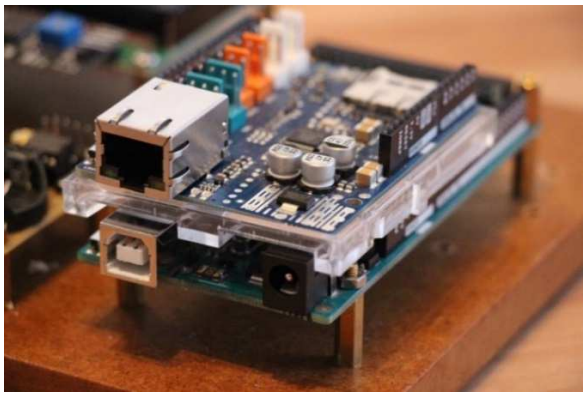
Getting started with WawiLib over Ethernet



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 using a wired Ethernet connection.

Many users know the Arduino “Blink” sketch. “Blink” was 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” and modify its internal variables over Ethernet.

“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 .xlsx, .xml 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 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, wired Ethernet is used as an interface between WawiLib and an Arduino board. The protocol used is UDP/IP.

The hardware you need is an Arduino board with an Ethernet 2 shield, a USB programming cable, a UTP cable, a router or switch and a Windows PC (32/64 bit). In this demo, we will use the Arduino MEGA2560 board but other boards can be used in a similar or even identical way (DUE, UNO, MKR1000, MKR1010, ZERO, …).

For the demo, the only difference between the UNO and other boards is the IO location of the LED. Samples for other boards are provided in the ‘Examples’ section of the Arduino IDE after installing WawiLib. You can modify the definition of the constant ‘LED’ yourself in the sample sketches if there is no sample for your board provided with WawiLib.

1.3. Required user experience

This demo assumes that the you know how to edit, compile and download Arduino programs. You should also have basic computer skills such as downloading and installing Windows programs. We will use Ethernet to connect to the Arduino board, therefore some basic knowledge of Ethernet 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 WawiEthernet Arduino library. If both have been correctly installed on your PC, you can skip this section.

- ✓ Download the WawiLib installer from 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:

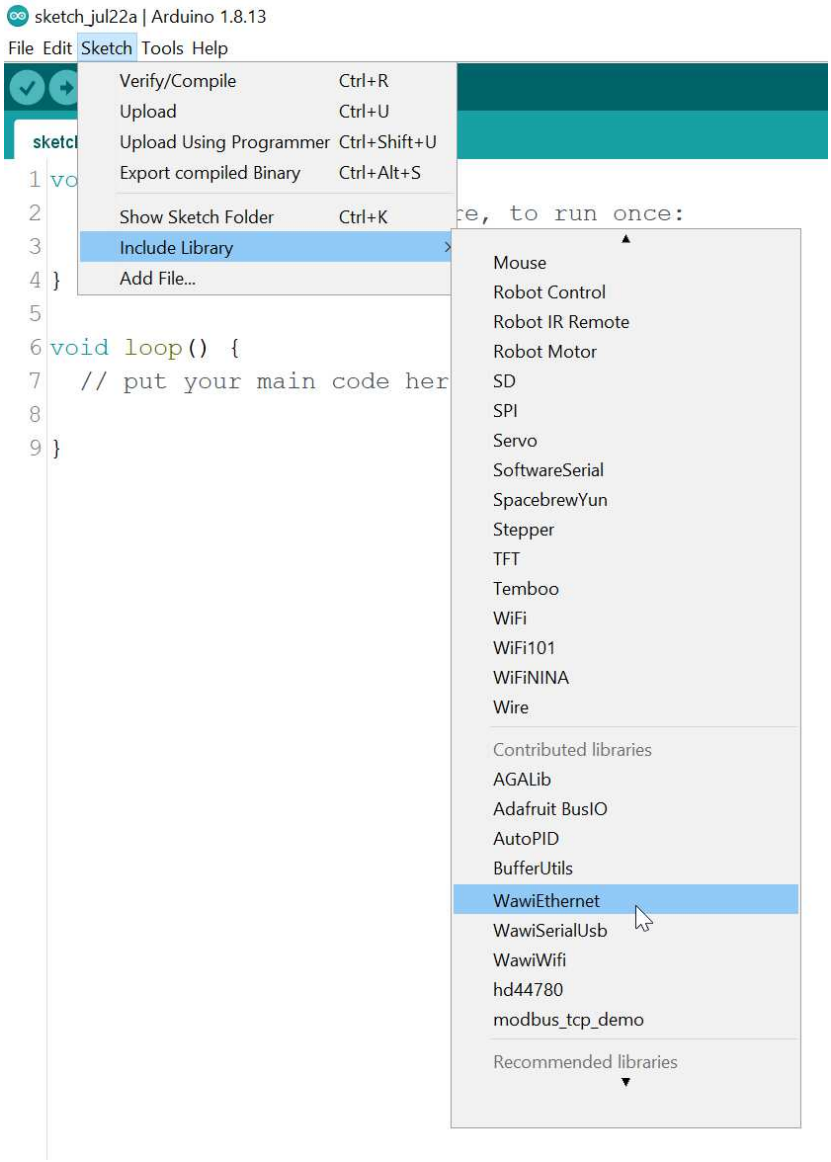


Fig. 2.1. Check the installation of the WawiEthernet library in the Arduino IDE.

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

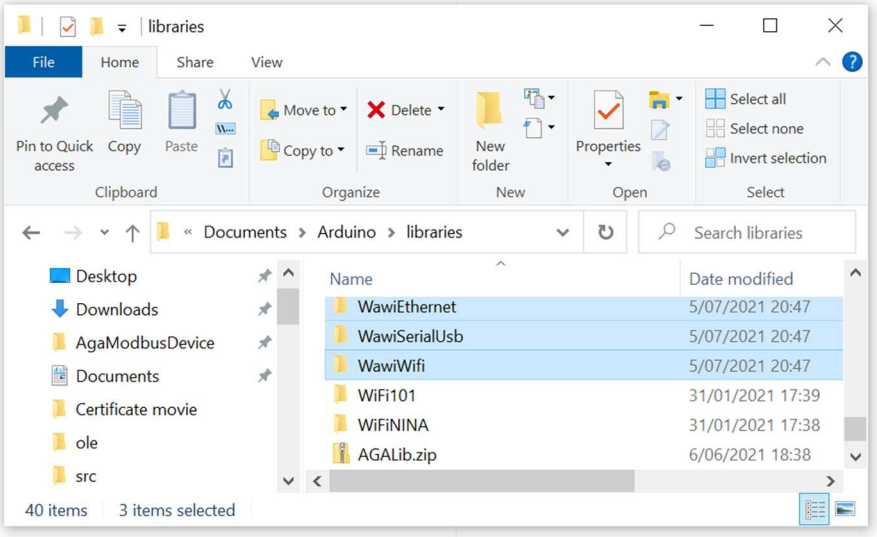


Fig. 2.2. Unpacked Libraries after installing WawiLib.

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".

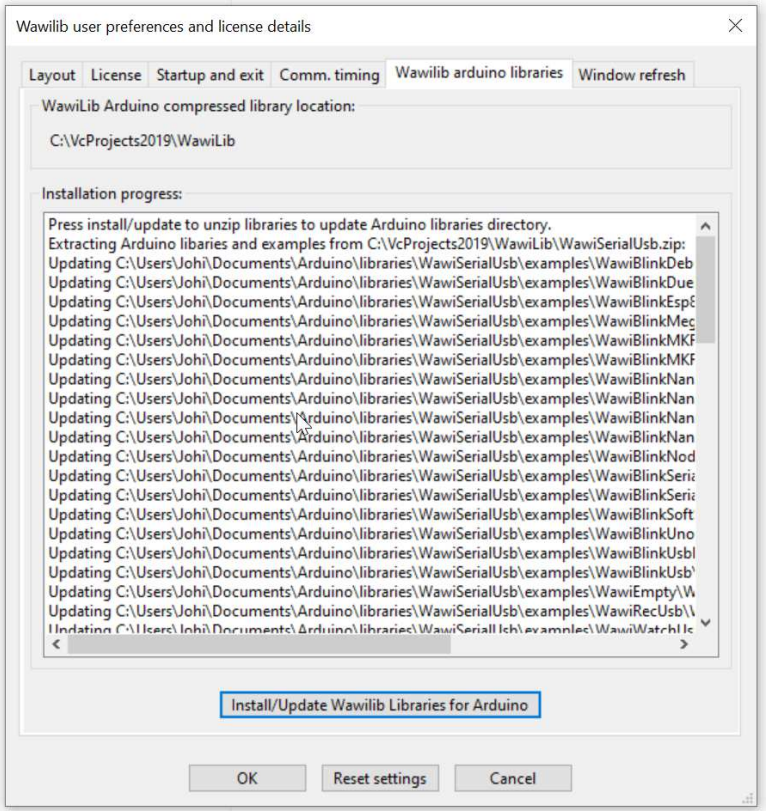


Fig. 2.3. Manual install of Arduino libraries.

3. Ethernet Network setup

3.1. Network layout

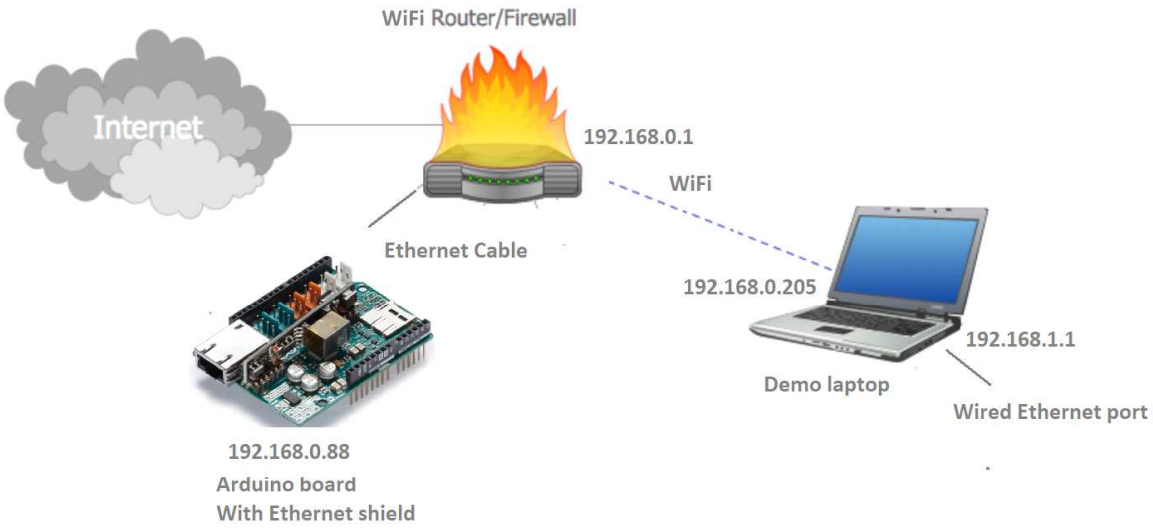


Fig. 3.1. Test network architecture overview.

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 a cabled connection between the router (or switch) and the Arduino board.

- ✓ Connect your Arduino board via an Ethernet UTP cable to your network switch/router.
- ✓ Make sure the Arduino board has power via USB or via a separate power supply.

3.2. Check your hardware connection.

The Arduino shield has 2 LEDs that are part of the Ethernet connector socket. The green one should burn and the orange one should blink or burn if the hardware connection is alive. If you see both LEDs off, there is a hardware problem that you need to solve first in order to continue with this demo.

3.3. Identify your network parameters.

- ✓ Open a command line window in Windows: press magnification glass on the taskbar and type "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.

Note: We could also connect the Arduino to the UTP network port of our laptop PC. If this is your setup intention, please make sure that the network card of the PC has fixed IP settings. Fixed IP settings are necessary as the PC is not connected to a DHCP server in this case. (this path is for more experienced users and out of scope for this demo).

3.4. 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 (remark: Some IP nodes on the network can disable their ping response for virus protection. Under these conditions the above strategy to determine a free fixed address is not completely watertight.)

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

As 192.168.0.1 is 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 the Arduino board with the demo sketch

Many of the Arduino libraries come with examples. WawiLib is not an exception. In this demo, we will use the sketch called WawiBlinkUdpCable.ino.

