ESP32_UniClock v2.3 program használata

Ebben a dokumentumban megpróbálom összefoglalni az UniClock általam fejlesztésű program működését, telepítését, adaptálását. A program már évek óta fejlődik, köszönhetően *Unferdorben Zoltánnak*, aki mindig új ötletekkel és remek óra panelekkel állt elő. ☺ És tesztelte a folyamatosan fejlődő és mindig új hibákkal rendelkező programverziókat.
Valamint az utóbbi időben *Pintér Sándor* is beszállt a projektbe remek ötletekkel, neki köszönhető a hőmérséklet páratartalom mérés továbbfejlesztése és a Neopixeles led világítás funkcionalitásának kitalálása, valamint végtelenül precíz és szorgalmas tesztelése. ☺
És *Gautier Bálintnak* köszönhető az új alapokra helyezett web oldal.
Az interneten létezik más hasonlóan Uniclock néven szereplő moduláris Nixie óra panel, ennek semmilyen köze sincsen ehhez a projekthez. Sajnálatos névegyezés.

Az UniClock egy Arduino környezetben működő univerzális óra program, amely ESP8266 és ESP32 mikrokontrolleren alapulva Nixie, VFD, Numitron csöves és LED-es kijelzőkkel is működik.
Az óra a pontos időt WiFi-n keresztül az internetes timeserverről, GPS modulról vagy saját DS3231 RTC modulról szinkronizálva biztosítja. Az óra lehet 4, 6 vagy 8 digites, tíz lépésben állítható nappali és éjszakai fényerővel, számjegy animációval. Teljesen önálló, internet független működés is lehetséges, nyomógombos óra/perc beállítással és RTC modullal.

Az óra működési paramétereit saját web oldalán keresztül lehet beállítani. Amennyiben nincsen folyamatos wifi kapcsolat, akkor az óra saját AP-ként működik, így biztosítja a web oldala elérhetőségét. Lehetőség van csövenkénti alá és háttérvilágításra Neopixel WS2812 címezhető ledekkel, választható animációval, fényerővel, stb.
Alarm funkció piezo hangjelzővel, opcionális leállító gombbal.

A github oldal tartalma:
A **https://github.com/gpeter62/ESP_UniClock** *oldalon számos bevált órának megtalálható a kapcsolási rajza és panelterve, Unferdorben Zoltánnak köszönhetően a „/schematics” alkönyvtárban. (Igény esetén gyári panelek és kész órák is hozzáférhetőek.)*

Vannak nyilvánosan elérhető internetes óra projektek, melyek szintén működtethetőek ezzel a programmal. Ezek közül kettő szintén megtalálható a kapcsolások között.

Letölthető sok adatlap a „/Datasheets” alkönyvtárból, amelyeket az órákban szoktunk használni.

Az „/old_versions” alatt korábbi, stabil program verziók találhatóak zip fájban tömörítve.

