WordClock

A clock that shows the time using words

Contents

History

Word clocks are kind-a cool, weird, quirky. Nerd stuff. I've seen them around for a few years now, and you can buy them for a few hundred dollars. They come in all sorts of sizes, shapes (usually square though), materials, colours and – sometimes – languages.

g00gle/bing/duckduckgo "word clock" or "word +clock" and you'll find them on the internet.

But buying is only half the fun. Building your own is better – you have more control over what you want and what you get – size, shape, colours, and built-in functionality. Raiding ebay™ for parts is much more fun than buying a complete clock that then perhaps doesn't know how to deal with your time zone or with your daylight savings settings.

A while ago we were travelling in Europe and saw a word clock in the local Bernese German language – so weird! Having lived in Switzerland for a long time (two decades!) we fluently speak the language and had no issues reading (or even pronouncing) the time from the clock. So, after returning home, I was told: "we need one of those". But also "in Zürich German, please" (we lived in Zurich, not in Berne).

Anyway, TL;DR: … here's a word clock in *Schwyzerdütsch* from Zürich, "made in Sydney".

The design is based upon Manuel Meister's clock. I saw his blog page (see links list at the end of this document) and had a look at his code page (see links) which was a great starting point. Over a few iterations, the code was rewritten from scratch, but many of the ideas from Manuel are good and so have been retained in principle, but most routines have been reimplemented in slightly different ways.

Design and functionality

I've kept the design relatively simple (there's no buttons, displays, weather stations, alarms, etc.) but the functionality is, relatively speaking, a bit more than just a clock. It features:

- Time (in words, in Swiss German, you can change this – it's a fun puzzle)
- WiFi access (credentials hard-coded, so be careful sharing your code)
- NTP time synchronisation (so it's always accurate)
- Timezone adjustment (default is for Australia/Sydney)
- Daylight savings time adjustment (automagic)
- Sunrise/sunset (shows if the sun is up or not)
- Moonrise/moonset (shows if the sun is up or not)
- NeoPixel gamma correction
- Dynamic (diurnal) brightness/contrast (bright during daytime, darker at night)
- Xmas tree (lights up 25 and 26/12)
- Easter egg (lights up on Easter, uses Gauss' algorithm)
- Special birthday icon

Below you'll find some instructions. Have fun making.

Take care. Relax, you've got time. Literally.

Bill of materials

Item	Source	AUD
uPesy ESP32 WRover DevKit C3 with WiFi antenna	LonelyBinary.com	$20.00
Adafruit Neopixel strip 5V, 30/m 3.33cm per pixel, 5m, 150 LEDs	ebay.com.au	$40.00
Connector 3 PIN RGB LED STRIP LIGHT SOLDERLESS WS2812B WS2811	ebay.com.au, 2 sets of 10	$40.00
AC 240V to DC 5V 2A Power Supply Transformer Adapter For LED Strip	ebay.com	$20.00
Plywood 897 x 600mm 7mm Pine BC Grade	Bunnings	$40.00
Dulux 340g Duramax Flat Black Spray Paint	Bunnings	$20.00
Cabot's 300g Oil Based Satin Cabothane Clear Polyurethane Timber spray varnish	Bunnings	$20.00
0.55mm #4 316 Stainless Steel 600x600mm, 2 sheets	Edcon Steel edconsteel.com.au	$110.00
Polypropylene Plastic Sheet Thickness 0.5mm Translucent Plastic Sheet Frosted	ebay.com	$25.00

Table 1 - BOM for a WordClock

Total costs approx. AUD 325.00

(AUD 270 for a high risk approach where you only order 1x steel sheet and have no spelling/grammar mistakes in your clockface layout)

Tools required

ruler, pencil, pen, tape measure, masking tape

hand saw, 45 deg mitre guide

circular saw or table saw

jig saw or small bandsaw

scissors

Araldite epoxy

hot glue gun

soldering iron + tin + flux

drill w/ 3mm bit and 13mm spade bit

a wood plane – electric is preferred

8-10 150mm quick grip bar clamps

sandpaper (320+ grit)

impact driver, wood screws 1/4x20mm

some electrical wire

soldering iron, tin, etc.