- ✓ Open the example via the menu “File\Examples\WawiEthernet\WawiBlinkUdpCable” in the Arduino IDE.

Demo files for other boards are also provided with WawiEthernet, the only difference with the demo for the Arduino Mega, Uno or Due is the definition of LED as in the MKR series the LED is not always at IO13.

```
/*
* Project Name: WawiBlinkUdpCable
* File: WawiBlinkUdpCable.ino
*
* Detailed manual:
* www.SylvesterSolutions.com\documentation -> "Getting started WawiLib Ethernet.pdf"
*
* Description: demo file library for WawiEthernet library.
* Uses cabled ethernet network to make connection with the Arduino board.
* Blinks LED at IO 13 with variable on and off periods.
* Counts the number of blinks.
* Variables can be checked & modified with the WawiLib-PC software.
*
* Author: John Gijs.
* Created Jan 2020
* More info: www.sylvestersolutions.com
* Technical support: support@sylvestersolutions.com
* Additional info: info@sylvestersolutions.com
*/

#include <WawiEthernet.h>

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

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

// communication port Arduino side for WawiLib communication
unsigned int port = 49152; // private free port number:

// standard Ethernet UDP communication object:
EthernetUDP UDPServer;

// WawiLib communications object:
WawiEthernet WawiSrv;

// LED position assumed at digital output 13
#define LED 13

// 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 setup()
{
	// serial port for diagnostics:
	Serial.begin(115200);
	delay(2000);

	// Setup Ethernet (standard Arduino code):
	Serial.println(F("A) Initializing Ethernet UDP lib... "));
	Ethernet.begin(mac, ipArd);
	UDPServer.begin(port);
	Serial.println(F("-> Init completed."));

	// wait for board to initialize
	delay(1000);

	// Setup WawiLib server object:
	Serial.println(F("B) Initializing WawiLib ... "));
	WawiSrv.begin(wawiVarDef, UDPServer, "MyArduino");
	Serial.println(F("-> OK."));

	// Setup DO for LED:
	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. 4.1. WawiBlinkUdpCable.ino source code.

- ✓ Fill in the free IP address identified in the previous section (192.168.0.88):

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

Fig. 4.2. WawiBlinkUdpCable.ino add your own IP address.

- ✓ Look at the value for *port*: the value of 49152 is the second parameter that identifies the connection between het Arduino board and the PC.

- ✓ Compile and download the sketch to your Arduino board.
- ✓ Open an Arduino serial monitor window:

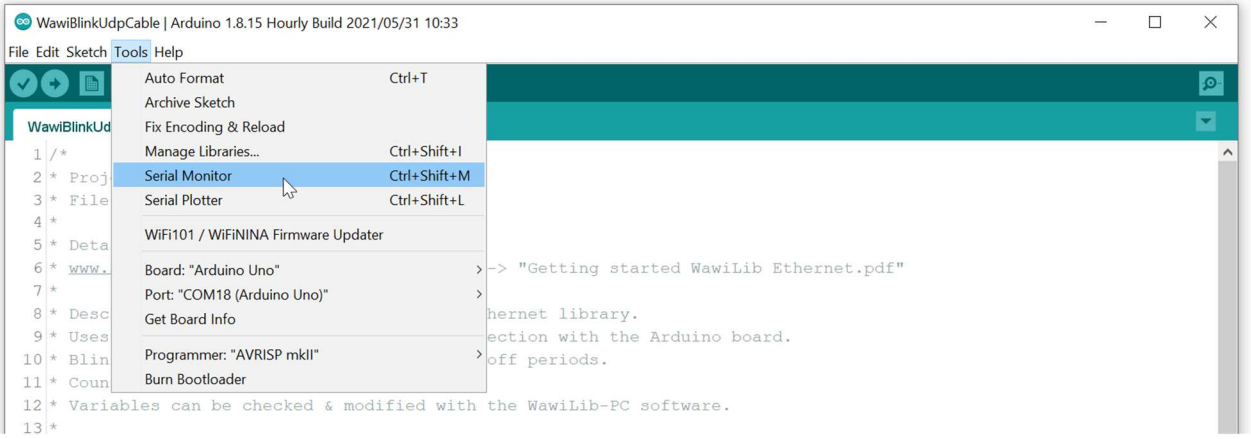


Fig. 4.3. How to make the serial output window visible.

- ✓ Restart your shield and look at the “serial monitor” output:

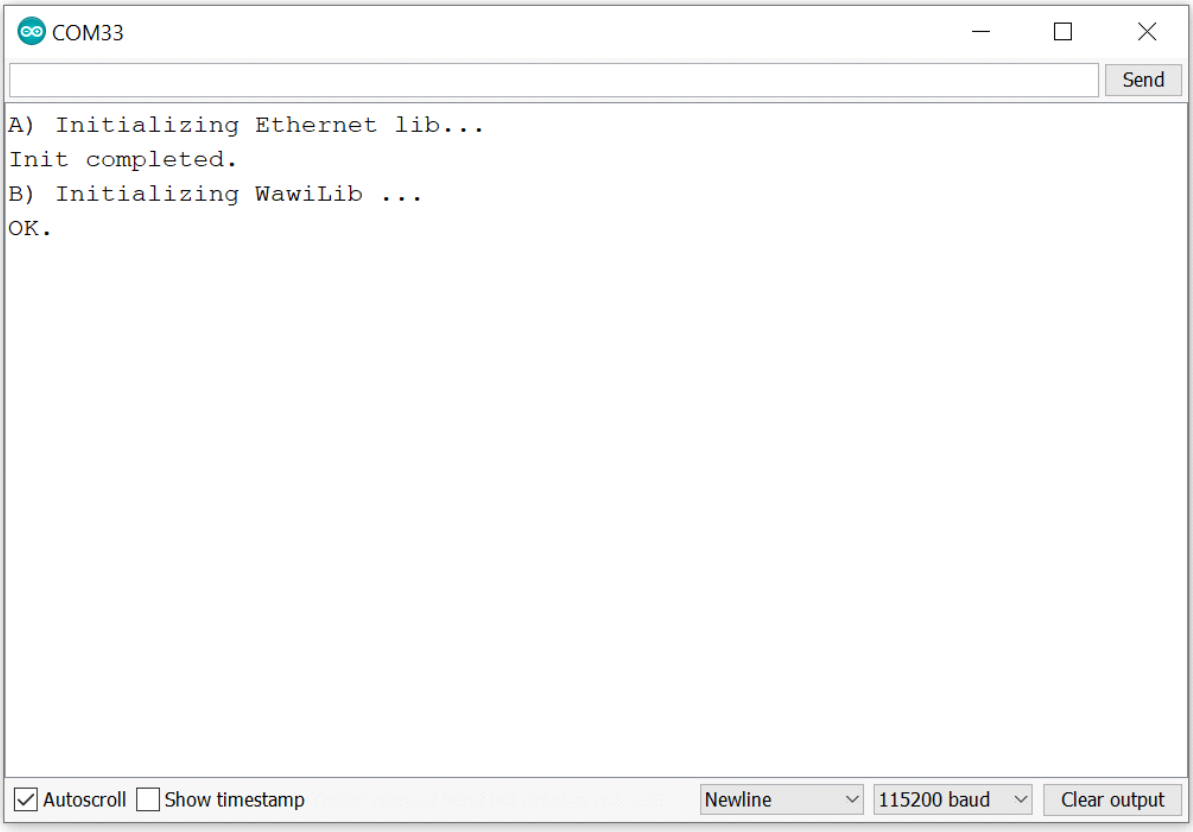


Fig. 4.4. The serial output Window.

- ✓ Ping to 192.168.0.88. there should be a response and no time-out (see below).

In the sketch above, you see that UDP port 49152 is used. You can use another port if you want. Remember to use the same port value in the WawiLib automatic scan range settings dialog box when making an Ethernet link (see later).

```
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. 4.5. 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.

It could be that the on-board LED is hidden by the Ethernet 2 shield. On the Ethernet shield, there is also a LED that is connected to Arduino I/O 13. This led is a small green LED mounted next to the Ethernet socket on the Ethernet Shield.

At this time, we have an Arduino board that is properly connected to one of our network segments and correctly parametrized to monitor and modify variables.

5. WawiLib user interface overview

✓ Start WawiLib on your PC:

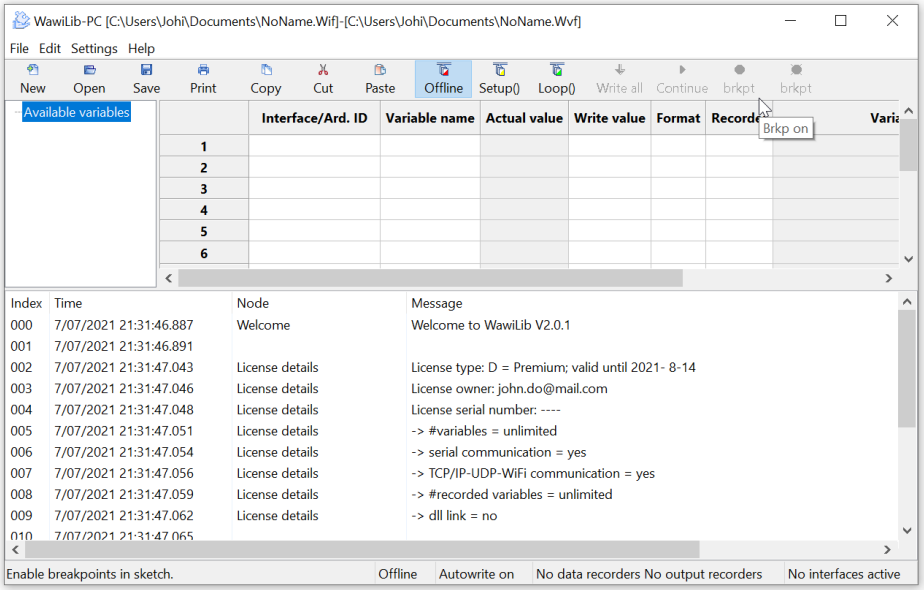


Fig. 5.1. WawiLib startup screen

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:

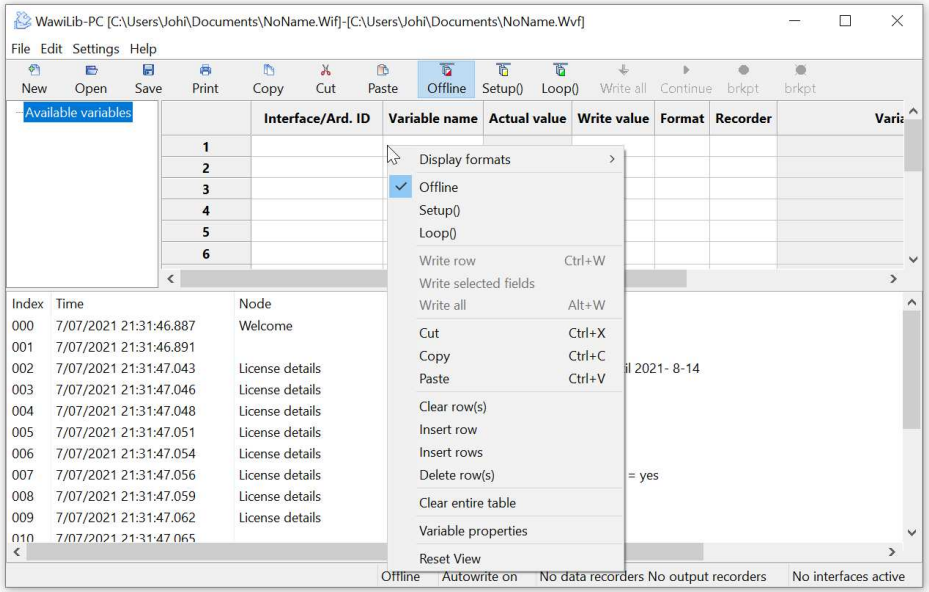


Fig. 5.2. WawiLib grid options.

