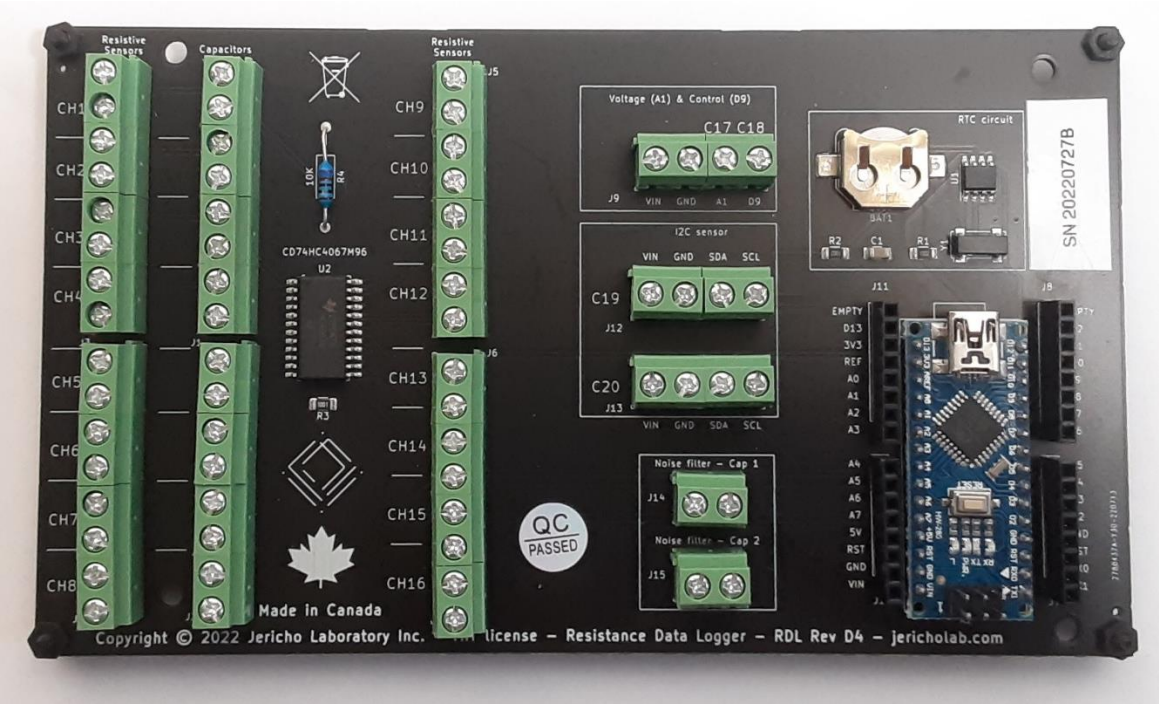
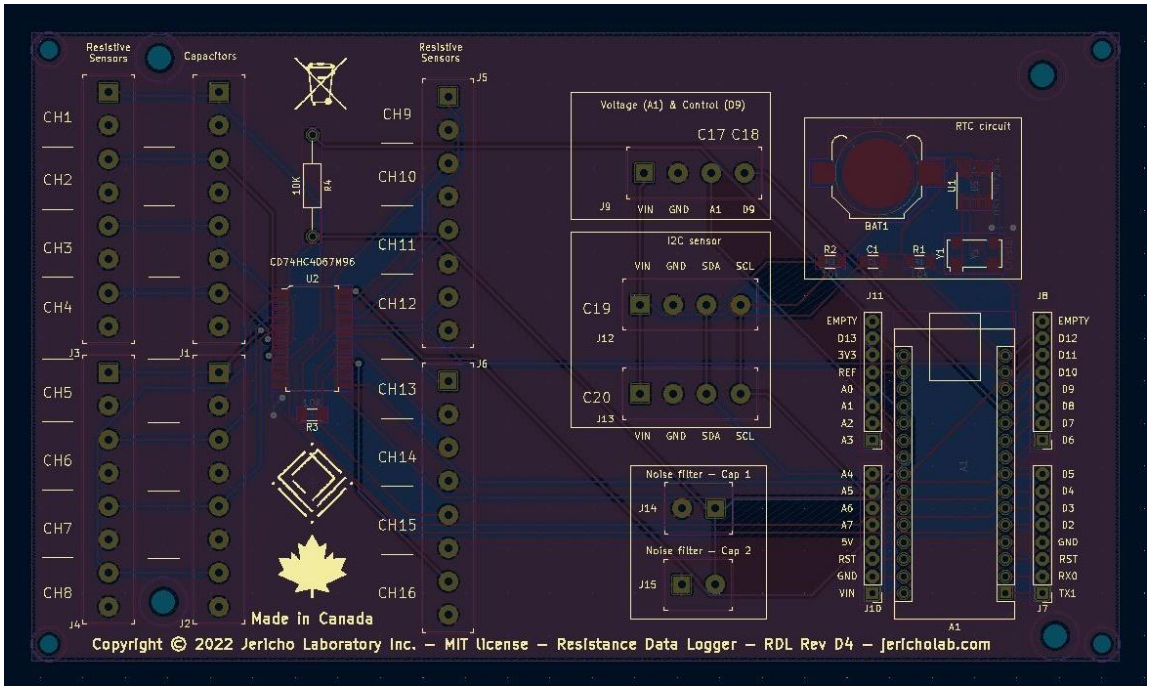
USER GUIDE – RESISTANCE DATA LOGGER (RDL) Rev D6 - WIP

JERICHO LABORATORY INC. - JERICHOLAB.COM

- -

Explore. Measure. Understand.





This page is intentionally left blank

Table of Contents

WARNING: This product contains small parts. Not for children under 5 years old.
WARNING: This product is not intended for critical use, internal use, medical use or food safety.
WARNING: One item of the Starter kit contains cadmium (photoresistor), which is a toxic substance. Cadmium is a forbidden substance in the European Union.
DISCLAIMER: Jericho Laboratory Inc. will not be held responsible for your safety or any damage other than to the device itself. We advise the user to apply great caution while using this product.

PRODUCT OVERVIEW

The Resistance Data Logger (RDL) is a low-cost, ready-to-use, open-source data logger for scientists and engineers. Mostly centered around high-quality temperature measurements, the 20-channel device can also measure humidity, luminosity, resistance, voltage, as well as do some basic control. Text-based, it focuses on data, providing only elementary graphing capabilities.

Customizable, transparent and well-documented, this product is for those who want to own their data, but also own their tool.

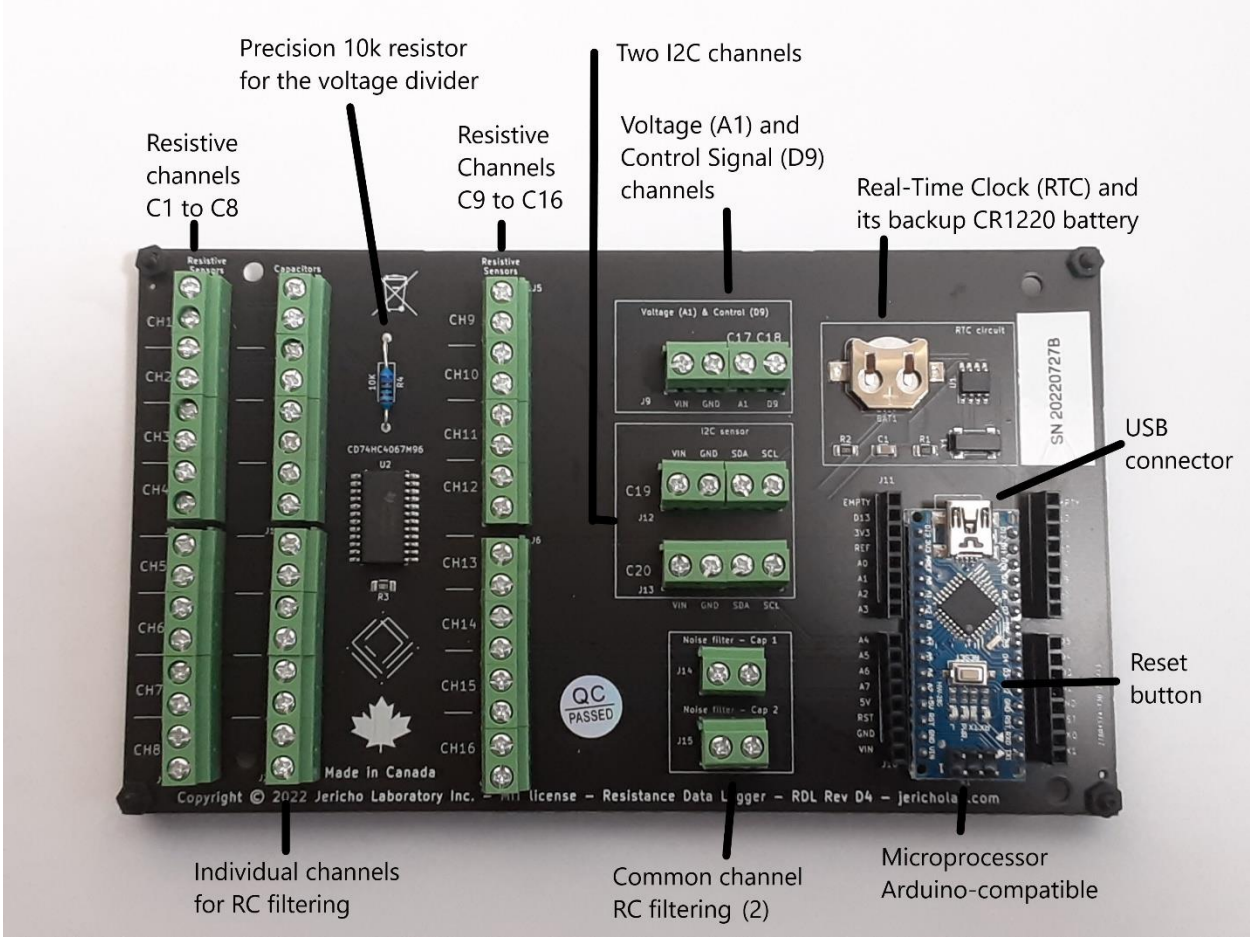


Figure 1 RDL rev D6 components layout

The RDL can measure a variety of variables over 20 channels, simultaneously. Exact availability of each measurement can be examined in Table 1. Humidity readings can be done natively with the use of wet bulbs. Humidity can also be read with a digital sensor (I2C) from the AHX family. Other I2C sensors (e.g., accelerometer) can be used by adding the required code. Digital control signals (0-5V, binary or PWM) can also be sent. Resistance measurements allow you to measure a range of phenomena from soil moisture to blood clot formation. Using our low-cost black globe and a wet bulb temperature, the RDL can monitor the comfort with the Wet Bulb Globe Temperature (WBGT) index.

The measured data is sent continuously to your computer or smartphone, for purposes of display and data storage. The device is equipped with an RTC clock and a 3V battery for accurate and resilient timestamp of the data. The device is compatible with Arduino Nano 3.0 ©.

IMPORTANT: The RDL is unable to log any data if it is not properly communicating with a PC or smartphone. Always make sure that the recording is active when using the product.

