OhmPi

Release v2024

OhmPi Team

Jun 26, 2024

INTRODUCTION

Summary

Release
v2024

Date
Jun 26, 2024

Date start
July 2016

Authors
Rémi CLEMENT, Nicolas FORQUET, Yannick FARGIER, Vivien DUBOIS, Hélène GUYARD, Olivier KAUFMANN, Guillaume BLANCHY, Arnaud WATLET

Target
users, researchers and developers

Status
some mature, some in progress

This documentation presents the development of a low-cost, open hardware resistivity meter to provide the scientific community with a robust and flexible tool for small-scale experiments. Called OhmPi, this resistivity meter features current injection and measurement functions associated with a multiplexer that allows performing automatic measurements.

CHAPTER

ONE

CONTENTS

1.1 OhmPi project

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OhmPi. The OhmPi team cannot be held responsible if the equipment does not work after assembly. You may redistribute and modify this documentation and make products using it under the terms of the CERN-OHL-P v2. This documentation is distributed WITHOUT ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING OF MERCHANTABILITY, SATISFACTORY QUALITY AND FITNESS FOR A PARTICULAR PURPOSE. Please see the CERN-OHL-P v2 for applicable conditions.

1.1.1 Introduction

The OhmPi project was initiated to develop an open-source, open-hardware resistivity meter, particularly designed for the research community, education, and humanitarian or not-for-profit organisations. In the last decade, geoelectrical monitoring has become a popular tool to study and monitor physical processes in hydrology. As novel applications emerge, the need for more accessible and flexible acquisition systems grows in the research community. The flexibility and adaptability of OhmPi makes it particularly suited to develop novel acquisition strategies or design innovative small-scale monitoring experiments.

Note: Anyone who wants to get involved is welcome to join the OhmPi project!

1.1.2 Authors

Rémi CLEMENT, Vivien DUBOIS, Nicolas Forquet, INRAE, REVERSAAL, Villeurbanne, France
Olivier KAUFMANN, Arnaud WATLET, Université de Mons, Mons, Belgium
Yannick FARGIER, GERS-RRO, Univ Gustave Eiffel, IFSTTAR, Lyon, France
Hélène GUYARD, IGE Grenoble, Université Grenoble Alpes, Grenoble, France
Guillaume BLANCHY, ULiège, Liège Belgium

1.1.3 Partners

1.1.4 Citing OhmPi

Rémi Clement, Yannick Fargier, Vivien Dubois, Julien Gance, Emile Gros, et al.. OhmPi: An open source data logger for dedicated applications of electrical resistivity imaging at the small and laboratory scale. HardwareX, Elsevier, 2020, 8, 24 p. ff10.1016/j.ohx.2020.e00122ff.

PDF version of the documentation (note the latest version is always the html one)

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

1.2 Hardware

This section contains the documentation needed to build an OhmPi.

The OhmPi is composed of different modules:

- a measurements board (`mb`): that measures the current and voltage and modulates the injected current
- 0, 1, … or n multiplexer boards (`mux`): that address different electrodes
- a power supply (`pwr`): either a 12V battery or a more advanced power supply where we can control the voltage/current

- a general controller (`ctrl`): to control the measurement board, multiplexer boards and power supply (=raspberrypi)

These module exists in different versions and can be combined using a configuration file. You can then upgrade your measurement board or power supply for specific applications.

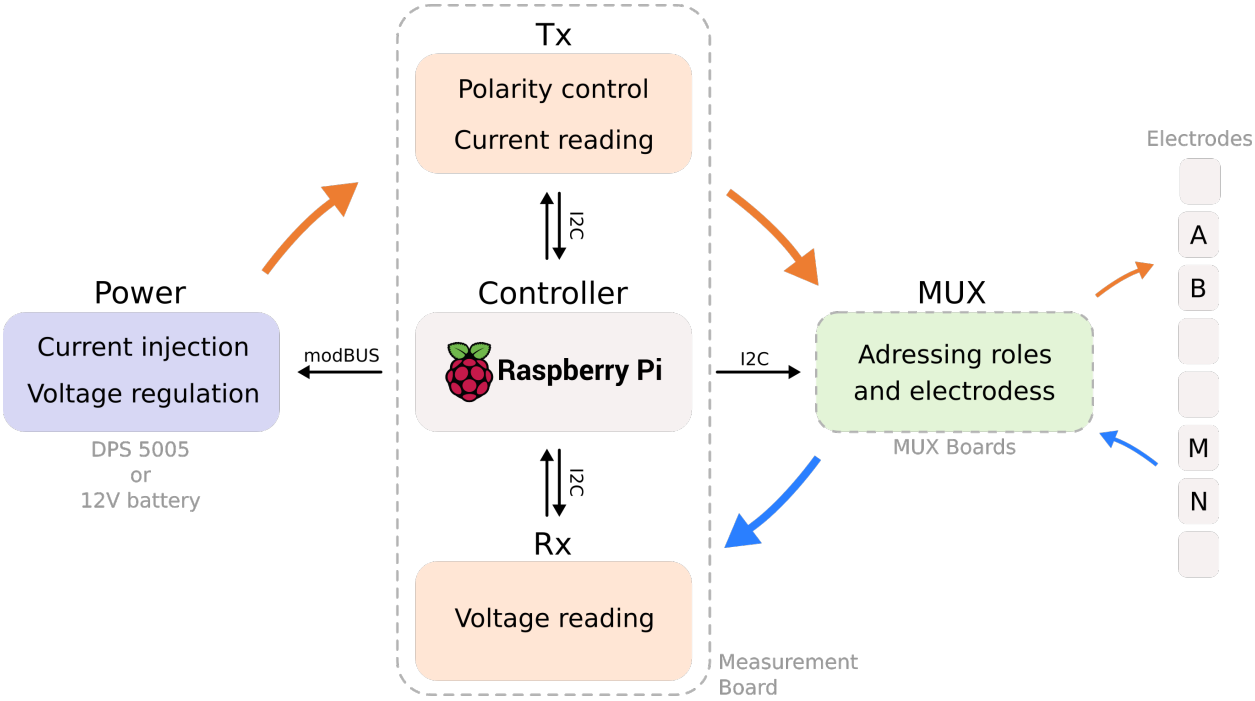


Fig. 1: OhmPi hardware flowchart.

1.2.1 OhmPi electronic design

Measurement board

The measurement board integrates different electronic components to

- measure the voltage at MN
- measure the current injected at AB
- switch the polarity of AB (to make different half-cycles/stack)

Some general explanation about the components is given below to help you understand the general electronics of the OhmPi. For more details, we redirect the reader to the datasheet of each component.

Measuring voltage

Voltage measurement is typically done through an **ADC (Analog to Digital Converter)**. In the OhmPi, the component *ADS1115*, a 16-bit ADC is used. The *ADS1115* is also equipped with a programmable gain control (PGA), which means it can scale up the measured voltage by a factor before digitizing it. Its gain can vary between 2/3 and 16. With a gain of 1, this component can measure voltages between 0 and 5 V with a precision of 5 / (2 ^ 16) = 0.076 mV.

However, we often measure voltage beyond 5 V. So to measure larger voltage with our ADC, we need to divide the received voltage. In `mb_2023`, this is done using a **resistor divider bridge**. The voltage at MN is the distributed across two resistors placed in series according to their respective resistances. For instance, if we see 12V at MN and have two

resistors in series of 150 and 300 Ohms. We will measure 12 * 150 / (150 + 300) = 4 V on the first resistor and 12 * 300 / (150 + 300) = 8 V on the second. The 4 V can be measured by our ADC.

Another technique to reduce the voltage consists in using **operational amplifier (opamp)**. These devices have multiple applications and using a given configuration with a known resistance, can be used to scale down the voltage input. In addition, opamp are used in 'follow-up' mode to ensure a **high input impedance** of the MN part. Indeed, if current is leaking in the MN part while we measure, it will affect our measurement.

In the measurement board 2024 (`mb_2024`), an opamp is also used in differential mode to measure the difference in voltage between M and N (N is used as a 'floating ground'). This enables us to measure much higher voltage as long as the difference between M and N is not too large.

Measuring current

Current measurement is usually obtained by measuring the voltage through a very accurate resistance called a **shunt resistance**. In `mb_2023`, a shunt resistance of 2 Ohms is used. As this resistance has a tiny value, the voltage drop measured through it is also very small and need to be amplified before being measured by the ADC. This is done through the INA282 component (also an opamp).

In `mb_2024`, the current measurement is done via a "Click module" where the shunt and amplifier (INA equivalent) are already soldered.

Polarity control

Each half-cycle has a different polarity. Current is first injected from A to B then from B to A with or without an off-time between the two injections. This cycle of polarity is ensured using four **relays** (optical in v2023, mechanical in v2024) that open or close alternatively. The relays are controlled from the MCP23008 which is **GPIO expander**.

Communication

The ADC (ADS1115) and GPI expander (MCP23008) communicate with the controller (raspberrypi) via two wires (SDA and SCL) using the **I2C protocol**. This protocol makes use of two wires. One wire is used to send pulses from a clock (SCL) and the other one to transmit data (SDA). These wires must be pulled high using pull-up resistors (meaning at rest, there should be 5V between these wires and the ground).

Multiplexer

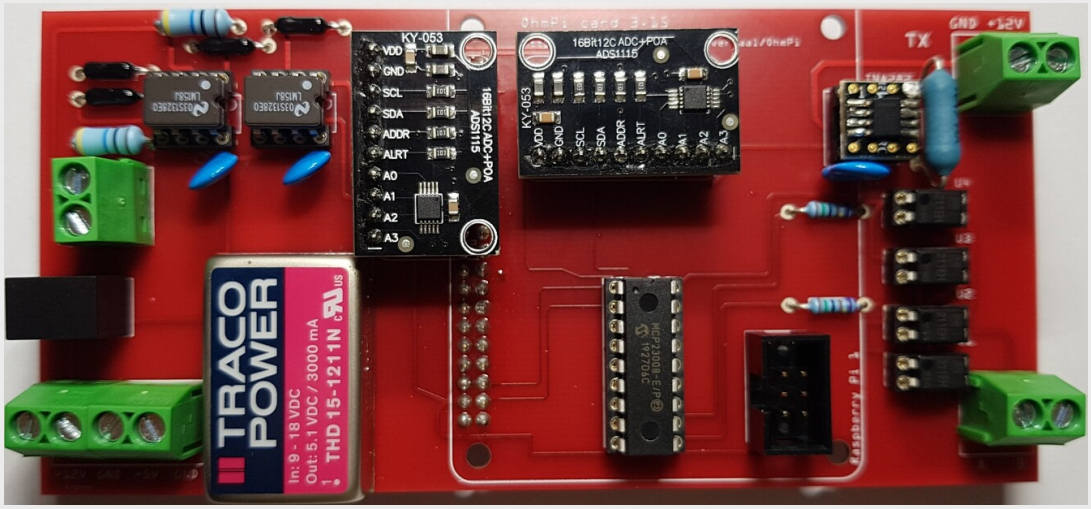
Multiplexer are used to address multiple electrodes. For this they use **relays** to create an electronic path between the electrode and the entry (A, B, M, or N) on the measurement board. The relays are usually controlled by a **GPIO expander** (MCP23017). Because too many GPIO expander cannot be addressed on the same I2C bus, we use a **I2C expander** (TCA9548A) which is itself connected via I2C to the controller (Rapsberry Pi).

1.2.2 Measurement board

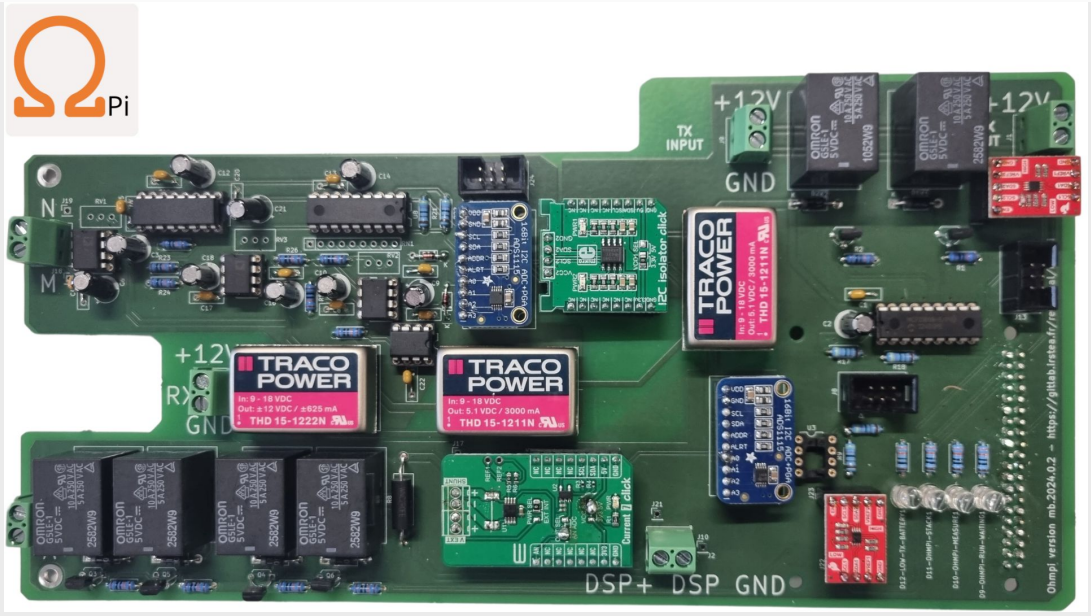
This section introduces the OhmPi measurement boards. Starting from this year, it has been possible to use any measurement board with the latest OhmPi code. Consequently, the OhmPi group provides a variety of board options tailored to your technical needs (e.g., laboratory measurement, field measurement), budget, and electronic skills.

The characteristics of each measurement board are described in the following table:

Recognize the version of the measurement board



Measurement board V2023.0.1



Measurement board V2024.0.2

Specifications

Parameters	v2023.0.1	v2024.0.2	Units
Vmn number of channels	1	1	•
Operating temperature	0 to 50	-25 to 50	°C
Max. permissible Vab	24	200	vdc
Power supply	12	12	vdc
Current with 2 ohms shunt resistor	0.11 to 40	0.11 to 500	mA
Min pulse duration	50	50	ms
Max pulse duration	15	15	second
Vmn input impedance	80	1000	MOhm
Vmn range	-/+ 5	-/+5	volt

Assemble measurement board (MB)

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

Build measurement board v2023

PART A Assembly of the measurement board

Required components

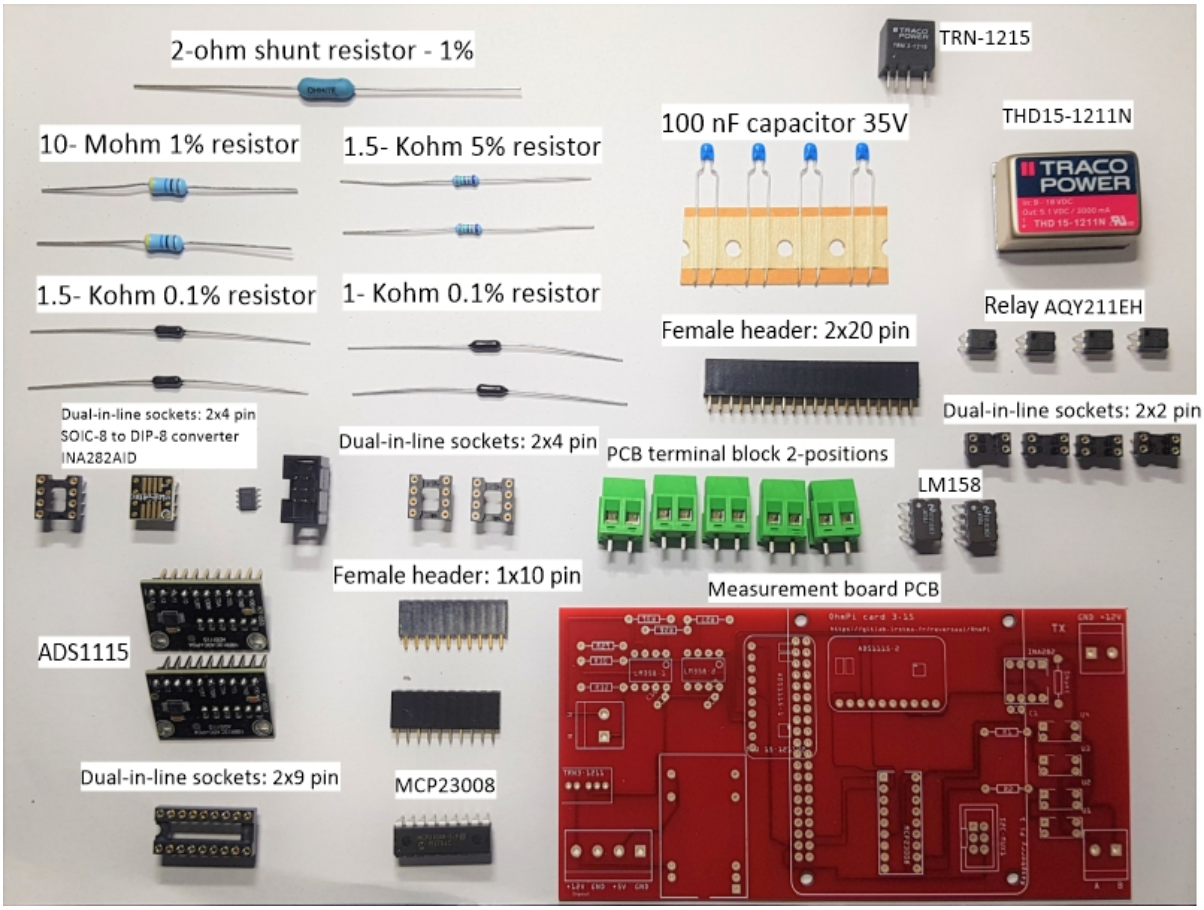

2-ohm shunt resistor - 1%
TRN-1215
100 nF capacitor 35V
THD15-1211N
10- Mohm 1% resistor
1.5- Kohm 5% resistor
1.5- Kohm 0.1% resistor
1- Kohm 0.1% resistor
Relay AQY211EH
Female header: 2x20 pin
Dual-in-line sockets: 2x4 pin
SOIC-8 to DIP-8 converter
INA282AID
Dual-in-line sockets: 2x4 pin
PCB terminal block 2-positions
Dual-in-line sockets: 2x2 pin
LM158
Female header: 1x10 pin
Measurement board PCB
ADS1115
Dual-in-line sockets: 2x9 pin
MCP23008

Table 1: List of components

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturers reference	Web reference
Raspberry Pi 4 Model B	1	58,75	58,75	Raspberry	Raspberry Pi 4 Model B	https://www.mouser.fr/ProductDetail/Seeed-Studio/102110421?qs=7MVldsJ5UaxeN3LYyh3sq3D%3D
LM158 AMP-o	2	14,5	58	Texas Instruments	LM358A	https://www.mouser.fr/ProductDetail/Texas-Instruments/LM158J?qs=X1J7HmVL2ZH8vpEfMl8%2FFQ%3D%3D
Printed circuit board	1	12	12	Asler	•	•
ADS11	2	11,9	23,8	Adafruit	1085	https://www.mouser.fr/ProductDetail/Adafruit/1085?qs=%2Fha2pyFaduhE%2FOGzuTWIQ9Iz5VjaqFO252Bg%3D%3D

continues on next page

Table 1 – continued from previous page

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturer s reference	Web reference
Capacitor 100nF 50Vdc 10% Ceramic	3	0,2	0,8	KEMET	C320C10	https://www.mouser.fr/ProductDetail/KEMET/C320C104K1R5TA7303?qs=c4UyoTs%2FLq1th4mcyOeTmA%3D%3D
Resistor 1 Kohm 0.5W +- 0.1%	2	1,3	2,6	TE Connectivity	H81K0B	https://www.mouser.fr/ProductDetail/TE-Connectivity-Holswort H81K0BYA?qs=%2Fha2pyFaduhUylh7Az% 2FmjFH2XjOUms6wZtUX 252BII%3D
Resistor 1.5 Kohms +- 0.1%	2	1,3	2,6	TE Connectivity	H81K5B	https://www.mouser.fr/ProductDetail/TE-Connectivity-Holswort H81K5BYA?qs=%2Fha2pyFadugy9tWham 252BX%2FM%3D

continues on next page

Table 1 – continued from previous page

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturer s reference	Web reference
Resistor 1.5 Kohms +- 5%	2	1,3	2,6	Vishay	CCF071K	https://www.mouser.fr/ProductDetail/Vishay-Dale/CCF071K50GKE36?qs=QKEOZdL6EQpA6LZRLQ3D%3D
Resistor 10 Mohms +-5%	2	0,762	1,524	VISHAY	CMF651	EK143 https://www.mouser.fr/ProductDetail/Vishay-Dale/CMF651M0000FKEK143?qs=CiayqK2gdcKzIA2LEVaLk3D%3D
2 ohm shunt resistor 1%	1	2,42	2,42	Ohmite	41F2R0E	https://www.mouser.fr/ProductDetail/Ohmite/41F2R0E?qs=IM6ToxQzGOAuEDprb19r3D%3D
Dual screw terminal (5.08-mm pitch)	5	0,648	3,24	CUI Devices	TB009-508-02BE	https://www.mouser.fr/ProductDetail/CUI-Devices/TB009-508-02BE?qs=vLWxofP3U2wCFk5uCkW3D%3D

continues on next page

Table 1 – continued from previous page

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturer s reference	Web reference
DC/DC converter 12 to 24V	1	15,58	31,16	TracoPower	TRN 3-1215	https://www.mouser.fr/ProductDetail/TRACO-Power/TRN-3-1215?qs=YCa%2FAAYMW02gqUicGQj0t%3D%3D
DIP Dual In Line Socket 2*4	3	0,72	2,16	Mill-Max	110-43-308-41-001000	https://www.mouser.fr/ProductDetail/Mill-Max/110-43-308-41-001000?qs=IGgAdOvCTsTu%2FqaUr8NArg%3D%3D&mgh=1&vip=1&gclid=EAIaIQobChMIn_TAxbCx8wIVQ5nVCh2QaD_BwE
AQY21	4	3,84	15,36	Panasonic Industrial Devices	AQY211	https://www.mouser.fr/ProductDetail/Panasonic-Industrial-Devic AQY211EH?qs=wKtUvITRialGIU8hcM7D%3D%3D

continues on next page

Table 1 – continued from previous page

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturer s reference	Web reference
DIP Dual In Line Socket 2*2	4	0,449	1,796	Preci-dip	110-83-304-41-001101	https://www.mouser.fr/ProductDetail/Preci-dip/110-83-304-41-001101?qs=%2Fha2pyFadujQKqx4wA2FMGNdxMCNv%2F33Nj0gBxRocuLUcYnpy3D%3D
MCP23	1	1,72	1,72	Adafruit	593	https://www.mouser.fr/ProductDetail/Adafruit/593?qs=sGAEpiMZZMsKEdP9slC3D
Header sets 1x10	2	2,12	4,24	Samtec	SSW-110-02-G-S	https://www.mouser.fr/ProductDetail/Samtec/SSW-110-02-G-S?qs=rU5fayqh%252BE0w1ORXZiBQpw%3D%3D

continues on next page

Table 1 – continued from previous page

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturer s reference	Web reference
SMT Breakout PCB for SOIC-8	1	2,5	2,5	Adafruit	1212	https://www.mouser.fr/ProductDetail/Adafruit/1212?qs=GURawfaeGuCAqqfvnVty
INA282	1	4,11	4,11	Texas Instruments	INA282A	https://www.mouser.fr/ProductDetail/Texas-Instruments/INA282AID?qs=Ze4%2FuFuz19ILFayZXOCfrA%
THD 15-1211N	1	39,72	39,72	TracoPower	THD 15-1211N	https://www.mouser.fr/ProductDetail/TRACO-Power/THD-15-1211N?qs=%2Fha2pyFadugpyEG4IDv

continues on next page

Table 1 – continued from previous page

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturer s reference	Web reference
DIP Dual In Line Socket 2*20	1	8,53	8,53	Samtec	SSQ-120-23-G-D	https://www.mouser.fr/ProductDetail/Samtec/SSQ-120-23-G-D?qs=rU5fayqh%252BE1BMVd%252BDZONqg%3D%3D
Pin strip no ejector	1	0,35	0,35	BLK electronic	1012055(	https://www.conrad.com/p/bkl-electronic-10120550-p searchTerm=741435& searchType=suggest& searchSuggest=product
Male Female spacer 2.5M HEXAC O-NALE	4	0,87	3,48	HARWIN	R25-3002002	https://www.mouser.fr/ProductDetail/Harwin/R25-3002002?qs=W0yvOO0ixfENUv0hsdC4 2FQ%3D%3D

continues on next page

Table 1 – continued from previous page

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturer s reference	Web reference
DIP Dual In Line Socket 2*9	1	1,86	1,86	Preci-dip	437-1108331	https://www.mouser.fr/ProductDetail/Preci-dip/110-83-318-41-001101?qs=FtMuP6KVi2TNQOezIAQ%2FPA%3D%3D

Description

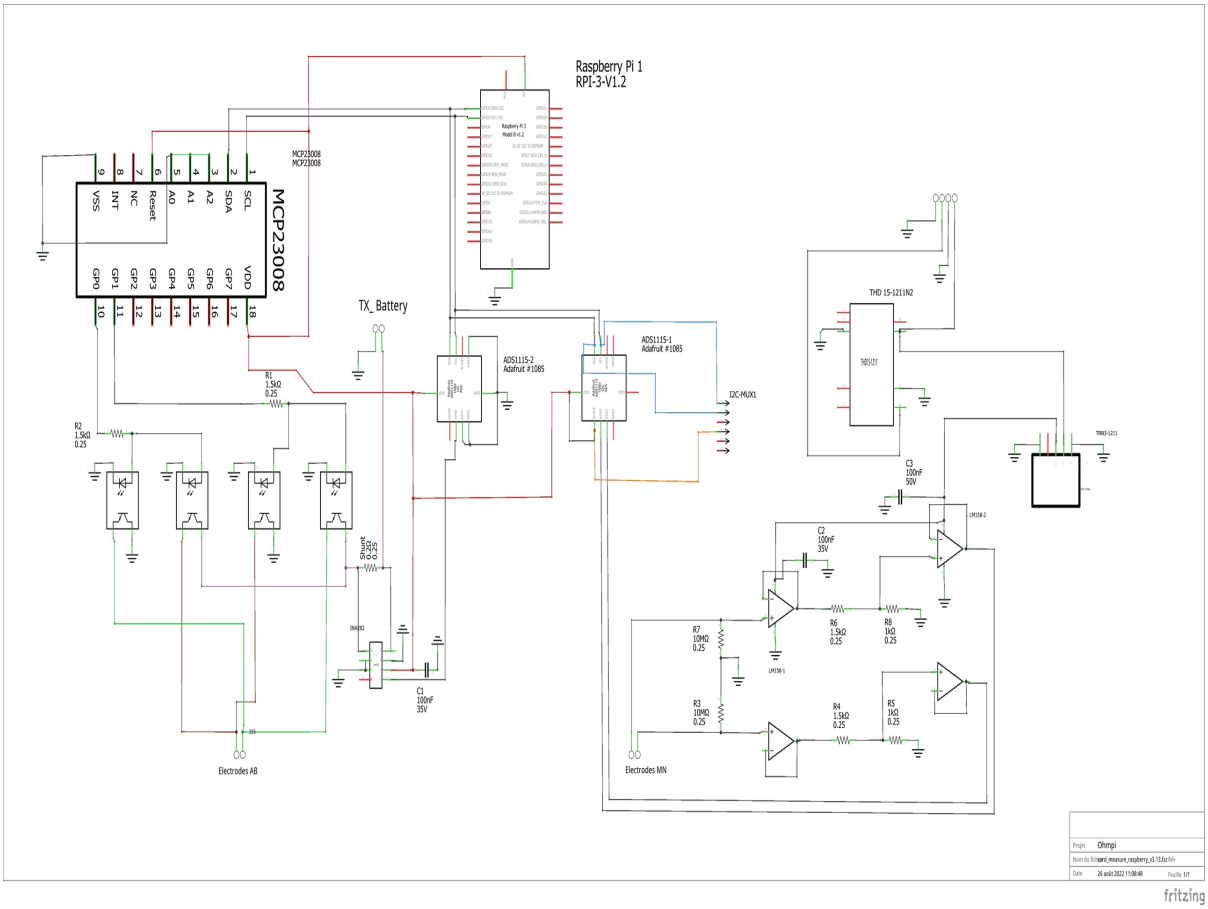
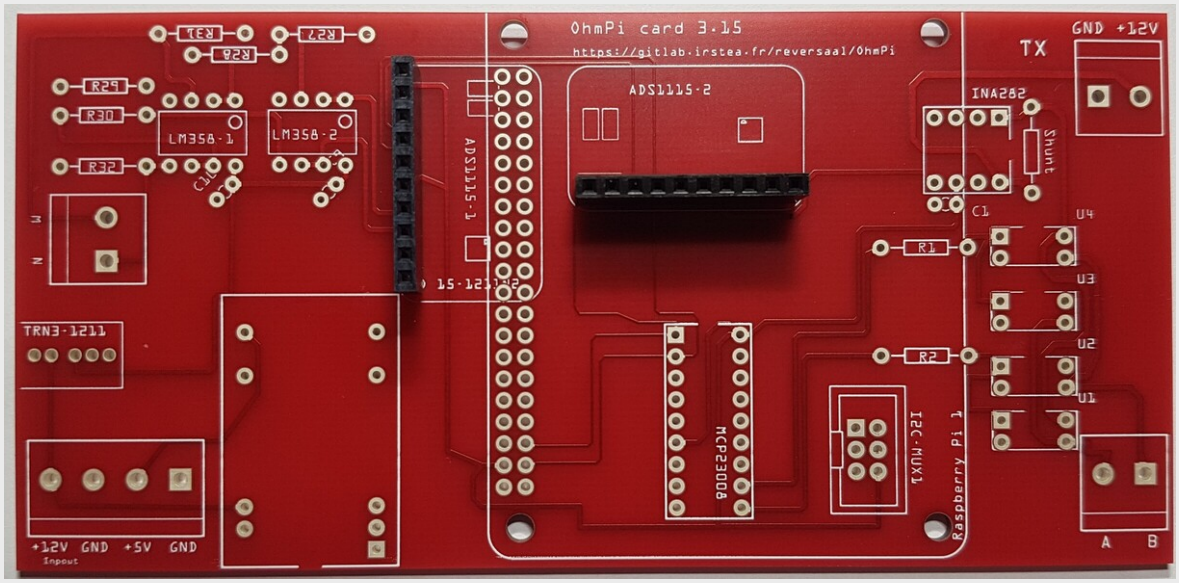


Figure shows the general schematics for the electronic measurement board developed. We have developed a complete

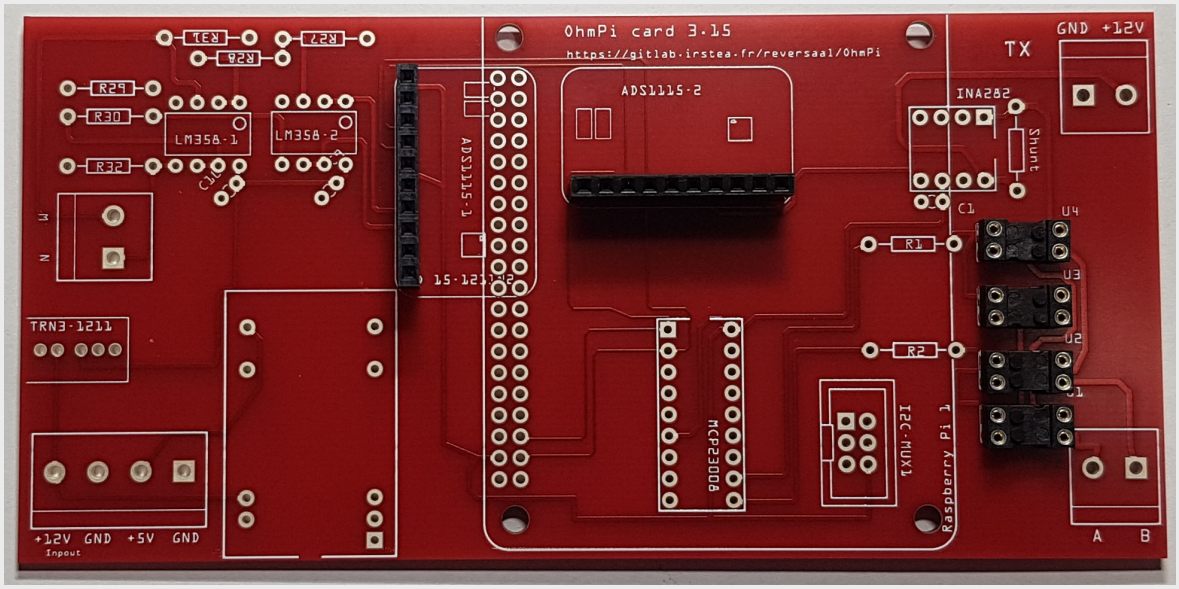
"plug and play" measurement board. To measure electrical resistivity with Raspberry Pi. two ADS1115 were used,one for the voltage measurement one for the current measurement, as proposed by Florsch [7]. The ADS1115 is a 16-bit ADC (Analog-to-Digital Converter), with an adaptable gain. The advantage of ADS1115 is that the input signal value could lie between - to + 6.114 V. For the current measurement we have directly integrated the INA282 component, which allows to realize precise current measurement around a shunt resistor. The assembly are described in the following steps:

1

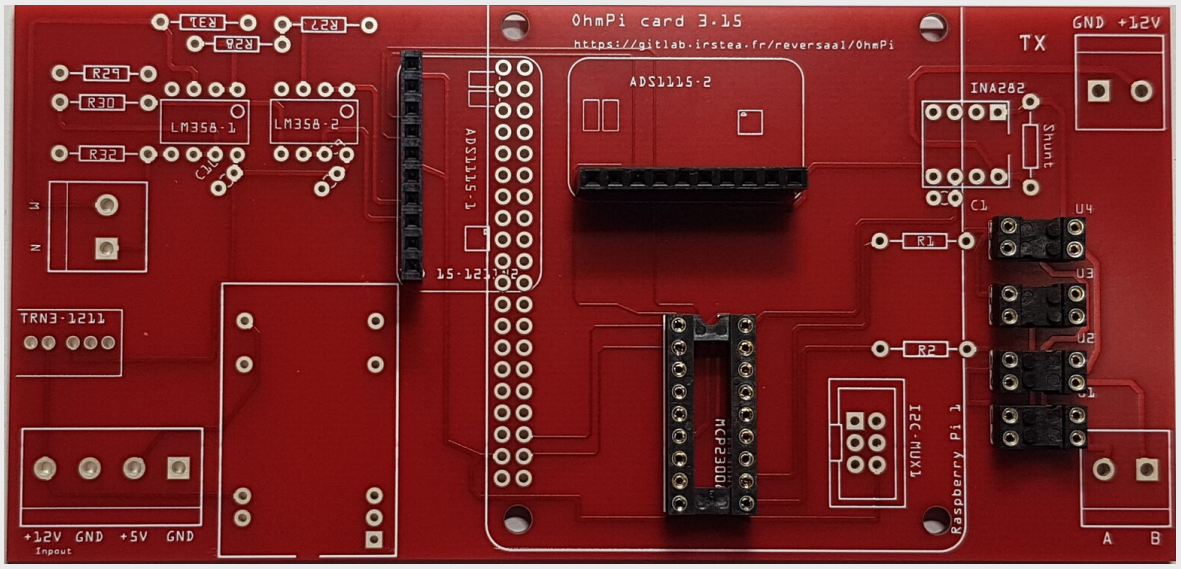
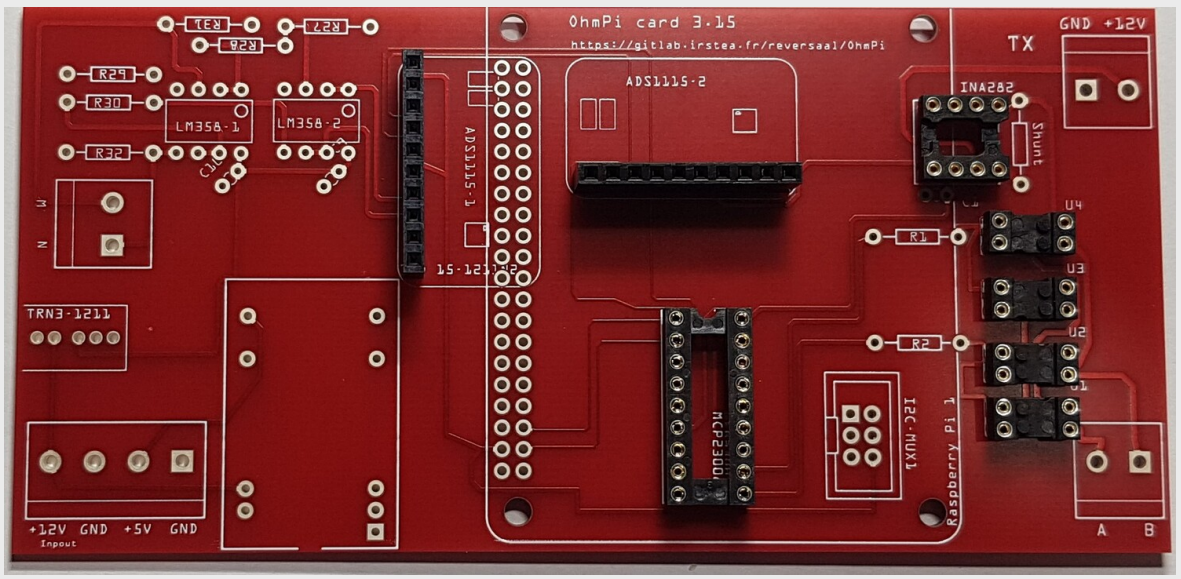


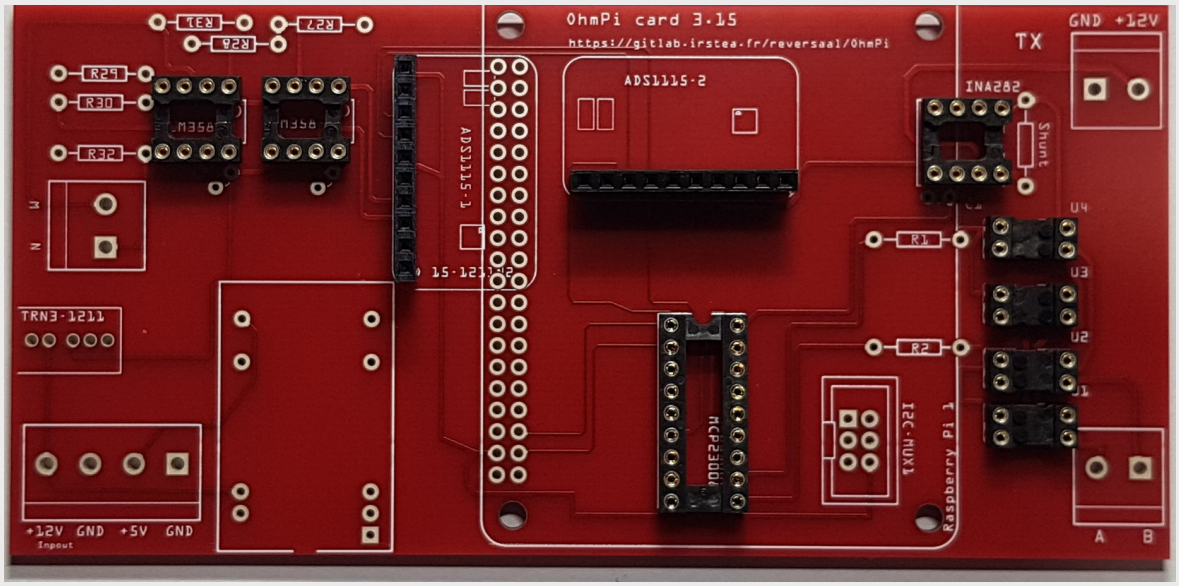
Installation of female header, 1 by 10 pins, for ADS1115

2

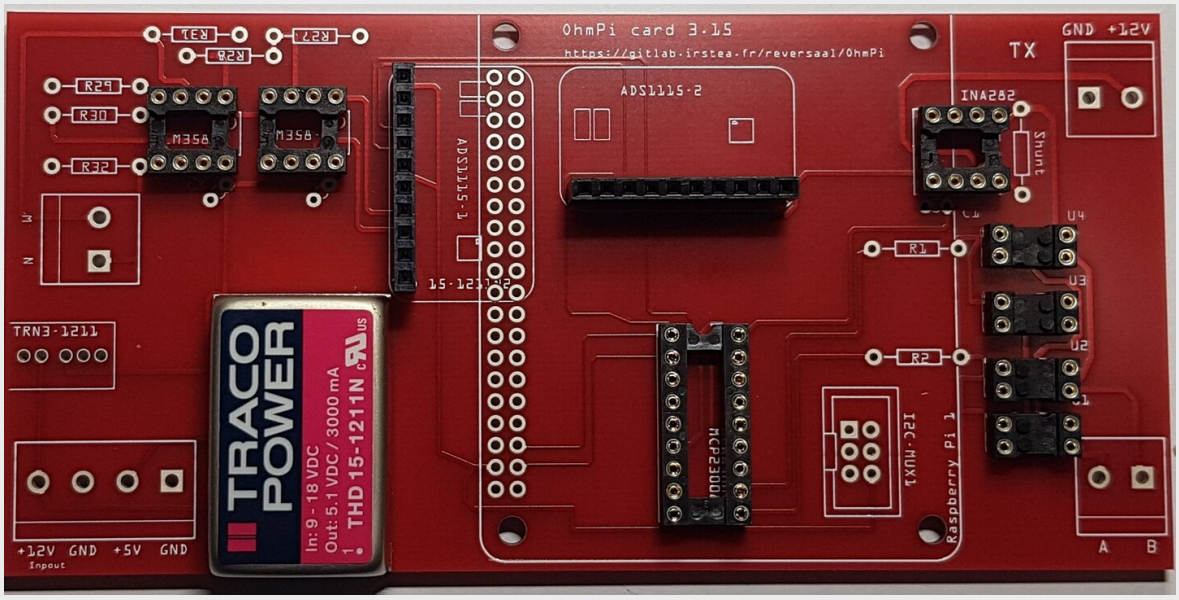


Soldering of 4 dual-in-line socket (2 by 2 pins) for optical relay, AQY211EH.

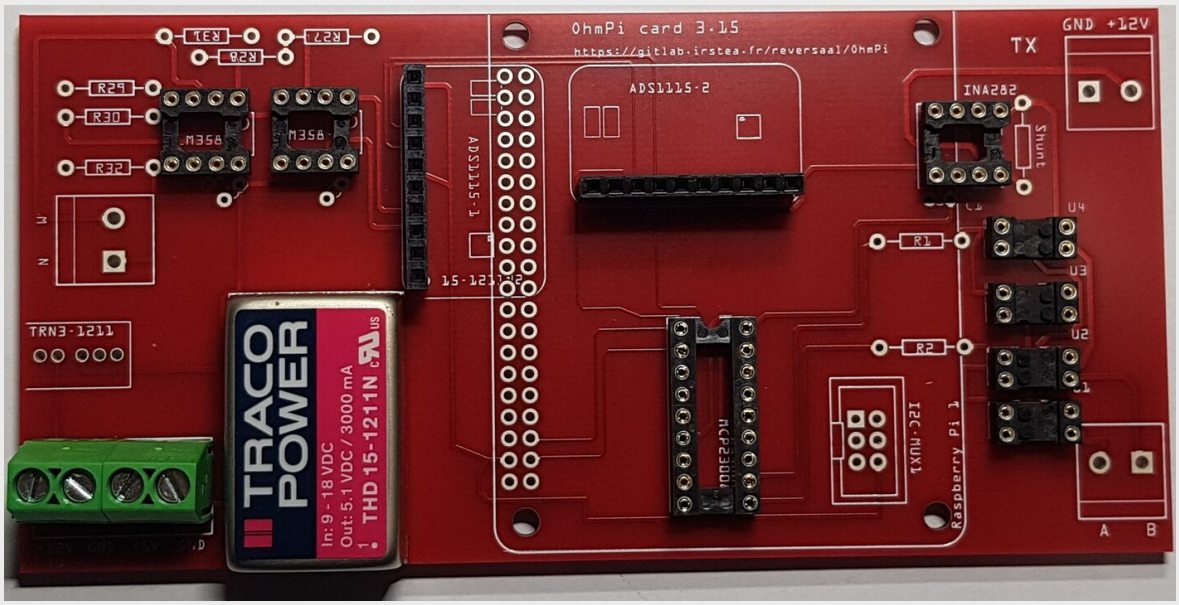
3	 Soldering of 1 dual-in-line socket (2 by 9 pins) for MCP23008.
4	 Soldering of 1 dual-in-line socket (2 by 4 pins)



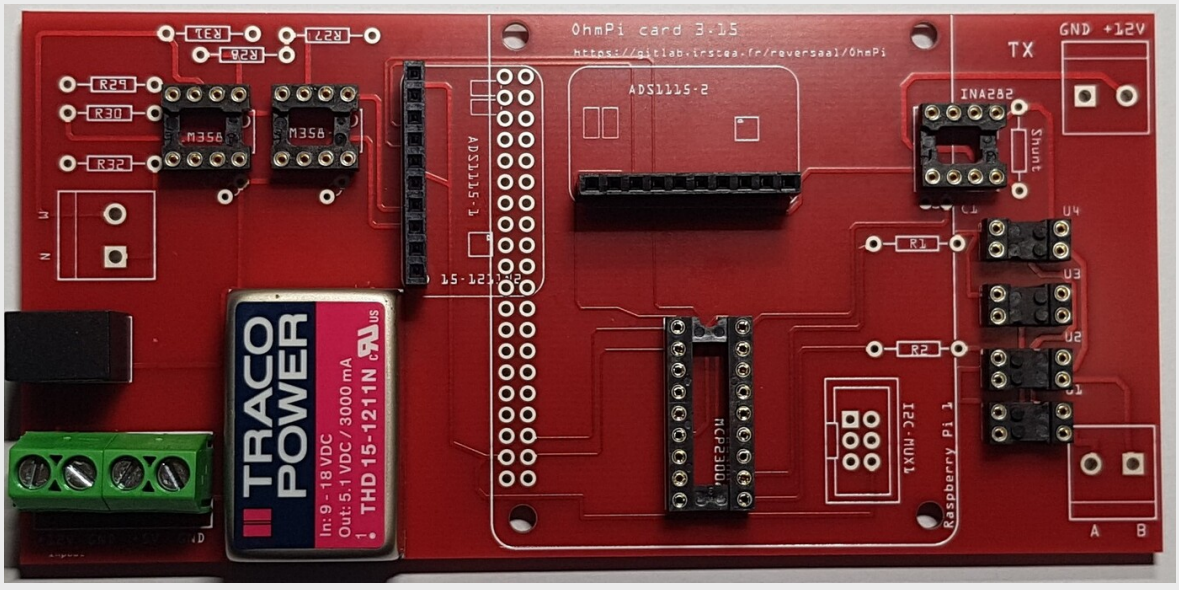
5 Soldering of 2 dual-in-line socket (2 by 4 pins)



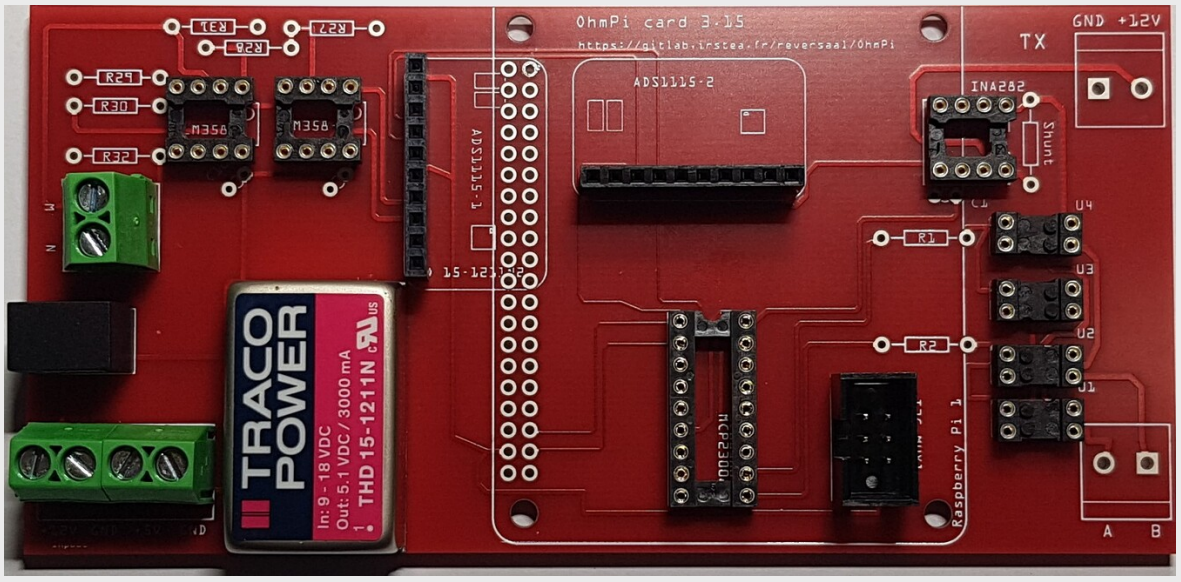
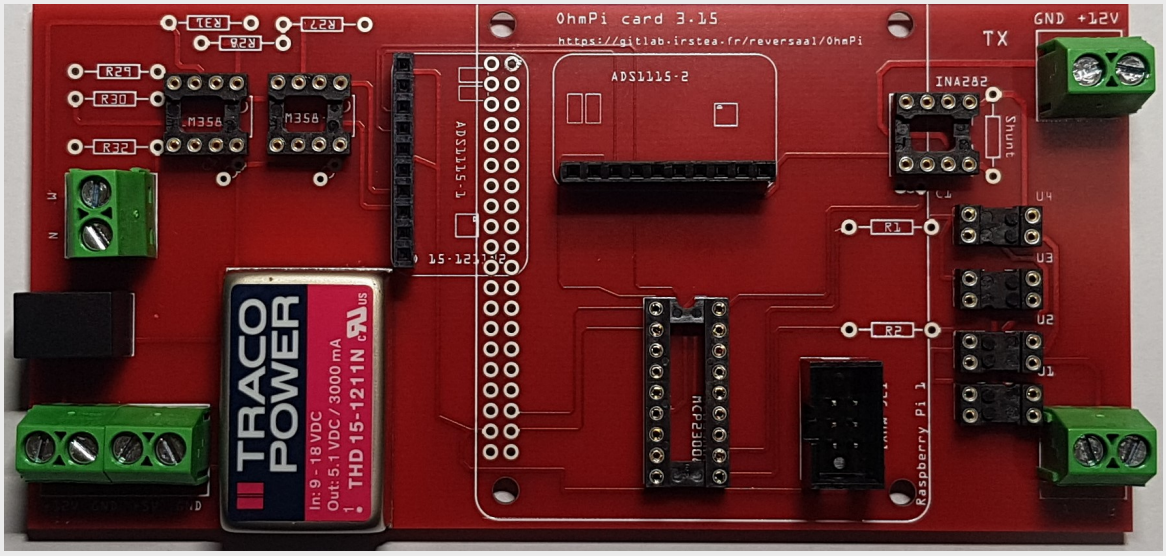
6 Traco Power Supply 12V to 5V, TDH15 - 1211N

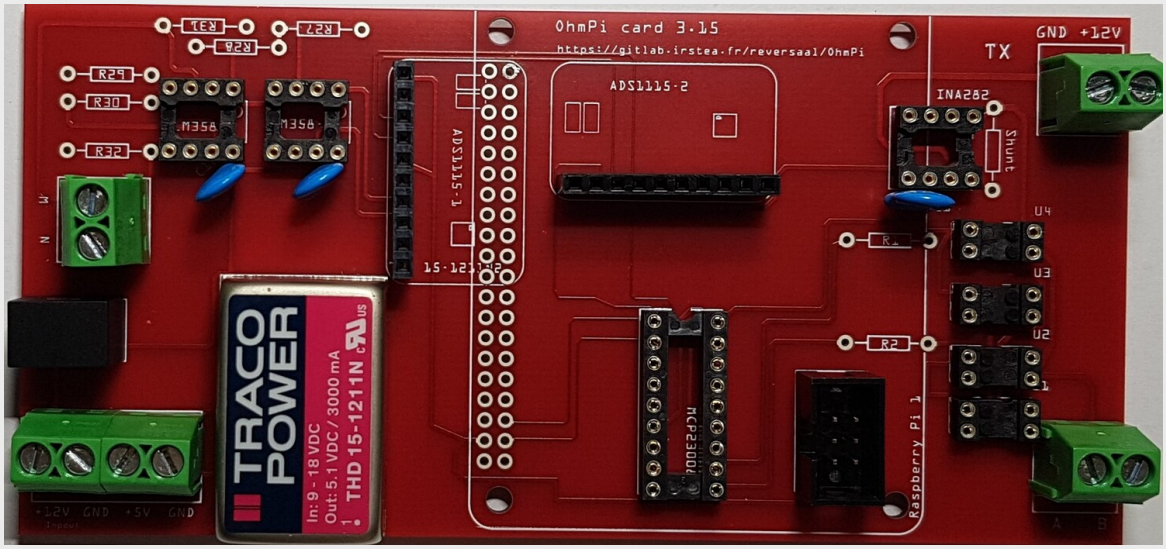


7 Four screw terminals for 12V inpout et 5V output



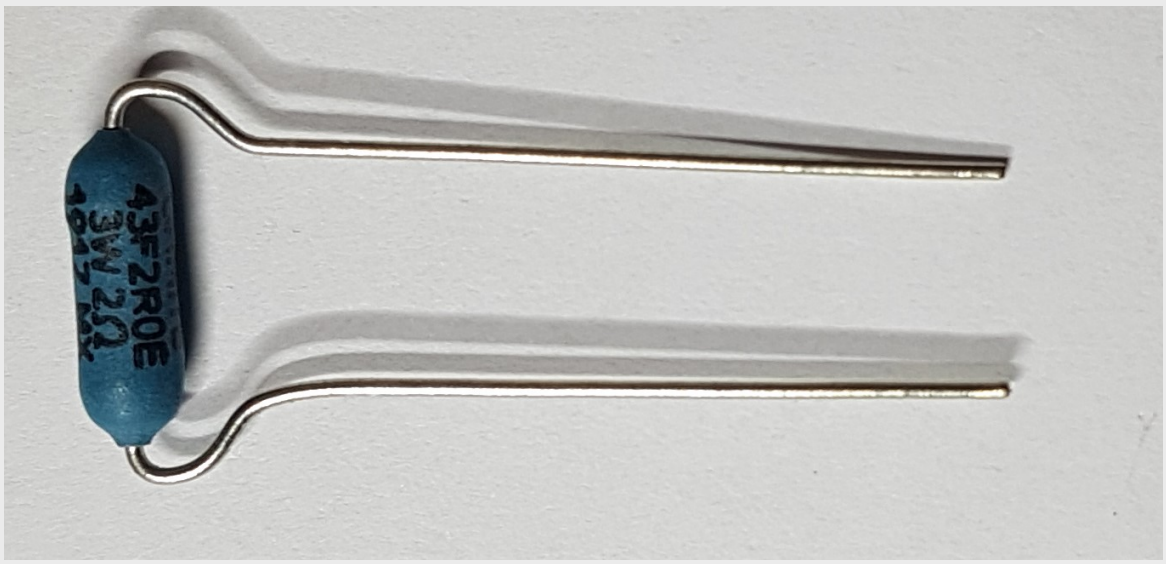
8 Traco power Supply 12V to 24V, TRN-1215

9	 Two screw terminals electrodes M and N
10	 Four screw terminals, for 12V power supply and for electrodes A and B



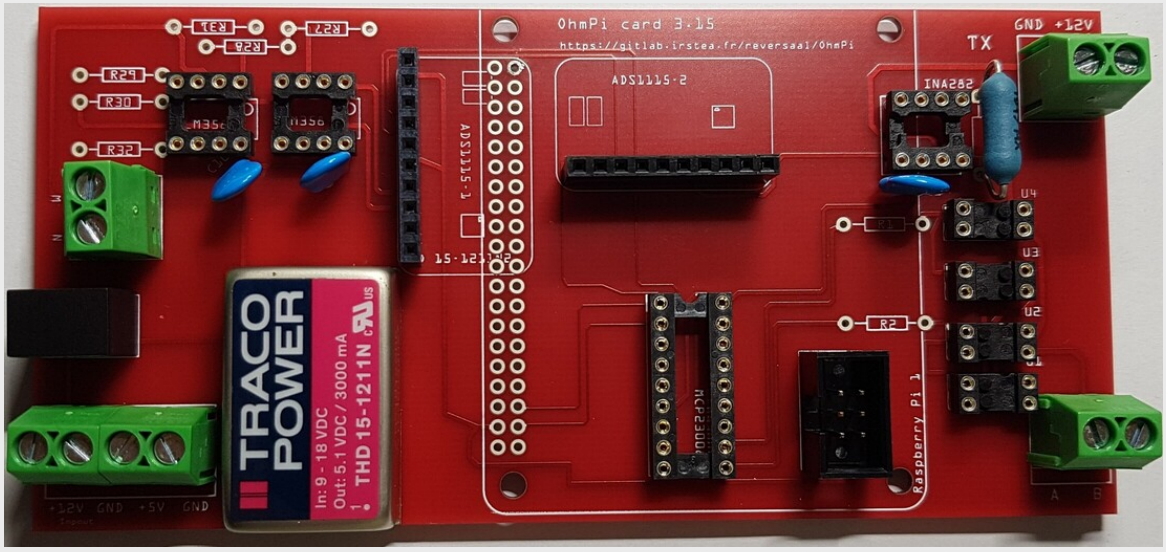
11 Soldering three capacitors (100nF)

Warning: In this version, we used a shunt resistor of 2 ohms, which limits the current measurement to 48 mA. If the current is higher than this value, you just have to decrease the value of the shunt resistor. Don't forget to change the shunt value in the config.py file (value associated to key 'R_shunt' in the OHMPI_CONFIG dict).



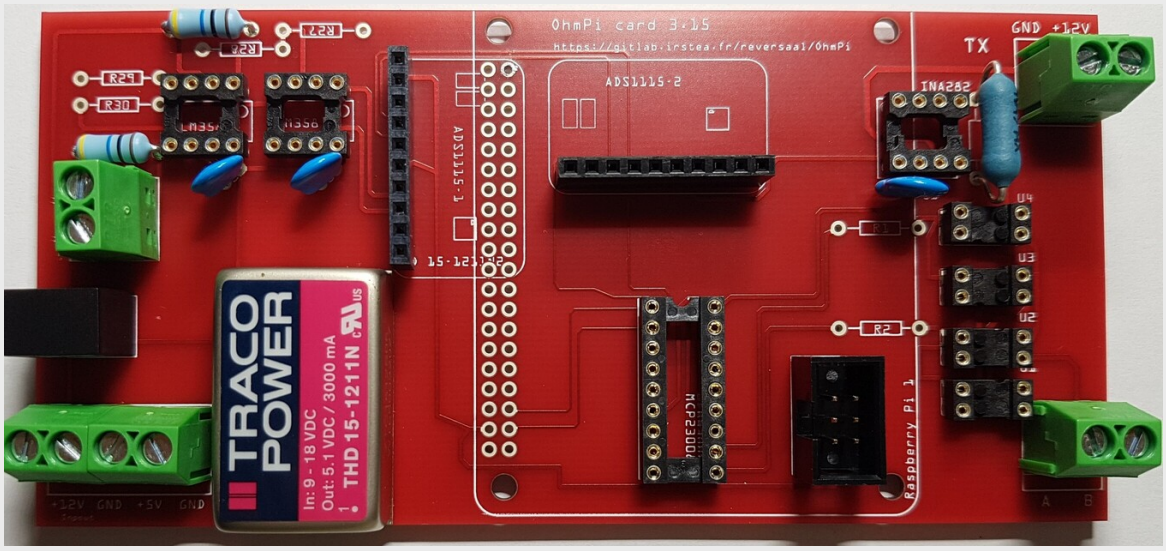
12 2 ohms shunt resistor pre-adjustment

13

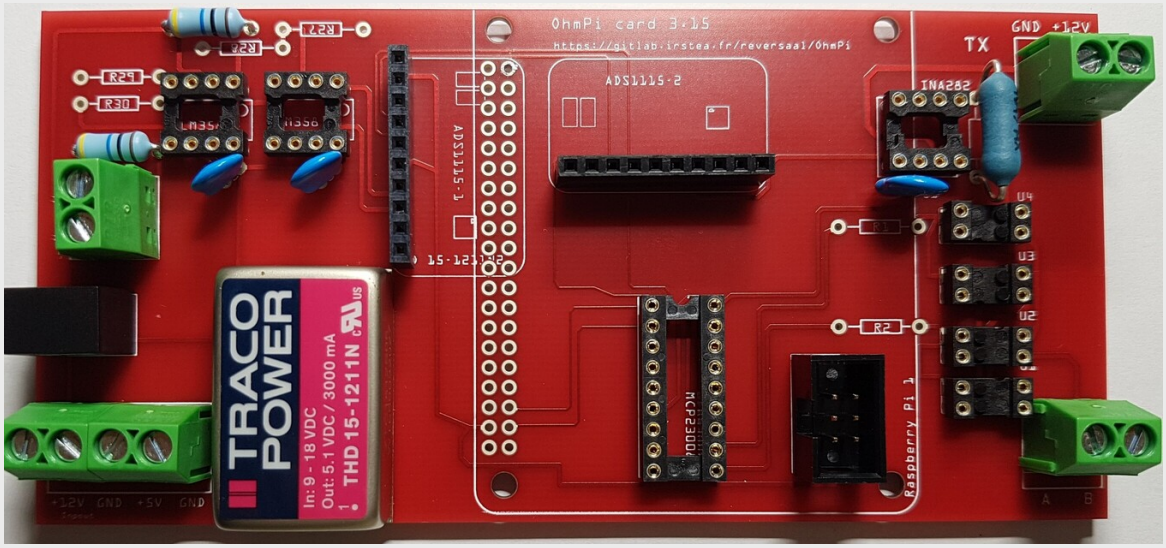


Soldering 2 ohms shunt resistor

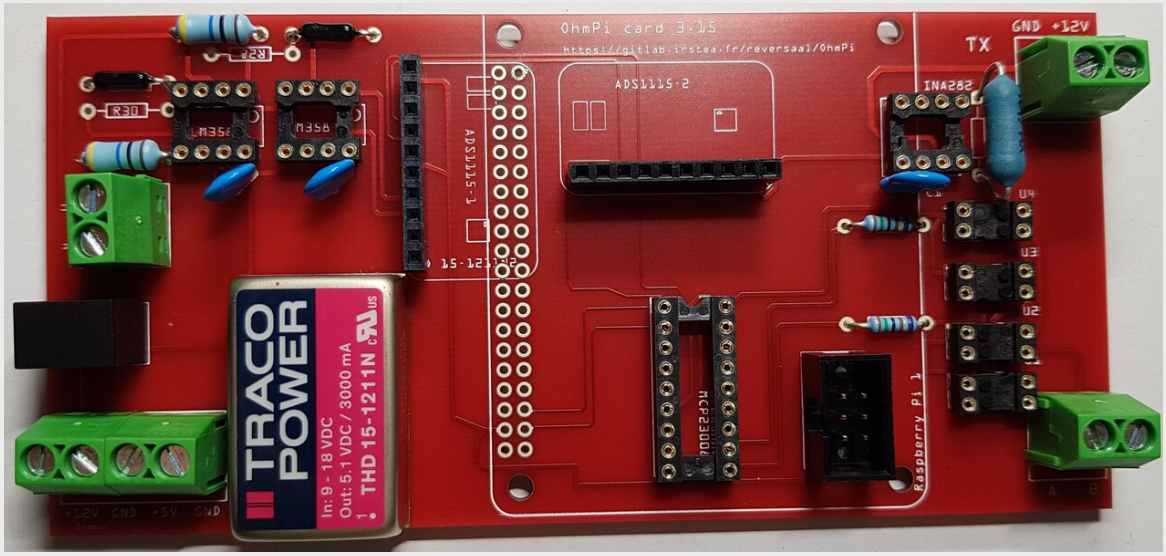
14



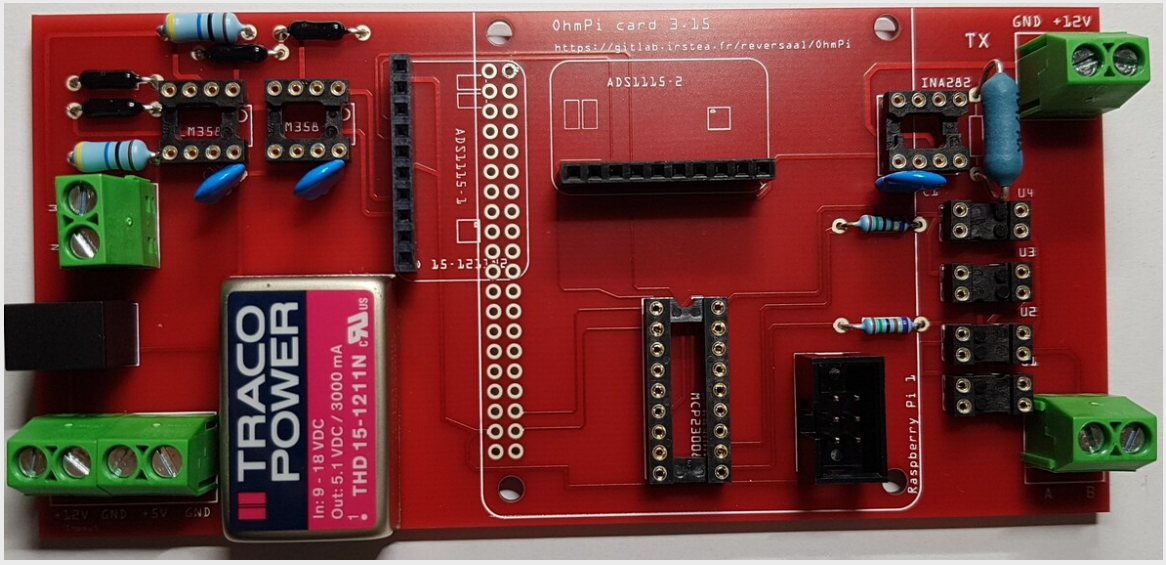
1 MOhm resistors



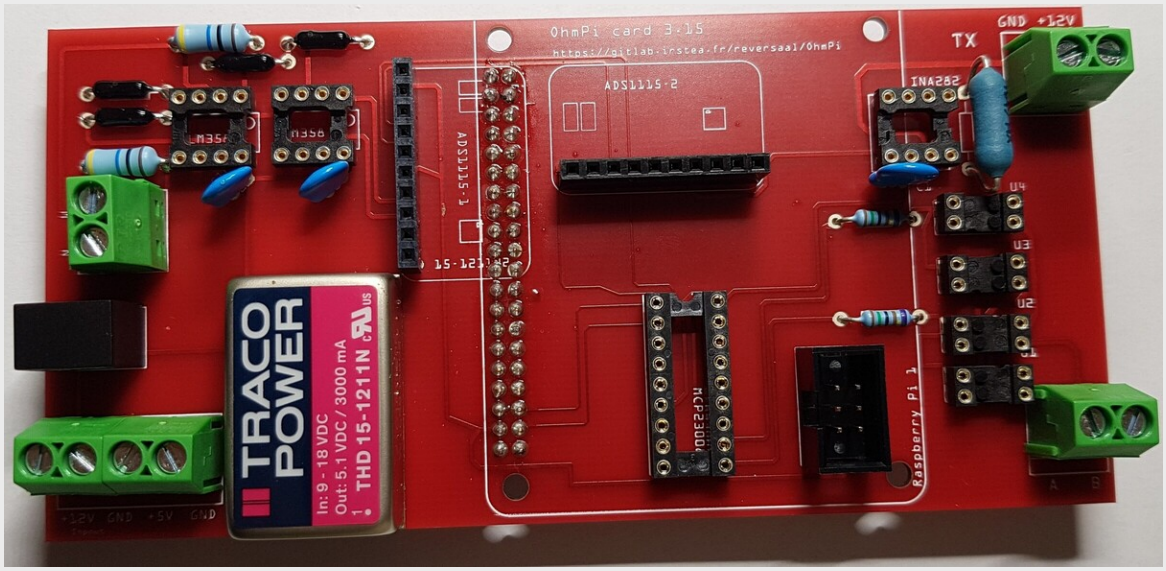
15 1 kOhm resistors



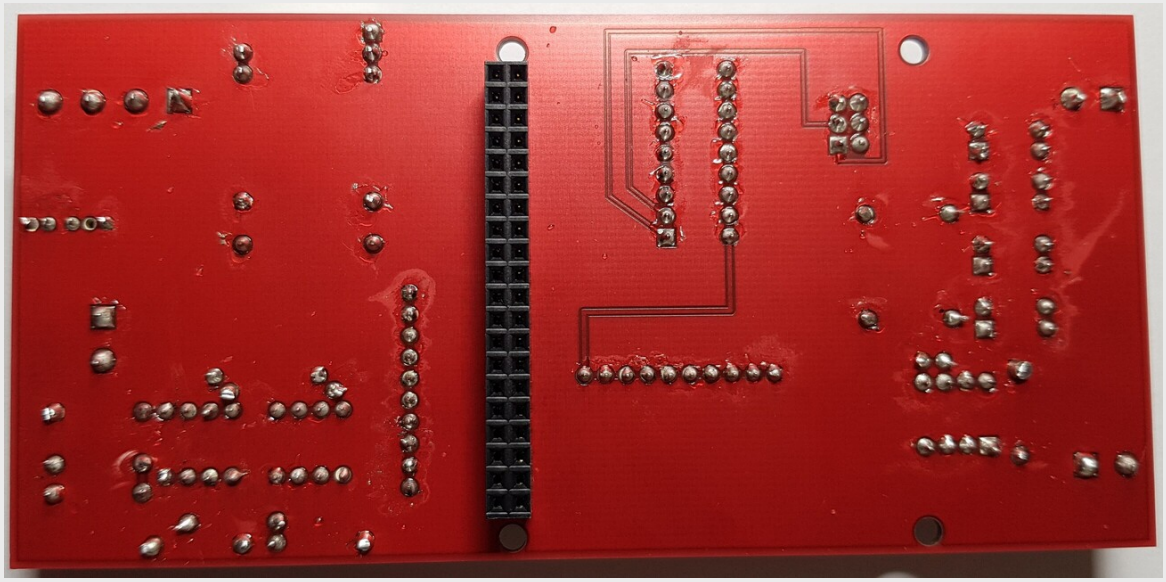
16 Soldering two 1.5 kOhm resistors



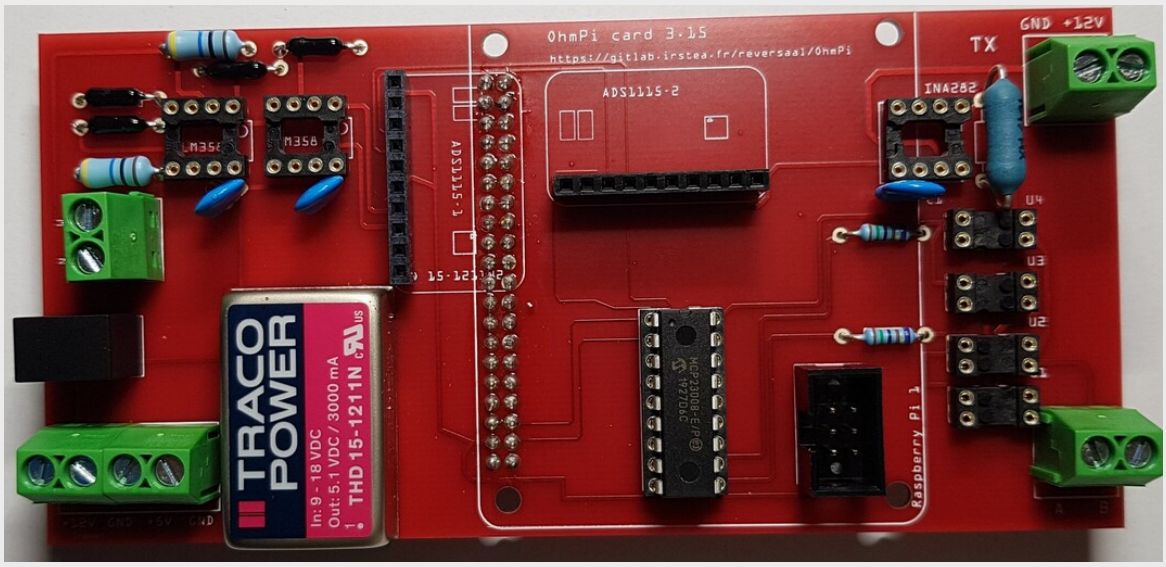
17 Soldering two 1.5 kOhms resistors



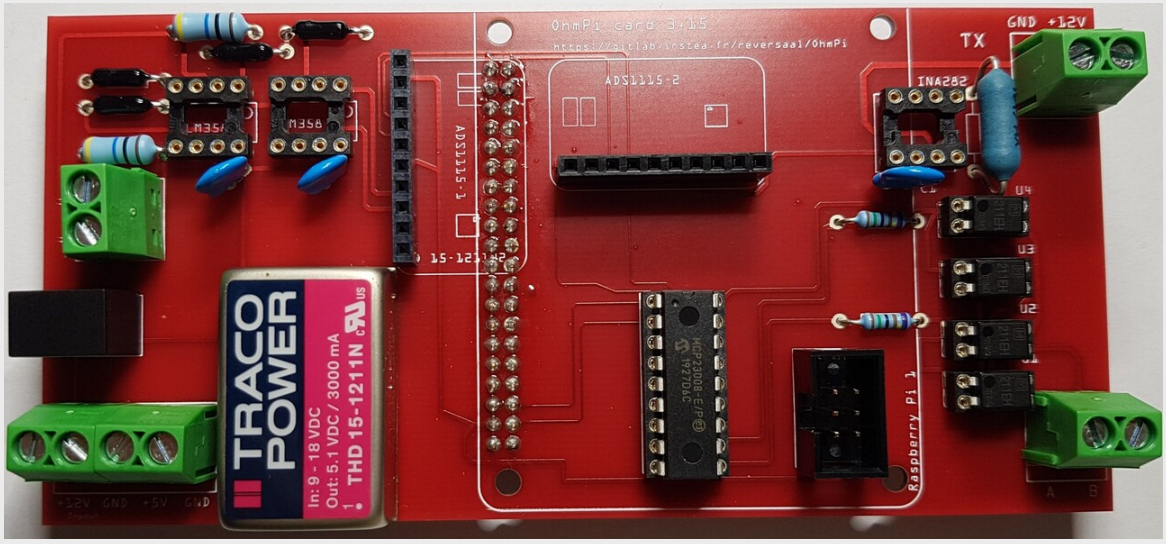
18 Female header 2 by 20 pins for Raspberry Pi connection



19 View of the female header 2 by 20 pins installation for Raspberry Pi connection

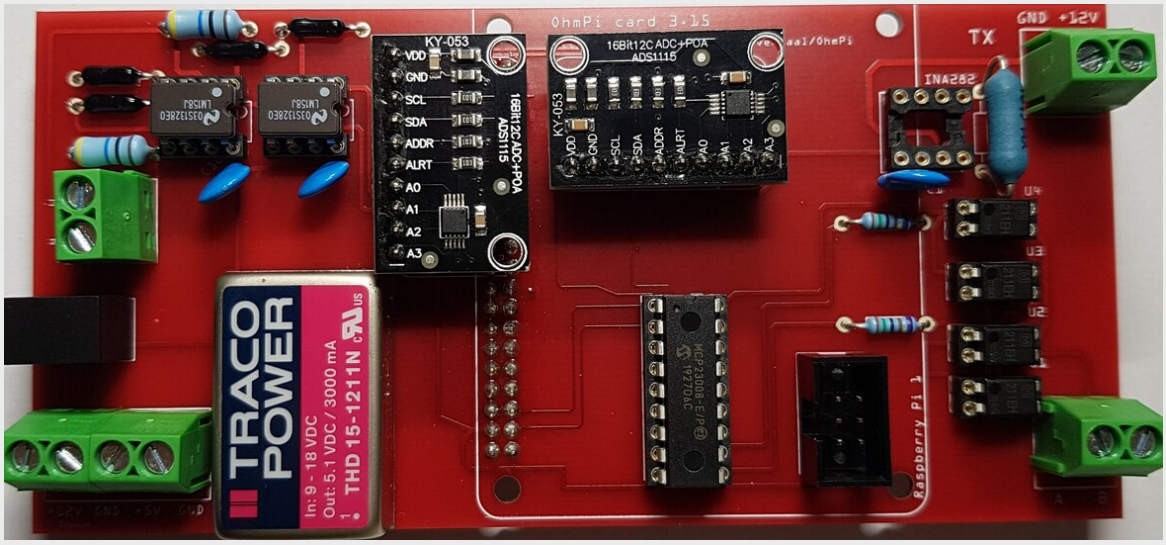


20 Fixing MCP23008 component (Dot mark on the top left corner)

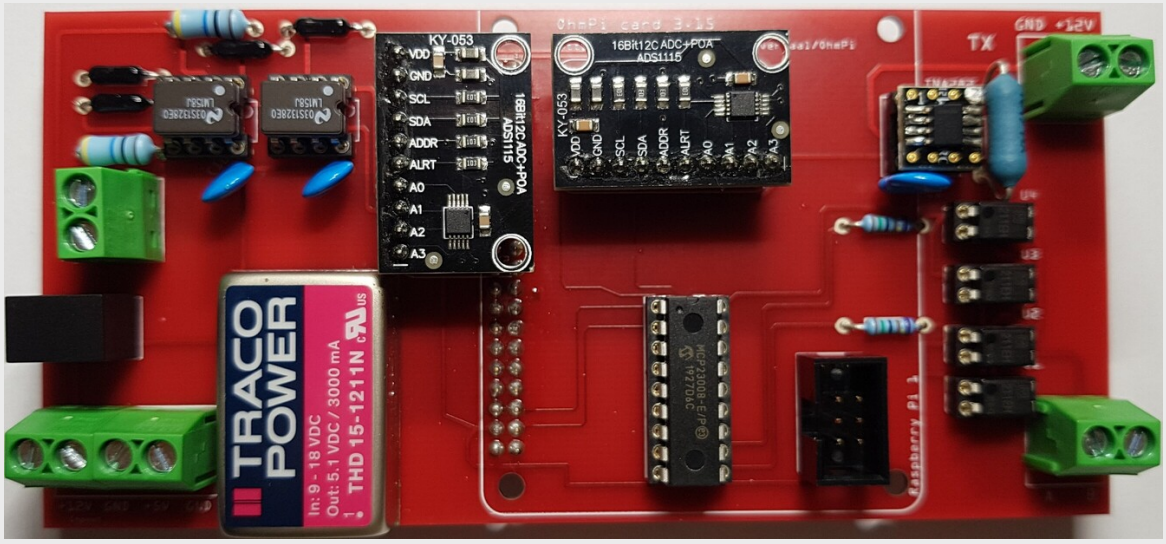


21 Mounting optical relay, AQY211EH (Dot mark in the top left corners)

22 Mounting LM158 operational amplifier



23 Mounting ADS115 board



24 Fixing the INA282 (Dot mark in the top right corner)

PART B Start-up of the measurement board

Required components

Table 2: List of components

Component	Number	Cost per unit €	Total cost €	Manufacturer	Manufacturer s reference	Web reference
Spacer 3x11 mm	8	0.39	3.12	Wurth Elektronik	9711103	https://www.mouser.fr/ProductDetail/Seeed-Studio/102110421?qs=7MVldsJ5UaxeN3LYyh3sq 3D% 3D
Screw	4	0.305	1.22	APM HEXSEAL	RM3X8M 2701	https://www.mouser.fr/ProductDetail/APM-HEXSEAL/RM3X8MM-2701?qs=JJSE% 2F12mKnS3VxSDrYXUH 3D% 3D

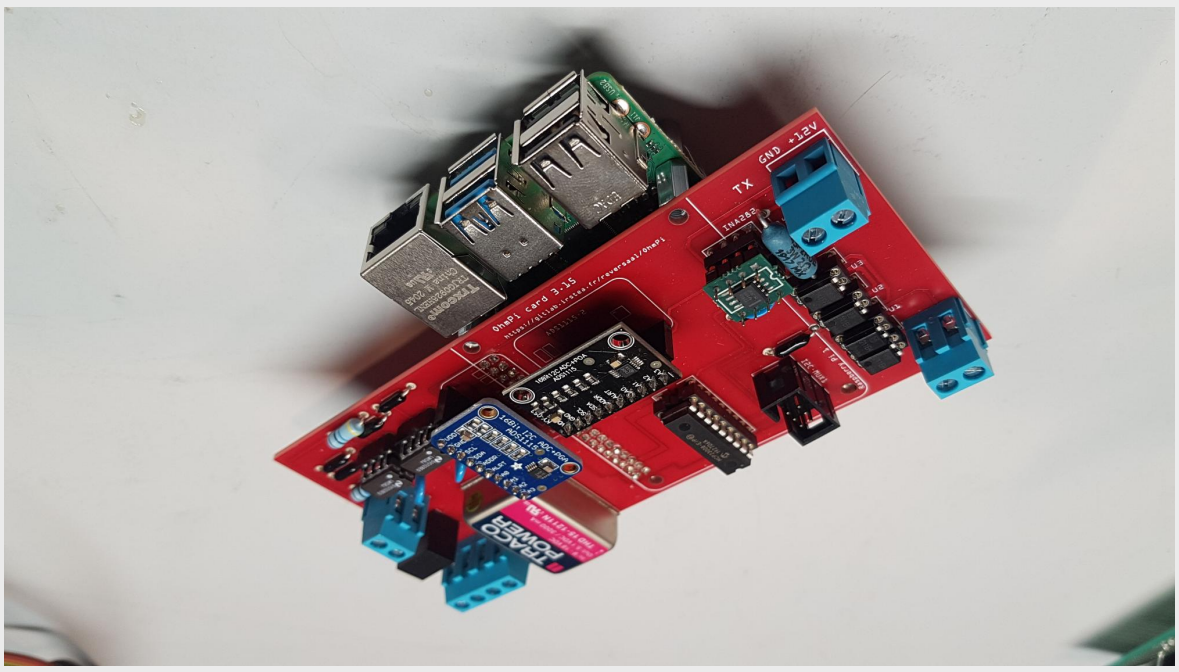
1

Shutdown the raspberry Pi and unplug the power supply

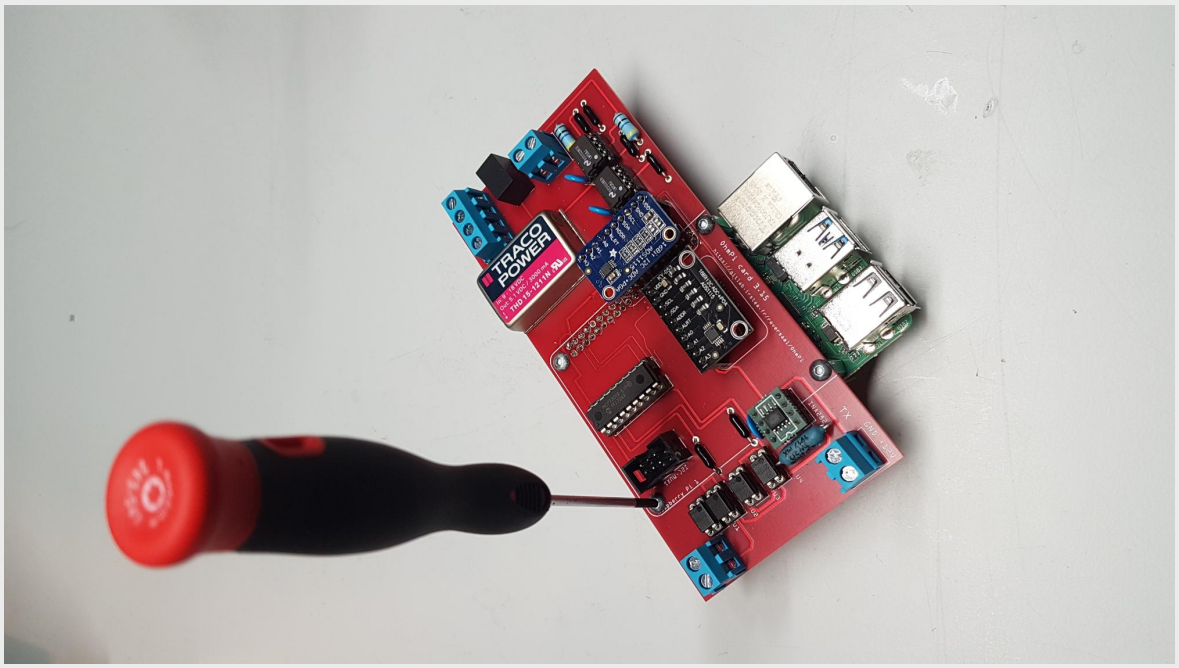
2

Mounting the bottom spacers on the Raspberry Pi (male/female, 11mm, M3)

3	
	Mounting the upper spacers on the Raspberry Pi (female/female, 11mm, M3)

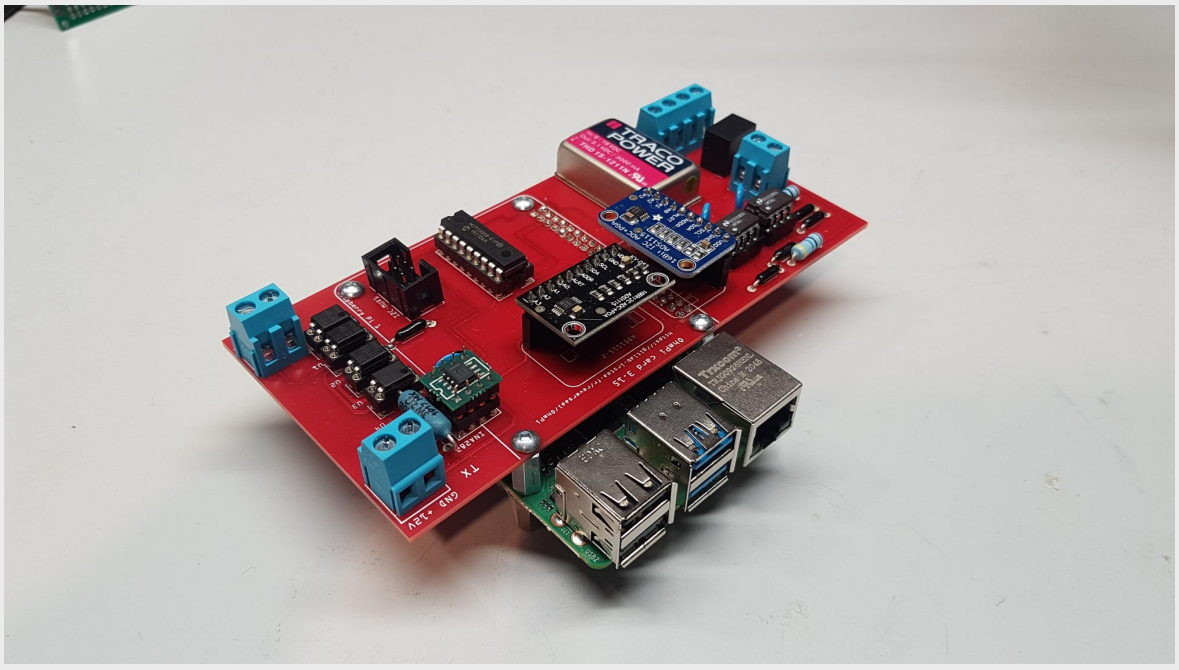
4	
	Mounting the OhmPi's measurement board on the Raspberry Pi

5



Mounting the OhmPi's measurement board on the Raspberry Pi

6

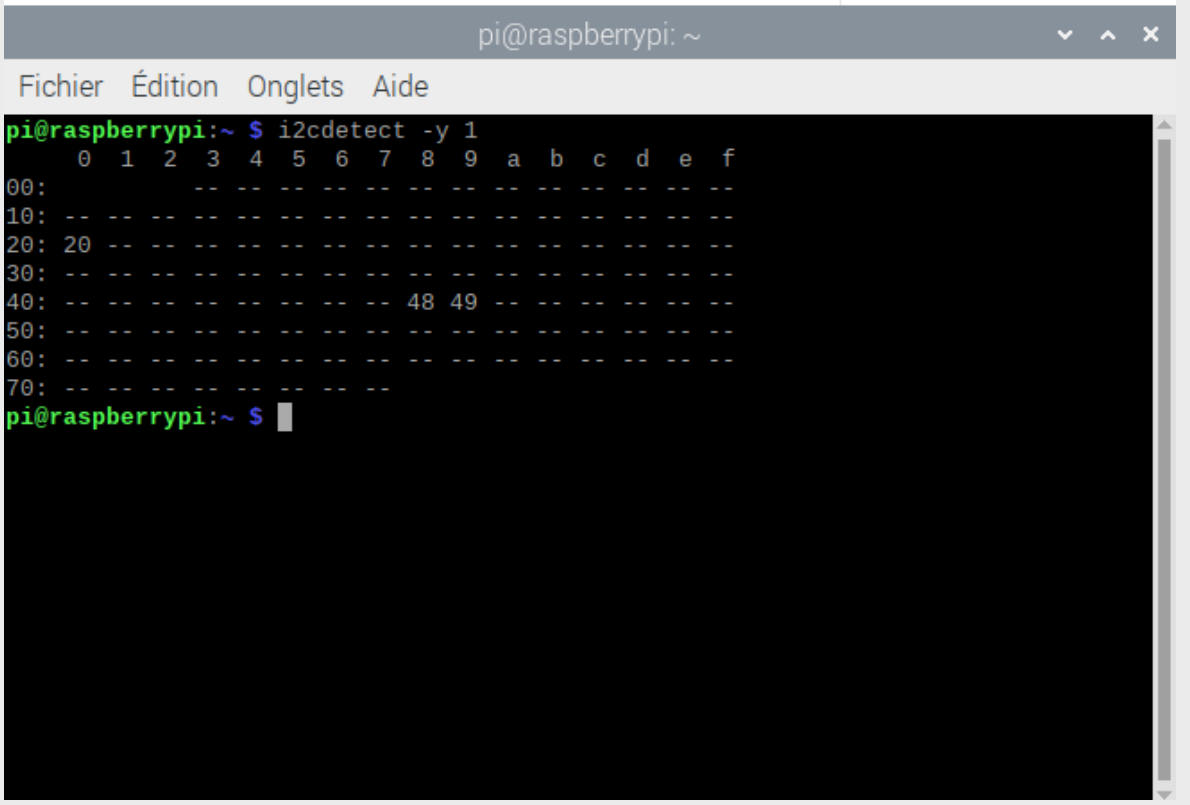


Plug the power supply into a socket and connect it to your Raspberry Pi's power port.

