can do all this work for you!
// https://milesburton.com/Dallas_Temperature_Control_Library

OneWire ds(20); // on pin 20 (a 4.7K resistor is necessary)

void setup(void) {
 Serial.begin(9600);
}

void loop(void) {
 byte i;
 byte present = 0;
 byte type_s;
 byte data[12];
 byte addr[8];
 float celsius, fahrenheit;

 if ( !ds.search(addr)) {
 Serial.println("No more addresses.");
 Serial.println();
 ds.reset_search();
 delay(250);
 return;
 }

 Serial.print("ROM =");
 for( i = 0; i < 8; i++) {
 Serial.write(' ');
 Serial.print(addr[i], HEX);
```

```
}

if (OneWire::crc8(addr, 7) != addr[7]) {
Serial.println("CRC is not valid!");
return;
}
Serial.println();

// the first ROM byte indicates which chip
switch (addr[0]) {
case 0x10:
Serial.println(" Chip = DS18S20"); // or old DS1820
type_s = 1;
break;
case 0x28:
Serial.println(" Chip = DS18B20");
type_s = 0;
break;
case 0x22:
Serial.println(" Chip = DS1822");
type_s = 0;
break;
default:
Serial.println("Device is not a DS18x20 family device.");
return;
}

ds.reset();
ds.select(addr);
ds.write(0x44, 1); // start conversion, with parasite power on at the end

delay(1000); // maybe 750ms is enough, maybe not
// we might do a ds.depower() here, but the reset will take care of it.
```

```
present = ds.reset();
ds.select(addr);
ds.write(0xBE); // Read Scratchpad

Serial.print(" Data = ");
Serial.print(present, HEX);
Serial.print(" ");
for ( i = 0; i < 9; i++) { // we need 9 bytes
data[i] = ds.read();
Serial.print(data[i], HEX);
Serial.print(" ");
}
Serial.print(" CRC=");
Serial.print(OneWire::crc8(data, 8), HEX);
Serial.println();

// Convert the data to actual temperature
// because the result is a 16 bit signed integer, it should
// be stored to an "int16_t" type, which is always 16 bits
// even when compiled on a 32 bit processor.
int16_t raw = (data[1] << 8) | data[0];
if (type_s) {
raw = raw << 3; // 9 bit resolution default
if (data[7] == 0x10) {
// "count remain" gives full 12 bit resolution
raw = (raw & 0xFFF0) + 12 - data[6];
}
} else {
byte cfg = (data[4] & 0x60);
// at lower res, the low bits are undefined, so let's zero them
if (cfg == 0x00) raw = raw & ~7; // 9 bit resolution, 93.75 ms
else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
```

```
 else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
 //// default is 12 bit resolution, 750 ms conversion time
 }
 celsius = (float)raw / 16.0;
 fahrenheit = celsius * 1.8 + 32.0;
 Serial.print(" Temperature = ");
 Serial.print(celsius);
 Serial.print(" Celsius, ");
 Serial.print(fahrenheit);
 Serial.println(" Fahrenheit");
}
```

16×2 LCD (MAXI)

The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

This example sketch prints “Hello World!” to the LCD and shows the time in seconds since the Arduino was reset.

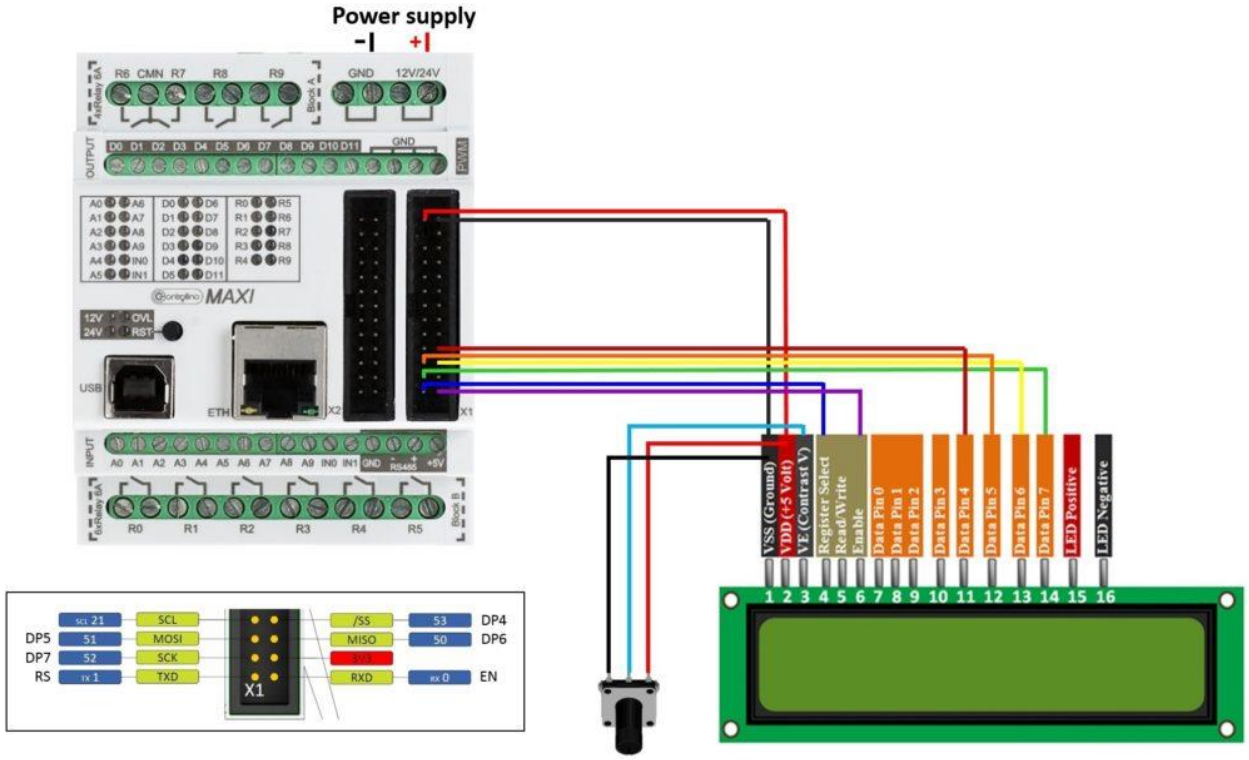
To see more about 16×2 LCD and related pins please visit https://www.arduino.cc/en/Tutorial/HelloWorld?from=Tutorial.LiquidCrystal

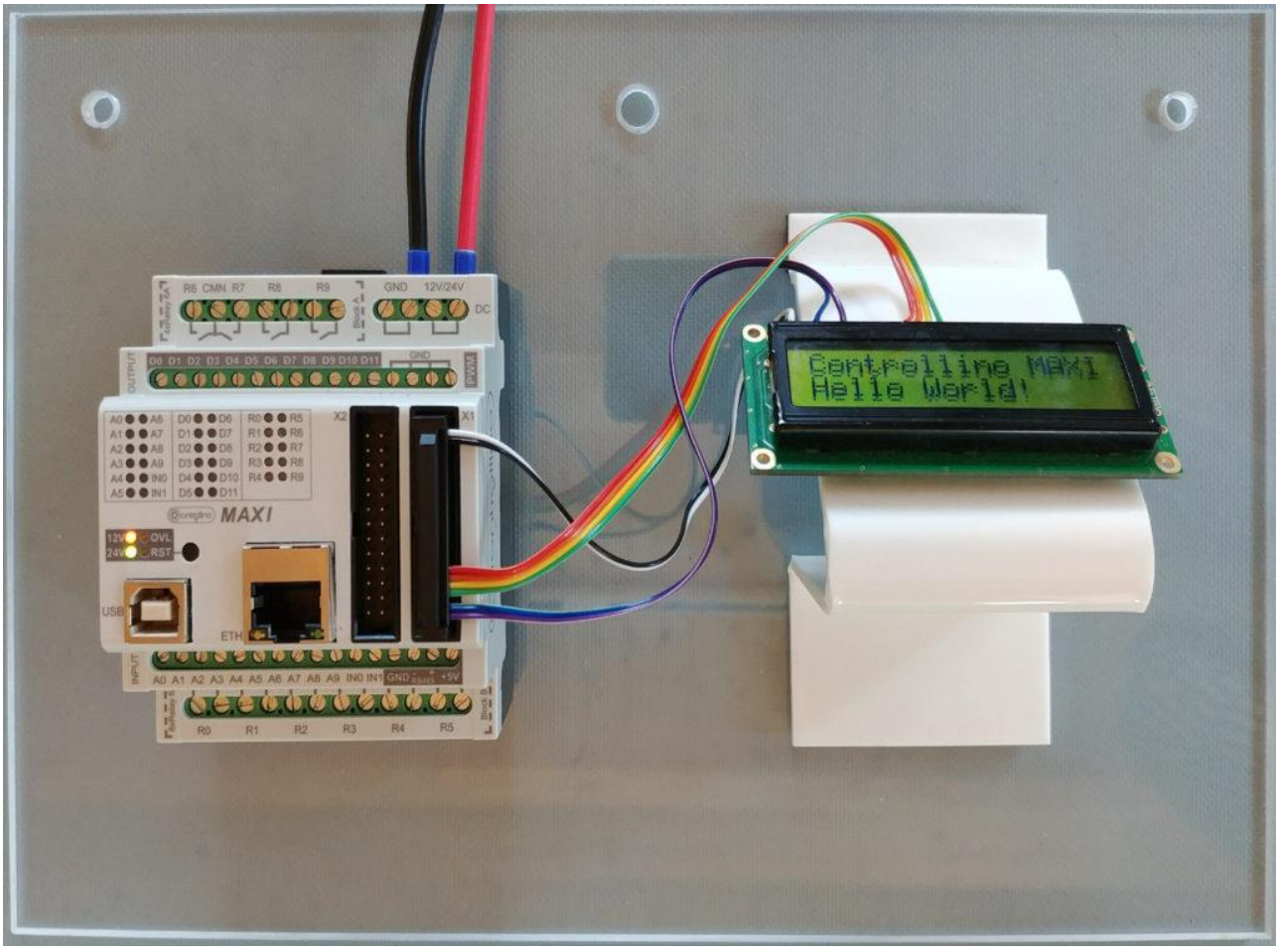
Hardware Required

- Controllino MINI/MAXI/MEGA
- 12/24V DC Power supply
- 16×2 LCD
- 10k ohm potentiometer
- hook-up wires

Circuit

Instead of the pins that we use in this example you can use every other communication, output and input pin. We suggest to use first communication and output pins because of internal resistors and then input pins.
In this example we don’t need UART and SPI communication and we are using these pins to have all input and output pins free.


Power supply
-| +|
R6 CMN R7 R8 R9 GND 12V/24V
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 GND
MAXI
12V 24V OVL RST
USB
ETH
X1 X2
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 IN0 IN1 GND RS485 +5V
R0 R1 R2 R3 R4 R5
VSS (Ground)
VDD (+5 Volt)
VE (Contrast V)
Register Select
Read/Write
Enable
Data Pin 0
Data Pin 1
Data Pin 2
Data Pin 3
Data Pin 4
Data Pin 5
Data Pin 6
Data Pin 7
LED Positive
LED Negative
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
DP5
DP7
RS
SCL
MOSI
SCK
TXD
/SS
MISO
RXD
S3
S0
DP4
DP6
EN
X1


Centrelline MAXI
Hello World!
MAXI
USB
ETH

Note*

Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

Code

Controllino MAXI

