Application note: potentiometer project

Contents

1 INTRODUCTION

1.1 Objective of this document

The objective of this document is to describe how to use WawiLib in a simple practical example.

In many applications you need some kind of rotary knob to modify the setpoint for a parameter. It can be the temperature for a temperature controller, it can be the setpoint for the speed of an engine or some other sort of setpoint.

In industry (certainly some years ago) one used a knop that drove some kind of variable resistor. When you turn the knob, the resistor value changes. So by measuring the resistance of this resistor, you know the position of the knob.

A potentiometer has 3 connections in most cases: 2 fixed to an internal resistor and 1 connection connected to a moving contact that can move over this resistor. The resistance value between the moving contact and the fixed connection points changes along with the user moving the knob or dial.

If you want to know the position of the potentiometer in an Arduino board, you connect one fixed end to 5V, the other to 0V and the moving contact clamp an Arduino analog input. As you turn the knob or dial the output value of the potentiometer goes from 0V to 5V. So you know the state of the knob or dial.

Not quite: it can be so that the resistance of the wires in the 0V and the 5V connection are not 0 or not the same depending on: the type of connection, the construction of the potentiometer and the length of the wires. So the 0 position of the potentiometer is not exactly 0V and the max position deviates a bit from 5V as well.

The objective of this demo is to show how to scale to exactly 0-100% whatever the deviations of the electrical configuration are. The technique used can be used in many other application as well. We will use WawiLib to look at the raw values, modify the scaling parameters and display the value of all the parameters involved.

1.2 Required user experience

You should be familiar with the tutorial “Getting started with WawiLib using the USB programming port”. There are no specific additional requirements.

1.3 Required hardware

- Arduino board (in this case A Mega2560 or compatible, Uno is also OK).
- PC:
  - Arduino IDE software.
  - WawiLib-PC software.
  - WawiSerialUsb Library installed in on the PC in the Arduino IDE.
- A potentiometer (20K in this example).
- Some wires to connect the potentiometer to the Arduino board.
- USB cable.

2 Hardware connections

2.1 Equivalent drawing of a potentiometer connected via wires to an Arduino.

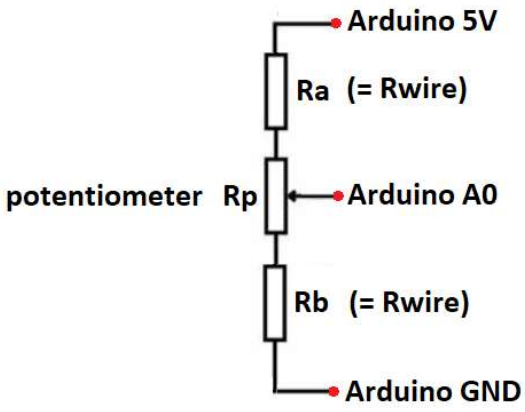

Fig 1. Equivalent drawing of potentiometer + connecting wires.

In Fig 1. Rp is the potentiometer, Ra represents the resistance of the wire between 5V of the Arduino and the first fixed clamp of the potentiometer, Rb does the same for the wire from the second fixed clamp of the potentiometer to the Arduino GND. The variable clamp of the potentiometer is connected to A0, an analog input of the Arduino board.

2.2 Calculation of the voltages V+ and V- (for the specialists)

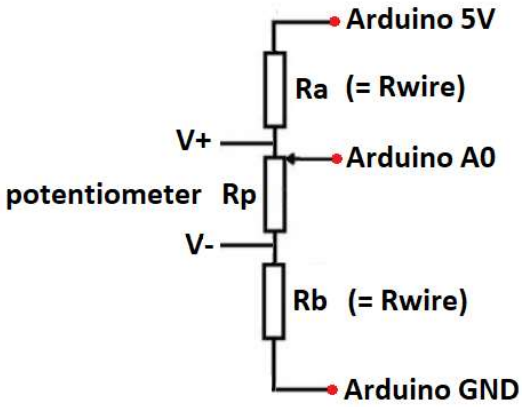

Fig 2. Voltage drop over Ra makes V+ < 5V, potentiometer in its upmost position.

$A0 = V+$=5V * (Rp+Rb)/(Ra+Rp+Rb) [voltage divider equation]

The equation above implies that as long as Ra is not 0, V+ will not be 5V. So the voltage at input A0 of the Arduino with the position of the potentiometer in its upmost position will not be 5V but a bit below 5V (for example 4.95V) hence we need to correct the input at A0 to make sure 4.95V is converted to the value 100% for the setpoint of our motor or other application.

Note: if the wires are very short you will see no practical influence of the wire resistance but in a larger application, they can play a significant role.