Table 1 RDL Channels Overview for revision D6

Zone identification	Connector type	Type of channel	Number of channels	Measurement ability	Control ability	Comments
J1, J2, J3, J4	Screw terminal	Voltage divider subcircuit (A0)	16	Temperature (resistor) Resistance Luminosity Soil humidity	N/A	Optional condensator can be connected in zones J5, J6
J9	Screw terminal	Arduino Analog pin (A1)	1	Voltage (0-5V)	Digital signal	For differential voltage measurements, connect the second pin to a ground.
J9	Screw terminal	Arduino Digital pin (D9)	1	N/A	Digital signal PWM signal	
J12, J13	Screw terminal	Arduino Analog pins (A4, A5)	2	I2C sensors	i2c sensors	SDA = A4, SCL = A5. VCC, GND terminals available nearby Already talking to to RTC clock address
J14, J15	Screw terminal	Filter	2	N/A	N/A	Can accommodate two condensators in parallel with the voltage divider analog input for noise filtering purposes.
Nano	Female header	D0, D1	2	Unavailable	Unavailable	Used to control Rx/Tx pins
Nano	Female header	D2, D3, D4, D5	4	Unavailable	Unavailable	Pin used to control multiplexer
Nano	Female header	D7, D8, D10, D11	5	Available	Available	5 digital pins available for temporary circuits
Nano	Female header	D12	1	Unavailable	Unavailable	Pin used to supply voltage to components
Nano	Female header	D13	1	Unavailable	Unavailable	Pin used to supply voltage to components. Can be used but also connected to the board LED.
Nano	Female header	A2, A3, A6, A7	4	Available	Available	5 analog pins available for temporary circuits

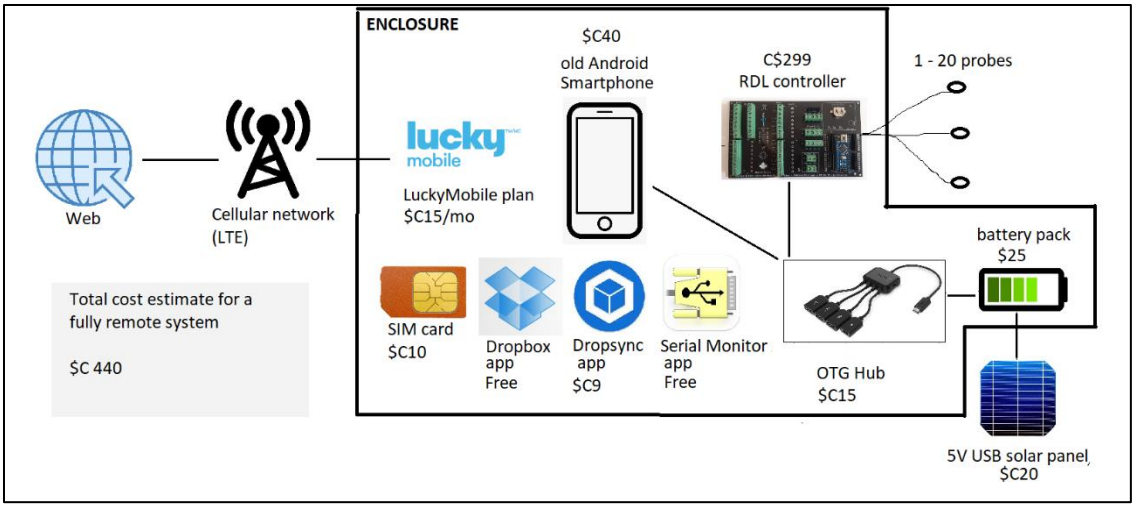


Figure 2 Fully autonomous RDL system components overview

PRODUCT FEATURES

- Low-cost portable electrical resistance multi probe data logger
- Designed for scientists, engineers and researchers
- Open-source software and hardware
- Arduino-compatible technology

- Measured variables:
  - dry bulb temperature (°C)
  - wet bulb temperature (°C), static
  - mean radiant temperature (with self-made black globe) (°C)
  - electrical resistance (ohm)
  - voltage (0 - 5 V)
  - illuminance (lux)
  - relative humidity (%)
  - Wet Bulb Globe Temperature index (WBGT) (thermal comfort index) (°C)
- 20 channels:
  - 16 input channels for resistive probes measurements only
  - 2 channels for I2C sensors
  - 1 channel for 0-5 V sensors
  - 1 channel for control signals
- Small (TH-1) or large (TH-2) temperature probes available
- 8 thermistor probes, calibrated, included in the Starter kit
- Factory-calibrated probes with an overall system accuracy of 0.5°C in the 0-100°C range (3-point calibration)

- Graph capability through a third-party software (e.g., Microsoft Excel®, LibreOffice ®, Python)
- Weatherproof case (inspired by NEMA Type 3 requirements)
- Waterproof probes (inspired by NEMA Type 6P requirements)
- Serial communication with PC/smartphone (ASCII data)
- Timestamp provided by a Real-Time Clock (RTC) subcircuit (day, month, year, hours, minutes, seconds)
- Data storage, power and communication provided by the PC/smartphone
- Acquisition rate up to 340 S/s (depending on many factors)

- Free access to the source code, calibration procedures, application notes
- Made in Canada
- Available on GitHub
- Ready-to-use
- Customizable

PRODUCT DESIGN AND CODE OVERVIEW

For more information about the product design choices, code architecture, calibration methodology and other topics, please consult the document titled “Product design and code overview”, available in the documentation section of our website (upcoming).

REQUIREMENTS

- Computer with minimal hardware:
  - CPU: 1 MHz; Memory: 250 MB RAM (Minimum Hardware)
  - Internet Access with Internet Explorer 9 or higher
  - 1 USB port (2.0 or 3.0)
  - Operating system: Windows®7 and newer, Mac OS X®, Linux®, Android® (See details below)
- If using with a smartphone, it must have “host capability” and be used with an OTG USB hub (not included)
- Wire-stripper for 22 to 28 AWG wire size

DISPLAY OUTPUT COLUMNS

Order into which the data is printed to the window or text file, from left to right. Each of these columns can be activated/deactivated separately. User judgement must be applied. For example, if asked to do so, the program will calculate and display thermistor temperatures even if no thermistor is connected.

1. Time (HH:MM:SS.MMM) (Arduino IDE Timestamp);
2. Date (YYYY/MM/DD) (RTC Timestamp);
3. Time (HH:MM:SS) (RTC Timestamp);
4. Identification (ID) number;
5. Thermistor temperature (channels 1-16);
6. Resistance values (channels 1-16);
7. Relative humidity values (channels 1-16);
8. I2C sensors readings (demonstration device is AM2301b) (channels 1-2);
9. Voltage readings (channels 1-2);
10. Luminosity readings (channels 1-16);
11. Soil humidity readings (channels 1-8) (Temperature compensated individually);
12. Wet Bulb Globe Temperature (WBGT) readings (Comfort index);
13. Control signal (0-5V).

QUICK START: MY FIRST EXPERIMENT USING THE RDL

This procedure will allow you to do a quick basic test of the RDL with the factory settings. It does not require modifying or compiling the source code. This will guide you through some measurements of an espresso machine brewing a cup of coffee. If you do not have access to a coffee machine, you can also use hot water.

Required time: 30 to 90 minutes.

Required material:

- Starter kit
- Access to a computer, an Internet connection and a spreadsheet software (e.g. MS Office, LibreOffice).
- Access to an espresso machine or hot water, and a cup.

Warning: DISCLAIMER: Jericho Laboratory Inc. will not be held responsible for your safety or any damage other than to the device itself. Be careful when manipulating hot liquids.

1- Get the RDL Starter Kit items out of the box.

Figure 3 Content of the Starter Kit rev D6

2- Insert a CR1220 battery (3V) in the battery holder of the RDL. The PLUS (+) sign of the battery should be facing up. (No battery is included with the package to reduce shipping restrictions). In the absence of a battery, the clock will not behave properly and might display wrong values.

*To finely synchronize the RTC timestamp with the official time, follow the instructions in the section titled “RTC timestamp adjustment”. This is not required for a first test.

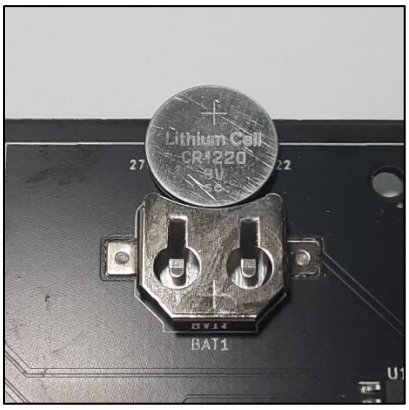


Figure 4 Inserting the CR1220 battery into the battery holder

3- Connect the USB cable from the RDL to the computer. The PWR (power) red light on the RDL should light up.

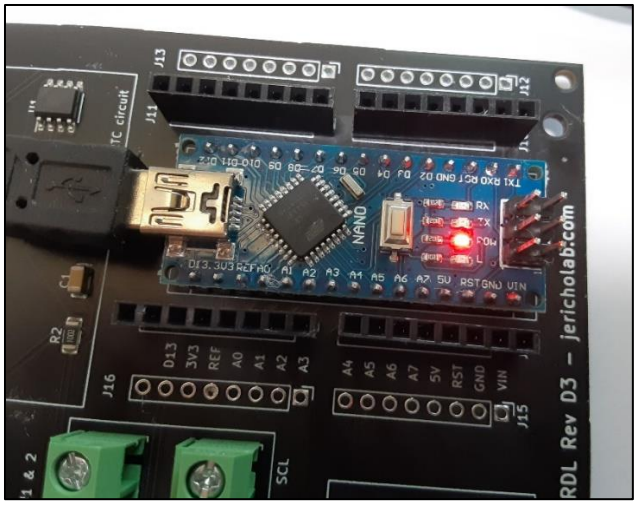


Figure 5 USB connection and power red light

4- If this is not already the case, connect each required thermistor to its dedicated channel with the provided screwdriver. For this quick test, three probes are required. Other probes can remain connected without problems. Thermistors have no polarity, so you can invert the two wires without consequence. Make sure that the thermistor ID is coherent with the channel in the source code (see Table 2). Make sure that the terminal contact is solid by pulling slightly on each wire. If using extension wire, the wire gauge should be AWG26 or larger to ensure a solid and easy connection to screw terminals. Smaller gauge can be difficult to use.

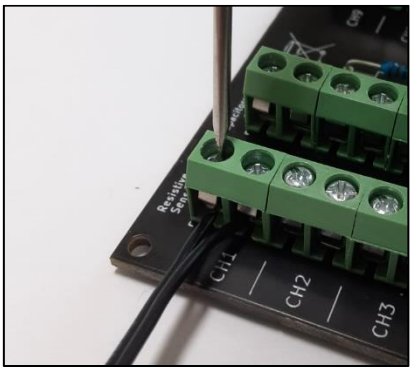


Figure 6 Connecting the thermistor probe wires to the dedicated channel C1

Table 2 Example of an 8-probe setup with matching probe ID's

Channel	Connected Probe ID	Source code Probe ID
C1	S002	S002
C2	S003	S003
C3	S004	S004
C4	S008	S008
C5	S009	S009
C6	S010	S010
C7	S011	S011
C8	S012	S012

5- Download the latest version of the free Arduino Software IDE for your operating system on the Arduino official website (www.arduino.cc). Follow the instructions provided by the website. We suggest installing the English version to more easily follow the User Guide instructions initially.

Figure 7 Arduino official website software download section

6- Launch the IDE software and modify the configuration settings. Go to Menu -> Tools.
   a. Board -> Arduino Nano
   b. Processor -> ATMega328P (Old Bootloader)
   c. Port -> COM4. The exact port name might vary depending on your system. (e.g., ttyUSB1, ttyUSB2, COM1, etc.) You can disconnect/reconnect the USB cable to see which port name appears when you connect the RDL.

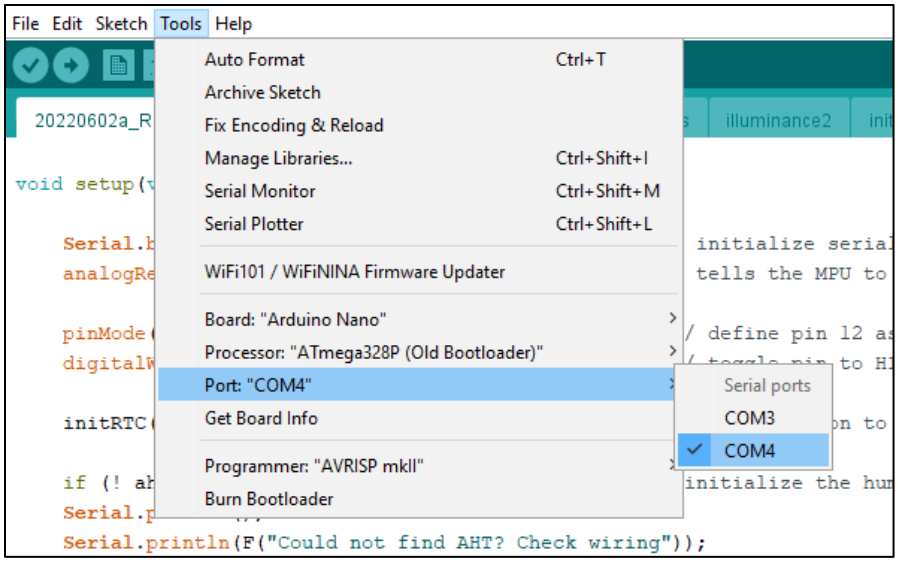


Figure 8 Serial port selection through the Arduino IDE menu

7- Go to Menu -> Tools -> Serial Monitor. A new window should pop-up. At first, the data being displayed on your screen will be gibberish, because the baud rate (i.e., communication rate) is wrong.