Metal capable laser cutter

optional - Wood laser cutter (7mm minimum ply)

Software

Arduino IDE, optional/alternative ESP-IDE

ESP32/Arduino libraries (refer to code)

InkScape, gimp, AutoCAD, etc. - SVG or DXF capable

Instructions

The clock is a stack of materials, from back to front:

- plywood back cover (has a little access door for ESP32, power cord, etc.)
- plywood spacers, approx 10mm wide, along the full edge
- LED array (12x12, 144 LEDs, wiring), ESP32, antenna, power connection, etc.
- plywood baffle/mask layer, 12x12 holes, approx 12-13mm diameter
- plywood spacers, approx 10mm wide, along the full edge
- stainless laser cut face plate

0. Design your clock face

The wordclock shows the time by showing it in words, something along the line of

"IT IS NOW TEN PAST SEVEN" ... depending on your preferences, language, whims, ideas, etc.

there are options for various variations. You'll need to take care with the correct order of the words. In general, the structure (in English) is a bit like this: "IT (IS [NEARLY]) | (HAS [JUST] BEEN) {MINUTES} | (QUARTER | HALF) (TO | PAST) {HOURS}"

The words "IT IS" would nearly always light up. If you have space (LEDs) you can also do "IT HAS BEEN". In Schwyzerdütsch, the correct way would be: "Äs isch bald Viertel vor Zähni" (it is soon quarter to ten) or "Äs isch Füf ab Halbi Achti gsi" (it has been twenty-five to eight), so the grammatical order is slightly different. It will be different again in French, Dutch, Swahili, etc.

The minutes change (nearly) every minute. Sometimes, you do not have enough space (LEDs)

to spell out all the minutes, so then you can leave out the one or two surrounding the "QUARTER" or the "HALF" hours, and it becomes something like "IT IS NEARLY QUARTER PAST THREE" when it's say 3:14 or even 3:13.

You'll need to juggle the ONE, TWO, THREE, ... TWENTYNINE minutes around such that they fit on a 12 point grid, or you need to consider a larger (13x13, 16x9, ...) grid instead. Sometimes you can borrow/share letters: SEVENINE for SEVEN and NINE, or EIGHTHREE for 8 and 3. Depending on your language/dialect, this may be easy or hard (Gaelic or Finnish or Romanian are probably hard). You usually do not need to have spaces between the characters, unless you get close to the TO/PAST words: QUARTERTO does not work, QUARTER_TO does work visually.

Sometimes you must have a space and there you can then choose a funny character like an easter egg, xmas tree, or - really tacky, I admit - a love heart that lights up in red on your partner's birthday.

The code here uses a 12x12 clockface, 144 LEDs, which only just worked for me. You can go full modern digital HD crypto AI with a 16x9 (also 144 LEDS). You can be more restrictive with say 9x11 or 10x10 or whatever. 4x4 probably will not work. Either way, you'll need to buy the right number of LEDs on a strip, and you'll need to adjust the code.

The clock was designed in Inkscape, and saved as scalable vector graphics - this is easily compatible with most laser cutters, the proprietary software there can usually import SVG and convert it into DXF, STL, Gerber, HPGL (remember that, anyone?) or whatever other vector format without losing any precision. Font line widths are important - you probably do not want pixelated/aliased characters on your face. You can use gimp, autocad, openscad, etc., whatever you like. MS paint is probably not good.

Pick you size - NeoPixels come in various pitches with 30/m (3.33cm per pixel) and 60/m (1.67cm/pixel) most frequent, but you can get them with lower or higher densities. Choose wisely. This design uses 30/m ones. A 5m strip contains 150 LEDs, just enough for the 144 we need.

1. Build ESP32/Neopixel strip

Connect (solder) Neopixel strip to 5V power supply.

Feed (solder) 5V/GND from there to ESP32.

Connect (solder) DATA wire from Neopixel strip to ESP32 pin 11 (NEO_PIN GPIO 27)

See e.g. https://esp32io.com/images/tutorial/ESP-WROOM-32-Dev-Module-pinout.jpg for pinout.

Test with some example code (make sure you adjust NEO_PIXELS and NEO_PIN numbers correctly).

