Word Clock

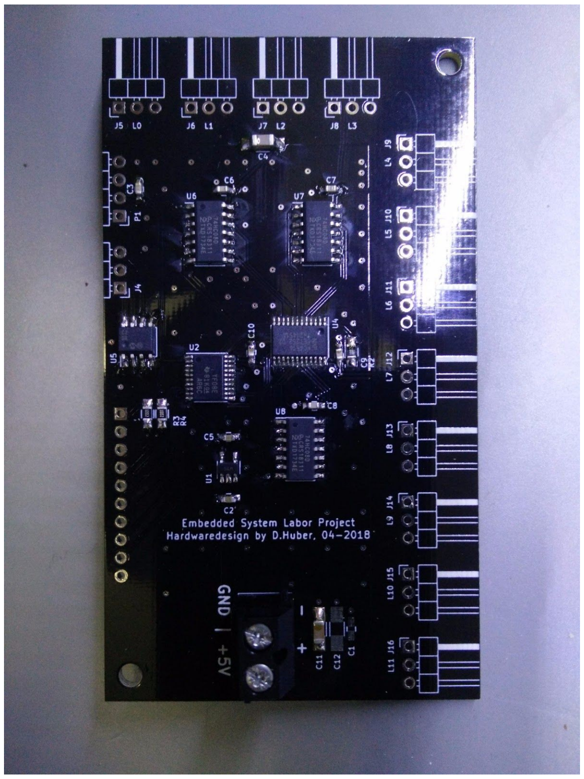
1 Summary of Project‘s Idea

We wanted to take advantage of the “Embedded Systems Laboratory” and build something we have been talking about for a while: a word clock. We have been looking at the “QLOCKTWO CLASSIC” for a while, however it is simply not affordable to us. With our design we wanted to take the clock to the next level and implement weather functionalities, automatic brightness adjustment and touch sensors to switch between modes.

2 Implementation

The realization of the project divides into three main topics, the electronic hardware, the software and the mechanical construction. As the “Embedded Systems Laboratory” is normally limited to hard- and software, we also focused our project report to these points. Some important details on the mechanical construction can be found in the appropriate section.

2.1 Hardware


Embedded System Labor Project
Hardwaredesign by D.Huber, 04-2018
GND
+5V

As hardware platform the Orange Pi Zero was used due to its feature set which perfectly fits to our project. It has WLAN and LAN for networked communication and SPI, I2C for the communication with low level hardware. Combined with Linux and the ability to use highlevel programming languages like Python this gives a outstanding flexibility.
For the characters we use RGB-LED-Stripes based on the known WS2812B controller. For these LEDs a level translator is necessary, which leads to the need of a PCB. This gives the opportunity to create a specialized interface board, containing the power distribution, levelshifter, sensor interfaces, supply capacitors and voltage regulators.Due to the fact, that we emulate the WS2812B protocol with the SPI unit of the Orange Pi, a slight timing error occurs. For a handful of LEDs this is within specification, but a chain of 110 LEDs would be out of specification. Therefore we also added a linewise addressing for our LEDs and an address decoder IC to get rid of this issue. As one of the group members has some experience in PCB design and the manufacturing process,

the PCB was quickly done. There have not been problems during the design process and component selection.
For the user interface we took the Microchip CAP1203 capacitiv proximity sensor which supports three inputs. This IC has no special design requirements and is a fully integrated sensor having only an I2C interface as output and for configuration. For the ambient light measurement we also sticked to a completely integrated sensor IC combining photodiode, amplifier and ADC in one package. As these ICs are mainly used in handheld devices the packages are rather small. We decided to use a bigger one to maintain hand-soldering capability, the Vishay VEML7700. To be able to freely choose the the sensor placement the ambient light sensor was connected with a short cable.
For the linewise addressing standard 74HC logic family ICs have been used. The level translator is a TI TXS0108 featuring a auto direction sensing, which is not used in our design. To provide the 3.3V power rail for the sensors an additional small LDO was integrated to be independent from the Orange Pi power supply. Two mounting holes and a nice black soldermask finish the PCB.