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:

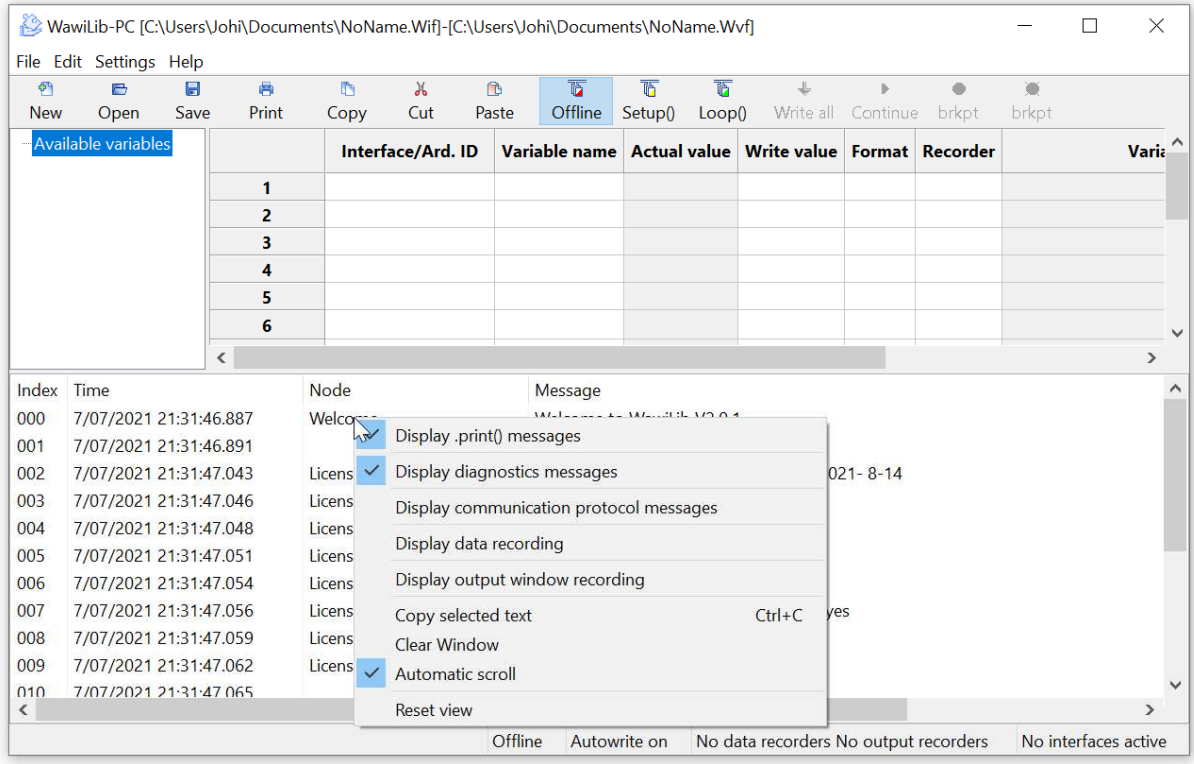


Fig. 5.3. WawiLib output window options.

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.

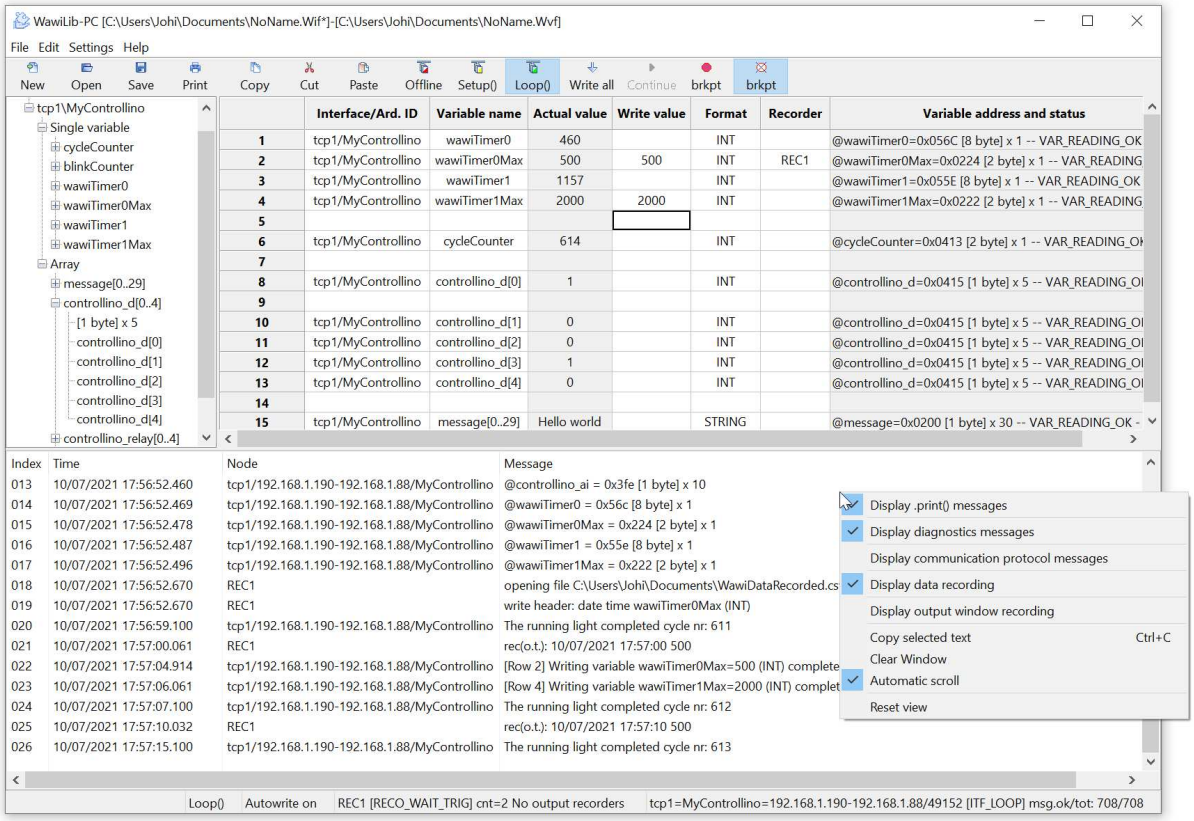


fig 5.4. WawiLib overview in a Controllino TCP configuration.

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 Ethernet communication link setup

One of the biggest challenges going online is finding the right UDP 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 the menu “Settings\Communication interfaces”.
- ✓ Select the tab “Ethernet UDP or TCP/IP communication scan settings”.

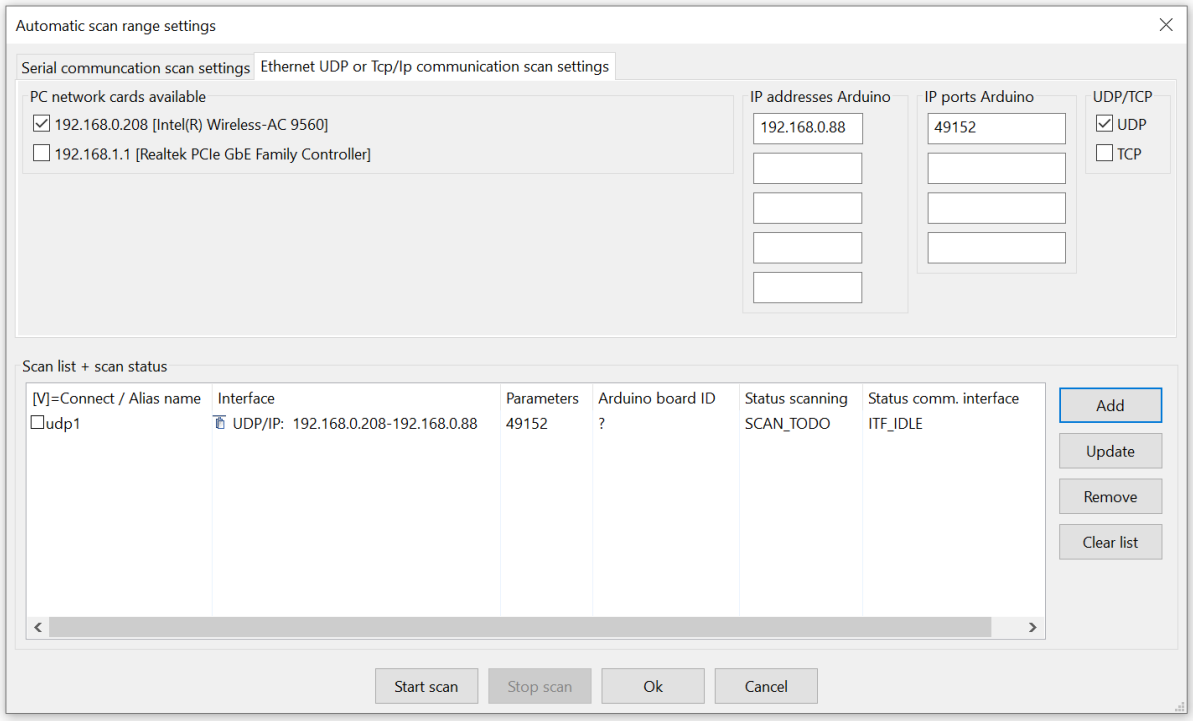


Fig. 6.1. Ethernet communication interface setup: scanning/setting up the connection.

- ✓ Check the PC network cards (adapters) to be used (i.e. the one linked to the network with the Arduino board.)
- ✓ Fill in the IP address and the IP port as indicated above (i.e. those settings were determined in §3.)
- ✓ Check “UDP” as protocol.
- ✓ Press “Add”.
- ✓ Press “Start scan”.

⇨ WawiLib will scan the network to identify Arduino boards on the Ethernet network using the selected parameters.

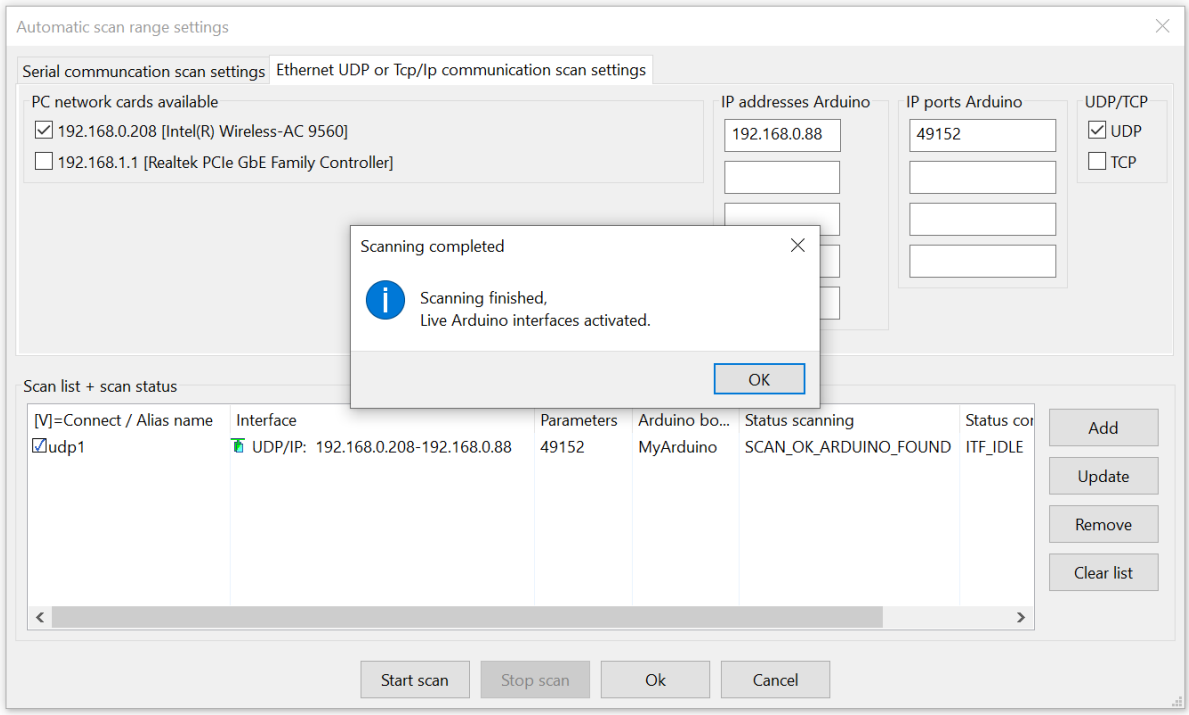


Fig. 6.2. Ethernet communication setup: indicating successful link check.