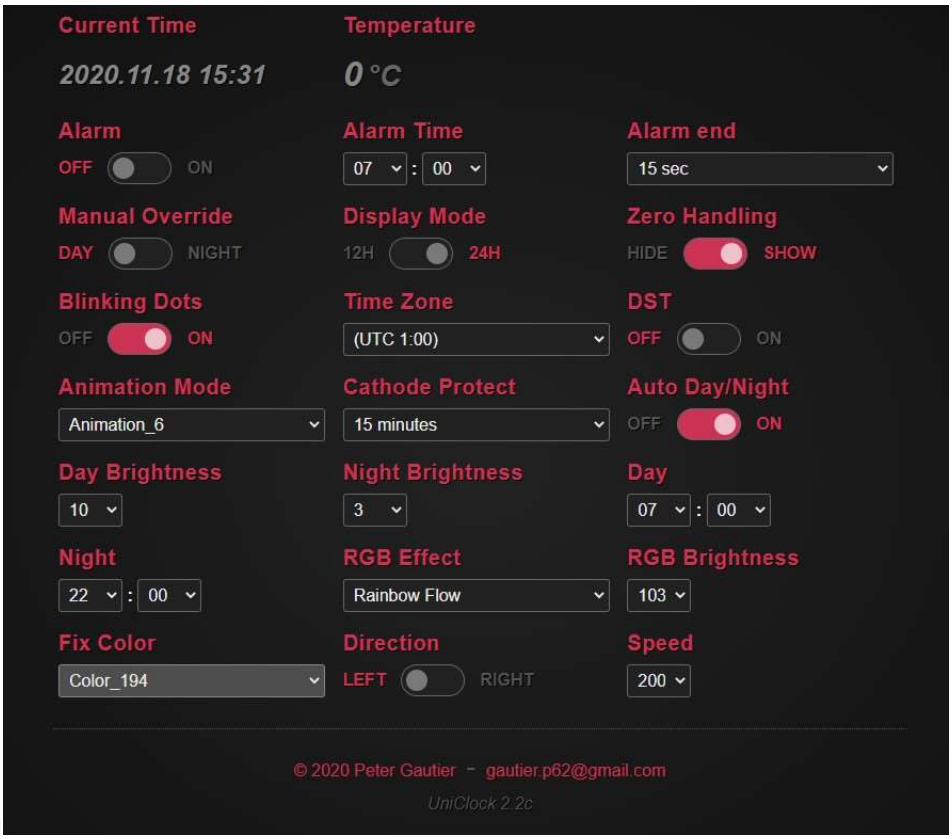
Felhasználói leírás - az óra működése

Az óra bekapcsolás után elszámol az összes csövön 0-tól 9-ig.

- Önálló működésű óra esetében (GPS vagy RTC szinkron)
Az óra azonnal létrehoz egy ***UNICLOCK*** nevű saját wifi hálózatot. (Vagy ami meg lett adva AP_NAME-ben.)
Erre a hálózatra bármikor fellépve például egy mobiltelefonnal azonnal be lehet jelentkezni az óra web oldalára, melynek címe 192.168.4.1, induláskor ezt ki is írja a kijelzőre.

- WiFi-s szinkonizálású óra esetében
Megpróbál fellépni az utoljára használt wifi hálózatra a korábban beállított jelszóval. Ha ez nem sikerül, akkor létrehoz egy ***UNICLOCK*** nevű saját wifi hálózatot. Erre fellépve például egy mobil telefonnal bejön egy web oldal, ahol ki lehet választani az elérhető wifi hálózatok közül, amit használni szeretnénk és megadhatjuk a jelszót. Ez tárolásra kerül és legközelebb már magától fellép ide. (Wifimanager funkció)
Ezután az óra fellép a kiválasztott hálózatra a megadott jelszóval. ha sikeres, a kijelzőre kiírja a kapott IP címét. (például 192.168.1.106)
Az óra web oldala ezen a címen érhető el bármilyen eszközről, amely ugyanezt a wifi hálózatot használja.

Az óra web oldala



Az oldalon jelenleg az összes funkció látható, akkor is, ha az adott órában esetleg ez nem lett beépítve. A későbbiekben ez finomításra kerül.
Nappali módban minden funkció működik. Az éjszakai módban kikapcsol a led világítás, az elválasztó pont villogtatása és átvált az óra éjszakai fényerőre.

Current time: pontos idő.
Temperature: Amennyiben van hőmérő, akkor a hőmérséklet, amúgy nem látható.
Humidity: Amennyiben van páratartalom mérés, akkor a páratartalom, amúgy nem látható.

Alarm: bekapcsolva az adott időpontban, maximum a magadott időtartamig ébreszt. Amennyiben van kikapcsoló gomb, akkor azzal bármikor leállítható. Amennyiben van Neopixel led az órában, akkor ezeket fehér színben nagy fényerővel villogtatja. Amennyiben van piezo hangjelző, akkor az egyre gyorsabb pittyegéssel jelez.

