

TÉMALABORATÓRIUM

JEGYZŐKÖNYV

WEMOS D1 MINI ÉS KIEGÉSZÍTŐ KÁRTYÁI ILLESZTÉSE STK3700-HOZ

KÉSZÍTETTE: DUDÁS TAMÁS ALEX
Konzulens: Naszály Gábor

KÉSZÍTÉS FÉLÉVE: 2022/23/1

TARTALOMJEGYZÉK

Specifikáció	3
Bevezetés	3
A téma értelmezése	3
A felhasznált eszközök bemutatása	3
Rendszerterv(ek)	3
Általános kiegészítő modulhoz (ún. shieldhez)	4
Specifikusan a Wemos D1 minihez	5
A kiegészítő áramkör részletes ismertetése	6
Kapcsolási rajz	6
NYÁK terv	7
Helyes használat	7
A megvalósított feladatok ismertetése	7
Alapvető kommunikáció	7
SHT30-as shield illesztése és annak használata	7
Webszerverbe épített WiFi scanner debuggerrel	8
Összegzés és tanulságok	9
Összegzés	9
Tanulságok	9
Irodalomjegyzék	9
[1]:LM75C adatlap:	9
[2]:VEML7700 adatlap:	9
[3]:SHT30 adatlap:	9
[4]: ESP dokumentáció (arduino környezetben):	9
[5]: Arduino beépített wifi scanner példakód:	9
Melléklet	9
Eszközökről képek:	9
Wemos D1 Mini/Tetszőleges shield helyes csatlakoztatása:	13
Kapcsolási rajz:	14
NYÁK:	16
Példa kódok:	16
Gecko – ESP egyszerű kommunikáció	17
Gecko – LM75C	19
Gecko – VEML770	22
Gecko – ESP – webserver	24

Specifikáció

A Giant Gecko STK3700-as mikrokontrollerhez illeszteni a Wemos D1 minit és kiegészítői kártyáit, ehhez megtervezni a nyomtatott áramkört, azt működésbe (bemérés, beültetés) hozni és példa alkalmazások útján demonstrálni, azok működésének helyességét.

Bevezetés

A téma értelmezése

A Giant Gecko STK3700-as fejlesztői kártyához olyan kiegészítő nyomtatott áramkör készítése és annak működésbe hozatala, amely lehetővé teszi, hogy más mikrokontrollereket és azok moduljait tudjuk hozzá illeszteni. Ez a specifikációban adott volt, amelyek az ESP modulok (továbbiakban: shield), míg másodlagos célként egy WiFi-vel rendelkező egység illesztése volt, ez a Wemos D1 Mini.

A felhasznált eszközök bemutatása

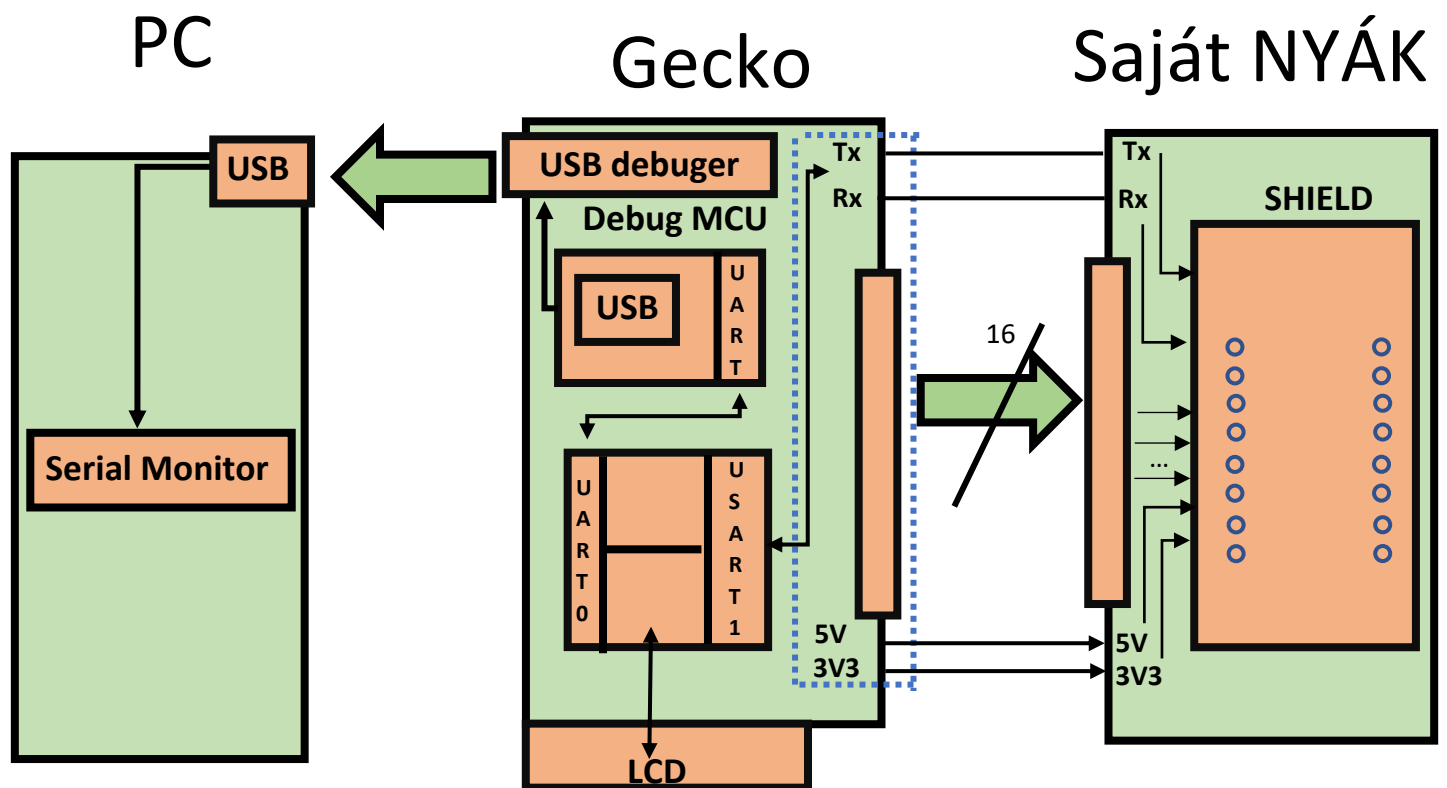
- Giant Gecko STK3700 (EFM32GG – STK3700):
 - Egy általános felépítésű mikrokontroller, számos hasznos perifériával és azok magába hordozott lehetőségeivel, a teljesség igénye nélkül: UART,USART,I2C,... . Pár technikai adat (Cortex M3 mag, 1Mbyte flash, max 48Mhz-es órajel). A feladat elvégzésre egy tökéletes választás volt, mert azon kívül, hogy gyors feldolgozást tesz lehetővé, az LCD kijelzőjén meg is tudtuk jeleníteni a kívánt mérési eredményeket, nem mellesleg már megismerhettük egy a tanszék gondozásába lévő tárgy keretein belül.
- SHT30-as shield:
 - Egy olyan kiegészítő modul (shield), amely hőmérséklet és páratartalom mérése alkalmas szenzort tartalmaz. A mért adatokat azonban nem egy az egyben szolgáltatja, hanem némi korrekciót igényel. (lásd: [1] 13.oldal, 13/20) A kommunikálást I2C alapon keresztül valósítja meg.
- Wemos D1 Mini:
 - Egy ESP8266-on alapuló kártya, amely WiFi képes. Tulajdonképpen egy WiFi-vel ellátott mikrokontroller, persze jóval szerényebb palettával (csak a két mikrokontrollert tekintve), mint a Giant Gecko STK3700-as, de így is számos ötlet megvalósítható vele.
- Továbbá felkerült a NYÁK-ra két darab szenzor is
 - LM75C (részletesebben [\[1\]](#))
 - hőmérséklet szenzor, amely I2C képes.
 - VEML7700 (részletesebben [\[2\]](#))
 - Fényesség mérő szenzor, szintén I2C képes.

Rendszerterv(ek)

A következőkben az általam tervezett NYÁK (nyomtatott áramkör) és az ahhoz kapcsolt egyéb eszközök rendszertervei találhatók. Mivel több fajta shield tetszőlegesen hozzáilleszthető (hardveres és szoftveres figyelmet is igényel) ezért készült azokhoz egy rendszerterv, de a Wemos D1 Minihez is, hogy miért kellett kettő, ez az adott szakaszban kerül megmagyarázásra.