PART C Check the measurement board

Run the terminal, and write

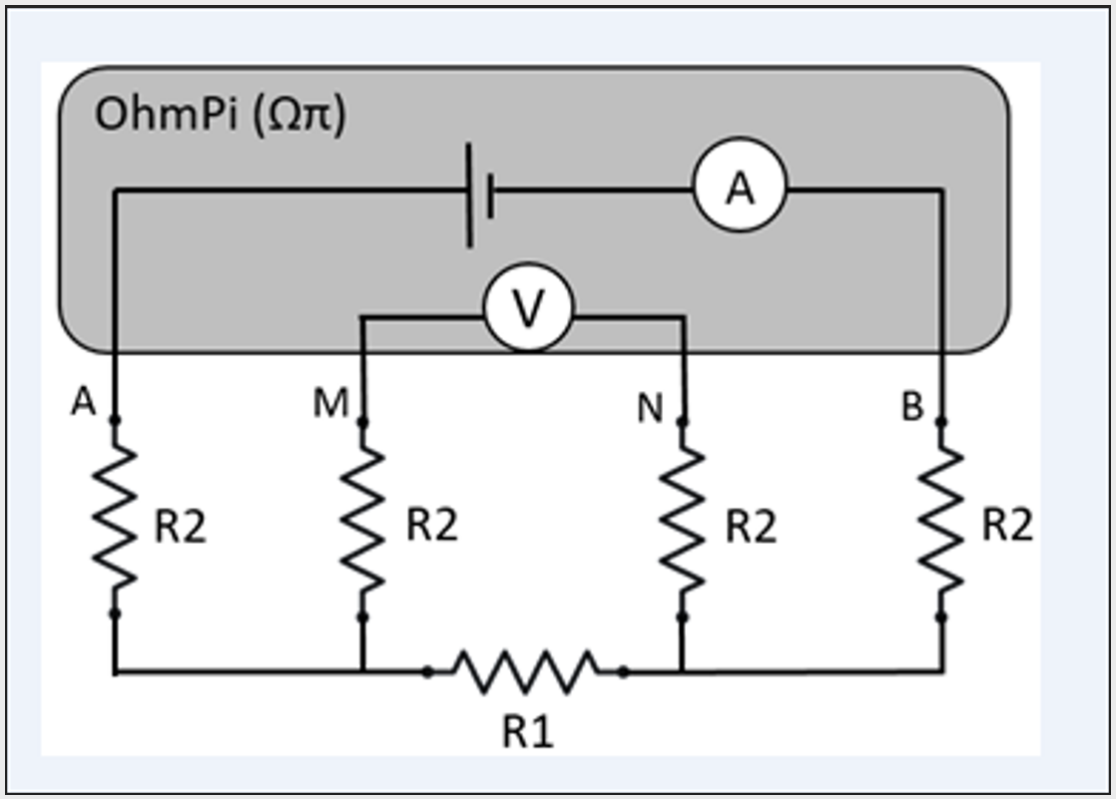
```
i2cdetect -y 1
```



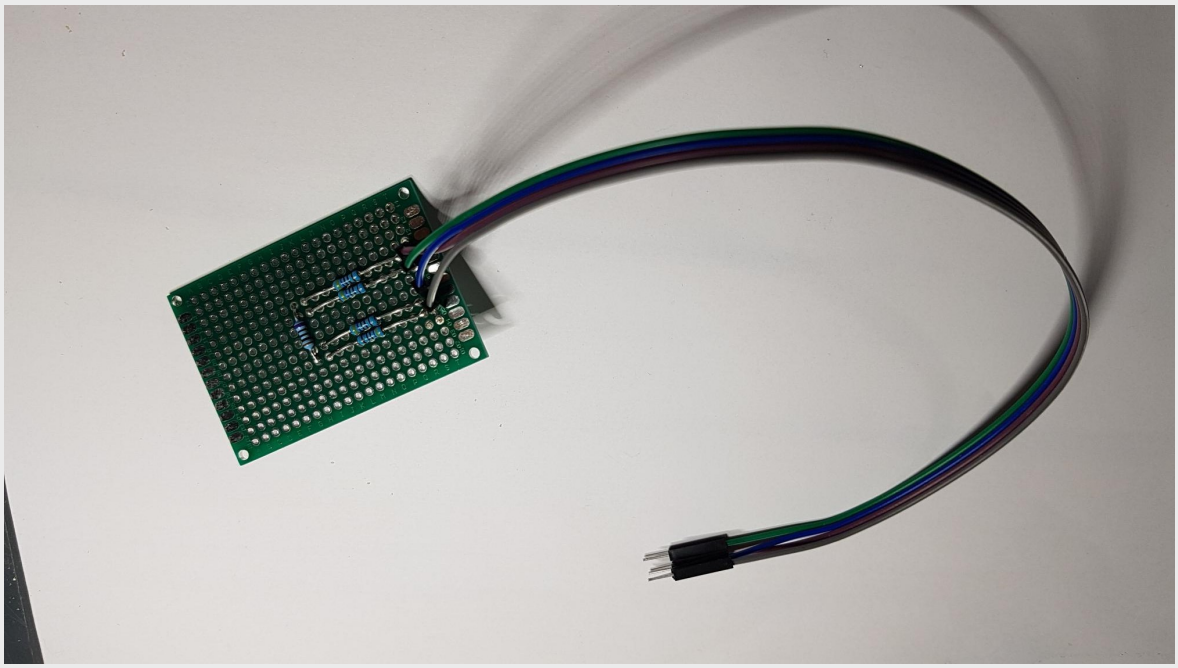
1 If everything is working, three I2C addresses should appear on the screen.

Setting up an equivalent electronic circuit, for this you will need:

- 4 1kOhm resistor (R2)
- 1 220 Ohm resistor (R1)
- 1 small padboard
- Spool of solder



2 Schematic of equivalent electronic circuit test

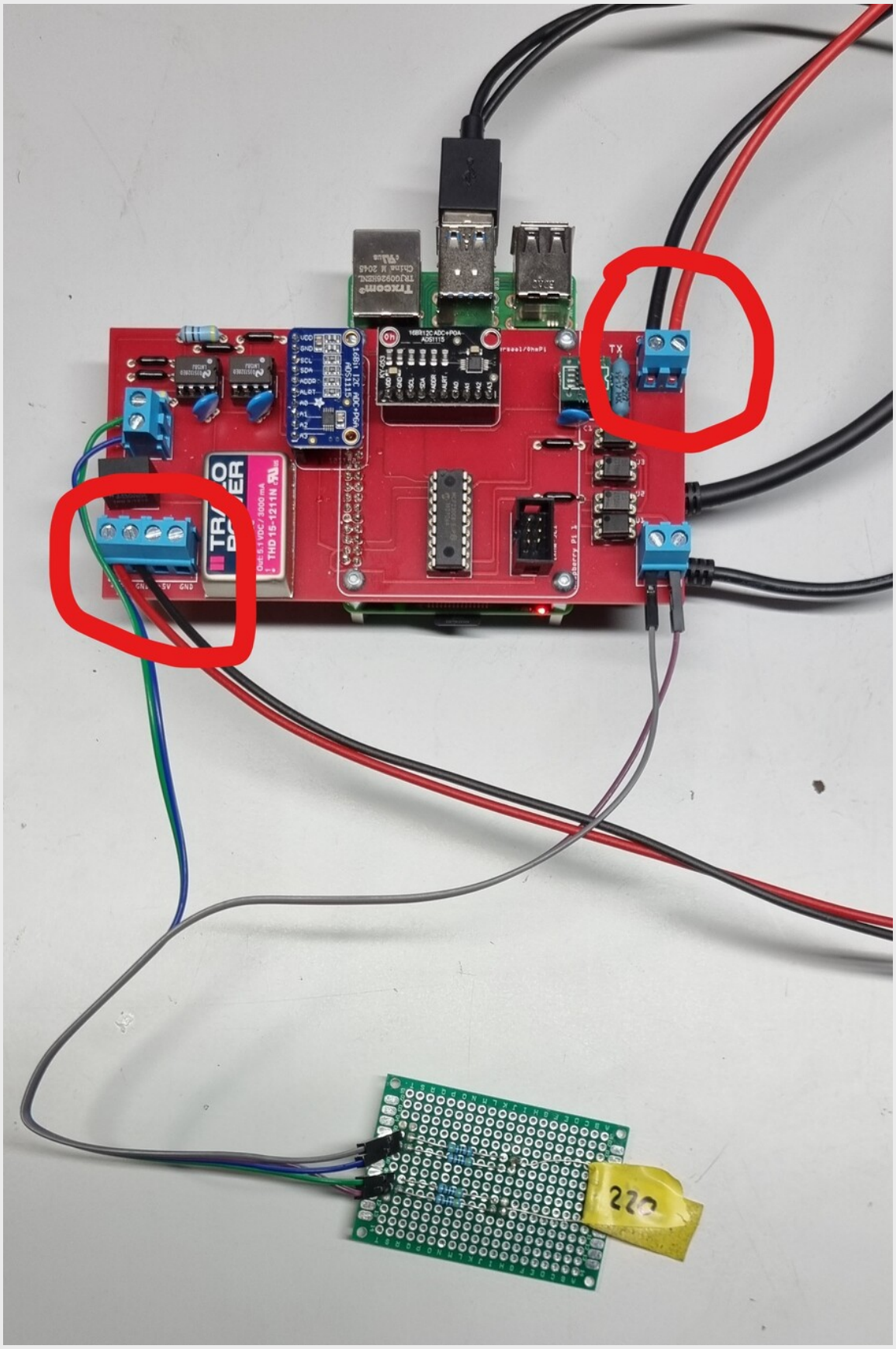


3 Prepare the equivalent electronic circuit test

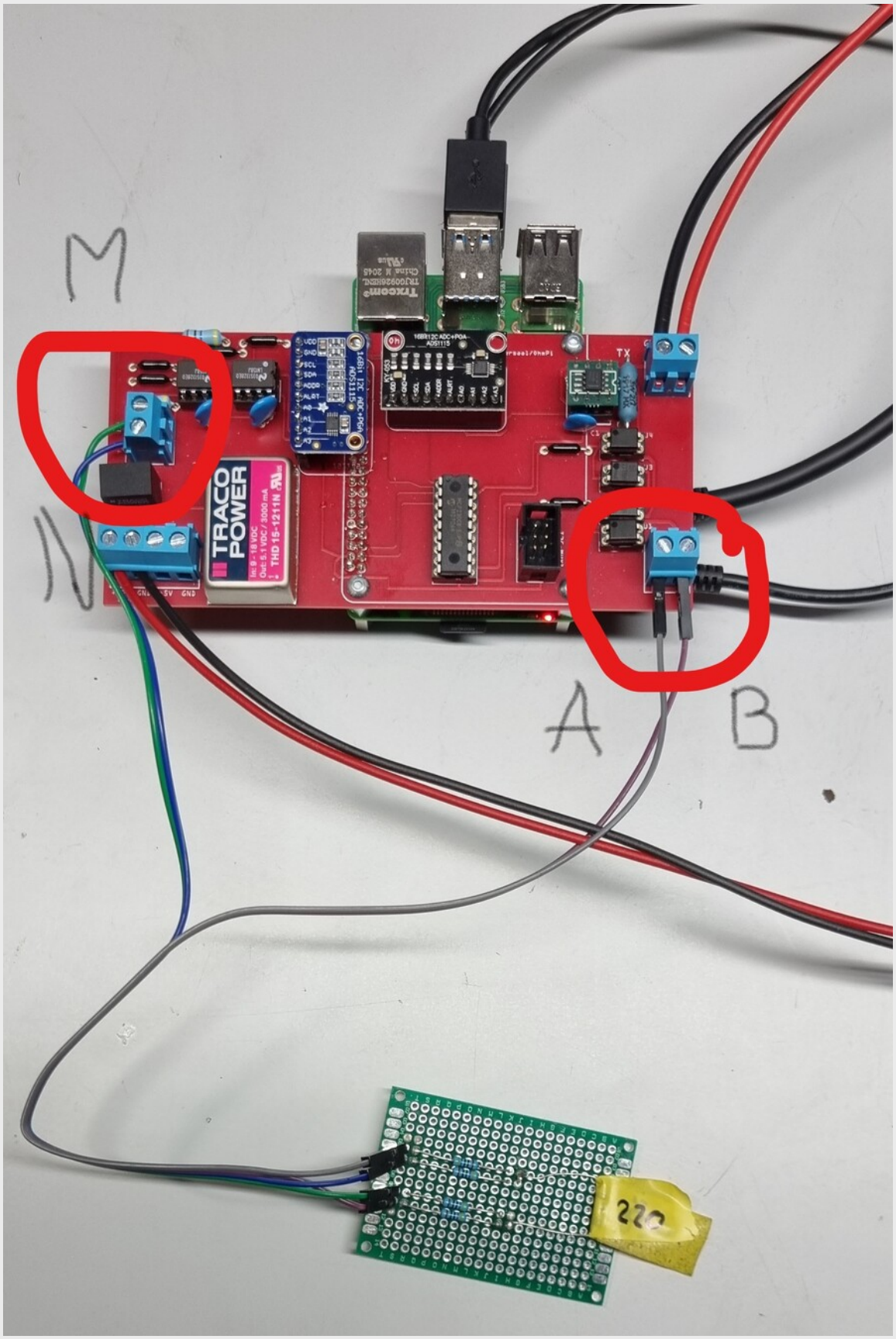
4 

Prepare the battery connections and the terminals.

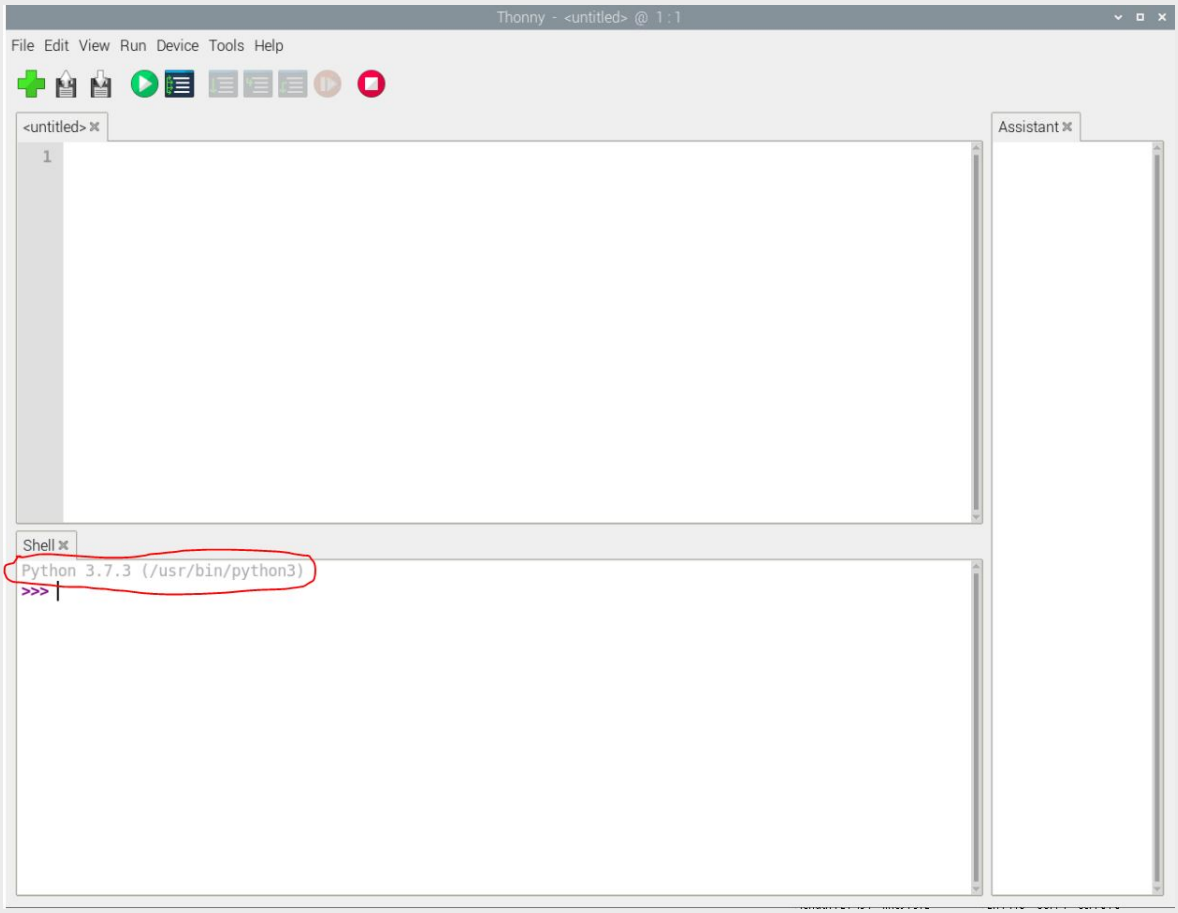
Soldering cables and terminals



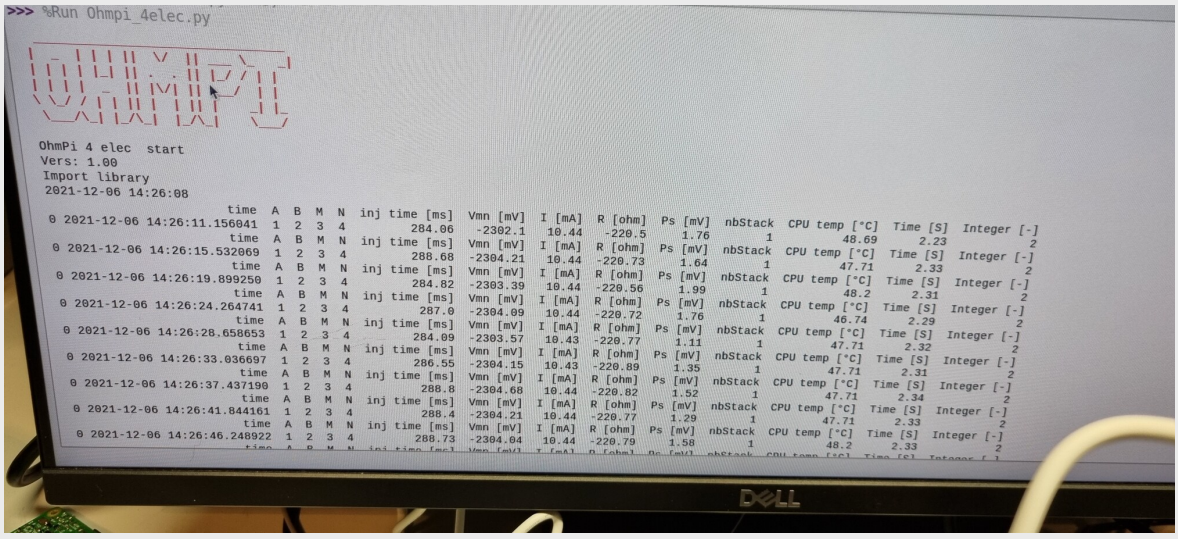
6 Install the red cables on the +12V terminal and black cable on the ground terminal. Connect to two different 12V batteries



7 Connect the equivalent electronic circuit test



8 Run the Thonny Interpreter

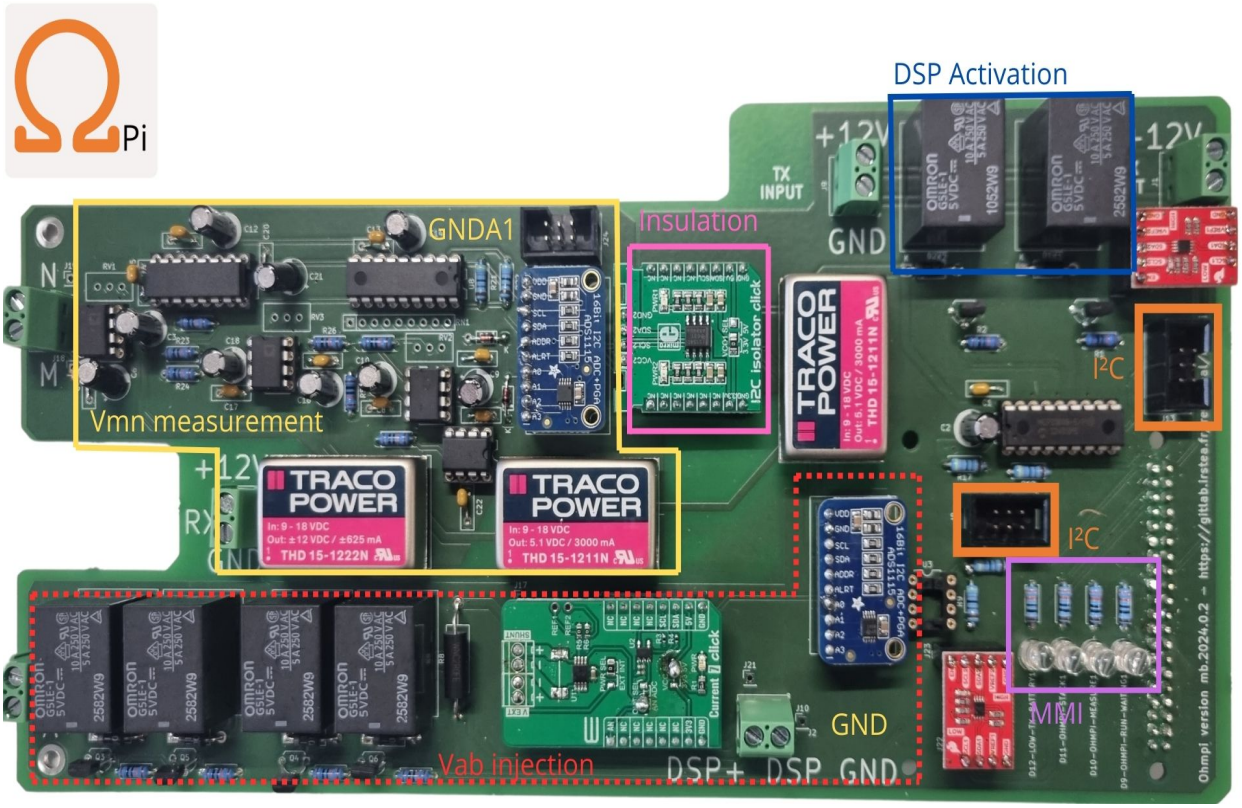


9 Run sample_measurement_example.py example, if everything works you should get the following result (220 Ohm)

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

Build measurement board v2024

The 2024.0.2 measurement board has been developed to replace the 2023.0.1 measurement board. It offers superior performance compared to its predecessor. The current measurement component has not evolved and presents no major differences. However, the major upgrade is the Mikroe-2C Isolator Click module ('https://www.mikroe.com/i2c-isolator-click') . Specifically, it provides electrical isolation for the Vmn measurement set. This isolation allows for injection voltages (Vab) up to 200V



Assembly of the measurement board

Schematics

Required components

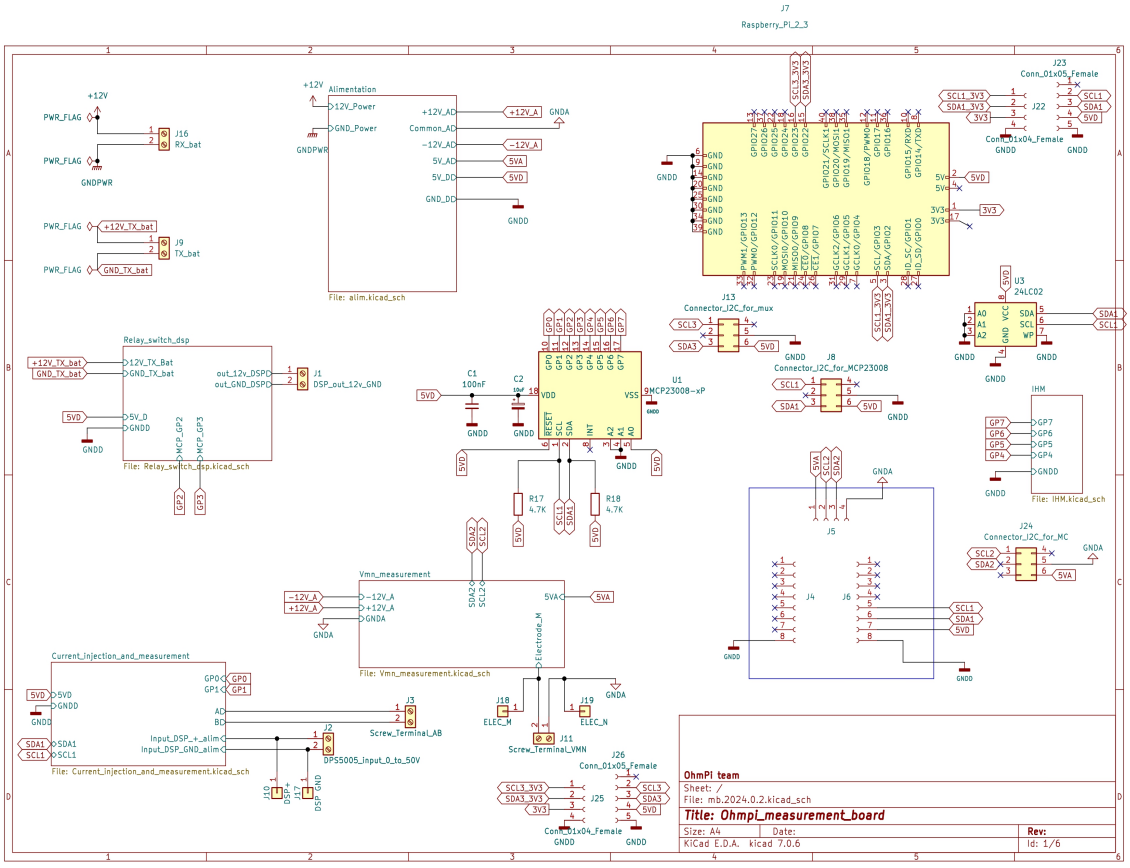


Fig. 2: Overview of the measurement board.

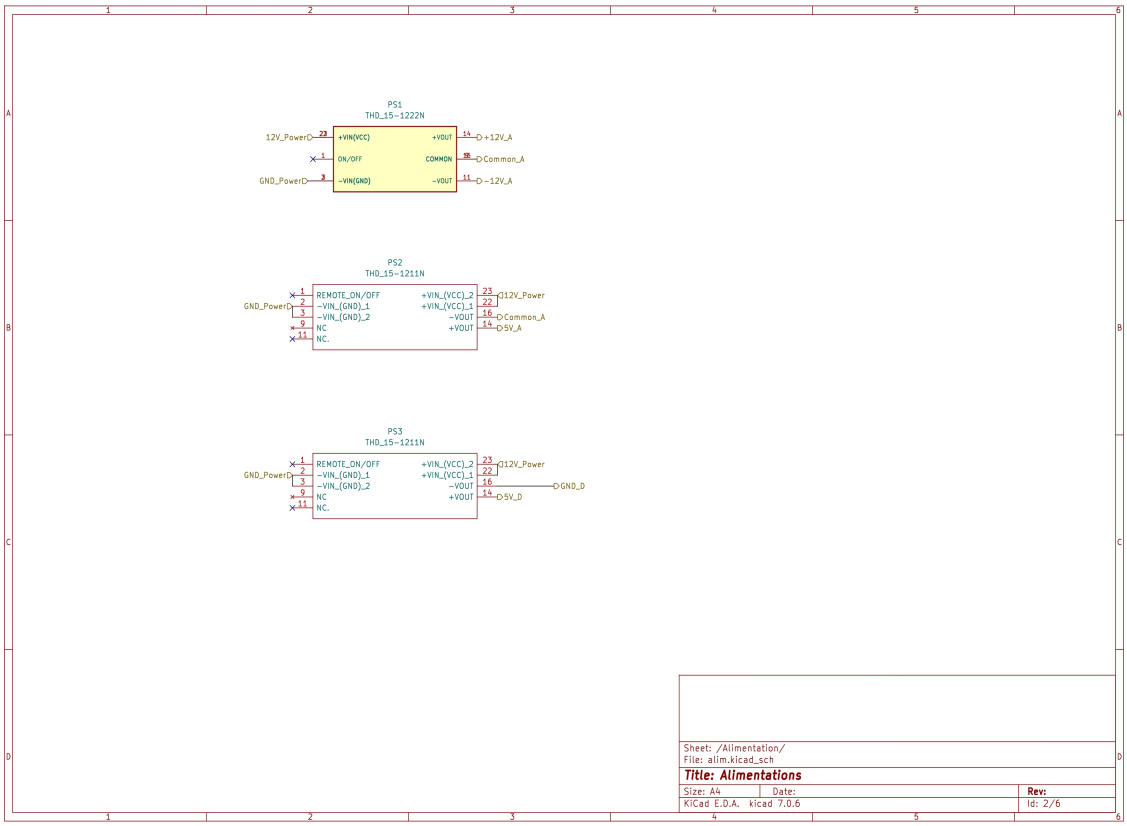


Fig. 3: Schematic of the power supply.

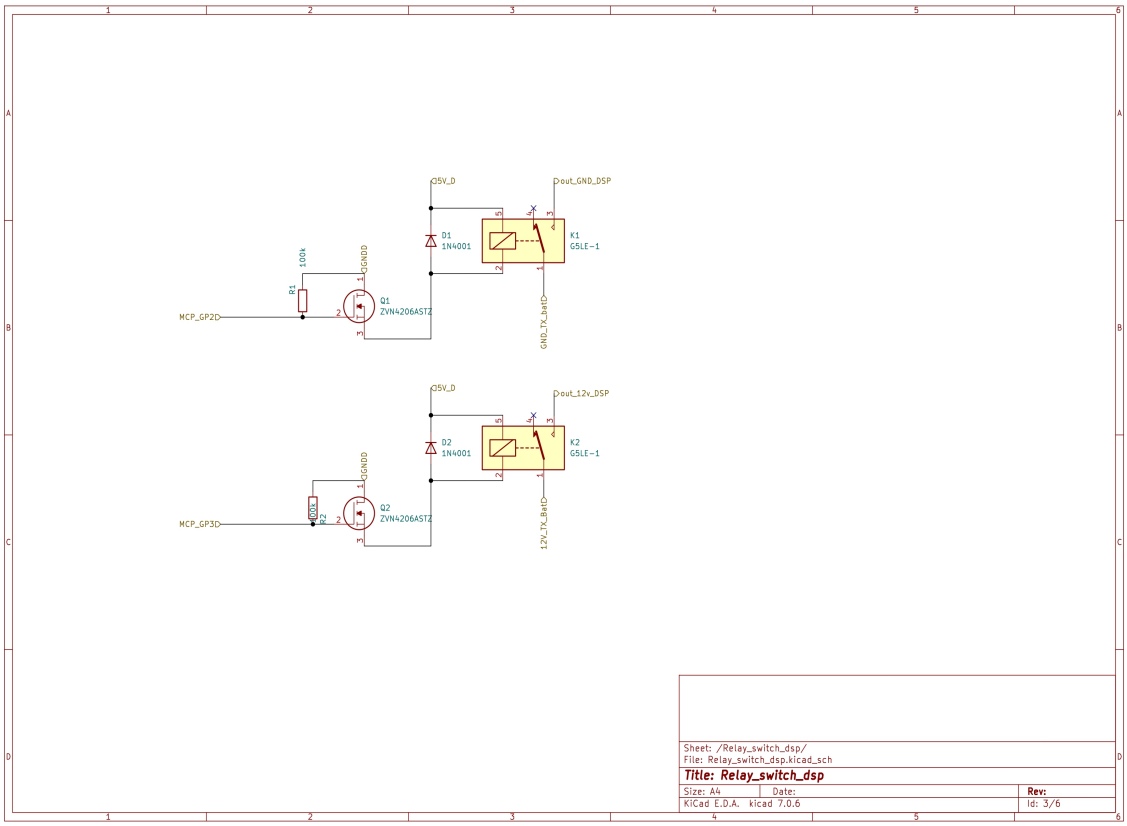


Fig. 4: Schematic of the DPS (digital power source) power supply (e.g. DPH5005).

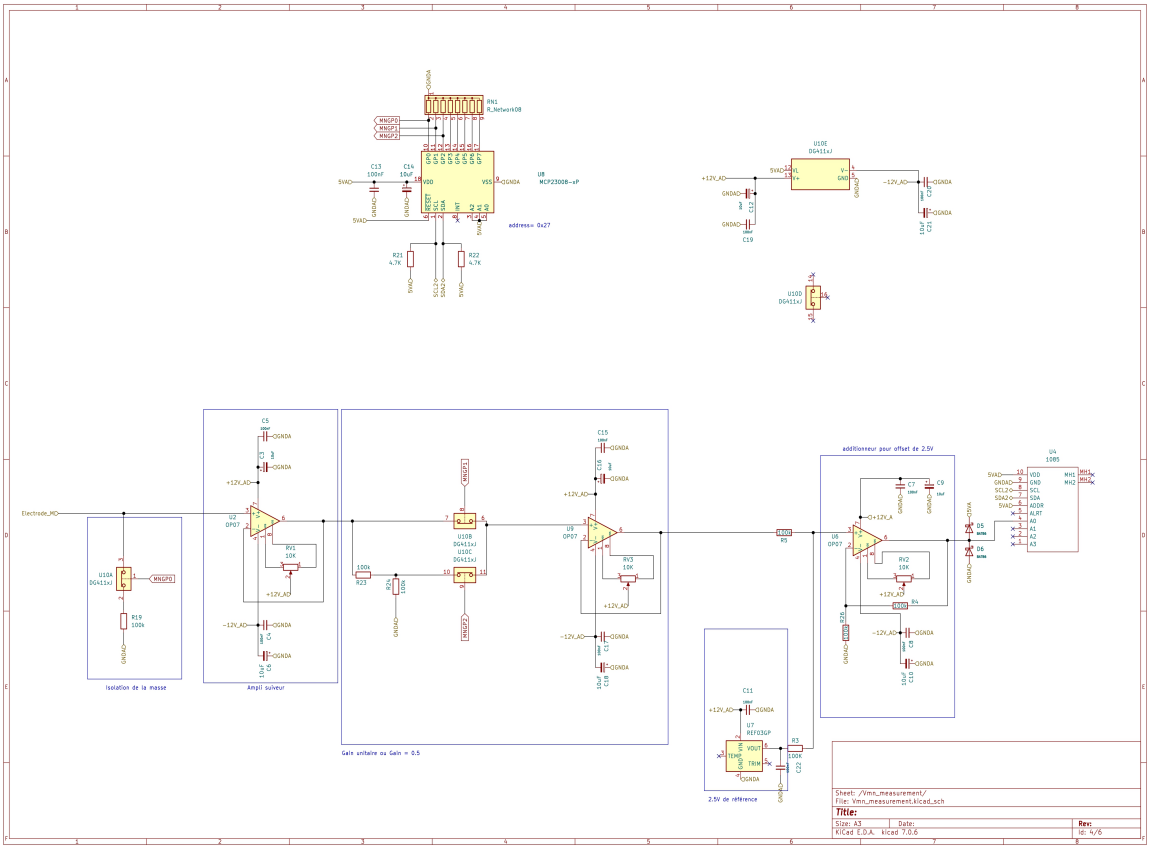


Fig. 5: Schematic of the Vmn signal conditioning.

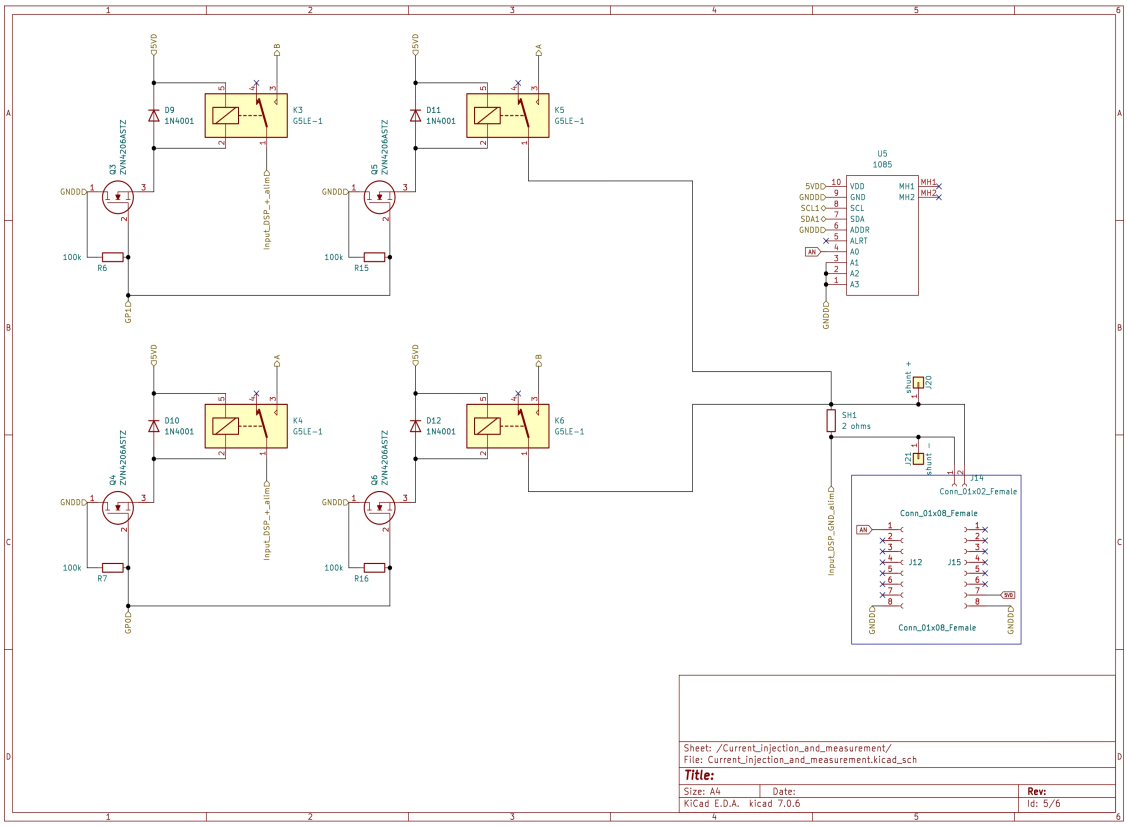


Fig. 6: Schematic of the current injection and measurement.

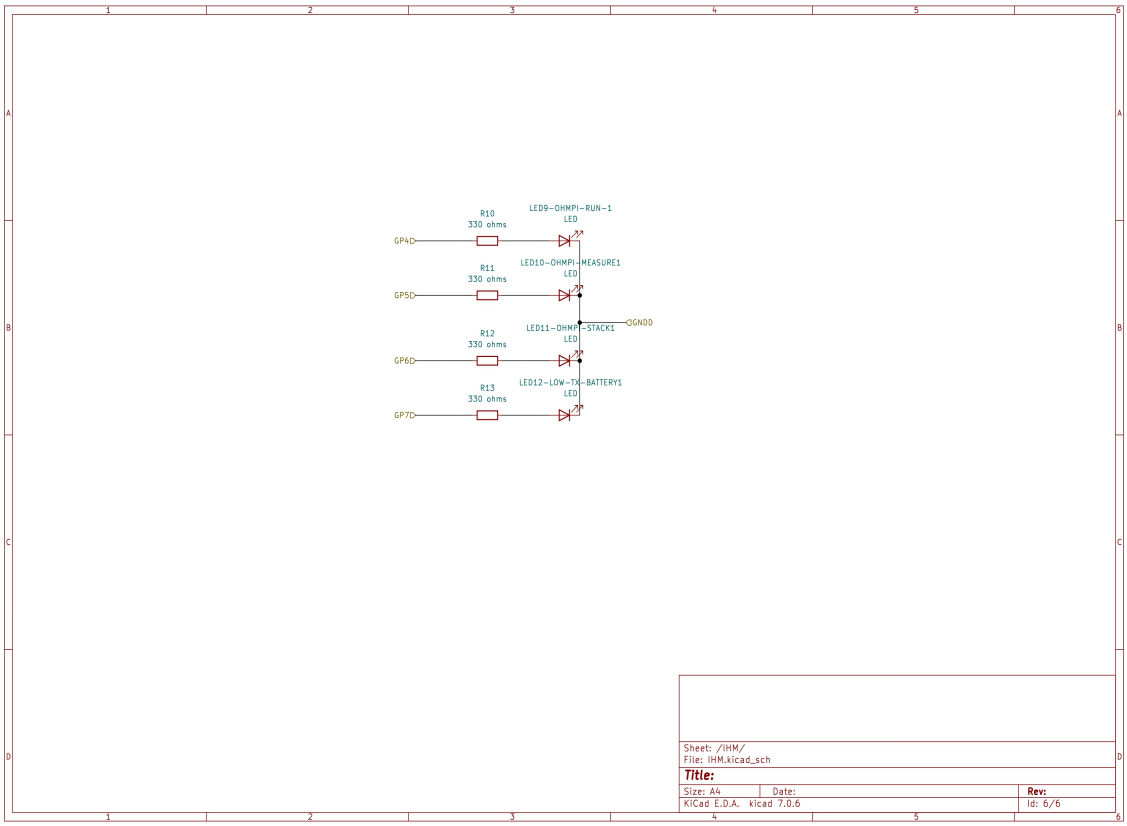


Fig. 7: Schematic of the human-machine interface.

Table 3: List of components

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
100 kOhm resistor	13	0.223	594-5063JD100	MBA02040	Vishay	Metal Film Resistors - Through Hole .4watt 100Kohms 1% 1/8watt body size	https://eu.mouser.com/ProductDet Vishay-Bey MBA02040 qs= mzRxyRlh 2F5Q% 3D%3D	2.899
330 Ohms resistor	4	0.307	279-CFR100J33	CFR100J33	TE Connectivity	Carbon Film Resistors - Through Hole 330Ohm 1W 500PPM	https://eu.mouser.com/ProductDet TE-Connec CFR100J33 qs= DDevMFO 3D%3D	1.228
4.7 kOhms resistor	4	0.307	603-CFR-25JB-52-4K7	CFR-25JB-52-4K7	YAGEO	Carbon Film Resistors - Through Hole 1/4W 4.7K Ohm 5%	https://eu.mouser.com/ProductDet YAGEO/ CFR-25JB- qs= oypCK0zG 2FvSUvCH 3D%3D	1.228
2 ohms shunt resistor	1	1.33	588-WNC2R0F	WNC2R0F	Ohmite	Wire-wound Resistors - Through Hole 2W 2 ohms 1%	https://eu.mouser.com/ProductDet Ohmite/ WNC2R0F qs= CDPnWzN 252Bw% 3D%3D	1.33

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
50V 1A general purpose rectifier diode DO-41	6	0.214	637-1N4001	1N4001	Diotec	Semiconductor Rectifiers Diode DO-41 50V 1A	https://eu.mouser.com/ProductDet Diotec-Sem 1N4001?qs= OlC7AqGiI 3D%3D	1.284
cree LED	4	0.279	941-C503BGAN	C503B-GAN-CD0E0781	Cree LED	Standard LEDs - Through Hole Green LED 527nm 5-mmRound 32900-64600mcd	https://eu.mouser.com/ProductDet Cree-LED/ C503B-GA qs= 7D1LtPJG0 252B% 252B5IGZv 3D%3D	1.116
50V 0.2 A small signal schottky diode DO-35	2	0.437	771-BAT86113	BAT86 113	Nexperia	Schottky Diodes & Rectifiers BAT86/SOI 34	https://eu.mouser.com/ProductDet Nexperia/ BAT86113 qs= me8Tqzrml 3D%3D	0.874
OP27E (single ultra offset 8DIP)	3	9.53	584-OP27EPZ	OP27EPZ	Analog Devices Inc.	Precision Amplifiers LOW-NOISE PRECISION OP AMP	https://eu.mouser.com/ProductDet Analog-Dev OP27EPZ? qs= WIvQP4zG 3D%3D	28.59
MCP23008 (GPIO expander)	2	1.89	485-593	593	Adafruit	Adafruit Accessories MCP23008 - i2c 8 input/output port expander	https://eu.mouser.com/ProductDet Adafruit/ 593?qs= GURawfae 2FuQ% 3D%3D	3.78

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
ADS1115 adafruit board (pack of 3)	2	14.9	485-1085	1085	Adafruit	Data Conversion IC Development Tools ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier	https://www.amazon.fr/AZDelivery dp/B07QHWL ref=sr_1_5?__mk_fr_FR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=18WHXZ3 keywords=ads1115&qid=170214021 sprefix=ads1115%2Caps%2C117&sr=8-5	14.99
2.5V precision voltage reference	1	6.12	584-REF03GPZ	REF03GPZ	Analog Devices Inc.	Voltage References PRECISION LOW-COST 2.5V R	https://eu.mouser.com/ProductDet Analog-De REF03GPZ qs=WIvQP4zG 3D%3D	6.12
DG411DJ analog switch	1	2.23	781-DG411DJ-E3	DG411DJ-E3	Vishay	Analog Switch ICs HIGH SPEED DG411 DIP-16	https://eu.mouser.com/ProductDet Vishay-Sen DG411DJ-I qs=xkjjIvogyb 252BuWCV 3D%3D	2.23

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
unpolar capacitor 100nF	12	0.214	594-K104K15X	K104K15X	Vishay	Multilayer Ceramic Capacitors MLCC - Leaded K 50V 100NF +/- 10 % X7R AMMO E3	https://eu.mouser.com/ProductDet Vishay-BC-K104K15X qs= rLgk8CAO 3D%3D	2.568
polarized capacitor 10uF	10	0.27	667-EEU-EB1J100S	EEU-EB1J100S	Panasonic	Aluminum Electrolytic Capacitors - Radial Leaded 10uF 63volts AEC-Q200	https://eu.mouser.com/ProductDet Panasonic/EEU-EB1J qs= cEAFgkeR 3D%3D	2.7
N channel 60V 600mA 700mW through hole transistor (ZVN4206	6	0.335	522-ZVN4206A	ZVN4206A	Diodes Incorporated	MOSFET N-Chnl 60V	https://www.mouser.be/ProductDet onsemi-Fai 2N7000BU qs= k2x4EL1% 2FKj6oeX 3D%3D	2.01

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
THD151211N (output 5V)	2	48.53	495-THD15-1211N	THD 15-1211N	TRACO Power	Isolated DC/DC Converters - Through Hole Product Type: DC/DC, Package Style: DIP-24, Output Power (W): 15, Input Voltage: 9-18 VDC, Output 1 (Vdc): 5.1, Output 2 (Vdc): N/A, Output 3 (Vdc): N/A	https://eu.mouser.com/ProductDet TRACO-Pc THD-15-12 qs= ckJk83FOL 3D%3D	97.06

nmg%

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
THD151222N (output 12V)	1	52	495-THD-15-1222N	THD 15-1222N	TRACO Power	Isolated DC/DC Converters - Through Hole Product Type: DC/DC, Package Style: DIP-24, Output Power (W): 15, Input Voltage: 9-18 VDC, Output 1 (Vdc): 12, Output 2 (Vdc): -12, Output 3 (Vdc): N/A	https://www.mouser.be/ProductDet TRACO-Pc THD-15-12 qs= ckJk83FOL 3D%3D	52
terminal block 12V	6	0.848	649-VI0221520	VI0221520	Amphenol	Fixed Terminal Blocks TB RIS CLA 180 STACK	https://eu.mouser.com/ProductDet Amphenol-VI0221520 qs= Mv7BduZu 3D%3D	5.088
mikroe i2C isolator	1	16.74	932-MIKROE-1878	MIKROE-1878	Mikroe	Interface Development Tools I2C Isolator click	https://eu.mouser.com/ProductDet Mikroe/MIKROE-1 qs= k5OWtXsT 3D%3D	16.74

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
mikroe shunt current sensor	1	24.18	932-MIKROE-4976	MIKROE-4976	Mikroe	Power Management IC Development Tools Current 7 Click	https://eu.mouser.com/ProductDet Mikroe/MIKROE-4 qs=MyNHzdoc 3D%3D	24.18
SparkFun Accessories Level Translator Breakout	2	4.03	474-BOB-15439	BOB-15439	SparkFun Electronics	SparkFun Accessories Level Translator Breakout - PCA9306	https://eu.mouser.com/ProductDet SparkFun/BOB-1543 qs=P1JMDcb9 252B0Xi4g 3D%3D	0
omhron G5LE relay	6	1.44	653-G5LE-1A4-DC5	G5LE-1A4-DC5	Omron	General Purpose Relays Power PCB Relay SPST-NO Sealed 5VDC	https://eu.mouser.com/ProductDet Omron-Ele G5LE-1A4 qs=pWf36BUt 3D%3D	8.64
DIP for MCP23008 (18 pins)	2	1.33	575-110473184	110-47-318-41-001000	Mill-Max	IC & Component Sockets STANDRD SOLDER TAIL DIP SOCKET	https://eu.mouser.com/ProductDet Mill-Max/110-47-318 qs=5aG0NVq1 3D%3D	2.66
DIP for OP27E (8 pins)	5	1.39	575-113308	110-13-308-41-001000	Mill-Max	IC & Component Sockets 8P GLD PIN GLD CONT	https://eu.mouser.com/ProductDet Mill-Max/110-13-308 qs=WZeyYeqM 3D%3D	6.95

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
DIP for DG411DJ (16 pins)	1	1.05	575-11044316	110-44-316-41-001000	Mill-Max	IC & Component Sockets 16P TIN PIN TIN CONT	https://eu.mouser.com/ProductDet Mill-Max/110-44-316 qs=IGgAdOvC 252BXEWı 3D%3D	1.05
header for raspberrypi	1	1.87	474-PRT-14017	PRT-14017	SparkFun Electronics	Raspberry Pi Accessories Raspberry Pi GPIO Tall Header - 2x20	https://eu.mouser.com/ProductDet SparkFun/PRT-14017 qs=a4BXICGg 2F% 252BaML8 3D%3D	1.87
header socket 1 row 5 positions	2	1.97	571-5-534237-3	5-534237-3	TE Connectivity	Headers & Wire Housings REC 1X05P VRT T/H	https://eu.mouser.com/ProductDet TE-Connec 5-534237-3 qs=Eln3I3szM 252BSZCsı 3D%3D	3.94
header socket 1 row 4 positions	3	1.46	571-5-534237-2	5-534237-2	TE Connectivity	Headers & Wire Housings REC 1X04P VRT T/H	https://eu.mouser.com/ProductDet TE-Connec 5-534237-2 qs=GYgf5Pdsjz 3D%3D	4.38

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
header socket 1 row 8 positions	4	1.74	571-5-535541-6	5-535541-6	TE Connectivity	Headers & Wire Housings REC 1X08P VRT T/H	https://eu.mouser.com/ProductDet TE-Connec 5-535541-6 qs= xDp7PGUN 252BuqVw 3D%3D	6.96
header socket 1 row 10 positions	2	2.71	200-SSW11002	SSW-110-02-G-S	Samtec	Headers & Wire Housings Tiger Buy Socket Strip with PCB Tails, .100" Pitch	https://eu.mouser.com/ProductDet Samtec/ SSW-110-0 qs= rU5fayqh% 252BE0w1 3D%3D	5.42
header socket 1 row 2 positions	1	1.02	200-SSW10202	SSW-102-02-G-S	Samtec	Headers & Wire Housings Tiger Buy Socket Strip with PCB Tails, .100" Pitch	https://eu.mouser.com/ProductDet Samtec/ SSW-102-0 qs= rU5fayqh% 252BE2ZE 2FBLw% 3D%3D	1.02
header pins 1 row 10 positions	1	1.24	571-1-826629-0	1-826629-0	TE Connectivity	Headers & Wire Housings 10P SINGLE ROW	https://eu.mouser.com/ProductDet TE-Connec 1-826629-0 qs= FazuUmnc 3D%3D	1.24

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
IDC sockets (go on the ribon cable)	2	0.43	710-612006230	612006230	Wurth Elektronik	Headers & Wire Housings WR-BHD 2.54mm Female 6P Strt IDC Conn	https://eu.mouser.com/ProductDet Wurth-Elek 612006230 qs= PhR8RmCi 3D%3D	0.86
IDC pins	3	1.32	571-1761681-1	1761681-1	TE Connectivity	Headers & Wire Housings IDC LOW PRO HDR 6P VERT HT BLACK	https://eu.mouser.com/ProductDet TE-Connec 1761681-1 qs= BqFpTYCC 3D%3D	3.96
PCB measurement board (pack of 3)	1						https://aisler.net/?lang=fr-FR	70
Raspberryp pi 4	1	63					https://www.kiwi-electr com/en/raspberry-p src= raspberrypi	63
socket to cut for 3 THD	3	1.73					https://www.mouser.be/ProductDet Mill-Max/ 801-47-012 qs= 5aG0NVq1 3D%3D	5.19
10k potentiometer	2	optional						0
R array sip9	1	optional						0

continues on next page

Table 3 – continued from previous page

Component	Quantity	Cost per unit	Mouser ref	Manufacturer	Manufacturer reference	Description	Web link	Total cost EUR excl.VAT
10k potentiometer bournds 3296W_ver	1	optional						0

Interactive BOM list

Interactive BOM list

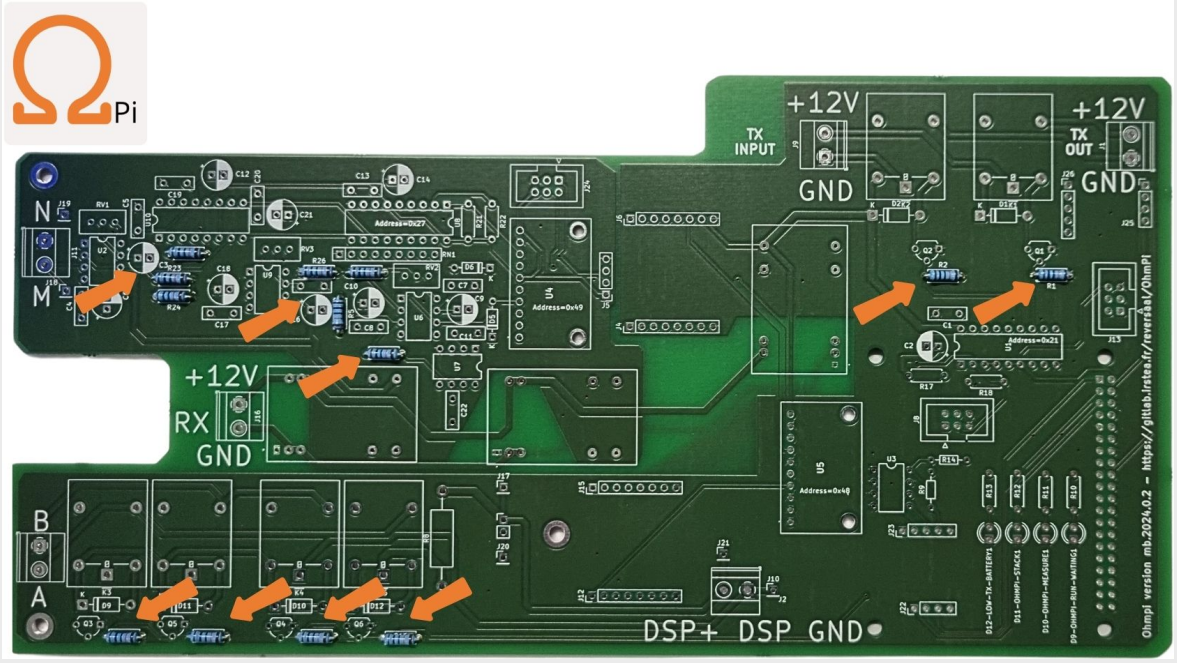
Before starting: how to soldering

How to Solder Electronic Components <https://www.sciencebuddies.org/science-fair-projects/references/how-to-solder>

Description

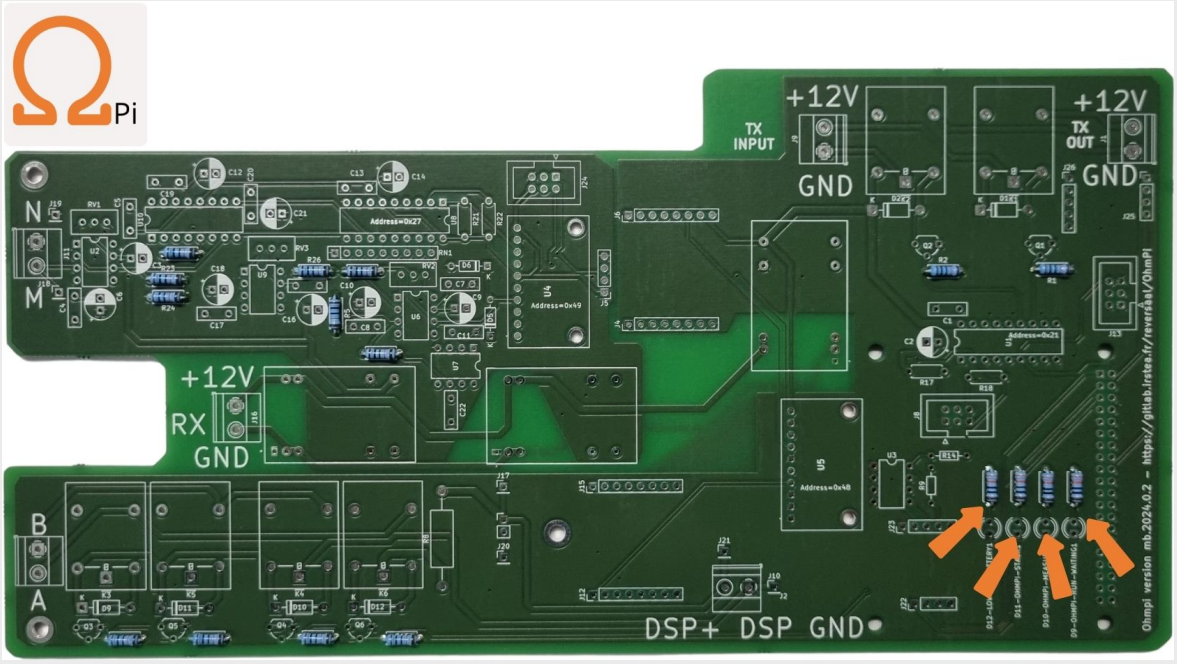
Soldering various RESISTOR on the measurement board PCB

1

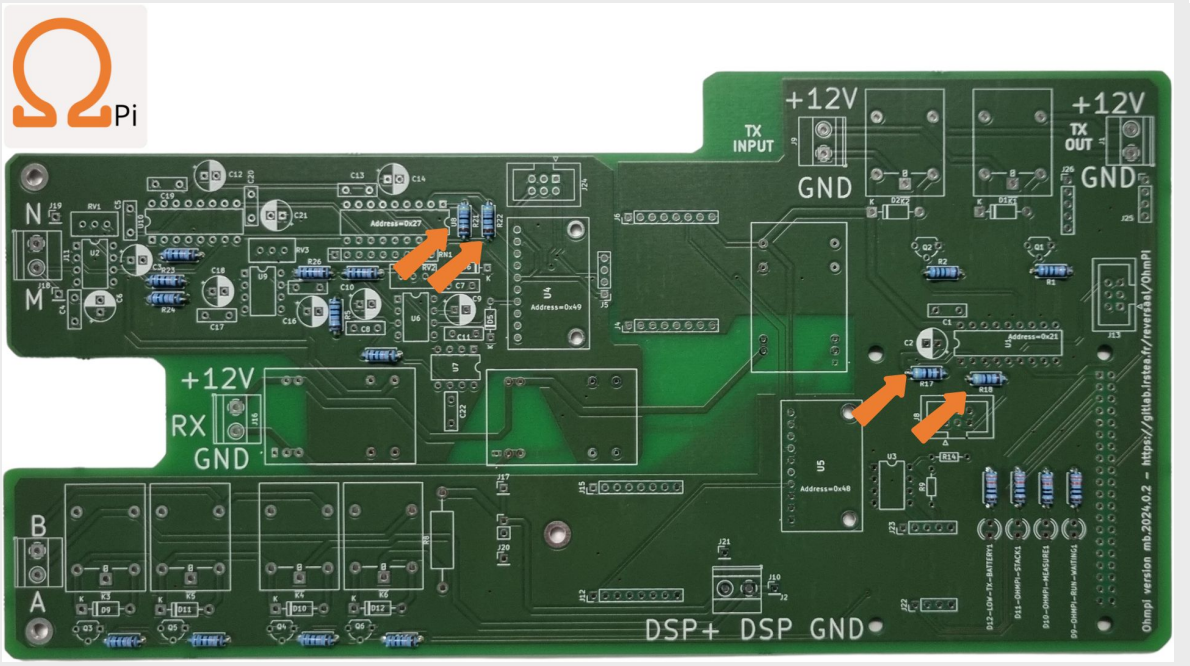


Soldering thirteen 100 kOhm resistors

2

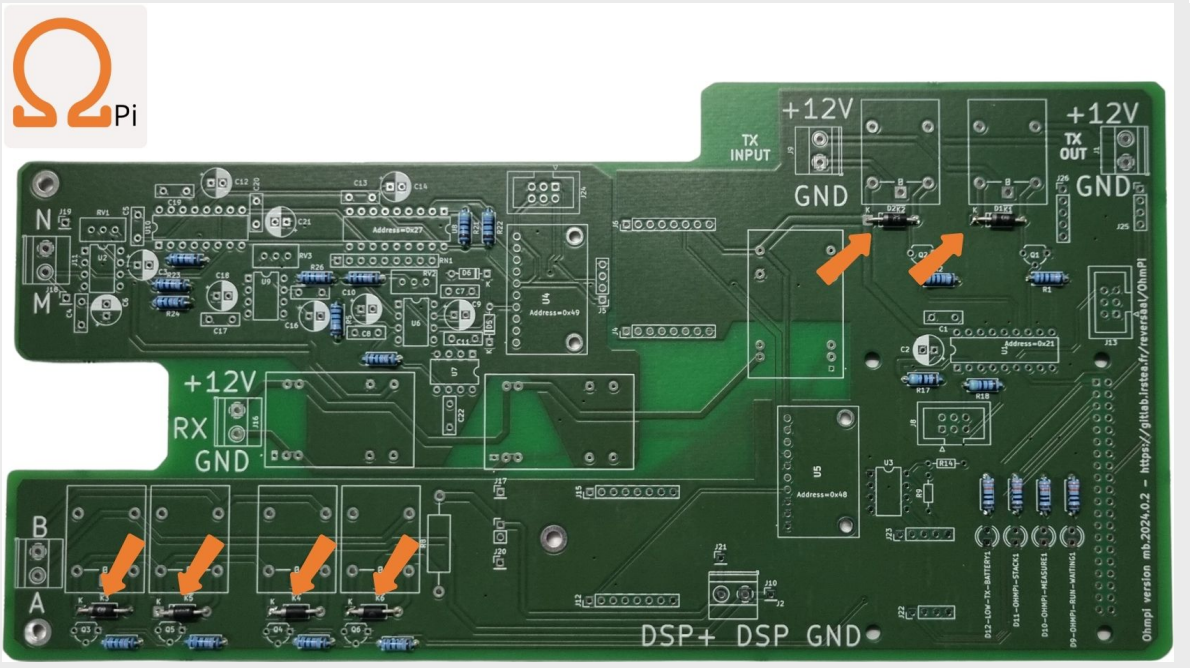


Soldering four 330 ohm resistors



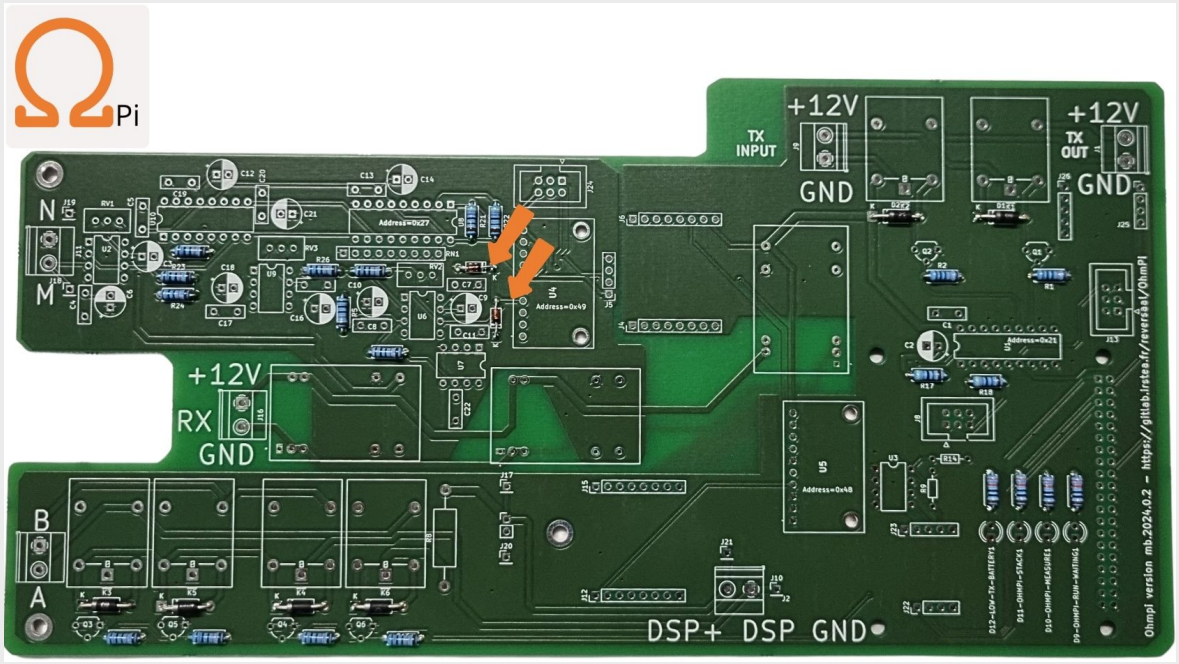
3 Soldering four 4K7 ohm resistors

Soldering DIODE on the measurement board PCB

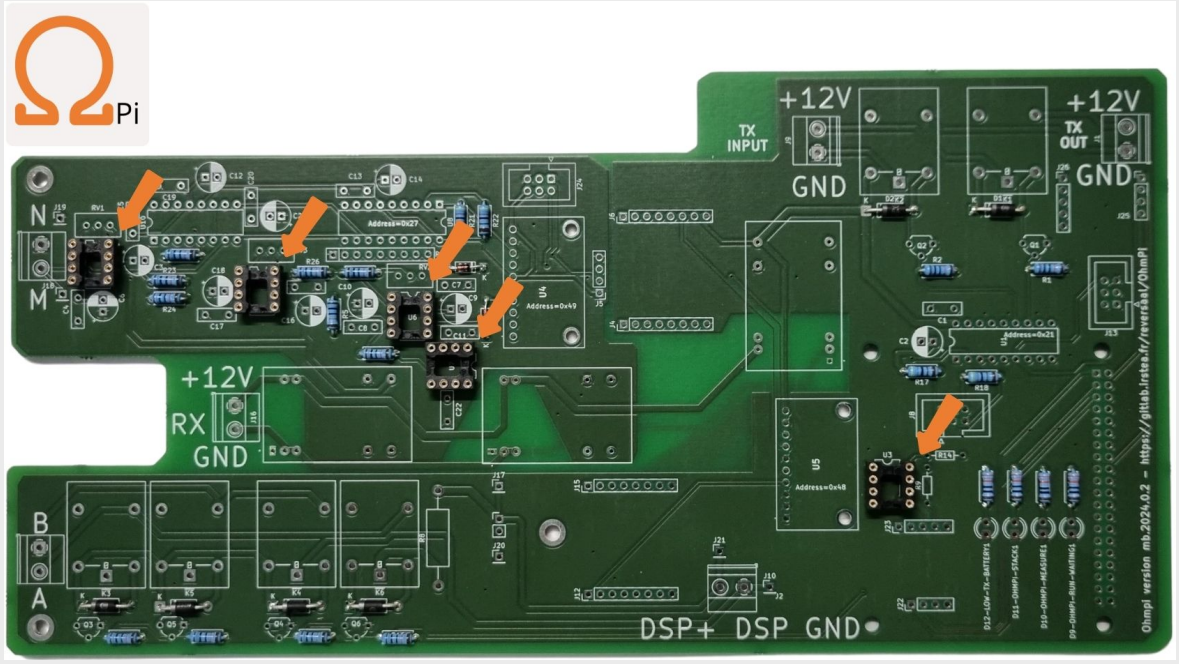


4 Soldering six diodes 1N4007

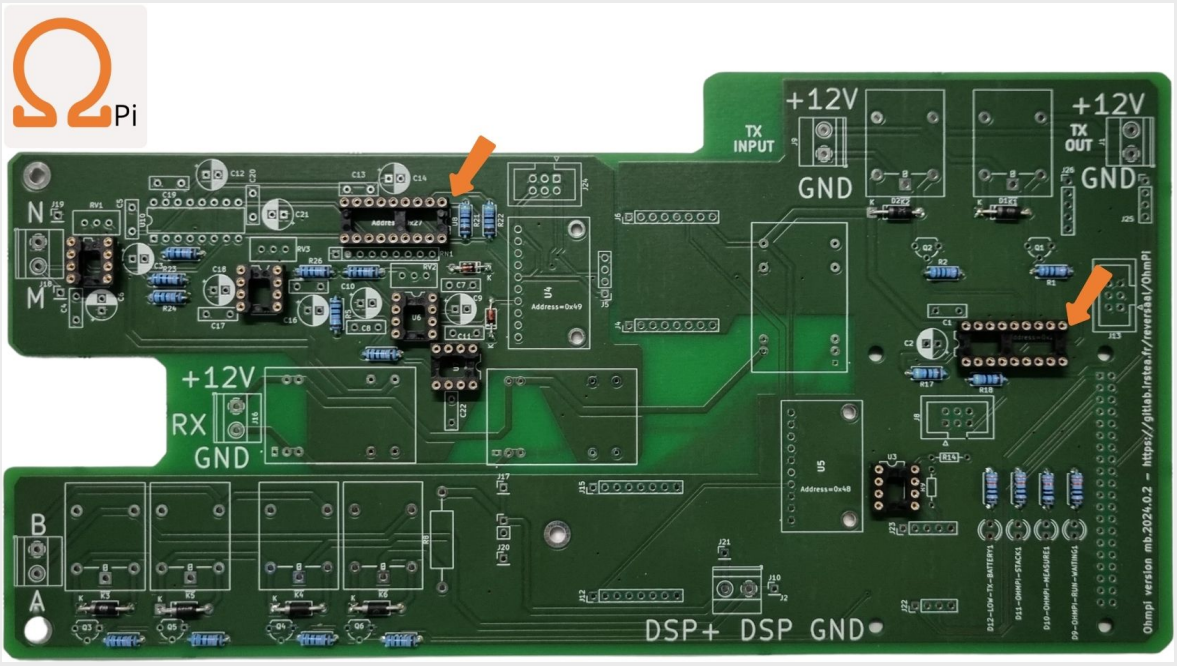
Soldering SCHOTTKY DIODE on the measurement board PCB



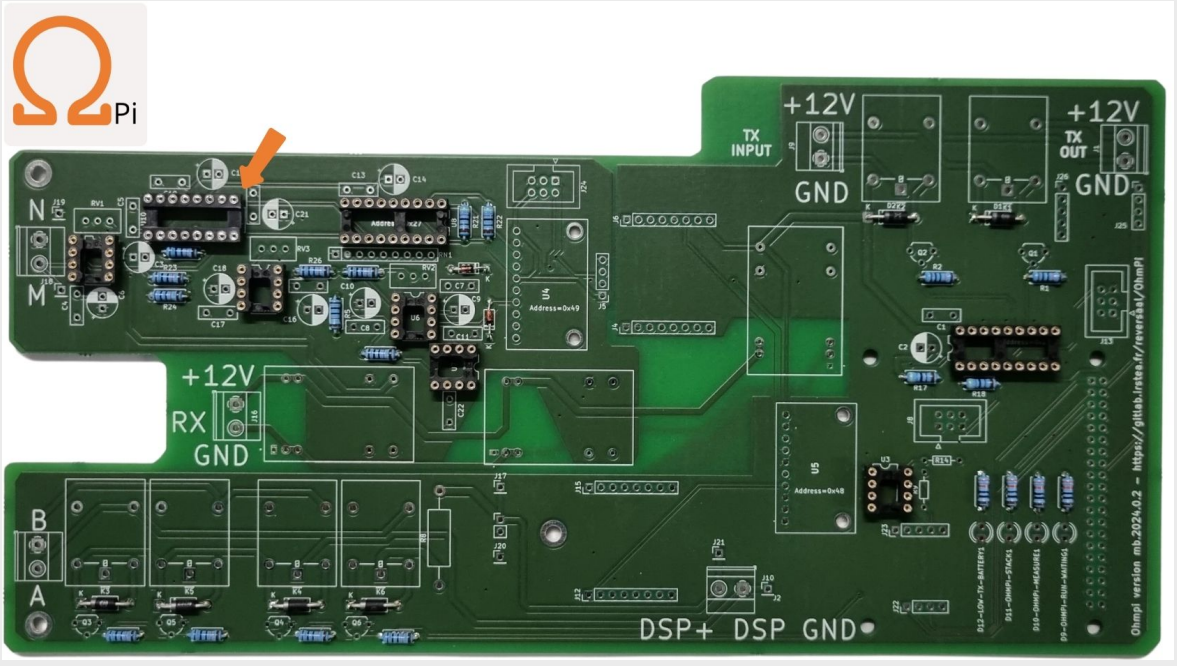
5 Soldering two Schottky diodes bat85 ou bat86



6 Soldering five DIP-8 sockets

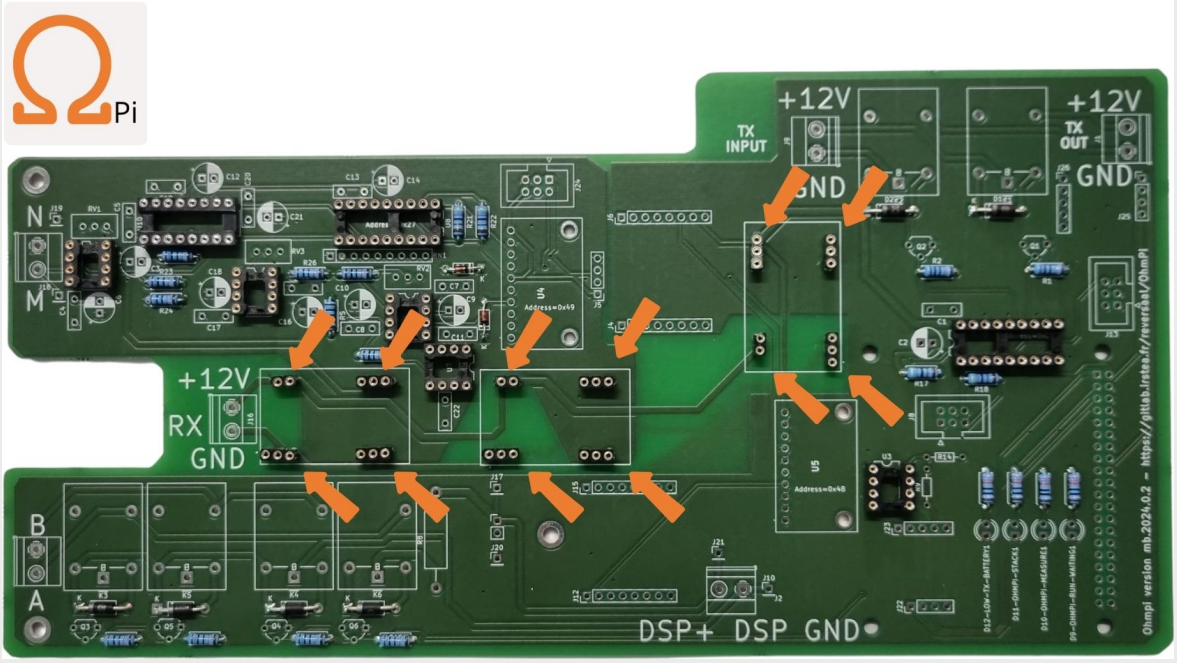


7 Soldering two DIP-18 sockets



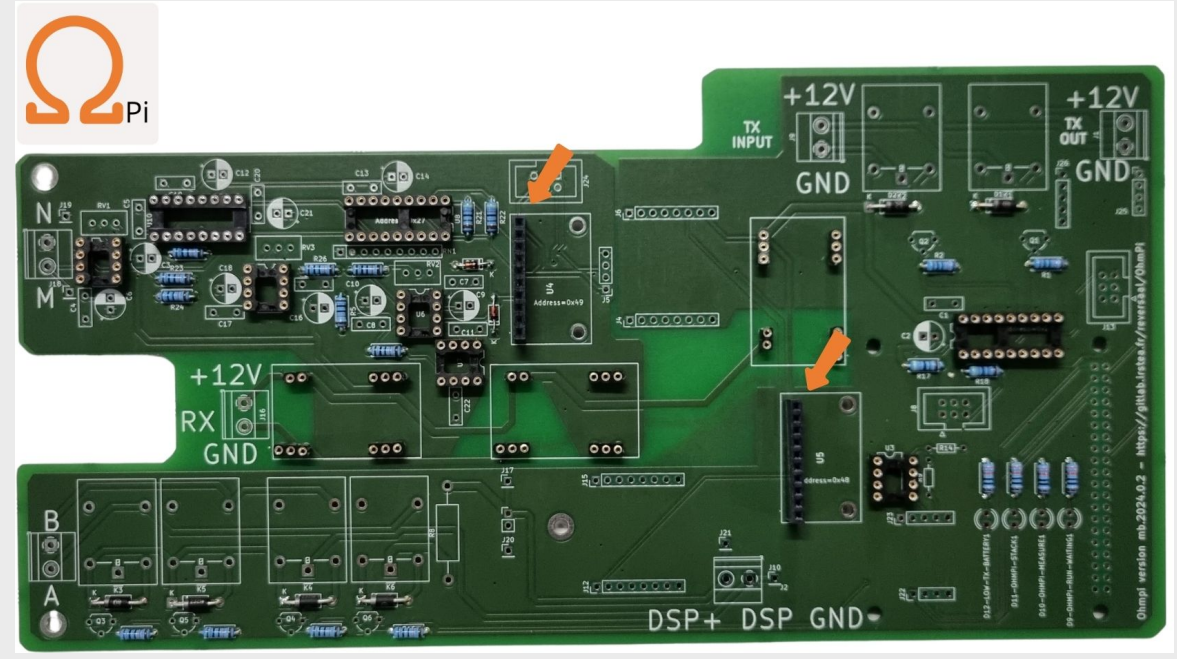
8 Soldering one DIP-16 sockets

9

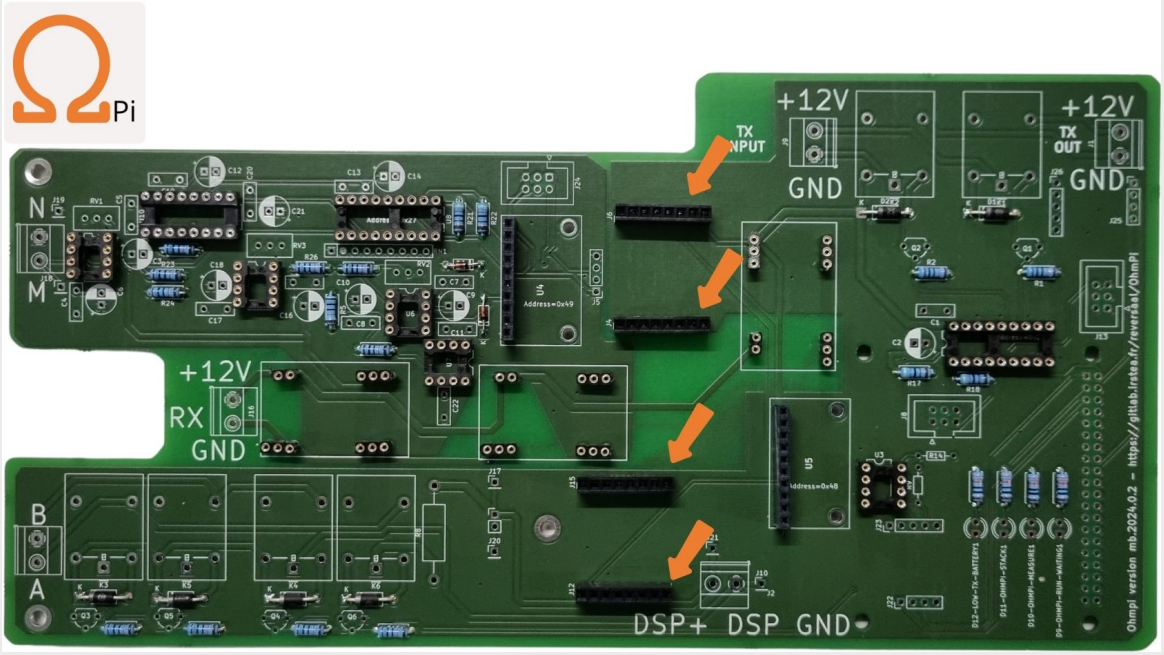


Soldering twelve cut sockets for 3 THD

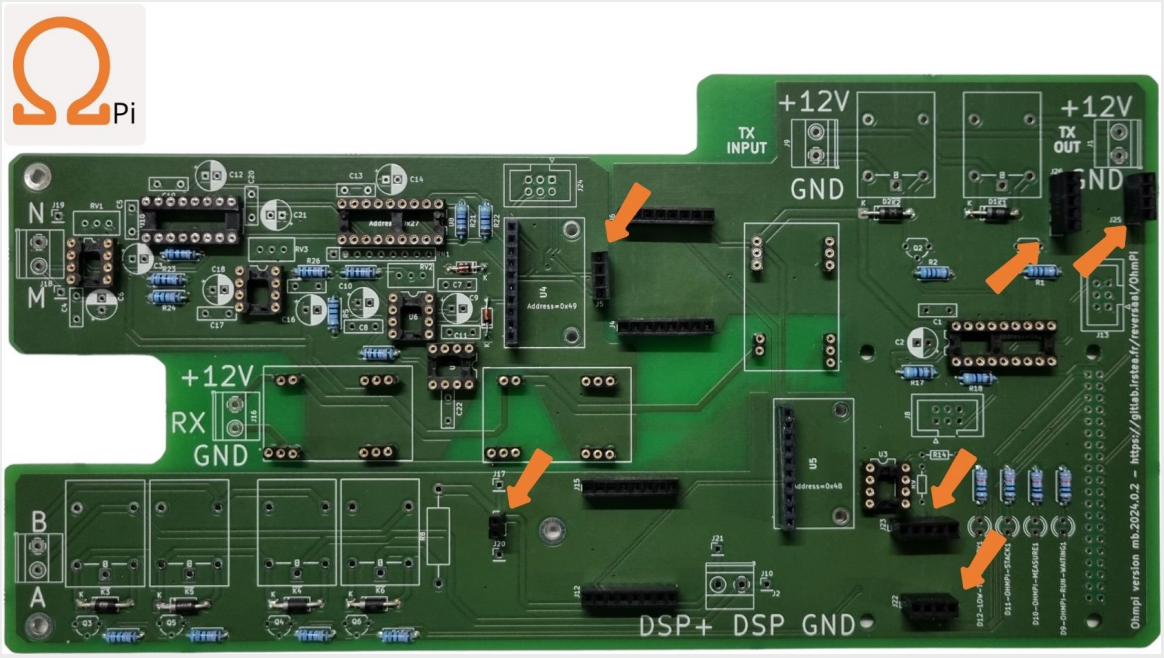
10



Soldering header socket 1 row 10 positions

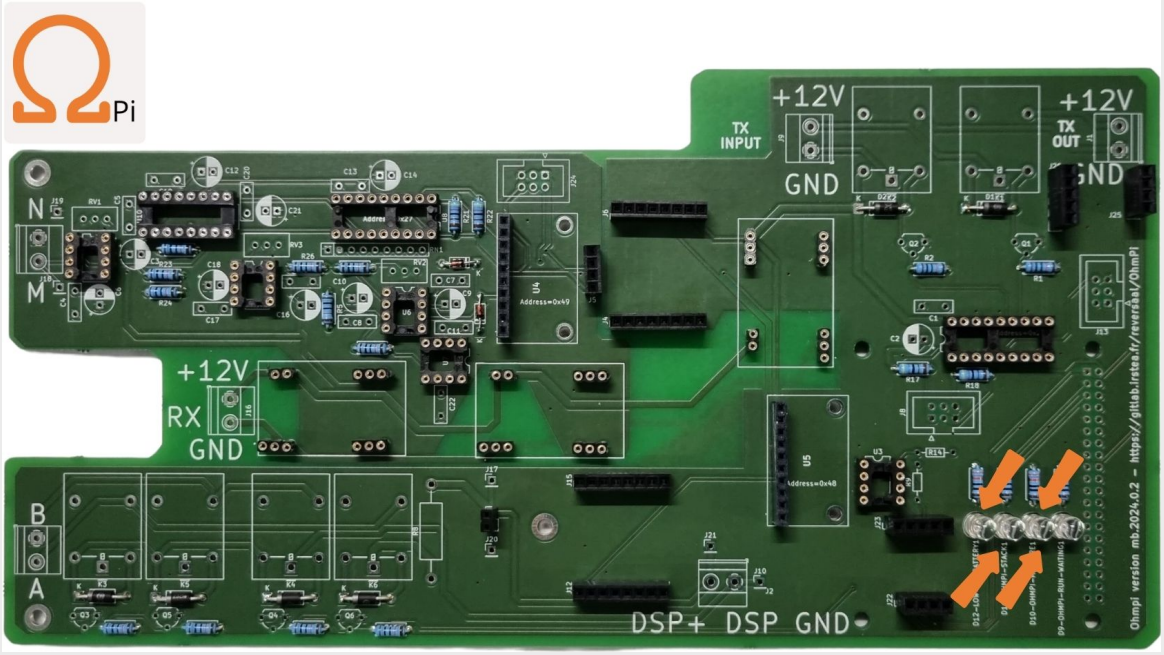


11 Soldering two header sockets with 1 row and 8 positions



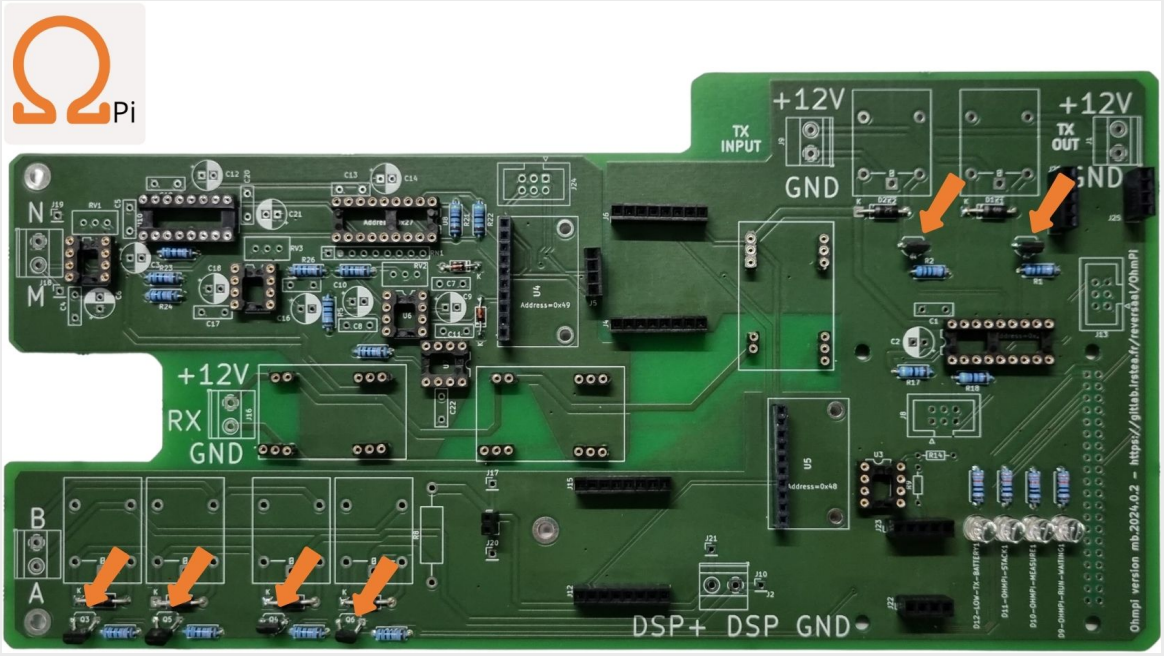
12 Soldering 1 header (1 row, 2 positions -> cut a bigger one), 3 * 1r4p and 2 * 1r5p.