Manual Override: Kézi átkapcsolás az éjszakai és nappali mód között.
Display Mode: 12/24**:** 12 vagy 24 órás kijelzési mód.
Zero Handling: 12 órás módban mutassa-e a nullát az óra tízes számjegyen.
Blinking Dots: Engedélyezi az elválasztó pont villogtatását. Különben folyamatosan világít.
Time Zone: Időzóna. Magyarországon „1”
DST: Nyári időszámítás bekapcsolása. (Szerencsére többé nem fog kelleni…)
Animation mode: a kijelzőn animálja a számjegyeket 5 különböző módon.
0 = kikapcsolva, 1-5 = animációs módok, 6 = mindig véletlen animáció az 1-5 közül.
Cathode Protect: a megadott időközönként körbefuttatja a kijelzőn a számokat. Ez a Nixie órák esetén fontos, hogy megvédje a csöveket a katód mérgezéstől. Egyébként pedig egy látványos effekt.

Auto Day/Night: engedélyezi az automatikus váltást az éjszakai és a nappali üzemmód között.
Day Brightness: nappali fényerő (1-10 között).
Night Brightness: éjszakai fényerő (0-10 között). Ha nulla, teljesen ki van kapcsolva a kijelző.
Day és ***Night***: a nappal és az éjszaka kezdő időpontja.

Amennyiben van neopixel led világítás, akkor megjelenik az oldalon:

RGB Effect: a Neopixel led animáció mód, jelenleg „OFF” állapot és 10 féle animáció választható.
RGB Brightness: fényerő
Fix Color: (0-256) amennyiben fix szín „animáció”van kiválasztva, akkor csak ez a szín fog világítani.
Direction Left/Right: a futó animáció iránya
Speed: az animáció sebessége

A színek 0..255 között a színkör alapján működnek, a 256 a fehér szín, a kakukktojás.

A program paraméterezése és feltöltése a mikrokontrollerre:

A környezet telepítése a szokásos módon történik:

Az Arduino IDE fejlesztő környezet sikeres telepítése után még telepíteni kell az „Eszközök/Alaplap kezelő”-ben a megfelelő ESP8266 vagy ESP32 alaplapokat. Ki kell választani a használt alaplapot. A CPU frequency 160MHz és a Flash Size beállításnál legalább 160kbyte FS-t engedélyezni kell az SPIFFS partíciónak.

Telepíteni kell az „***ESPxx Sketch DataUpload***” feltöltő programot, ami a /dat könyvtárból feltölti az SPIFFS fájlrendszerre a web oldalhoz szükséges fájlokat.

Ennek módját ez a leírás bemutatja:
8266 esetén: https://randomnerdtutorials.com/install-esp8266-filesystem-uploader-arduino-ide/

ESP32 esetén: https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/

A UniClock programot a szokott módon az Arduino könyvtárba kell telepíteni. (Általában Sajátgép/Dokumentumok/Arduino helyen található.)
A működéshez szükséges library-ket (ezek jelenleg aktuális változatás) szintén a zip fájlokból kibontva a szokásos módon az Arduino/libraries alá kell telepíteni. (Általában Sajátgép/ Dokumentumok/Arduino/libraries helyen található.)

A program beállítása az óra hardverhez:

A ***clocks.h*** fájlban fordítás előtt az alábbi paramétereket lehet a használt hardver ismeretében beállítani (a főprogramban és az alprogramokban elvileg már nem kell semmit beállítani.) Amelyik paraméter nem került beállításra egy óránál, az alapértelmezés szerint ki van kapcsolva illetve egy alapértelmezett értékre áll be. Tehát minden óránál csak azt kell beállítani, ami ott konkrétan használatva van a hardverben, a nem használt opciókat nem kell mindig beállítgatni.

```
/*_____________________________ USABLE PARAMETERS  ____________________________________
//#define DEBUG             //Enable Serial Monitor, 115200baud (only, if TX pin is not used
//---------------------------- CLOCK EXTRA OPTION PARAMETERS -------------------------------------------------
//#define USE_DALLAS_TEMP      //TEMP_SENSOR_PIN is used to connect the sensor, temperature measure
//#define USE_DHT_TEMP        //TEMP_SENSOR_PIN is used to connect the sensor,  temperature and humidity measure
//#define USE_RTC             //I2C pins are used!   SCL = D1 (GPIO5), SDA = D2 (GPIO4)
//#define USE_GPS             //use for standalone clock, without wifi internet access
//#define USE_NEOPIXEL_MAKUNA   //WS2812B led stripe, for tubes lightning. Don't forget to define tubePixels[] !

//----- DRIVER SELECTION - ----- Use only 1 driver from the following options in the clocks.h file!
//#define MULTIPLEX74141   //4..8 Nixie tubes generic driver for ESP8266 or ESP32
//#define MAX6921         //4..8 VFD tubes (IV18) driver for ESP8266 or ESP32

//---------------- ONLY 8266 clock drivers -------------------------------------------------
//#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

//------------------------------------------- pinout ----------------------------------------------------
#define COLON_PIN   -1       //Blinking Colon pin.  If not used, SET TO -1
#define TEMP_SENSOR_PIN -1    //DHT or Dallas temp sensor pin.  If not used, SET TO -1
#define DHTTYPE DHT22           //DHT sensor type, if used
#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   //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 MAXBRIGHTNESS 10 // (if MM5450, use 15 instead of 10)

//Display temperature and date in every minute between START..END seconds
#define ENABLE_CLOCK_DISPLAY true  //false, if no clock display is needed (for example: thermometer + hygrometer only)
#define SHIFT_TUBES_LEFT_BY_1      //shift left by 1 tube the display, if a thermometer is used with spec tube
#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"
#define GRAD_CHARCODE 16     //Thermometer grad
#define PERCENT_CHARCODE 17   //Hygrometer %

 #define AP_NAME "UNICLOCK"
 #define AP_PASSWORD ""
 #define WEBNAME "UniCLock v2.3"   //Name of the clock on web page
```

Ha a ***DEBUG*** engedélyezve van, akkor a Soros Monitoron a program induláskor mindent kiír, követhető, mi történik. Sebesség: 115200 baud.

A többi **CLOCK EXTRA OPTIONS** sor esetében ha a // kivesszük a sor elején, akkor engedélyezi az adott modul használatát.

USE_DALLAS_TEMP: Dallas 18b20 hőmérséklet szenzor engedélyezése.
A TEMP_SENSOR_PIN értékét be kell állítani, maximum 2 szenzor használata támogatott egy közös buszon. (2 szenzor esetén a szenzorok sorrendjét a a dallas_temp.ino modul elején lehet beállítani.)

USE_DHT_TEMP: DHT11, DHT21, vagy DHT22 hőmérő és páratartalom szenzor használható.
#define DHTTYPE DHT22 sorban adható meg a szenzor pontos típusa.
A TEMP_SENSOR_PIN értékét be kell állítani, maximum 1 szenzor használata támogatott.
Egyszerre nem használható a Dallas szenzorral!

USE_GPS : Ha a GPS engedélyezve van, akkor nem fog wifi hálózatot keresni, mindenképpen a GPS-ről szinkronizálja az órát, ez önálló működésű üzemmód. (***gps.ino*** modul bekapcsolásra kerül.) Jelenleg a GPS paramétereit is itt kell beállítani, amennyiben eltérnek a szokásostól.

USE_RTC: Ha RTC engedélyezve van, akkor DS3231 RTC modult fog használni, I2C buszon. (Az I2C busz SDA és SCL GPIO lábak megadhatóak a ***rtc.ino*** modulban.) Az rtc.ino modul be lesz kapcsolva, itt adható meg a nyomógombok és az üzemmód kapcsoló GPIO lába. RTC esetén a GPS nem választható. Ez a mód az üzemmód választó kapcsoló alapján működhet wifi óraként is, frissítve az RTC időt az internetről vagy önálló óraként is, csak az RTC-re támaszkodva. Ekkor kézzel beállítható a pontos idő a nyomógombokkal.

