# Wordclock Documentation

***Release latest***

**Apr 08, 2023**

# CONTENTS

CHAPTER

ONE

# INTRODUCTION

Here you will find information to build a wordclock inspired by QLOCKTWO (https://qlocktwo.com). The original concept has been adapted by Bernd Krolla into a elaborate python program running on Raspberry PI (https://rpi-wordclock.readthedocs.io). In a search for a more affordable alternative I made thorough alterations (in other words: started over) to run a wordclock program on an ESP8266 WiFi chip, running micropython. Where Krolla's program contains many plugins for showing the weather en even playing Tetris, I focused on the simplicity by only setting and showing the current time. In this documentation I wil give you insight in the process to build such a clock.

## 1.1 Overview

As mentioned in the introduction, this wordclock is based on the Bernd Krolla implementation of the original QLOCKTWO. Krolla's wordclock runs on a Raspberry Pi. I was searching for the most affordable implementation of a wordclock, therefore I ended up with a minimum of electrical components. The ESP8266 microcontroller is mostly shipped with firmware to run Arduino language. However, it also runs a port of Micropython which works great for this type of project.

To show the time in words, a led strip is placed behind a laser cut matrix of characters. The hardware can be divided in woodwork and electrical components. After some iterations I ended up with a wooden frame made out of laser cut slats that make sure if a led lights up, adjacent characters will not be illuminated.

The electrical components consist of a ESP8266 NodeMCU microcontroller, DS3231 Real Time Clock to remember the time (the ESP8266's internal clock is not reliable), TSL2561 luminocity sensor to adjust the brightness of the clock to the ambient light. Three small push buttons take care of setting the time and adjusting the color of the leds. A 5v DC power supply of 2A should be sufficient but I use a 3A version to be sure. The leds used are a WS2812B adressable led strip, in which each led can be controlled separately. The components are wired together with electrical wire and led strip connectors (to save a lot of soldering). Due to a maximum of laser cutting, it takes me less than 5 hours to assebmle a clock.

Wordclock

### 1.1.1 Normal operation

When the power adapter is plugged in, the ESP8266 will boot in about 5 seconds. When it is the first power up, the clock shows the time since putting the battery in the RTC. To change the time, hold the OK (middle) button until "HET IS" flashes 3 times. The clock wil enter time setting mode, which is a 4 step proces elaborated in tools.py (function set_time()).

1. Set the time in hours using de back and forward button. When the correct hour is selected, push the OK button, the time in hours will flash.
2. Set the time in 5 minutes using the back and forward button. When the correct time (rounded down to 5 minutes) is selected, push the OK button, the time will flash.

3. Set the exact time for the minute leds using the back and forward button. When the correct time is selected, push the OK button, "HET IS" and the minute leds will flash.
4. You now can change the yellow value of the clock using the back and forward button. At default the led color in RGB is 255,255,50 but you can change the 50 to the desired type of white to yellow. Push the OK button when ready. The clock wil turn off for 2 seconds and then show the time in the new color.

When you pull the plug of the power supply, the RTC will keep track of the time using the CR2032 battery. Once power is restored, the correct time should show up.

To compensate for daylight saving time, push the back or forward button once to change the current time by one hour.

### 1.1.2 Shopping list

Electrical components (total about €50):

- ESP8266 (NodeMCU) microcontroller
- DS3231 real time clock (RTC)
- TSL2561 luminosity sensor
- SN74AHCT125N (or similar) logic level shifter
- 3A 5V DC power supply
- Ledstrip with 114 WS2812B leds (30 leds/m)
- 13 led strip connectors
- 1 female DC plug
- 3 small push buttons
- 1 300-500 Ohm resistor
- Electrical wire (22 to 24 AWG)

Woodwork (total about €50 to €80):

- Laser cut front plate of 450x450mm
- Interior matrix made out of laser cut slats
- Slats for the border (could also be laser cut if desired)
- 4 small blocks to fixate screw on back plate
- Back plate (+- 440x440x4mm, for example plywood)

## 1.2 Hardware setup

### 1.2.1 Woodwork

The front of the wordclock shows a matrix of 11x10 characters and 4 dots for indicating the minutes. Behind the characters, regular printing paper takes care of diffusing the light from the leds. Due to the required level of detail, I ordered a laser cut front plate from plywood. The dimensions are 450x450 millimiters. I use a thickness of 2 millimiters to ensure that a wide viewing angle on the clock is possible. See a template in the downloads section.

