`tessie` User Manual

Urs Langenegger

urslangenegger@gmail.com

Paul Scherrer Institute
CH-5232 Villigen-PSI, Switzerland

February 18, 2026
`tessie version 2026/02/18-02`

Abstract

This document provides an overview of the `tessie` software and is intended to provide all information required to install the `tessie` software and safely operate the coldbox for the CMS phase-2 pixel module testing. For future reference, the necessary hardware information is also included.
This document is work in progress. Please send all comments, in particular bug reports and complaints, to the email address given above. Many thanks!

Intended Use: tessie, together with the hardware, is designed to allow the safe operation of a coldbox with climatic chamber for thermal cycling and well-defined running conditions for testing the CMS phase-2 inner tracker (pixel) modules. Following the instructions in this user guide will provide any user the required knowledge and understanding to safely operate tessie and the coldbox.

Safety instructions: It is imperative, every time before operating the coldbox, to

- ensure that the N2 flow is connected to the coldbox and works according to plan (all throttles are open, no sharp bends in the tubing, etc).
- ensure that the chiller is turned on, that the filter inside the chiller is not clogged, and that water is flowing.

The user bears the responsibility for damage and accidents caused by (1) not reading and understanding this user guide, (2) ignoring warnings and/or alarms issued by tessie or (3) modifying the coldbox hardware or deviating from the official build instructions.

Contents

1 Introduction **4**

2 Hardware aspects **6**
2.1 Raspberry Pi . . . 6
2.2 CAN bus . . . 6
2.3 `TEC` controller . . . 9
2.4 Module probe card . . . 9
2.5 Environmental monitoring . . . 9
2.6 CAN controlled FRAS/4 Relays . . . 11
2.7 Lid-closure sensor . . . 12
2.8 Cross section of module cooling infrastructure . . . 12

3 Installation **15**

4 Upgrading `tessie` or the `TEC` firmware **24**
4.1 `tessie` upgrades . . . 24
4.2 TEC firmware upgrades . . . 25

5 Operational aspects **26**
5.1 GUI - `Raspberry Pi` touchscreen interface to `tessie` . . . 26
5.2 Web interface to `tessie` . . . 28
5.3 MQTT - direct text-based communication with `tessie` . . . 29
5.4 Safe operations limits . . . 32
5.5 First steps . . . 33
5.6 Direct readout of probe card . . . 34
5.7 Traffic Lights . . . 35
5.8 Alarm channels . . . 35
5.9 Modes for cooling the `TECs` . . . 35
5.10 Temperatures on the `TEC` and module . . . 35
5.11 Reconditioning the HYT-223 RH/T sensor . . . 36

6 Frequently asked questions **40**
6.1 Operations . . . 40
6.1.1 Do's and don'ts . . . 40
6.1.2 `tessie` does not start . . . 40
6.1.3 An alarm goes off if I plug in a module and close the lid . . . 41
6.1.4 Alarm module temperature exceeds `SAFETY_MAXTEMPM` (single module) . . . 41
6.1.5 Alarm module temperature exceeds `SAFETY_MAXTEMPM` (single module) . . . 41
6.1.6 Alarm module temperature exceeds `SAFETY_MAXTEMPM` (many modules) . . . 42
6.1.7 The CANbus shows many errors . . . 42
6.1.8 Persistent issue with a single `TEC` . . . 42
6.1.9 Where is the printout/output/logfile/CSV file from tessie? . . . 42
6.1.10 The N2 flow valves are not working, but no CANbus errors appear . . . 43
6.2 Installation . . . 43
6.2.1 The touchscreen is white . . . 43

6.2.2 The touchscreen gradually turns white and stays white. 43
6.2.3 Server certificate verification 43
6.2.4 How to test the USB loudspeaker and `tessie` audio alarms 44

References **45**

1 Introduction

The `tessie`[1] program controls all aspects of the safe operation of the PSI coldbox developed for the testing of the CMS phase-2 pixel modules. It is hosted in a github respository [1].

`tessie` is a multi-threaded C++ program running on the Raspberry Pi with a custom hardware "hat" inside the coldbox. Originally, it started as a Qt5 GUI (graphical user interface) and it can still be operated in that way. However, in a production setup, it is mostly controlled through a web interface and/or scripts. The threads in `tessie` control, respectively, the graphical display, the underlying hardware (CAN [2] and I2C [3] bus), and the MQTT messaging service [4]. The web interface is driven by a node.js webserver (`tessieWeb`), running on the Raspberry Pi and listening to the MQTT messages and operating `tessie` through MQTT.

The coldbox, sketched in Fig. 1, comprises eight positions, where TEPX modules can be positioned in thermal contact to Peltier elements, each controlled by custom `TEC` controllers (`TEC` is the abbreviation for ThermoElectric Cooler and a synonym for Peltier element). A centrally placed PCB accomodates the electrical connections and readout of the TEPX modules and furthermore hosts an SHT85 air and humidity sensor [5].

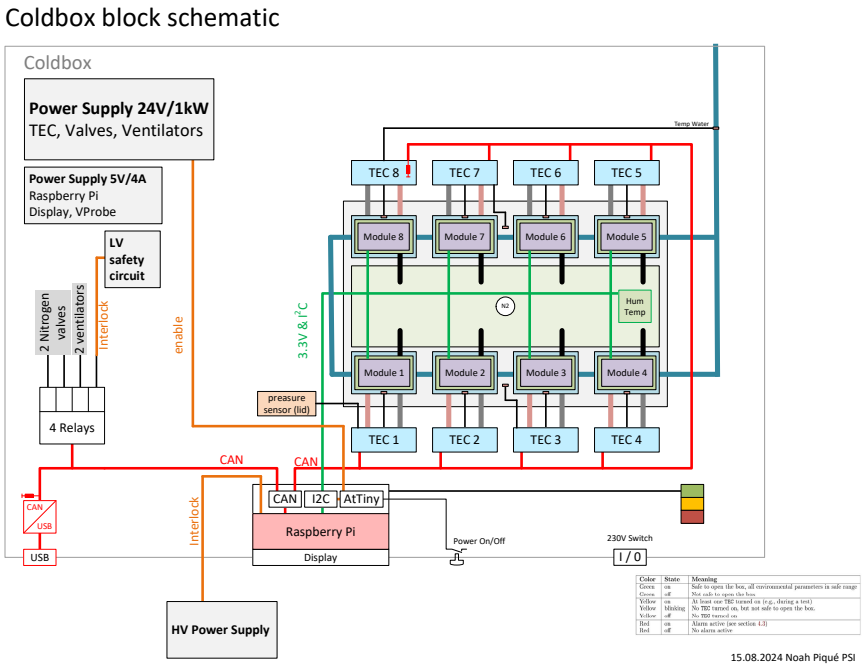


Figure 1: Sketch of the PSI coldbox showing the orientation and numbering scheme of the eight module positions, the CAN bus and I2C bus connection scheme, and the FRAS/4 Relays controlling the N2 flow. The three white boxes above the `Raspberry Pi` are implemented using a custom "hat". The thick blue line exiting at the top right indicates the water pipe to the chiller. Note that `TEC` numbering starts at 1, not 0 (zero). Figure from Ref. [6]

`tessie` enforces a safe operation environment in the coldbox. It will shutdown operations

[1] Etymology: tessie sounds better than TC (box), for temperature cycling (box)

if certain environmental parameters leave the safe region (cf. section 5.4). It will inhibit turning on the `TECs` if the chiller is not running. In addition, a watchdog process on the Raspberry Pi ensures that `tessie` is running and performing its required tasks.

The coldbox can be operated in a manual direct manner and through scripts. Manual direct operation at the box includes for example

- Opening and closing the lid to load or unload modules for testing
- Turning on/off the N2 flow controls via the GUI on the touchscreen.

The normal user interaction proceeds through a web interface and any browser[2]. Figure 2 illustrates the basics of the `tessie` control mechanisms.

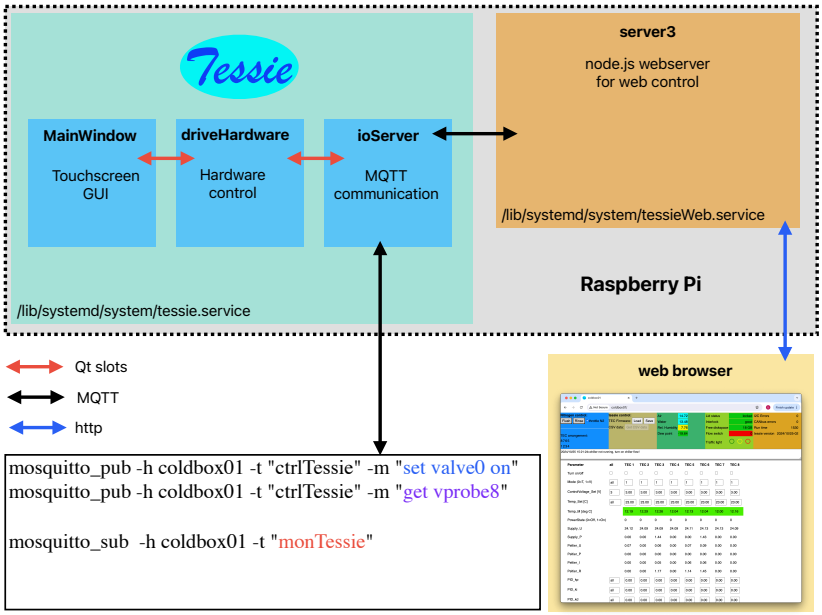


Figure 2: Sketch of the control mechanisms for `tessie`. The basic coldbox hardware control is encoded in the class `driveHardware`. This class accepts input directly from the touchscreen, the MQTT communication protocol, and the web interface via the node server3 (usin MQTT behind the scenes).

In addition, it may be required to log into the coldbox Raspberry Pi using `ssh` to update and recompile the code and/or restart the `tessie` program and/or its webserver.

[2]Not tested on `Edge` or `Internet Explorer`. If you encounter problems with `Edge`, please let us know.

2 Hardware aspects

2.1 Raspberry Pi

The coldbox is operated with a Raspberry Pi Model 4B. Table 1 provides an overview of `tessie`'s pin usage of the `J8` connector on the `Raspberry Pi` (*cf.* the result of the `pinout` command issued on the `Raspberry Pi`).

Table 1: `Raspberry Pi` pins (physical and GPIO numbering scheme) used by `tessie`.

phys.	GPIO	Name	Remarks
11	17	GPIORED	red light of traffic light
13	27	GPIOYELLO	yellow light of traffic light
15	22	GPIOGREEN	green light of traffic light
16	23	GPIOINT	interlock for external usage (`HIGH` = safe for operation)
18	24	GPIOPSUEN	always `HIGH` (not used to date)

A custom-made "hat" provides the interface between the `Raspberry Pi` and the CAN/I2C buses. The CAN bus provides the communication link to the `TECs` and the FRAS. The I2C bus allows the readout of an SHT85 sensor, a HYT-223 sensor, and the module probe card. However, because that "hat" has no direct consequences for `tessie`, it is not discussed further here.

2.2 CAN bus

Communication in the CAN bus proceeds through the transfer of frames, consisting (among others) of an ID (identifier), specifier of the data length (in our system: 0, 1, or 5), and the data. Any connected device can transmit data when the bus is free. All connected devices see all transmissions and decide whether or not to accept it, based on the board identifier and one specific bit in the ID. The ID consists of 11 bits illustrated in Fig. 3. Starting at

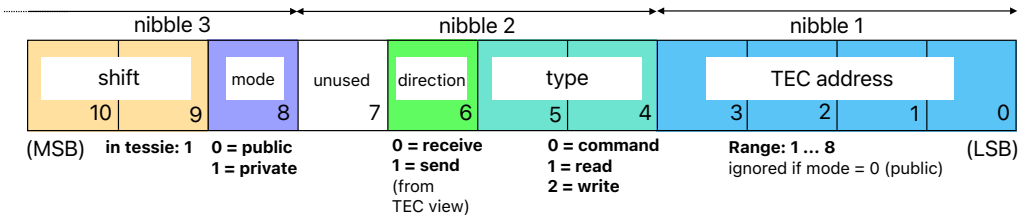


Figure 3: The CAN ID used in the coldbox's CAN bus.

the least-significant bit (LSB), 4 bits are required to encode the `TEC` ID because the first `TEC` has address 1 (not 0). This fact is utilized in the MQTT communication where address 0 corresponds to "all" `TECs`. The next two bits encode whether the frame contains a read (type = 1) or write (type = 2) request, or a command (type = 0). Bit 6 encodes the direction of

the communication, as seen from the TEC. Bit 7 is not used. Bit 8 determines whether the communication is targeted at a specific address (mode = 1) or is a broadcast (mode = 0). The two most significant bits (MSB) are used for a potential address space offset and are set to 0x01. A few remarks:

- Every nibble of the CAN ID has a range <0xA because (1) there are only 8 TECs, (2) bit 7 is not used (and set to 0), and (3) there are only 2 bits in the third nibble.
- The TEC address is immediately visible in the candump printout in the first digit of the displayed address
- The FRAS frames (emitted and received by the FRAS) follow a different bit-encoding scheme, *cf.* subsection 2.6.

