ESP32S2 GPS clock with OLED display

Revision 0

Date: 2025 May 31

Table of Contents

1 Introduction

A 24 hour digital clock with timing derived from satellite based GPS atomic clock can be assembled using three readily available hardware components:

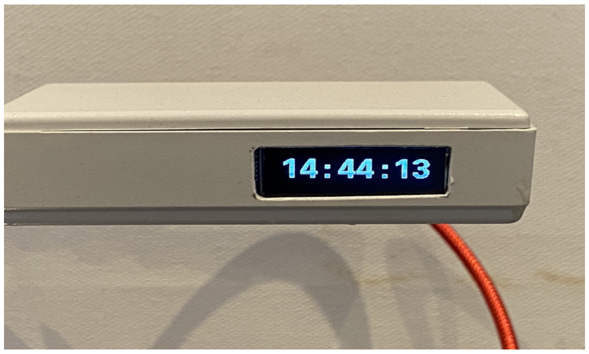
- ESP32 S2 Mini controller board
- NEO-6M GPS module and ceramic antenna
- 0.91” OLED display, 128x32 pixel, I2C interface

The firmware for the Espressif ESP32 S2 relies on software libraries for a lot of the processing tasks, such as the parsing of the NMEA messages received from the GPS module, for driving the OLED display and for handling the serial communications interfaces for peripheral functions.

The clock is housed in a small 8 cm L x 5 cm W x 2 cm H plastic enclosure. Power is provided via a USB cable that plugs into a 5 V cell-phone charger. When first powered-up, the GPS module may take a few minutes to acquire the GPS satellites signals. The clock will display something like 19:35:35 until it locks onto the satellite signals, then the current valid clock time is displayed.

The clock is programmed to display UTC – 5 hours time zone, which is New York time (Eastern Standard Time). If you operate the clock in another time zone you need to change the TimeZone constant in main.cpp to the correct offset value corresponding to the new time zone to show the correct local time.

Daylight time correction is enabled by installing a shorting jumper block across a 2 pin header on the ESP32S2 board.



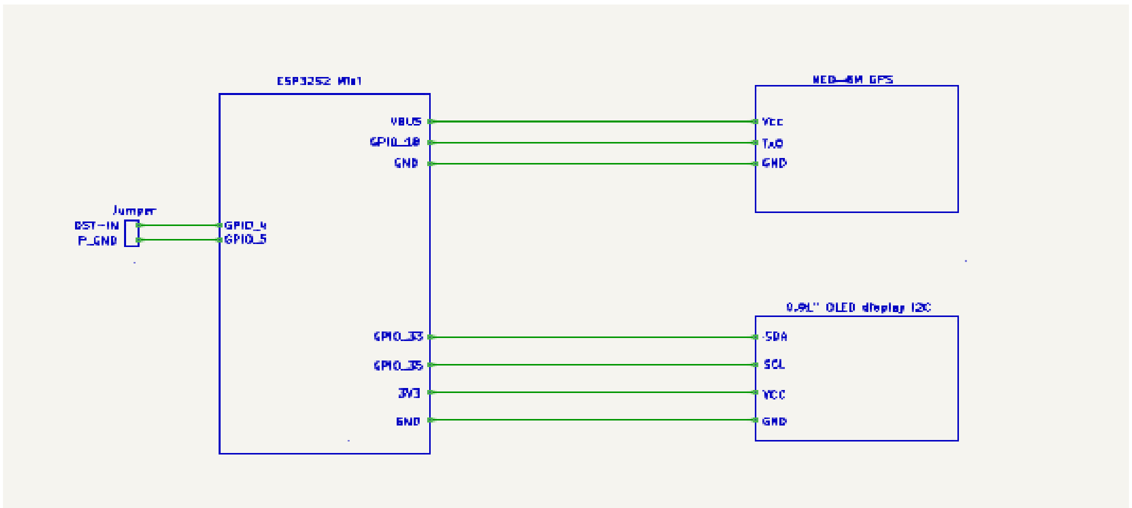
2 How the clock works

The NEO-6M module receives GPS satellite signals that enables it to determine the exact GPS time. The module, once synchronized, transmits the time signal information in the NMEA message format to the ESP32S2 board where the UTC time is parsed from the GSS sentence using the 107-ArduinoNmeaParser library.