Running all 150 LEDS of a new strip simultaneously at full brightness from a USB port may not work and you'll either send the ESP32 into a crash-and-burn boot loop, or you'll fry your device USB port, whatever you prefer, it's going to be ugly.

For now, leave it as is - cut up into pieces of 12 and rejoin with the connector wiring later.

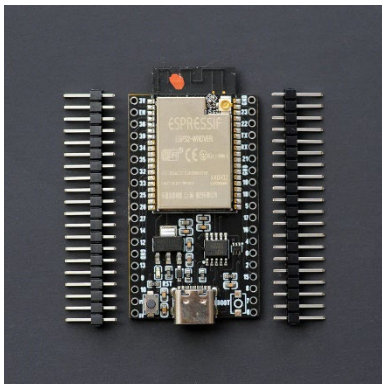


Figure 1 ESP32 WRover DevKit C3.

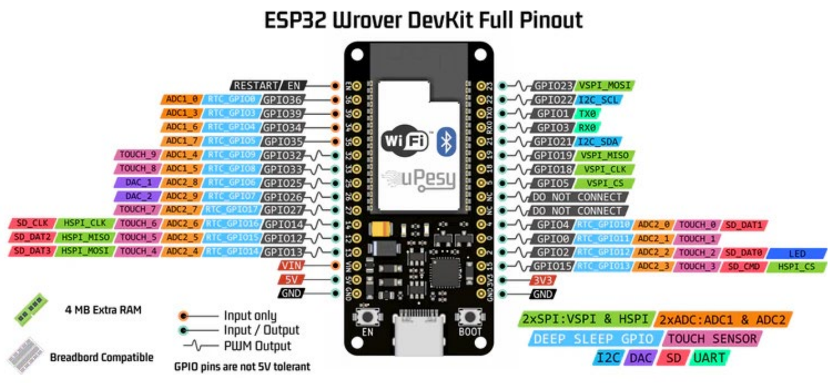


Figure 2 Pin-out for the ESP32 DevKit C3 device.

Note that the LonelyBinary "King of ESP WRover DevKit C3" does not have the exact same pinout as the uPesy WRover DevKit C3, although it identifies as the same in the Arduino IDE. The LonelyBinary version has a few additional pins on either end (38 pin board vs 32 pin board). Check your pinout.

2. Make the baffle/mask plate

Cut one piece of plywood to 500x500mm, ensure it is square and accurate, slightly larger size is better than slightly smaller. Keep the offcuts. Precisely mark centre points for a 12x12 array of 3.33cm spacings in X and Y, centred on the plywood. Hammer the marking down with a nail or with a centre punch. Drill a ~ 3mm hole in each hole. Carefully enlarge each hole with a ~13mm spade drill. Remove tear-outs, using some fine 320+ sandpaper. Optionally, if available, make holes using a wood laser cutter. Holes should be around 12-13mm diameter.

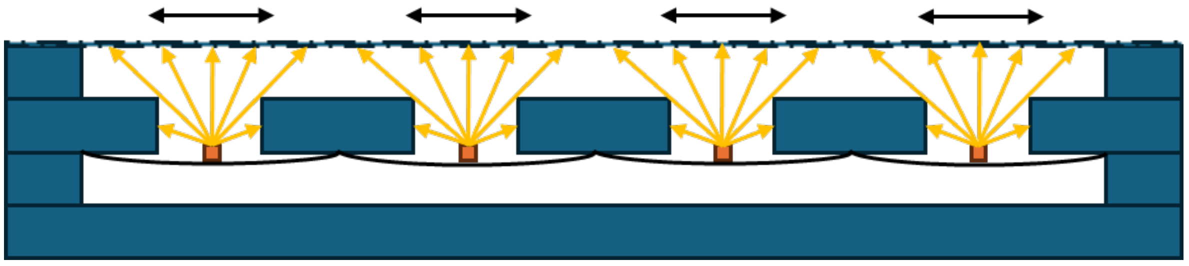


Figure 3- Baffle structure. This will help limit the light emission from one LED into the adjacent characters on the word clock. A 7mm plywood layer with 12-13mm holes will give an approximately 45 degrees illumination cut-off angle for the NeoPixel LEDs. A second 7mm spacer will define the illuminated diameter at approximately 25-30mm, or just one character.