The CAN bus traffic can be easily monitored with the candump tool:

```
pi@coldbox02:~ $ candump can0
can0  210   [1]  01
can0  251   [5]  01 00 00 00 00
can0  252   [5]  01 00 00 00 00
can0  253   [5]  01 00 00 00 00
can0  254   [5]  01 00 00 00 00
can0  255   [5]  01 00 00 00 00
can0  256   [5]  01 00 00 00 00
can0  257   [5]  01 00 00 00 00
can0  258   [5]  01 00 00 00 00
can0  210   [1]  02
.. snip ..
can0  318   [1]  08
can0  358   [5]  08 00 4C BB 41
can0  311   [1]  08
can0  351   [5]  08 00 C3 27 3F
can0  210   [1]  09
.. snip ..
can0  318   [1]  0A
can0  358   [5]  0A 00 F6 90 BF
.. snip ..
can0  042   [0]
can0  040   [1]  08
.. and so on ..
```

A short explanation of this printout:

- The first column shows the CAN bus, the second the CAN ID, the third the length of the data block, and the fourth (if present) the data.
- The 042 frames originate with the FRAS if without data (as in the first 042 line above), or corresponds to tessie talking to the FRAS with data: The four bits (starting at the MSB) correspond to LV On/Off, Fan On/Off, Flush, Rinse.

- The `25x` frames correspond to `TEC x` providing register readout, with the register number given in the first data byte (ranging from 0x00 to 0x14, as provided in Table 2). The digits `25` arise from the `shift bit` and `(send|read)`, respectively (cf. Fig. 3).
- Writing to the `TEC` registers would show up as `32x` frames. The digits `32` arise from the `(shift bit|private)` and `write`, respectively (cf. Fig. 3).
- Turning on/off the `TEC x` shows up as `30x [1] 01` and `30x [1] 02`, respectively.

Manual control of the `TECs` via `cansend` is also possible:

- Command: Turn on/off a `TEC` (# 1 in this example)

```
pi@coldbox01:~ $ cansend can0 301#01
pi@coldbox01:~ $ cansend can0 301#02
```

- Register write/read to/from (in this example set register 0x00 to 0x01; read it; reset it to 0x00—Note: the four bytes of the data payload start with the LSB and proceed to the MSB)

```
pi@coldbox01:~ $ cansend can0 321#0001000000
pi@coldbox01:~ $ cansend can0 311#00
pi@coldbox01:~ $ cansend can0 321#0000000000
```

- Read error register of `TEC` #1, clear it with command `Clear Error`, and read again

```
pi@coldbox01:~ $ cansend can0 311#13
pi@coldbox01:~ $ cansend can0 301#05
pi@coldbox01:~ $ cansend can0 311#13
```

- Turn on `TEC` #1, read `PowerState` register, turn off `TEC` #1

```
pi@coldbox01:~ $ cansend can0 301#01
pi@coldbox01:~ $ cansend can0 311#12
pi@coldbox01:~ $ cansend can0 301#02
pi@coldbox01:~ $ cansend can0 311#12
```

- Read the numerical value of the lidsensor (`Temp_W` of `TEC` 1)

```
pi@coldbox01:~ $ cansend can0 311#08
```

To see the results of these CAN bus interactions, you must have `candump can0` running in a different terminal. Note that if/when `tessie` is running, it will be non-trivial to see the CAN bus frames given the large CAN bus traffic from `tessie`.

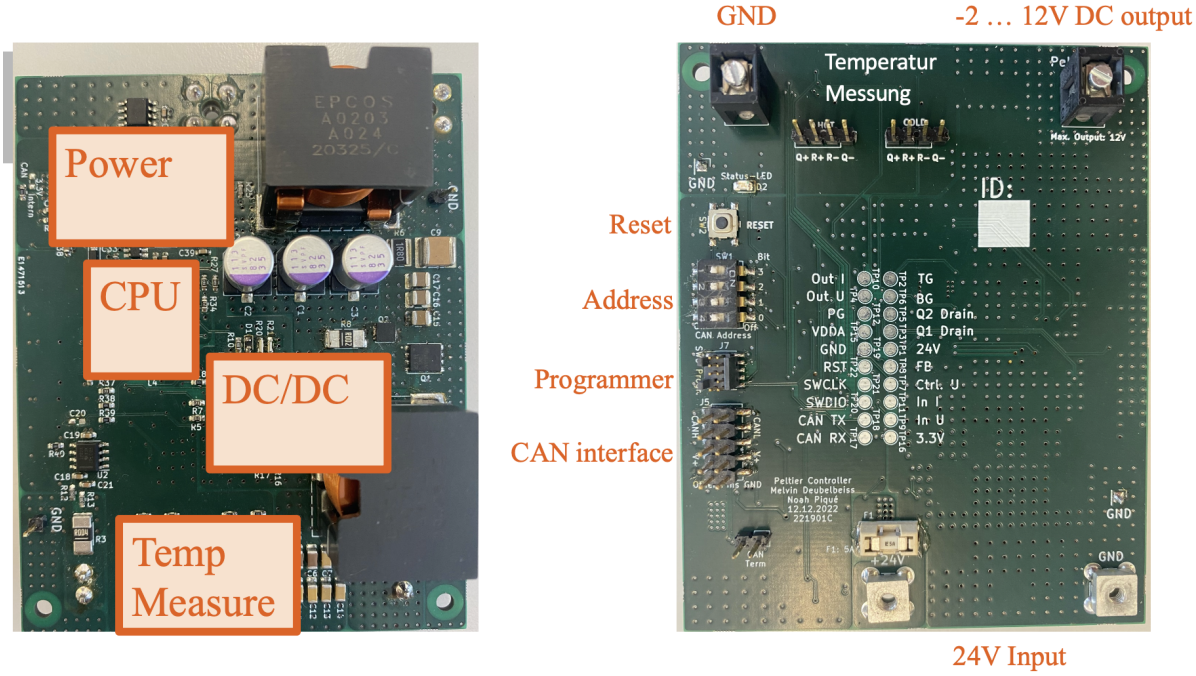


Figure 4: Photo of the (left) frontside and (right) backside of the `TEC` controller [6]. The fuse F1 is visible on the backside.

2.3 `TEC` controller

The `TEC` controllers are a central element of the coldbox and allow the direct or automated (PID) control of the `TECs`, in addition to their powering and monitoring. A coldbox houses 8 `TEC` controllers, one for each module position. The `TEC` controllers are attached to the CAN bus and have board IDs (addresses) 1…8. Controlling and obtaining the status of the `TEC` controllers is achieved through control commands and through writing to and reading values from various registers, provided in Table 2.

The `TEC` controllers indicate their internal state through an error register (register 19 `ErrorState`). No error is indicated through the value 0 in that register. Error states are categorized depending on their origin or affected impact, as shown in Table 3. These categories are sorted in order of increasing severity.

2.4 Module probe card

The module probe card [7], illustrated in Fig. 5 and sometimes referred to as "Vprobe" below, is used to measure voltage levels on test points on all four chips of a module. Since its readout is connected to the coldbox I2C bus, `tessie` is used to obtain the readings of this probe card.

2.5 Environmental monitoring

Initially an SHT85 sensor [5] was used to measure the air temperature and relative humidity inside the coldbox. Its readout proceeded in the "high-repeatability mode." The I2C device

Table 2: TEC Registers and commands. The flash variables are indicated with (F).

Register	Name	Type	Description
0	Mode	R/W (UInt32)	0=ConstTemp(PID) 1=ConstVoltage(manually)
1	ControlVoltage_Set	R/W (Float)	Setting the output peltier voltage
2	PID_kp	R/W (Float) (F)	PID loop parameter
3	PID_ki	R/W (Float) (F)	PID loop parameter
4	PID_kd	R/W (Float) (F)	PID loop parameter
5	Temp_Set	R/W (Float)	Target Module temperature
6	PID_Max	R/W (Float) (F)	Max. output Peltier voltage
7	PID_Min	R/W (Float) (F)	Min. output Peltier voltage
8	Temp_W	R (Float)	Water temperature
9	Temp_M	R (Float)	Module temperature
10	Temp_Diff	R (Float)	Temperature difference
11	Peltier_U	R (Float)	Voltage applied to Peltier element
12	Peltier_I	R (Float)	Current through Peltier element
13	Peltier_R	R (Float)	Resistance in Peltier element
14	Peltier_P	R (Float)	Power consumption of Peltier element
15	Supply_U	R (Float)	Voltage applied to TEC controller
16	Supply_I	R (Float)	Current through to TEC controller
17	Supply_P	R (Float)	Power consumption of TEC controller
18	PowerState	R (UInt32)	0=Off, 1=On
19	ErrorState	R (UInt32)	See dedicated error code table
20	Ref_U	R (Float) (F)	Reference voltage (ADC/DAC)
Address	**Command**	**Type**	**Description**
0	No Command	Command	Nothing
1	Power_On	Command	Turn on TEC
2	Power_Off	Command	Turn off TEC
3	Watchdog	Command	Send every <3 seconds
4	Alarm	Command	With Errorcode from TEC to master
5	Clear Error	Command	
6	Get SW Version	Command	Returns TEC firmware version
7	Save Variables	Command	Saves all variables into flash memory
8	Load Variables	Command	Loads all variable from flash memory
255	Reboot	Command	

address of the sensor is 0x44. The data transferred consist of 2×2 bytes together with 2 bytes for a CRC checksum.

Unfortunately, the SHT85 ages rapidly when exposed to extremely dry atmospheric conditions (< 5%RH) and needs to be re-conditioned at 200°C for ca 10 hours. This is not useful for a production system.

Therefore a new solution was found with the HYT-223 RH/T sensor [8]. This sensor allows for in-situ re-conditioning through an integrated heating circuit. To allow its addressing via the

Table 3: TEC error values in the ErrorState register (register number 19), cf. Table 2. The error categories are indicated by the most significant byte (the mask in the table).

Error name	Error code	Comment
Temperature Errors (TEMP_ERROR_MASK 0x05000000)		
TEMP_ERROR_SENSORM_MASK	0x00100000	
TEMP_ERROR_SENSORW_MASK	0x00200000	
TEMP_ERROR_SPI_FAILURE	0x00010000	
TEMP_ERROR_GENERAL_FAILURE	0x00020000	
TEMP_ERROR_SENSOR_FAILURE	0x00040000	
Variable Handler Errors (VARH_ERROR_MASK 0x06000000) Byte 2 (0x0000XX00) is the register number, where the Error is coming from		
VARH_ERROR_INVALID_VARIABLE	0x00000001	
VARH_ERROR_INVALID_DATA	0x00000002	
VARH_ERROR_READONLY	0x00000004	
VARH_ERROR_OUTOFRANGE	0x00000008	
VARH_ERROR_OUTOFRANGE_INT	0x00000010	(INT = intern)
VARH_ERROR_LOAD_FLASH	0x00000020	All variables will be set to default
Peltier Errors (PELTIER_ERROR_MASK 0x07000000)		
PELTIER_ERROR_SETVOLTAGE	0x00000001	
PELTIER_ERROR_SETVOLTAGE_PID	0x00000002	
PELTIER_ERROR_MAX_POWER	0x00000004	
Main Application Errors (PELTIER_ERROR_MASK 0x08000000)		
MAIN_ERROR_REG_NOT_FOUND	0x00000001	
MAIN_ERROR_CMD_NOT_FOUND	0x00000002	
MAIN_ERROR_SAVE_FLASH	0x00000004	
MAIN_ERROR_WATCHDOG	0x00000008	not implemented
Hard Fault Errors (HARDFAULT_ERROR_MASK 0x09000000)		
HARDFAULT_ERROR_MASK	0x09000000	
HARDFAULT_ERROR_IWDG	0x00000001	
HARDFAULT_ERROR_SWRST	0x00000002	Software Reset

CAN bus, a custom (small) PCB was developed and is shipped by PSI. It is the responsibility of the user (e.g. Dirigent, OHIO GUI, etc.) to periodically re-condition the HYT-223 as described in section 5.11. Note that the air temperature will rise considerably since the heater circuit will produce local temperatures of $> 120°C$.