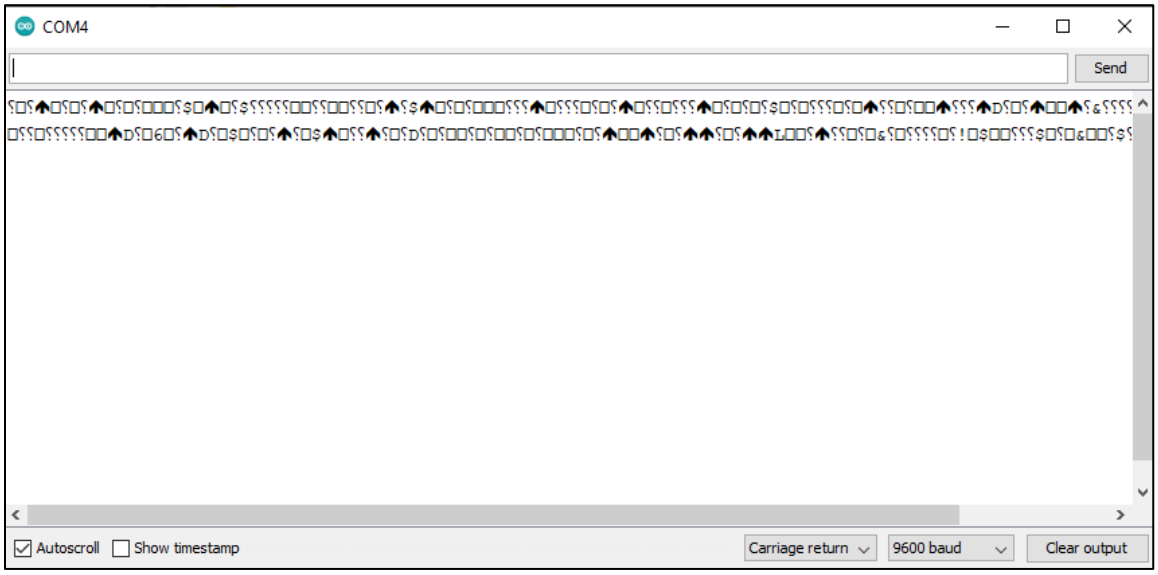


Figure 9 Example of gibberish data coming in due to a mismatch of the baud rates

8- The baud rate inside the RDL source code should match the baud rate of the computer. On the bottom right of the Serial Monitor window, set the baud rate to 57600 (program default is 9600). Once the baud rate is correct, readable data will start coming in through the Serial Monitor.

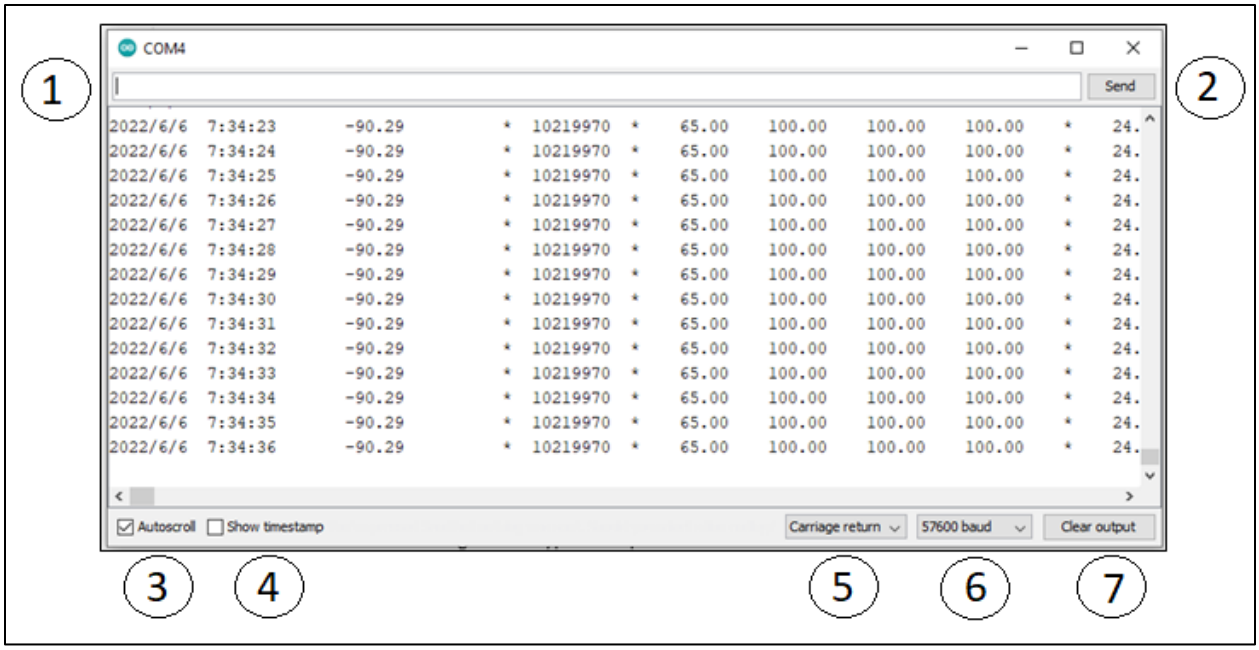


Figure 10 Serial Monitor window components. 1) Command line; 2) Send command button; 3) Activate/Deactivate autoscroll; 4) Show/Hide timestamp; 5) Selection of the end of line symbol; 6) Selection of the baud rate; 7) Clear the window.

9- Select the 'Carriage return' in the End of line options (item 5 in Figure 10). Otherwise, the command will not be recognized by the RDL.

10- The factory setting for the acquisition rate is 1 Hz (i.e., 1000 milliseconds interval). You can modify it to 2000 ms or leave it at 1000 ms. To modify that value, you must enter the text "INTERVAL" in the command line (item 1 in Figure 10).

11- If there are too many columns, you can zoom out by using CTRL + the mouse wheel. 'Autoscroll' can be deactivated to look at the data.

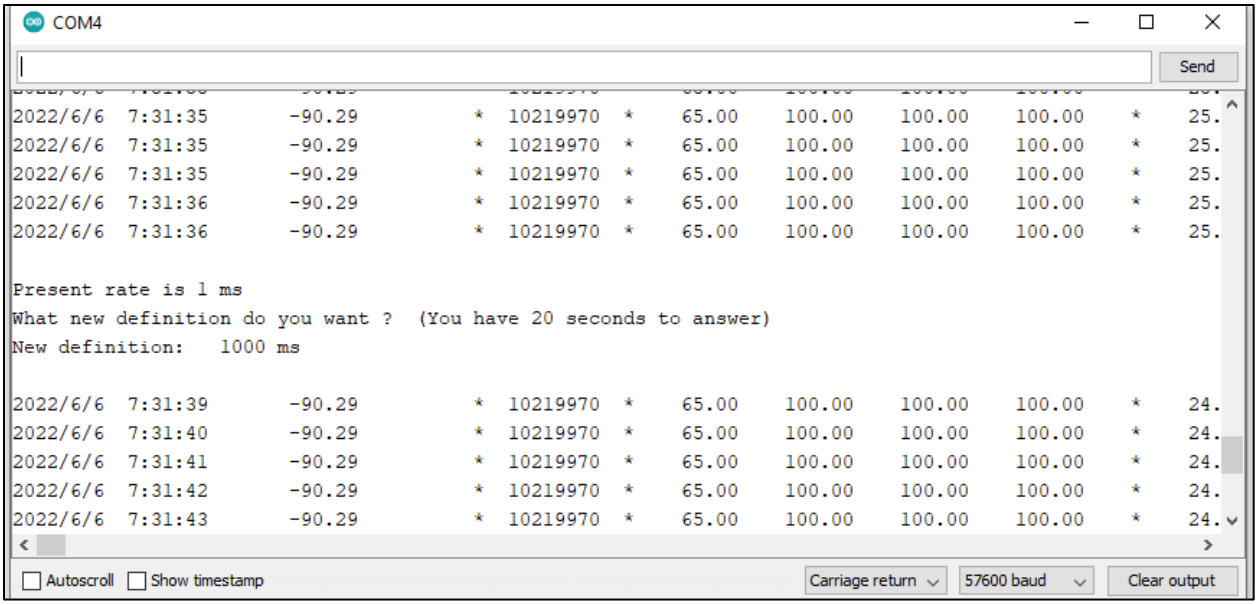


Figure 11 Example of an interval modification from 1 to 1000 ms through the command line

12- Before starting your experiment, you can clear the window by clicking on “Clear output” in the bottom right of the window (item 5 in Figure 10). Otherwise, non-relevant data can be erased later during post-treatment.

13- You can now proceed with your experiment. Meanwhile, using adhesive tape, position the three probes in the following way.

   a. Probe C1: Near the liquid exit to measure the temperature of the coffee outflow;
   b. Probe C2: Near the bottom of the cup, without touching the bottom surface, if possible. This will allow measuring the global temperature of the coffee accumulated in the cup.
   c. Probe C3: In the center of the top surface of the espresso machine. This surface becomes hot during the warm-up and operation phase since the boiler is right beneath the surface. It can be interesting to see how its temperature evolves over time.

14- Prepare the coffee as usual, and turn on the machine to preheat. Wait until the temperatures stabilize on your screen before activating the compressor (brewing phase).

Figure 12 Caption

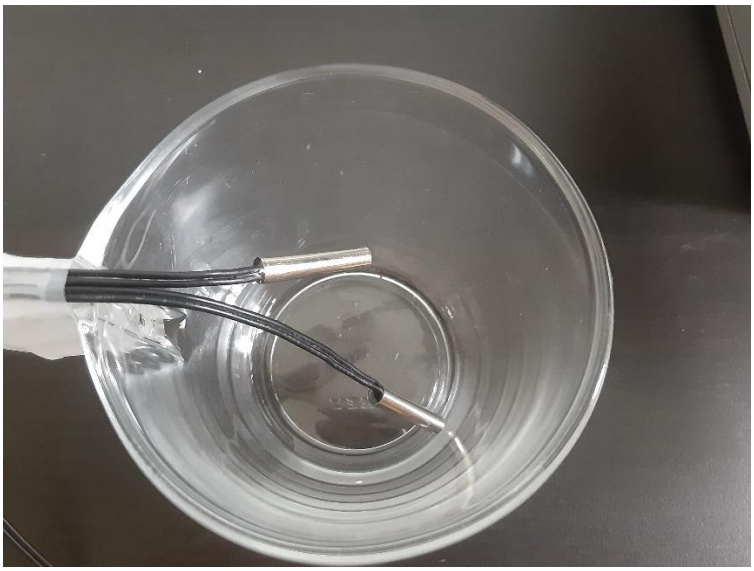


Figure 13 Position of the two probes inside the empty cup

15- As the experiment unfolds, watch the data accumulate in the Serial Monitor window. Each time, the RDL sends data, the light 'TX' on the controller should blink very quickly.
16- When you have gathered all your data, untick the case 'Autoscroll'. This will prevent the screen from moving while you select your data. Do not close your Serial Monitor window until you have saved your data in another reliable location.
17- Press "CTRL + A" to select all text. Then copy by pressing "CTRL + C".
18- Open a text editor and paste all text by pressing "CTRL + V".

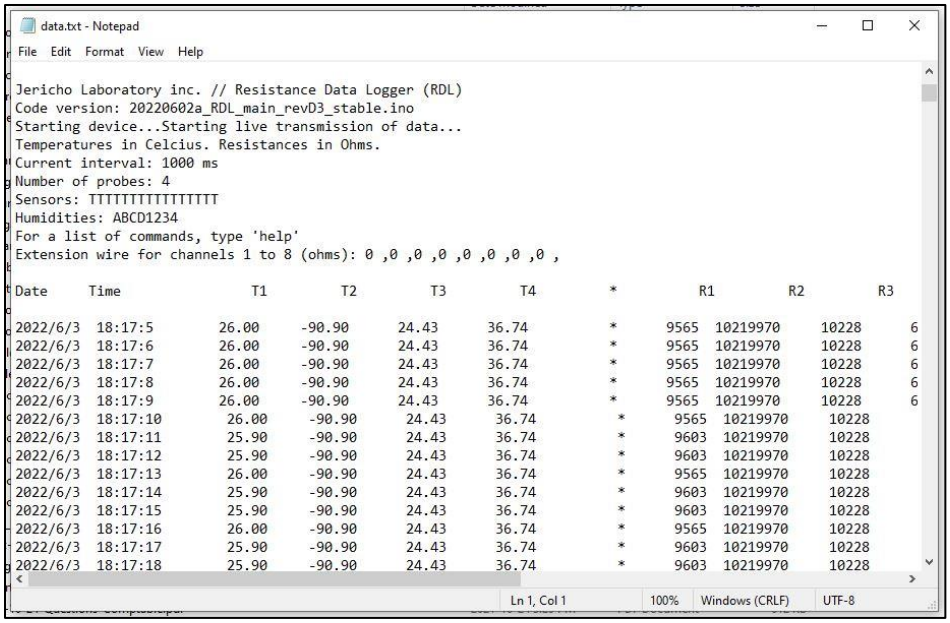


Figure 14 RDL text file output sample

19- Save the text file on your Desktop (ex: test_1.txt). This is an important step, since the data within the Serial Monitor is not recorded permanently. Data is lost when the window closes.

