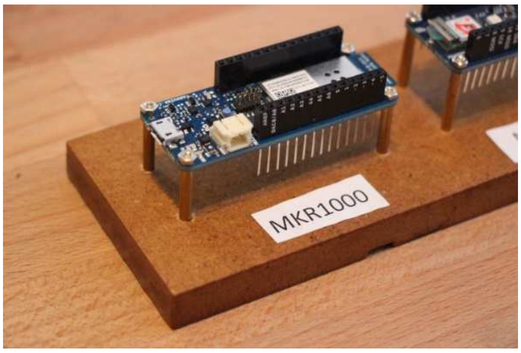
# Getting started with WawiLib over Wi-Fi



# 1. Introduction

## 1.1. Objective of this document.

The objective of this demo is to describe step by step how to get WawiLib up and running with a very small Arduino example program (a sketch). In this demo Wi-Fi will be used as communication link.

Many users know the Arduino "Blink" sketch. "Blink" is designed to blink an on-board LED as you can find on many Arduino boards. In this document you will learn how to create "WawiBlink" – the WawiLib version of "Blink".

"WawiBlink" blinks the same LED, but with variable time intervals. The time the LED is on and the time it is off is defined by 2 variables: *delayOn* and *delayOff*. The number of blinks is stored in the variable *blinkCounter*.

In this demo you will learn how to monitor and modify *delayOn*, *delayOff* and *blinkCounter* while the sketch is running on the Arduino board. The demo will also demonstrate how you can record the value of *blinkCounter* in an .xml, .xlsx or .csv file that can later be opened in Microsoft Excel, LibreOffice or a program you have written yourself.

You will also learn how to create diagnostics messages that will be displayed in the console output window of the WawiLib-PC application. The example uses this function to report the state changes of the on-board LED.

This demo also shows how to record the output of the sketch .print() statements in a disk file on your PC and the use of breakpoints in your sketch.

## 1.2. Software and hardware requirements

The Arduino IDE (in this example 1.8.15) and WawiLib V2.0.x both need to be installed on your PC. The demo runs with licensed and unlicensed versions of WawiLib. During the grace period of 2 months, you can test and use all functions without registration. After this period registration is required in order to access all functions. At this time registration is free. In the future a small contribution might be required to register in order to support the website.

WawiLib supports multiple interface types: serial, software serial, USB, USB-native, TCP/IP, UDP/IP via cable or Wi-Fi. In this demo Wi-Fi is used as interface between WawiLib and an Arduino board. The protocol used is UDP/IP.

The hardware you need is an Arduino MKR1000, a USB programming cable and a Windows PC (32 or 64 bit).  Examples that work with MKR1010, Arduino Wi-Fi shield and ESP8266 are also provided with WawiWifi. You also need a router with Wi-Fi capabilities that is connected to your PC. (The unfortunately obsolete Arduino Wi-Fi Shield is also supported by WawiLib.)

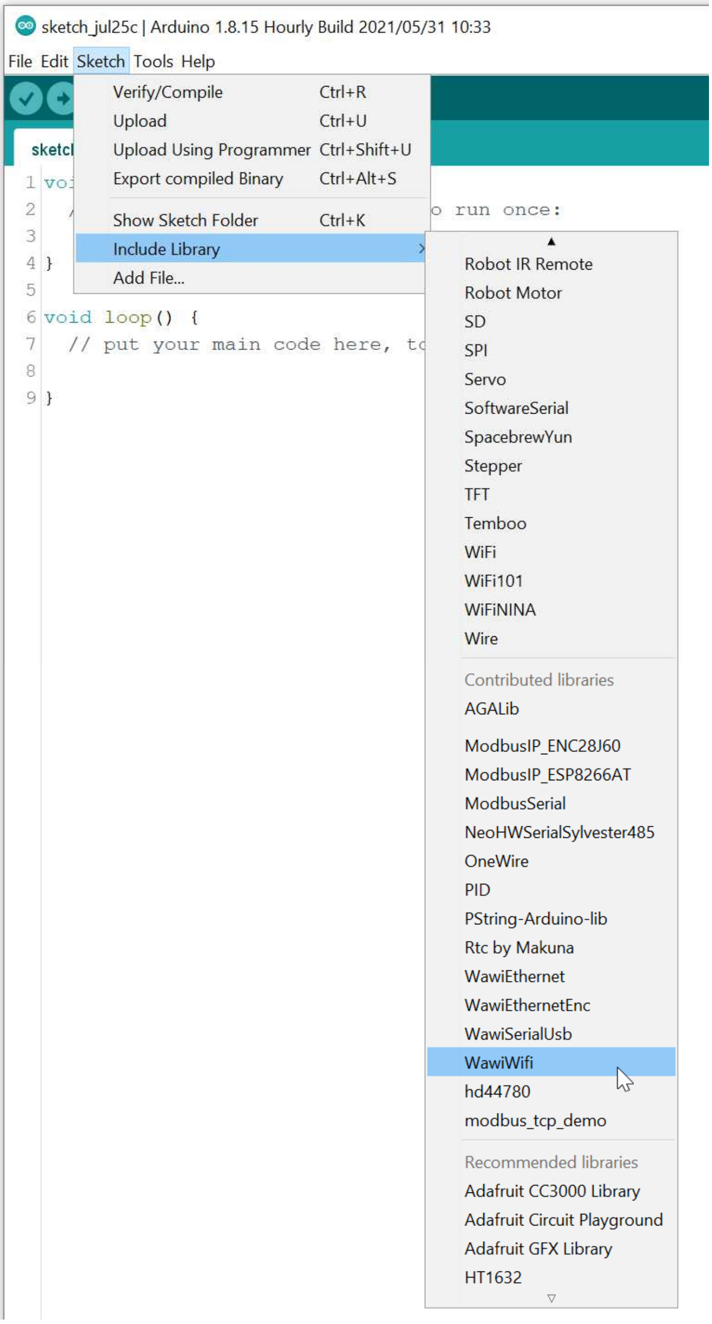
## 1.3. Required user experience

This demo assumes that you are able to edit, compile and download Arduino programs. You should also have basic computer skills such as downloading and installing Windows programs. We will use Wi-Fi to connect to the Arduino board, therefore some basic knowledge of Ethernet and Wi-Fi networks is also required.

## 2. Installing the WawiLib software

This section describes the steps you have to follow in order to install the WawiLib program and the WawiWifi Arduino library. If both have been correctly installed on your PC, you can skip this section.

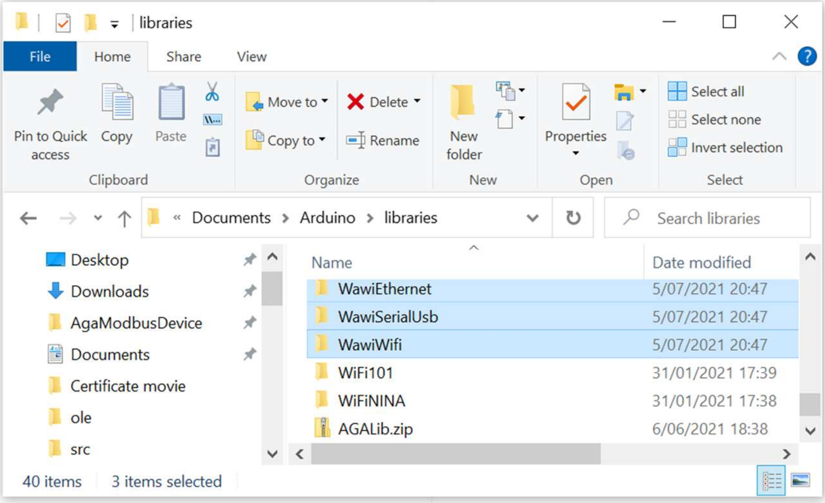
- ✓ Download the WawiLib installer from [www.sylvestersolutions.com](www.sylvestersolutions.com).
- ⇨ Install WawiLib using the downloaded WawiLib32.msi or WawiLib64.msi installer.
- ⇨ Start WawiLib.
- ➢ WawiLib will unpack the zipped WawiLib Arduino libraries and put them in the library directory of the Arduino IDE.
- ⇨ Open the Arduino IDE.
- ⇨ Check the presence of the installed libraries:



```
vo
  /                                          o run once:
}

void loop() {
  // put your main code here, t
}
```

Fig. 2.1. Check the installation WawiWifi library in the Arduino IDE.

The libraries WawiSerialUsb, WawiEthernet and WawiWifi can be found in: C:\Users\[your user name]\Documents\Arduino\libraries.

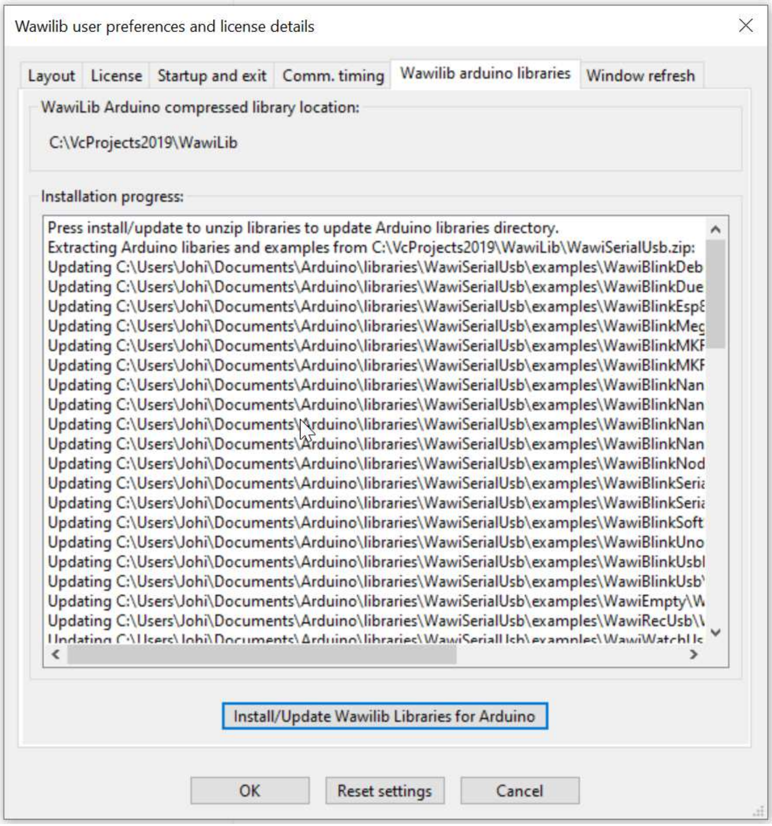


| Name | Date modified |
|---|---|
| WawiEthernet | 5/07/2021 20:47 |
| WawiSerialUsb | 5/07/2021 20:47 |
| WawiWifi | 5/07/2021 20:47 |
| WiFi101 | 31/01/2021 17:39 |
| WiFiNINA | 31/01/2021 17:38 |
| AGALib.zip | 6/06/2021 18:38 |

Fig. 2.2. Unpacked Libraries after installing WawiLib.

Note: 1) if, by exception, automatic installation of the libraries fails, you can manually unzip the WawiSererialUsb.zip, WawiEthernet.zip and WawiWifi.zip in the Documents\Arduino\Libraries directory. The libraries can be found in the installation directory of WawiLib.exe itself.

Note: 2) Manual installation of libraries can be triggered in the WawiLib menu "Settings\Preferences and license”. In tab “WawiLib Arduino libraries” press the button "Install\Update WawiLib Libraries for Arduino".


Wawilib user preferences and license details
Layout
License
Startup and exit
Comm. timing
Wawilib arduino libraries
Window refresh
Install/Update Wawilib Libraries for Arduino
OK
Reset settings
Cancel

WawiLib Arduino compressed library location:

C:\VcProjects2019\WawiLib

Installation progress:

```
Press install/update to unzip libraries to update Arduino libraries directory.
Extracting Arduino libaries and examples from C:\VcProjects2019\WawiLib\WawiSerialUsb.zip:
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkDeb
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkDue
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkEsp8
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkMeg
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkMKF
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkMKF
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkNan
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkNan
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkNan
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkNan
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkNod
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkSeria
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkSeria
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkSoft
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkUno
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkUsbl
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiBlinkUsb\
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiEmpty\W
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiRecUsb\\
Updating C:\Users\Johi\Documents\Arduino\libraries\WawiSerialUsb\examples\WawiWatchUs
```

Fig. 2.3. Manual install of Arduino libraries.

# 3. Wi-Fi network setup

## 3.1. Network layout

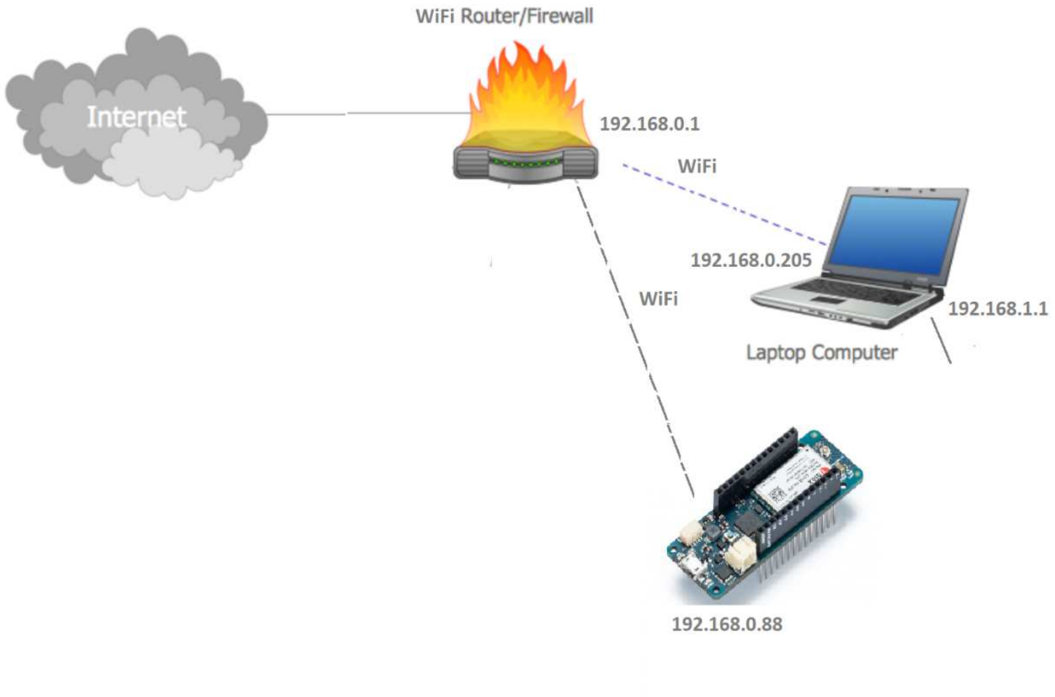


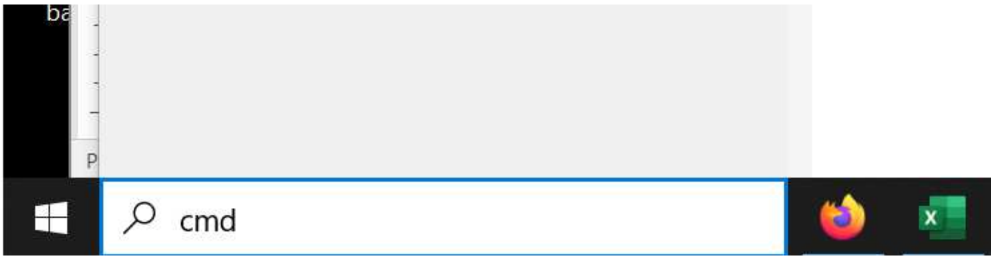
Fig. 3.1. Test network architecture overview.

In the drawing above you can see the demo test setup. The demo laptop is connected via Wi-Fi to the local network. There is also a Wi-Fi connection between the router (or switch) of the local network and the Arduino board.

- ✓ Make sure the Arduino board has power via USB or via a separate power supply.

## 3.2. Identify your network parameters.

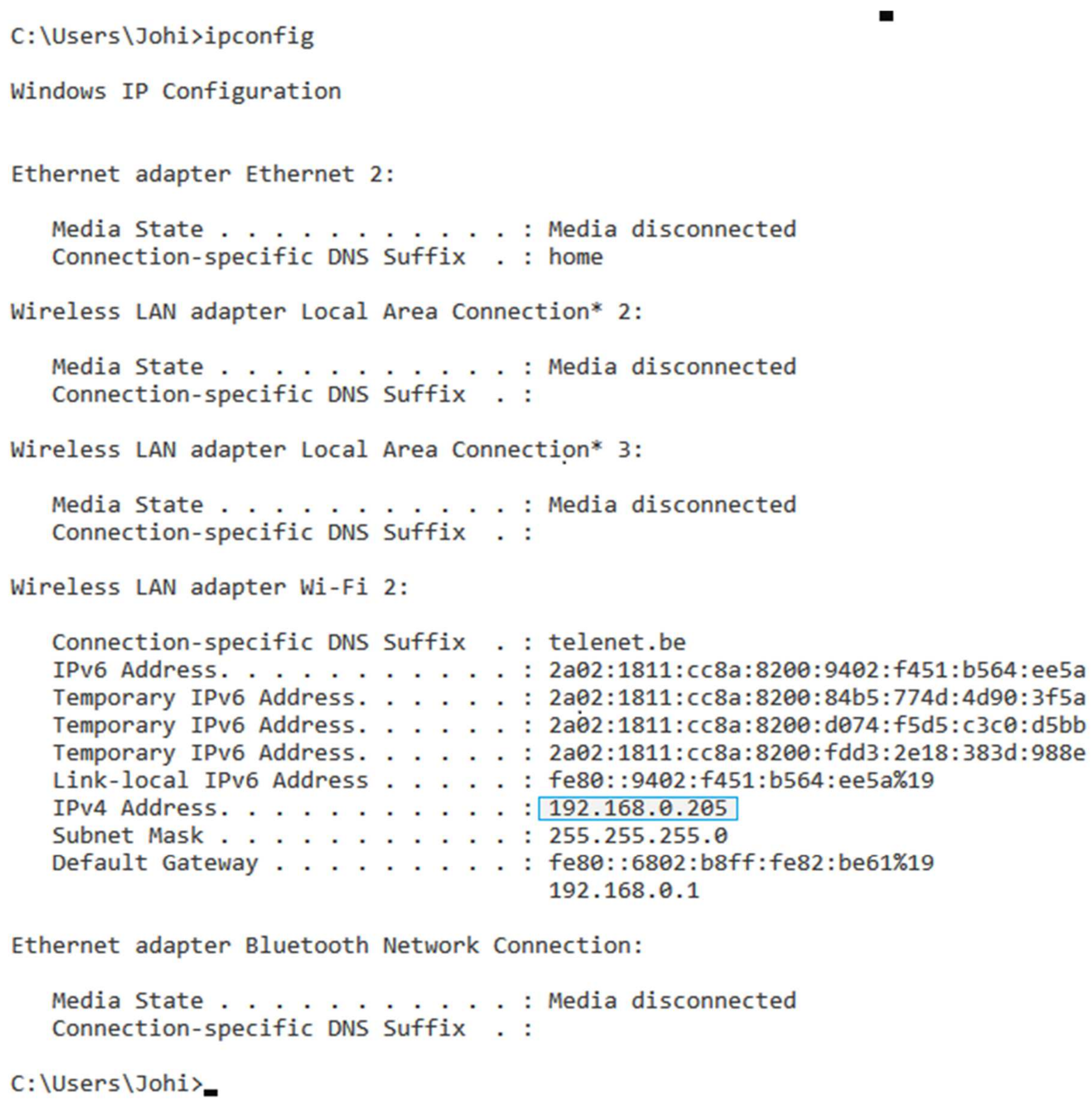
- ✓ Open a command line window on your PC: press magnification glass and type “CMD”.



```
cmd
```

Fig. 3.2. Opening a console window.

- ✓ On the command prompt type “IPCONFIG”.



```
C:\Users\Johi>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : home

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi 2:

   Connection-specific DNS Suffix  . : telenet.be
   IPv6 Address. . . . . . . . . . . : 2a02:1811:cc8a:8200:9402:f451:b564:ee5a
   Temporary IPv6 Address. . . . . . : 2a02:1811:cc8a:8200:84b5:774d:4d90:3f5a
   Temporary IPv6 Address. . . . . . : 2a02:1811:cc8a:8200:d074:f5d5:c3c0:d5bb
   Temporary IPv6 Address. . . . . . : 2a02:1811:cc8a:8200:fdd3:2e18:383d:988e
   Link-local IPv6 Address . . . . . : fe80::9402:f451:b564:ee5a%19
   IPv4 Address. . . . . . . . . . . : 192.168.0.205
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::6802:b8ff:fe82:be61%19
                                       192.168.0.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\Johi>_
```

Fig. 3.3. Console window with output of “ipconfig” to enumerate available network adapters.

In the window above, we see that my demo PC has 2 Ethernet adapters connected to 2 different networks: On 192.168.1.1, there is a hardware Ethernet adapter linked to the network segment 192.168.1.X. (disconnected) On 192.168.0.205, there is an Adapter connected to the network segment 192.168.0.X.

In this demo I will use 192.168.0.X. If your network has different properties, replace 192.168.0. appropriately in the text below.