✓ Wait for the scan to complete.

By now, you should see a green icon indicating that the link was successfully tested. If the connection fails, you might want to re-try with your virus scanner disabled or with your window firewall disabled. Checking for a very short period of time without firewall protection to identify the cause of the problem is a calculated risk. Do not operate your PC without these safeties activated for a longer period of time.

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 but blocking off other rogue network traffic. The Windows Firewall can be configured in such a way that the WawiLib has free access to TCP and UDP ports but other applications have not.

✓ Press OK.

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

⇨ If you prefer TCP instead of UDP, you can use this protocol as well. WawiLib comes with Ethernet examples that use TCP. But TCP is much slower as the protocol has more overhead and the Arduino processor and his/her shield have limited resources.

⇨ Do not underestimate the advantages of UDP. Indeed it is connectionless “fire and forget”, but for many applications this is a better approach as the upper network layers guarantee the retries. UDP is much handier for message-oriented applications whereas TCP is better for streaming. Voice over IP and Profinet are for example of very widespread applications using UDP based protocols.

7. Read and write variables with WawiLib

7.1. Watch variables

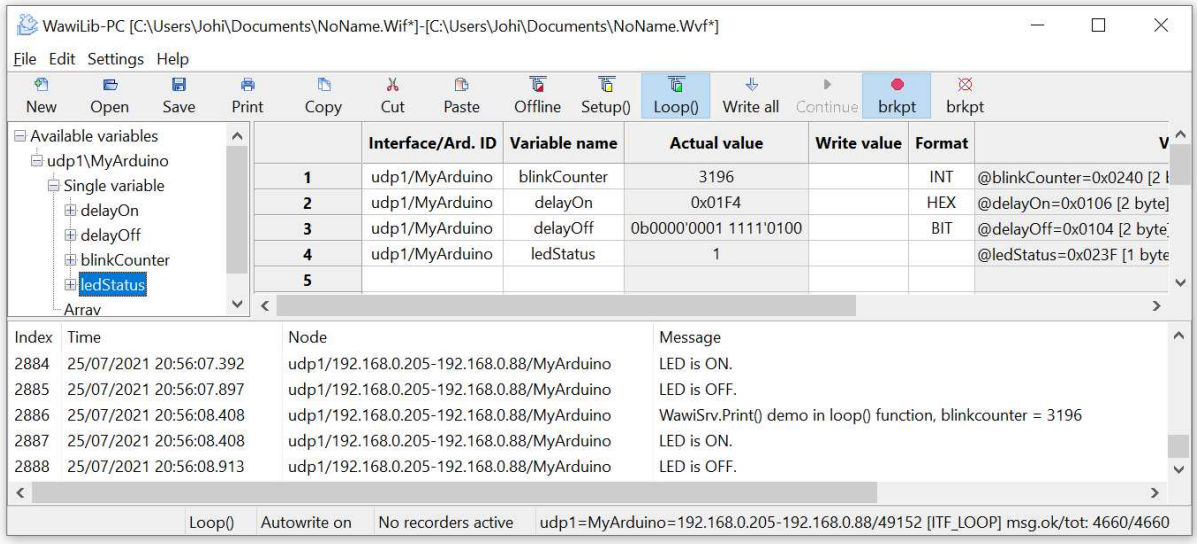


Fig. 7.1. Add variables to the grid using drag & drop.

- ✓ 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:

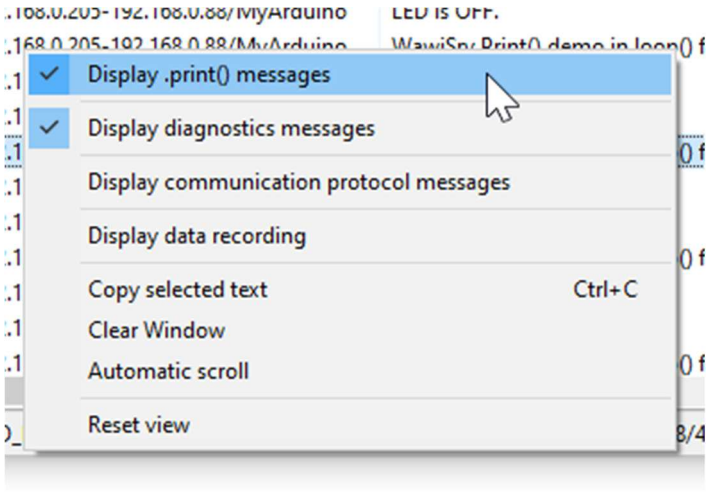


Fig. 7.3. The serial output window.

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

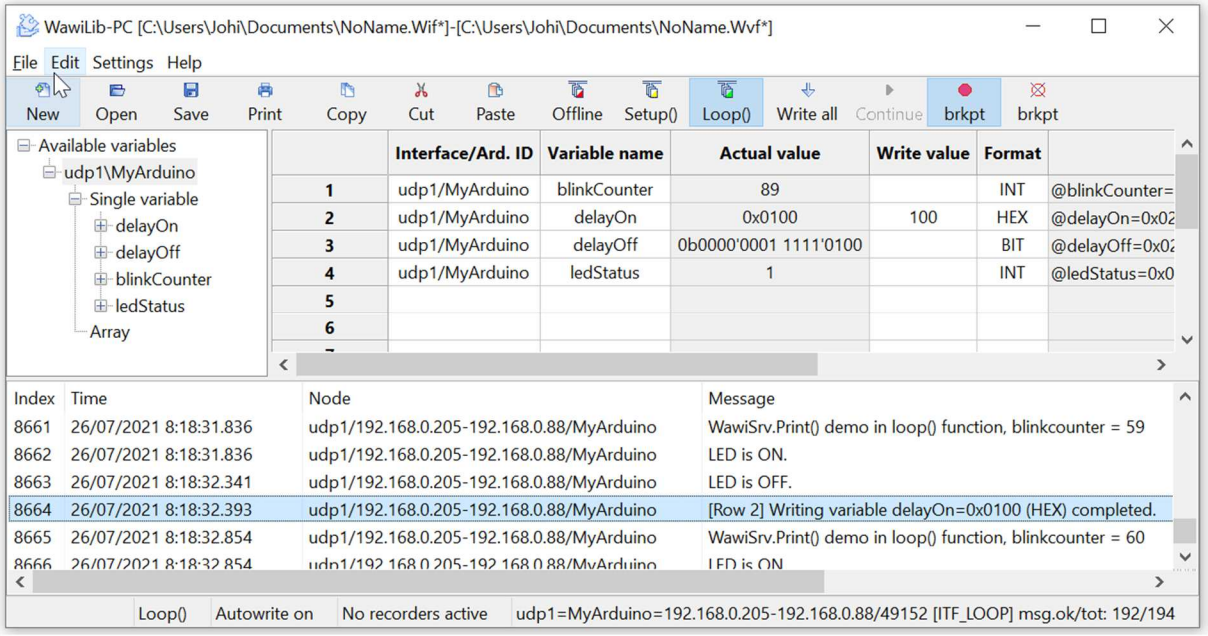


Fig. 7.4. Change the value of delayOn to 100 (hex).

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”

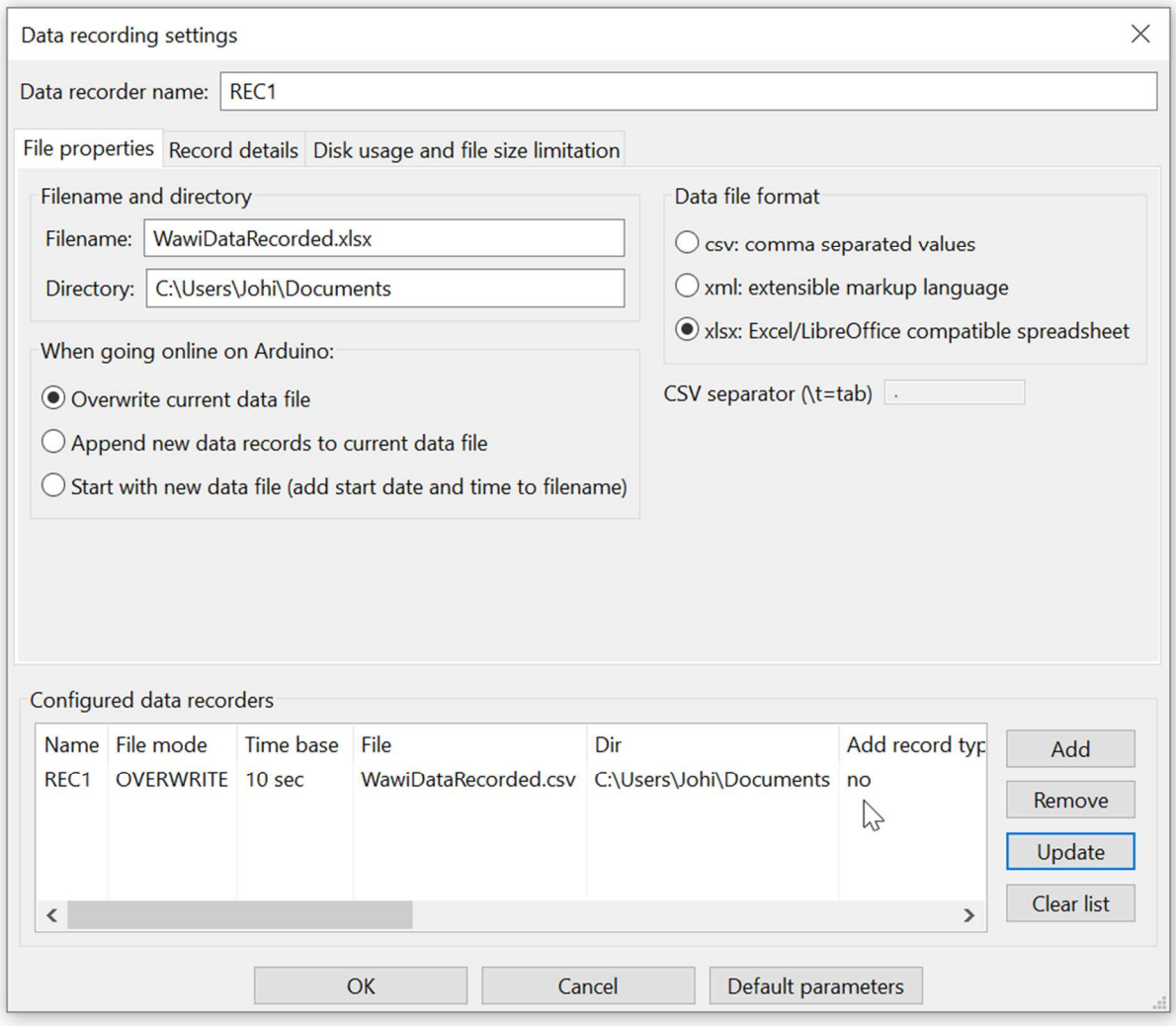


Fig. 8.1. The data recorder (for variables) setup dialog, main tab.