3. Add spacers

From the ply offcuts, cut eight 1-1.5cm wide strips of wood. Mitre the ends at a 45 deg angle, ensuring that the overall length is 500mm.

Glue these along the edges, 4 on each side, and press down with two quick grip bar clamps each. Let dry overnight, remove quick grip bar clamps, sand lightly along the edges.

4. Spray painting

Spray matte black spray paint on both sides of the panel with the 144 drilled holes, and on the inside of the back panel. Ensure that the inside of the 144 holes is also fully covered. Let dry overnight.

5. Mount LEDs

The following is best done one cut at a time.

Using scissors, cut the 150 LED neopixel strip into small strips of 12. If you have

chosen a different layout, adjust accordingly. From the power-supply side of the strip, count 12 LEDs, then cut carefully across the centre of the exposed set of 3 copper soldering contacts, leaving half a contact pad on either side. Rejoin the two strips (small and remaining long one) with the crimp connector cables. Make sure that GND remains GND, 5V remains 5V and DATA remains DATA. Press the caps on the crimp connectors hard, contact needles must penetrate the copper of the stip.

Test the strip before you continue. Repair/fix/replace if needed.

Repeat for the other 11 cuts, connect the other 10 crimp connector cables every 12 LEDs. Store the leftover 6 LEDs somewhere else. Maybe you have an idea for them. Or a shelf.

Once the extended 144 LED string works, us the hot glue gun to afix it to the back of the baffle panel.

Flip the panel upside down such that you are looking / working at the backside.

Start with the first powered LED (and the ESP32 and the antenna) in the top right corner, and afix it to the wood with a hot glue gun, such that the first LED is in the centre of the first hole.

Drip a small blob of hot glue in between every pair of two holes and carefully (hot!) press the Neopixel strip into it. Do NOT glue the LEDs themselves, put the glue halfway in between two.

Make sure the strip is carefully aligned, with each LED in the centre of a hole.

After 12 LEDS, fix the crimp connector to the wood with more hot glue.

Bend the strip around 180 deg and again glue the next crimp connector to the board, such that LED 13 is now in the centre of its hole. Fix the strip between every pair of LEDs, repeat for LEDs 14-24.

Repeat this process, zigzagging the strip from right to left, one down, from left to right, one down, etcetera, until all 144 LEDs are in place.

6. Cut clock face

Cut the clock face design out of the thin 0.55mm stainless steel, using the laser cutter.

Some data massaging from SVG to DXF to KDF to ... internal format is required. Make sure you leave sufficient edge around, to be able to cut the 500x500 edge line as well. Before cutting, ensure all the lead-in lines are located INSIDE the letter paths, otherwise they will be visible in the clock face later. Make sure laser parameters are properly selected for each layer.

Some letters (especially the funny characters but depending on your language others may do as well) will show signs of overheating/oxidation etc. after the cut. Careful work with very fine (380+) sandpaper, or with eg. a polishing brush on a Dremel may help to clean that up.

7. Translucent frosting

Cut the polypropylene (or PVC or polycarbonate or similar) frosted sheet to size, such that you can cover the entire backside of the clock face with them, without going across individual characters.

Cutting the sheets at 20x20cm will give you 4 sheets when you use a 3.33cm Neopixel strip.

Clean the metal clock face with a lightly soapy water, rinse with clean water, and dry.

Carefully use a little bit of well mixed Araldite epoxy to glue the sheets to the backside of the clock face, directly on the metal.

8. Final assembly

Make sure the ESP32 and the antenna are secured to the backside of the LED panel. You may need another blob of hot glue there to hold them in place.

Force all the crimp connector wires to be within the raised edges of the panel – this will be a little stiff, but it will be OK. Then close the backside, making sure the power connector can come out fine somewhere at the bottom – you choose where you want that.

Close the back cover and place four screws, one in each corner. Make sure they are long enough to reach the LED panel through the edge strips, a 20mm timber screw should be fine, 25 works as well, 30 is probably too long.

