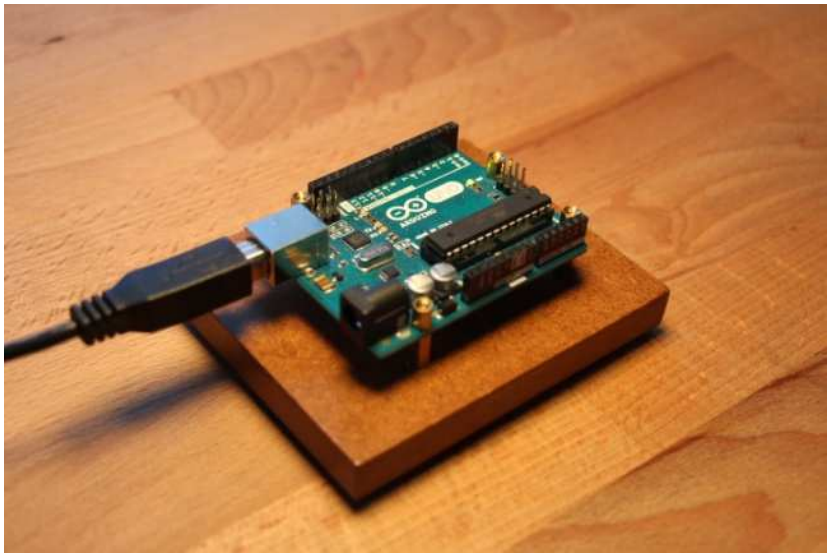
Getting started with WawiLib using the Arduino programming port



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 (sketch).

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 onboard LED.

This demo also shows how to record the output of sketch .print() statements in a disk file on your PC and the option to use 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, the USB programming port of the Arduino board is used.

The hardware you need is an Arduino board, a USB programming cable and a Windows PC (32 or 64 bit). The requirements are the same as those to run “Blink”. In this demo, we will use the Arduino UNO board but other boards can be used in a similar or even identical way.

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 you know how to edit, compile and download Arduino programs. You should also have basic computer skills such as downloading and installing Windows programs.

2. Install WawiLib Software

This section describes the steps you have to follow in order to install the WawiLib program and the WawiSerialUsb 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:

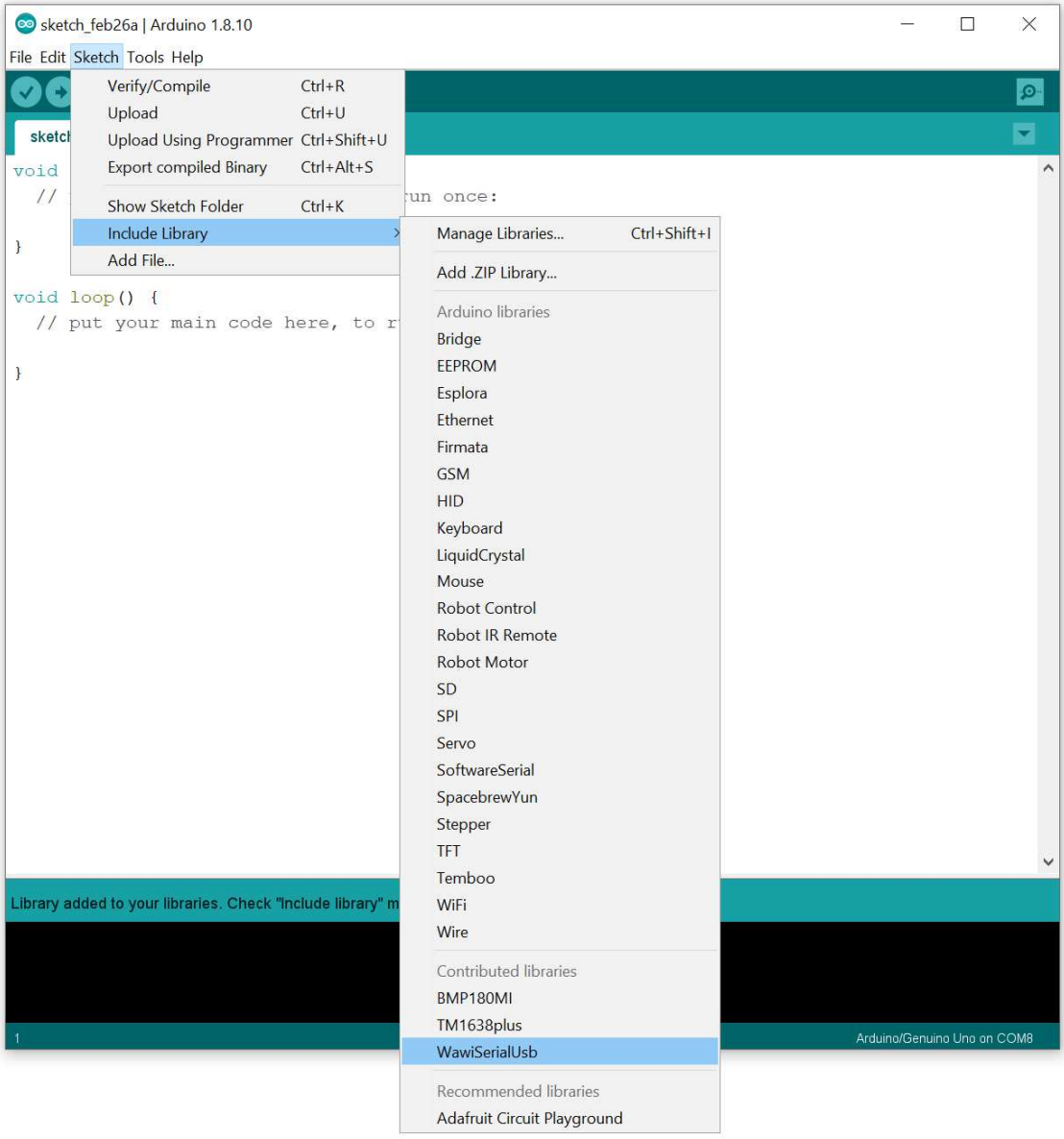


Fig. 2.1. Check the installation of the WawiLibSerialUsb library in the Arduino IDE.

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

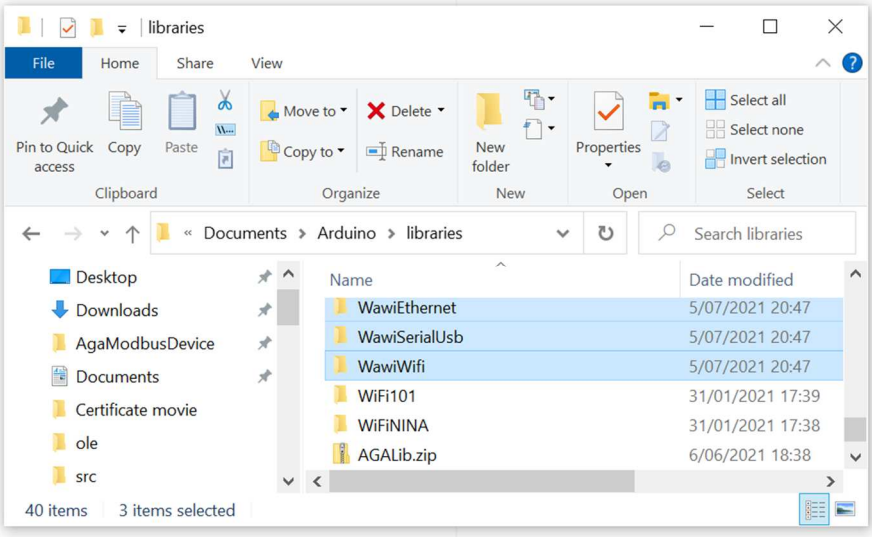


Fig. 2.2. Unpacked Libraries after installing WawiLib.

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".

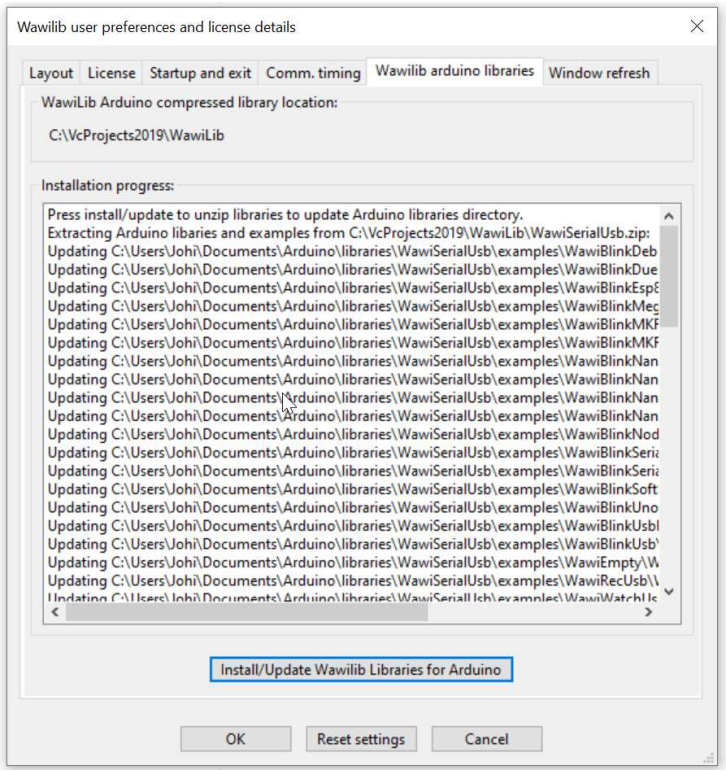


Fig. 2.3. Manual install of Arduino libraries.

3. 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 WawiBlinkUsb.ino.