20- Open Microsoft Excel or, alternatively, a free spreadsheet software (e.g., LibreOffice Calc).

21- Go to Menu -> Data -> From text. Select the text file. In some recent versions of Microsoft Office, you might have to activate “Legacy Wizards” in order to have access to data import from a text file. If this is your case, follow Microsoft instructions on the topic on their forum.

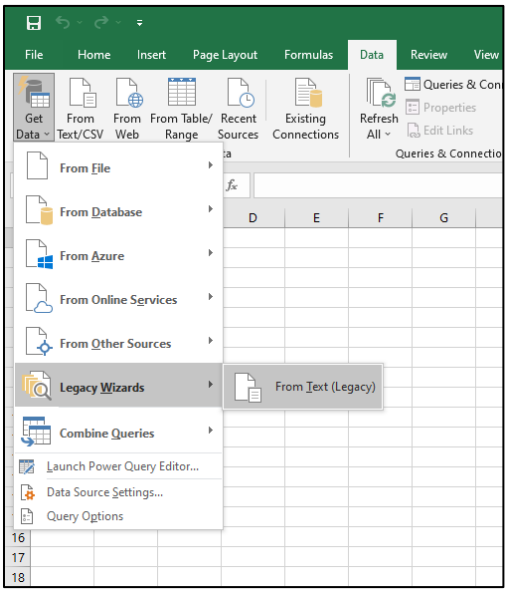
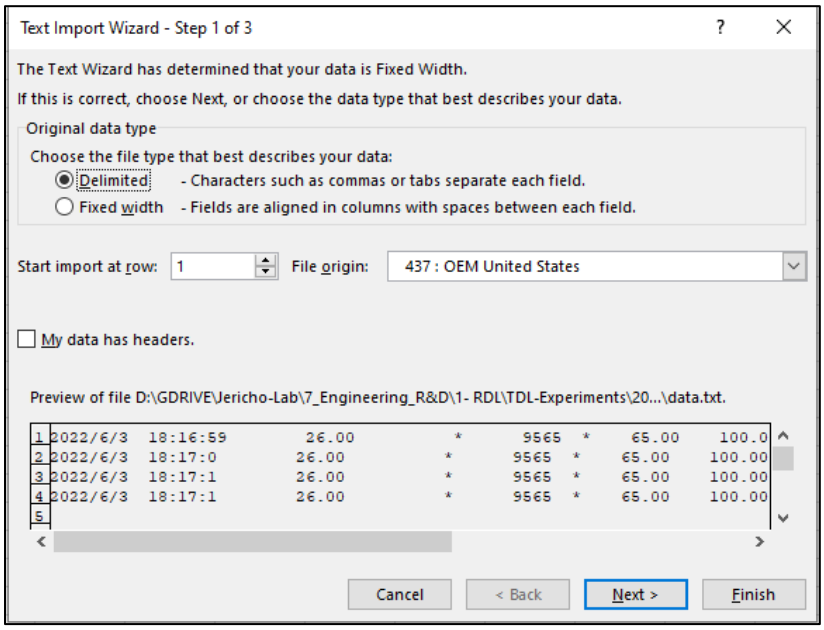
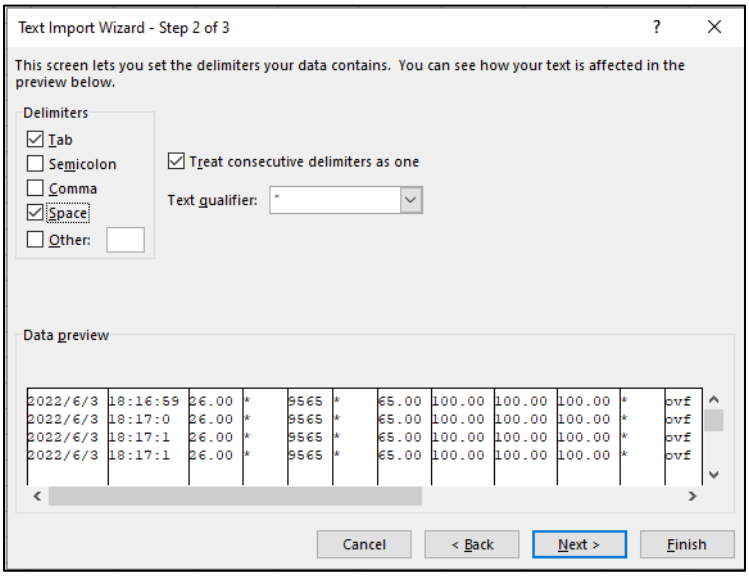


Figure 15 Microsoft Excel menu for the Text Data Import

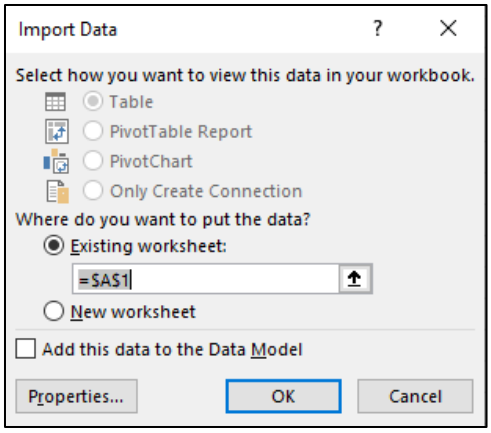
22- Select the ‘Delimited’ data type.



23- Select the ‘space’ character as the delimiter. Tick the box ‘Treat consecutive delimiters as one’. Click Finish.

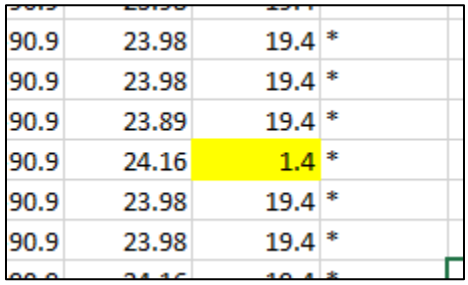


24- Import data to ‘Existing Worksheet’.



25- If necessary, adjust the 'timestamp' column cell format to 'Short Date' in order to display the time properly.

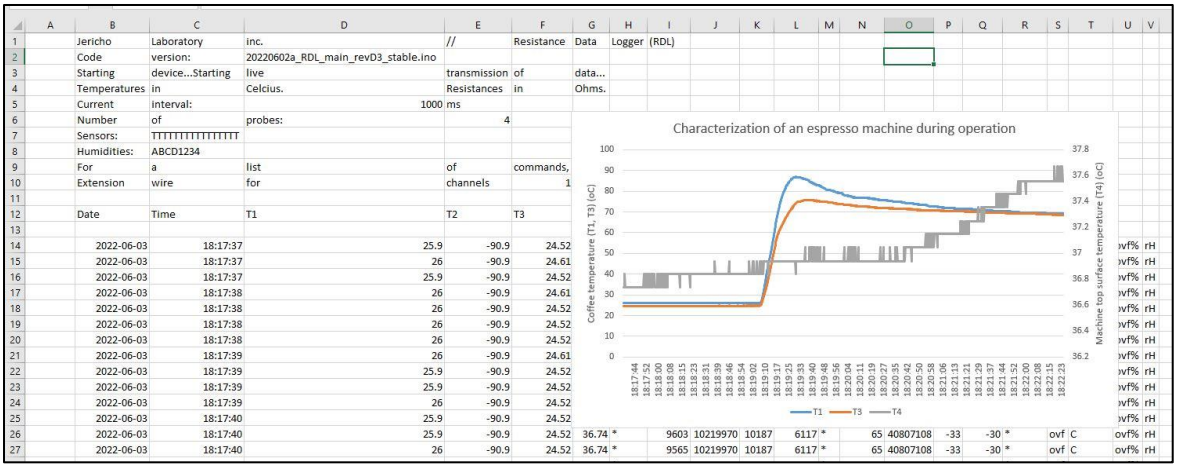
26- Clean up the data. Over long periods of measurements, a few lines will have a missing character. This happens more frequently with smartphones than computers.



90.9	23.98	19.4	*
90.9	23.98	19.4	*
90.9	23.89	19.4	*
90.9	24.16	1.4	*
90.9	23.98	19.4	*
90.9	23.98	19.4	*

Figure 16 Cleaning up the raw data

27- You can now create various graphs from your data. For example, you might create a "Line graph" with the column 'Time' as the x-axis and three temperatures as the y-axis. In our case, we decided to put data from channel C4 on a secondary vertical axis.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1		Jericho	Laboratory	inc.	//	Resistance	Data	Logger	(RDL)													
2		Code	version:	20220602a_RDL_main_revD3_stable.ino																		
3		Starting	device...Starting	live	transmission	of	data...															
4		Temperatures	in	Celcius.	Resistances	in	Ohms.															
5		Current	interval:	1000	ms																	
6		Number	of	probes:	4																	
7		Sensors:	TTTTTTTTTTTTTTT																			
8		Humidities:	ABCD1234																			
9		For	a	list	of	commands,																
10		Extension	wire	for	channels	1																
11																						
12		Date	Time	T1	T2	T3																
13																						
14		2022-06-03	18:17:37	25.9	-90.9	24.52															ovf%	rH
15		2022-06-03	18:17:37	26	-90.9	24.61															ovf%	rH
16		2022-06-03	18:17:37	25.9	-90.9	24.52															ovf%	rH
17		2022-06-03	18:17:38	26	-90.9	24.61															ovf%	rH
18		2022-06-03	18:17:38	26	-90.9	24.52															ovf%	rH
19		2022-06-03	18:17:38	26	-90.9	24.52															ovf%	rH
20		2022-06-03	18:17:38	26	-90.9	24.52															ovf%	rH
21		2022-06-03	18:17:39	26	-90.9	24.61															ovf%	rH
22		2022-06-03	18:17:39	25.9	-90.9	24.52															ovf%	rH
23		2022-06-03	18:17:39	25.9	-90.9	24.52															ovf%	rH
24		2022-06-03	18:17:39	26	-90.9	24.52															ovf%	rH
25		2022-06-03	18:17:40	25.9	-90.9	24.52															ovf%	rH
26		2022-06-03	18:17:40	25.9	-90.9	24.52	36.74	*	9603	10219970	10187	6117	*	65	40807108	-33	-30	*	ovf	C	ovf%	rH
27		2022-06-03	18:17:40	26	-90.9	24.52	36.74	*	9565	10219970	10187	6117	*	65	40807108	-33	-30	*	ovf	C	ovf%	rH

Figure 17 Data treatment inside Microsoft Excel

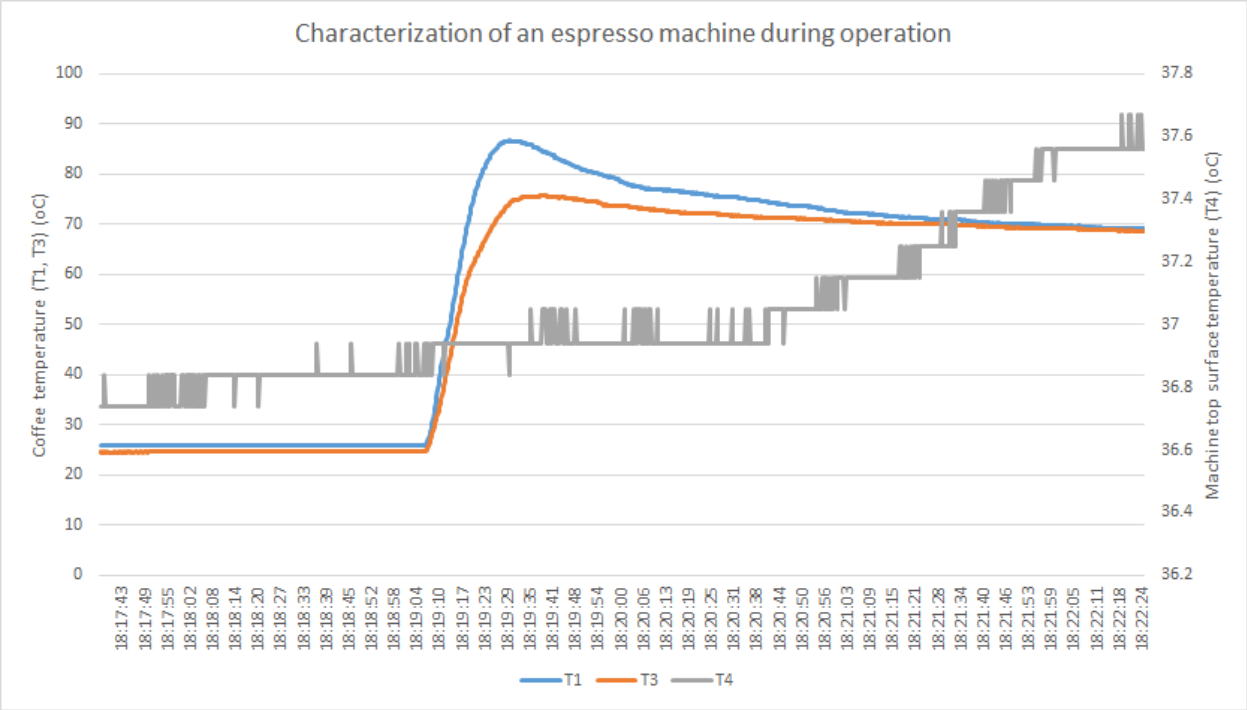


Figure 18 Example of data post-treatment with MS Excel ® for the espresso machine experiment

