Special thanks to:

Zoltán Unferdorben, who keeps coming up with new ideas, great PCB designs, and meticulously tests the latest program versions ☺.

Sándor Pintér who recently joined the project. His contribution includes improving the temperature/humidity measurement and Neopixel LED lighting. He is also a sharp-eyed and tireless tester. ☺

Balint Gautier for rebuilding the project website.

DISCLAIMER: There is another modular Nixie clock panel project on the Internet, also called UniClock. The two projects are not related – the fact that they bear the same name is mere coincidence.

About the software

UniClock is a universal timekeeper program for the Arduino Development Environment. It is based on the ESP8266 and ESP32 microcontrollers, and works with Nixie, VFD or Numitron tubes and LED displays.

Features

- Time set via WiFi from an Internet timeserver, from the GPS module or through synchronizing with its own DS3231-based RTC module
- Can have 4, 6 or 8 digits
- Daytime and nighttime brightness can be adjusted in 10 steps
- Animated digits
- Works offline (i.e. no Internet connection required
- Parameters can be set using a web browser. If there is no infrastructural Wi-Fi the clock works as an AP, providing access to its own website.
- Per-tube adjustable back- and underlighting (Neopixel WS2812 LEDs) with selectable animation, brightness etc.
- Alarm with piezo speaker, optional snooze, and mute button.
- Optional temperate and humidity sensors.
- Optional light sensor for automatic brightness control, for both tubes and LEDs.
- Optional radar / infra sensor usable to switch off the tubes display.
- Optional filament / anode ON/OFF switching to increase the tubes life.

GitHub project page:

https://github.com/gpeter62/ESP_UniClock contains verified wiring diagrams and PCB designs in the „/schematics” subdirectory, courtesy of Zoltán Unferdorben. PCBs, kits and ready-made clocks are also available – please contact me privately.

- There are publicly available clock projects that can also be used with this program. Two of these are included in the wiring diagrams.
- Datasheets can be downloaded from the "/Datasheets" subdirectory for parts we use in our designs.
- The "/old_versions" subdirectory contains earlier, stable program versions.

User's Manual

After being switched on, the clock counts from 0 to 9 on all tubes.

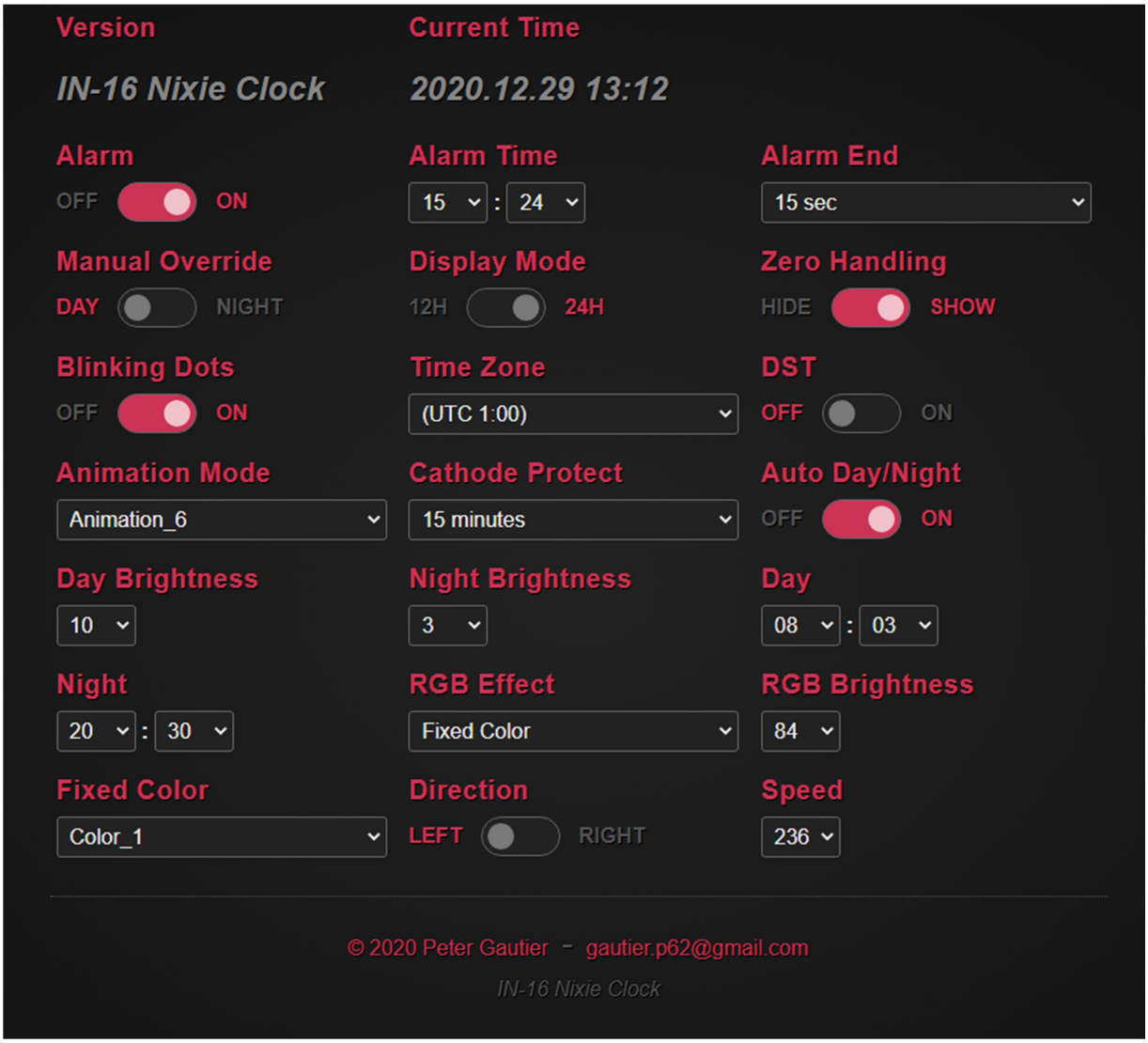
No-network operation (using GPS or RTC synchronization)