2.6 CAN controlled FRAS/4 Relays

To steer the two N2 valves, the ventilator, and to provide a second interlock signal for the module low-voltage Schütz, a "CAN Output FRAS4 4x Relais Out" device is connected to the CAN bus [9]. We refer to this device as "FRAS" in the following. Its baud rate should be at 125 kbits/s (default setting when delivered), following the instructions in the datasheet. The CAN bus ID (module address) should be set to 0/8 with the "top"/"bottom" rotary switches. This setting corresponds to module address 0x4y, where y = 0/1/2 for

process/service/control frames, respectively. The FRAS must receive (service) frames with a time interval smaller than three seconds. `tessie` sends such a frame every second. In case the FRAS is active, `tessie` sends a *process* frame every second.

2.7 Lid-closure sensor

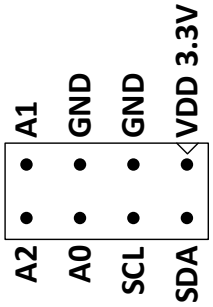
The lid-closure sensor is read out via the "water" temperature sensor of `TEC` 1, cf. the sketch in Fig. 1. Its reading is 4200 when open and below 4100 when the lid is closed. This value is transmitted periodically via the MQTT message `Temp_W = 175.775,-99,-99,-99,-99,-99,-99,19.5097`, where the lid sensor state and the water temperature are the only non-trivial values (cf. section 5.3).

2.8 Cross section of module cooling infrastructure

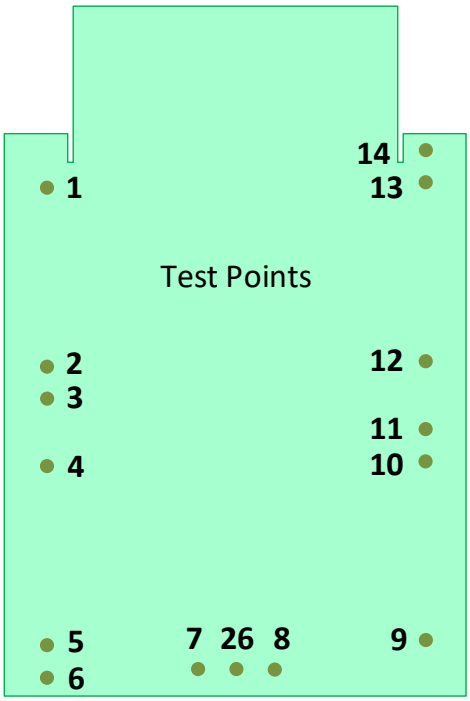
The cross section of the module cooling infrastructure is shown in Fig. 6. A few remarks may be useful.

- The PT1000 providing the temperature reading for the `TEC` register called Temp_M is not in thermal contact with the module but rather with the `TEC` (the Peltier module).
- The module has a thermal power of about 10-12 W (after accounting for voltage drops between the power supply and the module). This implies that the temperature on the module is substantially higher (of the order 15-20°C) on the module than the `TEC`. This is discussed and illustrated in Fig. 13 in section 5.10.

Module Probe V2



A2..A0: I2C Address
0: GND
1: Open



I²C Address Module Probe

0	1	1	A2	A1	A0	C

I²C Address Humidity Sensor

1	0	0	0	1	0	0

Module I²C Addresses (hex)

Pos	1	2	3	4	5	6	7	8
A[2:0]	0	1	2	3	4	5	6	7
C=0	30	32	34	36	38	3A	3C	3E
C=1	31	33	35	37	39	3B	3D	3F

Module ADC Readout Data 18 Bytes

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
C=0	TP 12		TP 8		TP 7		TP 11		TP 10		TP 9		TP 5		TP 6		201	1
C=1	TP 1		TP 8		TP 26		TP 14		TP 13		TP 2		TP 3		TP 4		201	1

ADC value conversion: $U[V] = \frac{V_{DD}}{65536} \cdot ADC$

Beat Meier PSI 12.6.2023

Figure 5: Module probe card documentation [7]. Note that $V_{DD} = 3.3114$, with a permille-level variation between boards. In the readout, bytes 16 have the value 201 as cross-check, and bytes 17 indicate the software version of the board.

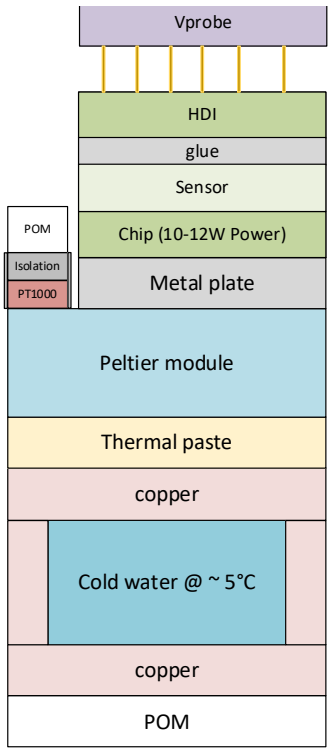


Figure 6: Cross section of the cooling infrastructure for a single module inside the coldbox. From the top: "Vprobe" is the probe card (subsection 2.4), the structures from the "HDI" down to and including the "Chip" represent the module, positioned on a metal plate in direct contact to the `TEC` (Peltier module). The cold water from the chiller is circulating in a copper enclosure. Note the placement of the PT1000 measuring the "`Temp_M`" register of the `TEC` controller: despite its name, it does *not* measure the module temperature but rather the `TEC` temperature of the module!

3 Installation

We assume that you have a coldbox where the hardware is completely configured according to the instructions [10] and connected to the internet. In this section we describe how to prepare the Raspberry Pi, starting from creating its boot device, installing all required software components, and setting up the automatic `tessie` startup at boot time.

A few important remarks before describing the installation procedure.

- If the Raspberry Pi's screen turns white (at the end of the boot process) very likely the screen flatband connector is not properly inserted. This can happen easily when inserting the SD card.

- It has been observed that a few power-cycles are required to have the touchscreen work properly (instead of displaying "nothing", which can mean a white screen or a black screen). Alternatively, try to connect via ssh and do (in a terminal) `sudo shutdown -r now`. Note that "nothing" is not the same as the white screen indicating a flatband cable-connector issue.

  It seems that this issue is due to newer releases of Debian version 12 (bookworm) in image files dated 2024-03-13 and 2024-03-15. It is not present in the image file dated 2023-12-05, referred to below.

- The following instructions have been tested verbatim (line by line copy-paste) with a Raspberry Pi 4 Model B Rev 1.4 with 8 GB RAM. Please provide feedback if you run into problems with a different Raspberry Pi.

- There is a script-based installation procedure provided by Branislav (Bane) Ristic at his forked repository. Depending on the image you start with, his setup may work well for you. It should be rather straightforward to merge that branch with the `master` branch of the main `tessie` repository since it is only about installation.

Installing `tessie` is straightforward, if the following steps are followed.

- Using the "Rasberry Pi Imager" [11], available for macOS, Windows, and Linux, burn a SD card with the 2023-12-05 image file, available from here. Make sure that you do *not* use XZ utils versions 5.6.0 or 5.6.1 since they contain malicious code (cf. link).

  It is recommended to apply a few changes to the default setup as illustrated in Fig. 7, in particular set the user name and password, the hostname, and allow `ssh` access for remote work; this is available from the Imager after you have specified the model, the OS, and the storage device (the SD card).

- Insert the SD card into the foreseen slot beneath the touchscreen flatband cable connector and power up the Raspberry Pi, i.e., plug in the (USB-C) power cable. If the touchscreen goes white, it is likely due to the "big" flatband connector not being plugged in correctly. Try again.

  On its first power-up, the Raspberry Pi will reboot various times. Be patient.

  At the end of the startup/boot sequence you should see a bluish background image showing a cormorant fisherman of Guilin, China.

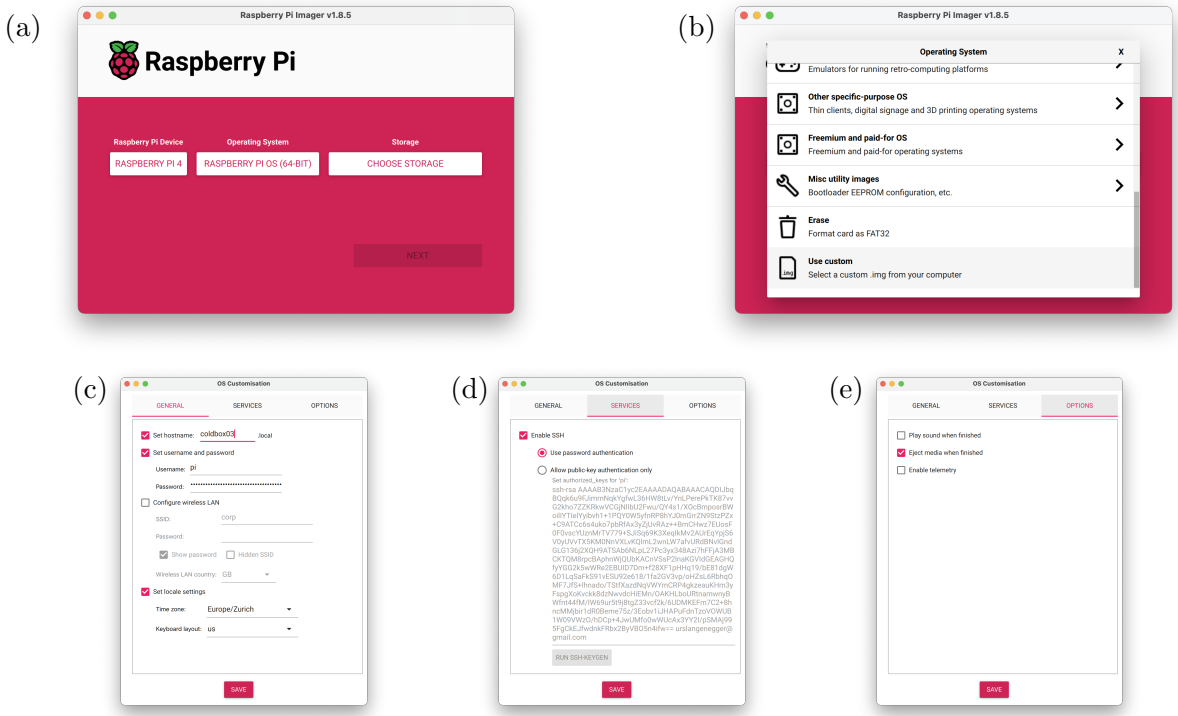


Figure 7: Screen shot of the Raspberry Pi Imager and customization examples. (a) Select the model and OS, (b) choose 'EDIT SETTINGS' to reach the lower panels, (c) enter the hostname and set a password for the default user 'pi' (use this!), (d) enable ssh connections, (e) if you want.

- *Installation of dependencies*
  Open a terminal (it should be accessible from one of the icons at the top of the display) or, better, login from another computer using `ssh` and do the following:

```
sudo date -s "Wed Apr 17 2024 10:00:00"

sudo apt-get update

sudo apt install -y nodejs
sudo apt install -y npm

sudo apt-get install -y libmosquitto-dev libmosquittopp-dev
sudo apt install -y mosquitto mosquitto-clients

sudo apt install -y libqt5charts5 libqt5charts5-dev

sudo apt install can-utils

sudo apt-get install -y nginx

sudo apt-get install -y cmake libusb-1.0
```

Do enter the correct day, date, and time in the first line above. Else you will get "server certificate verification" failures further down and other issues will arise.

- *Installation of tessie*
  Get the `tessie` software and compile it:

  ```
  cd /home/pi
  git clone https://github.com/ursl/tessie.git
  cd tessie/src
  qmake -o Makefile  tessie.pro
  make -j2
  ```

  Note 1: It is not recommended to use more than 2 cores for the compilation ("-j2") because of potential memory shortages.

  Note 2: In case you want to compile `tessie` on a non-Raspberry Pi host without I2C/CAN bus, invoke `qmake "CONFIG+=NOPI" -o Makefile tessie.pro`.

- *Mosquitto setup*
  Using the `nano` editor in `sudo` mode, *i.e.*, `sudoedit` in a vanilla system, edit the file `/etc/mosquitto/mosquitto.conf` and add the following two lines to the end of the file:

  ```
  listener 1883
  allow_anonymous true
  ```

In case the above instructions are unclear, the following is what you should type into the terminal: `sudoedit /etc/mosquitto/mosquitto.conf`, jump to the end, insert the two lines, and exit the editor (using in sequence: `CTRL-x  y  RET`).

