Detailed guide calibrating and using your MightyWatt R3

Guide revision: E (2017-04-07)
Board revision: 3.1
Minimum firmware version: 3.1.0 (main), 3.1.0 (calibration)
Arduino IDE tested: 1.8.1 (Arduino.cc)
Windows control program version: 3.1.0.0
Gnuplot version tested: 5.0.5 (gnuplot.info)

Safety first

Do not connect device under test to the terminals of the load when the load is not powered up. Before unplugging MightyWatt R3 from computer, disconnect any attached device under test from its input terminals. Before detaching MightyWatt R3 from its control board (Arduino), unplug the control board from computer.

The PWR+ voltage may appear on the heatsink and its screws. It typically does not but the possibility is there.

Calibration

This chapter will guide you through the calibration of your MightyWatt R3. Follow this guide **if your unit was *not* factory-calibrated** or you want to recalibrate it manually. MightyWatt R3 uses a precision ADC and DAC but there are still offsets and component tolerances that should be removed by a calibration. Because MightyWatt R3 has a dedicated ADC and DAC, the calibration is valid for the unit regardless of the control board (Arduino) used. If you strive for the best precision, recalibrate MightyWatt R3 every year. The calibration sketch is both for Arduino Uno and Zero (M0/M0 Pro). However, you must select the board also in the sketch in `Configuration.h`.

You will need one multimeter (voltmeter/ammeter) and a power supply with adjustable current and voltage. The best results are achieved when both the multimeter and power supply can measure/deliver the voltage and current to match the range of MightyWatt R3. If you have a multimeter or power supply with a limited range, the calibration is still possible.

Warning: During the calibration, neither the power limit nor the temperature of the transistor is monitored. Beware of the combination of high voltage and high current!

1) Connect your power supply to MightyWatt R3. Connect your ammeter in series with the load, preferably between the PWR+ terminal and positive terminal of your power supply. Set low voltage (~ 2 V) and maximum current on your power supply.
2) Open the calibration sketch in Arduino IDE.
3) Select the correct board in Arduino IDE in **Tools** → **Board**. Use *native USB* on Arduino Zero (M0/M0 Pro) boards. **Warning:** Arduino IDE discriminates between Arduino Zero and Arduino M0 Pro. Select the correct version, otherwise MightyWatt will not function properly.

4) Select the correct COM port in Arduino IDE in **Tools** → **Port**.
5) Go to `Configuration.h`. Uncomment the line (remove "//") that defines your board (*`#define UNO`* or *`#define ZERO`*), comment out the other line (put "//" at the start of the line). Uncomment the line that defines your ADC type – either 12-bit (*`#define ADC_TYPE_ADS1015`*) or 16-bit (*`#define ADC_TYPE_ADS1115`*), comment out the other line.
6) Go back to `MightyWattR3_Calibration` and find the line 100 (*`/* Calibration */`*) where the calibration area starts. Line numbers are shown at the left bottom corner of Arduino IDE. You can change the number of calibration points for every range by modifying the *`#define CALIBRATION_POINTS`* line.
7) Go to the line 120 where the current calibration (Part 1) starts. If your power supply or ammeter is not capable of delivering/measuring the maximum current your MightyWatt R3 can handle, limit the *`DAC_PERC_I_HI_RANGE_MAX`* value, if it can, leave the 99% there. This value determines the percentage of maximum possible current on high-current range that will be used in the calibration. For example, if your MightyWatt R3 is rated 24A and you only have 10A power supply, the calibration should end at 10 / (24 × 1.06) = 39%. Assume 6% overrange on the MightyWatt R3 current – multiply the rated maximum by 1.06. Write only the integer part of the percent, without the the percentage sign (e.g. *`#define DAC_PERC_I_HI_RANGE_MAX 39`*). Do not write more than 99%.
8) Upload the sketch into your board using the arrow in the top left corner of Arduino IDE.
9) Open your Excel calibration aid spreadsheet and clear the already-written sample values in Part 1.
10) Start the Serial Monitor console on the Arduino IDE by clicking on the magnyfying glass icon in the top right corner of Arduino IDE. If you are using Arduino Uno, select baud rate 9600 in the console window that will open (if not already selected) and restart the Serial Monitor. Arduino Zero (M0/M0 Pro) should work regardless of the baud rate set. When you open the console, it will prompt you to select the current range (low or high). If nothing is displayed, close the console, re-upload the sketch and open the console again. You will need to calibrate both current ranges. Select the one you want to start with.
11) Then, the console should show two kinds of values: the DAC value and the ADC value plus you should have a reading on your ammeter. It may take up to 3 seconds before the ADC value is displayed. Write the DAC, ADC and ammeter value into the spreadsheet calibration file and do not close the serial monitor.
12) Send any character (except "q") to the Arduino via the serial monitor. This will advance one iteration in the cycle, raising the current. Repeat until all points are measured and then send any character (except "q") once again until the console prompts you to continue with the calibration and ask for the current range you want to continue with. If you are satisfied with the results of the calibration, select the other range and repeat the same procedure. After you have measured all calibration points, type send "q" to finish.