The clock acts as an AP, creating a Wi-Fi network "***UNICLOCK***". (Or whatever name was already set for the AP_NAME variable)
Connect to this network to see the clock's website at 192.168.4.1. The IP address is also shown on the clock during initialization.

Internet operation

The clock will attempt to connect to the last known Wi-Fi network. If it fails, it will default back to the "No-network operation" mode, see description above. Connecting to the clock's AP will then show available Wi-Fi networks. Pick the one you prefer to use, and enter the password required for connection. Network name and password will be stored and used afterwards (also see "Wi-Fi manager" function). After a successful connection, the clock's IP address will be shown on the display (for example, 192.168.1.106). The clock's website can be accessed using this address from within the same network.

The clock's website



In daytime mode, all functions work as normal. In nighttime mode, the led lighting, and the separator blink both turn off, and the clock switches to "night brightness".

Current time: local time.
Temperature: If there is a thermometer attached, the temperature is displayed. Otherwise, it is empty.
Humidity: If there is humidity sensor attached, the humidity is displayed. Otherwise, it is empty.

Alarm: If "on", an alarm will sound at the set time of the day, for the duration set in "**alarm end**". If there is a snooze/off button connected, the alarm can be turned off using that. If there are Neopixel LEDs connected, those will flash white at maximum brightness. If there is a beeper connected, it will also start beeping with increasing frequency.

- ***Manual Override***: Manually switch between day- and nighttime mode.
- ***Display Mode: 12/24*****:** 12 or 24 hour display mode.
- **Zero Handling:** Show leading zero in in 12 hour mode.
- ***Blinking Dots***: Enables flashing of the date/time separator. When set to "off", separator is turned always on.
  Time Zone: Local time zone.
- ***DST***: Turns on DST, or Daylight Saving Time.

- ***Animation mode***: Digits may be animated, rolling over in 5 different ways:
  - 0 = turned off
  - 1-5 = animation modes
  - 6 = always picks animation randomly from 1-5
- ***Cathode Protect***: After the defined period, all numbers run through the display. For Nixie-tube clocks this is important to prevent cathode poisoning. For other displays it is just a nice effect to see.
- **Auto Day/Night**: Enables automatic switching between day and nighttime modes.
- ***Day Brightness:*** Daytime brightness (1-10).
- ***Night Brightness***: Nighttime brightness (0-10). If it is set to zero, the display turns off completely.
- ***Day*** and ***Night***: sets the time when "daytime" and "nighttime" brightness mode begins.

If there are Neopixel LEDs connected, the website also displays the following options:

- ***RGB Effect***: Color animation mode, adjustable between "OFF" and 10 different animations.
- ***RGB Brightness***: Total LED brightness.
- ***Fixed Color***: (0-256) Use only the selected color. Colors between 0-255 are based on the standard color wheel. Extra color: 192 = white.
  (example: 1=red, 20=yellow,60=green,160=blue, 192=WHITE, 244=magenta)
- ***Direction Left/Right***: Direction of animation.
- ***Speed***: Speed of animation.