Using some Araldite epoxy, carefully glue the stainless-steel faceplate to the front of the clock, making sure the characters are aligned with the LEDs. Use quick grip bar clamps to hold the face plate in place while the epoxy dries (24h). Drill a very small hole in each corner of the face plate, and make sure it's just the metal – do not drill deep into the wood. Nail a small round cap nail into each corner, to ensure the faceplate is properly affixed and cannot spontaneously fall at a later stage (it would be a guillotine for your toes, if you'd happen to watch it.

Mount a wall mounting kit (either two screws at the edge with a wire/string in between, or a hook/loop in the centre at the top) on the back panel, such that you can hang the clock from a wall. The clock will weigh approximately 5-6 kg.

Code

The code uses a WordClock class object, which does everything. It also uses a few other objects outside the class definition, such as the WiFi client and the NTP client.

The WordClock class has two methods, setup() and loop(). The setup() method, as is customary, deals with initialization, and is run once only. The loop() method does all the timing things.

Every loop, it checks if either the hour or the minute has changed. If the minute has changed, it updates the time (in UTC) and updates the NeoPixel string and the clock face. This includes checking for sunrise and sunset, moonrise and moonset times. It also checks for special dates such as Christmas, Easter, or a birthday. If the hour has changed, it checks for an active WiFi client connection (it shows up as device "wordclock.lan"), and if found, updates the system time with an NTP poll.

Time is kept in UTC, and then corrected using Sydney time zone and DST settings.

There are 4 files for the code. These are:

- utils.h, contains
    - WiFi access SSID and PASS
- WordClock.h, contains
    - List of called libraries
    - List of (macro defined) constants for NTP, TimeZone, NeoPixels, etc.
    - WordClock class declaration
- WordClock.cpp, contains
    - WordClock class definition
    - static constants for minutes/hours “words” (arrays)
- SwissWordClock_v2_20241023_LKG.ino, contains
    - setup();
    - loop();

Some things (like NeoPixel colours) are #defined through preprocessor macros. Most other code is simple straight C++ material. Most of the code is simple, and functional, not necessarily high level or awesome. It’s meant to just do its timing thing. The only exception is perhaps the hardware watchdog – an ESP32 has a hardware watchdog which can be used to reboot the system should it get stuck in some sort of loop (eg when trying to connect to a network or when trying to talk to an NTP server or so). If the watchdog is not reset within 30 seconds, the ESP will reboot. It’s like CTRL-ALT-DEL on steroids.

This code uses the Adafruit_Neopixel library. Instead, you should be able to use the FastLED library as well, if you like. Some minor modifications to the static constants and method calls may be required.

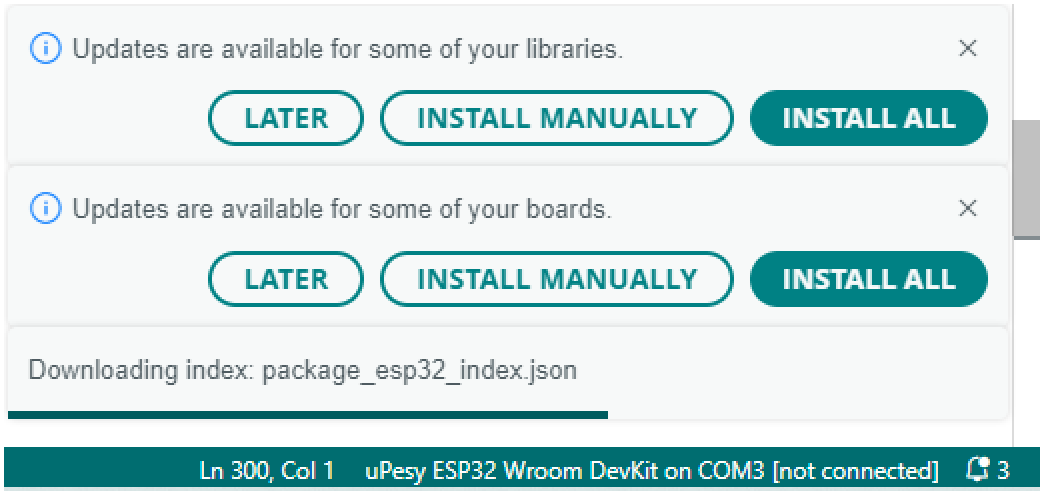
Watch out for updates

At some point during the development, the WordClock bricked itself. ESP32 ended up in an endless boot loop, spitting out errors/stack dumps. Of course, none of those hexadecimal numbers tell you why. Did we blow the esp32? Did we blow the NeoPixel strip? Or both?

