Nico Ahola
Modified: 07.08.2018

1. Introduction

This document explains the steps done during the making of the weather display. The idea was to create a wireless device that collects weather data from a weather service and displays it on a screen. It was inspired by Mikrobitti (February 2018), which provided the part list and a model code for a bus timetable display. The API and other parts of the code were changed for the weather display.

2. Hardware

The core of the weather display is Wemos D1 mini development board. It includes ESP8266 Wi-Fi module, which contains a useful deep sleep function. During deep sleep, the board consumes almost no power. The board is connected to a Waveshare E-ink display. Because E-ink displays only consume power when they are updated, combined with the deep sleep function of ESP8266, the weather display's power consumption is extremely low. Hence, it only needs a small battery. The battery used is H107-A24 380mAh battery from a Hubsan X4 quadcopter. To make sure the right amount of voltage goes through the battery, TP4056 battery charger module is used. A switch between Wemos and TP4056 is also needed to cut off the power during charging. Wemos and the display are separated from the other components by wire connectors. All components are shown in the figures below.

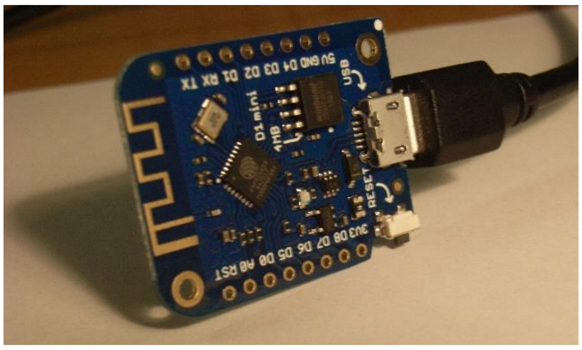
Figure 1: Wemos D1 mini

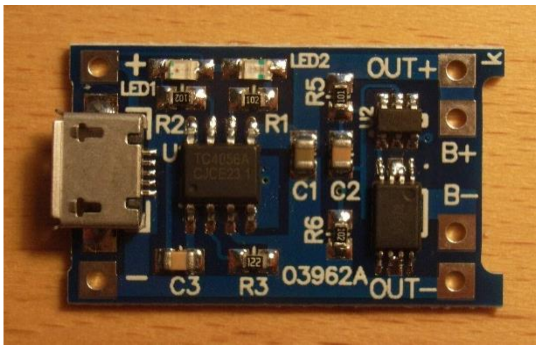
Figure 2: TP4056 module

Figure 3: 380 mAh battery

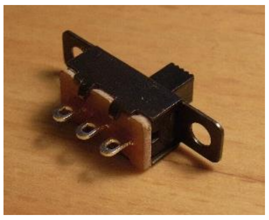
Figure 4: Switch

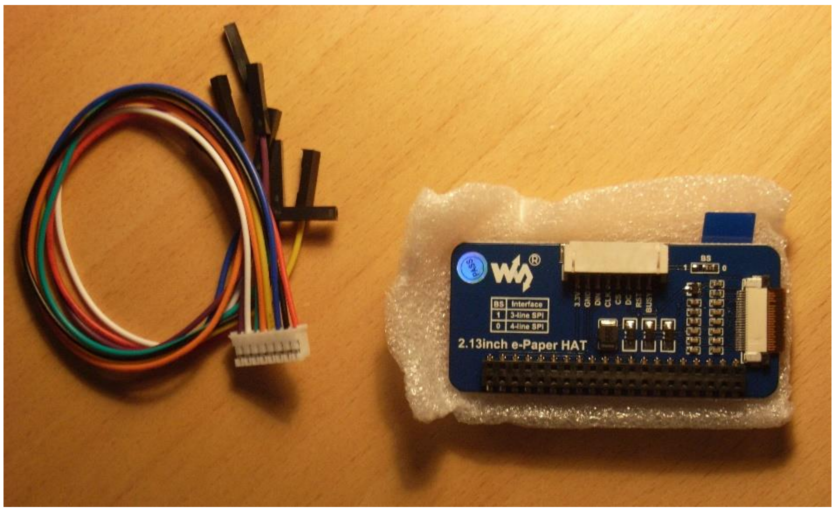
Figure 5: E-Ink Display and its wires

3. Software

The IDE used is Arduino Software, which uses C/C++ as its coding language. Adafruit GFX and GxEPD libraries are used to handle the display, such as writing text on it. The API used is from openweathermap.org. ArduinoJSON library is used to handle the http response, because the weather data is stored in JSON format. The http response string is parsed to JSON format, which can then be easily handled by []-operators to retrieve specific data. Openweathermap provides current weather data and 5 day / 3-hour forecast data for free. Current weather data was decided to be used here. From current weather data the display shows:

- Name of the city
- Current date and time
- Temperature (Tmp)
- Pressure (Prs)
- Humidity (Hum)
- Wind speed and direction (Wnd)
- Cloud coverage (Cld)
- Visibility (Vis)
- Weather condition(s)

Due to the deep sleep functionality of the ESP8266 chip, loop function can be left empty. When Wemos wakes up from the deep sleep, it sends a signal to D0-pin, which is connected to the reset-pin. Therefore, the board always resets itself and starts from the setup function after sleeping.