28- Analyzing the data is the final and most interesting part of the experiment. First, it is good practice to analyze the quality of the data. Are there any surprises, irregularities, noise in the data? Second, what are the trends that we observe in the data (peaks, valleys, plateau, oscillations, etc.)? Here are a few examples for the espresso machine graph.

   d. **The temperature of the coffee outflow sensor increased very fast initially**. Due to the thermal inertia of the large sensor, it is not clear at what time the coffee temperature outflow really peaked. It would be interesting to rerun the experiment with a small

probe, which has lower inertia. If the curve does not change, that would suggest that probe inertia does not play a significant role on this specific aspect.

e. **Both coffee temperature sensors peaked before the end of the brewing**. Since we know that the brewing lasted around 30 seconds, we can confirm that the heater element is not powerful enough to maintain the maximum temperature of the water. The maximum heating power [watts], the coffee grain thermal inertia and the water reservoir temperature are likely to play a role in the curve of the coffee temperature.
f. **There seems to be a slight inflexion point around 18:20:06**. This could be the time at which the brewing stopped. We could improve our observation by plotting a first order time derivative of the data for C1 and C3. It would be easier to see if our eye is being misled. That could be confirmed with a second experiment which measures the start and stop of the compressor. This could be measured in one of several ways: manually with a clock, with a current sensor, or with a digital switch.
g. **The two sensors eventually converged to a very similar temperature.** This is likely to indicate that after a few seconds without liquid inflow, the stratification became negligible due to conduction and convection within the cup.
h. **There was very little change in the temperature of the top surface (less than a degree)**. This was a surprising result as one could expect that the incoming cold water from the reservoir would have had some effect on the machine casing temperature. Also, with the help of the secondary axis, we can see that much of the temperature increase happened after the brewing. This challenges our understanding of the espresso machine.
i. **The top surface temperature had not peaked when the experiment ended.** It would be interesting to run the experiment longer for that reason.

   As you can see, much can be learned from a single graph. A first experiment often leads to a series of more refined experiments.

29- We advise you to keep the original data (i.e., text file) in case you eventually want to restart your analysis from scratch for a variety of reasons.

RDL FACTORY SETTINGS

Factory settings are the settings active when you first receive your device. Most of the RDL capacities are active in order to easily demonstrate every aspect of the product. The factory settings are a combination of memory parameters and source code parameters. You can alter the factory settings by combining serial commands and small source code alterations. You can always go back to factory settings by using the command “EEPROM-ERASE”.

Factory settings (EEPROM section)

- Acquisition interval: 1000 mS
- Number of probes: 16

- Temperature unit: Celsius
- Sensors probes: All 16 resistive channels are defined as temperature sensors. The corresponding 16-character code stored in the microcontroller is “TTTTTTTTTTTTTTTT”.
- Humidity probes: Channel C1 is associated with a dry bulb (A) and Channel C2 is associated with a wet bulb. The corresponding 8-character code stored in the microcontroller is “A1000000”.

Factory settings (source code section)

- Time display: Yes
- ID display: Yes
- Temperature display: Yes
- Resistance display: Yes
- Humidity display (wet bulb based): Yes
- I2C display: Yes
- Voltage reading display: Yes
- Luminosity display: Yes.
- Soil humidity display: Yes
- Wet Bulb Globe Temperature (WBGT) display: Yes
- Steinhart-Hart thermistor coefficients (A,B,C):
    - For the Starter Kit package
        - Channels C1 to C8: Custom coefficients specific to included thermistor probes
        - Channels C9 to C16: Generic coefficients for 10k NTC thermistor
    - For the Controller-only package
        - Channels C1 to C16: Generic coefficients for 10k NTC thermistor

AVAILABLE SERIAL COMMANDS

In order to interact with the RDL, the user must type in some of the following keywords at the top of the ‘Serial Monitor’ window and then click ‘Send’. The option ‘Carriage return’ must be selected for the RDL to detect the command. The commands are not case-sensitive.

HELP: The command ‘HELP’ prints out the available serial commands. Activity will resume after a few seconds.
INTERVAL: The command ‘INTERVAL’ allows adjusting the measurement interval (i.e., acquisition rate). The same rate applies to all sensors. After receiving the command, the program will wait for the user to enter a value between 1 and 96 400 000 ms (i.e., 24 h). Enter the value ‘1’ to force maximum speed, which will vary depending on the number of active probes. The default factory value is 1000 ms (1 second).

The maximum acquisition rate depends on a combination of factors:
- amount of sensors/values/data printed per poll
- baudrate (source code default is 57600, but the Nano can reach 115200)

- RTC use (DS3231b performance degrades above 57600)
- speed of the PC/smartphone (buffer overload)

Note: For intervals shorter than 1000 ms, we recommend activating the serial monitor native timestamp, which includes milliseconds. Otherwise, all datapoints taken within the same second will receive the same timestamp from the RTC (hh:mm:ss), which can be problematic for plotting time series afterwards. You can keep the RTC timestamp though, as the native timestamp does not include the date.

CELSIUS: The command 'CELSIUS' tells the device to display temperatures in Celsius (°C) units. Unit is stored in the permanent memory (EEPROM) and will be remembered after a shutdown or reset of the device. Celsius is the default factory setting. No confirmation or further instructions are required. Operation will resume after a few seconds.

FAHRENHEIT: Display temperature in Fahrenheit (°F) units. No confirmation or further instructions are required. Operation will resume after a few seconds.

KELVIN: Display temperatures in Kelvin (K) units. No confirmation or further instructions are required. Operation will resume after a few seconds.

COEFF: Print out the Steinhart-Hart coefficients presently stored in the controller memory. No confirmation or further instructions required. Operation will resume after a few seconds.

EEPROM-ERASE: Erase the EEPROM memory and return to the factory settings (RATE 1000 ms, Celsius units, all 16 channels active). No confirmation or further instructions required. Operation will resume after a few seconds.

RESET: This command restarts the microcontroller. It is the equivalent of pressing the RESET button on the Arduino Nano. This is convenient to print out the header when starting a clean measurement session for logging purposes. Operation will resume after a few seconds.

SENSORS: This command determines which sensor type is connected to each of the 16 multiplexed channels. After the initial command 'SENSORS' is sent, the program expects a 16-character string to be entered (e.g., "TPTPTPTPTTTTTTTT"), where T stands for temperature sensor, P stands for photoresistor. 30 seconds are allowed to enter the data. If no data is received within this delay, the previous value is maintained and operation resumes.

HUMIDITY: Relative humidity is calculated by post-treatment of the sensors data. A dedicated string called humidity informs the microcontroller of which channels are dry bulb temperature sensors and which ones are wet bulb temperatures sensors. 30 seconds are allowed to enter the data. If no data is received within this delay, the previous value is maintained and operation resumes.

Letters (A,B,C...) identify the dry bulb temperature sensors.

Digits (1,2,3...) identify wet bulb temperature sensors.

'0' identify the channels which are not concerned with humidity measurements.

The factory setting is 'A100 0000 0000 0000''.

Example #1: 'ABCD123400000000 '

This is the maximum number of independent humidity measurements that you can do with one RDL device. Channels C1, C2, C3, C4 are dry bulbs and are matched with wet bulbs from the channels C5,C6, C7, 8 respectfully. Channels 9 to 16 are not concerned with humidity.

Example #2: 'AB00120000000000'
This is an example of two humidity measurements and 12 channels not concerned with humidity measurements (could be surface temperatures or else).

Eample #3: 'A000120000000000'
This is an example of vector that will generate an error due to an unpaired dry bulb channel. The program will therefore keep the previous parameters.
- - - - - - - - - - - - - - - -

IMPORTANT USER PARAMETERS

Most variables within the source code should not be modified by the user during normal operations, but some important parameters are not available through the serial commands and need be modified within the source code before uploading it to the microcontroller.

- **Generic**
  - Activate (1) or Deactivate (0) the use of the generic Steinhart-Hart coefficients of thermistors (10 kohm @ 25C, B = 3950). The generic coefficients should be used when uncalibrated thermistors are connected. The generic coefficients should be deactivated when calibrated thermistors are connected to the RDL and the known coefficients have been substituted in the source code.
  - Generic = 1 (Active)
  - Generic = 0 (Inactive)
- **Time display**
  - Activate (1) or Deactivate (0) the use and display of the Real-Time Clock (RTC). The Real-Time Clock allows the RDL to keep track of the date and time with high accuracy. The RTC clock is polled at every cycle and its data is printed out in the serial monitor. This is especially useful for experiments that last more than 24 hours since most Serial Monitor Data Logging software do not keep track of the date. The RTC poses a restriction however in the maximal speed that the RDL can operate (approximately 5 Hz).
  - TimeDisplay = 1 (Active)
  - TimeDisplay = 0 (Inactive)
- **Identification Number display**
  - Activate (1) or deactivate (0) the display of the identification number (ID). By default, the source code attributes an identification number to each measurement (i.e., 1, 2, 3, etc.) The ID can supplement or substitute the timestamps in some cases, such as high-speed measurements. It is also useful during post-treatment to keep track of the data continuity (i.e., lines 2450 to 2460 are missing due to bad data)
  - idDisplay = 1 (Active)
  - idDisplay = 0 (Inactive)

- **Temperature/Illuminance display**
    - Activate (1) or Deactivate (0) the display of the temperature/illuminance value for the multiplexed channels. Deactivating the temperature display can be useful when the user is not using the multiplexed channels or is not using them with thermistors or photoresistors. In that latter case, the user might only activate the display of the resistance values.
    - tDisplay=1 (Active)
    - tDisplay=0 (Inactive)
- **Resistance display**
    - Activate / deactivate the display of the resistance value for the multiplexed channels. Deactivating the resistance display can be useful in cases where the user is not using the multiplexed channels or is only interested in temperatures.
    - ohmDisplay = 1 (Active)
    - ohmDisplay = 0 (Inactive)
- **Humidity display (wet bulb based)**
    - Activate/Deactivate the calculations and display of relative humidity based on the dry bulbs and wet bulbs available.
    - humDisplay = 1 (Active)
    - humDisplay = 0 (Inactive)
- **I2C display**
    - Activate (1) or deactivate (0) the use and display of the I2C sensors connected to the I2C channels of the RDL. The content of this tab can vary. The native source code demonstrates the I2C abilities of the RDL with the AM2301b digital temperature/humidity sensor.
    - I2CDisplay = 1 (Active)
    - I2CDisplay = 0 (Inactive)
- **Voltage display**
    - Activate/Deactivate the measurement, calculation and display of the voltage input in the corresponding channels (0-5V, continuous current).
    - VoltDisplay = 1 (Active)
    - VoltDisplay = 0 (Inactive)
- **Soil content humidity display**
    - Activate/Deactivate the calculations and display of the soil content humidity based on the gypsum blocks and the temperature probe available. The gypsum block resistance being sensitive to temperature, they must be corrected.
    - SoilDisplay = 1 (Active)
    - SoilDisplay = 0 (Inactive)
- **WBGT display**
    - Activate/Deactivate the calculations and display of the Wet Bulb Globe Temperature. The WBGT is calculated from three values: the natural wet bulb temperature, the dry bulb temperature and the black globe temperature.

- WBGTDisplay = 1 (Active)
- WBGTDisplay = 0 (Inactive)

HOW TO UPLOAD THE SOURCE CODE TO THE RDL

The RDL already comes ready-to-use, but you might want to make some changes to the source code or use the latest release.

1- If this is not already done, do steps 1 to 6 inclusively from the Quick Start section.
2- Download the latest source code on the Jericho website (zip file).
3- Unzip the source code in your working directory. The folder name should be the same as the main file without the file extension (e.g., 20220602a_RDL_main_revD3_stable)
4- Open this code file with the Arduino IDE software. (File -> Open)
5- Make the desired modifications to the code.
6- Insert the Steinhart-Hart coefficients (thermistor) to match your physical hardware into the main tab of the source code. The source code from the website has generic coefficients, which you can comment out using the '//' symbol. You can find the coefficients specific your serial numbers in the Jericho database.