At the back of the front plate, a raster of slats hold the led strips in place. For a prototype I sawed these slats by hand, but a laser cut is a lot more precise and easier (3mm MDF). See a template in the downloads section. The raster has a height of 18mm. In a prototype I used a height of 15mm, but when de leds are placed 15mm behind the front plate, the middle of a character is a bit more bright than the edges. Therefore I use 18mm height now. The slats in the laser cut template are numbered from 1 (top) to 11 (bottom).

For the outside border I use slats plywood of 4mm thickness. The lengths are 450mm for the left and right side, 450-2x4=442mm for the bottom and 2 times 191mm for the top. In the middle a open space is present for the 3 buttons. The template for the interior matrix contains parts for holding the buttons which can be glued under this cutout. The outside border should be as high as the height of the raster (for example 18mm) + few millimeters for the led strips and tape + the thickness of the back plate (for example 4mm). Total height should be about 24mm in my example.

The back plate consists of a plywood plate of 440x440x4 millimeters with some cut outs and holes in it.

**Steps for creating the woodwork:**

1. Paint or varnish the front plate and border slats in a color or varnish of your choice.
2. Cover the backside of the front plate with paper. You can fixate it with regular tape. See picture 1.
3. Put together the matrix of slats for the interior and glue it to the back of the front plate. See picture 2 and 3.
4. Put together the pieces for the minute-leds. Glue them to the back of the front plate. See picture 2.
5. Solder all electrical components and mount them in the woodwork, see chapter electrical setup. The template for the buttons can be used to hold them in place and glue them to the front plate when alle electrical components are ready to mount.
6. Glue the 4 blocks to the front plate for screwing the back plate to the frame. See picture 4.
7. Place the back plate. Make sure you have a cutout for the DC socket, 4 screwholes for screwing the back plate to the glued blocks and 1 hole for hanging the clock to a wall. See picture 5.

### 1.2.2 Electrical setup

The fritzing scheme (picture 7) provides an overview how to solder the components. It is most easy to separate the power circuit, led strip connectors and wires for communication.

**Power circuit:** It is important to understand that the led strip has contunious circuits for +5v, ground an data transfer. A WS2812B led can draw up to 60mA, which can add up to quite some current. Due to the small wire gauge in the led strip the resistance will grow and cause voltage drop. The leds at the end of the strip will be dimmer than at the beginning. This can be prevented by sourcing power to the led strip at multiple points.

The 5v power is sourced by the power supply through the female DC plug. After the plug the power wires split into a circuit to the bottom for sourcing the led strip at multiple points. The +5v and ground for the led strip do not have to be continious, the data wire should be.

The other circuit feeds both leds on the left side (seen from the back) and provide the microcontroller, RTC en luminocity sensor with 5v DC. Also the buttons need to be connected to ground on one side. The microcontroller can be sourced by a micro USB plug (with 5v and ground soldered to the wires) or you can use the Vin- and a ground pin. The breakout boards (RTC and luminosity sensor) can also be sourced direct from the power circuit. Some RTC provide soldering points to daisy chain the wires (SCL and SDA for the I2C bus, 5v and ground). You could use a 3v3 output from the microcontroller to source the RTC, if desired. A TSL2561 requires 5v to function properly.

**Led strip connectors:** I use 3 pin ledstrip connectors to easily connect led strips to each other without soldering (for example https://nl.aliexpress.com/item/32966732241.html). To connect the minute leds in the left and right corners, I extended the wires of a led strip connector. Further, connector wire together adjacent led strips. On the bottom, on 3 locations the 5v and groud wires are stripped to solder the power wires to avoid voltage drop.

**Data circuit:** From the microcontroller, several data wires run to the components. I recommend to solder all the wires (use an ESP8266 without GPIO header) to reduce required space and prevent errors due to wires slipping off. See picture 7 for a schematic.

1. D8 (GPIO 15) is the data in wire for the ledstrip, running to the logic level shifter to shift the logic signal from 3.3v to the required 5v of the WS2812B leds. My clock works without this level shifter, but when the voltage drops a little the signal could be disrupted. From the logic level shifter, a wire runs through a small 300-500 Ohm resistor to the data in wire at the led on the left bottom (seen from the back)
2. D1 (GPIO 5) is SCL, serial clock for the I2C interface, running to the RTC first and continues to the TSL2561 sensor. These breakout boards can also be connected to parallel to the I2C pins.
3. D2 (GPIO 4) is SDA, serial data for the I2C interface, runs alongside the SCL wire to the RTC and TSL2561.
4. D5 (GPIO 14) is the back botton, most right seen from the back. This pin is initially pulled up by the program and is pulled to ground when the button is pushed.
5. D6 (GPIO 12) is the OK button, in the center of the three. Also pulled up by the program.
6. D7 (GPIO 13) is the left button, most left seen from the back. Also pulled up by the program.