- ✓ Go to File\Examples\WawiSerialUsb\WawiBlinklUsb in the Arduino IDE.
- ✓ Open and compile WawiBlinkUsb and download the sketch to your Arduino board.
- ✓ Check if the program was properly downloaded by looking at the LED on the board. The LED should blink 500ms on and 500ms off.

```
#include<WawiSerialUsb.h>
WawiSerialUsb WawiSrv;
#defineLED 13

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

// make variables of interest known to WawiLib:
// thisfunction is used in WawiSrv.begin(....)
void wawiVarDef()
{
    WawiSrv.wawiVar(delayOn);
    WawiSrv.wawiVar(delayOff);
    WawiSrv.wawiVar(blinkCounter);
}
void setup()
{
    Serial.begin(115200);
    WawiSrv.begin(wawiVarDef, Serial, "My Arduino");
    pinMode(LED, OUTPUT);
}
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. 3.1. Minimal Arduino example WawiBlinkSerial.ino

4. WawiLib user interface overview

✓ Start WawiLib on your PC:

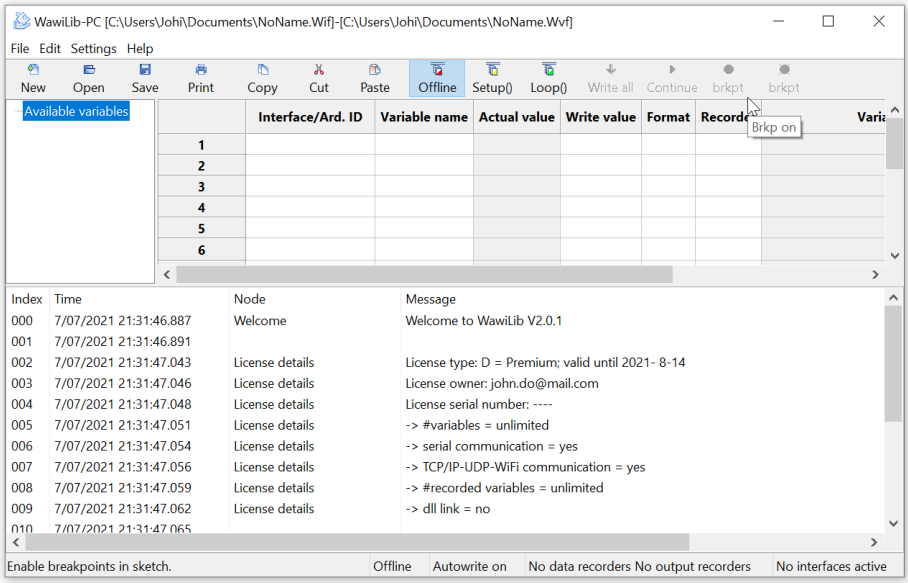


Fig. 4.1. WawiLib startup screen

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:

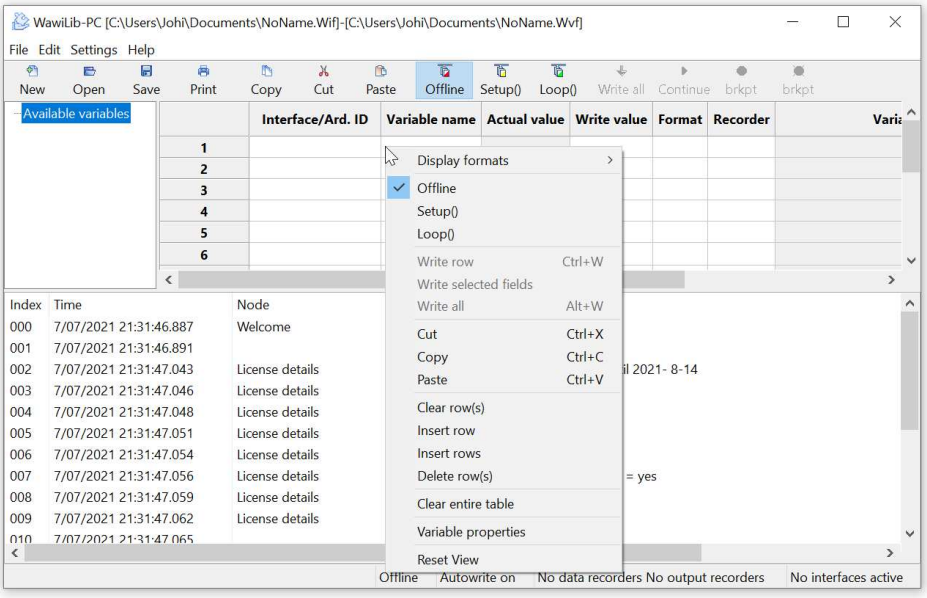


Fig. 4.2. WawiLib grid options.

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:

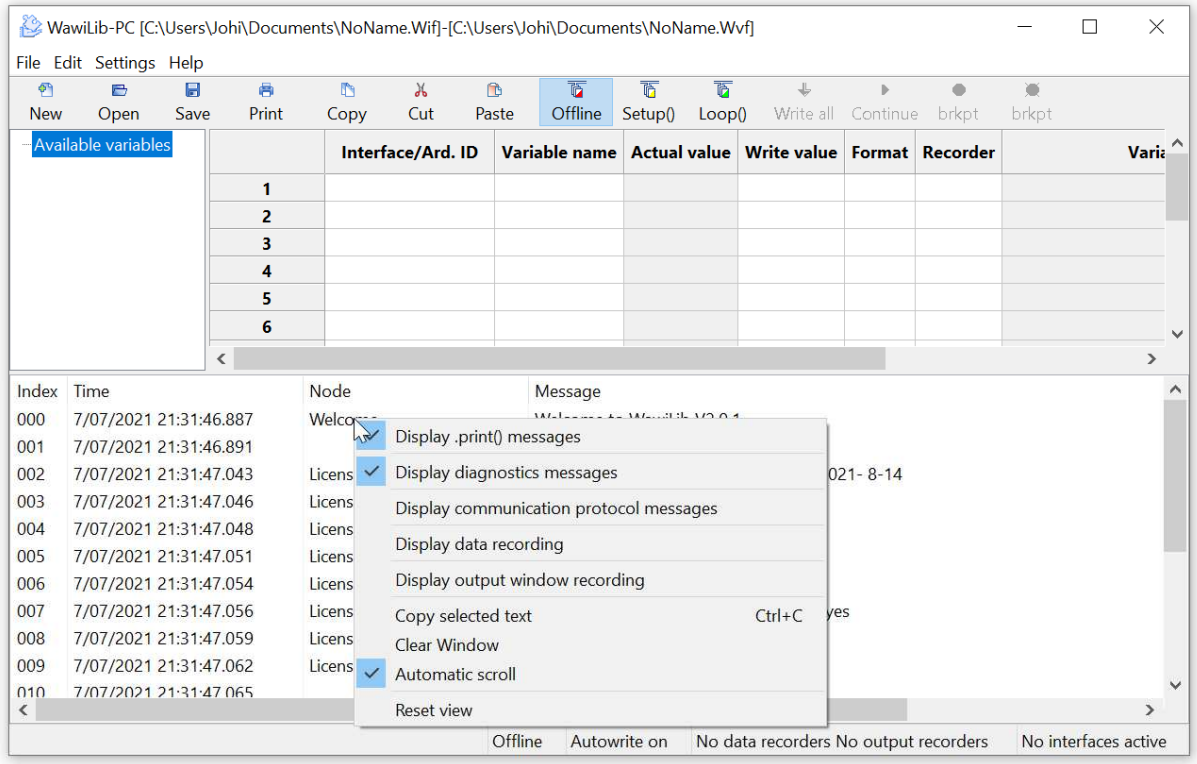


Fig. 4.3. WawiLib output window options.

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 WawiSerialUsb.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.

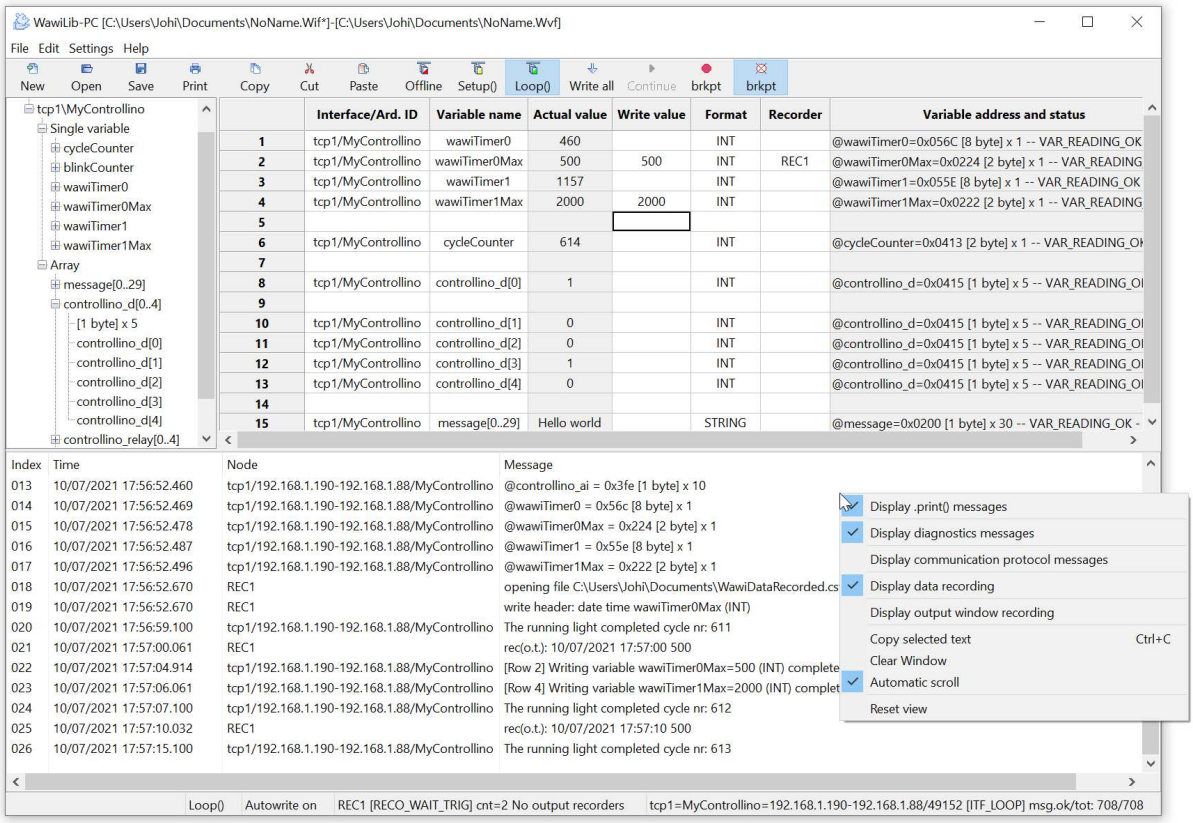


Fig. 4.4. WawiLib overview in a Controllino TCP configuration.

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.

5. WawiLib Serial-USB communication link setup