Information about light-emitting diode



13 Installation of four light-emitting diodes

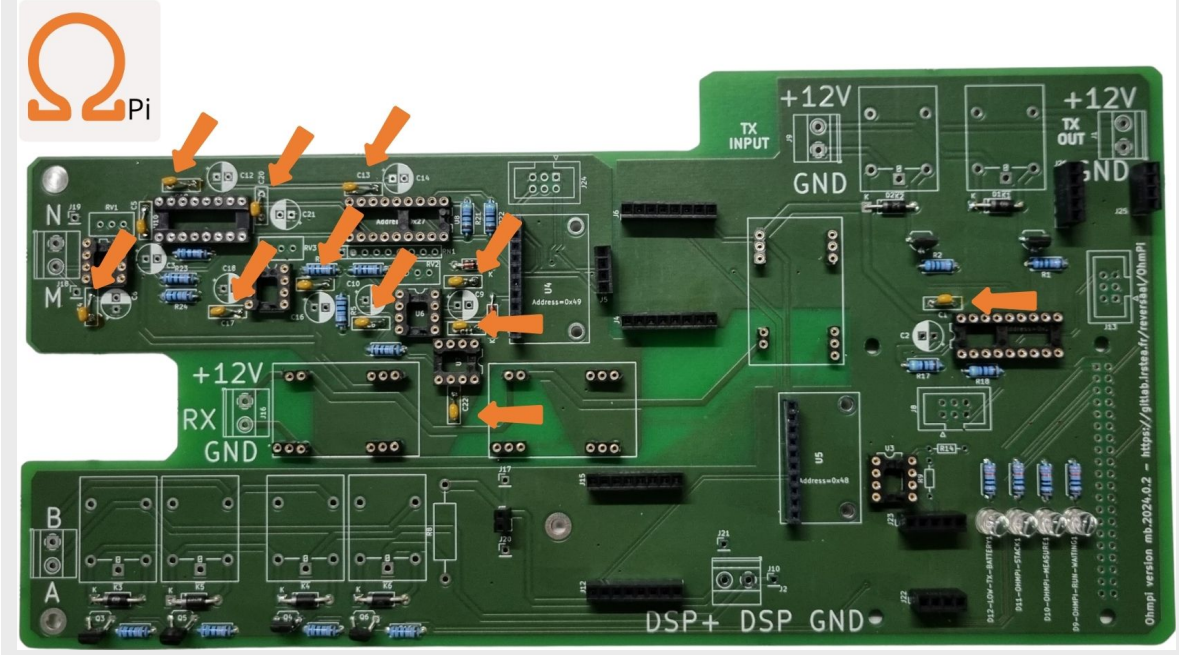
Information about MOSFET Metal Oxide Semiconductor Field Effect Transistor



14 Soldering six MOSFET ZVN4206 or ZVN4306

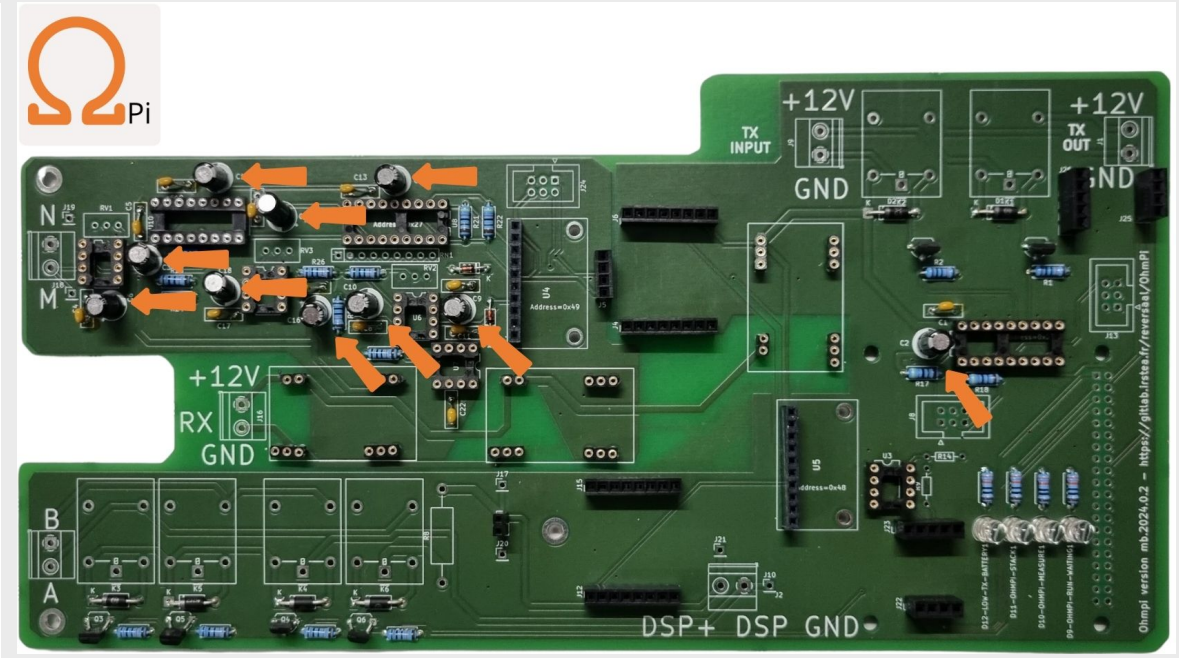
What is a CAPACITOR?

15



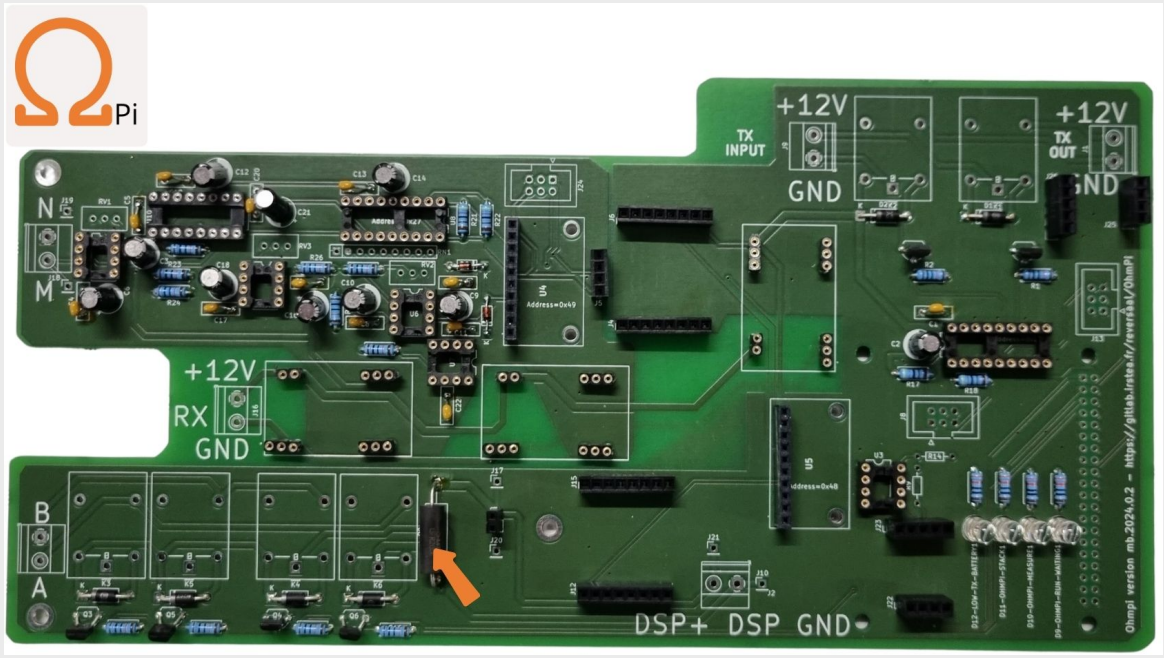
Soldering eleven 100 nF 50V tantalum capacitors

16

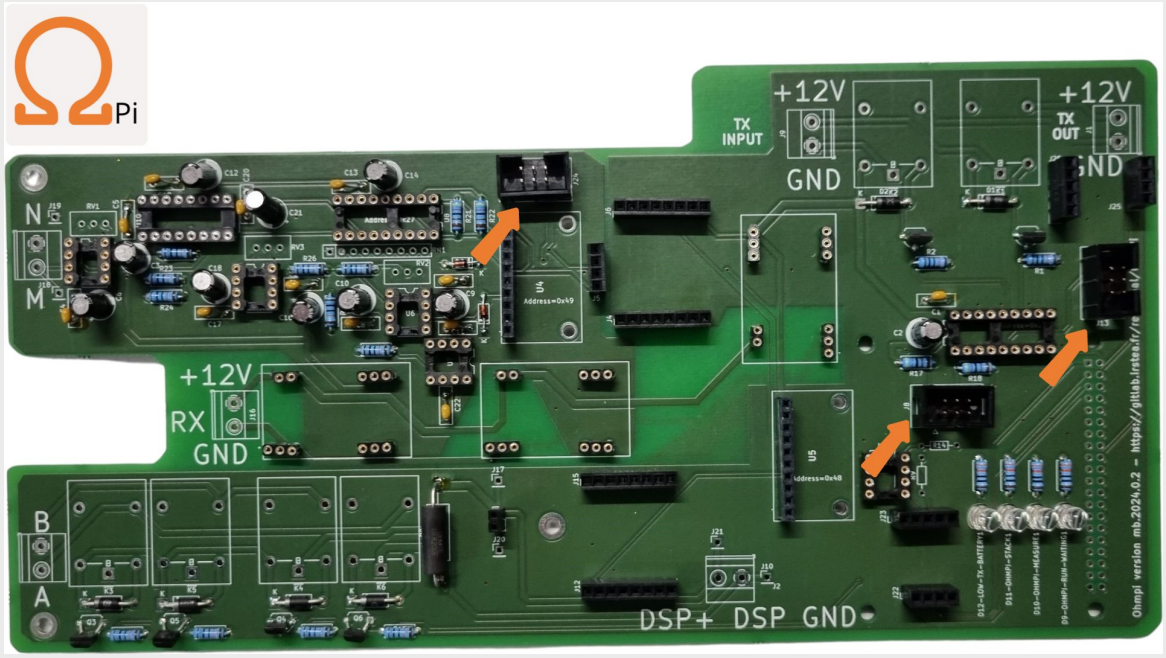


Soldering ten 10 µF 50V Electrolytic capacitors, **pay attention** to capacitor polarity

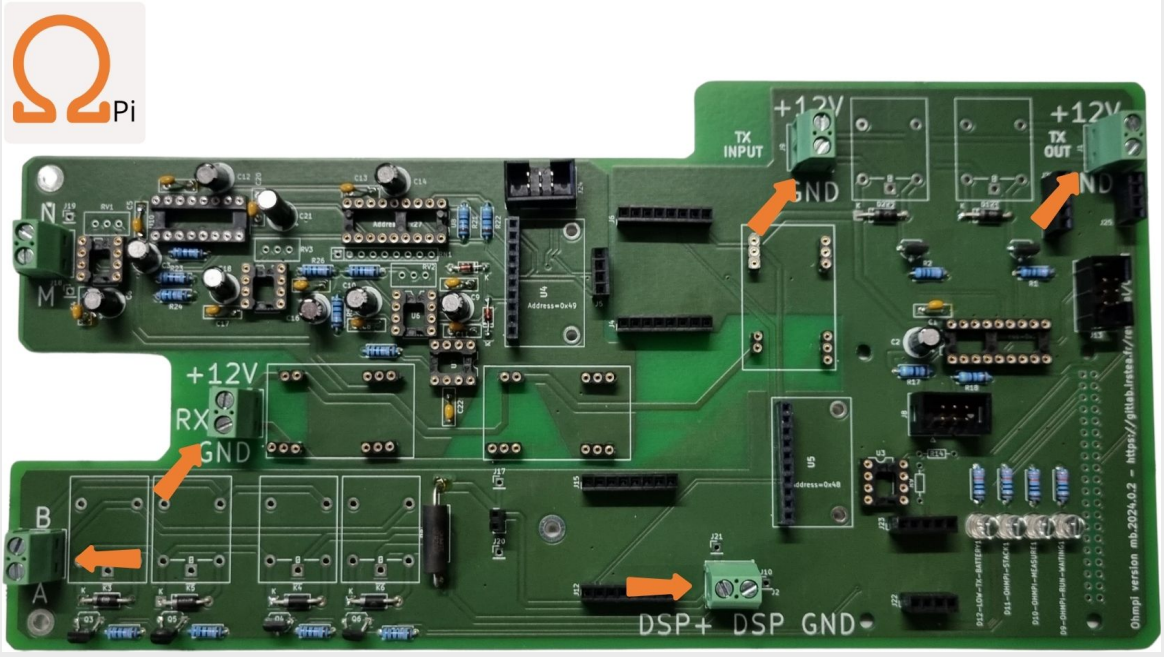
Warning: In this version, we used a shunt resistor of 2 ohms, which limits the current measurement to 48 mA. If the current is higher than this value, you just have to decrease the value of the shunt resistor. Don't forget to change the shunt value in the config.py file (value associated to key 'R_shunt' in the OHMPI_CONFIG dict).



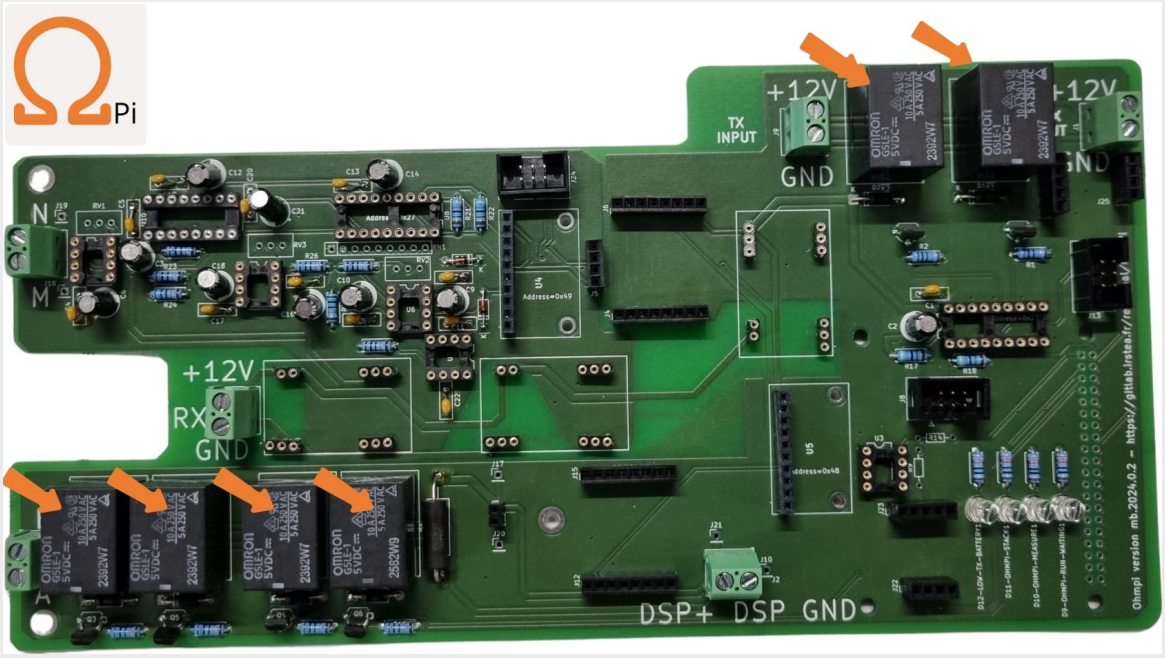
17 Soldering the 2 ohms shunt resistor



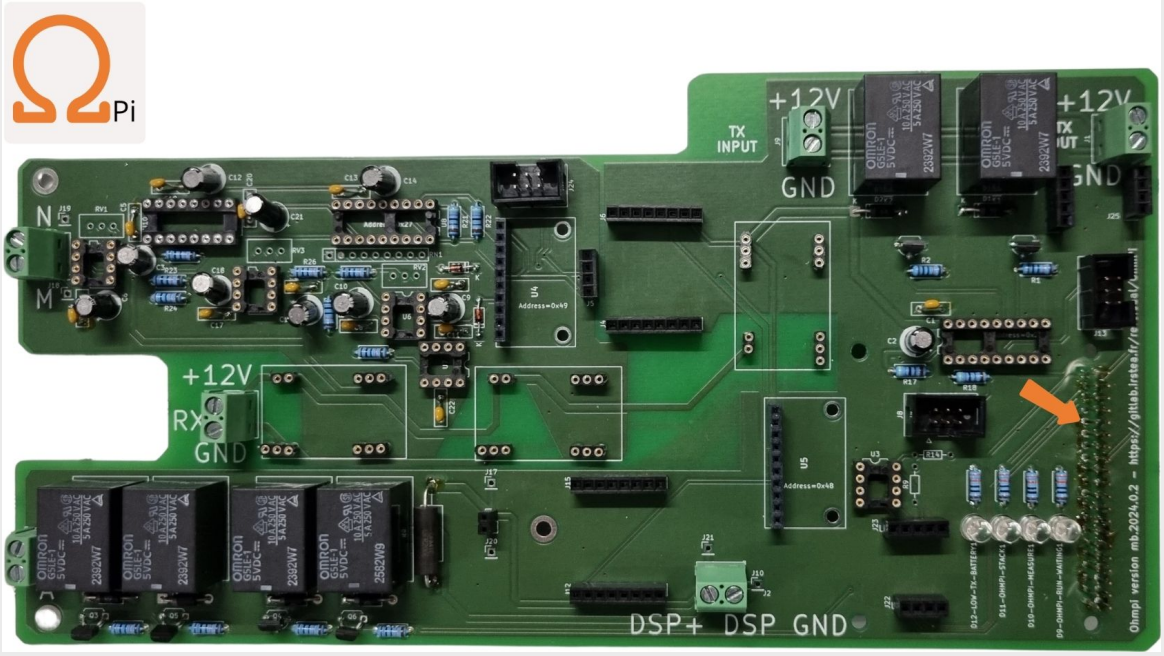
18 Soldering the two IDC 6 pins connectors. **pay attention to the connectors orientation**



19 Soldering six screw terminals for cable connection



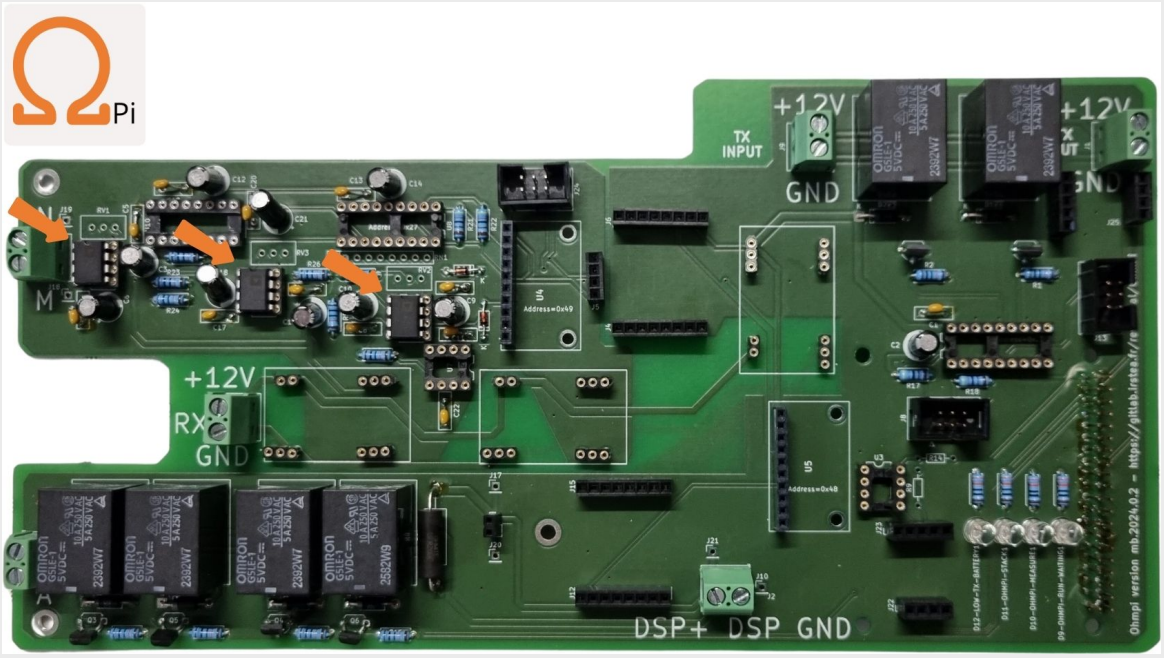
20 Soldering six omron G5LE relays 5 VDC



21 Soldering the 2x20 header for connection with the raspberry GPIO

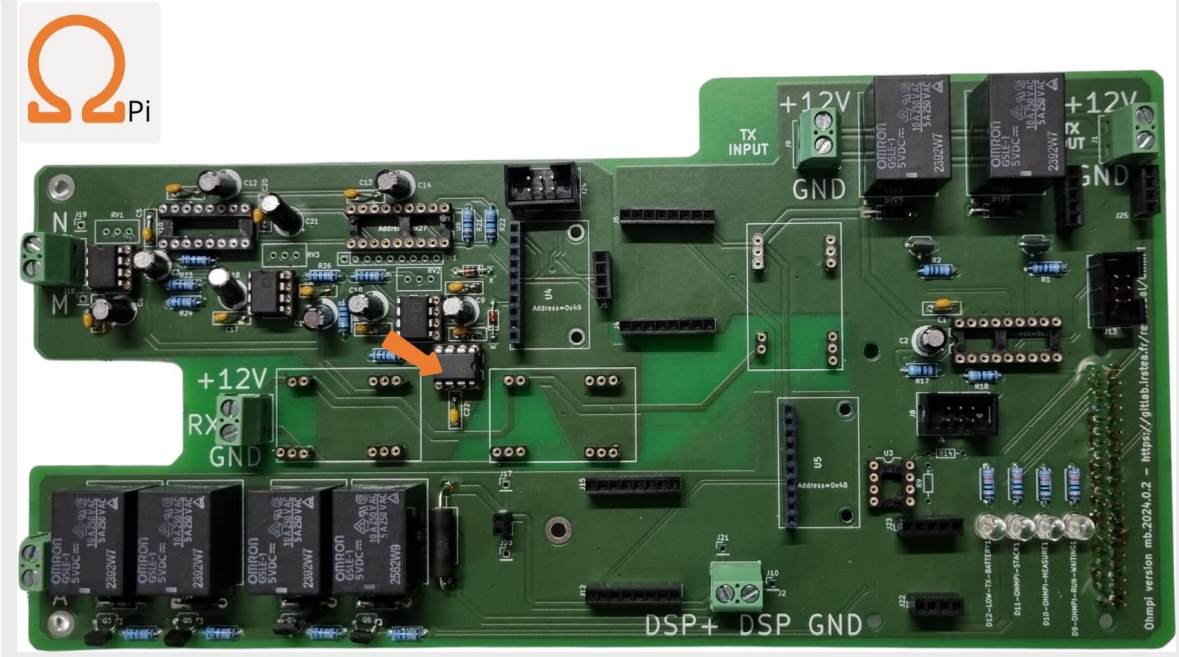
What is a Op-Amp?

In addition, the notch provides a way to visually identify the orientation of the package.



22 Place the three OP27 on their DIP-8 sockets **the notch must face upwards**

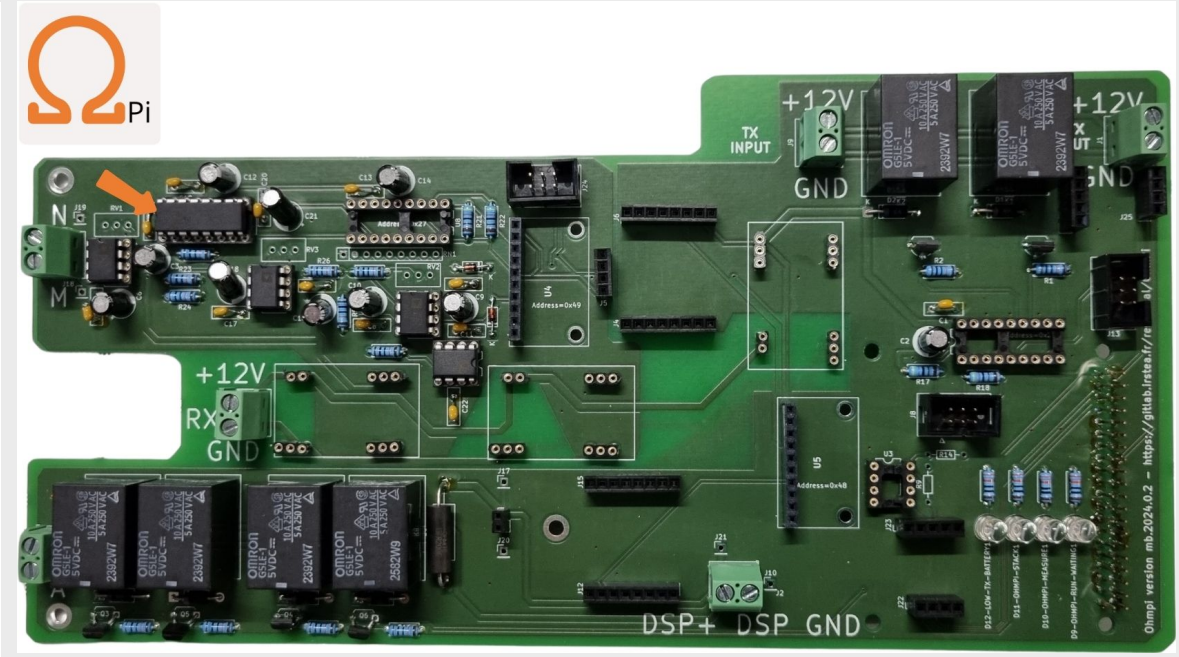
23



Place the REF03 reference voltage (2.5v) on its DIP-8 socket **the notch must face the right side**

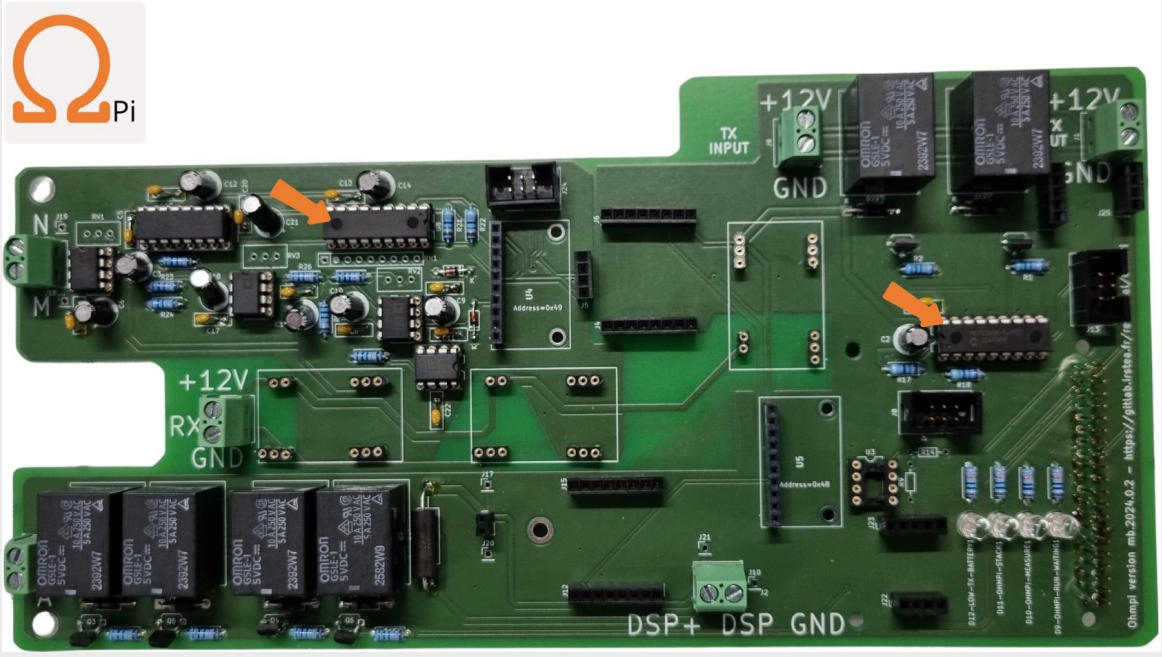
What is an analogue switch?

24



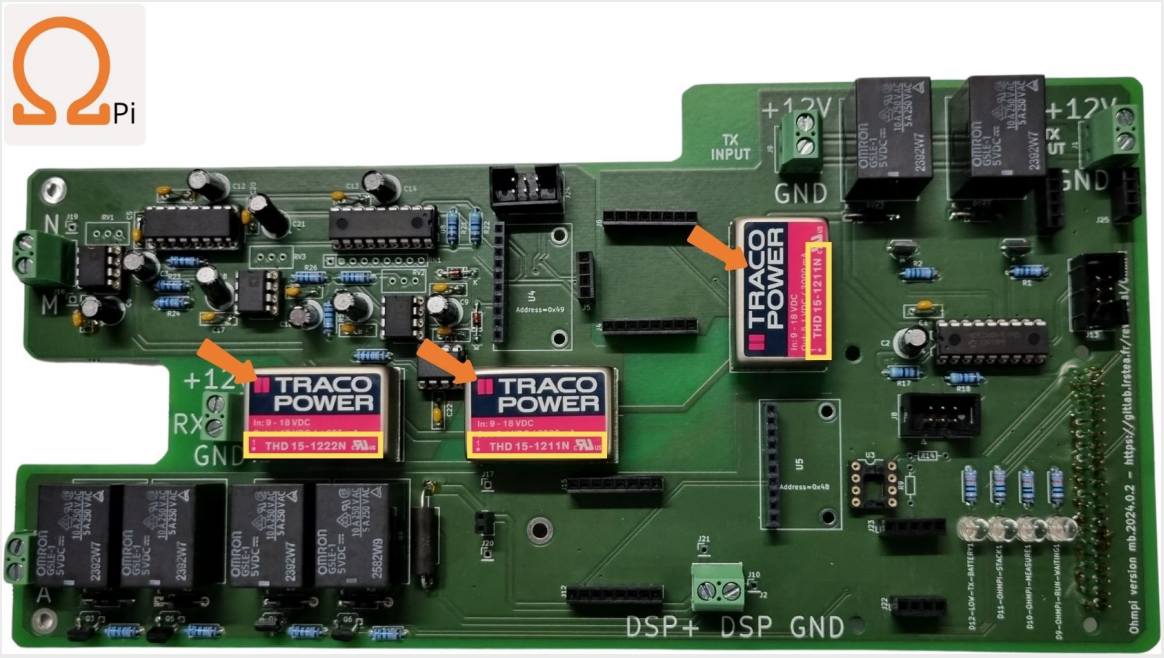
Place the DG411 **the notch must face the left side**

25

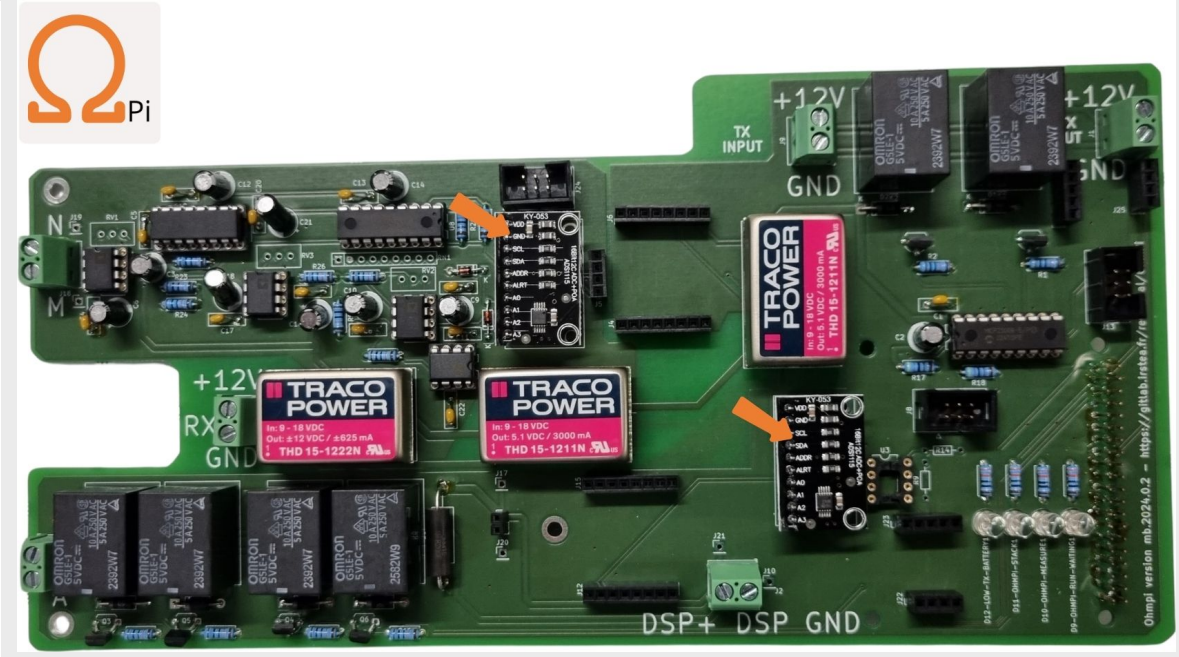


Place the MCP23008 on its DIP-16 socket **pay attention to the notches orientation**

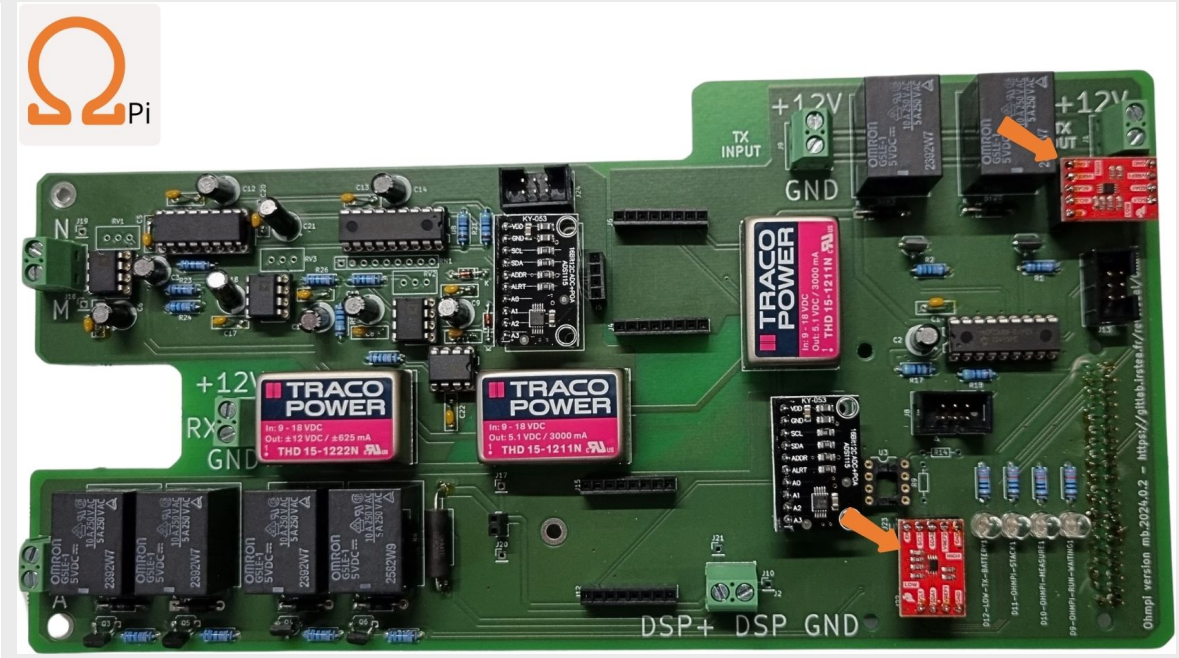
26



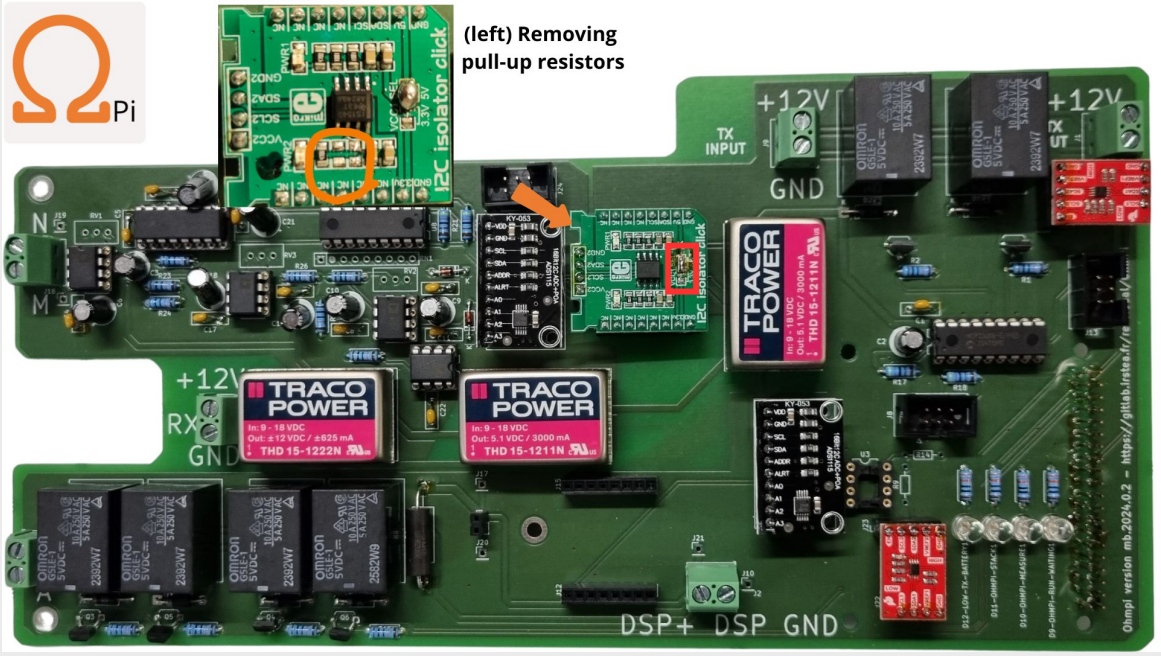
Place the Three THD, install the right reference at the right place according to the yellow boxes

27 

Place the ADS1115 board on its female header 1x10 pins

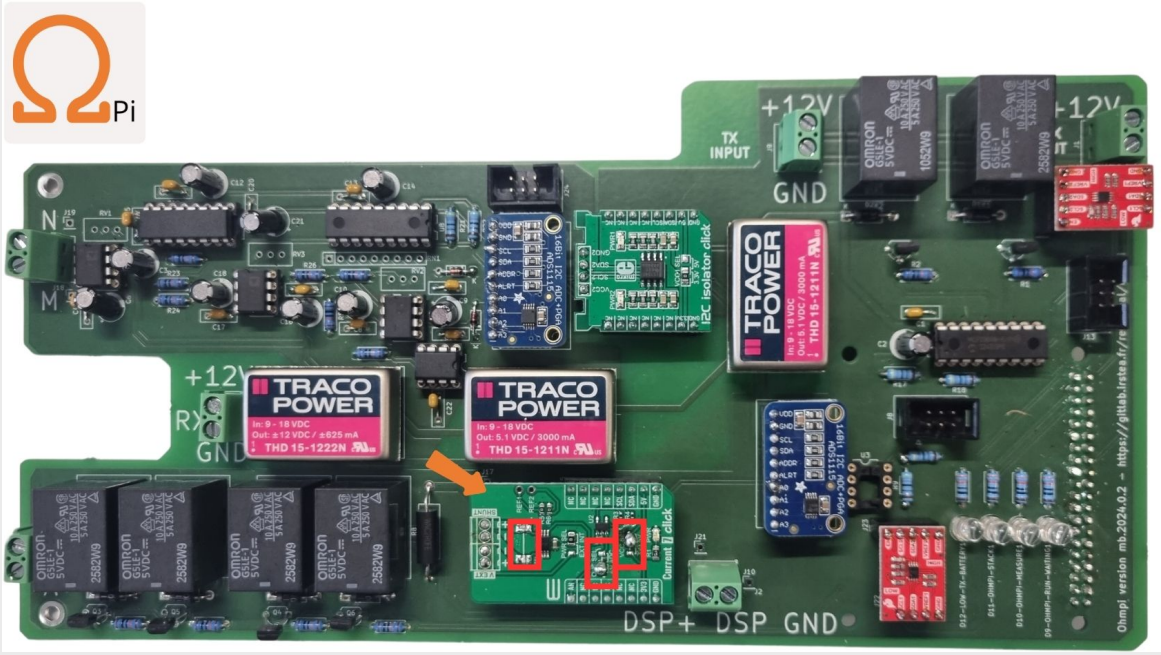
28 

Place the two I2C level adjusters



29 Place the I2C isolator add-on board **make sure you have right selection** according to the red box

Note: If you have issues with the I2C isolator (e.g. 0x49 and 0x27 are not visible), you may need to remove the pull-up resistor on the I2C isolator as shown above.



30 Place the current click add-on board **make sure you have right selections** according to the red boxes

Check measurement board v2024

Use the picture and table below to manually check with a multimeter for continuity and expected voltage in the measurement board.

If a continuity check does not pass it's likely means there is an issue with the soldering on the board. If the voltage with I2C (SDA and SCL pins) is not expected, there is likely an issue with pull-up resistors.

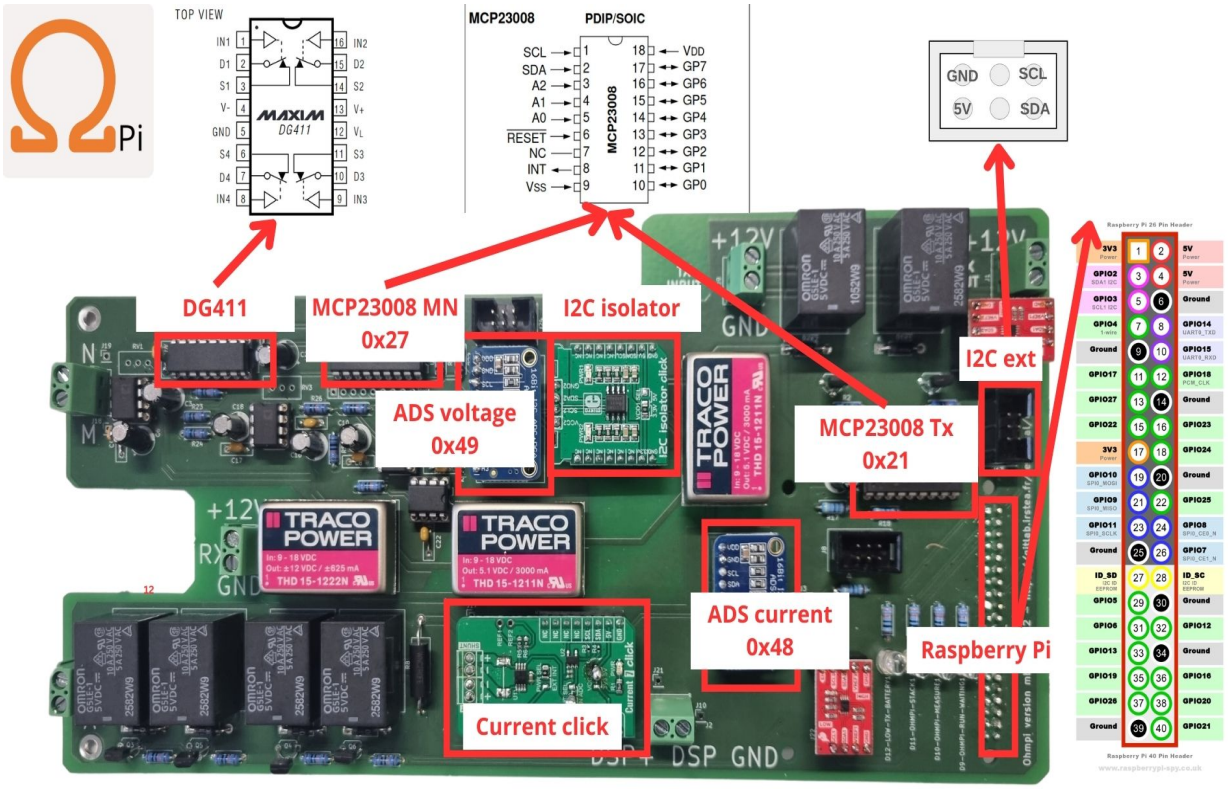


Table 4: Hardware check

Name	Power	Type	Multimeter BLACK probe	Multimeter RED probe	Expected
SC1	off	continuity	screw terminal Tx in GND	screw terminal Tx in +12V	no continuity
SC2	off	continuity	screw terminal Tx out GND	screw terminal Tx out +12V	no continuity
SC3	off	continuity	screw terminal A	screw terminal B	no continuity
SC4	off	continuity	screw terminal M	screw terminal N	no continuity
SC5	off	continuity	screw terminal DPS -	screw terminal DPS +	no continuity
SC6	off	continuity	ADS current 0x48 GND	ADS current 0x48 VDD	no continuity
SC7	off	continuity	ADS voltage 0x49 GND	ADS voltage 0x49 VDD	no continuity

Warning: Do not power the board if one of the SC (shortcircuit) test does not pass!

Table 5: Hardware check

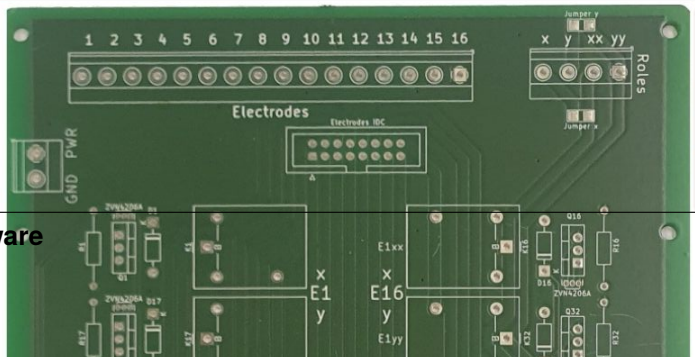
Name	Power	Type	Multimeter BLACK probe	Multimeter RED probe	Expected
C1	off	continuity	ADS current 0x48 GND	Current click GND	continuity
C2	off	continuity	ADS current 0x48 GND	Raspberry Pi GND	continuity
C3	off	continuity	ADS current 0x48 GND	MCP23008 Tx 0x21 VSS	continuity
C4	off	continuity	ADS current 0x48 GND	I2Cext GND	continuity
C5	off	continuity	screw terminal N	I2C isolator GND2	continuity
C6	off	continuity	screw terminal N	ADS voltage 0x49 GND	continuity
C7	off	continuity	screw terminal N	DG411 GND	continuity
C8	off	continuity	screw terminal N	MCP23008 MN 0x27 VSS	continuity
C9	off	continuity	ADS voltage 0x49 VDD	I2C isolator VDD2	continuity
C10	off	continuity	ADS voltage 0x49 VDD	MCP23008 VSS	continuity
C11	off	continuity	ADS voltage 0x49 SDA	I2C isolator SDA2	continuity
C12	off	continuity	ADS voltage 0x49 SDA	MCP23008 Tx 0x21 SDA	continuity
C13	off	continuity	ADS voltage 0x49 SCL	I2C isolator SCL2	continuity
C14	off	continuity	ADS voltage 0x49 SCL	MCP23008 Tx 0x21 SCL	continuity
C15	off	continuity	ADS current 0x48 SDA	Current click SDA	continuity
C16	off	continuity	ADS current 0x48 SDA	Raspberry Pi SDA1 (p3)	continuity
C17	off	continuity	ADS current 0x48 SDA	MCP23008 Tx 0x21 SDA	continuity
C18	off	continuity	ADS current 0x48 SDA	ADS voltage 0x49 SDA	no continuity
C19	off	continuity	ADS current 0x48 SCL	Current click SCL	continuity
C20	off	continuity	ADS current 0x48 SCL	Raspberry Pi SCL1 (p5)	continuity
C21	off	continuity	ADS current 0x48 SCL	MCP23008 Tx 0x21 SCL	continuity
C22	off	continuity	ADS current 0x48 SCL	ADS voltage 0x49 SCL	no continuity
C23	off	continuity	ADS current 0x48 VDD	Current click VDD	continuity
C24	off	continuity	ADS current 0x48 VDD	Raspberry Pi VDD (p2)	continuity
C25	off	continuity	ADS current 0x48 VDD	MCP23008 VDD	continuity
C26	off	continuity	ADS current 0x48 VDD	I2Cext 5V	continuity
V1	on	voltage	screw terminal Tx in GND	screw terminal Tx in +12V	12V
V2	on	voltage	screw terminal Rx GND	screw terminal Rx +12V	12V
V3	on	voltage	ADS current 0x48 GND	ADS current 0x48 VDD	5V
V4	on	voltage	ADS current 0x48 GND	ADS current 0x48 SDA	5V
V5	on	voltage	ADS current 0x48 GND	ADS current 0x48 SCL	5V
V6	on	voltage	ADS current 0x48 GND	I2Cext SDA	5V
V7	on	voltage	ADS current 0x48 GND	I2Cext SCL	5V
V8	on	voltage	screw terminal N	ADS voltage 0x49 VDD	5V
V9	on	voltage	screw terminal N	ADS voltage 0x49 SDA	5V
V10	on	voltage	screw terminal N	ADS voltage 0x49 SCL	5V

1.2.3 Multiplexer board

Measurement board are limited to four electrodes (A, B, M and N). Multiplexer board are composed or electrical relay (electronic switches) that enable to route the signal from A, B, M or N from the measurement board towards the specified electrodes. Multiplexer boards are needed for a multi-electrode system.

Recognize the version of the MUX boards

Measurement board V2023.0.1



Specifications

Parameters	v2023.X.X	v2024.X.X
Number of electrodes per board	64	16 (or 8)
Number of roles (A, B, M or N) per board	1	2 roles (or 4)
Power supply	12 V	12V (or 5V)

Assemble multiplexer board (MUX)

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

Build MUX board v2023

The multiplexing of the channels is a mechanical multiplexing based on OMRON's manufacturing relays (G5LE-1-VD 12 VDC). Each relay is combined with a ZVN4206A power MOFSET. The raspberry has only 30 GPIOs, which is not enough to activate all the 64 electrodes, which represent 512 GPIOs. We used gpio expander I2C (MCP23017). We have associated these components with an I2C multiplexer of type type TCA9548A from adafruit. This combination allows to go up to 512 GPIOs and up to 128 electrodes. Each card has its own digital address between 0X70 and 0X77. In the following presentation for an OhmPi 64 electrodes, we will use the addresses 0X70 for channel A, 0X71 for channel B, 0X72 for channel M and 0X73 for channel N. 0X73 for the N channel. 4 MUX board will be needed to multiplex an OhmPi 64 electrodes.

PART A Assembly of MUX board

Required components

Table 6: List of components

Components	Number	Cost per unit	Total cost	Manufacturer	Manufacturer s reference	Web reference
Printed circuit board	4	140	560	Aisler	•	https://aisler.net/
Pin strip no ejector 16 pins	16	0.62	9.92	BLK electronic	10120550	https://www.conrad.com/p/tru-components-1580994-pin-str

continues on next page

Table 6 – continued from previous page

Components	Number	Cost per unit	Total cost	Manufacturer	Manufacturer s reference	Web reference
diode-1n4007	256	0.091	23.296	Diodes Incorporated	1N4007-T	https://www.mouser.fr/ProductDetail/Diodes-Incorporated/1N4007-T?qs=sGAEpiMZZMueQxo7L%2FBPyAkbORUUMREn
Pin strip no ejector 6 pins	4	0.39	1.56	BLK electronic	10120550	https://www.conrad.com/p/tru-components-1580994-pin-str
Dual screw terminal (5.08-mm pitch)	12	0.648	7.776	CUI Devices	TB009-508-02BE	https://www.mouser.fr/ProductDetail/CUI-Devices/TB009-508-02BE?qs=vLWxofP3U2wCFk5uCkWTkA%3D%3D
Generic male header - 3 pins	12	0.205	2.46	TE Connectivity	4-103321-5	https://www.mouser.fr/ProductDetail/TE-Connectivity/4-103321-5?qs=5TwgZeq9E7HSYLqaljJYrw%3D%3D
MCP23017 I2C I/O Expander	16	2.5	40	Adafruit	732	https://www.mouser.fr/ProductDetail/Adafruit/732?qs=sGAEpiMZZMsKEdP9slC0Yfx13D
Omron G5LE-1-VD 12 VDC PCB relay 12 V DC 8 A 1	256	1.27	325.12	Omron	G5LE-1-VD 12 VDC	https://www.conrad.com/p/omron-g5le-1-vd-12-vdc-pcb-rel

continues on next page

Table 6 – continued from previous page

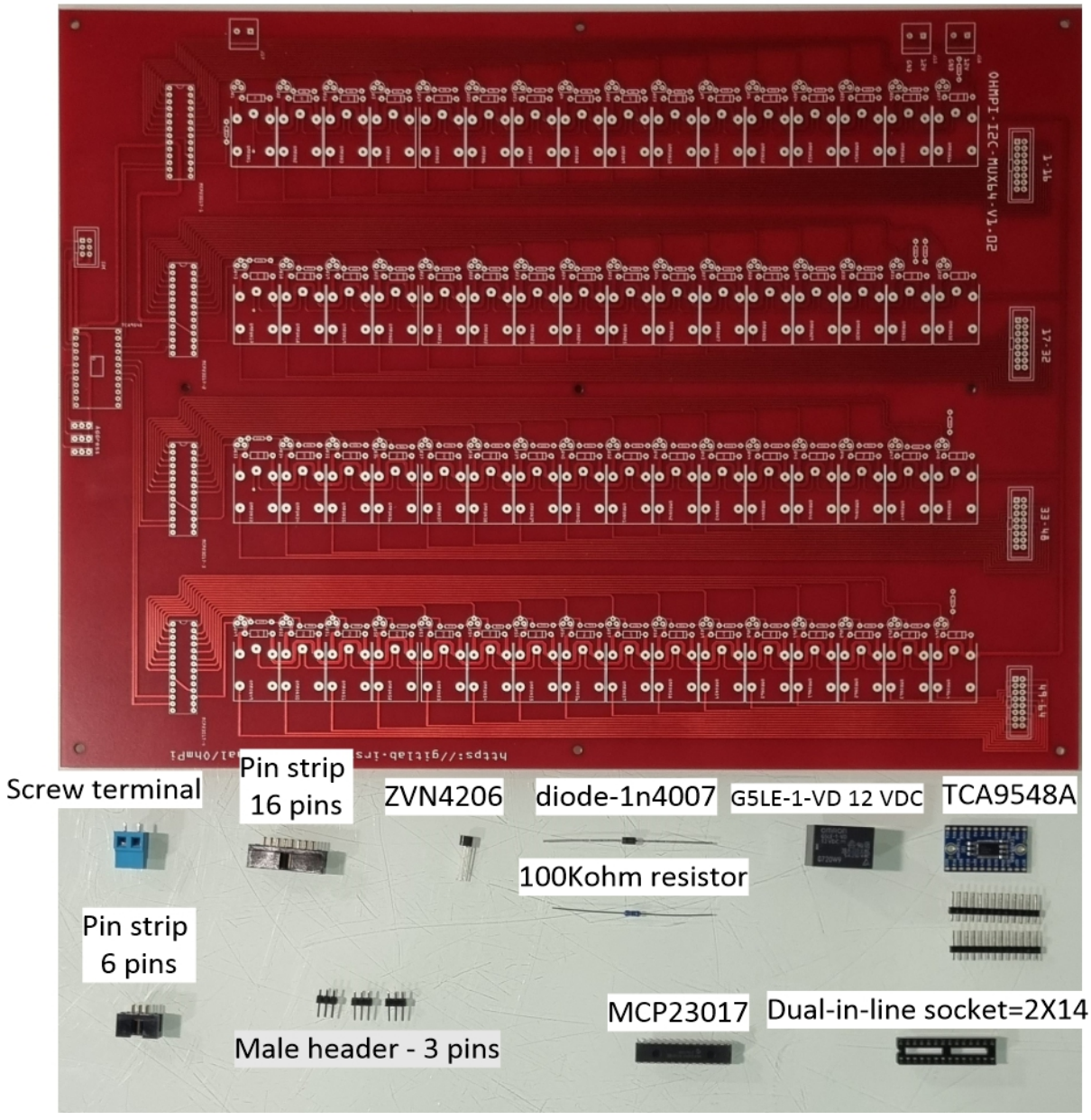
Components	Number	Cost per unit	Total cost	Manufacturer	Manufacturer s reference	Web reference
ZVN4206A MOSFET-NCHANNEL	256	0.471	120.576	Diodes Incorporated	ZVN4206A	https://www.mouser.fr/ProductDetail/Diodes-Incorporated/ZVN4206A?qs=vHuUswq2%252Bsz9b%2Ff6fcXt7g%3D%3D
100k Resistor	256	0.061	15.616	Vishay / Beyschlag	MBA02040C1(	https://www.mouser.fr/ProductDetail/Vishay-Beyschlag/MBA02040C1003FRP00?qs=mzRxyRlhVdt9crF7Zyf%2F5Q%3D%3D
Adafruit TCA9548A	4	5.89	23.56	Adafruit	2717	https://www.mouser.fr/ProductDetail/Adafruit/2717?qs=sGAEpiMZZMsyYdr3R27aV4E(252Baqg%252BZ3hVktao%3D
BKL Electronic 10120558 Pin strip no ejector Contact spacing: 2.54 mm Total number of pins: 16 No. of rows: 2 1 pc(s)	16	0.51	8.16	BLK electronic	10120558	https://www.conrad.com/p/bkl-electronic-10120558-pin-stri searchTerm=741727& searchType=suggest& searchSuggest=product

continues on next page

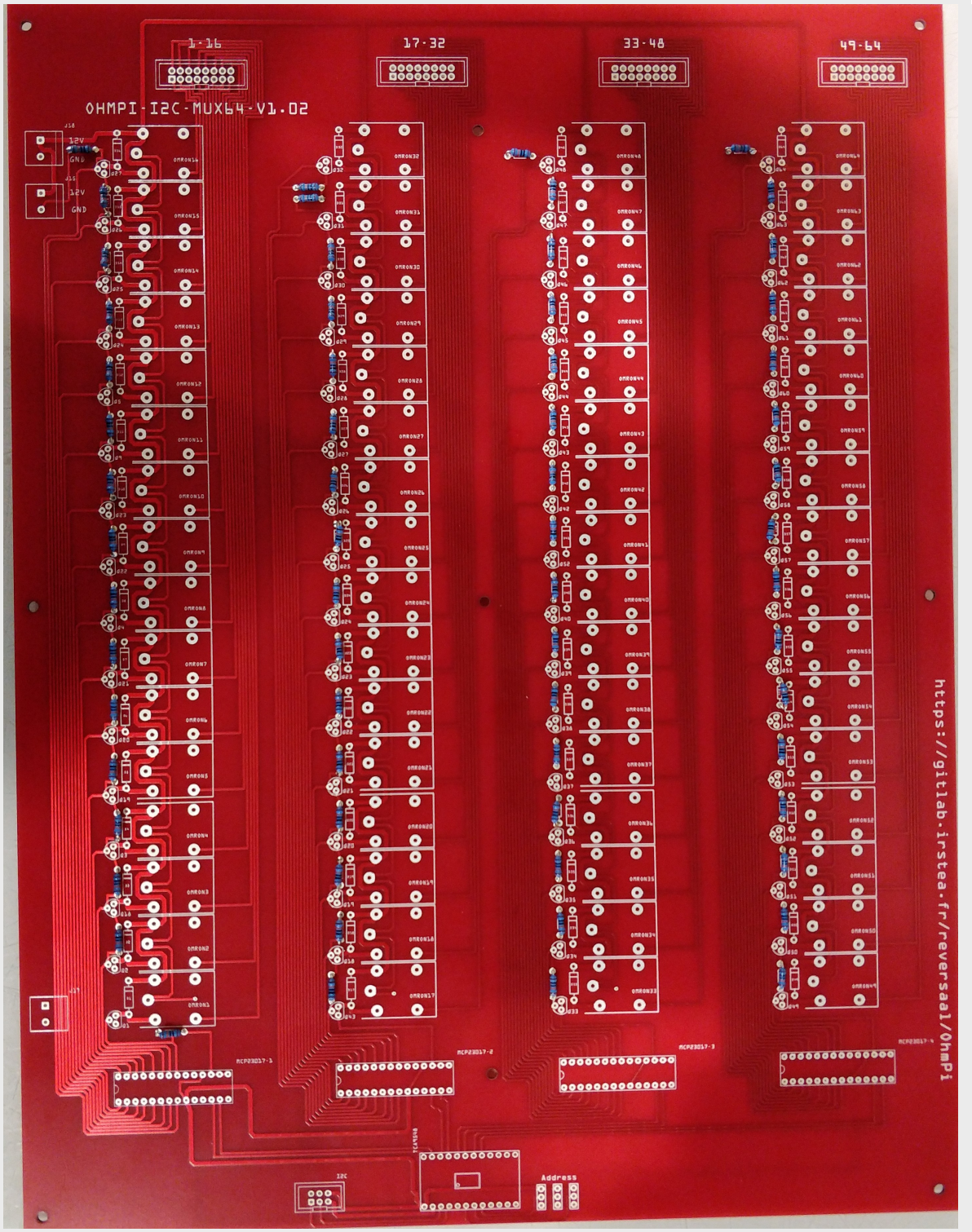
Table 6 – continued from previous page

Components	Number	Cost per unit	Total cost	Manufacturer	Manufacturer s reference	Web reference
BKL Electronic 10120862 Pin connector strain relief Contact spacing: 2.54 mm Total number of pins: 6 No. of rows: 2 1	16	0.84	13.44	BLK electronic	10120862	https://www.conrad.com/p/bkl-electronic-10120862-pin-con
BKL Electronic 10120158/10 Ribbon cable Contact spacing: 1.27 mm 16 x 0.08 mm Multi-coloured 10 m	1	27	27	BLK electronic	10120158/10	https://www.conrad.com/p/bkl-electronic-1012015810-ribbo searchTerm=1548658& searchType=suggest& searchSuggest=product
spacer 5.5 HEX 25 mm M3 male/female	31	2.79	86.49	Keystone Electronics	24300	https://www.mouser.fr/ProductDetail/Keystone-Electronics/24300?qs=UWqYQ%2F2cZWu0ejpOzmZC2A%3D%3D
Screw	9	0.305	2.745	APM HEXSEAL	RM3X8MM-2701	https://www.mouser.fr/ProductDetail/APM-HEXSEAL/RM3X8MM-2701?qs=JJSE%2F12mKnS3VxSDrYXUHw%3D%3D
spacer 5.5 HEX 25 mm M3 female/female	9	0.846	7.614	Keystone Electronics	25515	https://www.mouser.fr/ProductDetail/Keystone-Electronics/25515?qs=UWqYQ%2F2cZWuxuhUmfr%252BZuQ%3D%3D
2-way jumper	12					

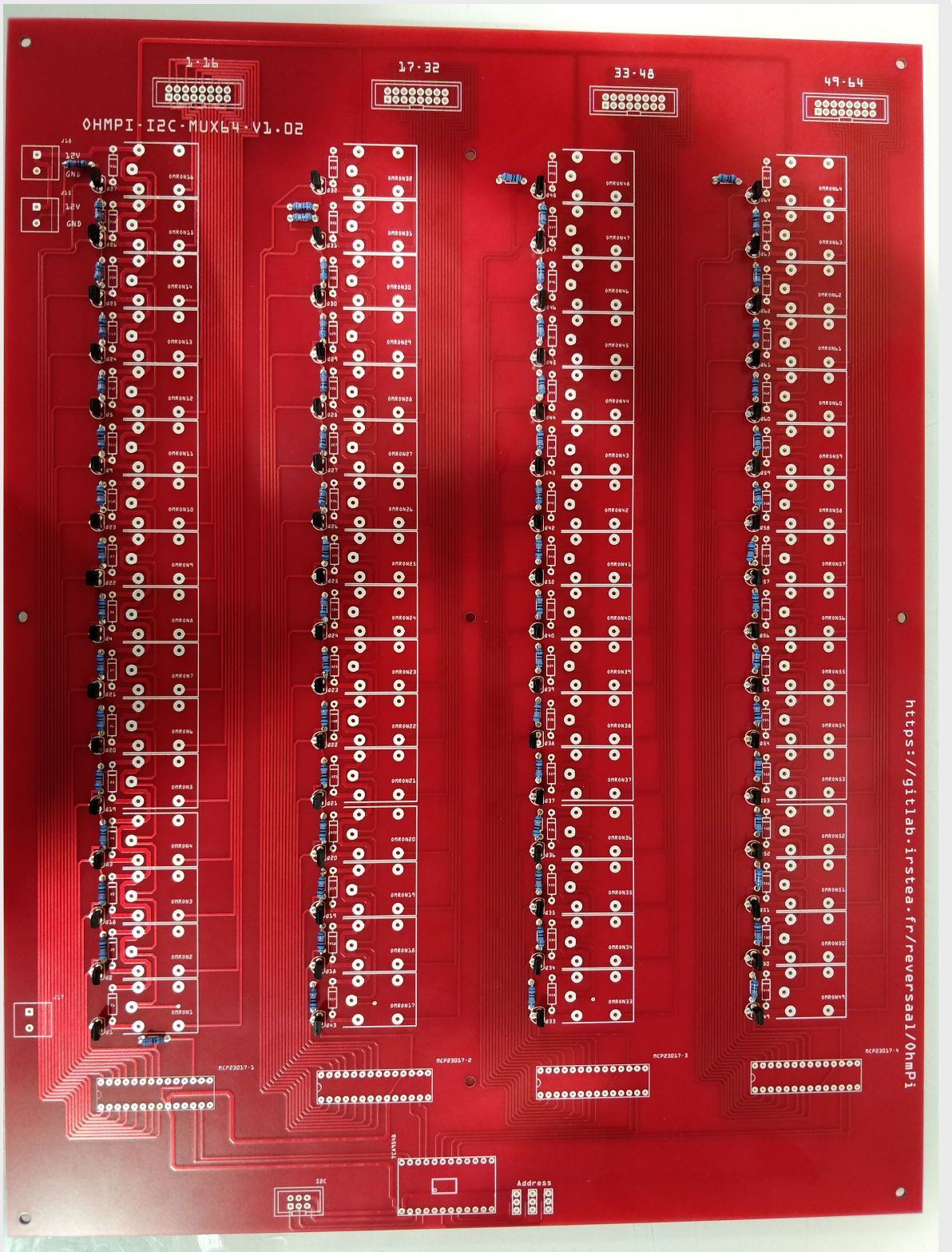
MUX board PCB



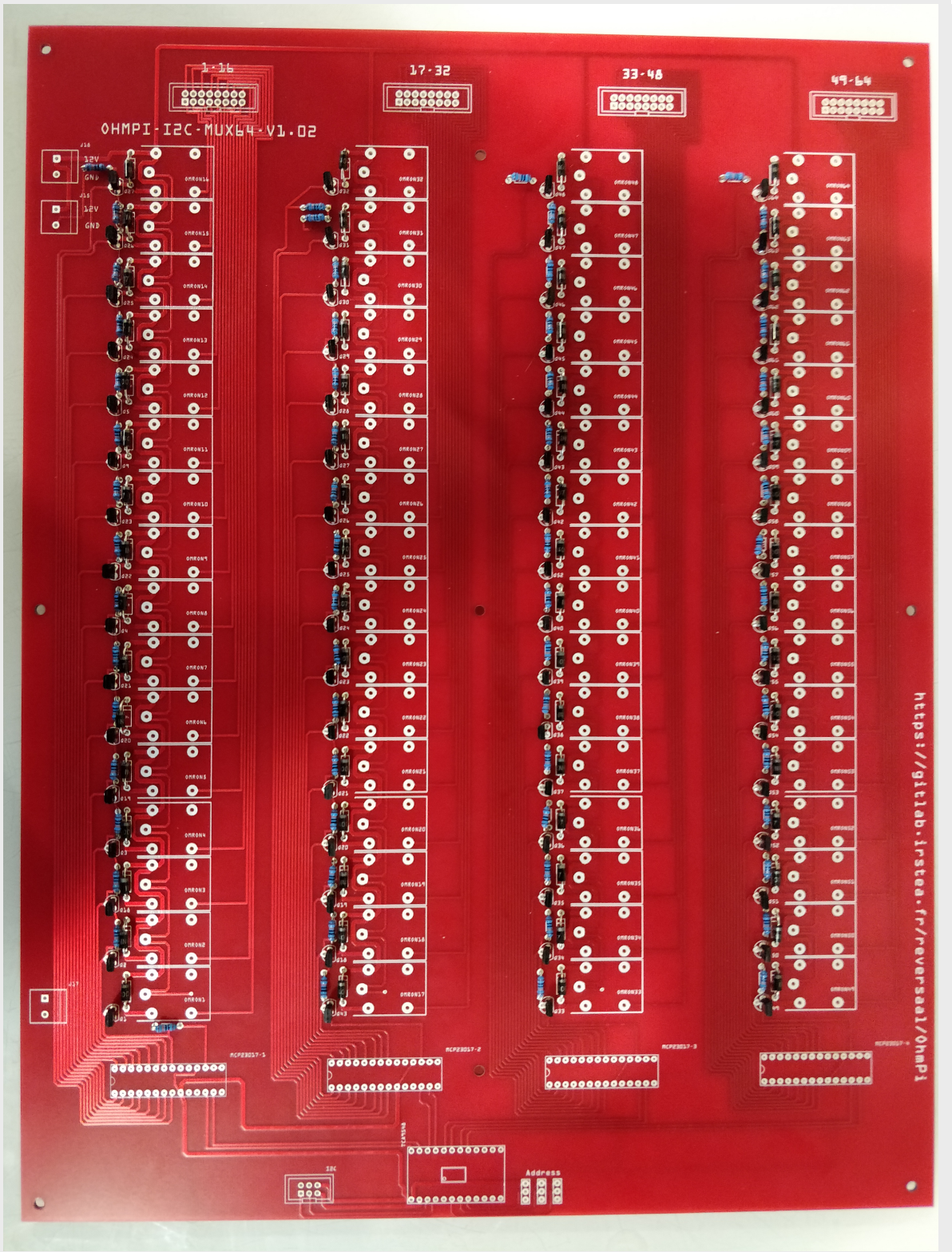
1 Mux board pcb



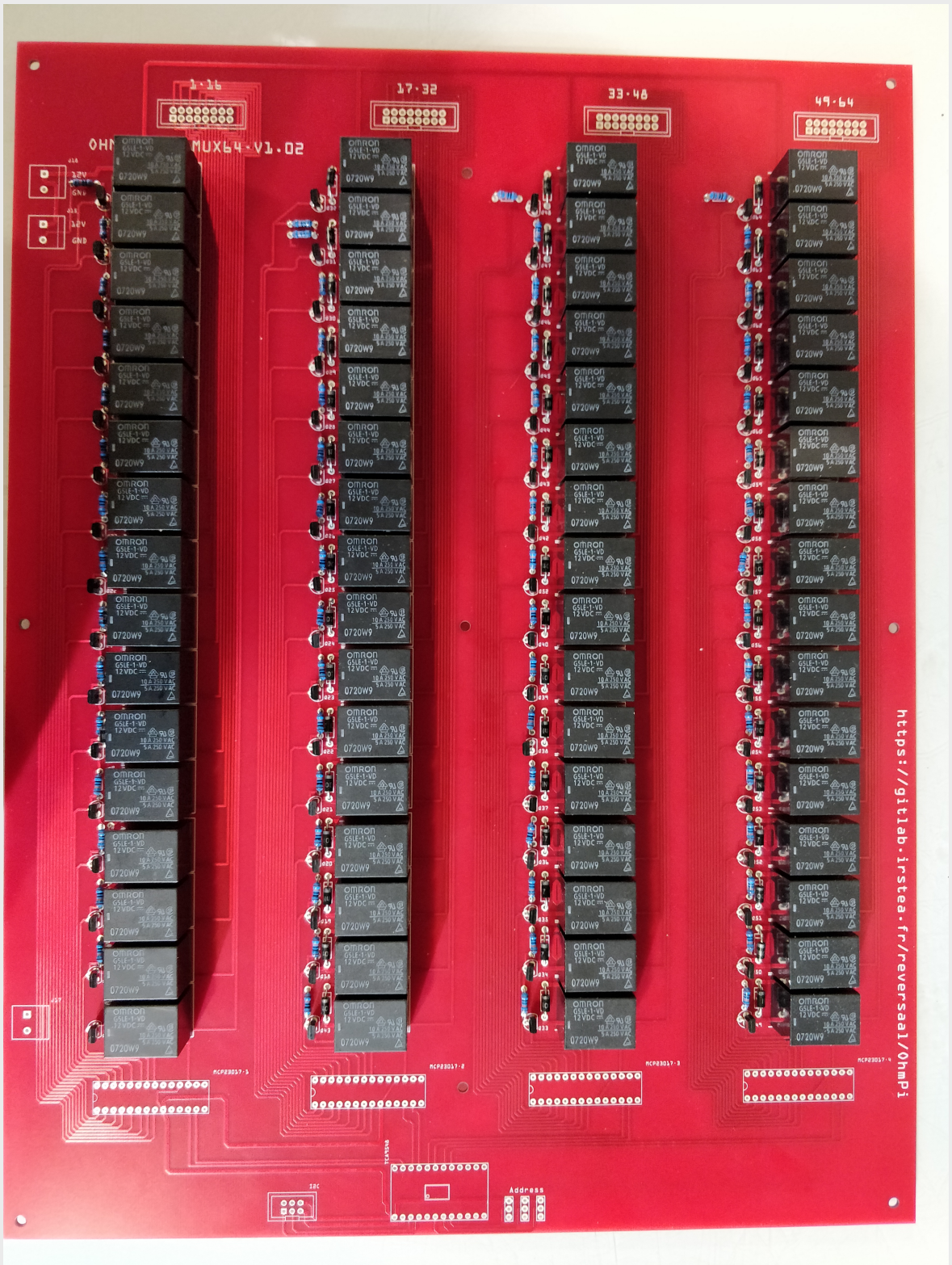
2 Installation of the 100 kOhm resistors



3 Installation of the MOSFET ZVN4206A



4 Installation of the diode 1N4007



5 Installation of the relay

6 Installation of the terminal screw

7 Installation of generic male header

8 Installation of Pin strip (6 pins) and pin strip (16 pins)

9 Installation of DIP Dual In Line Socket 2*14

Installation of Adafruit TCA9548A

Installation of MCP23017

Note: This step must be duplicated 4 times for every Mux card.

PART B MUX board address

To build an ohmpi it is necessary to have 4 MUX boards, with 4 different addresses. It is therefore necessary to identify each board, by assigning an address, which will be allocated in the OhmPi code. We present here the addresses selected by default.

For the A electrode board, we suggest addressing it with address 0x70:

1 Mount the jumpers and note the value of the address and the electrode name on the mux board (A).

For the B electrode board, we suggest addressing it with address 0x71:

2 Mount the jumpers and note the value of the address and the electrode name on the mux board (B).

For the N electrode board, we suggest addressing it with address 0x72:

3 Mount the jumpers and note the value of the address and the electrode name on the mux board (B).

For the M electrode board, we suggest addressing it with address 0x73:

4 Mount the jumpers and note the value of the address and the electrode name on the mux board (B).

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

Build MUX board v2024

The MUX board v2024 uses the same technology than the MUX board v2023, i.e. mechanical multiplexing. However it introduces a new level of flexibility by addressing multiple roles (A, B, M, N), which allows to build OhmPi system with multiple of 8 electrodes. Users can physically configure the MUX board to address 2 roles (A, B or M, N) or 4 roles (A, B, M, N). With only 32 relays, it can address 16 or 8 electrodes, for the 2- and 4-role configuration respectively. Given the reduced number of relays, the MUX board v2024 is interfaced with only two MCP23017 I/O expanders. This means that up to 4 MUX boards v2024 (i.e. 32-electrode system) can be directly connected to a measurement board v2024. A newly introduced I2C extension board (featuring a TCA9548A multiplexer) allows to connect up to 32 MUX board v2024, equating to a 256-electrode system. In theory, up to 8 I2C extension boards can be connected to the measurement board, which would allow to pilot 2048 electrodes. For obvious practical reasons, such a configuration couldn't be tested and is likely to be limited by the I2C bus being phisically too long, which would prevent to reach so many GPIOs. The MUX board v2024 also comes with both IDC connectors and screw connectors for the electrode takeouts, which allows to directly connect the electrode arrays to the board. In an effort to mitigate supply shortages, a last addition concerns the power mosfet associated with the relays, with the possibility to mount two types of components depending on market availability: either ZVN4206A or STP16NF06L.

Here, we will present how to assemble and configure a 32-electrode system, based on 4 MUX-board v2024 set up to address 2 roles / 16 electrodes each.

