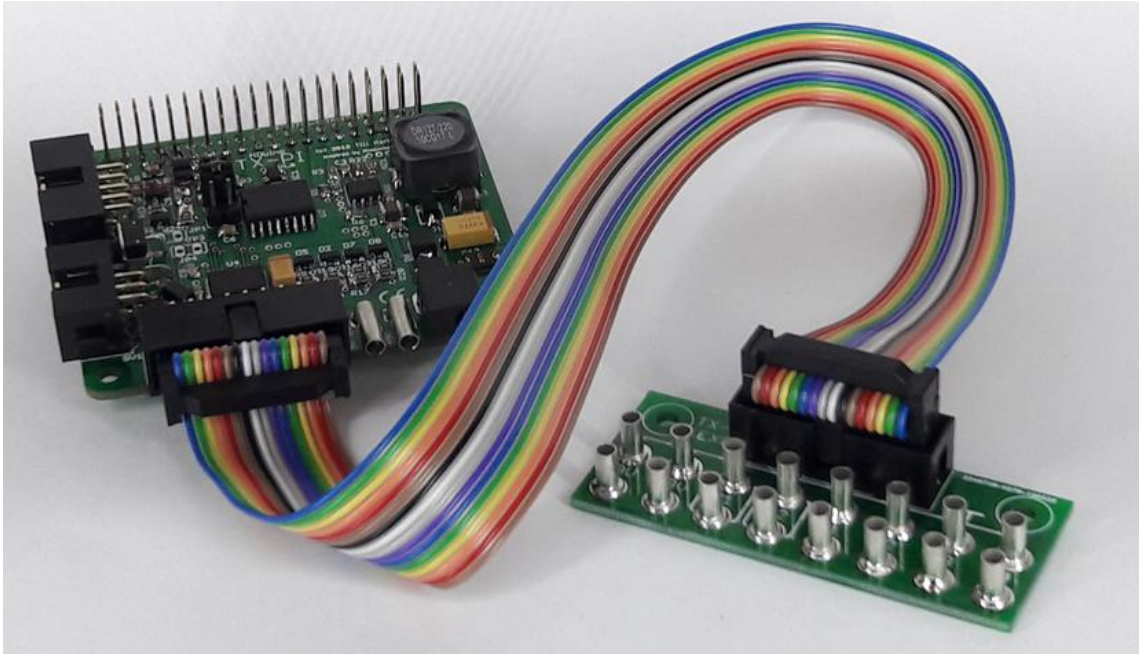
TX Pi HAT/ft HAT

a fischertechnik compatible HAT for the Raspberry Pi



Manual

Dr.-Ing. Till Harbaum

June 11, 2020

© 2019 Dr.-Ing. Till Harbaum <till@harbaum.org>

Project homepage: http://tx-pi.de
Forum: https://forum.ftcommunity.de/

Contents

Chapter 1

The TX Pi project

The TX Pi HAT is part of the TX Pi project located under http://tx-pi.de. The TX Pi project aims to integrate the Raspberry Pi with the fischertechnik construction toy. To achieve this goal the TX Pi project provides:

1. Hardware recommendations and custom hardware designs like the TX Pi HAT for the electrical integration
2. Software configurations for the Raspberry Pi suitable to control fischertechnik devices for the software integration
3. Case designs for all major components for the mechanical integration.

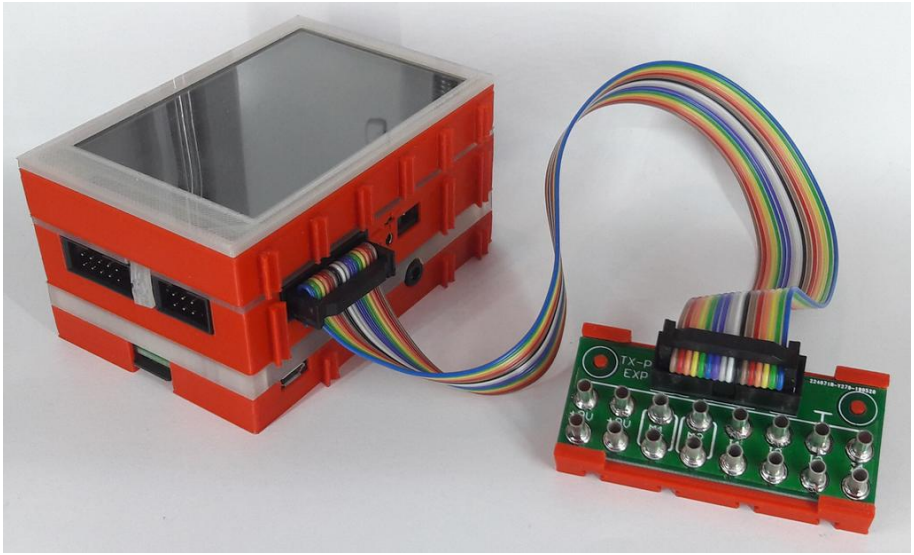


Figure 1.1: A complete TX Pi setup consisting of the Raspberry Pi3, the TX Pi HAT and a 3.5" display

The TX-Pi project is based in the Raspberry Pi. It also includes recommendations for hardware addons to use as well as custom hardware addons like the TX Pi HAT.

It also provides case designs for 3D printed cases to protect the various hardware components as well as making them mechanically compatible with the fischertechnik system.

1.1 Raspberry Pi

The Raspberry Pi is a full featured credit card sized Linux computer sold at a very low price. It's primarily targeted at the educational market but due to its low price and simple usage it has been widely adopted by the maker community.

The Raspberry Pi is being sold since 2012 and has reached its fourth version in 2019. The most popular versions of the Raspberry Pi include a multi core gigahertz arm CPU, up to 4GB RAM, several USB connectors incl. USB 3.0, Ethernet,

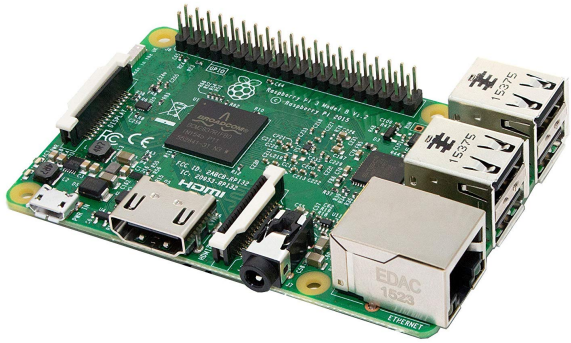
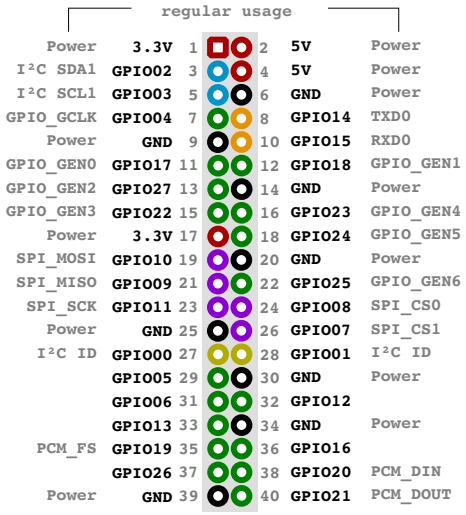


Figure 1.2: The Raspberry Pi version 3

WiFi and Bluetooth and HDMI. They can be powered from ubiquitous USB power supplies and are very affordable.