USE_NEOPIXEL_MAKUNA : WS2812 Neopixel ledek használhatóak. (Makuna-féle library van használva, ez a *z_neopixel_makuna.ino* modul)
A modulban a tubePixels[] tömbben kell definiálni, hogy a ledsor egyes pixelei melyik csőhöz tartoznak. Lehet, hogy csöveként csak egy led szükséges, de lehet több led is csövenként.
8266 esetében csak az RX láb használható vezérlésre, ESP32 esetén bármelyik 32-nél kisebb GPIO láb alkalmas.

Kijelző meghajtó modul kiválasztása:
Az ESP32 lapok esetében csak 2 meghajtó típusból választhatunk jelenleg:

```
//#define MULTIPLEX74141   (univerzális Nixie meghajtó)
//#define MAX6921   (univerzális 7 szegmenses VFD meghajtó)
```

8266 esetében még további meghajtó programok is rendelkezésre állnak:

```
#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_MULTIPLEX74141  //PCF pin expander for tube selection
```

Egyéb vezérlő lábak megadása:
A megadható láb kiosztás értelemszerűen megegyezik mindkét esetben. Ha egy funkció nincsen használva, akkor az értékét -1 –re kell állítani. **Mindig a GPIO számot célszerű megadni, ez nem függ a modulok típusától!!!**

```
#define COLON_PIN   -1      //Blinking Colon pin.  If not used, SET TO -1
#define TEMP_SENSOR_PIN -1   //DHT or Dallas temp sensor pin.  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   //Alarm buzzer pin
#define ALARMBUTTON_PIN -1    //Alarm switch off button pin
```

A ***COLON_PIN*** az óra és perc csövek között villogó pontokat kapcsolja. („***Blinking Dots***”) Nixie csöveknél szoktuk használni, ha nincsen tizedes pont a csőben.
A ***LED_SWITCH_PIN*** egy kapcsoló láb, amely nappal bekapcsol, éjjel kikapcsol. Bármilyen külső vezérlésre felhasználható.
A többi pin funkciója remélhetőleg értelemszerű mindenkinek.

```
#define MAXBRIGHTNESS 10
```

Ez a csövek maximális fényereje, nem érdemes piszkálni. Általában 10, egyetlen kivétel jelenleg a MM5450 led meghajtó modul, itt ez az érték 15, mivel az IC ezt támogatja közvetlenül.

A következő blokkban azt adhatjuk meg, hogy az óra egy percen belül hányadik másodpercnél melyik adatot írja ki a kijelzőre. (TEMP_START … DATE_END közötti paraméterek)
Az ***ENABLE_CLOCK_DISPLAY*** segítségével letiltható a teljes idő és dátum kijelzés, ennek csak akkor van értelme, ha

kizárólag hőmérséklet és páratartalom mérőt szeretnénk óra funkció nélkül.
A SHIFT_TUBES_LEFT_BY_1 paraméter 1 digittel balra eltolja a kijelzést, hőmérős óráknál.
Az ***ANIMSPEED*** érték a kijelző áttűnési animációk sebességét határozza meg.
A ***TEMP_CHARCODE***, a ***GRAD_CHARCODE*** és a ***PERCENT_CHARCODE*** értéke a Celsius, a fok és a „%” karakter kódját határozza meg. 7 szegmenses csöveknél ez a szokásos érték. Amennyiben speciális karakteres Nixi csövet használunk, akkor azt a számot kell beírni, amelyik számjegy helyén az adott jelzés található.

```
//Display temperature and date in every minute between START..END seconds
#define ENABLE_CLOCK_DISPLAY true   //false, if no clock display is needed (for example: thermometer + hygrometer only)
#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
```

A Neopixel ledeknél a megengedett minimum és maximum fényerőt adhatjuk meg itt:

```
byte c_MinBrightness = 8;
byte c_MaxBrightness = 255;
```

A következő paraméter blokkban a wifi kezelés adatait adhatjuk meg:

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

webName[] az weboldalon kiírt név, szabadon megadható (pl. „Pistike órája”)
AP_NAME a létrehozandó AccessPoint (röviden AP) neve
AP_PASSWORD az AP eléréséhez szükséges jelszó. Jelenleg nincsen beállítva, nyitott az AP.