PART A Assembly of MUX board 2024

Required components

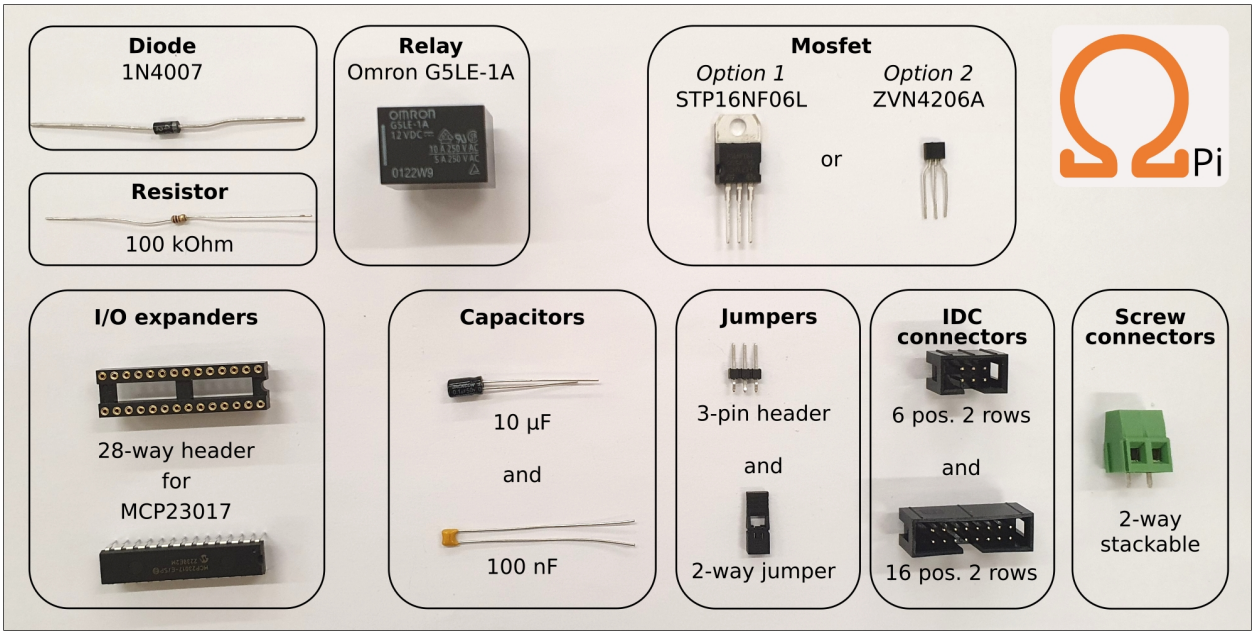


Table 7: List of components

Component	Quantity	Description	Vendor	Unit Price (EUR)	Costs (w. STP16N	Costs (w. ZVN06A)
Printed circuit board	1	1 PCB but often sold by 3	https://aisler.net/	•		
100nF	2	Unpolarized capacitor, small symbol	https://www.mouser.be/ProductDetail/KEMET/C320C104K1R5TA qs=c4UyoTs%2FLq1th4mcyOeTm 3D%3D	0.22	0.44	0.44
10uF	2	Polarized capacitor, small symbol	https://www.mouser.be/ProductDetail/Panasonic/ECA-1JHG100B?qs=sGAEpiMZZMsh%252B1woXyUXjwq 3D	0.24	0.48	0.48
1N_E4(	32	1000V 1A General Purpose Rectifier Diode, DO-41	https://eu.mouser.com/ProductDetail/Diodes-Incorporate 1N4007-T?qs=e%2FRqmsgwm9iVtgJ 3D%3D	0.13	4.16	4.16
Relays Pwr	1	Generic screw terminal, single row, 01x02, script generated (kicad-library-utils/schlib/autogen/	https://eu.mouser.com/ProductDetail/TE-Connectivity/282837-2?qs=A%252Bip%252BNCYi6O2H0N 3D%3D	0.55	0.55	0.55
6 pos. 2 rows IDC connector	1	Generic connector, double row, 02x03, odd/even pin numbering scheme (row 1 odd numbers, row 2 even numbers), script generated (kicad-library-utils/schlib/autogen/	https://eu.mouser.com/ProductDetail/Wurth-Elektronik/61200621621?qs=PhR8RmCirEbjX8n 3D%3D	0.44	0.44	0.44

continues on next page

Table 7 – continued from previous page

Component	Quantity	Description	Vendor	Unit Price (EUR)	Costs (w. STP16N	Costs (w. ZVN06A)
Screw_T (2*2P)	2	Generic screw terminal, single row, 01x04, script generated (kicad-library-utils/schlib/autogen/	https://eu.mouser.com/ProductDetail/TE-Connectivity/282837-2?qs=A%252Bip%252BNCYi6O2H0N3D%3D	0.55	1.10	1.10
16 pos. 2 rows IDC connector	1	Generic connector, double row, 02x08, odd/even pin numbering scheme (row 1 odd numbers, row 2 even numbers), script generated (kicad-library-utils/schlib/autogen/	https://eu.mouser.com/ProductDetail/Wurth-Elektronik/61201621621?qs=ZtY9WdtwX55qFf43D%3D	0.58	0.58	0.58
Screw_T (8*2P)	8	Generic screw terminal, single row, 01x16, script generated (kicad-library-utils/schlib/autogen/	https://eu.mouser.com/ProductDetail/TE-Connectivity/282837-2?qs=A%252Bip%252BNCYi6O2H0N3D%3D	0.55	4.40	4.40
3-pin header	2	Jumper, 3-pole, both open	https://eu.mouser.com/ProductDetail/TE-Connectivity/4-103321-5?qs=5TwgZeq9E7HSYL3D%3D	0.17	0.34	0.34
G5LE-1A DC12	32	Omron G5LE relay, Miniature Single Pole, SPDT, 10A	https://eu.mouser.com/ProductDetail/Omron-Electronics/G5LE-1A-DC12?qs=sGAEpiMZZMsqIr53D	1.18	37.76	37.76

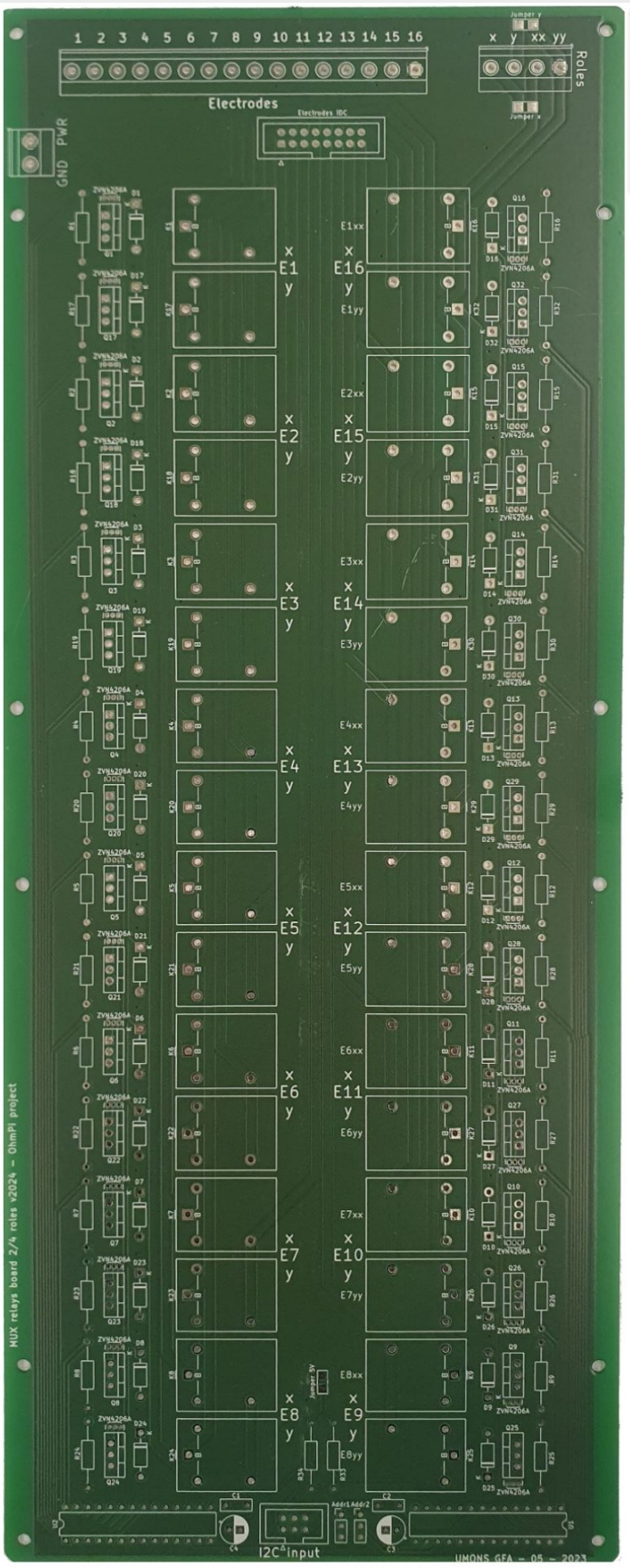
continues on next page

Table 7 – continued from previous page

Component	Quantity	Description	Vendor	Unit Price (EUR)	Costs (w. STP16N	Costs (w. ZVN06A)
STP16N	32	30A Id, 50V Vds, N-Channel Power MOSFET, TO-220	https://eu.mouser.com/ProductDetail/STMicroelectronics/STP16NF06?qs=FOlmdCx%252BAA3QgI0ylnH3D%3D	1.08	34.56	
ZVN42	32	30A Id, 50V Vds, N-Channel Power MOSFET, TO-220	https://eu.mouser.com/ProductDetail/Diodes-Incorporated/ZVN4206AVSTZ?qs=%2F4dsY8i%2FUxLhPG5M4pjuJ3D%3D	0.60		19.20
100k	34	Resistor	https://eu.mouser.com/ProductDetail/Vishay-Beyschlag/MBA02040C1003F?qs=mzRxyRlhVdt9crF72F5Q%3D%3D	0.10	3.40	3.40
MCP23	2	16-bit I/O expander, I2C, interrupts, w pull-ups, SPDIP-28	https://eu.mouser.com/ProductDetail/Microchip-Technolc/MCP23017-E-SP?qs=usxtMOJb1RyESXZ3D%3D	1.53	3.06	3.06
2-way jumper	2	•	https://eu.mouser.com/ProductDetail/TE-Connectivity/1-881545-2?qs=G55MHhPmvtILJr82FD4w%3D%3D	0.30	0.60	0.60
28-way socket	2	2.54mm Pitch Vertical 28 Way, Through Hole Turned Pin IC Dip Socket, 3A	https://befr.rs-online.com/web/p/dil-sockets/1831575	0.75	1.50	1.50
TOTAL					91.27	71.51

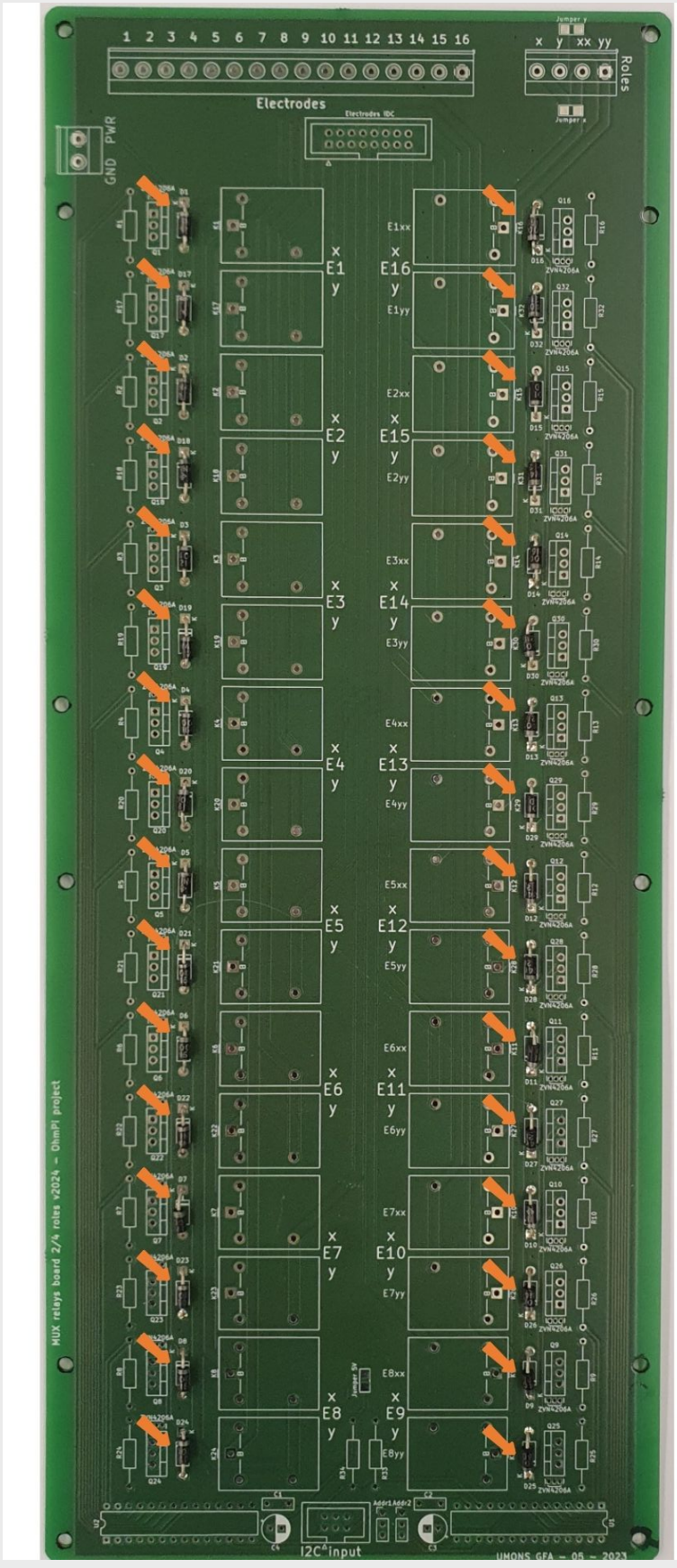
Mounting components on PCB board

MUX board v2024 PCB (mux.2024.0.0).


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Electrodes
Electrodes IDC
GND PWR
x y xx yy
Roles
Jumper y
Jumper x
E1 E16 E1xx E1yy
E2 E15 E2xx E2yy
E3 E14 E3xx E3yy
E4 E13 E4xx E4yy
E5 E12 E5xx E5yy
E6 E11 E6xx E6yy
E7 E10 E7xx E7yy
E8 E9 E8xx E8yy
MUX relays board 2/4 roles v2024 - OhmPi project
I2C input
UMONS GFA - 05 - 2023

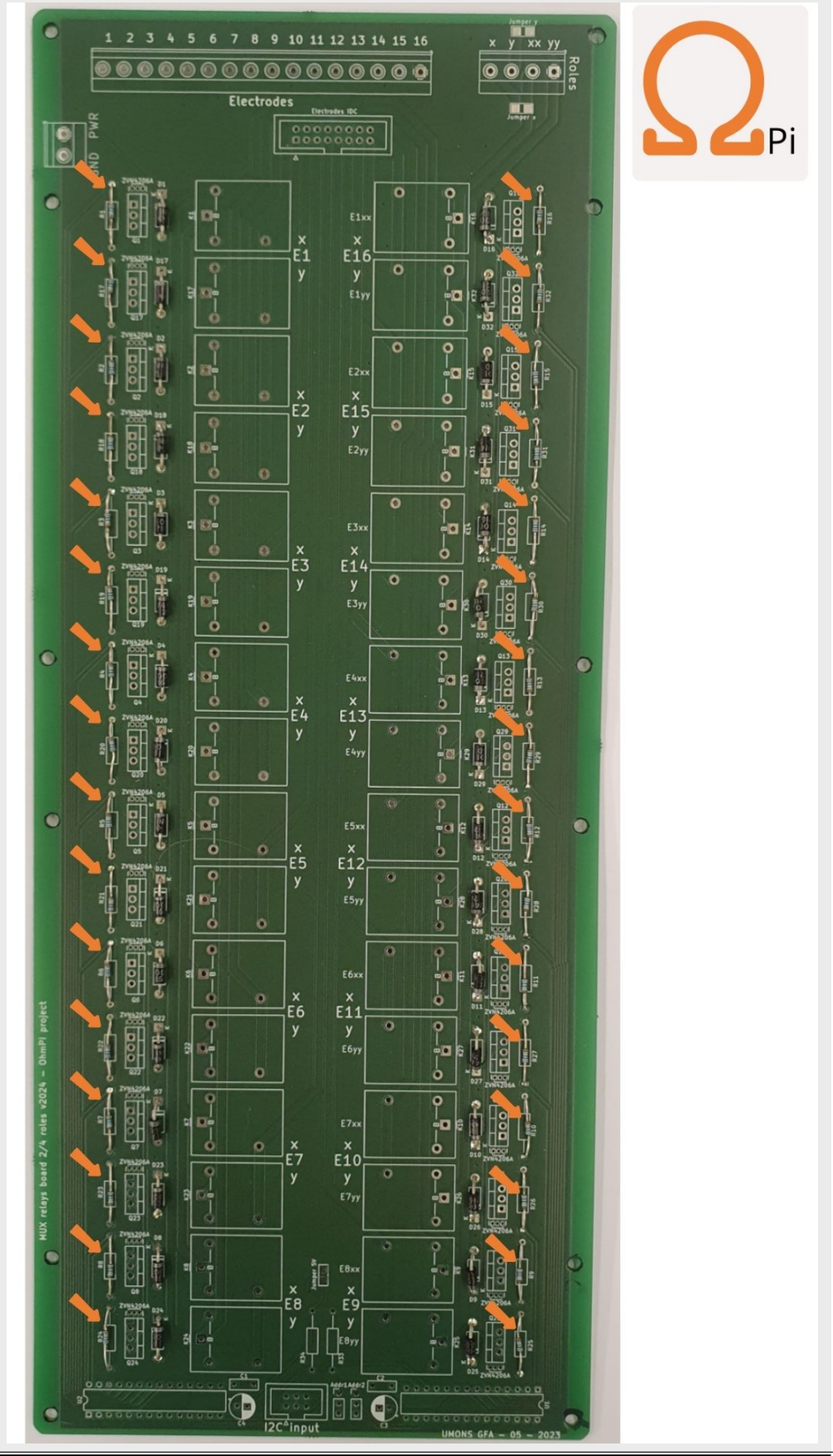
1

Mount the diodes.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Electrodes
Electrodes IDC
GND PWR
x y xx yy
Roles
Jumper y
Jumper x
E1 E16 E2 E15 E3 E14 E4 E13 E5 E12 E6 E11 E7 E10 E8 E9
E1xx E1yy E2xx E2yy E3xx E3yy E4xx E4yy E5xx E5yy E6xx E6yy E7xx E7yy E8xx E8yy
ZVN4206A
Jumper 5V
Addr1 Addr2
I2C input
UMONS GFA - 05 - 2023
MUX relays board 2/4 roles v2024 - OhmPi project

2

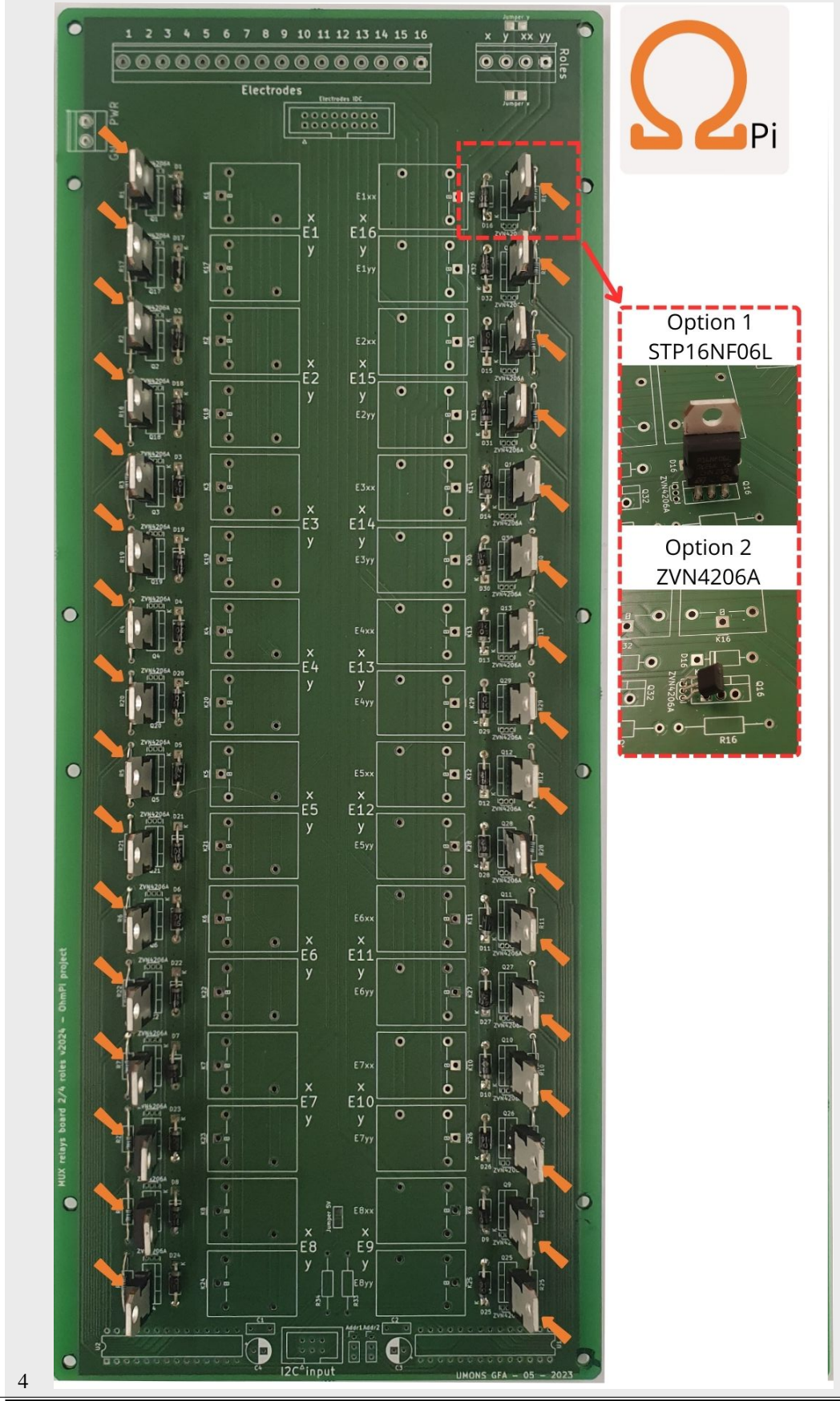
Mount the 100 kOhm resistors.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Electrodes
Electrodes IDC
x y xx yy
Roles
GND PWR
E1xx
x
E16
y
E1yy
E2xx
x
E15
y
E2yy
E3xx
x
E14
y
E3yy
E4xx
x
E13
y
E4yy
E5xx
x
E12
y
E5yy
E6xx
x
E11
y
E6yy
E7xx
x
E10
y
E7yy
E8xx
x
E9
y
E8yy
x
E1
y
x
E2
y
x
E3
y
x
E4
y
x
E5
y
x
E6
y
x
E7
y
x
E8
y
MUX relays board 2/4 roles v2024 - OhmPi project
I2C input
UMONS GFA - 05 - 2023

3

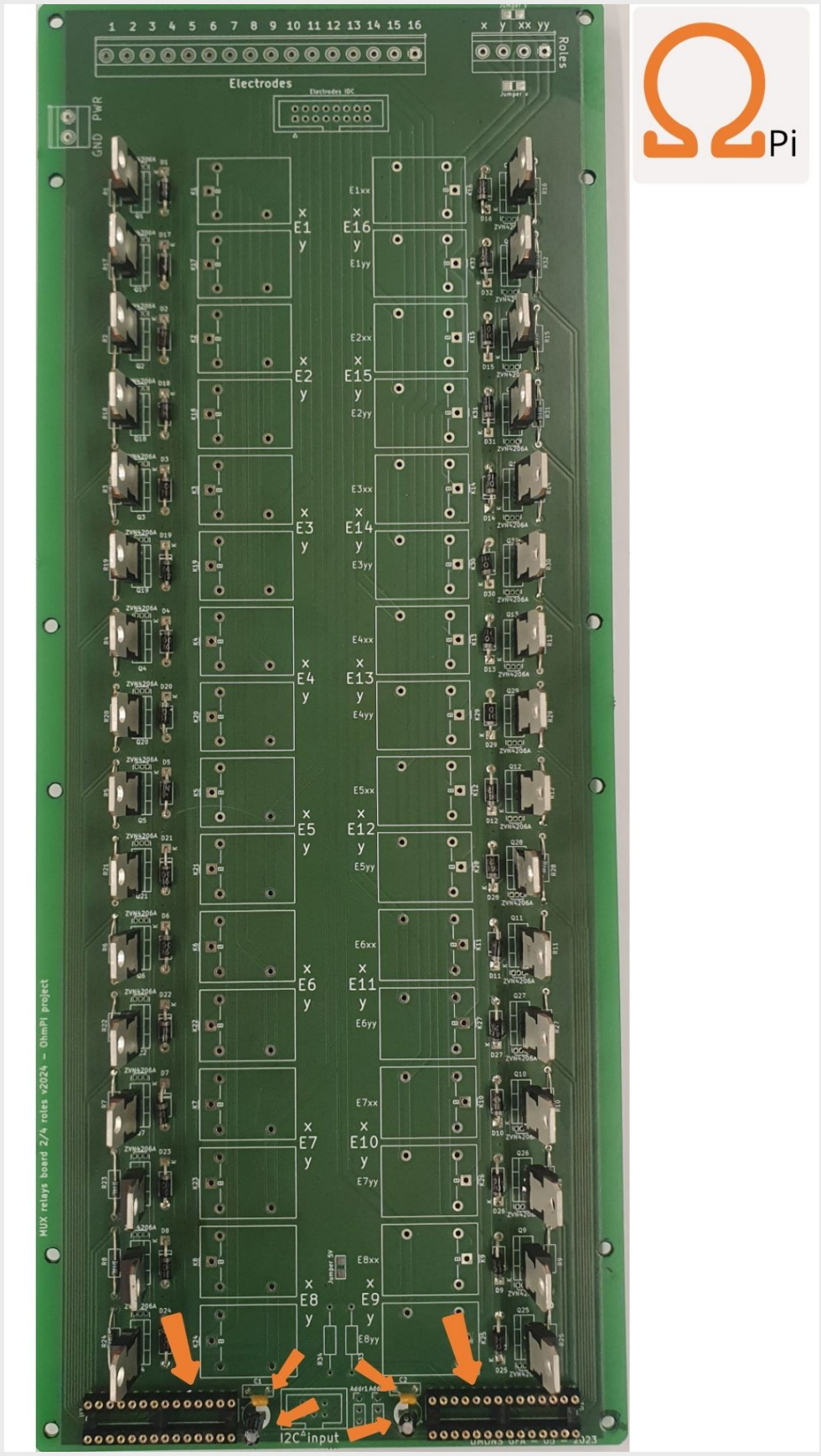
Mount the mosfets. You can chose between ZVN4206A or STP16NF06L.

Warning: In the PCB v2024.0.1, the white footprint of the ZVN4206A is upside-down. Please double check that the "drain" pint of the ZVN goes to the relay and the "source" pin goes to the ground. In doubt, refer to the ZVN datasheet. The white footprint on the PCB is corrected in v2024.0.2.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Electrodes
x y xx yy
Roles
PWR
E1 E16
E2 E15
E3 E14
E4 E13
E5 E12
E6 E11
E7 E10
E8 E9
MUX relays board 2/4 roles v2024 - OhmPi project
I2C input
UMONS GFA - 05 - 2023
Option 1
STP16NF06L
Option 2
ZVN4206A
Pi

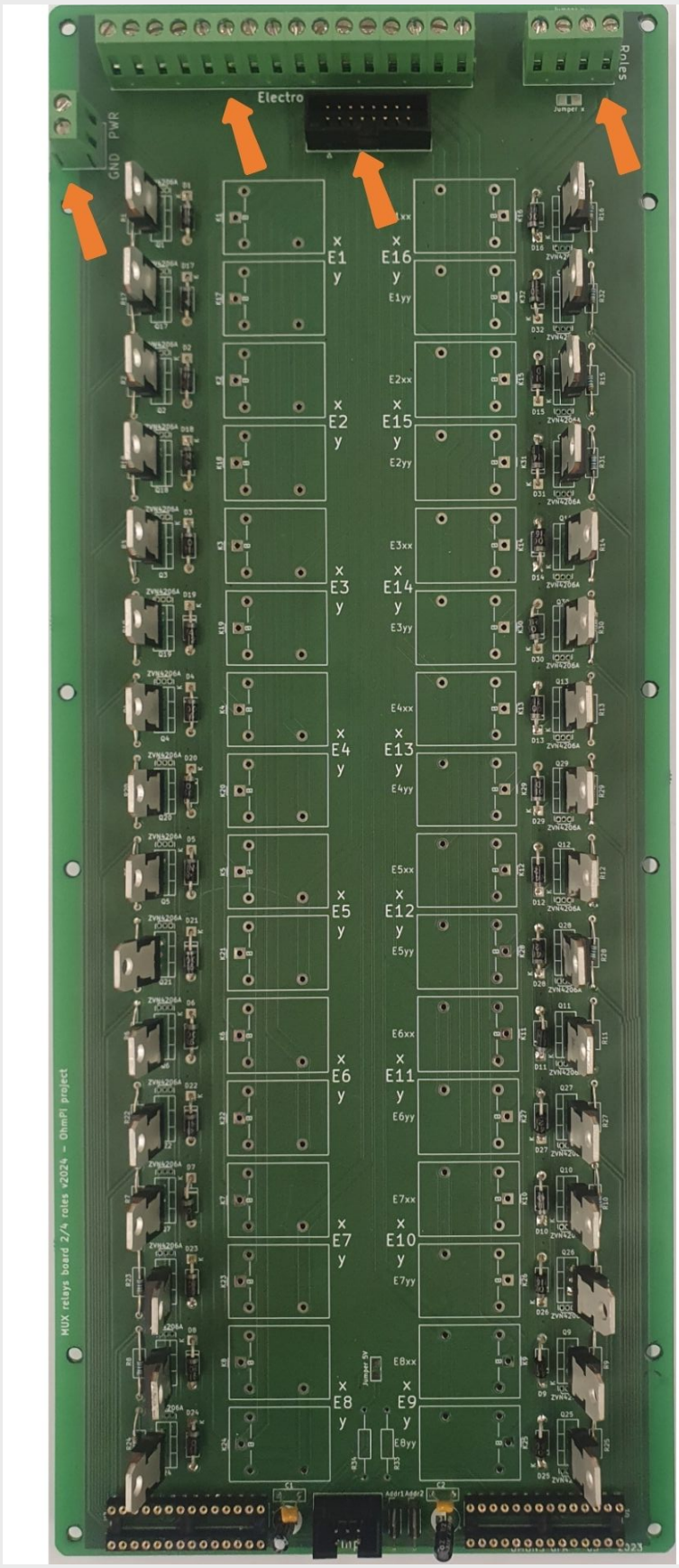
4

Mount the MCP23017 sockets and the capacitors.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Electrodes
GND PWR
x y xx yy
Roles
E1 E16 E2 E15 E3 E14 E4 E13 E5 E12 E6 E11 E7 E10 E8 E9
MUX relays board 2/4 roles v2024 – OhmPi project
I2C input

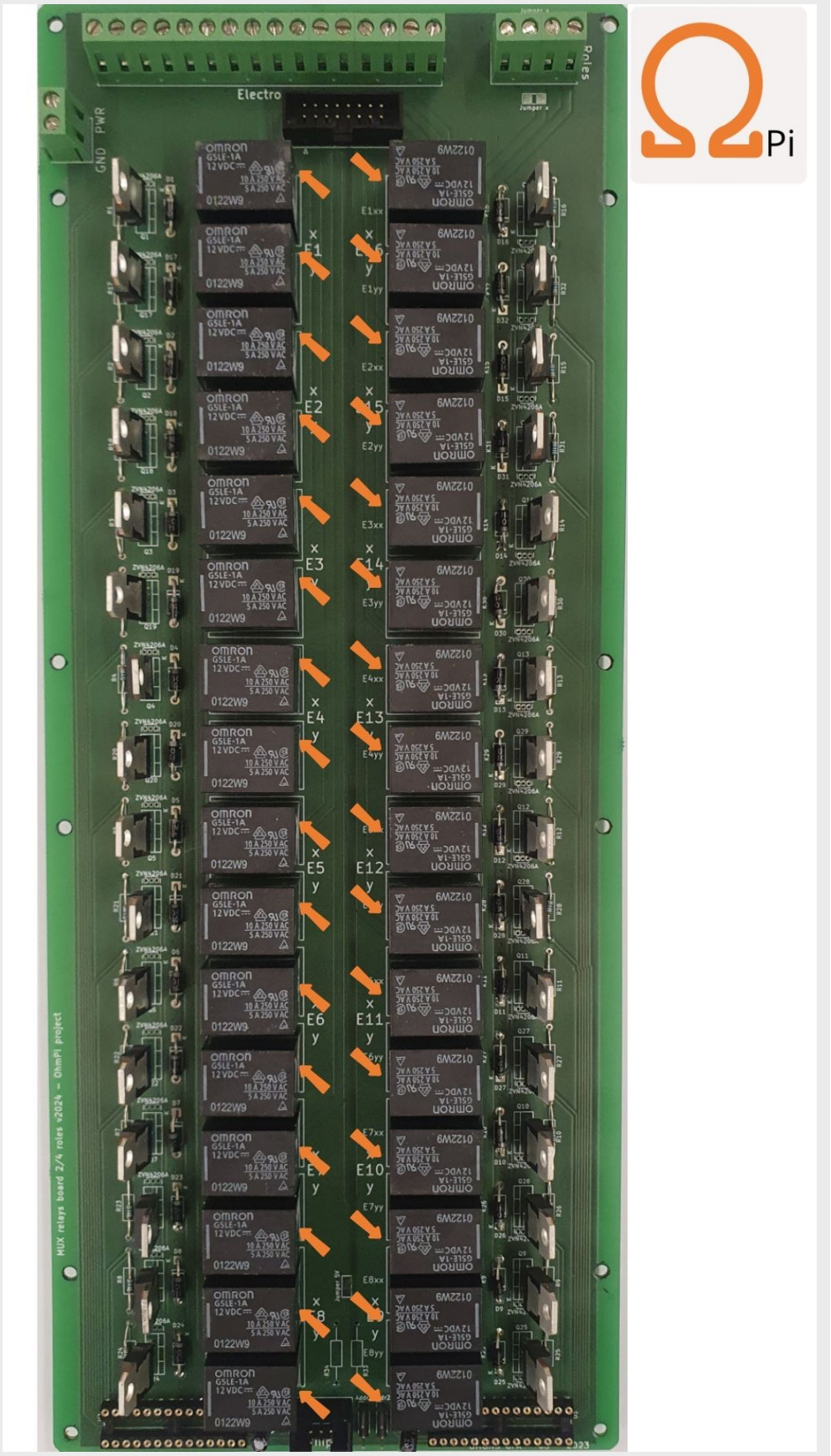
5

Mount the screw connectors and the IDC connectors. Note that if a board is dedicated to be configured in 2-role mode, the electrode takeouts can be mounted with 8 screw connectors only (on electrodes 1-8) and potentially a 8 position IDC connector for the IDC takeout.


Electro
Roles
GND PWR
MUX relays board 2/4 roles v2024 - OhmPi project
E1
E16
E2
E15
E3
E14
E4
E13
E5
E12
E6
E11
E7
E10
E8
E9

6

Mount the relays.


Electro
Roles
PWR
GND
Jumper x
Jumper 5V
MUX relays board 2/4 roles v2024 - OhmPi project
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10
E11
E12
E13
E14
E15
E16
OMRON
GSLE-1A
12VDC
0122W9

7

Defining role configuration

The MUX board v2024 can be configured in 2- or 4-role mode. This means that one board can either address 16 electrodes on 2 roles (X,Y for A,B or M,N), or 8 electrodes on 4 roles (labelled X,Y,XX,YY for A,B,M,N). An OhmPi system can manage a combination of 2-role and 4-role MUX boards as long as the 2-role boards come in pairs (e.g. 2 2-role MUX and 1 4-role MUX for 24 electrodes).

2-role configuration

This configuration is the prefered way to build systems with an even number of MUX boards (for 16, 32, 48 electrodes). However, when assembling an OhmPi, keep in mind that 2-role mode MUX boards have to come in pairs.

To enable the 2-role mode, 2 "roles" solder jumpers have to be bridged in the front side of the PCB next to the roles connector. To do so, the two jumper pads of each bridge have to be soldered together. This will connect roles X and roles XX together, as well as roles Y and YY together. In this way, the board is configure in 2-role mode. You can verify that the pair of roles X - XX and Y - YY are well connected by doing continuity checks with a digital voltmeter.

Warning: Make sure that the 8 "electrodes" solder jumpers at the back of the PCB are NOT bridged to avoid risks of shortcuts !

4-role configuration

Configuring a board in 4-role mode enables to use an odd number of MUX board (for systems with 8, 24, 40, 56,… electrodes). To do so, the 8 "electrodes" solder jumpers at the back of the PCB have to be bridged. In this way relays of electrodes relays of the following electrodes are paired together (albeit on different roles A, B, M and N): 1-16, 2-15, 3-14, 4-13, 5-12, 6-11, 7-10, 8-9. You can verify that these combinations are connected together by continuity checks with a digital voltmeter. It is best practice to only mount 8 screw connectors on the electrodes takeouts (and potentially only a 8 position IDC connector) to avoid confusion when cabling the system.

Warning: Make sure that the two "roles" solder jumpers at the front remain NOT bridged to avoid risks of shortcuts !

MUX board addresses

Each MUX board v2024 comes with 2 I/O expanders MCP23017, addressing 16 relays each. They expose a pair of two I2C addresses on the I2C bus in the range 0x20 - 0x27. Two 2-way jumpers placed on the 3-pin headers next to the IDC connector at the bottom of the board allow to shift the addesses two by two. There is 4 possible combinations for the jumpers which give the following addresses:

Jumper position Addr1	Jumper position Addr2	I2C addresses
Up	Up	0x20 - 0x21
Down	Up	0x22 - 0x23
Up	Down	0x24 - 0x25
Down	Down	0x26 - 0x27

The jumper positions of each ('up' or 'down' have to be carefully filled in the configuration file). One can check the I2C addresses visible on the I2C bus by typing the following command on the Raspberry Pi terminal, assuming that the MUX boards are powered and correctly connected to the measurement board:

```
i2cdetect -y 4
```

Replace "-y 4" by "-y 1" if the MUX is plugged on the "board" IDC connector, or if plugged in to a mb.2023.0.X board.

PART B Assembling MUX boards in an OhmPi system

The cabling of several MUX boards v2024 within an OhmPi system is entirely dependent on the role configuration of each board.

- 2-role MUX boards have to come in pairs. The 16 electrodes takeouts of each pair have to be cabled together. This is easily done with a ribbon cable plugged on the 16-way IDC connectors of the pair of boards. This also allows to stack two boards together leaving the screw connectors of the board on the top accessible to connect wires from the electrode arrays.
- 4-role MUX boards do not have to come in pairs. The 4 roles of each board have to be connected to the other 4 roles of the system (and at least to the ABMN connector on the measurement board). The electrodes connectors can only be used to address the first 8 or the last 8 positions. This is critical if wanting to connect the electrodes via the IDC connectors, which whill have to be carefully cabled.

Check MUX board v2024

Use the picture and table below to manually check with a multimeter for continuity and expected voltage in the board. Check your board against the correct expected column: 2-roles or 4-roles.

If a continuity check does not pass it's likely means there is an issue with the soldering on the board. If the voltage with I2C (SDA and SCL pins) is not expected, there is likely an issue with pull-up resistors. For the test with power "on", we expect the mux board to be connected to a 12V supply via the screw terminals and to the measurement board via the IDC ribbon cable.

Table 8: Hardware check

Name	Power	Type	Multiplexer BLACK probe	Multiplexer RED probe	Expected 4 roles	Expected 2 roles
SC1	off	continuity	screw terminal GND	screw terminal PWR	no continuity	no continuity
SC2	off	continuity	screw terminal role x	screw terminal role y	no continuity	no continuity
SC3	off	continuity	screw terminal role x	screw terminal role xx	no continuity	continuity
SC4	off	continuity	screw terminal role x	screw terminal role yy	no continuity	no continuity
SC5	off	continuity	screw terminal role y	screw terminal role xx	no continuity	no continuity
SC6	off	continuity	screw terminal role y	screw terminal role yy	no continuity	continuity
SC7	off	continuity	screw terminal role xx	screw terminal role yy	no continuity	no continuity
SC8	off	continuity	screw terminal elec 1	screw terminal elec 2	no continuity	no continuity
SC9	off	continuity	screw terminal elec 2	screw terminal elec 3	no continuity	no continuity
SC10	off	continuity	screw terminal elec 3	screw terminal elec 4	no continuity	no continuity
SC11	off	continuity	screw terminal elec 4	screw terminal elec 5	no continuity	no continuity
SC12	off	continuity	screw terminal elec 5	screw terminal elec 6	no continuity	no continuity
SC13	off	continuity	screw terminal elec 6	screw terminal elec 7	no continuity	no continuity
SC14	off	continuity	screw terminal elec 7	screw terminal elec 8	no continuity	no continuity
SC15	off	continuity	screw terminal elec 8	screw terminal elec 9	no continuity	continuity
SC16	off	continuity	screw terminal elec 9	screw terminal elec 10	no continuity	no continuity
SC17	off	continuity	screw terminal elec 10	screw terminal elec 11	no continuity	no continuity

continues on next page

Table 8 – continued from previous page

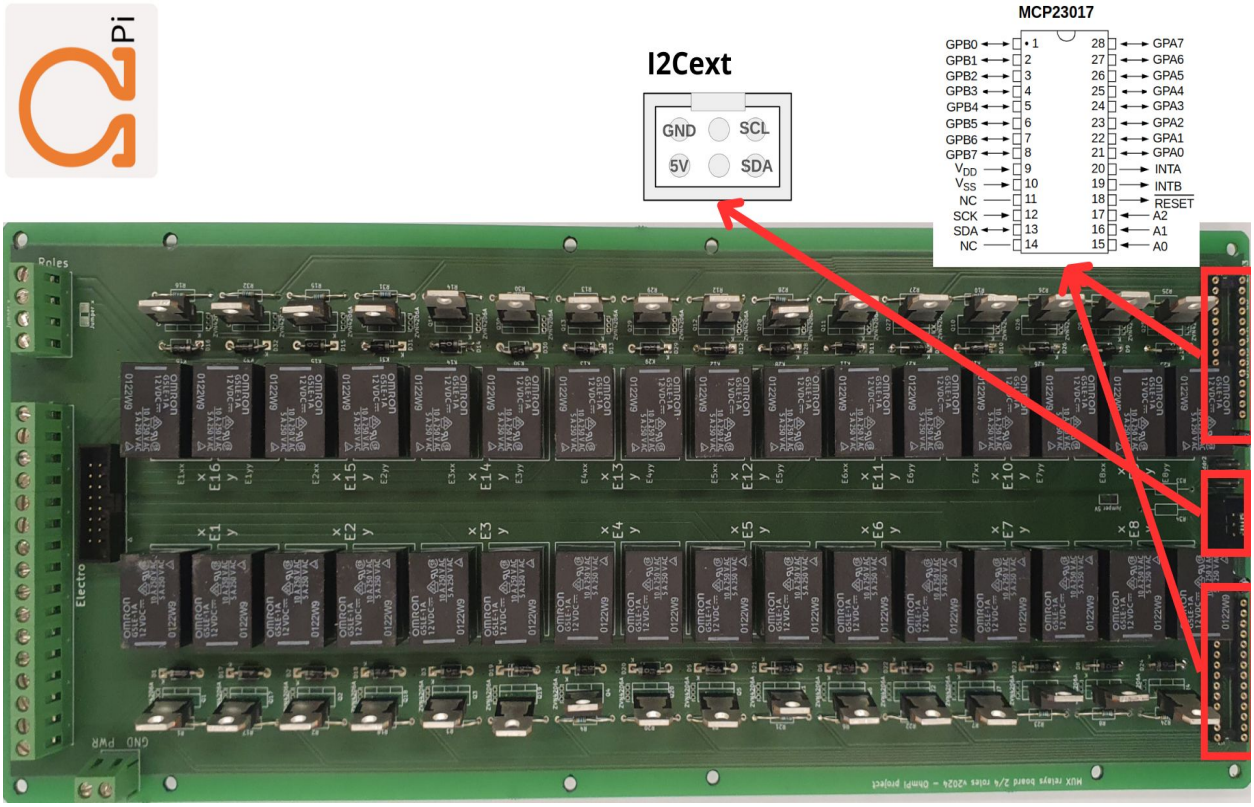
Name	Power	Type	Multiplexer BLACK probe	Multiplexer RED probe	Expected 4 roles	Expected 2 roles
SC18	off	continuity	screw terminal elec 11	screw terminal elec 12	no continuity	no continuity
SC19	off	continuity	screw terminal elec 12	screw terminal elec 13	no continuity	no continuity
SC20	off	continuity	screw terminal elec 13	screw terminal elec 14	no continuity	no continuity
SC21	off	continuity	screw terminal elec 14	screw terminal elec 15	no continuity	no continuity
SC22	off	continuity	screw terminal elec 15	screw terminal elec 16	no continuity	no continuity
SC23	off	continuity	screw terminal elec 1	screw terminal elec 16	no continuity	continuity
SC24	off	continuity	screw terminal elec 2	screw terminal elec 15	no continuity	continuity
SC25	off	continuity	screw terminal elec 3	screw terminal elec 14	no continuity	continuity
SC26	off	continuity	screw terminal elec 4	screw terminal elec 13	no continuity	continuity
SC27	off	continuity	screw terminal elec 5	screw terminal elec 12	no continuity	continuity
SC28	off	continuity	screw terminal elec 6	screw terminal elec 11	no continuity	continuity
SC29	off	continuity	screw terminal elec 7	screw terminal elec 10	no continuity	continuity
SC30	off	continuity	screw terminal elec 8	screw terminal elec 9	no continuity	continuity

Warning: Do not power the board if one of the SC (shortcircuit) test does not pass!

Table 9: Hardware check

Name	Power	Type	Multiplexer BLACK probe	Multiplexer RED probe	Expected roles 4	Expected roles 2
C1	off	continuity	screw terminal GND	both MCP23017 VSS	continuity	continuity
C2	off	continuity	screw terminal GND	I2Cext GND	continuity	continuity
C3	off	continuity	I2Cext SDA	both MCP23017 SDA	continuity	continuity
C4	off	continuity	I2Cext SDL	both MCP23017 SDL	continuity	continuity
V1	on	voltage	screw terminal GND	both MCP23017 SDA	5V	5V
V2	on	voltage	screw terminal GND	both MCP23017 SDL	5V	5V
V2	on	voltage	screw terminal GND	both MCP23017 VDD	5V	5V

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.



1.2.4 Power supply

Two sources of power are available now:

- a 12V battery
- a regulated power supply (DPH5005)

12V battery

When injecting, we actually connect the + and - of the battery to the A, B electrodes. Hence, we can only inject 12V maximum.

Digital power supply (DPH5005)

This alimentation enables us to inject up to 50 V and also to regulate the current. It needs to be connected to a 12V battery and can be controlled using *modbus* by the raspberrypi.

To assemble DPH5005, please follow the links:

DPH5005 manual

DPH5005 case manual

Note: Change the Baudrate from 9600 to 19200, press and maintain **SET**, and start DPH5005, you acces to a new menu change **BAUD**

Warning: Only use DPH5005 with the measurement board v2024

Warning: We sometimes refer to DPS (Digital Power Supply) as a general power supply different from the 12V battery. But this DOES NOT refer to the DPS5005 component (step down DC/DC). The component used in the documentation is the DPH5005 (boost DC/DC converter).

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

1.2.5 Raspberry Pi components

Required components	Quantity
Raspberry Pi 4 Model B	1
Micro SD 32 Go	1
HDMI Cable	1
Computer mouse	1
Computer Keyboard	1

The first step is to start up the Raspberry Pi board, including installation of an OS (operating system). For this step, the installation instructions are well described on the Raspberry website

1. Watch the video how to set up your raspberry Pi.

2. The authors recommend installing the latest stable and complete version of Raspberry Pi OS (Previously called Raspbian) by using Raspberry Pi Imager.
3. or you can visit this website.

Note: All the development tests were performed on Raspberry Pi 3 Model B and Raspberry Pi 4 Model B

To install the Raspberry Pi, please refer to the *Getting started* section.

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

1.2.6 Build your Ohmpi

In previous sections, we described the various components that compose Ohmpi. Today, version 1.0x is no longer maintained, but all boards from v2023 upwards are compatible with each other. This is the major innovation of 2024. Depending on your needs and applications, you can choose the board you are going to use.

Recommended configurations

Applications	Measurement Board	Mux	Raspberry Pi	Power supply	Config file name
64 or more electrodes for field monitoring	mb v2024	Mux v2023	Raspberry Pi 3 Model B or 4 model B	DPH5005	config_mb_2024_0_2__4_mux_2023_d
8, 16, 32, 48 electrodes for field monitoring	mb v2024	Mux v2024	Raspberry Pi 3 Model B or 4 model B	DPH5005	config_mb_2024_0_2__4_mux_2024_2
8, 16, 32, 48 electrodes for laboratory monitoring	mb v2024	Mux v2024	Raspberry Pi 3 Model B or 4 model B	12V Battery	config_mb_2024_0_2__4_mux_2024_2
4 electrodes concrete sample Laboratory (Rhoa and IP)	mb v2024	None	Raspberry Pi 3 Model B	DPH5005	config_mb_2024_0_2_dph5005.py
4 electrodes soil sample laboratory (Rhoa and IP)	mb v2024	None	Raspberry Pi 3 Model B	12V Battery	config_mb_2024_0_2.py.
4 electrodes-soil sample laboratory (only Rhoa)	mb v2023	None	Raspberry Pi 3 Model B	12V Battery	config_mb_2023.py

Another possible combination is to use MUX v2023 with MUX v2024 together, which allows to add series of 8 electrodes to a 64-electrode system. This could be handful if ones is looking to build e.g. a 96 electrode system, which would therefore feature 4 MUX 2023 (64 electrodes) + 4 MUX 2024 (32 electrodes).

Below we detail examples of OHMPI systems assemblies in different versions.

OhmPi systems assembly tutorials

Warning: We **strongly** recommend to test the assembled system in a controlled environment (in the lab, on resistor boards) before deploying in the field.

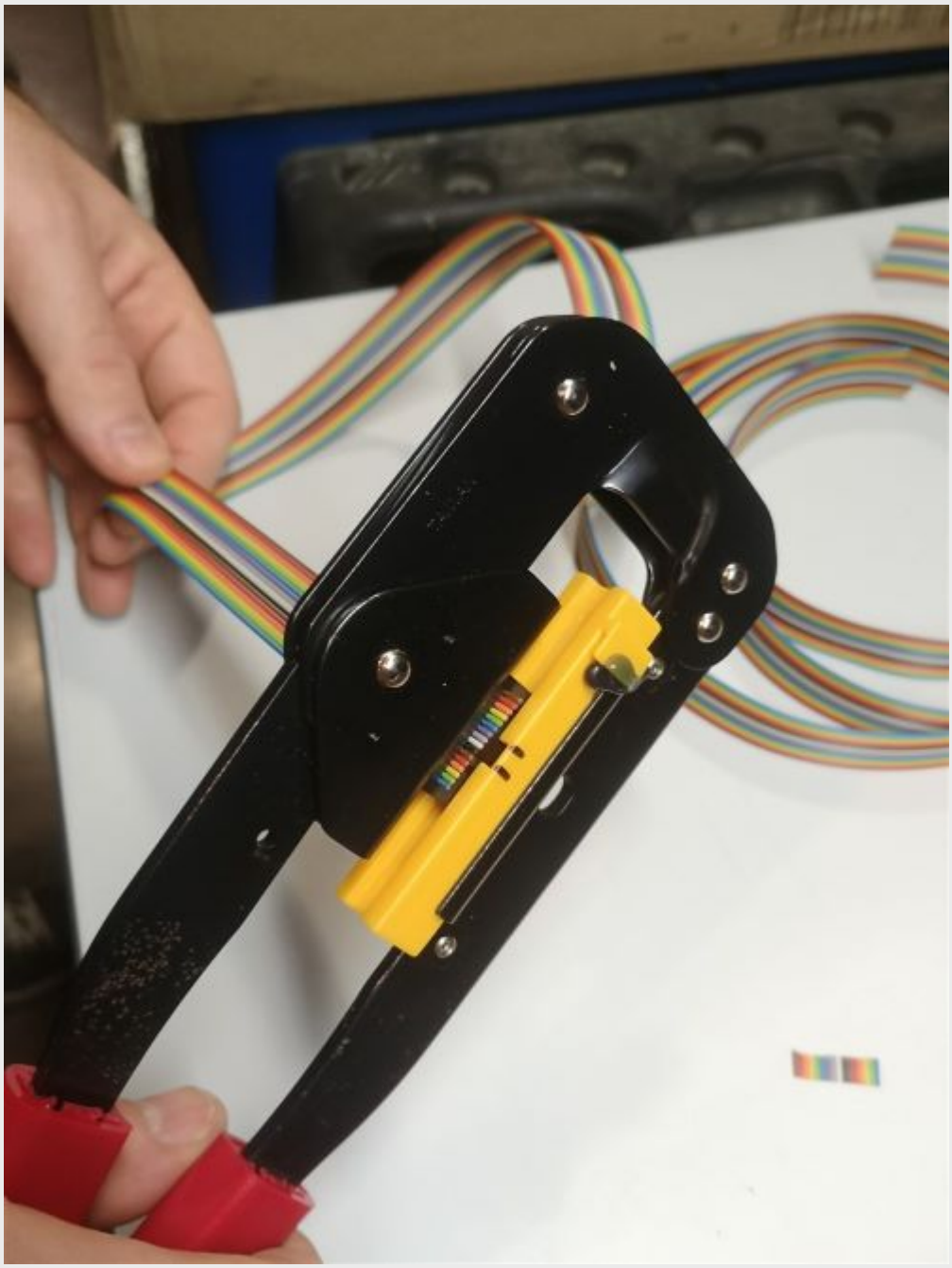
Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

Assembling the 64-electrode OhmPi

1 mb2024 + 4 mux2023 + 1 DPH5005)

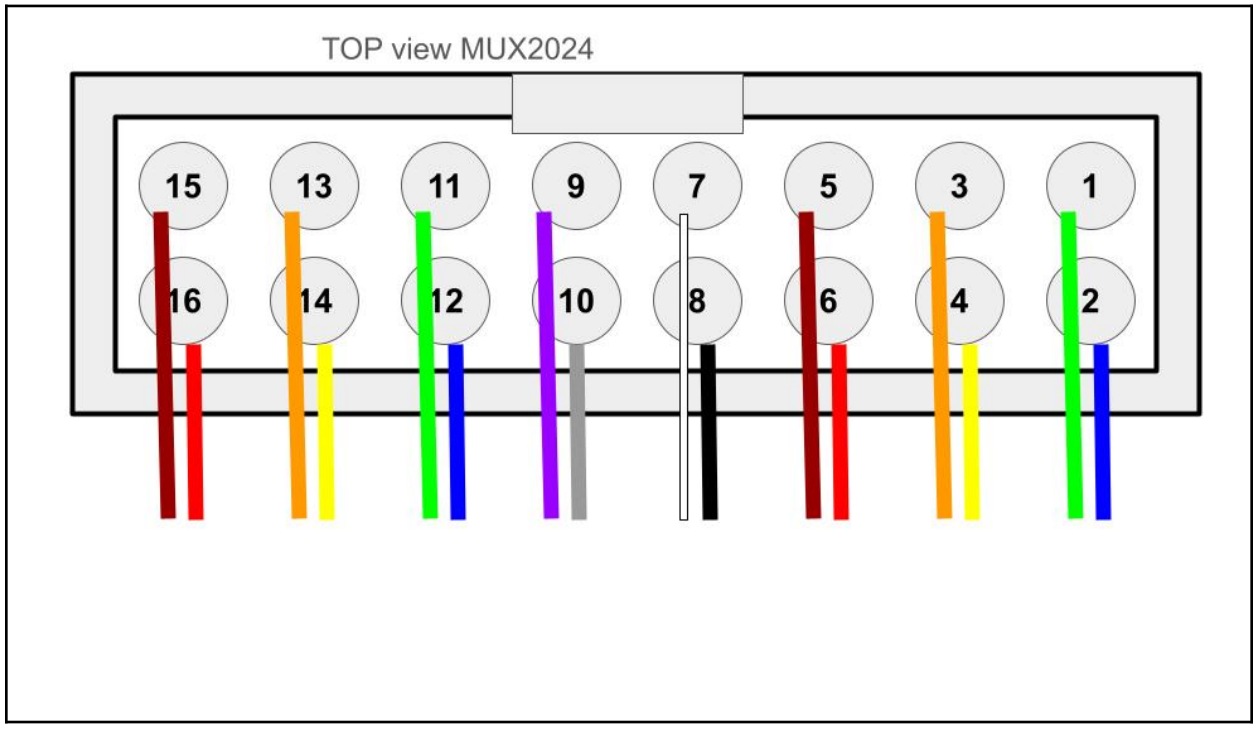
TODO :list on tools and components

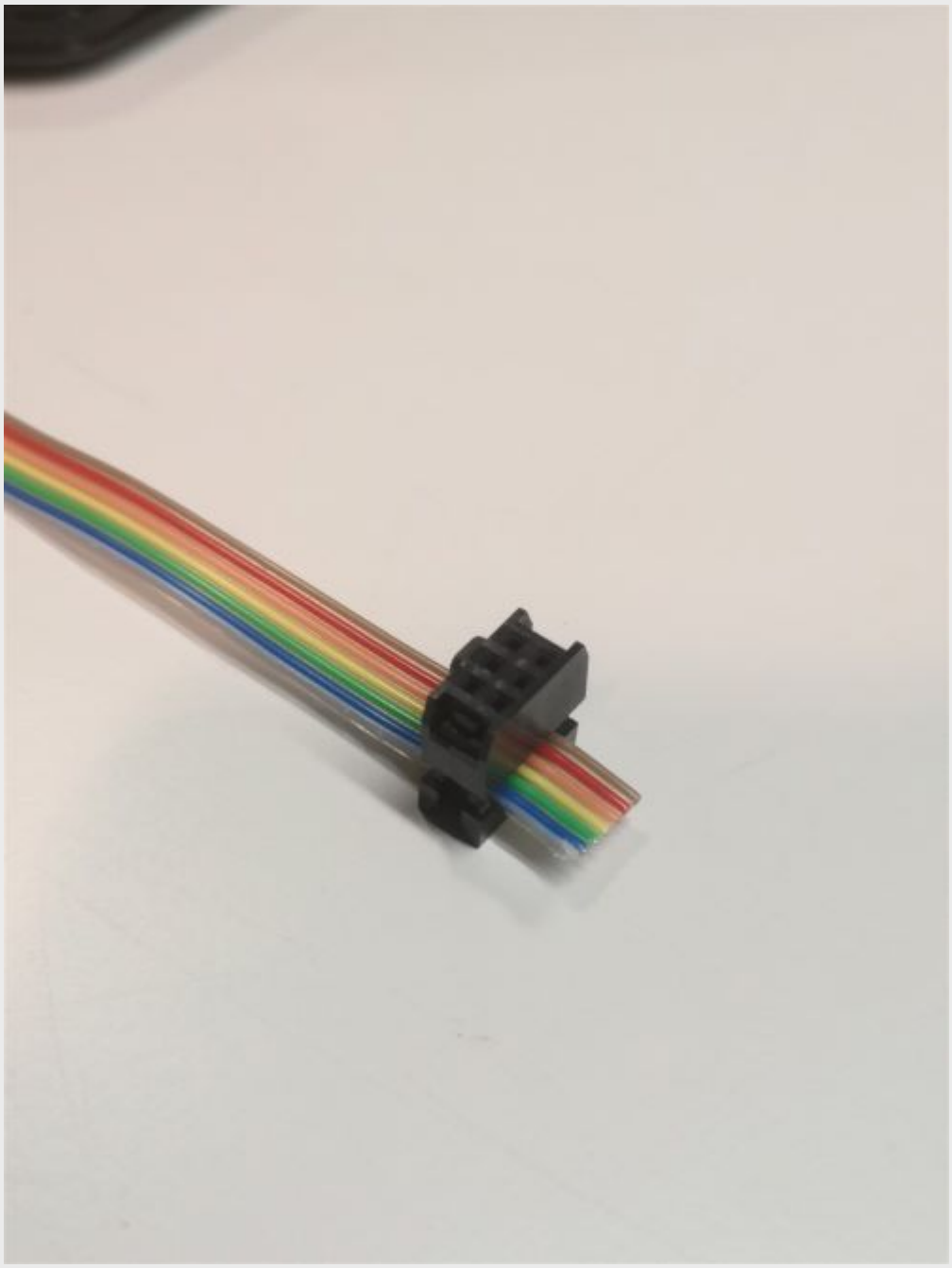
1	 Cut 4 ribbon cables composed of 16 wires each to the proper length (about 1.5m). Each wire corresponds to an electrode.

2	 Crimp the ribbon cable on the corresponding idc connector with a suitable clamp. Pay attention to the direction ofthe cables. Unbalanced IDC connector. The ribbon cable must be perpendicular to the connector.

3 Example of IDC connector mounting. The wires should run as perpendicular as possible to the IDC connector.

Warning: In MUX2024, the wiring of the electrodes from the IDC connector follows the order below (different from MUX2023). Take this into account if you wire your ribbon cable to further connectors or screw terminals.


TOP view MUX2024
15
13
11
9
7
5
3
1
16
14
12
10
8
6
4
2

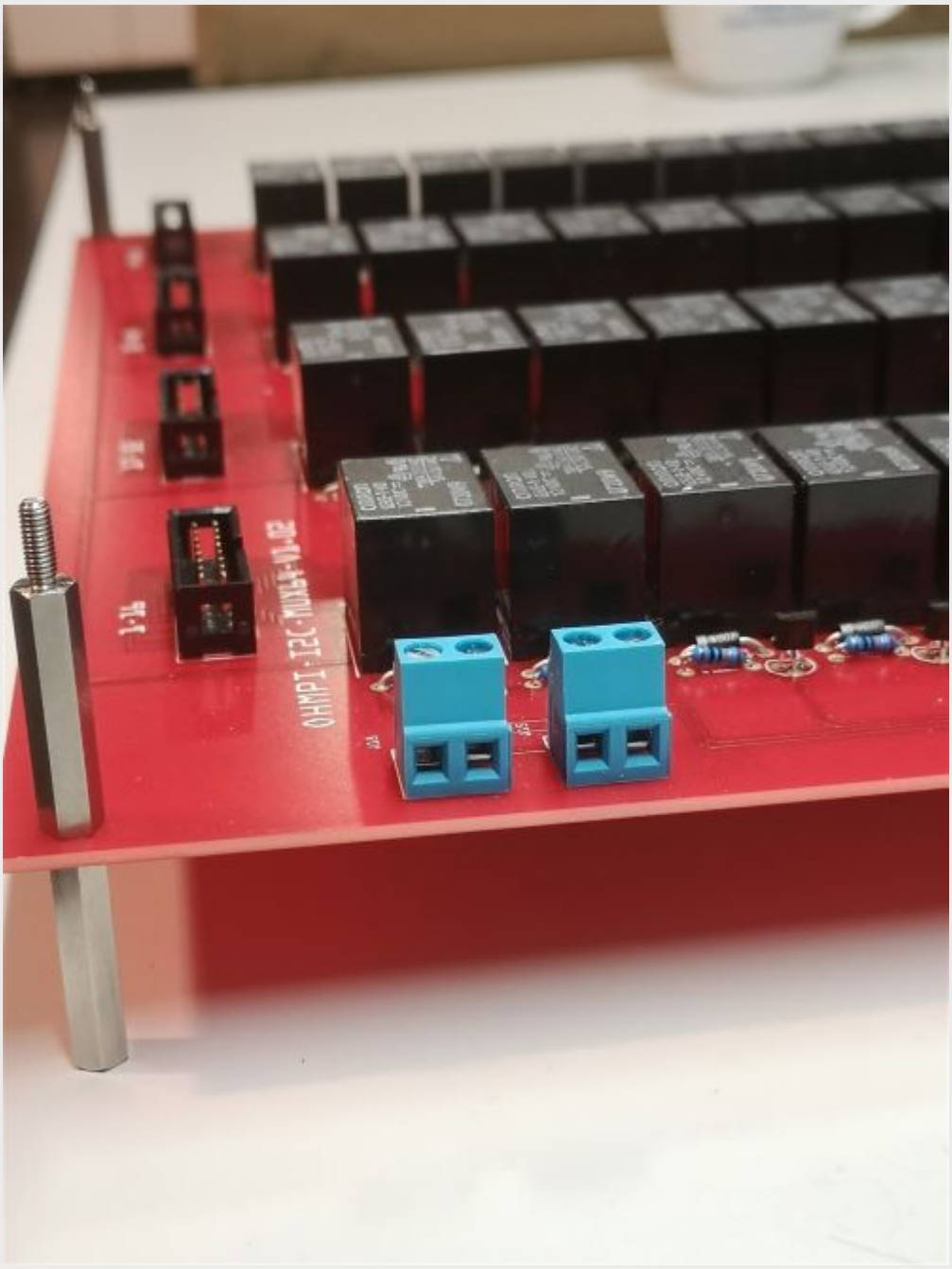


Same for a 6 wires ribbon cable of 1 m length.

5 

Cut the ribbon cable flush with the IDC connector.

6	
	Position 9 spacers above the MUX board, and 9 spacers below



Profile view for mounting the spacers above and below.

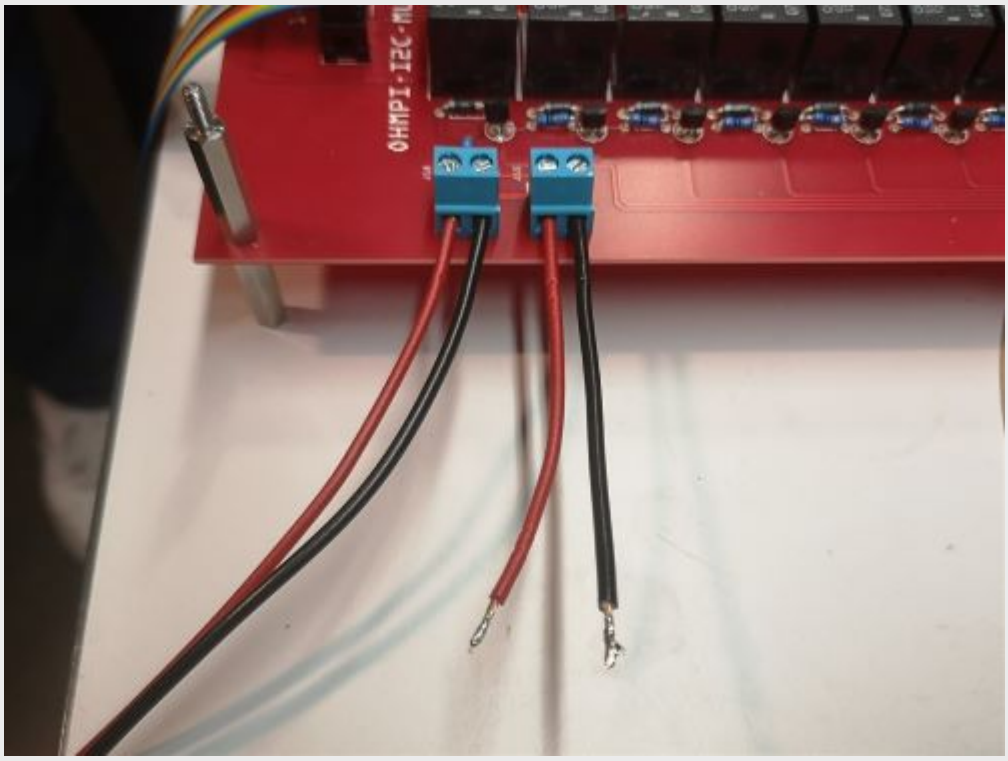
7	 Cut a 50 cm long wire of any color (here yellow). Strip and tin each end of the wire. Install the wire “A” on the screw terminal of MUX board « A »

8	
	Cut a red wire and a black wire of 50 cm length. Strip, tin and place the wires on the left screw terminal as shown in the picture: i)Red wire 12 V, ii) Black wire GND

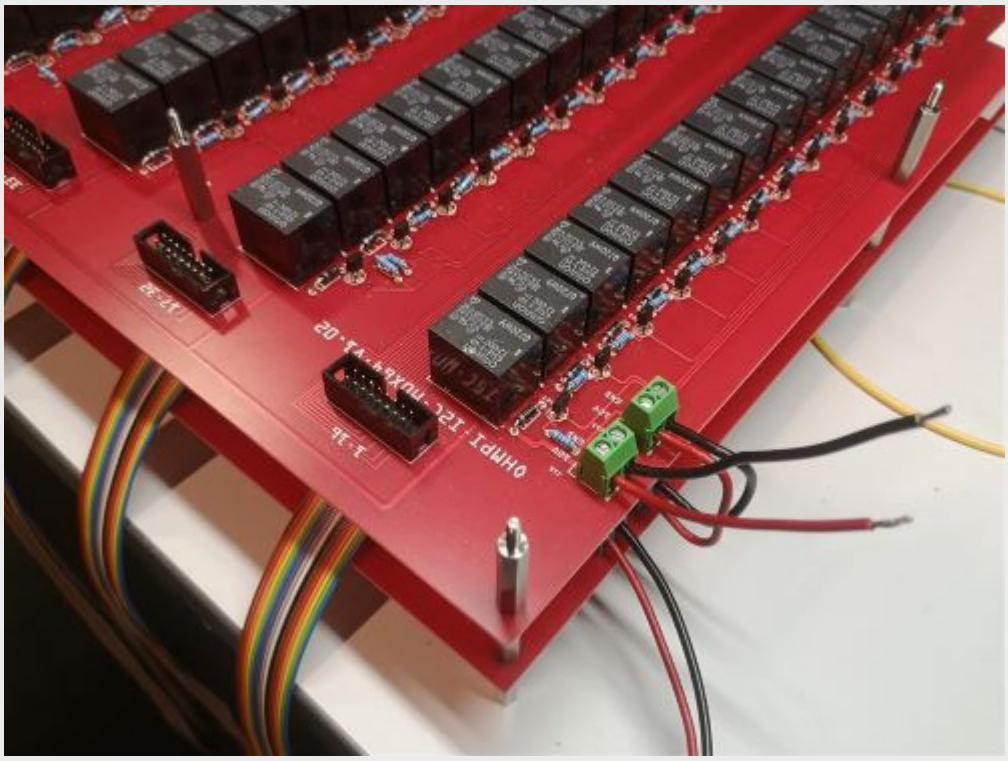
9	 Mount the 4 ribbon cables (16-wires each) with IDC connectors. A small noise is often heard when the IDC connector is clipped in place.

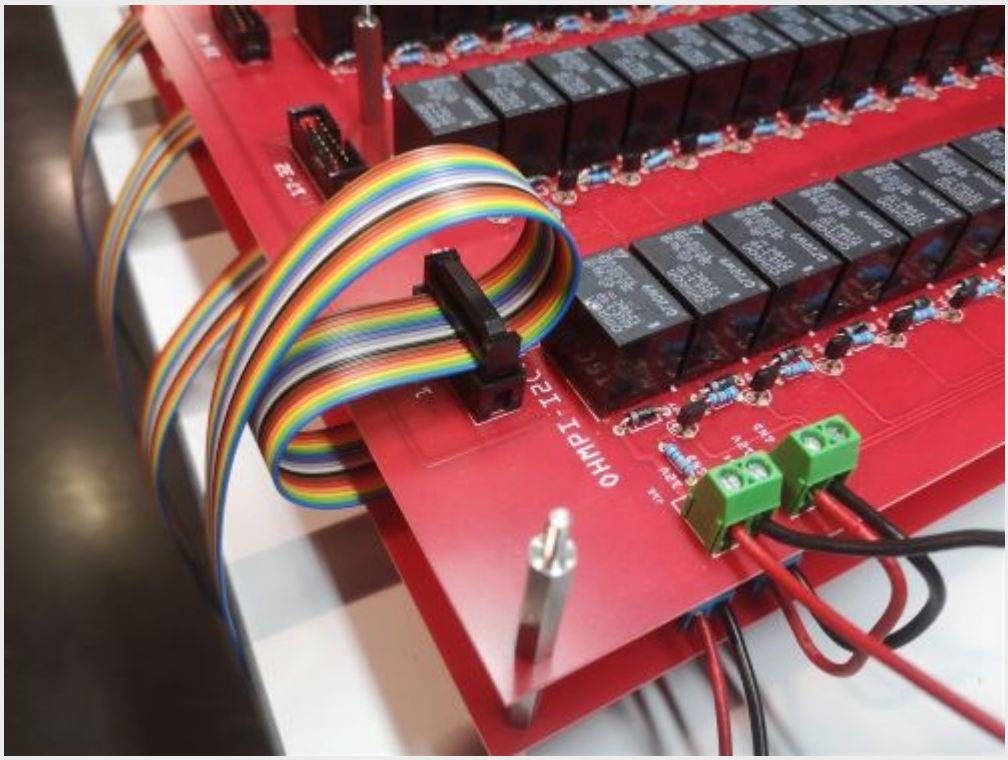
10

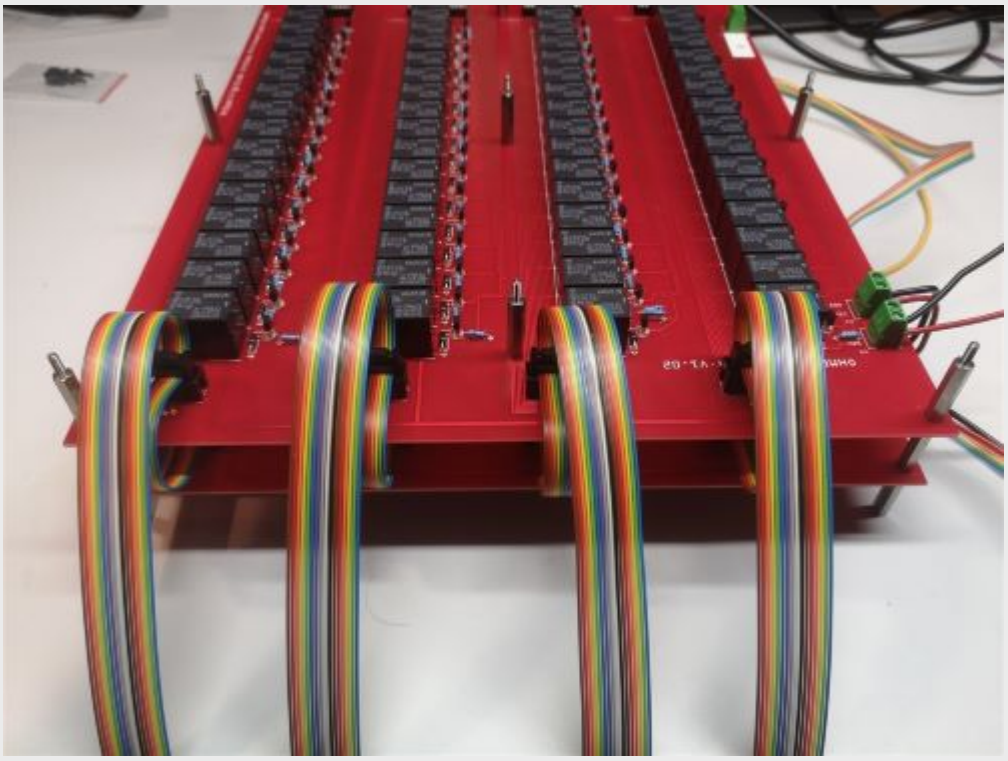
Mount the ribbon cables with 6-wires with the corresponding IDC connectors

11	 Cut a red wire and a black wire of 10 cm length. Strip and tin the wires at the ends. Mount the red wire on the 12V input and the black wire on the GND input on the right screw terminal.

12	 Mount and fix the second MUX board “B” on the first with the help of 9 spacers.

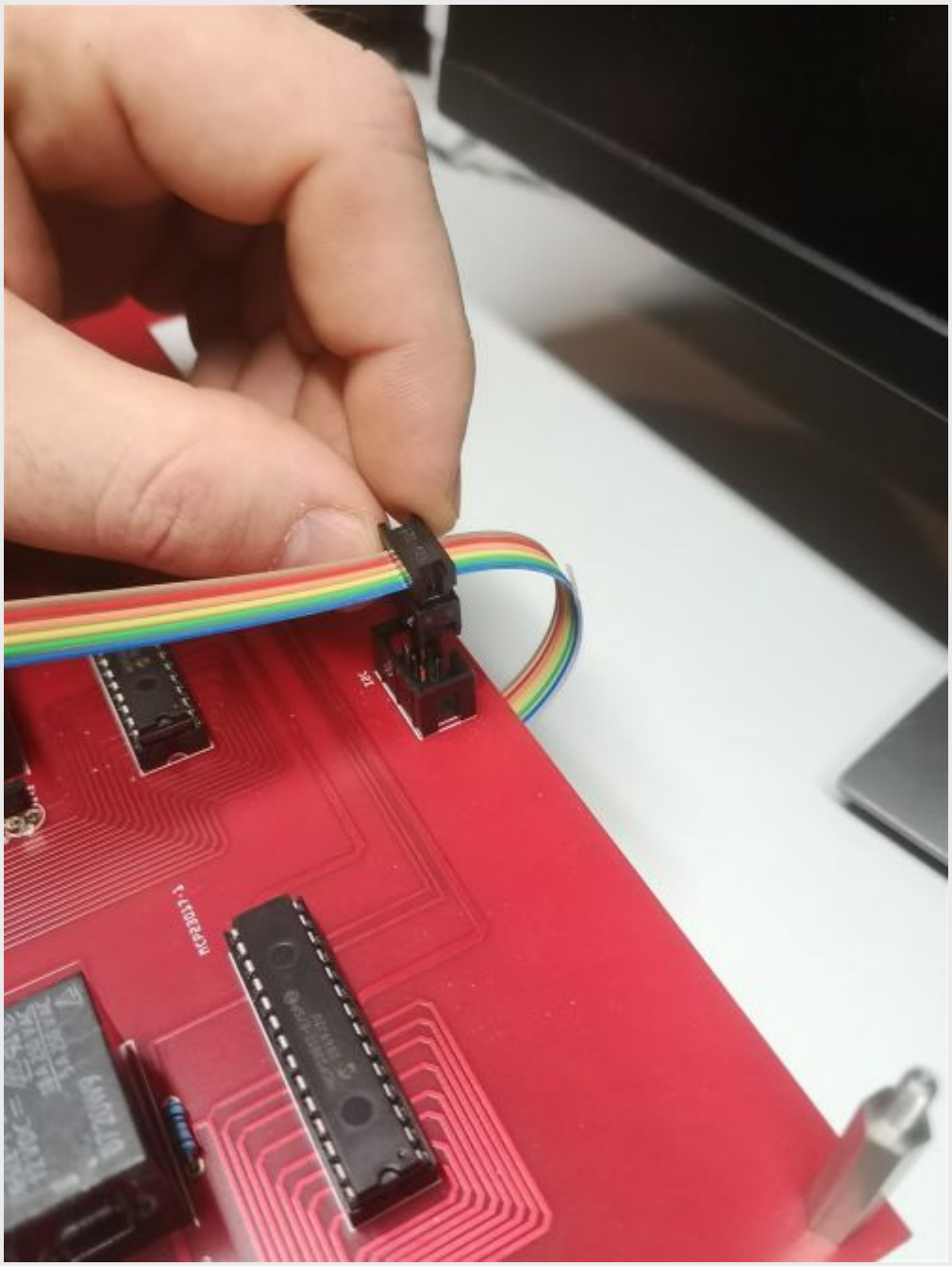
13		Cut, strip and tin a red wire and a black wire of 10 cm length. Mount the wires on the left screw terminal. Red wire 12V input, black wire GND input. Connect the red and black wires from board A to the right screw terminal of board B. Red wire 12V input. Black wire GND input.

14		Crimp a 16 wires IDC connector on the ribbon cable at about 15 cm from the previous connector. Please, pay attention to the direction of the cable before the crimp procedure. Mount the ribbon cable on the IDC connector on the board.

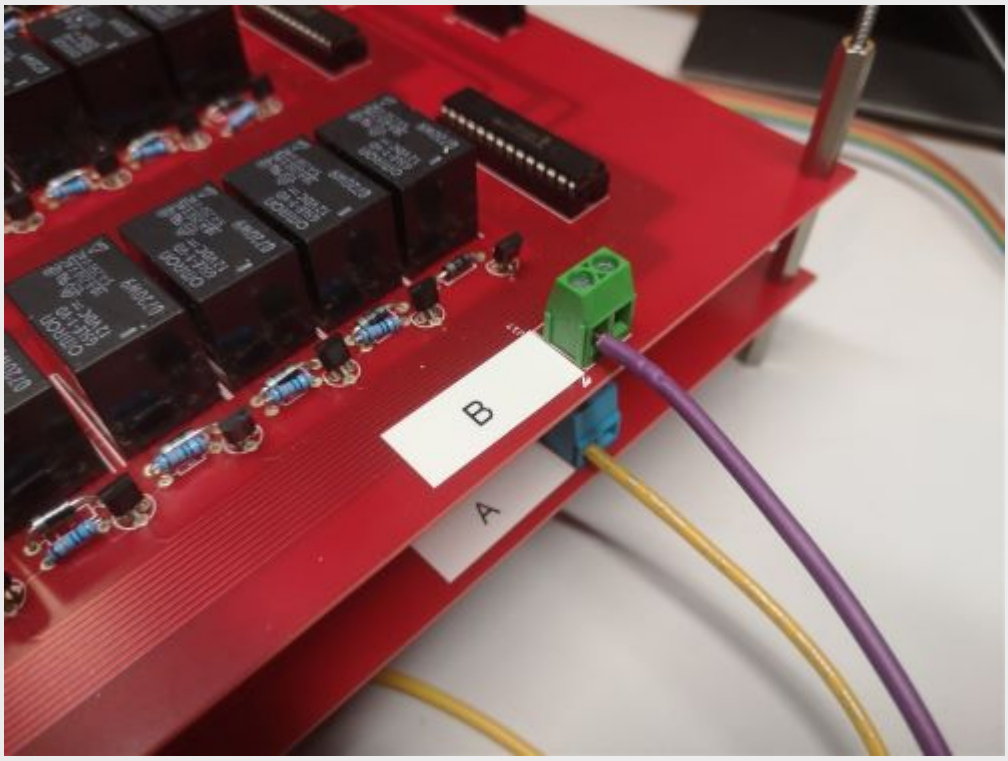
15 

Repeat the operation for the other 3 ribbon cables.

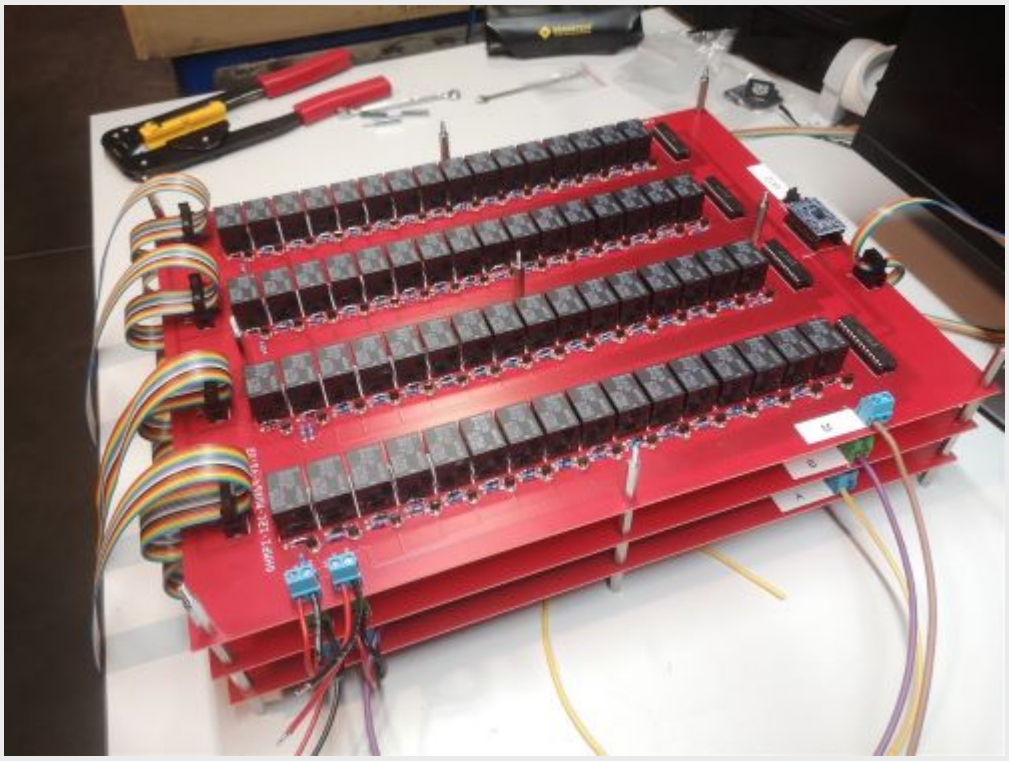
16



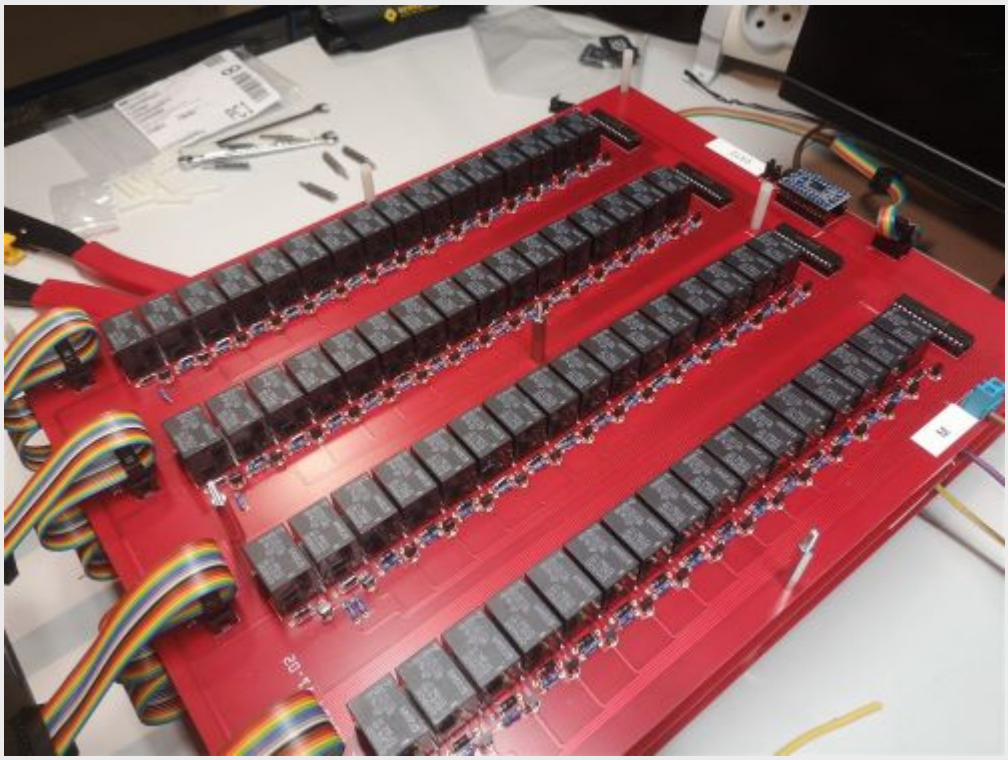
Repeat the operation for the 6 wires ribbon cable.

17	 Cut a 50 cm long wire “here purple” (Color not relevant but to be defined). Strip and tin the wire at its ends. Position the wire on the input B of the screw terminal of the multiplexing board B.

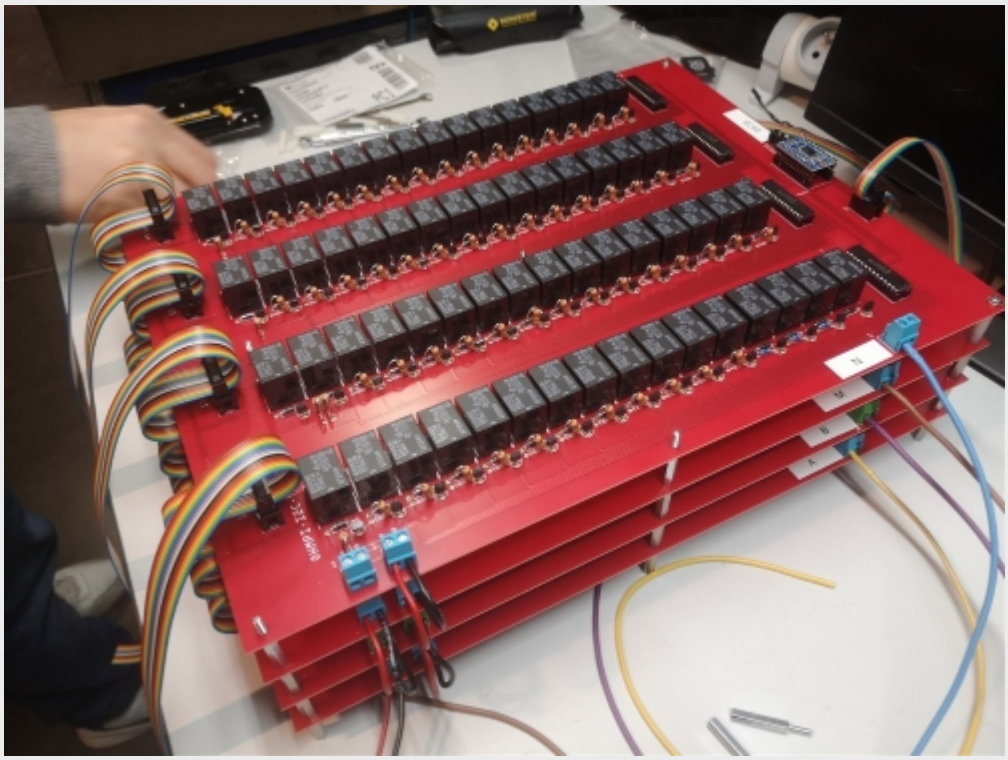
18



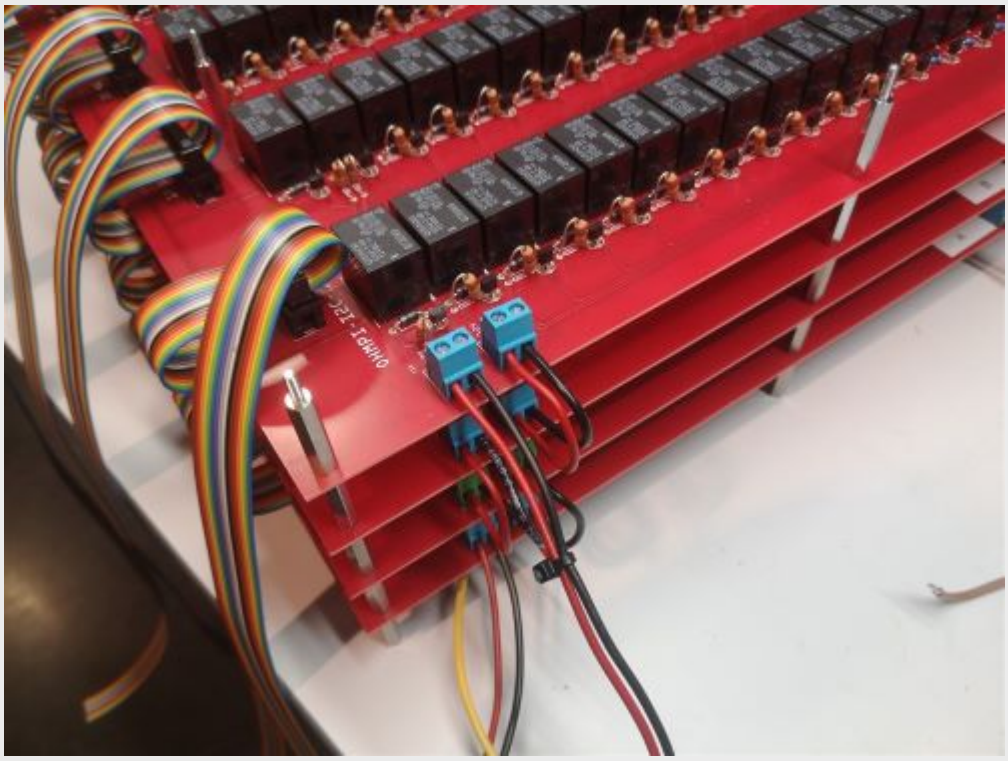
Repeat all these operations for the third MUX board called “M”.