2.3 Hardware connections

- ✓ Connect the GND pin of the interface converter to one fixed side of your potentiometer.
- ✓ Connect the 5V pin of the Arduino to the other side of your potentiometer.
- ✓ Connect the Arduino Mega to the PC using the USB cable.

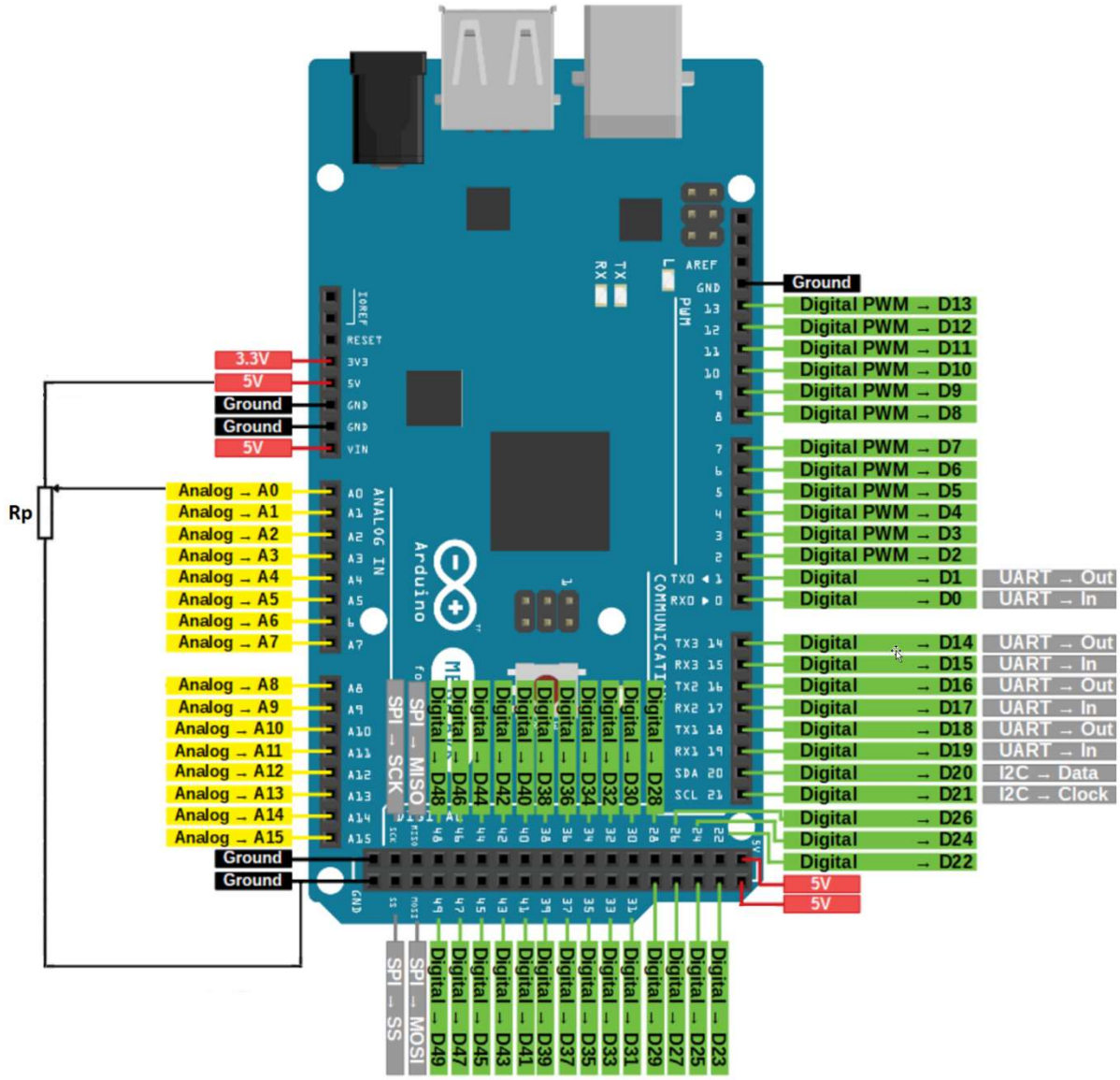

Fig3. Arduino - potentiometer connections.