- ✓ 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”

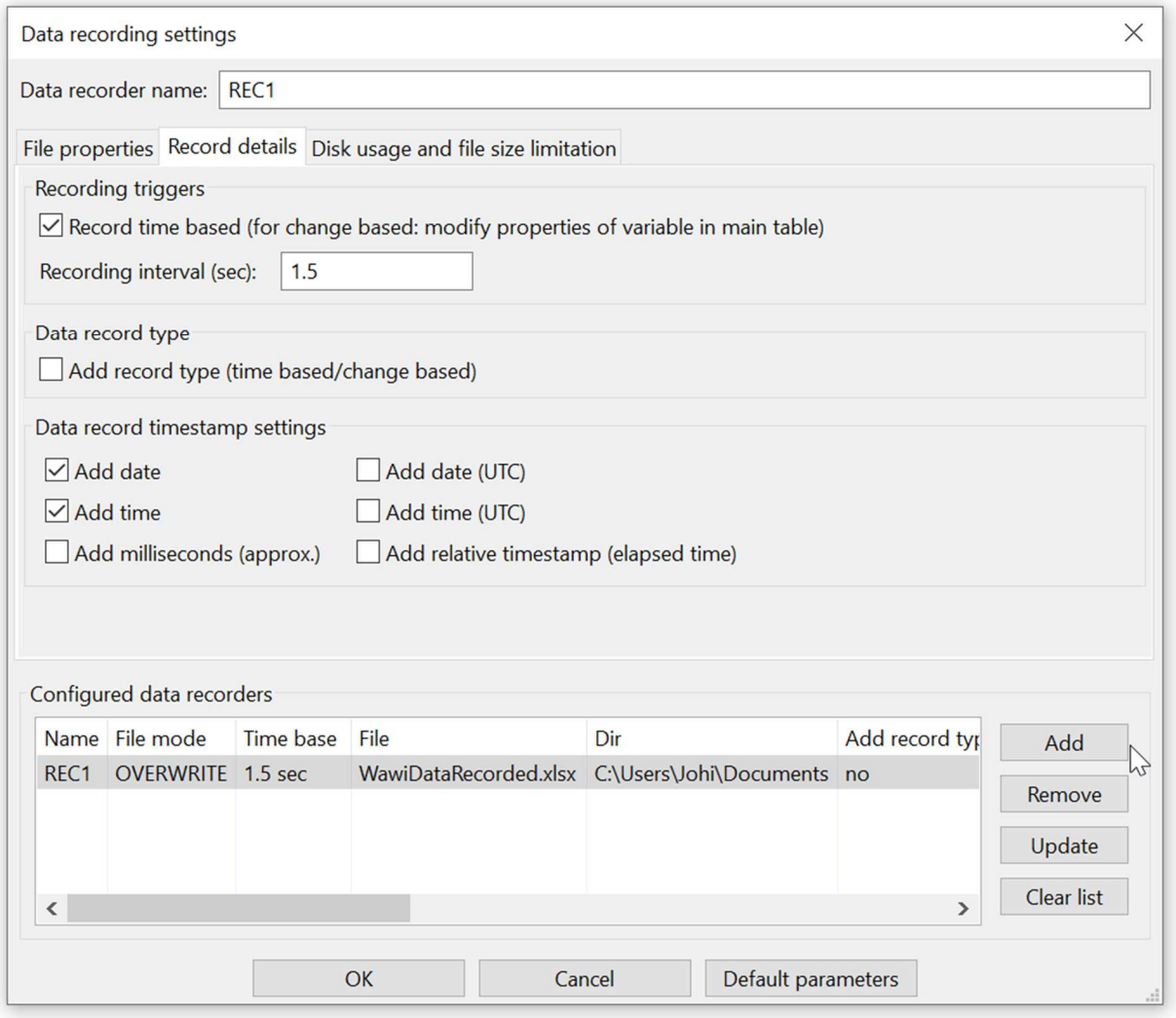


Fig. 8.2. The data recorder setup dialog, timing = 1.5 sec.

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.

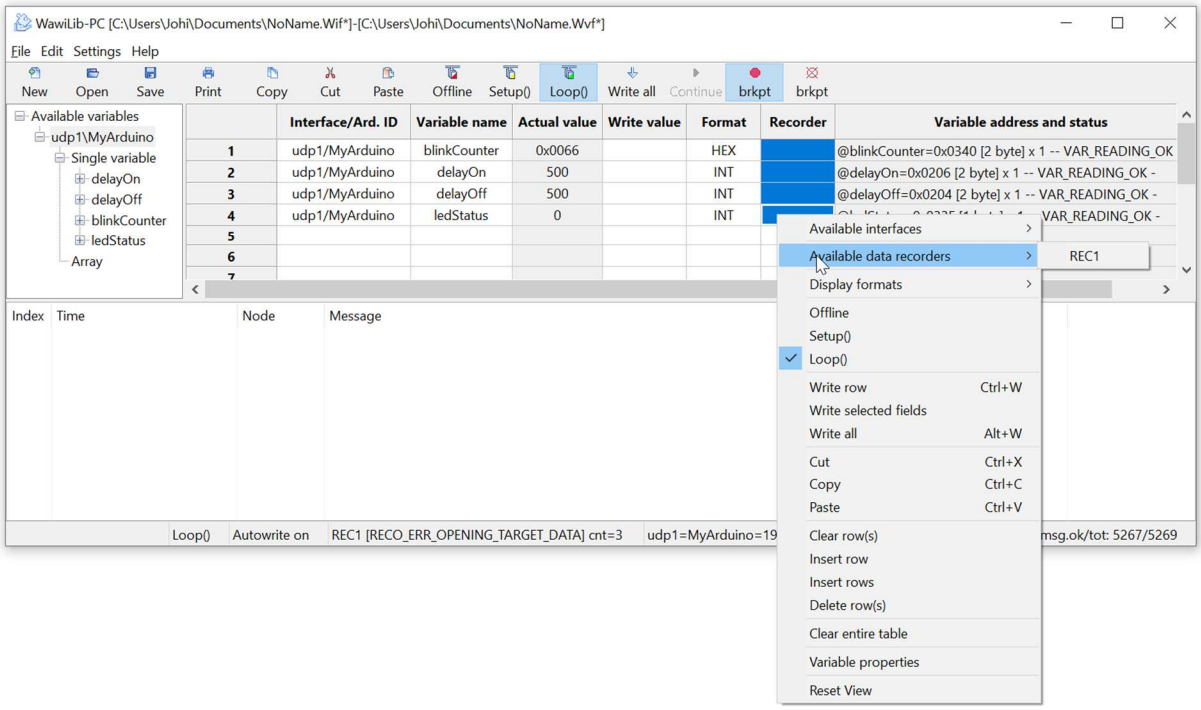


Fig. 8.3. Select a data recorder for all the variables.

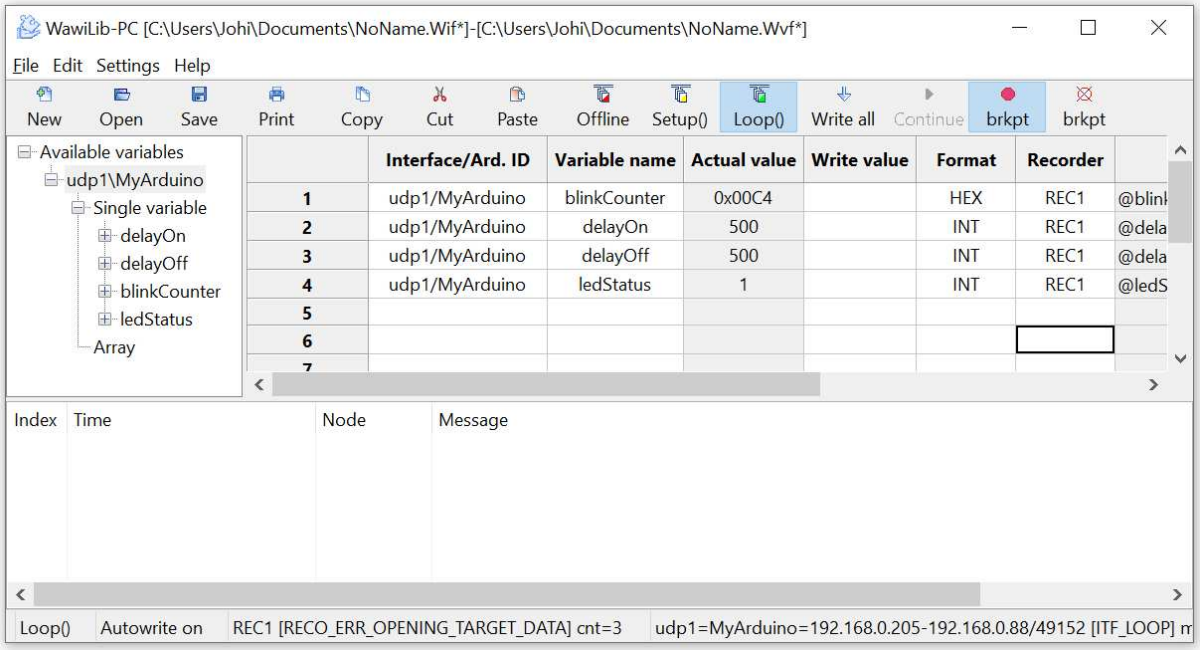


Fig. 8.4. Select a data recorder for all the variables: result.

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

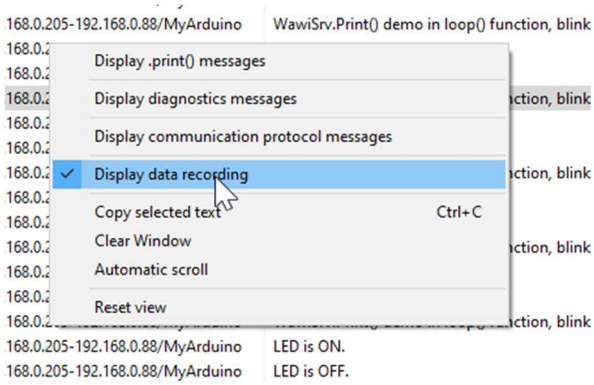


Fig. 8.5. Enable recording reporting in the output window.

- ✓ 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”