2.2 Software

The chosen software language was Python, because it not only supports low level programming that we need for controlling the LEDs and sensors, but also higher level programming that we need for querying the weather and the time over the internet.
The program itself is constructed like a typical microcontroller program, with an initialization at the beginning and a endless loop to operate in.

2.2.1 Setup of the Orange Pi

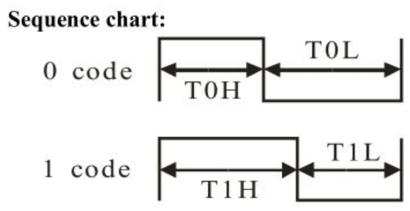
The operating system Armbian was picked to be installed on the Orange Pi. It was primarily chosen because the setup approach of the Ubuntu Core is not viable. After a successful installation the following python 3 libraries were installed: spidev, OPi, smbus, weather (which gets its data from the yahoo API).

2.2.2 Peripheral Communication and Control

The ambient light sensor and our proximity sensor are connected to the Orange Pi over I2C. The ambient light sensor is at address 0x10 and the three proximity sensors at 0x28. From both we can write using the python command `i2c.write_byte_data(address, offset, data)` and read with `return_value = i2c.read_byte_data(address, target_register)`

The LEDs are controlled over the Orange Pi’s SPI interface. The array `characterArray` is then used to build sending blocks which are then sent to the LEDs row by row. Depending on the row number, the corresponding pins on the Orange Pi for the address decoder IC are set using for example `GPIO.output(13, GPIO.HIGH)`.

At sending the LED data to the LEDs we had to manipulate the SPI sending because the LEDs distinguish between a received ‘1’ and ‘0’ not by level but by hightime and lowtime, like in the picture on the right. For that reason we had to encode each bit that should be received from the LEDs with four bits at the sender. Therefore we needed to use a SPI frequency four times higher than the baudrate of the LEDs. A ‘0’-code for the LEDs corresponded to 0b1000 = 0x8 at the sender, and a ‘1’-code to 0b1110 = 0xE.



Because of additional timing issues we had to add some zero bytes before and after the datablock for each row. Subsequently the datablocks are then sent with the command `spi.xfer(rowBuffer.tolist())`.

2.2.3 Configuration File

For the configuration of the clock we decided to use a file. In there the network SSID, password, and LED colours can be entered. Variables which contain the word “optional” do not need to be set. After the config file is set up, it should be put in the home directory of the operating system on the SD card.

2.2.4 Word Clock Software

After startup of the Orange Pi, the config file will be read and the variables set as environment variables. Afterwards the python file is called using the environment variables as transfer arguments. The call will then look like:

```
sudo python3 Wordclock.py <TIME_MODE> <COLOUR>
<DISABLE_AUTO_BRIGHTNESS> <NTP_ADDR>
```

Sudo rights are needed because of the peripheral communication. Time mode describes the possibility to switch between a display mode with dots for minutes or an average mode, where the clock will show “Es ist halb 5” (4:30) starting at 4:27:30 to 4:32:29. The color parameter specifies the color of the LEDs, where “0” is for off and “1” is white and many more options. The next option (`DISABLE_AUTO_BRIGHTNESS`) allows the user to disable the auto dimming of the LEDs. Some of us don’t want to be overruled by a sensor, so we implemented this function. If this parameter is “1” the ambient light sensor will be ignored. With the last variable the user can specify the NTP server which should be used to get an accurate time.
In the SmartWatch.py file this parameters will be handled at the beginning of the main function.

The `main` function continues with creating an array `characterArray` which holds the colour values for each LED of the Wordclock. Afterwards the SPI- and I2C-interfaces for sending data to the LEDs and receiving data from the sensors are initialized. A thread called `getOperatingModeThread` is started, which calls the function `getOperatingMode`. This function contains an infinite loop, in which the three proximity sensors are queried. They detect a switch intention by the user. The proximity sensors are hidden behind the 3D-printed surface with the characters. By holding a hand near the sensor at the bottom right corner of the Wordclock, the `weatherMode` is chosen, which prints the weather. By activating the sensor at the bottom left corner, `timeMode` is chosen to print the time and by activating the sensor at the top right corner, `offMode` is chosen, in which the LEDs are turned off, the Orange Pi however, will continue to run.