One of the biggest challenges going online on a board is finding the right port and the right settings. With this purpose in mind, WawiLib has a wizard to scan serial/USB ports with various settings to check for the presence of one (or multiple) Arduino board(s).

✓ Select the WawiLib menu Settings\Communication interfaces:

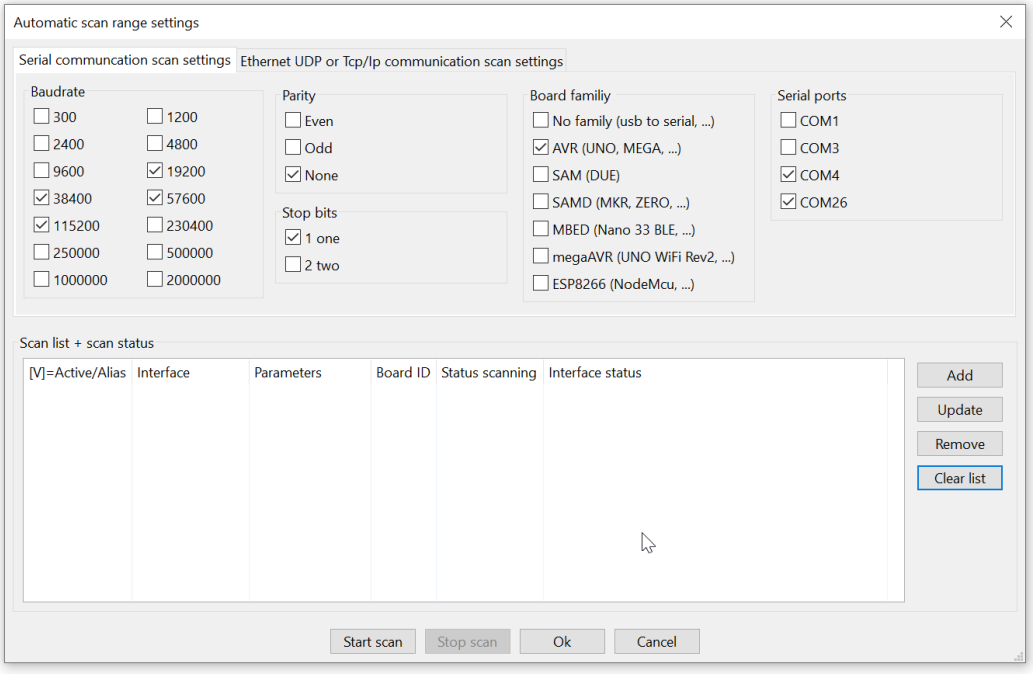


Fig. 5.1. Serial communication setup.

✓ Select the baud rates, board type(s) and serial port(s) that you want to check.
✓ Press the button “Add”:

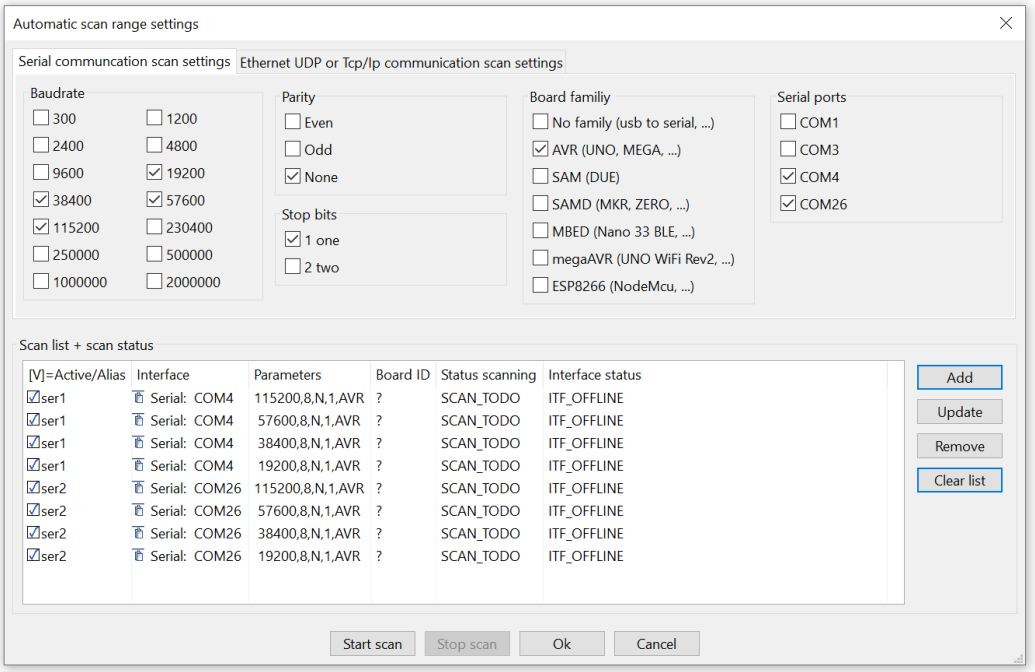


Fig. 5.2. Serial communication setup: scanning/setting up the connection.

⇨ You will see all combinations listed up in the “Scan list + scan status” table.
✓ Press the button “Start scan”.

WawiLib will try all setting combinations in the table one by one. When the scan is completed successfully, one (or more) icon(s) in the scan list will turn green. You can follow the process in the output window of WawiLib and in the “Scan list + scan status” window.

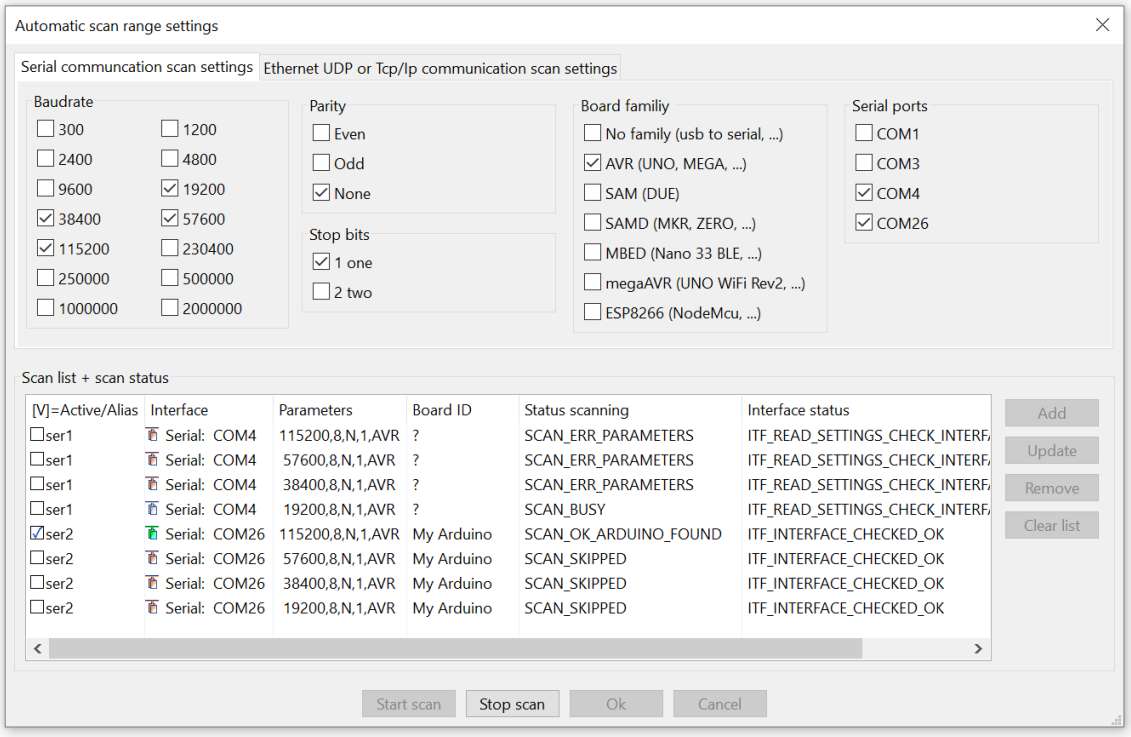


Fig. 5.3. Serial communication setup: indicating successful check.

In the table above, you see in the column “Arduino board ID” the parameter that was part of the WawiLib.begin(…) statement in the Sketch (see Fig. 3.1.). Here you can keep track if you use multiple Arduino boards.

✓ Click right in the list view and select “Remove Inactive”:
⇨ WawiLib will remove all interfaces that have failed their scan test.

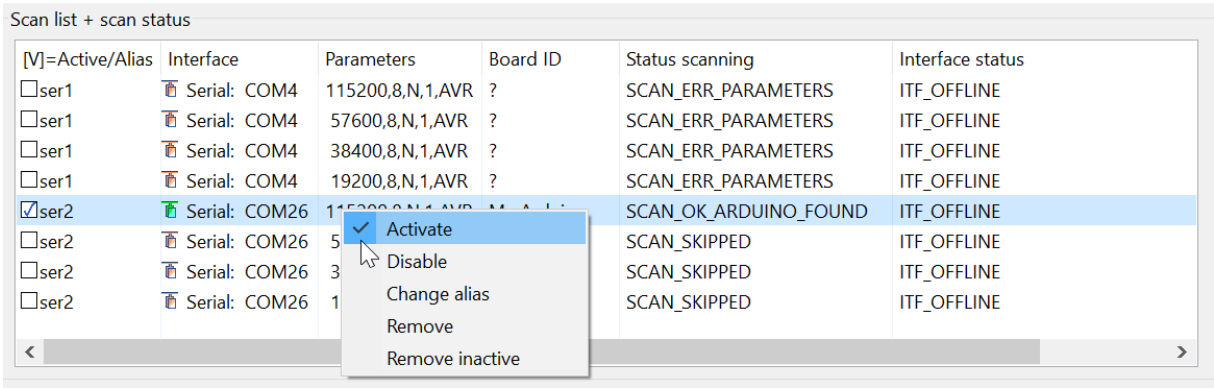


Fig. 5.4. Serial communication setup: remove failed interfaces.

- ✓ Press “OK” to close the “Scanning completed” dialog box.
- ✓ Press “OK” to close the “Automatic scan range settings” dialog box.

At this time, the connection parameters are identified.