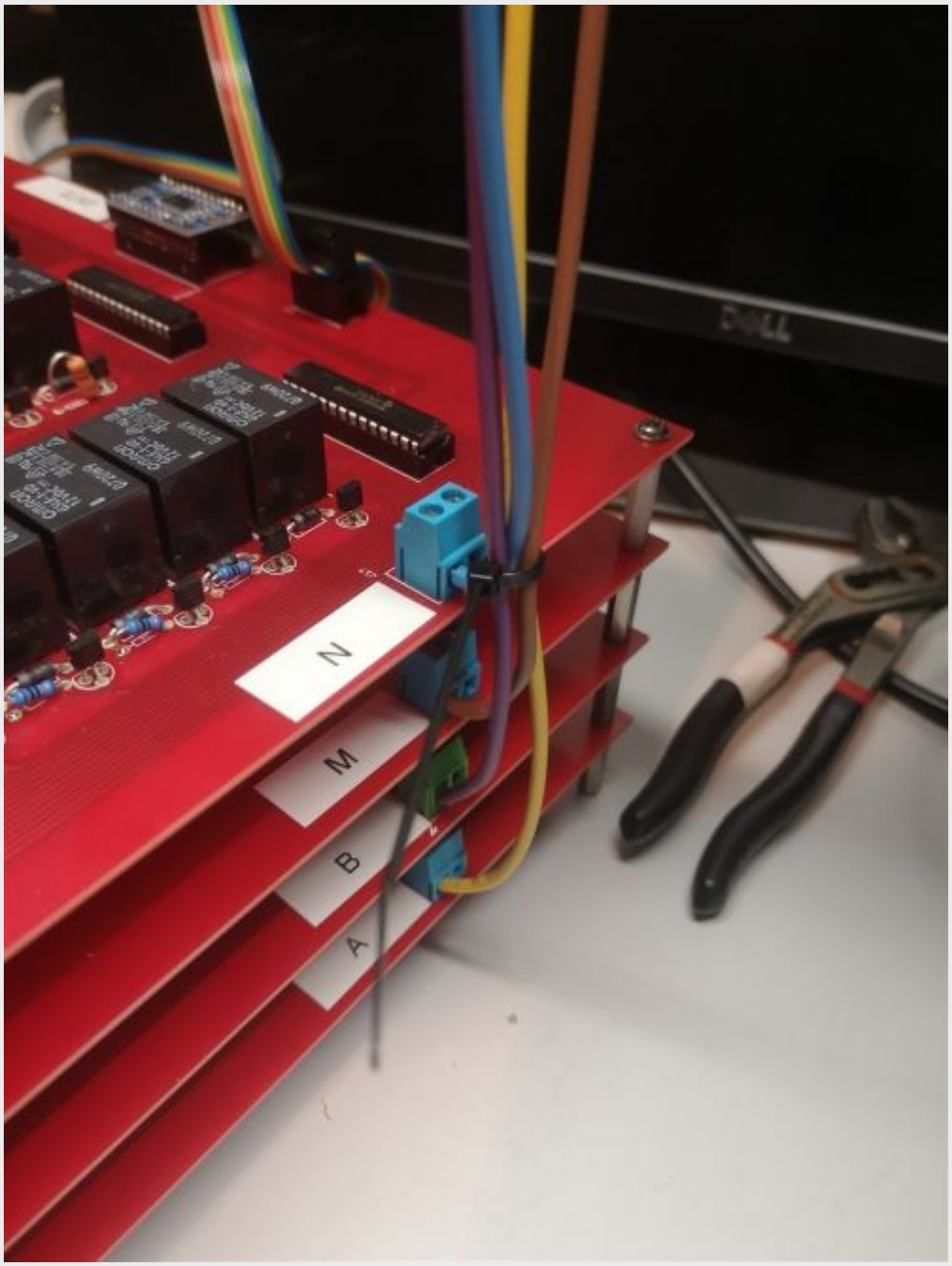
19		Repeat the operations for the fourth MUX Boards. Attention, it is necessary to position 5 different spacers (here nylon screw hex spacers) in between the “M” board and the “N” MUX Board (as shown on the photograph). Refer to the following photographs for more details on the assembly of the spacers

20



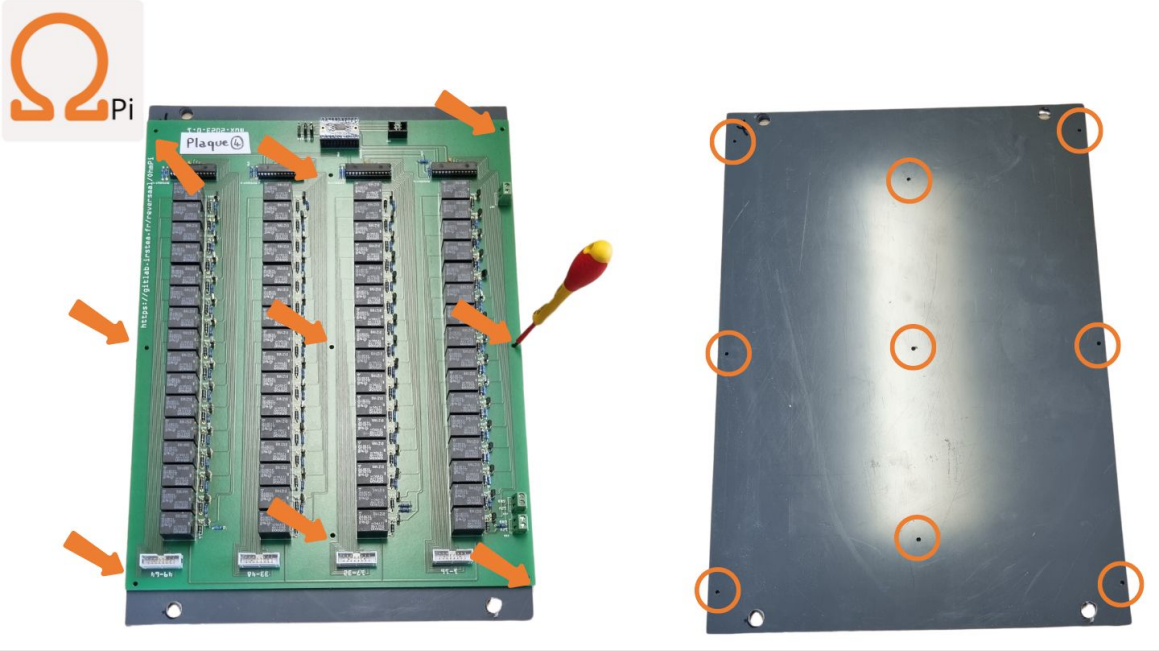
When mounting the 4th MUX board (“N”), screws can be placed on the nylon spacers to fix the boards together. Note that the other spacers could be used for this purpose.
Connect ribbon cables (16 wires) from board 3 to board 4 as previously described. Connect the red wire (12V) of MUX board “M” to the 12V terminal of the right screw terminal of MUX Board “N”. Connect the black wire (GND) of MUX board “M” to the GND screw terminal on MUX board “N”.

21	 Cut a red wire and a black wire of one meter length. Place the red wire on terminal “12V” and the black wire on terminal “GND” of the left screw terminal. Tie the wires together.



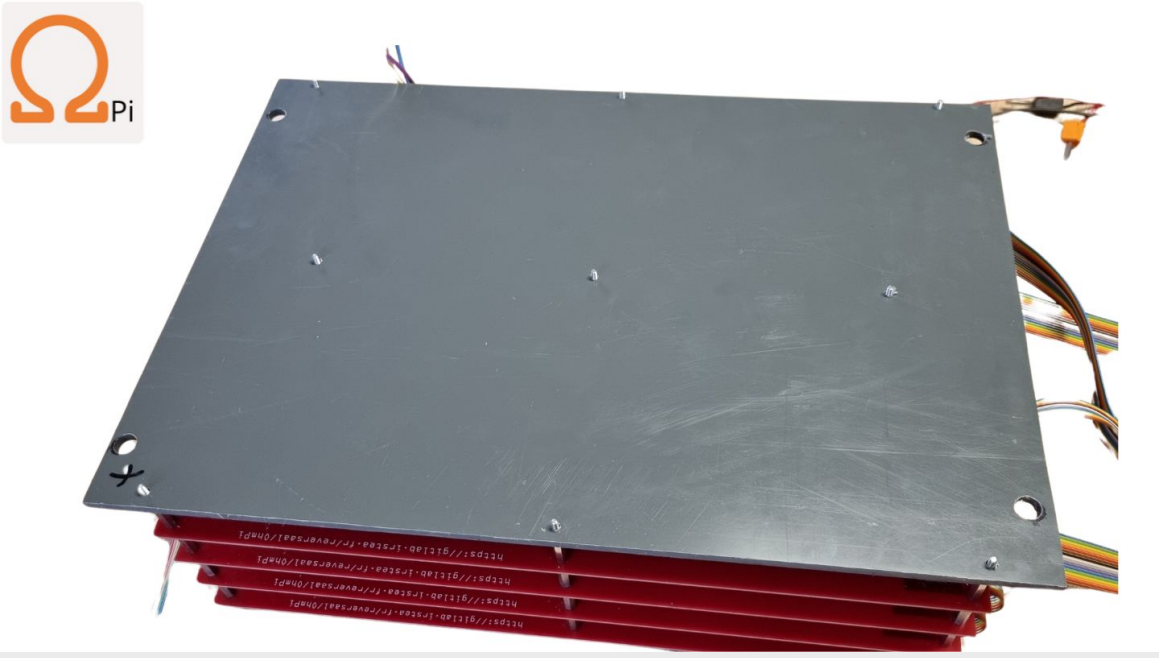
22 Tie the A, B, M and N wires together

23



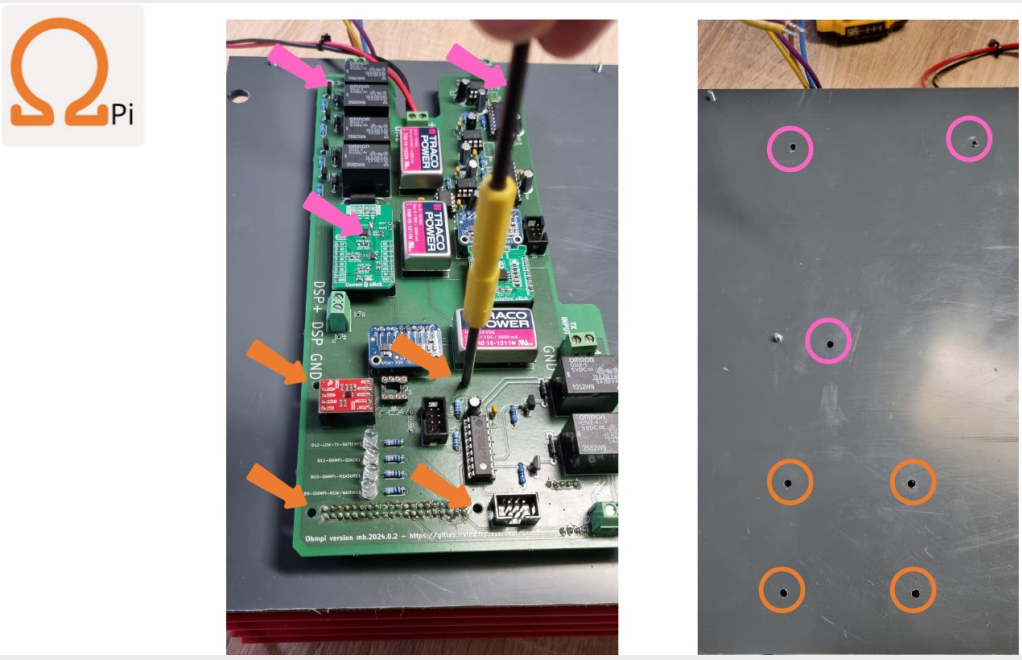
Cut a PVC/wood plate with the following minimum dimensions : 410 mm * 280 mm * 4 mm, and drill hole (M 3.5 mm)

24



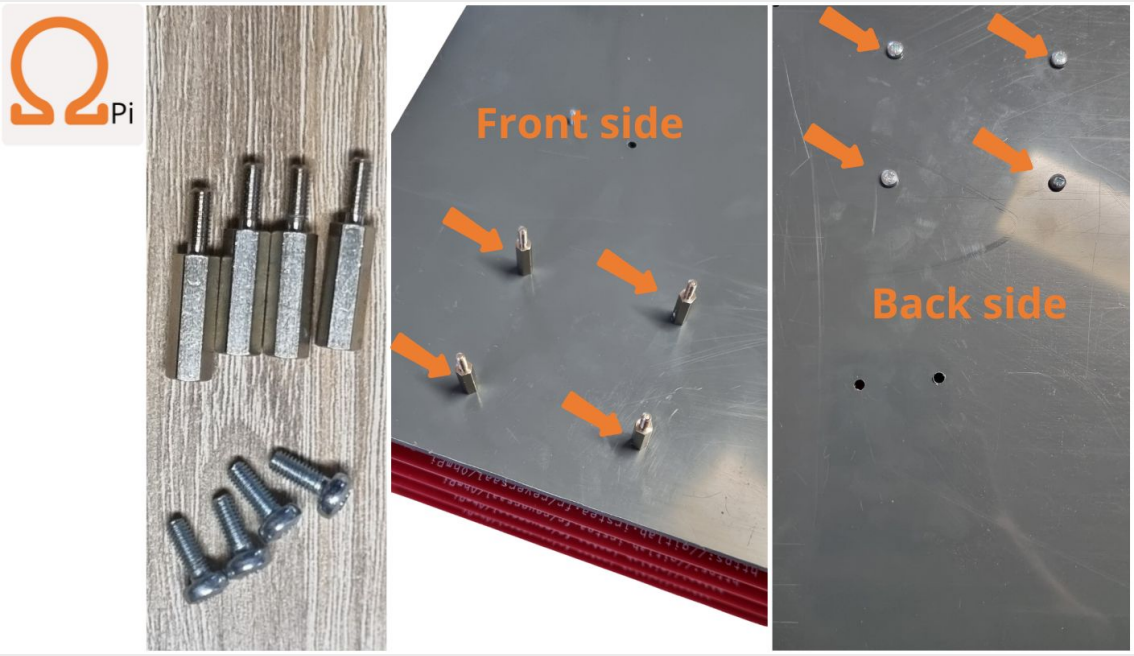
Fix the PVC plate

25



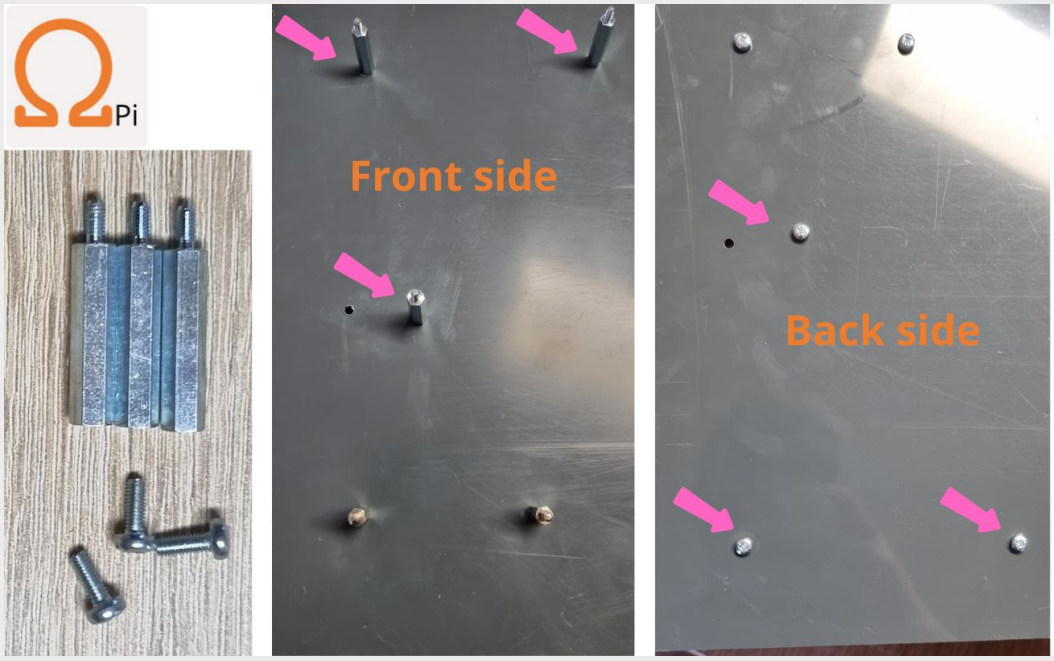
Drill holes for fixing Raspberry Pi and measurement board

26



Install spacer for Raspberry Pi on the pvc plate

27



Install spacer for measurement board on the pvc plate

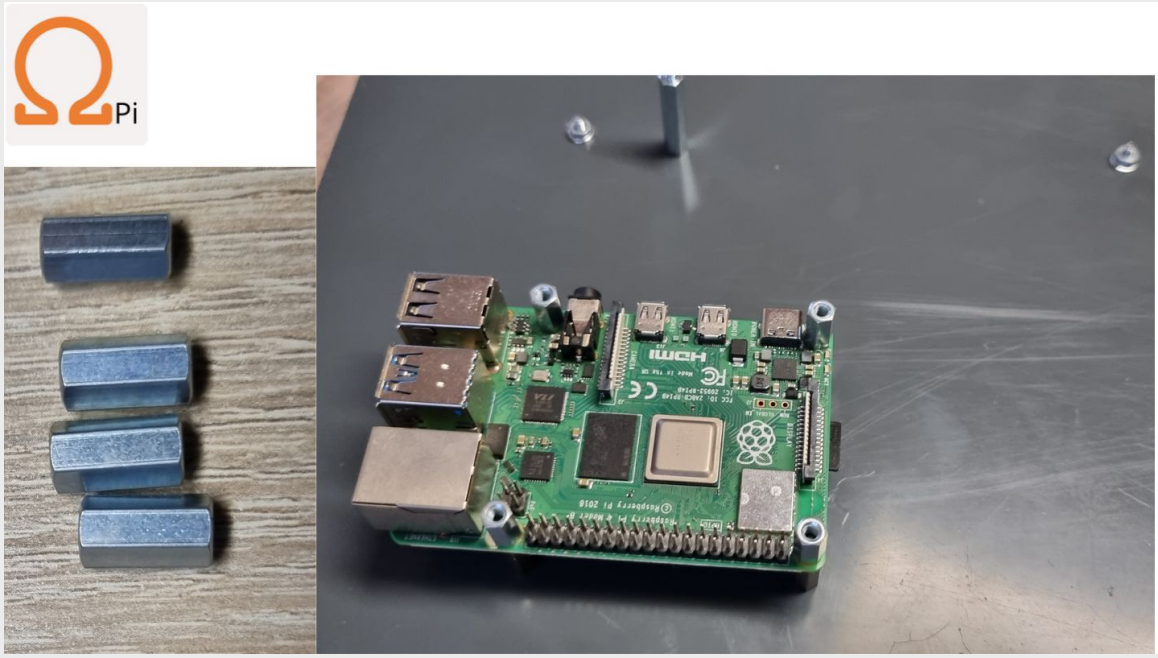
28

Fit 9 flat washers and nuts (M3)

29

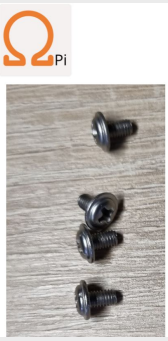
Install Raspberry Pi

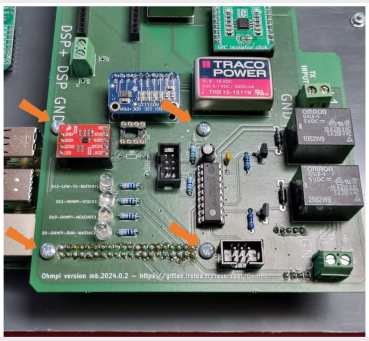
30



Fit 4 spacers (female/female, M3, 11 mm)

31



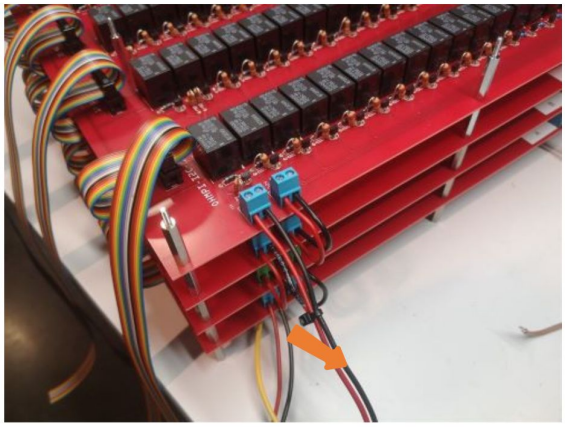


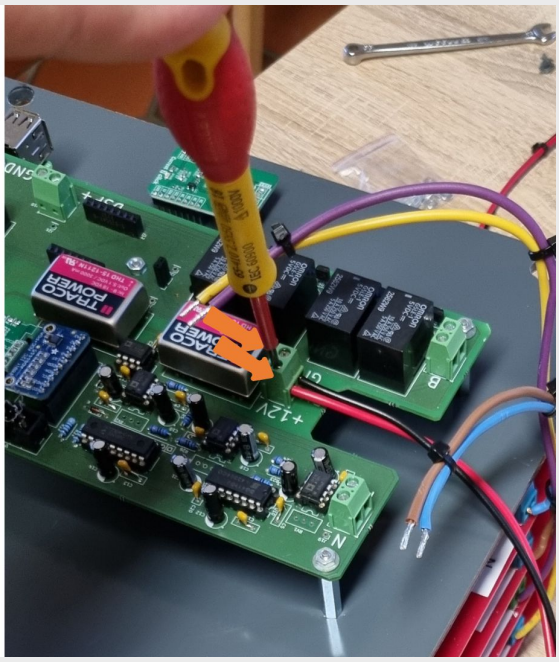
Install the measurement board on the Raspberry Pi, and fix the 4 screws (M3).

32

Fit 3 flat washers and nuts (M3) for measurement board.

33



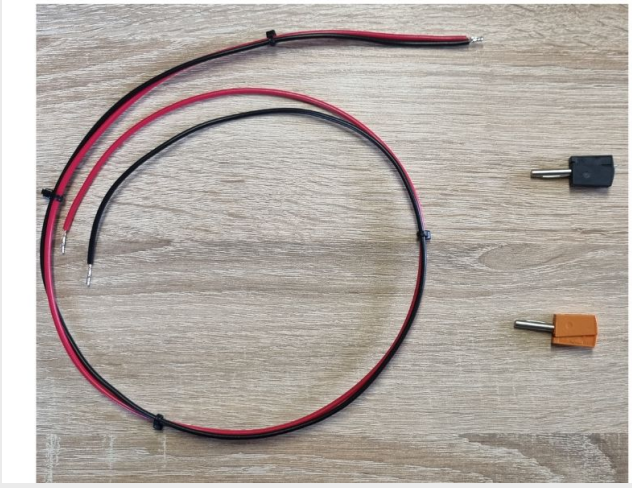


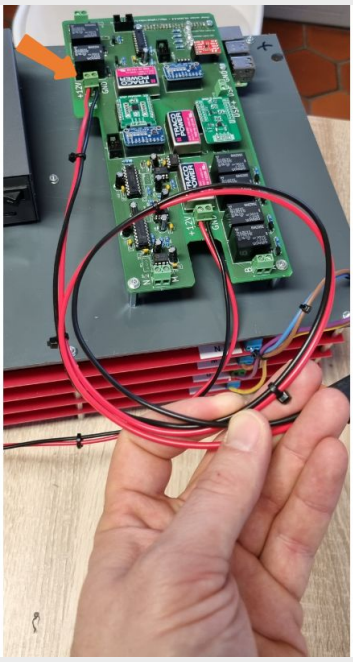
Connect 12V and GND cable from Mux to Measurement board

34

The choice is yours: position or fix the DPH5005.
connect USB cable between DPH5005 and Raspberry Pi

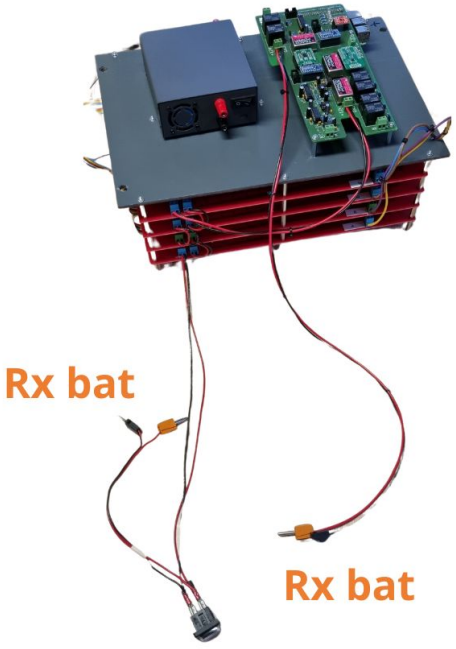
35





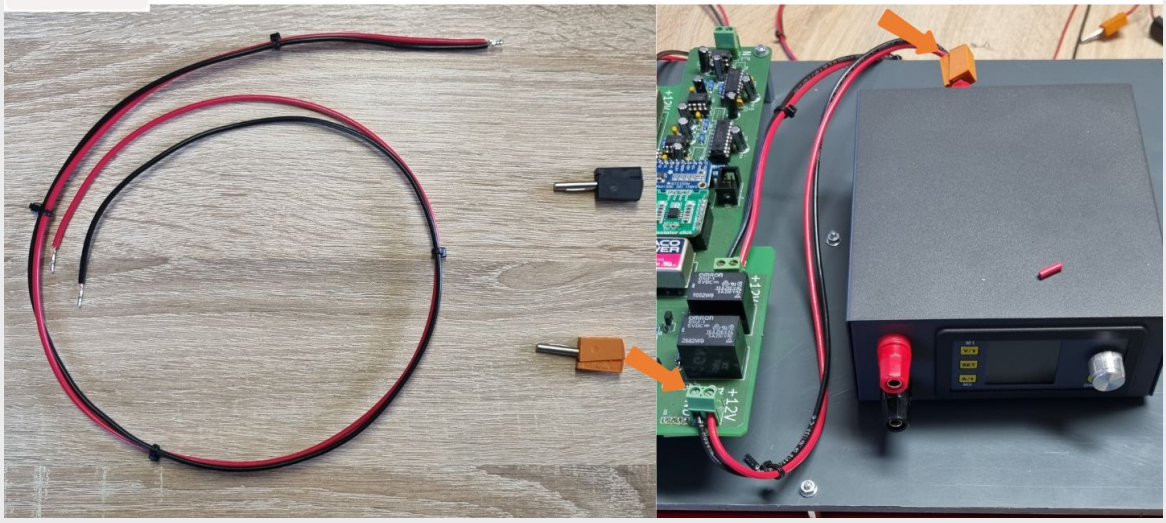
Prepare two wires (30 cm, 1.5 mm^2, black and red), and and install two banana plugs

36



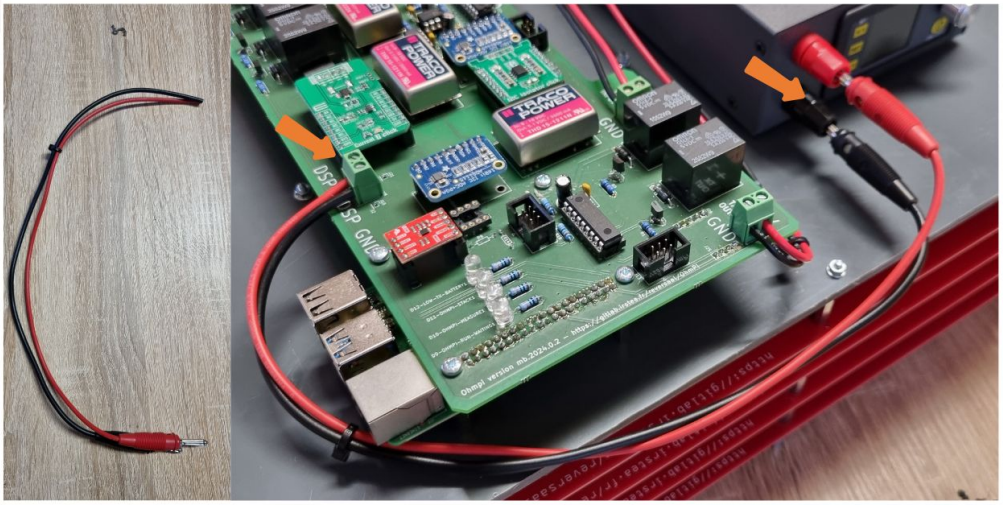
This is optional, but you could install a switch on the cable connecting to the 12V RX battery.

37

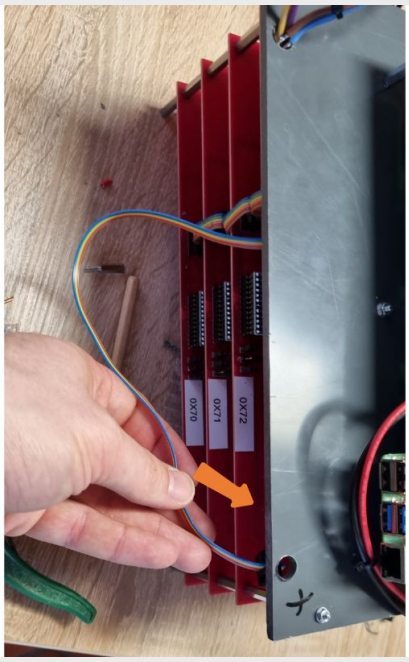
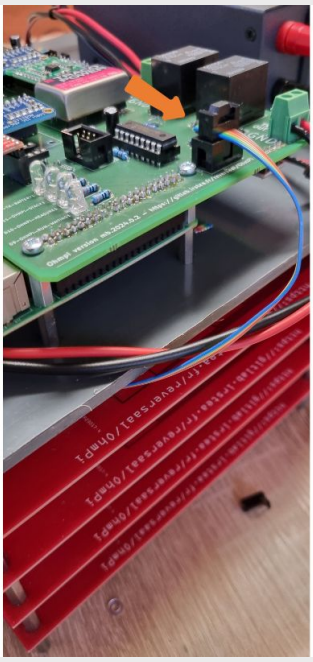
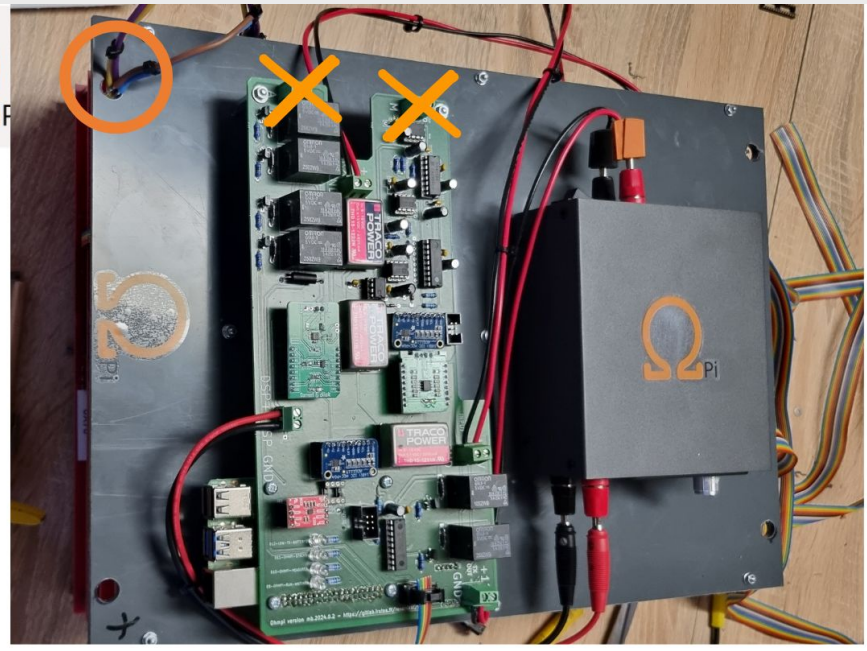


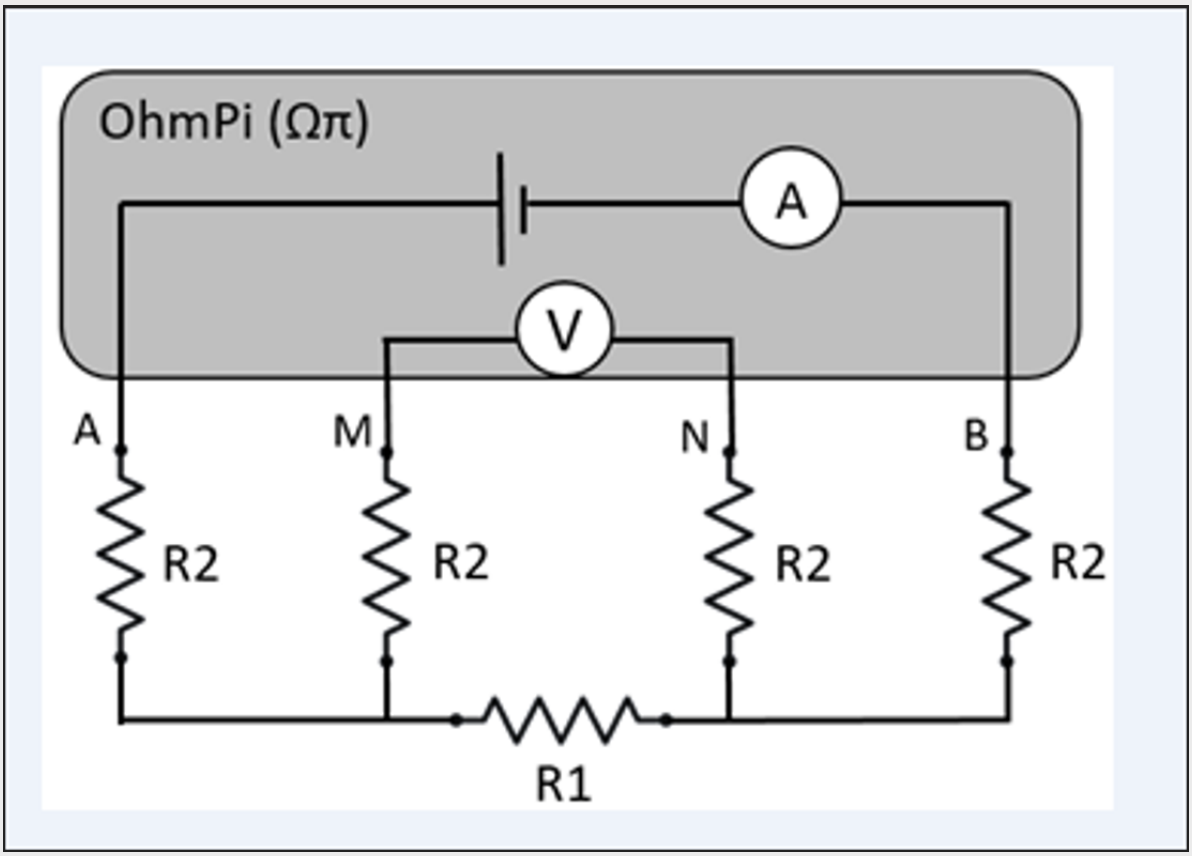
Prepare two wires (~15 cm, 1.5 mm^2, black and red), and and install two banana plugs and connect the measurement board and the input of DPH5005 (on the back side)

38

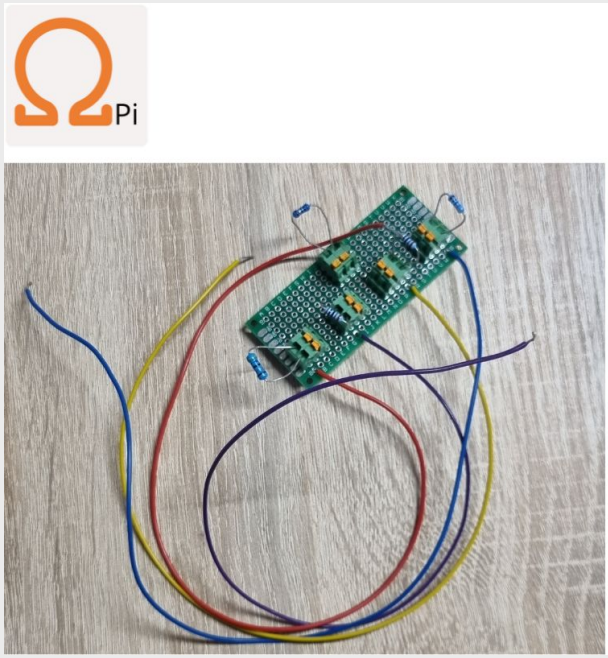


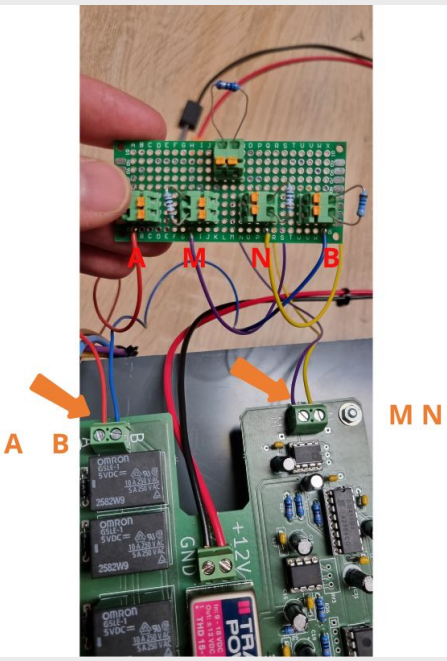
Prepare two wires (~20 cm, 1.5 mm^2, black and red), and and install two banana plugs and connect the measurement board (DPS+ and GND) and the output of DPH5005 (front side)

39	 	slide ribbon cable between MUX N and PCV plate, and connect ribbon cable to IDC connector
40		Do not connect the MUX electrode cables to the measurement board.



40





Connect a equivalent circuit
R2=1kOhm R1=100 ohm

Warning: At this point in the build, we consider that you have followed the instructions in *Getting started* section

Please connect both 12 V Battery for RX and TX.

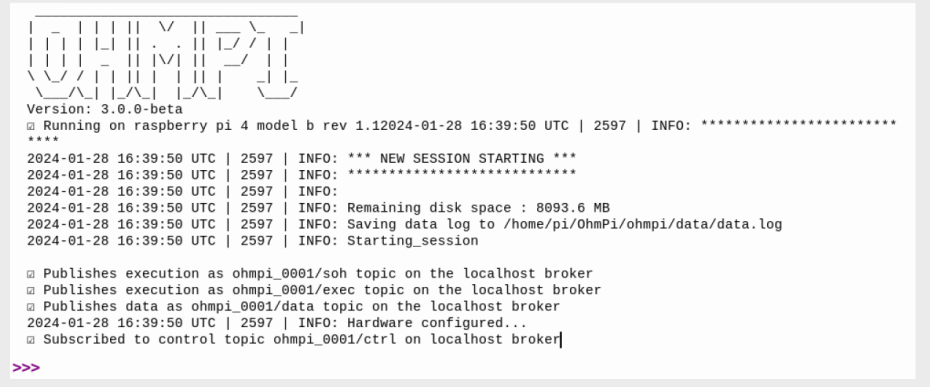
For direct use of Raspberry Pi Connect Screen, mouse and keybord, for remote control use SSH or VNC.

Now it is possible to carry out the first test on a reference circuit.

Write de following python script your OhmPi folder

```
import os
import numpy as np
import time
import matplotlib.pyplot as plt
os.chdir("/home/pi/OhmPi")
from ohmpi.ohmpi import OhmPi
k = OhmPi()
```

41



If everything is ok, you get the message upper, if not check all cable, and battery or refer to troubleshooting

```
k.test_mux()
```

You should hear each of the 256 MUX board relays activate and deactivate 1 at a time.

```
k.run_measurement(quad=[1,4,2,3], tx_volt = 5., strategy = 'constant', dutycycle=0.5)
```

A measurement will start, and you should obtain your first measurement, with a value of R = 100 ohm (R1 on the equivalent circuit).

If not check, your cable connection and batteries

You can now connect the 4 cables of each MUX to the screw terminals of the measurement board identified ABMN.

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

Assembling an OhmPi Lite (32 electrodes)

1 mb2024 + 4 mux2024 + 1 DPH5005

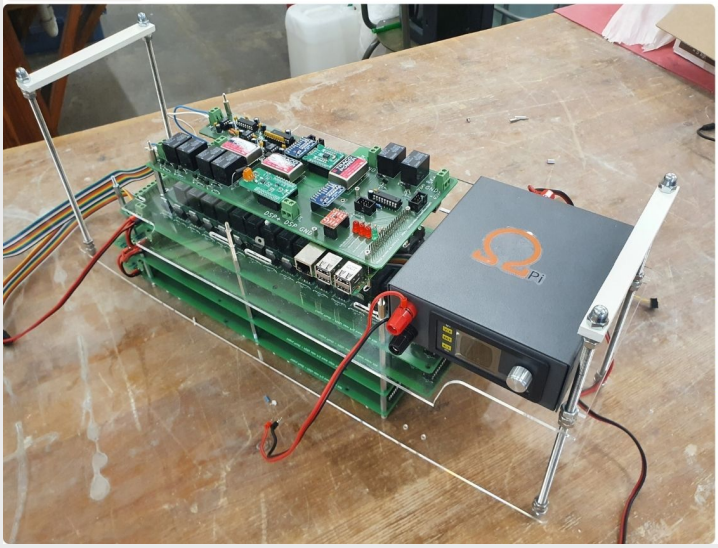
Here, we present how to build a 32-electrode OhmPi system using 1 mb.2024, 4 MUX.2024 and 1 DPH5005. This tutorial aims to illustrate one way of assembling a system using the MUX 2024 boards. It provides a working base, with room for improvement. Any idea to improve this design is welcome. Those who wish to build a 16-electrode OhmPi system can neglect steps 8-10, and adjust cable lengths/numbers accordingly.

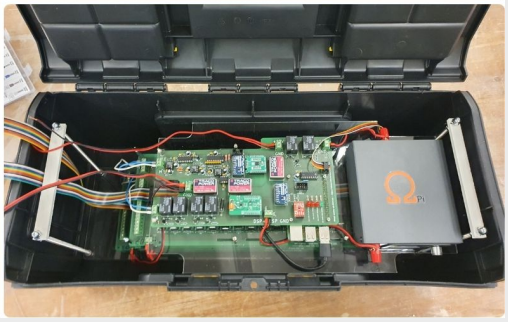
Warning: In this set-up, the MUX 2024 boards are configured in 2-role modes (see *2-role configuration*), so one should make sure that the MUX board are properly configured before going any further. Mounting a system with MUX 2024 boards set-up in 4-role modes imply cabling the boards differently as to what is illustrated in this tutorial.

It is also best to having tested each MUX board individually before mounting them together.

32-electrode OhmPi system
(OhmPi Lite)
MB.2024 + 4 MUX boards 2024 + 1 DPS 5005

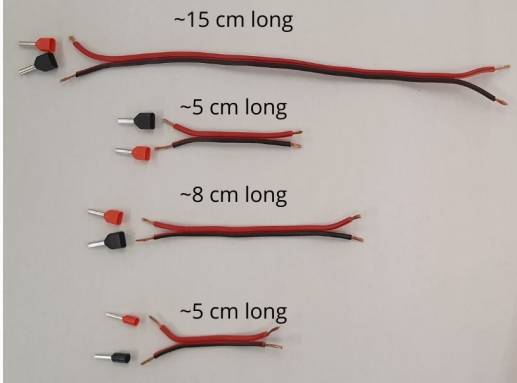
Fitting in a Stanley toolbox
59.5 x 28.1 x 26 cm (ref.: 1-79-218)





1. Prepare the power cables

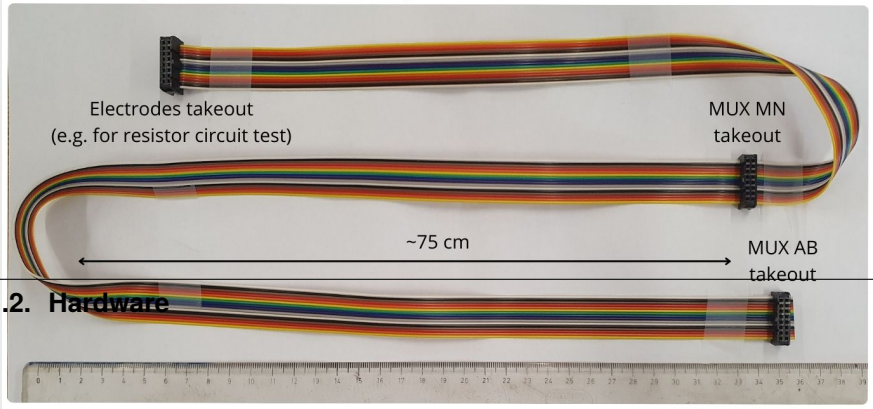
12 V Power cables for MUX boards
(e.g. 2 core loudspeaker cable)



Crimping pliers
to crimp multicore wires end

2. Prepare the ribbon cables

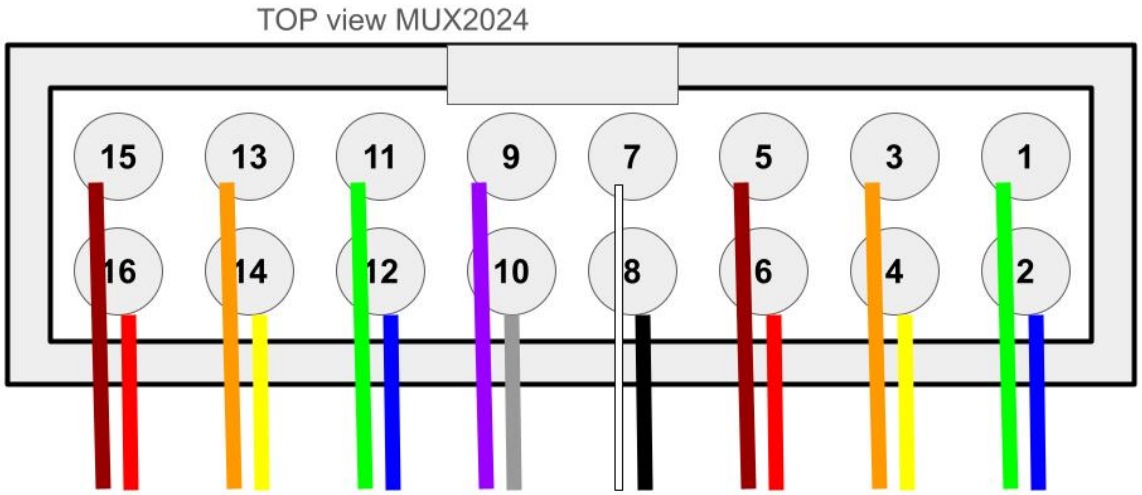
16-way ribbon cable for connecting 1 MUX board AB with 1 MUX board MN



Make sure that the connectors are always on the same orientation with respect to the wires colour scheme (check with respect to latch position)

Connectors can face up and down but colours scheme with respect to latch needs to match

Warning: In MUX2024, the wiring of the electrodes from the IDC connector follows the order below (different from MUX2023). Take this into account if you wire your ribbon cable to further connectors or screw terminals.



Warning: At this point in the build, we consider that you have followed the instructions in *Getting started* section

Please connect both 12 V Battery for RX and TX.

For direct use of Raspberry Pi Connect Screen, mouse and keyboard, for remote control use SSH or VNC.

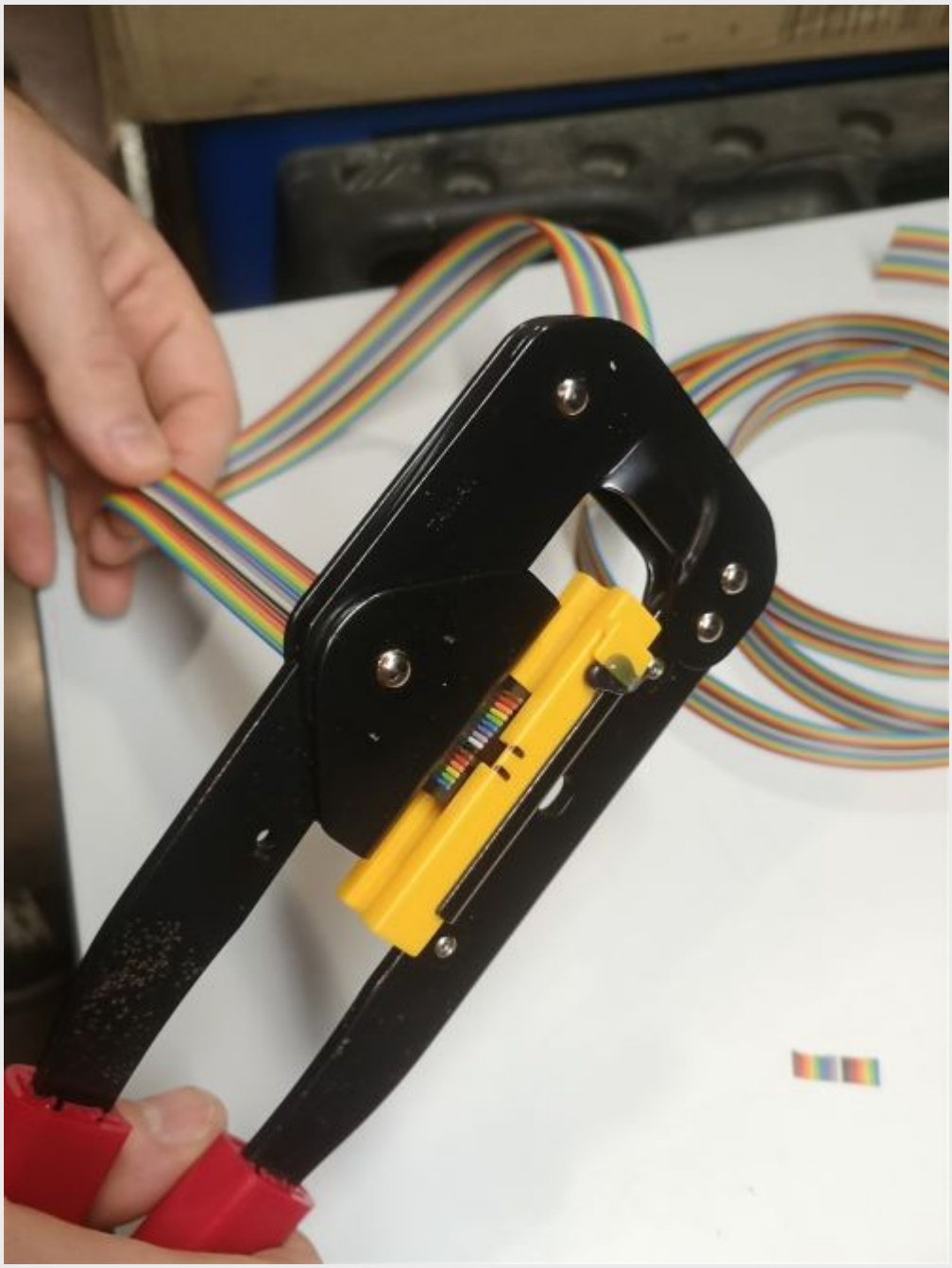
Now it is possible to carry out the first test on a reference circuit. See tests in *Assembling the 64-electrode OhmPi* for more details.

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

Assembling the OhmPi v2023

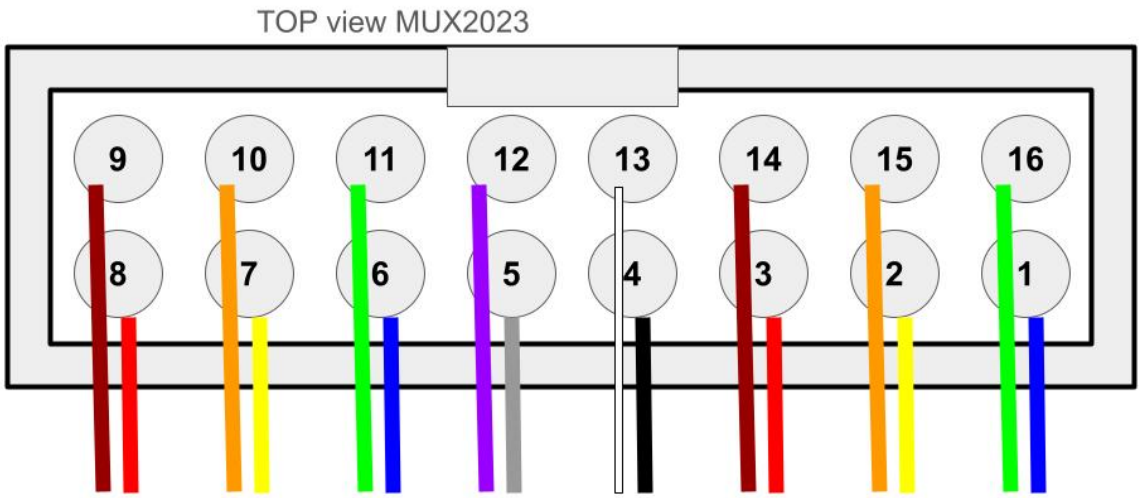
1 mb2023 + 4 mux2023 + 12V battery

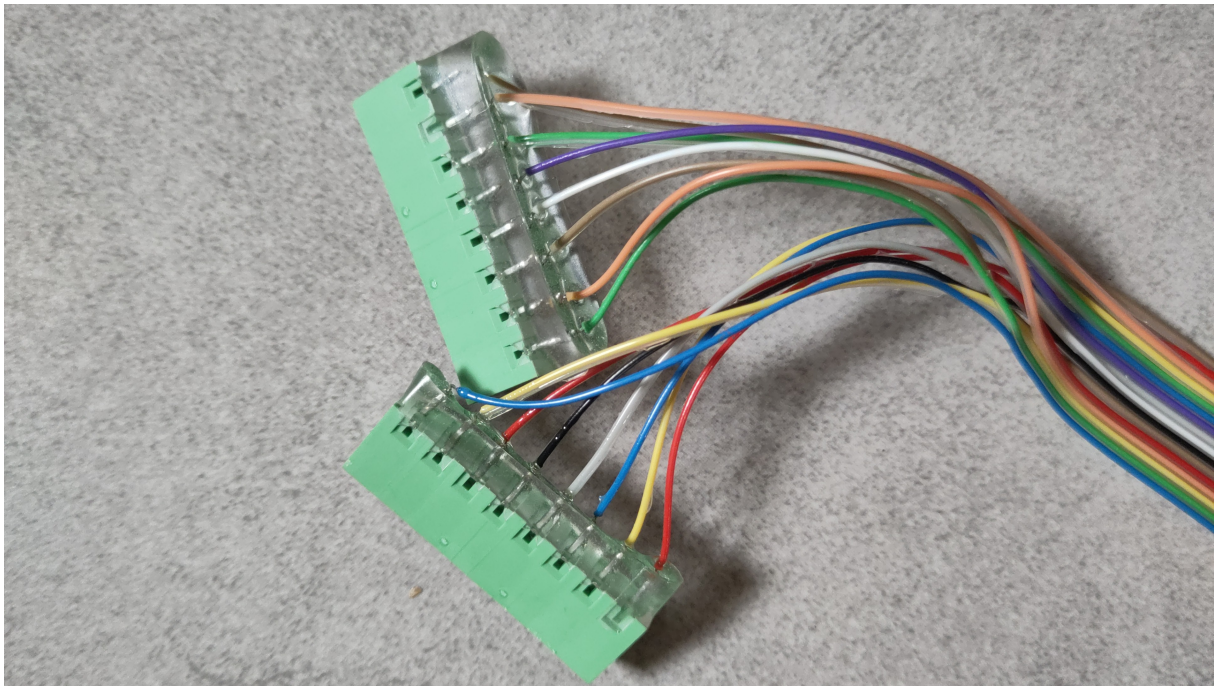
1	 Cut 4 ribbon cables composed of 16 wires each to the proper length (about 1.5m). Each wire corresponds to an electrode.

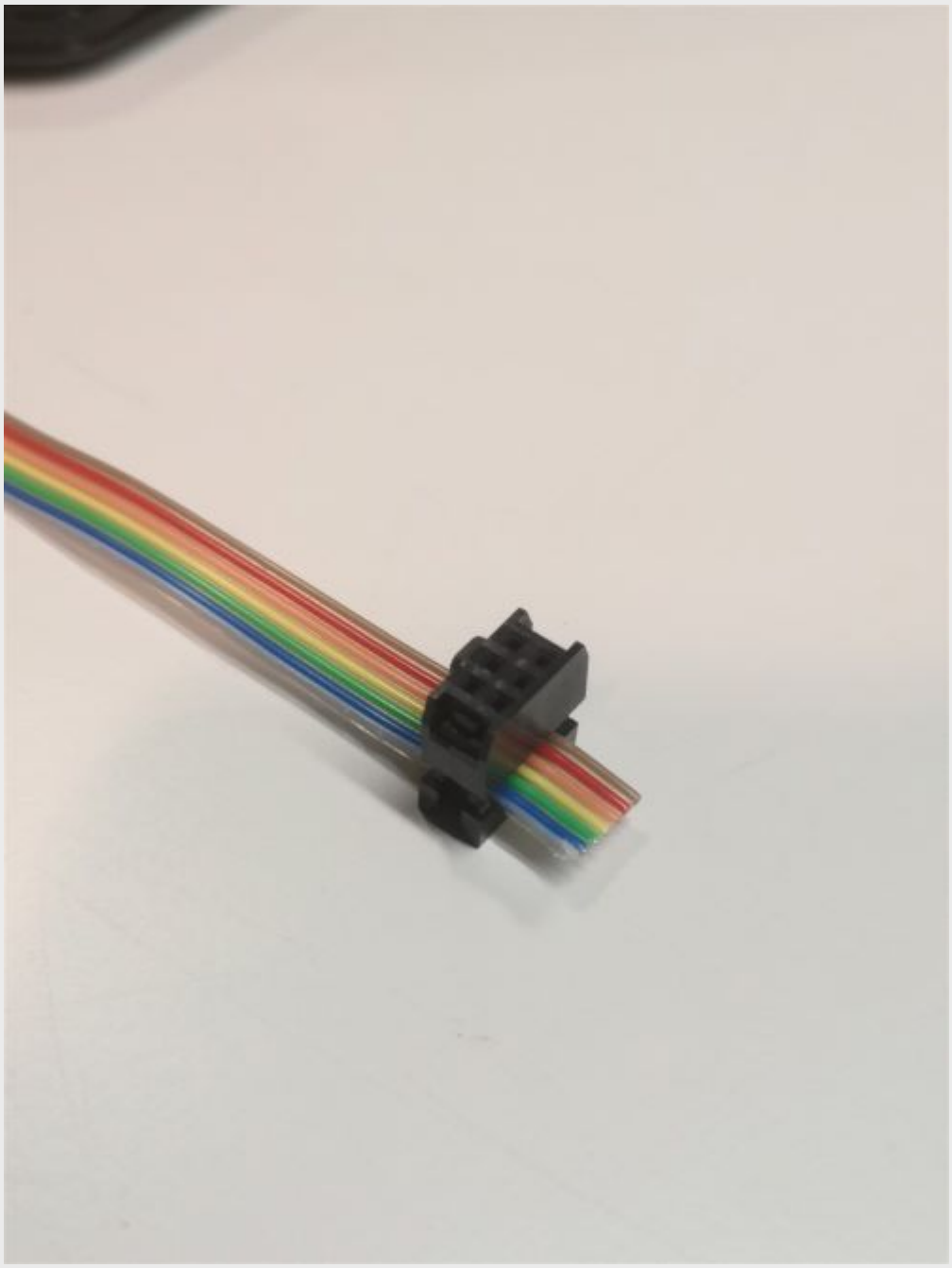
2	 Crimp the ribbon cable on the corresponding idc connector with a suitable clamp. Pay attention to the direction of the cables. Unbalanced IDC connector. The ribbon cable must be perpendicular to the connector.

3 Example of IDC connector mounting. The wires should run as perpendicular as possible to the IDC connector.

Warning: In MUX2023, the wiring of the electrodes from the IDC connector follows the order below (different from MUX2024). Take this into account if you wire your ribbon cable to further connectors or screw terminals.


TOP view MUX2023
9
10
11
12
13
14
15
16
8
7
6
5
4
3
2
1

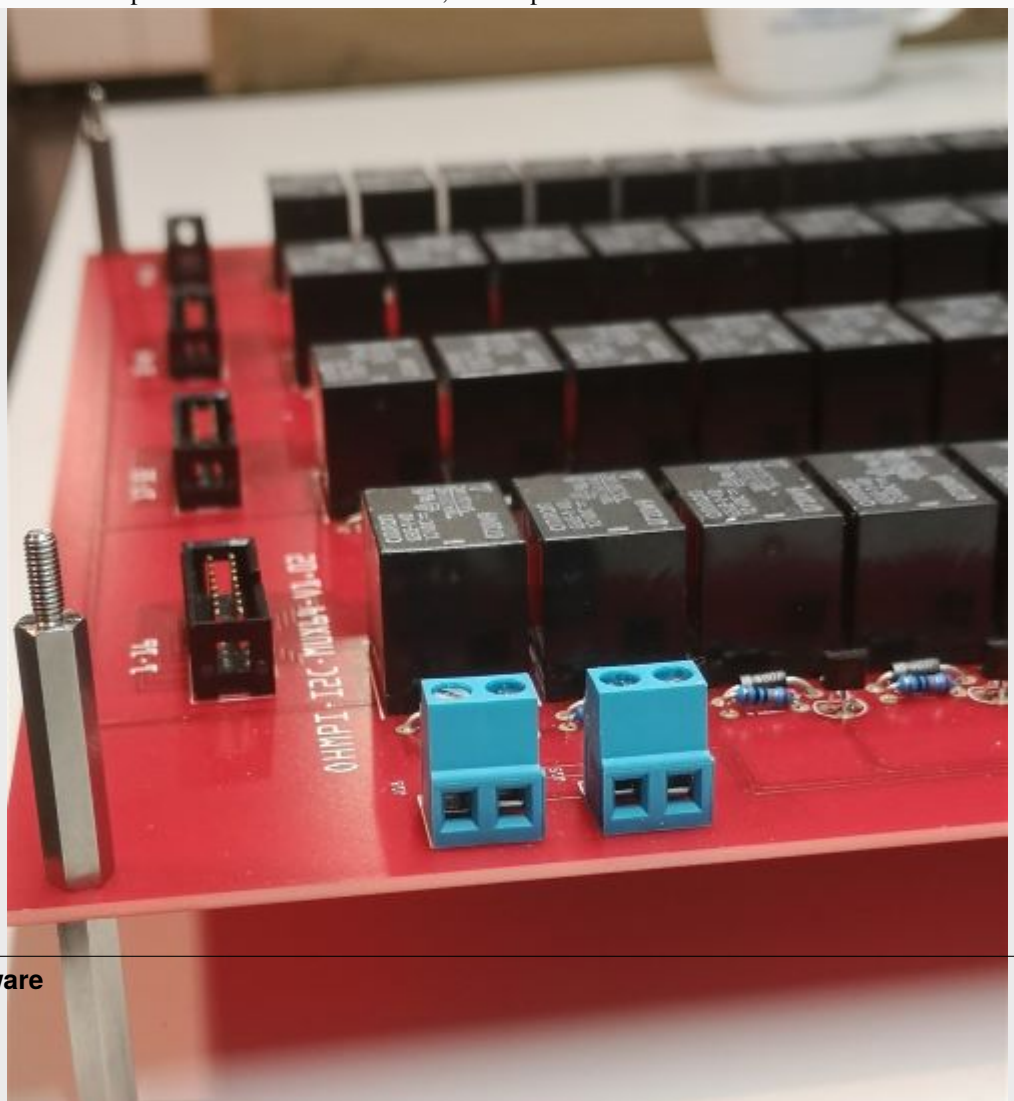




Same for a 6 wires ribbon cable of 1 m length.

5	 Cut the ribbon cable flush with the IDC connector.

6 Position 9 spacers above the MUX board, and 9 spacers below



7	 Cut a 50 cm long wire of any color (here yellow). Strip and tin each end of the wire. Install the wire “A” on the screw terminal of MUX board « A ».

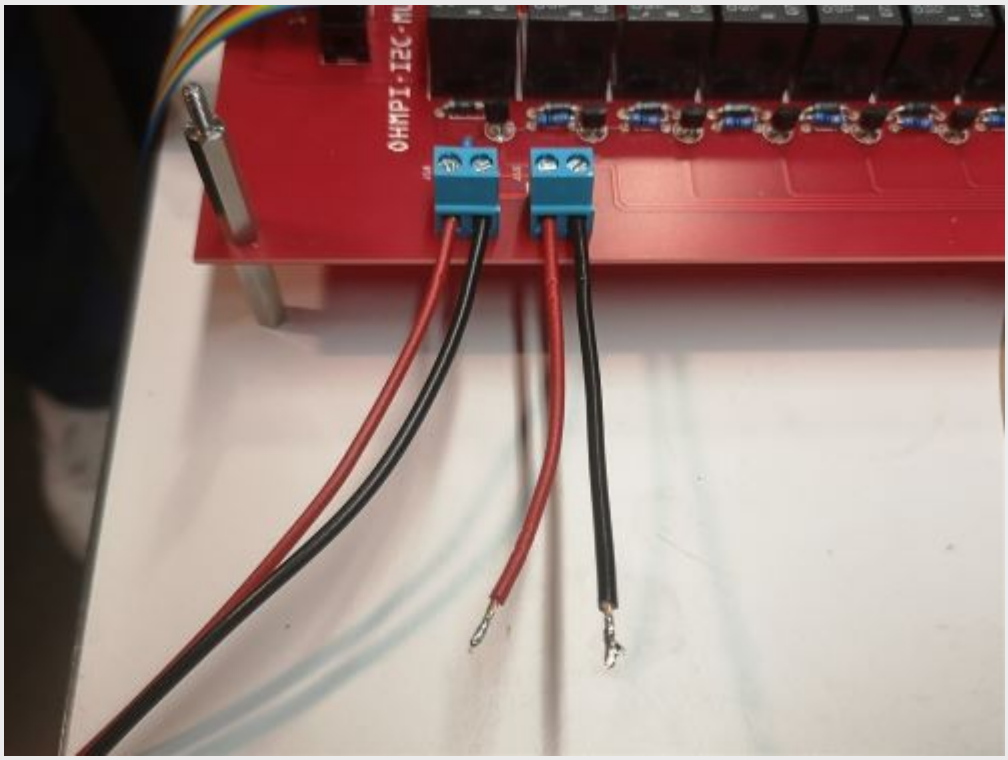
8	
	Cut a red wire and a black wire of 50 cm length. Strip, tin and position the wires on the left screw terminal as shown in the picture: i)Red wire 12 V, ii) Black wire GND

9 

Mount the 4 ribbon cables (16-wires each) with IDC connectors. A small noise is often heard when the IDC connector is clipped in place.

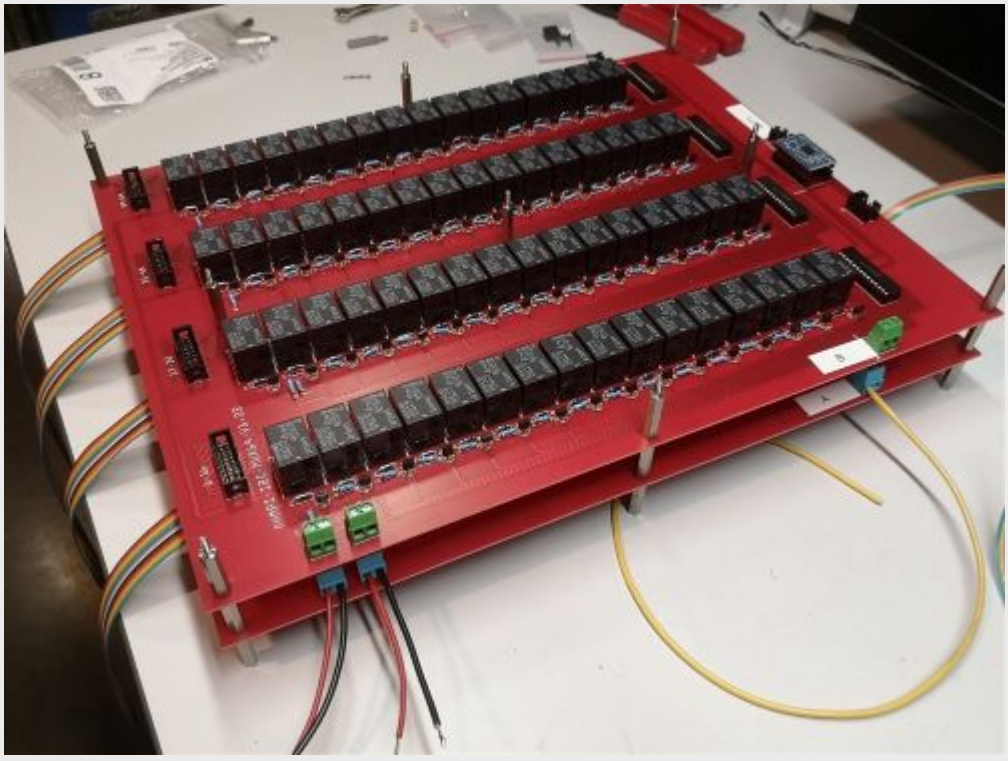
Mount the ribbon cables with 6-wires with the corresponding IDC connectors

11

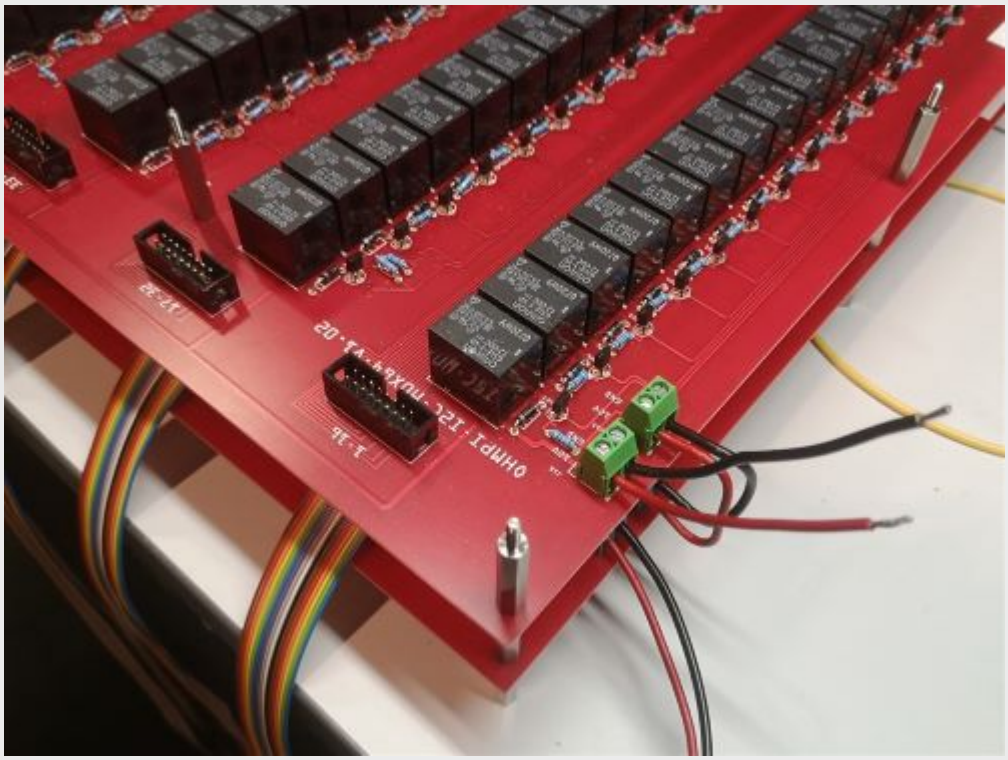


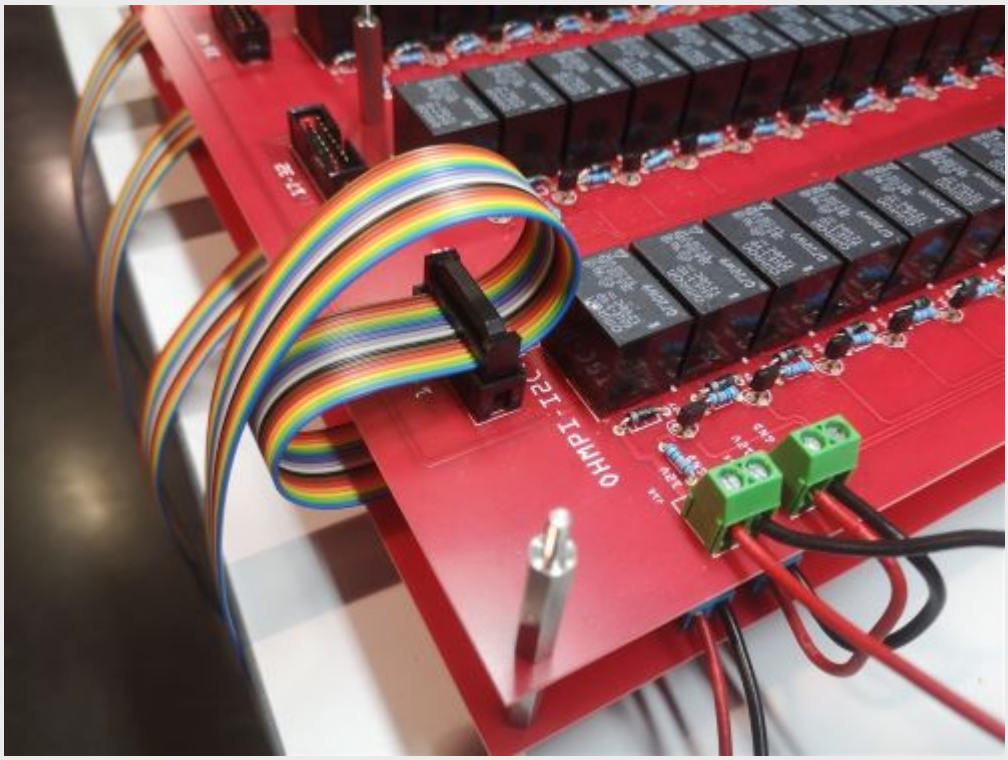
Cut a red wire and a black wire of 10 cm length. Strip and tin the wires at the ends. Mount the red wire on the 12V input and the black wire on the GND input on the right screw terminal.

12



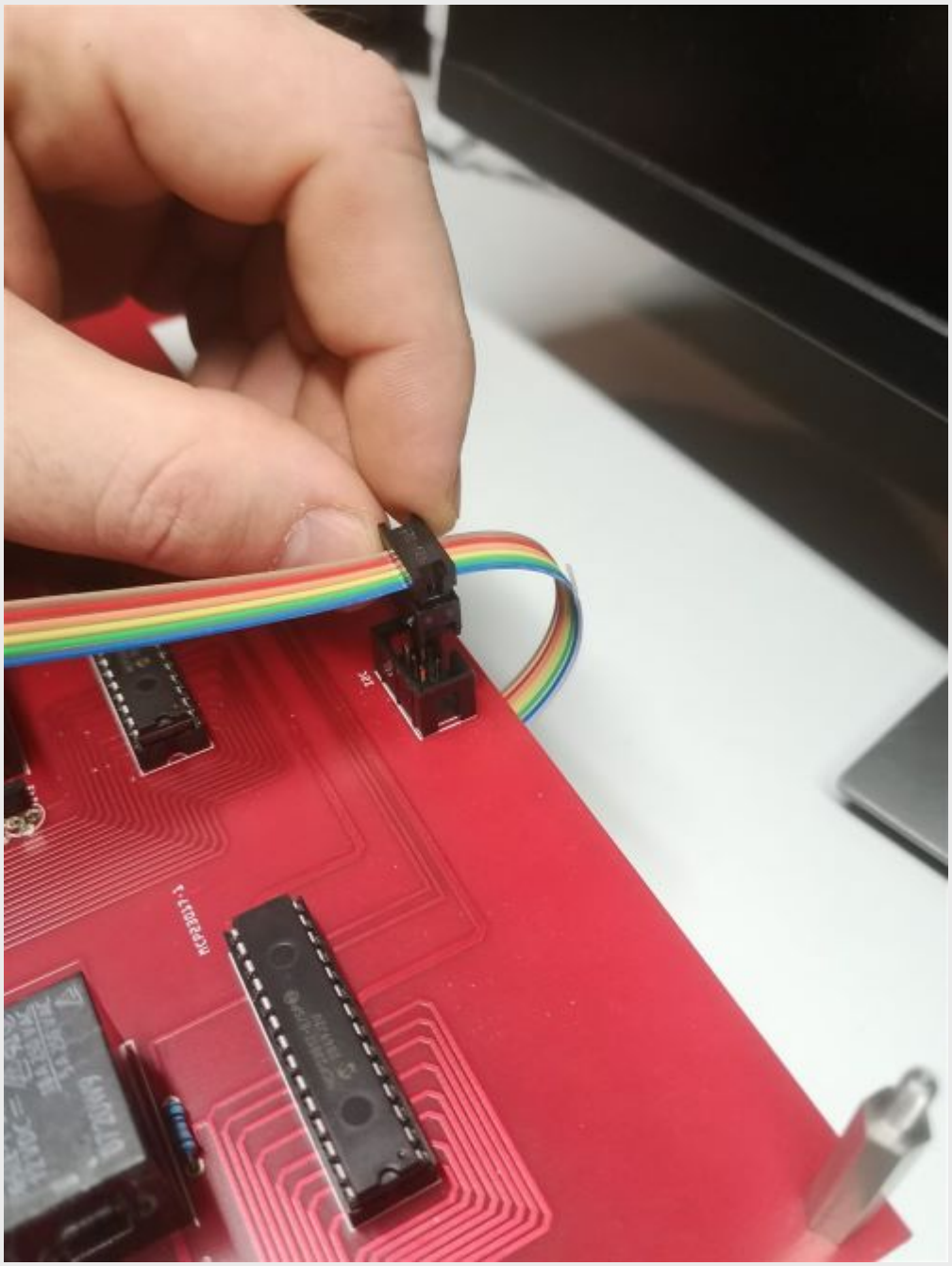
Mount and fix the second MUX board “B” on the first with the help of 9 spacers.

13	 Cut, strip and tin a red wire and a black wire of 10 cm length. Mount the wires on the left screw terminal. Red wire 12V input, black wire GND input. Connect the red and black wires from board A to the right screw terminal of board B. Red wire 12V input. Black wire GND input.

14	 Crimp a 16 wires IDC connector on the ribbon cable at about 15 cm from the previous connector. Please, pay attention to the direction of the cable before the crimp procedure. Mount the ribbon cable on the IDC connector on the board.

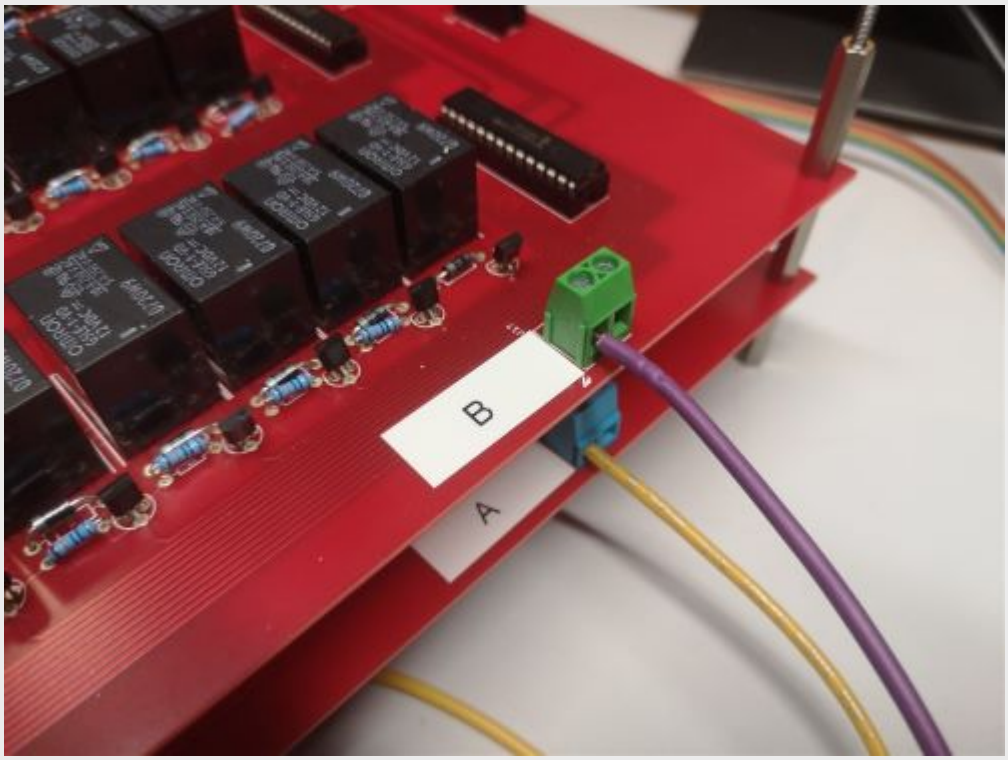
15 

Repeat the operation for the other 3 ribbon cables.

16 

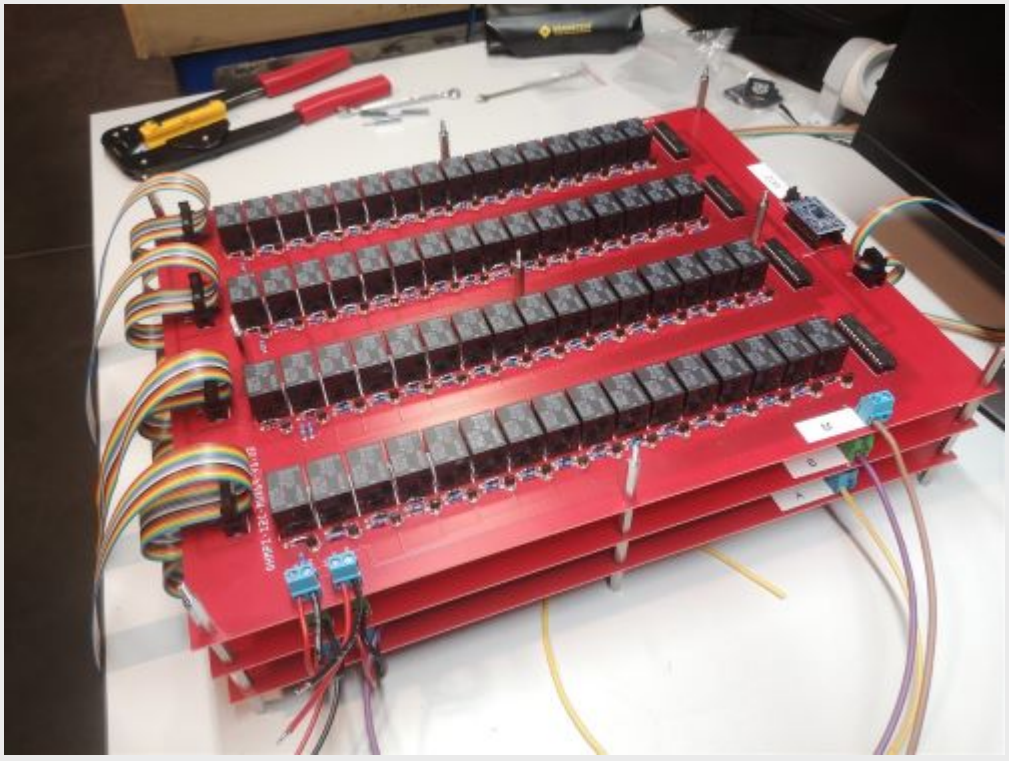
Repeat the operation for the 6 wires ribbon cable.

17

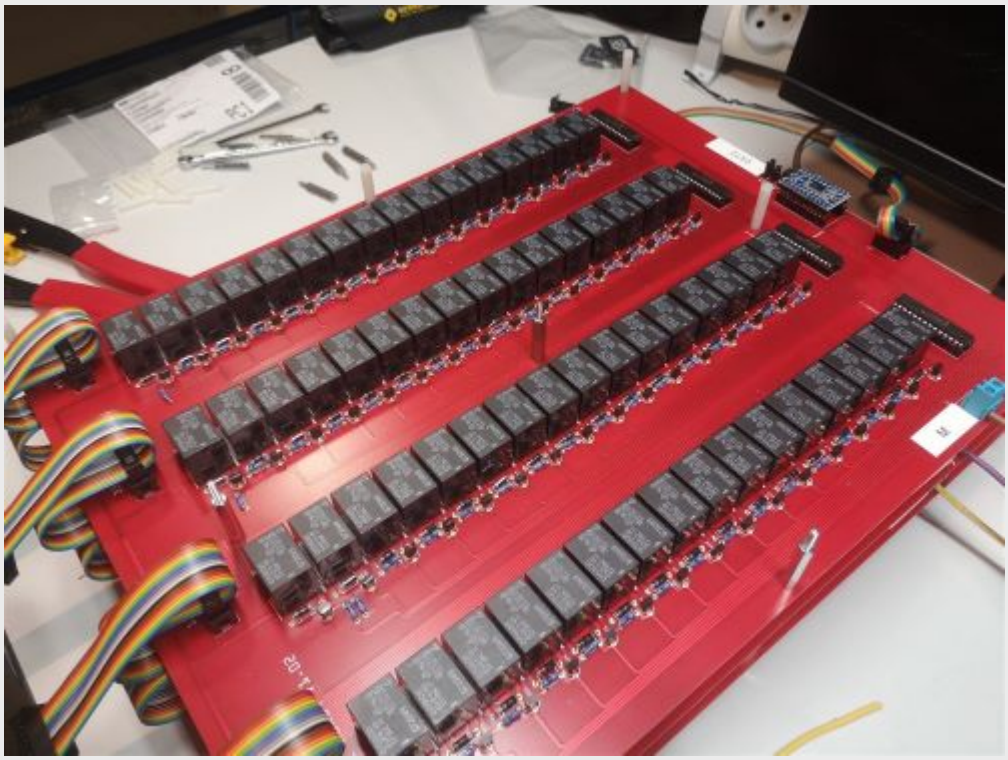


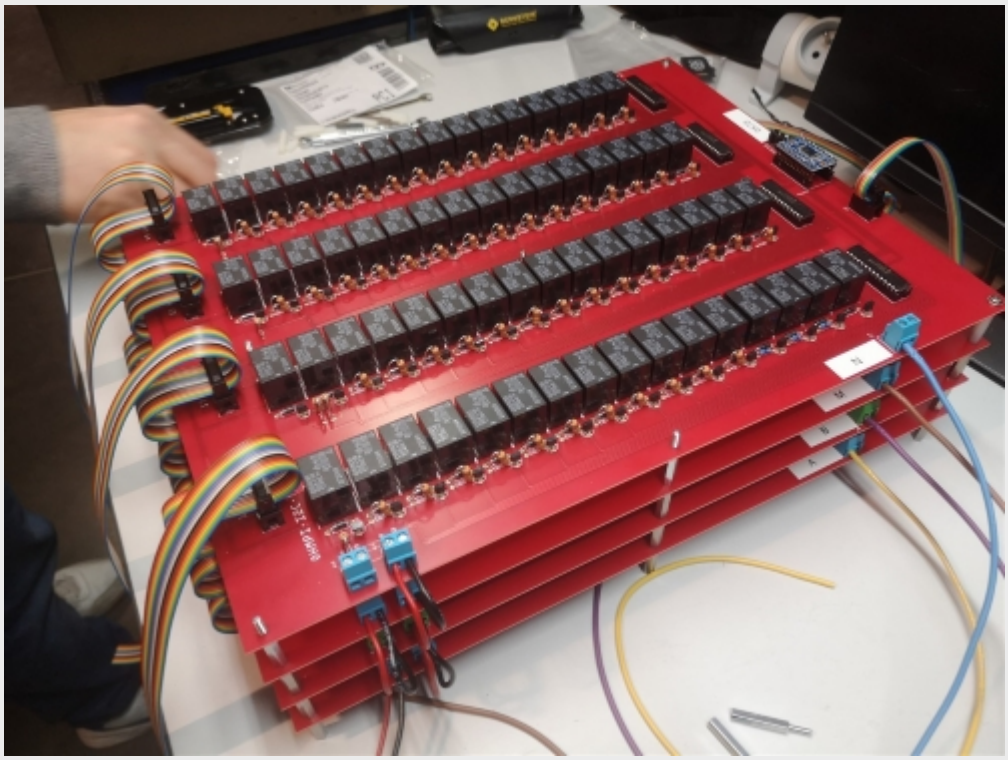
Cut a 50 cm long wire “here purple” (Color not relevant but to be defined). Strip and tin the wire at its ends. Position the wire on the input B of the screw terminal of the multiplexing board B.