```
/*
 LiquidCrystal Library - Hello World

 Demonstrates the use a 16x2 LCD display. The LiquidCrystal
 library works with all LCD displays that are compatible with the
 Hitachi HD44780 driver. There are many of them out there, and you
 can usually tell them by the 16-pin interface.

 This sketch prints "Hello World!" to the LCD

 This example code is in the public domain.

 https://www.arduino.cc/en/Tutorial/LiquidCrystal
 */

// include the library code:
#include <LiquidCrystal.h>
//#include <Controllino.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(1, 0, 53, 51, 50, 52);

void setup() {

 // set up the LCD's number of columns and rows:
 lcd.begin(16, 2);
```

```
 // Print a message to the LCD.
 lcd.print("Controllino MAXI");

}

void loop() {

 // set the cursor to column 0, line 1
 // (note: line 1 is the second row, since counting begins with 0):

 lcd.setCursor(0, 1);

 // print the number of seconds since reset:
 lcd.print("Hello World!");

 delay(5);

}
```

ANALOG READ SERIAL

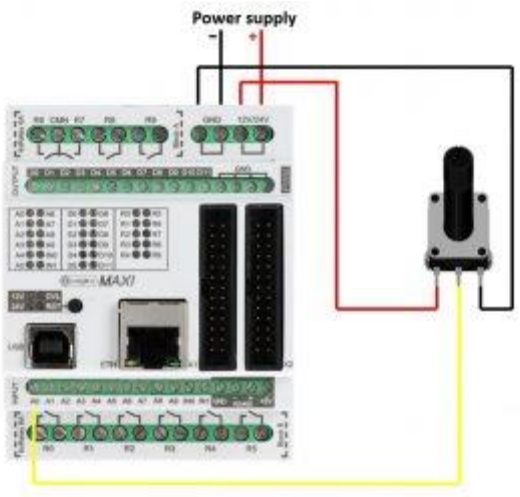
This example shows you how to read analog input from your Controllino device over serial communication. In order to do this, you have to connect potentiometer on one analog input and establish serial communication between your Controllino board and your computer running the Arduino Software(IDE).

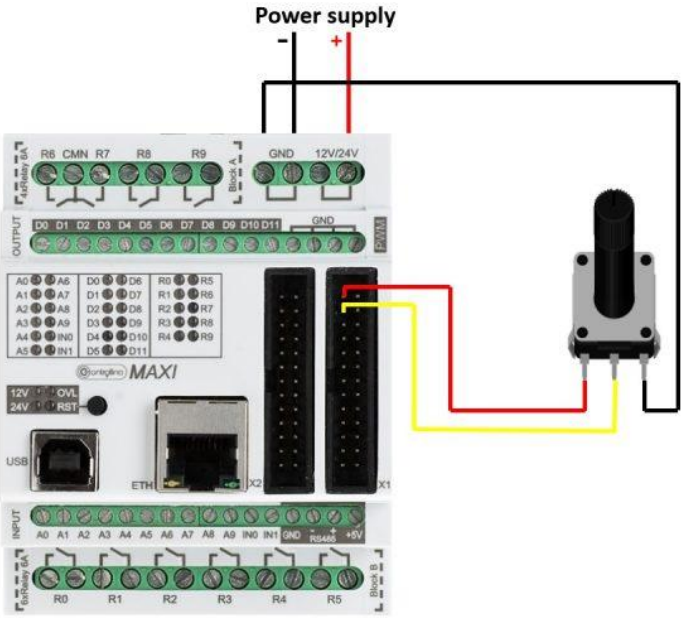
IMPORTANT INFORMATION!
Please, select proper target board in **Tools->Board->Controllino MINI/MAXI/MEGA** before Upload to your CONTROLLINO. (Please, refer to https://github.com/CONTROLLINO-PLC/CONTROLLINO_Library if you do not see the CONTROLLINOs in the Arduino IDE menu **Tools->Board**.)

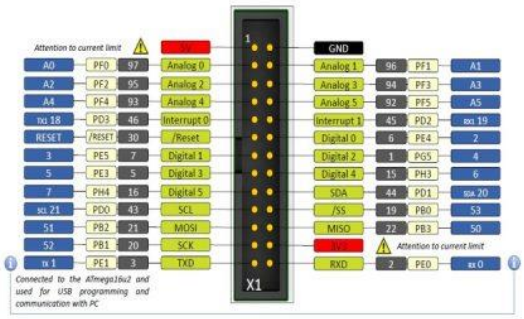
HARDWARE REQUIRED

- Controllino MINI/MAXI/MEGA
- 10k ohm potentiometer

CIRCUIT








```
#include <Controllino.h>

/* Usage of CONTROLLINO library allows you to use CONTROLLINO_xx aliases in your
sketch. */


// the setup routine runs once when you press reset:

void setup() {

    // initialize necessary pin as input pin

    pinMode(CONTROLLINO_A0, INPUT);


    // initialize serial communication at 9600 bits per second:

    Serial.begin(9600);

}


// the loop routine runs over and over again forever:

void loop() {

    // read the input on analog pin 0:

    int sensorValue = analogRead(CONTROLLINO_A0);