## 3.3. Identify a free IP address on your local network segment

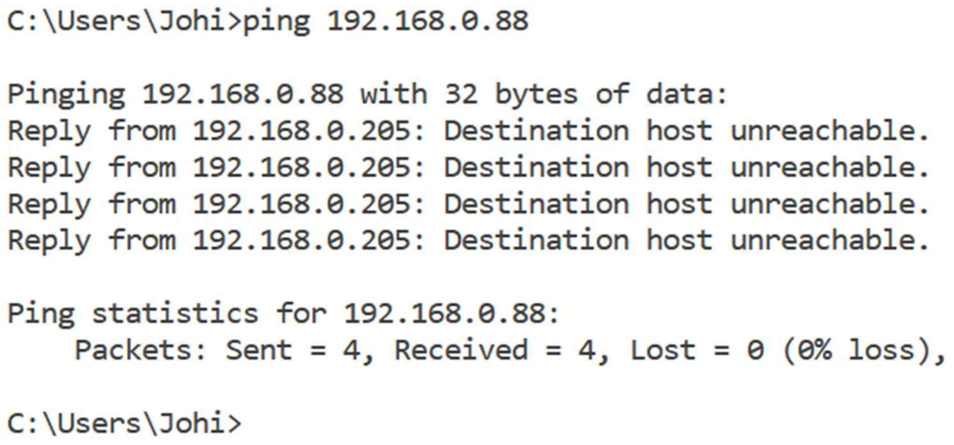
In this section we will use the command “ping” to identify a free network address on our network segment. First, we will ping the Ethernet adapter of the PC itself to see if ping works, then we will ping to 192.168.0.88 to see if there is a network adapter present on the address 192.168.0.88. If 192.168.0.88 is not used (timeout with ping), we can then use 192.168.0.88 as free Ethernet address for our Arduino network shield.

- ✓ Open a command window (as with the ipconfig CMD in the previous section).
- ✓ Type “ping 192.168.0.1”.

As 192.168.0.1 the PC network adapter address we will see a positive result (response time < time out).

- ✓ Type “ping 192.168.0.88”.

This command should time-out. If 192.168.0.88 does not time-out, replace .88 by other addresses (2...255) to find a free address on your local network segment:



```
C:\Users\Johi>ping 192.168.0.88

Pinging 192.168.0.88 with 32 bytes of data:
Reply from 192.168.0.205: Destination host unreachable.
Reply from 192.168.0.205: Destination host unreachable.
Reply from 192.168.0.205: Destination host unreachable.
Reply from 192.168.0.205: Destination host unreachable.

Ping statistics for 192.168.0.88:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

C:\Users\Johi>
```

Fig. 3.4. console window with output of “Ping” indicating 192.168.0.88 is not used.

## 4. Load Arduino board with demo sketch

Many of the Arduino libraries come with examples. WawiLib is not an exception. In this demo we will use WawiBlinkMkr1000UdpWifi.ino as demo sketch. Similar demos for other boards are available in the example sketches menu of the Arduino IDE.

✓ Open the example via the menu “File\Examples\WawiWifi\WawiBlinkMkr1000UdpWifi” in the Arduino IDE.

The largest part of the code is used to initialize the Wi-Fi library. The WawiLib specific part is there to create a WawiWifi object called WawiSrv:

```
// declare Arduino WiFi UDP communication object:
WiFiUDP server;

// WawiLib communications object:
WawiWifi WawiSrv;
```

Fig. 4.1. Declaring WiFi UDP and WawiLib objects to manage communication with WawiLib-PC.

The function wawiVarDef is called (only 1 time) to look up the address and size of a variable in memory. You should not call “wawiVarDef” from your sketch. The call is done by WawiWifi in the background. (The macro wawiVar and wawiVarArray do the translation.)

```
// make variables of interest known to WawiLib:
// this function is used in WawiSrv.begin(....)
void wawiVarDef()
{
    WawiSrv.wawiVar(delayOn);
    WawiSrv.wawiVar(delayOff);
    WawiSrv.wawiVar(blinkCounter);
}
```

Fig. 4.2. Making variables visible to WawiLib in your sketch.

The statement WawiSrv.begin(…) initializes WawiWifi. The parameters are: the name of the function “wawiVarDef()”, the initialized WiFiUDP server object and a name of your choice.

```
    // start UDP/IP server:
    server.begin(port);

    // Setup WawiLib:
    Serial.println(F("E) Initializing WawiLib:"));
    WawiSrv.begin(wawiVarDef, server, "MyArduino");
    Serial.println(F("-> Done"));

    pinMode(LED, OUTPUT);
```

Fig. 4.3. Initialize of the WawiLib object calling begin().

On every pass of WawiSrv.loop(), a check is made to see if there are incoming WawiLib PC requests. In the same way WawiSrv.delay() splits the waiting interval into small delay segments with checks for incoming WawiLib requests after each segment. At each print statement, data is put on a buffer and sent to the PC if the buffer is full. The buffer is also flushed during the .loop() command.

```
void loop()
{
    blinkCounter++;
    WawiSrv.print("WawiSrv.Print() demo in loop() function, blinkcounter = ");
    WawiSrv.println(blinkCounter);
    WawiSrv.println("LED is ON.");
    digitalWrite(LED, HIGH);
    WawiSrv.delay(delayOn);

    WawiSrv.println("LED is OFF.");
    digitalWrite(LED, LOW);
    WawiSrv.delay(delayOff);

    WawiSrv.loop();
}
```

Fig. 4.4. the main loop with .loop() commands and .delay() commands.

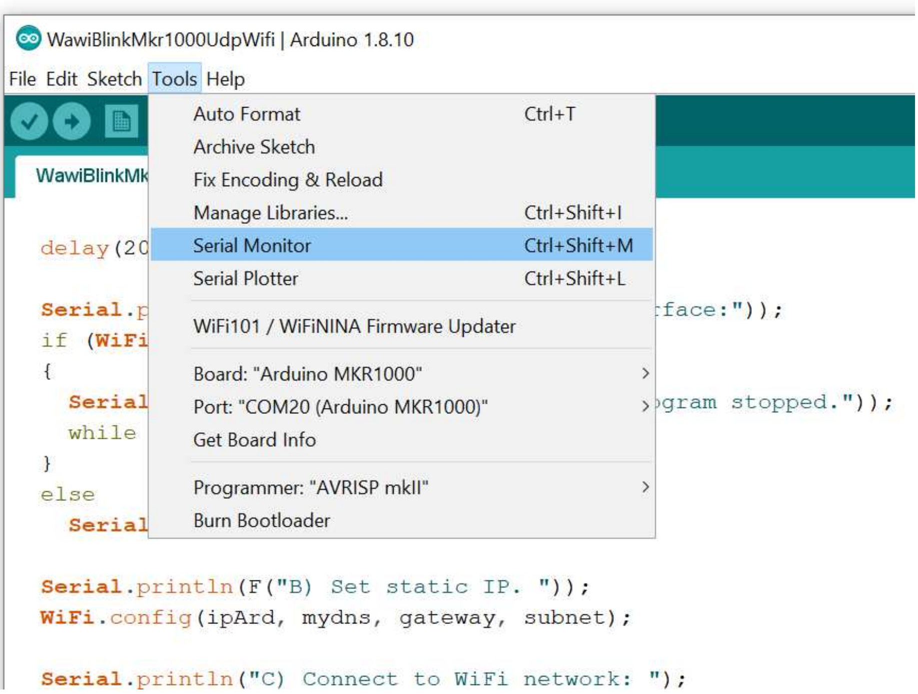
In order to make the demo work, you need to fill in several parameters determined by your local network and Wi-Fi configuration:

- ✓ Fill in the free IP address identified in the previous section (192.168.0.88).
- ✓ Modify ssid[] to the name of your wireless network (replace the text “--SSID--” & delete //).
- ✓ Modify key[] to the key of your wireless network (replace the text “--SSID KEY--” & delete //).
- ✓ Modify ipArd[] to the free IP address determined in the section above.
- ✓ Modify gateway[] to the IP address of the gateway identified in the section above.

```
// Wi-Fi parameters:
// char ssid[] = "--SSID--";   // network SSID (name)
// char key[] = "--SSID KEY--";   // network key

// the media access control (ethernet hardware) address for the shield:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x88 };

// the IP address of your Arduino shield:
byte ipArd[] = { 192, 168, 0, 88 };

// communication port Arduino side for WawiLib communication (can be changed)
unsigned int port = 49152;

// your router's gateway address:
byte gateway[] = { 192, 168, 0, 1 };

// your network subnet:
byte subnet[] = { 255, 255, 225, 0 };

// your dns
byte mydns[] = { 195, 130, 130, 5 }; // (not essential)
```

Fig. 4.5. parametrizing the WiFi connection and the fixed IP address.

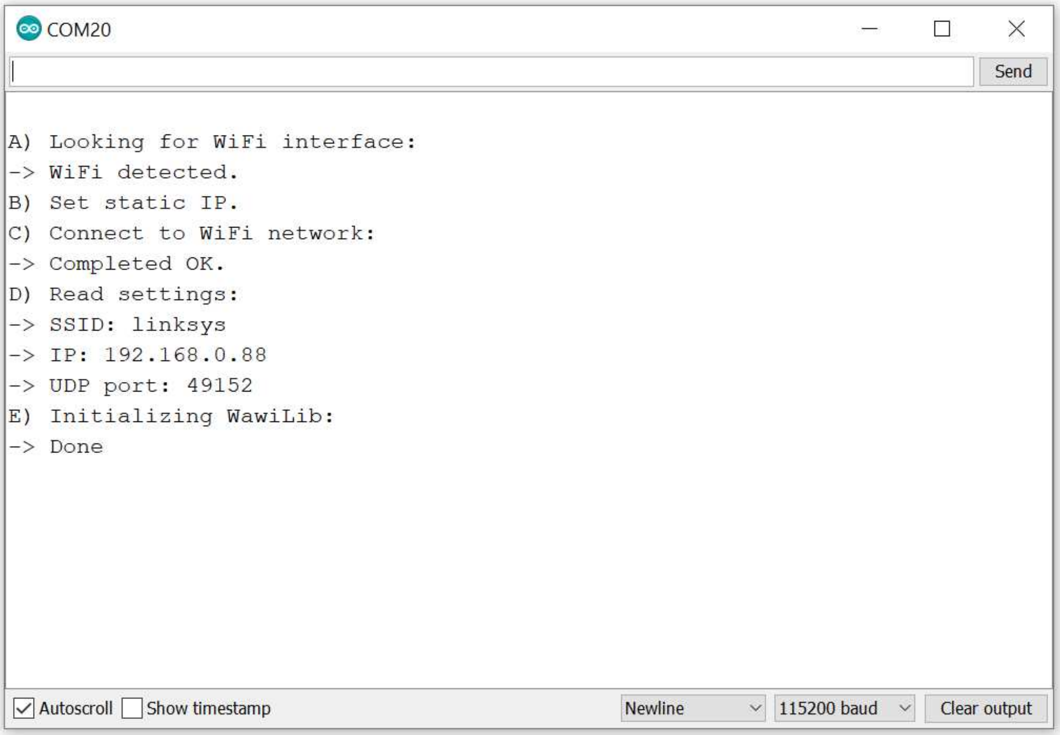
- ✓ Compile WawiWifi and download the file to your Arduino board.
- ✓ Open a serial monitor window in the Arduino IDE and check the output.


WawiBlinkMkr1000UdpWifi | Arduino 1.8.10
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino MKR1000"
Port: "COM20 (Arduino MKR1000)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
WawiBlinkMk

```
delay(20
Serial.p                                        face:"));
if (WiFi
{
  Serial                                        gram stopped."));
  while
}
else
  Serial

Serial.println(F("B) Set static IP. "));
WiFi.config(ipArd, mydns, gateway, subnet);

Serial.println("C) Connect to WiFi network: ");
```

Fig. 4.6. Open the serial monitor window.

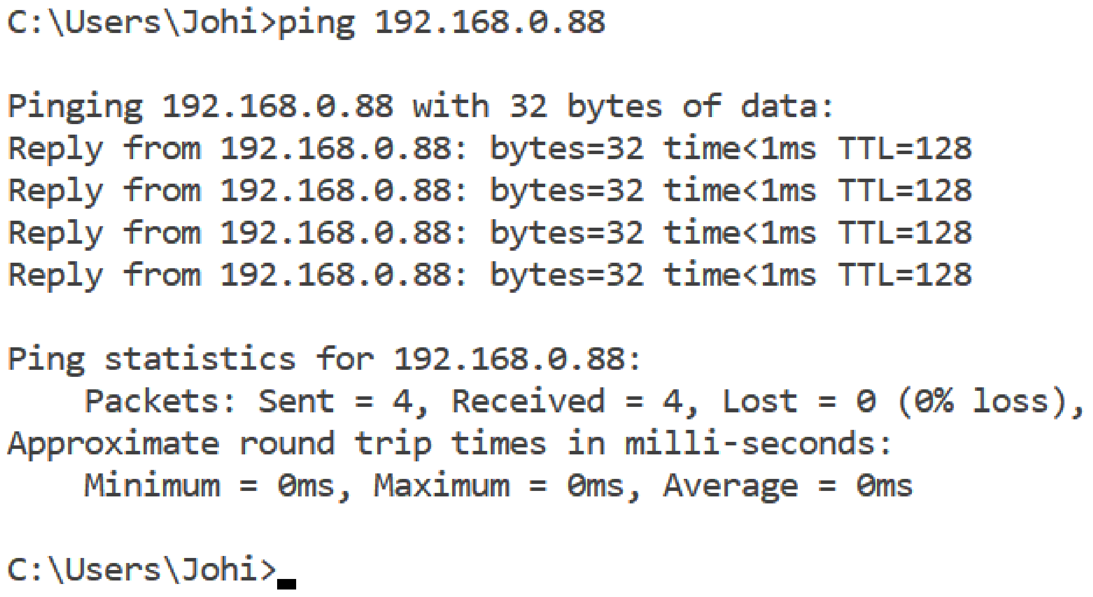
Below you see the output of the Arduino sketch when it starts up:



```
A) Looking for WiFi interface:
-> WiFi detected.
B) Set static IP.
C) Connect to WiFi network:
-> Completed OK.
D) Read settings:
-> SSID: linksys
-> IP: 192.168.0.88
-> UDP port: 49152
E) Initializing WawiLib:
-> Done
```

Fig. 4.7. Serial monitor output indicating an proper initialization of WawiLib in the Sketch.

✓ Validation: Ping to 192.168.0.88. There should be a timed response and not a time-out message.



```
C:\Users\Johi>ping 192.168.0.88

Pinging 192.168.0.88 with 32 bytes of data:
Reply from 192.168.0.88: bytes=32 time<1ms TTL=128
Reply from 192.168.0.88: bytes=32 time<1ms TTL=128
Reply from 192.168.0.88: bytes=32 time<1ms TTL=128
Reply from 192.168.0.88: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.88:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\Johi>_
```

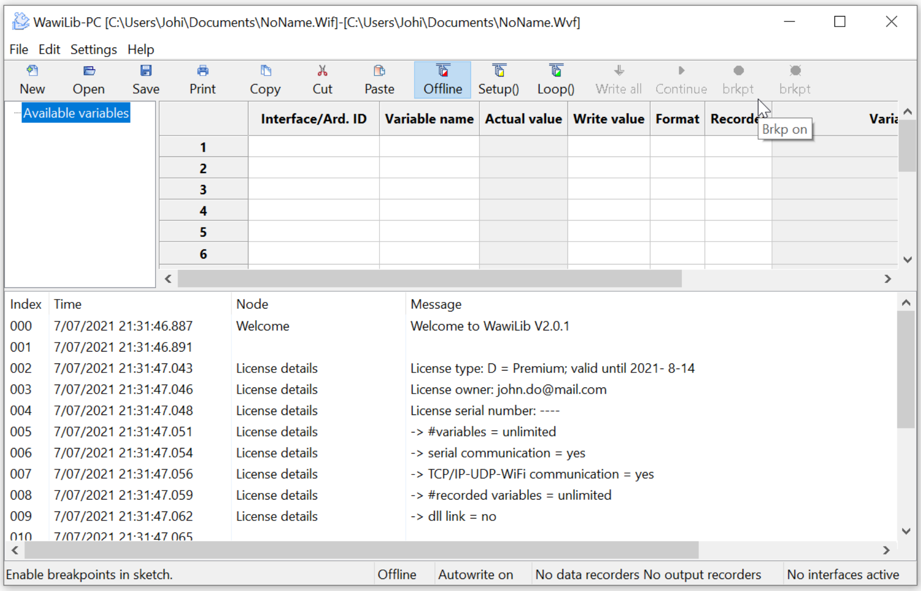
Fig. 4.8. Check the existence of Ethernet node 192.168.0.88. on the network (= ok).

✓ Check if the program was properly downloaded by looking at the LED on the board. It should blink 500ms on and 500ms off.

At this time we have an Arduino board that is properly connected to one of our network segments and correctly parametrized to go online and monitor and modify its (static) variables. In the next section we will go online, monitor and modify these variables. The LED on your Arduino board at I/O 6 should blink 500ms on 500ms off.

# 5. WawiLib user interface overview

✓ Start WawiLib on your PC:


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
Interface/Ard. ID Variable name Actual value Write value Format Recorder Varia
Brkp on
1 2 3 4 5 6
Enable breakpoints in sketch. Offline Autowrite on No data recorders No output recorders No interfaces active

```
Index  Time                     Node             Message
000    7/07/2021 21:31:46.887   Welcome          Welcome to WawiLib V2.0.1
001    7/07/2021 21:31:46.891
002    7/07/2021 21:31:47.043   License details  License type: D = Premium; valid until 2021- 8-14
003    7/07/2021 21:31:47.046   License details  License owner: john.do@mail.com
004    7/07/2021 21:31:47.048   License details  License serial number: ----
005    7/07/2021 21:31:47.051   License details  -> #variables = unlimited
006    7/07/2021 21:31:47.054   License details  -> serial communication = yes
007    7/07/2021 21:31:47.056   License details  -> TCP/IP-UDP-WiFi communication = yes
008    7/07/2021 21:31:47.059   License details  -> #recorded variables = unlimited
009    7/07/2021 21:31:47.062   License details  -> dll link = no
010    7/07/2021 21:31:47.065
```

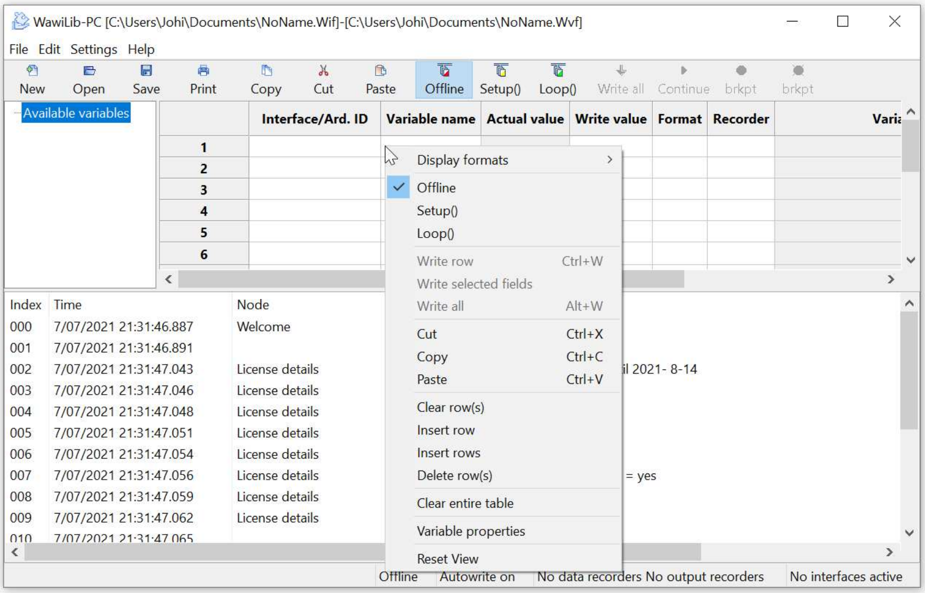
Fig. 5.1. WawiLib startup screen

The main window is split into 3 parts. The upper part contains a grid and a tree control, the bottom part contains a list box.

Once connected to the Arduino, the tree control shows all shared (static) variables in your sketch. In the grid control you enter the variables of your interest, the interface to be used, some parameters related to the variable itself and the data recorder(s) to be used. Drag & drop from the tree to the grid are also possible.

Interface and recorders can be configured using the “Settings” menu.