3 Arduino software.

3.1 Open the "WawiEmpty" example

✓ Open an empty framework example with WawiLib connection for the USB port:

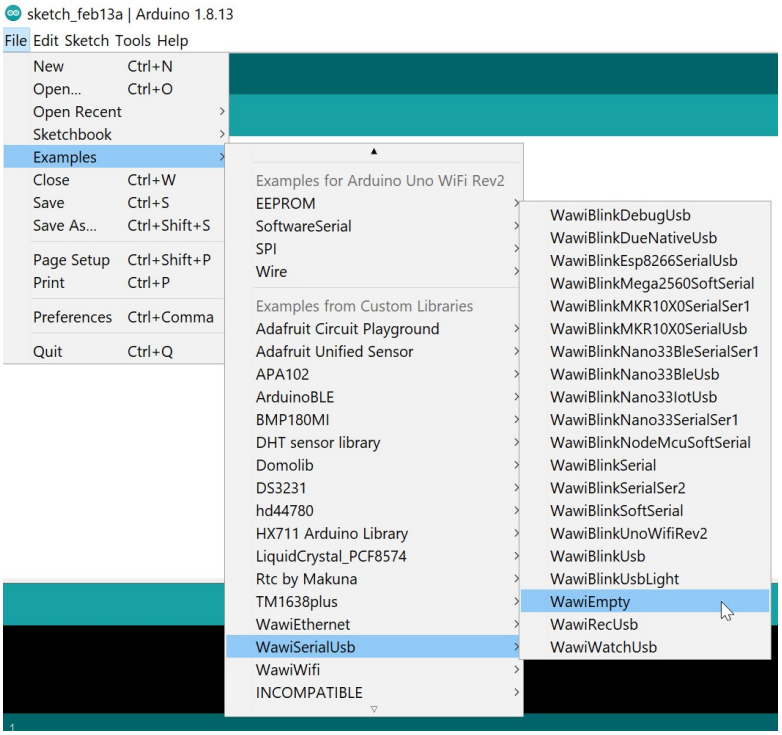

Fig 4. How to open the "WawiEmpty" example.

WawiEmpty is installed on your PC when you install the WawiSerialUsb library. To install the WawiSerialUsb library, see the "Getting started with WawiLib using the Arduino USB programming port." chapter 2: "INSTALL WAWILIB SOFTWARE".

```
/*
* Project Name: WawiEmpty
* File: WawiEmpty.ino
*
* Description: empty demo file to get you sarted with WawiSerialUsb libary.
* Use the USB programming port to make connection with the Arduino board.
* Variables can be checked & modified with the WawiLib-PC software.
*
* Remove or modify the lines with //? to get you started.
*
* Author: John Gijs.
* Created Jan 2021
* More info: www.sylvestersolutions.com
* Technical support: support@sylvestersolutions.com
* Additional info: info@sylvestersolutions.com
*/

#include <WawiSerialUsb.h>

WawiSerialUsb WawiSrv;

// declare your variables:
//? int a=500;
//? char s[20]="Hello world.";
```

```
void wawiVarDef()
{
    // make variables visible to WawiLib:
    //? WawiSrv.wawiVar(a);
    //? WawiSrv.wawiVarArray(s);
}

void setup()
{
    // intialize your serial port:
    Serial.begin(115200);
    // initialize the WawiLib library:
    WawiSrv.begin(wawiVarDef, Serial, "Empty Demo Arduino");
}

void loop()
{
    // print variables to the WawiLib-PC console output window:
    //? WawiSrv.println(a);
    //? WawiSrv.println(s);

    // wait a bit to make sure you can read the output in the WawiLib-PC output window
    //? WawiSrv.delay(1000);

    // do the internal WawiLib housekeeping:
    WawiSrv.loop();
}
```

Fig 5. "WawiEmpty" source code.

- ✓ Compile and load the example in the Arduino board.

3.2 Modify the example to read and scale analog input value.

- ✓ Add the yellow code to read the analog input and scale it (estimate scale) to 0-100%:

```
#include <WawiSerialUsb.h>

WawiSerialUsb WawiSrv;

// declare your variables:
int potMin = 0;
int potMax = 1000;
int potAct;
float potPct;

void wawiVarDef()
{
    // make variables visible to WawiLib:
    WawiSrv.wawiVar(potMin);
    WawiSrv.wawiVar(potMax);
    WawiSrv.wawiVar(potAct);
    WawiSrv.wawiVar(potPct);
}

void setup()
{
    // intialize your serial port:
    Serial.begin(115200);
    // initialize the WawiLib library:
    WawiSrv.begin(wawiVarDef, Serial, "Empty Demo Arduino");
    // define A0 as analog input:
    pinMode(A0, INPUT);
}
```

```
void loop()
{
    // read analog value
    potAct = analogRead(A0);
    // scale analog value
    potPct = (float)100.0 * (potAct - potMin) / (potMax - potMin);

    // do the internal WawiLib housekeeping, so we can monitor and modify potMin,
    // potMax, potIst and potPct with WawiLib-PC.
    WawiSrv.loop();
}
```

Fig 6. "WawiEmpty" source code modified to read and scale analog value.