1.1.1 GPIO port

Unlike most regular computers the Raspberry Pi includes a so-called 40 pin GPIO header (General Purpose Input and Output) as depicted in figure 1.3 which allows to connect arbitrary hardware ranging from various sensors over small touchscreen displays to powerful motor drivers and similar.



(a) GPIO pinout

(b) A display HAT

Figure 1.3: GPIO extension on the Raspberry Pi

The GPIO port allows to extend the Raspberry Pi using so-called HAT's. HAT stands for "Hardware attached on top" and describes hardware devices which have the mechanical footprint of the Raspberry Pi and which can be plugged directly into the GPIO connector and which then sit on top of the Raspberry Pi. Some of these boards can even be stacked so more than one HAT can be connected at a time. The ability to use more than one HAT requires careful selection of the HATs as the connections on the GPIO port can usually not be shared between several HATs. HATs being used simultaneously thus usually need to used separate signals of the GPIO port.

1.1.2 Raspberry Pi compared to fischertechnik TXT controller

Since 2014 fischertechnik is selling the TXT controller, a Linux based computing device with built-in 2.4" touchscreen and fischertechnik compatible power supply, inputs and outputs. Lego is also selling the similar EV3 controller for their system.

The following table compares the EV3, the TXT, the Raspberry Pi4 and the TX Pi[1].

	EV3	TXT	Raspberry Pi4	TX Pi
CPU type	TI-AM1808-Sitara	TI-AM3359-Sitara	Broadcom BCM2711	
Core types	32 bit Cortex A8	32 bit Cortex A8	64 bit Cortex A72	
No of CPU Cores	1	1	4	
CPU clock	300 MHz	600 MHz	1500 MHz	
Memory type	DDR2	DDR3	DDR4	
Memory size	64 MB	256 MB	4 GB	
On board flash	16 MB	128 MB	-	
USB host	1 * USB 2.0	1 * USB 2.0	2 * USB 2.0, 2 * USB 3.0	
USB device	1 * USB 2.0	1 * USB 2.0	-	
WiFi	-	SDIO: 802.11n 2.4 GHz	SDIO: 802.11ac (2.4 & 5 GHz, 1x1)	
Bluetooth	BT 4.0	BT 4.1 incl. BLE	BT 5.0 incl. BLE	
Ethernet	-	-	Gigabit Ethernet	
Display	178×128 mono	2.4" 240x320 TFT	-	3.5" 320x480 TFT
Touchscreen controller	-	software	-	XPT2046
Display controller	-	ILI9341	-	ILI9486/ILI9488
Power supply	6*AA (LR6)	ft 9V= 2.5A	USB-C 5V 3A	ft 9V= 2.5A
I²C	Lego custom	3.3V	3.3V	3.3V & 5V
9V analogue inputs	3	8	-	-
9V digital inputs	-	4	-	4
9V motor outputs	3	4	-	2
Price est.	200€	200€	35€	100€

The TXT is most useful when it comes to control complex fischertechnik models and when focus lies on the interconnection with many fischertechnik components.

The Raspberry Pi on the other hand has a huge advantage with respect to computing power. Whenever the focus lies on computing the Raspberry Pi is the right choice. If more fischertechnik compatible connections are required in a Raspberry Pi setup then one or more ftDuino's[2] can be attached to the Raspberry Pi via USB or I²C.

1.1.3 fischertechnik compatible cases for the Raspberry Pi

A case for the Raspberry Pi is provided at https://github.com/ftCommunity/tx-pi/tree/master/cases. Various case designs are available varying with respect to the openings available for use with cameras and displays or extender cables.

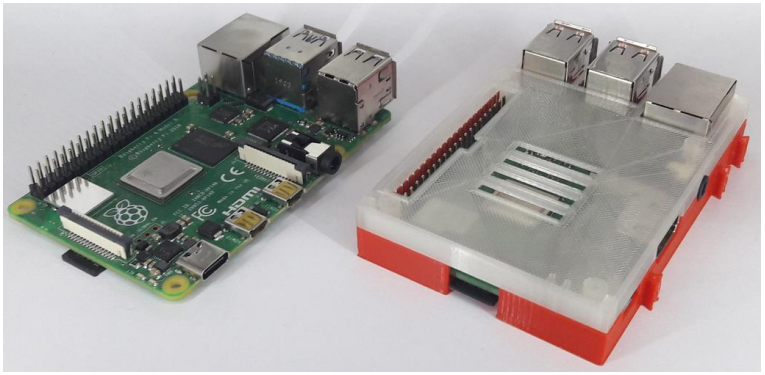


Figure 1.4: A case for the Raspberry Pi

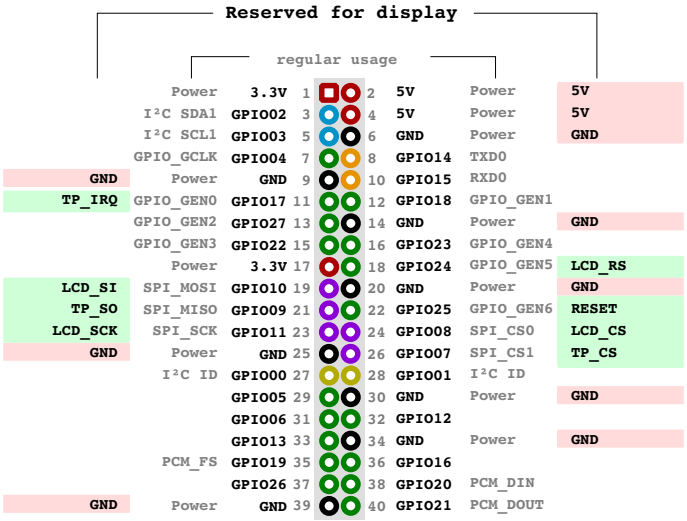
Image 1.4 shows a Raspberry Pi4 without case and a Raspberry Pi3 with case. All Raspberry Pi's from the Raspberry Pi B+ to the Raspberry Pi3 use the same mechanical layout and share the same case. The Raspberry Pi4 has slightly different connectors and the Ethernet and USB connectors have been swapped. Thus a different case is provided for this latest Raspberry Pi.

[1] TX Pi = Raspberry Pi4 + touch display + TX Pi HAT
[2] ftDuino, http://ftduino.de

1.2 Display screen

The TX Pi does not necessarily require a display. Like the regular Raspberry Pi it may also be used with a regular screen connected to the HDMI output or it may even be used headless with the option to use a remote screen setup like VNC.

But the standard use case ecpects a screen. Nearly all screens available for the Raspberry Pi can also be made to work with the TX Pi. But there are several limitations when being combined with the TX Pi HAT.



(a) GPIO pins reserved for display use

(b) Bottom of Waveshare 3.5" display

Figure 1.5: GPIO usage with display

As shown in figure 1.5 several pins of the GPIO connector have been reserved for use with a display HAT. This is due to the fact that most GPIO pins are expected to be exclusively used by a certain hardware. It this needs to make sure that the pins used by a display are not used by another addon and vice versa.

The different displays available for the Raspberry Pi differ significantly in the way they make use of the Raspberry Pi's GPIO pins. The popular Waveshare 3.5" touchscreen displays with 320x480 pixels resolution have been choosen as a reference. Several 3.4 inch third party displays are also compatible with this setup. Displays known to be compatible include:

Waveshare 3.5inch RPi LCD (A) Various compatible third party displays are available.

Waveshare 3.5inch RPi LCD (B) This display differs slightly in the display technology used and it requires a different driver but is otherwise compatible with the (A) version and with the default pin usage of the TX Pi.