✓ Right click on the grid in the top window for additional options:


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
Interface/Ard. ID Variable name Actual value Write value Format Recorder Varia
1 2 3 4 5 6
Display formats
✓ Offline
Setup()
Loop()
Write row Ctrl+W
Write selected fields
Write all Alt+W
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Clear row(s)
Insert row
Insert rows
Delete row(s)
Clear entire table
Variable properties
Reset View
Offline Autowrite on No data recorders No output recorders No interfaces active

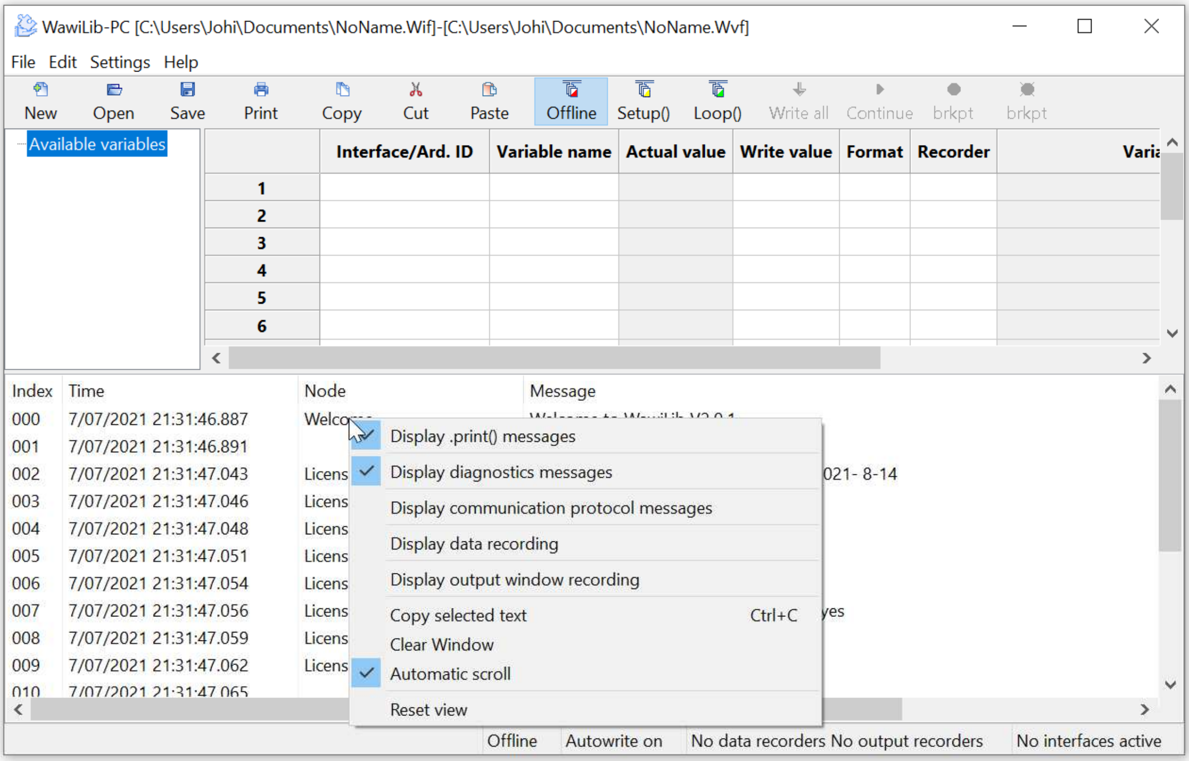
```
Index  Time                     Node
000    7/07/2021 21:31:46.887   Welcome
001    7/07/2021 21:31:46.891
002    7/07/2021 21:31:47.043   License details           il 2021- 8-14
003    7/07/2021 21:31:47.046   License details
004    7/07/2021 21:31:47.048   License details
005    7/07/2021 21:31:47.051   License details
006    7/07/2021 21:31:47.054   License details
007    7/07/2021 21:31:47.056   License details           = yes
008    7/07/2021 21:31:47.059   License details
009    7/07/2021 21:31:47.062   License details
010    7/07/2021 21:31:47.065
```

Fig. 5.2. WawiLib grid options.

Most of the options do not require additional comment, but the sub option “Display format” allows you to select various display formats for the variables in the grid.

The lower part is an output window used to report what WawiLib is doing. It is very handy if you have trouble going online on your board or if you want to see if a variable change was written to your Arduino board properly.

✓ Right click on the bottom window (output window) for additional options:


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
Interface/Ard. ID Variable name Actual value Write value Format Recorder Varia
1 2 3 4 5 6
✓ Display .print() messages
✓ Display diagnostics messages
Display communication protocol messages
Display data recording
Display output window recording
Copy selected text Ctrl+C
Clear Window
✓ Automatic scroll
Reset view
Offline Autowrite on No data recorders No output recorders No interfaces active

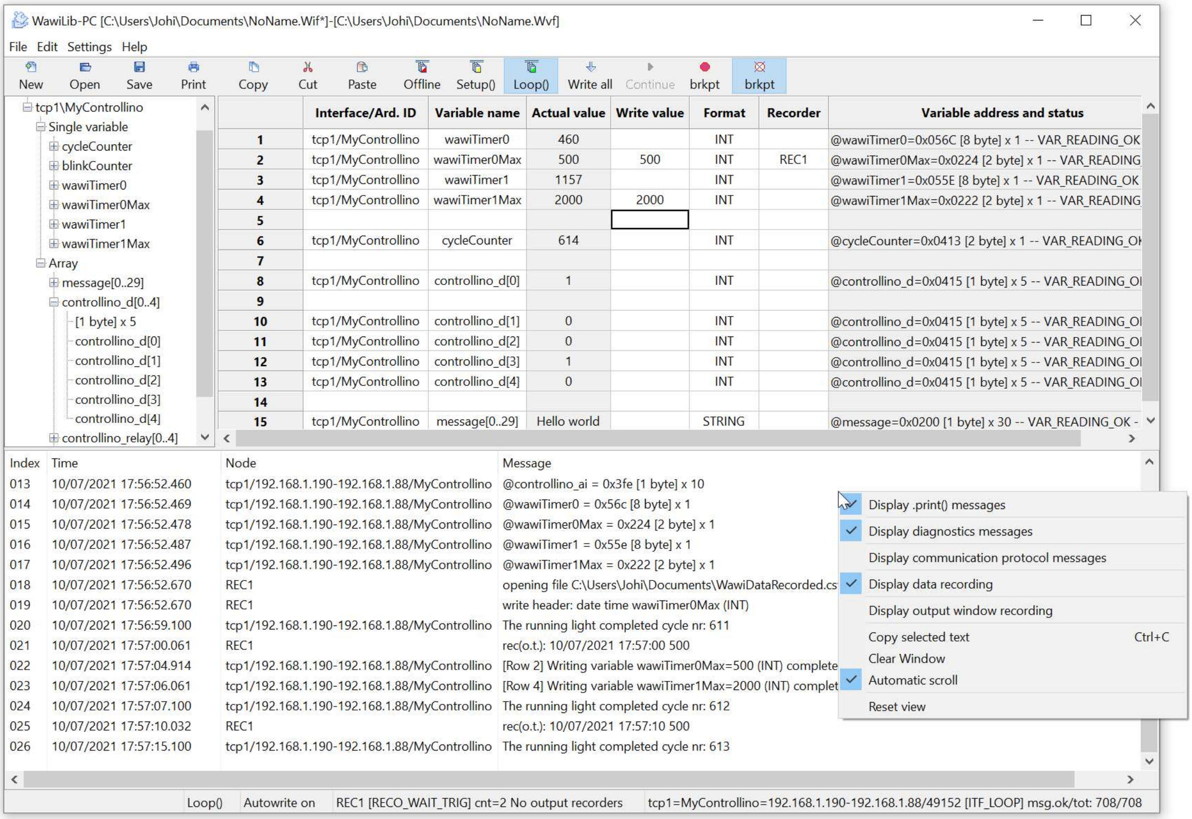
```
Index Time                     Node    Message
000   7/07/2021 21:31:46.887  Welco
001   7/07/2021 21:31:46.891
002   7/07/2021 21:31:47.043  Licens                  021- 8-14
003   7/07/2021 21:31:47.046  Licens
004   7/07/2021 21:31:47.048  Licens
005   7/07/2021 21:31:47.051  Licens
006   7/07/2021 21:31:47.054  Licens
007   7/07/2021 21:31:47.056  Licens                  yes
008   7/07/2021 21:31:47.059  Licens
009   7/07/2021 21:31:47.062  Licens
010   7/07/2021 21:31:47.065
```

Fig. 5.3. WawiLib output window options.

In the figure above, you see the popup menu where you can enable and disable different tracing settings.

- Display .print() messages: display the output of WawiSrv.print() messages used in your sketch for diagnostics and other purposes.
- Display diagnostics messages: display the output of general WawiLib diagnostics messages.
- Display communication protocol messages: display the communication messages as they are exchanged between the PC and the Arduino board.
- Display data recording: display the data written to disk by the data recorders (log variables).
- Display output recording: display the data written to disk by the output recorders (log .println() output).
- Automatic scroll: If activated, WawiLib will automatically scroll to the latest message in the output window every time a new message arrives.

The image above gives an incomplete overview of the various fields. Therefor I will use a more extended case for the bottom status line. This is the output of the WawiDemoControllinoTcpCable demo also included with WawiLib. The demo uses an Ethernet TCP interface on an Controllino Arduino Mega 2560 compatible PLC with generic WS5100/5500 Ethernet connection.



| | Interface/Ard. ID | Variable name | Actual value | Write value | Format | Recorder | Variable address and status |
|---|---|---|---|---|---|---|---|
| 1 | tcp1/MyControllino | wawiTimer0 | 460 | | INT | | @wawiTimer0=0x056C [8 byte] x 1 -- VAR_READING_OK |
| 2 | tcp1/MyControllino | wawiTimer0Max | 500 | 500 | INT | REC1 | @wawiTimer0Max=0x0224 [2 byte] x 1 -- VAR_READING |
| 3 | tcp1/MyControllino | wawiTimer1 | 1157 | | INT | | @wawiTimer1=0x055E [8 byte] x 1 -- VAR_READING_OK |
| 4 | tcp1/MyControllino | wawiTimer1Max | 2000 | 2000 | INT | | @wawiTimer1Max=0x0222 [2 byte] x 1 -- VAR_READING |
| 5 | | | | | | | |
| 6 | tcp1/MyControllino | cycleCounter | 614 | | INT | | @cycleCounter=0x0413 [2 byte] x 1 -- VAR_READING_Ol |
| 7 | | | | | | | |
| 8 | tcp1/MyControllino | controllino_d[0] | 1 | | INT | | @controllino_d=0x0415 [1 byte] x 5 -- VAR_READING_OI |
| 9 | | | | | | | |
| 10 | tcp1/MyControllino | controllino_d[1] | 0 | | INT | | @controllino_d=0x0415 [1 byte] x 5 -- VAR_READING_OI |
| 11 | tcp1/MyControllino | controllino_d[2] | 0 | | INT | | @controllino_d=0x0415 [1 byte] x 5 -- VAR_READING_OI |
| 12 | tcp1/MyControllino | controllino_d[3] | 1 | | INT | | @controllino_d=0x0415 [1 byte] x 5 -- VAR_READING_OI |
| 13 | tcp1/MyControllino | controllino_d[4] | 0 | | INT | | @controllino_d=0x0415 [1 byte] x 5 -- VAR_READING_OI |
| 14 | | | | | | | |
| 15 | tcp1/MyControllino | message[0..29] | Hello world | | STRING | | @message=0x0200 [1 byte] x 30 -- VAR_READING_OK - |

```
Index  Time                     Node                                          Message
013    10/07/2021 17:56:52.460  tcp1/192.168.1.190-192.168.1.88/MyControllino @controllino_ai = 0x3fe [1 byte] x 10
014    10/07/2021 17:56:52.469  tcp1/192.168.1.190-192.168.1.88/MyControllino @wawiTimer0 = 0x56c [8 byte] x 1
015    10/07/2021 17:56:52.478  tcp1/192.168.1.190-192.168.1.88/MyControllino @wawiTimer0Max = 0x224 [2 byte] x 1
016    10/07/2021 17:56:52.487  tcp1/192.168.1.190-192.168.1.88/MyControllino @wawiTimer1 = 0x55e [8 byte] x 1
017    10/07/2021 17:56:52.496  tcp1/192.168.1.190-192.168.1.88/MyControllino @wawiTimer1Max = 0x222 [2 byte] x 1
018    10/07/2021 17:56:52.670  REC1                                          opening file C:\Users\Johi\Documents\WawiDataRecorded.cs
019    10/07/2021 17:56:52.670  REC1                                          write header: date time wawiTimer0Max (INT)
020    10/07/2021 17:56:59.100  tcp1/192.168.1.190-192.168.1.88/MyControllino The running light completed cycle nr: 611
021    10/07/2021 17:57:00.061  REC1                                          rec(o.t.): 10/07/2021 17:57:00 500
022    10/07/2021 17:57:04.914  tcp1/192.168.1.190-192.168.1.88/MyControllino [Row 2] Writing variable wawiTimer0Max=500 (INT) complete
023    10/07/2021 17:57:06.061  tcp1/192.168.1.190-192.168.1.88/MyControllino [Row 4] Writing variable wawiTimer1Max=2000 (INT) complet
024    10/07/2021 17:57:07.100  tcp1/192.168.1.190-192.168.1.88/MyControllino The running light completed cycle nr: 612
025    10/07/2021 17:57:10.032  REC1                                          rec(o.t.): 10/07/2021 17:57:10 500
026    10/07/2021 17:57:15.100  tcp1/192.168.1.190-192.168.1.88/MyControllino The running light completed cycle nr: 613
```

```
Loop()   Autowrite on   REC1 [RECO_WAIT_TRIG] cnt=2 No output recorders   tcp1=MyControllino=192.168.1.190-192.168.1.88/49152 [ITF_LOOP] msg.ok/tot: 708/708
```

fig 5.4. WawiLib overview in a Controllino TCP configuration.

At the bottom of the WawiLib window there is a status line indicating the statuses of the application. The line is subdivided in different fields. I will describe the various fields using the example as displayed:

- “Loop()”: the target status of the communication interfaces {“Offline”, “Setup”, “Loop” } Setup()=Arduino is executing setup function, Loop()=Arduino is executing Loop() function. Note: Variable exchange is only available in Loop() mode, .print() is available in Setup() and in Loop() modes.
- “Autowrite on”: status Autowrite (See above; “ENTER” key triggers a variable write command for the line in the grid with the selected cell.)
- “REC1 [RECO_WAIT_TRIG] cnt=2”: the status name of the recorder named REC1, its FSM (finite state machine) status (=no tags selected for recording). The actual number of data records written to disk or memory (memory for excel .XLM file format) is 2.
- “No output recorders”: WawiLib can record .print() output from the sketch into an output file. In this case no recorders for this kind of data are defined.
- “TCP=MyControllino=192.168.1.190-192.168.1.88/49152 [ITF_LOOP]”: An interface of type TCP is active. The library was initialized (WawiSrv.begin() function) with parameter value “MyControllino” for the name of the board. The interface card on the PC has IP 192.168.1.168 and the Arduino board has IP 192.168.1.88. TCP port 49152 is used on the Arduino/Controllino side. The actual status of the communication FSM (finite state machine) is ITF_LOOP.

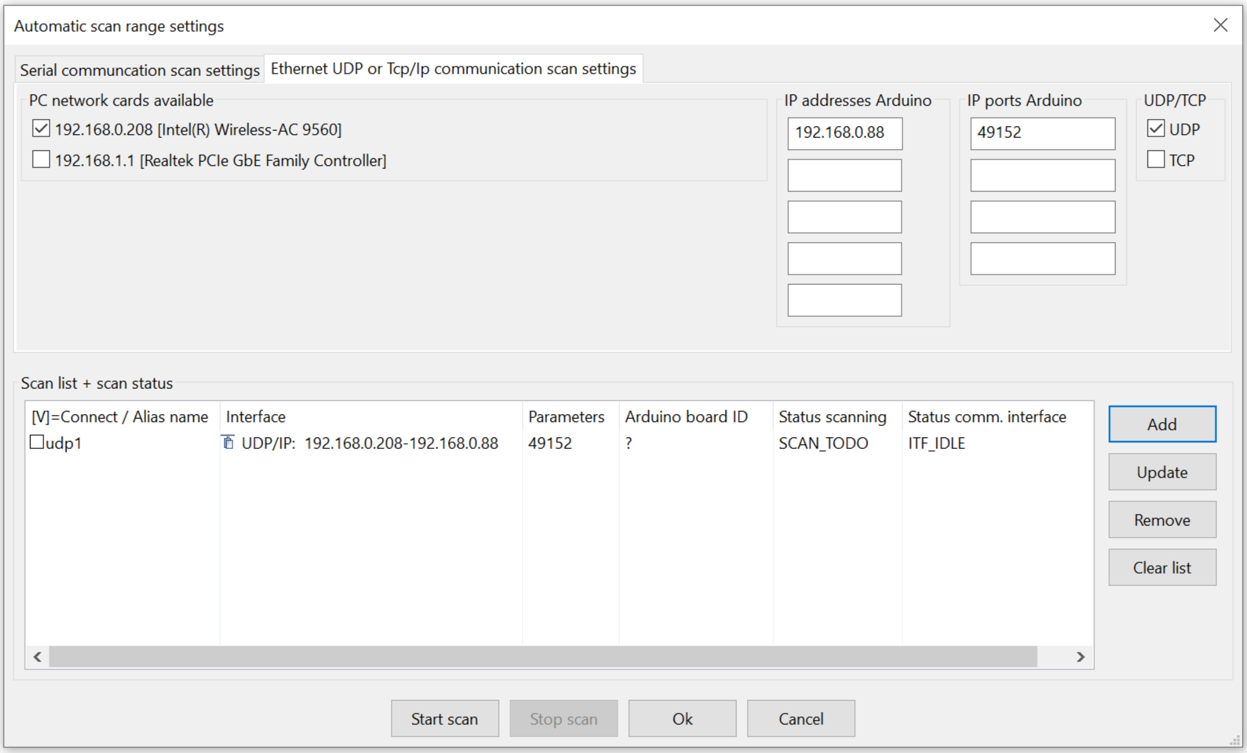
- “Msg.ok/.tot 708/708: There are 708 data telegrams exchanged OK between the Arduino on a total of 708 telegrams.

WawiLib supports multiple interfaces of multiple boards and multiple data recorders at the same time. Therefor the fields “TCP1[…]” and “REC1[…]” display the various recorders and various interfaces one after the other in an alternating way.

# 6. WawiLib communication link setup

One of the biggest challenges going online is to find the right port and the right settings. With this purpose in mind, WawiLib has a wizard to scan Ethernet IP/UDP ports with various settings to check for the presence of an Arduino board on the network.

- ✓ Open the automatic scan range settings dialog via “Settings\Communication interfaces”.
- ✓ Select the tab “Ethernet UDP or TCP/IP communication scan settings”.



Automatic scan range settings
Serial communcation scan settings
Ethernet UDP or Tcp/Ip communication scan settings
Add
Update
Remove
Clear list
Start scan
Stop scan
Ok
Cancel

PC network cards available

☑ 192.168.0.208 [Intel(R) Wireless-AC 9560]

☐ 192.168.1.1 [Realtek PCIe GbE Family Controller]

IP addresses Arduino: 192.168.0.88

IP ports Arduino: 49152

UDP/TCP: ☑ UDP ☐ TCP

Scan list + scan status

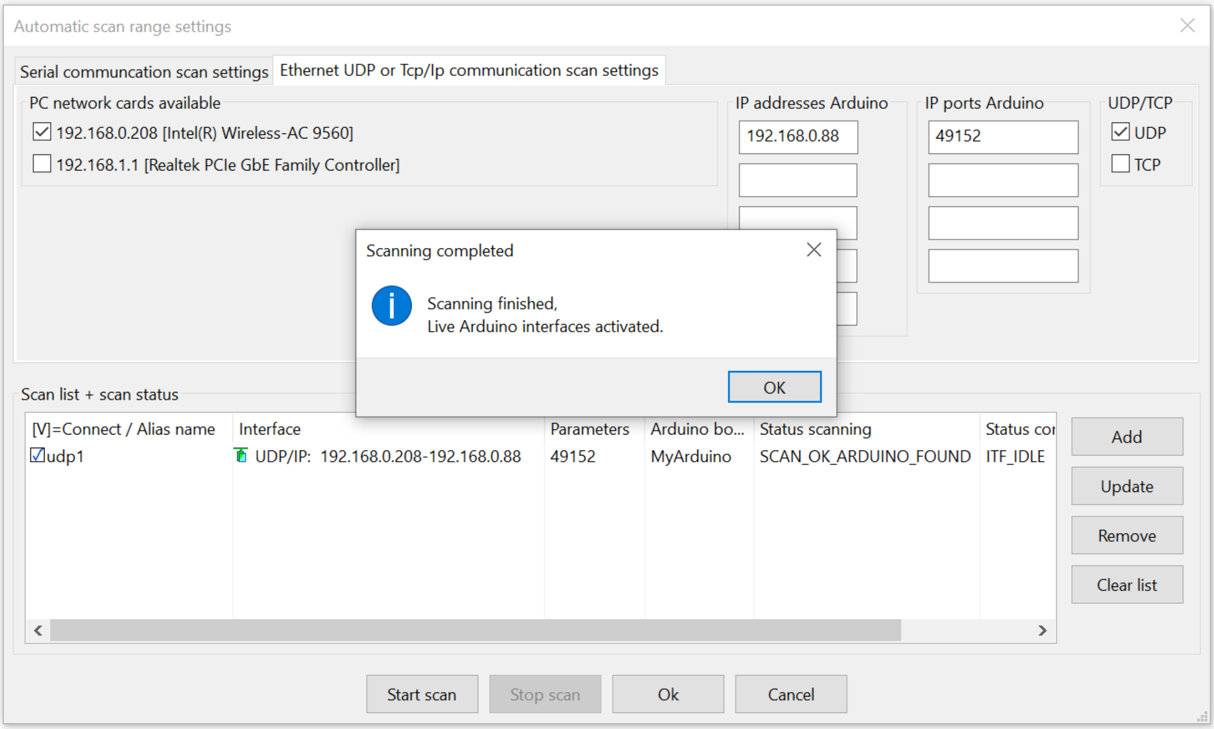
| [V]=Connect / Alias name | Interface | Parameters | Arduino board ID | Status scanning | Status comm. interface |
|---|---|---|---|---|---|
| ☐udp1 | UDP/IP: 192.168.0.208-192.168.0.88 | 49152 | ? | SCAN_TODO | ITF_IDLE |

Fig. 6.1. Ethernet communication interface setup: scanning/setting up the connection.

- ✓ Check the PC network cards (adapters) to be used to look for the Arduino board interface.
- ✓ Fill in the IP address and the IP port as indicated above (determined in the previous section).
- ✓ Check “UDP” as protocol.
- ✓ Press “Add”.
- ✓ Press “Start scan”.
- ✓ Wait for the scan to complete.

⇨ WawiLib will scan the network to discover Arduino boards on the net using the selected parameter(s). You can fill in multiple fields at one time, ‘Add’ will add all possible combinations of the parameters to the list box.

⇨ If a connection was successfully identified, it will be shown with a green label in the ‘interface’ interface, if not, the label will become red after scanning.