The parser is fed the incoming NMEA messages on the ESP32S2 Serial1 port using the parser.encode((char)Serial1.read()) function. When an updated message is received and parsed, the onRmsUpdate(nmea::RmcData const rmc) function processes the new UTC data to convert it to local time by adding the TimeZone offset for the clock's current time zone location. Then a daylight savings time correction is added if the DST_IN pin on the ESP32S2 is jumpered to ground.

The local clock time is displayed on a terminal screen, if one is open, in hh:mm:ss format. The same local time data time is then displayed on the OLED display using the U8g2 graphics library functions. Please see the main.cpp source code for details.

The block diagram for the clock is shown below and can be used for the interconnection wiring between the module components.



ESP32S2 GPS clock block diagram

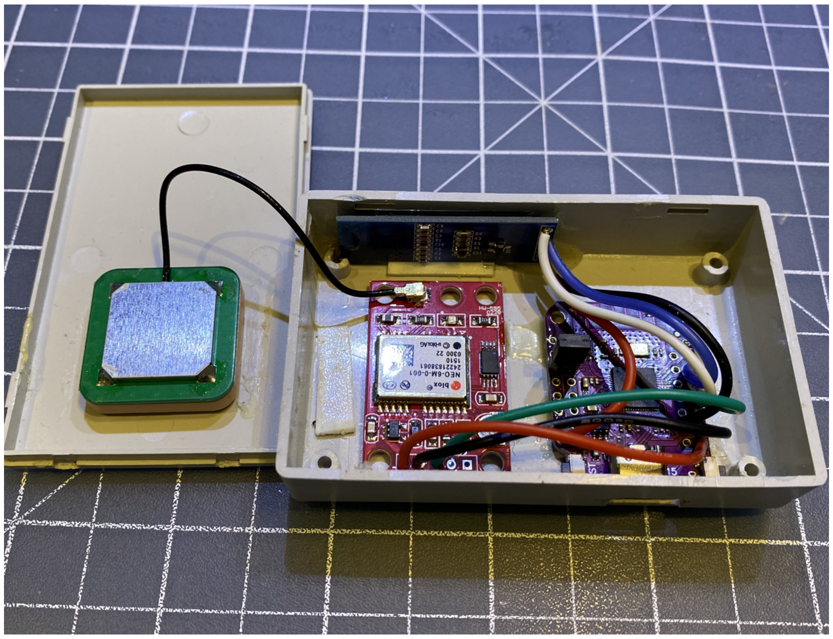
3 Assembly

Assembling the clock components is straightforward and simple as only 7 interconnecting wires are needed. The plastic enclosure case is prepared by cutting out the openings for the OLED display and for the USB plug to reach the board's type C connector.

The display module should be carefully positioned so that the front of the display can be clearly seen and that the case cover can be replaced without interference. The USB connector opening should have enough clearance so the plug pass through.

Next, the modules are wired together as per the block diagram. A 2 pin header is soldered to the pads for GPIO_4 and GPIO_5 on the board. A standard plastic strip header shorting jumper plugs onto these pins to enable the daylight savings time correction.

After wiring the modules is completed, test the clock before the modules are fixed into the case by connecting the ESP32S2 board to a 5 V supply with a USB cable. The clock should initialize and display a 'GPS Clock' sign-on message, then it will try to receive signals from several GPS satellites to synchronize itself. During the time when it is connecting to the satellites, you may see a static time display such as 19:35:35. When the module is synchronized, the correct local time is shown on the OLED display.



The clock has to be located where it has a clear reception path to the satellites for proper operation. If the satellite signals are blocked by structures like metal clad rooms or thick concrete walls, the clock will not work.

Once the clock is operating properly, the component modules can be glued inside the case with a bit of epoxy resin or other suitable adhesive and the case cover closed.

4 Software tools

The software code for the clock was developed using the Visual Studio Code editor with PlatformIO extension using the Arduino framework and several software libraries. Please see the platformio.ini files for the project configuration files.

PlatformIO is recommended if you plan to modify, build and upload the code to the ESP32S2 board.

5 License