✓ Compile and load the example in the Arduino board.

4 Test the application and tune the parameters *potMin* and *potMax*.

4.1 Connect your board with WawiLib-PC.

If you have followed the demo "getting started with WawiLib programming port", then you should be familiar with the following steps. If not, follow the demo first.

- ✓ Start WawiLib-PC.
- ✓ Go to "Settings\Communication interfaces".
- ✓ Select the settings for your board:

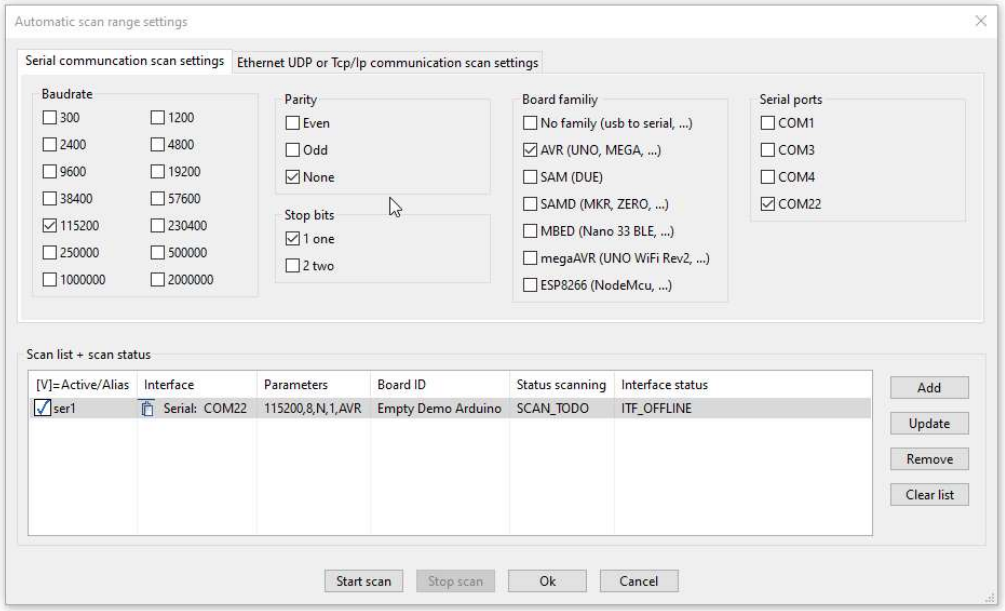

Fig 7. Connection parameters settings (serial port number can be different).

- ✓ Press "Add".
- ✓ Press "Start scan".
- ✓ The following message should appear to indicate that you are properly connected:

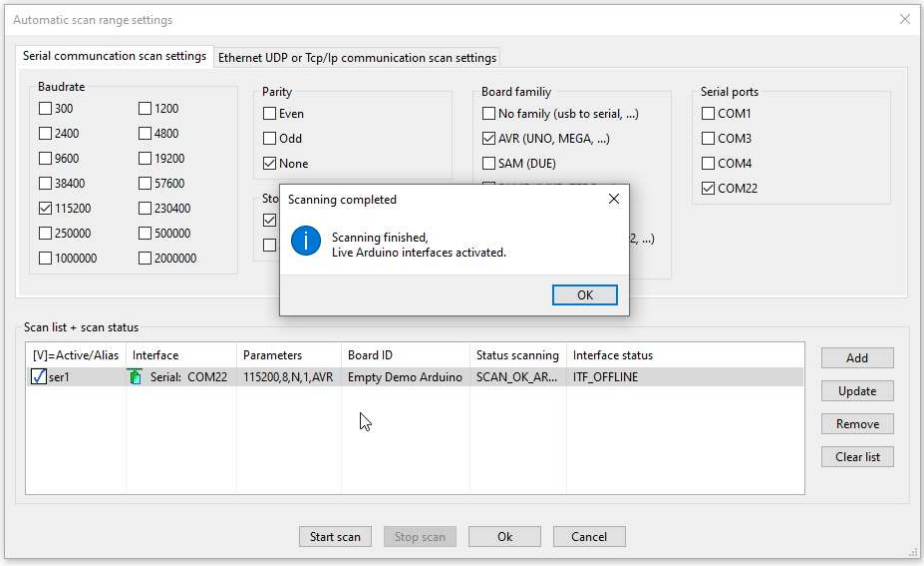

Fig 8. Connection parameters settings (scanning finished).