Automatic scan range settings

Serial communcation scan settings | Ethernet UDP or Tcp/Ip communication scan settings

PC network cards available

☑ 192.168.0.208 [Intel(R) Wireless-AC 9560]

☐ 192.168.1.1 [Realtek PCIe GbE Family Controller]

IP addresses Arduino: 192.168.0.88

IP ports Arduino: 49152

UDP/TCP: ☑ UDP ☐ TCP

Scanning completed

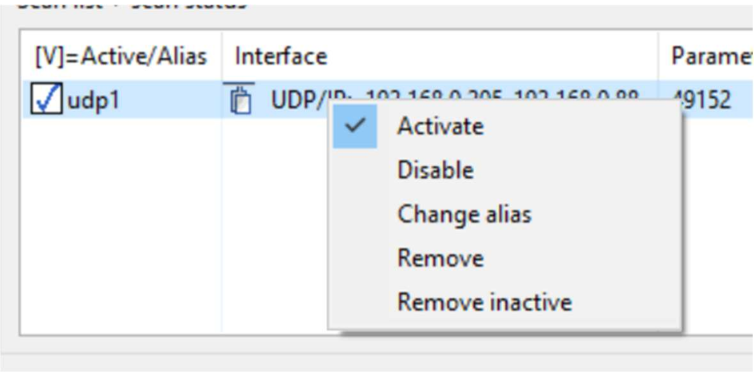
Scanning finished,
Live Arduino interfaces activated.

Scan list + scan status

| [V]=Connect / Alias name | Interface | Parameters | Arduino bo... | Status scanning | Status coı |
|---|---|---|---|---|---|
| ☑udp1 | UDP/IP: 192.168.0.208-192.168.0.88 | 49152 | MyArduino | SCAN_OK_ARDUINO_FOUND | ITF_IDLE |

Fig. 6.3. Ethernet communication setup indicating successful link check.

⇨ Click right on the mouse to make a pop-up menu appear:



| [V]=Active/Alias | Interface | Parame |
|---|---|---|
| ☑udp1 | UDP/IP: | 49152 |

Fig. 6.4. Modify the status of a connection without or after scanning.

- Activate/Disable: makes this choice available in the main window (checked automatically after successful interface scan check).
- Change alias: change the connection name (UDP1 in this case).
- Remove inactive: remove all combinations that were not scanned successfully.

If all goes as planned, you should see a green icon indicating that the link was successfully verified. If the connection fails, you might want to re-try with your virus scanner disabled or with your Windows Firewall disabled. Checking for a very short period of time without firewall protection to identify the cause of the problem is relatively safe. Do not operate your PC without these safeties active for a longer period.

If the connection is OK but fails with the firewall activated, you need to change your firewall settings in order to let the traffic for WawiLib trough while blocking off other rogue network traffic.

- ✓ Press OK.

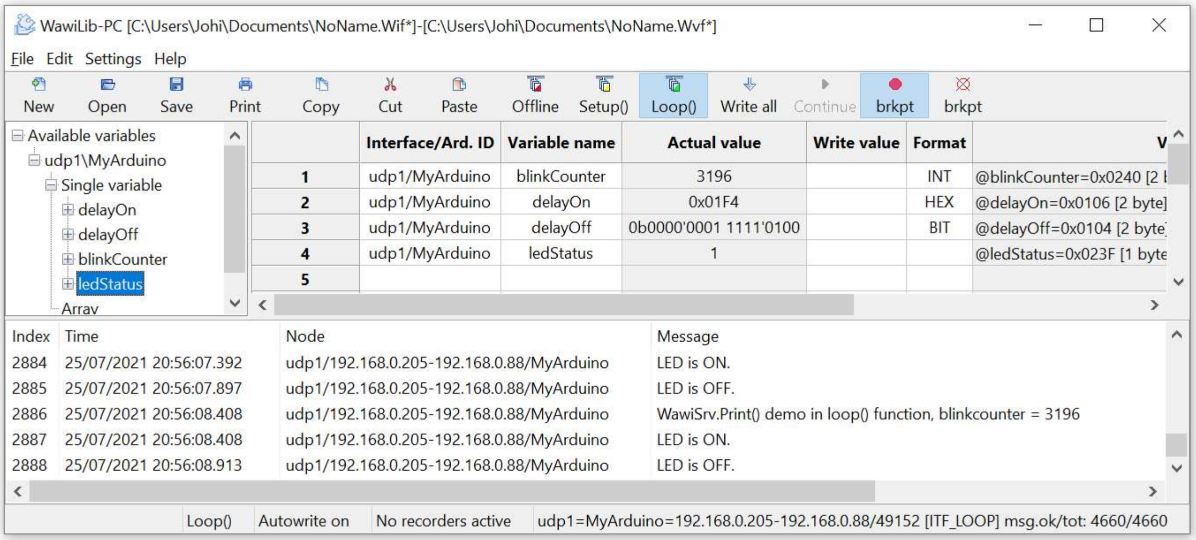
At this point the connection parameters are identified. The link will be effectively established once you press “Setup()” in the main window. WawiLib supports multiple boards and multiple connections at the same time. Serial, Ethernet and Wi-Fi connections can be operated together.

There is an obsolete Arduino Wi-Fi shield designed for the Uno, Mega and Due. Some of you might still have this type of shield. This shield has the bizarre property to send back a UDP answer message from a different port then the port where the message was received. You will see that many firewalls have issues with this particular type of behavior. The MKR1000 and MKR1010 nor the USB Wi-Fi Rev2 have this issue.

If you would want to use TCP instead of UDP, this is possible but I advise strongly against this. The performance of TCP on the Arduino shield is much slower compared to UDP. With UDP I get response times < 30ms, with TCP <300ms. The risk of UDP loosing frames is handled by the WawiWifi protocol itself at a higher level.

# 7. Read and write variables with WawiLib

## 7.1. Watch variables



| | Interface/Ard. ID | Variable name | Actual value | Write value | Format | V |
|---|---|---|---|---|---|---|
| 1 | udp1/MyArduino | blinkCounter | 3196 | | INT | @blinkCounter=0x0240 [2 b |
| 2 | udp1/MyArduino | delayOn | 0x01F4 | | HEX | @delayOn=0x0106 [2 byte] |
| 3 | udp1/MyArduino | delayOff | 0b0000'0001 1111'0100 | | BIT | @delayOff=0x0104 [2 byte |
| 4 | udp1/MyArduino | ledStatus | 1 | | | @ledStatus=0x023F [1 byte |
| 5 | | | | | | |

```
Index  Time                     Node                                        Message
2884   25/07/2021 20:56:07.392  udp1/192.168.0.205-192.168.0.88/MyArduino   LED is ON.
2885   25/07/2021 20:56:07.897  udp1/192.168.0.205-192.168.0.88/MyArduino   LED is OFF.
2886   25/07/2021 20:56:08.408  udp1/192.168.0.205-192.168.0.88/MyArduino   WawiSrv.Print() demo in loop() function, blinkcounter = 3196
2887   25/07/2021 20:56:08.408  udp1/192.168.0.205-192.168.0.88/MyArduino   LED is ON.
2888   25/07/2021 20:56:08.913  udp1/192.168.0.205-192.168.0.88/MyArduino   LED is OFF.
```

Loop() Autowrite on No recorders active udp1=MyArduino=192.168.0.205-192.168.0.88/49152 [ITF_LOOP] msg.ok/tot: 4660/4660

Fig. 7.1. Add variables to the grid using drag & drop.

- ✓ Go online (press Setup()) on the top toolbar.
- ✓ Drag the variables *blinkCounter* and *delayOn* from the tree control to the grid.
- ✓ Alternative: enter the names of the variables of interest in the grid.
- ✓ Modify the display format as indicated in fig. 7.1.
- ⇨ The “Interface/Arduino ID” column will be filled in automatically as there is only 1 board active.

(You can also click right and select “Available interfaces”. All configured links can be selected using this menu. This option is necessary if you want to exchange data with multiple boards.)

## 7.2. Modify variables

- ✓ Fill in 100 as new value for *delayOn* in the write column.
- ✓ Press “Write all”.
- ✓ Click right mouse on the output window and enable display messages as below:

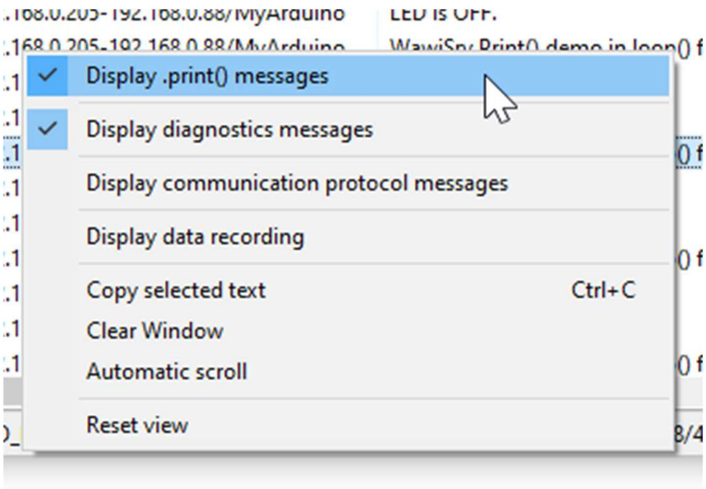
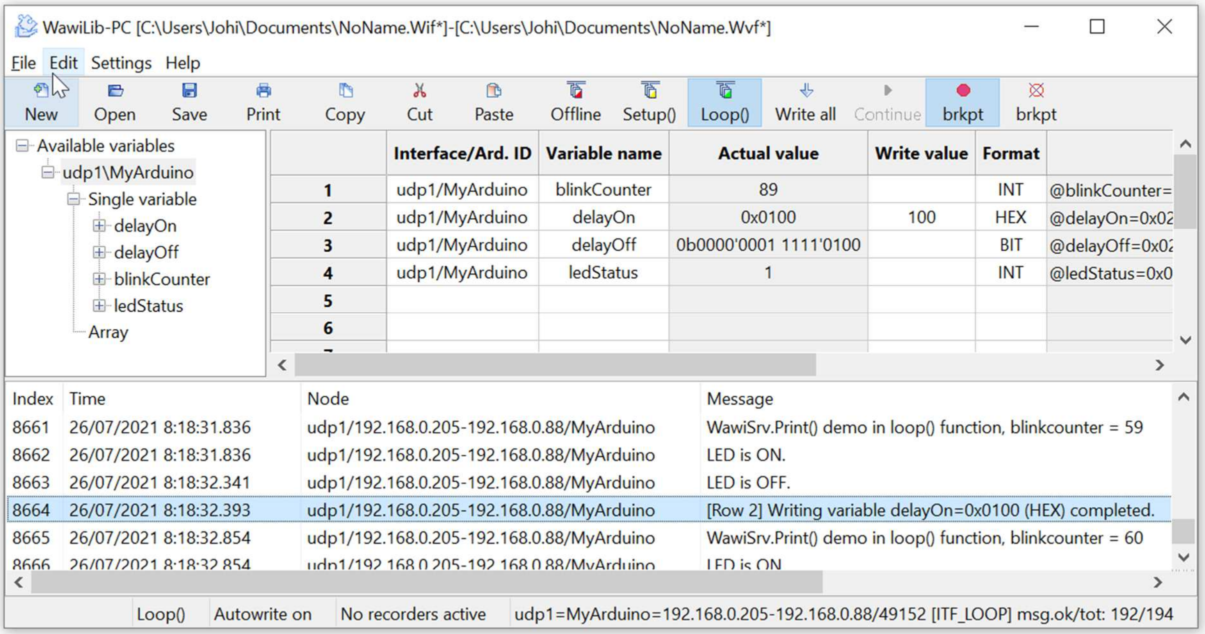

.168.0.205-192.168.0.88/MyArduino
LED is OFF.
.168.0.205-192.168.0.88/MyArduino
WawiSrv.Print() demo in loop() f
✓ Display .print() messages
✓ Display diagnostics messages
Display communication protocol messages
Display data recording
Copy selected text Ctrl+C
Clear Window
Automatic scroll
Reset view

Fig. 7.3. The serial output window.

You should see the actual value of *delayOn* change to 100. The time the LED is on will change to 100ms.



| | Interface/Ard. ID | Variable name | Actual value | Write value | Format | |
|---|---|---|---|---|---|---|
| 1 | udp1/MyArduino | blinkCounter | 89 | | INT | @blinkCounter= |
| 2 | udp1/MyArduino | delayOn | 0x0100 | 100 | HEX | @delayOn=0x02 |
| 3 | udp1/MyArduino | delayOff | 0b0000'0001 1111'0100 | | BIT | @delayOff=0x02 |
| 4 | udp1/MyArduino | ledStatus | 1 | | INT | @ledStatus=0x0 |
| 5 | | | | | | |
| 6 | | | | | | |

| Index | Time | Node | Message |
|---|---|---|---|
| 8661 | 26/07/2021 8:18:31.836 | udp1/192.168.0.205-192.168.0.88/MyArduino | WawiSrv.Print() demo in loop() function, blinkcounter = 59 |
| 8662 | 26/07/2021 8:18:31.836 | udp1/192.168.0.205-192.168.0.88/MyArduino | LED is ON. |
| 8663 | 26/07/2021 8:18:32.341 | udp1/192.168.0.205-192.168.0.88/MyArduino | LED is OFF. |
| 8664 | 26/07/2021 8:18:32.393 | udp1/192.168.0.205-192.168.0.88/MyArduino | [Row 2] Writing variable delayOn=0x0100 (HEX) completed. |
| 8665 | 26/07/2021 8:18:32.854 | udp1/192.168.0.205-192.168.0.88/MyArduino | WawiSrv.Print() demo in loop() function, blinkcounter = 60 |
| 8666 | 26/07/2021 8:18:32.854 | udp1/192.168.0.205-192.168.0.88/MyArduino | LED is ON. |

Loop() Autowrite on No recorders active udp1=MyArduino=192.168.0.205-192.168.0.88/49152 [ITF_LOOP] msg.ok/tot: 192/194

Fig. 7.4. Change the value of delayOn to 100 (hex).

Look at the status bar at bottom of the window:

- WawiLib has changed to Loop() mode as the Sketch is executing its Loop() function.
- “Autowrite” is on.
- UDP1 corresponds with an Arduino named “MyArduino” (see begin(“…”) in Sketch).
- 192.168.0.205 is the PC address.
- 192.168.0.88 is the Arduino address.
- UDP port 49152 is used at Arduino side.
- The status of the interface is “ITF_LOOP ()”.
- 192 message exchanges between PC and Arduino have been executed ok.
- 194 message exchanges PC and Arduino have been executed in total.

In the upper window, you will see the actual value of the variables. In the bottom window, you will see the communication telegrams that are exchanged with your Arduino board (provided tracing is enabled).

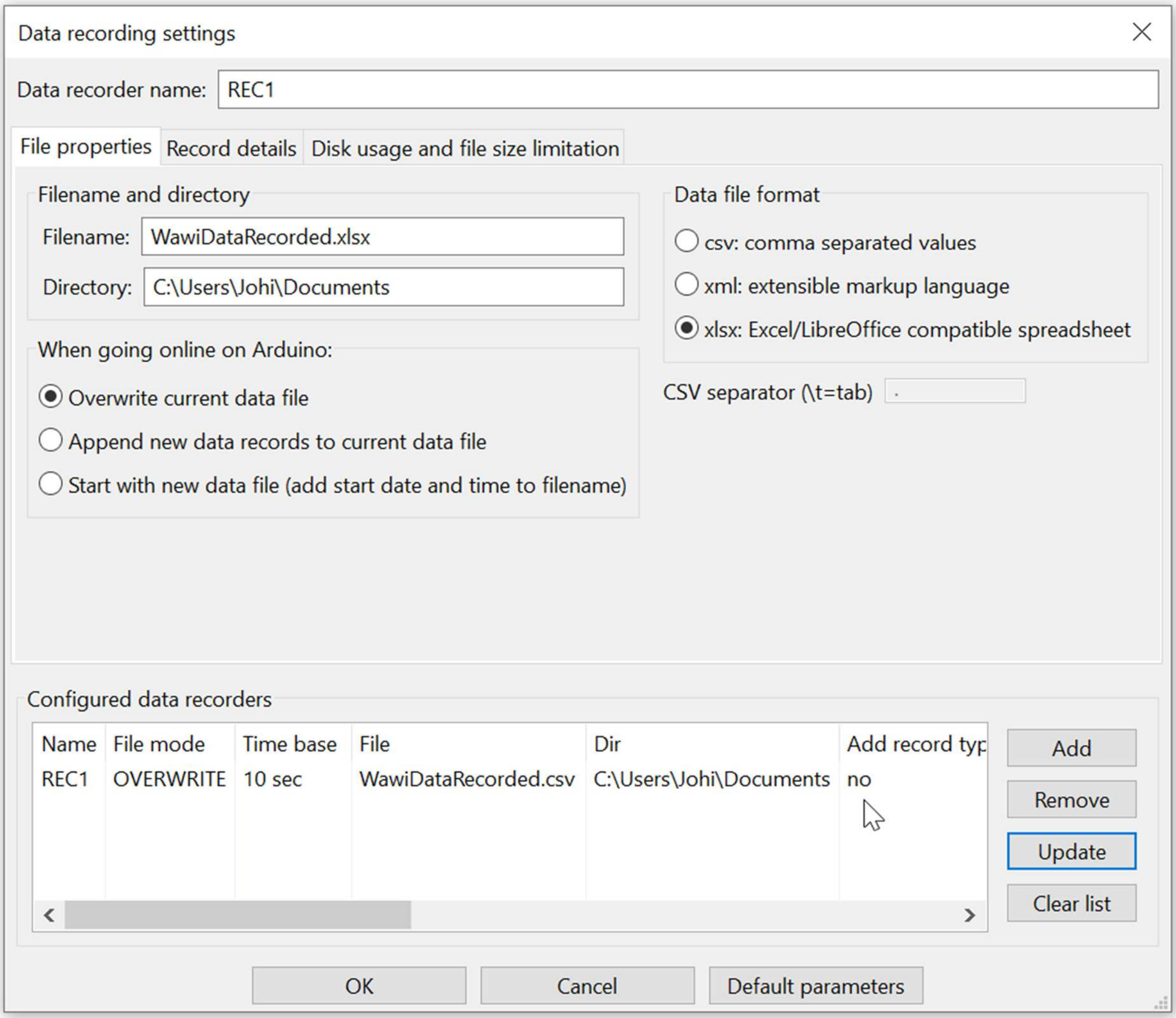
Look at the output window:

⇨ On line 8664 you see feedback of your write operation.
⇨ On the other lines you see the result of the WawiSrv.println(); statements in the sketch.

# 8. Record variables with WawiLib (introduction)

In this section, we will configure a data recorder to record the values of our parameters in an .xlsx file that is compatible with Microsoft Excel or LibreOffice calc.

- ✓ Select the menu “Settings/Data Recording”
- ✓ Select “Overwrite current data file”
- ✓ “xlsx: Excel/LibreOffice compatible spreadsheet” (and fill in the fields as indicated below)
- ✓ Press “Add”



Data recording settings

Data recorder name: REC1

Filename and directory

Filename: WawiDataRecorded.xlsx

Directory: C:\Users\Johi\Documents

When going online on Arduino:

● Overwrite current data file
○ Append new data records to current data file
○ Start with new data file (add start date and time to filename)

Data file format

○ csv: comma separated values
○ xml: extensible markup language
● xlsx: Excel/LibreOffice compatible spreadsheet

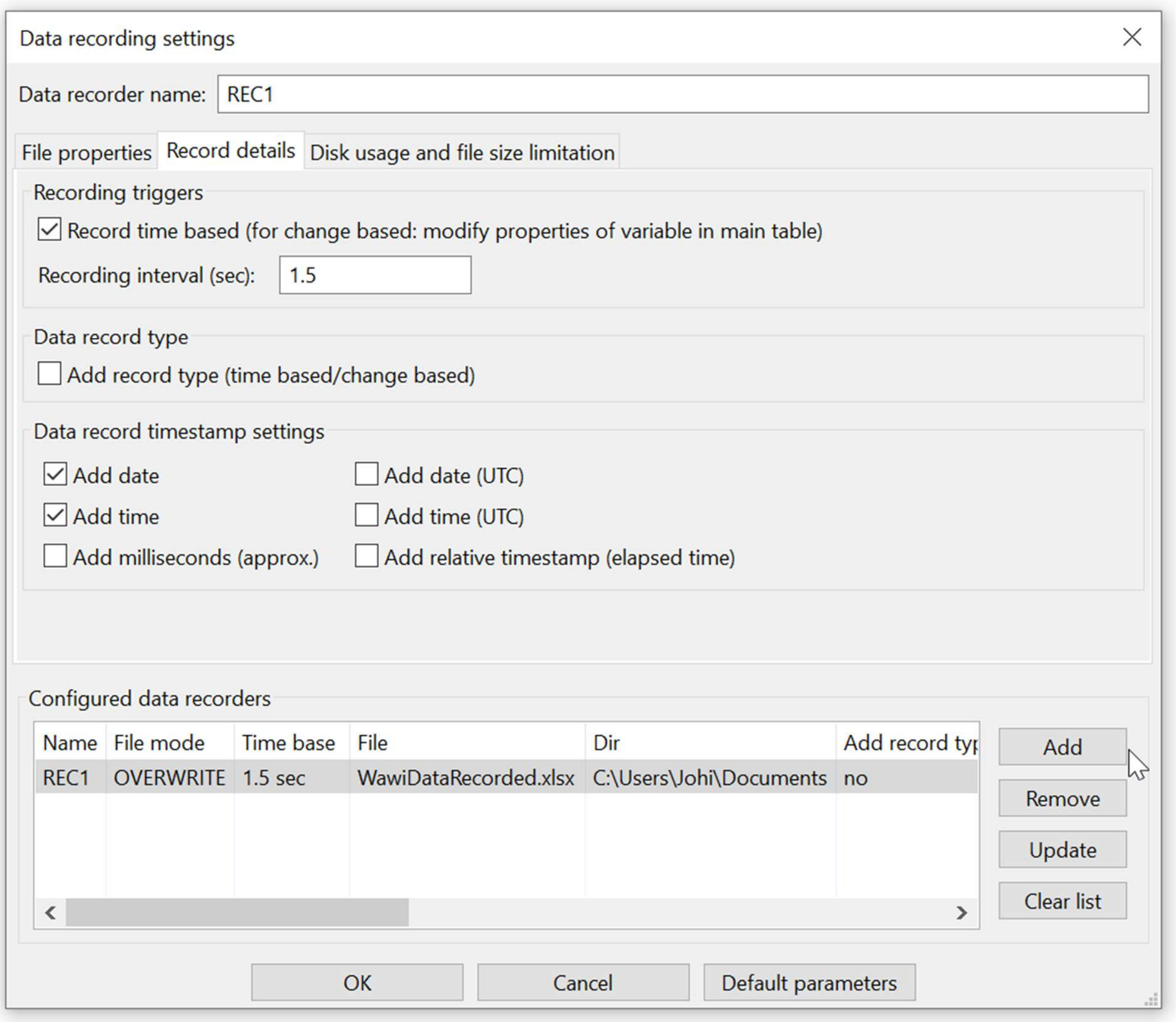
CSV separator (\t=tab) .

