wordclock_for_rpi Documentation

Release 0.0.1

Bernd Krolla

Jun 10, 2023

Contents

CHAPTER 1

General considerations

1.1 If you want to build a wordclock

Note: This project is currently still in an experimental state:

- The documentation contains the main steps to build a wordclock, but might not guide you through all steps in greatest detail
- Depending on your language-dependent stencil-layout, you might need to adapt the software. Feel free to contribute here! Currently available:
  - Stancil layout and software:
    - german (including swabian and bavarian: Thanks to Timo and Euchkatzl)
    - english (Thanks to Alexandre)
    - dutch (Thanks to svenjacobi, resolution is 10x9. Therefore, not all plugins are supported!)
    - swiss
    - french
    - romanian (Thanks to Darius)
    - swedish
  - Stancil layout only (requiring some python-implementations: See wordclock_plugins/time_default):
    - italian
    - spanish
    - turkish
  - Further languages/stencil layouts can be created using the ‘’config-file” and the script ‘’create_layout.py”

- A final note: Throughout this project, you will assemble electronic components, which can possibly harm you or others (or destroy your hardware). It's therefore important, that you know, what you are doing: By assembling this clock, you act on your own risk!

- Hardware requirements:
  - A (wooden) sceleton to hold LEDs, stencil, RPi, etc…
  - A stencil providing the letters * Find an overview over the different layouts here: https://github.com/bk1285/rpi_wordclock/tree/master/wordclock_layouts * You can create them on your own: Special plotters can produce adhesive stencils, which you can glue onto a glas plane. * Consider, that you might need to invert the layout to have the adhesive surface on top to attach to the glas plate. * Possible options for ordering a stencil are:
    - https://www.ponoko.com/ (thanks to StefanCarton).
    - http://www.mikrocontroller.net/articles/Word_Clock (thanks to euchkatzl)
    - Further reading: * http://diskussion.christians-bastel-laden.de/viewforum.php?f=12&sid=b90281d4a392f47503e9b9fc15495b19
  - A frame to enframe the wordclock
    - Possible materials: Wood, alumnium, etc.
  - A LED-strip running at 5V (e.g. WS2812 B Stripe 5m 150 LED)
    - Assure, that the spacing of the LEDs on the strip is equal or greater than the spacing of the letters of your stencil. If the spacing is smaller, you will not be able to get your LEDs into the correct position.
  - A Raspberry Pi (e.g. Review B, including SD-card)
  - A wifi-dongle to connect your RPi wireless to your local network
  - A power supply (e.g. 5V 10A 50W LED Power Supply)
    - 5V are required. The current, which needs to be provided at max depends on the number and power consumption of you LEDs.
  - A user-interface to run the wordclock
    - e.g. 3 buttons (each requiring a 1k and a 10k resistor)
    - e.g. a capacitive touch sensor
    - …
  - Some plugs to connect cables to the Raspberry Pis GPIO-pins
  - A micro-usb cable to connect the Raspberry Pi to the power supply
  - Optional: Hardware for levelshifting as oulined in https://learn.adafruit.com/neopixels-on-raspberry-pi/wiring and http://youtu.be/V9TwvranJnY?t=23m08s
  - Optional: A temperature sensor like an AM2302. To connect the sensor, an additial 10k resistor is required.
- You need to setup the software on your own
  - Some first documentation available here
  - This might require some python programming (to adopt the software to your needs)
- You should be ready to…
  - Setup the hardware (glueing, soldering, etc.)

    * Consider double connectors, if you want to minimize soldering efforts: https://github.com/bk1285/rpi_wordclock/issues/118
  - Setup a Raspberry Pi (raspbian)
    * Connect to the RPi via ssh
    * Install external dependencies of the wordclock project
    * Do some python programming (to adopt the software to your needs)
  - Contribute to this project
    * by sharing your implementations/improvements/enhancements/... ;)

CHAPTER 2

Hardware setup

2.1 Skeleton setup

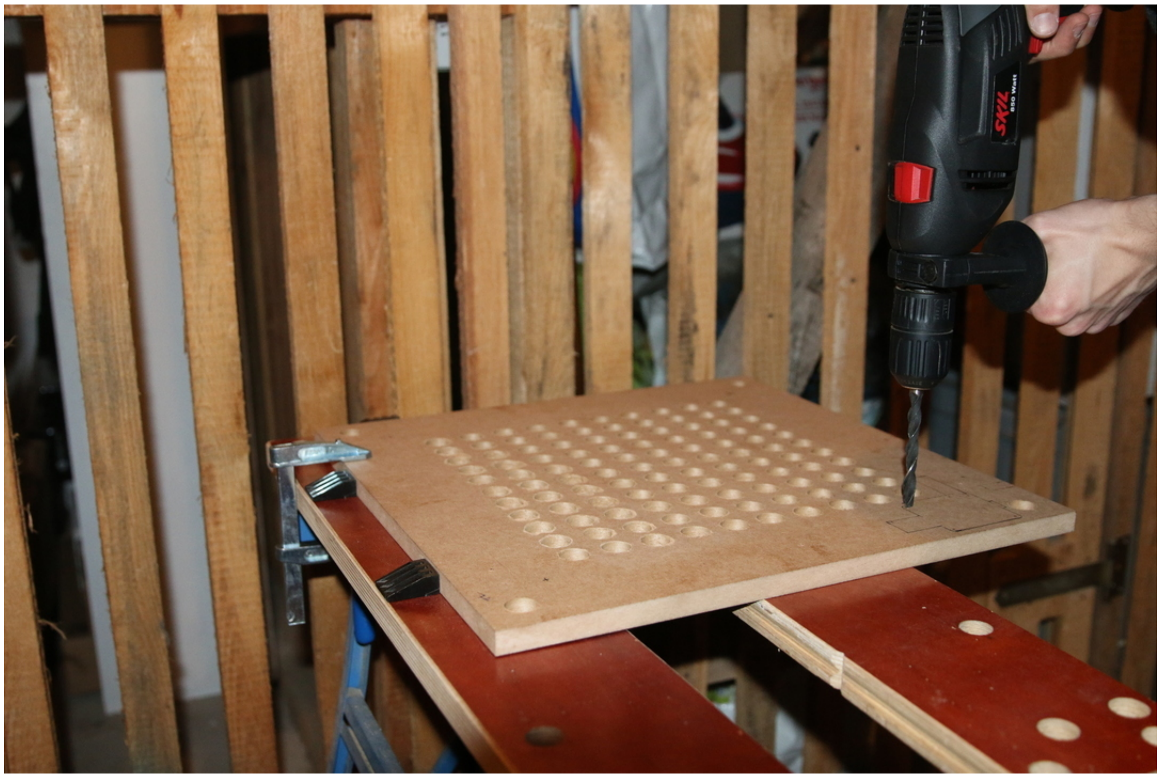


Fig. 1: According to the stencil layout, for each LED a hole needs to be prepared. The total number of 114 holes makes this pretty tedious.