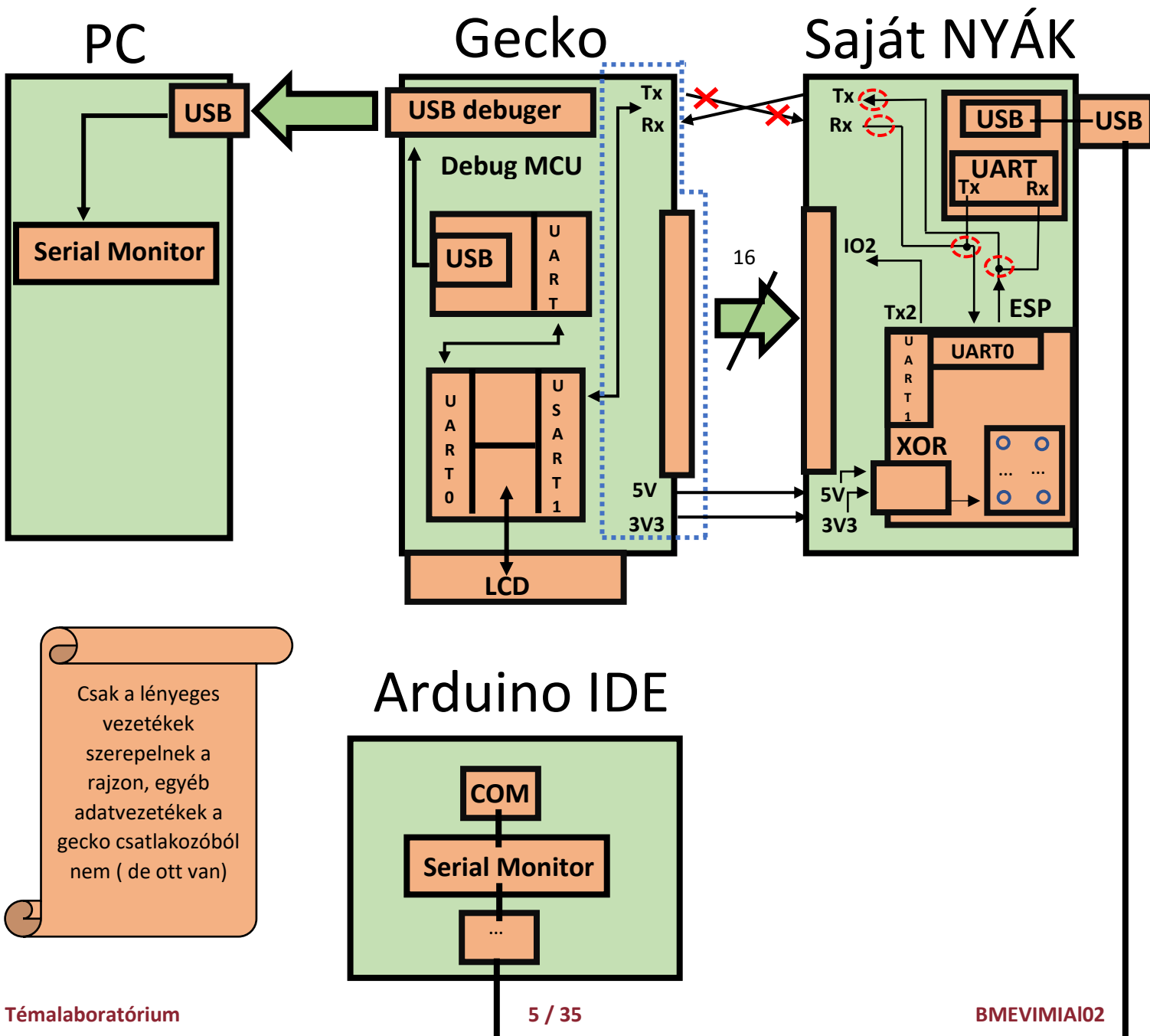
Általános kiegészítő modulhoz (ún. shieldhez)

Az elsődleges cél a shieldek illesztése volt, így ezzel kezdeném a részletezést. Shield használata esetében nem merül fel különösebb probléma az összeköttetésekkel. A kék szaggatott vonallal jelölt részlet egy egységbe tartozik csak fontos látni, hogy mégis ott hogyan mennek a vezetékek. A Tx,Rx lábak egyenesen vannak bekötve (mert a shield-eket kifejezetten a Wemos D1 minihez tervezik. Így a tervezőjük tudatában van annak, hogy a Wemos csatlakozón lévő UART jelek az ESP szemszögéből vannak elnevezve, így - ha lenne olyan shield, ami használná ezen jeleket - akkor ott a shield-en belül oldanák meg a keresztezést. A Gecko-t ilyenkor az egyenes összeköttetéssel tulajdonképpen az ESP modul helyett használjuk, azt emulálja), a két táp pedig aktív egyszerre. Így az 5V-os táp is meg a 3.3V-os táp is, a shield megkapja mindkettőt, de mint ismert ezek 3.3V-ról működnek, de a teljesség kedvéért az 5V-os tápot is elérhetővé tettem.



Specifikusan a Wemos D1 minihez

Másodlagos cél gyanánt a Wemos D1 mini illesztése volt a feladat. Az összeköttetés az elméletben a következőképpen néz ki. A Tx, Rx lábakat keresztbe kell kötni mert a Wemos bővítő csatlakozón az UART jelek az ESP szemszögéből vannak elnevezve (azaz pl. a Wemos csatlakozó TX jelét az ESP állítja elő), és a Gecko bővítő csatlakozón meg a Gecko szemszögéből vannak elnevezve (tehát a Gecko csatlakozó Tx jelét meg a Gecko állítja elő). Alternatív megoldásként felmerülhet az ESP UART1 perifériájának használata, de (csak Tx (Tx2)) azonban az csak küldeni képes így a fogadás nem meg oldott, tehát az ábrán látott összeköttetés a kulcs a rendeltetés szerű használathoz. A tápok bekötése is innen indul ki, ugyanis, ha az ESP-t szeretnénk felprogramozni, ahhoz szükség van egy USB kábelre, de ilyenkor megjelenik az 5V ott, tehát szintén nem köthetjük rá egy időpontban a Geckot, hogy ne tegyük tönkre (hisz akkor ott is megjelenik a feszültség). A maradék két táp (5V (belső) és 3V3 (3.3V)) bekötése is körültekintést igényel ugyanis csak az egyiket köthetjük rá. Ha a Geckora rátesszük az 5V-ot az a belső stabilizátorral előállítja a 3V3-at és ezt továbbítja az ESP-nek, azonban, ha nem ezt a megoldást választjuk, az is egy jó konfiguráció, ha a Gecko 5V-os "lábát" hozzuk összeköttetésbe az ESP 5V-os "lábával" és akkor az ESP hasonlóan a Geckohoz előállítja a saját maga számára szükséges feszültségszintet. A bekötéseket a NYÁK-on felhelyezett jumperekkel oldhatjuk meg, a mellékletben található egy példa ehhez a konfigurációhoz. A kék szaggatott vonallal jelölt részlet egy egységbe tartozik csak fontos látni, hogy mégis ott hogyan mennek a vezetékek.



A kiegészítő áramkör részletes ismertetése

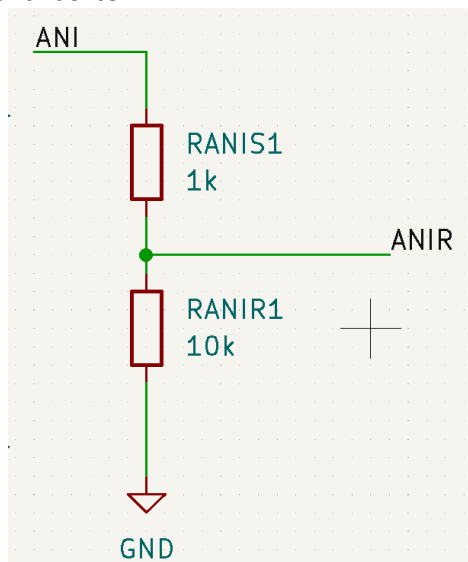
Kapcsolási rajz

A teljes kapcsolási rajz megtalálható a mellékletben. A lényegesebb gondolatok megtalálhatóak az adott illesztési feladathoz tartozó rendszertervénél. A lényegesebb gondolatok a rendszertervekben vannak, itt szeretnék kitérni a kapcsolási rajz egyéb érdekes részleteire. A hidegítő kondenzátorok a megszokott 100 nF-os értékkel rendelkeznek. Védő ellenállások melyek értéke 47 Ω (azért éppen ekkora mert az [3]-ben lévő adatlap (6/21) alapján így tűnt ideálisnak némi utána számolás alapján ($R = \frac{U}{I}$)). I2C kommunikációhoz használt órajel és adat "lábak" (SCL, SDA) felhúzó ellenállásai. Az analóg bementeknél 3V (+- pár század mV), ami az ideális működési tartományuk (van olyan egység, amely nem a szokványos tápfeszültségekről működik, hanem az előbb említett 3V-ról. Így biztosítva lett a teljeskörű kompatibilitás), így a 3.3V-ot le kellett osztani 3-ra (elméletben, hisz nem lesz sose pontos). Ehhez felhasználtam egy 1k Ω -os és egy 10k Ω -os ellenállást, amit a következőképpen számoltam:

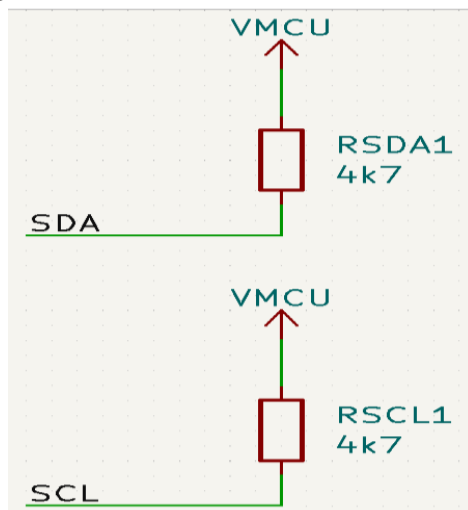
$$U_{analóg} = U_{VMCU} * \frac{\text{vonatkozó ág}}{\text{összes}} = 3.3V * \frac{10k\Omega}{10k\Omega + 1k\Omega} = 3V.$$

Kapcsolási rajzon:

- Ellenállásztó:



- I2C:



NYÁK terv

A teljes NYÁK-tervről készült kép, szintén elérhető a mellékletben. Itt igyekeztem minden beforrasztandó komponenst egy oldalra helyezni, ez sikerült is.