Waveshare 4.0inch RPi LCD (A) This display is slghtly bigger than the 3.5 inch version but is otherwise compatible and features the same resolution and the same pin usage.

Waveshare 3.5inch RPi LCD (C) This is a high speed version of the (A) model. The **MHS-3.5inch RPi Display** is known to be compatible with this.

In general all displays able to work with the default drivers for the Waveshare displays mentioned above are compatible with the pin usage of the TX Pi.

1.2.1 Display case

The TX Pi project also provides case designs for the displays. These can be found at https://github.com/ftCommunity/tx-pi/tree/master/cases and on Thingverse[3]

Selecting the right bottom and top halves depends on the Raspberry Pi being used as the dispay case requires different extra cutouts for the different USB and Ethernet connectors of the Raspberry Pi4 or previous versions. Also if a camera cable is to be routed through the display case then a custom version may be needed.

[3] ft TX-Pi compatible 4in. display case, https://www.thingiverse.com/thing:3605290

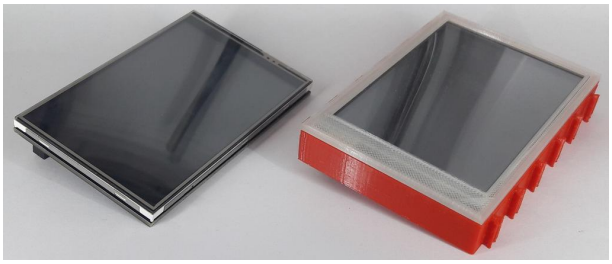


Figure 1.6: A case for the Waveshare 3.5inch (A) display

Chapter 2

The TX Pi HAT

The TX Pi HAT is a so called "HAT" for the Raspberry Pi. HAT stands for "Hardware attached on top" and describes a piece of hardware which can be attached to the top of the Raspberry Pi using its 40 pin GPIO header.

Figure 2.1: Version 1.0 of the TX Pi HAT mounted on a Raspberry Pi

The TX Pi HAT attaches to the top of the GPIO header and extends the pins on its own top to allow for further HATs to be stacked on top of the TX Pi HAT. The TX Pi HAT makes intensive use of the signals exposed by the Raspberry Pi on its 40 pin expansion connector. Care has thus to be taken if multiple HATs are being used at once.

The Raspberry Pi or TX Pi by itself does not require the TX Pi HAT to be present. The TX Pi HAT is optional but it makes the integration of the Raspberry Pi into the fischertechnik system a significantly more straight forward. The TX Pi HAT was designed to address the following topics:

- The Raspberry Pi is typically powered via a USB connector from a 5V power source like a phone charger. The TX Pi HAT allows the Pi to be powered from a 9V power source instead.
- The Raspberry Pi can not control it's own power source. This means that a raspberry pi cannot completely switch off by itself. The TX Pi HAT contains a power supply controllable by the Raspberry Pi allowing to save power e.g. in battery driven setups.
- The GPIO pins of the Raspberry Pi are not electrically compatible with fischertechnik sensors and actors. The TX Pi HAT gives the Raspberry Pi four fischertechnik compatible digital inputs and two fischertechnik compatible analog motor outputs.

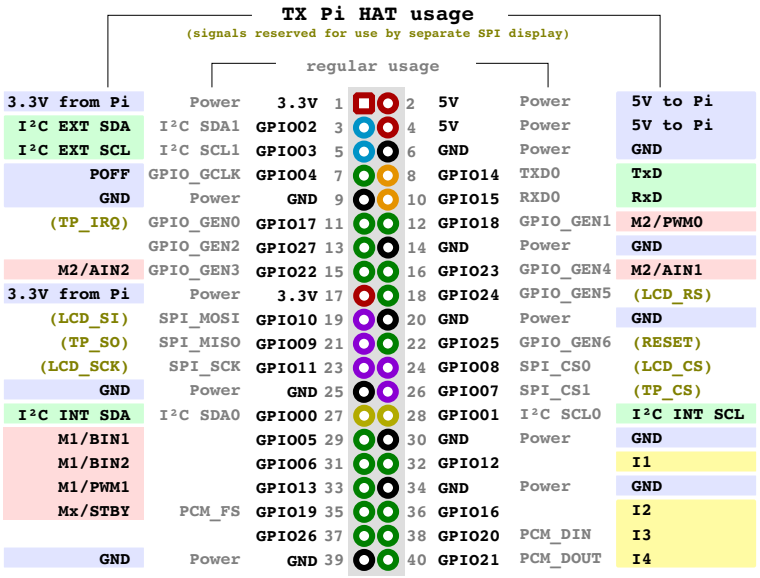


Figure 2.2: TX Pi HAT usage of extension pins on the Raspberry Pi

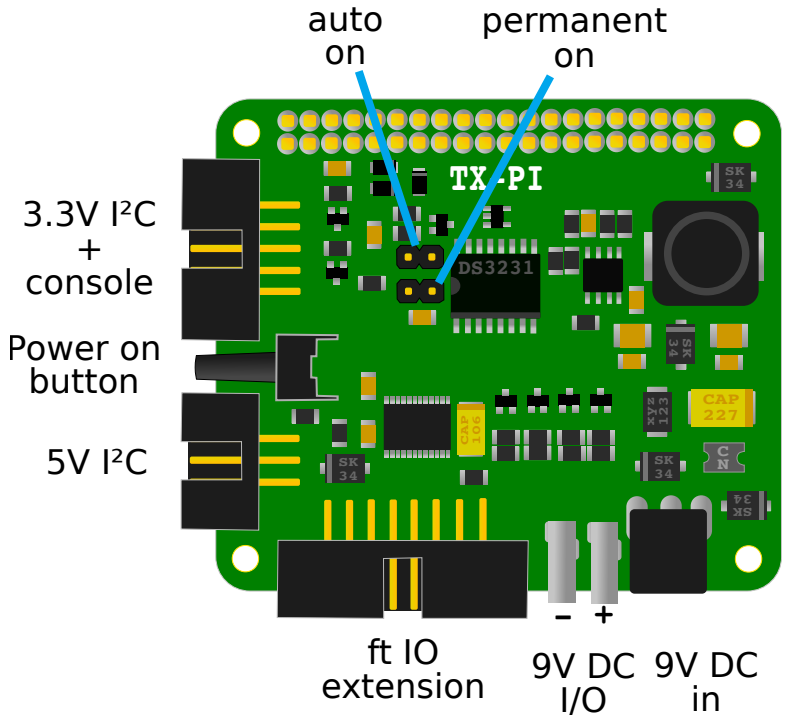
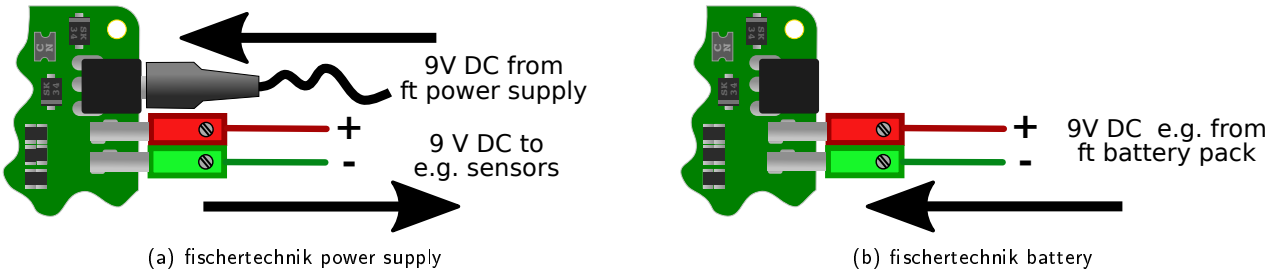


Figure 2.3: The IO ports of the TX Pi HAT

- I²C sensors are popular in the fischertechnik world as well as in the Raspberry Pi community. The TX Pi HAT provides the Raspberry Pi with two fischertechnik compatible I²C ports. The 10 pin 3.3V port is compatible with the EXT port of the fischertechnik TXT and like this carries the console UART signals besides the I²C signals. The additional 6 pin port implements a 5V I²C bus and is compatible with the fischertechnik TX controller as well as the ftDuino and various other third party add on's. Both, the 5V and the 3.3V port can be used at the same time. But since they share the same logical I²C bus they also share the same I²C address space.