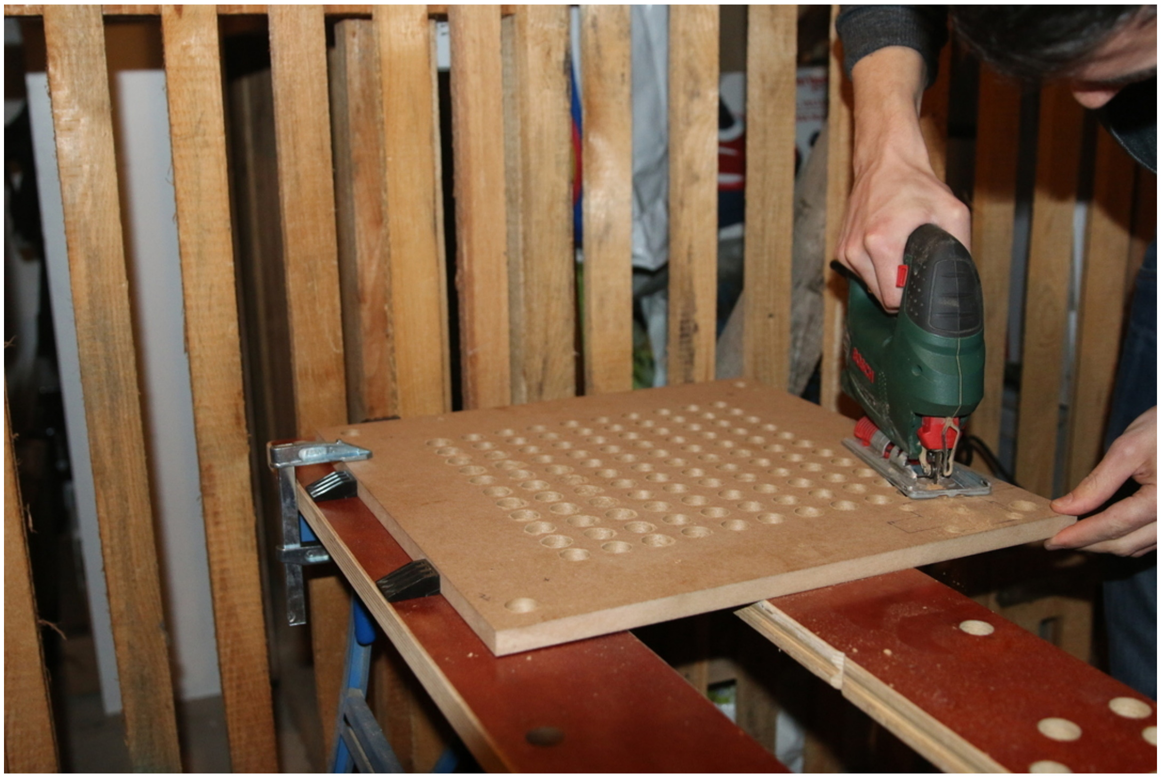


Fig. 2: To place the Raspberry Pi within the skeleton, some sawing is required. . .

2.2 LED setup

2.3 Raspberry Pi setup

2.4 Button setup

2.5 Stancil setup

2.6 Final clock

2.7 Video documentation on the wiring layout

Link to wiring documentation

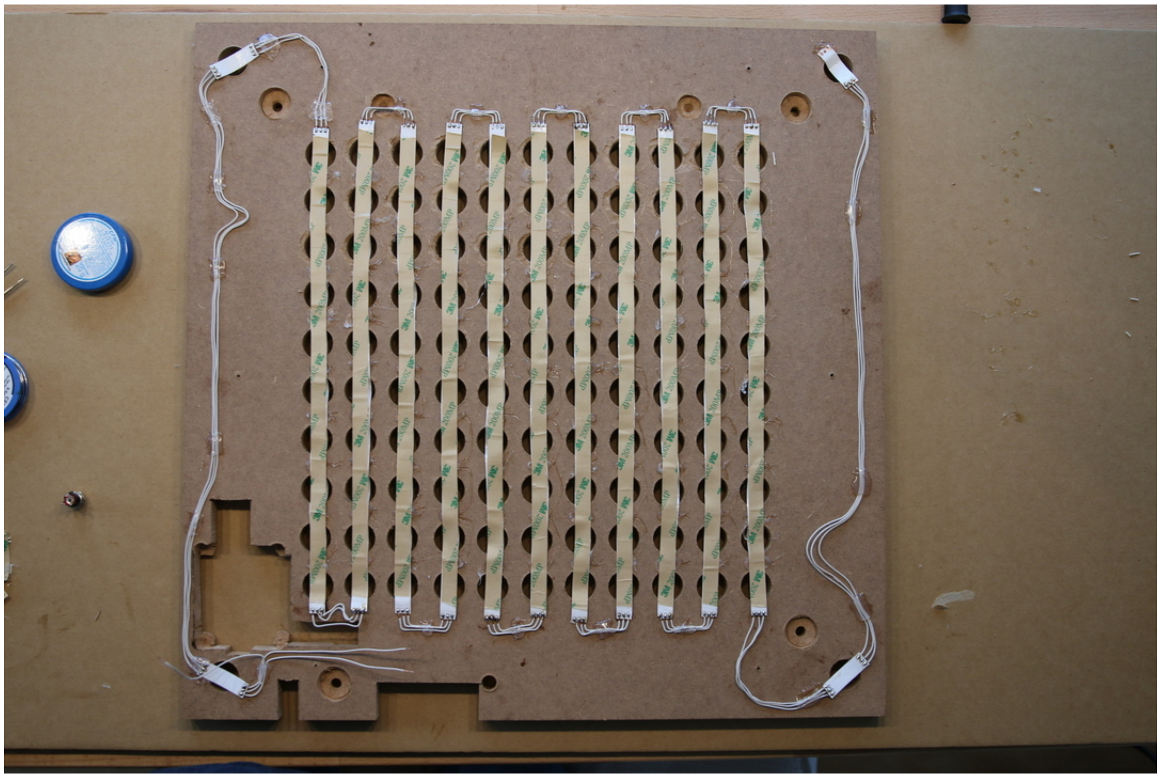


Fig. 3: After soldering the LED strip, the clock looks like this. The soldering needs to be done according to the wiring layout. E.g. based on 11*10 letters:

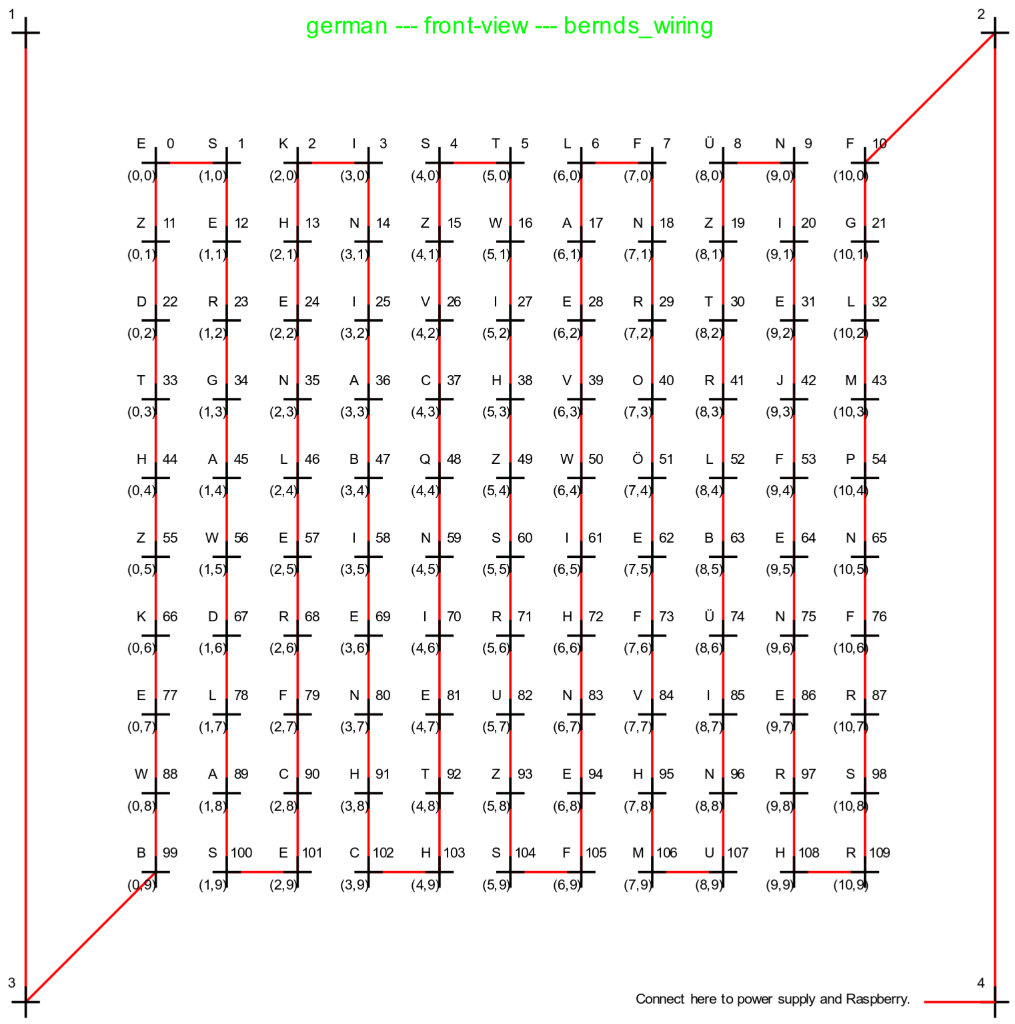


Fig. 4: Further wiring layouts are available. Assure to connect the LED strip in the right direction. Little arrows indicate that along the strip.

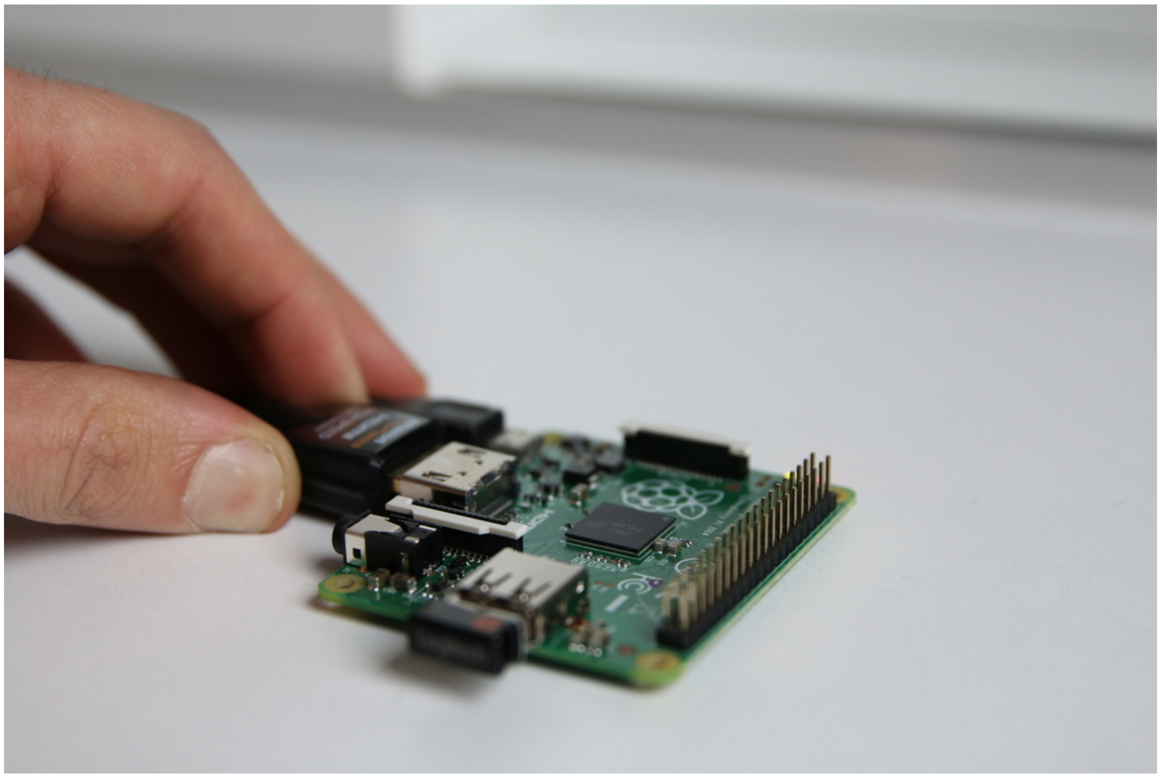


Fig. 5: Before you mount your raspberry inside the clock, install the latest Raspbian, connect it to your local wifi and ensure that you can ssh to it.