Setting the parameters and uploading to the microcontroller:

Install the Arduino IDE as per instructions on the website (www.arduino.com). Then - from within the IDE - add the ESP8266 or ESP32 boards (check the "Tools/Boards" menu and add whatever board you use). Set the CPU frequency to 160 MHz adjust Flash Size to at least 160 Kbyte (the minimum for SPIFFS partitions).

Also install the "***ESPxx Sketch DataUpload***" program. This is used to upload the files for the clock's website from the /dat library.

Details for installing the above program:

- for 8266: https://randomnerdtutorials.com/install-esp8266-filesystem-uploader-arduino-ide/
- for ESP32: https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/

The UniClock program must be added to the Arduino library (usually at **This PC/Documents/Arduino**). Libraries used by UniClock can also be found in the ZIP file. These must be installed too (usually into **This PC/Documents/Arduino/libraries**)

Setting the program for the clock hardware:

Before compiling adjust the parameters In the **clocks.h** file (as required by the hardware you use). If a parameter is not set, it will be either turned off set to default. In other words, you only need to set parameters related to your specific hardware – the rest can be left unchanged.

```
/*_____________________________ USABLE PARAMETERS ____________________________________________
  //#define DEBUG  //Enable Serial Monitor, 115200baud (only, if TX pin is not used anywhere!!!)
  //---------------------------- CLOCK EXTRA OPTION PARAMETERS --------------------------
  //#define USE_DALLAS_TEMP //TEMP_DALLAS_PIN is used to connect DS18B20 temperature sensors
  //#define USE_DHT_TEMP    //TEMP_DHT_PIN is sensor pin
  //#define USE_BME280      //I2C Temperature + humidity + pressure, SDA+SCL I2C pins are used!
  //#define USE_BMP280      //I2C Temperature + barometric pressure, SDA+SCL I2C pins are used!
  //#define USE_AHTX0       //I2C Temperature + humidity, SDA+SCL I2C pins are used!
  //#define USE_SHT21       //I2C Temperature + humidity, SDA+SCL I2C pins are used!
  //#define USE_RTC         //DS3231 realtime clock, SDA+SCL I2C pins are used!
  //#define USE_GPS         //use for standalone clock, without wifi internet access
  //#define USE_NEOPIXEL //WS2812B led stripe,tubes backlight.
    Don't forget to define tubePixels[] !

  //----- DRIVER SELECTION -- Use only 1 driver from the following options in the clocks.h file!
  //#define MULTIPLEX74141_ESP32  //4..8 Nixie tubes generic driver for ESP32
  //#define MAX6921_ESP32         //4..8 VFD tubes (IV18) driver for ESP8232
  //#define HV5122                //4..8 Nixie driver - development in progress
  //-------------- 8266 clock drivers ------------------------------------------------
  //#define MULTIPLEX74141        //4..8 Nixie tubes generic driver for ESP8266
  //#define MAX6921               //4..8 VFD tubes (IV18) driver for ESP8266
  //#define NO_MULTIPLEX74141     //4..6 Nixie tubes, serial latch driver, 74141 for each tube
  //#define MM5450                //6..8 LEDS
  //#define MAX7219CNG            //4..8 LED
  //#define Numitron_4511N        //Numitron 4x tube clock
  //#define SN75512               //4..8 VFD tubes
  //#define samsung               //samsung serial display
  //#define PCF_74141             //PCF pin expander for tube selection
  //#define PT6355                //VFD clock - development in progress

  //--------------------- PINOUT & PIN PARAMETERS ---------------------------------------
  //#define PIN_SDA xx             // you can set the used SDA and SCL pins
  //#define PIN_SCL xx             // if it is not default value
  //#define COLON_PIN   -1        //Blinking Colon pin.  If not used, SET TO -1
  //#define TEMP_DALLAS_PIN -1    //Dallas DS18B20 temp sensor pin.  If not used, SET TO -1
```

```
//#define TEMP_DHT_PIN -1       //DHT temp sensor pin.  If not used, SET TO -1
//#define DHTTYPE DHT22         //DHT sensor type, if used...
//#define LED_SWITCH_PIN -1     //external led backlight ON/OFF.  If not used, SET TO -1
//#define DECIMALPOINT_PIN -1   //Nixie decimal point between digits. If not used, SET TO -1
//#define ALARMSPEAKER_PIN -1   //Alarm buzzer pin
//#define ALARMBUTTON_PIN -1    //Alarm switch off button pin
//#define ALARM_ON HIGH         //HIGH or LOW level is needed to switch ON the buzzer?
//#define NEOPIXEL_PIN 3        //8266 Neopixel LEDstripe pin is always the RX pin!!!
//#define RGB_MIN_BRIGHTNESS 8   //Neopixel leds minimum brightness
//#define RGB_MAX_BRIGHTNESS 255 //Neopixel leds maximum brightness
//#define RADAR_PIN 34          //Radar sensor pin
//#define RADAR_TIMEOUT 300     //Automatic switch off tubes after xxx sec
//#define TUBE_POWER_PIN 23     //Filament or HV switch ON/OFF pin
//#define TUBE_POWER_ON HIGH    //HIGH or LOW level is needed to switch ON the TUBE POWER?
//#define LIGHT_SENSOR_PIN -1   //Environment light sensor, only ADC pins are usable!
//                                ESP32 for example: 34,35,36,39... 8266: only A0

//#define REF_RESISTANCE    10000.0  // Resistor value is 10k, between LDR sensor and GND
//#define MAXIMUM_LUX 150            //Lux level for maximum tube brightness
//#define LUX_CALC_SCALAR   12518931 * 1.2  //Calibrate here the LDR sensor
//#define LUX_CALC_EXPONENT -1.405          //LDR sensor characteristic

//#define MAXBRIGHTNESS 10      //Do not change this value!

//Display temperature and date in every minute between START..END seconds
//#define ENABLE_CLOCK_DISPLAY true  //false, if no clock display (thermometer+humidity only)
//#define SHIFT_TUBES_LEFT_BY_1   //shift leftIP address by 1 tube the display,
                                  //if a thermometer is used with spec tube
//#define LEFTDECIMAL false       //set true (Z574M), if decimal point is on the left
                                  //side on the tube.  Else set false (Z573M)!
//#define TEMP_START  35          //Temperature display start..end
//#define TEMP_END    40
//#define HUMID_START 40        //Humidity% display start..end
//#define HUMID_END   45
//#define DATE_START  45        //Date is displayed start..end
//#define DATE_END    50
//#define ANIMSPEED   50        //Animation speed in millisec
//#define TEMP_CHARCODE 15      //Thermometer "C", set to -1 to disable
//#define GRAD_CHARCODE 16      //Thermometer grad, set to -1 to disable
//#define PERCENT_CHARCODE 17   //Humidity %
//#define DOUBLE_BLINK          //both separator points are blinking (6 or 8 tubes VFD clock)


//#define AP_NAME "UNICLOCK"   //Access Point name
//#define AP_PASSWORD ""             //AP password

//#define WEBNAME "LED UniClock"  //Clock's name on the web page
```