- *Hardware (I2C and CAN) bus configuration*
  Using `sudoedit`, edit the file `/boot/firmware/config.txt` to contain the following two lines:

  ```
  dtparam=spi=on
  dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=25
  dtoverlay=spi-bcm2835-overlay
  dtparam=i2c_vc=on
  ```

- *tessie webserver*
  Setup the tessie web server by installing all required node packages

  ```
  cd /home/pi/tessie/node/test1
  npm install --save express socket.io mqtt
  ```

- *tessie and related services startup at boot time*
  With `sudoedit` create the file `/lib/systemd/system/tessie.service` with the following content (i.e. do `sudoedit /lib/systemd/system/tessie.service` and copy-paste the following):

  ```
  [Unit]
  Description=tessie
  After=network.target

  [Service]
  Type=idle
  User=pi
  Group=staff
  Environment="XAUTHORITY=/home/pi/.Xauthority"
  Environment="DISPLAY=:0"
  WorkingDirectory=/home/pi/tessie/src
  ExecStartPre=/home/pi/tessie/resetCAN.sh
  ExecStart=/home/pi/tessie/src/tessie -f
  StandardOutput=inherit
  StandardError=inherit
  Restart=on-failure
  RestartSec=2s

  [Install]
  WantedBy=graphical.target
  ```

  Note that `tessie` is run under the user pi. For the tessie webserver do `sudoedit /lib/systemd/system/tessieWeb.service` with the following contents

```
[Unit]
Description=tessie
After=multi-user.target

[Service]
Type=idle
WorkingDirectory=/home/pi/tessie/node/test1
ExecStart=/usr/bin/node /home/pi/tessie/node/test1/server3.js
Restart=on-failure
RestartSec=2s

[Install]
WantedBy=multi-user.target
```

FIXME A description of the watchdog service and its installation will follow in due time.

- *Configure nginx*
  This optional section allows connecting to `http://coldbox03` instead of `http://coldbox03:3000`.
  Create the `nginx` configuration file with the command `sudoedit /etc/nginx/sites-available/default` and replace the contents of the file with the following contents

```
server {
    listen 80;
    server_name coldbox03.psi.ch;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

  Do change `coldbox03.psi.ch` to your coldbox hostname and domain! Be careful when copy-pasting the inverted commas! Start the service with

```
sudo service nginx start
```

- *Configure sound*
  This extremely important section provides information on how to enable sound output to an attached USB loudspeaker. First determine which sound card your loudspeaker is attached to with

```
cat /proc/asound/cards
```

This will result in a list similar to

```
 0 [vc4hdmi0       ]: vc4-hdmi - vc4-hdmi-0
                      vc4-hdmi-0
 1 [vc4hdmi1       ]: vc4-hdmi - vc4-hdmi-1
                      vc4-hdmi-1
 2 [Headphones     ]: bcm2835_headpho - bcm2835 Headphones
                      bcm2835 Headphones
 3 [UACDemoV10     ]: USB-Audio - UACDemoV1.0
                      Jieli Technology UACDemoV1.0 at usb-0000:01:00.0-1.3, full speed
```

Here the loudspeaker is attached to card 3. With this information `sudoedit` the file `/etc/asound.conf` with the contents

```
defaults.pcm.card 3
defaults.ctl.card 3
```

You should make sure that the USB loudspeaker works and is easily audible (cf. section 6.2 for instructions if necessary)

- *Startup services*
  Now enable the startup of the two low-level components at boot time plus tessie and its webserver

  ```
  sudo systemctl enable pigpiod
  sudo systemctl enable mosquitto.service

  sudo systemctl enable tessie.service
  sudo systemctl enable tessieWeb.service
  ```

  You can always monitor the status of these "services" with

  ```
  systemctl status tessie
  systemctl status tessieWeb
  ```

- *Hardware power buttton configuration*
  Download the auxiliary software package and install it:

  ```
  cd /home/pi
  git clone https://github.com/Howchoo/pi-power-button.git
  ./pi-power-button/script/install
  ```

  Warning: If you do this step on a Raspberry Pi that is *not* in a coldbox with a central power button, it will likely shutdown down and not properly power up!

- *Splash screen configuration*
  (Note: This is not compulsory. There is no real need to change the splash screen.)
  Using `sudoedit`, edit the file `/boot/firmware/cmdline.txt` to contain *on one line* the following two lines (they are provided here on two lines such that they can be copied in their entirety):

  ```
  console=serial0,115200 console=tty1 root=PARTUUID=7a0cea11-02  rootfstype=ext4
  fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles
  ```

  Using `sudoedit`, edit the file `/boot/firmware/config.txt` to contain

  ```
  disable_splash=1
  ```

  Enter the following in a terminal:

  ```
  cd /usr/share/plymouth/themes/pix/
  sudo mv splash.png splash.png.bac
  sudo cp /home/pi/tessie/splash.png ./
  ```

Now reboot the system, e.g., with `sudo shutdown -r now`. If the shutdown process gets stuck, hit the central power button. If all goes well, the touchscreen of the Raspberry Pi will show the GUI featured in Fig. 8. You can connect from any PC.
The normal manual way to interact with `tessie` is through a webbrowser. Point your favorite browser to `http://coldbox03`, cf. Fig 9.

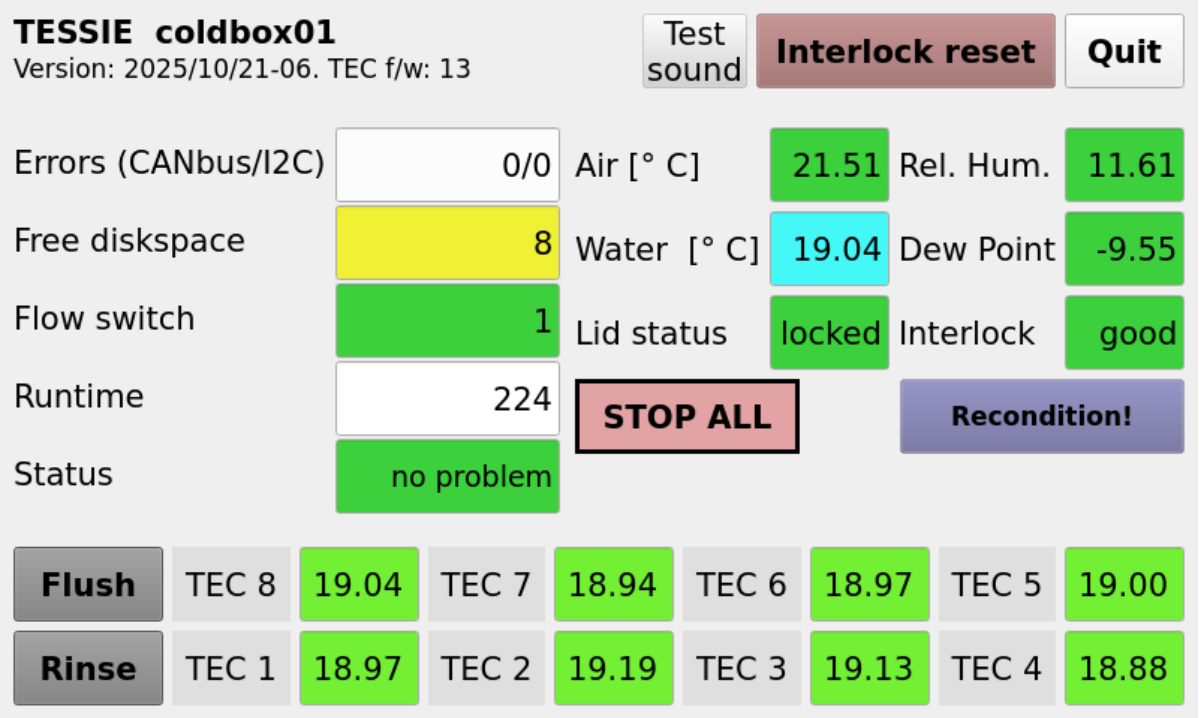


Figure 8: The GUI appearing on the Raspberry Pi touch screen. The interaction possibilities are limited to (1) controlling the N2 flow with `Rinse` and `Flush`, (2) stop operations with "STOP ALL" which will turn off all `TECs`, turn on maximal N2 flow by opening both the rinse and flush valves, and breaking the HV/LV interlocks, (3) resetting the interlocks, (4) "Quit"-ing the program, and (5) sounding the test alarm. The `Rinse` and `Flush` buttons will have a greenish background color when these valves are open and N2 is flowing. Their font is white in case N2 throttling is active and black in case it is not. For full control of `tessie` you should use the web interface shown in Fig. 9.

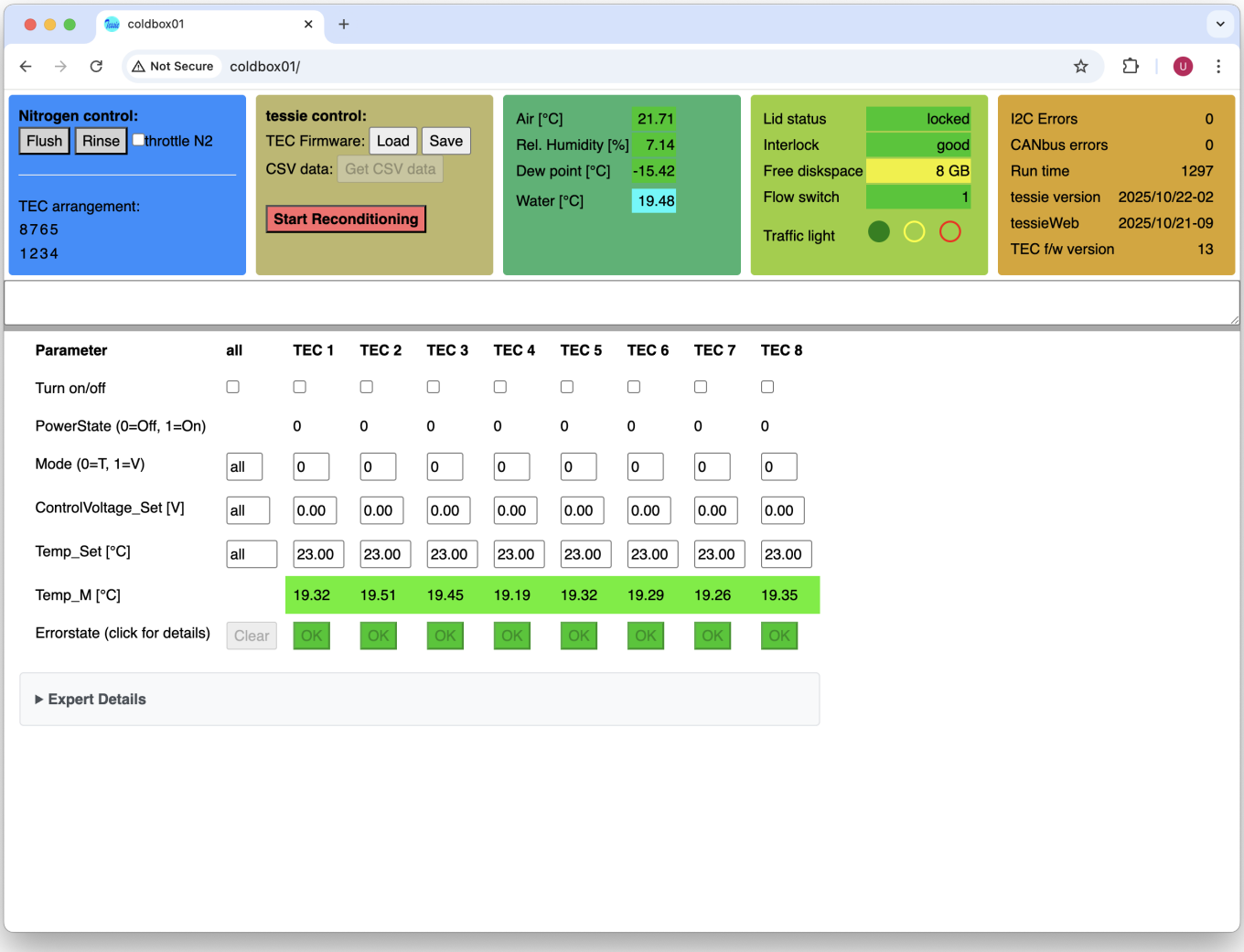


Figure 9: Web graphical interface to `tessie`. The top right brown-yellow box will turn red in case the connection to `tessieWeb` is interrupted.

4 Upgrading `tessie` or the `TEC` firmware

4.1 `tessie` upgrades

Note: It is strongly recommended to frequently update the `tessie` source code, and run an uptodate version of the code!

Upgrading **`tessie`** is straightforward in a terminal. If you are using the HEAD of the master branch, do:

```
cd /home/pi/tessie
git pull
git checkout
cd src
qmake -o Makefile  tessie.pro
make -j2
```

It can be argued that you should work with tagged versions of **`tessie`** to make sure that you are working with a tested, stable, and well-defined version of the software. Note that the tagged versions are described on the tessie github page [1]. You start with a tagged version using

```
cd /home/pi/tessie
git pull origin master
git fetch --tags
git checkout 2025/05/23-01
cd src
qmake -o Makefile  tessie.pro
make -j2
```

Note that git will tell you that you are in a 'detached HEAD' state. Unless you intend to do code development, you can safely ignore it. If you want to do code development and commit your changes, read the rest of the warning message and do as told.
You can check which tags are available with