- ✓ Press 2 x "ok" to close the message and the scan range dialog box.

4.2 Display the variables of interest.

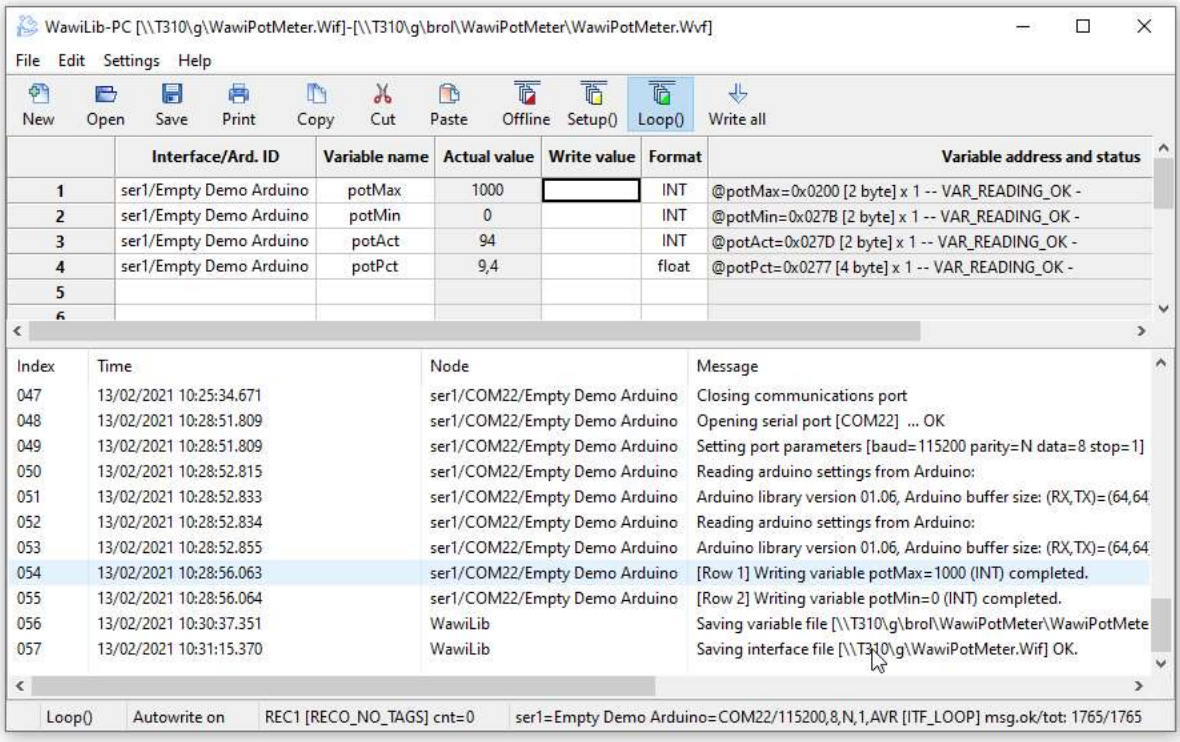

Fig 9. WawiLib-PC with variable names of interest.

- ✓ Fill in the names of the variables in the column "Variable name".
- ✓ Press "Setup()".
- ➢ The actual values of *potMax*, *potMin, potAct* and *potPct* should appear.
- ✓ Turn on the dial of your potentiometer.
- ➢ The actual values of *potIst* and *potPct* should change.