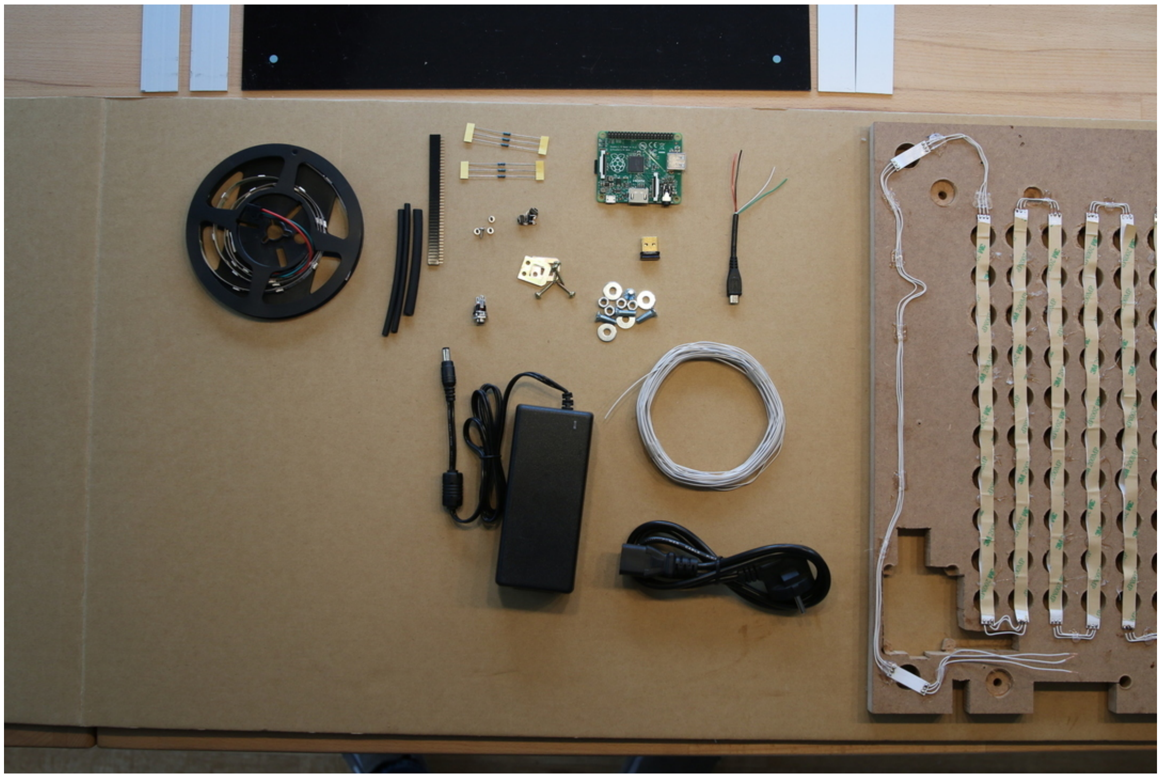


Fig. 6: At this stage, the displayed components are required for the further setup.

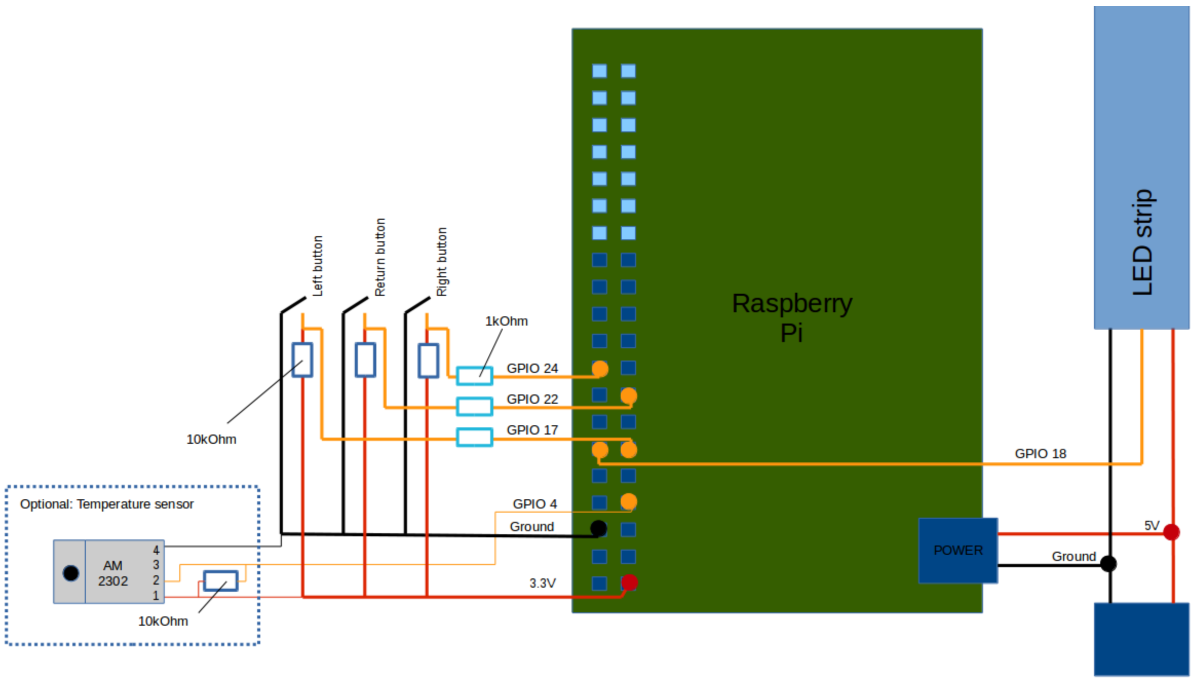


Fig. 7: Conceptual wiring layout to connect RPi, buttons, etc. See also Power connectors, USB-Pinouts