További testreszabási lehetőségek a főprogramban:
A factoryReset() programrészben adjuk meg a gyári alapállapothoz tartozó értékeket.

WiFi web oldal módosítása:

A „/dat” könyvtárban találhatóak a web oldal fájlai. Azt az SPIFFS uploaderrel kell feltölteni az ESP modulra.

ESP32_UniClock v2.3 program használata

A legfontosabb két univerzális kijelző meghajtó program beállítása:

multiplex74141.ino

A program multiplex módon egy darab 74141 segítségével hajt meg tetszőleges számú Nixie csövet.

Értelemszerűen meg kell adni a csövek engedélyező lábait kapcsoló GPIO számokat:
const byte digitEnablePins[] = {14,12,13,15};

És a 74141 ABCD vezérlő lábait vezérlő GPIO számokat: const byte ABCDPins[4] = {2,4,5,0};

A maxDigits egy nagyon fontos érték, ez a csövek darabszáma, ezt a program innen fogja megtudni, kézzel nem szabad változtatni! const int maxDigits = sizeof(digitEnablePins);

const int PWMrefresh=11000; //msec, Multiplex time period. Greater value => slower multiplex freq
const int PWMtiming[MAXBRIGHT+1] = {0,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000};

A ***PWMrefresh*** egy darab cső multiplex időállandója.
A ***PWMtiming[]*** az egyes 0..10 fényerőértékekhez tartozó időállandó.
A program a fényerőhöz tartozó értékig kigyújtja a csövet, majd a
PWMrefresh - PWMtiming[fényerő] ideig kikapcsolva tartja.
Majd lép tovább a következő csőhöz.
Látható, hogy a PWMrefresh értéke nagyobb kb.10%-kal, mint a legnagyobb fényerőhöz tartozó időzítés. Ez azért van, hogy a mindenképpen stabilan kikapcsoljon a cső és ne legyen ghost effektus.
Amennyiben a hardver tranzisztoros meghajtású, akkor ez a különbség lehet minimális, de optocsatolós meghajtásnál ez feltétlenül szükséges, mert az elég lassan olt ki.
A csövek számától függően szükséges lehet az időzítések változtatása, hogy sok csőnél se legyen vibrálás. Illetve, bárki ízlése szerint szabadon beállíthatja, hogy mekkora fényer értékek tartozzanak az egyes fényerő szintekhez.

MAX6921.ino

A program egy darab MAX6921 IC segítségével hajt meg tetszőlegesen bekötött VFD kijelzőket.
A ***segmentEnablePins[]*** tömbben meg kell adni, hogy az MAX6821 melyik OUT lábain találhatóak az a,b,c,d,e,f,g,h szegmensek. Mindig kötelező 8 értéket megadni!!! A „h” szegmens értelemszerűen a tizedespont.
digitEnablePins[] tömbben meg kell adni, hogy melyik OUT lábak kapcsolják a digiteket. Annyi lábat kell megadni, ahány számjegy van. (Ez a példa egy IV18 hosszú, sok számjegyes csőhöz készült.)
A MAX6921 pins szekcióban értelemszerűen az IC-t vezérlő ESP GPIO lábai vannak megadva.

//Fill this table with the OUT positions of the MAX6921 chip!
byte segmentEnablePins[] = {0,2,5,6,4,1,3,7}; //segment enable OUTbits of MAX6921 (a,b,c,d,e,f,g,DP)
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)

A charDefinition[] táblázat értelemszerűen a karakterkészletet definiálja. Itt látható, hogy korábban miért a #define TEMP_CHARCODE 15 és a #define GRAD_CHARCODE 16 határozta meg Celsius és % jeleket. ☺

Az alábbi értékek a ***multiplex74141.ino*** modulnál leírt logikával működnek.