```
cd /home/pi/tessie
git --no-pager tag
```

If you want to go to the HEAD of the master branch (where all development takes place), do

```
cd /home/pi/tessie
git checkout master
```

To make the changes take effect, the minimal commands are:

```
sudo systemctl restart tessie
sudo systemctl restart tessieWeb
```

Alternatively, you can reboot the coldbox, either by turning it off/on (pressing the central power button) or in a terminal:

```
sudo shutdown -r now
```

In both cases, you can verify the update by comparing the version string of the GUI displayed on the Raspberry Pi's touch screen and the web GUI (a reload of the page may be required).

4.2 TEC firmware upgrades

Upgrading the TEC firmware is done with the OpenBLT Bootloader [12], a very convenient tool for downloading micro-controller firmware over the CAN bus (plus other options, which are not relevant for this context). No additional cabling is required. A minimal version of this software is distributed with the tessie repository.

Start by compiling the bootloader's library and executable with the following commands:

```
cd /home/pi/tessie/tecFirmware/Source/LibOpenBLT
mkdir _build && cd _build
cmake ..
make

cd ../../BootCommander
mkdir _build && cd _build
cmake ..
make
```

Stop tessie before uploading the TEC firmware! So:

```
sudo systemctl stop tessie
```

After this you can upload, via the CANbus the firmware, contained in file srec/tecware.srec, of a single TEC with the following command:

```
cd /home/pi/tessie/tecFirmware
./BootCommander -s=xcp -t=xcp_can -d=can0 -b=125000 -tid=0x301 \
                -rid=0x341 srec/tecware_v13.srec
```

Note: This is the example command to upload TEC 1. For TEC i, both the -tid=0x30i and -rid=0x30i command line arguments should be modified accordingly.

To upload all TECs with the same firmware file, a (very) trivial shell script is provided:

```
cd /home/pi/tessie/tecFirmware
./flashAllTECs srec/tecware_v13.srec
```

Note 1: You need at least v12 of the TEC firmware (better v13) to run tessie. tessie will refuse to run if your TEC firmware versions are not uptodate or inconsistently loaded.
Note 2: Uploading the TEC firmware will *not* change the variable values you store(d) in the flash memory. This can be regarded a feature or not, it depends on your point of view. To write the variables contained in the new firmware to the flash memory, you need to run the following after uploading the firmware:

```
cd /home/pi/tessie/tecFirmware
./saveToFlash
```

We *strongly* recommend to always run this command after uploading a new firmware version!

After this procedure, you can restart tessie:

```
sudo systemctl start tessie
```

5 Operational aspects

There are three methods to control the behavior of `tessie`:

- Graphical User Interface on the `Raspberry Pi` touchscreen
- Web Interface accessible with a web browser
- Text based interface based on MQTT

This section starts with a description of these three methods and afterwards discusses various aspects relevant to operations.

5.1 GUI - `Raspberry Pi` touchscreen interface to `tessie`

The GUI `Raspberry Pi` touchscreen interface is shown in Fig. 10.

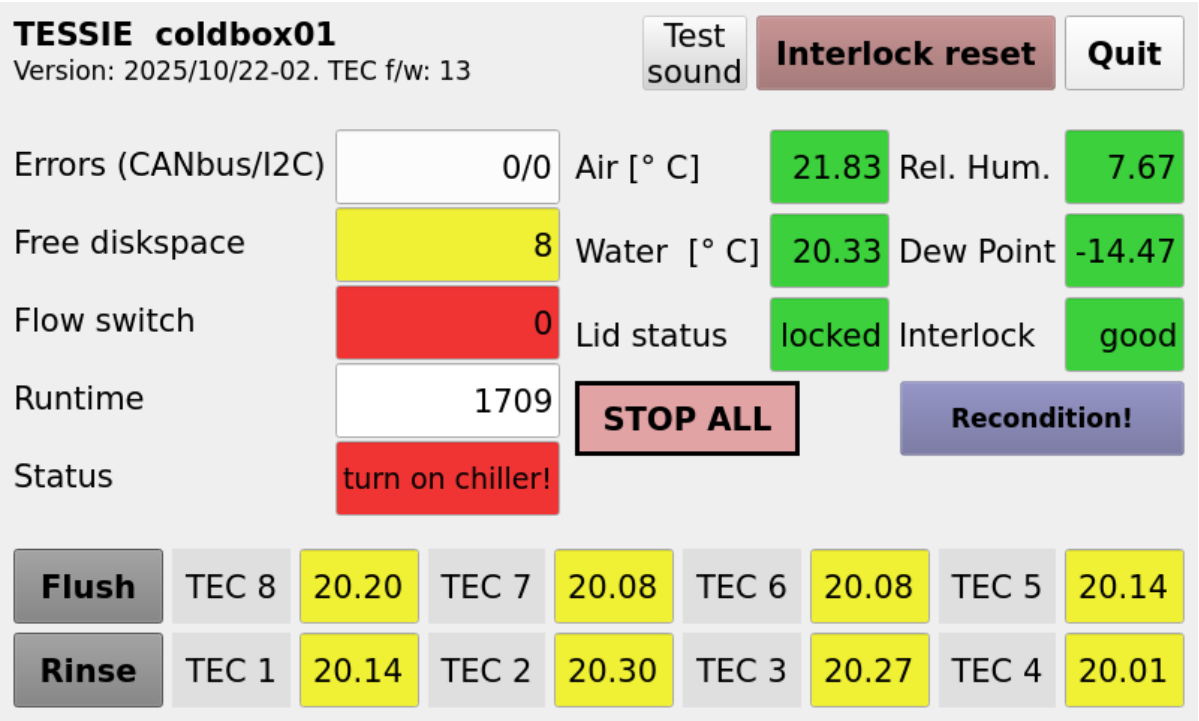


Figure 10: The GUI `Raspberry Pi` touchscreen interface to `tessie`. Here it tells you that no cooling liquid flow is measured with the flow switch and that you should turn on the chiller. You will not be able to turn on the `TECs` without first turning on the chiller.

This GUI starts up on the `Raspberry Pi` touch screen after booting and provides a minimalistic control surface to `tessie`. It allows for

- turning off and on the **"Flush"** and **"Rinse"** valves controlling the N2 flow. In normal operation, these buttons have a black font color. If nitrogen throttling is enabled, the font color will turn white.

- stopping all "dangerous" activites with the **"STOP ALL"** button that will turn off all `TECs`, turn on both "Flush" and "Rinse" valves, and disable the interlock (*i.e.*, switch it to `LOW`). It will make sure that the relative humidity in the box is minimal and that the power dissipated in the box is also minimal. This is the button you should hit if things go berserk.
- resetting the `tessie` interlock with the **"Interlock reset"** button. This is required to put `tessie` into a normal state again after a condition occurred where the interlock was disabled. In addition to resetting `tessie`, you very likely will have to do something additional for your HV and LV poweersupplies.
- testing the attached loudspeaker wit the **Test sound"** button. Note that the entire mp3 file will be played, and pressing this multiple times will not shorten the sound duration.
- quitting the `tessie` program.

All other controlling interactions with `tessie` should be initiated from the web interface, described in section 5.2.

The GUI displays the status of the system in three different blocks with multiple text displays with changing background colors.

System status The "Errors" field accumulates the total number of CANbus errors and I2C bus errors over `tessie`'s running time. It will turn (shortly) red in case of a new CANbus/I2c bus error, *cf.*, Fig. 11. If there is a persisting CANbus problem, it will stay red. The "Free diskspace" field shows the free storage space on the `Raspberry Pi` (`tessie` is writing considerable output, and if the storage space runs low you'll have to clean up and make space). This field should turn yellow with $< 2\,\mathrm{GB}$ free storage and it will turn red when $< 1\,\mathrm{GeV}$ is free. The "Flow switch" field shows the status of the the chiller flow, as (if) measured by the flow switch. "Runtime" shows the `tessie` running time in seconds. The "Status" box tries to indicate the status of the box in normal terms (note that a single line can not always reflect a complex situation. For instance if the chiller is off and the conditions inside the box require keeping the lid closed, only one requirement is reflected in this field).

Environmental status The "Air" field indicates the air temperature as measured with the SHT85, located at one end of the PCB inside the coldbox. Its background changes with the air temperature between blue, green, yellow, and red. The "Water" field provides the same information for the cooling water temperature inside the coldbox, with the same color coding as for the "Air" temperature field. The "Rel. Hum." field shows the air relative humidity in percent. It will turn green for values $< 5\%$. The "Dew Point" shows the calculated dew point and is green if both the air temperature is at least 2°C warmer than the dew point. The "Lid status" indicates whether the lid is locked or not and changes color accordingly.

`TEC` status The labels with the `TEC` numbers change the color depending whether the `TEC` is powered or not, in a similar fashion as the valve buttons color scheme. The fields provide the temperaturec [°C] of the PT-1000 on the `TEC` (*cf.* Fig. 6) with a changing background color scheme.

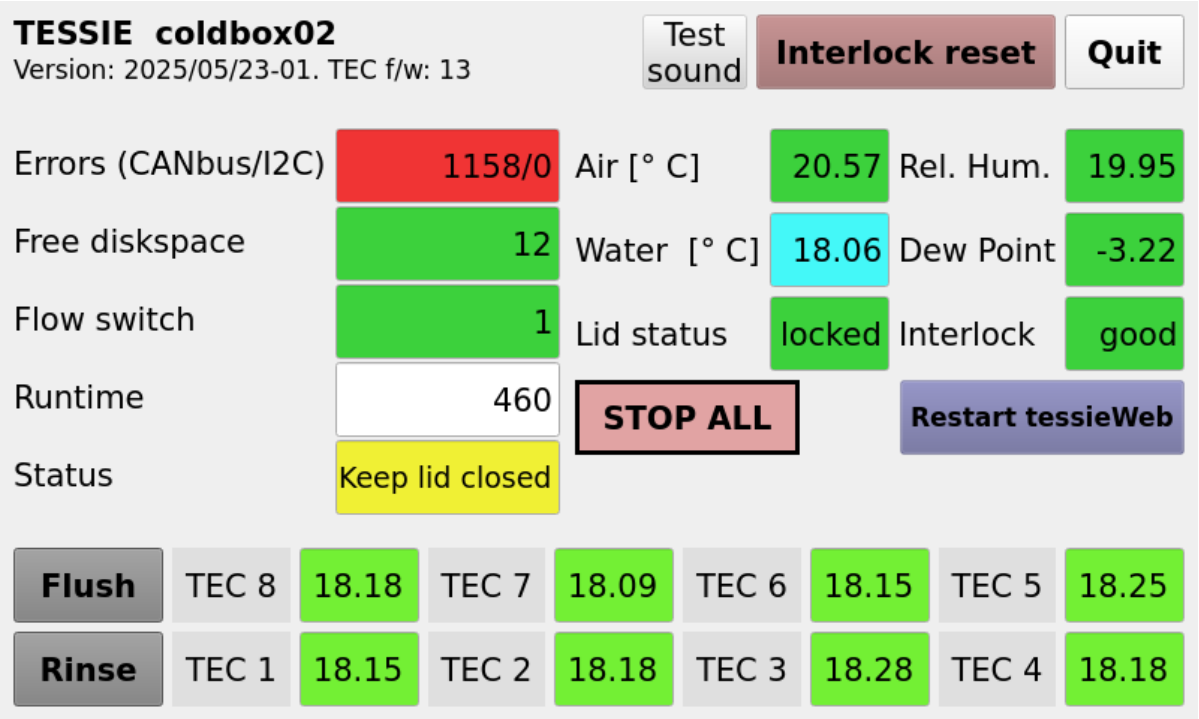


Figure 11: Illustration of the red background in case of many CANbus errors.

5.2 Web interface to `tessie`

The normal way to interact with `tessie` is via the web interface, shown in Fig. 12. The display is divided into three horizontal sections. At the top, the status and some controls are located: The blue part shows the Nitrogen control and indicates the numbering of the `TECs`. In addition there is a checkbox for N_2 throttling—this allows an automatic control that the relative humidity stays in the range of about 5–10% (this is helpful for the longevity of the RH/T sensor). The brownish part allows to `Load` and `Save` the `TEC` flash firmware (*e.g.*, if some values have been manually altered and you want to make these changes persistent). The dark green part shows environmental parameters inside the coldbox—note that the Water temperature is read out via the CAN bus, while the three other parameters are obtained via the I2C bus (it is, therefore, possible that the Water temperature is updating but the others are not, if there is a problem with the I2C bus or the SHT85 sensor). The light green section provides information relevant to the safety situation and mirrors the "traffic light" on the coldbox. The marron section at the right provides auxiliary information about errors on the buses, the run time and the software version.

The second horizontal section shows the MQTT messages sent to the ctrlTessie thread.

The third and largest section is the main control grid for the `TECs`. It allows to turn on/off the `TECs`, chose their operation mode (between constant voltage and PID-regulated temperature setting), set the `ControlVoltage_Set` or the `Temp_Set`. The temperature display provides essential information about the `TEC` status and should be monitored visually in case of experimental activity. Colored text background fields provide fast visual feedback. The other control possibilities are towards the bottom and concern the PID regulation parameters.

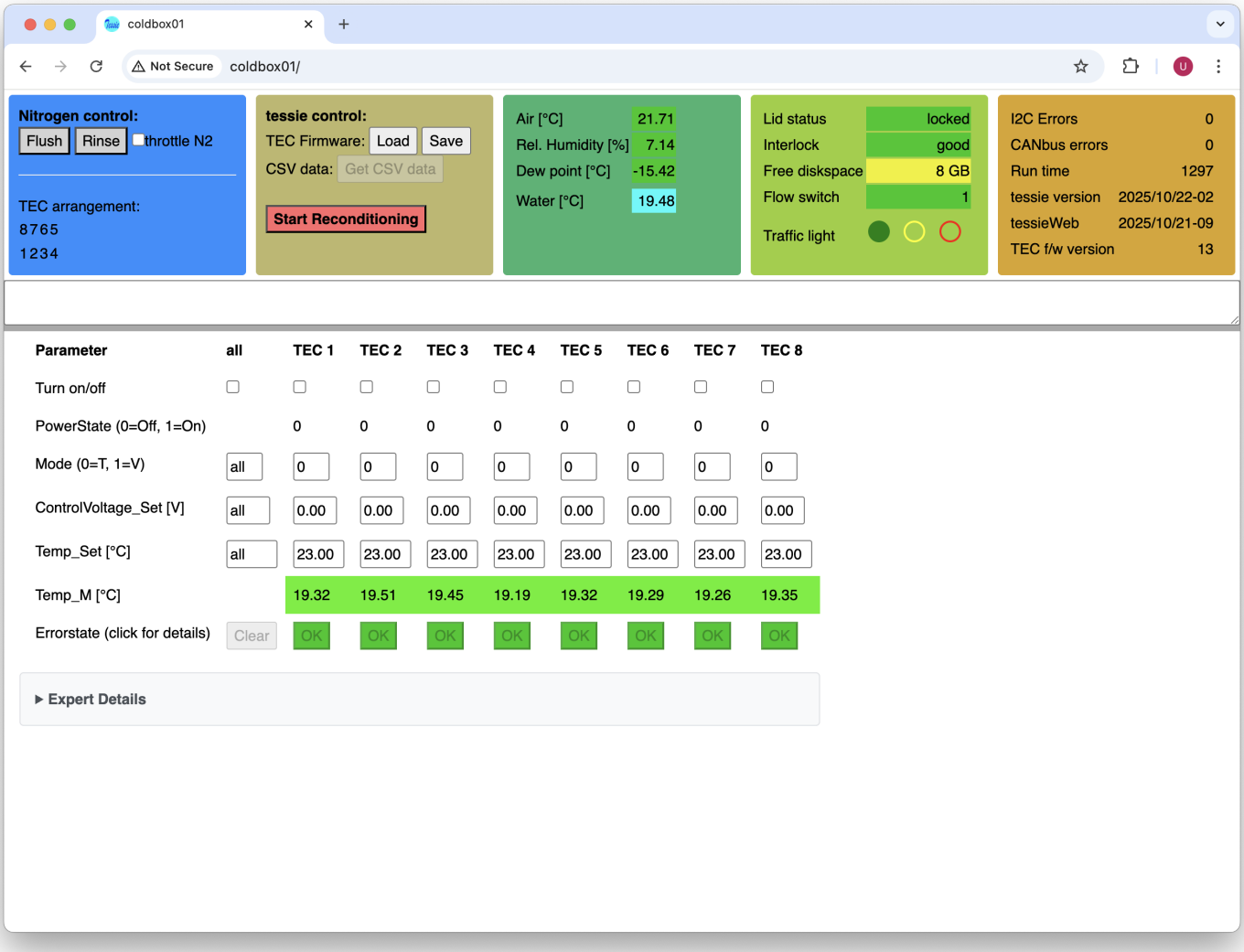


Figure 12: Web graphical interface to `tessie`.

The web interface is driven by a node.js webserver (tessieWeb) running on the `Raspberry Pi`. It can happen that the client/server connection is lost (because of shutting down the `Raspberry Pi`, connectivity issues, etc.). In that case the marron section turns red. It can also happen that `tessie` seemingly cannot be operated via the web interface (*e.g.*, hitting the `Flush` button will not result in the Flush valve changing state). In that case, you should close the browser tab and open a new one pointing to the `tessie` web address.

5.3 MQTT - direct text-based communication with `tessie`

Direct communication with `tessie` proceeds via two channels, either through the touchscreen GUI or through the MQTT protocol. The web GUI is not a *direct* communication channel,

since it is only an interface to MQTT messaging.

The interface with MQTT proceeds through two threads, `ctrlTessie` and `monTessie`. The former is used to communicate commands while the latter display regular monitoring information broadcast by `tessie`.
To see the traffic on a thread, you "subscribe" to it with (*e.g.*)