After starting the `getOperatingModeThread`, the `main` function proceeds with an endless loop, which operates in the corresponding `operatingMode`. In each mode the required data, weather or time, is queried over the internet and written into the `characterArray`, which is then sent to the LEDs over SPI.

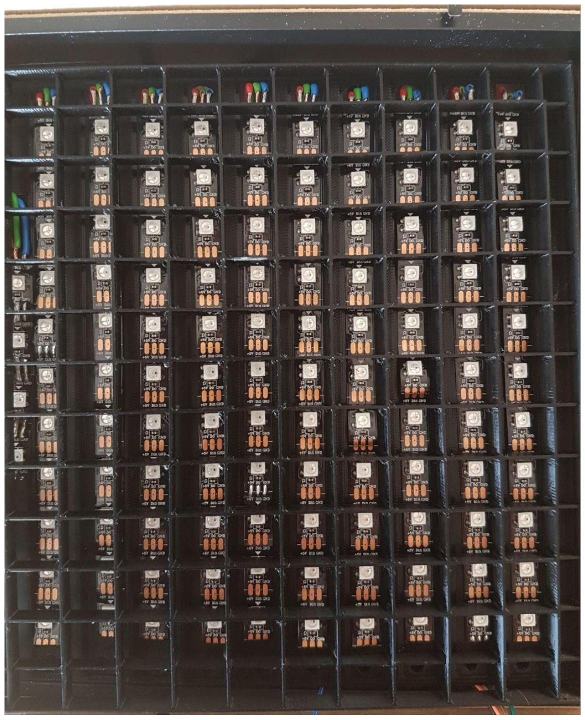
The time is queried from a network time protocol (ntp) server, because it offers one of the lowest latencies at updating the time. If the ntp-server is not accessible, the system time of the Orange Pi is used, which is updated at the startup of the Wordclock.

The weather information is received from the yahoo weather API. Depending on the temperature, the wind speed and the weather code a decision is made what will be printed on the clock. The clock can show 13 different weather messages and, depending on the current forecast and a priority list, one is chosen. The priority list favours important information like rain over not so important ones like “it is cold/warm”. If the current state can not be displayed on our clock (for instance if it is hailing) and it is neither cold nor warm, the default message “Es ist kalt” will be shown.

2.3 Mechanical construction

The front of the clock was 3D printed using a Hilbert Curve, which creates a rustic look. Just behind the letters a very thin white layer was printed, to act as a diffuser for our LEDs.
The three proximity sensors are located at the left and right bottom corner and the top right corner of the backside of the front panel.

For the LEDs a back panel and a grid were printed. The LEDs are glued to the back panel and holes for wiring were made. Afterwards a grid is put on top, so that the LEDs don’t light more than one letter. This construction can be seen on the figure to the right.



3 Changes to Original Proposal

From the functionality point of view there have been changes to the proposal because functions were added. The clock now features also four minute LEDs at the lower end. In addition, the mechanical construction was adopted to the new features and optimized for 3D printing.

Another change we made, was the option to show the current (outdoor) temperature instead of an animation when switching to weather mode. While an animation would be eye-catching and funny, when thinking about usability, we prefer to know how many degree celsius it is.

4 Conclusion

All in all we are pleased to note, that everything worked out as planned. Our hardware works fine, we made the right choices for our sensors and our 3D printed design looks not only appealing but is also functional. We still think, that the decision to work with the Orange Pi Zero was a good call and we learned a lot along the way.
We did meet our goal of spending less than 40€ on this clock: The PCB and the hardware components sum up to roughly 12€, the LEDs were 7€, for the Orange Pi Zero we made a bargain for 6€ and the 3D printing costs 8€. Not considering soldering tin and a few cables (which obviously everyone has laying around at home) the clock was 38€ and many hours designing, soldering, coding, testing, bug searching and going on coffee breaks.

Our video can be found here: https://youtu.be/vUvrZr4AJRo