2.1 Power supply

One of the main features of the TX Pi HAT is it's ability to power the Raspberry Pi from most regular fischertechnik power sources. Power sources can either be connected via the 4mm DC jack or via the pair of fischertechnik 2.5mm jacks left of the DC jack as depicted in figure 2.4.



(a) fischertechnik power supply (b) fischertechnik battery

Figure 2.4: Power supply options of the TX Pi HAT

Supported and tested are:

- fischertechnik power supply 505287 as delivered by fischertechnik for the TXT controller.
- Most stabilized 9V DC power supplies with matching adapters. The power supply should be able to deliver at least 2.5A.
- The fischertechnik 9V battery case using a regular disposable 9V battery.
- The fischertechnik rechargeable battery pack.

Both power supply options are protected against reversed polarity. The DC jack is additionally protected against reverse current flow from the TX Pi HAT into the DC jack.

The 2.5mm fischertechnik jacks are not protected against reversed current flow as they are supposed to be used in both directions. They can be used to either power the TX Pi HAT from an external power source as depicted in figure 2.4(b). Or when powered from the DC jack they can be used to power external devices like sensors from the power supply connected to the DC jack as depicted in figure 2.4(a).

Care has thus to be take to not connect a battery and the DC power source at the same time. In this case the battery would be sourced from the power supply and the battery will likely be damaged.

2.1.1 Power indicator LED

The TX Pi HAT includes a power LED below the power button. This LED will light up whenever the TX Pi HAT's power supply is enabled.

2.1.2 Raspberry Pi power consumption

The latest Raspberry Pi with a few peripherals connected draws at most 3A at 5V under full load. The power supply on the TX Pi HAT was designed to deliver this amount of power. Therefore more than 1.6A on 9V side may be required under full load. In most cases and under regular load the Raspberry Pi typically draws at most around 500mA on 9V via the TX Pi HAT.

E.g. the fischertechnik rechargeable battery pack is rated at 1500mAh. It will thus allow to power the Raspberry Pi for about an hour under full load or nearly three hours under normal load. The fischertechnik power supply 505287 delivers up to 2.5A and easily power the Raspberry Pi even under full load. Some care still has to be taken as additional motors and lamps on the fischertechnik model may also consume significant power.

2.1.3 Power jumper settings

The power supply feature of the TX Pi HAT can be configured using two jumpers on the board. These jumpers control the power-on behavior of the setup:

auto on If this jumper is installed then the Raspberry Pi will automatically be power on whenever power is applied to the TX Pi HAT. If this jumper is removed then the TX Pi HAT will stay off when power is applied and the power button needs to be used to power on the Raspberry Pi.

permanent on Installing this jumper overdrives any power control and if it's set the power supply will be permanently enabled. The Raspberry Pi is then not able to power itself off. This jumper is also needed if the TX Pi HAT is being used stand alone without Raspberry Pi.

Usually the `auto on` jumper is being installed and and the `permanent on` jumper is not installed.

2.1.4 Power on

There are several ways to control the power supply of the TX Pi HAT. All of these can only be used to enable the power supply. If any one of these is enabled then the power supply will be on.

permanent on jumper If this jumper is installed the power supply will be on regardless of any other signal.

Raspberry Pi `GPIO04` This GPIO pin is controlled by the Raspberry Pi and is used to keep power on during regular operation. The Raspberry Pi can be configured to use this pin to release the power supply once the PI is shut down.

Power button The power button enables the power supply as long as the power button is being pressed. In a typical setup the Raspberry Pi will take over power control via `GPIO04` once the power button is released.

Applying power If the `auto on` jumper is installed applying power will enabled the power supply for a short period of time. In a typical setup the Raspberry Pi will take over power control via `GPIO04` once power is stable for some time.

RTC The on board real time clock (RTC) can activate the power supply using its interrupt output. This alarm usually needs to be acknowledged by the Raspberry Pi via I^2C. Otherwise this signal stays active and will keep the Raspberry Pi powered even if it's shut down.

If the TX Pi HAT is powered off it goes into a low power state and the power consumption is less than 30 µA. The RTC is still enabled during power down in order to keep the time and to be able to power the system on via its alarm triggered interrupt output.

2.1.5 Raspberry Pi auto power off configuration

The TX Pi HAT gives the Raspberry Pi control over it's own power supply. During regular operation the TX Pi HAT power supply will be powered on for a short period of time by e.g. the power button. This will immediately wake up the Raspberry Pi itself which takes over and keeps the TX Pi HAT power supply on via its `GPIO04` pin. The Raspberry Pi will do this by default and without further configuration due to a Raspberry Pi internal pull-up resistor on this GPIO pin.

The Raspberry Pi can be configured to release this pin on shut down allowing it to disable its own power supply once it has successfully finished its shut down sequence. This feature is not enabled by default. In order to allow the Raspberry Pi to power off on shut down the following line needs to be added to the `/boot/config.txt` file on the Raspberry Pi.

```
dtoverlay=gpio-poweroff,gpiopin=4,active_low=1
```

2.2 Internal I^2C port for RTC power control

The on board RTC of the TX Pi HAT can control the power supply. By default the RTC is not configured and will not influence power control. The RTC is controlled via the I^2C bus and e.g. shows up under address 0x68 on I^2C bus `i2c-0`[1].

The following needs to be added to the `/boot/config.txt` file on the Raspberry Pi in order to enable `i2c-0`:

[1] On early prototypes the RTC shows up on I^2C bus `i2c-1`

```
dtparam=i2c_vc=on
```

Afterwards the I²C bus should be accessible:

```
$ i2cdetect -l
i2c-0   i2c             bcm2835 I2C adapter                     I2C adapter
```

And the RTC chip should show up at address 0x68 while the optional EEPROM chip may also show up at address 0x50:

```
$ sudo i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The RTC can be configured via I²C to enable the TX Pi HAT's power supply. Example code for this is e.g. provided by the `tx-pi-hat-demo` app.

Once the RTC's alarm is active it will permanently drive the power supply. Any pending alarm needs to be acknowledged in order to give the Raspberry Pi back control over the power supply. The `tx-pi-hat-demo` also does that when being launched. If the Raspberry Pi has been started by the RTC the `tx-pi-hat-demo` needs to be launched to acknowledge the RTC's alarm. Afterwards the Raspberry Pi can be powered off as usual.

2.2.1 RTC alarm auto acknowledge

The RTC alarm acknowledge can be automated by adding the command `i2cset -y 0 0x68 0x0f 0x00` to the end of the `/etc/rc.local` file e.g. as follows:

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true

if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

# ack pending RTC wakeup
/usr/sbin/i2cset -y 0 0x68 0x0f 0x00

exit 0
```

This will clear the alarm bit on every boot and will this allow to totally power down the Raspberry Pi again.