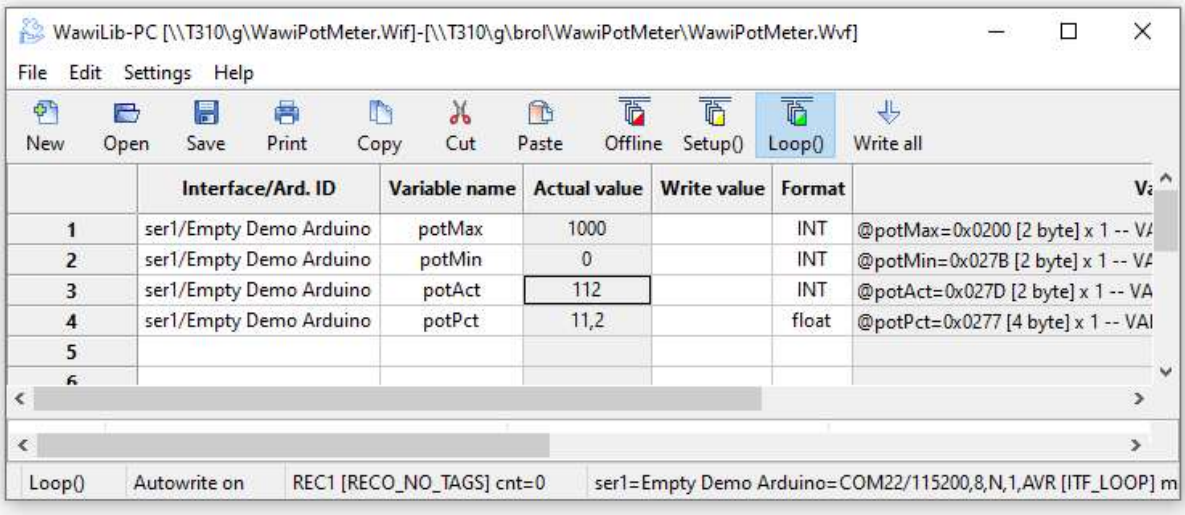

Fig 10. Variable names of interest after change of potentiometer position.

4.3 Tune the low scale value.

- ✓ Turn on the dial of your potentiometer until the value of *potAct* minimum, but no further and then stop turning.
- ➢ You have now reached the minimum position of your potentiometer.

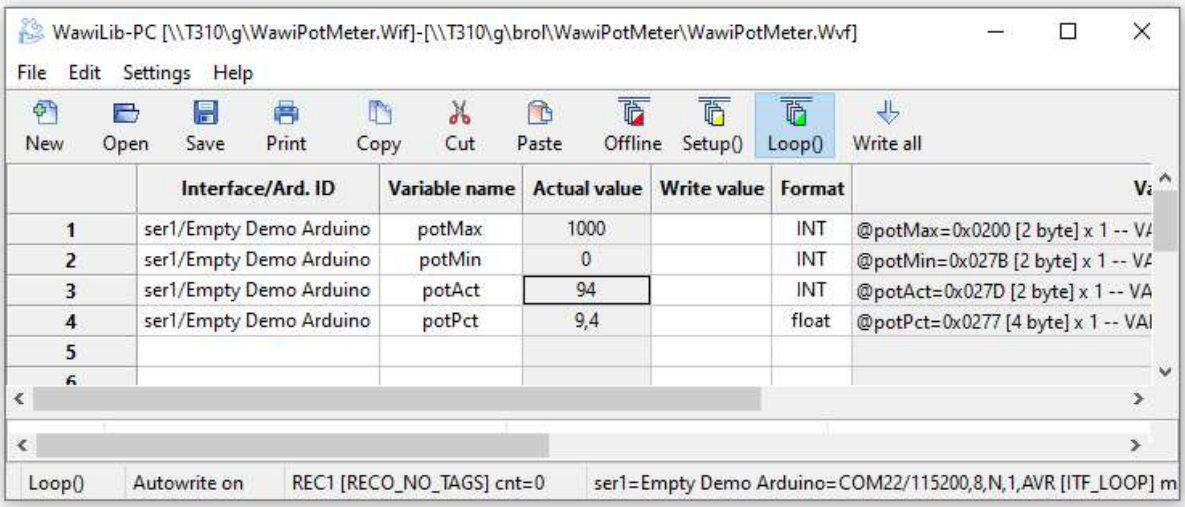

Fig 11. Variable names of interest after at minimum potentiometer position.

- ✓ Copy the value *potAct* (in my case 94) into the field "Write value" next to *potMin*.
- ✓ Press "Write all".
- ➢ You have now changed the minimum scaling value to 94.
- ➢ In the same time as *potPct* should change to something close to 0.