    // print out the value you read:
```

```
    Serial.println(sensorValue);
    delay(1); // delay in between reads for stability
}
```

Note*

Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

CONTROLLINO HMI WITH MODBUS

Controllino HMIs are industrial HMI panels with high-resolution touch-screens and modern design. The panels combine IP65 corrosion resistant plastic housing with the full version of the iX software, providing an advanced HMI solution for every kind of applications.
Combine HMI with Controllino and you can create simple as well as the sophisticated applications in very short time.

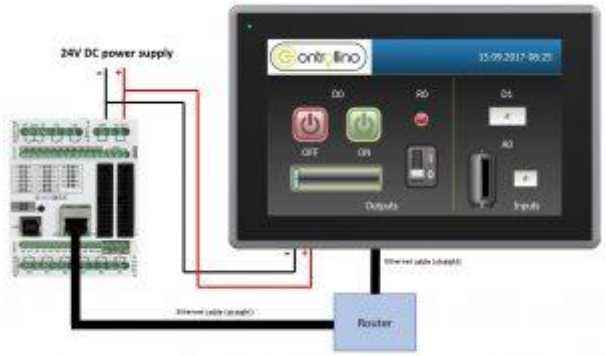
IMPORTANT: Please, select proper target board in **Tools->Board->Controllino MINI/MAXI/MEGA** before Upload to your CONTROLLINO. (Please, refer to https://github.com/CONTROLLINO-PLC/CONTROLLINO_Library if you do not see the CONTROLLINOs in the Arduino IDE menu **Tools->Board**.)

MODBUS TCP/IP

Hardware Required

- Controllino MAXI/MEGA
- 24V DC Power supply
- Controllino HMI
- 2x Ethernet cable
- Router

Circuit

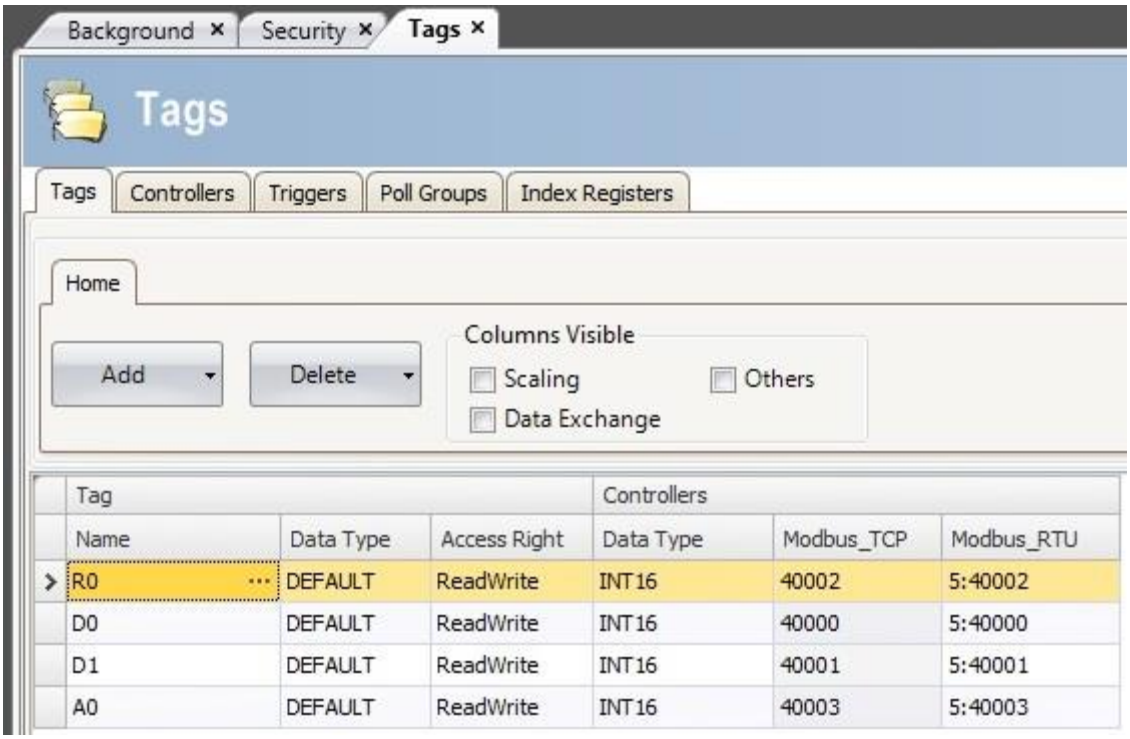


Note*
Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

HMI setup

To set up the Controllino HMI for communication and application we use the iX Software. iX Software is a revolutionary software that features drivers to communicate with your automation equipment, enhanced HMI functionality, state-of-the-art graphics, an intuitive design environment and a truly open platform for today's automation market. iX Developer is a licensed application that runs on a Windows PC and is used to program iX operator panels. Here we are going to show some steps on how to set it up properly. If you want to check more information and the iX user manual click here.

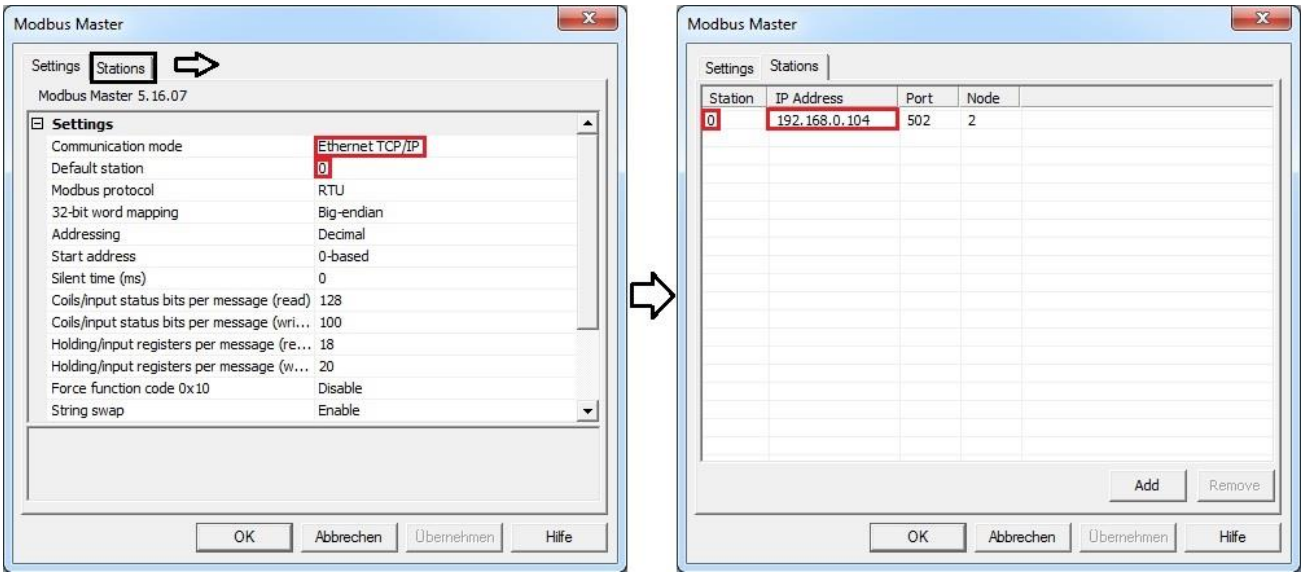
Download the Controllino HMI – TCP/IP example and run it in the iX software. From the left side of the main screen in the iX example there is the *Tags* button. Open the *Tags* to set your application addresses. You can see example of this on the next picture:



For more information about *Tags* and addressing of them navigate to *Controllers/Settings* and click on the help button of the *Modbus Master* controller.

After that go to the *Controllers tab* to choose the right controller for communication. When we are using the Controllino device we want to choose *MODICON -> Modbus Master* controller. In this example it is already done and they are named *Modbus_TCP* and *Modbus_RTU*.

Press on the *Modbus_TCP* controller, activate it and click on the *Settings* button to do the folowing:



Set communication mode to the Ethernet TCP/IP and the IP address of the station to your Controllino IP address. To get your Controllino IP address and to ensure that the connections work, run the **ArduinoIDE->File->Examples->Ethernet->DhcpAddressPrinter** example, upload it and open Serial Monitor. If everything works, you should get your IP address on the screen.

Code

Before uploading the program to the Controllino you have to install the Modbus TCP library. You can download it here. To install it copy the Modbus folder to your **Arduino->libraries** folder.

After getting the IP address of the Controllino the new program has to be uploaded to the device. The only thing that needs to be changed in the next code is this line:

***uint8_t ip[] = {192, 168, 0, 104}; // change this to the IP address of your Controllino device**

```
#include<Controllino.h>
#include <SPI.h>
#include <Ethernet.h>

#include "Mudbus.h"
```

```
Mudbus Mb;


int D0 = CONTROLLINO_D0;
int D1 = CONTROLLINO_D1;
int R0 = CONTROLLINO_R0;
int A0_ = CONTROLLINO_A0;



//Function codes 1(read coils), 3(read registers), 5(write coil), 6(write regist
er)
//signed int Mb.R[0 to 125] and bool Mb.C[0 to 128] MB_N_R MB_N_C
//Port 502 (defined in Mudbus.h) MB_PORT


void setup()
{
 uint8_t mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
 uint8_t ip[] = { 192, 168, 0, 103 };
 uint8_t gateway[] = { 192, 168, 0, 254 };
 uint8_t subnet[] = { 255, 255, 255, 0 };
 Ethernet.begin(mac, ip, gateway, subnet);

 delay(5000);
 Serial.begin(9600);
 Serial.println("Started");

 pinMode(D0, OUTPUT);
 pinMode(D1, INPUT);
 pinMode(R0, OUTPUT);
 pinMode(A0_, INPUT);
}

void loop()
{
```

```
 Mb.Run();

 analogWrite(D0, Mb.R[0]);
 digitalWrite(R0, Mb.R[2]);
 Mb.R[1] = digitalRead(D1);
 Mb.R[3] = analogRead(A0_);