2.3 On-board EEPROM

The usage of the EEPROM is optional. It is not required for proper operation of the TX Pi HAT. However, it can be used to store helpful system information that gives the Raspberry Pi some additional information about the HAT already at a very early boot stage.

In order to be able to access the EEPORM its I^2C bus may need to be enabled as explained in section 2.4.3.

The optional on board EEPROM can be accessed by the Raspberry Pi via the same I^2C port as the RTC. The EEPROM can be used to store additional configuration information to be used by the Raspberry Pi at boot time to setup the system. The EEPROM is optional and is usually not required[2].

Some TX Pi HAT specific content of the EEPROM could be stored in file `eeprom_settings.txt` like this:

```
##############################################################################
# Vendor info

# 128 bit UUID.
product_uuid 392b690a-c74d-4049-8a37-2f2789a7946e

# 16 bit product id
product_id 0x4711

# 16 bit product version
product_ver 0x0000

# ASCII vendor string  (max 255 characters)
vendor "Till Harbaum"

# ASCII product string (max 255 characters)
product "fischertechnik HAT"

# If board back-powers Pi via 5V GPIO header pins:
# 2 = board back-powers and can supply the Pi with a minimum of 2A
# If back_power=2 then USB high current mode will be automatically
# enabled on the Pi
back_power 2
```

The EEPROM can be read by e.g. the following command:

```
$ sudo eepflash.sh -r -f=dump.eep -t=24c256 -d=0 -a=50
This will attempt to talk to an eeprom at i2c address 0x50 on bus 0. Make sure there is an
eeprom at this address.
This script comes with ABSOLUTELY no warranty. Continue only if you know what you are doing.
Do you wish to continue? (yes/no): yes
Reading...
32256 Bytes (32 kB, 32 KiB) kopiert, 3,01154 s, 10,7 kB/s
64+0 Datensätze ein
64+0 Datensätze aus
32768 Bytes (33 kB, 32 KiB) kopiert, 3,06013 s, 10,7 kB/s
Closing EEPROM Device.
Done.
```

Afterwards the 32768 bytes contents of the EEPROM are stored in the file dump.eep.

To write the aforementioned data from `eeprom_settings.txt` to the EEPROM the textual representation first has to be converted to bonary:

```
$ eepmake eeprom_settings.txt eeprom_settings.eep
Opening file eeprom_settings.txt for read
```

[2] Raspberry Pi HAT EEPROM data: https://github.com/raspberrypi/hats/tree/master/eepromutils

```
Done reading
Writing out...
Done.
```

Afterwards the data can be written to the EEPROM:

```
$ sudo eepflash.sh -w -f=eeprom_settings.eep -t=24c256 -d=0 -a=50
This will attempt to talk to an eeprom at i2c address 0x50 on bus 0. Make sure there is an
eeprom at this address.
This script comes with ABSOLUTELY no warranty. Continue only if you know what you are doing.
Do you wish to continue? (yes/no): yes
Writing...
0+1 Datensätze ein
0+1 Datensätze aus
114 Bytes kopiert, 0,600914 s, 0,2 kB/s
Closing EEPROM Device.
Done.
```

If everything worked correctly the HATs EEPROM config will be detected after next boot and shows up under /proc:

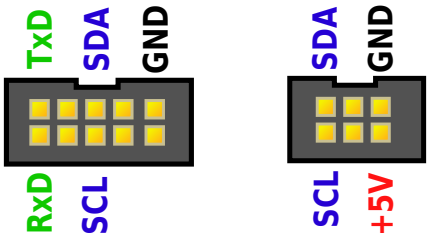
```
$ cat /proc/device-tree/hat/product
fischertechnik HAT
```

2.4 External I²C ports

The Raspberry Pi exposes two I²C busses on its expansion connector. The I²C bus i2c-0 is present on pins 27 and 28 (GPIO00 and GPIO01) while a second I²C bus i2c-1 is provided on pins 3 and 5 (GPIO02 and GPIO03).

Bus i2c-0 is primarily meant to be used for a so called HAT ID EEPROM[3] while bus i2c-1 is free for any other usage. The ftDuino makes use of both busses and uses i2c-0 for its own I²C components and uses i2c-1 for externally connected devices. Using seperate busses for both purposes avoids address collisions. E.g. the ftDuino's RTC uses the same address 0x68 as the gyro part of the fischertechnik combi sensor 158402.

The TX Pi HAT exposes i2c-1 port via two connectors as depicted in figure 2.5. The two connectors are compatible to the ports provided by the original fischertechnik TXT and TX controllers as well as the I²C port of the ftDuino.



(a) 3.3V (b) 5V

Figure 2.5: The I²C ports of the TX Pi HAT

2.4.1 3.3V I²C port

The ten pin I²C port of the TX Pi HAT depicted in figure 2.5(a) is compatible to the EXT port of the fischertechnik TXT controller. All signals on this port use a 3.3V signal level and should be connected to 3.3V signals only. Like the EXT connector of the TXT this port carries a I²C bus as well as the console TxD (transmit data) and RxD (receive data) of a serial console. Both the I²C signals as well as the console signals are connected directly to the according pins of the Raspberry Pi's expansion header. Over voltages and short circuits on these signals may therefore harm the Raspberry Pi.

[3] HAT ID EEPROM: https://github.com/raspberrypi/hats/tree/master/eepromutils

The serial console

The TX Pi HAT just like the fischertechnik TXT controller provides access to the internal serial console of the Linux system through pins 9 and 10 of the ten pin I²C port. On both systems a adapter using e.g. a USB to 3.3v UART converter may be used to get access to the serial console of the TXT or the Raspberry Pi.

This console usually runs at 115200 bit/s and can be accessed from the PC side using regular terminal emulator software like Teraterm (Windows) or Minicom (Linux).

2.4.2 5V I²C port

The six pin I²C port of the TX Pi HAT depicted in figure 2.5(b) is compatible with the EXT port of the fischertechnik TX controller and the I²C port of the ftDuino. It is connected via a level shifter to the same I²C port as the ten pin TXT compatible port. The level shifter provides 5 volt signal levels on the I²C signals. This port should therefore be used whenever a 5V I²C device is to be connected to the TX Pi HAT.

The same logical I²C bus is present on both connectors. From the Raspberry Pi's point of view these port are indistinguishable and they e.g. share the same address space.

The six pin port also provides a 5V power source. This port can source up to 250mA for external devices like sensors or e.g. a ftDuino connected via I²C. The 5V connection on the six pin port does now allow to power the TX Pi HAT or the attached Raspberry Pi. A protection diode makes sure that any external power source connected to this port is not being used. This protects any potential external power source like the ftDuino from overload by the up to 3A current the Raspberry Pi draws under normal operating conditions.

2.4.3 Raspberry Pi integration of I²C

Both external I²C ports of the TX Pi HAT carry the same `i2c-1` bus of the Raspberry Pi. In regular Raspberry Pi setups this port is not enabled by default and needs to be enabled using e.g. the standard `raspi-config` tool of the Raspberry Pi.