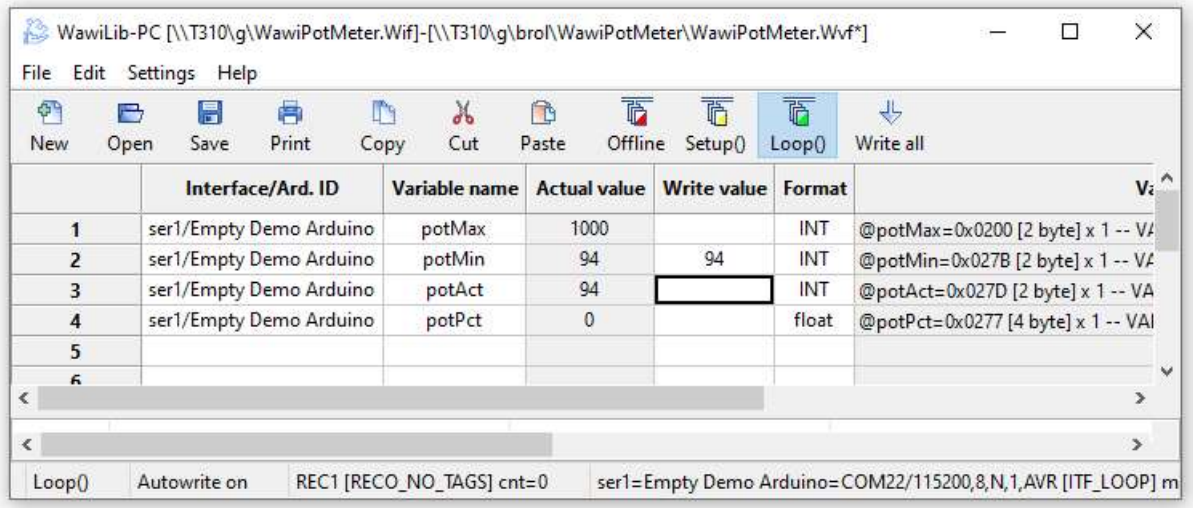

Fig 12. Variable names of interest after at minimum potentiometer position, scale corrected.

4.4 Tune the high scale value.

- ✓ Turn on the dial of your potentiometer until the value of *potAct* maximum, but no further and then stop turning.
- ➢ You have now reached the maximum position of your potentiometer.

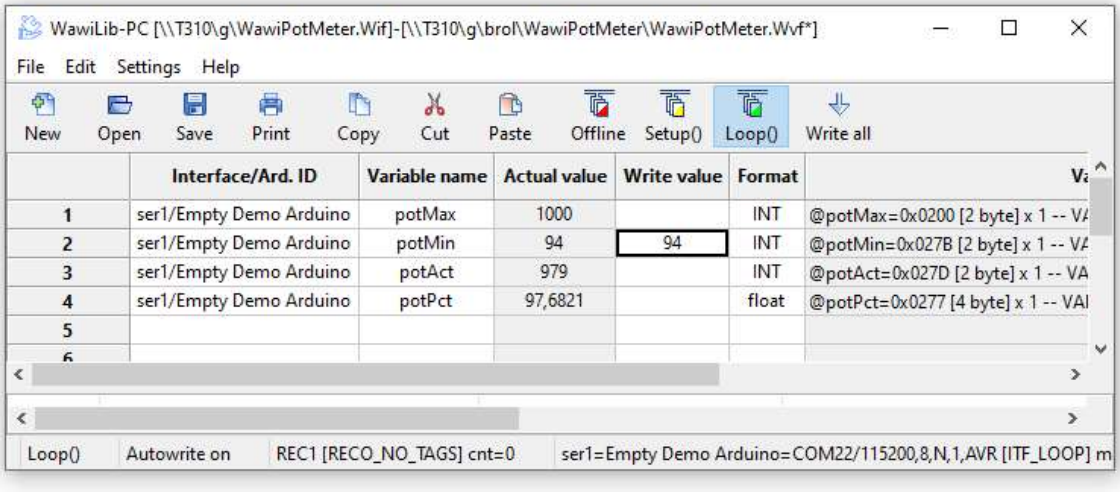

Fig 13. Variable names of interest after at maximum potentiometer position.

- ✓ Copy the value *potAct* (in my case 979) into the field "Write value" next to *potMax*.
- ✓ Press "Write all".
- ➢ You have now changed the maximum scaling value to 979.
- ➢ In the same time as *potPct* should change to something close to 100.

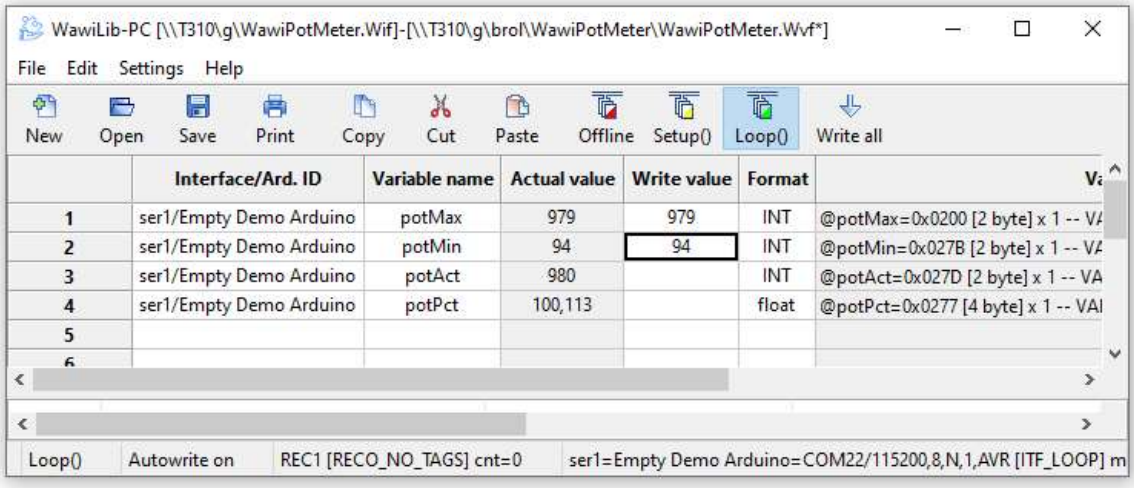

Fig 14. Variable names of interest after at maximum potentiometer position, scale corrected.