}
```

MODBUS RTU VIA RS485

Hardware Required

- Controllino MAXI/MEGA
- 24V DC Power supply
- Controllino HMI
- 3 wires or serial port adapter (male)

Circuit

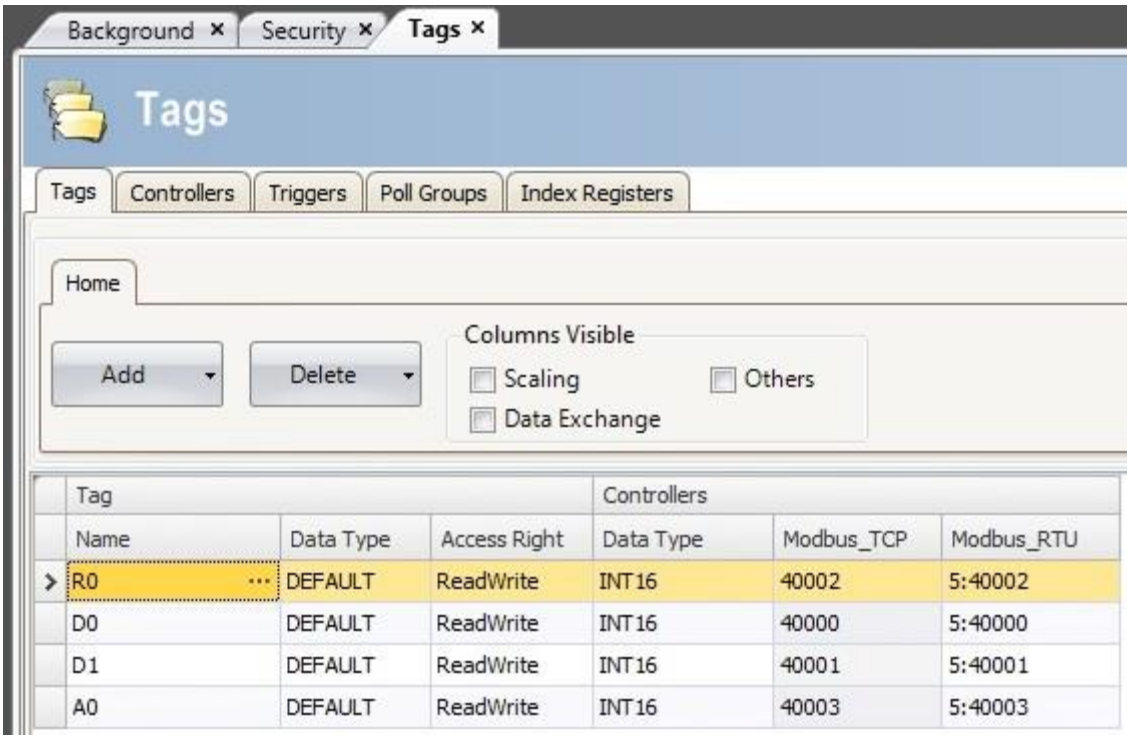
Note*

Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

HMI setup

To set up the Controllino HMI for communication and application we use the iX Software. iX Software is a revolutionary software that features drivers to communicate with your automation equipment, enhanced HMI functionality, state-of-the-art graphics, an intuitive design environment and a truly open platform for today's automation market. iX Developer is a licensed application that runs on a Windows PC and is used to program iX operator panels.
Here we are going to show some steps on how to set it up properly. If you want to check more information and the iX user manual click here.

Download the Controllino HMI – RTU example and run it in the iX software. From the left side of the main screen in the iX example there is the *Tags* button. Open the *Tags* to set your application addresses. You can see example of this on the next picture:

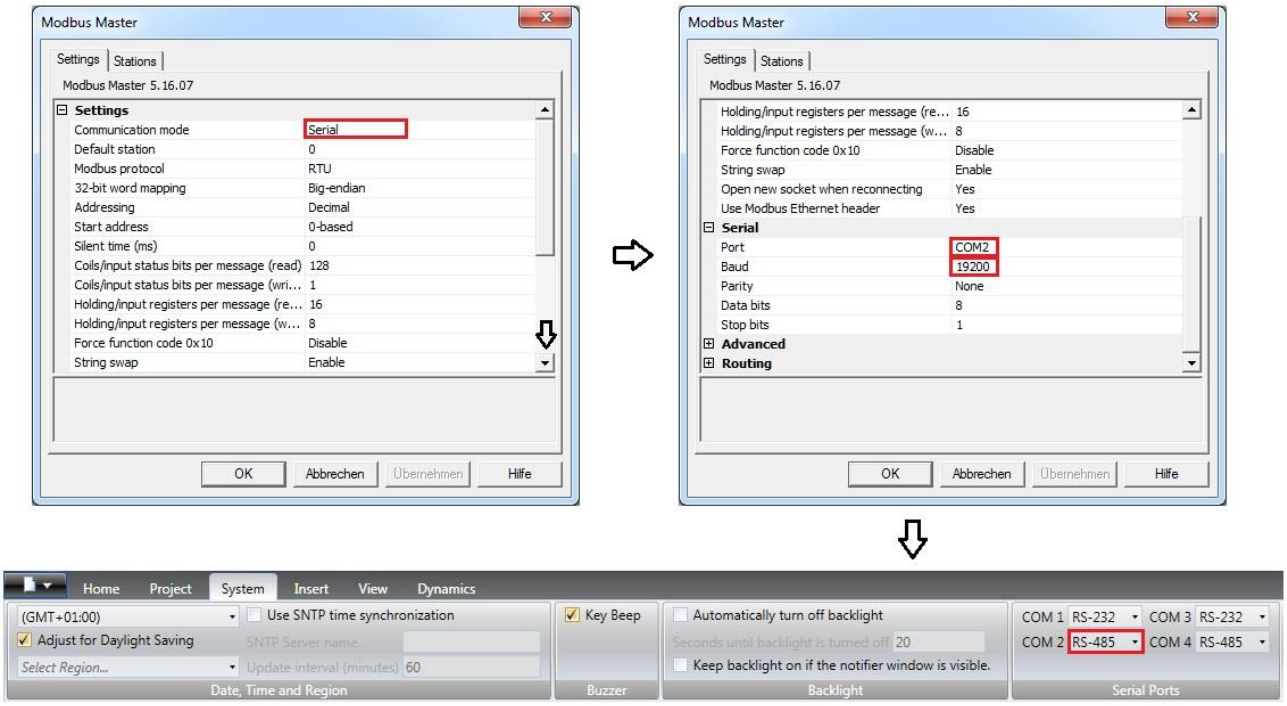


For more information about *Tags* and addressing of them navigate to *Controllers/Settings* and click on the help button of the *Modbus Master* controller.

Notice here that the Modbus_RTU tags are starting with “5:”. For communication with other stations than the default station, the station number is given as a prefix to the device. This is stated either as a fixed number or as an index register between I1 and I8. The station number is referring to a specific unit id. (5:40001 addresses holding register 40001 in station 5.) In our case the Controllino is slave device on station 5. We can always define his station in the code that we upload.

After that go to the *Controllers tab* to choose the right controller for communication. When we are using the Controllino device we want to choose *MODICON -> Modbus Master* controller. In this example it is already done and they are named *Modbus_TCP* and *Modbus_RTU*.

Press on the *Modbus_RTU* controller, activate it and click on the *Settings* button to do the folowing steps:



Set communication mode to the Serial and the COM2 port of your Controllino HMI. Don't forget to match the baud rate of Controllino and HMI. Choose the COM2 port to be RS-485.

Code

The Controllino HMI is communicating to the Controllino slave device on the station 5 so we have to set it in the next line:

***define SlaveModbusAdd 5**

```
#include <Controllino.h> /* Usage of CONTROLLINO library allows you to use CONTR
OLLINO_xx aliases in your sketch. */

#include "ModbusRtu.h" /* Usage of ModBusRtu library allows you to implement the
Modbus RTU protocol in your sketch. */

/*

 CONTROLLINO - Modbus RTU protocol Slave example for MAXI and MEGA, Version 01.0
0


 The sketch is relevant only for CONTROLLINO variants MAXI and MEGA (because of
necessity of RS485 interface)!
```

```
 This sketch is intended as an example of the communication between devices via
RS485 with utilization the

 ModbusRTU protocol.

 In this example the CONTROLLINO is used as the Modbus slave!

 For more information about the Modbus protocol visit the website: https://modbu
s.org/


 Modbus master device can read Modbus 16bit registers (provided by the slave):

 0 - analog CONTROLLINO_A0 value (0 - 1024)

 1 - digital CONTROLLINO_D0 value (0/1)

 2 - Modbus messages received

 3 - Modbus messages transmitted


 Modbus master device can write Modbus 16bit registers:

 4 - relay CONTROLLINO_R0 (0/1)

 5 - relay CONTROLLINO_R1 (0/1)

 6 - relay CONTROLLINO_R2 (0/1)

 7 - relay CONTROLLINO_R3 (0/1)


 To easily evaluate this example you need a second CONTROLLINO as Modbus master
running DemoModbusRTUMaster

 example sketch.

 Please note that both CONTROLLINOs need 12/24V external supply and you need to
interconnect GND, -, + signals

 of RS485 screw terminal.


 Modbus Master-Slave library for Arduino (ModbusRtu.h) was taken from the websit
e:

 https://github.com/smarmengol/Modbus-Master-Slave-for-Arduino

 It was necessary to modify setting of the PORTJ for pins DE and RE control. The
se pins are located at the

 PORJ and on the pins PIN6(DE) and PIN5(RE).