Helyes használat

A rendszerterveknél részletezett megfontolások/konklúziók az irányadóak, az attól eltérő használat nem ajánlott.

A megvalósított feladatok ismertetése

Alapvető kommunikáció

A megvalósított feladat 3 részfeladat tulajdonképpen.

1. Az UART vonalak tesztelése:
 - a. A Gecko UART-on kommunikál az ESP-vel (Wemos D1 Mini), ahol egy darab karakter kerül küldésre, ami az 'a' (ASCII 97), majd ezt a Gecko a PUTTY segítségével megjeleníti.
 - b. Sebesség: 115200 baudrate
2. Az I2C vonalak tesztelése:
 - a. A kártyán lévő VEML7700 felélesztése és az I2C kommunikáció kipróbálása
3. Az I2C vonalak tesztelése:
 - a. A kártyán lévő LM75C felélesztése és az I2C kommunikáció kipróbálása

A mellékletben megtalálható mindegyik kód

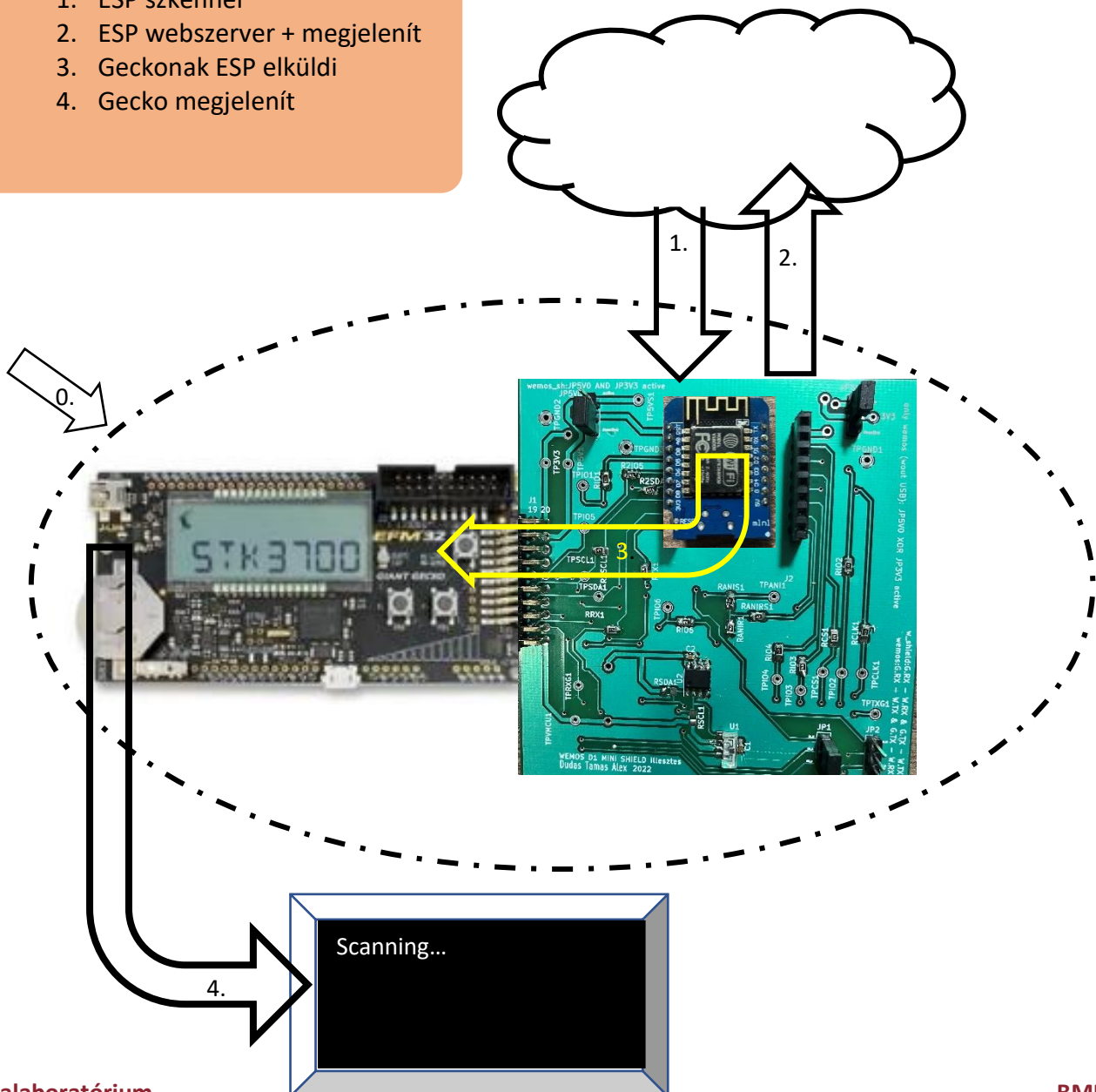
SHT30-as shield illesztése és annak használata

A feladat a Gecko és az SHT30 együttes munkavégzése, amellyel letesztelhetők az I2C vezetékek. Mellékletben megtalálható minden ehhez kapcsolódó kód.

Webszerverbe épített WiFi scanner debuggerrel

A feladat a Gecko és az ESP (részletes dokumentáció bevezető példákkal [4]) együtt dolgozásának elérése. A terv a következő: az ESP-re írni egy WiFi szkennelő programot (a template [5]), ezután az ESP-n meg kell írni a webszervert megvalósító programot. Az ESP-n van egy beépített LED, amelyet inicializálni kell kimenetként, mert ezt használok arra, hogy lássa a programozó, hogy dolgozik-e vajon az ESP. Ha nem szkennel, akkor csak világít, szkennelés esetén pedig villog némi késleltetéssel. Ezután elkészült a webszerver, amellyel a hálózaton keresztül követhetjük a megjelenő adatokat a szkennelés elindítása után. De mivel nem lehet rajta egyszerre a Gecko és az Arduino IDE (alias PC) [a Wemos D1 mini rendszertervéneél tárgyaltak miatt](#), ezért a Geckot használjuk soros portnak. Ábrán a következőképpen néz ki. Mellékletben megtalálható minden kód.

0. Helyes összekötés
1. ESP szkennel
2. ESP webszerver + megjelenít
3. Geckonak ESP elküldi
4. Gecko megjelenít



Összegzés és tanulságok

Összegzés

A feladat során betekintést kaphattam egy teljes fejlesztési folyamatba, amelyet nagyon élveztem. Voltak kisebb, nagyobb megpróbáltatások mind hardveres, mind szoftveres oldalon. Hardveres oldalon például a végtelen mennyiségű gondolkodás és képzelőerő, amely elvezet addig, hogy akkor mi, hogy legyen, akár tervezést nézünk akár mikor meg kell valósítani a huzalozás. Persze végtelen nagy figyelmességgel, hogy minden lehetséges rossz használatra fel legyen készítve a hardver és ne legyen senkinek se bántódása. Szoftveres oldalon se voltak nagyobb problémák, inkább utána nézni mindennek, az adatlapon, hogy melyik eszköz hogyan érhető el például az I2C és a VEML7700-as ALS regisztere, nagyon érdekes és komplex feladatnak gondolok (persze azért a Témalaboratórium határai között). A feladat magába foglalta a hardver tervezését, annak beültetését, bemérését, tesztelését, illetve a megfelelő összeköttetések felügyeletét is (jumperekkel). A megvalósított feladat szoftveres oldalon igen sokszínű, mert tartalmaz beágyazott C kódot, az Arduino környezetben fejlesztett kódok C++ kódok, a webes rész felépítése HTML kiegészítve Bootstrap5-tel. Mivel nem jutott idő Javascript felélesztésére, így HTML kóddal valósult meg a dinamikus táblázat, a háttér animációhoz CSS használtam fel.

Tanulságok

Szerintem egy remek alapot kaptam, hogy ha esetleg ilyen jellegű témához/feladathoz/munkához kerülök, helyt álljak mind hardveres részen, mind szoftveres részen. Itt inkább a beágyazott részre gondolok, mint például egy I2C beüzemelése.