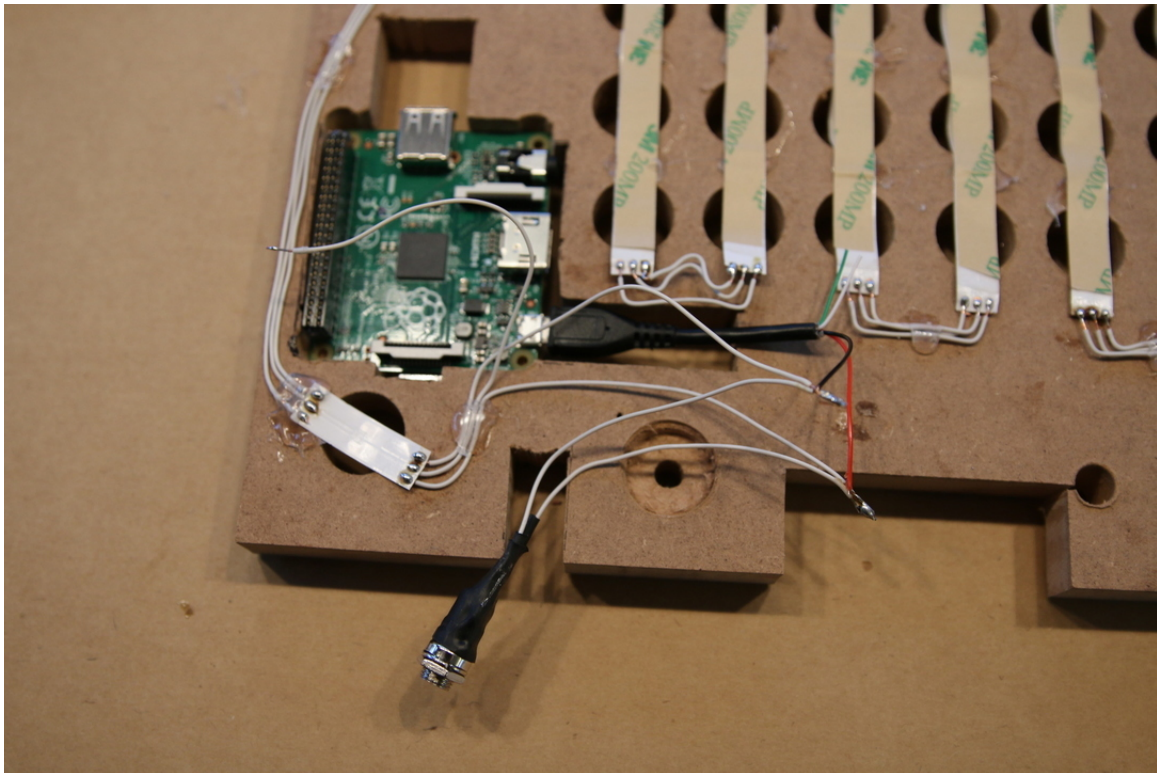


Fig. 8: The connection of buttons, LED strip and power brings the wordclock close to its final hardware configuration.

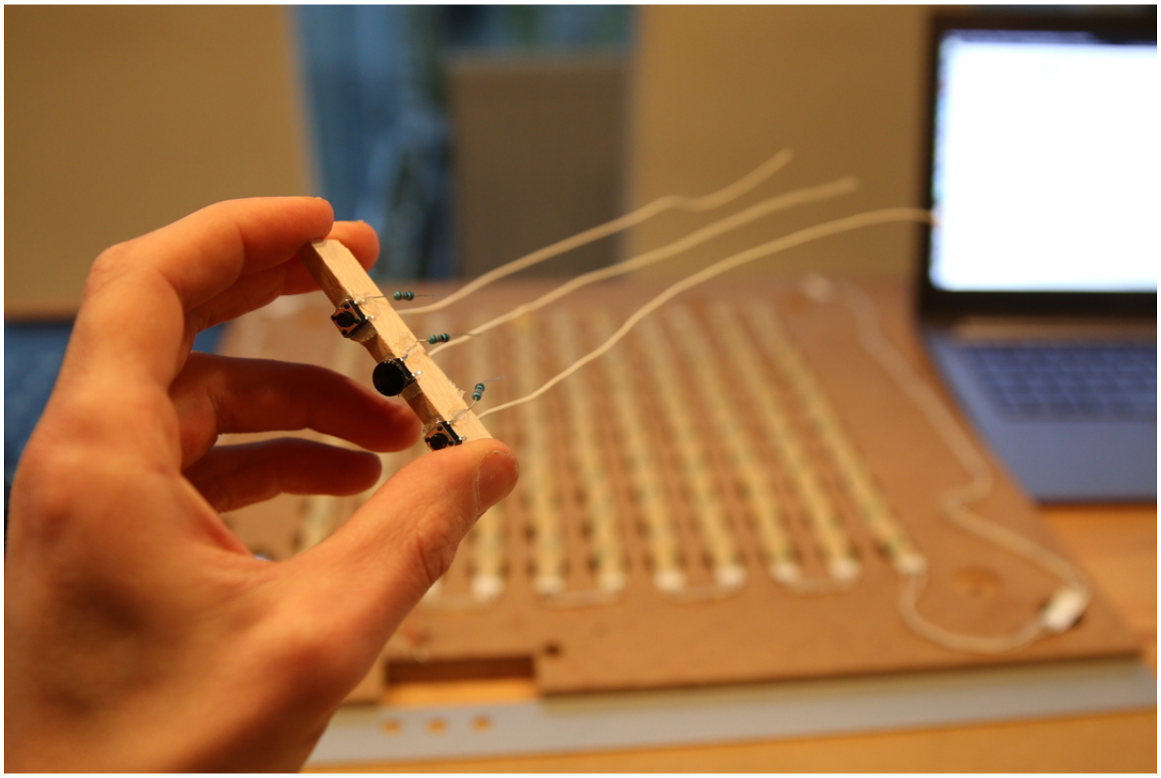


Fig. 9: Buttons with attached resistors. The center button has already its final tip.

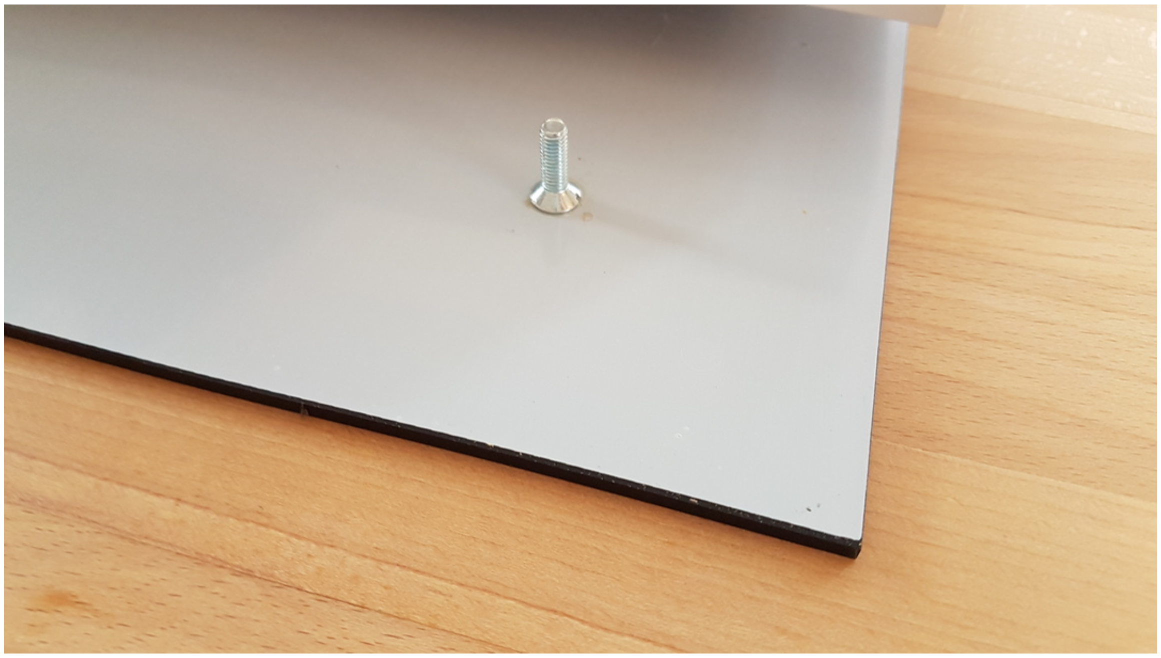


Fig. 10: Fixation of 4 screws within the 4 corners of the stancil using two-component adhesive.