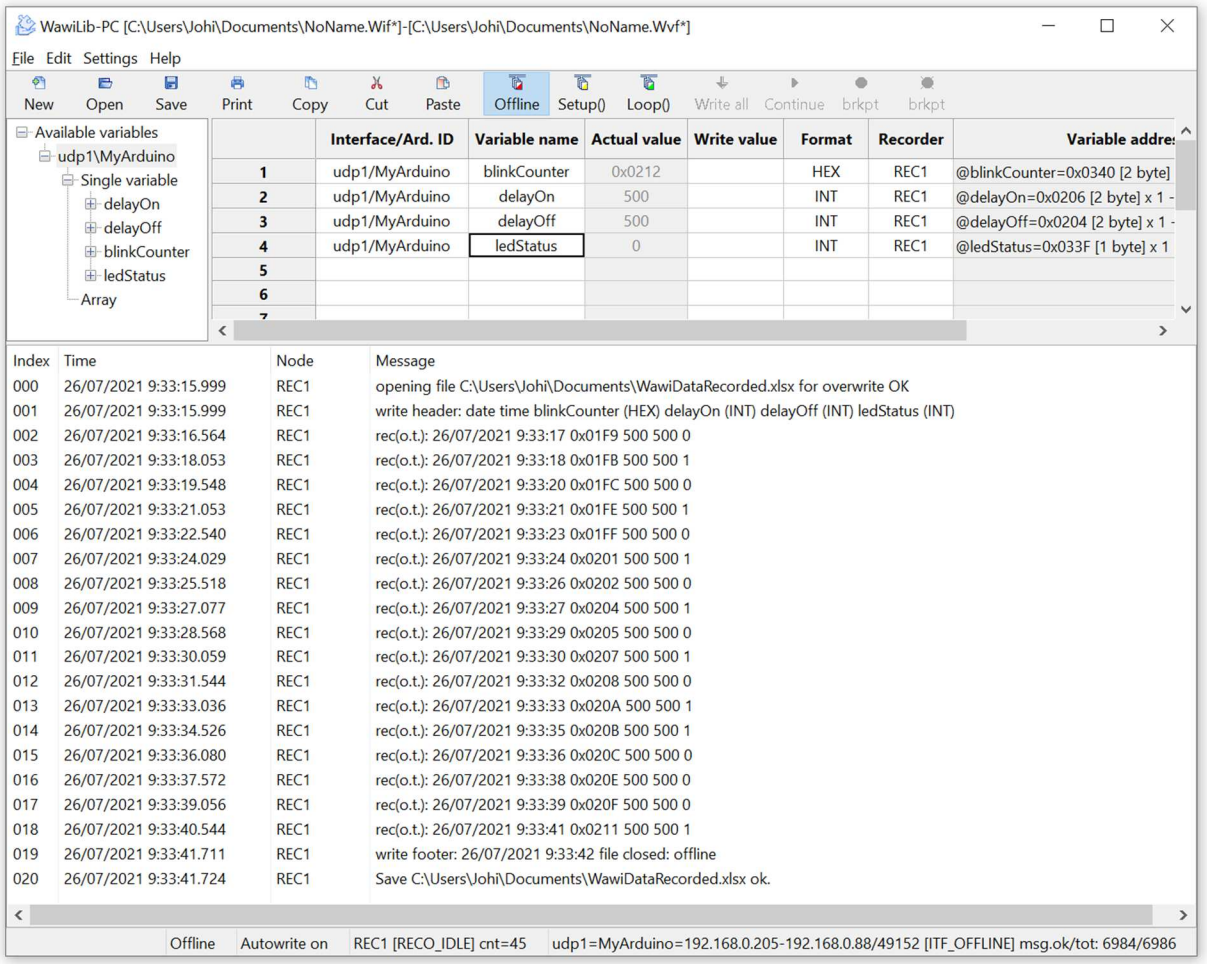
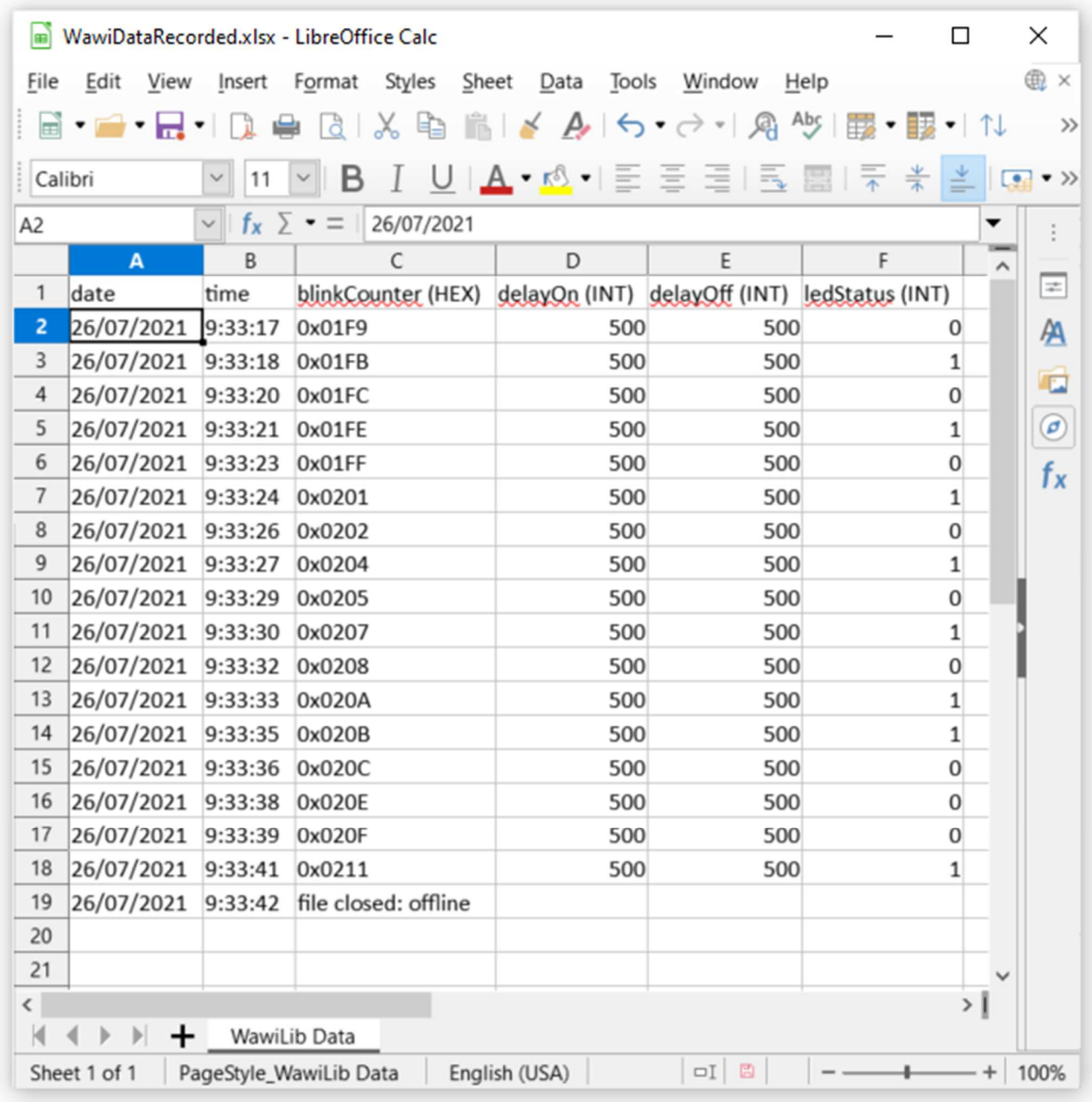


Fig. 8.6. The data recording mirrored in the output window.

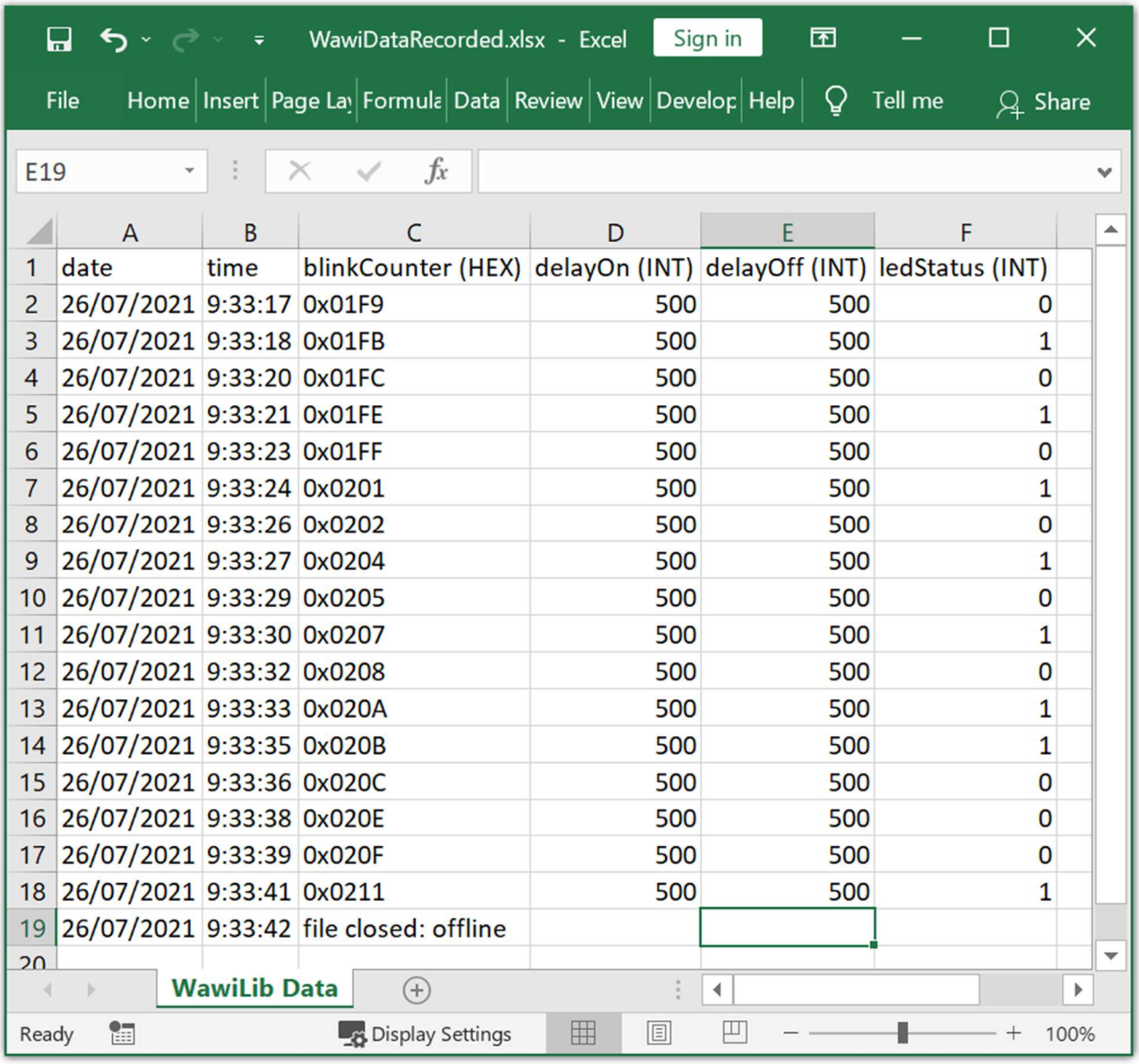
- ✓ Press “Offline”.
- ✓ Open the recorded .xlsx file in LibreOffice calc:



	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”:

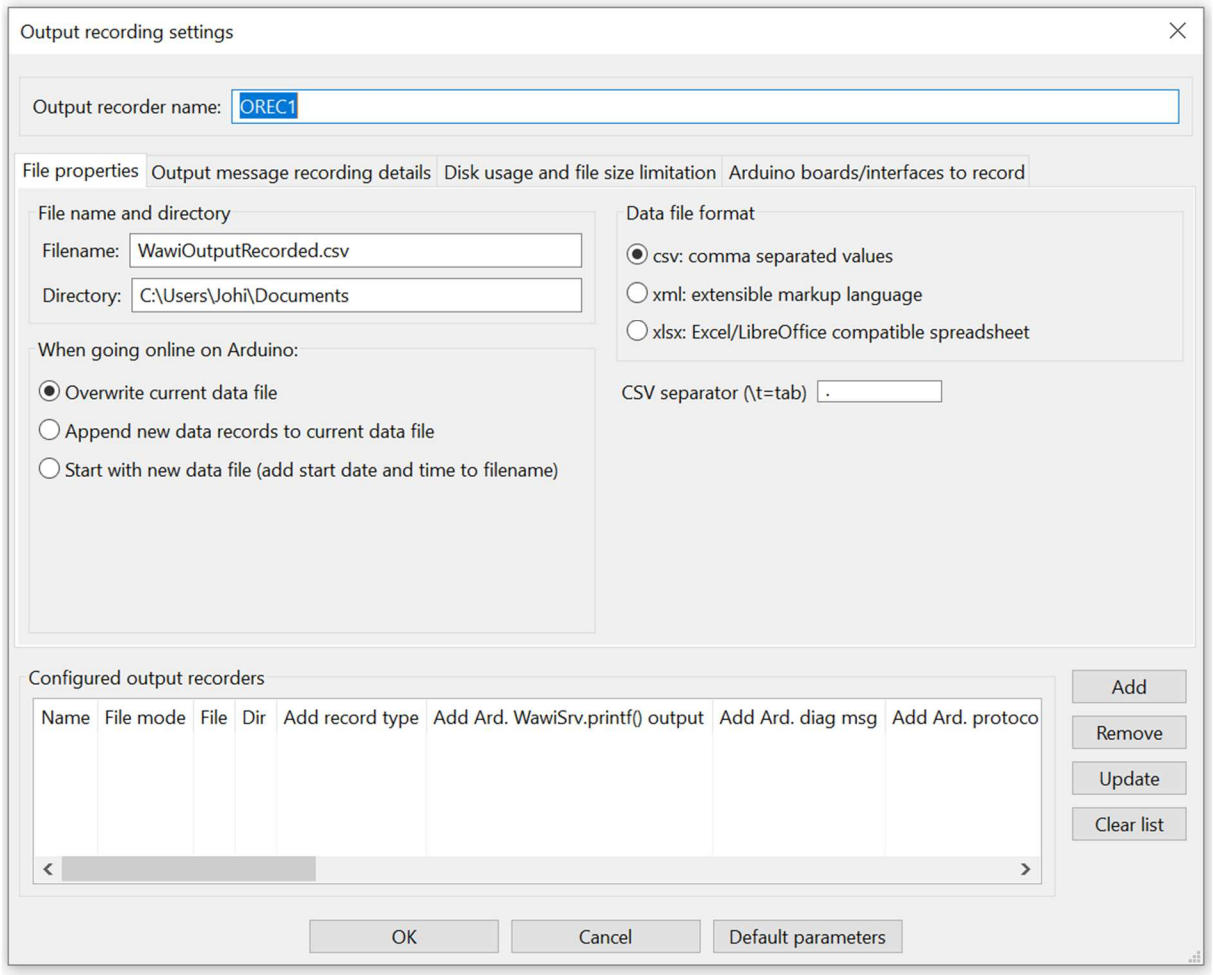


fig 9.1. Define a new output recorder.