Configured data recorders

| Name | File mode | Time base | File | Dir | Add record typ |
|---|---|---|---|---|---|
| REC1 | OVERWRITE | 10 sec | WawiDataRecorded.csv | C:\Users\Johi\Documents | no |

Fig. 8.1. The data recorder (for variables) setup dialog, main tab.

- ✓ Select the tab “Record Details” (fig. 8.2)
- ✓ Click on the line “REC1” in the list box.
- ✓ Enter 1.5 for the “recording interval (sec)” parameter field.
- ✓ Press “Update”



Data recording settings

Data recorder name: REC1

Recording triggers

☑ Record time based (for change based: modify properties of variable in main table)

Recording interval (sec): 1.5

Data record type

□ Add record type (time based/change based)

Data record timestamp settings

☑ Add date
□ Add date (UTC)
☑ Add time
□ Add time (UTC)
□ Add milliseconds (approx.)
□ Add relative timestamp (elapsed time)

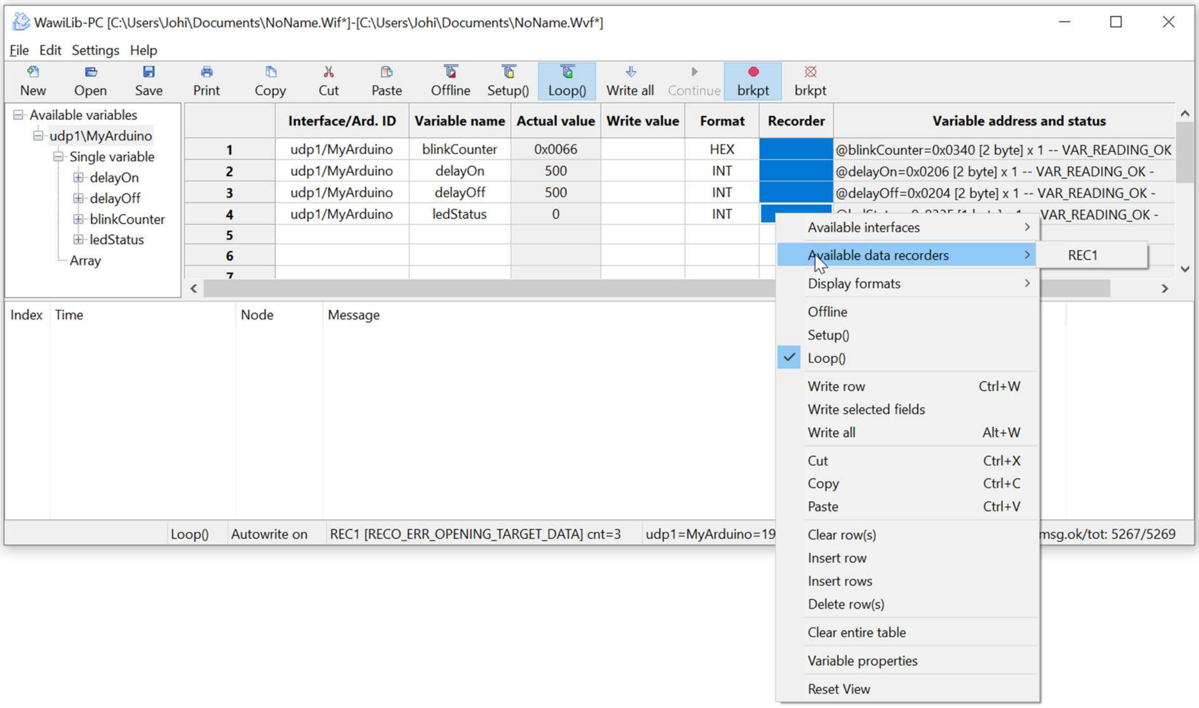
Configured data recorders

| Name | File mode | Time base | File | Dir | Add record typ |
|---|---|---|---|---|---|
| REC1 | OVERWRITE | 1.5 sec | WawiDataRecorded.xlsx | C:\Users\Johi\Documents | no |

Fig. 8.2. The data recorder setup dialog, timing = 1.5 sec.

This will create a data recorder in line with your actual settings.

- ✓ Press “OK”.
- ✓ Fill in the table as in fig. 8.3.
- ✓ Select all grid fields linked to variables in the recorder column.
- ✓ Click right
- ✓ “Available data recorders/Rec1”.
- ✓ Note: You can also enter “REC1” in the fields for the variables in the Recorder column.

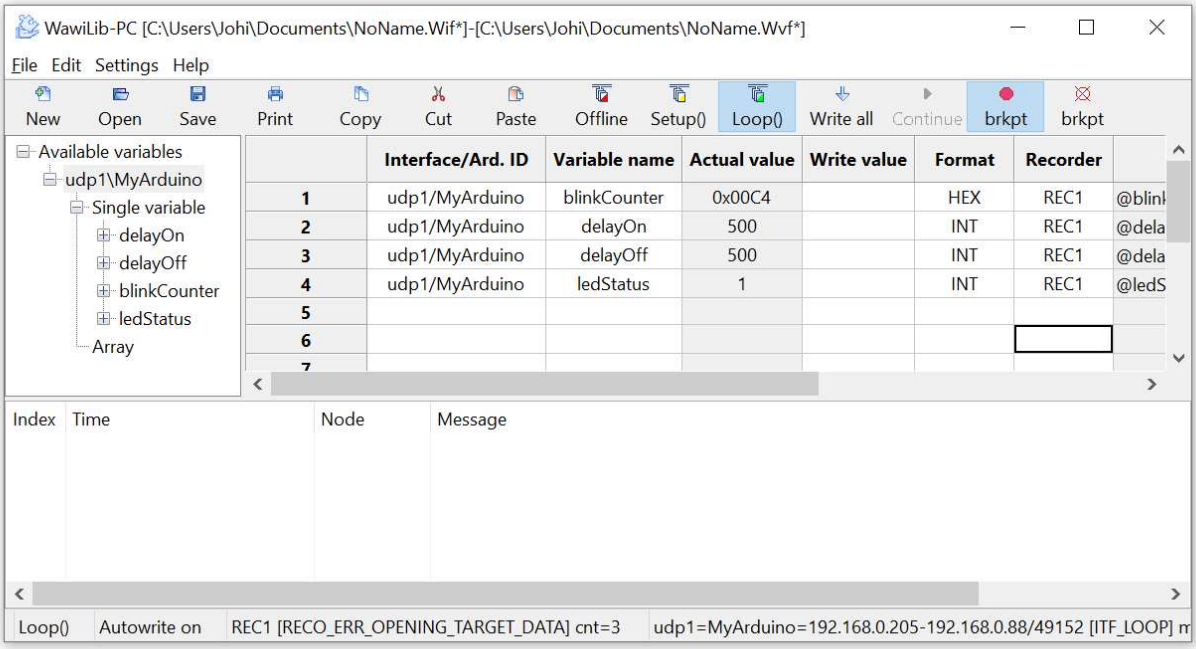


| | Interface/Ard. ID | Variable name | Actual value | Write value | Format | Recorder | Variable address and status |
|---|---|---|---|---|---|---|---|
| 1 | udp1/MyArduino | blinkCounter | 0x0066 | | HEX | | @blinkCounter=0x0340 [2 byte] x 1 -- VAR_READING_OK |
| 2 | udp1/MyArduino | delayOn | 500 | | INT | | @delayOn=0x0206 [2 byte] x 1 -- VAR_READING_OK - |
| 3 | udp1/MyArduino | delayOff | 500 | | INT | | @delayOff=0x0204 [2 byte] x 1 -- VAR_READING_OK - |
| 4 | udp1/MyArduino | ledStatus | 0 | | INT | | [illegible] VAR_READING_OK - |
| 5 | | | | | | | |
| 6 | | | | | | | |

Index Time Node Message

Loop() Autowrite on REC1 [RECO_ERR_OPENING_TARGET_DATA] cnt=3 udp1=MyArduino=19 msg.ok/tot: 5267/5269

Fig. 8.3. Select a data recorder for all the variables.


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif*]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
udp1\MyArduino
Single variable
delayOn
delayOff
blinkCounter
ledStatus
Array

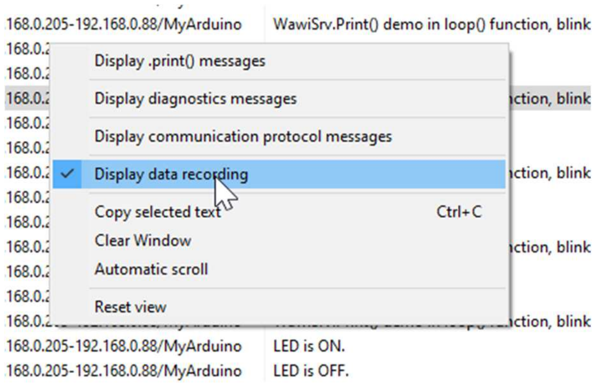
| | Interface/Ard. ID | Variable name | Actual value | Write value | Format | Recorder | |
|---|---|---|---|---|---|---|---|
| 1 | udp1/MyArduino | blinkCounter | 0x00C4 | | HEX | REC1 | @blinl |
| 2 | udp1/MyArduino | delayOn | 500 | | INT | REC1 | @dela |
| 3 | udp1/MyArduino | delayOff | 500 | | INT | REC1 | @dela |
| 4 | udp1/MyArduino | ledStatus | 1 | | INT | REC1 | @ledS |
| 5 | | | | | | | |
| 6 | | | | | | | |

Index Time Node Message

Loop() Autowrite on REC1 [RECO_ERR_OPENING_TARGET_DATA] cnt=3 udp1=MyArduino=192.168.0.205-192.168.0.88/49152 [ITF_LOOP] m

Fig. 8.4. Select a data recorder for all the variables: result.

- ✓ Disable all the options in the output window (right click in output window):
- ✓ Enable “Display data recording” (fig. 8.5.)

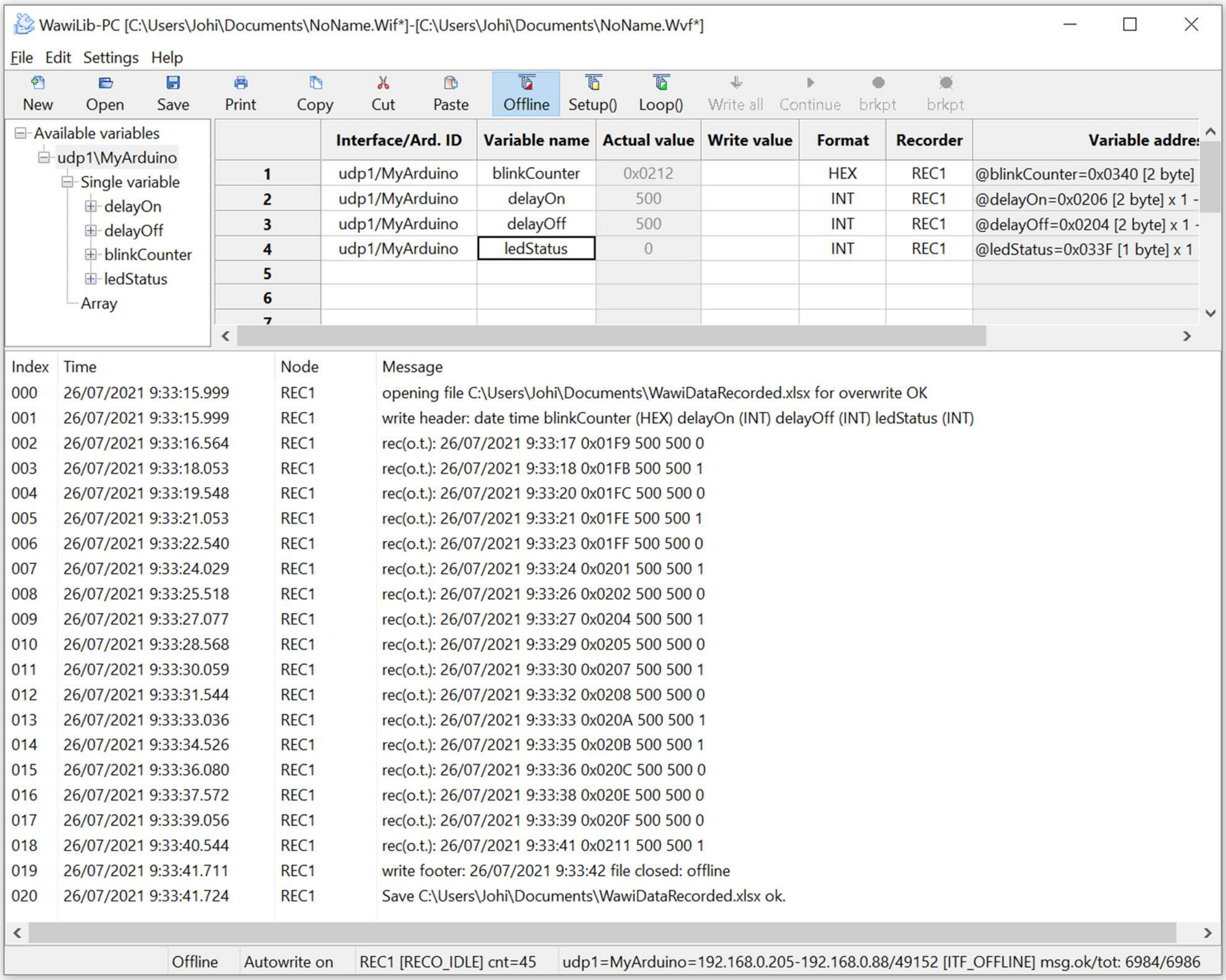


Display .print() messages
Display diagnostics messages
Display communication protocol messages
✓ Display data recording
Copy selected text Ctrl+C
Clear Window
Automatic scroll
Reset view

```
168.0.205-192.168.0.88/MyArduino    WawiSrv.Print() demo in loop() function, blink
168.0.205-192.168.0.88/MyArduino    LED is ON.
168.0.205-192.168.0.88/MyArduino    LED is OFF.
```

Fig. 8.5. Enable recording reporting in the output window.

- ✓ Press “Setup()”
- ✓ Wait 15 seconds.
- ⇨ You will now see the different values of the variables as they are written to the .xlsx file in the output window.
- ✓ Press “Offline”



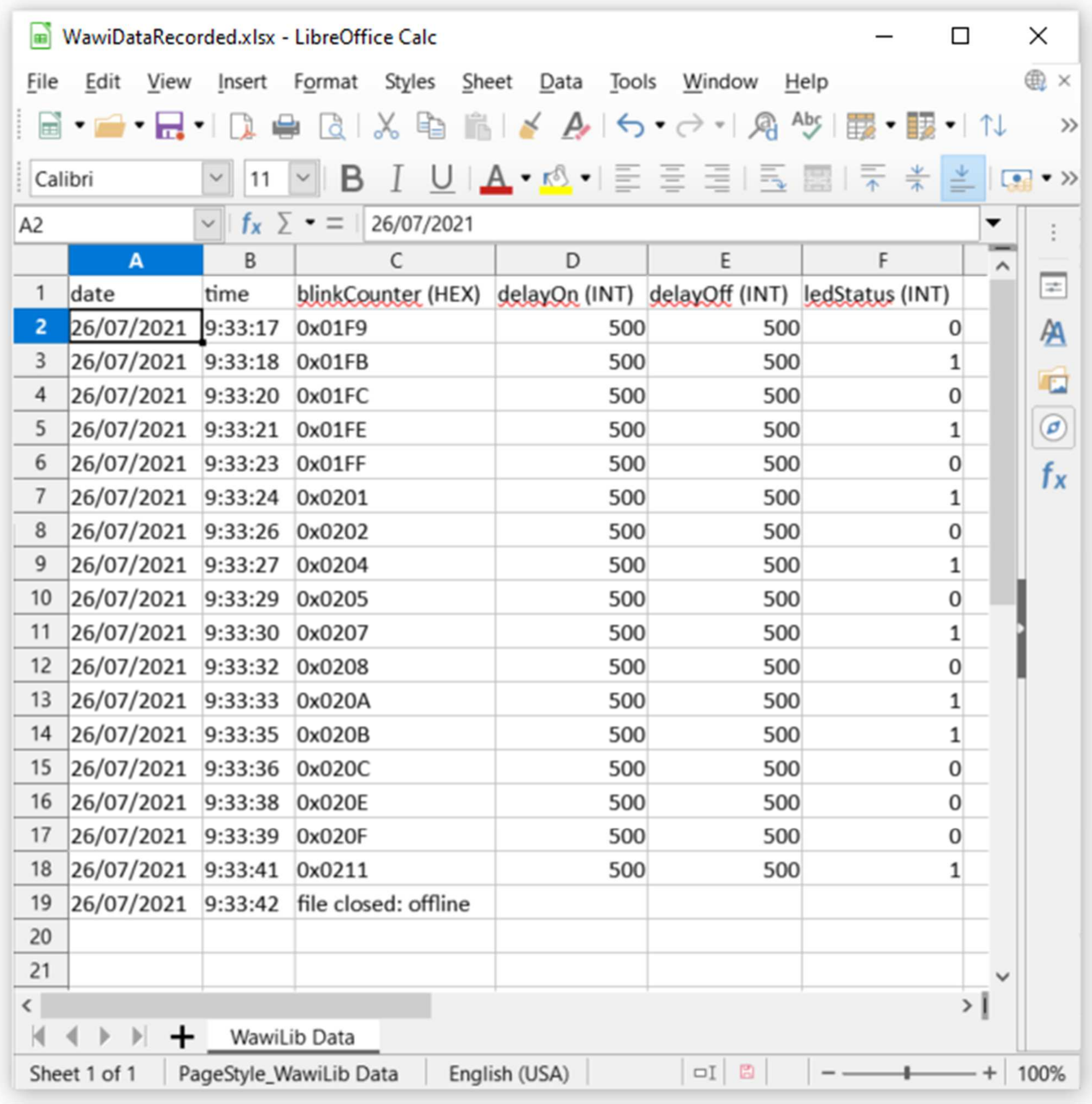
| | Interface/Ard. ID | Variable name | Actual value | Write value | Format | Recorder | Variable addre: |
|---|---|---|---|---|---|---|---|
| 1 | udp1/MyArduino | blinkCounter | 0x0212 | | HEX | REC1 | @blinkCounter=0x0340 [2 byte] |
| 2 | udp1/MyArduino | delayOn | 500 | | INT | REC1 | @delayOn=0x0206 [2 byte] x 1 - |
| 3 | udp1/MyArduino | delayOff | 500 | | INT | REC1 | @delayOff=0x0204 [2 byte] x 1 - |
| 4 | udp1/MyArduino | ledStatus | 0 | | INT | REC1 | @ledStatus=0x033F [1 byte] x 1 |
| 5 | | | | | | | |
| 6 | | | | | | | |

```
Index  Time                     Node   Message
000    26/07/2021 9:33:15.999   REC1   opening file C:\Users\Johi\Documents\WawiDataRecorded.xlsx for overwrite OK
001    26/07/2021 9:33:15.999   REC1   write header: date time blinkCounter (HEX) delayOn (INT) delayOff (INT) ledStatus (INT)
002    26/07/2021 9:33:16.564   REC1   rec(o.t.): 26/07/2021 9:33:17 0x01F9 500 500 0
003    26/07/2021 9:33:18.053   REC1   rec(o.t.): 26/07/2021 9:33:18 0x01FB 500 500 1
004    26/07/2021 9:33:19.548   REC1   rec(o.t.): 26/07/2021 9:33:20 0x01FC 500 500 0
005    26/07/2021 9:33:21.053   REC1   rec(o.t.): 26/07/2021 9:33:21 0x01FE 500 500 1
006    26/07/2021 9:33:22.540   REC1   rec(o.t.): 26/07/2021 9:33:23 0x01FF 500 500 0
007    26/07/2021 9:33:24.029   REC1   rec(o.t.): 26/07/2021 9:33:24 0x0201 500 500 1
008    26/07/2021 9:33:25.518   REC1   rec(o.t.): 26/07/2021 9:33:26 0x0202 500 500 0
009    26/07/2021 9:33:27.077   REC1   rec(o.t.): 26/07/2021 9:33:27 0x0204 500 500 1
010    26/07/2021 9:33:28.568   REC1   rec(o.t.): 26/07/2021 9:33:29 0x0205 500 500 0
011    26/07/2021 9:33:30.059   REC1   rec(o.t.): 26/07/2021 9:33:30 0x0207 500 500 1
012    26/07/2021 9:33:31.544   REC1   rec(o.t.): 26/07/2021 9:33:32 0x0208 500 500 0
013    26/07/2021 9:33:33.036   REC1   rec(o.t.): 26/07/2021 9:33:33 0x020A 500 500 1
014    26/07/2021 9:33:34.526   REC1   rec(o.t.): 26/07/2021 9:33:35 0x020B 500 500 1
015    26/07/2021 9:33:36.080   REC1   rec(o.t.): 26/07/2021 9:33:36 0x020C 500 500 0
016    26/07/2021 9:33:37.572   REC1   rec(o.t.): 26/07/2021 9:33:38 0x020E 500 500 0
017    26/07/2021 9:33:39.056   REC1   rec(o.t.): 26/07/2021 9:33:39 0x020F 500 500 0
018    26/07/2021 9:33:40.544   REC1   rec(o.t.): 26/07/2021 9:33:41 0x0211 500 500 1
019    26/07/2021 9:33:41.711   REC1   write footer: 26/07/2021 9:33:42 file closed: offline
020    26/07/2021 9:33:41.724   REC1   Save C:\Users\Johi\Documents\WawiDataRecorded.xlsx ok.
```

```
Offline   Autowrite on   REC1 [RECO_IDLE] cnt=45   udp1=MyArduino=192.168.0.205-192.168.0.88/49152 [ITF_OFFLINE] msg.ok/tot: 6984/6986
```

Fig. 8.6. The data recording mirrored in the output window.