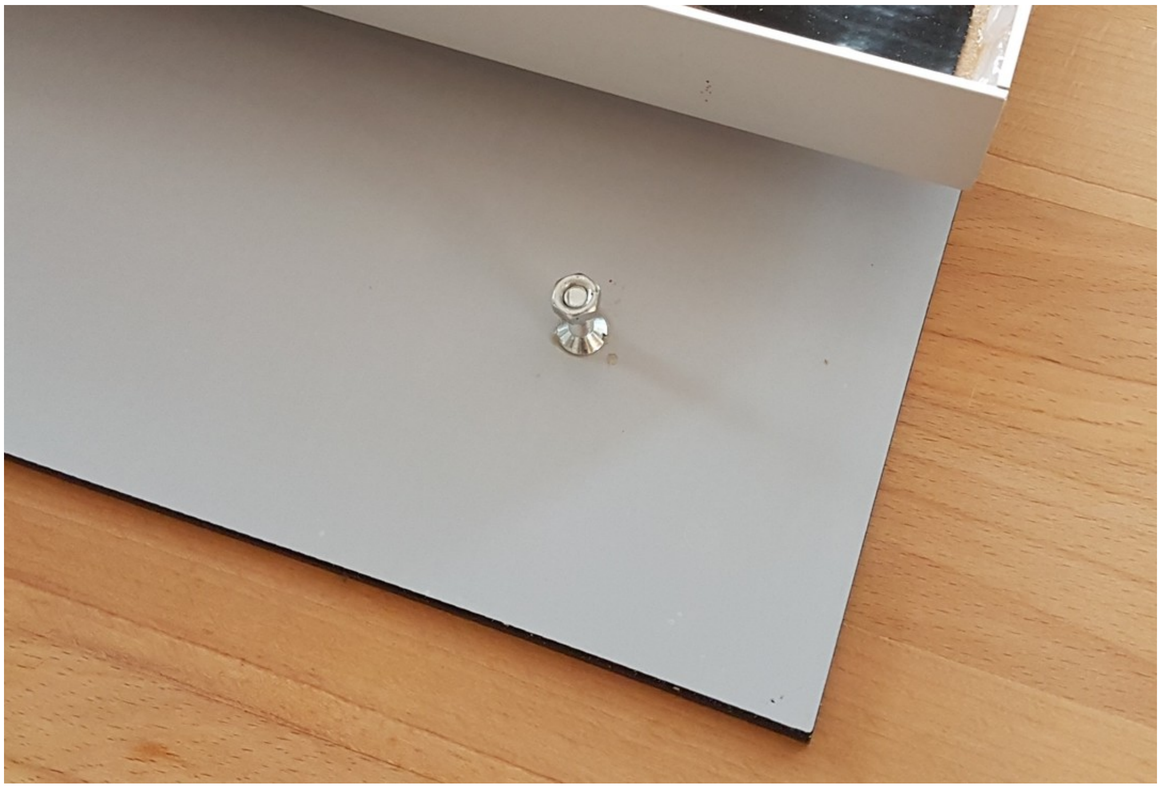


Fig. 11: Allows to fix the stancil with screw-nuts to the sceleton.

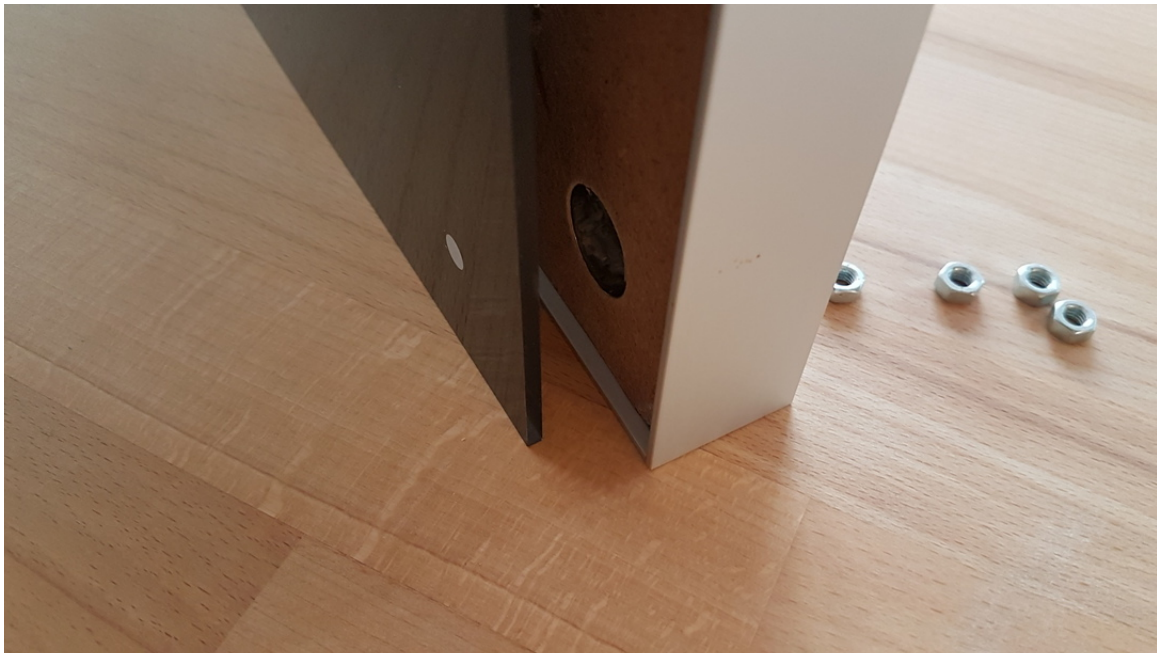


Fig. 12: To increase stability, consider an overlapping of the frame to hold the major weight of the stancil.