 IMPORTANT INFORMATION!
```

```
 Please, select proper target board in Tools->Board->Controllino MAXI/MEGA befor
e Upload to your CONTROLLINO.

 (Please, refer to https://github.com/CONTROLLINO-PLC/CONTROLLINO_Library if you
do not see the CONTROLLINOs in

 the Arduino IDE menu Tools->Board.)


 Created 30 March 2017

 by David



 (Check https://github.com/CONTROLLINO-PLC/CONTROLLINO_Library for the latest CO
NTROLLINO related software stuff.)

*/

int D0 = CONTROLLINO_D0;

int D1 = CONTROLLINO_D1;

int R0 = CONTROLLINO_R0;

int A0_ = CONTROLLINO_A0;


// This MACRO defines Modbus slave address.

// For any Modbus slave devices are reserved addresses in the range from 1 to 24
7.

// Important note only address 0 is reserved for a Modbus master device!


#define SlaveModbusAdd 5


// This MACRO defines number of the comport that is used for RS 485 interface.

// For MAXI and MEGA RS485 is reserved UART Serial3.


#define RS485Serial 3


// The object ControllinoModbuSlave of the class Modbus is initialized with thre
e parameters.

// The first parametr specifies the address of the Modbus slave device.

// The second parameter specifies type of the interface used for communication b
etween devices - in this sketch

// is used RS485.
```

```
// The third parameter can be any number. During the initialization of the objec
t this parameter has no effect.


Modbus ControllinoModbusSlave(SlaveModbusAdd, RS485Serial, 0);


// This uint16 array specified internal registers in the Modbus slave device.

// Each Modbus device has particular internal registers that are available for t
he Modbus master.

// In this example sketch internal registers are defined as follows:

// (ModbusSlaveRegisters 0 - 3 read only and ModbusSlaveRegisters 4 - 7 write on
ly from the Master perspective):

// ModbusSlaveRegisters[0] - Read an analog value from the CONTROLLINO_A0 - retu
rns value in the range from 0 to 1023.

// ModbusSlaveRegisters[1] - Read an digital value from the CONTROLLINO_D0 - ret
urns only the value 0 or 1.

// ModbusSlaveRegisters[2] - Read the number of incoming messages - Communicatio
n diagnostic.

// ModbusSlaveRegisters[3] - Read the number of number of outcoming messages - C
ommunication diagnostic.

// ModbusSlaveRegisters[4] - Sets the Relay output CONTROLLINO_R0 - only the val
ue 0 or 1 is accepted.

// ModbusSlaveRegisters[5] - Sets the Relay output CONTROLLINO_R1 - only the val
ue 0 or 1 is accepted.

// ModbusSlaveRegisters[6] - Sets the Relay output CONTROLLINO_R2 - only the val
ue 0 or 1 is accepted.

// ModbusSlaveRegisters[7] - Sets the Relay output CONTROLLINO_R3 - only the val
ue 0 or 1 is accepted.


uint16_t ModbusSlaveRegisters[8];


// The setup function runs once when you press reset (CONTROLLINO RST button) or
connect the CONTROLLINO to the PC

// In the setup function is carried out port setting and initialization of commu
nication of the Modbus slave protocol.


void setup()

{

 delay(5000);
```

```
 Serial.begin(9600);
 Serial.println("Started");


 pinMode(D0, OUTPUT);
 pinMode(D1, INPUT);
 pinMode(R0, OUTPUT);
 pinMode(A0_, INPUT);


 ControllinoModbusSlave.begin( 19200 ); // Start the communication over the Modb
usRTU protocol. Baund rate is set at 19200.
 Serial.begin(9600);


}


// The loop function runs over and over again forever
void loop()
{
 // This instance of the class Modbus checks if there are any incoming data.
 // If any frame was received. This instance checks if a received frame is Ok.
 // If the received frame is Ok the instance poll writes or reads corresponding
values to the internal registers
 // (ModbusSlaveRegisters).
 // Main parameters of the instance poll are address of the internal registers a
nd number of internal registers.


 ControllinoModbusSlave.poll(ModbusSlaveRegisters, 8);


 // While are not received or sent any data, the Modbus slave device periodicall
y reads the values of analog and
 // digital outputs.
 // Also it updates the other values of registers.


 analogWrite(D0, ModbusSlaveRegisters[0]);
 digitalWrite(R0, ModbusSlaveRegisters[2]);
 ModbusSlaveRegisters[1] = digitalRead(D1);
```

```
 ModbusSlaveRegisters[3] = analogRead(A0_);


}
```

CURRENT OUTPUTS (MAXI AUTOMATION)

The MAXI Automation is the version of Controllino MAXI specifically tailored for the needs of automation specialists! It is the perfect compromise between compact

size and big input and output number. The core competence is its flexibility. In this example the usage of Controllino real analog (0-10V) outputs will be shown. Controllino Maxi Automation has special:

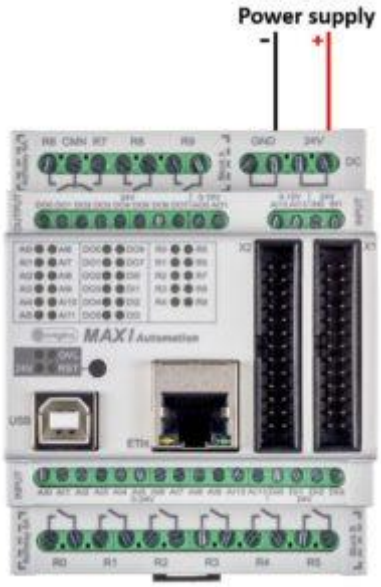
- **2x Analog Inputs 0-10V**
- **2x Analog Outputs – 0-10V (0-20mA)**

If you need two current (0-20mA), and not voltage (0-10V) outputs for your project, you can change them into current outputs by simply removing two 0 Ω resistors on Maxi Automation controll board. This process will be shown in next steps.

Hardware Required

- Controllino MAXI Automation
- 24V DC Power supply
- Soldering iron
- Tweezers

Circuit



Note*

Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

Code

Controllino MAXI Automation

To set your real analog outputs for your needs you can use the builtin example from Controllino Library, or you can copy the example from below.

```
#include <Controllino.h> /* Usage of CONTROLLINO library allows you to use CONTR
OLLINO_xx aliases in your sketch. */


// the setup function runs once when you press reset (CONTROLLINO RST button) or
connect the CONTROLLINO to the PC
void setup() {
 // initialize all used digital output pins as outputs
 pinMode(CONTROLLINO_AO0, OUTPUT);
 pinMode(CONTROLLINO_AO1, OUTPUT);
}