- ✓ Activate “Display .Print() messages” and “Display communication protocol messages” window. (Click right and use the output window menu.)
- ✓ Press “Setup()” in the main window toolbar.

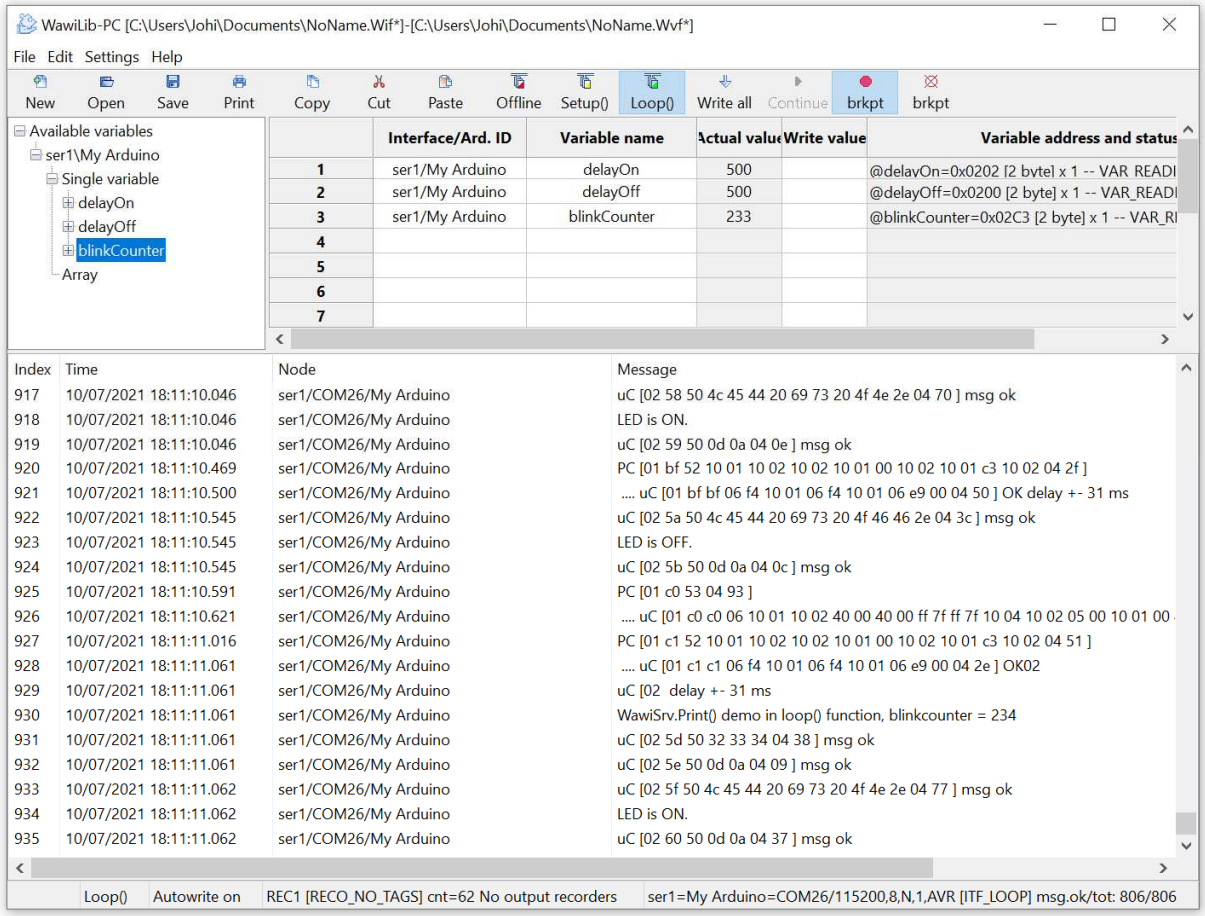


Fig. 5.5. Serial communication setup: remove failed interfaces.

WawiLib will establish a connection to the Arduino board using the parameters as they were identified in the section above (connections to multiple boards at the same time are supported).

Notes:

- If you are using different types and families of Arduino boards, they all use different types of RTS/DTR handshake types. In order to resolve this, you can select different board families.
- Be careful (do not use) with 1200 baud as this can trigger a function on the Arduino Due that starts a firmware reset.
- When you open a “Serial Monitor Window” in the Arduino IDE, this can trigger a reset on some boards. A reset of your board can be triggered in the same way when you make a connection between WawiLib and your Arduino board.

- If you do not know which port to use for your Arduino connection, you can select all of them at the same time and press “add” and scan. The right combinations will light up in green after scanning.

6. Read and write variables with WawiLib

6.1. Watch variables

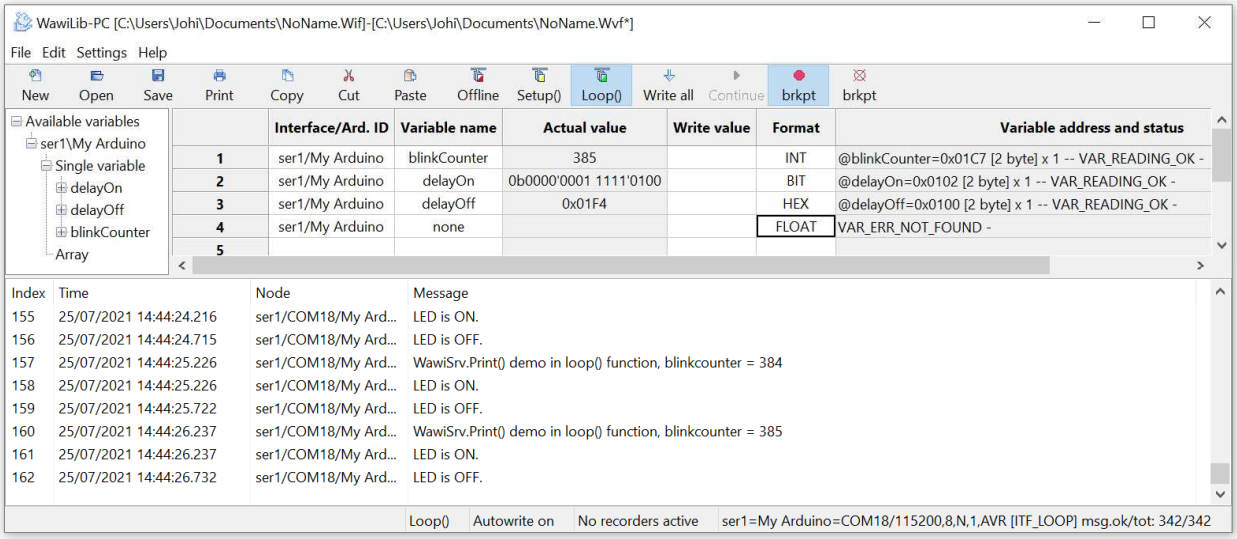


Fig. 6.1. Add variables to the grid using drag & drop.

- ✓ 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. 6.1.

The “Interface/Arduino ID” column will be filled in automatically as there is only 1 board active.

You can also click right on the grid and select “Available interfaces”. Any active interfaces can be selected using this menu. This option is used to exchange data with multiple boards at the same time.