The code follows four main steps: connecting to Wi-Fi, handling http request and response, collecting all necessary data from the response and finally putting the device to sleep. Length of the deep sleep was decided to be 30 minutes, because Openweathermap updates its current weather data approximately every 30 minutes (xx:50 and xx:20). More detailed explanation of the code and its functions can be found from the comments in the code.

4. Steps

The project started by making sure Wemos D1 mini works. This was done by connecting it to a computer (figure 1) and sending an example blink code from Arduino Software. The next step was to solder all the wires from the display to Wemos (result in figure 6). After that one could test functionality of the display. Mikrobitti provided a test code for this purpose and “Sun shines” was indeed successfully printed on the screen as shown in figure 7.

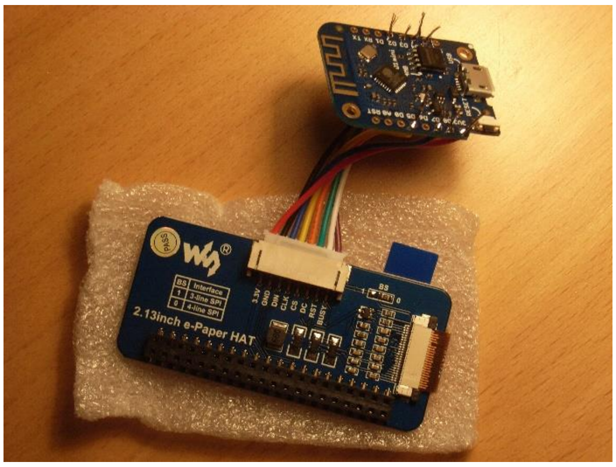
Figure 6: Wemos and display

At this point the purpose was to get the actual code working and print the actual weather data on the screen (figure 8). The code is explained in the previous section. After one was happy with the code, one could move on to the charging unit to make the device wireless.

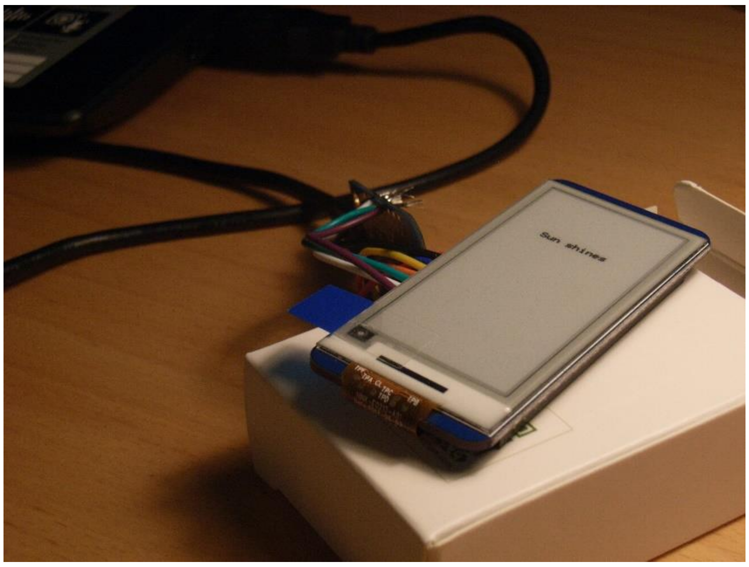

Figure 7: Successful print on the display

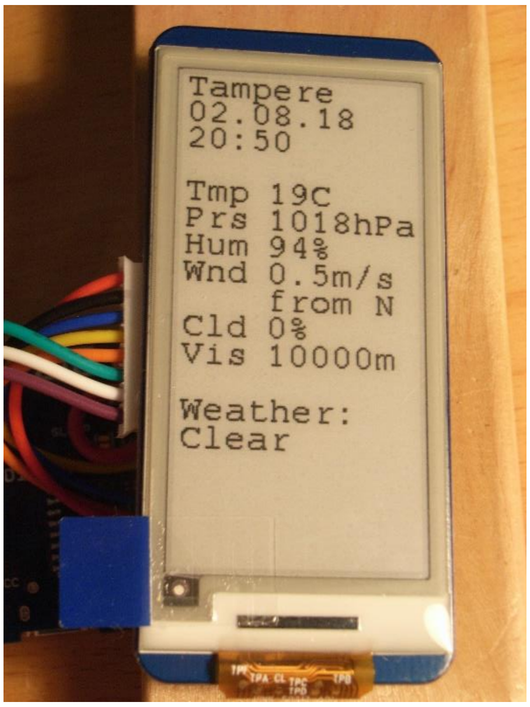

Figure 8: Finished display

Making of the charging unit was simple: wire connector was soldered to OUT+ and OUT- of TP4056 and the battery was soldered to B+ and B- of TP4056. A switch was also soldered between the charging module and the wire connector, as seen in figure 9. The center pin of the switch was connected to TP4056 and one of the other two pins would connect to Wemos's ground-pin when the wire connectors are connected. When the switch is on the same side as the empty pin, there's an open circuit and no current flows. The device connects to Wi-Fi and starts updating itself automatically when the switch closes the circuit. The finished device is shown in figure 10.