// the loop function runs over and over again forever
```

```
void loop() {
 int analogOut0 = 127;                      // 0 - 255 to be set (0 - 10 000 mV, or 0 - 20 000 uA)
 int analogOut1 = 255;                      // 0 - 255 to be set (0 - 10 000 mV, or 0 - 20 000 uA)
 analogWrite(CONTROLLINO_AO0, analogOut0); // set the analog output 0 to 5V or 10mA
 analogWrite(CONTROLLINO_AO1, analogOut1); // set the analog output 1 to 10V or 20mA
}
```

If you are not able to compile the sketch, choose the Controllino MAXI Automation board!

To make the outputs and relays work, CONTROLLINO pins have to be set up as OUTPUTs!

*pinMode(CONTROLLINO_xx, OUTPUT);

Steps

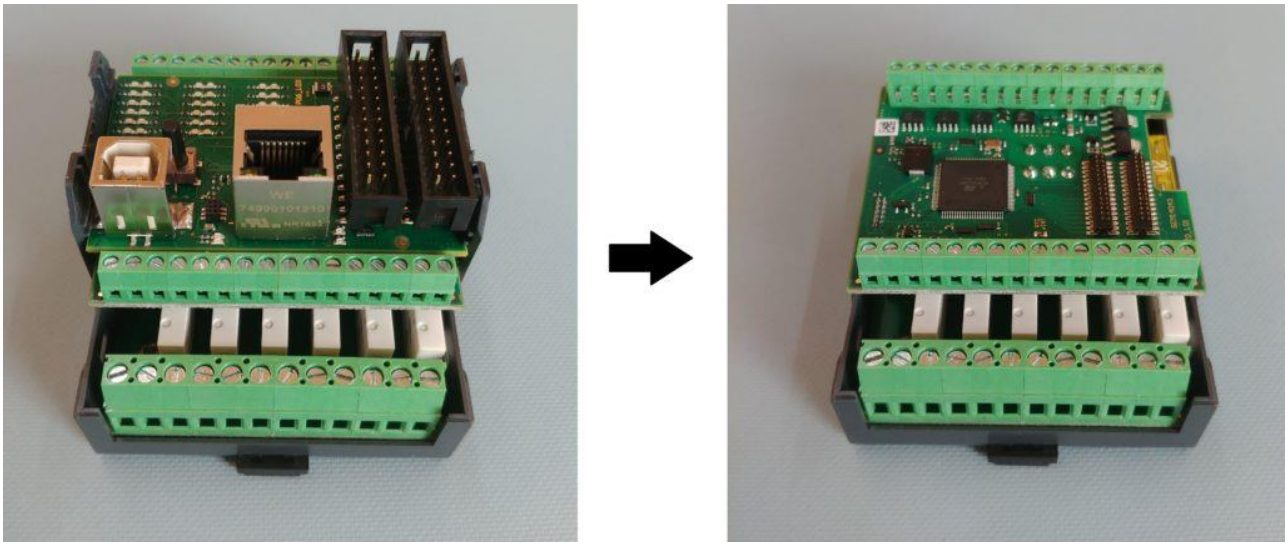
In the following steps we will show you how to turn the voltage to the current outputs.

Step 1:

Remove the Controllino MAXI Automation cover by lifting marked sides of the cover with flat-head screwdriver:

Step 2:

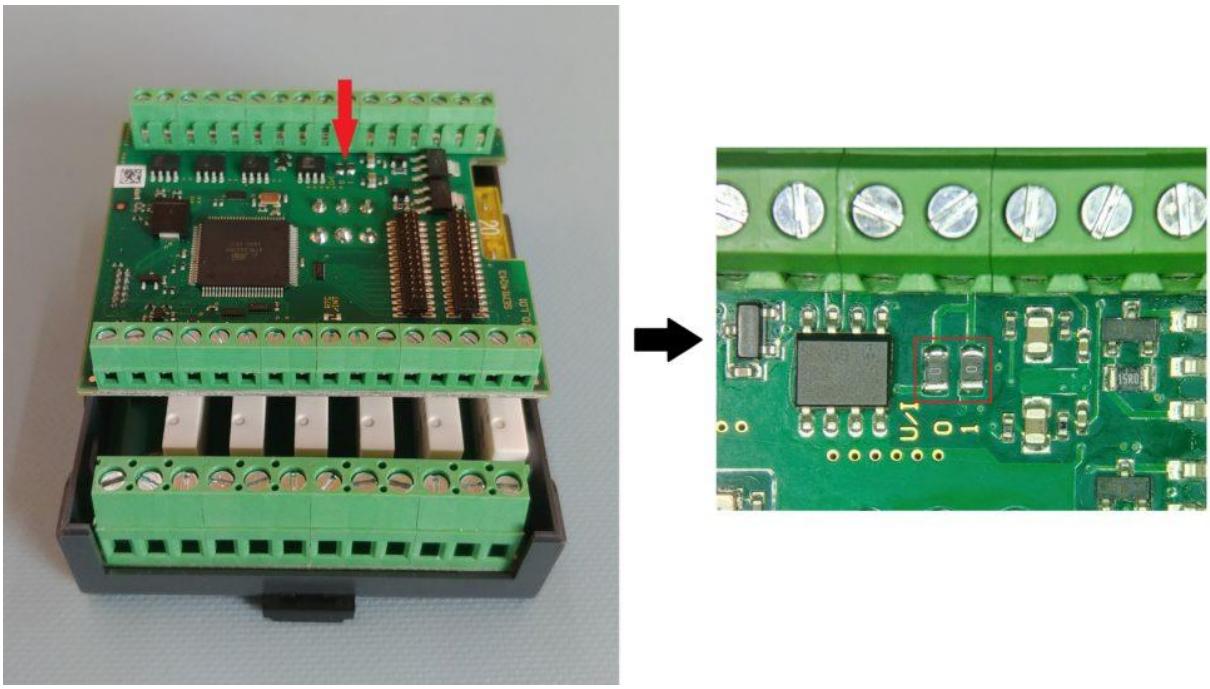
Remove the plastic sides and disconnect the Controllino connection board



from the Controllino MAXI Automation.

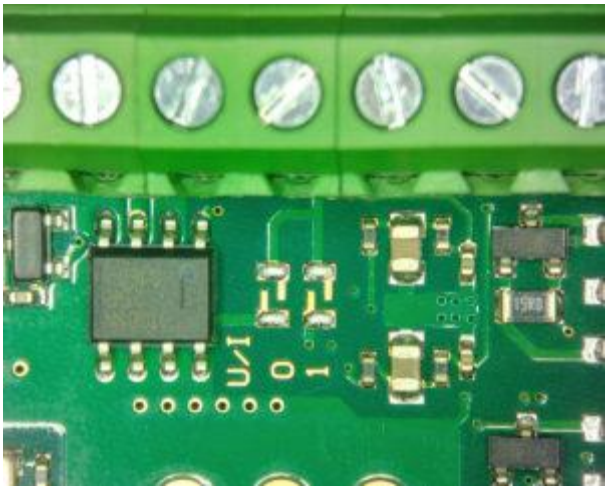
Step 3:

Locate the two 0 Ω resistors of the Controllino relay board:



Step 4:

In order to get the two current outputs (0-20 mA) on Controllino MAXI Automation, apply heat and take the tweezers to remove two 0 Ω resistors:



To get back the voltage outputs (0-10V), simply solder the resistors back on Controllino MAXI Automation control board.

Note*

To test the outputs use previous example or example from the CONTROLLINO library for the Controllino MAXI Automation.

0-10 V -> 0-20 mA

If you are not able to compile the sketch, choose the Controllino MAXI Automation board!

DIGITAL & RELAY BLINK

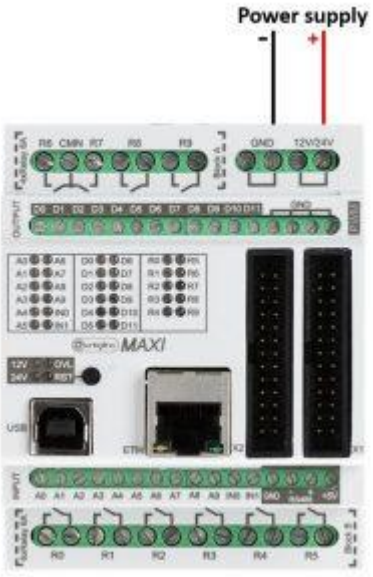
This example shows you how to use our powerful digital outputs and relays. The CONTROLLINO PLCs have “High-Side Switch” outputs, “Half-Bridge” outputs (only MEGA) and potential free relay outputs. Some of these outputs are also capable of generating PWM (Pulse Width Modulation) signals. Therefore it is possible to dim a lamp or to control the speed of a DC motor.

IMPORTANT INFORMATION!
Please, select proper target board in **Tools->Board->Controllino MINI/MAXI/MEGA** before Upload to your CONTROLLINO. (Please, refer to https://github.com/CONTROLLINO-PLC/CONTROLLINO_Library if you do not see the CONTROLLINOs in the Arduino IDE menu **Tools->Board**.)

Hardware Required

- Controllino MINI/MAXI/MEGA
- 12/24V DC Power supply

Circuit



Note*

Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

Code

Controllino MINI

In case of the Controllino MINI, the relays are connected parallelly to the digital outputs D0-D5 and thus are named D0-D5.

```
#include <Controllino.h> /* Usage of CONTROLLINO library allows you to use CONTR
OLLINO_xx aliases in your sketch. */


// the setup function runs once when you press reset (CONTROLLINO RST button) or
connect the CONTROLLINO to the PC

void setup() {

 // initialize all used digital output pins as outputs

 pinMode(CONTROLLINO_D0, OUTPUT); // note that we are using CONTROLLINO aliases
for the digital outputs

 pinMode(CONTROLLINO_D1, OUTPUT);

 pinMode(CONTROLLINO_D2, OUTPUT); // the alias is always like CONTROLLINO_

 pinMode(CONTROLLINO_D3, OUTPUT); // and the digital output label as you can see
at the CONTROLLINO device

 pinMode(CONTROLLINO_D4, OUTPUT); // next to the digital output screw terminal

}


// the loop function runs over and over again forever

void loop() {

 digitalWrite(CONTROLLINO_D0, HIGH); // turn the LED on (HIGH is the voltage lev
el)

 delay(100);                         // wait for 100 milliseconds which is 1/10
of a second

 digitalWrite(CONTROLLINO_D0, LOW);  // turn the LED off by making the voltage L
OW

 delay(100);                         // wait for 100 milliseconds which is 1/10
of a second

 digitalWrite(CONTROLLINO_D1, HIGH);
```

```
 delay(100);
 digitalWrite(CONTROLLINO_D1, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_D2, HIGH); // please, visit https://www.controllino.biz/downloads/
 delay(100);                         // if you want to know more about the mapping of the CONTROLLINO
 digitalWrite(CONTROLLINO_D2, LOW);  // digital outputs to the Arduino pins
 delay(100);
 digitalWrite(CONTROLLINO_D3, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_D3, LOW);  // by using CONTROLLINO aliases instead of Arduino pin numbers
 delay(100);                         // you ensure sketch portability between all CONTROLLINO variants
 digitalWrite(CONTROLLINO_D4, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_D4, LOW);
 delay(100);
}
```

To make the outputs and relays blink, CONTROLLINO pins have to be set up as OUTPUTs!

***pinMode(CONTROLLINO_xx, OUTPUT);**

Controllino MAXI/MEGA