- ✓ Open the recorded .xlsx file in LibreOffice calc:

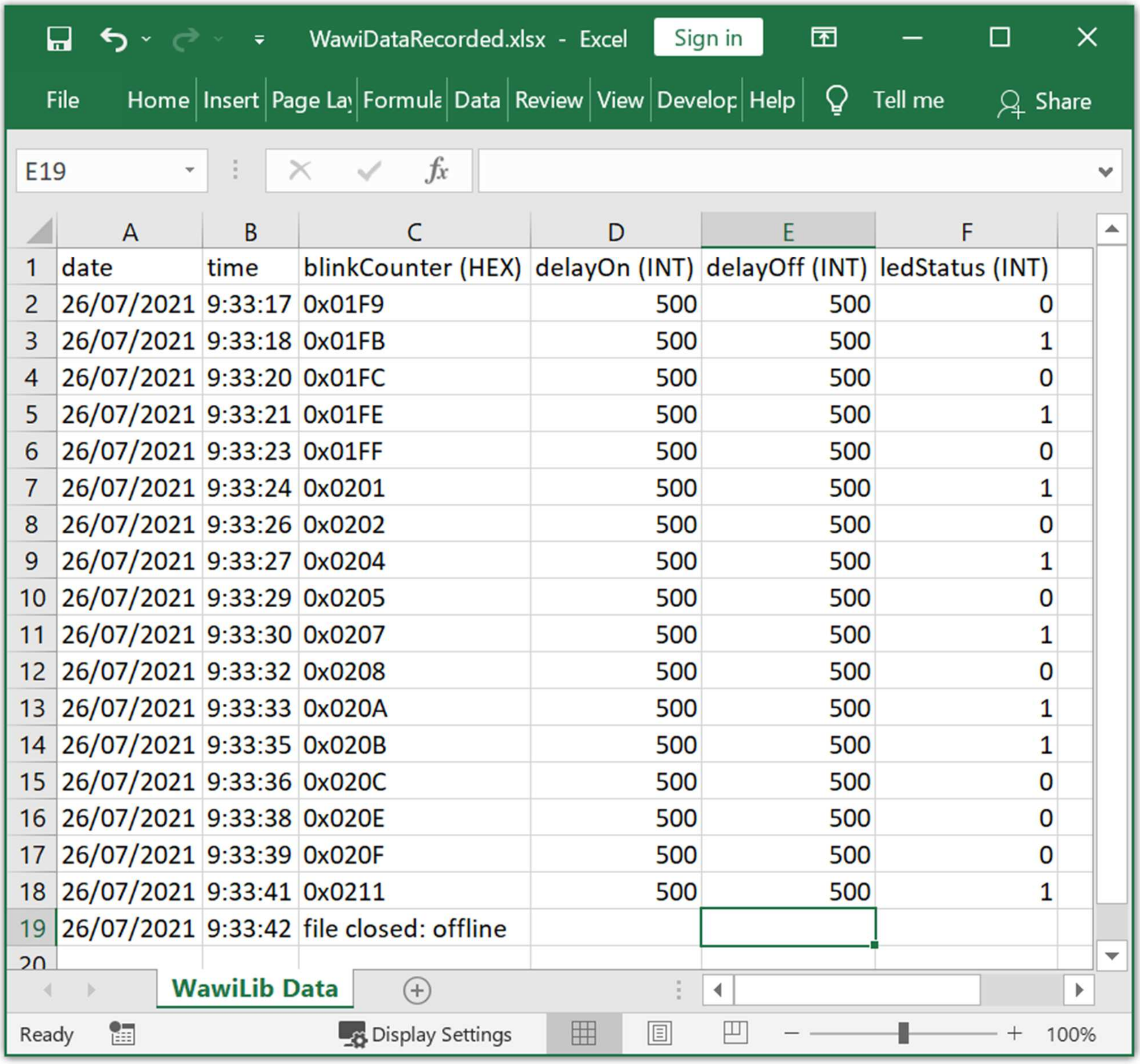


WawiDataRecorded.xlsx - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Calibri 11
A2
26/07/2021
WawiLib Data
Sheet 1 of 1
PageStyle_WawiLib Data
English (USA)
100%

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | date | time | blinkCounter (HEX) | delayOn (INT) | delayOff (INT) | ledStatus (INT) |
| 2 | 26/07/2021 | 9:33:17 | 0x01F9 | 500 | 500 | 0 |
| 3 | 26/07/2021 | 9:33:18 | 0x01FB | 500 | 500 | 1 |
| 4 | 26/07/2021 | 9:33:20 | 0x01FC | 500 | 500 | 0 |
| 5 | 26/07/2021 | 9:33:21 | 0x01FE | 500 | 500 | 1 |
| 6 | 26/07/2021 | 9:33:23 | 0x01FF | 500 | 500 | 0 |
| 7 | 26/07/2021 | 9:33:24 | 0x0201 | 500 | 500 | 1 |
| 8 | 26/07/2021 | 9:33:26 | 0x0202 | 500 | 500 | 0 |
| 9 | 26/07/2021 | 9:33:27 | 0x0204 | 500 | 500 | 1 |
| 10 | 26/07/2021 | 9:33:29 | 0x0205 | 500 | 500 | 0 |
| 11 | 26/07/2021 | 9:33:30 | 0x0207 | 500 | 500 | 1 |
| 12 | 26/07/2021 | 9:33:32 | 0x0208 | 500 | 500 | 0 |
| 13 | 26/07/2021 | 9:33:33 | 0x020A | 500 | 500 | 1 |
| 14 | 26/07/2021 | 9:33:35 | 0x020B | 500 | 500 | 1 |
| 15 | 26/07/2021 | 9:33:36 | 0x020C | 500 | 500 | 0 |
| 16 | 26/07/2021 | 9:33:38 | 0x020E | 500 | 500 | 0 |
| 17 | 26/07/2021 | 9:33:39 | 0x020F | 500 | 500 | 0 |
| 18 | 26/07/2021 | 9:33:41 | 0x0211 | 500 | 500 | 1 |
| 19 | 26/07/2021 | 9:33:42 | file closed: offline | | | |
| 20 | | | | | | |
| 21 | | | | | | |

Fig. 8.7. File with recorded variables in .XLSX format opened in LibreOffice calc.

✓ Open the recorded .xlsx file in Microsoft Excel:



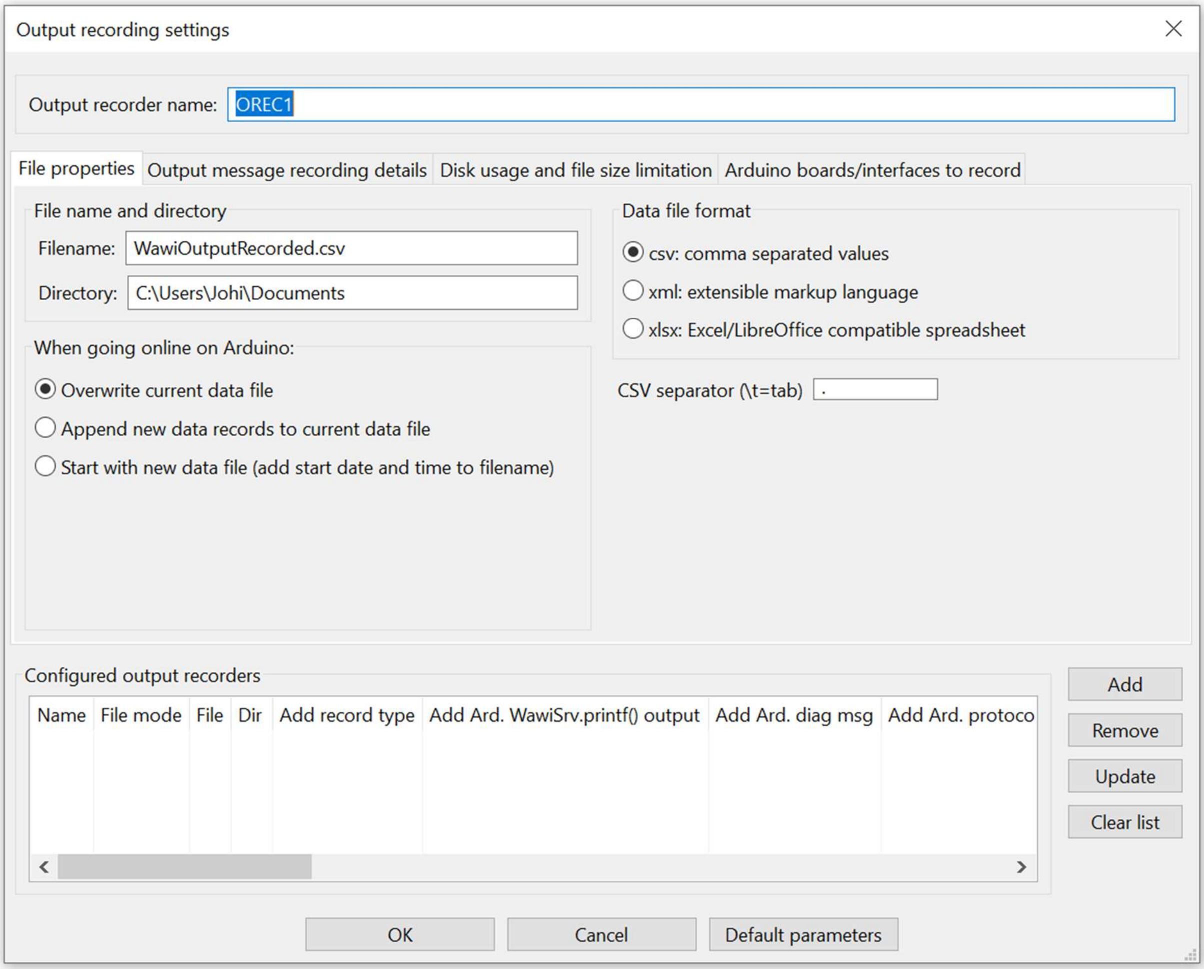
| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | date | time | blinkCounter (HEX) | delayOn (INT) | delayOff (INT) | ledStatus (INT) |
| 2 | 26/07/2021 | 9:33:17 | 0x01F9 | 500 | 500 | 0 |
| 3 | 26/07/2021 | 9:33:18 | 0x01FB | 500 | 500 | 1 |
| 4 | 26/07/2021 | 9:33:20 | 0x01FC | 500 | 500 | 0 |
| 5 | 26/07/2021 | 9:33:21 | 0x01FE | 500 | 500 | 1 |
| 6 | 26/07/2021 | 9:33:23 | 0x01FF | 500 | 500 | 0 |
| 7 | 26/07/2021 | 9:33:24 | 0x0201 | 500 | 500 | 1 |
| 8 | 26/07/2021 | 9:33:26 | 0x0202 | 500 | 500 | 0 |
| 9 | 26/07/2021 | 9:33:27 | 0x0204 | 500 | 500 | 1 |
| 10 | 26/07/2021 | 9:33:29 | 0x0205 | 500 | 500 | 0 |
| 11 | 26/07/2021 | 9:33:30 | 0x0207 | 500 | 500 | 1 |
| 12 | 26/07/2021 | 9:33:32 | 0x0208 | 500 | 500 | 0 |
| 13 | 26/07/2021 | 9:33:33 | 0x020A | 500 | 500 | 1 |
| 14 | 26/07/2021 | 9:33:35 | 0x020B | 500 | 500 | 1 |
| 15 | 26/07/2021 | 9:33:36 | 0x020C | 500 | 500 | 0 |
| 16 | 26/07/2021 | 9:33:38 | 0x020E | 500 | 500 | 0 |
| 17 | 26/07/2021 | 9:33:39 | 0x020F | 500 | 500 | 0 |
| 18 | 26/07/2021 | 9:33:41 | 0x0211 | 500 | 500 | 1 |
| 19 | 26/07/2021 | 9:33:42 | file closed: offline | | | |
| 20 | | | | | | |

Fig. 8.8. File with recorded variables in .XLSX format opened in Microsoft Excel..

# 9. Record .print() output to file (introduction)

In this section, we will configure an output recorder to record the output of WawiSrv.print() statements to a .csv file.

- ✓ Open the menu “Settings/Output Recording” in the main window.
- ✓ Press “Clear list”.
- ✓ Select as data file format in the first tab: “csv: comma separated values”.
- ✓ Select “Overwrite current data file”.
- ✓ Go to the second tab.
- ✓ Eanble “Arduino WawiSrv.print() messages” recording (see fig. 9.2.)
- ✓ Press “Add”:



Output recording settings

Output recorder name: OREC1

File properties
Output message recording details
Disk usage and file size limitation
Arduino boards/interfaces to record

File name and directory

Filename: WawiOutputRecorded.csv

Directory: C:\Users\Johi\Documents

When going online on Arduino:

● Overwrite current data file

○ Append new data records to current data file

○ Start with new data file (add start date and time to filename)

Data file format

● csv: comma separated values

○ xml: extensible markup language

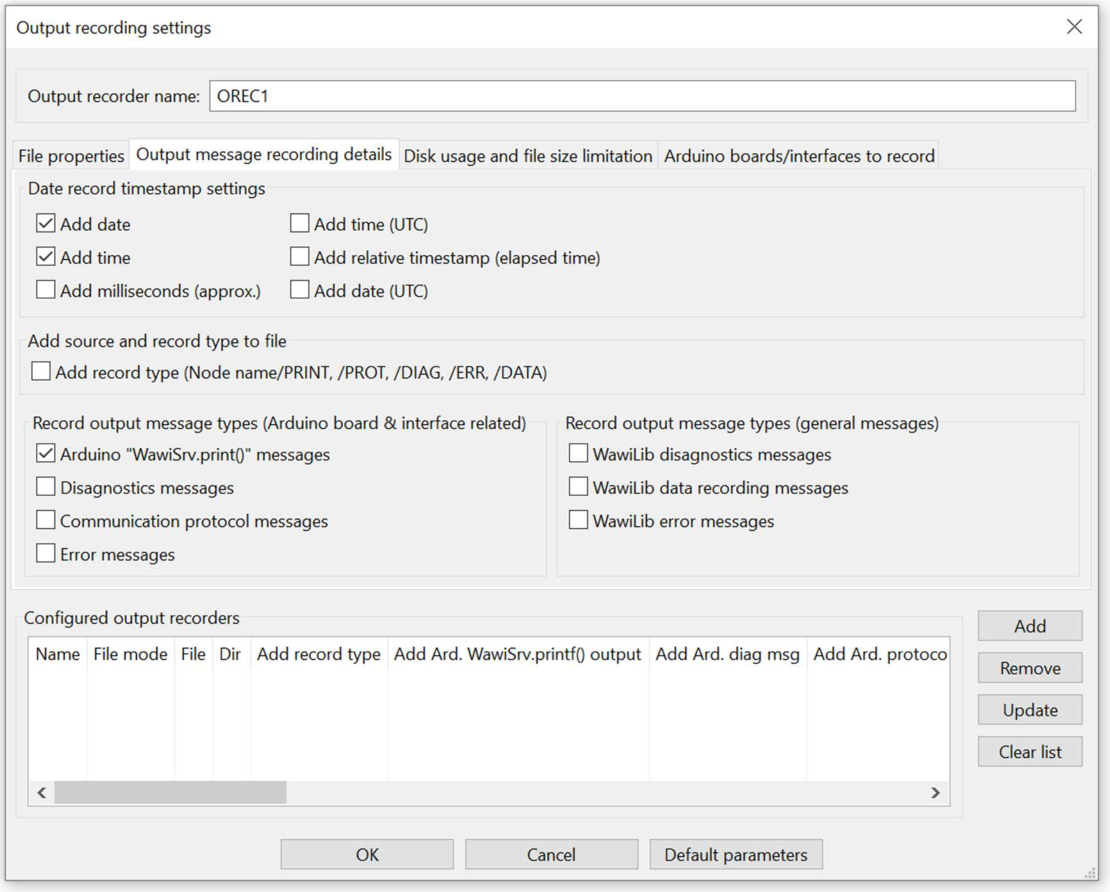
○ xlsx: Excel/LibreOffice compatible spreadsheet

CSV separator (\t=tab) .

Configured output recorders

| Name | File mode | File | Dir | Add record type | Add Ard. WawiSrv.printf() output | Add Ard. diag msg | Add Ard. protoco |
|---|---|---|---|---|---|---|---|

fig 9.1. Define a new output recorder.



Output recording settings

Output recorder name: OREC1

File properties
Output message recording details
Disk usage and file size limitation
Arduino boards/interfaces to record

Date record timestamp settings

☑ Add date
☑ Add time
□ Add milliseconds (approx.)
□ Add time (UTC)
□ Add relative timestamp (elapsed time)
□ Add date (UTC)

Add source and record type to file

□ Add record type (Node name/PRINT, /PROT, /DIAG, /ERR, /DATA)

Record output message types (Arduino board & interface related)

☑ Arduino "WawiSrv.print()" messages
□ Disagnostics messages
□ Communication protocol messages
□ Error messages

Record output message types (general messages)

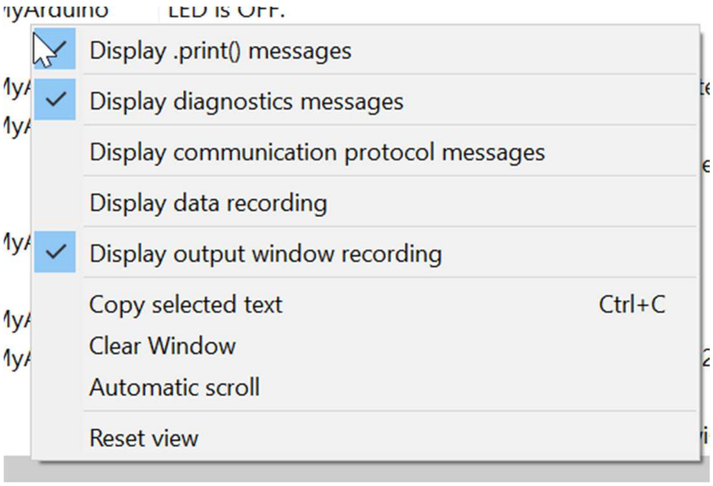
□ WawiLib disagnostics messages
□ WawiLib data recording messages
□ WawiLib error messages

Configured output recorders

| Name | File mode | File | Dir | Add record type | Add Ard. WawiSrv.printf() output | Add Ard. diag msg | Add Ard. protoco |
|---|---|---|---|---|---|---|---|

fig 9.2 Enable output “WawiSrv.print()” statements recording to disk file.

- ✓ Press “OK” to close the dialog box.
- ✓ Enable only “Display output window recording () and “Display .print() messages” .



lyArduino LED is OFF.
✓ Display .print() messages
✓ Display diagnostics messages
Display communication protocol messages
Display data recording
✓ Display output window recording
Copy selected text Ctrl+C
Clear Window
Automatic scroll
Reset view

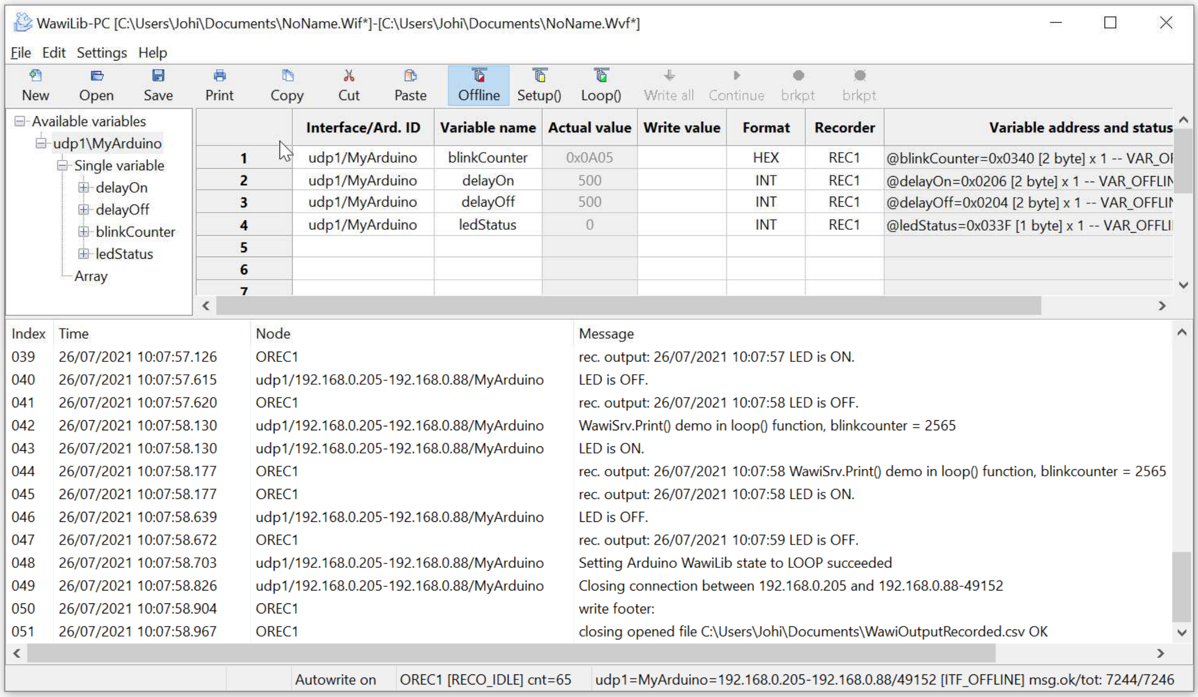
fig 9.3. Enable “Output recording” and “Display print messages” to the WawiLib output window.