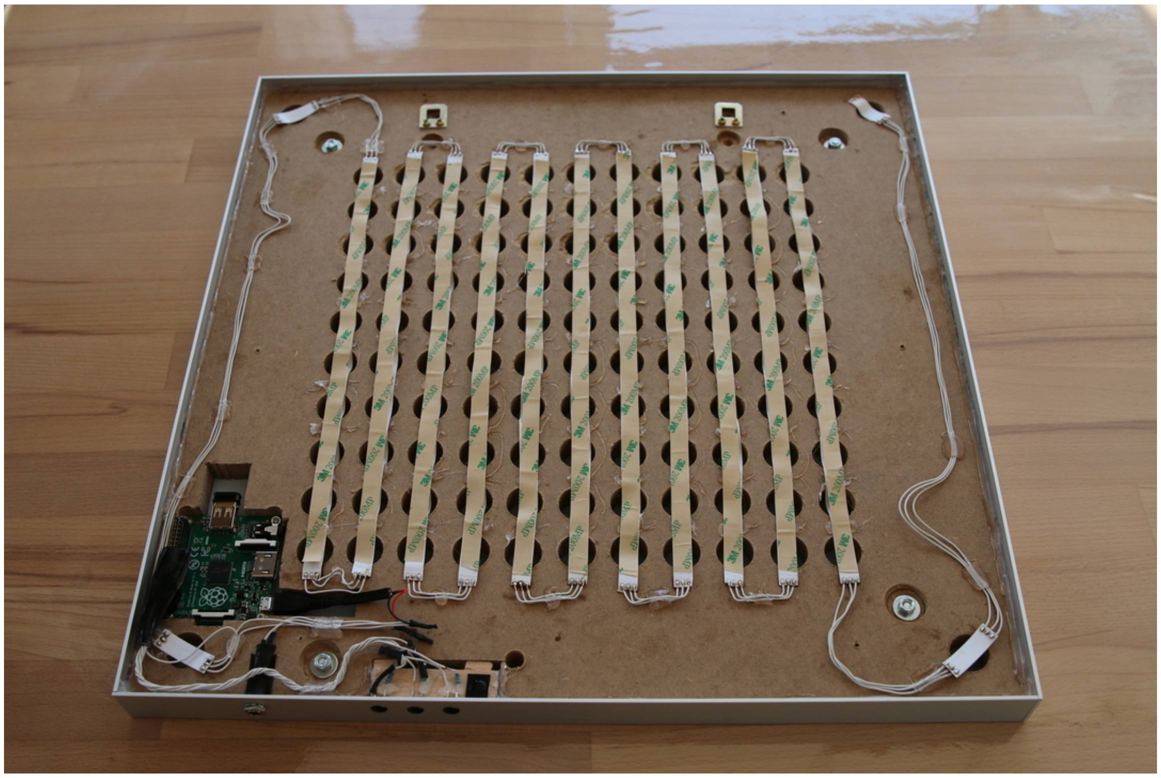


Fig. 13: Backside of the final wordclock.

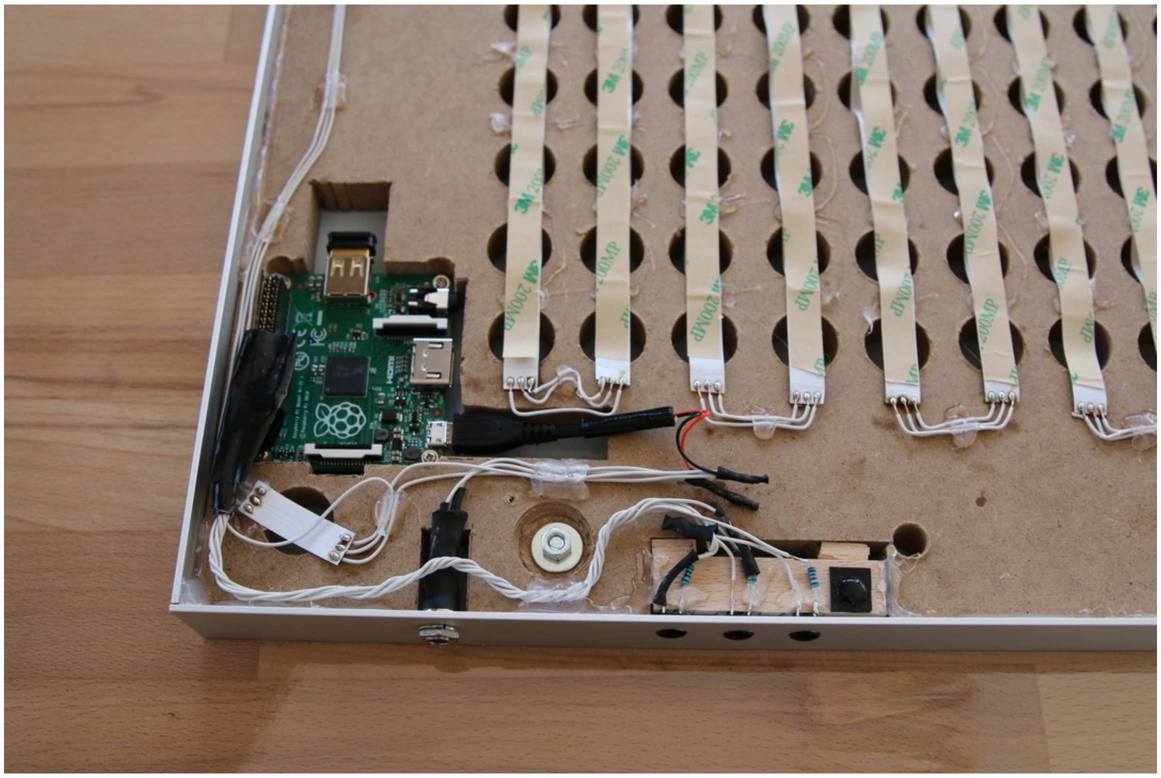


Fig. 14: Closeup of the final wordclock.

CHAPTER 3

Software setup

3.1 Setup your raspberry pi

Setup your raspberry pi (e.g. using Raspberry Pi Imager) by downloading and installing latest Raspian on the SD card.

During the installation process, configure the location according to your needs (incl. time zone, etc.)

3.2 Set locales

Since the config-file contains layouts for multiple languages, assure to have a utf-8 compatible locale after setting up your raspberry:

```
echo $LANG
```

should return something, containing utf-8 at the end. E.g.:

```
en_US.UTF-8
```

If not, check this website, to adjust it: http://perlgeek.de/en/article/set-up-a-clean-utf8-environment

3.3 The wordclock software

3.3.1 Download software

Clone the wordclock software to the directory ~/rpi_wordclock (to run the actual wordclock):

```
cd ~
git clone https://github.com/bk1285/rpi_wordclock.git
```

3.3.2 3rd party dependencies (packages)

To install 3rd party dependencies (packages) enter in a terminal/commandline:

```
sudo apt-get install python3-pip python3-scipy scons git swig fonts-freefont-ttf␣
↪libopenjp2-7
```

3.3.3 (Optional) dependencies to readout temperature sensor

To read out an temperature sensor (AM2302), which can additionally be connected to the raspberry via GPIOs, install the according dependencies:

These dependencies are http://www.airspayce.com/mikem/bcm2835/index.html

and:

```
sudo pip install am2302_rpi
```

3.3.4 3rd party dependencies (python packages)

To install 3rd party python dependencies (packages) run:

```
cd ~/rpi_wordclock
sudo pip3 install -r requirements.txt
```

3.3.5 Adopt software

To adjust the wordclock to your own settings, create and edit the file ~/rpi_wordclock/wordclock_config/wordclock_config.cfg

To start over, you might just copy the file ~/rpi_wordclock/wordclock_config/wordclock_config.example.cfg and adopt this file.