```
#include <Controllino.h> /* Usage of CONTROLLINO library allows you to use CONTROLLINO_xx aliases in your sketch.*/


// the setup function runs once when you press reset (CONTROLLINO RST button) or connect the CONTROLLINO to the PC
void setup() {
 // initialize all used digital output pins as outputs
 pinMode(CONTROLLINO_D0, OUTPUT);
```

```
 pinMode(CONTROLLINO_D1, OUTPUT); // note that we are using CONTROLLINO aliases
for the digital outputs

 pinMode(CONTROLLINO_D2, OUTPUT);

 pinMode(CONTROLLINO_D3, OUTPUT); // the alias is always like CONTROLLINO_

 pinMode(CONTROLLINO_D4, OUTPUT); // and the digital output label as you can see
at the CONTROLLINO device

 pinMode(CONTROLLINO_R0, OUTPUT); // next to the digital output screw terminal

 pinMode(CONTROLLINO_R1, OUTPUT);

 pinMode(CONTROLLINO_R2, OUTPUT);

 pinMode(CONTROLLINO_R3, OUTPUT);

 pinMode(CONTROLLINO_R4, OUTPUT);

}


// the loop function runs over and over again forever

void loop() {

 digitalWrite(CONTROLLINO_D0, HIGH); // turn the LED on (HIGH is the voltage lev
el)

 delay(100);                         // wait for 100 milliseconds which is 1/10
of a second

 digitalWrite(CONTROLLINO_D0, LOW);  // turn the LED off by making the voltage L
OW

 delay(100);                         // wait for 100 milliseconds which is 1/10
of a second

 digitalWrite(CONTROLLINO_D1, HIGH);

 delay(100);

 digitalWrite(CONTROLLINO_D1, LOW);

 delay(100);

 digitalWrite(CONTROLLINO_D2, HIGH); // please, visit https://www.controllino.bi
z/downloads/

 delay(100);                         // if you want to know more about the mappi
ng of the CONTROLLINO

 digitalWrite(CONTROLLINO_D2, LOW);  // digital outputs to the Arduino pins

 delay(100);

 digitalWrite(CONTROLLINO_D3, HIGH);

 delay(100);

 digitalWrite(CONTROLLINO_D3, LOW);  // by using CONTROLLINO aliases instead of
Arduino pin numbers
```

```
 delay(100);                          // you ensure sketch portability between al
l CONTROLLINO variants
 digitalWrite(CONTROLLINO_D4, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_D4, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_R0, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_R0, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_R1, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_R1, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_R2, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_R2, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_R3, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_R3, LOW);
 delay(100);
 digitalWrite(CONTROLLINO_R4, HIGH);
 delay(100);
 digitalWrite(CONTROLLINO_R4, LOW);
 delay(100);


}
```

DISCONNECT RELAYS

If you have a CONTROLLINO MINI device you can use the relay outputs “D0” to “D5” to connect and switch external circuits. The contact type as well as the contact connections are marked on the PLCs. The maximum permissible switching current per relay is 6A (at 250V / AC) or 6A (at a maximum of 30V / DC). The relay outputs are potential free!

On the CONTROLLINO MINI, the relays are connected parallelly to the digital outputs D0-D5 and thus are named D0-D5. If you switch the digital output, the relay digital output will also automatically be switched. In the case you don’t want that to happen you can disconnect the relay outputs by removing the solder line on the Controllino relay board.

Hardware Required

- Controllino MINI
- 12/24V DC Power supply
- Knife or scalpel

Circuit

Note*
Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

Code

Controllino MINI

To check if your relay switches together with the digital output on your Controllino MINI, you can run this code for certain digital output.

```
#include <Controllino.h> /* Usage of CONTROLLINO library allows you to use CONTR
OLLINO_xx aliases in your sketch. */


// the setup function runs once when you press reset (CONTROLLINO RST button) or
connect the CONTROLLINO to the PC

void setup() {

 // initialize all used digital output pins as outputs

 pinMode(CONTROLLINO_D0, OUTPUT);

}


// the loop function runs over and over again forever

void loop() {

 digitalWrite(CONTROLLINO_D0, HIGH); // turn the LED on (HIGH is the voltage lev
el)

 delay(100);                         // wait for 100 milliseconds which is 1/10
of a second

 digitalWrite(CONTROLLINO_D0, LOW);  // turn the LED off by making the voltage L
OW

 delay(100);                         // wait for 100 milliseconds which is 1/10
of a second