```
mosquitto_sub  -h coldbox03 -t "ctrlTessie"
```

To send commands over the `ctrlTessie` thread, you "publish" with (*e.g.*)

```
mosquitto_pub -h coldbox03 -t "ctrlTessie" -m "set valve0 on"
```

The `tessie` commands can be obtained with

```
mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "help"
```

This will result in

```
> ==================
> hostname: coldbox01
> thread:  ctrlTessie
> ==================
>
> Note: [tec {0|x}] can be before or after {get|set|cmd XXX}, e.g.
>       cmd Power_On tec 7
>       tec 7 cmd Power_Off
>
> Note: tec numbering is from 1 .. 8. tec 0 refers to all TECs.
>
> Note: Once heatOn has been called, tessie will only react to heatOff
>
> cmd messages:
> -------------
> cmd valve0
> cmd valve1
> cmd throttleN2On
> cmd throttleN2Off
> cmd heatOn
> cmd heatOff
> cmd startReconditioning
> [tec {0|x}] cmd Power_On
> [tec {0|x}] cmd Power_Off
> [tec {0|x}] cmd ClearError
> [tec {0|x}] cmd GetSWVersion
> [tec {0|x}] cmd SaveVariables
> [tec {0|x}] cmd LoadVariables
> [tec {0|x}] cmd Reboot
>
> messages to write information:
```

```
> ------------------------------
> [tec {0|x}] set Mode {0,1}
> [tec {0|x}] set ControlVoltage_Set 1.1
> [tec {0|x}] set PID_kp 1.1
> [tec {0|x}] set PID_ki 1.1
> [tec {0|x}] set PID_kd 1.1
> [tec {0|x}] set Temp_Set 1.1
> [tec {0|x}] set PID_Max 1.1
> [tec {0|x}] set PID_Min 1.1
> set valve0 {on|off}
> set valve1 {on|off}
>
> messages to obtain information:
> ------------------------------
> get Temp
> get RH
> get DP
> get throttleN2
> get valve0
> get valve1
> get vprobe[1-8]
> get vprobegnd
>
> get {monitoring|allMonTessie}
>
> [tec {0|x}] get Mode
> [tec {0|x}] get ControlVoltage_Set
> [tec {0|x}] get PID_kp
> [tec {0|x}] get PID_ki
> [tec {0|x}] get PID_kd
> [tec {0|x}] get Temp_Set
> [tec {0|x}] get PID_Max
> [tec {0|x}] get PID_Min
> [tec {0|x}] get Temp_W
> [tec {0|x}] get Temp_M
> [tec {0|x}] get Temp_Diff
> [tec {0|x}] get Peltier_U
> [tec {0|x}] get Peltier_I
> [tec {0|x}] get Peltier_R
> [tec {0|x}] get Peltier_P
> [tec {0|x}] get Supply_U
> [tec {0|x}] get Supply_I
> [tec {0|x}] get Supply_P
> [tec {0|x}] get PowerState
> [tec {0|x}] get Error
> [tec {0|x}] get Ref_U
```

```
> Tutorial for getting started:
> mosquitto_pub -h coldbox01 -t "ctrlTessie" -m " set valve0 on"
> mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "set valve1 on"
> mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "set ControlVoltage_Set 4.5"
> mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "cmd Power_On"
> mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "cmd Power_Off"
> mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "set ControlVoltage_Set 0.0"
> mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "set valve0 off"
> mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "set valve1 off"
```

You should carefully read the three Notes. Issuing a command like, for instance, `cmd tec 7 Power_off` will not work, has not been promised to work, and there are no intentions to change this behavior.

The `monTessie` thread contains messages according the following format:

- `VAR = G1, Y0, R0, L1, I1, D17, F1, T0, H0`
  contains various information: The "traffic light" status Green, Yellow, Red, where 0 (1) means off (on). L indicates the lid status, I the interlock (0 = broken interlock, *i.e.*, an unsafe state), D diskspace (GB, rounded to the next integer number), F the flowswitch status (0 = no flow detected,*i.e.*, the chiller is off or in standby mode). T indicates whether N2 throttling is active. H indicates the "heater" status of the HYT-223. If the integer number following H is larger than zero, the coldbox is in a reconditioning state and will ignore virtually all MQTT commands (except for `cmd heatOff`).

- `Env = 20.9091, 18.1523, 26.6148, 1.00364, 0, 40, 285631, 0, 0`
  contains environmental data: air temperature, cooling liquid temperature, relative humidity in per cent, dew point, CAN bus error counter, I2C error counter, run time in seconds, N2 flush valve status, and N2 rinse valve status. All temperatures in degree Celsius.

- In addition, the `TEC` registers (cf. 2) are broadcast every 10 seconds with the values for all `TECs` separated by commas. If these values leave a "safety" window, then they are printed within one second. The "safety" window is set to 0.1 for all values except for `Supply_I` (1.0), `Supply_P` (1.0), and `Error` (any change).

  You can invoke the broadcast of the complete set of information, *i.e.*, including the full `TEC` information, using the MQTT command `mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "get monitoring"` (in case you preferred you could replace the keyword `"monitoring"` with `"allMonTessie"`).

Furthermore, alarm and warning messages are broadcast over these channels (cf. 5.8).

5.4 Safe operations limits

To ensure the safety of the coldbox equipment, the `TEC` controllers and the FRAS relais require a "heartbeat" command from `tessie` at periodic intervals (3 seconds for both; `tessie` sends the command every second). If that expected "heartbeat" signal is not registered, these components stop operating (*e.g.*, the `TECs` are shut off and no longer provide cooling for the modules).

In addition, `tessie` continuously monitors environmental parameters to ensure a safe operation of the coldbox in case of operator error. Table 4 provides a summary of the safe operation parameters. An alarm is raised if `tessie` registers a violation, *cf.* Section 5.8. If

Table 4: Safe operations parameters monitored by `tessie`. In this table "module temperature" indicates the PT1000 temperature reading mounted on the Peltier module.

Parameter	**min** [deg]	**max** [deg]
box air temperature	n/a	40
water temperature	n/a	30
module temperature	n/a	30
difference between box air temperature and dew point	2	n/a
difference between module temperature and dew point	2	n/a

any temperature (box air, water, modules, copper block) exceeds 40°C, `tessie` will turn off all `TECs` and trigger the interlock procedure. In addition, it will ensure maximum N2 flow by opening the `Rinse` and `Flush` valves.

5.5 First steps

To start operations on (*e.g.*) coldbox03, the following instructions might be helpful (in case you need them).

Turn on Load the module(s) into slot(s). Close the lid and ensure that it is properly locked.

- Turn on the chiller and make sure it is not in standby mode.
- On any computer connected to the same network as your coldbox03 point a browser to `http://coldbox03`. All manual interactions described below are done on this web GUI.
- Reduce the relative humidity inside the coldbox by opening the N2 valves, initially both `Flush` and `Rinse`. Once the dew point is sufficiently low, you may turn off the `Flush` flow. Note: it is a matter of your hardware (screw) settings whether `Rinse` is sufficient to keep the relative humidity low.
- Set the voltage on the `TECs` to be operated (a good starting value is 3 V) by entering the value into the boxed field.
- Turn on the `TEC` by clicking the corresponding checkbox.
- Observe how the module temperature drops.

It is now safe to power (LV/HV) the module and run electrical tests.

Turn off Before turning off the coldbox, turn off the power (LV/HV) on the modules in the coldbox and then proceed according to the following list

- Turn off the `TECs` by clicking the checkbox.

- Turn on the N2 flow `Flush` to faster warm up the coldbox and keep the relative humidity low.
- Wait until the temperature inside the box is close to the room temperature.
- Turn off the N2 flow, both `Flush` and `Rinse`.

It is now safe to open the lid and remove the module(s).

5.6 Direct readout of probe card

In principle, the probe card [7] readout should be handled by higher-level software. However, it is also possible to do a direct readout of the probe card with MQTT. In one terminal subscribe to the `ctrlTessie` thread with