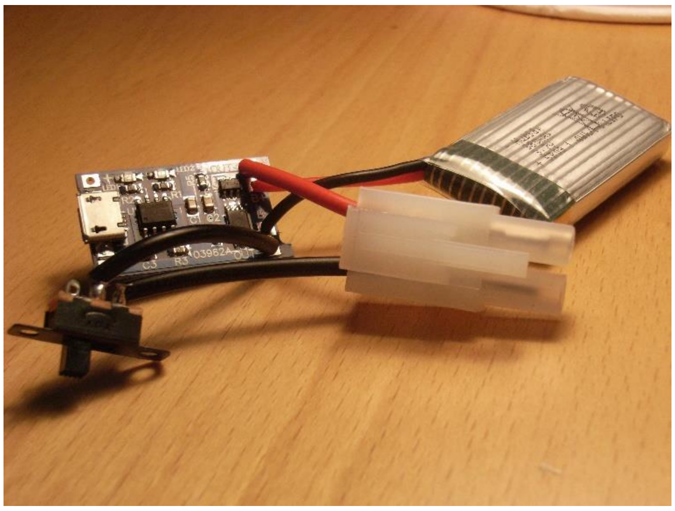
Figure 9: The charging unit

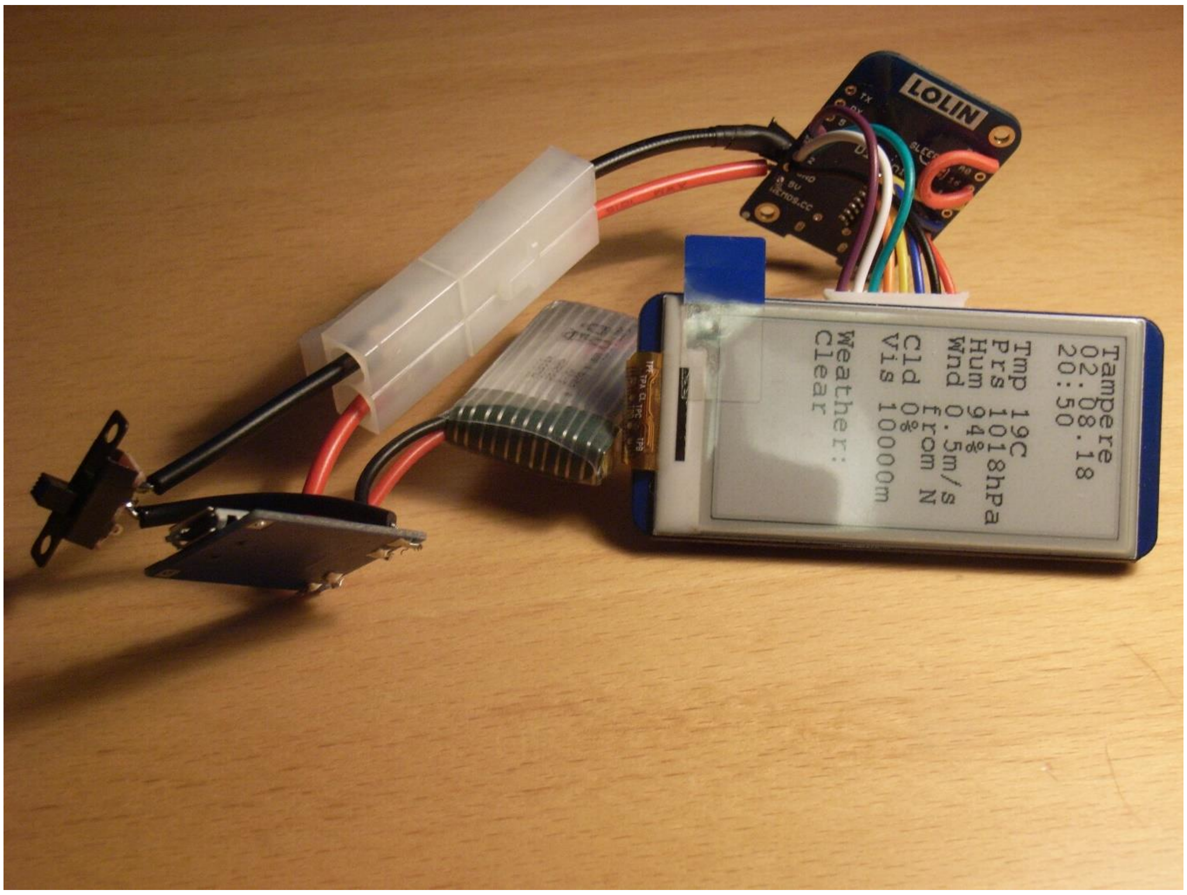

Figure 10: Finished device

5. Further improvements

- A case for the weather display. Either by 3D printing or from cardboard etc.
- Option for 5 day / 3-hour forecast. Either by just commenting in code or a possibility to switch from the hardware.