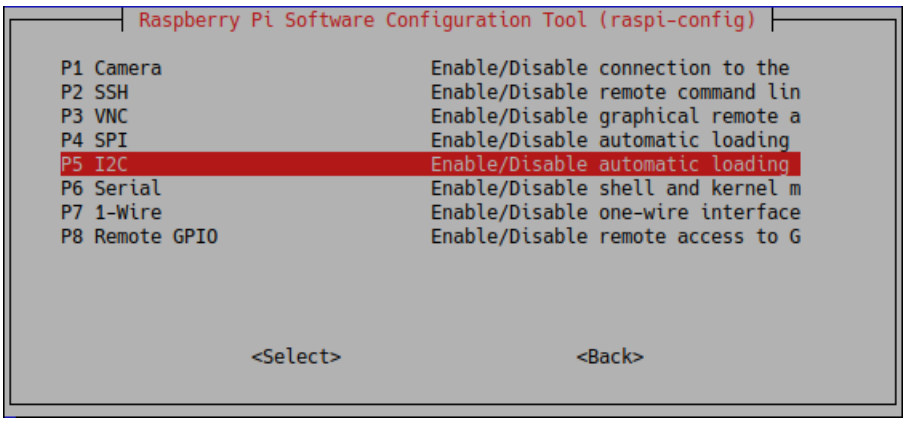


Figure 2.6: Enabling `i2c-1` in the `raspi-config` tool

The second (internal) I²C bus should have also been enabled as described in section 2.2. Once enabled both I²C busses are available to the I²C related command line tools like `i2cdetect` as `i2c-0` and `i2c-1`.

The bus itself can be detected using the `i2cdetect -l` command:

```
$ i2cdetect -l
i2c-0   i2c             bcm2835 I2C adapter                     I2C adapter
i2c-1   i2c             bcm2835 I2C adapter                     I2C adapter
```

If the I²C has been setup correctly this will make the busses show up as `i2c-0` and `i2c-1`. The commands `i2cdetect 0` and `i2cdetect 1` can then be used to scan the bus for I²C devices.

Bus `i2c-0` will expose the on-board RTC (real time clock) of the TX Pi HATat address `0x68` and if installed as well the ob-board ID EEPROM at address `0x50`:

```
$ i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

A similar scan on bus i2c-1 will only reveal devices if there's something connected to the external I^{2}C ports of the TX Pi HAT. In the following case a mini servo adapter[4] is attached to the 5V I^{2}C port.

```
$ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- 11 -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The I^{2}C also needs to be enabled to gain access to the RTC of the TX Pi HAT in order to be able to use the RTC alarm controlled power on.

In the image above two devices show up. The internal RTC of the TX Pi HAT under address 0x68 and an externally connected mini servo adapter under address 0x11.

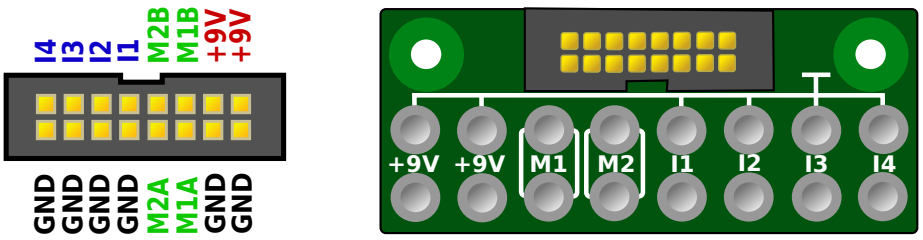
Similar information can be obtained from the touchscreen GUI of the TX Pi using the I^{2}C scanner app as depicted in figure 2.9(a).

2.5 fischertechnik inputs and outputs

The TX Pi HAT was designed to integrate the Raspberry Pi into the fischertechnik system. Besides a fischertechnik compatible power supply it also provides a few fischertechnik compatible inputs and output which can be used to connect switches, lamps, motors and similar devices from the fischertechnik system with the TX Pi HAT.

The TX Pi HAT was designed to be stacked on top of the Raspberry Pi and below a optional display add-on. Thus connectors can only be placed on the sides of the TX Pi HAT which significantly limits the available space.

Thus all fischertechnik compatible connections have been moved to a single 16 pin IO connector as depicted in figure 2.7.



(a) TX Pi HAT IO connector

(b) optional breakout board

Figure 2.7: fischertechnik IO connections on the TX Pi HAT

[4] Mini servo adapter: https://github.com/harbaum/ftduino/tree/master/addons/i2c-servo

A cable carrying a matching connector on one end and regular 2.5mm fischertechnik plugs on the other can directly be connected to this port.

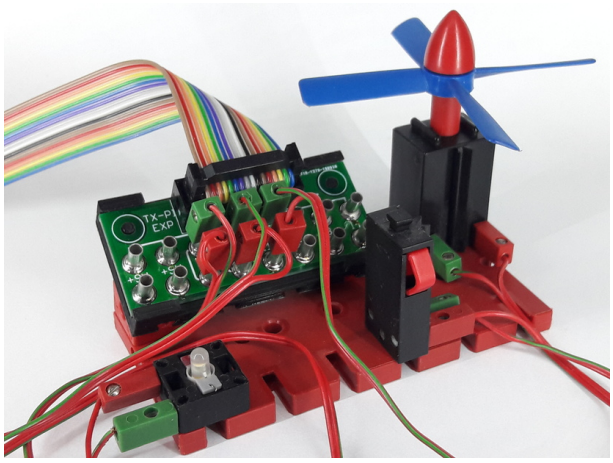


Figure 2.8: Connecting fischertechnik to the breakout board

To make usage easier the TX Pi HAT comes with a breakout board carrying the regular fischertechnik sleeves which can be used with the regular 2.5mm fischertechnik plugs.

2.5.1 Inputs

The TX Pi HAT features four fischertechnik compatible inputs `I1`, `I2`, `I3` and `I4`. These are digital inputs and can be used to connect buttons, switches and the fischertechnik photo transistor.

The inputs are protected against over voltage and can directly be connected to any fischertechnik power source. The regular use case is to connect the input against ground.

The `TX Pi HAT demo` app available from the cfw-apps repository [5] displays the state of the inputs as depicted in figure 2.9(b).

The four inputs `I1` to `I4` are mapped to the Raspberry Pi's pins `GPIO12`, `GPIO16`, `GPIO20` and `GPIO21` as depicted in figure 2.2.

2.5.2 Outputs

The TX Pi HAT features two fischertechnik compatible motor outputs `M1` and `M2`. Each motor output consists of two individual outputs. The outputs can both be adjusted in polarity allowing to e.g. control the motors direction and the speed can be adjusted using a separate PWM signal.

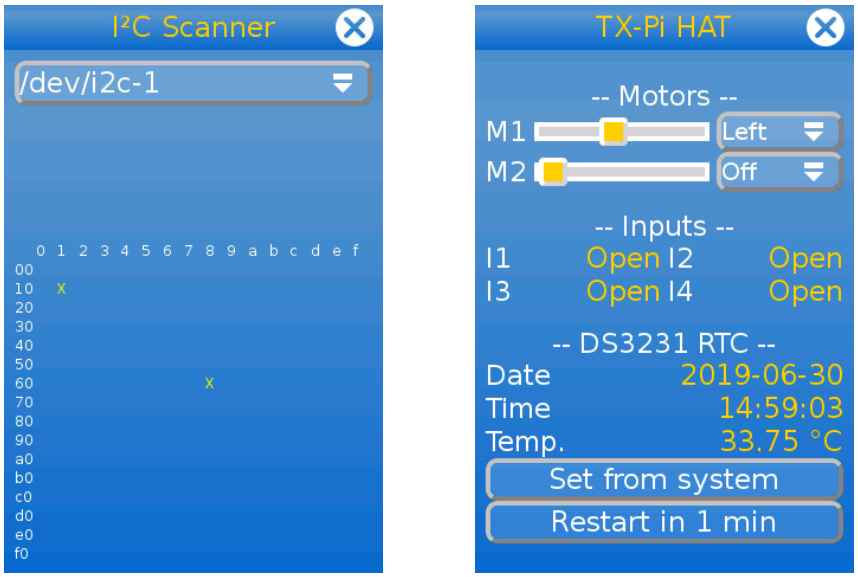
The motor outputs are driven by a `TB6612` dual motor bridge. The input pins of the `TB6612` are mapped to the Raspberry Pi's GPIO pins as follows:

	M1	M2
IN1	BIN1/GPIO05	AIN1/GPIO23
IN2	BIN2/GPIO06	AIN2/GPIO22
PWM	PWM1/GPIO13	PWM0/GPIO18
STBY	GPIO19	

The state of the two motor outputs can be controlled from the Raspberry Pi by driving `GPIO05`, `GPIO06` and `GPIO13` for `M1` and `GPIO23`, `GPIO22` and `GPIO18` for `M2`. The `STBY` signal is shared by both outputs and is controlled from the Raspberry Pi's `GPIO19`.