```
mosquitto_sub -h coldbox01 -t "ctrlTessie"
```

In a second terminal, assuming that you have a probe card at slot 8, issue the read command

```
mosquitto_pub -h coldbox01 -t "ctrlTessie" -m "get vprobe8"
```

You will receive, in the first terminal (the one where you have subscribed to the `ctrlTessie` thread), two lines with the following format:

```
get vprobe8
vprobe8 = 0.0073,0.064,0.0042,0.0015,0.0045,0.0016,0.0041,0.0012,0.0034,0.0009
```

The first line repeats the command given in the second terminal and then the result of that readback is provided. In the printout above we have suppressed digits to an avoid excessive line length. Normally, seven significant digits are provided for all measurements. The interpretation of the numbers is as follows

```
probePosition = vin  voffs  vdda0  vddd0  vdda1  vddd1  vdda2 vddd2  vdda3  vddd3
```

corresponding to the input voltage, offset, and the read digital and analog voltages of the 4 chips on the module. In case no probe card is installed at the queried position (7 in the example below), you should receive the following response:

```
get vprobe7
vprobe7 = -999
```

If you do not receive any response (beyond the command repeat), power-cycle the coldbox: push the green button below the touch-screen, wait until the `Raspberry Pi` has turned off, turn off the red switch on the left side of the front panel, wait 20 seconds, and turn on all switches in the reverse order.

For debugging purposes you can also obtain the ground levels (normally subtracted in the above printout) by issuing, directly after a probe card readout, the following command:

```
get vprobegnd
vprobegnd = 0,0.00025264,5.0528e-05,0,5.0528e-05
```

The first line repeats the command given in the second terminal and then the result of that readback is provided. The interpretation of the numbers is as follows

```
gnd3 gnd6 gnd11 gnd14 gnd26
```

corresponding to the labeling in Fig. 5.

5.7 Traffic Lights

Three lights are used for a visual display of the operations status of the coldbox, cf. Table 5.

Table 5: "Traffic light" display of the `tessie` status.

Color	State	Meaning
Green	on	Safe to open the box, all environmental parameters in safe range
Green	off	Not safe to open the box
Yellow	on	At least one `TEC` turned on (e.g., during a test)
Yellow	blinking	No `TEC` turned on, but not safe to open the box.
Yellow	off	No `TEC` turned on
Red	on	Alarm active (see section 5.8)
Red	off	No alarm active

During reconditioning 5.11 all lights (green, yellow, and red) of the traffic light will light up simultaneously.

5.8 Alarm channels

`tessie` raises an alarm in case operational issues require human intervention. The alarm is raised as soon as `tessie` observes a measurement violating the safe operation region. The alarms are propagated via various means

- the alarm condition is broadcast to the `ctrlTessie` and `monTessie` MQTT channels
- an alarming sound is played through an attached loudspeaker inside the coldbox. All connected web GUIs will also play the sound in case the user has given the browser (tab) permission to play audio.
- the "traffic" light display of the coldbox displays a constant red light

In addition to the alarms raised, `tessie` also issues warnings for I2C and CAN bus errors. These warnings are accumulated as counters in the GUIs (both the web GUI and the GUI running on the coldbox touch screen). In addition, an warning sound is played.

5.9 Modes for cooling the `TECs`

There are two modes for cooling the `TECs`, *cf.* Figs. 8 and 9. The setting `Mode = 0` enables the `TEC` controller's PID control algorithm to set the `TEC` temperature to the value specified in the `Temp_Set` value, while the setting `Mode = 1` applies a fixed voltage `ControlVoltage_Set` to the `TECs` (in the range $0 <$ `ControlVoltage_Set` < 11 V). Table 6 provides an approximate translation between the setting of `ControlVoltage_Set` and the temperature measured on the PT1000 attached to the `TEC`, *cf.* Fig. 6.

5.10 Temperatures on the `TEC` and module

The temperature measured with an NTC on the chip of a digital module has been studied and is compared to the temperatures on the module `TEC`. The accuracy of this NTC is of the

Table 6: Approximate translation between `ControlVoltage_Set` and PT1000 temperature `Temperature_M`, with the temperature measured on a powered HDI. Note: this translation depends on the cooling liquid temperature!

`ControlVoltage_Set` [V]	`Temperature_M` [°C]	**HDI temperature [°C]**
1	20	35–38
2	15	30–35
3	7	20–25
4	0	15–20
5	-4	12–17
FIXME	add	more

order a few degrees, as estimated from the difference between the temperatures measured in chips 14 and 15 as shown in Fig. 13.
The NTC readout was obtained with the Ph2-ACF software (version v4-15) using the command

```
CMSITminiDAQ -f CMSIT_RD53B.xml -c ntc
```

It should be noted that the `TEC` temperature differs by about -15°C from the chip temperature. This may be similar to the situation in the experiment, where the chips will also not be at the same temperature as the cooling.

5.11 Reconditioning the HYT-223 RH/T sensor

The HYT-223 RH/T sensor allows in-situ reconditioning which is required from time to time. To quote the technical documentation [8]: "HYT223 contains a microheating structure which allows for thermal reconditioning. A reconditioning cycle is recommended in challenging atmospheres and conditions. Length and interval required depend on the application environment. A possible reconditioning setting is heating the module with 8 to 9 V and 700mW power for 10 minutes every 24 hours."

The best indication that the HYT-223 requires reconditioning is that the relative humidity reading no longer goes down to very low values if the box is flushed with N2. It will stay at a few percent. This will make it impossible to operate the coldbox at very low temperatures because `tessie` will complain about the air temperature being too close to the dewpoint.
A dedicated small PCB was developed to allow this in-situ reconditioning through the CAN bus. The reconditioning process can be initiated through the touchscreen button or via the web interface. Both will trigger the same behavior:

- both N2 valves will be closed to allow less cooling at the RH/T sensor
- the heater circuit will be activated until the target (air) temperature of 90°C is exceeded. (The air temperature is used because no direct RH sensor temperature reading is available.)
- once (and only once) 90°C has been reached, heating continues for 10 minutes.

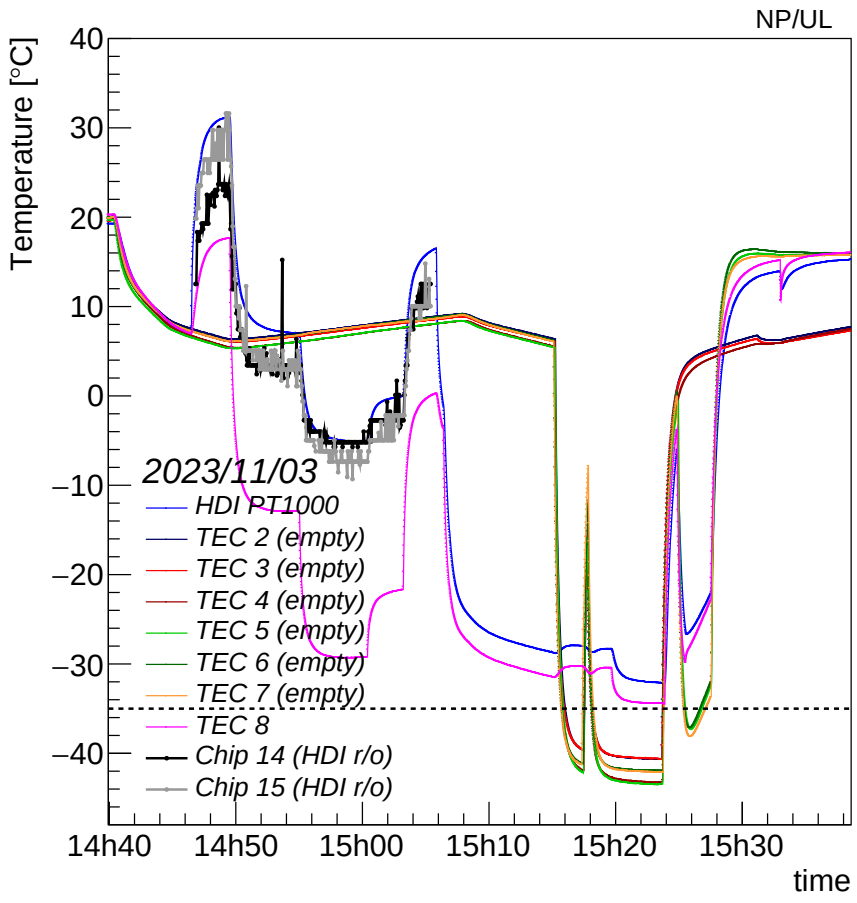


Figure 13: Temperature measurements with a digital module. Note that there is a difference of about 15°C between the chip temperature and the module `TEC` temperature with a powered module (there would be no HDI temperature readout without powering). The short-dashed line is at $T = -35$°C, to guide the eye.

- after this wait time of 10 minutes, the heating circuit is turned off and both valves are opened (to speed up the cool down). After three minutes, `tessie` should return to its normal state. The Air temperature will still be a bit elevated (between approximately 25–30°C), but this is in a safe region and `tessie` will not complain.

During reconditioning all lights (green, yellow, and red) of the "traffic" light will light up simultaneously.
It is also possible to turn on and off the heater manually via MQTT commands, cf. 5.3. Note that once the heater is turned on, `tessie` will ignore all MQTT commands except `cmd heatOff`.

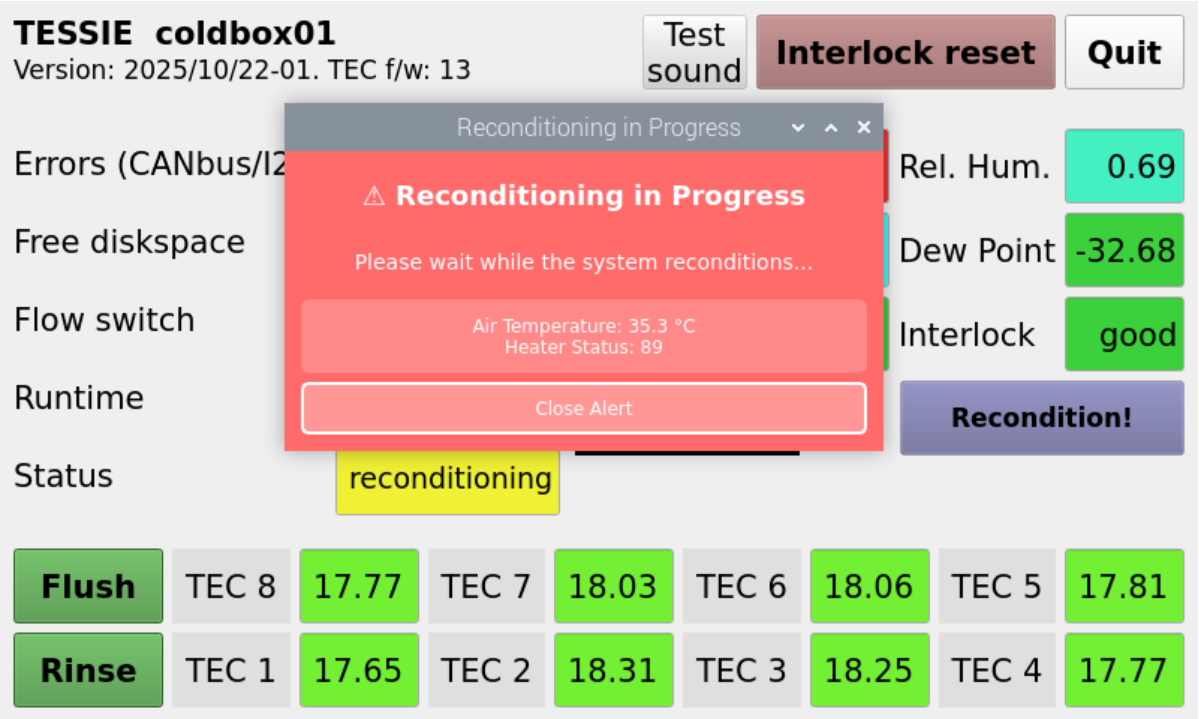


Figure 14: Display on touchscreen while reconditioning is in progress.

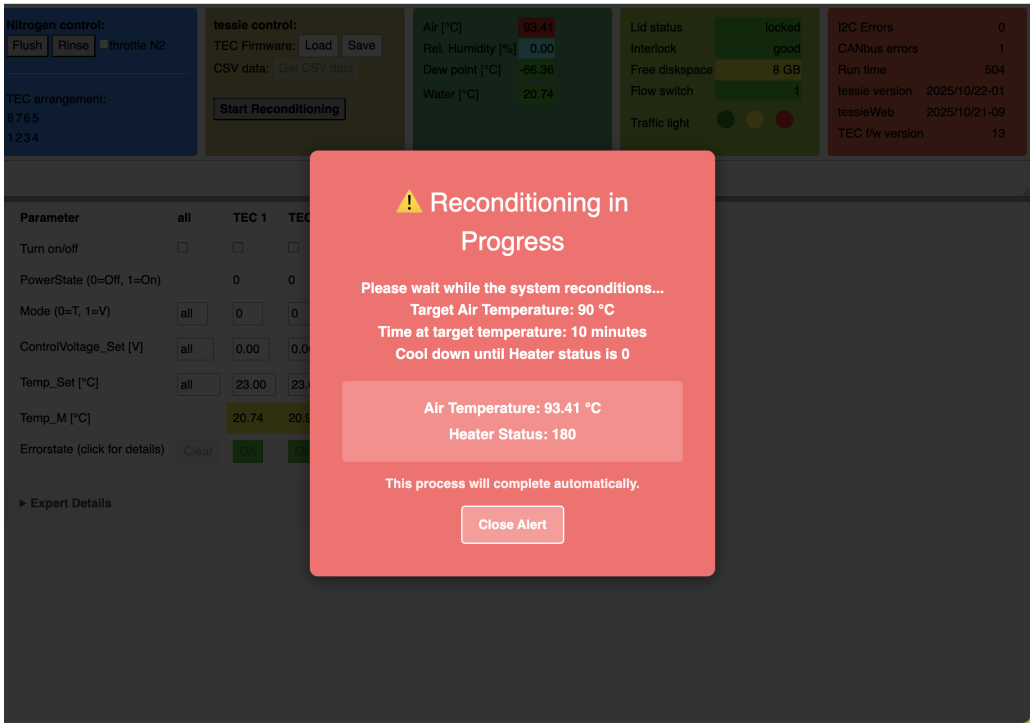


Figure 15: Display on web graphical interface while reconditioning is in progress.

6 Frequently asked questions

This sections aims to provide help for errors and issues that may be encountered with the coldbox and/or `tessie`, during operation or installation.

6.1 Operations

Many of the issues discussed here are indicated by error, warning, and/or alarm messages being broadcast to the MQTT channels `ctrlTessie` and `monTessie`, the red light of the traffic light turning on, sound messages being played, and other highlighting background colors in the `tessie` GUI on the coldbox touchscreen (cf. Fig. 10).

6.1.1 Do's and don'ts

What are the most important aspects when operating the coldbox?
Answer: The Do's are

- Turn on the chiller
- Ensure the chiller is running
- Ensure that a loudspeaker is connected (play the test sound via the GUI button or see below in subsection 6.2.4)
- Ensure that the browser tab is unmuted
- Once a day, recondition the HYT-223 RH/T sensor. It is recommended to remove all modules from coldbox for this process.

Do not

- operate the coldbox without the chiller running
- be impatient with the web UI or the GUI. Other web UI instances might be connected and need to be synchronized.
- ignore error messages or alarms
- disconnect the loudspeaker or lower its volume
- mute the browser tab with the web UI
- leave any module in the coldbox while reconditioning the HYT-223

6.1.2 `tessie` does not start

It can happen that after a reboot `tessie` cannot display the GUI on the Raspberry Pi touchscreen. This happens possibly because of a race conditions after a fresh Raspberry Pi powerup. The quick-n-dirty fix: Simply reboot the Raspberry Pi, either through the power button or with `sudo shutdown -r now` on the command line. That should work.

The better approach is to add the following text to the file `/lib/systemd/system/tessie.service` to have the following content:

```
[Unit]
Description=tessie
After=network.target

[Service]
Type=idle
User=pi
Group=staff
Environment="XAUTHORITY=/home/pi/.Xauthority"
Environment="DISPLAY=:0"
WorkingDirectory=/home/pi/tessie/src
ExecStartPre=/home/pi/tessie/resetCAN.sh
ExecStart=/home/pi/tessie/src/tessie -f
StandardOutput=inherit
StandardError=inherit
Restart=on-failure
RestartSec=2s

[Install]
WantedBy=graphical.target
```

Note the two "Restart" lines.

6.1.3 An alarm goes off if I plug in a module and close the lid

A possible reason for this condition could be that you turned on the chiller, opened the lid to connect a module, closed the lid and then did *not* turn on the N2 flow. The chiller water will cool the Peltier element to whatever temperature you set the water temperature to and this temperature may well be below the dew point (especially in Summer, when the ambient air humidity is high). This will trigger the alarm `module ... too close to dew point ...`
Solution: Turn of N2 flow (either flush or rinse or both).

6.1.4 Alarm module temperature exceeds `SAFETY_MAXTEMPM` (single module)

The reason for this condition is likely that you powered a module without turning on the corresponding `TEC`.
Solution: Turn on the corresponding `TEC` cooling or power-off the module.

6.1.5 Alarm module temperature exceeds `SAFETY_MAXTEMPM` (single module)

The reason for this condition is likely that you turned on one `TEC`, and then off again, without turning on the chiller (or there is no flow in the water). This can warm the entire side (1–4 or 5–8) of the coldbox and, because of variations between the different `TEC` temperature sensing, a single module can go above the limit before the others do so. It can take a while to get into this state. Note: If, from previous interactions with the coldbox, the water is still below 20°C, `tessie` will allow turning on the chiller despite it not running.
Solution: Turn on the chiller.

6.1.6 Alarm module temperature exceeds SAFETY_MAXTEMPM (many modules)

The reason for this condition is likely that you turned on (some or all) TEC without turning on the chiller. It takes a few minutes to get into this state. Note: If, from previous interactions with the coldbox, the water is still below 20°C, tessie will allow turning on the chiller despite it not running.
Solution: Turn on the chiller.

6.1.7 The CANbus shows many errors

The reason for this problem is likely some electrical condition affecting the CANbus due to operator manipulations or to missing components (lid sensor) or to bridged components (*e.g.* replacing the lid sensor with a resistor).
Solution: Restart tessie either by power-cycling the coldbox with the central power button or by logging in from a remote computer and doing

```
ssh coldbox
sudo systemctl restart tessie
```

6.1.8 Persistent issue with a single TEC

The "issue" could be CAN bus errors (showing up as, e.g. temperature readings of −999°C) or other examples. This could be due to a blown fuse on the TEC.
Solution: Replace the fuse on the TEC, cf. Fig. 4.

6.1.9 Where is the printout/output/logfile/CSV file from tessie?

For a better understanding of possible issues with tessie sometimes you need access to the printout, especially if you have looked into the code.
Solution: You can see the printout with journalctl.

```
journalctl -u tessie -b
journalctl --since "24 hour ago" -u tessie
journalctl --follow -u tessie
```

The option -b yields all messages since the last boot, the second example restricts this to the specified time period, and the third example shows continuous output.

There are logfiles in the running directory of tessie, normally in /home/pi/tessie/src. The running instance of tessie is writing to tessie.txt and all previous logfiles are kept with timestamps attached (*e.g.*, tessie.txt.20241004_10:22:25.366)

In addition to the logfiles, a CSV file (comma-separated values), tessie.csv, is filled by tessie. The values in that file correspond to

```
date,T-air,RH,DP,T-water,powerState[8],Vset[8],Tset[8],T-module[8]
```

where date is the date and timestamp of the following values, T-air, RH, DP are the temperature, relative humidity, and dew point of the air inside the coldbox, respectively. T-water provides the cooling liquid temperature. The remainder of the line come in groups of 8 values for the 8 TECs: powerState indicates whether the TECs are turned on (enabled), Vset provides the ControlVoltage_Set in Volts, Tset shows the programmed Temp_Set in degrees Centigrade, and finally T-module provides the module temperatures.

6.1.10 The N2 flow valves are not working, but no CANbus errors appear

One possibility is that a connector on the FRAS has a shaky contact, *cf.* Fig. 16. This can be indicated by no LED on the FRAS being lighted.

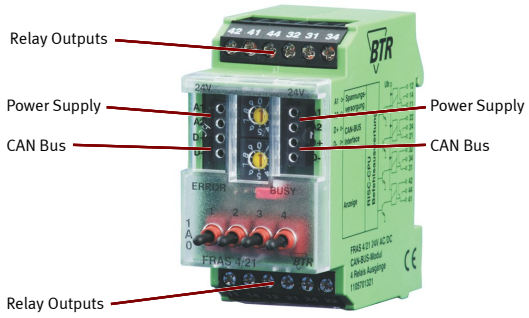


Figure 16: Photo of the FRAS. The connectors for the sockets (A1/A2/D−/D+, labeled with "Power Supply" and "CAN Bus") supplied by the manufacturer with this device, are prone to shaky contacts and should be secured by some (improvised) means.

Solution: Open the box (take off the lid for that purpose) and ensure a proper contact of the connector, *e.g.*, by tying it down with a cable tie.

6.2 Installation

6.2.1 The touchscreen is white

This is an indication that the flatband cable connector is not properly inserted into its socket. The fact that initially the touchscreen showed a colorful pattern is no contra-indication, that simply indicates that it has power.
Solution: Unplug the flatband cable and insert once again.

6.2.2 The touchscreen gradually turns white and stays white.

This can happen during the first boot after inserting a fresh SD card and normally changes after repeated reboots (done automatically by the system). If it persists, then most likely you used an image that is not the recommended version.
Solution: Burn a new iso image using the following img. This issue was posted to stackexchange (here), but it remains unresolved.

6.2.3 Server certificate verification ...

... error messages when trying to run `apt-get`. This can happen if you have not set the correct time and date.
Solution: Set the date and time using the following command (replacing the example values with the correct value)

```
sudo date -s "Wed Apr 17 2024 10:00:00"
```

6.2.4 How to test the USB loudspeaker and `tessie` audio alarms

The easiest (and still reasonably safe) way is to make sure that the coldbox is empty (no module loaded!) set one `TEC` to `Mode = 1` and `ControlVoltage_Set` to 3 V. (This should be done via the web interface.) This will turn on the `TEC`, which will lower the (Peltier module) temperature below the dew point. Once the temperature is closer than 2°C to the dew point (cf. Table 4), the alarm will go off. You will see this visually on the GUI (`TEC` temperature field indicator flashing red) and you should hear the alarm. At this point, you can turn off the `TEC` and all should be well again. Note: You should hear the alarm from two sources! (1) the USB loudspeaker in the coldbox should emit the alarm, and (2) the web browser should sound the alarm on the computer. If one of these two sources is not present, fix it.

Another possibility is to hit the "Test sound" button on the GUI. This button is not available on the web UI. Currently, it tests only the coldbox loudspeaker and will not sound the alarms in a web browser.
Note: This will play the entire mp3 file for a duration of about 50 seconds. Hitting the button again will not stop the playing, but will prolong the playing.

References

[1] Urs Langenegger, "tessie". https://github.com/ursl/tessie.

[2] Steve Corrigan, "Introduction to the Controller Area Network (CAN)". https://www.ti.com/lit/an/sloa101b/sloa101b.pdf.

[3] Jonathan Valdez, Jared Becker, "Understanding the I2C Bus". https://www.ti.com/lit/an/slva704/slva704.pdf.

[4] MQTT.org, "MQTT: The Standard for IoT Messaging". https://mqtt.org/.

[5] Sensirion - The sensor company, "Datasheet SHT85 - Humidity and Temperature Sensor". https://sensirion.com/media/documents/4B40CEF3/61642381/Sensirion_Humidity_Sensors_SHT85_Datasheet.pdf.

[6] Noah Piqué, "Coldbox Development at PSI". CMS Inner Tracker Modules Meeting, November 30, 2023 (https://indico.cern.ch/event/1349094/).

[7] Beat Meier, et al., "Voltage probecard". https://psi-lab.docs.cern.ch/coldbox/vprobe/ https://gitlab.cern.ch/psi/moduleprobe/.

[8] IST AG, Innovative Sensor Technology, "Data sheet Humidity module HYT223 with integrated heater". https://www.ist-ag.com/sites/default/files/downloads/DHHeatedHYT_E.pdf.

[9] Wilke Technology, "CAN Output FRAS4 4x Relais Out". https://wilke.de/fileadmin/templates/daten/DATA_Sheet_DV-CANFRAS4-01_EN.pdf.

[10] Noah Piqué, "Coldbox assembly manual". https://psi-lab.docs.cern.ch/coldbox/assembly/.

[11] Raspberry Pi OS, "Raspberry Pi Imager". https://www.raspberrypi.com/software/.

[12] FEASER, "OpenBLT Bootloader". https://www.feaser.com/openblt/doku.php?id=homepage.