18

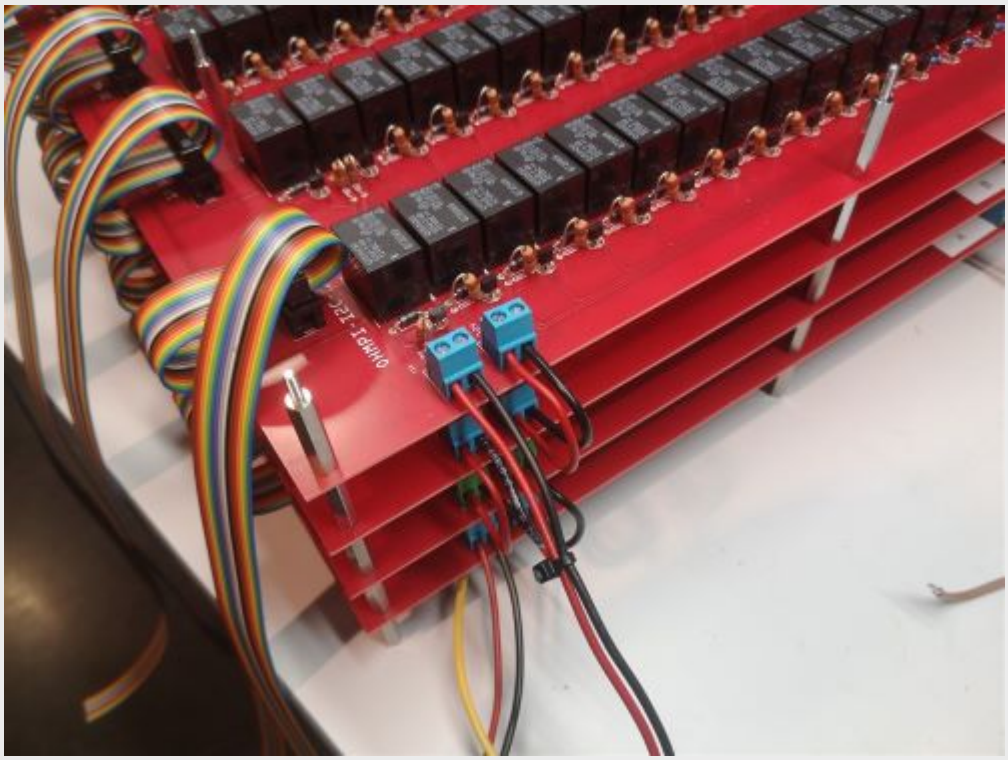


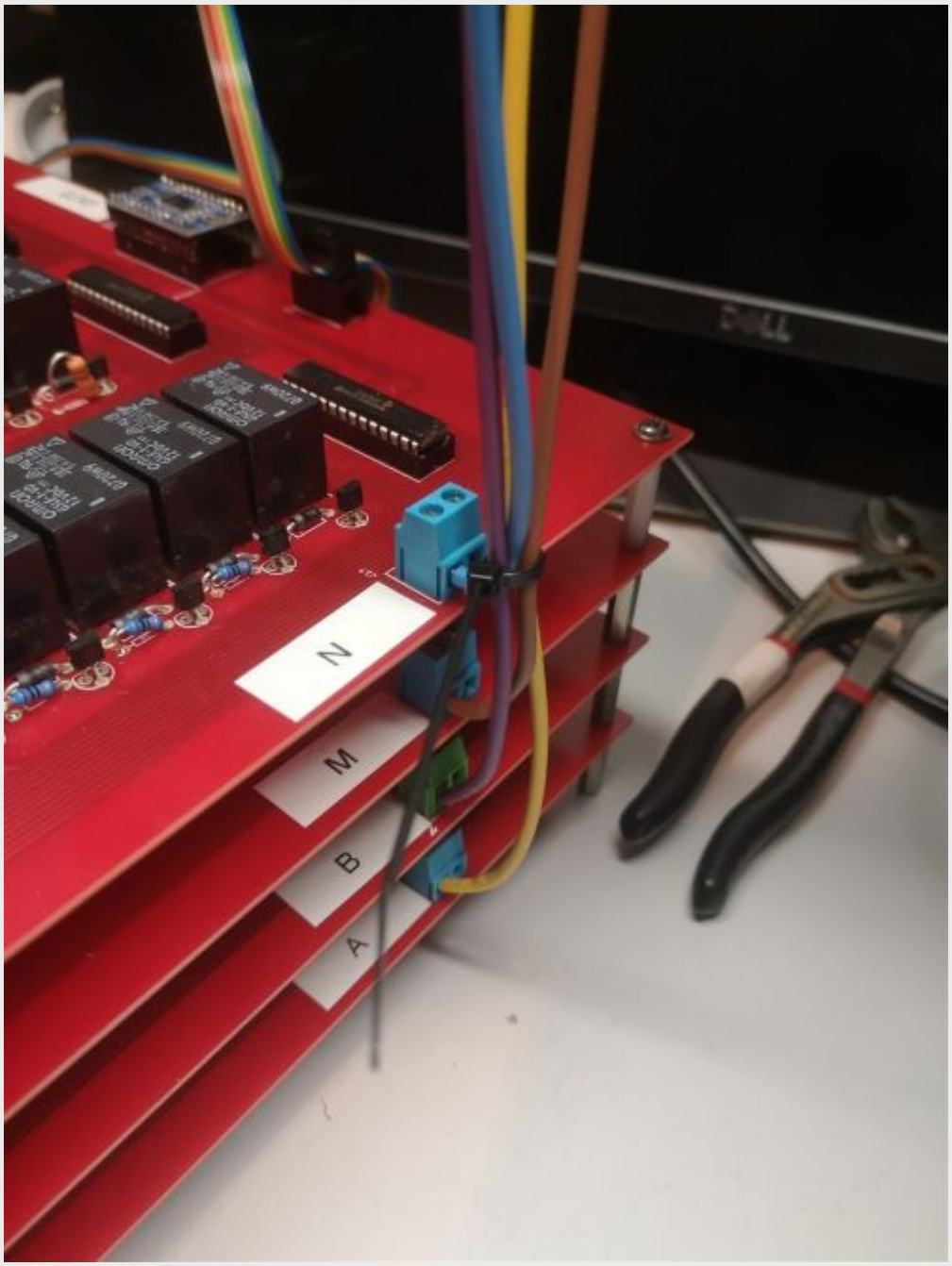
Repeat all these operations for the third MUX board called “M”.

19	
	Repeat the operations for the fourth MUX Boards. Attention, it is necessary to position 5 different spacers (here nylon screw hex spacers) in between the “M” board and the “N” MUX Board (as shown on the photograph). Refer to the following photographs for more details on the assembly of the spacers



When mounting the 4th MUX board (“N”), screws can be placed on the nylon spacers to fix the boards together. Note that the other spacers could be used for this purpose. Connect ribbon cables (16 wires) from board 3 to board 4 as previously described. Connect the red wire (12V) of MUX board “M” to the 12V terminal of the right screw terminal of MUX Board “N”. Connect the black wire (GND) of MUX board “M” to the GND screw terminal on MUX board “N”.

21	 Cut a red wire and a black wire of one meter length. Place the red wire on terminal “12V” and the black wire on terminal “GND” of the left screw terminal. Tie the wires together.



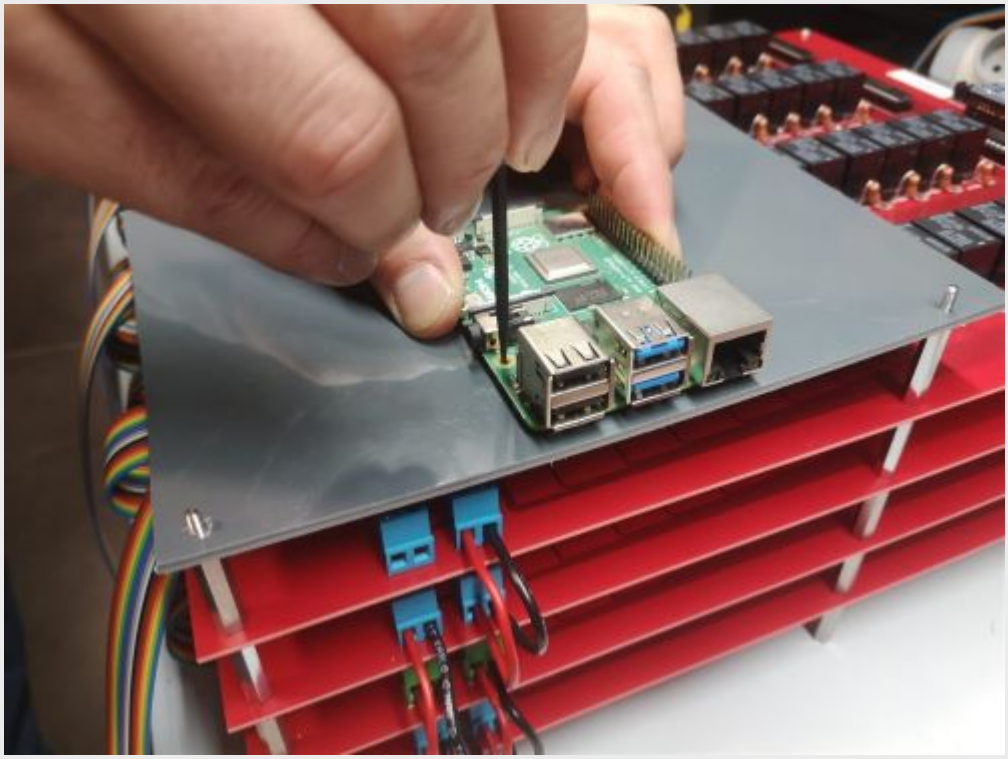
22 Tie the A, B, M and N wires together

23	 Cut a PVC plate with the following minimum dimensions : 200 mm * 150 mm * 5 mm

24

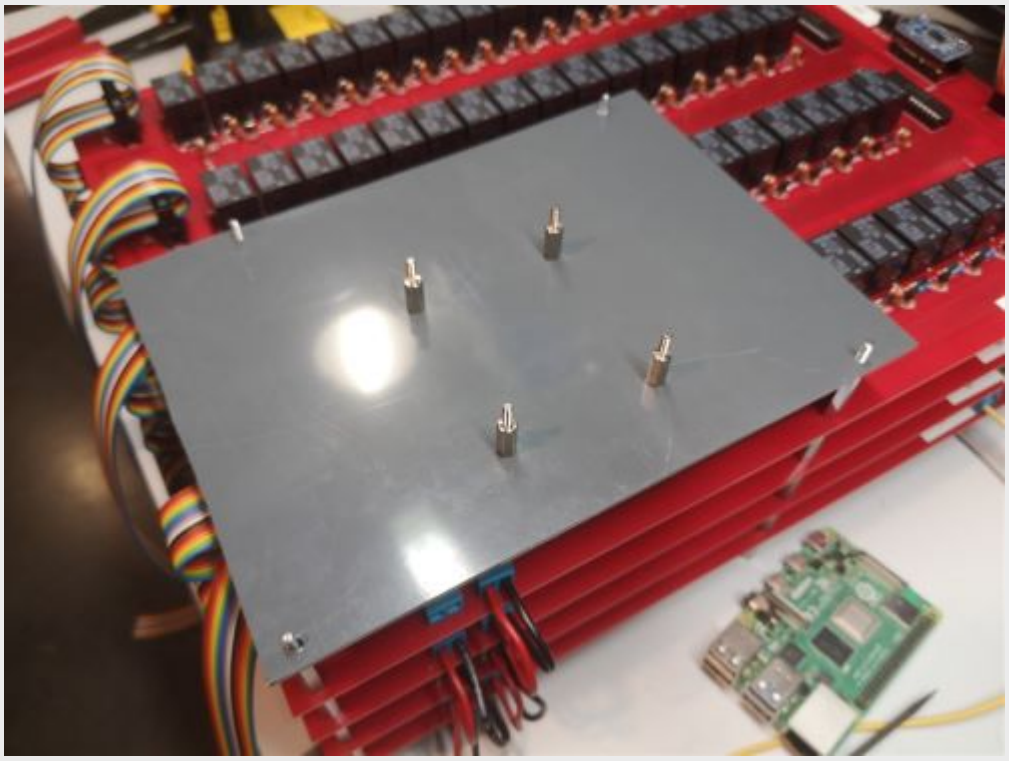
Drill the plate to mount it on the remaining metal spacers. Do not tighten the assembly.

25



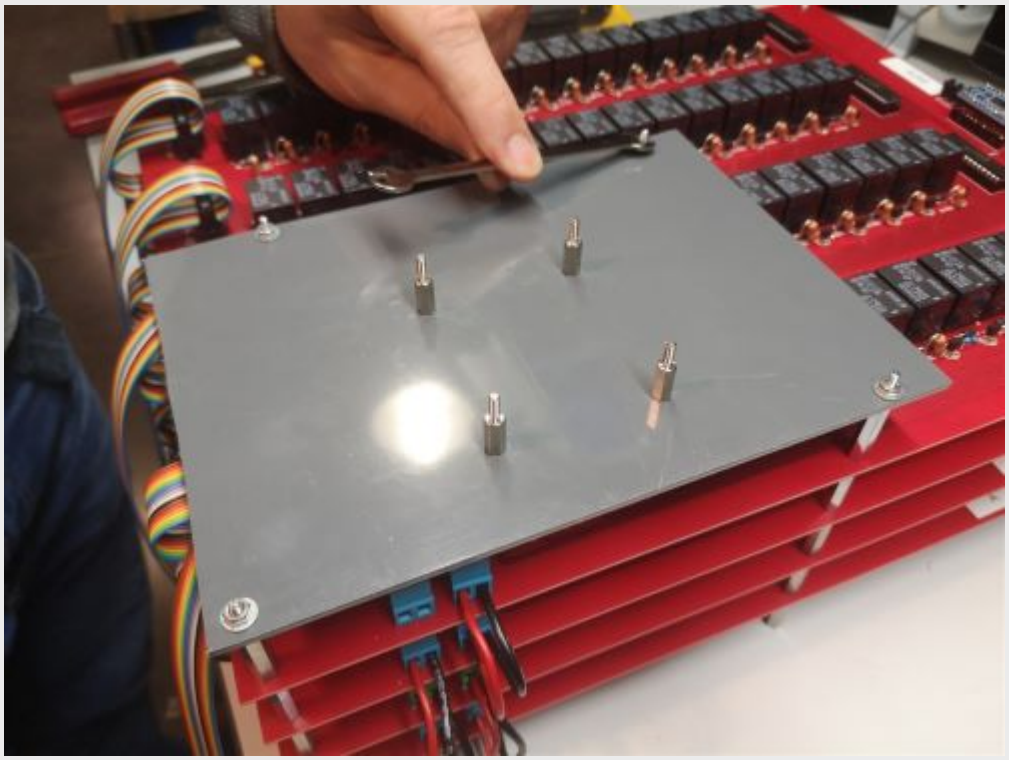
Position the Raspberry Pi (RPI) board on the plate so that you can access the USB ports. Mark the holes of the RPI board on the plate for mounting.

26



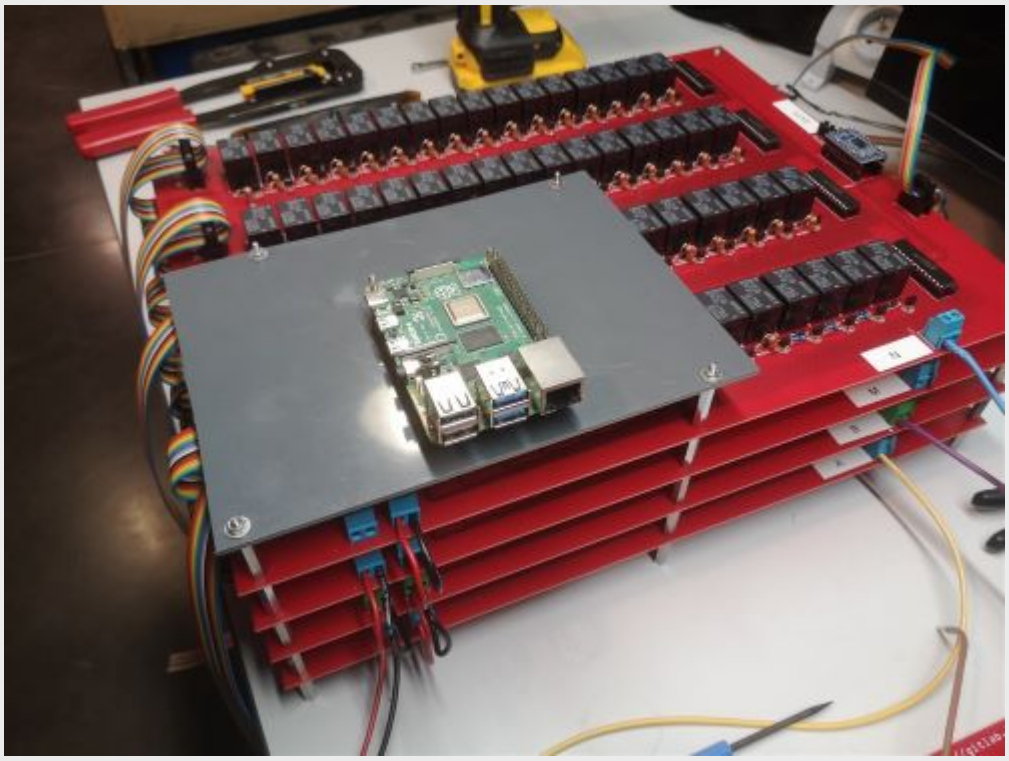
Add spacers on the PVC plate.

27



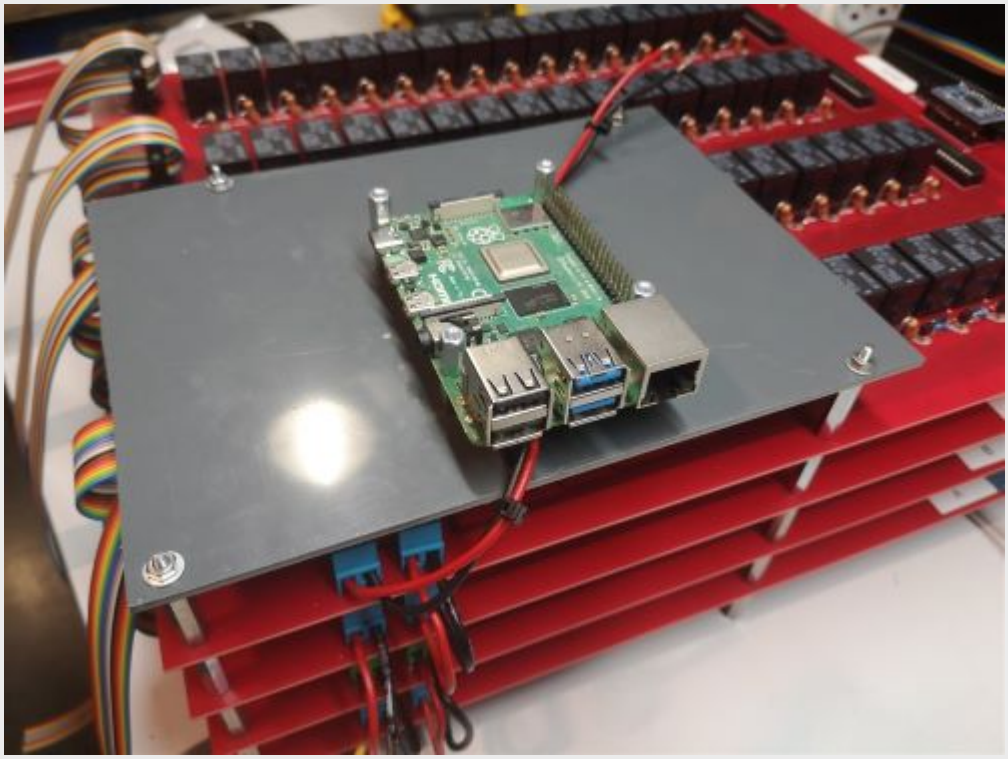
Attach the PVC plate to the metal spacers with washers and nuts.

28

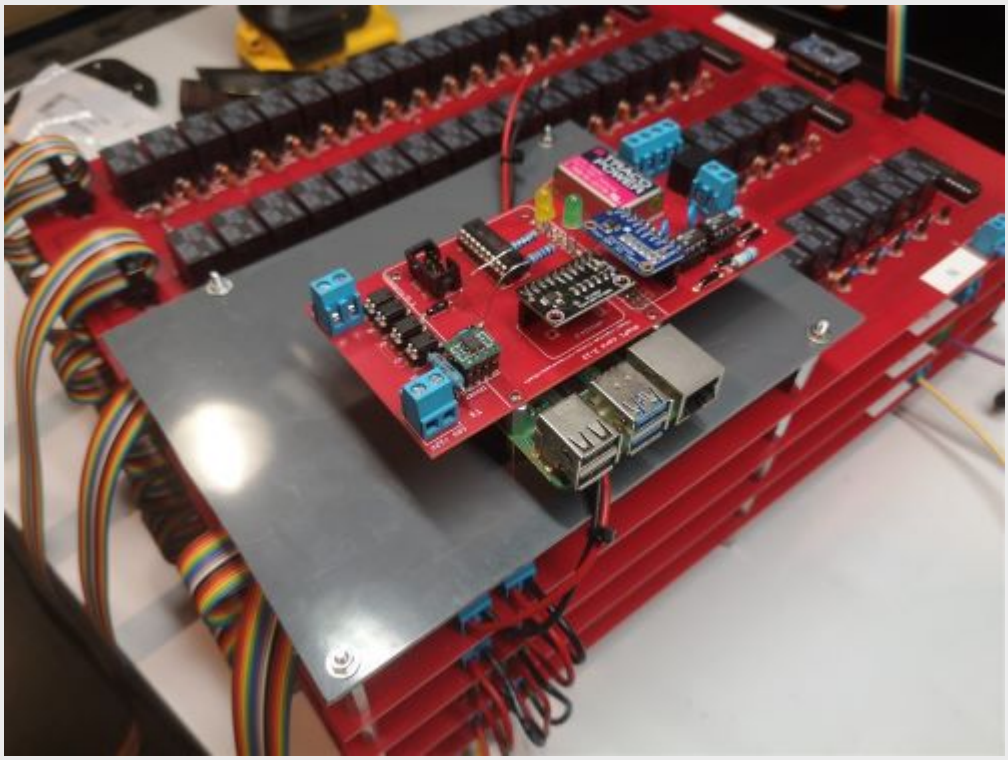


Position and fix the RPI card on the spacers

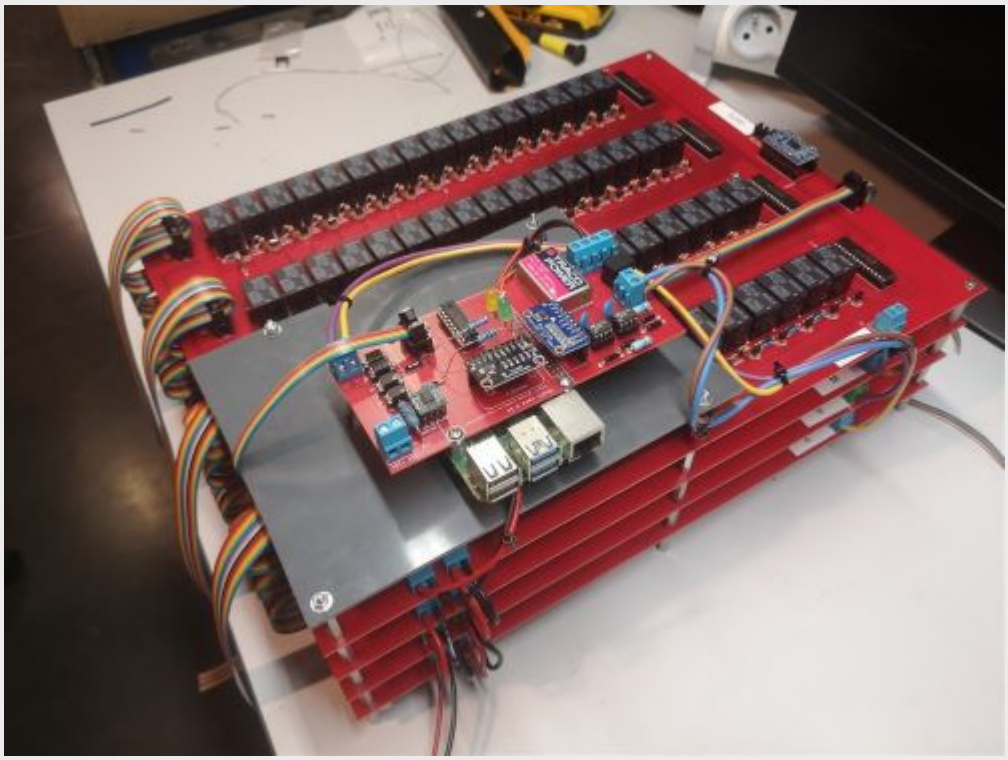
29

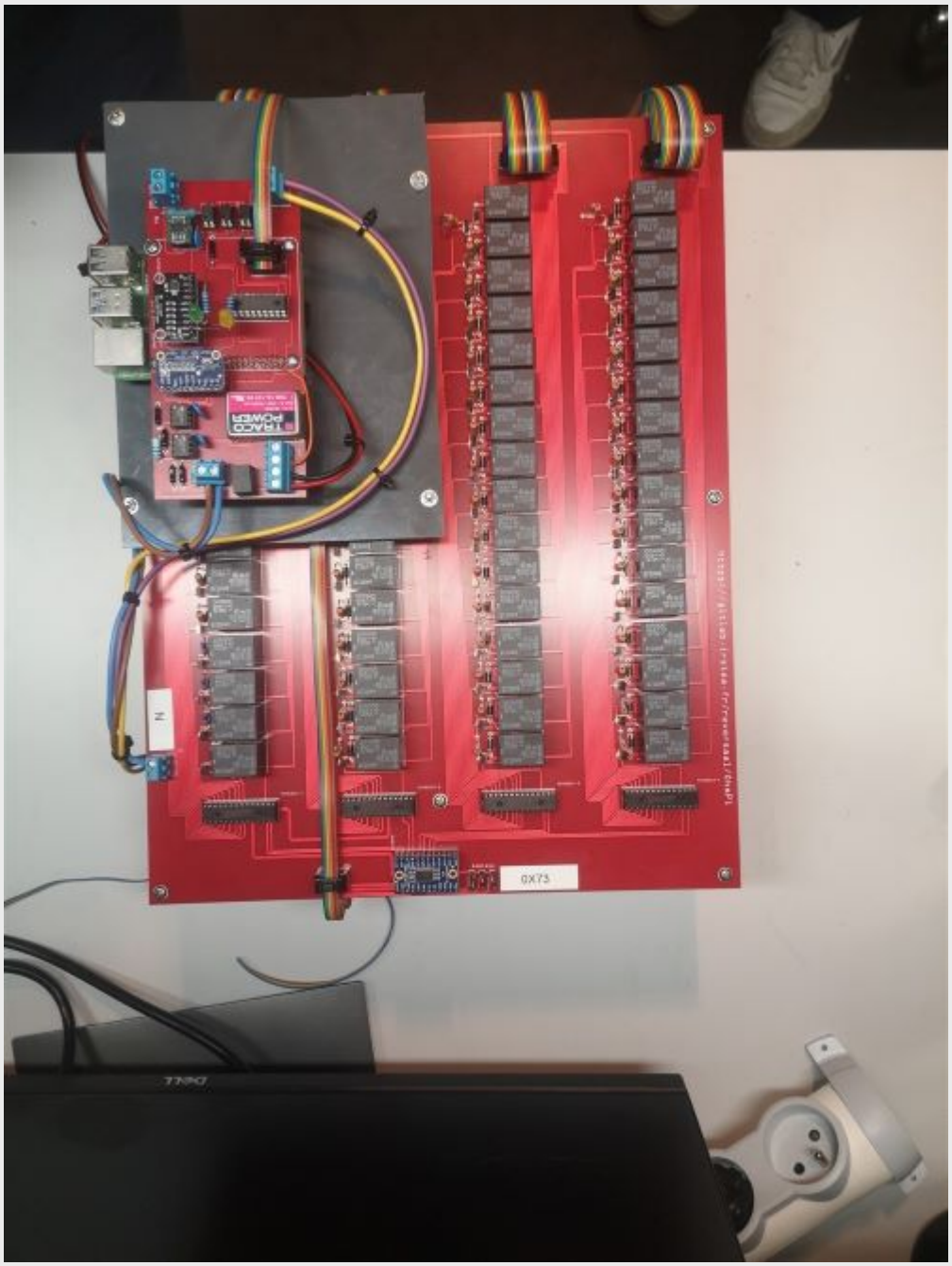


Add spacers on the RPI board. The red (12V) and black (GND) wires coming out of the “M” MUX board must pass under the RPI board.

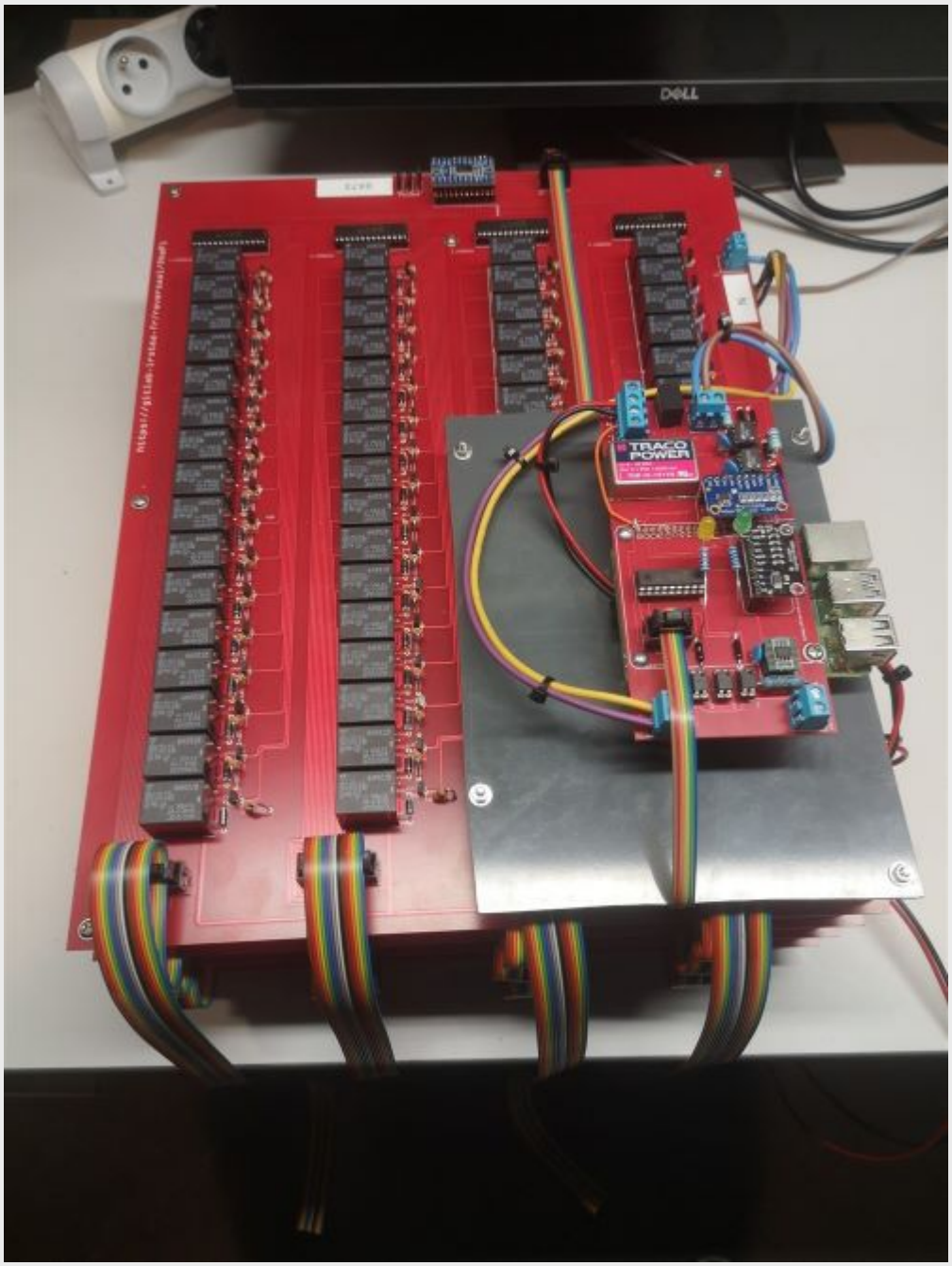
30 

Place the measurement board on the RPI GPIO outputs and on the pre-positioned spacers. Note that LEDs are present on this measurement board with an associated resistance simply for testing purposes (do not consider this temporary modification of the board). Same for the orange wire present on the board.

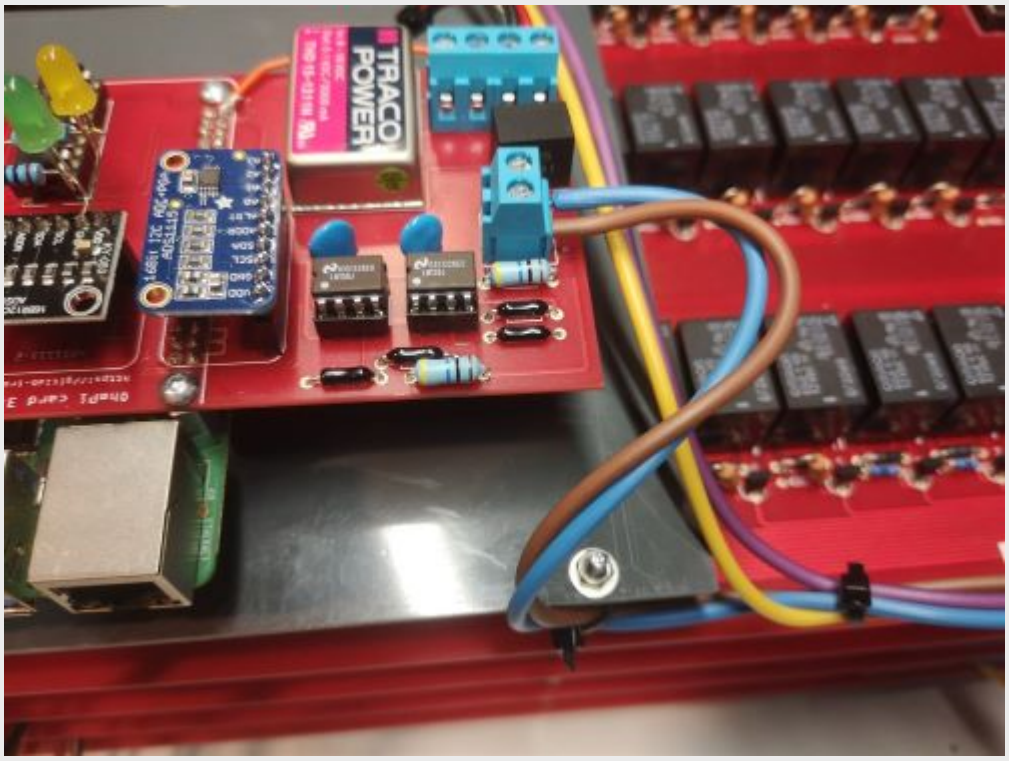
31	 Connect the wires ” A ” (here yellow), ” B ” (here purple), ” M ” (here brown) and ” N ” (here blue) on the corresponding terminal blocks on the measurement board. Connect the 6 wires ribbon cable on the measurement board by passing under the PVC plate.Connect the red and black wires to the 12 V and GND terminal block.



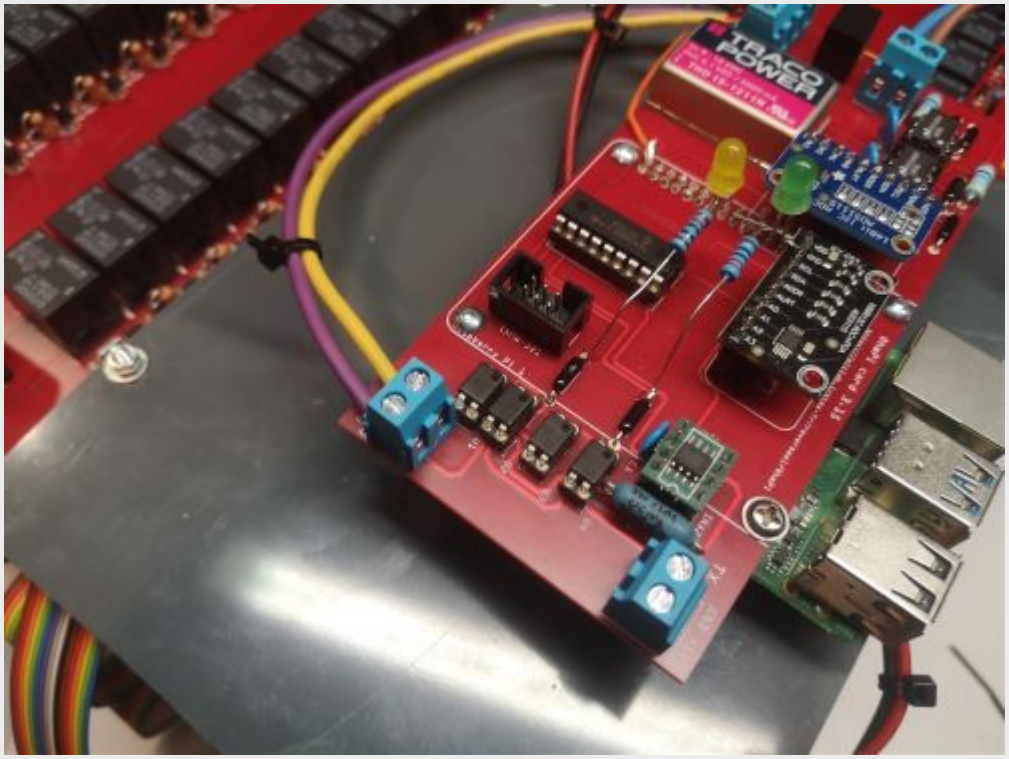
32 Top view of the assembly. Add clips to secure the wires together.



33 Second view.

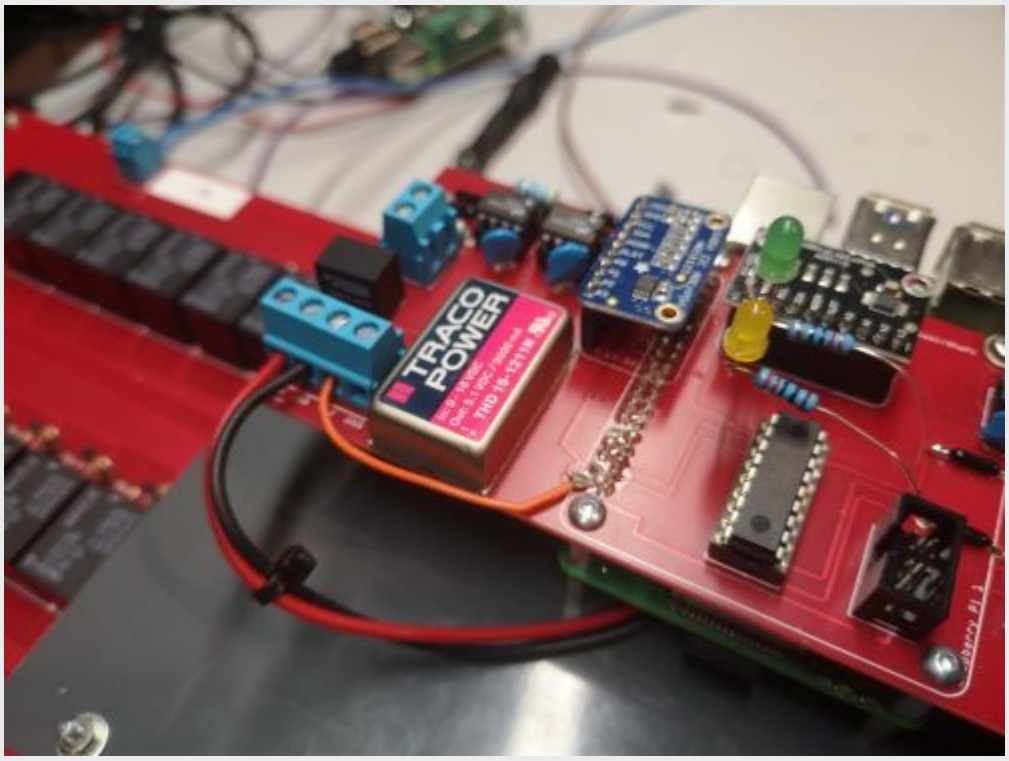


34 Zoom in on the connection of the M and N wires.

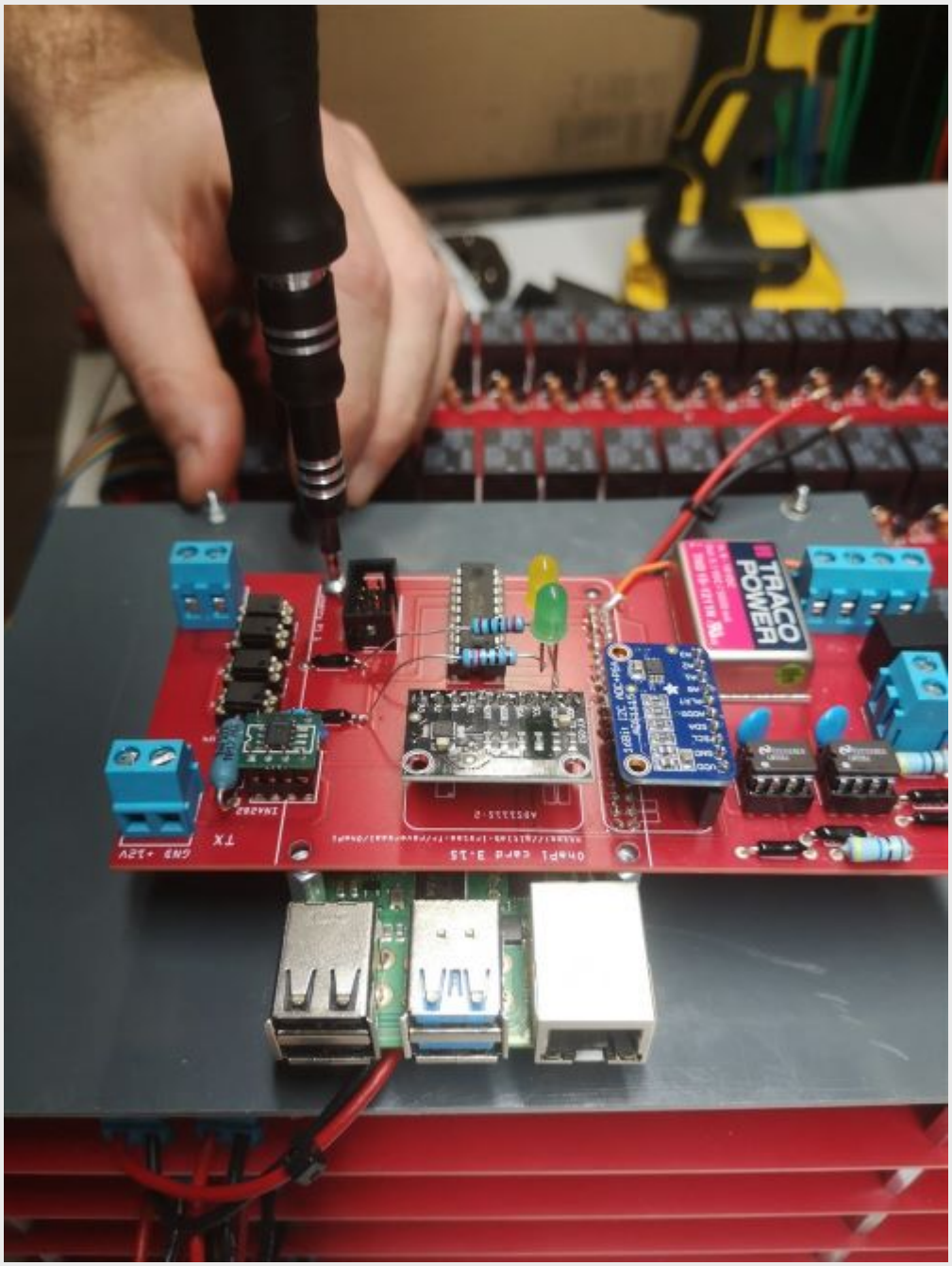


35 Zoom in on the connection of the A and B wires.

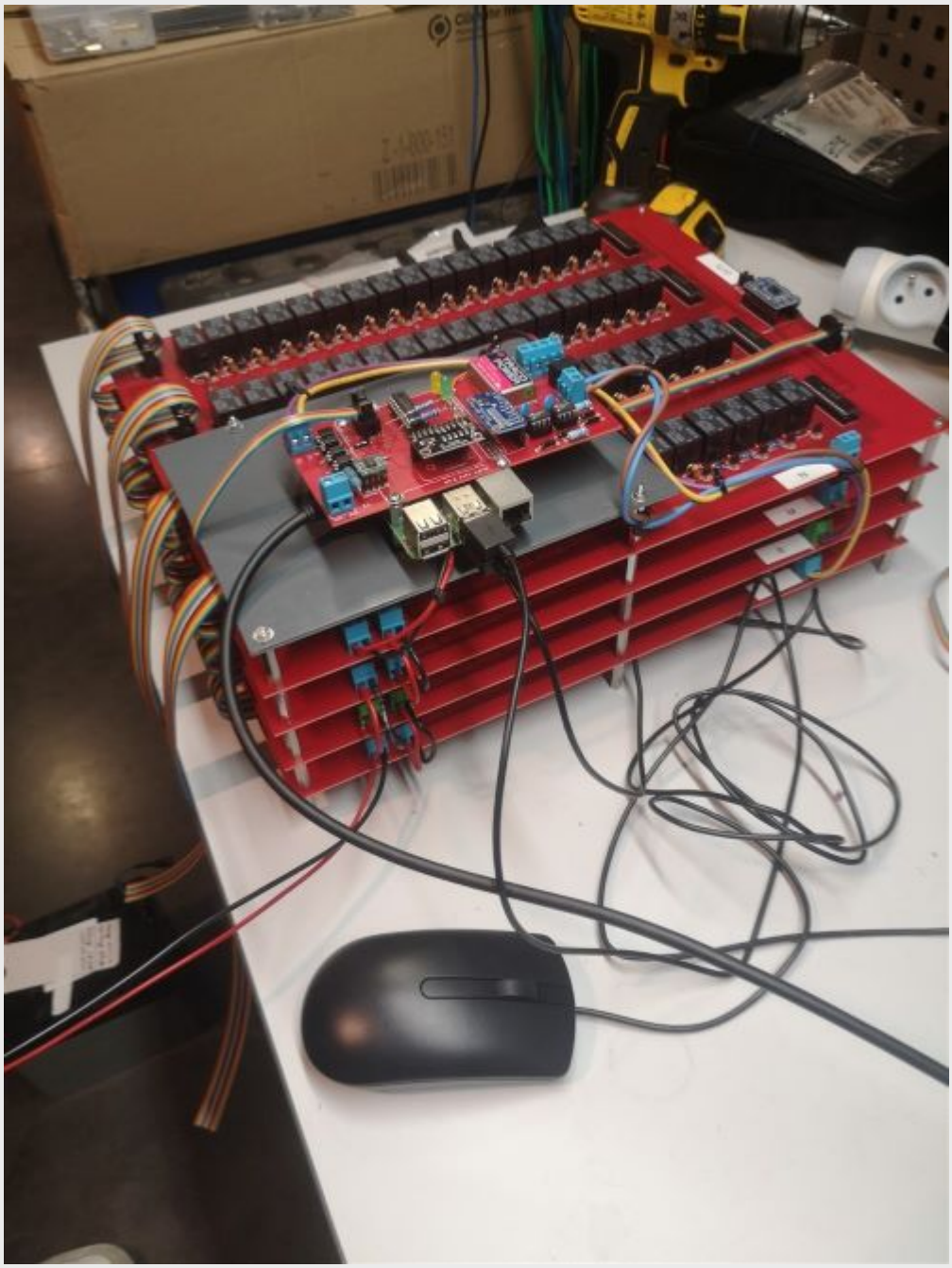
36



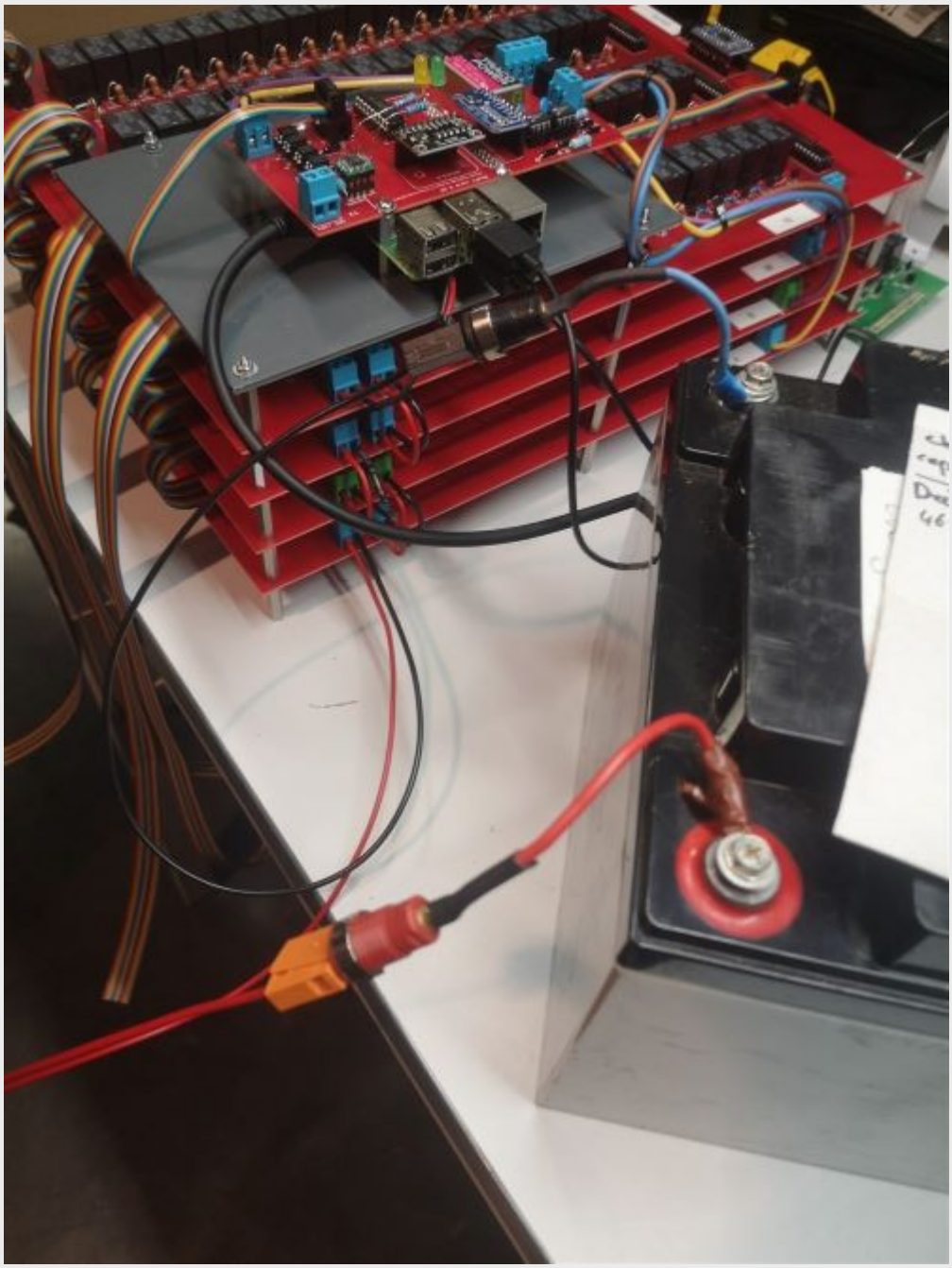
Zoom in on the connection of the « 12V » and « GND » wires.



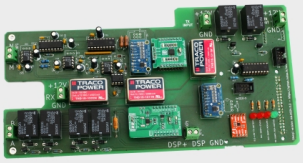
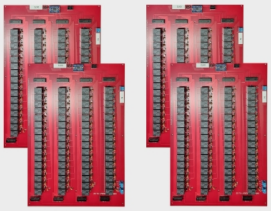
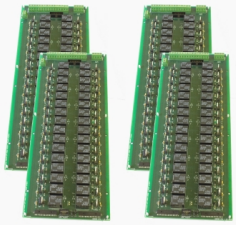
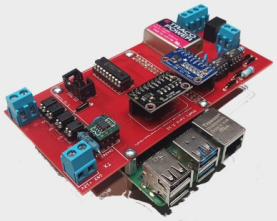
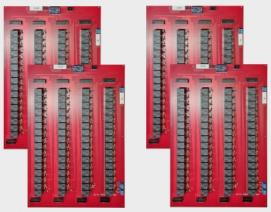
37 Fixing the measurement board on the spacers present on the RPI board.



38 Place the SD card containing the OS and the pre-installed programs. Connect a mouse and a keyboard to the USB inputs of the RPI board. Connect a monitor to the HDMI output of the RPI board.



39 Connect the red and black cables of board A to a 12V battery or other laboratory power supply delivering a 12VDC voltage. Enjoy

	1. Controller	2. Measurement board	3. MUX boards	4. Power module
OhmPi v2024	 Raspberry Pi 3B or 4	 mb.2024	 **4 mux.2023** 64 electrodes	 **Digital Power Supply** Joy-It DPS 5005 (0-50 V)
OhmPi Lite v2024	 Raspberry Pi 3B or 4	 mb.2024	 **4 (or 2) mux.2024** 32 (or 16) electrodes	 **Digital Power Supply** Joy-It DPS 5005 (0-50 V)
OhmPi v2023	 Raspberry Pi 3B	 mb.2023	 **4 mux.2023** 64 electrodes - 1 role each	 **12V battery**

1.3 Software and operation

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OhmPi. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

This section describes the Python software and how to interact with an OhmPi instrument.

1.3.1 Software architecture

The OhmPi v2024 software has been completely re-structured to enable increased flexibility for both users and developers. The software is based on an object-oriented module with a class exposing the OhmPi functionalities used to interact with the OhmPi instrument via a web interface, IoT communication protocols (e.g. MQTT) and/or directly through the Python API.

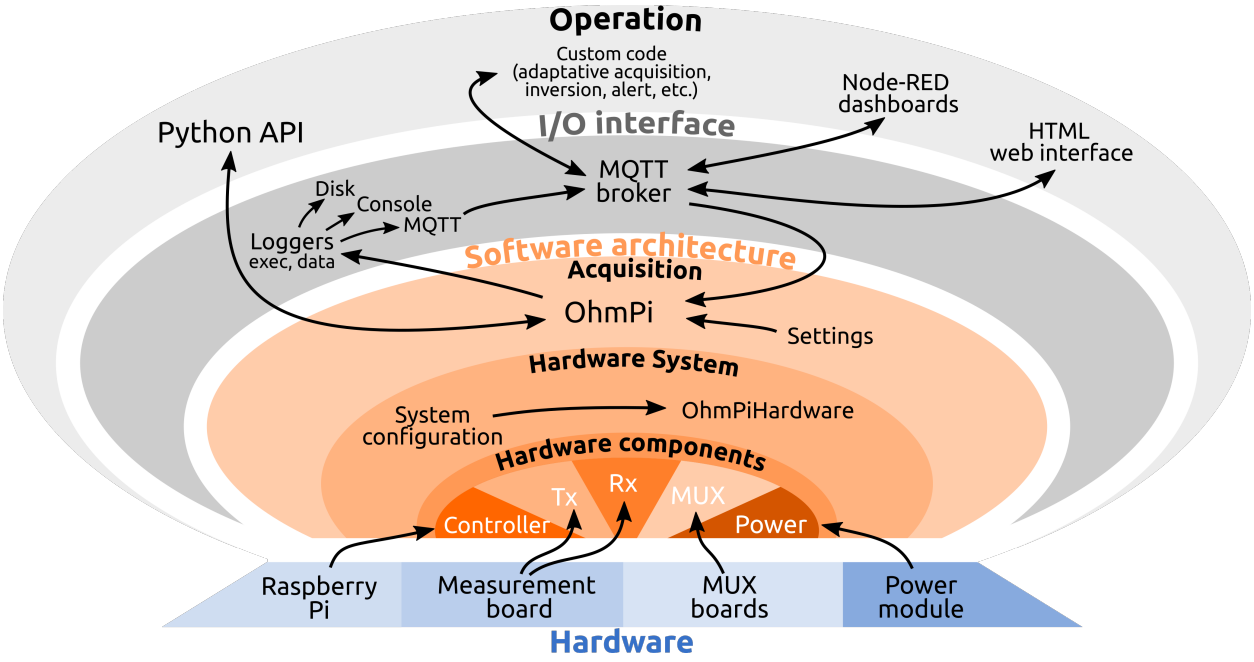


Fig. 8: Software architecture of OhmPi v2024.

The software is organised in several modules describing the system in a hierarchy including three levels of description and operation of the OhmPi.

Acquisition

On the top level, the `OhmPi` class (in ohmpi/ohmpi.py) includes all the higher-level methods and properties allowing to operate the system (e.g. acquire measurement sequences). The OhmPi class exposes the user-oriented API, generates logs and handles IoT messages. Generic users are expected to interact with the system through these higher-level functionalities, which are designed to remain as stable as possible while the hardware evolves. Only the introduction of new end-user functionalities should imply new developments at this level.

Hardware system

On the medium level, the `OhmPiHardware` class provides a mean to assemble and operate the acquisition system. The methods of this class orchestrate atomic operations of the system components in order to expose basic system functionalities such as cross-MUX switching, square wave voltage injection or full waveform voltage and current reading during injection cycles. These functionalities are implemented using synchronization mechanisms between threads in order to insure that each component keeps in step with the others. The whole system is described in a *configuration file* listing the hardware components and versions used. Through a dynamic import mechanism the modules containing the classes corresponding with the physical hardware modules of a particular OhmPi system are instantiated and associated with the system object instantiated from the OhmPiHardware class. In this way, it is relatively simple to build customised systems once the concrete classes describing the system components have been written. This part of the

software architecture should remain stable if the overall system functionalities do not evolve. However, the introduction of new functionalities at the system level or radical changes in the way the components work together will require adaptations at this level.

Hardware components

On the base level, the five main hardware components are represented by distinct classes exposing the components atomic functionalities. Theses classes are abstract classes in order to provide a common interface for different implementations of a component. From these abstract classes concrete classes are implemented representing the default properties, actual capabilities and ways to interact with the physical modules or boards. Improving an existing hardware component or introducing a new design may be desirable in order to, e.g. reduce costs, improve performance, adapt measurement range to specific applications, or incorporate easily available electronic components. It is at this level that software developments are mainly expected to occur following updates on the hardware. The component class should expose the minimal functionalities required by the hardware system for this type of component.

Warning: OhmPi is a participative project open to all, it requires skills in electronics and to respect the safety rules. OhmPi must be assembled in a professional context and by people competent in electronics. The OhmPi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The OhmPi team cannot be held responsible if the equipment does not work after assembly.

1.3.2 Getting started

Step 1: Set up the Raspberry Pi

First, install an operating system on the Raspberry Pi by following the official instructions

Then connect to the Raspberry Pi either via ssh or using an external monitor.

For all questions related to Raspberry Pi operations, please refer to the official documentation

In the "Raspberry Pi Configuration" (graphically: start button > Preferences > Raspberry Pi Configuration; in command line: *raspi-config*), in the 'Interfaces' tab, make sure **I2C is enabled**. That will allow the Pi to communicate with the OhmPi measurement board.

Step 2: Clone the OhmPi project

You need to clone the OhmPi repository on the Raspberry Pi with the following command:

```
git clone https://gitlab.com/ohmpi/ohmpi.git
```

Note that the project moved from the Gitlab IRSTEA to gitlab.com in January 2024. The Gitlab IRSTEA is synced as a read-only clone from the gitlab.com.

Step 3: Run the installation script

Simply navigate to the OhmPi folder:

```
cd ohmpi
```

And run the following command on the terminal:

```
./install.sh
```

The install script:

- creates an **python virtual environment** called "ohmpy" in which all dependencies will be installed;
- installs all **dependencies** specified in requirements.txt;
- installs a **local MQTT broker** which will be used to centralize all the communication between the hardware, the software and the interfaces;
- configures the **I2C buses** on the Raspberry Pi.

When the installation is performed, we need to add the OhmPi folder to the PYTHONPATH by editing the .bashrc file as follows:

```
nano ~/.bashrc
```

And add the following line:

```
export PYTHONPATH=$PYTHONPATH:/home/<username>/OhmPi
```

Replace <*username*> by your username on the Raspberry Pi (e.g.: /home/pi/OhmPi).

Step 4: Activate the *ohmpy* virtual environment

Before operating the instrument, we need to activate the *ohmpy* virtual environment with the following command:

```
cd ~/OhmPi
source ohmpy/bin/activate
```

If you need to leave the virtual environment, simply type:

```
deactivate
```

Following these steps, you are now ready to operate the OhmPi.

1.3.3 Operating the system

This section details describes how to operate an OhmPi instrument.

Contents:

Configuration

The configuration file *config.py* defines how the OhmPi system is assembled and expected to behave. It tells the software how the hardware is set up, cabled and configured. In certain cases, it also allows you to define hardware specifications, such as the maximum voltage that a specific MUX board can handle. For general purpose, most specifications can be left on default values.

Warning: Not to be confused with *Acquisition settings*. One configuration specified in a config.py file can handle multiple combinations of acquisition settings.

Default configuration

A default version of *config.py* is provided in the repository. This file should be edited to customize the configuration following the user's needs and preferences. A series of default configuration files are available in the configs folder. A simple helper command can help you select the appropriate configuration file depending on your version of the meausurement board and type of MUX boards. The helper will ask you a few questions and will select the right configuration for your case. It can be called in via the terminal as

```
python setup_config.py
```

Still, it is best practice to open the configuration file and check that the parameters are correctly configured. Updating the configuration file manually is mandatory for custom systems combining different versions of the measurement and MUX boards.

Warning: One should make sure to understand the parameters before altering them. It is also recommended to keep a copy of the default configuration.

Configuration file structure

Listing 1: Config file header

```
import logging
from ohmpi.utils import get_platform

from paho.mqtt.client import MQTTv31  # noqa

_, on_pi = get_platform()
# DEFINE THE ID OF YOUR OhmPi
ohmpi_id = '0001' if on_pi else 'XXXX'
# DEFINE YOUR MQTT BROKER (DEFAULT: 'localhost')
mqtt_broker = 'localhost' if on_pi else 'NAME_YOUR_BROKER_WHEN_IN_SIMULATION_MODE_HERE'
# DEFINE THE SUFFIX TO ADD TO YOUR LOGS FILES
logging_suffix = ''
```

The configuration is written in a python file structured in a series of dictionaries related to:

1. OHMPI_CONFIG: the OhmPi instrument information (id of the instrument and default settings).

Listing 2: OHMPI_CONFIG: Dictionary containing basic informations about the OhmPi instrument

```
# OhmPi configuration
OHMPI_CONFIG = {
    'id': ohmpi_id,  # Unique identifier of the OhmPi board (string), default = '0001'
    'settings': 'settings/default.json',  # INSERT YOUR FAVORITE SETTINGS FILE HERE
}
```

1. HARDWARE_CONFIG: the hardware system in which the five different modules 'ctl' (controller), 'tx' (transmitter), 'rx' (receiver), 'mux' (multiplexers), 'pwr' (power).

Table 10: HARDWARE_CONFIG

Main Key	Module Key	Value Description	Expected Value	Value description
ctl	model	Controller of the OhmPi system.	raspberry_pi	Defines a Raspberry Pi as controller.
pwr	model	Type of power unit.	pwr_batt	Defines an external battery as power unit.
			pwr_dph5005	Defines an external DPH5005 as power unit
	voltage	Defines default output voltage in V.	*float*, e.g. 12.	Sets 12 V as default voltage.
	interface_name	Interface used for communication with controller.	none	Sets no software communication (e.g. for 'pwr_batt')
			modbus	Sets a modubs connection
tx	model	Type of transmitter.	mb_2024_0_2	Load TX defined in *ohmpi.hardware_components.mb_2024_0_2()*
			mb_2023_0_X	Load TX defined in *ohmpi.hardware_components.mb_2023_0_X()*
	voltage_max	Maximum voltage supported by the TX board [V]	*float*, e.g. 50.	
	current_max	Maximum current supported by TX board [A]	*float*, e.g. 0.05	Is function of r_shunt. Can be calculated as 4.80/(50*r_shunt)
	r_shunt	Value (in Ohms) of shunt resistor mounted on TX.	*float*, e.g. 2.	2 Ohms resistor.
	interface_name	Name of interface used for communication with controller	i2c	I2C connector 1
			i2c_ext	I2C connector 2
rx	model	Type of transmitter.	mb_2024_0_2	Load RX defined in *ohmpi.hardware_components.mb_2024_0_2()*
			mb_2023_0_X	Load RX defined in *ohmpi.*

Table 11: MUX board general config in HARDWARE_CONFIG

Module Key	Value Description	Expected Value	Value description
model	Type of Mux board.	mux_2024_0_X	Load RX defined in *ohmpi.hardware_components.mux_2024_0_X()*
		mux_2023_0_X	Load RX defined in *ohmpi.hardware_components.mux_2023_0_X()*
electrodes	List of electrodes addressed by the MUX board	*array-like*, e.g. range(1,65)	Sets electrode IDs addressed by the MUX board
roles	Roles addressed by the MUX board	* *string*: 'A', 'B', 'M', 'N' * or *list*, e.g. ['A, 'B'] * or *dict*, e.g. {'A':'X','B':'Y', 'M':'XX','N':'YY'}	Sets roles addressed by the MUX board. If *string*, MUX addresses only 1 role (for MUX 2023) For MUX 2024: * Number of roles defines if MUX set up in 2 or 4 roles mode. * *list* or *array* order determines physical cabling * *dict* values rely on annotation on MUX 2024 board 'X', 'Y', 'XX', 'YY'
voltage_max	Maximum injected voltage managed by the MUX board	*float*, e.g. 50.	Sets maximum voltage to 50 V.
current_max	Maximum current [in A] managed by the MUX board	*float*, e.g. 3.	Sets maximum current to 3 A.

Table 12: MUX 2023 board specific config in HARDWARE_CONFIG

Module Key Value Description	Expected Value	Value description
mux_tca_address I2C address of MUX board	*hex integer* 0x70 - 0x77	Address of MUX board

Table 13: MUX 2024 board specific config in HARDWARE_CONFIG

Module Key Value Description	Expected Value	Value description
addr1 Physical position of jumper on addr1	*string* 'up' or 'down	This will compute I2C address of MUX board based on addr1 and addr 2 configuration. See *MUX board addresses*.
addr2 Physical position of jumper on addr1	*string* 'up' or 'down	This will compute I2C address of MUX board based on addr1 and addr 2 configuration. See *MUX board addresses*.
tca_address I2C address of I2C extension	None *(default)* *hex integer*, e.g. 0x71	No I2C extensions cabled. Address of I2C extension
tca_channel Channel of the I2C extension	*int* 0 - 7	Channel used in case I2C extension configured.

Here's an example of the HARDWARE_CONFIG:

Listing 3: HARDWARE_CONFIG: Dictionary containing configuration of the hardware system and how it is assembled.

```
r_shunt = 2. # Value of the shunt resistor in Ohm.
HARDWARE_CONFIG = {
    'ctl': {'model': 'raspberry_pi'}, # contains informations related to controller unit,
↪ 'raspberry_pi' only implemented so far
    'pwr': {'model': 'pwr_batt', 'voltage': 12., 'interface_name': 'none'},
    'tx':  {'model': 'mb_2024_0_2',
            'voltage_max': 50.,  # Maximum voltage supported by the TX board [V]
            'current_max': 4.80/(50*r_shunt),  # Maximum voltage read by the current␣
↪ADC on the TX board [A]
            'r_shunt': r_shunt,  # Shunt resistance in Ohms
```

(continues on next page)

(continued from previous page)

```
            'interface_name': 'i2c'
           },
    'rx':  {'model': 'mb_2024_0_2',
            'latency': 0.010,  # latency in seconds in continuous mode
            'sampling_rate': 50,  # number of samples per second
            'interface_name': 'i2c'
           },
    'mux': {'boards':
                {'mux_00':
                     {'model': 'mux_2024_0_X',
                      'electrodes': range(1, 9),
                      'roles': ['A', 'B', 'M', 'N'],
                      'tca_address': None,
                      'tca_channel': 0,
                      'addr1': 'down',
                      'addr2': 'down',
                      },
                 },
            'default': {'interface_name': 'i2c_ext',
                        'voltage_max': 50.,
                        'current_max': 3.}
           }
    }
```

The logging dictionaries divided in:

Listing 4: EXEC_LOGGING_CONFIG: dictionary configuring how the execution commands are being logged by the system. Useful for debugging.

```
# SET THE LOGGING LEVELS, MQTT BROKERS AND MQTT OPTIONS ACCORDING TO YOUR NEEDS
# Execution logging configuration
EXEC_LOGGING_CONFIG = {
    'logging_level': logging.INFO,
    'log_file_logging_level': logging.DEBUG,
    'logging_to_console': True,
    'file_name': f'exec{logging_suffix}.log',
    'max_bytes': 262144,
    'backup_count': 30,
    'when': 'd',
    'interval': 1
}
```

Listing 5: DATA_LOGGING_CONFIG: Dictionary configuring the data logging capabilities of the system

```
# Data logging configuration
DATA_LOGGING_CONFIG = {
    'logging_level': logging.INFO,
    'logging_to_console': True,
    'file_name': f'data{logging_suffix}.log',
    'max_bytes': 16777216,
```

(continues on next page)

(continued from previous page)

```
    'backup_count': 1024,
    'when': 'd',
    'interval': 1
}
```

Listing 6: SOH_LOGGING_CONFIG: Dictionary configuring how the state of health of the system is logged

```
# State of Health logging configuration (For a future release)
SOH_LOGGING_CONFIG = {
    'logging_level': logging.INFO,
    'log_file_logging_level': logging.DEBUG,
    'logging_to_console': True,
    'file_name': f'soh{logging_suffix}.log',
    'max_bytes': 16777216,
    'backup_count': 1024,
    'when': 'd',
    'interval': 1
}
```

The MQTT dictionaries divided in:

Listing 7: MQTT_LOGGING_CONFIG

```
# MQTT logging configuration parameters
MQTT_LOGGING_CONFIG = {
    'hostname': mqtt_broker,
    'port': 1883,
    'qos': 2,
    'retain': False,
    'keepalive': 60,
    'will': None,
    'auth': {'username': 'mqtt_user', 'password': 'mqtt_password'},
    'tls': None,
    'protocol': MQTTv31,
    'transport': 'tcp',
    'client_id': f'{OHMPI_CONFIG["id"]}',
    'exec_topic': f'ohmpi_{OHMPI_CONFIG["id"]}/exec',
    'exec_logging_level': logging.DEBUG,
    'data_topic': f'ohmpi_{OHMPI_CONFIG["id"]}/data',
    'data_logging_level': DATA_LOGGING_CONFIG['logging_level'],
    'soh_topic': f'ohmpi_{OHMPI_CONFIG["id"]}/soh',
    'soh_logging_level': SOH_LOGGING_CONFIG['logging_level']
}
```

Listing 8: MQTT_CONTROL_CONFIG

```
# MQTT control configuration parameters
MQTT_CONTROL_CONFIG = {
    'hostname': mqtt_broker,
    'port': 1883,
    'qos': 2,
```

(continues on next page)

(continued from previous page)

```
    'retain': False,
    'keepalive': 60,
    'will': None,
    'auth': {'username': 'mqtt_user', 'password': 'mqtt_password'},
    'tls': None,
    'protocol': MQTTv31,
    'transport': 'tcp',
    'client_id': f'{OHMPI_CONFIG["id"]}',
    'ctrl_topic': f'ohmpi_{OHMPI_CONFIG["id"]}/ctrl'
}
```

Interfaces

Three interfaces can be used to interact with the OhmPi:

- a *Web interface*: user friendly graphical interface to achieve basic operations for everyday use, such as running a sequence or repeated sequences.
- a *Python interface*: based on the *API reference*, the Python interface allows basic and more advanced operations such as custom acquisition strategies and automation.
- a *IoT interface*: based on the MQTT messaging protocol used in IoT, it is a framework to incorporate the OhmPi system within complex experiments designs comprising other IoT sensors.

Web interface

This is a user friendly graphical interface for new users as well as running quick and easy acquisitions. The webinterface enables to upload sequences, change parameters, run a sequence and download data. To start the interface, open the terminal in the OhmPi folder and type:

```
./run_http_interface.sh
```

The Raspberry Pi of the OhmPi is runs a small webserver to serve the 'index.html' interface. This interface can be accessed locally by opening a browser on the Raspberry Pi and going to *http://localhost:8000*.

Alternatively to a local webserver, the Raspberry Pi can be configured as a used as a Wi-Fi Access Point (AP). Then other laptop or mobile devices can connect to the WiFi of the Raspberry Pi and interact with the webinterface. To configure the Raspberry Pi to act as an access point and run the webserver automatically on start, see instructions on raspap.com and in 'run_http_interface_on_start.txt'.

Once configured, the webserver should start by itself on start and once connected to the Pi, the user can go to 10.3.141.1:8080 to access the interface.

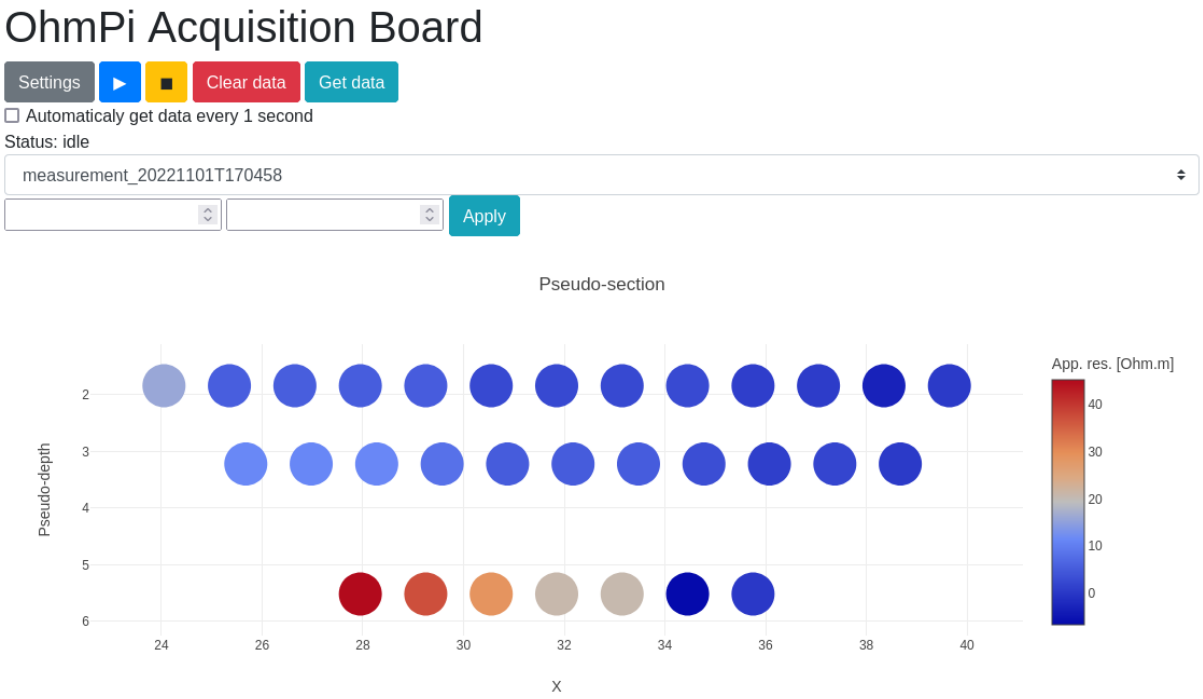


Fig. 9: Web interface with its interactive pseudo-section.

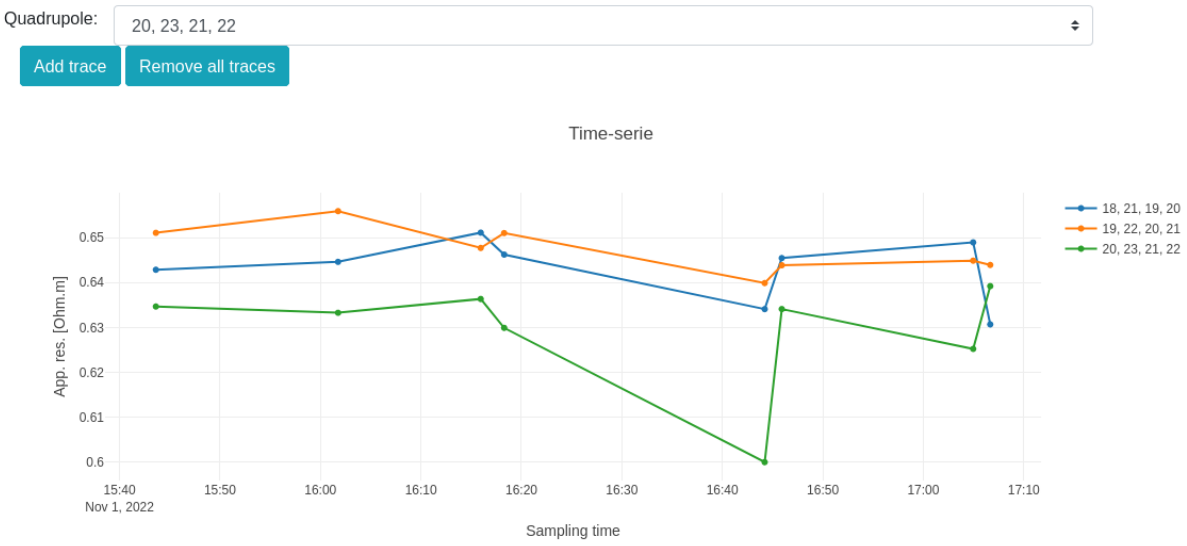


Fig. 10: Evolution of quadrupole apparent resistivity with time.

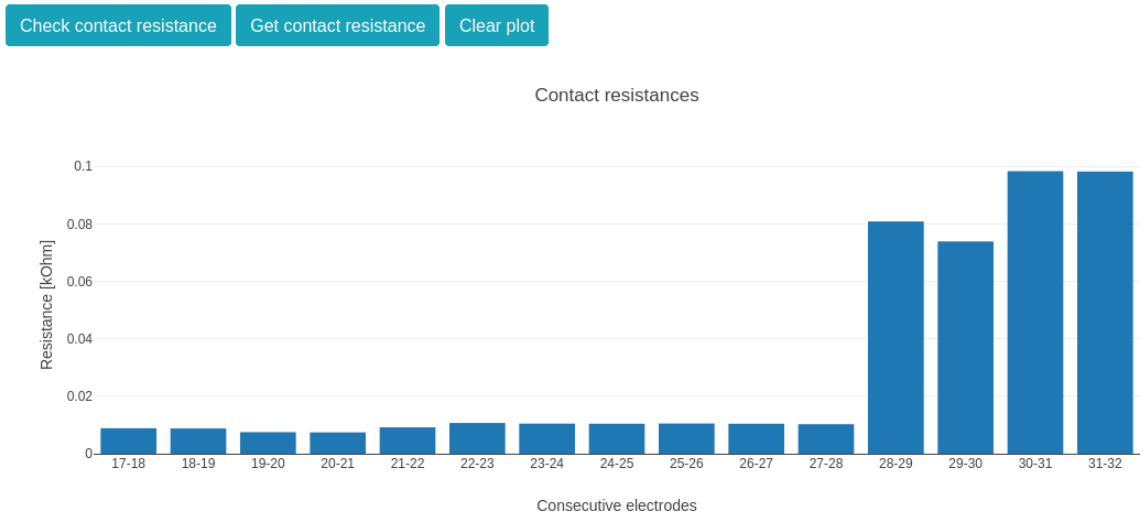


Fig. 11: Contact resistance check.

Python interface

This interface offers a more direct access to the software components, specifically suited for testing or automating acquisition.

By importing the *OhmPi* class from the ohmpi.py, we can control the OhmPi instrument via a Python script or interactively with IPython. It involves using the terminal or a Python IDE such as Thonny on the Raspberry Pi. A connection can also be established via SSH (see PuTTY on Windows or ssh command on macOS/Linux).

To access the Python API, make sure that: - the PYTHONPATH has been correctly configured to export the location of the ohmpi module; - you are in the Python environment created earlier (*souce ohmpy/bin/activate*)

Both of these can be done by executing

```
source env.sh
```

Listing 9: Example of using the Python API to control OhmPi

```
from ohmpi.ohmpi import OhmPi

### Define object from class OhmPi
k = OhmPi()  # this loads default parameters from the disk

### Default parameters can also be edited manually
k.settings['injection_duration'] = 0.5  # injection time in seconds
k.settings['nb_stack'] = 1  # one stack is two half-cycles
k.settings['nbr_meas'] = 1  # number of time the sequence is repeated

### Update settings if needed
k.update_settings({"injection_duration":0.2})

### Set or load sequence
k.sequence = np.array([[1,2,3,4]])    # set numpy array of shape (n,4)
```

(continues on next page)

(continued from previous page)

```
# k.set_sequence('1 2 3 4\n2 3 4 5')    # call function set_sequence and pass a string
# k.load_sequence('ABMN.txt')    # load sequence from a local file

### Run contact resistance check
k.rs_check()

### Run sequence (synchronously - it will wait that all
# sequence is measured before returning the prompt
k.run_sequence()
# k.run_sequence_async()  # sequence is run in a separate thread and the prompt returns␣
↪immediately
# time.sleep(2)
# k.interrupt()  # kill the asynchron sequence

### Single measurement can also be taken with
quadrupole = [1, 4, 2, 3]
k.run_measurement(quadrupole)  # use default acquisition parameters

### Custom or adaptative argument, see help(k.run_measurement)
k.run_measurement(quadrupole,
                  nb_stack=4,  # do 4 stacks (8 half-cycles)
                  injection_duration=1,  # inject for 2 seconds
                  duty_cycle = 0.5) # duty_cycle is
```

For detailed usage, please see the *API reference* or look in the 'examples' folder.

IoT interface

This is an interface designed for an advanced remote usage of the OhmPi such as remote automation, data consumption by multiple processes and interaction with other sensors in the scope of a monitoring. It is based on the MQTT protocol, designed for the Internet of Things (IoT), to interact with the OhmPi.

This option allows interacting remotely with a single OhmPi, a network of OhmPis, as well as auxiliary instruments and sensors. The communication is based on a publish/subscribe approach and involves a MQTT broker.

An example of MQTT broker that can be used is Mosquitto. Depending on the monitoring needs, an MQTT broker can be set up locally on the Raspberry Pi, on a local network or any remote server reachable through the net. A local Mosquitto broker can be set up and enabled to run as a service on the OhmPi using the bash script install_local_mqtt_broker.sh.

MQTT messages include logging messages from the OhmPi and commands sent to the OhmPi. These messages can be examined easily using a third party software such as MQTT Explorer.

Commands sent on the broker are received by the ohmpi.py script that runs on the OhmPi (make sure ohmpi.py starts on reboot) and further processed. MQTT commands are sent in JSON format following the Python API with kwargs as illustrated below:

Listing 10: Updating acquisition settings. Depending on the experiment needs, MQTT brokers can be set up locally on the Raspberry Pi or on a local or remote server.

```
{
  "cmd_id": "3fzxv121UITwGjWYgcz4xw",
```

(continues on next page)

(continued from previous page)

```
  "cmd": "update_settings",
  "kwargs": {
    "settings": {
      "nb_meas": 2,
      "nb_electrodes": 10,
      "nb_stack": 2,
      "injection_duration": 2,
      "sequence_delay": 100
    }
  }
}
```

Listing 11: Check contact resistances

```
{
  "cmd_id": "3fzxv121UITwGjWYgcz4xw",
  "cmd": "rs_check",
}
```

Listing 12: Running a single measurement

```
{
  "cmd_id": "3fzxv121UITwGjWYgcz81x",
  "cmd": "run_measurement",
  "kwargs": {"quad":[1,2,3,4]}
}
```

Listing 13: Running a sequence.

```
{
  "cmd_id": "3fzxv121UITwGjWYgcz4Yw",
  "cmd": "run_sequence",
}
```

Listing 14: Running same sequence multiple times (nb_meas).

```
{
  "cmd_id": "3fzxv121UITwGjWYgcz4Yw",
  "cmd": "run_multiple_sequences",
}
```

Listing 15: Interrupt current acquisition.

```
{
  "cmd_id": "3fzxv121UITwGjWYgcz4xw",
  "cmd": "interrupt",
}
```

Custom processing of messages and tailor-made dashboards for monitoring experiments may be designed using a browser-based flow editor such as Node-red. This may help designing complex IoT experiments and monitoring systems in which OhmPi is a component.

Examples incorporating execution commands and data outputs from OhmPi can be found in the OhmPi examples.

Once Node-RED is installed on the OhmPi, these examples can be accessed separately by running a command in the console such as :

```
node-red basic_ohmpi_flows_node-red.json
```

These examples may require installing some additional node packages in order to work properly. This can be done in the Palette Manager within Node-RED.