The four pins on one channel control the state of one motor output as follows:

[5] cfw-apps repository, https://github.com/harbaum/cfw-apps



(a) I²C scanner app (b) TX Pi HAT demo app

Figure 2.9: Apps making use of the TX Pi HAT

STBY	IN1	IN2	PWM	state
0	x	x	x	standby (high impedance)
1	0	0	1	stop (high impedance)
1	1	1	x	short brake (both outputs low)
1	0	1	1	turn CCW (one output low, one high)
1	0	1	0	short brake (both outputs low)
1	1	0	1	turn CW (one output high, one low)
1	1	0	0	short brake (both outputs low)

The `TX Pi HAT demo` app available from the cfw-apps repository [6] allows to control the state of the outputs as depicted in figure 2.9(b).

More information about programming the inputs and outputs is being given in section 3.

[6] cfw-apps repository, https://github.com/harbaum/cfw-apps

Chapter 3

Programming the TX Pi HAT

The TX Pi HAT extends existing interfaces of Raspberry Pi to make them accessible for fischertechnik usage.

Part of these extensions are accessed via I^{2}C and part of them is accessed using the Raspberry Pi's GPIO pins.

3.1 I^{2}C

The I^{2}C bus used for the Raspberry Pi is the regular `i2c-1` bus available on the 40 pin GPIO port of as depicted in figure 2.2. Before this port can be accessed by software it needs to be enabled as explained in section 2.4.3.

The TX Pi HAT itself already contains a I^{2}C client chip. Further I^{2}C peripherals can be connected externally either via the 10 pin 3.3V port or via the 6 pin 5V port.

3.1.1 Built-in DS3231 RTC

The built-in RTC shows up under address 0x68. Since the DS3231 is supported by a standard I^{2}C kernel driver it can be made available to the kernel using the Linux on-board tools and drivers.

Command line

Since there is no reliable way of auto detecting hardware on the I^{2}C bus the Linux kernel needs to be told that there is a ds3231 device at address 0x68 of bus `i2c-1`:

```
$ sudo modprobe rtc_ds1307
$ echo ds3231 0x68 | sudo tee -a /sys/class/i2c-adapter/i2c-1/new_device
ds3231 0x68
```

If this is wanted permanently and at boot time, then the following line added to `/boot/config.txt` will achieve the same result:

```
dtoverlay=i2c-rtc,ds3231,addr=0x68
```

Afterwards the RTC should show up in the sys file system and the date and time can be read:

```
$ cat /sys/class/rtc/rtc0/date
2000-01-01
$ cat /sys/class/rtc/rtc0/time
00:10:15
```

The command `hwclock` can then be used to read the time from the RTC:

```
$ hwclock -r
2000-01-01 01:11:08.923619+0100
```

It's obvious that the RTC does not know the correct time. In most cases the Raspberry Pi will know the correct time already from a network service. This time can be written into the RTC:

```
$ hwclock -w
$ hwclock -r
2019-07-01 09:22:48.677249+0200
```

The time can be moved from the RTC back to the system time using the `hwclock -s` command.

Please be aware that the RTC does not come with a seperate power supply. If the TX Pi HAT is disconnected from a power supply the time and date stored on the RTC will be reset to 2000-01-01 00:00:00 UTC. But the date and time is kept if while the TX Pi HAT is being connected to a power source even if the main power supply is switched off and if the Raspberry Pi is switched off. This allows to use the RTC to power up the Raspberry Pi at a given time.

Unfortunately the Linux kernel driver used this way does not provide access to the alarm functions of the RTC which are required to wake up the Raspberry Pi timed by the RTC alarm. Thus using the linux kernel driver for the RTC is not recommended. Instead the RTC should be accessed directly bypassing any RTC driver as outlined in the following section.

Python

In Python like most other programming languages it's possible to directly access the I²C bus and thus send arbitrary commands to the any I²C device like the RTC. A matching Python library for the DS3231 RTC can e.g. be found at github[1].

This library can be used from within any Python program:

```
ds3231 = DS3231.DS3231(1, 0x68)
# set RTC time from system time
ds3231.write_now()
# set alarm one minute in the future
ds3231.setAlarm(datetime.datetime.now() + datetime.timedelta(minutes=1))
# shut down pi to let it wake up again
call(["sudo", "poweroff"])
```

A second snippet of code is needed once the RTC has woken up the Raspberry Pi since the RTC will keep the alarm signal engaged which in turn would prevent future attemps to power the Raspberry Pi down again.

```
# after RTC triggered powerup the RTC needs to be acknowledged
# to release the power supply
ds3231.ackPendingAlarm()
```

The direct approach from Python will not work if the kernel driver is loaded as this gains exclusive access to the chip. The `i2cdetect -y 1` command will show 'UU' if the address is blocked by kernel usage.

```
$ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

In this case the driver can be unloaded:

```
$ echo 0x68 | sudo tee -a /sys/class/i2c-adapter/i2c-1/delete_device
```

[1] DS3231.py: https://github.com/harbaum/cfw-apps/tree/master/packages/tx-pi-hat-test

3.1.2 fischertechnik environmental sensor 167358

The environmental sensor from the fischertechnik home automation and IoT kits is based on the Bosch BME680 sensor.

Figure 3.1: The fischertechnik environmental sensor 167358

A matching Python library is available for Raspbian and can be installed:

```
sudo apt-get install python3-bme680
```

This library can be used in a Python program as follows[2]:

```
import bme680
import time

sensor = bme680.BME680()

sensor.set_humidity_oversample(bme680.OS_2X)
sensor.set_pressure_oversample(bme680.OS_4X)
sensor.set_temperature_oversample(bme680.OS_8X)
sensor.set_filter(bme680.FILTER_SIZE_3)

sensor.set_gas_status(bme680.ENABLE_GAS_MEAS)
sensor.set_gas_heater_temperature(320)
sensor.set_gas_heater_duration(150)
sensor.select_gas_heater_profile(0)

while True:
    if sensor.get_sensor_data():
        output = "{0:.2f} C,{1:.2f} hPa,{2:.2f} %RH".format(↩
           sensor.data.temperature, sensor.data.pressure, sensor.data.humidity)

        if sensor.data.heat_stable:
            print("{0},{1} ohms".format(output, sensor.data.gas_resistance))
        else:
            print(output)

    time.sleep(1)
```

The output will include temperatures, pressure and humidity:

```
$ python3 ./bme680.py
30.60 C,1005.56 hPa,27.84 %RH
30.63 C,1005.56 hPa,27.85 %RH,35178.86404694175 ohms
30.69 C,1005.59 hPa,27.82 %RH,49038.265142913355 ohms
30.76 C,1005.61 hPa,27.73 %RH,57726.21732917038 ohms
30.81 C,1005.60 hPa,27.65 %RH,63817.64696100763 ohms
30.86 C,1005.58 hPa,27.57 %RH,68219.75818501839 ohms
...
```

[2]The small arrow ↩ indicates that the line does not end there but continues with the contents of the next line

The ohms output is a value that can be used to determine air quality. Examples for this can be found on github[3].

3.1.3 fischertechnik combi sensor 158402

The combi sensor sold by fischertechnik under part number 158402 is based on the Bosch Sensortec BMX055 integrated circuit.

Figure 3.2: The fischertechnik combi sensor 158402

This integrated circuit bascially combines three seperate units into one case. These three units are a accelerometer, a gyroscope and a magnetometer (compass). All three are accessible under their own I²C addresses:

BMX055 component	I²C address
Accelerometer	0x18
Gyroscope	0x68
Magnetometer	0x10

Using the combi sensor thus shows up like this when being scanned using `i2cdectect`:

```
$ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: 10 -- -- -- -- -- -- -- 18 -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The fact that this gyroscope uses address 0x68 is the main reason why the TX Pi HAT's internal RTC is connected to the Raspberry Pi's other I²C bus `i2c-0` as it uses the same address 0x68 and having both devices on the same bus would let their addresses collide and making them inaccessible.

Python examples for this can be found at github[4]. However, to make this work with the fischertechnik combi sensor on a Raspberry Pi we had to comment line 78 like so:

```
# BMX055 Mag address, 0x10(16)
# Select Mag register, 0x4B(75)
#               0x83(121)       Soft reset
#bus.write_byte_data(0x10, 0x4B, 0x83)
```

After this line is commented this python program will read the sensor data:

```
$ python ./BMX055.py
Acceleration in X-Axis : 2
Acceleration in Y-Axis : -1
Acceleration in Z-Axis : 1018
X-Axis of Rotation : -14
Y-Axis of Rotation : -24
```

[3] bme680-python indoor-air-quality.py, https://github.com/pimoroni/bme680-python/blob/master/examples/indoor-air-quality.py
[4] BMX055 python demo: https://github.com/ControlEverythingCommunity/BMX055/blob/master/Python/BMX055.py

```
Z-Axis of Rotation : 44
Magnetic field in X-Axis : -34
Magnetic field in Y-Axis : -74
Magnetic field in Z-Axis : -133
```

Another python example making use of the BMX055 can be founf in the CFW apps repository[5]. It ws written for the fischertechnik combi sensor and runs on the fischertechnik TXT controller as well as the Raspberry Pi. Its output is depicted in figure 3.3.

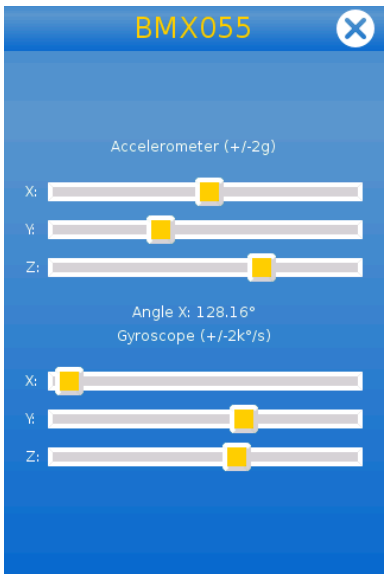


Figure 3.3: BMX055 CFW app

3.1.4 Third party sensors

Besides the sensors sold by fischertechnik the I²C has also become popular in the aftermarket. The TX Pi HAT allows to connect these directly if they follow the connection scheme used by the fischertechnik TXT or TX controllers.

In the following image are the two fischertechnik sensors connected to the 10 pin 3.3V port while a OLED display and a mini servo adapter are connected to the 6 pin 5V port. All devices show up on bus `i2c-1`:

```
$ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: 10 11 -- -- -- -- -- -- 18 -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- 76 --
```

The I²C bus is popular in the Raspberry Pi community and thus drivers and examples can easily be found in the internet for most common I²C peripherals.

Example code for the 128*64 OLED display showing up at address 0x3c can e.g. found in the Adafruit repository [6].

The mini servo adapter on the other hand is a community project aimed at the ftDuino[7] and thus does not come with a ready to use library for the Raspberry Pi. But the standard I²C command line tools can be used to control the mini servo adapter. E.g. the command

[5] BMX055 CFW app: https://github.com/harbaum/cfw-apps/blob/master/packages/bmx055/bmx055.py
[6] Adafruit OLED/SSD1306 python library: https://github.com/adafruit/Adafruit_Python_SSD1306
[7] ftDuino: http://ftduino.de

```
i2cset -y 1 0x11 0 100
```

will move the servo connected to servo output 0 of the mini servo adapter to position 100. The same can be achieved from within python:

```
$ python3
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import smbus
>>> bus = smbus.SMBus(1)
>>> bus.write_byte_data(0x11,0,100)
>>>
```

This approach works for many simple I^{2}C devices and allows to easily write a python software making use of nearly any I^{2}C chip.

3.2 Programming the fischertechnik inputs and outputs

The four fischertechnik inputs `I1` to `I4` and the two motor outputs `M1` to `M2` are connected directly to the GPIO pins of the Raspberry Pi. Programming them is thus done using the regular methods of dealing with the GPIOs of the Raspberry Pi.

The GPIOs of the Raspberry Pi can be controlled from most popular programming languages as well as from the command line.

3.2.1 Command line

Command line usage makes use of the `/sys` file system. E.g. the following command will activate `GPIO12` as input and then read the state of `GPIO12`. Since `GPIO12`is connected with the input `I1` this will read the current state of that input:

```
echo "12" > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio12/direction
cat /sys/class/gpio/gpio12/value
```

The other inputs `I2` to `I4` can be evaluated in the same way using 16, 20 and 21 as the GPIO values.

Outputs can be configured in a similar way. The following code will bring the motor outputs out of standby:

```
echo "19" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio19/direction
echo "1" > /sys/class/gpio/gpio19/value
```

Enabling M1 to turn full speed several GPIO pins need to be driven according to the tables in section 2.5.2:

```
echo "19" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio19/direction
echo "1" > /sys/class/gpio/gpio19/value

echo "5" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio5/direction
echo "1" > /sys/class/gpio/gpio5/value

echo "6" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio6/direction
echo "0" > /sys/class/gpio/gpio6/value

echo "13" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio13/direction
echo "1" > /sys/class/gpio/gpio13/value
```

3.2.2 Python

Similar can be done in most programming languages. In python reading `I1` via `GPIO12` looks like:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(12, GPIO.IN)
print("I1:", GPIO.input(12))
```

Again, the other inputs `I2` to `I4` can be evaluated in the same way using 16, 20 and 21 as the GPIO values.

Leaving the standby mode of the motor bridge can be done as follows:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(19, GPIO.OUT)
GPIO.output(19, GPIO.HIGH)
```

The four pins required to turn the motor `M1` can e.g. set in the following manner:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

pins = { 19:GPIO.HIGH, 5:GPIO.HIGH, 6:GPIO.HIGH, 13:GPIO.HIGH }
for p in list(pins.keys()):
    GPIO.setup(p, GPIO.OUT)
    GPIO.output(p, pins[p])
```