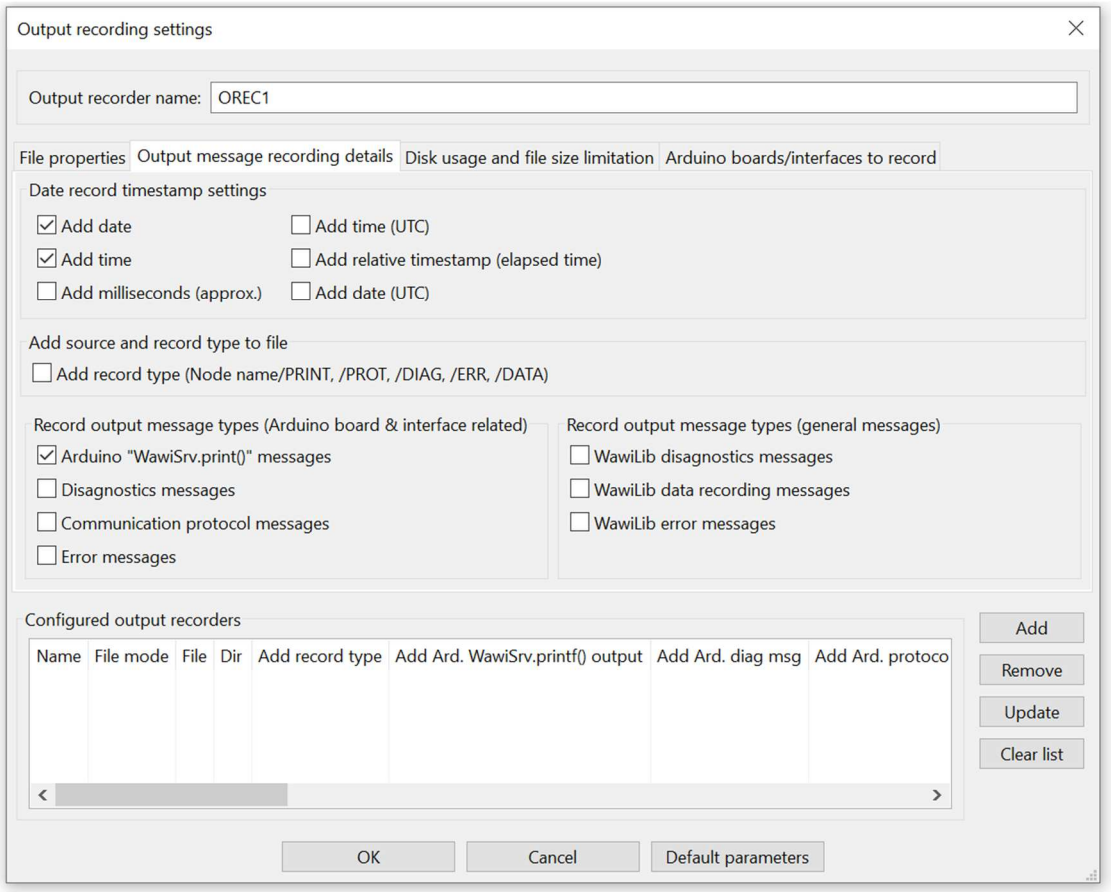


fig 9.2 Enable output “WawiSrv.print()” statements recording to disk file.

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

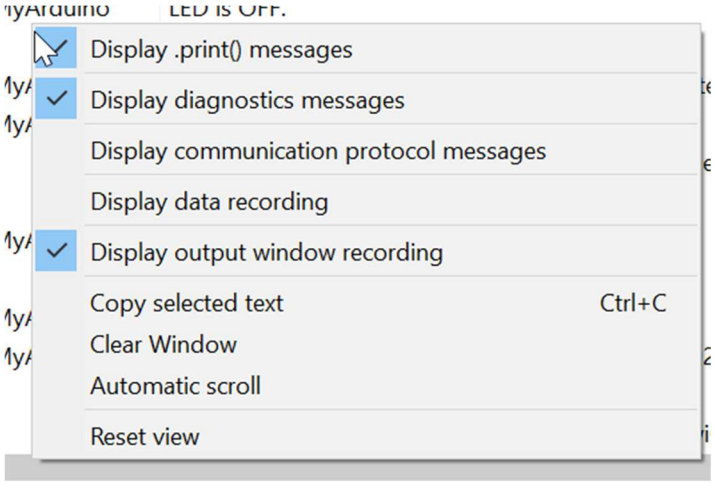


fig 9.3. Enable “Output recording” and “Display print messages” to the WawiLib output window.

- ✓ 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 relationship 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.

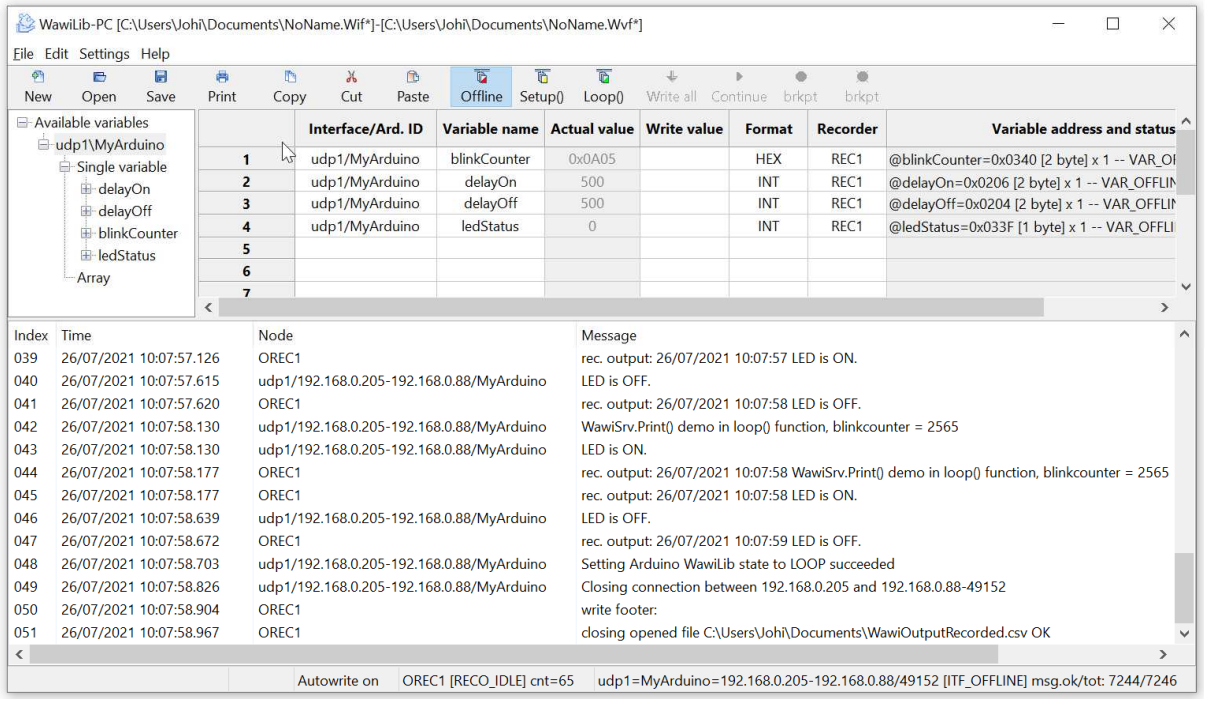


Fig 9.4. WawiLib output recording + tracing of output recording active.

```
/*
* Project Name: WawiBlinkUdpCable
* File: WawiBlinkUdpCable.ino
*
* Detailed manual:
* www.SylvesterSolutions.com\documentation -> "Getting started WawiLib
Ethernet.pdf"
*
* Description: demo file library for WawiEthernet library.
* Uses cabled ethernet network to make connection with the Arduino board.
* Blinks LED at IO 13 with variable on and off periods.
* Counts the number of blinks.
* Variables can be checked & modified with the WawiLib-PC software.
*
* Author: John Gijs.
* Created Jan 2020
* More info: www.sylvestersolutions.com
* Technical support: support@sylvestersolutions.com
* Additional info: info@sylvestersolutions.com
*/

#include <WawiEthernet.h>

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

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

// communication port Arduino side for WawiLib communication
unsigned int port = 49152; // private free port number:

// standard Ethernet UDP communication object:
EthernetUDP UDPServer;

// WawiLib communications object:
WawiEthernet WawiSrv;

// LED position assumed at digital output 13
#define LED 13

// 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 setup()
{
    // serial port for diagnostics:
    Serial.begin(115200);
    delay(2000);

    // Setup Ethernet (standard Arduino code):
    Serial.println(F("A) Initializing Ethernet UDP lib... "));
    Ethernet.begin(mac, ipArd);
    UDPServer.begin(port);
    Serial.println(F("-> Init completed."));

    // wait for board to initialize
    delay(1000);

    // Setup WawiLib server object:
    Serial.println(F("B) Initializing WawiLib ... "));
    WawiSrv.begin(wawiVarDef, UDPServer, "MyArduino");
    Serial.println(F("-> OK."));

    // Setup DO for LED:
    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 WawiBlinkEthernetUdp.ino

✓ Press “Offline”.

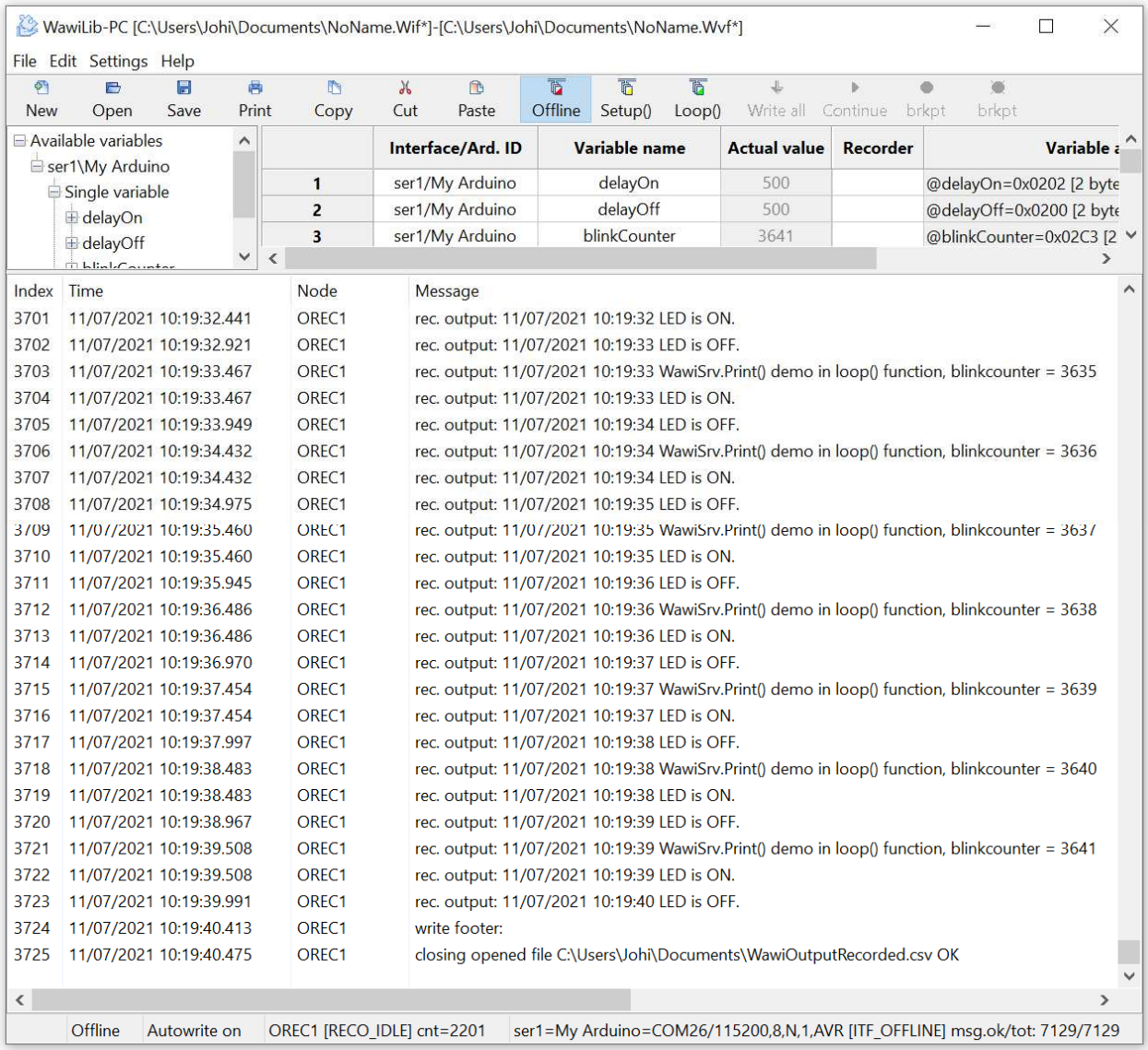
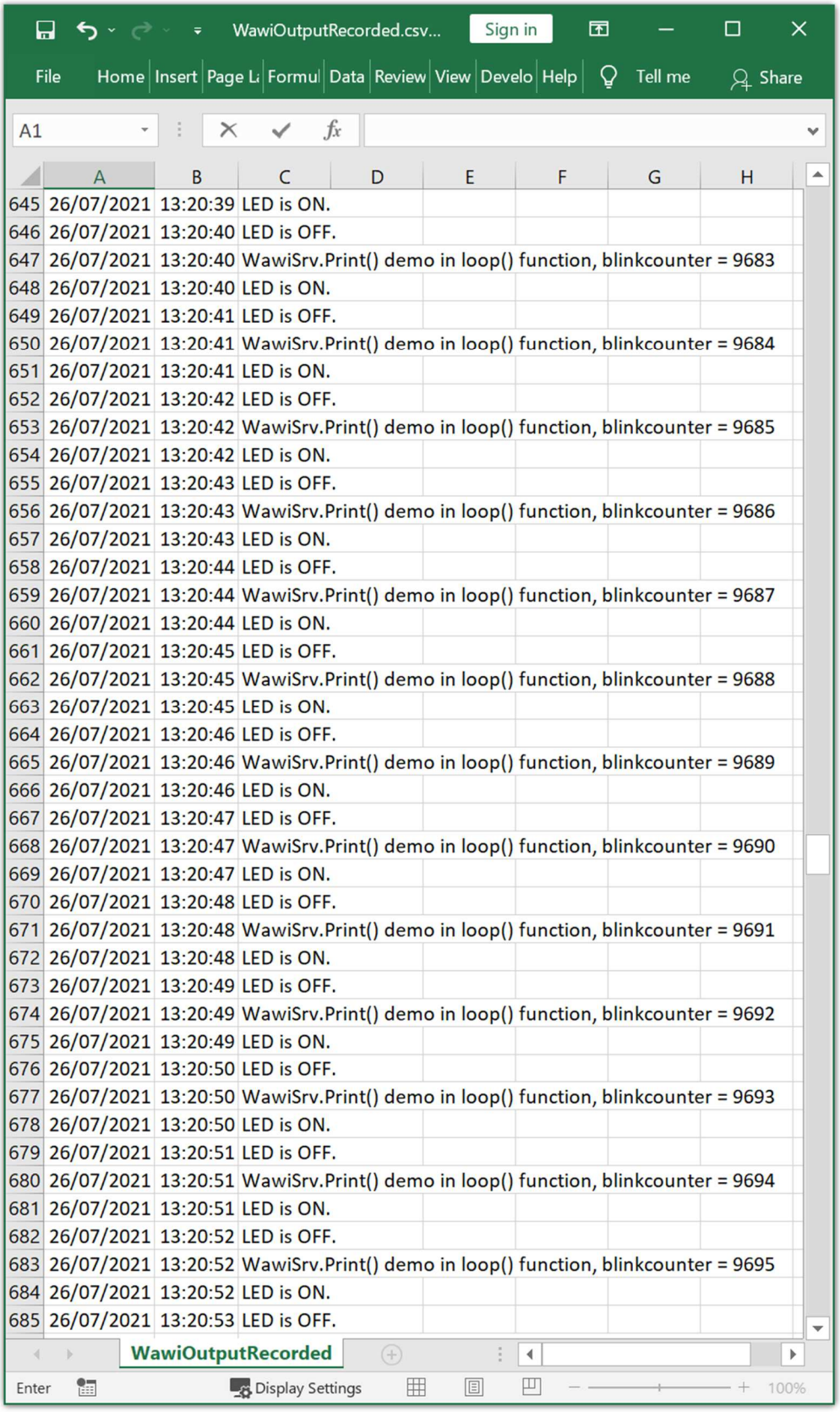


fig 9.5. Arduino output recording displayed in window.

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