Some simple test code (just run the example from the NeoPixel library, ignore all the fancy WiFi, NTP etc stuff) led to the same result – boot loops galore, bit no lights on.

Eventually, after madly reverting code to LKG (last known good) status etc., and finding it still bricked itself, g👀gling around the interwebs led to the magic find on reddit/r/ that this was likely due to an out-of-sequence update issue. Arduino IDE nags you occasionally about available library updates, and apparently the esp32 board library had received an update (to V3.0.0) which was incompatible with some of the ESP32 board driver hooks. Reverting to an older version 2.0.17 unbricked the whole thing, and brought the clock back to life.

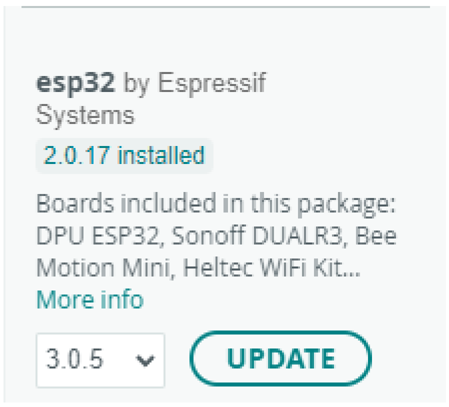
Lesson learnt: be careful when blindly updating libraries.


Updates are available for some of your libraries.

Updates are available for some of your boards.

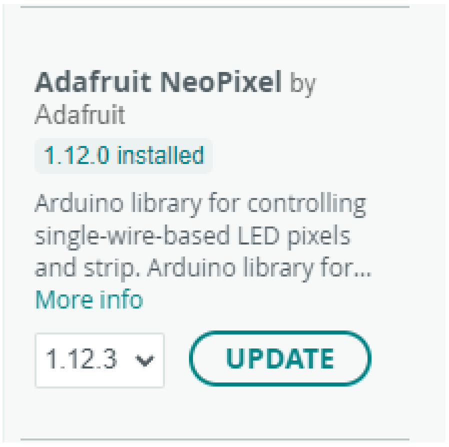
Downloading index: package_esp32_index.json

Ln 300, Col 1 uPesy ESP32 Wroom DevKit on COM3 [not connected] 3


esp32 by Espressif Systems

2.0.17 installed

Boards included in this package: DPU ESP32, Sonoff DUALR3, Bee Motion Mini, Heltec WiFi Kit...
More info


Adafruit NeoPixel by Adafruit

1.12.0 installed

Arduino library for controlling single-wire-based LED pixels and strip. Arduino library for...
More info