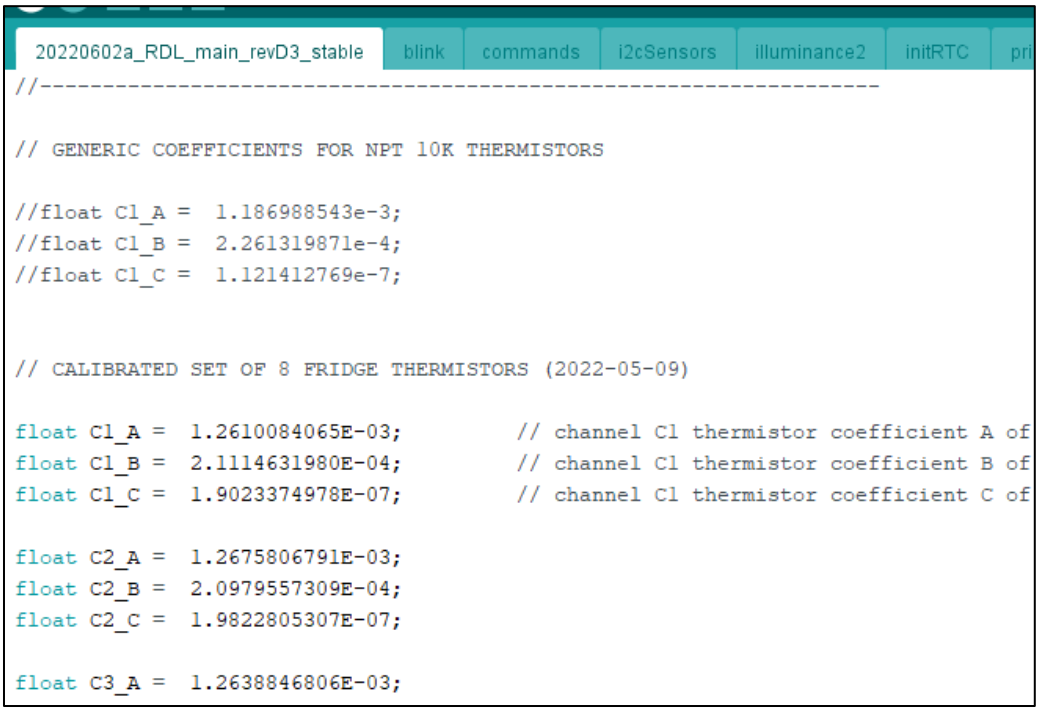


Figure 19 Custom thermistor coefficients updated in the source code, based on calibration certificate

7- Connect the RDL to the computer via the USB cable.
8- Make sure that the proper USB port is selected by the software.
9- Install all the required libraries. Go to Menu -> Manage Libraries. This is necessary even if you don't plan on using some of the features (e.g., I2C humidity sensors) due to their presence in the

source code, even when the features are not activated. EEPROM.h is included in the source code but is not required to install since it is native to the IDE software.

a. RTClib.h by Adafruit
b. Adafruit_AHTX0 by Adafruit

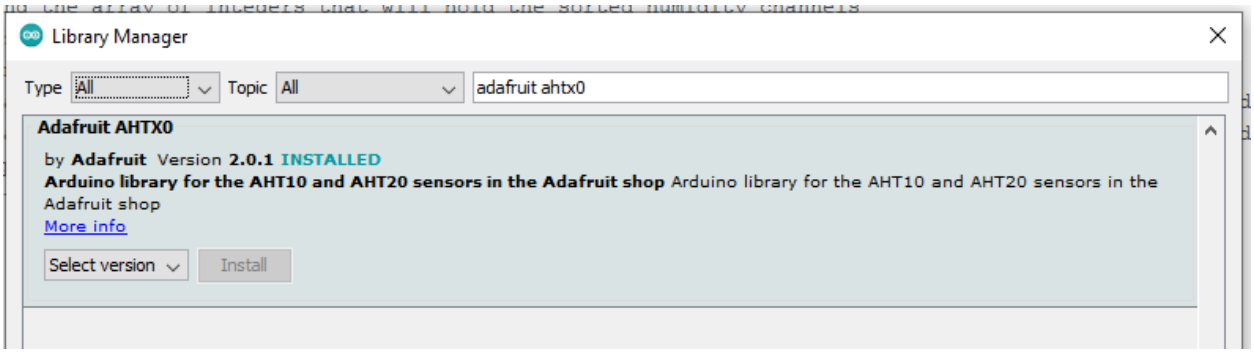


Figure 20 Searching the Adafruit AHTX0 library within the Arduino IDE Library Manager

10- Go to Menu -> Sketch -> Upload. This will compile then immediately upload the code to the RDL. The "RX" and "TX" LED light should flash quickly while the code is being uploaded. Normal upload time should be 10 seconds. Normal compile time should 10 to 15 seconds. Total time should then be 20 to 25 seconds.
11- Follow the steps from the Quick Start guide to open the Serial Monitor and see if you get the expected results.

For more help about Arduino coding and debugging, visit the official forum on arduino.cc or ask your local community: the Arduino platform is very common worldwide.

HOW TO LOG DATA TO FILE FOR COMPUTERS (UPCOMING FEATURE)

The Arduino IDE can hold large quantities of data inside the serial monitor window. However, this data is kept in memory temporarily, and eventually the data would have to be copied manually to a text file for permanent storage. There is always a chance that the computer or program would shut down before doing that operation. It is therefore best to save data as we go to a text file. There are a few different ways to log the data, depending on your setup and skills.

The 'logging script' can be used.

1- Download the latest 'logging script' from the Jericho website.
2- Open a new Terminal window.
3- Go to the script folder.
4- Launch the script.
5- After a few seconds of data logging, open the text file to make sure that data is being saved on a continuous basis.
6- (Optional) Launch the gnuplot script in a second window for a live visualization of the data.

HOW TO USE THE RDL WITH A COMPUTER

While the Arduino IDE can be used for logging with some projects, here is an overview of the different ways of using the RDL with a computer.

Important: Whatever the method chosen, it is important that the computer does not sleep. When going to sleep (e.g., closing the lid) the data logging is interrupted. The USB voltage supply is maintained but the serial communication is interrupted.

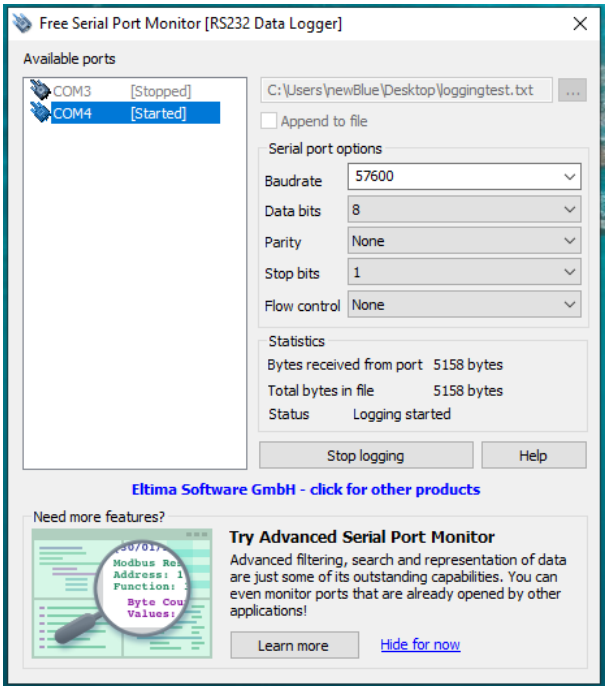


Figure 21 Example of a free Serial Monitor Program for Windows

Coding (alternatives to the official Arduino IDE)

- Atom (with the PlatformIO add-on)

Data Logging (Windows)

- Jericho Processing script (free, open source, tested)
- Free Serial Port Monitor by Eltima (free, proprietary, tested)
- YAT (Yet Another Terminal) (free, open source, not tested)
- HyperTerminal (paid, proprietary, not tested)

Data Logging (Linux)

- Jericho Processing script (free, open source, tested)
- CuteCom (free, proprietary, only saves to file at closing, which is a data loss risk)

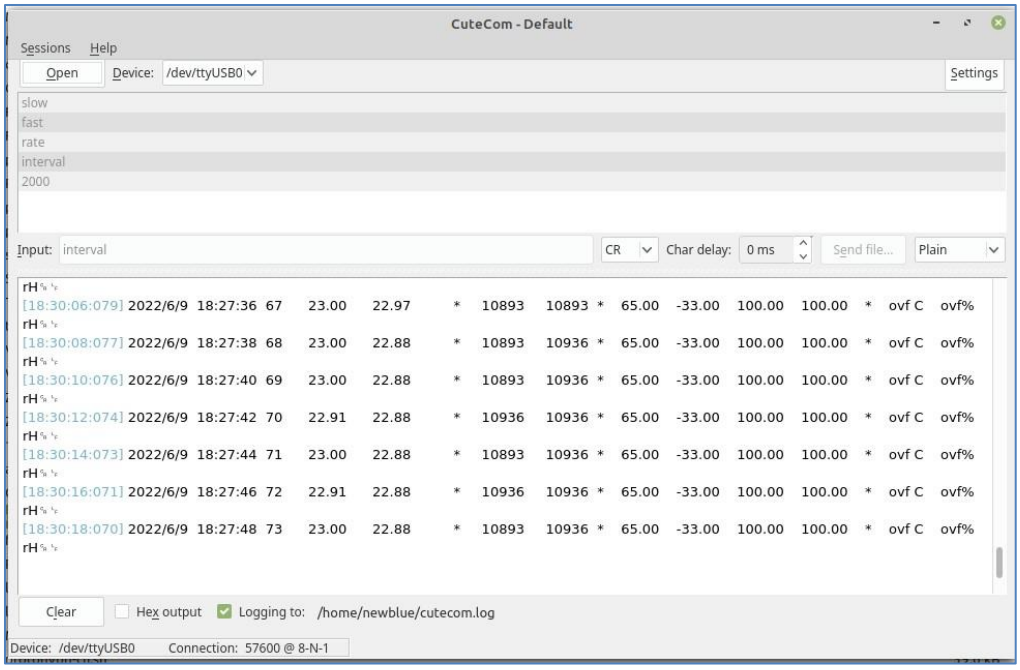


Figure 22 Screenshot of CuteCom while data logging from the RDL

Data Logging (Apple)

We currently do not support Apple devices. The Processing script should work, some Serial Monitor apps are available (e.g., CuteCom), but we did not do any test with the Apple OS environments.

Live Graph (all OS)

The RDL is not aimed at users with live visualization needs. There are however two methods to do so minimally, and it might satisfy some users needs.

1- Serial Plotter

The official Arduino IDE has a Plot Monitor, which allows to automatically plot the incoming serial data if the data is clean. This requires to deactivate the prints that are not required or compatible. For plotting the thermistor temperature values, you will have to deactivate: ID, RTC timestamp, resistance, humidity, I2C sensors. This will make sure that the program can detect the relevant data (i.e., temperatures) and select a suitable axis range to display it.

Important: it is impossible to save the data while plotting with the Serial Plotter. The Serial Plotter does not keep the data in memory. Even if you try to activate a second program, like CuteCom, it is not possible for two programs to maintain a connection with the microcontroller simultaneously. Therefore, the live graph must be used for debugging purposes, or applications that do not require data logging.

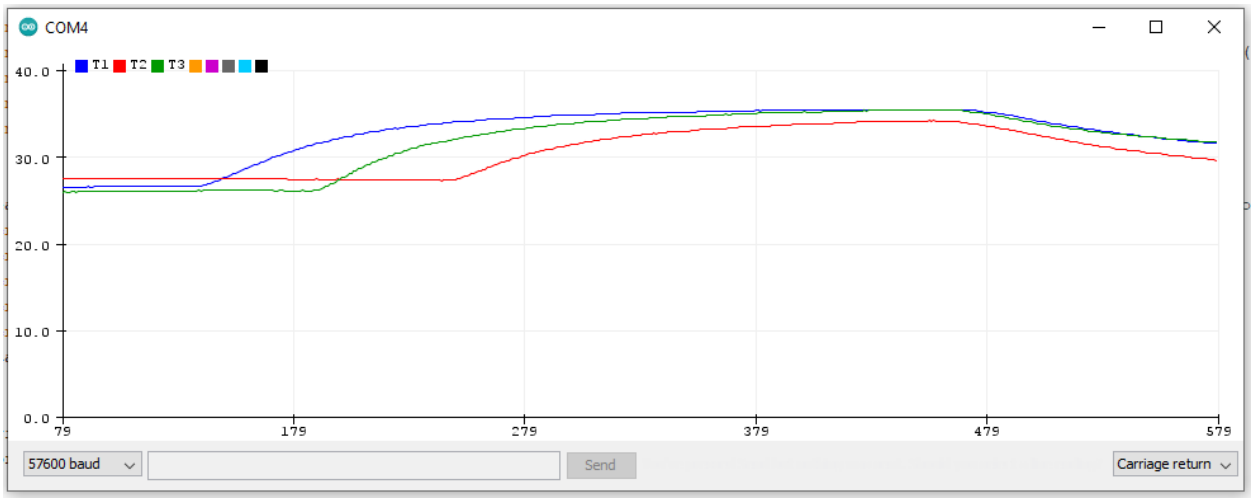


Figure 23 Example of a simple live temperature plot available by combining the RDL and the Serial Plotter feature by Arduino IDE

2- Processing + Gnuplot

There is the possibility of logging the data continuously with a Processing script, and then write a second script (with Gnuplot or Python) that would poll the text file continuously. This would allow the user to visualize live data while logging as well. However, we do not currently support this option.