Irodalomjegyzék

[1]:LM75C adatlap:

https://www.ti.com/lit/ds/symlink/lm75b.pdf?ts=1671637244198&ref_url=https%253A%252F%252Fwww.google.com%252F

[2]:VEML7700 adatlap:

<https://www.vishay.com/docs/84286/veml7700.pdf>

[3]:SHT30 adatlap:

https://www.mouser.com/datasheet/2/682/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital-971521.pdf

[4]: ESP dokumentáció (arduino környezetben):

<https://arduino-esp8266.readthedocs.io/en/latest/index.html>

[5]: Arduino beépített wifi scanner példakód:

<https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/scan-examples.html>

Melléklet

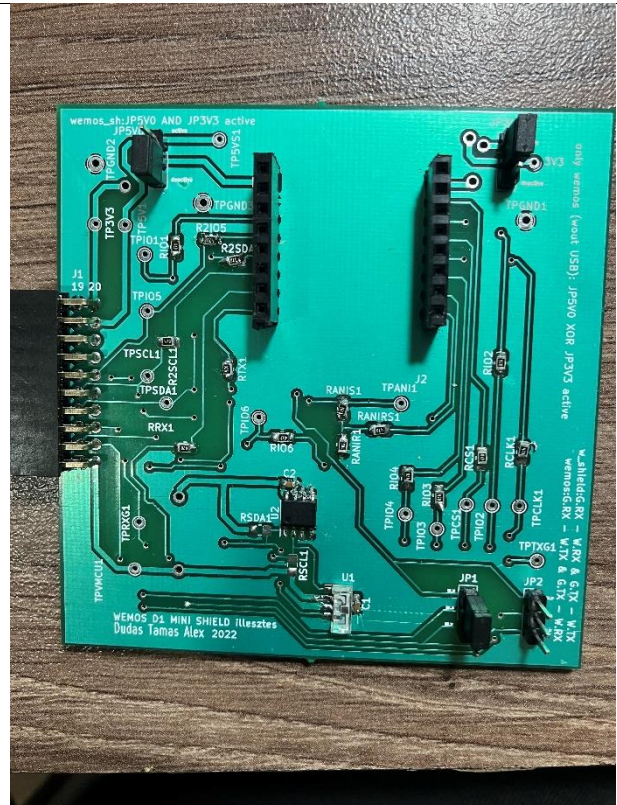
Eszközökről képek:

Eszközök	Kép
----------	-----

Giant Gecko STK3700



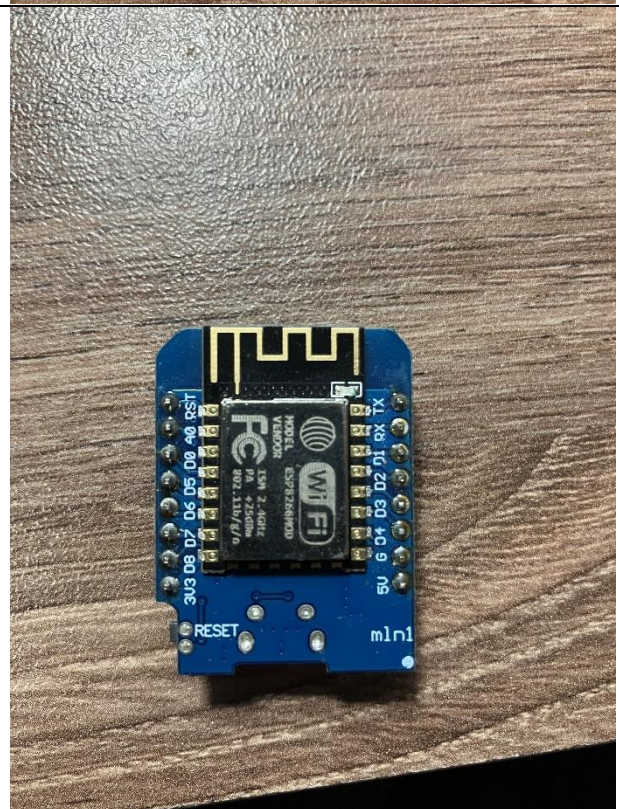
Saját NYÁK



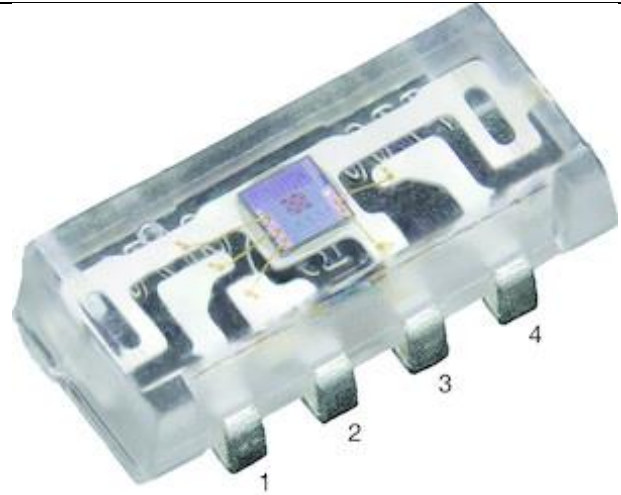
SHT30 shield



Wemos D1 Mini



VEML770 fénymérő

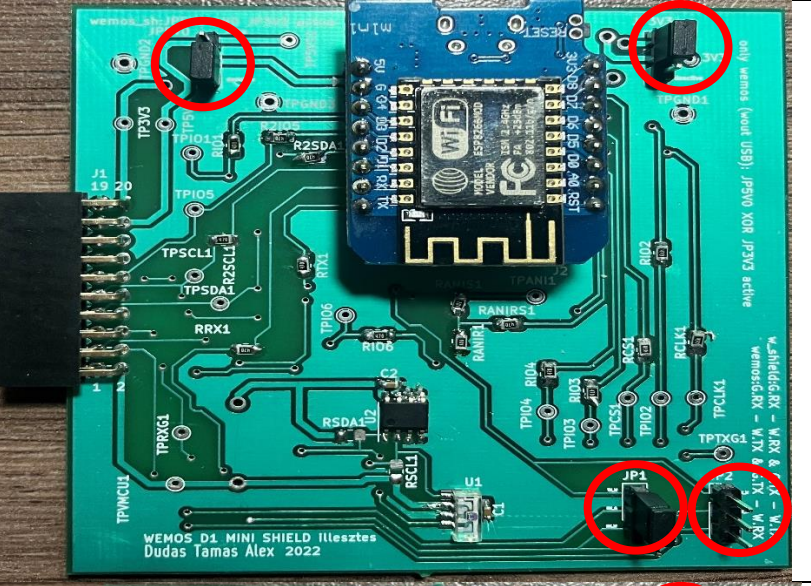
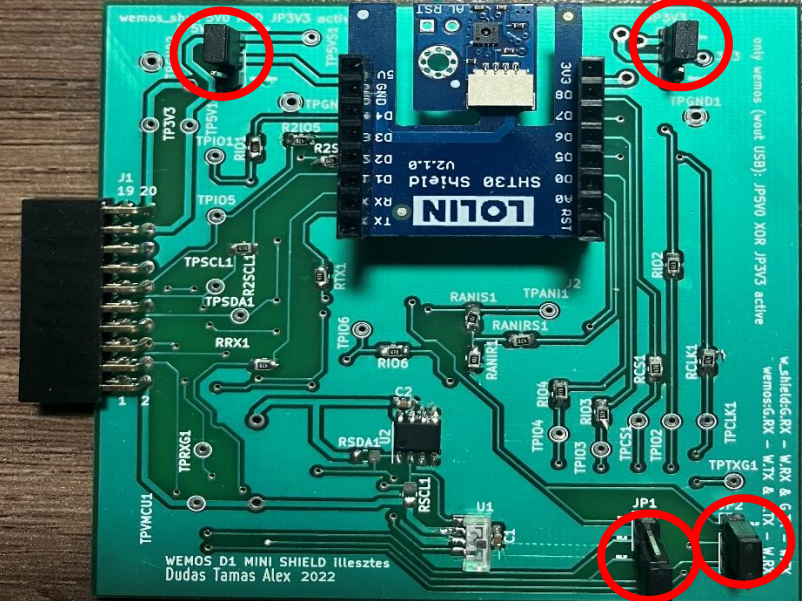