Do note the “Variable address and status” column: @delayOn=0x0102 means this variable is located at address 0x0102 in the Arduino board and its size is 2 bytes. The statement x 1 indicates that this variable is not an array. VAR_READING_OK is the status of the variable data exchange FSM.

Look at line 4: if a variable cannot be found in the Arduino board, the status of the FSM is VAR_ERR_NOT_FOUND.

6.2. Modify variables

In the upper window, you see the actual value of the variables. In the bottom window, you see the communication telegrams that are exchanged over USB with your Arduino board when it is online.

- ✓ In the output window, disable all output but enable “Display diagnostics messages”.
- ✓ Fill in 100 as new value for *delayOff* in the write column.
- ✓ Press “Write all”.
- ⇨ You see the actual value of delayOn change to 100 (in the upper window). The time the LED is on will change to 100ms.
- ⇨ In the bottom window, you see the result of your write action, if there is a format error, it will be displayed here.

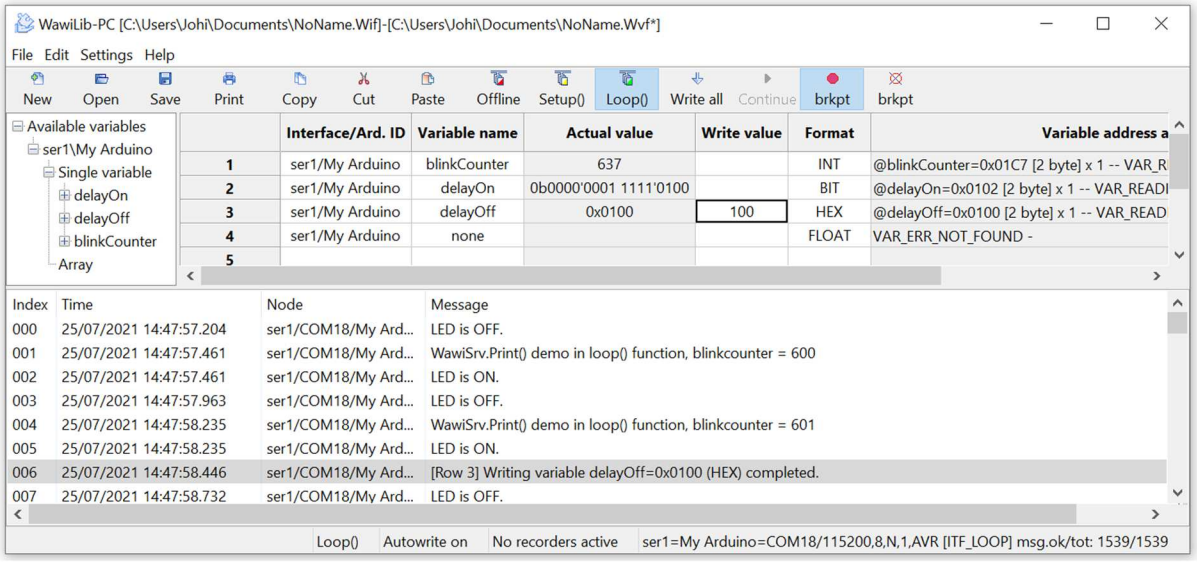


Fig. 6.2. Change the value of the parameter delayOff.

In the bottom window you can find the following information:

- The target status of the communication interfaces is “Loop()”.
- “Autowrite” is on.
- No data recorders were defined.
- No output recorders were defined.
- ser1 corresponds to an Arduino named “My Arduino”.
- COM 18 is the (virtual) USB serial port used.
- Baud rate = 115200 bits/second.
- 8 data bits.
- N=No parity.
- 1=1 stop bit.
- AVR=AVR family type of board.
- The state of the communication interface Ser1 is ITF_LOOP.
- 1539 message exchanges between PC and Arduino have been executed successfully.