- ✓ Press “Setup()”.
- ✓ Wait 15 seconds
- ✓ Press “Offline”

⇨ You will now see the output of the .print() statements in the output window at the same time this info is also recorded to file.

⇨ The LED on your board should blink 500ms on and 500ms off.

There is no link between the activation of the “Display output window recording” in the output window and the actual recording (=writing of the data to the file) on disk.



| | Interface/Ard. ID | Variable name | Actual value | Write value | Format | Recorder | Variable address and status |
|---|---|---|---|---|---|---|---|
| 1 | udp1/MyArduino | blinkCounter | 0x0A05 | | HEX | REC1 | @blinkCounter=0x0340 [2 byte] x 1 -- VAR_OF |
| 2 | udp1/MyArduino | delayOn | 500 | | INT | REC1 | @delayOn=0x0206 [2 byte] x 1 -- VAR_OFFLIN |
| 3 | udp1/MyArduino | delayOff | 500 | | INT | REC1 | @delayOff=0x0204 [2 byte] x 1 -- VAR_OFFLIN |
| 4 | udp1/MyArduino | ledStatus | 0 | | INT | REC1 | @ledStatus=0x033F [1 byte] x 1 -- VAR_OFFLI |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |

```
Index  Time                     Node                                        Message
039    26/07/2021 10:07:57.126  OREC1                                       rec. output: 26/07/2021 10:07:57 LED is ON.
040    26/07/2021 10:07:57.615  udp1/192.168.0.205-192.168.0.88/MyArduino   LED is OFF.
041    26/07/2021 10:07:57.620  OREC1                                       rec. output: 26/07/2021 10:07:58 LED is OFF.
042    26/07/2021 10:07:58.130  udp1/192.168.0.205-192.168.0.88/MyArduino   WawiSrv.Print() demo in loop() function, blinkcounter = 2565
043    26/07/2021 10:07:58.130  udp1/192.168.0.205-192.168.0.88/MyArduino   LED is ON.
044    26/07/2021 10:07:58.177  OREC1                                       rec. output: 26/07/2021 10:07:58 WawiSrv.Print() demo in loop() function, blinkcounter = 2565
045    26/07/2021 10:07:58.177  OREC1                                       rec. output: 26/07/2021 10:07:58 LED is ON.
046    26/07/2021 10:07:58.639  udp1/192.168.0.205-192.168.0.88/MyArduino   LED is OFF.
047    26/07/2021 10:07:58.672  OREC1                                       rec. output: 26/07/2021 10:07:59 LED is OFF.
048    26/07/2021 10:07:58.703  udp1/192.168.0.205-192.168.0.88/MyArduino   Setting Arduino WawiLib state to LOOP succeeded
049    26/07/2021 10:07:58.826  udp1/192.168.0.205-192.168.0.88/MyArduino   Closing connection between 192.168.0.205 and 192.168.0.88-49152
050    26/07/2021 10:07:58.904  OREC1                                       write footer:
051    26/07/2021 10:07:58.967  OREC1                                       closing opened file C:\Users\Johi\Documents\WawiOutputRecorded.csv OK
```

Autowrite on | OREC1 [RECO_IDLE] cnt=65 | udp1=MyArduino=192.168.0.205-192.168.0.88/49152 [ITF_OFFLINE] msg.ok/tot: 7244/7246

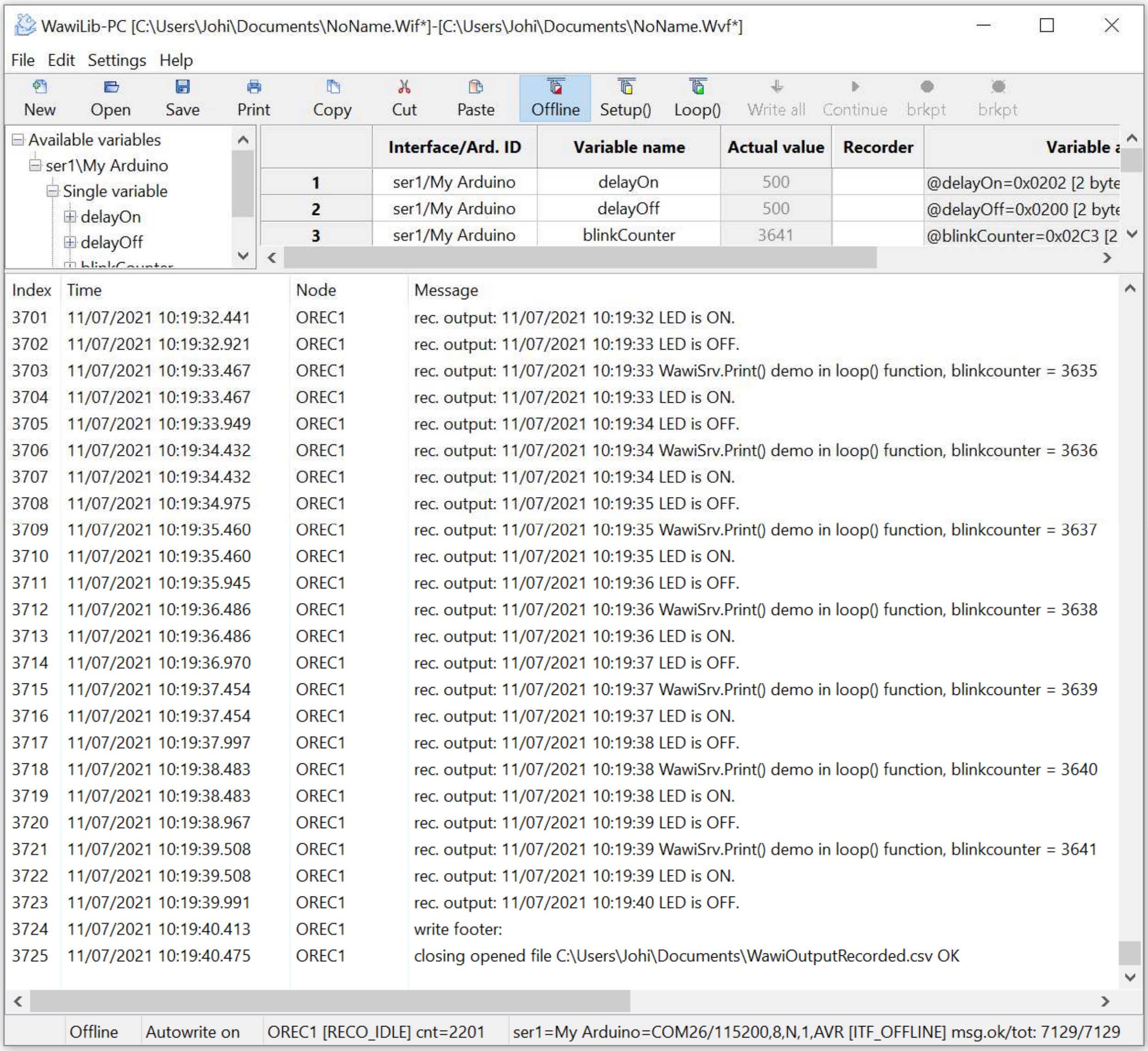
Fig 9.4. WawiLib output recording + tracing of output recording active.

```
/*
* Project Name: WawiBlinkMkr1000UdpWifi
* File: WawiBlinkMkr1000UdpWifi.ino
*
* Detailed manual:
* www.SylvesterSolutions.com\documentation -> "Getting started WawiLib WiFi.pdf"
*
* Description: demo file library for WawiWifi library.
* Lets you monitor and modify variables of different type and sizes.
* Uses WiFi to make connection with the Arduino board.
* Variables can be checked & modified with the WawiLib-PC software.
*
* Author: John Gijs.
* Created March 2020
* More info: www.sylvestersolutions.com
* Technical support: support@sylvestersolutions.com
* Additional info: info@sylvestersolutions.com
*/

#include <WawiWifi.h>

// WiFi parameters:
// char ssid[] = "--SSID--";   // network SSID (name)
// char key[] = "--SSID KEY--";   // network key

const char ssid[] = "telenet-F8665";    // my network SSID (name)
const char key[] = "vyzvveXbuh45";      // my network SSID (name of alternative
network)
```

```
// the media access control (ethernet hardware) address for the shield:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x88 };

// the IP address of your Arduino shield:
byte ipArd[] = { 192, 168, 0, 88 };

// communication port Arduino side for WawiLib communication (can be changed)
unsigned int port = 49152;

// your router's gateway address:
byte gateway[] = { 192, 168, 0, 1 };

// your network subnet:
byte subnet[] = { 255, 255, 225, 0 };

// your dns
byte mydns[] = { 195, 130, 130, 5 }; // your dns (not essential)

#define LED 6

// declare communication object:
WiFiUDP server;

// WawiLib communications object:
WawiWifi WawiSrv;

// test variables for demo:
int delayOn = 500;
int delayOff = 500;
int blinkCounter = 0;
bool ledStatus;

// make variables of interest known to WawiLib:
// this function is used in WawiSrv.begin(....)
void wawiVarDef()
{
    WawiSrv.wawiVar(delayOn);
    WawiSrv.wawiVar(delayOff);
    WawiSrv.wawiVar(blinkCounter);
    WawiSrv.wawiVar(ledStatus);
}

void printWiFiStatus();

void setup()
{
    Serial.begin(115200);

    // wait for serial monitor window to connect & open or time-out:
    int holdTime = 10000;
    while (!Serial && holdTime > 0)
    {
        delay(10);
        holdTime -= 10;
    }

    Serial.println(F("\nA) Looking for WiFi interface:"));
    if (WiFi.status() == WL_NO_SHIELD)
    {
        Serial.println(F("-> WiFi not detected. Program stopped."));
        while (true);
    }
    else
        Serial.println("-> WiFi detected.");

    Serial.println(F("B) Set static IP. "));
    WiFi.config(ipArd, mydns, gateway, subnet);

    Serial.println("C) Connect to WiFi network: ");
    int status = WiFi.begin(ssid, key);
    do
    {
        Serial.println(F("-> Connecting ..."));
        status = WiFi.begin(ssid, key);
        int i = 0;
        do
        {
            delay(1000);
            status = WiFi.status();
        } while (i++ < 10 && status != WL_CONNECTED);
    } while (status != WL_CONNECTED);
    Serial.println(F("-> Completed OK."));

    Serial.println(F("D) Read settings: "));
    Serial.print("-> SSID: ");
    Serial.println(WiFi.SSID());
    Serial.print("-> IP: ");
    IPAddress ip = WiFi.localIP();
    Serial.println(ip);
    Serial.print("-> UDP port: ");
    Serial.println(port);

    // start UDP/IP server:
    server.begin(port);

    // Setup WawiLib:
    Serial.println(F("E) Initializing WawiLib:"));
    WawiSrv.begin(wawiVarDef, server, "MyArduino");
    Serial.println(F("-> Done"));
    pinMode(LED, OUTPUT);
}

void loop()
{
    blinkCounter++;
    WawiSrv.print("WawiSrv.Print() demo in loop() function, blinkcounter = ");
    WawiSrv.println(blinkCounter);

    WawiSrv.println("LED is ON.");
    ledStatus = HIGH;
    digitalWrite(LED, ledStatus);
    WawiSrv.delay(delayOn);

    WawiSrv.println("LED is OFF.");
    ledStatus = LOW;
    digitalWrite(LED, ledStatus);
    WawiSrv.delay(delayOff);

    WawiSrv.loop();
}
```

fig 9.4. Minimal Arduino example `WawiBlinkMkr1000UdpWifi.ino`

- ✓ Press “Offline”.



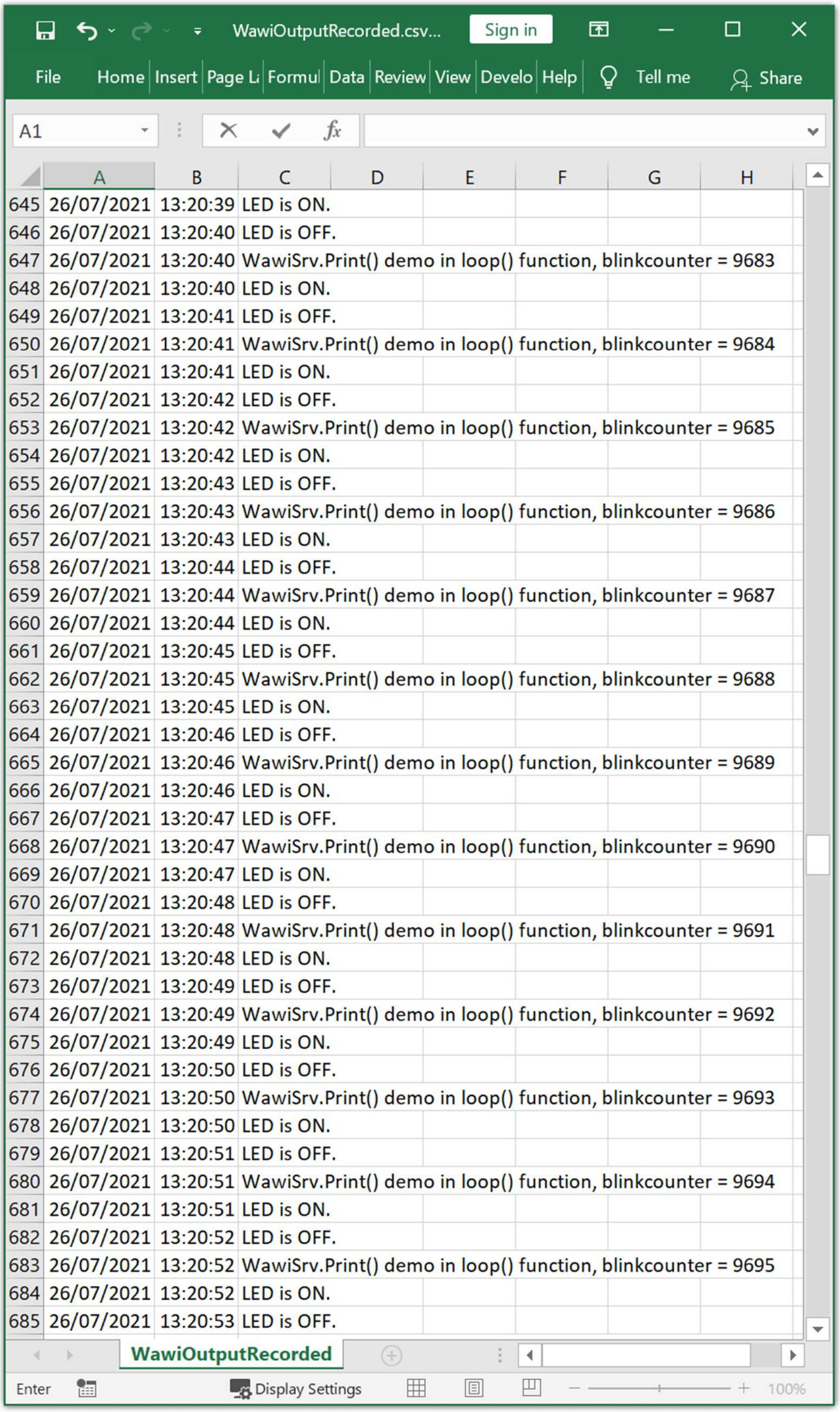
| | Interface/Ard. ID | Variable name | Actual value | Recorder | Variable a |
|---|---|---|---|---|---|
| 1 | ser1/My Arduino | delayOn | 500 | | @delayOn=0x0202 [2 byte |
| 2 | ser1/My Arduino | delayOff | 500 | | @delayOff=0x0200 [2 byte |
| 3 | ser1/My Arduino | blinkCounter | 3641 | | @blinkCounter=0x02C3 [2 |

```
Index Time                     Node    Message
3701  11/07/2021 10:19:32.441  OREC1   rec. output: 11/07/2021 10:19:32 LED is ON.
3702  11/07/2021 10:19:32.921  OREC1   rec. output: 11/07/2021 10:19:33 LED is OFF.
3703  11/07/2021 10:19:33.467  OREC1   rec. output: 11/07/2021 10:19:33 WawiSrv.Print() demo in loop() function, blinkcounter = 3635
3704  11/07/2021 10:19:33.467  OREC1   rec. output: 11/07/2021 10:19:33 LED is ON.
3705  11/07/2021 10:19:33.949  OREC1   rec. output: 11/07/2021 10:19:34 LED is OFF.
3706  11/07/2021 10:19:34.432  OREC1   rec. output: 11/07/2021 10:19:34 WawiSrv.Print() demo in loop() function, blinkcounter = 3636
3707  11/07/2021 10:19:34.432  OREC1   rec. output: 11/07/2021 10:19:34 LED is ON.
3708  11/07/2021 10:19:34.975  OREC1   rec. output: 11/07/2021 10:19:35 LED is OFF.
3709  11/07/2021 10:19:35.460  OREC1   rec. output: 11/07/2021 10:19:35 WawiSrv.Print() demo in loop() function, blinkcounter = 3637
3710  11/07/2021 10:19:35.460  OREC1   rec. output: 11/07/2021 10:19:35 LED is ON.
3711  11/07/2021 10:19:35.945  OREC1   rec. output: 11/07/2021 10:19:36 LED is OFF.
3712  11/07/2021 10:19:36.486  OREC1   rec. output: 11/07/2021 10:19:36 WawiSrv.Print() demo in loop() function, blinkcounter = 3638
3713  11/07/2021 10:19:36.486  OREC1   rec. output: 11/07/2021 10:19:36 LED is ON.
3714  11/07/2021 10:19:36.970  OREC1   rec. output: 11/07/2021 10:19:37 LED is OFF.
3715  11/07/2021 10:19:37.454  OREC1   rec. output: 11/07/2021 10:19:37 WawiSrv.Print() demo in loop() function, blinkcounter = 3639
3716  11/07/2021 10:19:37.454  OREC1   rec. output: 11/07/2021 10:19:37 LED is ON.
3717  11/07/2021 10:19:37.997  OREC1   rec. output: 11/07/2021 10:19:38 LED is OFF.
3718  11/07/2021 10:19:38.483  OREC1   rec. output: 11/07/2021 10:19:38 WawiSrv.Print() demo in loop() function, blinkcounter = 3640
3719  11/07/2021 10:19:38.483  OREC1   rec. output: 11/07/2021 10:19:38 LED is ON.
3720  11/07/2021 10:19:38.967  OREC1   rec. output: 11/07/2021 10:19:39 LED is OFF.
3721  11/07/2021 10:19:39.508  OREC1   rec. output: 11/07/2021 10:19:39 WawiSrv.Print() demo in loop() function, blinkcounter = 3641
3722  11/07/2021 10:19:39.508  OREC1   rec. output: 11/07/2021 10:19:39 LED is ON.
3723  11/07/2021 10:19:39.991  OREC1   rec. output: 11/07/2021 10:19:40 LED is OFF.
3724  11/07/2021 10:19:40.413  OREC1   write footer:
3725  11/07/2021 10:19:40.475  OREC1   closing opened file C:\Users\Johi\Documents\WawiOutputRecorded.csv OK
```

```
Offline   Autowrite on   OREC1 [RECO_IDLE] cnt=2201   ser1=My Arduino=COM26/115200,8,N,1,AVR [ITF_OFFLINE] msg.ok/tot: 7129/7129
```

fig 9.5. Arduino output recording displayed in window.