13) After you put all the calibration values into the spreadsheet, Part 1, it will show the calibration values, which you will later put into `Configuration.h` of the main sketch.
14) Close the console and go to line 103. Comment out *`#define CALIBRATE_CURRENT`* and uncomment *`#define CALIBRATE_VOLTAGE`*.
15) Set your power supply to low current (~ 0.2 A) and 32 V (or maximum voltage if your power supply cannot go to 32 V). **For safety, lower the current first, then increase the voltage**. Connect MightyWatt R3 directly to the power supply, connect sense cables from SENS-/SENS+ to the power supply terminals too and connect your voltmeter to the same point where the sense cables end. This way the measured voltage by the MightyWatt R3 will not be affected by the cable resistance. If you are unable to use the 4-wire connection (PWR and SENS terminals), use only 2-wire connection (PWR terminals only) and comment out line 263.
16) Voltage calibration (Part 2, line 236) follows the same principle as Part 1. Adjust the maximum calibration voltage (percentage of rated range) in *`DAC_PERC_V_HI_RANGE_MAX`* in the same fashion as you did with the current. For example, if your MightyWatt R3 is rated 30V and you only have 20V power supply, the calibration should end at 20 / (30 × 1.06) = 63%. Assume 6% overrange on the MightyWatt R3 voltage – multiply the rated maximum by 1.06. Write only the integer part of the percent, without the the percentage sign (e.g. *`#define DAC_PERC_V_HI_RANGE_MAX 39`*). Upload the changed sketch into Arduino and start the Serial Monitor. Again, two sets of values are needed – one for the low voltage and one for the high voltage range. Do not write more than 99%.
17) Put the measured values into the Excel calibration aid file, Part 2, replacing any values that may have already been there. It will show your calibration values, which you will put into `Configuration.h` of the main sketch.
18) Open the main sketch and locate `Configuration.h` (not the one in the calibration sketch). Copy the calculated calibration values and the date of calibration from Excel into the */* *`Calibration`* *** part of the `Configuration.h` file. The cells in Excel are conveniently arranged so that you can easily copy&paste them into the `Configuration.h` file.
19) Continue with the next part: **Main sketch**.

Main sketch

The main sketch was tested for Arduino Uno and Zero (M0/M0 Pro). However, you must select the board also in the sketch.

1) Open the sketch in the Arduino IDE.
2) Select the correct board in the Arduino IDE in **Tools** → **Board**. Use *native USB* on Arduino Zero (M0/M0 Pro) boards. **Warning:** Arduino IDE discriminates between Arduino Zero and Arduino M0 Pro. Select the correct version, otherwise MightyWatt will not function properly.
3) Select the correct COM port in Arduino IDE in **Tools** → **Port**.
4) Go to `Configuration.h`. Uncomment the line (remove "//") that defines your board (*`#define UNO`* or *`#define ZERO`*), comment out the other line (put "//" at the start

of the line). Uncomment the line that defines your ADC type – either 12-bit (*#define ADC_TYPE_ADS1015*) or 16-bit (*#define ADC_TYPE_ADS1115*), comment out the other line.

5) Upload the sketch into your board using the arrow in the top left corner of Arduino IDE.