```
/*
* Project Name: WawiBlinkUdpCableBreakpoint
* File: WawiBlinkUdpCable.ino
*
* Detailed manual:
* www.SylvesterSolutions.com\documentation -> "Getting started WawiLib Ethernet.pdf"
*
* Description: demo file library for WawiEthernet library.
* Uses cabled ethernet network to make connection with the Arduino board.
* Blinks LED at IO 13 with variable on and off periods.
* Counts the number of blinks.
* Variables can be checked & modified with the WawiLib-PC software.
*
* Author: John Gijs.
* Created July 2021
* More info: www.sylvestersolutions.com
* Technical support: support@sylvestersolutions.com
* Additional info: info@sylvestersolutions.com
*/

#include <WawiEthernet.h>

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

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

// communication port Arduino side for WawiLib communication
unsigned int port = 49152; // private free port number:

// standard Ethernet UDP communication object:
EthernetUDP UDPServer;

// WawiLib communications object:
WawiEthernet WawiSrv;

// LED position assumed at digital output 13
#define LED 13

// 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 setup()
{
    // serial port for diagnostics:
    Serial.begin(115200);
    delay(2000);

    // Setup Ethernet (standard Arduino code):
    Serial.println(F("A) Initializing Ethernet UDP lib... "));
    Ethernet.begin(mac, ipArd);
    UDPServer.begin(port);
    Serial.println(F("-> Init completed."));

    // wait for board to initialize
    delay(1000);

    // Setup WawiLib server object:
    Serial.println(F("B) Initializing WawiLib ... "));
    WawiSrv.begin(wawiVarDef, UDPServer, "MyArduino");
    Serial.println(F("-> OK."));

    // Setup DO for LED:
    pinMode(LED, OUTPUT);

    // Enable/disable breakpoints at startup:
    // 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()”.

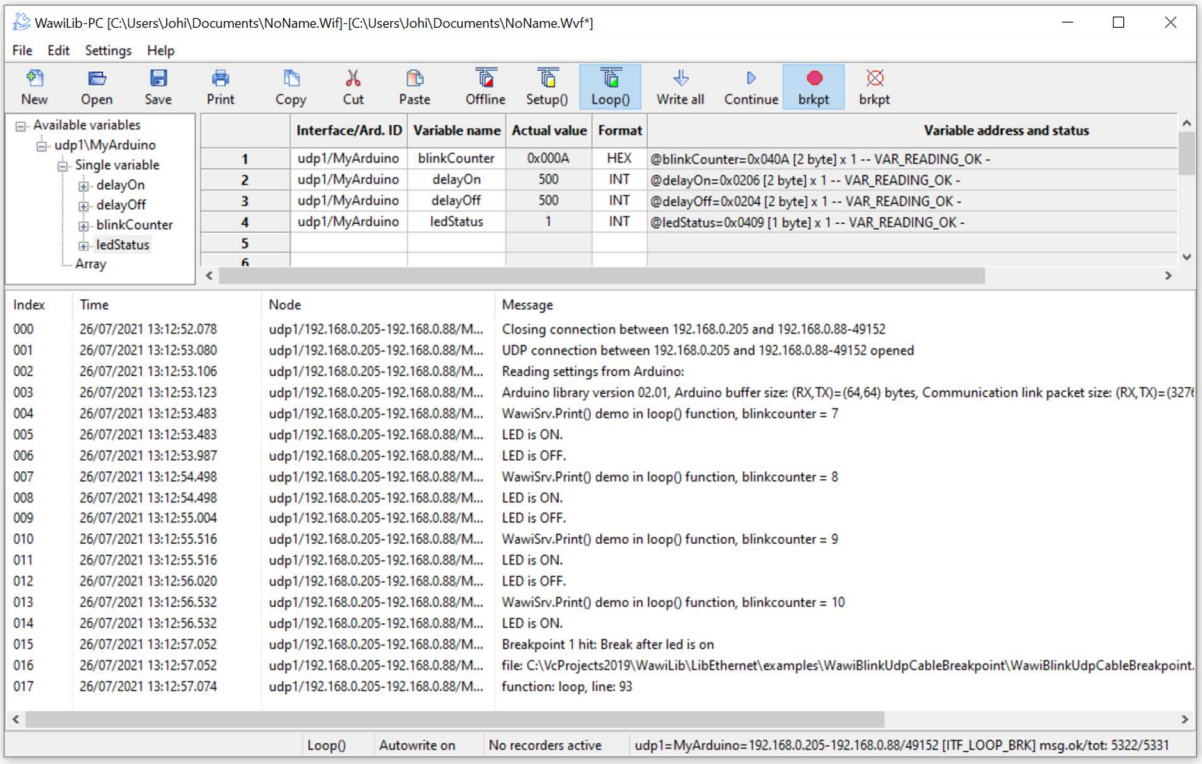


Fig 10.2. WawiBlinkUdpCableBreakpoint hit a breakpoint.

- ⇨ 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. WawiBlinkUdpCableBreakpoint 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. WawiBlinkUdpCableBreakpoint activation at startup.

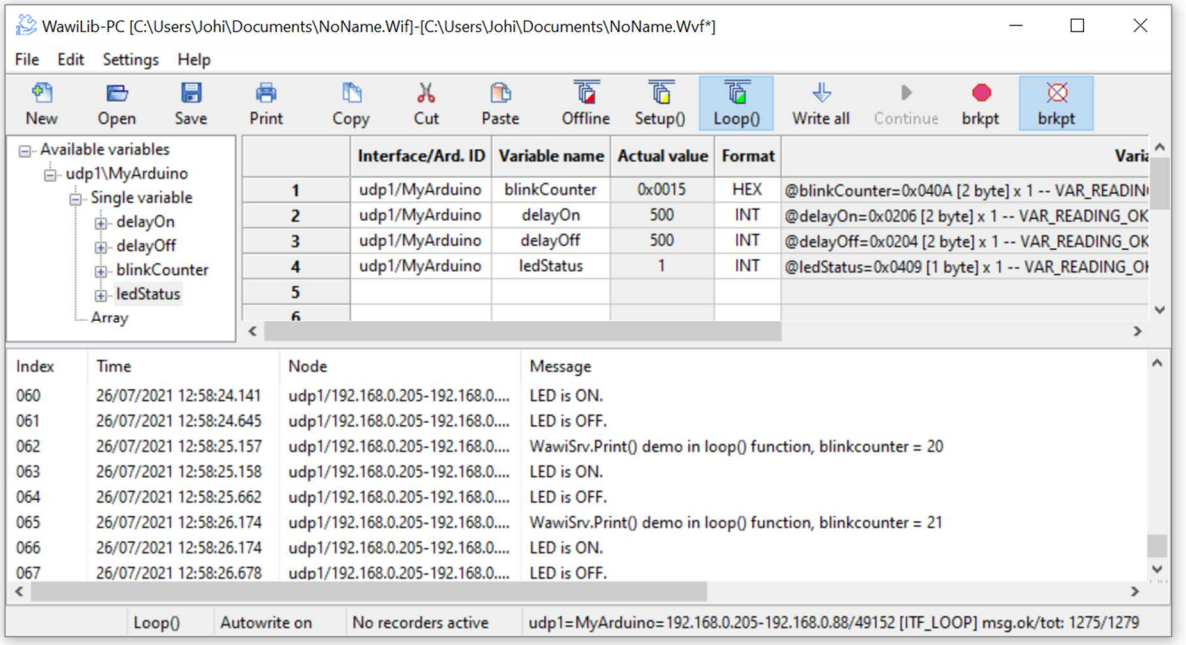


Fig 10.5. WawiBlinkUdpCableBreakpoint with breakpoints deactivated.

11. Further reading

This demo demonstrates the concept of WawiLib using a wired Ethernet connection. 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.