LM75C hőmérséklet mérő

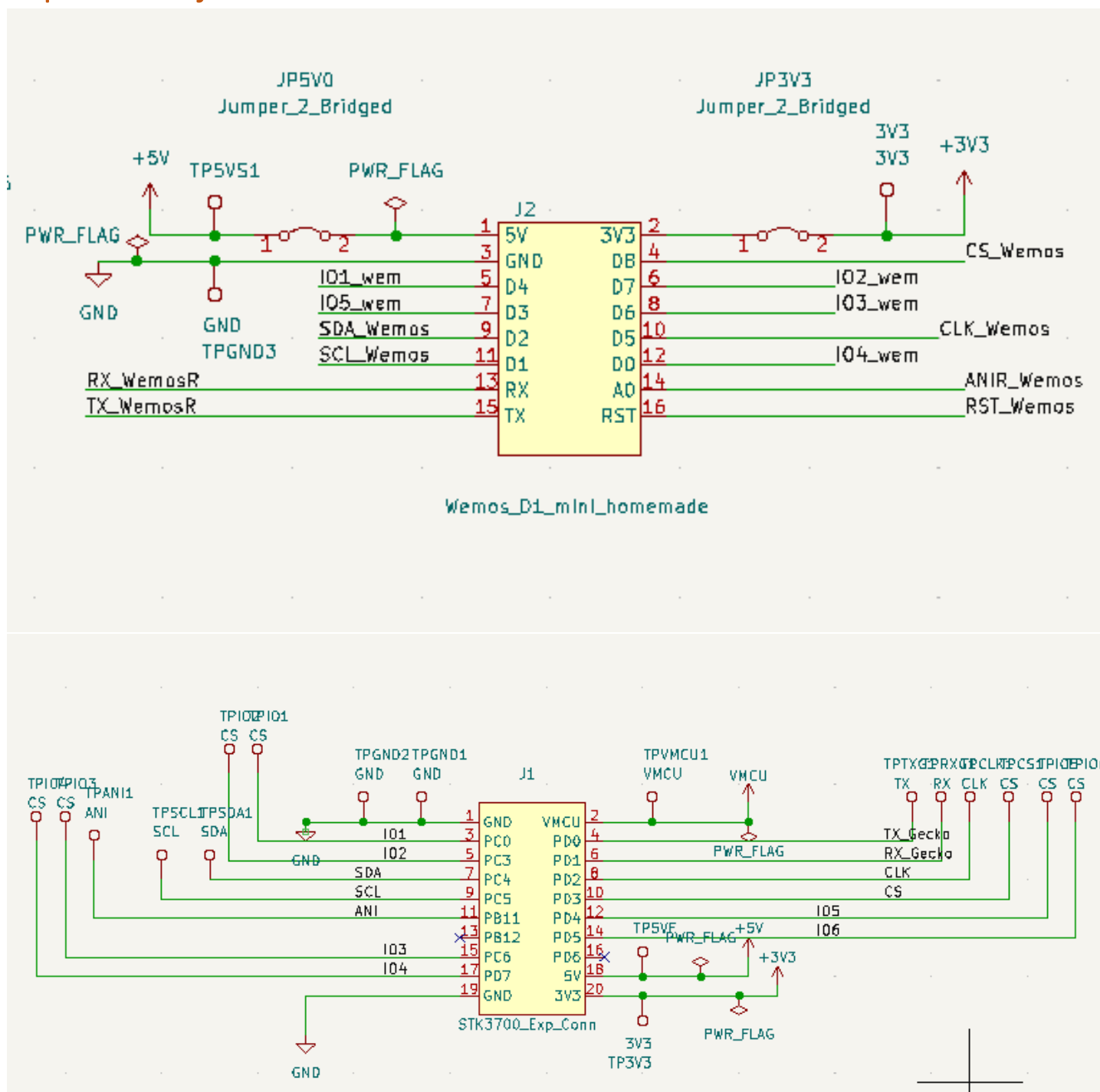


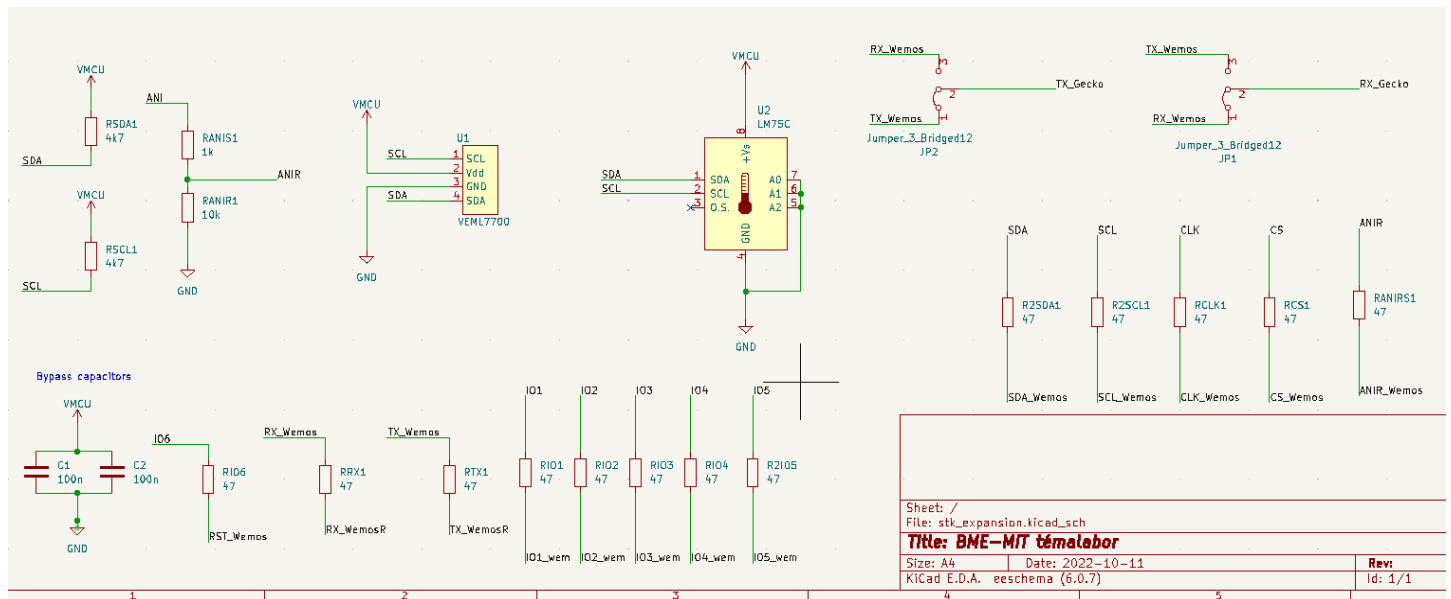
Wemos D1 Mini/Tetszőleges shield helyes csatlakoztatása:

Jelen példában a tetszőleges shield az SHT30.

Csatlakoztatandó eszköz	Kép
Wemos	
Shield	

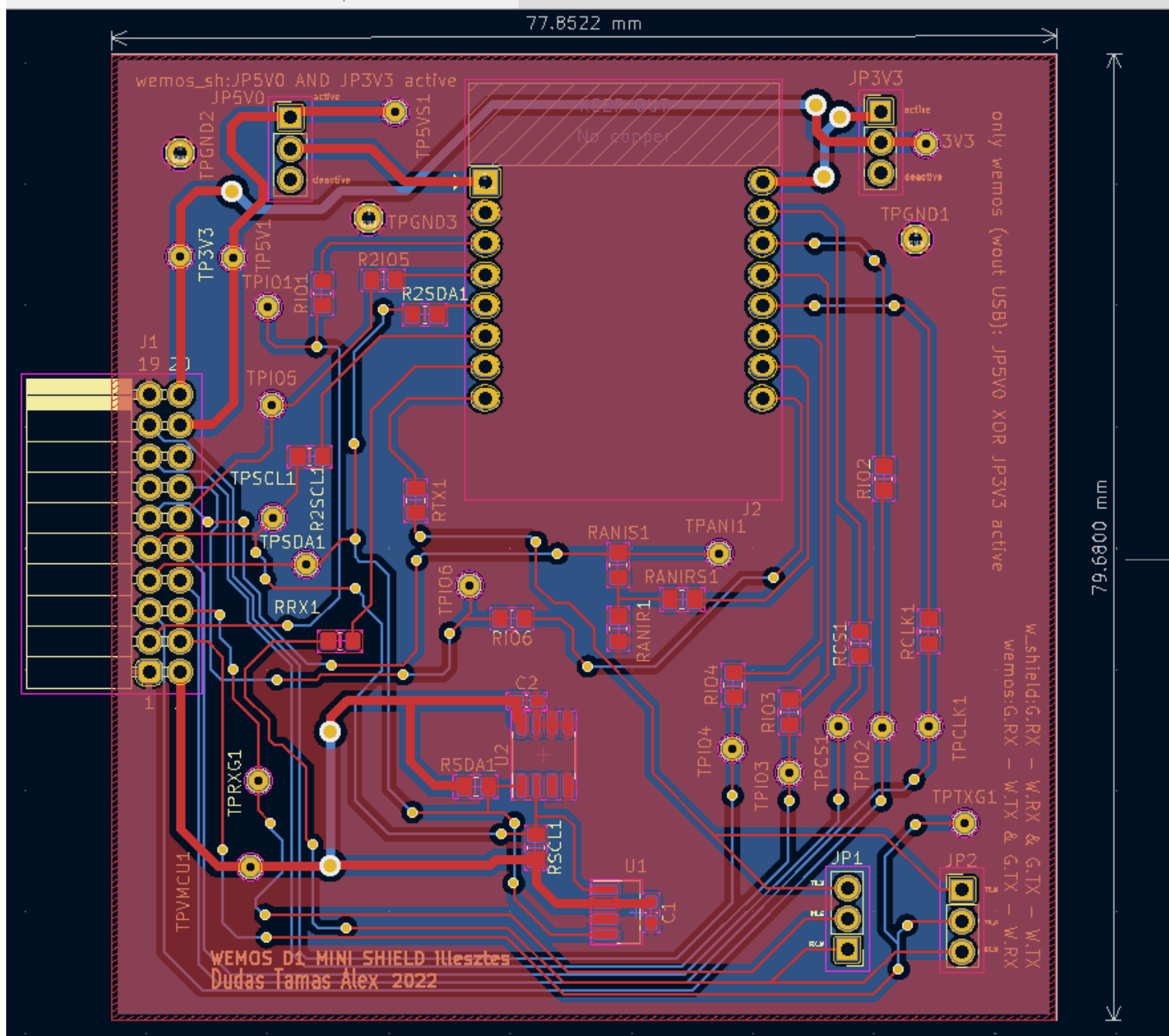
Kapcsolási rajz:





NYÁK:

A felhasznált NYÁK tervező szoftver (+ verzió): KiCad 6.0



Példa kódok:

Gecko – ESP egyszerű kommunikáció

Kód:

Gecko oldala

```
#include "em_device.h"
#include "em_cmu.h"
#include "em_gpio.h"
#include "em_usart.h"

#include "em_chip.h"
uint8_t a = 0;

int main(void)
{
    /* Chip errata */
    CHIP_Init();

    // Enable UART0 through Board Cont.

    // Enable CLK for GPIO
    // Enable HFPERCLK (enabled by def)
    // Enable GPIO CLK branch
    CMU->HFPERCLKEN0 |= CMU_HFPERCLKEN0_GPIO;

    // PF7 high
    // Set PF7 to output (push-pull)
    GPIO->P[5].MODEL |= GPIO_P_MODEL_MODE7_PUSHPULL;
    // Set PF7 high
    GPIO->P[5].DOUTSET = 1 << 7;

    // Configure UART0
    // Enable clock for USART1
    CMU_ClockEnable(cmuClock_USART1, true);
    CMU_ClockEnable(cmuClock_UART0, true);

    // Init UART0 115200 Baud, 8N1 frame format)
    USART_InitAsync_TypeDef usart1_init = USART_INITASYNC_DEFAULT;
    usart1_init.baudrate = 115200;
    usart1_init.refFreq = 0;
    usart1_init.databits = usartDatabits8;
    usart1_init.parity = usartNoParity;
    usart1_init.stopbits = usartStopbits1;
    usart1_init.mvdis = false;
    usart1_init.oversampling = usartOVS16;
    usart1_init.prsRxEnable = false;
    usart1_init.prsRxCh = 0;
    usart1_init.enable = usartEnable;
    USART_InitAsync(USART1, &usart1_init);
```

```
// UART0
USART_InitAsync_TypeDef uart0_init = USART_INITASYNC_DEFAULT;
USART_InitAsync(UART0, &uart0_init);

// Set PD0 (TX) push-pull output
GPIO_PinModeSet(gpioPortD, 0, gpioModePushPull, 1);
// Set PD1 (RX) input
GPIO_PinModeSet(gpioPortD, 1, gpioModeInput, 0);
//
// Set PE0 (TX) push-pull output
GPIO_PinModeSet(gpioPortE, 0, gpioModePushPull, 1);
// Set PE1 (RX) input
GPIO_PinModeSet(gpioPortE, 1, gpioModeInput, 0);
//
GPIO_PinModeSet(gpioPortC, 4, gpioModeDisabled, 0);
GPIO_PinModeSet(gpioPortC, 5, gpioModeDisabled, 0);
// Use Location 1 for UART0
USART1->ROUTE = USART_ROUTE_TXPEN | USART_ROUTE_RXPEN | USART_ROUTE_LOCATION_LOC1 ;
UART0->ROUTE = USART_ROUTE_TXPEN | USART_ROUTE_RXPEN | USART_ROUTE_LOCATION_LOC1 ;

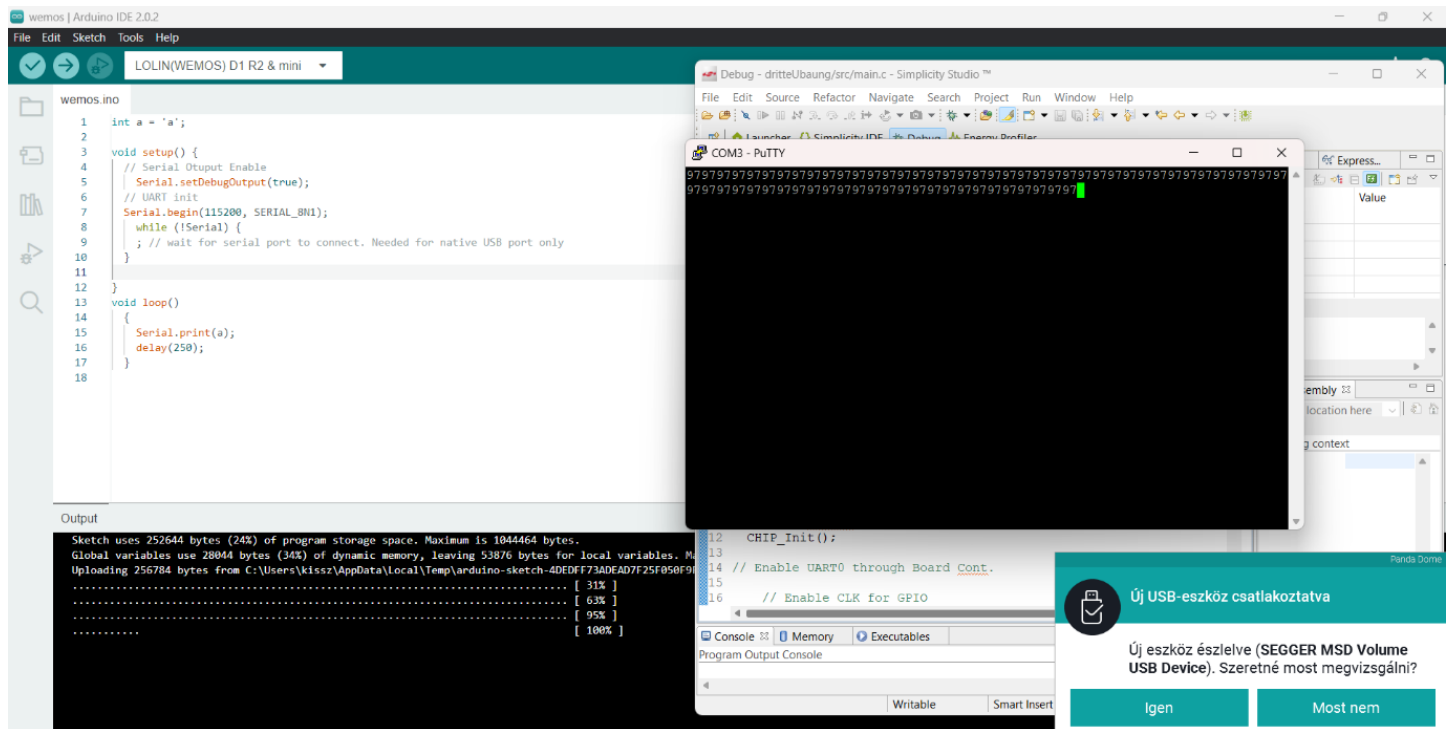
// Send a char
//USART_Tx(USART1, '+');

/* Infinite loop */
while (1) {
    a = USART_Rx(USART1);
    USART_Tx(UART0, a);
}
}
```

ESP oldala

```
int a = 'a';
void setup(){
    // Serial Output Enable
    Serial.setDebugOutput(true);
    // UART init, with 8N1 format
    Serial.begin(115200, SERIAL_8N1);
    // while not found serial
    while(!Serial){;}
}
```

Kép a működésről:



Gecko – LM75C

Kód:

```

#include "em_device.h"
#include "em_chip.h"

#include <segmentlcd.h>
#include <i2cspm.h>
#include <em_cmu.h>
#include <udelay.h>

int main(void)
{
    /* Chip errata */
    CHIP_Init();

    /******
     * Init LCD
     *****/
    SegmentLCD_Init(false);

    /******
     * Init I2CSPM driver
     *****/

    /* Set init structure to:

```

```

    * - Use I2C1 at location #0
    * - PC4 as SDA, PC5 as SCL
    */
I2CSPM_Init_TypeDef init;

init.port          = I2C1;
init.portLocation = 0;

init.sdaPort = gpioPortC;
init.sdaPin  = 4;
init.sclPort = gpioPortC;
init.sclPin  = 5;

init.i2cRefFreq = 0; // Driver will get the clock freq of I2C1
init.i2cMaxFreq = I2C_FREQ_STANDARD_MAX;
init.i2cClhr    = i2cClockHLRStandard;

/*
 * Enable clock for GPIO
 */
CMU_ClockEnable(cmuClock_GPIO, true);

/*
 * Call the init function of the I2CSPM driver
 */
I2CSPM_Init(&init);

/*
 * Power on ALS (Ambient Light Sensor)
 *
 * - After reset all the bits in the Configuration Register are cleared
 *   (set to 0) except bit 0 (ALS Shut Down). If we want to turn on ALS,
 *   we need to clear this bit as well.
 *
 * - A "WRITE" I2C transaction is needed:
 *   - I2C address of VEML7700: 0b_001_0000 (0x10)
 *   - Bytes to send: 3
 *     - Command Code: 0x00 (designates the Configuration Register)
 *     - Lower Data Byte: 0x00 (clear all bits)
 *     - Higher Data Byte: 0x00 (clear all bits)
 */

/*
 * Set transfer sequence structure as needed
 */
I2C_TransferSeq_TypeDef seq;
uint8_t buf0[2];
/*

```

```

seq.addr      = 0x10 << 1; // !!!
seq.flags     = I2C_FLAG_WRITE;
seq.buf[0].len = 3;
seq.buf[0].data = buf0; // &buf0[0];

buf0[0] = 0x00; // Designates the Configuration Register
buf0[1] = 0x00; // Config data (power on) /LSB/
buf0[2] = 0x00; // Config data (power on) /MSB/
*/

/*
 * Call the transfer function of the I2CSPM driver
 */
// I2CSPM_Transfer(I2C1, &seq);

/*
 * Periodically read the ambient light data
 */

/*
 * Set transfer sequence structure as needed
 *
 * This time we need a "WRITE_READ" I2C transaction:
 * - WRITE part:
 * - Bytes to send: 1
 * - Data to send: 0x04 (designates the ALS data register)
 * - READ part:
 * - Bytes to read: 2
 * - Data to read: 16 bit ALS data
 */
uint8_t buf1[2];

seq.addr      = 0x48 << 1; // !!!
seq.flags     = I2C_FLAG_READ;

seq.buf[0].len = 2;
seq.buf[0].data = buf0;
// seq.buf[1].len = 2;
// seq.buf[1].data = buf1;

// buf0[0] = 0x04; // Designates ALS register

/* Infinite loop */
while (1) {
    // Call the transfer function of the I2CSPM driver
    I2CSPM_Transfer(I2C1, &seq);

    // Put ambient light data onto lower display of the LCD