*/If ***DEBUG*** is enabled, the serial monitor will display every step during initialization. Set serial speed to 115200 baud.

For other **CLOCK EXTRA OPTIONS**, uncomment the line (remove "//") to enable the given module.

Temperature and humidity sensors

The clock can have many sensors connected, but now maximum two temperature and two humidity sensor data are displayed on the tubes. (The Serial Monitor will show all of them.)

USE_DALLAS_TEMP enables Dallas 18b20 temperature sensor. The value of TEMP_SENSOR_PIN must be set. A maximum of 2 sensors may be used on one shared bus. (If there are 2 sensors, the order of the sensors can be set at the top of the **dallas_temp.ino** module.)

USE_DHT_TEMP: DHT11, DHT21, or DHT22 thermometer and humidity sensors can be used. The exact type of the sensor can be set in the **#define DHTTYPE** line (for example, #define DHTTYPEDHT22). The value of TEMP_SENSOR_PIN must be set. Only one DHT sensor is allowed. **DHT and Dallas sensors may not be used at the same time!**

USE_BME280, USE_BMP280 , USE_AHTX0, USE_SHT21: Different I2C environment temperature and humidity sensors. The pins ***PIN_SDA*** and ***PIN_SCL*** must be defined. Only one sensor can be used from the same type.

USE_GPS: If the GPS module is allowed, the clock will not look for a Wi-Fi network. Instead, it will synchronize the clock from the GPS module. The ***gps.ino*** program will be turned on. As of now, the GPS parameters must be adjusted in this file if necessary.

USE_RTC: If the RTC is enabled, it will use the DS3231 RTC module on I2C. (The I2C PIN_SDA and PIN_SCL pins must be defined.) GPIO ports for the push buttons and the mode switch can be set here too. If the RTC module is used, GPS may not be used. RTC mode (selectable via the mode switch) may work as a WiFi clock as well, refreshing the time from the Internet. If there is no internet connection, the RTC can be adjusted using the push buttons.

USE_NEOPIXEL: WS2812 Neopixel LEDs used via the Makuna library: *z_neopixel.ino*. Within this file, the **tubePixels[]** array must be set to represent what tube the LED row individual pixels belong to. Each tube may use either one LED or multiple LEDs. With the 8266, only the RX pin can be used for control. With the ESP32 any GPIO pin lower than 32 can be used.