Images

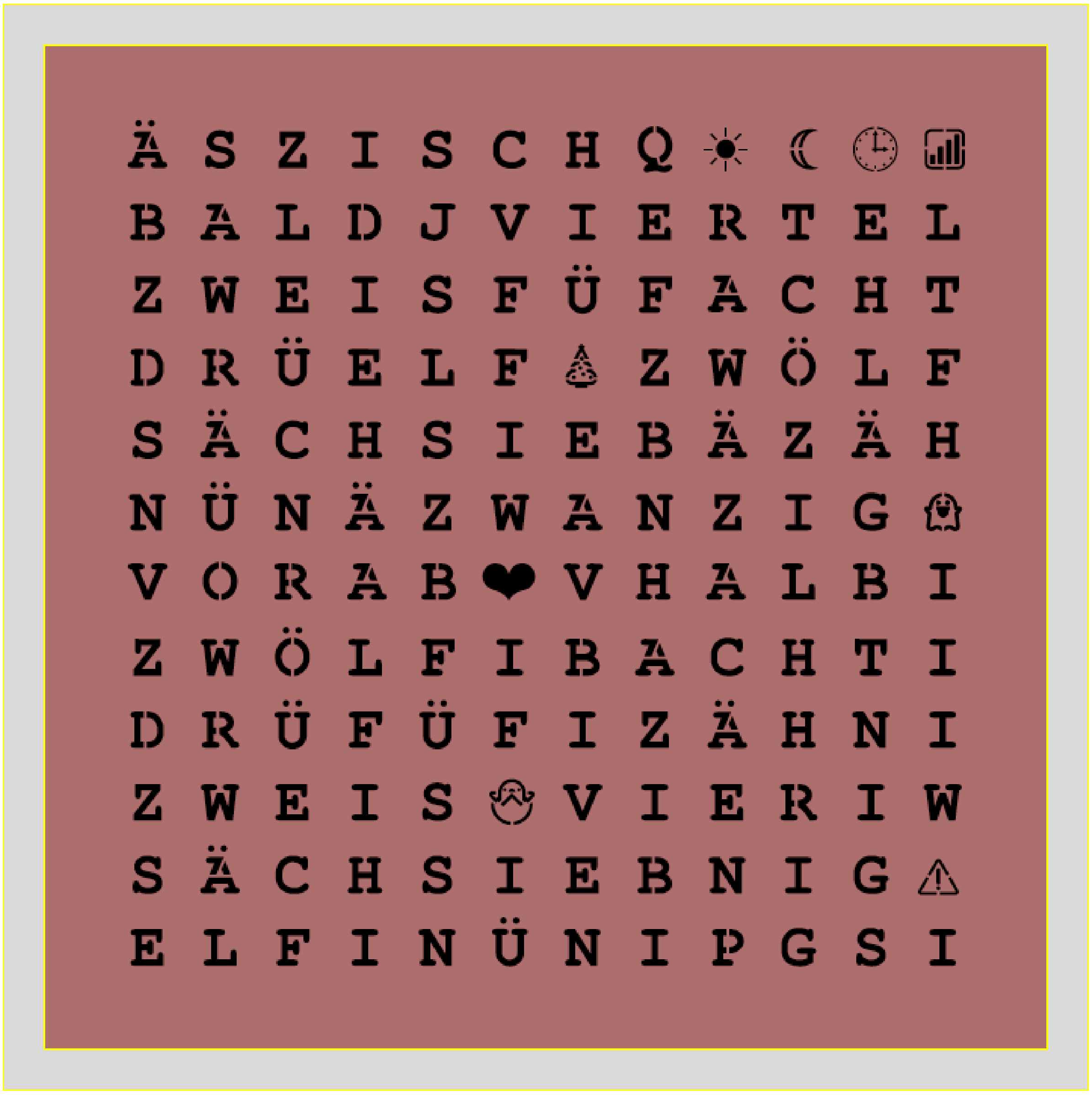
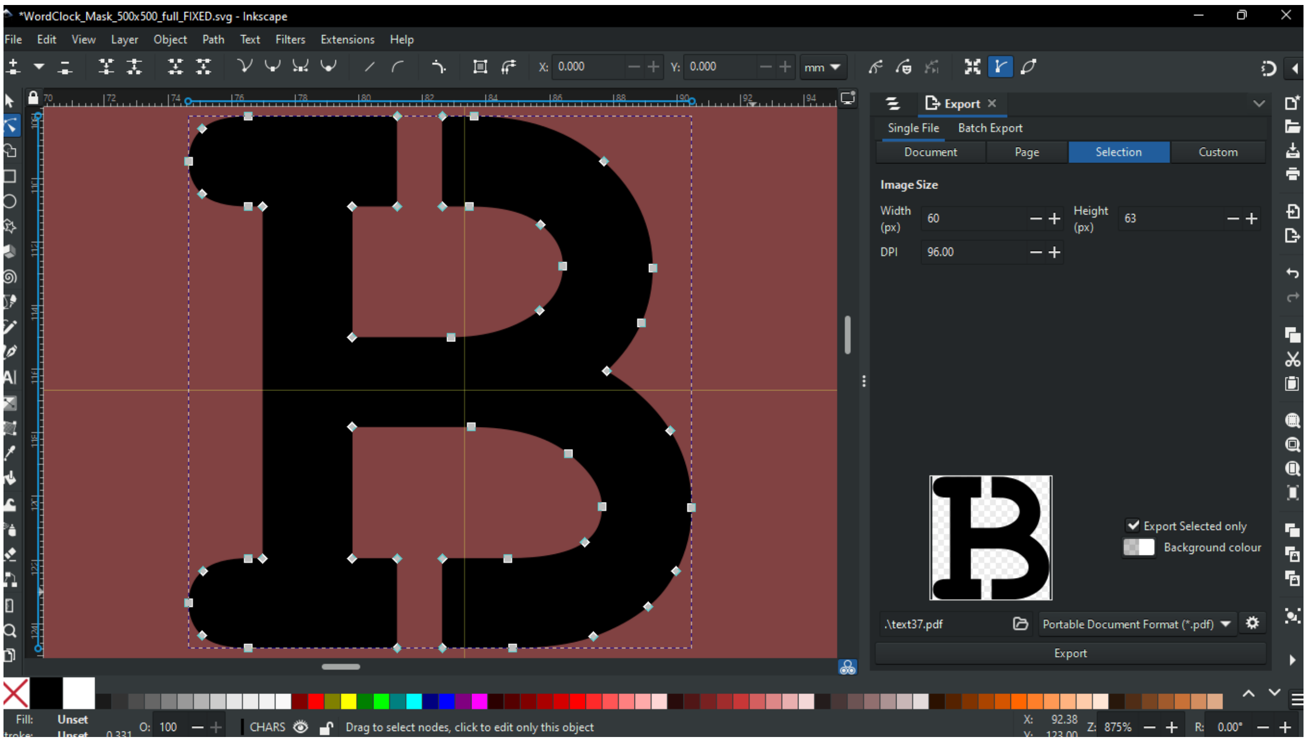