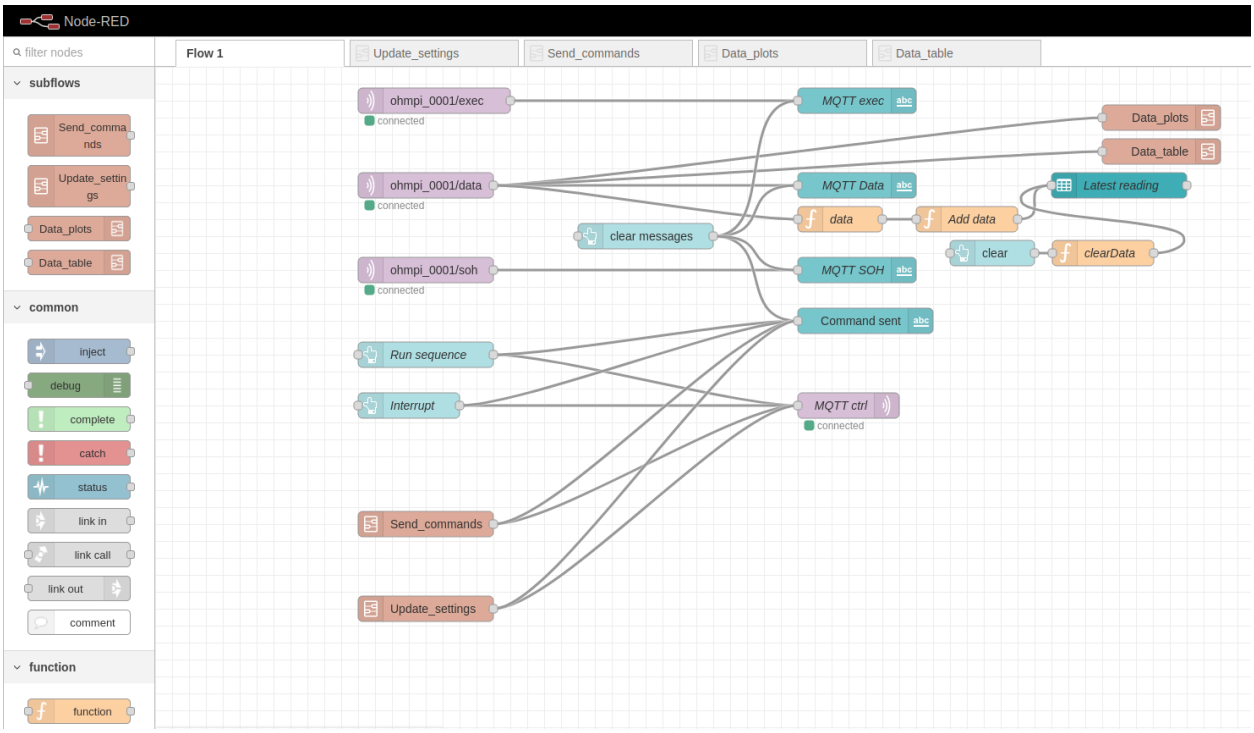


Fig. 12: Example flow in node-red to interact with an OhmPi.

For more documentation dedicated to node-red, please refer to the Node-red cookbooks.

Acquisition settings

This section details the acquisition settings that can be specified for measurement on a quadrupole.

Listing 16: json dictionary containing the default settings contained in `settings/default.json`

```
 {
 "injection_duration": 0.2, # injection duration of one pulse within an injection cycl
 "nb_stack": 1, # number of injection cycles (e.g. 'nb_stack'=1 means one positive and␣
↪one negative pulse)
 "sampling_interval": 2, # sampling interval in ms
 "tx_volt": 5, # injection voltage in V for strategy 'constant' or starting V_AB for␣
↪strategy 'vmax' or 'vmin'
 "duty_cycle": 0.5, # duty cycle for the injection (0-1)
 "strategy": "constant", # injection strategy ("constant", "vmax" or "vmin")
 "fw_in_csv": true, # full waveform saved in a the main csv file. Read by run_sequence()
 "fw_in_zip": true, # full waveform saved in a separate zip file. Read by run_sequence()
```

(continues on next page)

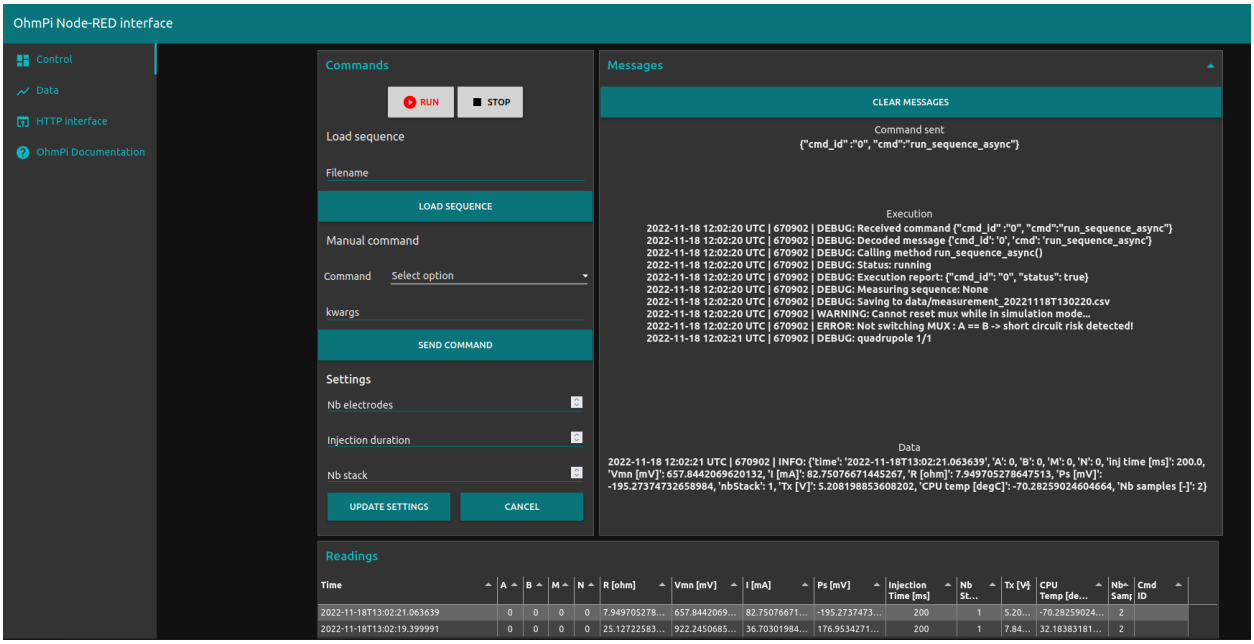


Fig. 13: Example of a dashboard UI created with node-red to interact with an OhmPi - control tab.

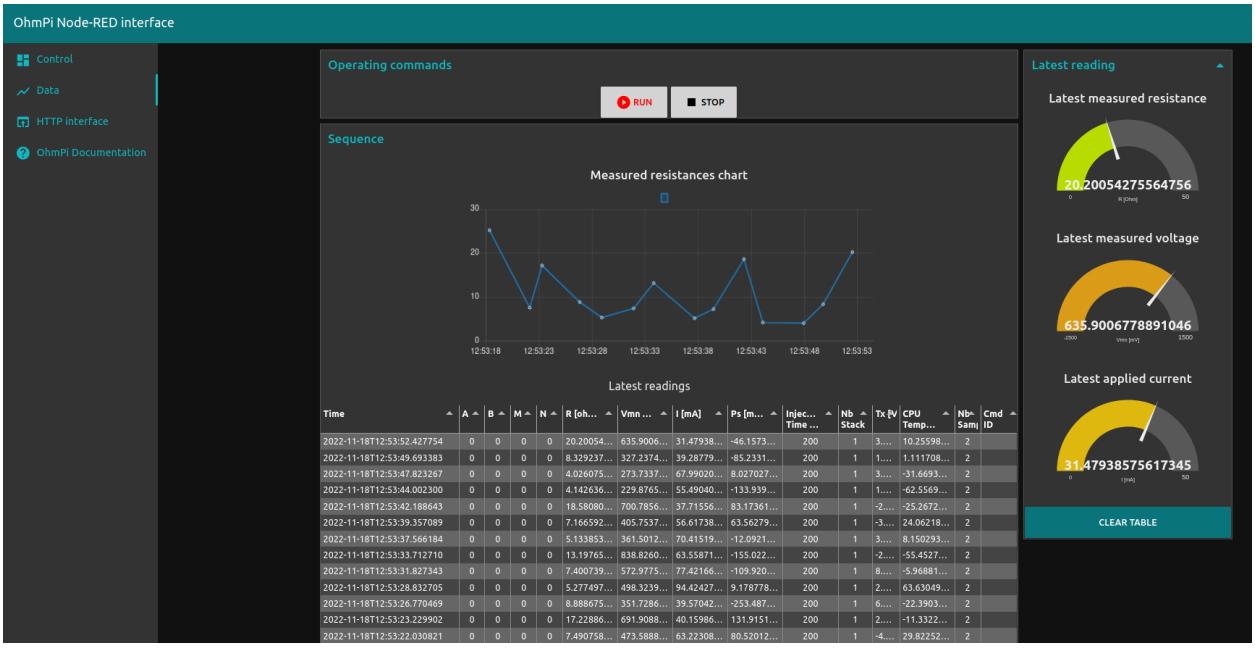


Fig. 14: Example of a dashboard UI created with node-red to interact with an OhmPi - data visualization tab.

(continued from previous page)

```
  "export_path": "data/measurements.csv" # path for data output. Read by run_sequence()
  "nb_meas": 1, # number of sequences repeated when calling run_mutlitple_sequences()
  "sequence_delay": 1, # sleeping time between repeated sequences when calling run_
↪multiple_sequences()
  }
```

For more information on these settings, see the API doc for *ohmpi.ohmpi.OhmPi.run_measurement()*, *ohmpi.ohmpi.OhmPi.run_measurement()* and *ohmpi.ohmpi.OhmPi.run_measurement()*

In addition to these default settings, and for advanced users, additional settings related to the injection strategy "vmax" and "vmin" can also be specified in the json settings file as follows:

Listing 17: Dictionnary containing the advanced settings that can be added to `settings/default.json`

```
  {
  "vab_max": null, # maximum V_AB (in V) bounding the vmax injection strategy. Value is␣
↪capped by vab_max from hardware config. Default is None, which means vmax strategy␣
↪bounded by hardware vab_max from hardware config.
  "iab_max": null, # maximum I_AB (in mA) bounding the vmax injection strategy. Value is␣
↪capped by iab_max from hardware config. Default is None, which means vmax strategy␣
↪bounded by hardware iab_max from hardware config.
  "vmn_max": null, # maximum V_mn (in mV) bounding the vmax injection strategy. Value is␣
↪capped by vmn_max from hardware config. Default is None, which means vmax strategy␣
↪bounded by hardware vmn_max from hardware config.
  "vmn_min": null, # minimum V_mn (in mV) bounding the vmax injection strategy. Value is␣
↪capped by vmn_min from hardware config. Default is None, which means vmax strategy␣
↪bounded by hardware vmn_min from hardware config.
  }
```

For more information on these settings, see the API donc for `OhmPi.run_measurement()`.

Loggers

Loggers have been introduced in this release. They use the excellent logging python package. Specific handlers have been implemented for running with ohmpi.py (one for logging to an MQTT broker (see *IoT interface* for more details) and one for creating zipped rotated logs on disk).

Two loggers have been defined. The first one is dedicated to log operations execution. It is named exec_logger. The second one, named data_logger, is dedicated to log data. A third one is planned to log the state of health (SOH) of the system in a future version.

By default, logs are written to the console (print-like), stored locally in files (a zip is created after some time i.e. every day and/or when the size of the log exceeds a maximum size) and sent to an MQTT broker. Different logging levels may be defined for the different logs and handlers in the *Configuration*.

Advanced users may write new handlers and edit the *setup_loggers.py* file to customize the logging mechanisms to their needs.

Monitoring application

This section details ways to automate measurement acquisition in order to set up the OhmPi as a monitoring tool.

Repeated acquisition at fixed intervals

The easiest way to set up time-lapse acquisition is to perform repeated acquisition of a sequence at fixed intervals. Repeated acquisition can be initiated from the three different *Interfaces*.

Listing 18: Example of code for monitoring.

```
### Run multiple sequences at given time interval
k.settings['nb_meas'] = 3  # run sequence three times
k.settings['sequence_delay'] = 100 # every 100 s
k.run_multiple_sequences()  # asynchron
# k.interrupt()  # kill the asynchron sequence
```

Scheduled acquisition using crontab

Example run_sequence script and crontab screenshot

IoT acquisition and sensor trigger

Example node-red experiment

1.3.4 API reference

created on January 6, 2020. Updates dec 2023; in-depth refactoring May 2023. Hardware: Licensed under CERN-OHL-S v2 or any later version Software: Licensed under the GNU General Public License v3.0 Ohmpi.py is a program to control a low-cost and open hardware resistivity meters within the OhmPi project by Rémi CLEMENT (INRAE), Vivien DUBOIS (INRAE), Hélène GUYARD (IGE), Nicolas FORQUET (INRAE), Yannick FARGIER (IFSTTAR) Olivier KAUFMANN (UMONS), Arnaud WATLET (UMONS) and Guillaume BLANCHY (FNRS/ULiege).

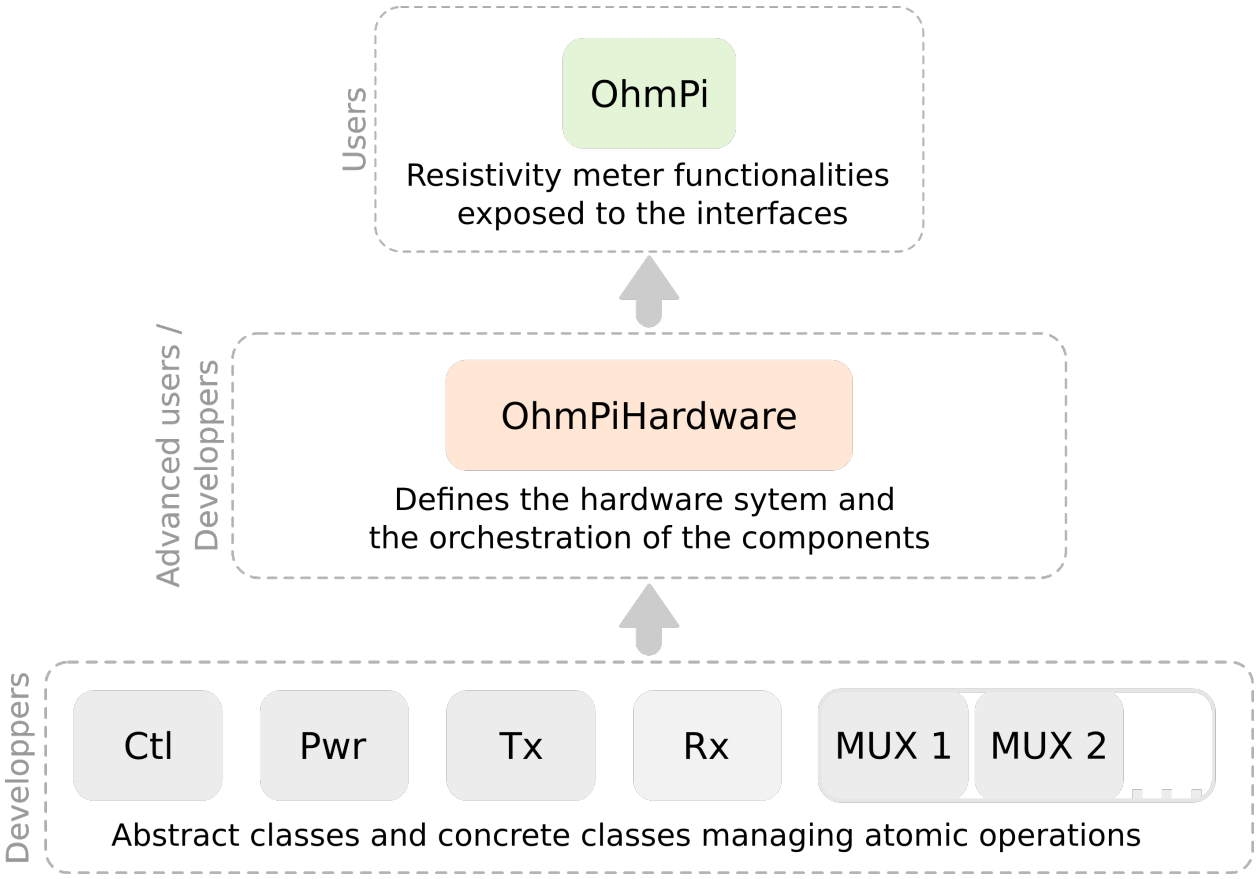
class ohmpi.ohmpi.**OhmPi**(*settings=None*, *sequence=None*, *mqtt=True*, *config=None*)

OhmPi class.

Attributes

sequence

Gets sequence


Users
OhmPi
Resistivity meter functionalities
exposed to the interfaces
Advanced users /
Developpers
OhmPiHardware
Defines the hardware sytem and
the orchestration of the components
Developpers
Ctl
Pwr
Tx
Rx
MUX 1
MUX 2
Abstract classes and concrete classes managing atomic operations

Methods

append_and_save(filename, last_measurement)	Appends and saves the last measurement dict.
create_sequence(nelec[, params, ...])	Creates a sequence of quadrupole.
download_data([cmd_id])	Create a zip of the data folder to then download it easily.
export([fnames, outputdir, ftype, elec_spacing])	Export surveys stored in the 'data/' folder into an output folder.
find_optimal_vab_for_sequence([which, n_samples])	Find optimal Vab based on sample sequence in order to run sequence with fixed Vab. Returns Vab Parameters ---------- which : str Which vab to keep, either "min", "max", "mean" (or other similar numpy method e.g. median) If applying strategy "full_constant" based on vab_opt, safer to chose "min" n_samples: int number of samples to keep within loaded sequence.
get_data([survey_names, cmd_id])	Get available data.
interrupt([cmd_id])	Interrupts the acquisition.
load_sequence(filename[, cmd_id])	Reads quadrupole sequence from file.
plot_last_fw([save_fig, filename])	Plots last full waveform measurement
quit([cmd_id])	Quits OhmPi.
remove_data([cmd_id])	Remove all data in the ´export_path´ folder on the raspberrypi.
repeat_sequence(**kwargs)	Identical to run_multiple_sequences().
reset_mux([cmd_id])	Switches off all multiplexer relays.
restart([cmd_id])	Restarts the Raspberry Pi.
rs_check([vab, cmd_id, couple, tx_volt])	Checks contact resistances.
run_inversion([survey_names, elec_spacing])	Run a simple 2D inversion using ResIPy (https://gitlab.com/hkex/resipy).
run_measurement([quad, nb_stack, ...])	Measures on a quadrupole and returns a dictionary with the transfer resistance.
run_multiple_sequences([sequence_delay, ...])	Runs multiple sequences in a separate thread for monitoring mode.
run_sequence([fw_in_csv, fw_in_zip, cmd_id, ...])	Runs sequence synchronously (=blocking on main thread).
run_sequence_async([cmd_id])	Runs the sequence in a separate thread. Can be stopped by 'OhmPi.interrupt()'.
set_sequence([sequence, cmd_id])	Sets the sequence to acquire.
shutdown([cmd_id])	Shutdown the Raspberry Pi.
switch_mux_off(quadrupole[, cmd_id])	Switches off multiplexer relays for given quadrupole.
switch_mux_on(quadrupole[, bypass_check, cmd_id])	Switches on multiplexer relays for given quadrupole.
test_mux([activation_time, mux_id, cmd_id])	Interactive method to test the multiplexer boards.
update_settings(settings[, cmd_id])	Updates acquisition settings from a json file or dictionary.

get_deprecated_methods
measure
read_quad
stop

append_and_save(*filename: str*, *last_measurement: dict*, *fw_in_csv=None*, *fw_in_zip=None*, *cmd_id=None*)

Appends and saves the last measurement dict.

Parameters

filename
[str] Filename of the .csv.

last_measurement
[dict] Last measurement taken in the form of a python dictionary.

fw_in_csv
[bool, optional] Wether to save the full-waveform data in the .csv (one line per quadrupole). As these readings have different lengths for different quadrupole, the data are padded with NaN. If None, default is read from default.json.

fw_in_zip
[bool, optional] Wether to save the full-waveform data in a separate .csv in long format to be zipped to spare space. If None, default is read from default.json.

cmd_id
[str, optional] Unique command identifier.

create_sequence(*nelec*, *params=[('dpdp', 1, 8)]*, *ireciprocal=False*, *fpath=None*)

Creates a sequence of quadrupole. The sequence is saved automatically to sequences/mysequence.csv and used within OhmPi class. Several type of sequence or sequence with different parameters can be combined together.

Parameters

nelec
[int] Number of electrodes.

params
[list of tuple, optional] Each tuple is the form (<array_name>, param1, param2, ...) Dipole spacing is specified in terms of "number of electrode spacing". Dipole spacing is often referred to 'a'. Number of levels is a multiplier of 'a', often refere to 'n'. For multigradient array, an additional parameter 's' is needed. Types of sequences available are : - ('wenner', a) - ('dpdp', a, n) - ('schlum', a, n) - ('multigrad', a, n, s)

ireciprocal
[bool, optional] If True, will add reciprocal quadrupoles (so MNAB) to the sequence.

fpath
[str, optional] Path where to save the sequence (including filename and extension). By default, sequence is saved in ohmpi/sequences/sequence.txt.

download_data(*cmd_id=None*)

Create a zip of the data folder to then download it easily.

export(*fnames=None*, *outputdir=None*, *ftype='bert'*, *elec_spacing=1*)

Export surveys stored in the 'data/' folder into an output folder.

Parameters

fnames
[list of str, optional] List of path (not filename) to survey in ohmpi format to be converted.

outputdir
[str, optional] Path of the output directory where the new files are stored. If None, a directory called 'output' is created in OhmPi.

ftype
[str, optional] Type of export. To be chosen between: - bert (same as pygimli) - pygimli (same as bert) - protocol (for resipy, R2 codes)

elec_spacing
[float, optional] Electrode spacing in meters. Same electrode spacing is assumed.

`find_optimal_vab_for_sequence`(*which='mean'*, *n_samples=10*, ***kwargs*)

Find optimal Vab based on sample sequence in order to run sequence with fixed Vab. Returns Vab Parameters ———- which : str

Which vab to keep, either "min", "max", "mean" (or other similar numpy method e.g. median) If applying strategy "full_constant" based on vab_opt, safer to chose "min"

n_samples: int
number of samples to keep within loaded sequence

kwargs : kwargs passed to Ohmpi.run_sequence

Returns

Vab_opt
[float [in V]] Optimal Vab value

`get_data`(*survey_names=None*, *cmd_id=None*)

Get available data.

Parameters

survey_names
[list of str, optional] List of filenames already available from the html interface. So their content won't be returned again. Only files not in the list will be read.

cmd_id
[str, optional] Unique command identifier.

`interrupt`(*cmd_id=None*)

Interrupts the acquisition.

Parameters

cmd_id
[str, optional] Unique command identifier.

`load_sequence`(*filename: str*, *cmd_id=None*)

Reads quadrupole sequence from file.

Parameters

filename
[str] Path of the .csv or .txt file with A, B, M and N electrodes. Electrode index start at 1.

cmd_id
[str, optional] Unique command identifier.

Returns

sequence
[numpy.ndarray] Array of shape (number quadrupoles * 4).

plot_last_fw(*save_fig=False*, *filename=None*)

Plots last full waveform measurement

Parameters

save_fig: boolean, optional - default (False)
filename: str, optional. Path to save plot. By default figures/test.png

quit(*cmd_id=None*)

Quits OhmPi.

Parameters

cmd_id
[str, optional] Unique command identifier.

remove_data(*cmd_id=None*)

Remove all data in the ´export_path´ folder on the raspberrypi.

Parameters

cmd_id
[str, optional] Unique command identifier.

repeat_sequence(***kwargs*)

Identical to run_multiple_sequences().

reset_mux(*cmd_id=None*)

Switches off all multiplexer relays.

Parameters

cmd_id
[str, optional] Unique command identifier.

restart(*cmd_id=None*)

Restarts the Raspberry Pi.

Parameters

cmd_id
[str, optional] Unique command identifier.

rs_check(*vab=5*, *cmd_id=None*, *couple=None*, *tx_volt=None*)

Checks contact resistances. Strategy: we just open A and B, measure the current and using vAB set or assumed (12V assumed for battery), we compute Rab.

Parameters

vab
[float, optional] Voltage of the injection.

couple
[array, for selecting a couple of electrode for checking resistance]

cmd_id
[str, optional] Unique command identifier.

tx_volt
[float, optional DEPRECATED] Save as vab.

run_inversion(*survey_names=None*, *elec_spacing=1*, ***kwargs*)

Run a simple 2D inversion using ResIPy (https://gitlab.com/hkex/resipy).

Parameters

survey_names
[list of string, optional] Filenames of the survey to be inverted (including extension).

elec_spacing
[float (optional)] Electrode spacing in meters. We assume same electrode spacing everywhere. Default is 1 m.

kwargs
[optional] Additional keyword arguments passed to *resipy.Project.invert()*. For instance *reg_mode* == 0 for batch inversion, *reg_mode* == *2* for time-lapse inversion. See ResIPy document for more information on options available (https://hkex.gitlab.io/resipy/).

Returns

xzv
[list of dict] Each dictionnary with key 'x' and 'z' for the centroid of the elements and 'v' for the values in resistivity of the elements.

run_measurement(*quad=None*, *nb_stack=None*, *injection_duration=None*, *duty_cycle=None*, *strategy=None*, *tx_volt=None*, *vab=None*, *vab_init=None*, *vab_min=None*, *vab_req=None*, *vab_max=None*, *iab_min=None*, *iab_req=None*, *min_agg=None*, *iab_max=None*, *vmn_min=None*, *vmn_req=None*, *vmn_max=None*, *pab_min=None*, *pab_req=None*, *pab_max=None*, *cmd_id=None*, ***kwargs*)

Measures on a quadrupole and returns a dictionary with the transfer resistance.

Parameters

quad
[iterable (list of int)] Quadrupole to measure, just for labelling. Only switch_mux_on/off really create the route to the electrodes.

nb_stack
[int, optional] Number of stacks. A stack is considered two pulses (one positive, one negative). If 0, we will look for the best voltage.

injection_duration
[int, optional] Injection time in seconds.

duty_cycle
[float, optional] Duty cycle (default=0.5) of injection square wave.

strategy
[str, optional, default: constant] Define injection strategy (if power is adjustable, otherwise constant vab, generally 12V battery is used). Either: - vmax : compute Vab to reach a maximum Vmn_max and Iab without exceeding vab_max - vmin : compute Vab to reach at least Vmn_min - constant : apply given Vab but checks if expected readings not out-of-range - full_constant: apply given Vab with no out-of-range checks for optimising duration at the risk of out-of-range readings Safety check (i.e. short voltage pulses) performed prior to injection to ensure injection within bounds defined in vab_max, iab_max, vmn_max or vmn_min. This can adapt Vab. To bypass safety check before injection, vab should be set equal to vab_max (not recommended)

vab_init
[float, optional] Initial injection voltage [V] Default value set by config or boards specs

vab_min
[float, optional] Minimum injection voltage [V] Default value set by config or boards specs

vab_req
[float, optional] Requested injection voltage [V] Default value set by config or boards specs

vab_max
[float, optional] Maximum injection voltage [V] Default value set by config or boards specs

iab_min
[float, optional] Minimum current [mA] Default value set by config or boards specs

iab_req
[float, optional] Requested iab [mA] Default value set by config or boards specs

iab_max
[float, optional] Maximum iab allowed [mA]. Default value set by config or boards specs

pab_min
[float, optional] Minimum power [W]. Default value set by config or boards specs

pab_req
[float, optional] Requested power [W]. Default value set by config or boards specs

pab_max
[float, optional] Maximum power allowed [W]. Default value set by config or boards specs

vmn_min: float, optional
Minimum Vmn [mV] (used in strategy vmin). Default value set by config or boards specs

vmn_req: float, optional
Requested Vmn [mV] (used in strategy vmin). Default value set by config or boards specs

vmn_max: float, optional
Maximum Vmn [mV] (used in strategy vmin). Default value set by config or boards specs

min_agg
[bool, optional, default: False] when set to True, requested values are aggregated with the 'or' operator, when False with the 'and' operator

tx_volt
[float, optional # deprecated] For power adjustable only. If specified, voltage will be imposed.

vab
[float, optional] For power adjustable only. If specified, voltage will be imposed.

cmd_id
[str, optional] Unique command identifier.

`run_multiple_sequences`(*sequence_delay=None*, *nb_meas=None*, *fw_in_csv=None*, *fw_in_zip=None*, *cmd_id=None*, ***kwargs*)

Runs multiple sequences in a separate thread for monitoring mode.
Can be stopped by 'OhmPi.interrupt()'. Additional arguments are passed to run_measurement().

Parameters

sequence_delay
[int, optional] Number of seconds at which the sequence must be started from each others.

nb_meas
[int, optional] Number of time the sequence must be repeated.

fw_in_csv
[bool, optional] Wether to save the full-waveform data in the .csv (one line per quadrupole). As these readings have different lengths for different quadrupole, the data are padded with NaN. If None, default is read from default.json.

fw_in_zip
[bool, optional] Wether to save the full-waveform data in a separate .csv in long format to be zipped to spare space. If None, default is read from default.json.

cmd_id
[str, optional] Unique command identifier.

kwargs
[dict, optional] See help(OhmPi.run_measurement) for more info.

`run_sequence`(*fw_in_csv=None*, *fw_in_zip=None*, *cmd_id=None*, *save_strategy_fw=False*, ***kwargs*)

Runs sequence synchronously (=blocking on main thread).
Additional arguments are passed to run_measurement().

Parameters

fw_in_csv
[bool, optional] Whether to save the full-waveform data in the .csv (one line per quadrupole). As these readings have different lengths for different quadrupole, the data are padded with NaN. If None, default is read from default.json.

fw_in_zip
[bool, optional] Whether to save the full-waveform data in a separate .csv in long format to be zipped to spare space. If None, default is read from default.json.

cmd_id
[str, optional] Unique command identifier.

`run_sequence_async`(*cmd_id=None*, ***kwargs*)

Runs the sequence in a separate thread. Can be stopped by 'OhmPi.interrupt()'.
Additional arguments are passed to run_measurement().

Parameters

cmd_id
[str, optional] Unique command identifier.

`property sequence`
Gets sequence

`set_sequence`(*sequence=None*, *cmd_id=None*)
Sets the sequence to acquire.

Parameters

sequence
[list of list or array_like] Sequence of quadrupoles (list of list or array_like).

cmd_id: str, optional
Unique command identifier.

shutdown(*cmd_id=None*)

Shutdown the Raspberry Pi.

Parameters

cmd_id
[str, optional] Unique command identifier

switch_mux_off(*quadrupole*, *cmd_id=None*)

Switches off multiplexer relays for given quadrupole.

Parameters

quadrupole
[list of 4 int] List of 4 integers representing the electrode numbers.

cmd_id
[str, optional] Unique command identifier.

switch_mux_on(*quadrupole*, *bypass_check=False*, *cmd_id=None*)

Switches on multiplexer relays for given quadrupole.

Parameters

quadrupole
[list of 4 int] List of 4 integers representing the electrode numbers.

bypass_check: bool, optional
Bypasses checks for A==M or A==N or B==M or B==N (i.e. used for rs-check).

cmd_id
[str, optional] Unique command identifier.

test_mux(*activation_time=0.2*, *mux_id=None*, *cmd_id=None*)

Interactive method to test the multiplexer boards.

Parameters

activation_time
[float, optional] Time in seconds during which the relays are activated.

mux_id
[str, optional] ID of the mux_board to test.

cmd_id
[str, optional] Unique command identifier.

update_settings(*settings: str*, *cmd_id=None*)

Updates acquisition settings from a json file or dictionary. Parameters can be: - nb_electrodes (number of electrode used, if 4, no MUX needed) - injection_duration (in seconds) - nb_meas (total number of times the sequence will be run) - sequence_delay (delay in second between each sequence run) - nb_stack (number of stack for each quadrupole measurement) - strategy (injection strategy: constant, vmax, vmin) - duty_cycle (injection duty cycle comprised between 0.5 - 1) - export_path (path where to export the data, timestamp will be added to filename)

Parameters

settings
[str, dict] Path to the .json settings file or dictionary of settings.

cmd_id
[str, optional] Unique command identifier.

class ohmpi.hardware_system.**OhmPiHardware**(***kwargs*)

OhmPiHardware class. A class to operate the system of assembled components as defined in the ohmpi/config.py file

Attributes

pulses
pwr_state

Methods

compute_vab([vab_init, vab_min, vab_req, ...])	Estimates best Vab voltage based on different strategies.
reset_mux()	Switches off all multiplexer relays.
switch_mux(electrodes[, roles, state])	Switches on multiplexer relays for given quadrupole.
test_mux([channel, activation_time])	Interactive method to test the multiplexer.
vab_square_wave(vab, cycle_duration[, ...])	Performs a Vab injection following a square wave and records full waveform data.

calibrate_rx_bias
discharge_pwr
last_dev
last_iab
last_iab_dev
last_resistance
last_sp
last_vmn
last_vmn_dev
select_samples

compute_vab(*vab_init=5.0*, *vab_min=None*, *vab_req=None*, *vab_max=None*, *iab_min=None*, *iab_req=None*, *iab_max=None*, *vmn_min=None*, *vmn_req=None*, *vmn_max=None*, *pab_min=None*, *pab_req=None*, *pab_max=None*, *min_agg=False*, *polarities=(1, -1)*, *pulse_duration=0.1*, *delay=0.05*, *diff_vab_lim=2.5*, *n_steps=4*, *n_sigma=2.0*, *filename=None*, *quad_id=0*)

Estimates best Vab voltage based on different strategies. In “vmax” and “vmin” strategies, we iteratively increase/decrease the vab while checking vmn < vmn_max, vmn > vmn_min and iab < iab_max. We do a maximum of n_steps and when the difference between the two steps is below diff_vab_lim or we reached the maximum number of steps, we return the vab found.

Parameters

pulse_duration
[float, optional] Time in seconds for the pulse used to compute optimal Vab.

vab_init
[float, optional] Voltage to apply for guessing the best voltage. 5 V applied by default. If strategy “constant” is chosen, constant voltage to applied is “vab”.

vab_max
[float, optional] Maximum injection voltage to apply to tx (used by all strategies).

vmn_max
[float, optional] Maximum voltage target for rx (used by vmax strategy).

vmn_min
[float, optional] Minimum voltage target for rx (used by vmin strategy).

polarities
[list of int, optional] Polarity of the AB injection used to compute optimal Vab. Default is one positive, then one negative.

p_max
[float, optional] Maximum power that the device can support/sustain.

diff_vab_lim
[float, optional] Minimal change in vab between steps for continuing the search for optimal vab. If change between two steps is below the diff_vab_lim, we have found the optimal vab.

n_steps
[int, optional] Number of steps to try to find optimal vab. Each step last at least injection_duration*len(polarities) seconds.

Returns

vab
[float] Proposed Vab according to the given strategy. :param vmn_min: :param vmn_max: :param iab_max:

`reset_mux`()

Switches off all multiplexer relays.

`switch_mux`(*electrodes*, *roles=None*, *state='off'*, ***kwargs*)

Switches on multiplexer relays for given quadrupole.

Parameters

electrodes
[list] List of integers representing the electrode ids.

roles
[list, optional] List of roles of electrodes, optional

state
[str, optional] Either 'on' or 'off'.

`test_mux`(*channel=None*, *activation_time=1.0*)

Interactive method to test the multiplexer.

Parameters

channel
[tuple, optional] (electrode_nr, role) to test.

activation_time
[float, optional] Time in seconds during which the relays are activated.

`vab_square_wave`(*vab*, *cycle_duration*, *sampling_rate=None*, *cycles=3*, *polarity=1*, *duty_cycle=1.0*, *append=False*)

Performs a Vab injection following a square wave and records full waveform data. Calls in function Vab_pulses.

Parameters

vab: float
Injection voltage [V]

cycle_duration: float
Duration of one cycle within the square wave (in seconds)

sampling_rate: float, None Default None
Sampling rate for Rx readings

cycles: integer, Default: 3
Number of cycles

polarity: 1, 0 , -1
Starting polarity

duty_cycle: float (0 to 1)
Duty cycle of injection wave

append: bool, optional
Default: False

1.4 Troubleshooting

We encourage users to report any issue, or bugs as an issue in the official repository on GitLab. Please have a look at existing open and closed issues before posting a new one. We have compiled here below a list of common issues and and explanations on how to fix them. For issue with the hardware, make sure your board passes the hardware checks (*Check measurement board v2024*, *Check MUX board v2024*).

1.4.1 Communication issue between components (I2C, pull-up)

If you get an I2C communication error or cannot see some I2C address with *i2cdetect*.

Most components of the OhmPi communicate via I2C protocol. This protocol works with two lines (SDA and SCL) that **must be pulled-up** at rest. The pull-up resistor consist in placing a 100k (or similar values) resistor between the line and VDD (5V in this case).

Check with the multimeter the voltage between SDA/SCL and the ground to see if it reaches 5V at rest. If it's not the case, you may need stronger pull-up (smaller value of pull-up resistor).

Note: On the measurement board v2024, the I2C isolator from Mikroe, already has pull-up that adds to the pull-up already on the ADS1115 board. If the ADS1115 of the Vmn part cannot be seen by i2cdetect, we recommend to remove the pull-up resistors on the Mikroe I2C isolator board (see note fig29 in *Build measurement board v2024*)

1.4.2 Issue with the pulses between A and B

In the measurement board v2023, this is likely due to the optical relays not opening or closing properly. These relays are quite fragile and, from experience, are easily damaged. Check if the optical relay are still working by measuring if they are conductor when turned on using a multimeter without connecting any electrodes to A and B.

If an optical relay is broken, you will have to replace it with a new one.

In the measurement board v2024, these optical relays are replaced by mechanical relays which are more robust and shoudn't cause any issue.

1.4.3 Values given is not the correct one

One possible cause is that the **shunt resistor was burned**. Once burned, the value of the resistor is not correct anymore and we advice to change it. To see if the shunt is burned, you can measure the value of the shunt resistor to see if it still has the expected value.

Another possibility is that the MN voltage you are trying to measure is **over the range of the ADC** (+/- 4.5 V effective range for ADS1115). You can easily check that by measuring the voltage at MN with a voltmeter.

In the measurement board v2024, the current sensing part is replaced by a click board. It is possible that the shunt resistance on this click board is burned due to malfunction. In this case, erroneous value of current will be given. The click board must be replaced to solve the issue.

1.4.4 Noise in the Vmn signal

The OhmPi system does not filter the signal for 50 or 60Hz power noise. This noise can appear in the Vmn reading if the Tx or Rx battery is connected to a charger connected to the grid.

To solve this, you may need to design a system that disconnect the charger (turn if off) when doing a measurement.

1.4.5 Unexpected electrode takeout

The IDC socket of the mux2023 and mux2024 are not wired identically. Double check that you connected the right electrode to the right ribbon cable (see drawings in the assembling tutorials)

1.4.6 Strong decay in current

A strong decay in current can be an indication that the battery cannot supply enough power to the DPH5005 to main the requested voltage.

1.4.7 Modbus error

Modbus is the procotol used to communicated between the DPH5005 and the Raspberry Pi via a USB cable. If the Pi cannot detect the DPS, a modbus error can happen. Make sure the USB cable is ok and that the DPH5005 is supplied.

1.5 How to contribute

OhmPi is an open source system and contributions in terms of hardware and software are welcome. However, in order to maintain the project on tracks and promote exchange and reuse, it is necessary that contributors wishing to develop new software or hardware components follow a few basic steps detailed below. Contributors are also kindly asked to get in touch with the OhmPi developing team.

1.5.1 Developing hardware components

Here is a non exhaustive wish list of new hardware features that are planned/hoped to be developed in the future. Contributors are welcomed to join forces to make this list come true, or propose new ideas by creating a new issue in the Gitlab repository.

1.5.2 Developing software features

If the new developments purely concern the software (e.g. bug fix, new acquisition strategy, etc.), then please follow the git best practices by first creating an new issue and then create a local branch linked to this issue. Once the new feature is implemented, a pull request can be initiated.

1.5.3 Software interface to new hardware components

This section is intended for developers of a new hardware component as part of an OhmPi system.

It presents some advices and best practices that should help developing new hardware components to work within an OhmPi system.

Two cases should be distinguished when dealing with hardware development components:

1- Developments of a hardware component that comply with the way the OhmPi Hardware System works. Such developments typically focus on improving an existing component (reduce cost, improve performance, adapt range to specific applications, design with easily available parts…). The newly created hardware component will expose the minimal functionalities required by hardware_system for this type of component.

2- Developments of a hardware component that introduce changes in the way the OhmPi Hardware System works. Such developments do not only focus on improving a single component but also on the way to operate the system. A discussion with developers of the OhmPi_Hardware and OhmPi classes should be initiated at a very early stage to investigate the best ways to design and implement a working solution.

If you are dealing with case 1 or have designed a development path and strategy, you are ready to start.

First the hardware board/device should be conceived and designed. The Ohmpi team recommends that contributors design or import their boards within KiCAD and that both schemas and PCB are shared.

When developing a new component Class, always start your development in a new branch. 1- Create a new python file or make a copy and modify an existing similar component file. All hardware component modules are stored in the ohmpi/hardware_component directory. In the newly created python file, define a new class based on the relevant abstract class of abstract_hardware_components.py. Implement the abstract methods to interact with your hardware. Name the file according to the name of the component. Make sure to place this new python module in the ohmpi/hardware_component directory.

2- Create a new configuration file or make a copy and modify an existing configuration file. All existing config files are stored in the ohmpi/hardware_component directory. In this newly created config file, describe your system including the new component in the HARDWARE_CONFIG dictionary. Name it config_XXX.py where XXX should be replaced by an expression describing the system. Make sure to place your new config file in the ohmpi/configs directory.

3- Create a new script or make a copy and modify of an existing script for testing the component. In this script, write python code where you will conduct the tests of the required functionalities of the new component.

1.5.4 Hardware components API

class ohmpi.hardware_components.abstract_hardware_components.**CtlAbstract**(***kwargs*)

CTlAbstract Class Abstract class for controller

Attributes

cpu_temperature

class ohmpi.hardware_components.abstract_hardware_components.**MuxAbstract**(***kwargs*)

MUXAbstract Class Abstract class for MUX

Attributes

barrier

Methods

switch([elec_dict, state, bypass_check, ...])	Switch a given list of electrodes with different roles.
switch_one([elec, role, state])	Switches one single relay.
test(elec_dict[, activation_time])	Method to test the multiplexer.

reset

switch(*elec_dict=None*, *state='off'*, *bypass_check=False*, *bypass_ab_check=False*)

Switch a given list of electrodes with different roles. Electrodes with a value of 0 will be ignored.

Parameters

elec_dict
[dictionary, optional] Dictionary of the form: {role: [list of electrodes]}.

state
[str, optional] Either 'on' or 'off'.

bypass_check: bool, optional
Bypasses checks for A==M or A==M or B==M or B==N (i.e. used for rs-check)

bypass_check: bool, optional
Bypasses checks for A==B (i.e. used for testing r_shunt). Should be used with caution.

abstract switch_one(*elec=None*, *role=None*, *state=None*)

Switches one single relay.

Parameters

elec
role
state
[str, optional] Either 'on' or 'off'.

test(*elec_dict*, *activation_time=1.0*)

Method to test the multiplexer.

Parameters

elec_dict
[dictionary, optional] Dictionary of the form: {role: [list of electrodes]}.

activation_time
[float, optional] Time in seconds during which the relays are activated.

class ohmpi.hardware_components.abstract_hardware_components.**PwrAbstract**(***kwargs*)

PwrAbstract Class Abstract class for Power module

Attributes

current
pwr_state
voltage

Methods

battery_voltage
reload_settings
reset_voltage

class ohmpi.hardware_components.abstract_hardware_components.**RxAbstract**(***kwargs*)

RXAbstract Class Abstract class for RX

Attributes

bias
Gets the RX bias

gain
latency
Gets the Rx latency

sampling_rate
voltage
Gets the voltage VMN in Volts

Methods

gain_auto
reset_gain

property **bias**

Gets the RX bias

property **latency**

Gets the Rx latency

abstract property **voltage**

Gets the voltage VMN in Volts

class ohmpi.hardware_components.abstract_hardware_components.**TxAbstract**(***kwargs*)

TxAbstract Class Abstract class for TX module

Attributes

current
Gets the current IAB in Amps

gain
injection_duration
latency
Gets the Tx latency

measuring
polarity
pwr_state
tx_bat
voltage
Gets the voltage VAB in Volts

Methods

inject([polarity, injection_duration, ...])	Abstract method to define injection.
voltage_pulse([voltage, length, polarity])	Generates a square voltage pulse

current_pulse
discharge_pwr
reset_gain

abstract property current

Gets the current IAB in Amps

abstract inject(*polarity=1*, *injection_duration=None*, *switch_pwr=False*)

Abstract method to define injection.

Parameters

polarity: int, default 1
Injection polarity, can be eiter 1, 0 or -1.

injection_duration: float, default None
Injection duration in seconds.

switch_pwr: bool
Switch on and off tx.pwr.

property latency

Gets the Tx latency

property voltage

Gets the voltage VAB in Volts

voltage_pulse(*voltage=0.0*, *length=None*, *polarity=1*)

Generates a square voltage pulse

Parameters

voltage: float, optional
Voltage to apply in volts, tx_v_def is applied if omitted.

length: float, optional
Length of the pulse in seconds

polarity: 1,0,-1
Polarity of the pulse

class ohmpi.hardware_components.mb_2023_0_X.**Rx**(***kwargs*)

RX class

Attributes

bias
Gets the RX bias

gain
latency
Gets the Rx latency

sampling_rate
voltage
Gets the voltage VMN in Volts

Methods

gain_auto
reset_ads
reset_gain

property **voltage**

Gets the voltage VMN in Volts

class ohmpi.hardware_components.mb_2023_0_X.**Tx**(***kwargs*)

Tx Class

Attributes

current
Gets the current IAB in Amps

gain
injection_duration
latency
Gets the Tx latency

measuring
polarity
pwr_state
tx_bat
voltage
Gets the voltage VAB in Volts

Methods

inject([polarity, injection_duration, ...])	Abstract method to define injection.
voltage_pulse([voltage, length, polarity])	Generates a square voltage pulse

current_pulse
discharge_pwr
gain_auto
reset_ads
reset_gain
reset_mcp

property current

Gets the current IAB in Amps

inject(*polarity=1*, *injection_duration=None*, *switch_pwr=False*)

Abstract method to define injection.

Parameters

polarity: int, default 1
Injection polarity, can be eiter 1, 0 or -1.

injection_duration: float, default None
Injection duration in seconds.

switch_pwr: bool
Switch on and off tx.pwr.

voltage_pulse(*voltage=None*, *length=None*, *polarity=1*)

Generates a square voltage pulse

Parameters

voltage: float, optional
Voltage to apply in volts, tx_v_def is applied if omitted.

length: float, optional
Length of the pulse in seconds

polarity: 1,0,-1
Polarity of the pulse

class ohmpi.hardware_components.mb_2024_0_2.**Rx**(***kwargs*)

RX Class

Attributes

bias
Gets the RX bias

gain
latency
Gets the Rx latency

sampling_rate
voltage
Gets the voltage VMN in Volts

Methods

gain_auto
reset_ads
reset_gain
reset_mcp

property voltage

Gets the voltage VMN in Volts

class ohmpi.hardware_components.mb_2024_0_2.**Tx**(***kwargs*)

TX Class

Attributes

current
Gets the current IAB in Amps

gain
injection_duration
latency
Gets the Tx latency

measuring
polarity
pwr_state
tx_bat
voltage
Gets the voltage VAB in Volts

Methods

current_pulse([current, length, polarity])	Generates a square current pulse.
inject([polarity, injection_duration])	Abstract method to define injection.
voltage_pulse([voltage, length, polarity])	Generates a square voltage pulse

discharge_pwr
gain_auto
reset_ads
reset_gain
reset_mcp

current_pulse(*current=None*, *length=None*, *polarity=1*)

Generates a square current pulse. Currenttly no DPS can handle this…

Parameters

voltage: float, optional
Voltage to apply in volts, tx_v_def is applied if omitted.

length: float, optional
Length of the pulse in seconds

polarity: 1,0,-1
Polarity of the pulse

inject(*polarity=1*, *injection_duration=None*)

Abstract method to define injection.

Parameters

polarity: int, default 1
Injection polarity, can be eiter 1, 0 or -1.

injection_duration: float, default None
Injection duration in seconds.

switch_pwr: bool
Switch on and off tx.pwr.

class ohmpi.hardware_components.mux_2023_0_X.**Mux**(***kwargs*)

Attributes

barrier

Methods

switch([elec_dict, state, bypass_check, ...])	Switch a given list of electrodes with different roles.
switch_one([elec, role, state])	Switches one single relay.
test(elec_dict[, activation_time])	Method to test the multiplexer.

reset
reset_i2c_ext_tca
reset_one
reset_tca

switch_one(*elec=None*, *role=None*, *state=None*)

Switches one single relay.

Parameters

elec
role
state
[str, optional] Either 'on' or 'off'.

class ohmpi.hardware_components.mux_2024_0_X.**Mux**(***kwargs*)

Attributes

barrier

Methods

switch([elec_dict, state, bypass_check, ...])	Switch a given list of electrodes with different roles.
switch_one([elec, role, state])	Switches one single relay.
test(elec_dict[, activation_time])	Method to test the multiplexer.

reset
reset_i2c_ext_tca
reset_one

switch_one(*elec=None*, *role=None*, *state=None*)

Switches one single relay.

Parameters

elec
role
state
[str, optional] Either 'on' or 'off'.

class ohmpi.hardware_components.pwr_batt.**Pwr**(***kwargs*)

Attributes

current
pwr_state
voltage

Methods

battery_voltage
reload_settings
reset_voltage

class ohmpi.hardware_components.pwr_dph5005.**Pwr**(***kwargs*)

Attributes

current
current_max
pwr_state
voltage
voltage_max

Methods

battery_voltage
current_max_default
current_overload
power_max
reload_settings
reset_voltage
voltage_default

class ohmpi.hardware_components.raspberry_pi.Ctl(***kwargs*)

Attributes

cpu_temperature

Methods

reset_modbus

1.6 Examples of applications

TODO (e.g. OhmPi in Inrae, OhmPi in Rocherfort,)

1.7 Archived versions

These versions are not supported anymore.

1.7.1 OhmPi V 1.01 (limited to 32 electrodes)

Warning: This version corresponds to the version published in the Hardware X journal. However, we have corrected the bugs that existed on this version and explained the missing mounting points in detail below. We invite you to refer to this document to assemble Ohmpi V1.01.

Warning: Ohmpi is a participative project open to all, it requires skills in electronics and to respect the safety rules. Ohmpi must be assembled in a professional context and by people competent in electronics. The Ohmpi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The Ohmpi team cannot be held responsible if the equipment does not work after assembly.

The philosophy of Ohmpi

The philosophy of Ohmpi V1.01 is to offer a multi electrode resistivity meter, from a set of commercially available electronic cards it is a resistivity meter limited to 32 electrodes only. It is limited to low-current injection, but suitable for small laboratory experiments and small field time monitoring

Technical data

Parameter	**Specifications**	Units
Electrodes	32	
Operating temperature	0 to 50	°C
Power consumption of CPU and control system	18.5	W
Voltage injection	9	V
Battery	12	V
Current	0 to 50	mA
Min pulse duration	150	ms
Input impedance	36	MOhm
Data storage	micro SD card	
Resolution	O.O1	Ohm

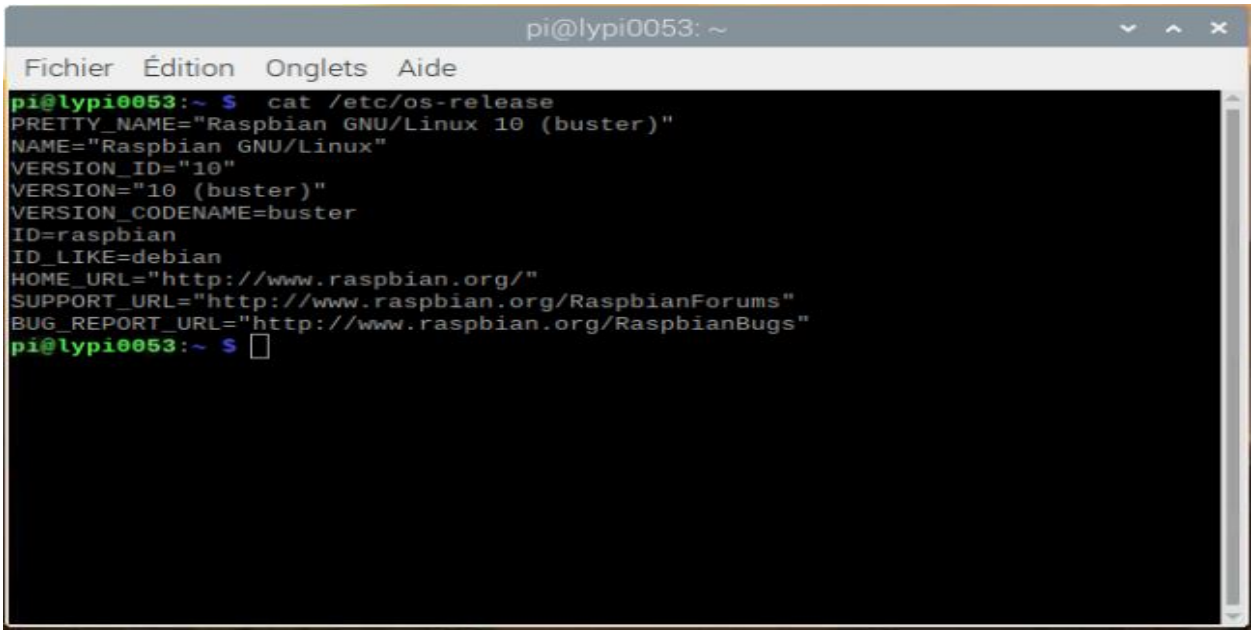
Raspberry Pi configuration

OS installation

The first step is to start up the Raspberry Pi board, including installation of an OS (operating system). For this step, the installation instructions are well described on the Raspberry website

1. Watch the video “how to set up your raspberry Pi” (https://www.youtube.com/watch?v=wjWZhV1v3Pk)
2. The authors recommend installing the latest stable and complete version of Raspbian by using NOOBS (a simple-to-use operating system installer).

Note: All the development tests were performed on Raspberry Pi 3 Model B, we used the following version of Raspbian:



Warning: Once the OS has been installed, **1-wire, spi and GPIO remote option** must be deactivated via the Raspbian GUI settings menu. Failure to carry out this task may cause damage to the relay shield cards during measurements.

3. When the relays are connected to the GPIO, make sure that all the GPIOs are in the low position when the raspberry starts up. If not, the relays will activate unexpectedly. To ensure that the GPIOs are in Low position, you will need to modify the /boot/config.txt file.

   Run the terminal, and write

```
cd /boot/
```

4. Open config.txt with GNU nano editor

```
sudo nano config.txt
```

5. At the end of the file write :

```
gpio=8=op,dl
gpio=7=op,dl
```

6. Press Ctrl +O to save the modifications and press enter
7. Press Ctrl +x to escape and return to the terminal
8. Close the terminal

Virtual Environment and packages

All dependencies are specified in requirements.txt

Note: All instructions below should be typed in the terminal

It is first necessary to ensure that the libatlas-base-dev library is installed:

```
sudo apt-get install libatlas-base-dev
```

We strongly recommend users to create a virtual environment to run the code and installed all required dependencies. It can be done either in a directory gathering all virtual environments used on the computer or within the ohmpy directory.

Create the virtual environment:

```
python3 -m venv ohmpy
```

Activate it using the following command:

```
source ohmpy/bin/activate
```

Install packages within the virtual environment. Installing the following package should be sufficient to meet dependencies:

```
pip install RPi.GPIO adafruit-blinka numpy adafruit-circuitpython-ads1x15 pandas
```

Check that requirements are met using

```
pip list
```

You should run you code within the virtual environment to leave the virtual environment simply type:

```
deactivate
```

Activate virtual environment on Thonny (Python IDE) (on Raspberry Pi)

If you decided to use a virtual environment, it is necessary to setup Thonny Python IDE the first time you use it.

1- Run the Thonny Python IDE software, Click on raspberry access **menu > programming> Thonny pythonIDE**

2- Thonny opens, Python runs on the root (Python 3.7.3 (/usr/bin/python3))

3-Click on **Run>select interpreter**, a new window opens click on interpret

4-On the new open windows select **alternative Python3 or virtual environment**

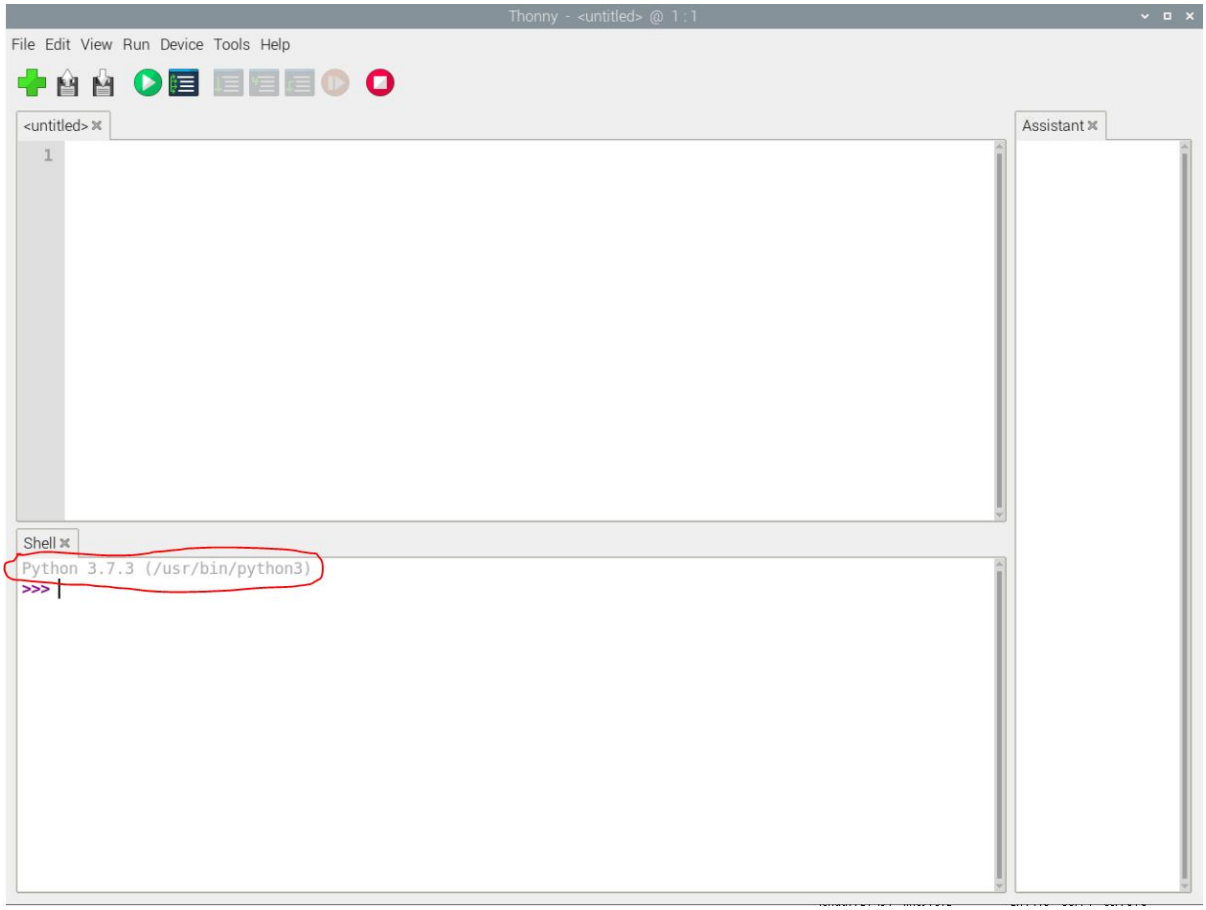
5- New buttons appeared, selected **"locate another python executable "**

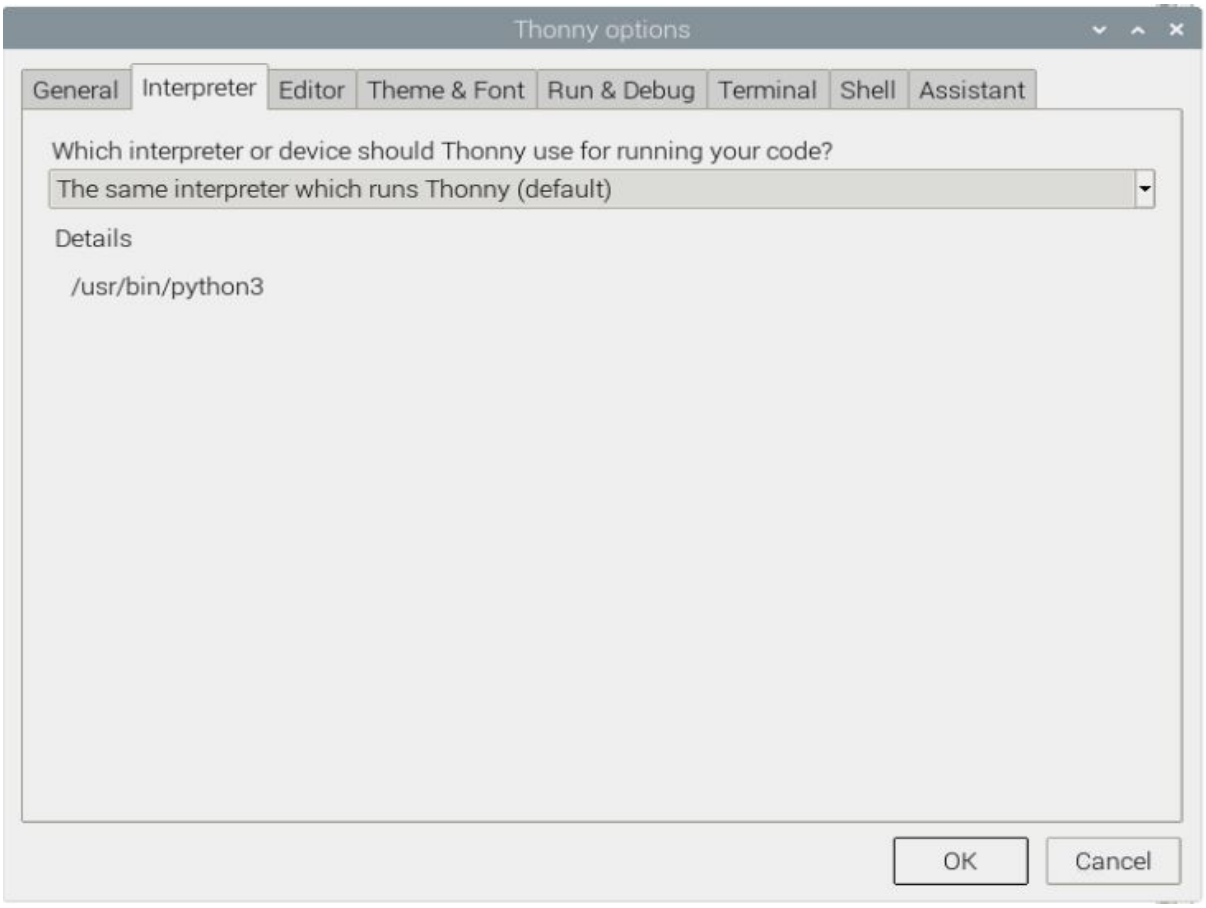
6- A new window opens, find the folder where there is the python 3 file in the virtual environment folder previously created **/home/pi/ohmpi/bin/python3**.

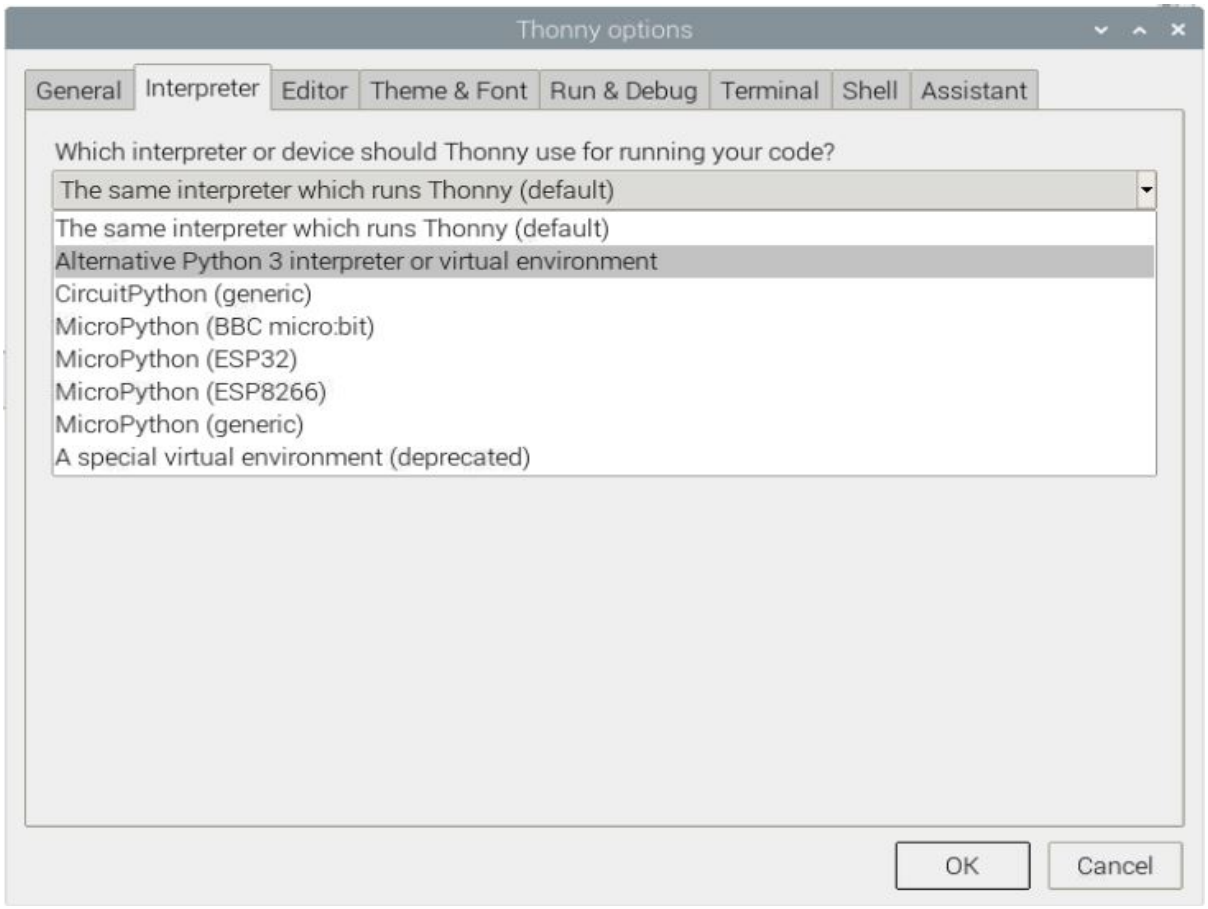
7- In the **known interpreter** tab the path of the virtual environment should appear

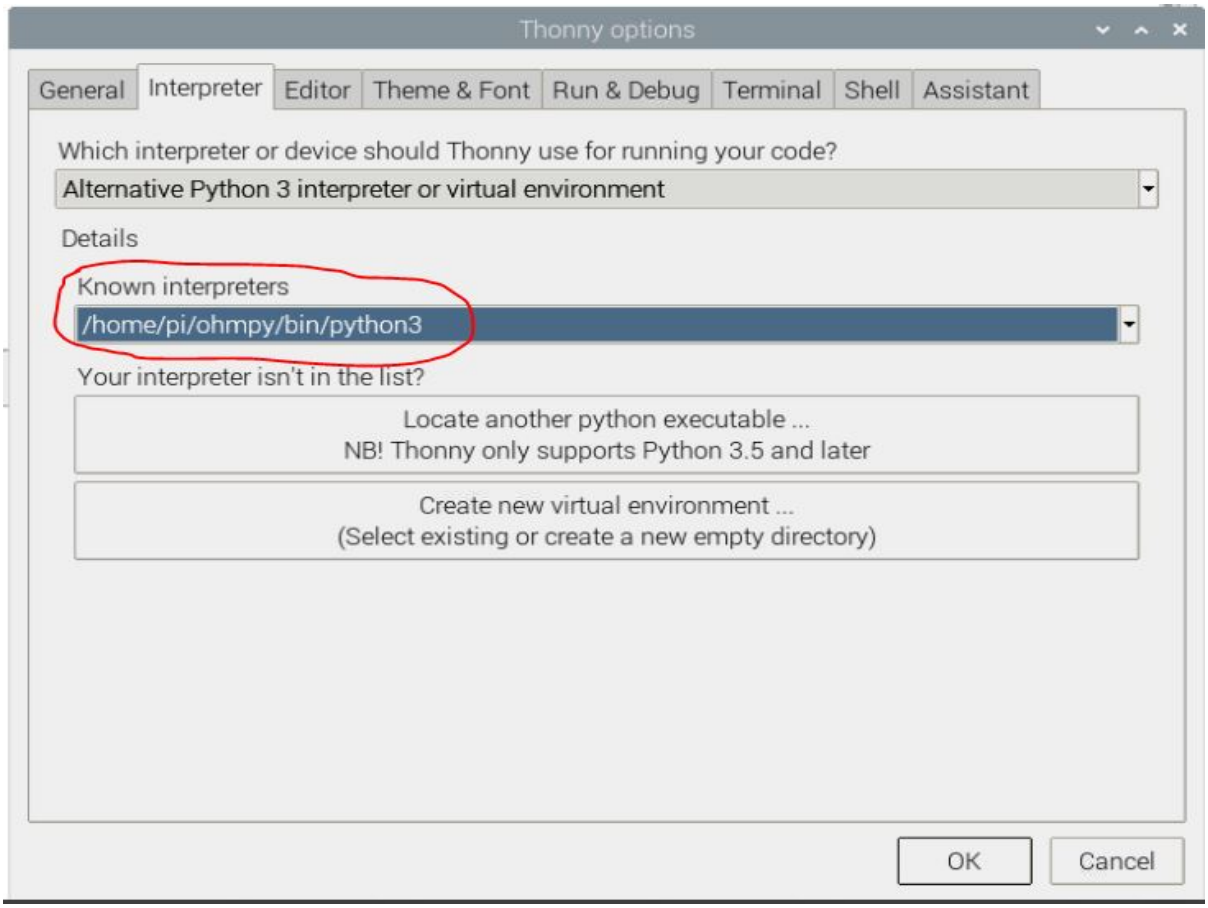
8- Close the window by clicking on **ok**.

9- Close thonny to save modifications


Thonny - <untitled> @ 1 : 1
File Edit View Run Device Tools Help
<untitled>
1
Assistant
Shell
Python 3.7.3 (/usr/bin/python3)
>>>


Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Which interpreter or device should Thonny use for running your code?
The same interpreter which runs Thonny (default)
Details
/usr/bin/python3
OK
Cancel


Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Which interpreter or device should Thonny use for running your code?
The same interpreter which runs Thonny (default)
The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment
CircuitPython (generic)
MicroPython (BBC micro:bit)
MicroPython (ESP32)
MicroPython (ESP8266)
MicroPython (generic)
A special virtual environment (deprecated)
OK
Cancel


Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Which interpreter or device should Thonny use for running your code?
Alternative Python 3 interpreter or virtual environment
Details
Known interpreters
/home/pi/ohmpy/bin/python3
Your interpreter isn't in the list?
Locate another python executable ...
NB! Thonny only supports Python 3.5 and later
Create new virtual environment ...
(Select existing or create a new empty directory)
OK
Cancel

Assembly of the measuring/current injection cards, and connection with the Raspberry Pi

Electrical resistivity measurements board

a) Description

To measure electrical resistivity with Raspberry Pi, an ADS1115 was introduced, as proposed by Florsch [7]. The ADS1115 is a 16-bit ADC (Analog-to-Digital Converter), with an adaptable gain. Its value has been set at 2/3 in this study. The input signal value could lie between - to + 6.114 V. The ADS1115 is mounted on a board adapted from an in-house design. Figure 5 shows the general diagram for the electronic measurement board developed. This figure also displays the test circuit used to test the board in the laboratory, which mimics the behavior of a soil subjected to current injection. In this test circuit, resistance R11 represents the soil resistance. Soil resistance R11 is connected to electrodes A and B for the current injection. Resistors R10 and R12 constitute the contact resistances between soil and electrodes; they are typically made of stainless steel. The battery, which allows for direct current injection, is connected in series with resistors R10, R11 and R12. In this part of the board, resistance R9 has been added to measure the current flowing between electrodes A and B. This resistance value has been set at 50 ohms in order to ensure: • a precise resistance, • a resistance less than the sum of resistors R10, R11 and R12; indeed, R10 and R12 generally lie between 100 and 5,000 ohms. To measure the current intensity between A and B, the electrical potential difference at the pole of the reference resistor (R9) is measured. The intensity (in mA) is calculated by inserting the resulting value into the following: ? To measure the potential difference needed to measure current intensity, the ADS 1115 is connected to the ground of the circuit. In our case, the ground reference is electrode B. The analog inputs A1 and A0 of the ADS1115 are connected to each pole of the reference resistor (R9). In order to increase input impedance and adapt the signal gain, tracking amplifiers have been included and completed by a divider bridge (R5, R8, R6 and R7) located between the two amplifiers. The resistance of the divider bridge ensures that the signal remains between 0 and 5 V, in accordance with the ADS1115 signal gain. To measure the potential difference, the M and N electrodes are connected to analog inputs A2 and A3 of the ADS 1115. Between the ADC and the electrodes, two tracking amplifiers and a divider bridge have been positioned so as to obtain a potential lying within the 0-5 V range at the analog input of the ADS 1115. Let's note that the potential difference value would equal the potential measured with ADS1115 multiplied by the voltage reduction value of the divider bridge (see Section 5.2). Despite the use of high-resolution resistance (i.e. accurate to within 1%), it is still necessary to calibrate the divider bridge using a precision voltmeter. For this purpose, the input and output potentials of the divider bridge must be measured using an equivalent circuit for various electrical potential values. These values serve to calculate the gain. With this electronic board, it is possible to measure the potential and intensity without disturbing the electric field in the ground, with the total input impedance value being estimated at 36 mega-ohms. A shortcut between Electrodes A and B will generate excessive currents, whose intensities depend on the type of battery used. A lithium ion battery or automobile-type lead-acid battery can deliver a strong enough current to damage the board and, as such, constitutes a potential hazard. We therefore recommend adding a 1.5-A fuse between the battery and resistor R9.

b) Implementation

The measurement board must be printed using the PCB file (Source file repository), with components soldered onto it by following the steps described below and illustrated in the following figure :

- **Step no. 1: test divider bridge**

  For each measurement channel, we have installed a bridge divider, it is necessary to test with ohmmeter the value of the resistances, to adjust each coefficients (coef_p0, coef_p1, coef_p2, coef_p3) in the Ohmpi.py code..

$$coefpo = (R1 + R2)/R1$$

$$coefp1 = (R3 + R4)/R3$$

$$coefp2 = (R7 + R6)/R7$$

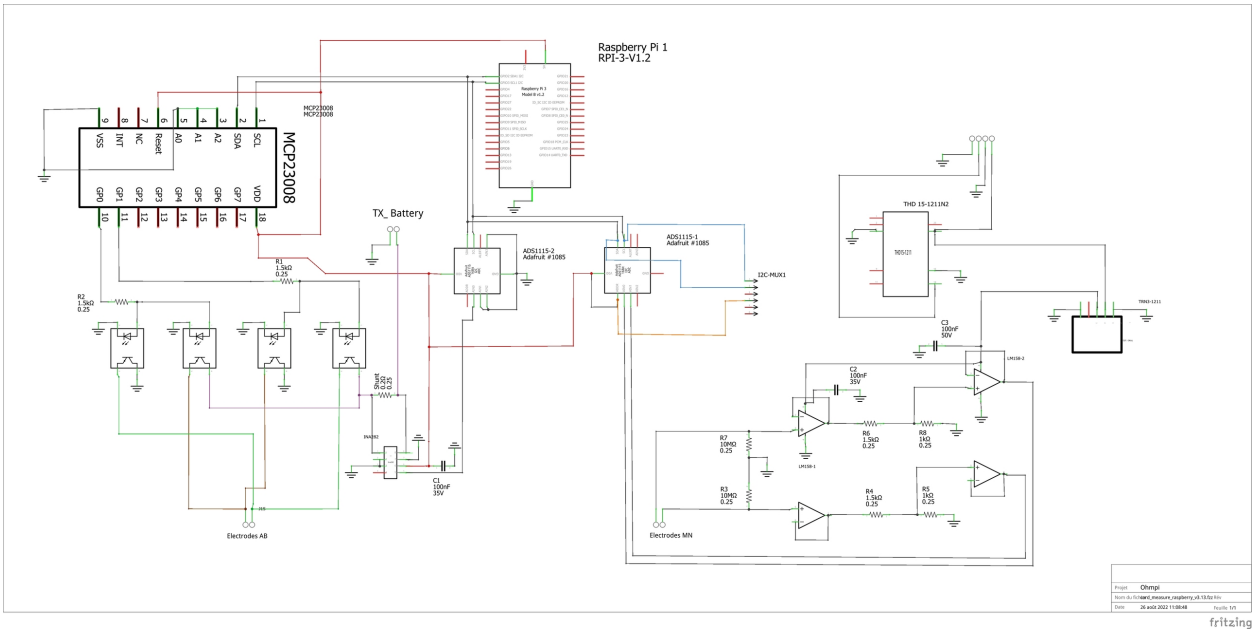


Fig. 15: Measurement board

$$coefp3 = (R9 + R8)/R9$$

```
36  """
37  hardware parameters
38  """
39  R_ref = 50 # reference resistance value in ohm
40  coef_p0 = 2.5 # slope for current conversion for ADS.P0, measurement in V/V
41  coef_p1 = 2.5 # slope for current conversion for ADS.P1, measurement in V/V
42  coef_p2 = 2.5 # slope for current conversion for ADS.P2, measurement in V/V
43  coef_p3 = 2.5 # slope for current conversion for ADS.P3, measurement in V/V
```

The coefficient parameters can be adjusted in lines 40 to 43 of the ohmpi.py code.

- Step no. 2: installation of the 1-KOhm resistors with an accuracy of ± 1%.
- Step no. 3: installation of the 1.5-KOhm resistors with an accuracy of ± 1%.
- Step no. 4: installation of both the black female 1 x 10 header and the 7-blue screw terminal blocks
- Step no. 5: installation of the 50-Ohm reference resistor ± 0.1%, please check the value and correct the line 39 in ohmpi.py code
- Step no. 6: addition of both the ADS115 directly onto the header (pins must be plugged according to the figure) and the LM358N operational amplifiers (pay attention to the direction).

1-KOhm and 1.5-KOhm resistors apply to the divider bridge. If, for example, you prefer using a weaker or stronger power supply, it would be possible to adjust the divider bridge value by simply modifying these resistors. Once all the components have been soldered together, the measurement board can be connected to the Raspberry Pi and the battery terminal, according to Figure 9. Between the battery and the TX+ terminal of the measurement board, remember to place a fuse holder with a 1.5-A fuse for safety purposes.

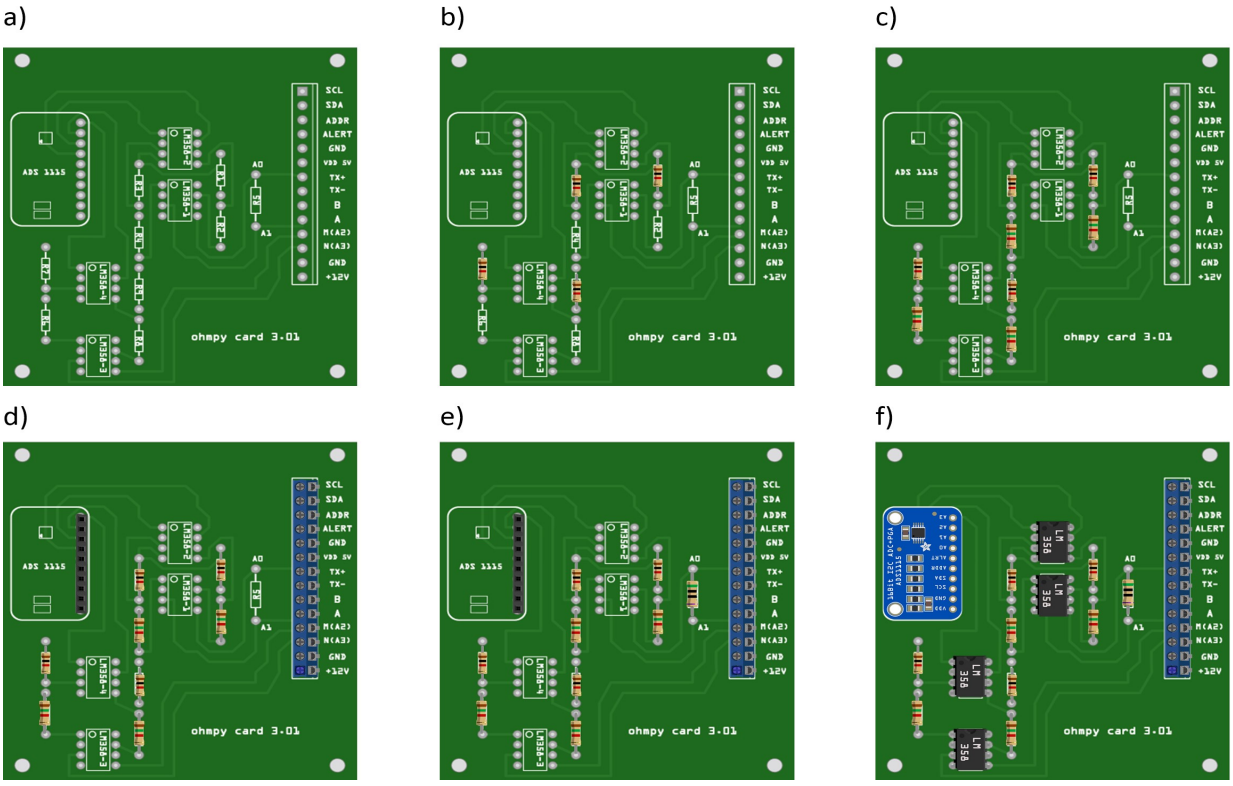


Fig. 16: Measurement circuit board assembly: a) printed circuit board, b) adding the 1-KOhm resistors ± 1%, c)adding the 1.5-KOhm resistors ± 1%, d) adding the black female 1 x 10 header and the 7-blue screw terminal block(2 pin, 3.5-mm pitch), e) adding the 50-ohm reference resistor ± 0.1%, and f) adding the ADS1115 and the LM358N low-power dual operational amplifiers

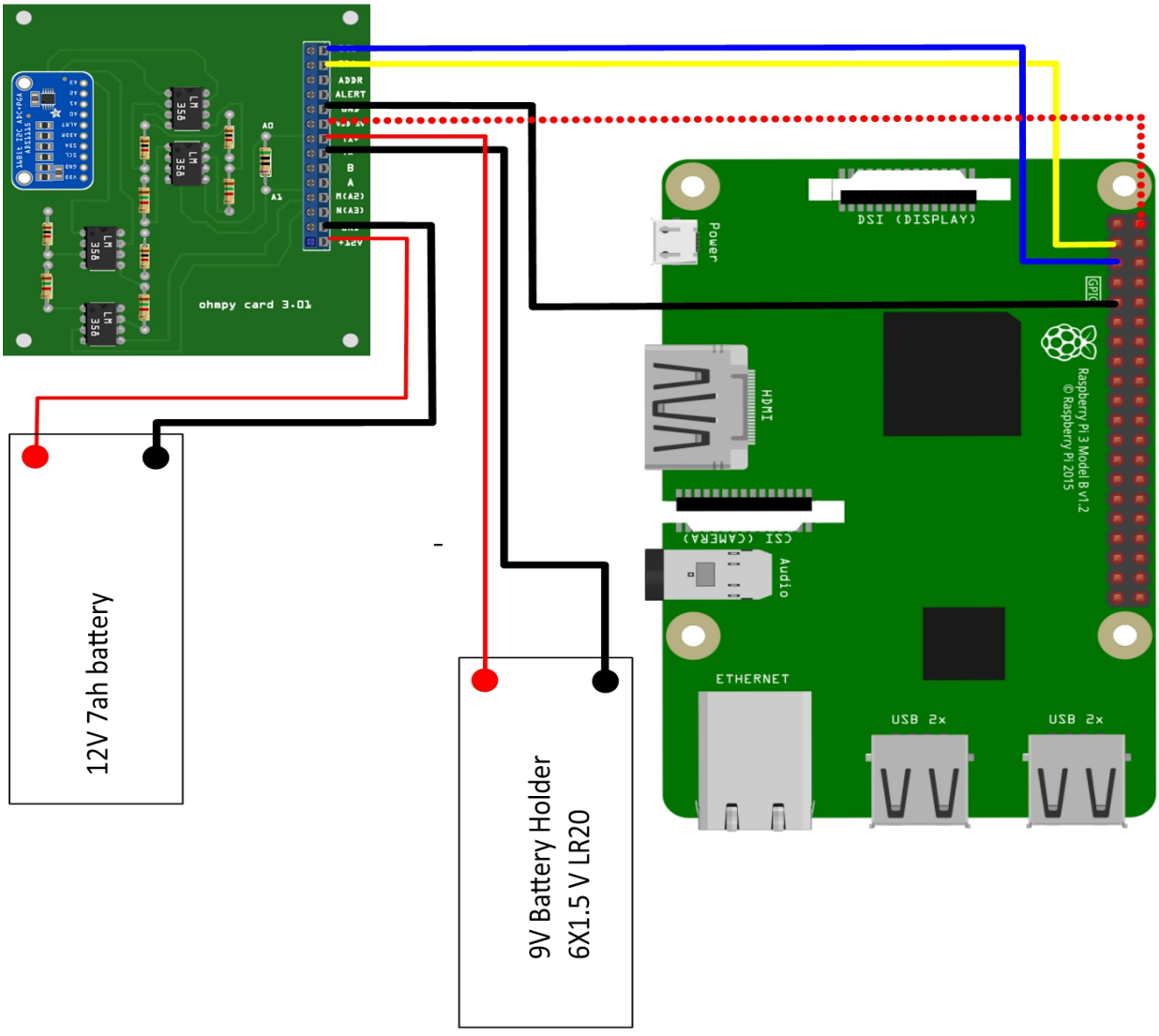


Fig. 17: Measurement board installation with Raspberry Pi

Current injection board

To carry out the electrical resistivity measurement, the first step consists of injecting current into the ground. In our case, a simple 9-V lead-acid battery is used to create an electrical potential difference that results in current circulating into the ground. The current is injected through electrodes A and B (see Fig. 2). This injection is controlled via a 4-channel relay module board connected to the Raspberry Pi. The mechanical relay module board is shown in Figure 4. Relays 1 and 2 serve to switch on the current source. The common contacts of relays 1 and 2 are connected to the positive and negative battery poles, respectively. The normally open contacts of both relays are connected to the common contacts of relays 3 and 4. Relays 1 and 2 are connected to the GPIO 7 on the Raspberry Pi and therefore activate simultaneously. The role of relays 3 and 4 is to reverse the polarity at electrodes A and B. Thus, when relays 3 and 4 are energized by the GPIO 8 in the open position, the positive battery pole is connected to electrode A and the negative pole to electrode B. When not energized, they remain in the normally closed position. This set-up offers a simple and robust solution to inject current.