It is wise to check for short circuits with a multimeter (+5v should not be in contact with ground). Also make sure that the data wire continues in the same zigzag pattern so that numbering of characters corresponds with the program.

### 1.2.3 Mounting electrical components

With all components soldered together you can place them in the woodwork. All electrical components can be mounted with a glue gun. See picture 4 for all componentes. The order of installation I perfer:

1. Microcontroller with RTC and TSL2561. It is recommended to write the program to the microcontroller first and test before glueing it together. See page about software setup.
2. Power circuits from the chips to the power plug
3. Power circuit at the bottom of the clock for sourcing the led strips
4. Minute leds can be glued in the pieces. Cover them with duct tape to prevent leakage of light.
5. The wiring can be glued to the front plate or matrix for better fixation
6. The leds strips, cut in strings of 10 leds, can be connected by led strip connectors and glued to the cut outs in the top and bottom slats (nr. 1 and 11). It could be wise to extend the strip with one length a time and test if leds do light up if you plug the power supply. This enables finding wiring errors. See picture 6 for a wiring diagram.
7. When all led strips are mounted, cover the back with duct tape. This prevents leakage of light from one character to the other.
8. Test if your clock works and mount the back plate. See step 6 and 7 in woodwork.

### 1.2.4 Pictures

### 1.2.5 Downloads

- Laser cut template for the front plate in Dutch
- Laser cut template for the interior matrix

## 1.3 Software setup

The program for the ESP8266 runs on micropython. See next chapter for more information how to program the python files (https://github.com/robsloetjes/wordclock) to the microcontroller.

On power up, the ESP8266 loops through main.py, which contains the main program.

The modules you need to run the wordclock work as follows:

- main.py - Program starts with initiating the different parts/modules. When started an infinite loop is run through in which the program checks if a button is pressed (to change the time). If not it updates the time. When the time (in hours and minutes) is unchanged, nothing happens.
- display.py - Module that takes care of translating the current time into ranges of leds that need to light up (currently Dutch matrix layout only). It also interacts with the led strip using module ledstrip.py.
- ledstrip.py - Makes shure that the leds will light up, also in the right color and brightness. Uses values from config.py and illuminance values from the TSL2561 sensor using tsl2561.py
- tools.py - Contains several modules to get and set the time and led color.
- config.py - Contains variables which mostly can be altered to change some functionalities.
- tsl2561.py - Driver for the TSL2561 luminosity sensor.
- DS3231.py - Diver for the RTC which tracks the time more precise than an ESP8266 can.

### 1.3.1 Writing program to ESP8266 & debug

The program for the ESP8266 runs on micropython. See https://docs.micropython.org/en/latest/esp8266/tutorial/intro.html for more information and the firmware. The firmware can be flashed by the ESPtool (https://github.com/espressif/esptool/), but I use a program called uPyCraft (https://randomnerdtutorials.com/install-upycraft-ide-windows-pc-instructions/) to burn the firmware and write the program to the microcontroller. Use the option 'Burn firmware' to write the Micropython firmware to the ESP8266. The computer needs to be connected by USB to the ESP8266. When the firmware is burned, create a serial connection through the COM port (e.g. using uPyCraft).

Copy the python files to the workSpace folder or create new files and copy the content. Each file needs to be written separately to the ESP8266 (Tools -> Download).

When downloading is complete, push the reset button on the ESP board. While the serial connection is live, you can read the print outs of the program, which makes debugging possible. It is therefore recommended to test the program with all components connected to the microcontroller before glueing it all into the frame.

## 1.4 Troubleshooting

- Wordclock doesn't turn on

Probably there is something wrong in the power circuit. Check with a multimeter if the there are no short circuits and all parts get 5v DC supplied. When the power supplie is plugged in, the onboard led of the ESP8266 should flash once. When the power circuit is not the problem, a software issue could be present. You can connect the microcontroller to a computer and read out the serial output of the program (for example use uPyCraft). Be aware that you keep the power supply plugged in, since the ESP8266 cannot provide enough power when accidentally many leds light up.

- The ledstrip only partially works

Or the data wire is interrupted (between the parts that work and don't) or a part of the led strip recieves no or not enough power.

- The clock shows strange colors

Probably a small shortage is present beween the 5v wand the data wires, or the led strip gets insufficient power at some point to pass through the information in the data wire.

- Some characters are more dim than others

Too much voltage drop is present. Check if you source the led strip with 5v DC at multiple points.

- Wrong characters light up

Does the wiring follow the diagram as provided in the hardware section?