- 1539 message exchanges between PC and Arduino have been executed in total.

7. Record variable to file (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.

- ✓ Open the menu “Settings/Data Recording” in the main window.
- ✓ Press “Clear list”.
- ✓ Select as data file format in the first tab: xlsx “Excel/LibreOffice compatible spreadsheet”.
- ✓ Select “Overwrite current data file”.
- ✓ Go to the second tab. (Fig.. 7.2.)
- ✓ Select 10 seconds as time base.
- ✓ Press “Add”:

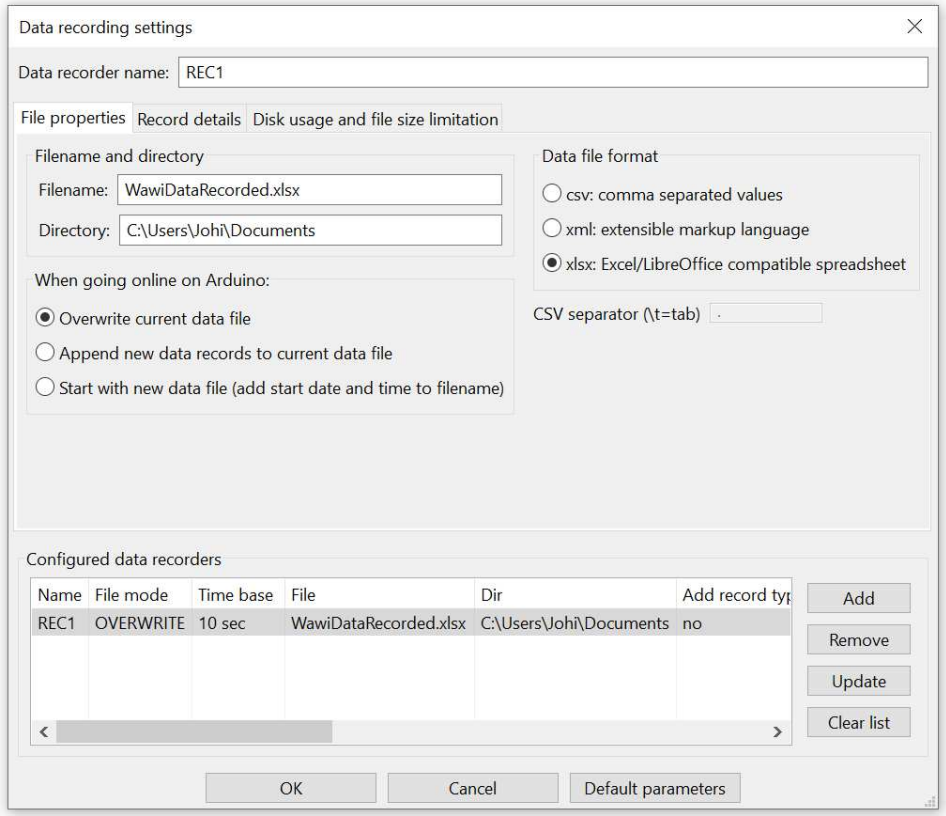


Fig. 7.1. Define a new data recorder.

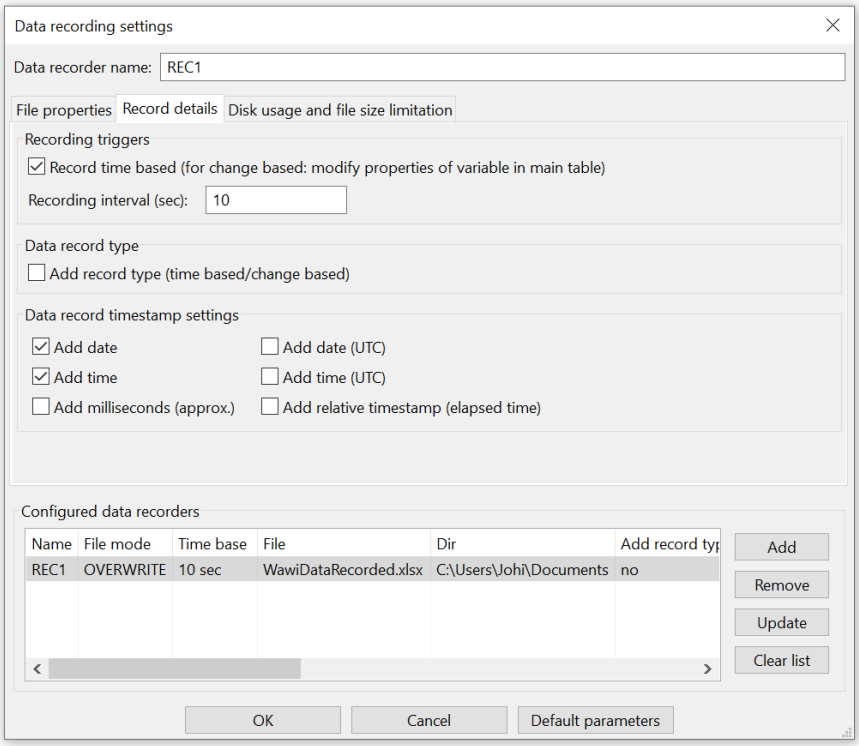


Fig. 7.2. Define a new data recorder time base = 10 sec.

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

- ✓ Press “OK” to close the dialog box.
- ✓ In the main grid, select all variables recorder fields (Fig. 7.3 in blue), click right and select “Available data recorders/Rec1”:

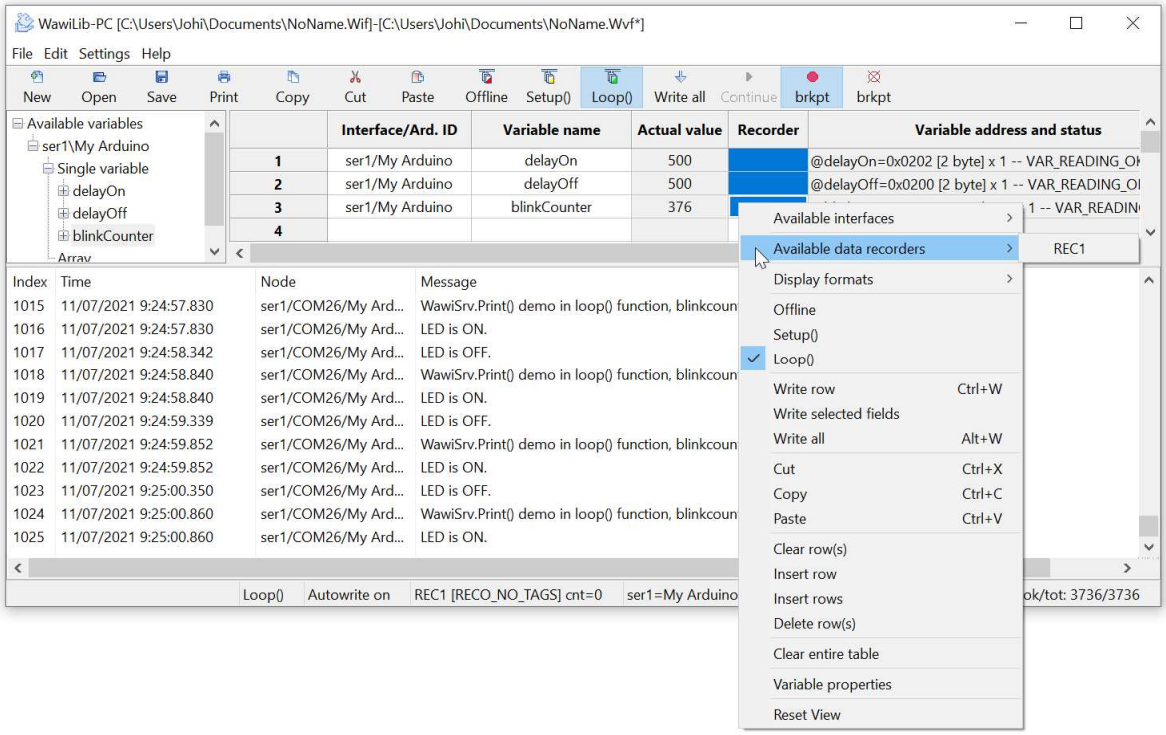


Fig. 7.3. Link all the variables to a data recorder REC1.

- ✓ Disable the “Trace protocol” option in the output window.
- ✓ Enable “Automatic scroll” & “Display data recording”
- ✓ Disable all other options ad indicated in Fig. 7.4.:

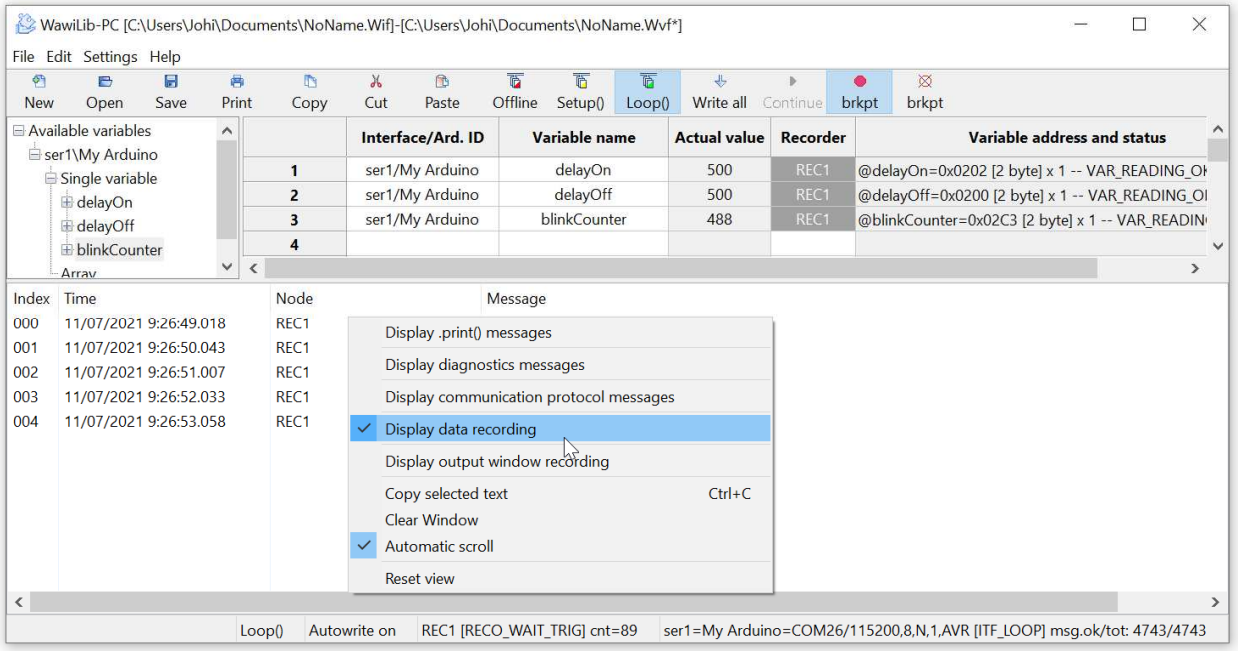


Fig. 7.4. Enable display the recorded data in the output window.

- ✓ Press “Setup()”.

You will now see the different values of your variables as they are written to the .xlsx file in the output window.

- ✓ Press “Offline”.

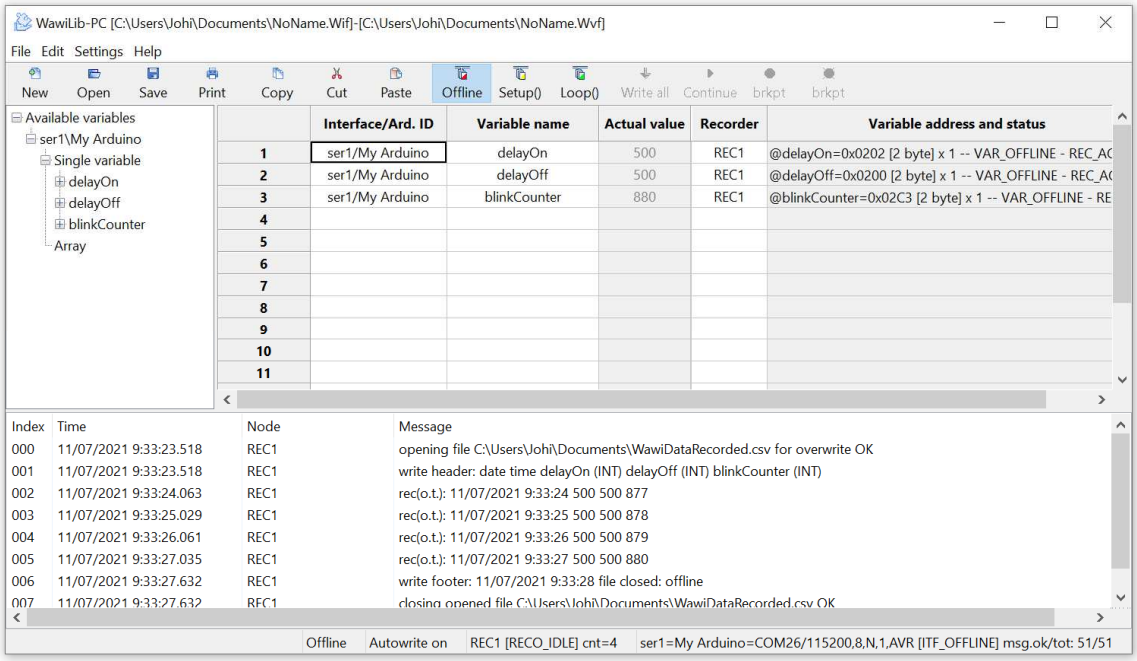
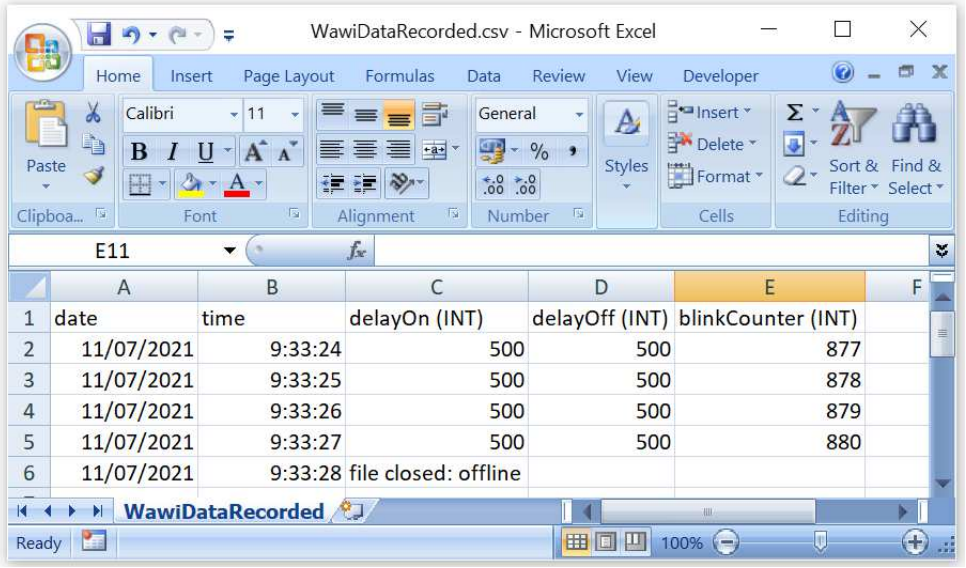


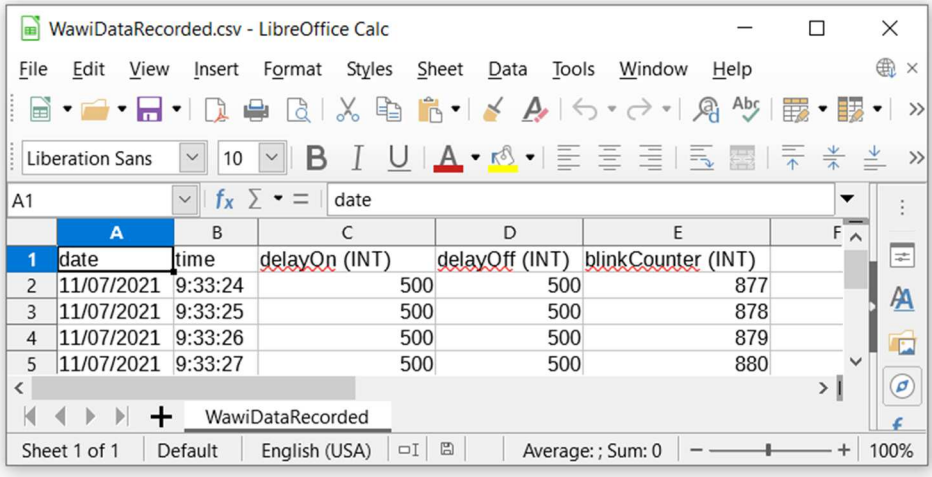
Fig. 7.5. After going offline, the output window displays the name of the file created.

✓ Open the recorded .xlsx file in Microsoft Excel or equivalent.



	A	B	C	D	E	F