}
```

To make the outputs and relays blink, CONTROLLINO pins have to be set up as OUTPUTs!

*pinMode(CONTROLLINO_xx, OUTPUT);

Steps

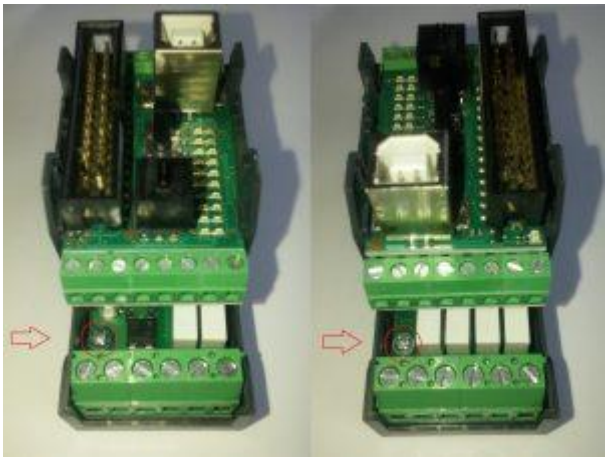
In the following steps we will show you how to disconnect relays on the Controllino relay board.

Step 1:

Remove the Controllino MINI cover by lifting marked sides of the cover with flat-head screwdriver:

Step 2:

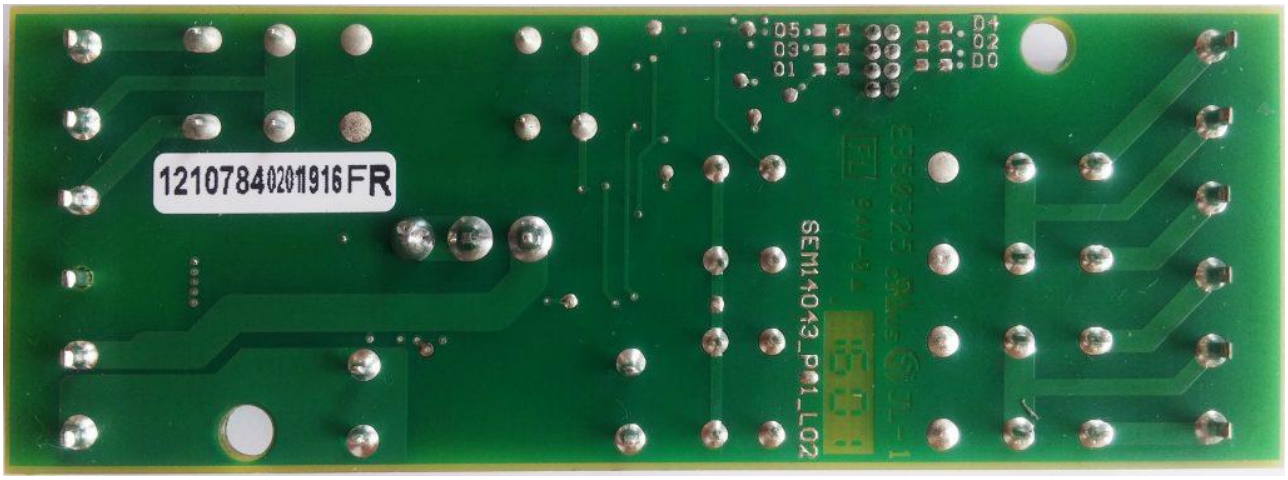
Remove these two screws that hold Controllino relay board in place:



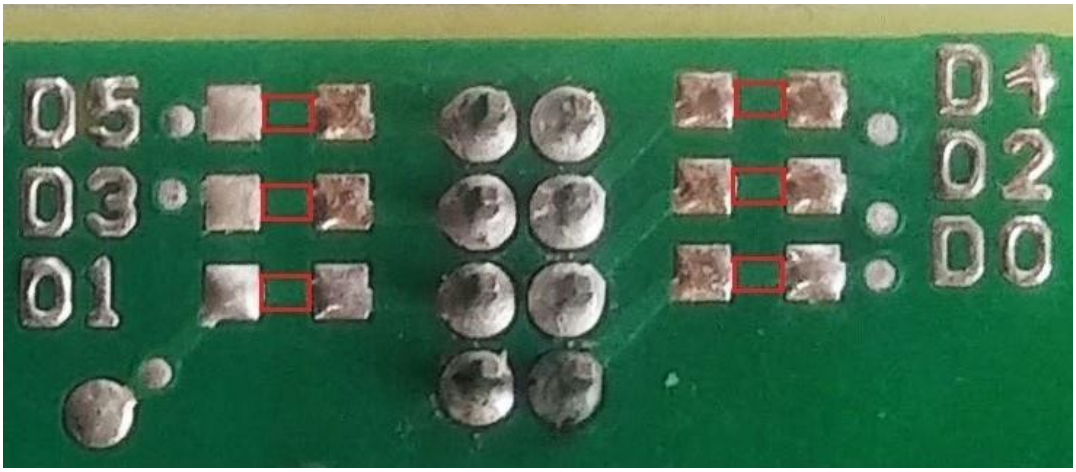
and separate the device from the bottom case.

Step 3:

Turn to the bottom side of Controllino relay board:

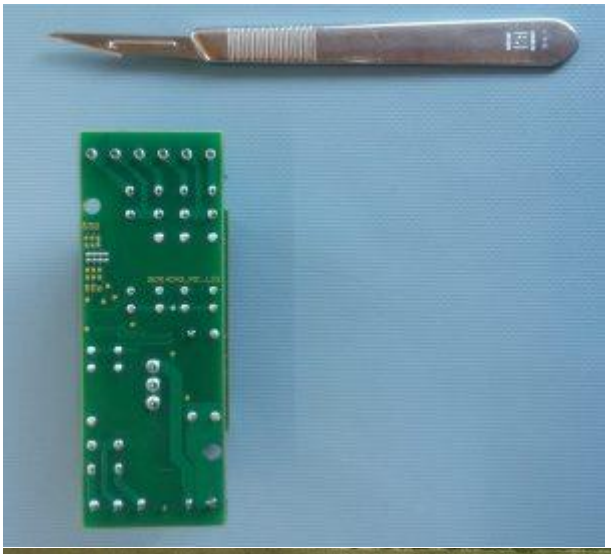


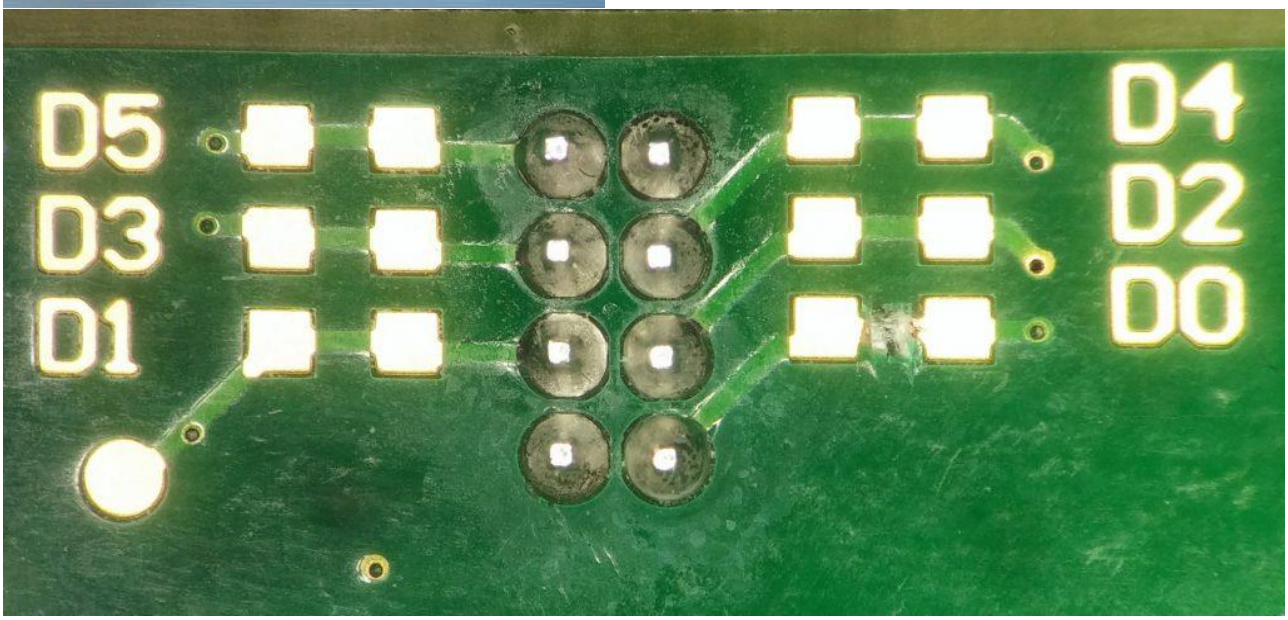
and locate this section on the right corner of the board:



Step 4:

Take the sharp knife or scalpel and remove the solder line of the wanted digital output, marked with a red square on a previous picture. When removing the marked solder line, the best way is to make a cut on each side of the marked square and then remove the middle part.





In this example the relay D0 is disconnected from the digital output. When you are finished you can upload the same program to check it again.

Note*

You can allways reconnect the relay outputs by connecting the 0 ohm smd resistor or some kind of conductor to reestablish the broken connection.

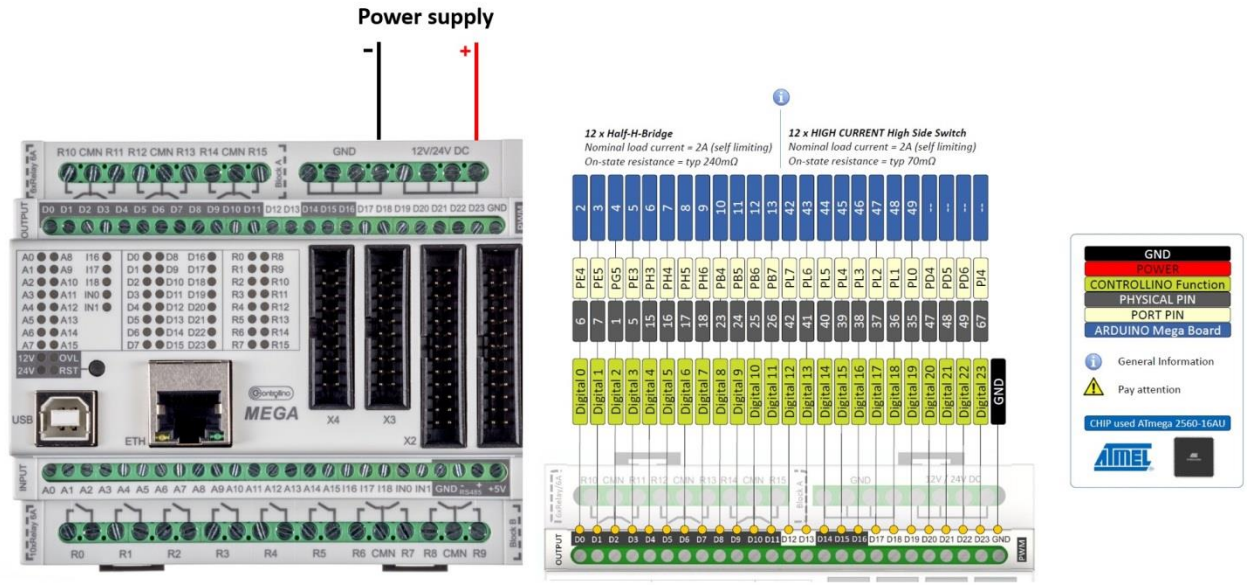
ENABLE D20-D23 PINS (MEGA)

Controllino MEGA has 24 High Side digital outputs available. The first 20 outputs we can control using the standard Arduino functions (e.g. digitalWrite and analogWrite), but for D20-D23 digital outputs we have to know how to use PORT manipulation on ATmega2560. On the connection picture we can see that there are no Arduino pins assigned to these outputs.

Hardware Required

- Controllino MEGA
- 12/24V DC Power supply

Circuit



Note*
Pin header is working on 5V TTL levels. Voltage levels over 5.5V can damage the Controllino permanently.

Code

Controllino MEGA

```
void setup()
{
 DDRD = DDRD | B01110000; //Set the ports PD4, PD5, PD6 as outputs
 DDRJ = DDRJ | B00010000; //Set the port PJ4 as output
}

void loop() {
 int del = 100;
 //Digital output 20
 PORTD = PORTD | B00010000; //Set HIGH
 delay(del);
 PORTD = PORTD & B11101111; //Set LOW
 delay(del);

}



/*
//Digital output 21
 PORTD = PORTD | B00100000;
 delay(del);
 PORTD = PORTD & B11011111;
 delay(del);



//Digital output 22
 PORTD = PORTD | B01000000;
 delay(del);
 PORTD = PORTD & B10111111;
 delay(del);
```

```
//Digital output 23
 PORTD = PORTD | B10000000;
 delay(del);
 PORTD = PORTD & B01111111;



 PORTJ = PORTJ | B00010000;
 delay(del);
 PORTJ = PORTJ & B11101111;
 delay(del);



 PORTD = PORTD | B01110000; // sets Digital Outputs 20,21,22 in one shot to HIGH
                            // -> turns the LEDs ON
 PORTJ = PORTJ | B00010000; // sets Digital Output 23 in one shot to HIGH
                            // -> turns the LED ON



 PORTD = PORTD & B10001111; // sets Digital Outputs 20,21,22 in one shot to LOW
                            // -> turns the LEDs OFF
 PORTJ = PORTJ & B11101111; // sets Digital Output 23 in one shot to LOW
                            // -> turns the LED OFF
*/
```