Selecting display driver module:

If you use the ESP32 board these drivers can be selected for now:

- #define MULTIPLEX74141_ESP32 // *ESP32 Nixie driver*
- #define MAX6921_ESP32 // ESP32 7-segment VFD driver
- #define HV5122 // 4..8 tubes Nixie driver – development in progress
- #define PT6355 //VFD clock, multi segment tubes - development in progress

For the 8266 these drivers are available:

- #define NO_MULTIPLEX74141 //4..6 Nixie tubes, serial latch driver, 74141 for each tube
- #define MM5450 // *6..8 LEDs*
- #define MAX7219CNG // *4..8 LEDs*
- #define Numitron_4511N // 4x *Numitron tube clock*
- #define SN75512 // *4..8 VFD tubes*
- #define Samsung // Samsung serial display
- #define PCF_MULTIPLEX74141 PCF // pin expander tube selection

Defining other control pins:

Pin definitions are the same for both the ESP32 and the 8266.If a function is not used, set that pin to -1. **Always use GPIO number, since it is independent from the board type.**

- #define PIN_SDA xx // you can set the used SDA and SCL pins
- #define PIN_SCL xx // if it is not default value
- #define COLON_PIN-1 // Blinking colon. If not used, SET TO -1
- #define TEMP_DALLAS_PIN -1 // Dallas temp sensor. If not used, SET TO -1
- #define TEMP_DHT_PIN -1 // DHTxx temp sensor. If not used, SET TO -1
- #define LED_SWITCH_PIN -1 // External led lightning ON/OFF. If not used, SET TO -1
- #define DECIMALPOINT_PIN -1 // Nixie decimal point between digits. If not used, SET TO -1
- #define ALARMSPEAKER_PIN -1 // *Buzzer*
- #define ALARMBUTTON_PIN -1 // *Snooze / alarm off button*

- #define RADAR_PIN 34 //Radar sensor pin
- #define TUBE_POWER_PIN 23 //Filament or HV switch ON/OFF pin
- #define LIGHT_SENSOR_PIN -1 //Environment light sensor, only ADC pins are usable!

PIN_SDA and ***PIN_ SCL*** pins are used by RTC and every I2C temperature and humidity sensors.

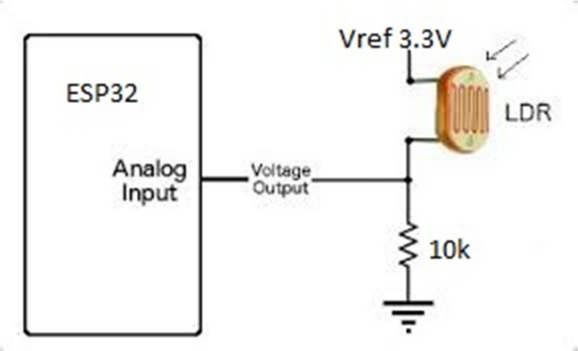
COLON_PIN output pin switches the blinking dot between hours and minutes (see „***Blinking Dots***"). It is used in Nixie clocks if there is no decimal point in the tube.

LED_SWITCH_PIN is an output pin. It is ON during daytime and OFF during nighttime. Can be used to switch any external circuit.

RADAR_PIN: Input pin for radar module or infrared motion sensor. If installed, the display will switch off after ***RADAR_TIMEOUT*** seconds.

TUBE_POWER_PIN: Output pin for filament or anode HighVoltage ON/OFF pin. The ON output value can be defined here: ***TUBE_POWER_ON*** **HIGH** or **LOW**. The power will switch off, if the actual tube brightness is zero or no Radar sensor movement is detected (if there is a radar installed).

LIGHT_SENSOR_PIN: Input pin, for ***e***nviroment light sensor, Lux calculation for automatic tube and LED brightness control in daytime. Only analog ADC input pins are usable.
MAXIMUM_LUX sets the Lux limit, when the maximum brightness will be set. Default value is 150 Lux.
Automatic brightness control is continuos without brightness steps between tubes brightness 1-10 and LED brightness ***RGB_MIN_BRIGHTNESS*** and ***RGB Brightness*** setting on web page.
ESP32 Vref = 3.3 Volt, but ***8266 Vref = 1.0 Volt only***!



I hope, the other pins are self-explanatory…

In the following section you can define what data to be displayed at a given second of every minute. (These are the parameters between TEMP_START and DATE_END)

ENABLE_CLOCK_DISPLAY inhibits time and date display – useful if you want only temperature and humidity displayed, without the clock function.

SHIFT_TUBES_LEFT_BY_1 shifts the display to the left by 1 digit, useful for thermometer clocks.

ANIMSPEED value sets the speed of display transitions.