Figure 4 inkscape layout for clock face. There are 12x12=144 characters/icons.


*WordClock_Mask_500x500_full_FIXED.svg - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help
Export
Single File
Batch Export
Document
Page
Selection
Custom
Image Size
Export
Drag to select nodes, click to edit only this object

X: 0.000 Y: 0.000 mm

Width (px) 60 Height (px) 63
DPI 96.00

✓ Export Selected only
Background colour

.\text37.pdf Portable Document Format (*.pdf)

Fill: Unset O: 100 CHARS X: 92.38 Z: 875% R: 0.00°

Figure 5 inkscape layout. Characters are closed path objects. Paths are bezier splines. Typeface is Courier New boldface, and font is 76pt. Small Character line width is approx. 2.83mm. For characters with enclosed islands, 1.4mm wide sections have been removed using a Path Difference Boolean logic step with a sacrificial rectangle. Character grid pitch is 33.3mm in both horizontal and vertical direction.

Links

WordClock

https://meister.io/blog/zyt/

https://github.com/manuelmeister/WordClockBerndeutsch

Libraries

https://github.com/arduino-libraries/WiFi

https://www.arduino.cc/reference/en/libraries/wifi/wifiudp/

https://github.com/JChristensen/Timezone

https://github.com/PaulStoffregen/Time

https://github.com/arduino-libraries/NTPClient

https://github.com/signetica/SunRise

https://github.com/signetica/MoonRise

https://github.com/signetica/MoonPhase

https://github.com/adafruit/Adafruit_NeoPixel

ESP32 WRover DevKit C3

https://www.upesy.com/blogs/tutorials/upesy-esp32-wrover-devkit-board-documentation-version-latest https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate-guide

Hardware watchdog

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/wdts.html

https://iotassistant.io/esp32/enable-hardware-watchdog-timer-esp32-arduino-ide/

ESP Core 3 and Neopixel bug

https://stackoverflow.com/questions/61265671/neopixel-sample-code-crashing-when-using-a-higher-number-of-pixels

https://forum.arduino.cc/t/neopixel-crash-with-75-pixels-using-esp32-core-3-0-x/1273500

https://github.com/adafruit/Adafruit_NeoPixel/issues/402

https://github.com/teknynja/Adafruit_NeoPixel/tree/esp32_rmt_memory_allocation_fix_safe/

Software

https://www.arduino.cc

https://www.espressif.com/en/products/devkits

https://inkscape.org

https://www.gimp.org

https://openscad.org

Parts sourcing

https://LonelyBinary.com/

https://littlebirdelectronics.com.au

https://edconsteel.com.au/

https://ebay.com.au/

https://bunnings.com.au/