```

```

        SegmentLCD_LowerNumber(buf0[0]);
    }
}

```

Gecko – VEML770

Kód:

```

#include "em_device.h"
#include "em_chip.h"

#include <segmentlcd.h>
#include <i2cspm.h>
#include <em_cmu.h>
#include <udelay.h>

int main(void)
{
    /* Chip errata */
    CHIP_Init();

    /******
     * Init LCD
     *****/
    SegmentLCD_Init(false);

    UDELAY_Calibrate();

    /******
     * Init I2CSPM driver
     *****/

    /* Set init structure to:
     * - Use I2C1 at location #0
     * - PC4 as SDA, PC5 as SCL
     */
    I2CSPM_Init_TypeDef init;

    init.port          = I2C1;
    init.portLocation = 0;

    init.sdaPort = gpioPortC;
    init.sdaPin  = 4;
    init.sclPort = gpioPortC;
    init.sclPin  = 5;
}

```

```

init.i2cRefFreq = 0; // Driver will get the clock freq of I2C1
init.i2cMaxFreq = I2C_FREQ_STANDARD_MAX;
init.i2cClhr      = i2cClockHLRStandard;

/*
 * Enable clock for GPIO
 */
CMU_ClockEnable(cmuClock_GPIO, true);

/*
 * Call the init function of the I2CSPM driver
 */
I2CSPM_Init(&init);

/*
 * Power on ALS (Ambient Light Sensor)
 *
 * - After reset all the bits in the Configuration Register are cleared
 *   (set to 0) except bit 0 (ALS Shut Down). If we want to turn on ALS,
 *   we need to clear this bit as well.
 *
 * - A "WRITE" I2C transaction is needed:
 *   - I2C address of VEML7700: 0b_001_0000 (0x10)
 *   - Bytes to send: 3
 *     - Command Code: 0x00 (designates the Configuration Register)
 *     - Lower Data Byte: 0x00 (clear all bits)
 *     - Higher Data Byte: 0x00 (clear all bits)
 */

/*
 * Set transfer sequence structure as needed
 */
I2C_TransferSeq_TypeDef seq;
uint8_t buf0[3];

seq.addr      = 0x10 << 1; // !!!
seq.flags     = I2C_FLAG_WRITE;
seq.buf[0].len = 3;
seq.buf[0].data = buf0; // &buf0[0];

buf0[0] = 0x00; // Designates the Configuration Register
buf0[1] = 0x00; // Config data (power on) /LSB/
buf0[2] = 0x00; // Config data (power on) /MSB/
/*
 * Call the transfer function of the I2CSPM driver
 */
I2CSPM_Transfer(I2C1, &seq);

```

```

/*
 * Periodically read the ambient light data
 */

/*
 * Set transfer sequence structure as needed
 *
 * This time we need a "WRITE_READ" I2C transaction:
 *   - WRITE part:
 *     - Bytes to send: 1
 *     - Data to send: 0x04 (designates the ALS data register)
 *   - READ part:
 *     - Bytes to read: 2
 *     - Data to read: 16 bit ALS data
 */
uint8_t buf1[2];

seq.addr      = 0x10 << 1; // !!!
seq.flags     = I2C_FLAG_WRITE_READ;

seq.buf[0].len = 1;
seq.buf[0].data = buf0;
seq.buf[1].len = 2;
seq.buf[1].data = buf1;

buf0[0] = 0x04; // Designates ALS register

/* Infinite loop */
while (1) {
    // Call the transfer function of the I2CSPM driver
    I2CSPM_Transfer(I2C1, &seq);

    for (uint16_t i = 0; i < 1000; i++) {
        UDELAY_Delay(1000); // 1 ms
    }

    // Put ambient light data onto lower display of the LCD
    SegmentLCD_LowerNumber(256 * buf1[1] + buf1[0]);
}
}

```

Gecko – ESP – webserver

Gecko oldala (beágyazott)

```

#include "em_device.h"
#include "em_cmu.h"
#include "em_gpio.h"

```



```
#include "em_usart.h"

#include "em_chip.h"
uint8_t a = 0;

int main(void)
{
    /* Chip errata */
    CHIP_Init();

    // Enable UART0 through Board Cont.

    // Enable CLK for GPIO
    // Enable HFPERCLK (enabled by def)
    // Enable GPIO CLK branch
    CMU->HFPERCLKEN0 |= CMU_HFPERCLKEN0_GPIO;

    // PF7 high
    // Set PF7 to output (push-pull)
    GPIO->P[5].MODEL |= GPIO_P_MODEL_MODE7_PUSH_PULL;
    // Set PF7 high
    GPIO->P[5].DOUTSET = 1 << 7;

    // Configure UART0
    // Enable clock for USART1
    CMU_ClockEnable(cmuClock_USART1, true);
    CMU_ClockEnable(cmuClock_UART0, true);

    // Init UART0 115200 Baud, 8N1 frame format)
    USART_InitAsync_TypeDef usart1_init = USART_INITASYNC_DEFAULT;
    usart1_init.baudrate = 115200;
    usart1_init.refFreq = 0;
    usart1_init.databits = usartDatabits8;
    usart1_init.parity = usartNoParity;
    usart1_init.stopbits = usartStopbits1;
    usart1_init.mvdis = false;
    usart1_init.oversampling = usartOVS16;
    usart1_init.prsRxEnable = false;
    usart1_init.prsRxCh = 0;
    usart1_init.enable = usartEnable;
    USART_InitAsync(USART1, &usart1_init);

    // UART0
    USART_InitAsync_TypeDef uart0_init = USART_INITASYNC_DEFAULT;
    USART_InitAsync(UART0, &uart0_init);

    // Set PD0 (TX) push-pull output
    GPIO_PinModeSet(gpioPortD, 0, gpioModePushPull, 1);
    // Set PD1 (RX) input
```

```

GPIO_PinModeSet(gpioPortD, 1, gpioModeInput, 0);
//
// Set PE0 (TX) push-pull output
GPIO_PinModeSet(gpioPortE, 0, gpioModePushPull, 1);
// Set PE1 (RX) input
GPIO_PinModeSet(gpioPortE, 1, gpioModeInput, 0);
//
GPIO_PinModeSet(gpioPortC, 4, gpioModeDisabled, 0);
GPIO_PinModeSet(gpioPortC, 5, gpioModeDisabled, 0);
// Use Location 1 for UART0
USART1->ROUTE = USART_ROUTE_TXPEN | USART_ROUTE_RXPEN | USART_ROUTE_LOCATION_LOC1 ;
UART0->ROUTE = USART_ROUTE_TXPEN | USART_ROUTE_RXPEN | USART_ROUTE_LOCATION_LOC1 ;

// Send a char
//USART_Tx(USART1, '+');

/* Infinite loop */
while (1) {
    a = USART_Rx(USART1);
    USART_Tx(UART0, a);
}
}

```

ESP oldala (beágyazott)

```

// TODO meta datas
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include "index_html.h";
// For visualisation the results on the web
String arraySSID[40]="";
String arrayRSSI[40]="";
// Counting the data and the flag which is stands for controlling the data flow
int scanned = 0;
bool flag = true;

// WiFi config
ESP8266WebServer server(80); // HTTP
const char* ssid = "<SSID>"; // SSID
const char* password = "<PW for the prev SSID>"; // RSSI

void webpageStart(){
    // connect
    server.send(200,"text/html", webpageFormat);
}
void scanning_handle(){
    if(flag){
        Serial.println("scan start");
        digitalWrite(BUILTIN_LED, HIGH); // turn on LED with voltage HIGH
        delay(2000); // wait one second
        digitalWrite(BUILTIN_LED, LOW); // turn off LED with voltage LOW
        delay(2000); // wait one second
    }
}

```

```
// WiFi.scanNetworks will return the number of networks found
int n = WiFi.scanNetworks();
scanned = n;
// Scanning and filling the arrays
Serial.println("scan done");
if (n == 0) {
    Serial.println("no networks found");
} else {
    Serial.print(n);
    Serial.println(" networks found");
    for (int i = 0; i < n; ++i) {
        // Print SSID and RSSI for each network found
        Serial.print(i + 1);
        Serial.print(": ");
        Serial.print(WiFi.SSID(i));
        arraySSID[i] = WiFi.SSID(i);
        Serial.print(" (");
        Serial.print(WiFi.RSSI(i));
        arrayRSSI[i] = WiFi.RSSI(i);
        delay(10);
        Serial.println("");
        if(i == (n-1)){ // If the results have printed
            flag = false;
        }
    }
}
showResult();
}

void showResult(){
    WiFiClient client = server.client();
    client.println("<!DOCTYPE html> <html>");
    client.println("<title>Result</title>");
    client.println("<head><meta charset = \"utf-8\">");
    client.println("<link href=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css\"");
    client.println("rel=\"stylesheet\" integrity=\"sha384-1BmE4kWBq78iYhF1dvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3\"");
    client.println("crossorigin=\"anonymous\">"); // for bootstrap
    client.println("</head>");
    client.println("<body>");
    client.println("<p>");
    client.println("<table class=\"table table-hover\">");
    if(scanned > 0){ // If the ESP doesn't found any network then it doesnt need to show anything
        // table's head
        client.println("<thead>");
        client.println("<tr>");
        client.println("<th scope=\"col\">#</th>");
        client.println("<th scope=\"col\">SSID</th>");
        client.println("<th scope=\"col\">RSSI</th>");
        client.println("</tr>");
        client.println("<tbody>");
        for(int j = 0; j < scanned; ++j){
            client.println("<tr>");
            client.print("<th scope=\"row\">");client.print(j);client.println("</th>");
            client.print("<td>");client.print(arraySSID[j]);client.println("</td>");
            client.print("<td>");client.print(arrayRSSI[j]);client.println("</td>");
            client.println("</tr>");
        }
    }
}
```

```

        client.println("</tbody>");
    }
}
client.println("<a class=\"button button-on\" href=\"/scanre\">AGAIN</a>"); // switches the flag's
states
client.println("</p>");
client.println("</body>");
client.println("<style>.button{ text-align: center;display: block;width: 140px;background-color:
#1abc9c;border: none;color: white;padding: 13px 30px;text-decoration: none;font-size: 25px;margin: 0px
auto 35px;cursor: pointer;border-radius: 4px;}");
    client.println(".button-on{background-color: #1abc9c;}");
    client.println(".button-on:active{background-color: #16a085;}");
    client.println("</style></head></html>");
}
void scanrestart_handle(){
    flag = true; // flag enable
    server.send(200,"text/html",webpageFormat);
}
void setup(){
    // initialize onboard LED as output
    pinMode(BUILTIN_LED, OUTPUT);
    // Adjust baud rate
    Serial.begin(115200);
    // WiFi settings
    // Set WiFi to station mode and disconnect from an AP if it was previously connected
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    WiFi.begin(ssid, password);
    while(WiFi.status() != WL_CONNECTED){delay(500);Serial.print(".");}
    Serial.println();
    Serial.print("IP Address: "); Serial.println(WiFi.localIP());
    // connect
    server.on("/", webpageStart);
    server.on("/scanon",scanning_handle);
    server.on("/scanre",scanrestart_handle);
    // request
    server.begin();
    Serial.println("HTTP server started");
}

void loop(){
    server.handleClient();
}