Windows control program

MightyWatt R3 is controlled over virtual COM port. This is used in the Windows software.

1. Connecting to MightyWatt R3

Connect the USB cable from the load to the computer, then start the program and in the **Connection** menu select the Arduino board you are using and then the correct COM port.

2. Manual control and measured values

Manual control immediately sends the desired setting to MightyWatt R3. **Stop** button will set the load to zero current. Continuously measured values are displayed in **Values** box.

3. Software-controlled modes

MightyWatt R3 has internal analog circuitry to keep constant current (CC) or constant voltage (CV). Other modes are maintained in a software feedback loop but physically are either CC or CV.

3.1. Constant power and constant resistance

Can be selected to be internally either constant current or constant voltage. If you don't know which mode to select, start with CC mode.

3.2. Software-controlled voltage

This mode is handy when the phase of the device under test (DUT) is opposite of the normally-expected phase for voltage. Typically, a device will decrease its voltage when it is loaded (more current is drawn from it). In specific situations, the voltage can increase with the current. In such cases, the analog circuitry that keeps CV will not be able to function properly. However, it is possible to achieve CV in a software-feedback loop. That is what this mode does. Internally, it is set to CC and tries to maintain constant voltage by adjusting the current.

3.3. Maximum power point tracker (MPPT)

Some devices, namely photovoltaic cells and fuel cells, have an operating point where their output power is at a maximum. This mode can adaptively track the peak power. Internally, it is set to CV and tries to manipulate the voltage in such a way that the power is the highest possible. When you are setting the MPPT, you can enter initial voltage. If you leave zero in the box, MightyWatt R3 will set 90% of the open-circuit voltage as the initial guess. This function is experimental and may not work correctly with all types of DUTs.

4. Programmable watchdog

Watchdog can stop the load in case current, voltage, power or resistance rises above or drops below the set value.

5. Voltage sensing

Voltage can be **sensed** either locally at the power (PWR) terminals or remotely using another pair of cables – from the sense (SENS) terminals. Remote voltage sense (4-wire, Kelvin) effectively removes the voltage drop introduced by the cable resistance from measurements. This is the preferred mode of measurement when the voltage is important to be measured precisely.

6. Programmatic control

More complex behaviour can be programmed using the **Program** box. You can select any combination of controllable variables (current, voltage, power, resistance, software-controlled voltage or maximum power point tracker) and assign it any **value** for any **duration**. Moreover, a condition similar to the watchdog function can be set to each program item. The **skip if** condition will terminate the current program item and jump to the next one. If **Watchdog** is enabled, it is active in program mode as well and will terminate the program if it trips.

It is possible to create several program items which will be executed consecutively after you click the **Start** button. You can use either **constant** value or linear **ramp** from an initial value to a final value. The initial value can be copied from **previous** program item. In this case, the last measured value of the preceding program item will be used as the starting value for the next item. You can repeat the program procedure by enabling **loop** and setting it to either a specific number of loops or run the program in an infinite loop. It is possible to save the current program procedure to a XML file. To do so, click **Program items → Save**.

Loading a program procedure can be done in two ways: Either adding the items from a XML file or overwriting existing items with those from a XML file. Adding does not change any other settings, overwriting will also overwrite the loop setting, logging period setting and if the load is connected at that moment, then also 2-wire/4-wire mode, measurement filter, LED brightness, LED rules and fan rules. To add items, select **Program items → Add**. To overwrite items and settings, select **Program items → Replace.**

7. Data logging

Data can be saved into a simple tab-delimited text file that can be easily imported into Excel. A data file must be created before logging is possible. Use the **Logging** menu to create a **New File**. The sampling period can be adjusted in **Logging → Settings**. For the maximum logging speed, set the value to zero. The true maximum logging frequency is approximately 220 Hz when using Arduino Uno and 237 Hz when using Arduino Zero (M0/M0 Pro). Logging is activated and deactivated separately for manual control (**Log Manual**) and for program (**Log Program**). Program logging will start after clicking the **Start** button and will terminate when the program finishes. Manual logging will be active when a user program is not running. The time value in logged data is referenced to the time when the file was created. Data is written into the file continuously so you won't lose them in case the program crashes.