Note: Each plugin of the wordclock project has its own section in the config-file (create it, if needed, but not existant)

Note: If your wordclock has a stencil layout or display resolution, which is not supported yet, you might need to adopt the software by providing your own *wiring*-class (to the file wordclock_tools/wiring.py)

3.3.6 Run software

To run the wordclock software (with adapted wiring and config-file) do:

```
cd ~/rpi_wordclock
sudo python3 wordclock.py
```

In case, the whole thing is not working as expected: Maybe the section trouble-shooting might help…

3.3.7 Make software run on every startup

Add the python-script to crontab by calling the command:

```
sudo crontab -e
```

Add here:

```
@reboot sudo python3 /home/pi/rpi_wordclock/wordclock.py
```

3.3.8 Access the wordclock via webinterface

Visit the wordclocks webinterface by entering the wordclocks IP to your browers address bar.

CHAPTER 4

Further reading

4.1 Concepts and background

- WCA (Word Clock Array): The center matrix, without minute-LEDs and other stuff
- WCA_WIDTH, WCA_HEIGHT: Height and width of the WCA.
  - Part of the wordclock software are png-files, which need to fit to these values.
  - Currently available: 11x10 png-files.
  - Support for wordclocks with other resolution available (untested).
- WCD (Word Clock Display): Includes any led attached to the wordclock (such as minutes, possible/future ambilights/etc.)
- Coordinates (or: WCA-coordinates): Can be 1d or 2d, used to adress a LED on the word clock array
- Index (or: strip index): Used to adress a LED depending on the position on the LED-strip

4.2 Expanding the functionality of the wordclock

4.2.1 Remote control of the wordclock

The wordclock comes with a REST-API to control the major functionality of the clock.

To access the API documentation, visit:

```
http://wordclock-ip/api
```

4.2.2 Adding a new plugin

You might be interested in expanding the wordclocks functionality by adding a new plugin to the wordclock

To do so, you need to…

- Think about the name of this plugin: E.g. *new_stuff*
- Add a new folder *new_stuff* to the folder wordclock_plugins
  - Create a plugin.py-file with a class *plugin*, which has at least the following functions implemented:
    - __init__(self, config): You can use the config-object to pass data from the config-file for initialization purposes
    - run(): Run the actual plugin
- For the actual implementation, you can access the provided methods of the class *wordclock_display* * If necessary you might extend it… ;)
- Add an icon (with resolution 11x10 pixel) for the new plugin to the directory wordclock_plugins/*new_stuff*/icons/11x10/*logo.png*
- Add a section *[plugin_new_stuff]* to the reference config-file (wordclock_config/wordclock_config.reference.cfg) and store there all necessary config-values in a way that they are suitable for all wordclock users by default and out of the box. * Even consider disableing your plugin by default.. * Disabling is mandatory, if additional hardware is required to run the plugin.
- Add the same section *[plugin_new_stuff]* to your local config-file (wordclock_config/wordclock_config.cfg), holding the (custom) values you want to have setup for your own clock. * Your plugin will extract config values from wordclock_config.cfg first. If they are not set their, it will default to wordclock_config.reference.cfg
- Document everything properly, so that others (and maybe you as well) can later understand it… ;)
- Commit your changes using git and consider to create a pull-request at https://www.github.com/bk1285/rpi_wordclock
- Consider, that this repository uses nvie's branching model: http://nvie.com/posts/a-successful-git-branching-model/

CHAPTER 5

Trouble shooting

Something is not working?

- The command:

  ```
  sudo pip install pytz astral feedparser pillow svgwrite freetype-py
  ```

  fails to install properly? If so, try to install further dependencies (thanks to SEBatHome):

  ```
  sudo apt-get build-dep python-imaging libjpeg8 libjpeg62-dev libfreetype6␣
  ↪libfreetype6-dev
  ```

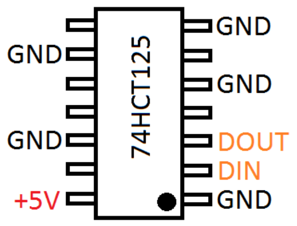
- The leds do not light up as expected?
  - It is important to have common ground for LEDs and RPi. Assure, ground is same for all of them (Thanks to euchkatzl).
  - Assure to connect the LED strip in the right direction. Little arrows indicate that along the strip (Thanks to euchkatzl).
  - Assure correct functionality of leds:

    cd ~/rpi_ws281x/python/examples vim strandtest.py # Set number of leds, pin, etc. sudo python strandtest.py

    The leds should light up now…
  - Disable the RPis soundcard (since it might interfere with the PMW-channel, sending data to the LEDs. Thanks to ELViTO12 for reporting):

    ```
    sudo sh -c "echo blacklist snd_bcm2835 >> /etc/modprobe.d/alsa-
    ↪blacklist.conf";
    sudo reboot;
    ```

  - In case the LEDs are flickering as shown in this video https://www.youtube.com/watch?v=UHxVS8SkXOU (Thanks to oxivanisher), consider the usage of a level-shifter to connect the GPIO-pin of the raspberry to the LED-strip. Further reading: https://github.com/jgarff/rpi_ws281x/issues/127 https://github.com/bk1285/rpi_wordclock/issues/38



- When starting the wordclock-script, "Pin 17 pressed" is logged all the time?

  To get rid of this message, you first need to finish the wordclock setup by attaching all 3 buttons to it.

  If you aim to run the wordclock without buttons, change the config-file settings as follows:

  ```
  [wordclock_interface]
  type = gpio_high
  ```

Note: The provided information might be completely unsatifying, leaving you here frustrated and annoyed without a working wordclock. . . :/

However, if you have any issues during the setup, consider:

- To update the provided documentation (or this trouble shooting section), as soon as you resolved your problem.
- To report a software issue here: https://github.com/bk1285/rpi_wordclock/issues

CHAPTER 6

Further documentation

Two youtube videos, which outline the main functionalities and features are available here and here.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

CHAPTER 8

Acknowledgements

- Christian (idea and first efforts for realization)
- Daniel and Markus (technical support and hints to make the project advance)
- Jeremy (providing the great rpi_ws281x-library)