```

ESP oldala (webes része)

```

const char webpageFormat[] =
R"=====(
<!DOCTYPE html>
<html lang="hu">
    <meta charset="utf-8">
    <title>Témalabor</title>
    <body>
        <div class="header">
            <div class="inner-header flex">

```

```

        <h1 style="font-family:Arial, Helvetica, sans-serif;"> <strong>WiFi
Scanner</h1> </strong>
    </div>
    <a class="button button-on" href="/scanon">G0</a>
        <div>
            <svg class="waves" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
                viewBox="0 24 150 28" preserveAspectRatio="none" shape-
rendering="auto">
                <defs>
                    <path id="gentle-wave" d="M-160 44c30 0 58-18 88-18s 58 18
88 18 58-18 88-18 58 18 88 18 v44h-352z" />
                </defs>
                <g class="parallax">
                    <use xlink:href="#gentle-wave" x="48" y="0"
fill="rgba(255,255,255,0.7)" />
                    <use xlink:href="#gentle-wave" x="48" y="3"
fill="rgba(255,255,255,0.5)" />
                    <use xlink:href="#gentle-wave" x="48" y="5"
fill="rgba(255,255,255,0.3)" />
                    <use xlink:href="#gentle-wave" x="48" y="7"
fill="#fff" />
                </g>
            </svg>
        </div>
    </div>
</body>
</html>
<!--Formatting waves ref: Dudás Tamás Alex Informatika 2 Webshop HF -->
<style>
    p
    {
        font-size: 20px;
        margin: auto;
    }
    .header
    {
        position:relative;
        text-align:center;
        background: linear-gradient(60deg, rgba(84,58,183,1) 0%, rgba(0,172,193,1) 100%);
        color:white;
    }
    .logo
    {
        width:50px;
        fill:white;

```

```
padding-right:15px;
display:inline-block;
vertical-align: middle;
}
.inner-header
{
height:65vh;
width:100%;
margin: 0;
padding: 0;
}
.flex
{ /*Flexbox for containers*/
display: flex;
justify-content: center;
align-items: center;
text-align: center;
}
.waves
{
position:relative;
width: 100%;
height:15vh;
margin-bottom:-7px; /*Fix for safari gap*/
min-height:100px;
max-height:150px;
}
.content
{
position:relative;
height:20vh;
text-align:center;
background-color: white;
}
/* Animation */
.parallax > use
{
animation: move-forever 25s cubic-bezier(.55,.5,.45,.5) infinite;
}
.parallax > use:nth-child(1)
{
animation-delay: -2s;
animation-duration: 7s;
}
.parallax > use:nth-child(2)
{
animation-delay: -3s;
animation-duration: 10s;
```

```
}  
.parallax > use:nth-child(3)  
{  
  animation-delay: -4s;  
  animation-duration: 13s;  
}  
.parallax > use:nth-child(4)  
{  
  animation-delay: -5s;  
  animation-duration: 20s;  
}  
@keyframes move-forever  
{  
  0% {  
    transform: translate3d(-90px,0,0);  
  }  
  100% {  
    transform: translate3d(85px,0,0);  
  }  
}  
/*Shrinking for mobile*/  
@media (max-width: 768px)  
{  
  .waves  
  {  
    height:40px;  
    min-height:40px;  
  }  
  .content  
  {  
    height:30vh;  
  }  
  h1 {  
    font-size:24px;  
  }  
}  
.alert  
{  
padding: 20px;  
background-color: #f44336;  
color: white;  
}  
  
.closebtn  
{  
margin-left: 15px;  
color: white;  
font-weight: bold;
```

```
float: right;
font-size: 22px;
line-height: 20px;
cursor: pointer;
transition: 0.3s;
}

.closebtn:hover
{
color: black;
}
footer
{
text-align: left;
font-size: 15px;
font-family: 'Times New Roman', Times, serif;
}
* {
box-sizing: border-box;
}
form.search-bar input[type=text] {
padding: 10px;
font-size: 17px;
border: 1px solid grey;
float: left;
width: 80%;
background: #f1f1f1;
}

form.search-bar button {
float: left;
width: 20%;
padding: 10px;
background: #2196F3;
color: white;
font-size: 17px;
border: 1px solid grey;
border-left: none;
cursor: pointer;
}
</style>
<!--Formatting buttons-->
<style>
.button{
text-align: center;
display: block;
width: 140px;
background-color: #1abc9c;
```



```
border: none;
color: white;
padding: 13px 30px;
text-decoration: none;
font-size: 25px;
margin: 0px auto 35px;
cursor: pointer;
border-radius: 4px;
text-align: center;
}
.button-on{
    background-color: #1abc9c;
}
.button-on:active{
    background-color: #16a085;
}
.button-off{
    background-color: #34495e;
}
.button-off:active{
    background-color: #2c3e50;
}

</style>
<!--Formatting paragraphs-->
)====";
```

Gecko – Shield (SHT30)

Kód:

```
#if 1
#include "em_device.h"
#include "em_chip.h"

#include <segmentlcd.h>
#include <i2cspm.h>
#include <em_cmu.h>

float cTemp;
float fTemp;
float humidity;

int main(void)
{
    /* Chip errata */
    CHIP_Init();

    /******
    * Init LCD
    *****/
    SegmentLCD_Init(false);
    /******
    * Init I2CSPM driver
```

```

*****/
/* Set init structure to:
 * - Use I2C1 at location #0
 * - PC4 as SDA, PC5 as SCL
 */
I2CSPM_Init_TypeDef  init;

init.port             = I2C1;
init.portLocation     = 0;

init.sdaPort          = gpioPortC;
init.sdaPin           = 4;
init.sclPort          = gpioPortC;
init.sclPin           = 5;

init.i2cRefFreq       = 0; // Driver will get the clock freq of I2C1
init.i2cMaxFreq       = I2C_FREQ_FASTPLUS_MAX;
init.i2cClhr          = i2cClockHLRStandard;

/*
 * Enable clock for GPIO
 */
CMU_ClockEnable(cmuClock_GPIO, true);

/*
 * Call the init function of the I2CSPM driver
 */
I2CSPM_Init(&init);

/*
 * SHT30 Shield
 * - A "WRITE" I2C transaction is needed:
 * - I2C address of SHT30: 0x45, 8'b 0100_0101
 * - Bytes to send: 2
 * - Command Code:      0x2C (Clock stretching enable)
 * - Second byte:       0x06
 */
/*
 * Set transfer sequence structure as needed
 */
I2C_TransferSeq_TypeDef  seq;
uint8_t                  buf0[2];

seq.addr                 = 0x45 << 1;
seq.flags                 = I2C_FLAG_WRITE;
seq.buf[0].len           = 2;
seq.buf[0].data           = buf0; // &buf0[0];

buf0[0] = 0x2C; // measurement command
buf0[1] = 0x06; // measurement command
/*
 * Call the transfer function of the I2CSPM driver
 */
I2CSPM_Transfer(I2C1, &seq);

/*
 * Periodically read the ambient light data
 */
/*
 * Set transfer sequence structure as needed

```

```
*
* This time we need a "READ" I2C transaction:
*   - READ part:
*     - Bytes to read: 6
*     - 1st and 2nd are the Temp bytes, then the
*     - 3rd is Checksum
*     - 4th and 5th are relative humidity
*     - 6th is again Checksum
*/
uint8_t  buf1[6];

seq.addr      = 0x45 << 1;
seq.flags     = I2C_FLAG_READ;

seq.buf[0].len = 6;
seq.buf[0].data = buf1;

/* Infinite loop */
while (1) {
    // Call the transfer function of the I2CSPM driver
    I2CSPM_Transfer(I2C1, &seq);
    // convert data
        float Temp = 256 * buf1[0] + buf1[1];
        // float hTemp = 256 * buf1[3] + buf1[4];
        cTemp = -45 + 175 * ( Temp / (65535.0) ) ;
        // fTemp = -49 + 347 * ( Temp / ((double)_2_POWER_16 - 1) )
;

        // humidity = hTemp * 100 / ((double)_2_POWER_16 - 1);
    // Put ambient light data into lower display of the LCD
        SegmentLCD_LowerNumber(cTemp);
}
}
#endif
```