1	date	time	delayOn (INT)	delayOff (INT)	blinkCounter (INT)	
2	11/07/2021	9:33:24	500	500	877	
3	11/07/2021	9:33:25	500	500	878	
4	11/07/2021	9:33:26	500	500	879	
5	11/07/2021	9:33:27	500	500	880	
6	11/07/2021	9:33:28	file closed: offline			

Fig. 7.6. Data file opened in Microsoft Excel.



	A	B	C	D	E	F
1	date	time	delayOn (INT)	delayOff (INT)	blinkCounter (INT)	
2	11/07/2021	9:33:24	500	500	877	
3	11/07/2021	9:33:25	500	500	878	
4	11/07/2021	9:33:26	500	500	879	
5	11/07/2021	9:33:27	500	500	880	

Fig. 7.7. Data file opened in LibreOffice Calc.

You can see the date, the time, the relative timestamp, the type of record (on timer or on change recording), the name of the variables and their value in the Excel table.

8. 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 .txt 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.
- ✓ Select Arduino WawiSrv.print() messages (see Fig.. 8.2.)
- ✓ Press “Add”:

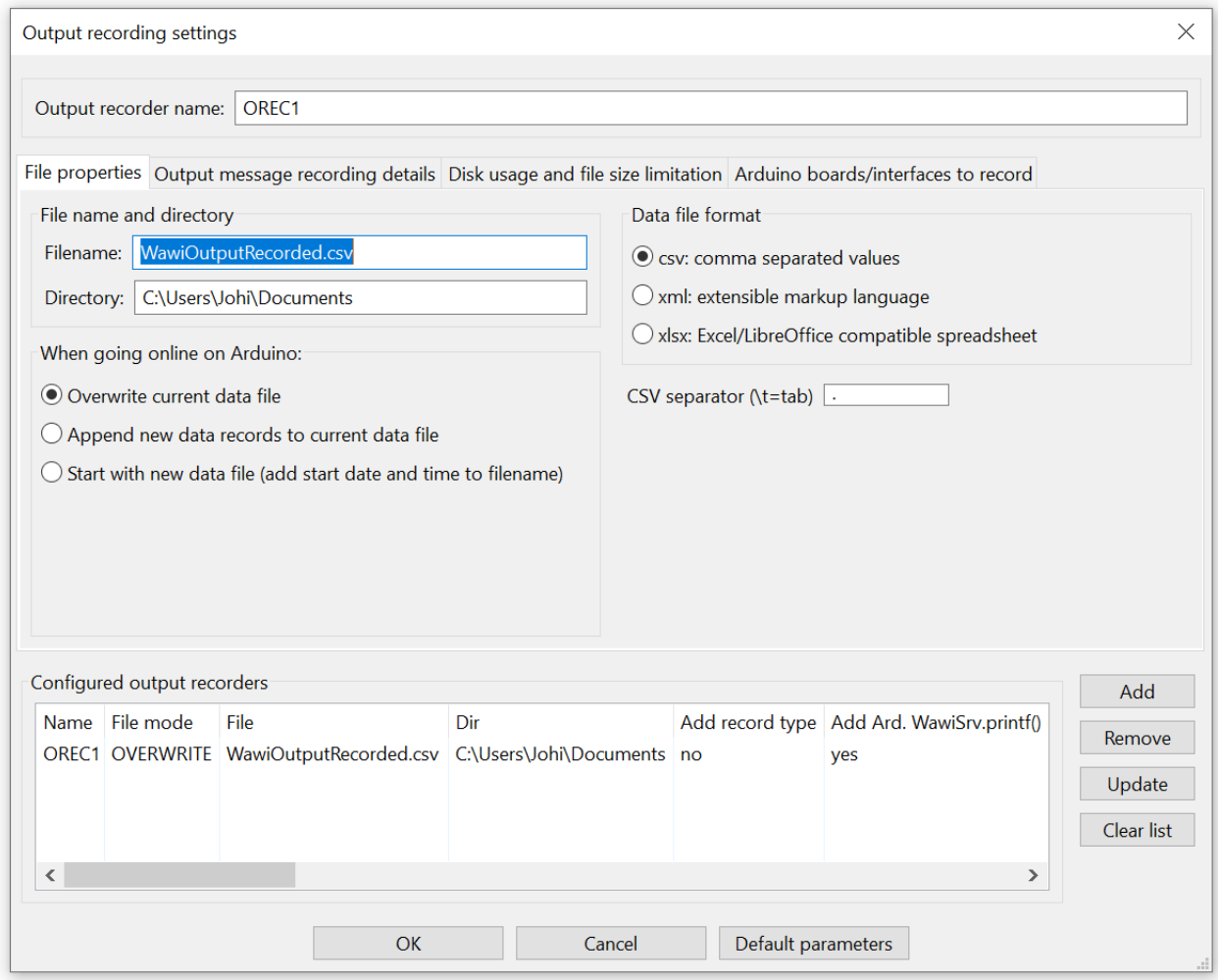


Fig. 8.1. Define a new output recorder.

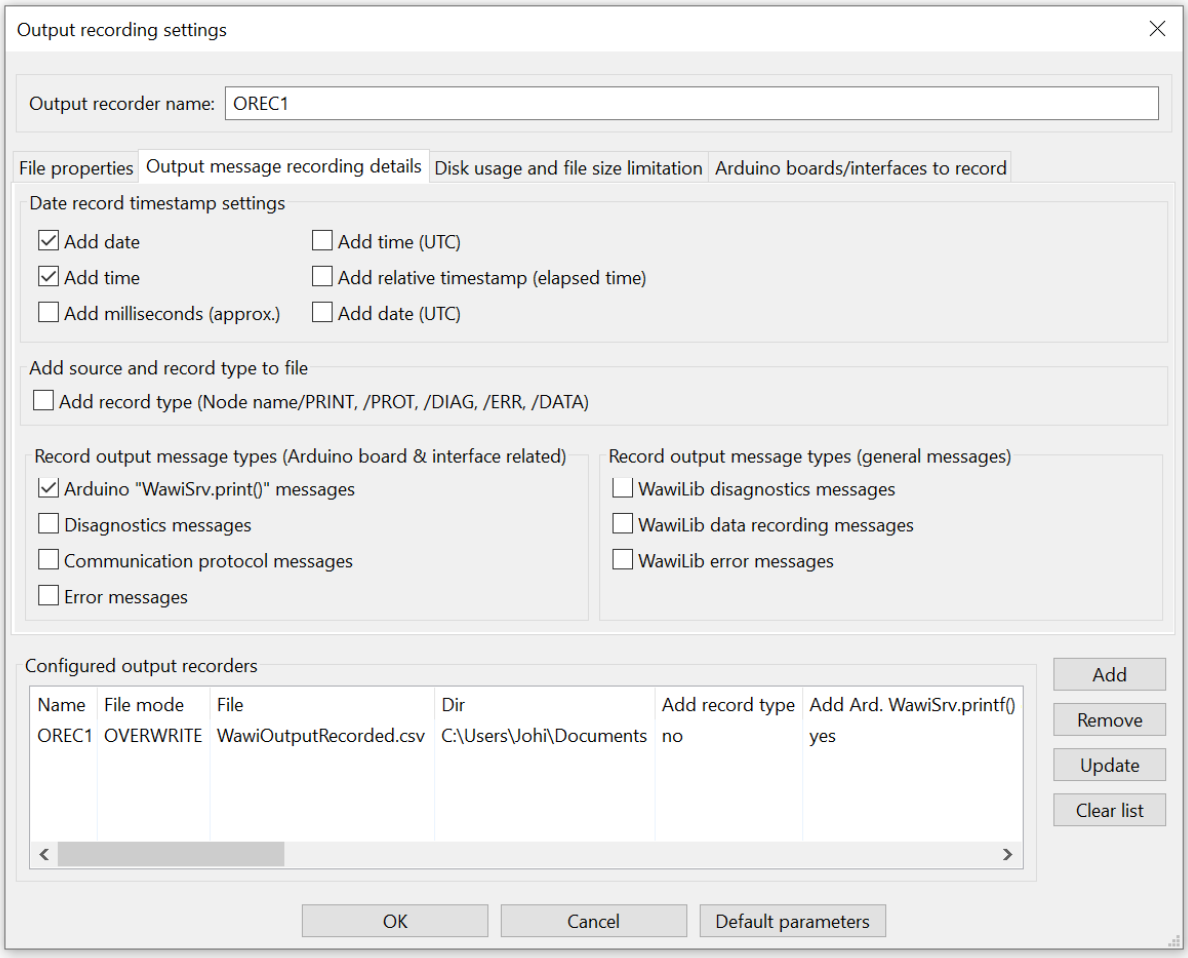


Fig. 8.2. Select subset of output recorder messages to be recorded.

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

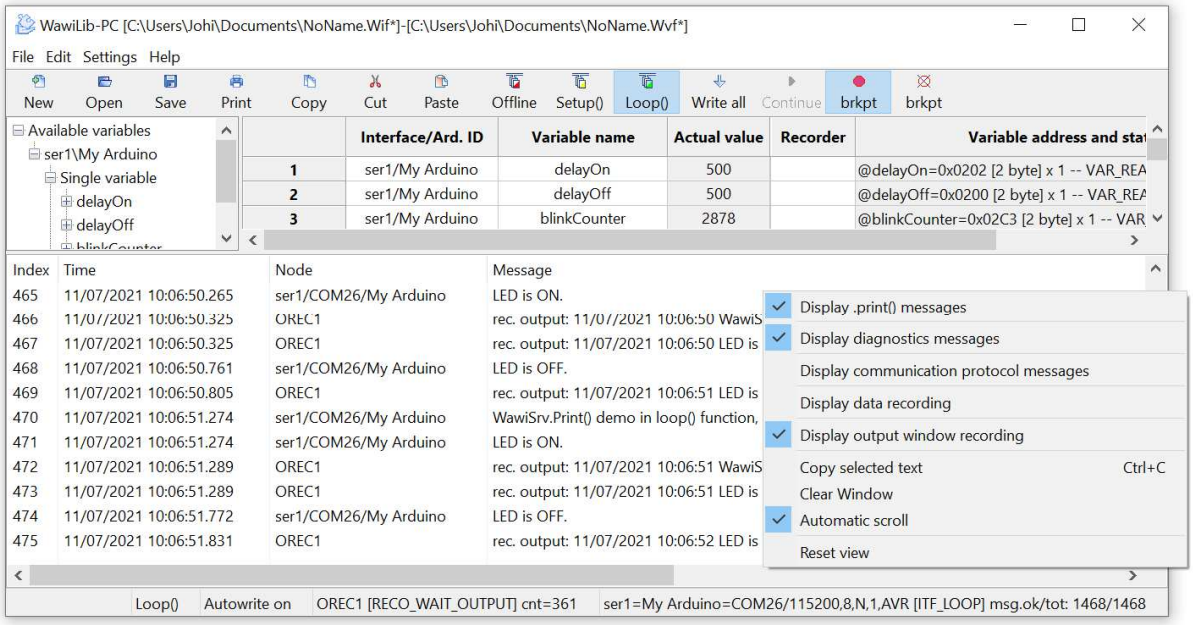


Fig. 8.3. Enable Display print() messages & Display output window recording.

✓ Press “Setup()”.

You will now see the output of the print() statements in the output window. The output is generated in your sketch by the lines indicated in yellow in Fig.. 24.

Do note (Fig.. 8.5.) that the output itself is displayed and what is written to file as well. (See Node column.) The settings of the output window do not influence the recoding itself.

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

