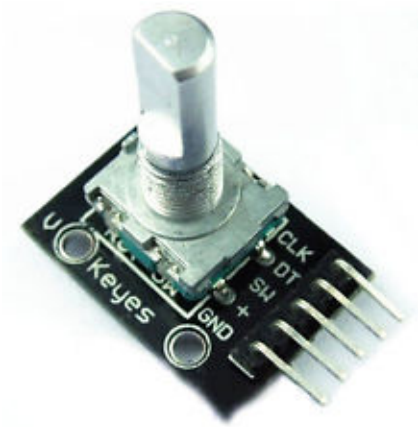
Astro-Clock is a small GPS-controlled alarm clock which offers some additional features (displaying of indoor and outdoor temperatures as well as some astronomical data like position, rise- and set-times of Moon and Sun, …). It reads the datasets broadcast by the GPS-outdoor unit which main purpose is to control a WSPR beacon (for details please see my homepage). Originally planned as a little gadget and birthday present for my spouse, its software-part finally inflated to almost 2000 lines of code, driving the Arduino Nano to its limits. In my opinion the clock has become an attractive and rewarding project worth sharing with others. Great attention was put on keeping the hardware simple and easy to rebuild, using cheap, pre-assembled modules from online sources.

Software:

The main program (sketch) as well as all necessary libraries are available for download on my homepage. Most of the software was written by me, open-source libraries were added wherever it seemed expedient.

Required hardware:

- a rotary switch with integrated push-button (Source: eBay)



- a DS18B20 temperature sensor(Source: eBay)

- a 433MHz ISM receiver (often sold as a package with transmitter-unit, needed for the GPS-beacon / Source: eBay)

- a 1,8“ TFT display, based on ST7735 (128x160 pixel, model KMR-1.8 / Source: eBay)

- an Arduino Nano or one of its clones (Source: eBay)

- a small dynamic speaker (in my case salvaged from an old intercom system)

- some small parts (resistors, capacitors, ...)

- a photoresistor (my source: https://www.ebay.de/itm/20-PCS-GL5528-5528-Fotowiderstand-ldr-Erfassung-Widerstande-Licht-abhangig-CCYE/401176239861?hash=item5d67f79ef5:g:zj0AAOSwHoFXvYl6

GL5528 5528 Photoresistor
Spectral peak: 540nm
Bright resistance: 10-20K
Dark resistance: 1M

- a power supply (in my case an old cellphone-charger / 5V switching type)

Some tips and tricks:

- Rotary switch:

„CLK“ and „DT“ are the two outputs of the encoder, „SW“ is connected to the push button. The two encoder-outputs require 10k pull-up resistors to +5V. In my case they were already on the board.
Values of capacitors & resistors are a bit critical and should not differ more than +/- 20% for the debouncing to work properly.

- DS18B20 temperature sensor:

Here a pull-up resistor is mandatory too and must be added if not already on the board. 4,7 to 10k are OK.

- Brightness sensor / photoresistor:

The value of the series resistor depends on the photoresistor used. The clock provides a function allowing to adjust the day/night threshold value (0...99%). This might be used to choose an appropriate value for the resistor. A sensor reading of 30...70% at switching threshold should be reasonable.
The value of the capacitor isn't critical (10n...100n).

- Speaker:

If a piezo is to be used instead of a dynamic speaker, the common connector circuit is not suitable since it wouldn't provide a path for the piezo to discharge. In this case please consult page 3 of the schematics.
The 1k resistor at the base protects the Arduino in case of a fault (current limitation). Any value from 470 Ohm to 2k is OK. The transistor can be any low power type with sufficient Collector current and B>200. The Arduino can drive currents of up to 40mA, so connecting the speaker directly to the Arduino's output (with a current-limiting resistor) should work too. At least driving a piezo this way is definitely OK.

- 433MHz receiver:

Please provide a 10k pull-up resistor to +5V (there was no internal one at my module)

- TFT-display:

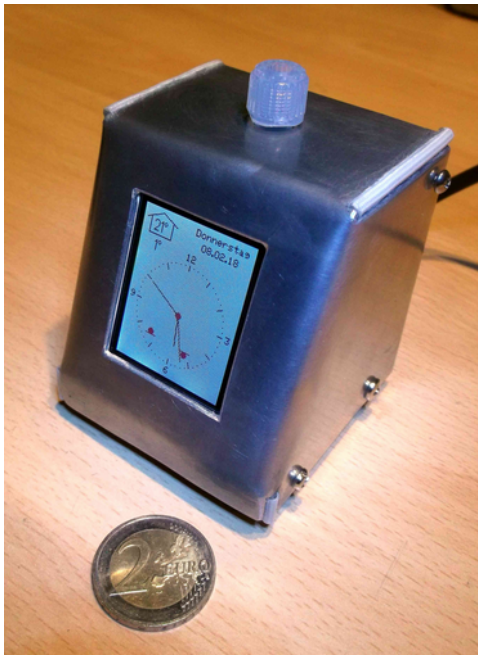
A network of voltage dividers converts the Arduino's 5V to the 3.3V required by the display. Any combination providing the necessary division ratio is OK. As a rule of thumb the resistors connected to ground should be in the range of 1k...10k. There are some designs circulating on the internet that do not convert voltage at all or have only one resistor (in series to the line). I did neither test those designs nor would I recommand such an approach.

The backlight is controlled by a common collector circuit just like the speaker, and the same design rules apply.