- ✓ Open the file WawiOutputRecordd.csv from your "Documents" folder.



| | A | B | C |
|---|---|---|---|
| 645 | 26/07/2021 | 13:20:39 | LED is ON. |
| 646 | 26/07/2021 | 13:20:40 | LED is OFF. |
| 647 | 26/07/2021 | 13:20:40 | WawiSrv.Print() demo in loop() function, blinkcounter = 9683 |
| 648 | 26/07/2021 | 13:20:40 | LED is ON. |
| 649 | 26/07/2021 | 13:20:41 | LED is OFF. |
| 650 | 26/07/2021 | 13:20:41 | WawiSrv.Print() demo in loop() function, blinkcounter = 9684 |
| 651 | 26/07/2021 | 13:20:41 | LED is ON. |
| 652 | 26/07/2021 | 13:20:42 | LED is OFF. |
| 653 | 26/07/2021 | 13:20:42 | WawiSrv.Print() demo in loop() function, blinkcounter = 9685 |
| 654 | 26/07/2021 | 13:20:42 | LED is ON. |
| 655 | 26/07/2021 | 13:20:43 | LED is OFF. |
| 656 | 26/07/2021 | 13:20:43 | WawiSrv.Print() demo in loop() function, blinkcounter = 9686 |
| 657 | 26/07/2021 | 13:20:43 | LED is ON. |
| 658 | 26/07/2021 | 13:20:44 | LED is OFF. |
| 659 | 26/07/2021 | 13:20:44 | WawiSrv.Print() demo in loop() function, blinkcounter = 9687 |
| 660 | 26/07/2021 | 13:20:44 | LED is ON. |
| 661 | 26/07/2021 | 13:20:45 | LED is OFF. |
| 662 | 26/07/2021 | 13:20:45 | WawiSrv.Print() demo in loop() function, blinkcounter = 9688 |
| 663 | 26/07/2021 | 13:20:45 | LED is ON. |
| 664 | 26/07/2021 | 13:20:46 | LED is OFF. |
| 665 | 26/07/2021 | 13:20:46 | WawiSrv.Print() demo in loop() function, blinkcounter = 9689 |
| 666 | 26/07/2021 | 13:20:46 | LED is ON. |
| 667 | 26/07/2021 | 13:20:47 | LED is OFF. |
| 668 | 26/07/2021 | 13:20:47 | WawiSrv.Print() demo in loop() function, blinkcounter = 9690 |
| 669 | 26/07/2021 | 13:20:47 | LED is ON. |
| 670 | 26/07/2021 | 13:20:48 | LED is OFF. |
| 671 | 26/07/2021 | 13:20:48 | WawiSrv.Print() demo in loop() function, blinkcounter = 9691 |
| 672 | 26/07/2021 | 13:20:48 | LED is ON. |
| 673 | 26/07/2021 | 13:20:49 | LED is OFF. |
| 674 | 26/07/2021 | 13:20:49 | WawiSrv.Print() demo in loop() function, blinkcounter = 9692 |
| 675 | 26/07/2021 | 13:20:49 | LED is ON. |
| 676 | 26/07/2021 | 13:20:50 | LED is OFF. |
| 677 | 26/07/2021 | 13:20:50 | WawiSrv.Print() demo in loop() function, blinkcounter = 9693 |
| 678 | 26/07/2021 | 13:20:50 | LED is ON. |
| 679 | 26/07/2021 | 13:20:51 | LED is OFF. |
| 680 | 26/07/2021 | 13:20:51 | WawiSrv.Print() demo in loop() function, blinkcounter = 9694 |
| 681 | 26/07/2021 | 13:20:51 | LED is ON. |
| 682 | 26/07/2021 | 13:20:52 | LED is OFF. |
| 683 | 26/07/2021 | 13:20:52 | WawiSrv.Print() demo in loop() function, blinkcounter = 9695 |
| 684 | 26/07/2021 | 13:20:52 | LED is ON. |
| 685 | 26/07/2021 | 13:20:53 | LED is OFF. |

fig 9.6. Arduino output recording file opened in Excel.

If you like, you can write also to an XML database file and WawiLib also supports closing the file each hour so your recordings remain limited in file size.

## 10. WawiLib breakpoints (introduction)

Sometimes you want your code to stop at a certain point. Advanced debuggers have these functions standard. WawiLib is no substitute for these tools. However sometimes a simple breakpoint can come in handy. Therefor WawiLib contains a basic breakpoint functionality.

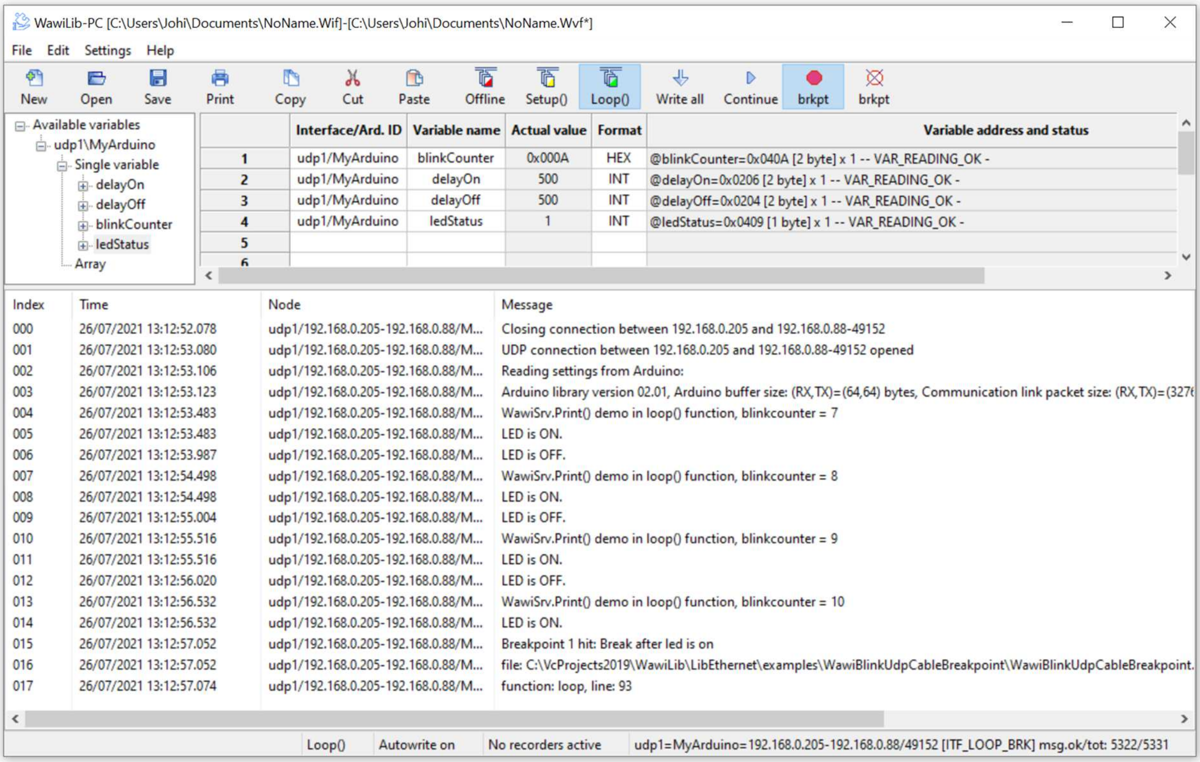
- ✓ Open the example File\Examples\WawiSerialUsb\WawiWifiBreakpoint.ino in the IDE.
- ✓ Compile and download the example.
- ✓ Connect WawiLib the board using “Settings\Communication interfaces” as in §5.
- ✓ Press “Setup()”.

```
/*
* Project Name: WawiBlinkMkr1000UdpWifiBreakpoint
* File: WawiBlinkMkr1000UdpWifi.ino
*
* Detailed manual:
* www.SylvesterSolutions.com\documentation -> "Getting started WawiLib WiFi.pdf"
*
* Description: demo file library for WawiWifi library.
* Lets you monitor and modify variables of different type and sizes.
* Uses WiFi to make connection with the Arduino board.
* Variables can be checked & modified with the WawiLib-PC software.
*
* This program will stop once it hit a breakpoint and can be resumed
* using WawiLib-PC.
*
* Author: John Gijs.
* Created March 2020
* More info: www.sylvestersolutions.com
* Technical support: support@sylvestersolutions.com
* Additional info: info@sylvestersolutions.com
*/

#include <WawiWifi.h>

// WiFi parameters:
// char ssid[] = "--SSID--";   // network SSID (name)
// char key[] = "--SSID KEY--";   // network key

// the media access control (ethernet hardware) address for the shield:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0x88 };

// the IP address of your Arduino shield:
byte ipArd[] = { 192, 168, 0, 88 };

// communication port Arduino side for WawiLib communication (can be changed)
unsigned int port = 49152;

// your router's gateway address:
byte gateway[] = { 192, 168, 0, 1 };

// your network subnet:
byte subnet[] = { 255, 255, 225, 0 };

// your dns
byte mydns[] = { 195, 130, 130, 5 }; // your dns (not essential)

#define LED 6

// declare communication object:
```

```
WiFiUDP server;

// WawiLib communications object:
WawiWifi WawiSrv;

// test variables for demo:
int delayOn = 500;
int delayOff = 500;
int blinkCounter = 0;
bool ledStatus;

// make variables of interest known to WawiLib:
// this function is used in WawiSrv.begin(....)
void wawiVarDef()
{
    WawiSrv.wawiVar(delayOn);
    WawiSrv.wawiVar(delayOff);
    WawiSrv.wawiVar(blinkCounter);
    WawiSrv.wawiVar(ledStatus);
}

void printWiFiStatus();

void setup()
{
    Serial.begin(115200);

    // wait for serial monitor window to connect & open or time-out:
    int holdTime = 10000;
    while (!Serial && holdTime > 0)
    {
        delay(10);
        holdTime -= 10;
    }

    Serial.println(F("\nA) Looking for WiFi interface:"));
    if (WiFi.status() == WL_NO_SHIELD)
    {
        Serial.println(F("-> WiFi not detected. Program stopped."));
        while (true);
    }
    else
        Serial.println("-> WiFi detected.");

    Serial.println(F("B) Set static IP. "));
    WiFi.config(ipArd, mydns, gateway, subnet);

    Serial.println("C) Connect to WiFi network: ");
    int status = WiFi.begin(ssid, key);
    do
    {
        Serial.println(F("-> Connecting ..."));
        status = WiFi.begin(ssid, key);
        int i = 0;
        do
        {
            delay(1000);
            status = WiFi.status();
        } while (i++ < 10 && status != WL_CONNECTED);
    } while (status != WL_CONNECTED);
    Serial.println(F("-> Completed OK."));
    Serial.println(F("D) Read settings: "));
    Serial.print("-> SSID: ");
    Serial.println(WiFi.SSID());
    Serial.print("-> IP: ");
    IPAddress ip = WiFi.localIP();
    Serial.println(ip);
    Serial.print("-> UDP port: ");
    Serial.println(port);

    // start UDP/IP server:
    server.begin(port);

    // Setup WawiLib:
    Serial.println(F("E) Initializing WawiLib:"));
    WawiSrv.begin(wawiVarDef, server, "MyArduino");
    Serial.println(F("-> Done"));
    pinMode(LED, OUTPUT);

    // Enable/disable breakpoints at startup:
    // (remove comment to activate)
    // WawiSrv.wawiBreakDisable();
    // WawiSrv.wawiBreakEnable();
}

void loop()
{
    blinkCounter++;
    WawiSrv.print("WawiSrv.Print() demo in loop() function, blinkcounter = ");
    WawiSrv.println(blinkCounter);
    WawiSrv.println("LED is ON.");
    ledStatus = HIGH;
    digitalWrite(LED, ledStatus);
    WawiSrv.delay(delayOn);
    if (blinkCounter % 5 == 0)
        WawiSrv.wawiBreak(1, "Break after led is on");
    WawiSrv.println("LED is OFF.");
    ledStatus = LOW;
    digitalWrite(LED, ledStatus);
    WawiSrv.delay(delayOff);
    if (blinkCounter % 10 == 0)
        WawiSrv.wawiBreak(2, "Break after led is off");

    WawiSrv.loop();
}
```

fig 10.1. WawiLib breakpoint support demo.

- ✓ Add the variables to the grid as indicated in fig. 10.2.
- ✓ Press “Setup()”.



| | Interface/Ard. ID | Variable name | Actual value | Format | Variable address and status |
|---|---|---|---|---|---|
| 1 | udp1/MyArduino | blinkCounter | 0x000A | HEX | @blinkCounter=0x040A [2 byte] x 1 -- VAR_READING_OK - |
| 2 | udp1/MyArduino | delayOn | 500 | INT | @delayOn=0x0206 [2 byte] x 1 -- VAR_READING_OK - |
| 3 | udp1/MyArduino | delayOff | 500 | INT | @delayOff=0x0204 [2 byte] x 1 -- VAR_READING_OK - |
| 4 | udp1/MyArduino | ledStatus | 1 | INT | @ledStatus=0x0409 [1 byte] x 1 -- VAR_READING_OK - |
| 5 | | | | | |
| 6 | | | | | |

```
Index  Time                     Node                                Message
000    26/07/2021 13:12:52.078  udp1/192.168.0.205-192.168.0.88/M...  Closing connection between 192.168.0.205 and 192.168.0.88-49152
001    26/07/2021 13:12:53.080  udp1/192.168.0.205-192.168.0.88/M...  UDP connection between 192.168.0.205 and 192.168.0.88-49152 opened
002    26/07/2021 13:12:53.106  udp1/192.168.0.205-192.168.0.88/M...  Reading settings from Arduino:
003    26/07/2021 13:12:53.123  udp1/192.168.0.205-192.168.0.88/M...  Arduino library version 02.01, Arduino buffer size: (RX,TX)=(64,64) bytes, Communication link packet size: (RX,TX)=(327
004    26/07/2021 13:12:53.483  udp1/192.168.0.205-192.168.0.88/M...  WawiSrv.Print() demo in loop() function, blinkcounter = 7
005    26/07/2021 13:12:53.483  udp1/192.168.0.205-192.168.0.88/M...  LED is ON.
006    26/07/2021 13:12:53.987  udp1/192.168.0.205-192.168.0.88/M...  LED is OFF.
007    26/07/2021 13:12:54.498  udp1/192.168.0.205-192.168.0.88/M...  WawiSrv.Print() demo in loop() function, blinkcounter = 8
008    26/07/2021 13:12:54.498  udp1/192.168.0.205-192.168.0.88/M...  LED is ON.
009    26/07/2021 13:12:55.004  udp1/192.168.0.205-192.168.0.88/M...  LED is OFF.
010    26/07/2021 13:12:55.516  udp1/192.168.0.205-192.168.0.88/M...  WawiSrv.Print() demo in loop() function, blinkcounter = 9
011    26/07/2021 13:12:55.516  udp1/192.168.0.205-192.168.0.88/M...  LED is ON.
012    26/07/2021 13:12:56.020  udp1/192.168.0.205-192.168.0.88/M...  LED is OFF.
013    26/07/2021 13:12:56.532  udp1/192.168.0.205-192.168.0.88/M...  WawiSrv.Print() demo in loop() function, blinkcounter = 10
014    26/07/2021 13:12:56.532  udp1/192.168.0.205-192.168.0.88/M...  LED is ON.
015    26/07/2021 13:12:57.052  udp1/192.168.0.205-192.168.0.88/M...  Breakpoint 1 hit: Break after led is on
016    26/07/2021 13:12:57.052  udp1/192.168.0.205-192.168.0.88/M...  file: C:\VcProjects2019\WawiLib\LibEthernet\examples\WawiBlinkUdpCableBreakpoint\WawiBlinkUdpCableBreakpoint.
017    26/07/2021 13:12:57.074  udp1/192.168.0.205-192.168.0.88/M...  function: loop, line: 93
```

Loop() | Autowrite on | No recorders active | udp1=MyArduino=192.168.0.205-192.168.0.88/49152 [ITF_LOOP_BRK] msg.ok/tot: 5322/5331

Fig 10.2. WawiBlinkUdpWifiBreakpoint hit a breakpoint.

⇨ The sketch will run until a breakpoint is hit.

⇨ Fig 10.2. shows that a breakpoint was hit:

- ○ The status of the board is ITF_LOOP_BRK (see status bar)
- ○ The blue arrow “Continue” on the toolbar is enabled.

✓ Press “Continue” in the toolbar.

⇨ The sketch will run further until blinkCounter is a multiple of 5 or 10 and then show a message in the output window as indicated in figure 29.

⇨ The output window contains the line and a message you defined yourselves in your code.

```
⇨      if (blinkCounter % 5 == 0)
⇨          WawiSrv.wawiBreak(1, "Break after led is on");
```

Fig 10.3. WawiBlinkUdpWifiBreakpoint hit a breakpoint.

⇨ The output window also contains the source file name, the function and the source line where the breakpoint was hit. (fig. 10.2)

✓ Wait until another breakpoint is hit.

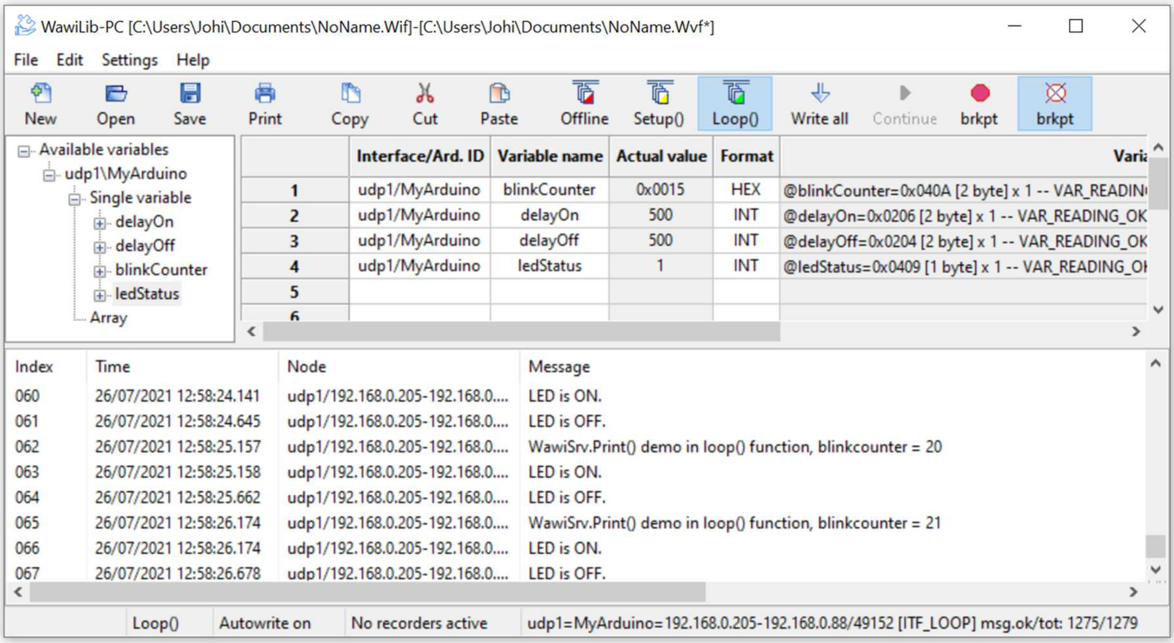
✓ Press the hollow circle “brkpt” in the toolbar to disable all breakpoints.

⇨ The sketch will run further disregarding breakpoints.

⇨ Note: You can define the initial activation state of the breakpoints using the following statements:

```
  // WawiSrv.wawiBreakDisable(); // enable or disable breakpoints at startup
  // WawiSrv.wawiBreakEnable();  // enable or disable breakpoints at startup
```

Fig 10.4. WawiBlinkUdpWifiBreakpoint activation at startup.



| | Interface/Ard. ID | Variable name | Actual value | Format | Varia |
|---|---|---|---|---|---|
| 1 | udp1/MyArduino | blinkCounter | 0x0015 | HEX | @blinkCounter=0x040A [2 byte] x 1 -- VAR_READIN |
| 2 | udp1/MyArduino | delayOn | 500 | INT | @delayOn=0x0206 [2 byte] x 1 -- VAR_READING_OK |
| 3 | udp1/MyArduino | delayOff | 500 | INT | @delayOff=0x0204 [2 byte] x 1 -- VAR_READING_OK |
| 4 | udp1/MyArduino | ledStatus | 1 | INT | @ledStatus=0x0409 [1 byte] x 1 -- VAR_READING_OI |
| 5 | | | | | |
| 6 | | | | | |

| Index | Time | Node | Message |
|---|---|---|---|
| 060 | 26/07/2021 12:58:24.141 | udp1/192.168.0.205-192.168.0.... | LED is ON. |
| 061 | 26/07/2021 12:58:24.645 | udp1/192.168.0.205-192.168.0.... | LED is OFF. |
| 062 | 26/07/2021 12:58:25.157 | udp1/192.168.0.205-192.168.0.... | WawiSrv.Print() demo in loop() function, blinkcounter = 20 |
| 063 | 26/07/2021 12:58:25.158 | udp1/192.168.0.205-192.168.0.... | LED is ON. |
| 064 | 26/07/2021 12:58:25.662 | udp1/192.168.0.205-192.168.0.... | LED is OFF. |
| 065 | 26/07/2021 12:58:26.174 | udp1/192.168.0.205-192.168.0.... | WawiSrv.Print() demo in loop() function, blinkcounter = 21 |
| 066 | 26/07/2021 12:58:26.174 | udp1/192.168.0.205-192.168.0.... | LED is ON. |
| 067 | 26/07/2021 12:58:26.678 | udp1/192.168.0.205-192.168.0.... | LED is OFF. |

```
Loop()    Autowrite on    No recorders active    udp1=MyArduino=192.168.0.205-192.168.0.88/49152 [ITF_LOOP] msg.ok/tot: 1275/1279
```

Fig 10.5. WawiBlinkUdpWifiBreakpoint with breakpoints deactivated.

# 11. Further reading

This demo demonstrates the concept of WawiLib using Wi-Fi. WawiLib has more extended functions that will be presented in other demos. Functions of interest to you can be the monitoring and modification of strings or the use of various representation formats (HEX/INT/UINT/CHAR/ STRING/FLOAT/DOUBLE).

Arrays of variables are also supported with WawiLib.  Recording of variables can be executed "on change", "on timer" or both.  Data recording can also be done with one file per hour or per day to make the generated files more manageable.

In the same way WawiLib supports recording of the output of .print() statements to a file on the disk of the PC. Files remain manageable as they can also be saved per hour or per day.

WawiLib also supports an elementary breakpoint facility that can be very handy debugging smaller Arduino's that have no on-board debug support or by absence of a special cable.

WawiLib supports links via Wi-Fi, cabled Ethernet, hardware serial, software serial and via USB to serial converters.

I hope you enjoyed this demo. Visit us on www.sylvestersolutions.com for more demos.