8. Settings

You can set the LED behaviour, fan behaviour and the measurement filter in **Settings**. Choose the LED brightness in **Settings → LED → Brightness** and the cases when it will be lit in

Settings → LED → Rules. The rules for when the LED is on are combined (ORed). Similarly, the rules of fan control can be selected in **Settings → Fan**. For safety reasons, the fan can never be turned fully off. It can either be always on or off in situations where the power dissipation is low and the temperature also. **Auto quiet** has more relaxed rules than **Auto cool**. For the best accuracy of measurement, keep the system as cool as possible by letting the fan to run continuously. A 42-period (~200 ms) triangular weighted moving average filter is implemented in the firmware to the smooth noise from measurements. This filter can be enabled or disabled in **Settings → Measurement filter**. Use unfiltered setting only for fast measurements.

Because the load can impose millisecond-long glitches during range switching, the autoranging feature can be disabled for current and voltage in their control modes in **Settings → Autoranging**. That means the current autoranging will be disabled when the load is in CC mode. Voltage autoranging will not be affected in CC mode. In CV mode, it will be the opposite. Accuracy for low current or voltage may be lowered when autoranging is disabled.

You can save and load settings to/from a XML file. This is the same file used for saving Program items and the save (**Settings → Save**) does the same as in Program items. Load (**Settings → Load**) will only load settings, without affecting Program items.

9. Tools

The tools menu contains **Integrators and statistics** toolbox that displays statistical information for the measured values and allows the integration of charge and dissipated energy (heat). In the toolbox, press **Start (Stop)** to initiate (halt) the data gathering. Press **Log snapshot** to log the displayed values to the log file. The button is enabled only when a log file is opened. You can enter an additional header for the data in the **User note** text box. Check **Offset log** to offset the logged data to the right so they are in different columns than the regular log data. This may help with the subsequent processing in your favourite spreadsheet application.

A plot of the recent voltage, current, power and resistance is possible if you have installed gnuplot – a free software for creating graphs (www.gnuplot.info) – and MightyWatt connected. Select **Tools → Plot (gnuplot)** and the period from which the data will be displayed. If the control software does not find gnuplot installed in the default location, it will ask you to locate its executable manually and then will create a text file (gnuplotpath.txt) with this path so you won't have to locate it again. Data is gathered automatically but can be cleared manually using **Tools → Plot (gnuplot) → Clear data**. Graphs are updated once per second. Because gnuplot is started as an external application by the Windows control program, it must be closed again through the Windows control program by selecting **Tools → Plot (gnuplot) → Close plot**.

10. Compensating maximum power dissipation for series resistance

When you run the load in 4-wire mode, the cable resistance is not reflected in the maximum power dissipation. The load will dissipate less power because some of the power is dissipated in the cables. You can also deliberately add a power resistor in series with the load to lower the heat burden at the load. The load will, however, limit the power to the calculated value – not the actual power it is dissipating. For these cases, it is possible to compensate for the extra series resistance. Select **Settings → Series resistance**. A pop-up window will appear that lets you

specify the series resistance. The load will now assume that you are dissipating some of the power ($R \cdot I^2$) externally and raise the total power rating in the 4-wire mode. You can also enter the power rating of your external resistance and select to stop the load automatically when the power is exceeded for more than one second. This automatic stop works in both 2-wire and 4-wire mode. Use series resistance feature with caution as wrong values may cause MightyWatt R3 to overheat.

11. Information on device ranges

You can see some of the hardware limits of MightyWatt R3 by clicking on **Help → Device Info**. The displayed values are calculated from the calibration values and generally taken from `Configuration.h`.

12. Thermal protection

Because power dissipation can be high in MightyWatt R3, it constantly monitors the temperature of the main MOSFET. The temperature displayed is lower than the true junction temperature so a healthy safety margin is preferable. That is why the maximum temperature is set at 110 °C, which corresponds to about 145 °C junction temperature at 25 °C ambient temperature. MightyWatt R3 will automatically stop when overheat is detected.

Your feedback is appreciated

Tell me how you like this guide and how you like MightyWatt R3. Send me a note at jpolonsky@gmail.com!