TEMP_CHARCODE, ***GRAD_CHARCODE*** and ***PERCENT_CHARCODE*** sets the character code for Celsius, degree, and percentage symbols. For 7-segment tubes the proper value is already defined. For Nixie- tubes with special symbols

the “number” must be set where the symbol is displayed. For example, if the tube has a “%” symbol instead of the number 7, you want to put 7 in here to display that percentage symbol.

Code sample:

```
//Display temperature and date in every minute between START and END seconds
#define ENABLE_CLOCK_DISPLAY true //false, if no clock display is needed (for example:
thermometer + humidity only)
#define LEFTDECIMAL false
#define TEMP_START  35
#define TEMP_END    40
#define HUMID_START 40
#define HUMID_END   45
#define DATE_START  45
#define DATE_END    50
#define ANIMSPEED   50  //Animation speed in millisec
#define TEMP_CHARCODE 15
#define GRAD_CHARCODE 16
#define PERCENT_CHARCODE 17   //Humidity %
#define DOUBLE_BLINK    //both separator points are blinking (6 or 8 tubes VFD clock)
```

Minimum and maximum brightness for Neopixel LEDs can be set here:

```
#define RGB_MIN_BRIGHTNESS 8   //Neopixel leds minimum brightness
#define RGB_MAX_BRIGHTNESS 255 //Neopixel leds maximum brightness
```

Wi-Fi network data goes into the next block:

```
char webName[] = "UniClock 2.2c";
#define AP_NAME "UNICLOCK"
#define AP_PASSWORD ""
```

webName[] Name displayed on the website (i.e. "Joe’s clock")
AP_NAME Name of the Wi-Fi network to be created
AP_PASSWORD Wi-Fi password. If not set, then the network is open.

Further options for customization in the main program:

The factoryReset() section contains the factory pre-set values.

Customizing the clock’s website:

The files of the website can be found in the **/dat** library. This must be uploaded to the clock using the SPIFFS uploader program.

After successful program upload switch on Serial Monitor (115200 baud) and check it!
You will see, what happens, when the clock starts. For example:

```
Starting ESP32 IN-11 UniClock 2.5
- ALARMSPEAKER_PIN: GPIO33
- RADAR_PIN: GPIO21
- LIGHT_SENSOR_PIN: GPIO35
- TEMP_DALLAS_PIN: GPIO23
DS18B20 sensors found:1
Temperature requested
```

```
Setup MAX6921 pins for VFD Clock...
- PIN_LE: GPIO14
- PIN_BL: GPIO12
- PIN_DATA: GPIO27
- PIN_CLK: GPIO13
--- Generating segment pins bitmap ---
--- Generating digit pins bitmap ---
---- Generated Character / Pins table -----
Number of digits:6
Disable tubes
Loading setting from EEPROM. EEPROM ver:200
Enable tubes:0
Setup NeoPixel LEDS
- NEOPIXEL_PIN: GPIO2
Pixel count: 10
Brightness:8 - 255
___ USED CLOCK PINS ___
2: NEOPIXEL_PIN
12: PIN_BL
13: PIN_CLK
14: PIN_LE
21: RADAR_PIN
23: TEMP_DALLAS_PIN
27: PIN_DATA
33: ALARMSPEAKER_PIN
35: LIGHT_SENSOR_PIN
_________________________
```

Settings of the most often used universal display driver programs

multiplex74141.ino and multiplex74141_ESP32.ino

Any number of tubes can be driven by one 74141 in multiplex mode.

The GPIO numbers used for the tubes' ENABLE pins must be set:

```
byte digitEnablePins[] = {14,12,13,15};
```

Also the ABCD control pins for the 74141:

```
byte ABCDPins[4] =  {2,4,5,0};
```

maxDigits is the number of tubes (or digits):

```
const int maxDigits = sizeof(digitEnablePins); Don't change manually!
int PWMrefresh=11000; //msec, Multiplex period. Greater value => slower multiplex freq
int PWMtiming[MAXBRIGHT+1] = {0,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000};
```

PWMrefresh Multiplex time constant for one tube

PWMtiming[] Time constant for each brightness level 0..10.

The program will switch on a digit for **PWMtiming[brightness]** period and switch off for **PWMrefresh - PWMtiming[brightness]**

PWMrefresh is higher by about 10% than the maximum brightness time value (**PWMtiming[10]**). This is to make sure that the tube is turned off, to avoid the „ghost effect". If the tubes are driven by transistors this can be a small value, but with optocouplers it must be set properly (some optocouplers may turn off very slowly).

Depending on the number of tubes (digits) it may be necessary to fine tune the above timing to avoid light scintillation. Also, you can adjust the brightness level for each (0..10) value to your liking.