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

Általános megjegyzés:
Mivel általában nem illik ilyen hosszú programokat megszakítás rutinba tenni, ezért a kritikus flash memória hozzáféréskor a ***EEPROMsaving*** változó jelzi a programban, hogy kicsit ne frissítsen, hogy ne zavarjon bele a flash időzítésbe. Ilyenkor egy pillanatra leáll a frissítés.

Letölthető: https://github.com/gpeter62/ESP_UniClock

További info és segítségkérés,észrevétel, javaslat: gautier.p62@gmail.com

Készítette: Gautier Péter, 2020. december 14

Minta óra beállítások a clocks.h fájlban:

Szabadon bővíthető az órák száma., mindenki elkészíthető a saját órájainak megfelelő profilokat Szoftver frissítésnél nem kell belenyúlni az egyes program modulokba, az összes paraméter itt megadható és nem változik. Nyilván egy konkrét órára feltöltéskor mindig azt az órát kell engedélyezni, amit feltölteni akarunk. (Itt a példában a CLOCK_3 az.)

```
//#define CLOCK_1
#define CLOCK_3
//#define CLOCK_31
//#define CLOCK_21

#ifdef CLOCK_1   //8266, UNFI PCB clock, 4x IN-16 tubes
  #define DEBUG
  #define USE_NEOPIXEL_MAKUNA
  byte tubePixels[] = {3,2,1,0};       //4 tubes, single leds, reverse direction
  //#define USE_DALLAS_TEMP
  #define TEMP_SENSOR_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};
  const int DpPin = -1;        // decimalPoint inside Nixie tube, set -1, if not used!
  #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
  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!!!)
  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



#ifdef CLOCK_31   //ESP32, UNFI board, 6 x Z573M Nixie tubes
  #define DEBUG
  #define USE_NEOPIXEL_MAKUNA
  byte tubePixels[] = {0,1,2,3,4,5};    //6 tubes, single leds
  #define USE_DALLAS_TEMP
  #define TEMP_SENSOR_PIN 23    //DHT or Dallas temp sensor pin.  If not used, SET TO -1
  #define MULTIPLEX74141
  const byte digitEnablePins[] = {4,16,17,5,18,19};   //ESP32 6x tube Clock
  const byte ABCDPins[4] =  {12,27,14,13};
  const int DpPin = 15; // 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 ALARMSPEAKER_PIN 33   //Alarm buzzer pin
  #define ALARMBUTTON_PIN 32    //Alarm switch off button pin
  #define ALARM_ON HIGH
  #define AP_NAME "UNICLOCK32"
  #define AP_PASSWORD "cicaMica2"
  #define WEBNAME "ESP32UniClock 2.3"
#endif

#ifdef CLOCK_21   //8266 D1-mini, P.S. PCB 4xIN14  thermometer / hygrometer
  //#define DEBUG
  #define USE_DHT_TEMP
  #define DHTTYPE DHT22
  #define TEMP_SENSOR_PIN  3  //RX
  #define MULTIPLEX74141
  const byte digitEnablePins[] = {15,13,12,14};
  const byte ABCDPins[4] =  {2,4,5,0};
  const int DpPin = -1; // decimalPoint inside Nixie tube, set -1, if not used!
  #define ENABLE_CLOCK_DISPLAY false  //don't display date/time!!!
```

```
  #define SHIFT_TUBES_LEFT_BY_1 //shift left by 1 tube the display, if a thermometer is used with spec tube
  #define TEMP_CHARCODE 4
  #define GRAD_CHARCODE 16
  #define PERCENT_CHARCODE 7
  #define LED_SWITCH_PIN   16   //external led lightning ON/OFF.  If not used, SET TO -1
  #define DECIMALPOINT_PIN 1 //TX   //Nixie decimal point between digits. If not used, SET TO -1
  #define AP_NAME "Nixie Homero 5.6"
  #define AP_PASSWORD "kutyus55!"
  #define WEBNAME "Nixie Homero 5.6"
#endif
```