4.5 Update the sketch with the values for *potMin* and for *potMax*.

✓ Add the yellow code to read the analog input and scale it (true scale) to 0-100%:

```
#include <WawiSerialUsb.h>

WawiSerialUsb WawiSrv;

// declare your variables:
int potMin = 94;
int potMax = 979;
int potAct;
float potPct;

void wawiVarDef()
{
    // make variables visible to WawiLib:
    WawiSrv.wawiVar(potMin);
    WawiSrv.wawiVar(potMax);
    WawiSrv.wawiVar(potAct);
    WawiSrv.wawiVar(potPct);
}

void setup()
{
    // intialize your serial port:
    Serial.begin(115200);
    // initialize the WawiLib library:
    WawiSrv.begin(wawiVarDef, Serial, "Empty Demo Arduino");
    // define A0 as analog input:
    pinMode(A0, INPUT);
}

void loop()
{
    // read analog value
    potAct = analogRead(A0);
    // scale analog value
    potPct = (float)100.0 * (potAct - potMin) / (potMax - potMin);

    // do the internal WawiLib housekeeping, so we can monotor and modify potMin,
    // potMax, potIst and potPct with WawiLib-PC.
    WawiSrv.loop();
}
```

Fig 15. "WawiEmpty" source code after updating of scaling.

✓ Go offline in WawiLib-PC.
✓ Compile and download the new code to your board.
✓ Go online with WawiLib-PC and check the result.

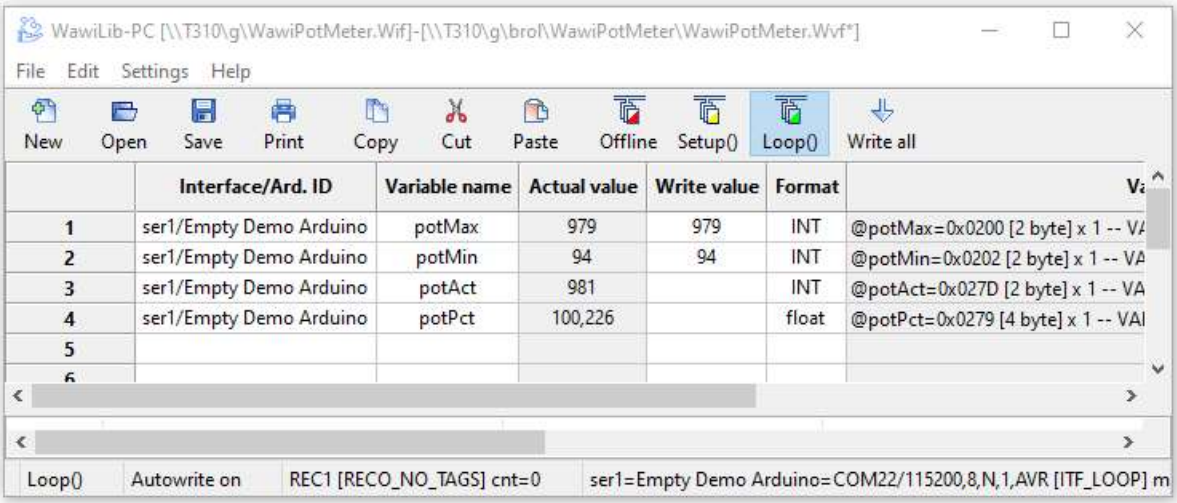

Fig 16. Check of updated settings for *potMax* and *potMin*.

5 FURTHER READING

This demo demonstrates how to tune and test a simple potentiometer application using WawiLib. We first opened a framework sketch, then we added some lines of code to define the scaling variables, we exposed the variables to WawiLib using WawiVar(), then we added code to read and to scale the variables themselves from the analog IO and the last step was the actual scaling.

As this is a very simple task, other ways to obtain the same result exist, but a your application becomes more and more complex, the WawiLib way is simply a more structured, user friendly and powerful way to do things.

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