MAX6921.ino and MAX6921_ESP32.ino

The program can drive multiple VFD tubes with one MAX6921 IC in any configuration.

segmentEnablePins[] defines which OUT pins of the MXX6921 are connected to the a,b,c,d,e,f,g,h segments. **There must be 8 pins defined!** Segment "h" is the decimal point.

digitEnablePins[] defines which OUT pins will turn on the digits. You must define one pin per each digit. (This example is for the IV18 multi-digit long tube.)

MAX6921 pins section defines the GPIO control pins.

```
//Fill this table with the OUT positions of the MAX6921 chip!
const byte segmentEnablePins[] = {0,2,5,6,4,1,3,7}; //segment enable OUTbits of MAX6921
(a,b,c,d,e,f,g,DP)
const byte digitEnablePins[] = {18,11,17,12,16,13,14,15}; //digit enable OUTbits of MAX6921

//MAX6921 pins
#define PIN_LE   12 // D6 Shift Register Latch Enable
#define PIN_CLK  13 // D7 Shift Register Clock
#define PIN_DATA 14 // D5 Shift Register Data
#define PIN_BL   15 // D8 Shift Register Blank (1=display off, 0=display on)
```

charDefinition[] contains the "special" character codes. Now you can see why the **#define TEMP_CHARCODE 15** and **#define GRAD_CHARCODE 16** program lines were used for Celsius and percentage symbols…

Values below work the same way as explained with the **multiplex74141.ino** module.

```
int PWMrefresh=5500;
int PWMtiming[MAXBRIGHT+1] = {0,250,500,1000,2000,2500,3000,3500,4000,4500,5000};
```

General remarks

It is generally not recommended to put long code within an interrupt handler. To maintain critical timing during

EEPROM updates, the **EEPROMsaving** flag prevents the program from refreshing the clock. This might be perceived as a momentary blink or freeze of the display.

Project files can be downloaded from:

https://github.com/gpeter62/ESP_UniClock

For further information, support, feedback and suggestions please contact me via email: gautier.p62@gmail.com

DISCLAIMER

About the Clock Projects

The Author gives absolutely no guarantee to the project, descriptions, plans or any piece of information presented in these files and documents are 100% correct. Best efforts has been made to minimize the errors and the author believes all the samples present do work. Readers must do their research and understand the concepts to the core before proceeding with any of these projects.

I reserve the right to change any information in these files and documents without prior notice either to make correction or improvements.

Copyrights

Damage/loss:

The author and his associates cannot be held responsible for any damages in the form of physical loss / monetary loss or any kind of loss that comes as the result of making projects that are presented in these documents and files.

The projects and tutorials present are only for educational purpose and should not be replicated or modified for the sake of hurting any human (including self-hurting) / animals / private or public property.

The person / group of people involving in making the projects presented are solely responsible for their results, either good or bad.

Precautions:

The person / group of people must take enough precautions before proceeding any project presented, such as wearing rubber shoes and electrical gloves while working with high voltage parts, wearing eye goggles and allowing fresh air while soldering etc to say a few.

Beginners and inexperienced individuals in electronics / electrical industry should NOT make any high voltage / AC mains projects or parts.

The readers who agree to this disclaimer may follow the information / ideas / make projects, readers who disagree may not follow any information or make any projects presented.

Document version v1.0 (12/14/2020)