HOW TO USE THE RDL WITH A SMARTPHONE

Smartphones can be used to communicate and log the data from the RDL for significant periods of time. There is currently no way to draw live graphs from a smartphone. A free third-party Android apps exists, ArduinoDroid by Anton Smirnov, to write, compile and upload code to the Arduino-compatible microcontroller, but it currently has limited features and reliability.

To use the RDL controller with a smartphone, you will need:

- RDL controller;
- Computer with internet access (for the initial configuration of the RDL and the post-treatment)
- On-The-Go hub such as the SUMAYA, with the good USB type (around $C15 on Amazon);
- Serial Terminal Android app, such as SERIAL USB TERMINAL, by Kai-Moriche (free, available on PlayStore)
- Cloud account, such as Dropbox (free up to 10 GB);
- Cloud Android client, such as Dropsync by MetaCtrl
    - free up to 10MB files, $C9 for unlimited size
    - use the one-way sync mode (upload only) to avoid file writing conflicts
- Wifi access or a dedicated smartphone data plan

Figure 24 Samsung S3 logging from the RDL rev D3 through an OTG hub

Jericho does not sell the accessories required for smartphone operation (i.e., OTG cable, OTG hub). We do not recommend USB adapters, since the ones we tested have proved to be unreliable (e.g., intermittent contacts). Hubs have shown to be of better physical constructions and reliability. While shopping, be careful to avoid “charging-only’ cables. These cables have four pins instead of five, and will not allowing communication.

In the field, it is better to not use a computer. Because you want to have a more compact system and avoid leaving a high-cost computer in the field. The solution is to use a smartphone. The smartphone role is to provide energy, data storage and communication. Smartphone mode operation can also be useful to reduce cost, since an early-generation smartphone can be cheaper than a desktop or laptop computer.

Smartphone also allow the use of the cellular network (e.g., LTE), without the use of any additional component. It can simply upload the data through the cellular network. This allows the monitoring of stuff in remote locations such as forests, agricultural fields without the need to go pick up the data on a regular basis. The live transmission of data also reduces the length of missing data in case of problem, since the dysfunction of the apparatus would show up instantly in the bad or missing data. It also allows using the inner sensors of the smartphone.

Important: Most USB OTG hub have a CHARGE mode and an OTG. To be able to charge the smartphone while logging, you must position the hub to CHARGE mode.

Smartphone inner sensors

It is not very common knowledge that smartphones sensors raw values can be accessed and logged. There are a few free applications that allow to do that. Some common smartphone sensors include:

- Geolocation (Latitude, Longitude, Altitude);
- Accelerometer/Gravity;
- Velocity;
- Orientation;
- Proximity;
- Luminosity;
- Electromagnetic field;
- Battery temperature and charge;
- Sound level.

The accuracy of those sensors will vary for each smartphone, but the specifications are available through the manufacturer. These internal sensors can be a good complement to the RDL sensors. For example, it might be extremely useful to combine GPS data with temperature, if you are monitoring the temperature of car component as you drive.

In practice, you will get one text file for the RDL and one text file for the inner sensors, because these are produced by two different apps. Both files are then synchronized to your Cloud. They can later be merged into a single spreadsheet.

There are some limitations, however. First, the timestamps will not be synchronized, which will cause some discrepancy when you merge the datasets. Second, this will increase the smartphone power consumption.

Android app example: “AndroSensor” by Androsensor
Apple app example: “Physics Toolbox Sensor Suite” by Christian Vieyra.

Not all apps can log the data they display (e.g., Sensors Multitool” by weRed Software). Not all sensors can be logged by all apps. For example, Sensors Multitool is unable to log battery temperature (bug).

Specific notes about the Serial Terminal App 1.44 by Kai-Moriche

We recommend using at least 50 kB buffer (Settings in the App Menu). A smaller buffer will cause some issues (i.e., overflow, lag, gibberish symbols, crash) especially when a larger number of sensors are active.

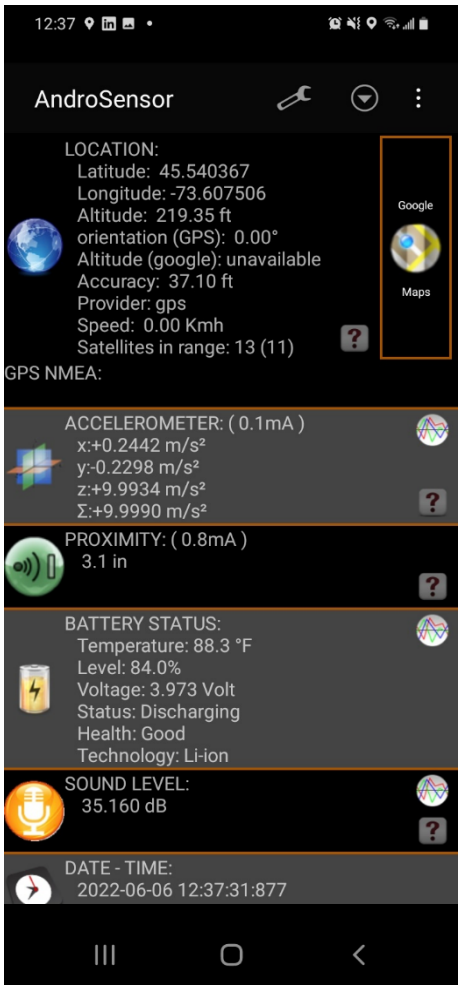


Figure 25 Screenshot sample of the AndroSensor app for Android Samsung A10 (not all sensors are visible)

Choosing a smartphone

For RDL operation, we recommend the use of Android smartphones. Apple products could also be used, although we have not made any test. The old Samsung S3 (2013) has proven to be a suitable candidate for smartphone operation. It has been sold in 40 million copies worldwide, hence it is available for cheap on the second-hand market (C$75), and it has a higher number of inner sensors than more recent phones. One downside is that it is not officially supported any more by Android, and so the security, support and compatibility levels are decreasing day by day. Original batteries have usually expired by now, but replacement batteries from third-parties are available on e-commerce website for around C$15. The battery can be installed in a few minutes, since batteries were not glued back in earlier smartphone designs.

Figure 26 Two units of the Samsung S4 smartphone, with a replacement battery

This is a great way to reuse electronics that are otherwise mostly obsolete. Users with a higher budget can use entry-level brand-new phones, such as the Samsung A10 or A03S (approximately $C200). When choosing a smartphone, you should try to select one with minimal power usage. Older smartphones are usually less power hungry nominally. More recent smartphones are usually more power hungry, due to ever increasing performance and larger screens. However, recent smartphone also benefits from more efficient low-power modes. This will increase the duration of the system power supply (internal or external) and reduce the size of the required solar panel, if applicable.

If you are unsure about your smartphone OTG abilities, apps such as “USB OTG Checker” can be used to verify that a phone is compatible with OTG communication. The recent Note 7 smartphone from Chinese manufacture Cubot, for example, does not have OTG abilities.

Figure 27 Example of a 5V-regulated USB solar panel connected to a RDL-smartphone setup

Power

The smartphone has a USB connection - usually type micro or mini. Therefore, they can provide a regulated 5V signal just like computers. When a smartphone is connected to a computer, it usually becomes a SLAVE to the computer HOST. This means that the computer leads the communication. However, an increasing number of smartphone models are able to act as a host. This allows smartphones to use a mouse and keyboard, for example. Detailed documentation is widely available on the internet to describe the variants of the SLAVE-HOST paradigm.

Figure 28 Example of OTG use
(Photo by Hans Haase - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=40618659)

Hubs are a variant of the OTG cables. OTG hubs usually have two-modes of operation: CHARGE and ON-THE-GO. The Charge mode is intended to charge the phone with the hub. The On-The-Go mode is intended to communicate with the USB device (e.g., mouse). In that mode, the phone is not being charged. A reduced set of phones have what is called a SIMULTANEOUS mode. The Samsung S3 is one of those. It means that when the hub is in CHARGE mode, the phone is still able to communicate. This is the most desirable mode of operation, since it allows indefinite operation of the RDL.

USB hubs are usually limited to 2 amps or less, so be careful not to use them with charging blocks that can deliver more 2 amps. Failure to respect the manufacturers indication on that regard can result in equipment melt-down and fire hazard. Always verify the specifications of the charger before using it with a USB hub.

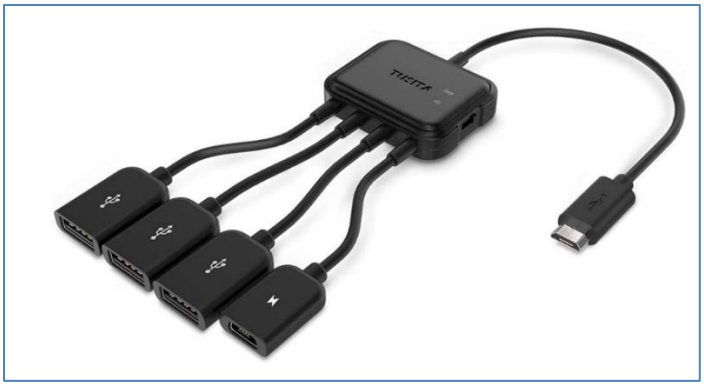


Figure 29 Example of a compatible OTG hub sold by TUSITA

It is important to note that in this simultaneous mode, the smartphone does not supply energy to the slave. The power must come from the hub. This means that a constant supply of energy must be provided in order to create a continuous measurement. This proscribes the use of solar-panel-only systems. It requires either a charger wall adapter, a large battery or a solar-panel-battery system (Figure 2).

Data Storage

Data storage is an important consideration in the design and operation of any data logger system. An independent storage (i.e., internal for the RDL) increases the complexity and cost of the system. Jericho's choice of smartphone reduces the complexity and cost of the RDL, as well as having important synergies with the communication aspect (next section). Internal data storage of the 2012 Samsung S3 is 8 GB, which far exceeds the needs of most data logging activities.

Communication

Communication between the RDL and the smartphone is done through serial communication, a standard established for decades.

There are a few ways to retrieve the data:

- **USB upload**: It is possible to just get the data at the end of the experiment. That means recovering the system and connecting to a computer. You can then access the internal memory of the phone and copy your data files to your computer. The user needs to know the folder path. The folder path is usually indicated within the options of the serial monitor app.
- **Manual email**: It is possible to occasionally reach your system, and send the data file by email. The user needs to know the folder path. The folder path is usually indicated within the options of the serial monitor app.
- **Cloud sync**: You must subscribe to a Cloud service, then find a client app for your phone. This can be a problem for very old phones like the Samsung S3. The client needs not only to allow connecting to your account, but ideally, to automatically sync. This can be problematic for some

services. For example, Google Drive official Android client does not support Android 4 anymore. This means you won't be able to download it from the Play Store.

- For Dropbox, there is a functioning client but it does not have the auto sync function. This is why, a third-party, such as Dropsync is needed. Dropsync requires Android 5.0 (Samsung S4) and is free to use under 10MB files. However, if using a large number of sensors or a high measurement frequency, the data size can be significant. Data logging apps like Kai-Moriche do not split the file daily. This means that the file will grow indefinitely (>100MB). This causes two problems. One, the file will stop uploading after it reaches 10MB. This can be solved by buying the full license, which cost C16$ and has no size limit. Second, with the one-file strategy, you will keep re-uploading the complete data set. This is not a huge problem if you are using Wi-Fi, but it will be problematic when using the cellular network, as it might require you to buy a larger data plan. An example of minimal service plan available in Canada is Lucky Mobile. You can buy a SIM card for around $C10 and a 500 Mb data plan for $C15. This is a prepaid plan, that requires no contract and the data is valid for one month. You can replenish as needed. Other companies offer similar plans. It can be more affordable in other parts of the world, such as Europe or India. The overall cost of a fully remote system is presently estimated to be under $C500.

Contrarily to computers, it is not possible at this stage to control more than one RDL device per smartphone. This is because the Serial Monitor apps are designed to talk to one serial device at a time, and Android does not allow opening two instances of the same app.