```
#include<WawiSerialUsb.h>
WawiSerialUsb WawiSrv;
#defineLED 13

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

// make variables of interest knownto WawiLib:
// thisfunction is used in WawiSrv.begin(....)
void wawiVarDef()
{
    WawiSrv.wawiVar(delayOn);
    WawiSrv.wawiVar(delayOff);
    WawiSrv.wawiVar(blinkCounter);
}
void setup()
{
    Serial.begin(115200);
    WawiSrv.begin(wawiVarDef, Serial, "My Arduino");
    pinMode(LED, OUTPUT);
}
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. 8.4. Minimal Arduino example WawiBlinkSerial.ino

✓ Press “Offline”.

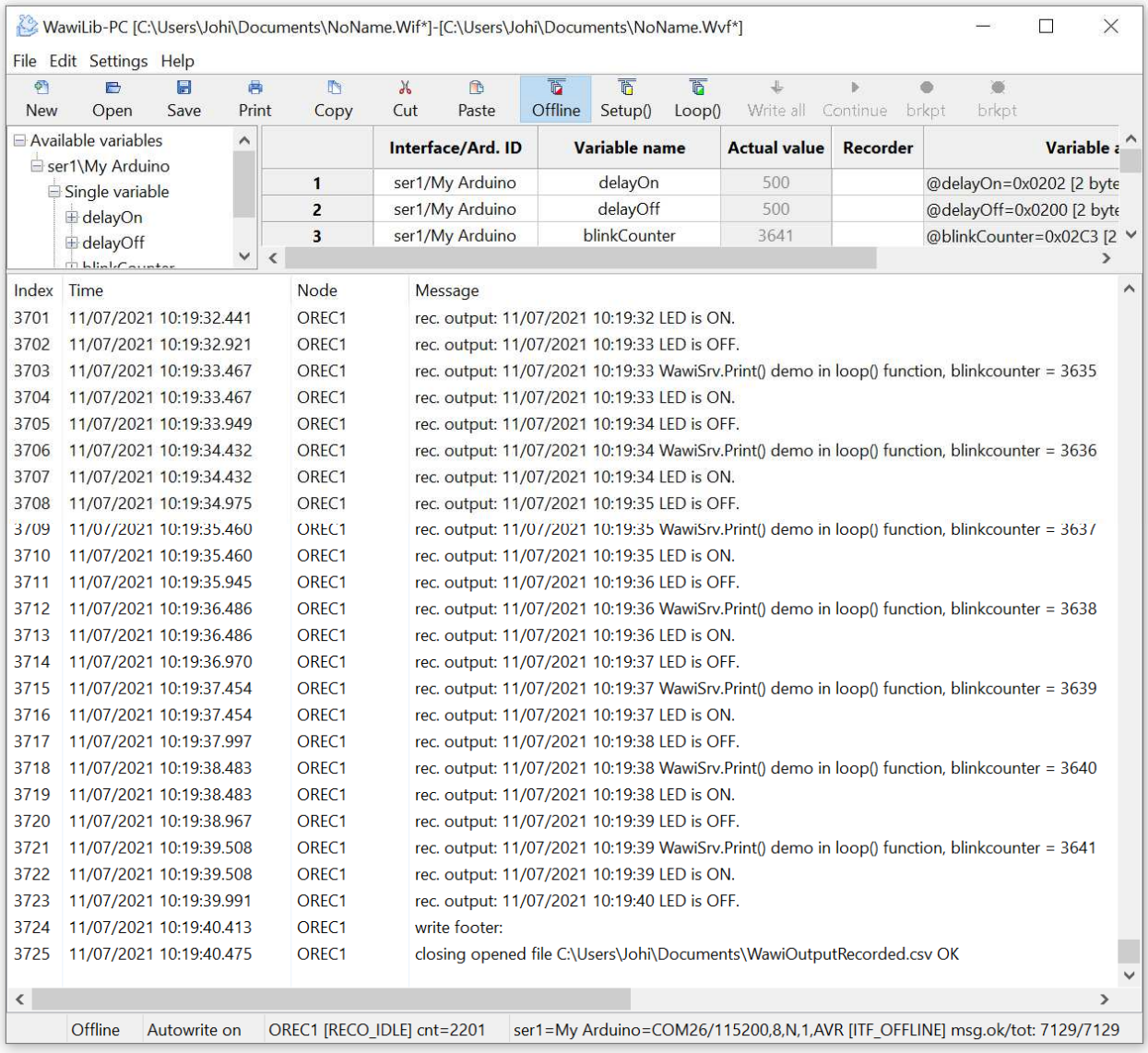


Fig. 8.5. Arduino output recording displayed in window.

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

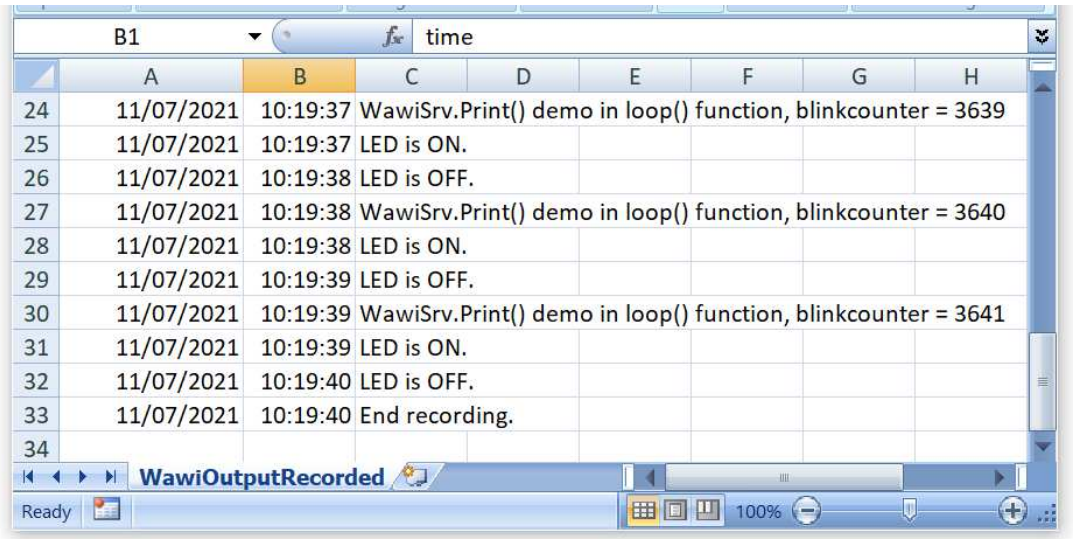


Fig. 8.6. Arduino output recording file opened in Excel.

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.

9. Introduction to WawiLib breakpoints

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

```
#include <WawiSerialUsb.h>
WawiSerialUsb WawiSrv;
#define LED 13

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

// 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(led);
}

void setup()
{
    Serial.begin(115200);
    WawiSrv.begin(wawiVarDef, Serial, "My Arduino");
    pinMode(LED, OUTPUT);
    WawiSrv.wawiBreakDisable();
}

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

    WawiSrv.loop();
}
```

Fig. 9.1. WawiLib breakpoint support demo.

✓ Add the variables to the grid as indicated in Fig. 9.2.

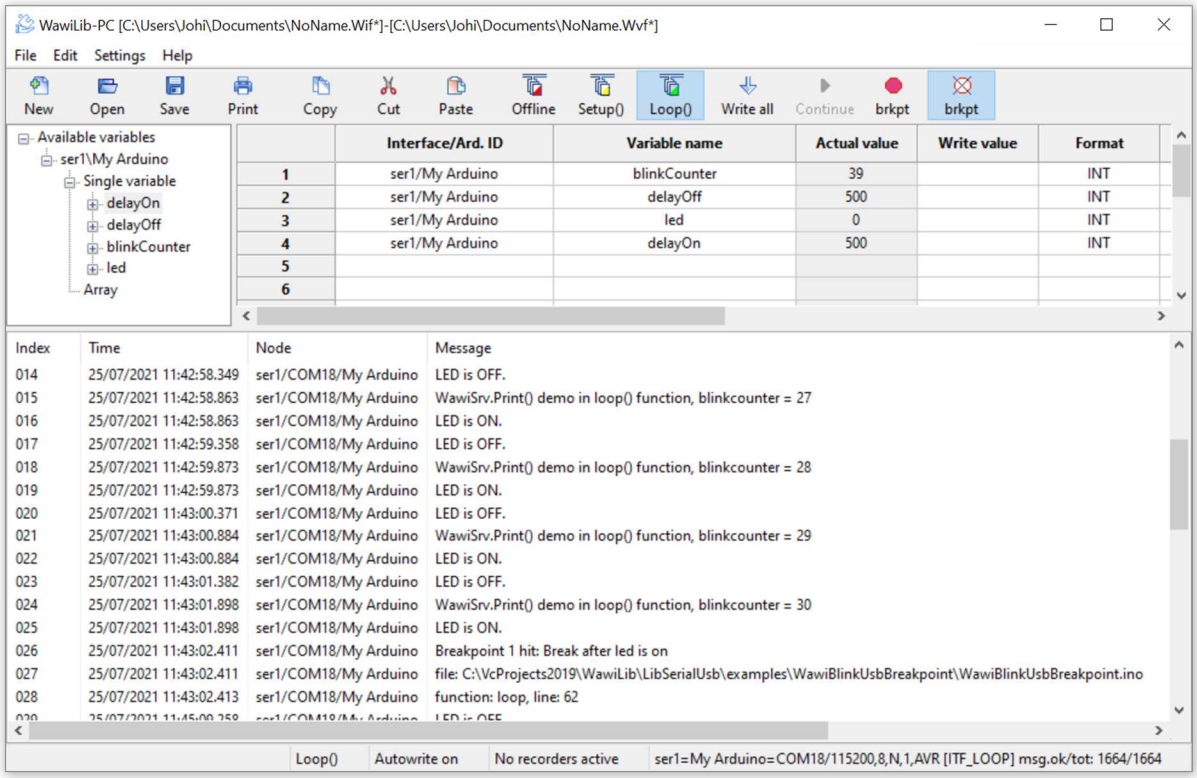


Fig 9.2. Variables of WawiBlinkUsbBreakpoint added to grid.

✓ Press “brkpt” 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");
```

⇨ The output window also contains the source file, the function and the source line where the breakpoint was hit.

✓ Press “continue” in the toolbar.

✓ The sketch will run further another breakpoint is hit.

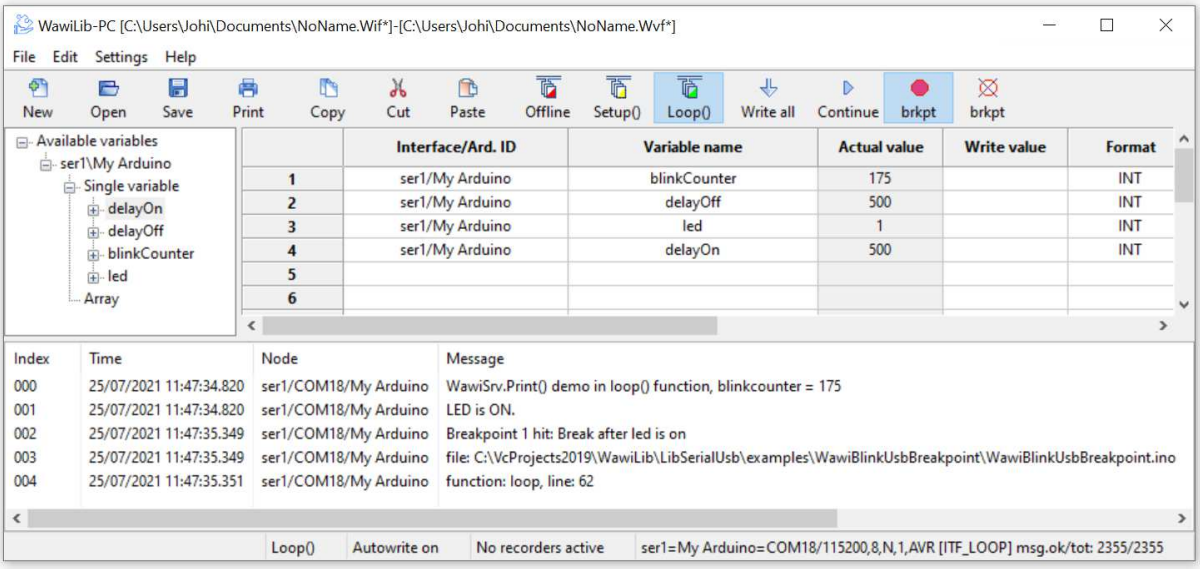


Fig 9.3. WawiBlinkUsbBreakpoint hit a breakpoint again.

10. Further reading

This demo demonstrates the concept of WawiLib using the USB programming port of your Arduino board. WawiLib has more extended functions that are 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.

Arrays of variables are also supported by WawiLib. Recording 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. WawiLib supports links via USB, Wi-Fi, cabled Ethernet, RS232C, 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.