Sample clock settings in clocks.h

The number of clocks may be freely expanded – you can create a profile for each clock you have. When updating the software there is no need to work with individual program files – every parameter can be set in **clocks.h** and remains unchanged. When uploading a clock, that specific one has to be enabled for upload: that will be CLOCK_3 in this example.

```
//#define CLOCK_1
#define CLOCK_3
//#define CLOCK_40
//#define CLOCK_41
//#define CLOCK_21

#ifdef CLOCK_1   //8266, UNFI PCB clock, 4x IN-16 tubes
  #define DEBUG
  #define USE_NEOPIXEL
  byte tubePixels[] = {3,2,1,0};        //4 tubes, single leds, reverse direction
  //#define USE_DALLAS_TEMP
  #define TEMP_DALLAS_PIN -1    //DHT or Dallas temp sensor pin.  If not used, SET TO -1
  #define MULTIPLEX74141
  const byte digitEnablePins[] = {14,12,13,15};   //IN16 4x tube clock
  const byte ABCDPins[4] =  {2,4,5,0};
  #define COLON_PIN   16        //Blinking Colon pin.  If not used, SET TO -1
  #define AP_NAME "UNICLOCK"
  #define AP_PASSWORD ""
  #define WEBNAME "IN-16 Nixie Clock"
#endif


#ifdef CLOCK_3   //8266, PCB less clock, IV-18 VFD tube
https://www.thingiverse.com/thing:3417955
  #define DEBUG
  #define USE_DALLAS_TEMP
  #define TEMP_SENSOR_PIN 4    //DHT or Dallas temp sensor pin.  If not used, SET TO -1
  #define MAX6921
  const byte segmentEnablePins[] =  {0,2,5,6,4,1,3,7};   //segment enable OUTbits of MAX6921
(a,b,c,d,e,f,g,DP)  (You MUST define always 8 Pins!!!)
  const byte digitEnablePins[] = {18,11,17,12,16,13,14,15}; //19};  //digit enable OUTbits of
MAX6921 1,2,3,4,5,6,7,8)  (You may define any number)
  //MAX6921 pins
    #define PIN_LE    12  // D6 Shift Register Latch Enable
    #define PIN_CLK   13  // D7 Shift Register Clock
    #define PIN_DATA  14  // D5 Shift Register Data
    #define PIN_BL    15  // D8 Shift Register Blank (1=display off     0=display on)
  #define AP_NAME "UNICLOCK"
  #define AP_PASSWORD ""
  #define WEBNAME "IV-18 VFD Clock"
#endif


//_______ESP-32 CLOCKS  (2x20pin Wemos D1 mini ESP32 modul) _______________
#ifdef CLOCK_40   //ESP32 WEMOS D1 mini, UNFI 6 x IV-11 VFD tubes clock
  #define DEBUG
  #define USE_NEOPIXEL
  #define NEOPIXEL_PIN 22
  byte tubePixels[] = {0,1,2,3,4,5};    //6 tubes, single leds
  //#define USE_DALLAS_TEMP
  #define USE_DHT_TEMP
  #define DHTTYPE DHT22
  #define TEMP_DHT_PIN 25    //DHT temp sensor pin.  If not used, SET TO -1
  #define MAX6921_ESP32
  const byte segmentEnablePins[] =  {19,17,15,12,13,16,18,14};   //segment enable OUTbits of
```

```
MAX6921 (a,b,c,d,e,f,g,DP)  (You MUST define always 8 Pins!!!)
  const byte digitEnablePins[] = {9,8,7,2,1,0};  //digit enable OUTbits of MAX6921 (1,2,3,4,5,6)
(You may define any number)
  #define DOUBLE_BLINK  //both separator points are blinking   //MAX6921 pins
    #define PIN_LE    4  // Shift Register Latch Enable
    #define PIN_CLK   17  // Shift Register Clock
    #define PIN_DATA  16  // Shift Register Data
    #define PIN_BL    32  // Shift Register Blank (1=display off     0=display on)
  #define ALARMSPEAKER_PIN 2   //Alarm buzzer pin
  #define ALARMBUTTON_PIN 0    //Alarm switch off button pin
  #define ALARM_ON HIGH         //How to switch ON alarm buzzer
  #define AP_NAME "UNICLOCK32"
  #define AP_PASSWORD ""
  #define WEBNAME "ESP32 IV-11 VFD-Clock"
#endif



#ifdef CLOCK_41   //ESP32 WEMOS D1 mini, UNFI board, 6 x Z573M Nixie tubes
  #define DEBUG
  #define USE_NEOPIXEL
  #define NEOPIXEL_PIN 22
  byte tubePixels[] = {0,1,2,3,4,5};    //6 tubes, single leds
  #define USE_DALLAS_TEMP
  //#define USE_DHT_TEMP
  //#define DHTTYPE DHT22
  //#define TEMP_DHT_PIN 25
  #define TEMP_DALLAS_PIN 25    //Dallas temp sensor pin.  If not used, SET TO -1
  #define MULTIPLEX74141_ESP32
  const byte digitEnablePins[] = {26,18,33,19,23,5};   //ESP32 6x tube Clock
  const byte ABCDPins[4] =  {32,16,4,17};
  #define DP_PIN 27             // decimalPoint inside Nixie tube, set -1, if not used!
  #define LEFTDECIMAL false   //set true (Z574M), if decimal point is on the left side on the
tube. Else set false (Z573M)!
  //#define TEMP_CHARCODE -1   //disable char => shift display right with 1 digit
  //#define GRAD_CHARCODE -1   //disable char => shift display right with 1 digit
  #define ALARMSPEAKER_PIN 2   //Alarm buzzer pin
  #define ALARMBUTTON_PIN 0    //Alarm switch off button pin
  #define ALARM_ON HIGH
  #define AP_NAME "UNICLOCK32"
  #define AP_PASSWORD ""
  #define WEBNAME "ESP32 Z573M Nixie-Clock"
#endif
```