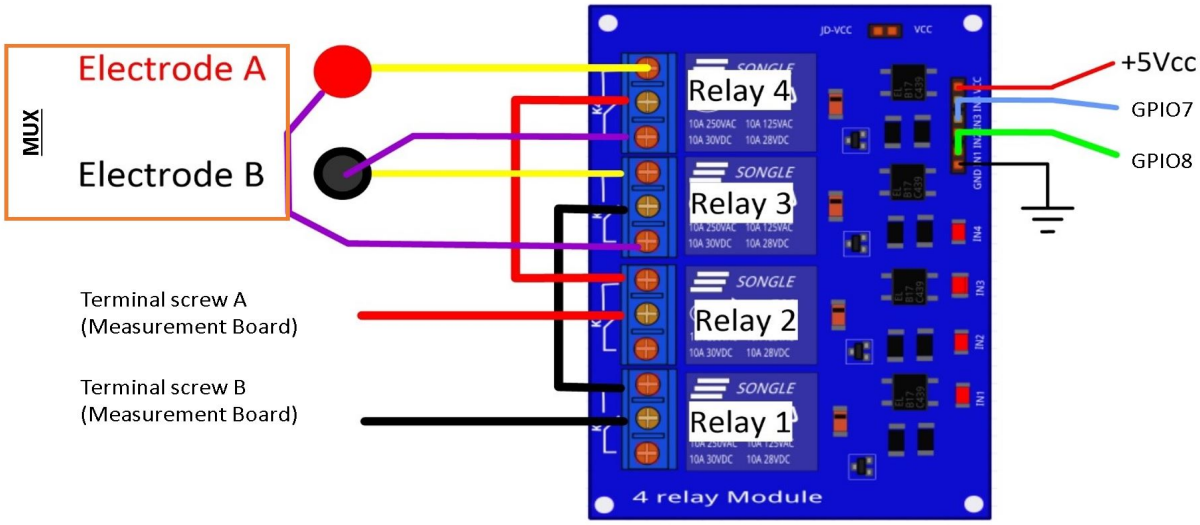


Fig. 18: Wiring of the 4-channel relay module board for current injection management

The next step consists of featuring the 4-channel relay module used for current injection and its assembly. The wiring between the relays must be carried out in strict accordance with Fig. 10. This card must then be connected to the Raspberry Pi and the measurement card. On the Raspberry Pi, it is necessary to connect inputs In1 and In2 to the same GPIO. For this purpose, it is necessary to solder together the two pins on the 4-channel relay shield module and connect them to the Raspberry Pi GPIO-7 (Fig. 10). The same must be performed for inputs In3 and In4 with GPIO-8. Connect the GND and 5Vdc pins of the relay card's 4 channels respectively to the GND pin and 5Vcc of the Raspberry Pi. Now connect relays 1, 2, 3 and 4, as shown in the diagram, using 1-mm2 cables (red and black in Fig. 10). Lastly, connect the inputs of relay 1 and 2 respectively to terminals B and A of the measurement board.

Congratulations, you have build a 4 electrodes resistivity-meter.

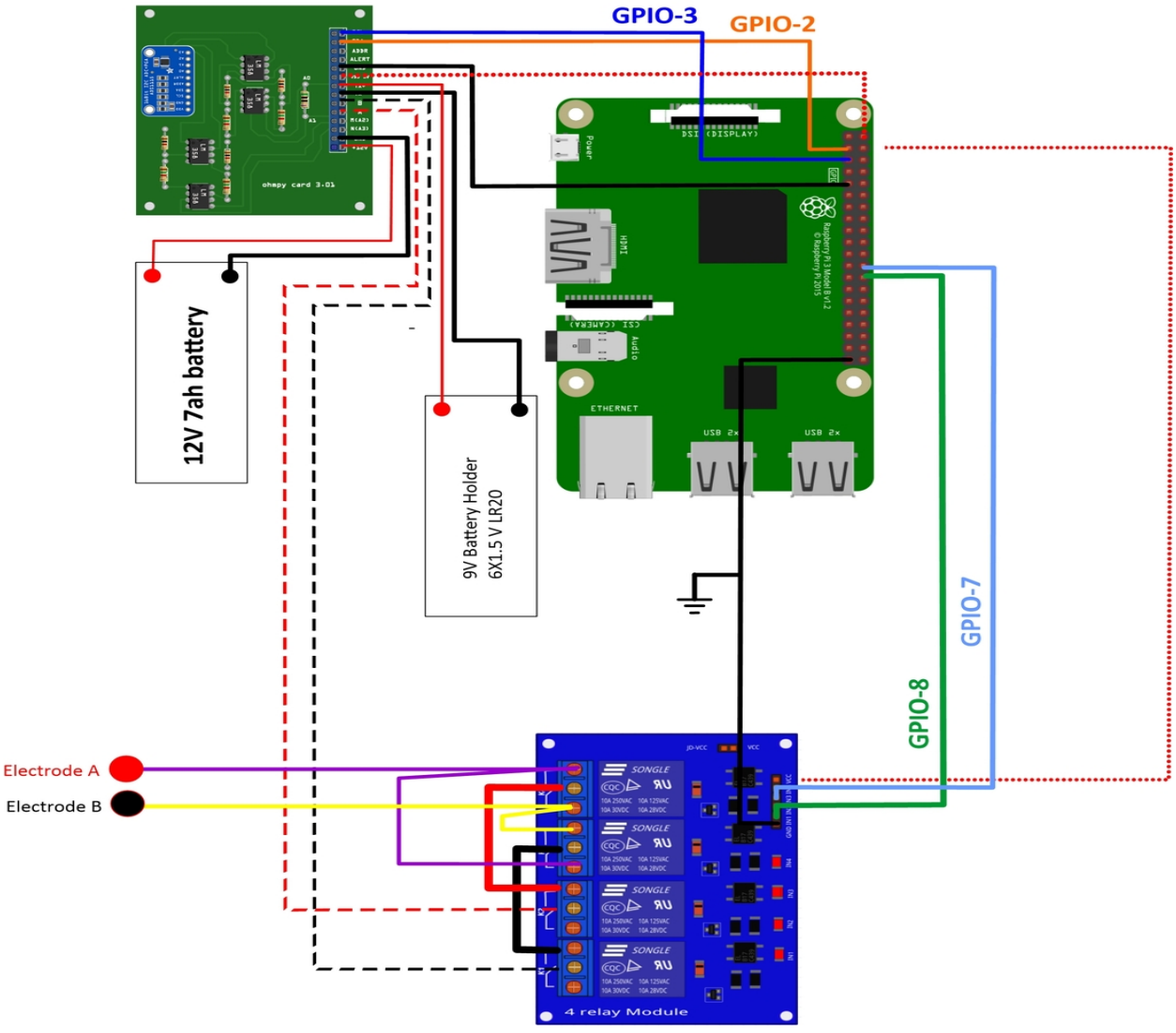


Fig. 19: Current injection board installation with Raspberry Pi

First four electrodes resistivity measurement

Under construction !

Describe the way to validate the first part of the instruction. Electrical resistivity measurement on test circuit

Multiplexer implementation

The resistivity measurement is conducted on four terminals (A, B, M and N). The user could perform each measurement by manually plugging four electrodes into the four channel terminals. In practice, ERT requires several tens or thousands of measurements conducted on different electrode arrays. A multiplexer is therefore used to connect each channel to one of the 32 electrodes stuck into the ground, all of which are connected to the data logger.

We will describe below how to assemble the four multiplexers (MUX), one per terminal. A multiplexer consists of 2 relay modules with 16 channels each. On the first board, on each MUX, 15 relays out of the 16 available will be used. Please note that the suggested configuration enables making smaller multiplexers (8 or 16 electrodes only). On the other hand, if you prefer upping to 64 electrodes, which is entirely possible, a GPIO channel multiplier will have to be used. To prepare the multiplexer, the channels of the two relay boards must be connected according to the wiring diagram shown below.

For this purpose, 0.5-mm^2 cables with end caps are used and their length adjusted for each connection in order to produce a clean assembly. The length was adjusted so that the distance between the two points to be connected could be directly measured on the board once they had been assembled one above the other, in adding an extra 3 cm. The wires at the ends need to be stripped and the end caps added. As a final step, connect the cables to the correct connectors. This operation must be repeated in order to carry out all the wiring shown in Figure below.

Once the operation has been completed, the 16 control pins of each 16-channel relay shield card must be prepared. Each card actually contains 16 input channels for activating each relay (Fig. 12). However, we will be activating several relays with a single GPIO (to limit the number of GPIOs used on Raspberry Pi, see Section 2.4). To execute this step, it will be necessary to follow the protocol presented in Figure.

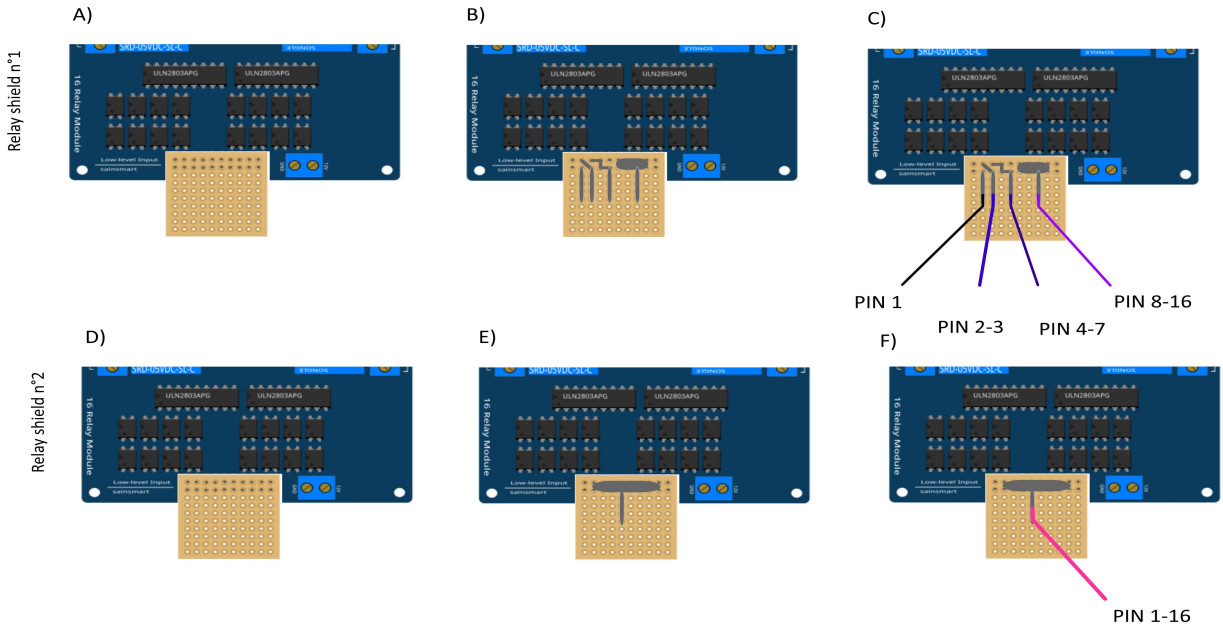


Fig. 20: Connection to the 16-channel relay shield

For the 16-channel relay shield no. 1, these steps must be followed: * Position a test circuit with 10 horizontal and 10 vertical holes on the pins of the 16-channel relay shield board. * Follow the diagram and solder the pins as shown in Fig. * Lastly, solder 0.5-mm^2 wires 1 m in length to the test circuit.

For relay shield no. 2, follow the same procedure, but solder all the pins together (d-e-f). This same operation must be repeated for the other three multiplexers as well. The next step consists of connecting the relay card inputs to the Raspberry Pi according to Table 5 for all four multiplexers.

	Relay shield n°1				Relay Shield n°2
	Pin 1	Pin 2-3	Pin 4-7	Pin 8-16	Pin 1- 16
Multiplexer A	12	16	20	21	26
Multiplexer B	18	23	24	25	19
Multiplexer M	06	13	04	17	27
Multiplexer N	22	10	09	11	05

Connection of the GPIOs to each multiplexer

Electrode connection

At this point, all that remains is to connect the electrodes of each multiplexer to a terminal block (Fig. 13). In our set-up, screw terminals assembled on a din rail were used. According to the chosen multiplexer configuration, all the relays of each multiplexer will be connected to an electrode and, consequently, each electrode will have four incoming connections. Instead of having four cables connecting an electrode terminal to each multiplexer, we recommend using the cable assembly shown in the following Figure.

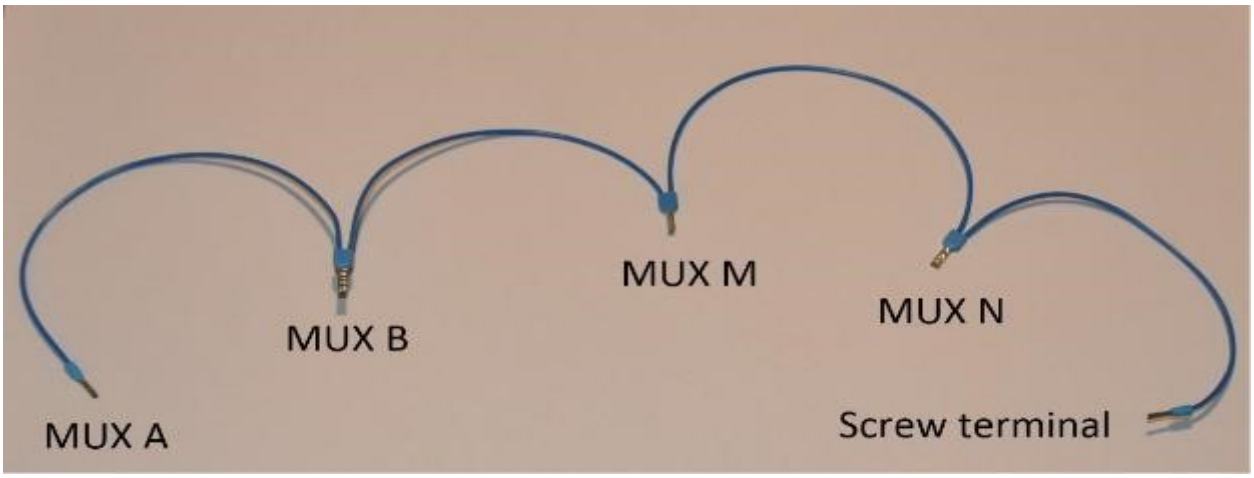


Fig. 21: Wire cabling for multiplexer and terminal screw connection

the next figure provides an example of multiplexer relay connections for electrode no. 1: this electrode of multiplexer MUX A must be connected to electrode no. 1 of MUX B. Moreover, electrode no. 1 of MUX B must be connected to electrode no. 1 of MUX N, which in turn must be connected to electrode no. 1 of MUX M. Lastly, electrode no. 1 of MUX M is connected to the terminal block. This operation must be repeated for all 32 electrodes.

Warning: The 16 channel relay cards exist in 5-V and 12-V , in the bottom figure we have 12-V cards that we will directly connect to the battery. In case you bought 16 channel relay 5-V cards, you will need to add a DC/DC 12-V/5-V converter. You can use a STEP DOWN MODULE DC-DC (Velleman WPM404) and set the voltage to 5V with the potentiometer.

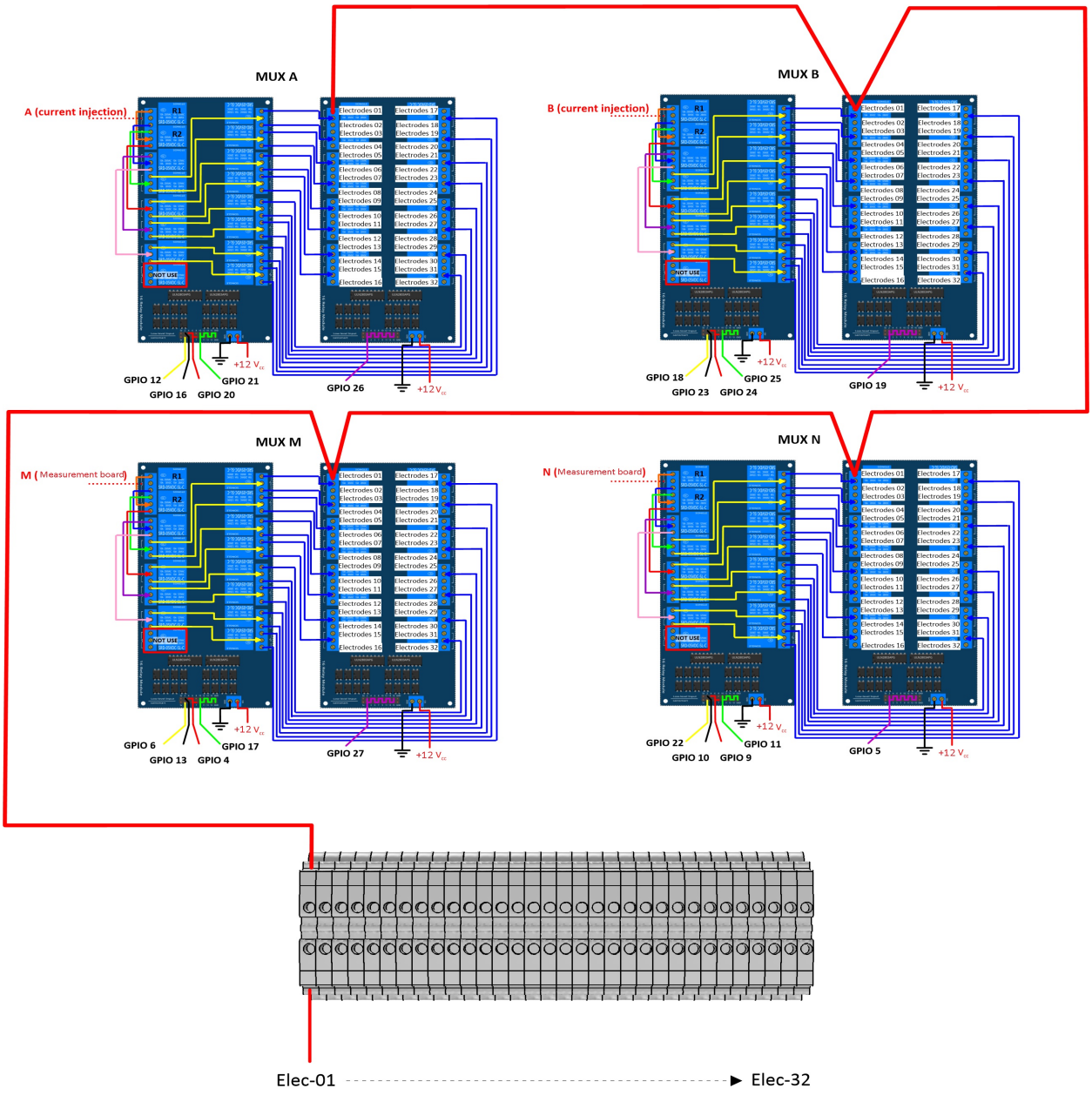


Fig. 22: Example of a multiplexer connection to the screw terminal for electrode no. 1.

Operating instruction

Preliminary procedure (Only for the initial operation)

The open source code must be downloaded at the Open Science Framework source file repository for this manuscript (https://osf.io/dzwb4/) or at the following Gitlab repository address: https://gitlab.irstea.fr/reversaal/OhmPi. The code must be then unzipped into a selected folder (e.g. OhmPi-master). A "readme" file is proposed in the directory to assist with installation of the software and required python packages. It is strongly recommended to create a python virtual environment for installing the required packages and running the code.

Startup procedure

As an initial operating instruction, all batteries must be disconnected before any hardware handling. Ensure that the battery is charged at full capacity. Plug all the electrodes (32 or fewer) into the screw terminals. The Raspberry Pi must be plugged into a computer screen, with a mouse and keyboard accessed remotely. The Raspberry Pi must then be plugged into the power supply (for laboratory measurements) or a power bank (5V - 2A for field measurements). At this point, you'll need to access the Raspbian operating system. Inside the previously created folder "ohmPi", the protocol file "ABMN.txt" must be created or modified; this file contains all quadrupole ABMN numeration (an example is proposed with the source code). Some input parameters of the main "ohmpi.py" function may be adjusted/optimized depending on the measurement attributes. For example, both the current injection duration and number of stacks can be adjusted. At this point, the9 V and 12-V battery can be plugged into the hardware; the "ohmpi.py" source code must be run within a python3 environment (or a virtual environment if one has been created) either in the terminal or using Thonny. You should now hear the characteristic sound of a relay switching as a result of electrode permutation. After each quadrupole measurement, the potential difference as well as the current intensity and resistance are displayed on the screen. A measurement file is automatically created and named "measure.csv"; it will be placed in the same folder.

Electrical resistivity measurement parameters description

```
27 """
28 measurement parameters
29 """
30 nb_electrodes = 32 # maximum number of electrodes on the resistivity meter
31 injection_duration = 0.5 # Current injection duration in second
32 nbr_meas= 1 # Number of times the quadrupole sequence is repeated
33 sequence_delay= 30 # Delay in seconds between 2 sequences
34 stack= 1 # repetition of the current injection for each quadrupole
```

The measurement parameters can be adjusted in lines 27 to 30 of the ohmpi.py code.

Complete list of components

Warning: The list evolve a little bit after the publication of the article, it is necessary to refer to this list, the article is out of date

Table 14: Table Title

Component	Number	Cost per unit	Total cost	Manufacturer	Manufacturer s reference
Raspberry Pi 3 Model B+	1	37	37	Raspberry	Raspberry Pi 3 Model B
Raspberry Pi 1 2 and 3 Power Supply	1	8.37	8.37	Raspberry	Raspberry Pi 1 2 and 3 Power Supply
SainSmart 16-Channe Canal 12V Relay Relais Module pour Arduino DSP AVR PIC ARM	8	24.99	199.92	Sain Smart	101-70-103
4-Channe 5V Relay Module	1	7.99	7.99	Sain Smart	20-018-101-CMS
cable 1X1 mm2 (50 m)	1	19.66	19.66	TRU COMPONENTS	1568649
cable 1X0.5 mm2 (100 m)	1	29.71	29.71	TRU COMPONENTS	1565235
Printed circuit board (packaging quantity x 3)	1	12	12	Asler	0
Header	1	2.68	2.68	Samtec	SSW-110-02-G-S

1.7.2 OhmPi V 1.02 (limited to 32 electrodes)

Warning: Ohmpi is a participative project open to all, it requires skills in electronics and to respect the safety rules. Ohmpi must be assembled in a professional context and by people competent in electronics. The Ohmpi team cannot be held responsible for any material or human damage which would be associated with the use or the assembly of OHMPI. The Ohmpi team cannot be held responsible if the equipment does not work after assembly.

Note: In this version, we have improved the electronic measurement board. To upgrade from version 1.01 to 1.02, you just have to replace the measurement board by the new one proposed here.

The philosophy of Ohmpi

The philosophy of Ohmpi V1.01 is to offer a multi electrode resistivity meter, from a set of commercially available electronic cards it is a resistivity meter limited to 32 electrodes only. It is limited to low-current injection, but suitable for small laboratory experiments and small field time monitoring

Technical data

Parameter	**Specifications**	Units
Electrodes	32	
Operating temperature	0 to 50	°C
Power consumption of CPU and control system	18.5	W
Voltage injection	9	V
Battery	12	V
Current	0 to 50	mA
Min pulse duration	150	ms
Input impedance	36	MOhm
Data storage	micro SD card	
Resolution	O.O1	Ohm

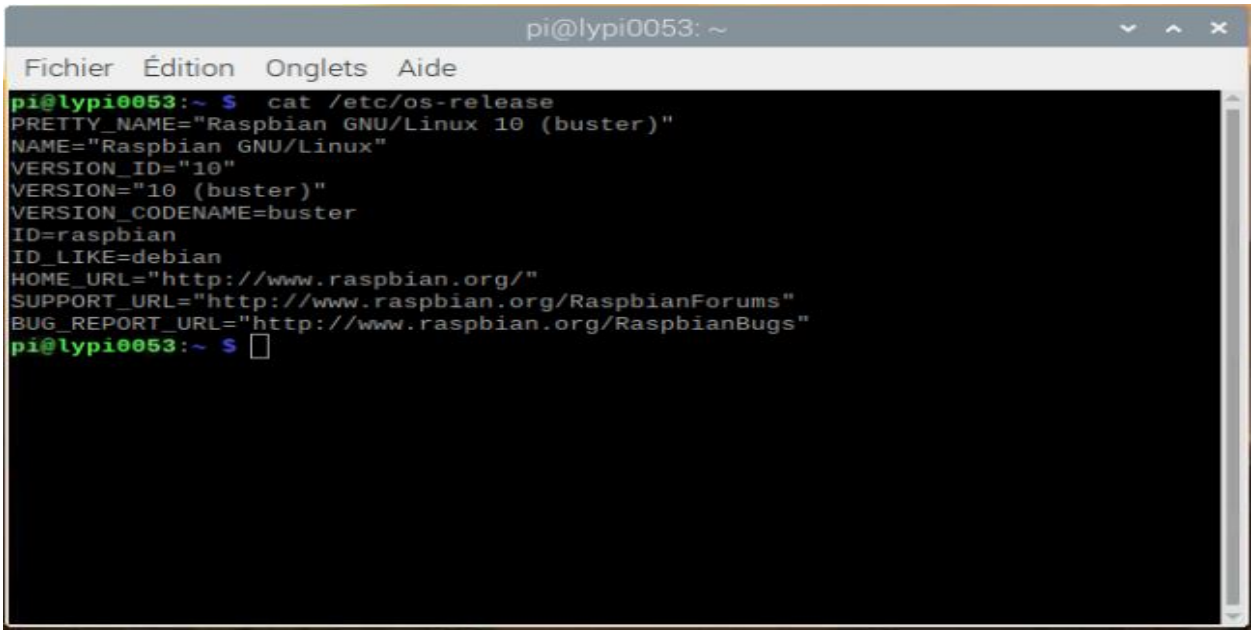
Raspberry Pi configuration

OS installation

The first step is to start up the Raspberry Pi board, including installation of an OS (operating system). For this step, the installation instructions are well described on the Raspberry website

1. Watch the video “how to set up your raspberry Pi” (https://www.youtube.com/watch?v=wjWZhV1v3Pk)
2. The authors recommend installing the latest stable and complete version of Raspbian by using NOOBS (a simple-to-use operating system installer).

Note: All the development tests were performed on Raspberry Pi 3 Model B, we used the following version of Raspbian:



Warning: Once the OS has been installed, **1-wire, spi and GPIO remote option** must be deactivated via the Raspbian GUI settings menu. Failure to carry out this task may cause damage to the relay shield cards during measurements.

3. When the relays are connected to the GPIO, make sure that all the GPIOs are in the low position when the raspberry starts up. If not, the relays will activate unexpectedly. To ensure that the GPIOs are in Low position, you will need to modify the /boot/config.txt file.

   Run the terminal, and write

```
cd /boot/
```

4. Open config.txt with GNU nano editor

```
sudo nano config.txt
```

5. At the end of the file write :

```
gpio=8=op,dl
gpio=7=op,dl
```

6. Press Ctrl +O to save the modifications and press enter
7. Press Ctrl +x to escape and return to the terminal
8. Close the terminal

Virtual Environment and packages

All dependencies are specified in requirements.txt

Note: All instructions below should be typed in the terminal

It is first necessary to ensure that the libatlas-base-dev library is installed:

```
sudo apt-get install libatlas-base-dev
```

We strongly recommend users to create a virtual environment to run the code and installed all required dependencies. It can be done either in a directory gathering all virtual environments used on the computer or within the ohmpy directory.

Create the virtual environment:

```
python3 -m venv ohmpy
```

Activate it using the following command:

```
source ohmpy/bin/activate
```

Install packages within the virtual environment. Installing the following package should be sufficient to meet dependencies:

```
pip install RPi.GPIO adafruit-blinka numpy adafruit-circuitpython-ads1x15 pandas
```

Check that requirements are met using

```
pip list
```

You should run you code within the virtual environment to leave the virtual environment simply type:

```
deactivate
```

Activate virtual environment on Thonny (Python IDE) (on Raspberry Pi)

If you decided to use a virtual environment, it is necessary to setup Thonny Python IDE the first time you use it.

1- Run the Thonny Python IDE software, Click on raspberry access **menu > programming> Thonny pythonIDE**

2- Thonny opens, Python runs on the root (Python 3.7.3 (/usr/bin/python3))

3-Click on **Run>select interpreter**, a new window opens click on interpret

4-On the new open windows select **alternative Python3 or virtual environment**

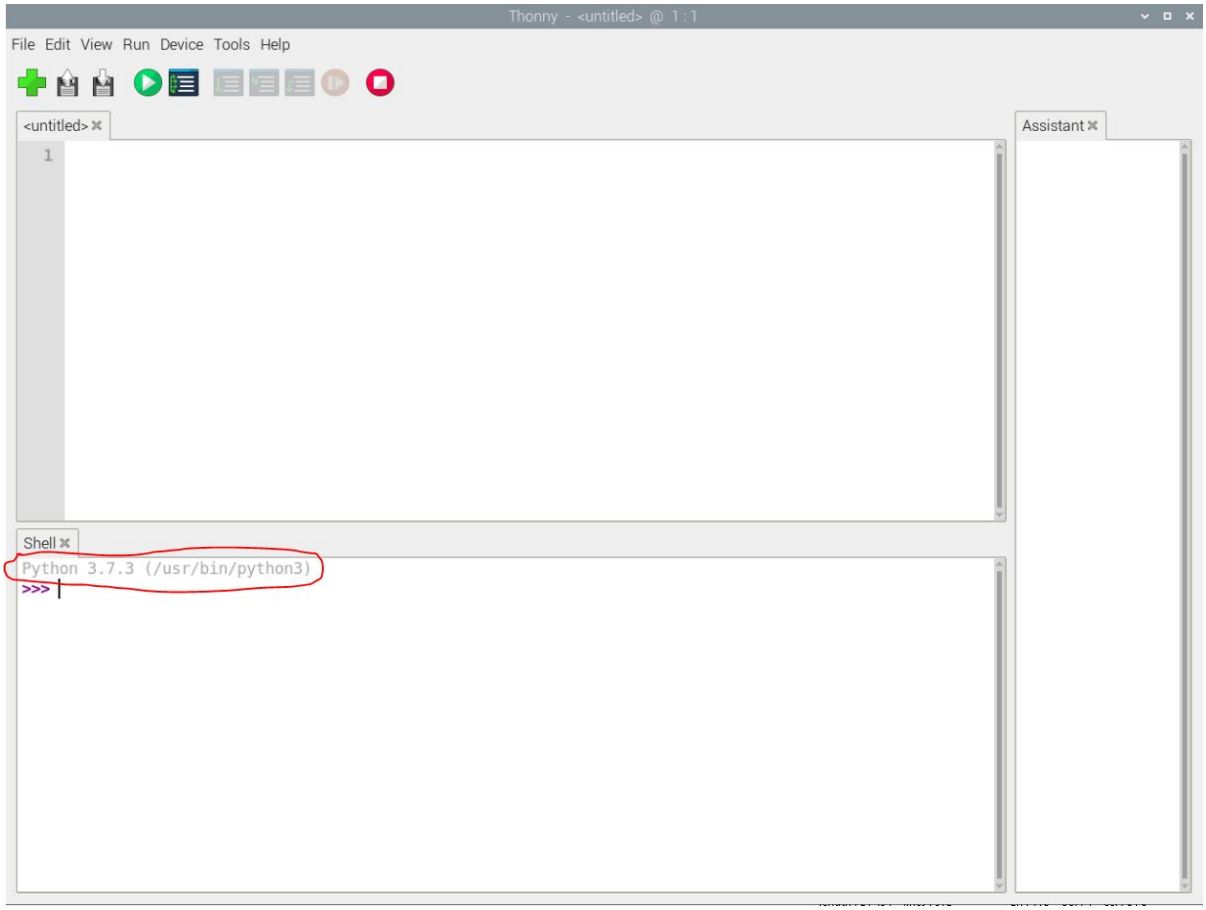
5- New buttons appeared, selected **"locate another python executable "**

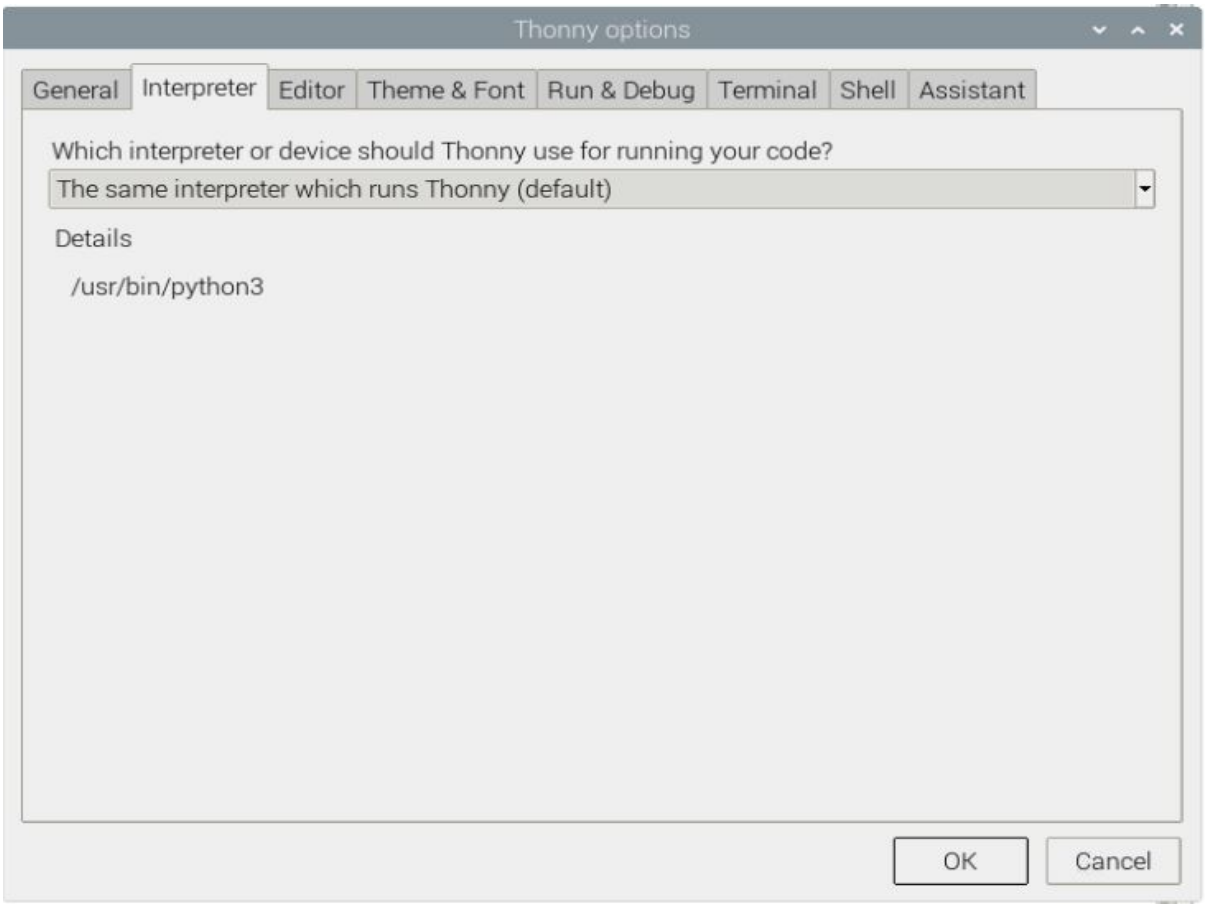
6- A new window opens, find the folder where there is the python 3 file in the virtual environment folder previously created **/home/pi/ohmpi/bin/python3**.

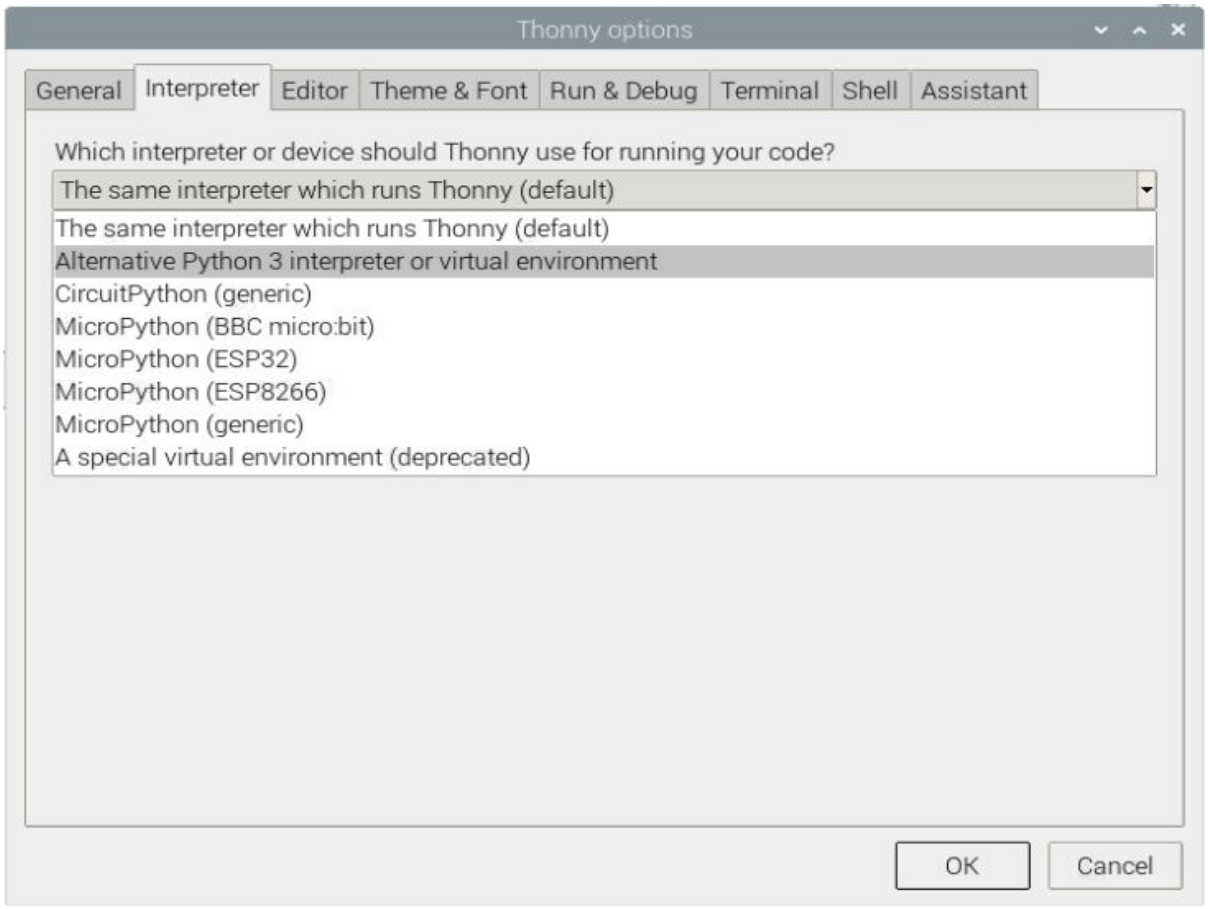
7- In the **known interpreter** tab the path of the virtual environment should appear

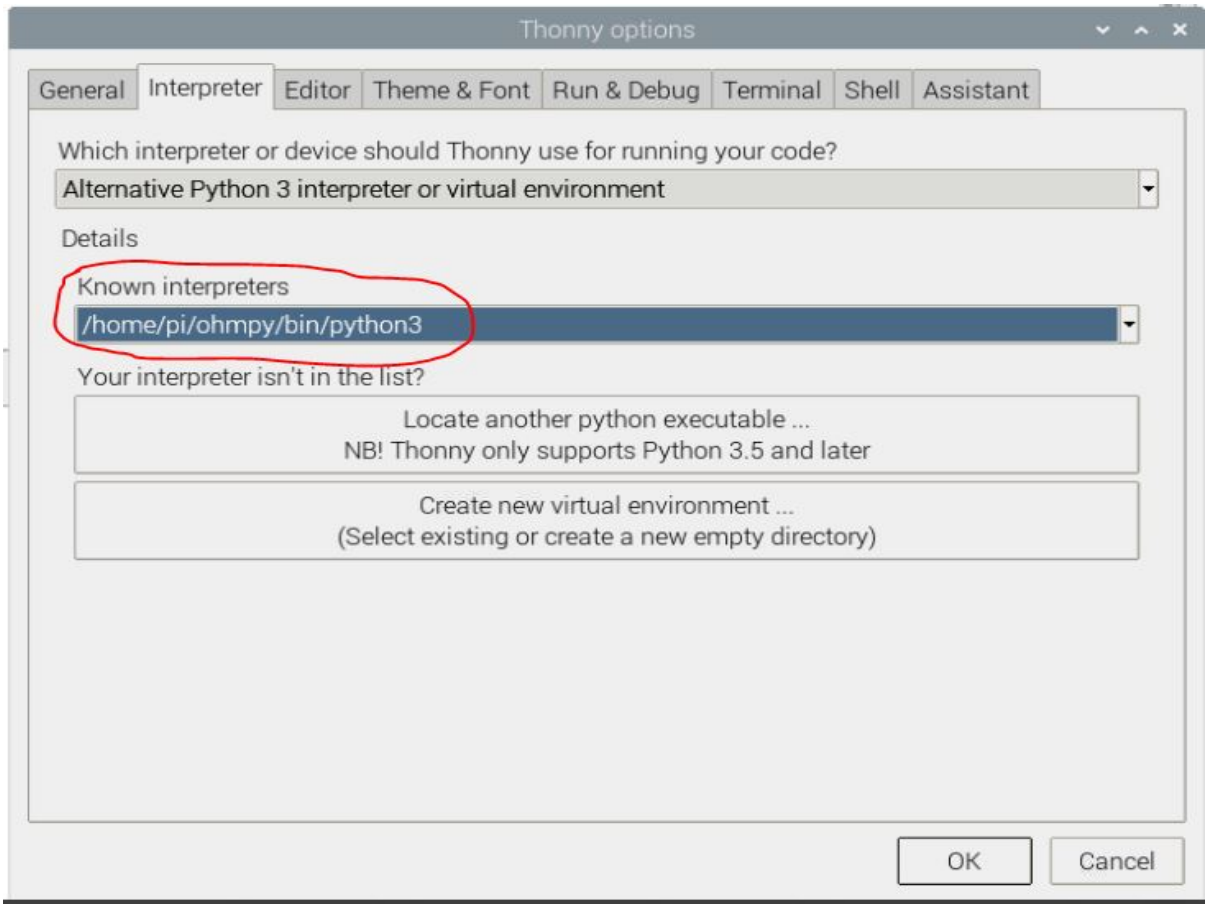
8- Close the window by clicking on **ok**.

9- Close thonny to save modifications


Thonny - <untitled> @ 1 : 1
File Edit View Run Device Tools Help
<untitled>
1
Assistant
Shell
Python 3.7.3 (/usr/bin/python3)
>>>


Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Which interpreter or device should Thonny use for running your code?
The same interpreter which runs Thonny (default)
Details
/usr/bin/python3
OK
Cancel


Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Which interpreter or device should Thonny use for running your code?
The same interpreter which runs Thonny (default)
The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment
CircuitPython (generic)
MicroPython (BBC micro:bit)
MicroPython (ESP32)
MicroPython (ESP8266)
MicroPython (generic)
A special virtual environment (deprecated)
OK
Cancel


Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Which interpreter or device should Thonny use for running your code?
Alternative Python 3 interpreter or virtual environment
Details
Known interpreters
/home/pi/ohmpy/bin/python3
Your interpreter isn't in the list?
Locate another python executable ...
NB! Thonny only supports Python 3.5 and later
Create new virtual environment ...
(Select existing or create a new empty directory)
OK
Cancel

Assembly of the measuring/current injection cards, and connection with the Raspberry Pi

Electrical resistivity measurements board

a) Description

To measure electrical resistivity with Raspberry Pi, an ADS1115 was introduced, as proposed by Florsch [7]. The ADS1115 is a 16-bit ADC (Analog-to-Digital Converter), with an adaptable gain. Its value has been set at 2/3 in this study. The input signal value could lie between - to + 6.114 V. The ADS1115 is mounted on a board adapted from an in-house design. Figure 5 shows the general diagram for the electronic measurement board developed. This figure also displays the test circuit used to test the board in the laboratory, which mimics the behavior of a soil subjected to current injection. In this test circuit, resistance R11 represents the soil resistance. Soil resistance R11 is connected to electrodes A and B for the current injection. Resistors R10 and R12 constitute the contact resistances between soil and electrodes; they are typically made of stainless steel. The battery, which allows for direct current injection, is connected in series with resistors R10, R11 and R12. In this part of the board, resistance R9 has been added to measure the current flowing between electrodes A and B. This resistance value has been set at 50 ohms in order to ensure: • a precise resistance, • a resistance less than the sum of resistors R10, R11 and R12; indeed, R10 and R12 generally lie between 100 and 5,000 ohms. To measure the current intensity between A and B, the electrical potential difference at the pole of the reference resistor (R9) is measured. The intensity (in mA) is calculated by inserting the resulting value into the following: ? To measure the potential difference needed to measure current intensity, the ADS 1115 is connected to the ground of the circuit. In our case, the ground reference is electrode B. The analog inputs A1 and A0 of the ADS1115 are connected to each pole of the reference resistor (R9). In order to increase input impedance and adapt the signal gain, tracking amplifiers have been included and completed by a divider bridge (R5, R8, R6 and R7) located between the two amplifiers. The resistance of the divider bridge ensures that the signal remains between 0 and 5 V, in accordance with the ADS1115 signal gain. To measure the potential difference, the M and N electrodes are connected to analog inputs A2 and A3 of the ADS 1115. Between the ADC and the electrodes, two tracking amplifiers and a divider bridge have been positioned so as to obtain a potential lying within the 0-5 V range at the analog input of the ADS 1115. Let's note that the potential difference value would equal the potential measured with ADS1115 multiplied by the voltage reduction value of the divider bridge (see Section 5.2). Despite the use of high-resolution resistance (i.e. accurate to within 1%), it is still necessary to calibrate the divider bridge using a precision voltmeter. For this purpose, the input and output potentials of the divider bridge must be measured using an equivalent circuit for various electrical potential values. These values serve to calculate the gain. With this electronic board, it is possible to measure the potential and intensity without disturbing the electric field in the ground, with the total input impedance value being estimated at 36 mega-ohms. A shortcut between Electrodes A and B will generate excessive currents, whose intensities depend on the type of battery used. A lithium ion battery or automobile-type lead-acid battery can deliver a strong enough current to damage the board and, as such, constitutes a potential hazard. We therefore recommend adding a 1.5-A fuse between the battery and resistor R9. In version 1.02, we have improved the electronic board of measurement. we have added a DC/DC converter to supply the operational amplifiers (2 Traco power DC/DC converter TRN3-1215). These converters allow to limit the suppression of the signal when the injected voltage is higher than 10V. We also added 4 capacitors on the +12v inputs of the fast operational amplifiers. These are decoupling capacitors (typically 100nF ceramic) between each power supply terminal and ground. The last point, we have added a four very high resistances of 10 MOhm, between the ground and the signal input on the operational amplifiers. This prevents the operational amplifiers from overheating.

Note: If you want to have very accurate measurements you can replace the resistors with a tolerance of 1% by resistors with a tolerance of 0.01% which will improve the measurement, but the cost will be higher.

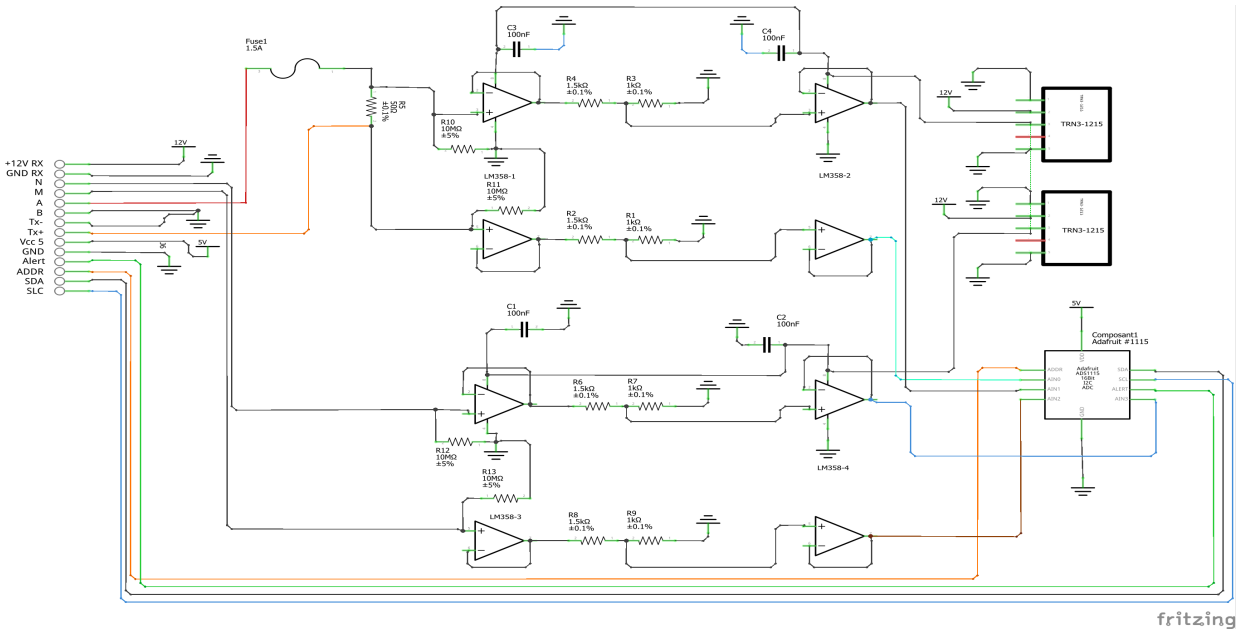


Fig. 23: Measurement board (Ohmpi version 1.02)

b) Implementation

The measurement board must be printed using the PCB file (Source file repository), with components soldered onto it by following the steps described below and illustrated in the following figure :

- **Step no. 1: test divider bridge**

  For each measurement channel, we have installed a bridge divider, it is necessary to test with ohmmeter the value of the resistances, to adjust each coefficients (coef_p0, coef_p1, coef_p2, coef_p3) in the Ohmpi.py code..

$$coefpo = (R1 + R2)/R1$$

$$coefp1 = (R3 + R4)/R3$$

$$coefp2 = (R7 + R6)/R7$$

$$coefp3 = (R9 + R8)/R9$$

```
36 """
37 hardware parameters
38 """
39 R_ref = 50 # reference resistance value in ohm
40 coef_p0 = 2.5 # slope for current conversion for ADS.P0, measurement in V/V
41 coef_p1 = 2.5 # slope for current conversion for ADS.P1, measurement in V/V
42 coef_p2 = 2.5 # slope for current conversion for ADS.P2, measurement in V/V
43 coef_p3 = 2.5 # slope for current conversion for ADS.P3, measurement in V/V
```

  The coefficient parameters can be adjusted in lines 40 to 43 of the ohmpi.py code.

- Step no. 2: installation of the 1-KOhm resistors with an accuracy of ± 1% (b-in the figure).
- Step no. 3: installation of the 1.5-KOhm resistors with an accuracy of ± 1%(C-in the figure).

- Step no. 4: installation of both the black female 1 x 10 header and the 7-blue screw terminal blocks (c-in the figure)
- Step no. 5: installation of the 50-Ohm reference resistor ± 0.1%, please check the value and correct the line 39 in ohmpi.py code (d-in the figure)
- Step no. 6: addition of both the ADS115 directly onto the header (pins must be plugged according to the figure) and the LM358N operational amplifiers (pay attention to the orientation) (e-in the figure).
- Step no. 7: installation of the 10-MOhm resistors with an accuracy of ± 5% (f-in the figure).
- Step no. 8: installation of the two DC/DC converter TRN3-1215 (h-in the figure).
- Step no. 9: installation of the four capacitor on 100-nF/50vDC and the fuse of 10-A (h-in the figure).

1-KOhm and 1.5-KOhm resistors apply to the divider bridge. If, for example, you prefer using a stronger power supply, it would be possible to adjust the divider bridge value by simply modifying these resistors. Once all the components have been soldered together, the measurement board can be connected to the Raspberry Pi and the battery terminal, according to Figure 9. Between the battery and the TX+ terminal of the measurement board, remember to place a fuse holder with a 1.5-A fuse for safety purposes.

Current injection board

To carry out the electrical resistivity measurement, the first step consists of injecting current into the ground. In our case, a simple 9-V lead-acid battery is used to create an electrical potential difference that results in current circulating into the ground. The current is injected through electrodes A and B (see Fig. 2). This injection is controlled via a 4-channel relay module board connected to the Raspberry Pi. The mechanical relay module board is shown in Figure 4. Relays 1 and 2 serve to switch on the current source. The common contacts of relays 1 and 2 are connected to the positive and negative battery poles, respectively. The normally open contacts of both relays are connected to the common contacts of relays 3 and 4. Relays 1 and 2 are connected to the GPIO 7 on the Raspberry Pi and therefore activate simultaneously. The role of relays 3 and 4 is to reverse the polarity at electrodes A and B. Thus, when relays 3 and 4 are energized by the GPIO 8 in the open position, the positive battery pole is connected to electrode A and the negative pole to electrode B. When not energized, they remain in the normally closed position. This set-up offers a simple and robust solution to inject current.

The next step consists of featuring the 4-channel relay module used for current injection and its assembly. The wiring between the relays must be carried out in strict accordance with Fig. 10. This card must then be connected to the Raspberry Pi and the measurement card. On the Raspberry Pi, it is necessary to connect inputs In1 and In2 to the same GPIO. For this purpose, it is necessary to solder together the two pins on the 4-channel relay shield module and connect them to the Raspberry Pi GPIO-7 (Fig. 10). The same must be performed for inputs In3 and In4 with GPIO-8. Connect the GND and 5Vdc pins of the relay card's 4 channels respectively to the GND pin and 5Vcc of the Raspberry Pi. Now connect relays 1, 2, 3 and 4, as shown in the diagram, using 1-mm2 cables (red and black in Fig. 10). Lastly, connect the inputs of relay 1 and 2 respectively to terminals B and A of the measurement board.

Congratulations, you have build a 4 electrodes resistivity-meter.

First four electrodes resistivity measurement

Under construction !

Describe the way to validate the first part of the instruction. Electrical resistivity measurement on test circuit

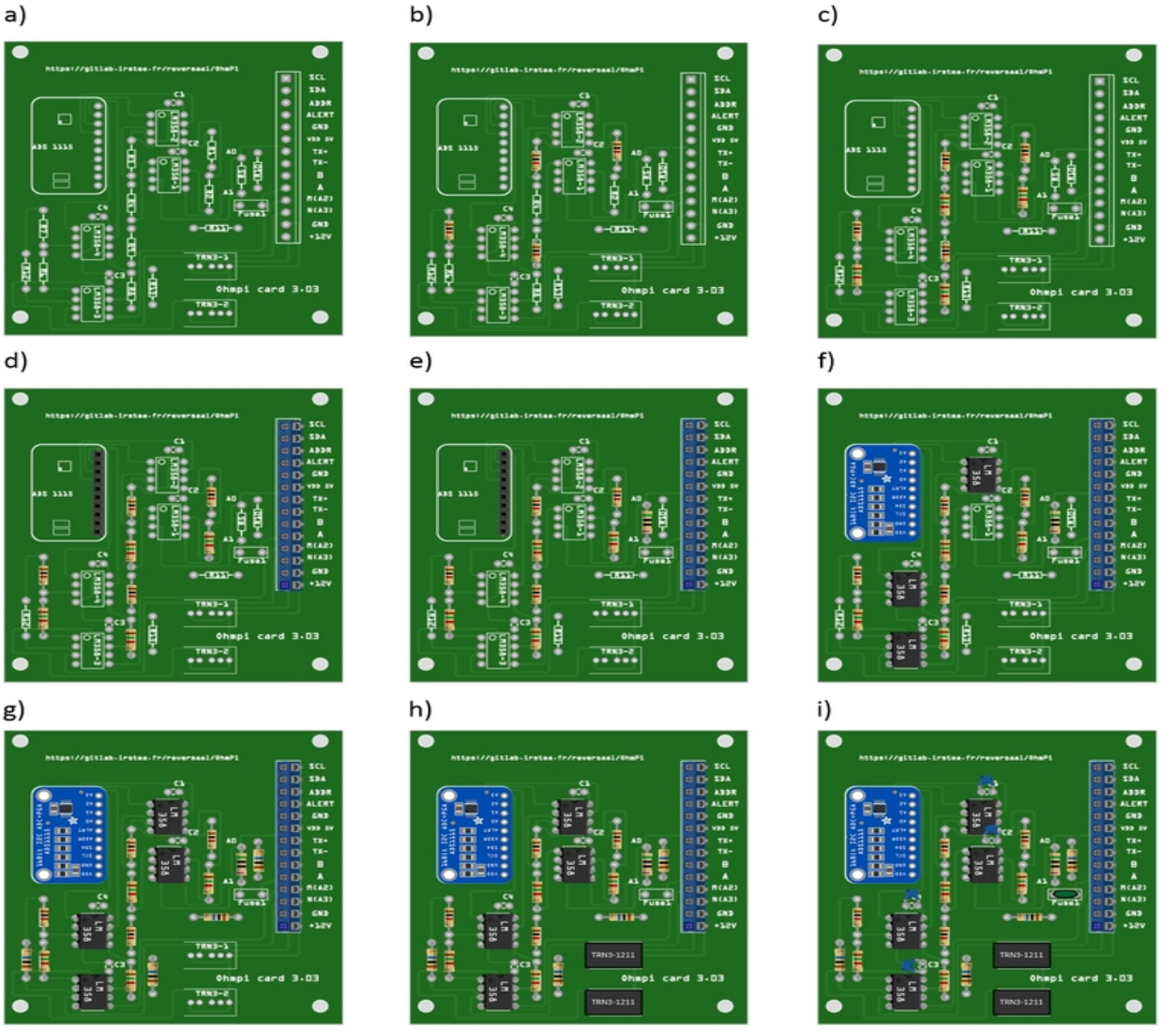


Fig. 24: Measurement circuit board assembly: a) printed circuit board, b) adding the 1-KOhm resistors ± 1%, c)adding the 1.5-KOhm resistors ± 1%, d) adding the black female 1 x 10 header and the 7-blue screw terminal block(2 pin, 3.5-mm pitch), e) adding the 50-ohm reference resistor ± 0.1%, and f) adding the ADS1115 and the LM358N low-power dual operational amplifiers

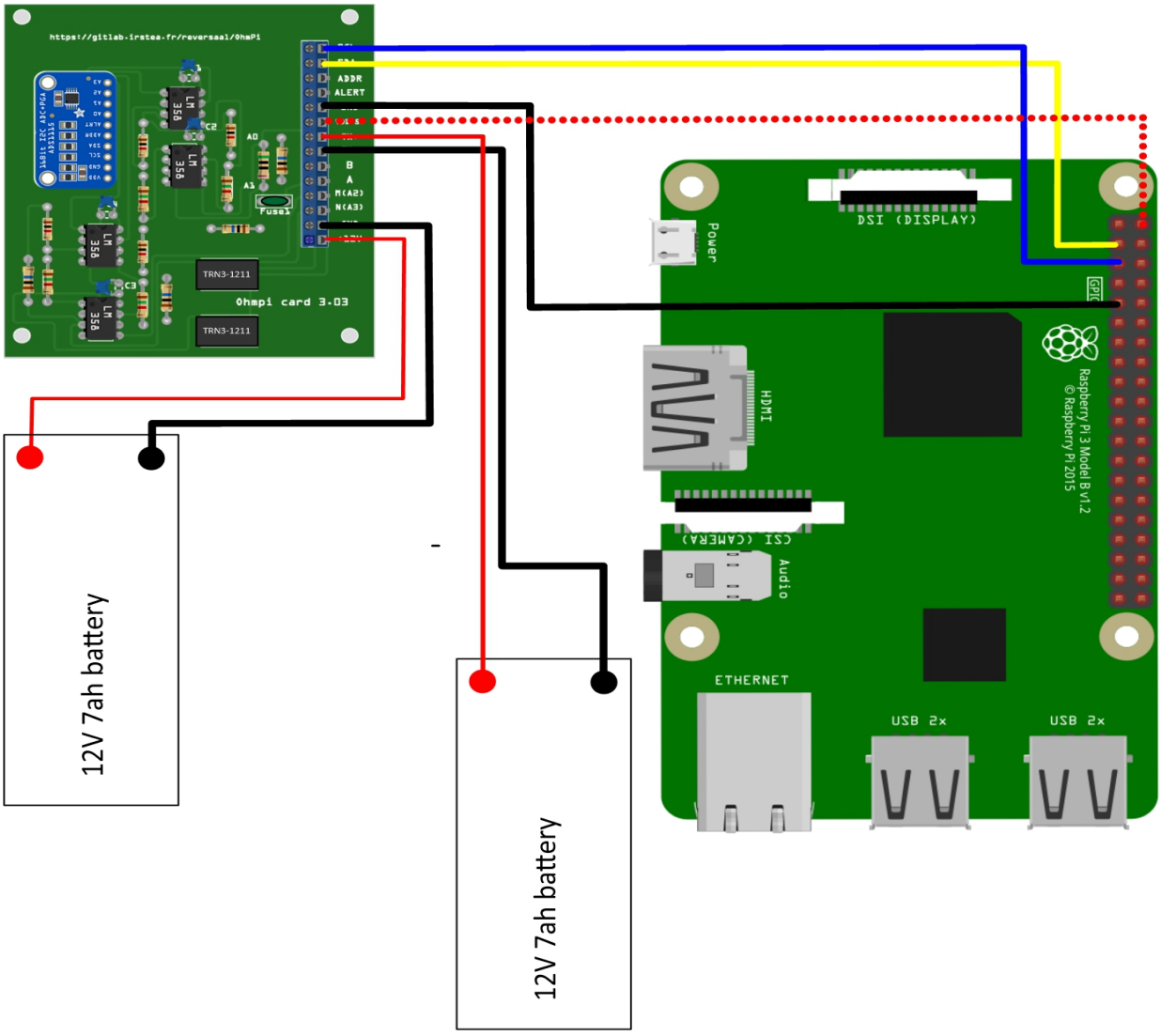


Fig. 25: Measurement board installation with Raspberry Pi

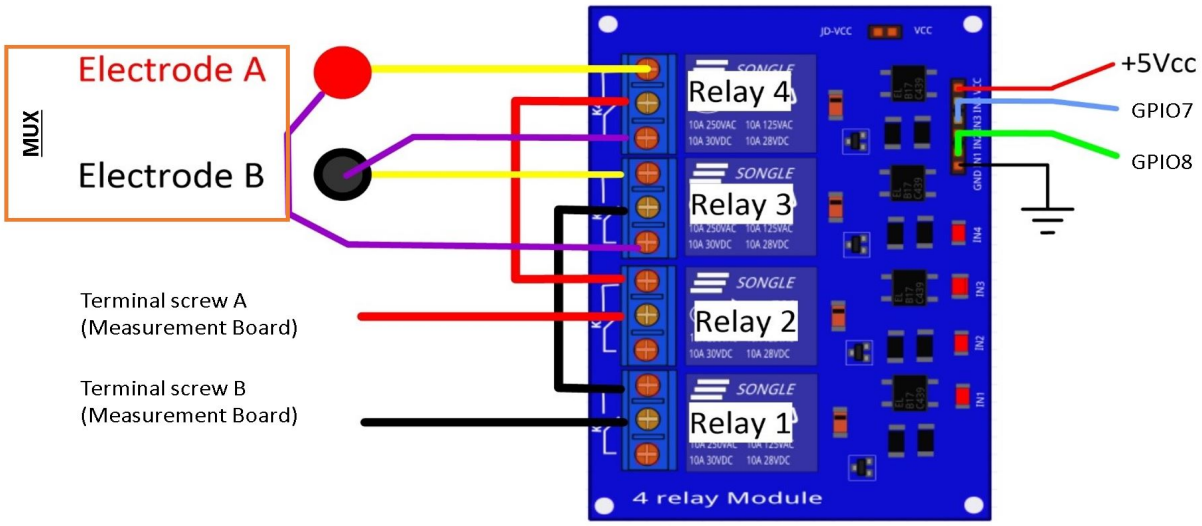


Fig. 26: Wiring of the 4-channel relay module board for current injection management

Multiplexer implementation

The resistivity measurement is conducted on four terminals (A, B, M and N). The user could perform each measurement by manually plugging four electrodes into the four channel terminals. In practice, ERT requires several tens or thousands of measurements conducted on different electrode arrays. A multiplexer is therefore used to connect each channel to one of the 32 electrodes stuck into the ground, all of which are connected to the data logger.

We will describe below how to assemble the four multiplexers (MUX), one per terminal. A multiplexer consists of 2 relay modules with 16 channels each. On the first board, on each MUX, 15 relays out of the 16 available will be used. Please note that the suggested configuration enables making smaller multiplexers (8 or 16 electrodes only). On the other hand, if you prefer upping to 64 electrodes, which is entirely possible, a GPIO channel multiplier will have to be used. To prepare the multiplexer, the channels of the two relay boards must be connected according to the wiring diagram shown below.

For this purpose, 0.5-mm^2 cables with end caps are used and their length adjusted for each connection in order to produce a clean assembly. The length was adjusted so that the distance between the two points to be connected could be directly measured on the board once they had been assembled one above the other, in adding an extra 3 cm. The wires at the ends need to be stripped and the end caps added. As a final step, connect the cables to the correct connectors. This operation must be repeated in order to carry out all the wiring shown in Figure below.

Once the operation has been completed, the 16 control pins of each 16-channel relay shield card must be prepared. Each card actually contains 16 input channels for activating each relay (Fig. 12). However, we will be activating several relays with a single GPIO (to limit the number of GPIOs used on Raspberry Pi, see Section 2.4). To execute this step, it will be necessary to follow the protocol presented in Figure.

For the 16-channel relay shield no. 1, these steps must be followed: * Position a test circuit with 10 horizontal and 10 vertical holes on the pins of the 16-channel relay shield board. * Follow the diagram and solder the pins as shown in Fig. * Lastly, solder 0.5-mm^2 wires 1 m in length to the test circuit.

For relay shield no. 2, follow the same procedure, but solder all the pins together (d-e-f). This same operation must be repeated for the other three multiplexers as well. The next step consists of connecting the relay card inputs to the Raspberry Pi according to Table 5 for all four multiplexers.

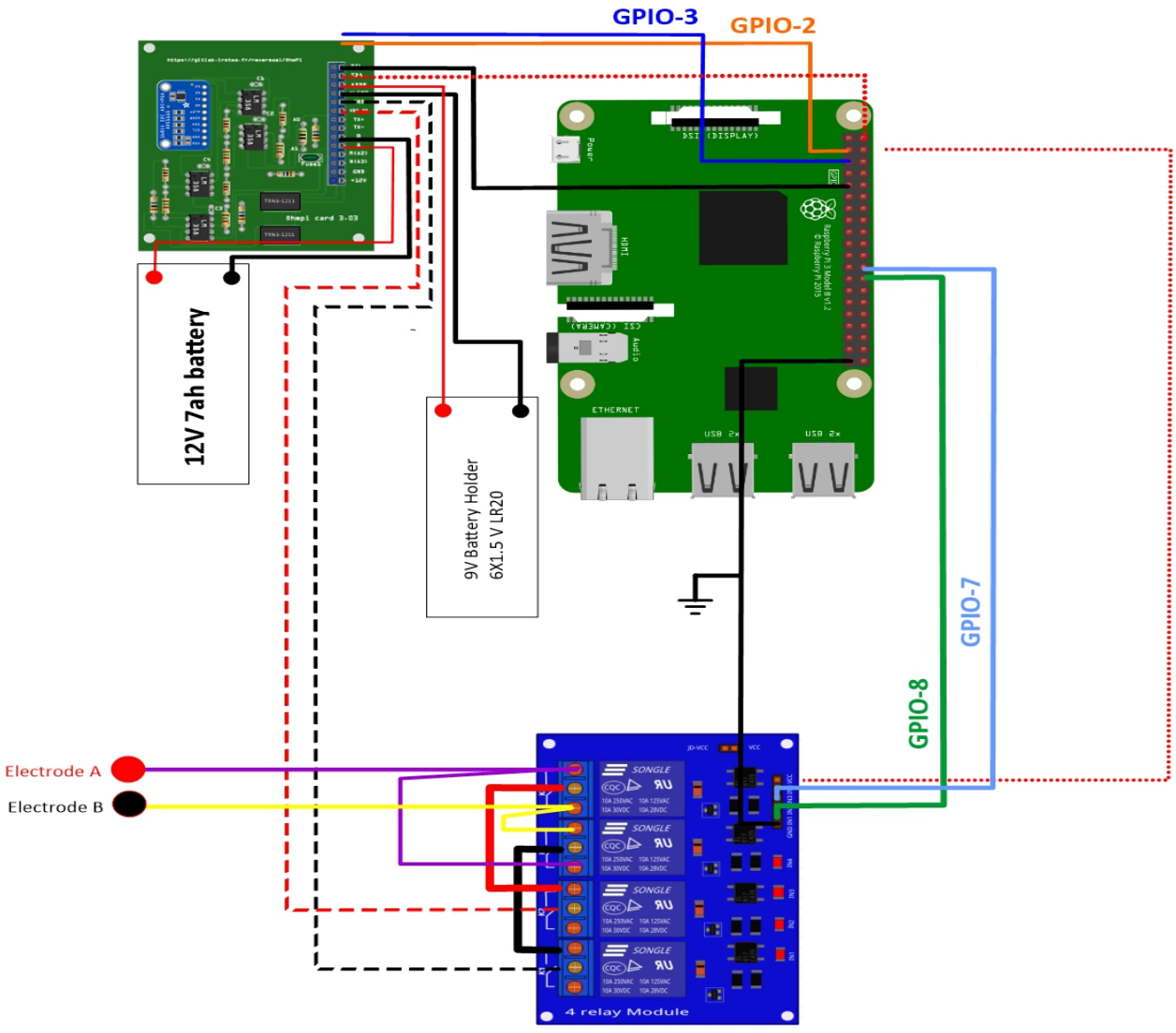


Fig. 27: Current injection board installation with Raspberry Pi

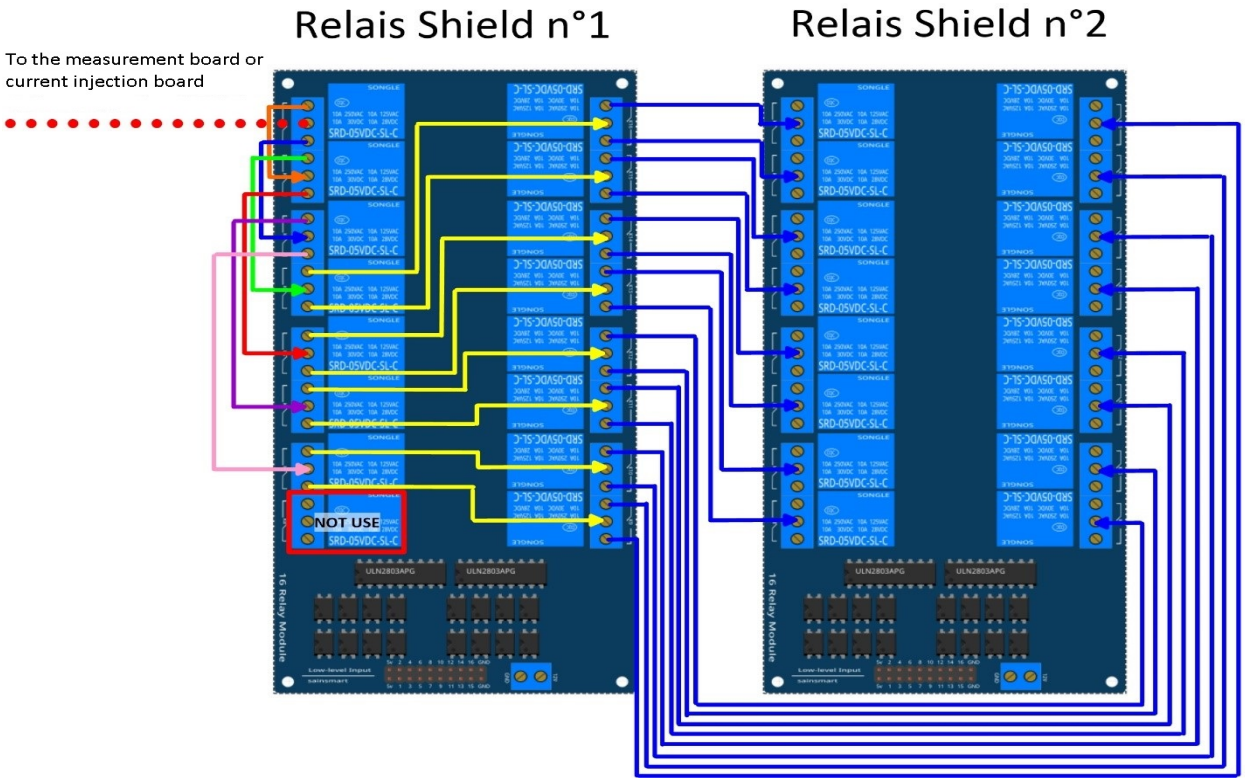


Fig. 28: Schematic diagram of the wiring of two 16-channel relay shields

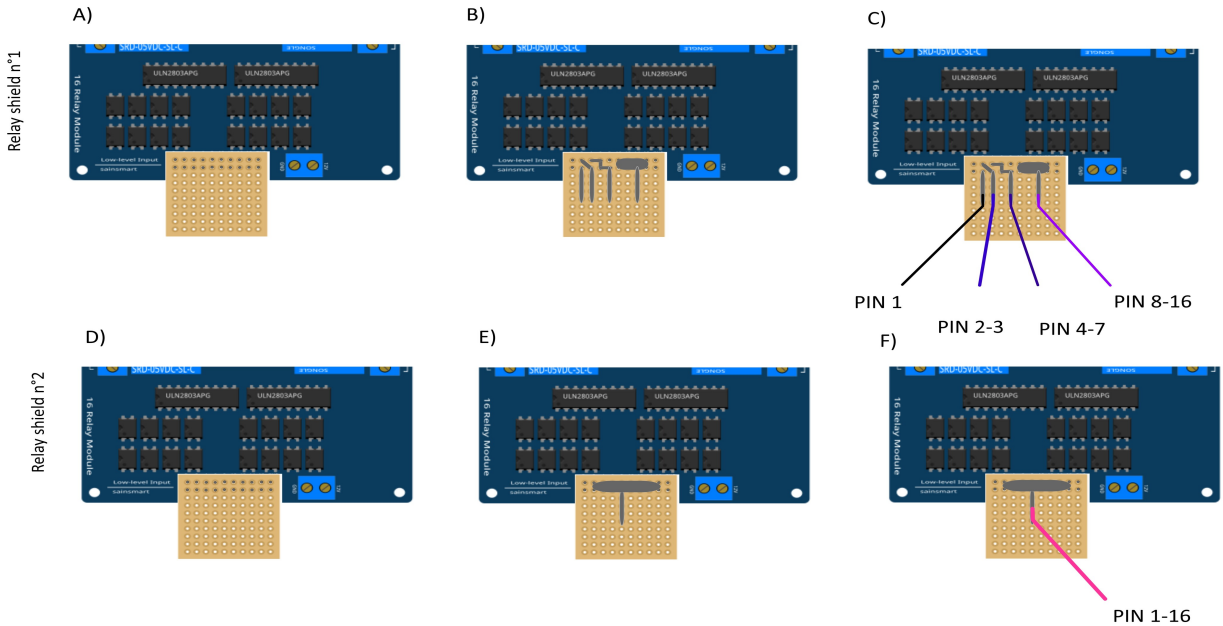


Fig. 29: Connection to the 16-channel relay shield

	Relay shield n°1				Relay Shield n°2
	Pin 1	Pin 2-3	Pin 4-7	Pin 8-16	Pin 1- 16
Multiplexer A	12	16	20	21	26
Multiplexer B	18	23	24	25	19
Multiplexer M	06	13	04	17	27
Multiplexer N	22	10	09	11	05

Connection of the GPIOs to each multiplexer

Electrode connection

At this point, all that remains is to connect the electrodes of each multiplexer to a terminal block (Fig. 13). In our set-up, screw terminals assembled on a din rail were used. According to the chosen multiplexer configuration, all the relays of each multiplexer will be connected to an electrode and, consequently, each electrode will have four incoming connections. Instead of having four cables connecting an electrode terminal to each multiplexer, we recommend using the cable assembly shown in the following Figure.

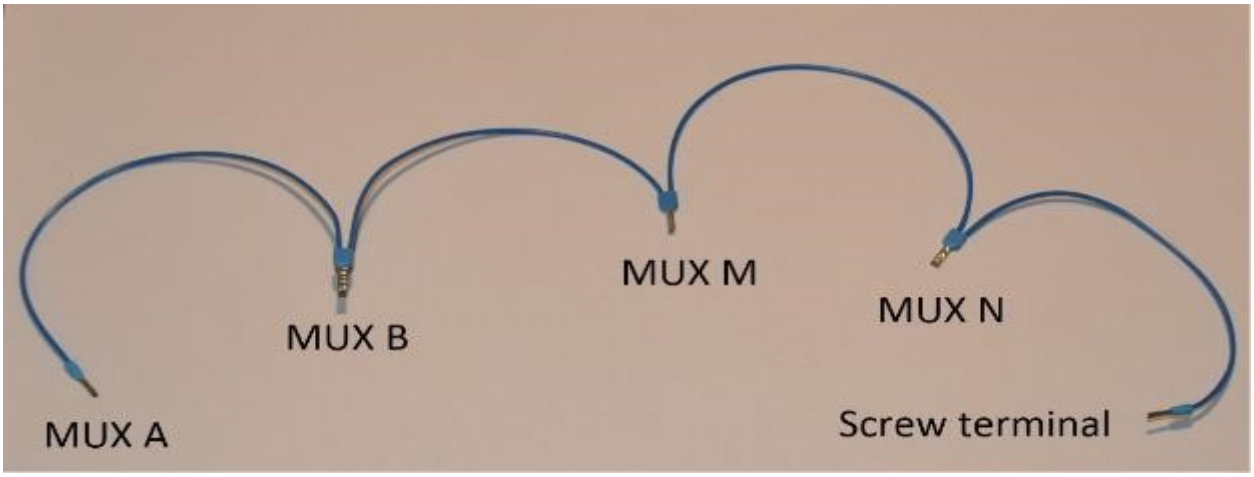


Fig. 30: Wire cabling for multiplexer and terminal screw connection

the next figure provides an example of multiplexer relay connections for electrode no. 1: this electrode of multiplexer MUX A must be connected to electrode no. 1 of MUX B. Moreover, electrode no. 1 of MUX B must be connected to electrode no. 1 of MUX N, which in turn must be connected to electrode no. 1 of MUX M. Lastly, electrode no. 1 of MUX M is connected to the terminal block. This operation must be repeated for all 32 electrodes.

Warning: The 16 channel relay cards exist in 5-V and 12-V , in the bottom figure we have 12-V cards that we will directly connect to the battery. In case you bought 16 channel relay 5-V cards, you will need to add a DC/DC 12-V/5-V converter. You can use a STEP DOWN MODULE DC-DC (Velleman WPM404) and set the voltage to 5V with the potentiometer.

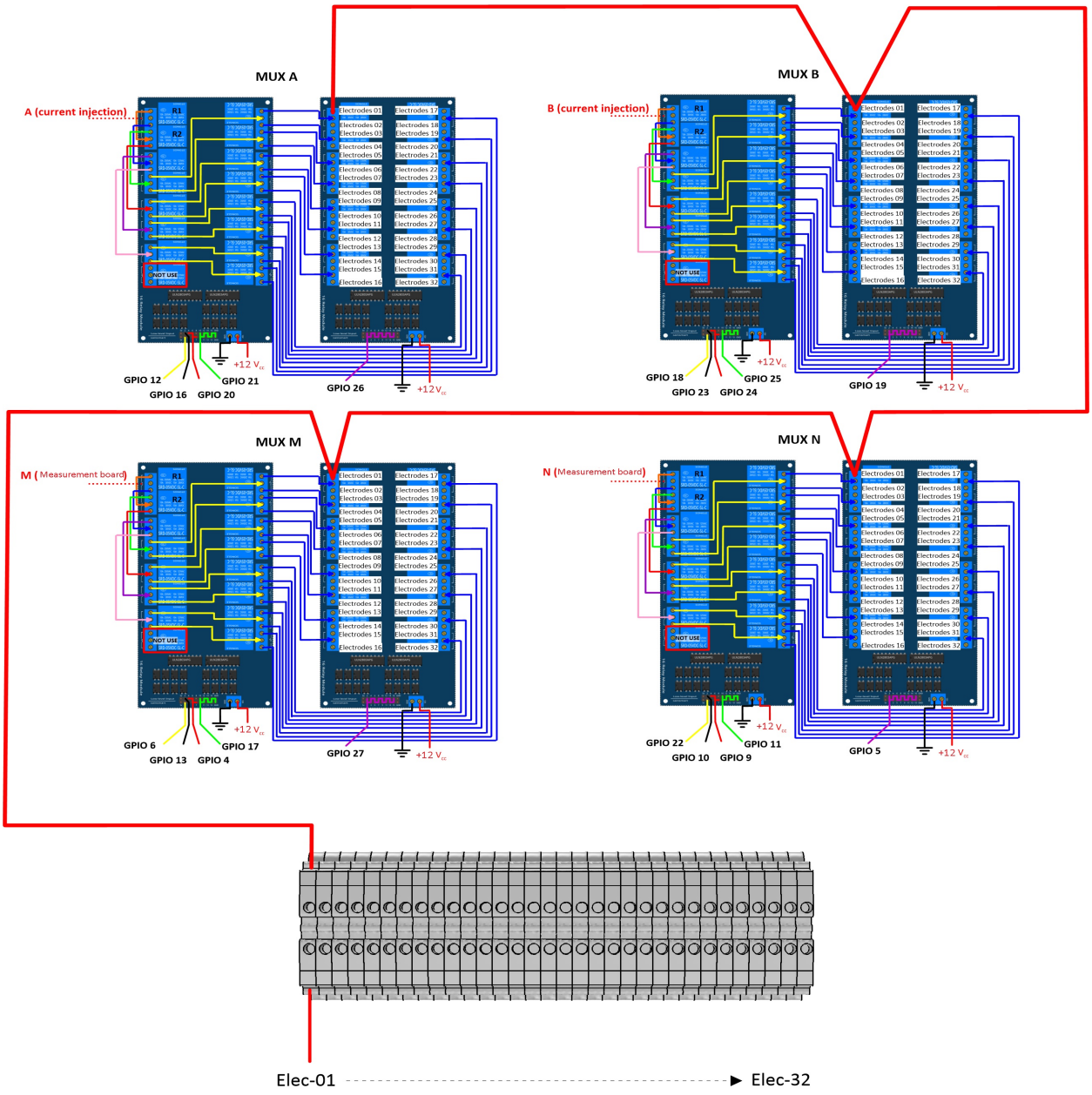


Fig. 31: Example of a multiplexer connection to the screw terminal for electrode no. 1.

Operating instruction

Preliminary procedure (Only for the initial operation)

The open source code must be downloaded at the Open Science Framework source file repository for this manuscript (https://osf.io/dzwb4/) or at the following Gitlab repository address: https://gitlab.irstea.fr/reversaal/OhmPi. The code must be then unzipped into a selected folder (e.g. OhmPi-master). A "readme" file is proposed in the directory to assist with installation of the software and required python packages. It is strongly recommended to create a python virtual environment for installing the required packages and running the code.

Startup procedure

As an initial operating instruction, all batteries must be disconnected before any hardware handling. Ensure that the battery is charged at full capacity. Plug all the electrodes (32 or fewer) into the screw terminals. The Raspberry Pi must be plugged into a computer screen, with a mouse and keyboard accessed remotely. The Raspberry Pi must then be plugged into the power supply (for laboratory measurements) or a power bank (5V - 2A for field measurements). At this point, you'll need to access the Raspbian operating system. Inside the previously created folder "ohmPi", the protocol file "ABMN.txt" must be created or modified; this file contains all quadrupole ABMN numeration (an example is proposed with the source code). Some input parameters of the main "ohmpi.py" function may be adjusted/optimized depending on the measurement attributes. For example, both the current injection duration and number of stacks can be adjusted. At this point, the9 V and 12-V battery can be plugged into the hardware; the "ohmpi.py" source code must be run within a python3 environment (or a virtual environment if one has been created) either in the terminal or using Thonny. You should now hear the characteristic sound of a relay switching as a result of electrode permutation. After each quadrupole measurement, the potential difference as well as the current intensity and resistance are displayed on the screen. A measurement file is automatically created and named "measure.csv"; it will be placed in the same folder.

Electrical resistivity measurement parameters description

In the version 1.02, the measurement parameters are in the Jason file (ohmpi_param.json).

```
nb_electrodes = 32 # maximum number of electrodes on the resistivity meter
injection_duration = 0.5 # Current injection duration in second
nbr_meas= 1 # Number of times the quadrupole sequence is repeated
sequence_delay= 30 # Delay in seconds between 2 sequences
stack= 1 # repetition of the current injection for each quadrupole
export_path= "home/pi/Desktop/measurement.csv"
```

Complete list of components

Warning: The list evolve a little bit after the publication of the article, it is necessary to refer to this list, the article is out of date

Table 15: List of components

Component	Number	Cost per unit	Total cost	Manufacturer	Manufacturer s reference
Raspberry Pi 3 Model B+	1	37	37	Raspberry	Raspberry Pi 3 Model B
Raspberry Pi 1 2 and 3 Power Supply	1	8.37	8.37	Raspberry	Raspberry Pi 1 2 and 3 Power Supply
SainSmart 16-Channe 12V Relay	8	24.99	199.92	Sain Smart	101-70-103
4-Channe 5V Relay Module	1	7.99	7.99	Sain Smart	20-018-101-CMS
cable 1X1 mm2 (50 m)	1	19.66	19.66	TRU COMPONENTS	1568649
cable 1X0.5 mm2 (100 m)	1	29.71	29.71	TRU COMPONENTS	1565235
Printed circuit board (packaging quantity x 3)	1	12	12	Asler	0
Header sets 1x10	1	2.68	2.68	Samtec	SSW-110-02-G-S
Dual screw terminal (2.5-mm pitch)	7	0.648	4.55	RS PRO	897-1332
Resistor 1	4	0.858	3.44	TE Connectivity	H81K0BYA

PDF version of this documentation

PYTHON MODULE INDEX

O

INDEX

P

Q

R

S

T

U

V