- Power supply:

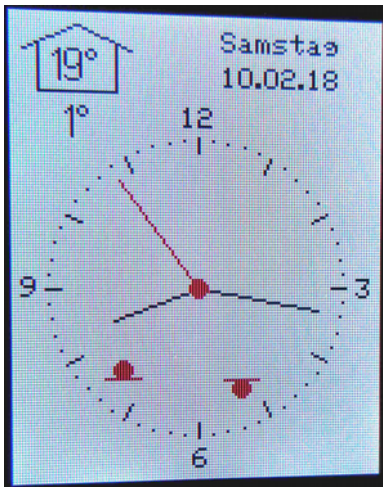
Basically there are two ways of powering the circuit. One can either use Arduino's onboard line regulator, feeding 7...8V into „Vin“. Alternatively you may connect a properly stabilized 5V source to Arduino's 5V pin, bypassing the regulator.
The 220µF buffer capacitor is definitely recommendable but the 2x100µH line filter is not a must (I’m a ham radio enthusiast and hence quite sensitive when it comes to electromagnetic pollution :-). Speaking of filtering and noise, despite its CMOS technology the Arduino is quite noisy and installing the ISM receiver in its proximity will result in a reduced sensitivity. Therefore I decided to put both, the receiver and the temperature sensor into a piece of shrink tube and attach them to the power cable about 20cm (8 inch) away from the clock. This way you can also avoid false temperature readings. It takes only two extra lines to transfer the data (I used an old 4-wire USB-cable for this).

Operation:

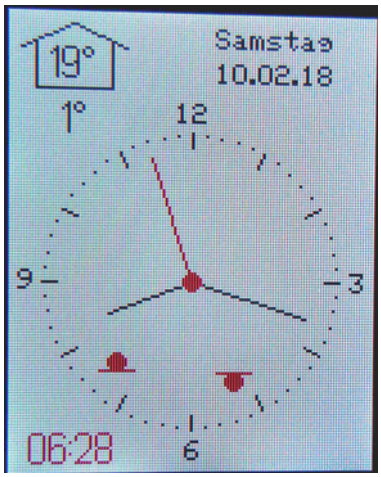


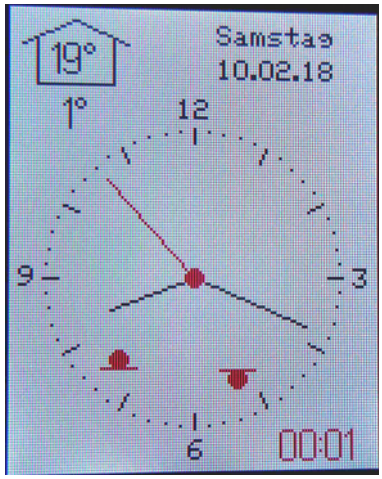
This is the finalized clock. The knob on the top is connected to the rotary switch and push button. Keep it pressed during start-up to enter the day/night threshold setup menu. In this menu the reading of the brightness sensor as well as the currently set threshold are being displayed. You can adjust the threshold by turning and acknowledge/save the selected value by pressing the knob. After that the desired display brightness for day- and night-mode, the timezone and observing of daylight saving time (DST) can be set in the same manner.

Once the clock has started you can change between various menus by turning the knob.

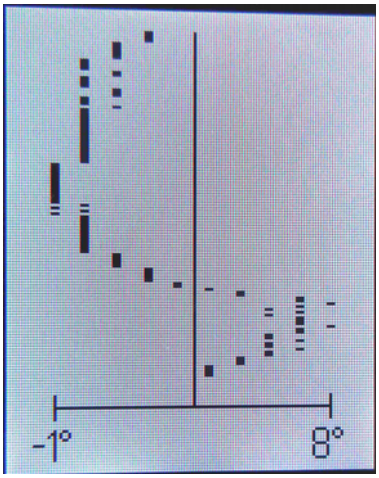


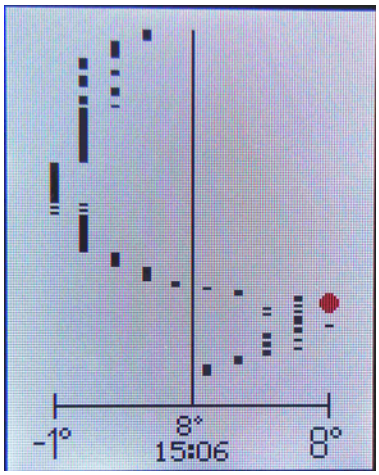
The main screen in day- and night-mode. In the upper area date as well as indoor and outdoor temperatures are being displayed. The language can be set to either German or English by changing a parameter at the source code. Temperatures are in °C. The two icons at the clockface mark the times of sunrise and sunset.





The main screen with active alarm clock (left) and timer (right). Shortly press the knob to select one of the two. In order to adjust the setting, keep the knob pressed after selecting the respective item until the numbers start flashing. Turn the knob to adjust the value. Pressing the knob once again will store the value and activate the alarm (numbers turn Red). In case of the alarm clock values will be stored non-volatile and hence not being erased during power-off. To turn off the alarm first select the respective item by pressing the knob and keep it pressed until the display is cleared. Once an alarm has fired it can be muted for 5 minutes by turning the knob. Pressing the knob will turn the alarm off. In case of the alarm clock (left picture) it would be triggered again after 24h.





In this screen you can see the trend of the outside temperature over the past 24h (most recent data at the top). Pressing the knob will toggle a cursor on/off allowing to select a certain datapoint (values are shown at the bottom).

Here you can see the current position of Sun and Moon as well as their rise- and set-times, calculated for the observers current coordinates. The Moons phase and tilt are also calculated and depicted based on the current time and coordinates (with some limitations due to the resolution of the display). With regard to the elevation, the typical approach of projecting the symbols to a virtual plain was slightly modified. At this orrery the center stands for 90°, the horizon-circle equals 0° and 45° is at half distance between the two.

In case an objet is set (below horizon) its symbol will be replaced by either “M” or “S”.