RTC TIMESTAMP ADJUSTMENT

To precisely synchronize your Real-Time-Clock (RTC) timestamp with the official time, execute the following procedure:

1. Make sure that your computer clock is synchronized with the official time of your area.
2. Remove the battery in the battery holder (if any).
3. Compile and upload the main code to the RDL by using the UPLOAD button in the Arduino IDE interface.
4. Put in the battery immediately after the upload has finished (see the message at the bottom of the IDE screen).
5. Push the reset button immediately.
6. Time will be set. You might see a warning message if your serial monitor was already opened.
7. Timestamp error is small (20 seconds or less) due to the delay between the start of the compile time and the pressing of the reset button.

After that, any reset or upload will not affect the RTC timestamp until the battery is removed or dies (several years).

Make sure that the code is actually recompiled at the time of uploading, so that the code contains the current time, and note the time registered at the previous compile. To make sure of that, restart the Arduino IDE and change a line of code with a trivial change (like deactivating/activating sensors).

NOISE CONTROL

Electromagnetic (EM) noise control is an important part of signal measurements. The amount of noise present depends on several factors among which the prevalence of EM noise in the environment and the length of the sensor lead wire. Long lead wires – several meters - tend to act as an antenna and are likely to pick up strong EM noise even in low noise environment such as a residence.

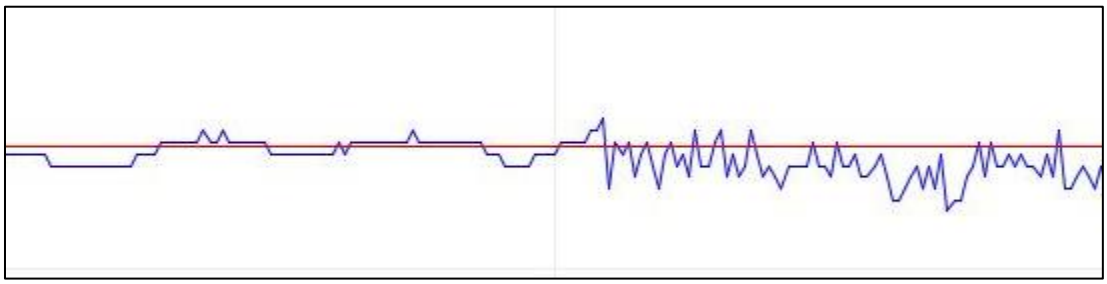


Figure 30 RDL Measurement example, with (left) and without (right) RC filter noise control

The RDL has two distinct mechanisms by which it lowers the prevalence of EM noise. Both mechanisms are optional, independent from one another and can be combined.

1) RC filters

   A low-pass filter of type Resistor-Capacitor is present in the circuit.

   a. One RC filter is available for each of the 16 resistance channels. This allows individual filtering. The capacitor type and value can be individualized. It is connected to a screw terminal in parallel with the sensor. It is not necessary to add a distinct resistor, as the resistive sensor acts provides the resistance. The filter is optional and can be left empty, when EM noise is not a concern. As an example, for a lead wire of 10 meters, Jericho has had good results with a capacitor of 0.1 uF.

   b. One global RC filter is available at the analog input. This allows to filter all channels with a single filter. The use of this filter is indicated when all filters need the same amount of filtering. Two screw terminals are available. This allows the user to combine two capacitors to obtain the desired performance. The filter is also optional and can be left empty, when EM noise is not a concern.

      With both techniques, a reading delay (e.g. 100 ms) might have to be implemented. Since the multiplexer only polls each channel for a limited amount time, the condensator needs time to equilibrate each time. Without an RC filter, the multiplexer can poll each channel for a minimal amount of time (i.e. 15 ms), but with an RC filter, some time is needed for the condensator to reach equilibrium. Without a proper delay (e.g. 100 ms) which varies for each case, the ADC might read a lower voltage value than without the RC filter, a significant source of error.

2) Data averaging
   Noise filter can also be treated numerically by averaging numerous sample (e.g., 10 samples) for each channel. This is simpler to implement than physical filtering: it consists of modifying the value for the parameter 'NUMSAMPLES' in the source code. For maximum speed, use value of 1. For heavy filtering, you can use values of 5 to 20. Since the ADC takes 15 ms per reading, a value of 20 would delay by at least 300 ms per channel. For 16 channels, that would mean a delay of 4800 ms, without accounting for the processing time (i.e., arithmetic averaging). This would limit your ability to gather data at a faster interval than 5s.

DIGITAL SENSORS

The RDL is able to communicate with digital sensors who use the I2C protocol. Since all sensors use the same data line, each sensor must use a different I2C address. This implies that two units of the same sensors cannot be used simultaneously, as they will both "talk" at the same time. You should also verify that each added sensor does not have an address that is already used. For example, the I2C address of a DS1307 RTC clock is 0x68. This address cannot be used by the sensors. The AM2301b temperature and humidity sensor has the 0x38 address.

HOW TO RECALIBRATE YOUR THERMISTORS (UPCOMING)

Section to be written in the near future.

HOW TO USE THE PRODUCT ENCLOSURE

The RDL Starter Kit is provided with a basic enclosure. The enclosure consists of a plastic food container with a snap lid (see figure below).

There is some stigma around the use of plastic food containers as PCB enclosures, as it can appear as "non-professional". However, this type of item is a great choice for several reasons:

1) Weatherproof, due to its intended purpose to contain liquid food.
2) Extremely affordable component (~$3-4), which helps keep our product price low;
3) Can be substituted by another similar food container if it breaks;
4) Daily tested by thousands of people;
5) Large enough to contain the whole RDL system;
6) Does not conduct electricity;
7) Temperature resistant (freeze, hot liquids)
8) Lightweight (which reduces shipping cost and carbon footprint);
9) Transparent (useful for some sensors or seeing the power ON light) but can be painted.
10) Can be drilled easily to install the PCB standoffs and cable gland(s);
11) Meets the mechanical requirements of most applications;
12) Recyclable (polypropylene (PP) food containers usually have the recycle logo)

Cable gland

To reduce cost, a single cable gland is used for all the wires that must go through the enclosure. This is a non-conventional use of cable glands, as they are usually intended for a single cable and provide a very tight seal. To reduce the infiltration of dust, humidity, insects, etc. inside the enclosure, a little piece of foam can be used to seal the space around the wires. It is also recommended to position the cable gland downwards, when possible, as this will reduce the inflow of water or rain into the enclosure. In applications requiring a better waterproofness, a small amount of urethane can be applied to seal the exit. This will however make it permanent and can be difficult to remove/clean.

Desiccant

To reduce and stabilize the humidity level inside the enclosure, always include a desiccant bag (silicone gel) inside the enclosure for field use. This reduces the likelihood of water condensation on the PCB, which can cause misreading and malfunctions.

PCB standoff screws

Four 3.5 mm nylon standoff screws are used to hold the PCB to the enclosure lid. These plastic screws have low mechanical properties and should be tightened lightly to avoid breaking them. The nylon screws could be replaced by stainless screws if needed. The PCB screws also constitute a possible entry for water. Thin rubber gasket or silicone could be applied to improve the impermeability. Positioning the enclosure with the lid facing down is another way to reduce susceptibility to rain entry.

Outdoor use

The use of the RDL outdoors often requires a weatherproof enclosure to protect the RDL and the smartphone from the elements. Obviously, the food container has no official water-resistant certification like NEMA-3 or IP6, but it aims to be weatherproof. We stress-test this enclosure in a regular dishwasher. In regular cycles (60°C, 60 minutes), less than 30 mL should enter the bottom of the enclosure. Insulation can be added inside the enclosure to help tolerate extreme temperatures (e.g., cold depleted smartphone battery when logging in the winter).

OPEN-SOURCE LICENSES

Jericho Laboratory operates the RDL product with the following open-source licenses:
SOFTWARE: GPL 2.0
HARDWARE: MIT
WEBSITE, DOCUMENTS: Creative Commons CC-BY-NC-SA (Non-commercial use only)

For more information about the user permissions and responsibilities associated with each license, please refer to our website or contact us via email at info@jericholab.com.

WARRANTY

Our manufacturer warranty includes a 60-day full refund excluding shipping cost. We also provide a 1-year limited warranty on defects. Any misuse, physical damage or modification of the device by the user will void the warranty. This includes but is not limited to: adding or removing components, soldering, bending, cutting, sanding, painting. Made In Canada with International Components.

CONTACT

For more information, tutorials, FAQ, software updates, accessories and replacement parts, please visit us at jericholab.com. We also appreciate you leaving comments and suggestions.

ABOUT JERICHO LAB

Jericho is a company based in Montreal, Canada. Our mission is to improve access to scientific equipment by providing provide low-cost high-quality products to students, scientists and engineers. The Rose of Jericho (*Anastatica hierochuntica*) is a desert plant that can survive years of complete desiccation (absence of water).

ABOUT ARDUINO

The RDL is compatible with the Arduino language, which is based on the 'Processing 3' language, which itself is similar to the C language. Most of the C/C++ core language can be used with the RDL device and the Arduino IDE software. You can create classes, use inheritance, composition, and other functionalities, but STL and exceptions cannot be used. Arduino first appeared around 2005 and is now used worldwide by over a million users. Jericho is not endorsed or linked in any sense with the Arduino company. Arduino clones are produced by hundreds of manufacturers around the world, with different qualities.


Please do not throw away this product to garbage or recycle bin as it can contain toxic substances and/or heavy metals. Please send your Jericho product to your local electronic waste drop-off site so that it can be disposed of properly.

FREQUENTLY ASKED QUESTIONS (FAQ)

- What is a thermistor?

A thermistor is a tiny passive device which has an electrical resistance that varies with the temperature of its material. It is a word play with the words ‘thermal’ and ‘resistor’.

- Why do I have to connect the thermistor to a specific channel?

Each thermistor has unique parameters called Steinhart-Hart coefficients. Each channel is associated with a specific set of coefficients. Therefore, switching the thermistors would provide inaccurate temperature readings.

- ***Can I use the RDL to do something else***?

Yes, you can modify the product, software and hardware to add or remove capabilities. However, any physical modification of the device will void the warranty.

- ***I dropped the RDL in the water, what should I do?***

Water will create short-circuits that will render the device unusable temporarily. Put it in dry uncooked rice with closed container for 24 to 48h to remove humidity, like you would do for a smartphone. It should work again.

- **What other software than Arduino Software can I use to control the RDL device?**

The IDE software provided by Arduino is specifically designed for Arduino and Arduino-compatible devices and provides an easy interface to beginners and advanced users. However, some users might prefer to use more professional programming tools. A free option that is often recommended is the software Atom, combined with the Platformio add-on.

- I broke some component. How do I find replacement parts?

You can consult the Bill of Material (BOM) to find the exact part number and order a new part online. If you believe this is due to a manufacturing defect and you are still covered by your warranty, contact us by email.

- Should I tin the wire endings to be used with screw terminals?

No, NASA procedures for example require to NOT thin the wires due to different thermal expansion coefficients of the wire and solder.

TROUBLESHOOTING

- **Error: “An error occurred while uploading the sketch. (avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x31)**
    - This can happen when the RDL was busy sending sensor data when the PC tried to communicate.
    - Try reuploading the program.
    - Try disconnecting the USB cable and reconnecting, then reuploading.
    - Try another USB port of the PC.
    - Try to restart the program.
- **Error: “Board at COM4 is unavailable.”**
    - The selected serial port is not detecting the RDL/Arduino.
    - Verify that the USB cable is connected to both the PC and the RDL.
    - Verify that the corresponding port is selected in the Arduino menu.
- **The printed output of a channel makes no sense. Example, the temperature of the channel C2 is -273.15°C.**
    - Negative, random-looking values are usually associated with empty or opened channels (false contact, broken probe). Verify that there really is a sensor connected to the channel and that the connection is solid.
    - *Verify that the channel is properly set up. If the channel is set up as a photoresistor (P) but that physically the sensor is a thermistor, the calculated value will make no physical sense.*
- **Error: “Variable X was not declared in this scope”**
    - Two very common reasons for this error is 1) a function was indeed declared at the wrong place 2) a missing bracket or semi-colon has created a complete fuckup of the entire code. It can be hard to find the exact spot. Undo the latest change or go back to your previous working saved version.

REFERENCES

Arduino Uno Rev3, Digikey, https://media.digikey.com/pdf/Data%20Sheets/Arduino%20PDFs/A000066_Web.pdf

ATmega 328p datasheet: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Processing language official website, Processing.org

Free Arduino tutorials, https://www.tutorialspoint.com/arduino/

Yunus A. Çengel. 2007. *Heat & Mass Transfer: A Practical Approach*. McGraw-Hill Education (India) Pvt Limited.

Learn: Basic knowledge about principles and techniques behind the Arduino ecosystem. https://docs.arduino.cc/learn