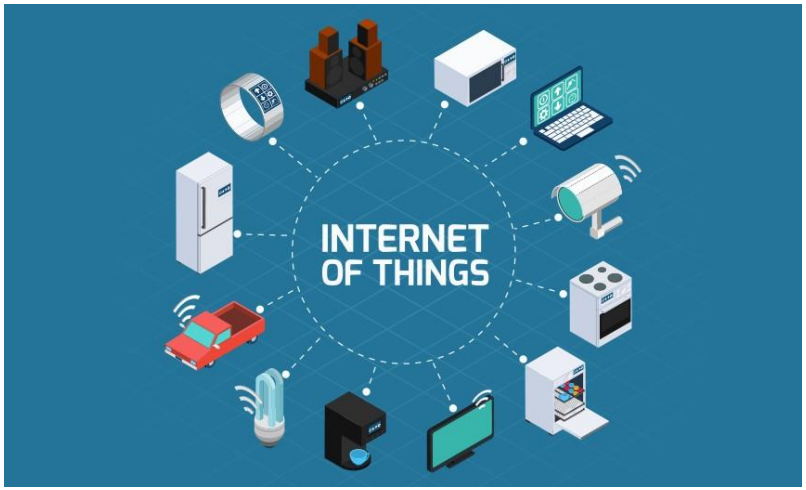
IoT eszköz alapú mérőrendszer

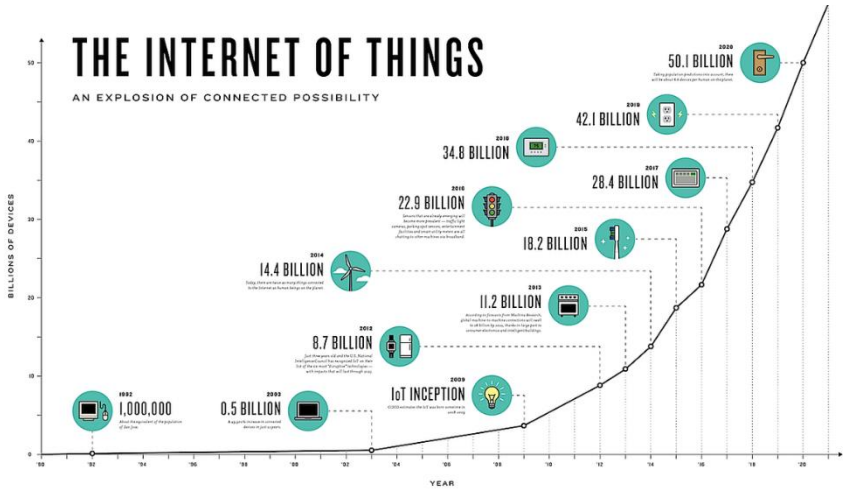
Gyulai Péter
Konzulens: Krébesz Tamás István

Méréstechnika és
Információs Rendszerek
Tanszék

Internet of Things (IoT)

*„IoT is simply the **network** of interconnected things/devices which are **embedded** with sensors, software, network connectivity and necessary electronics that enables them to collect and exchange **data** making them **responsive**."*

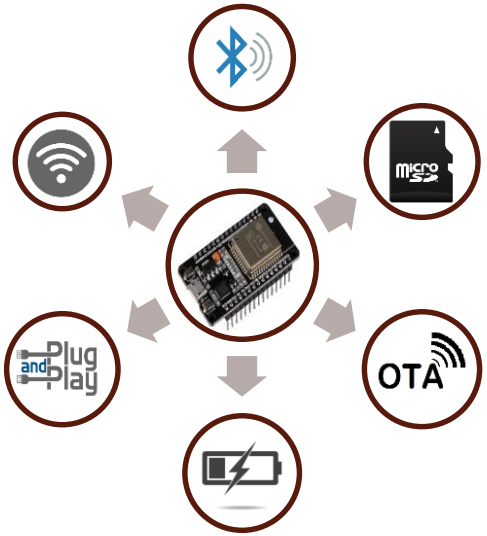




A teljes elképzelés

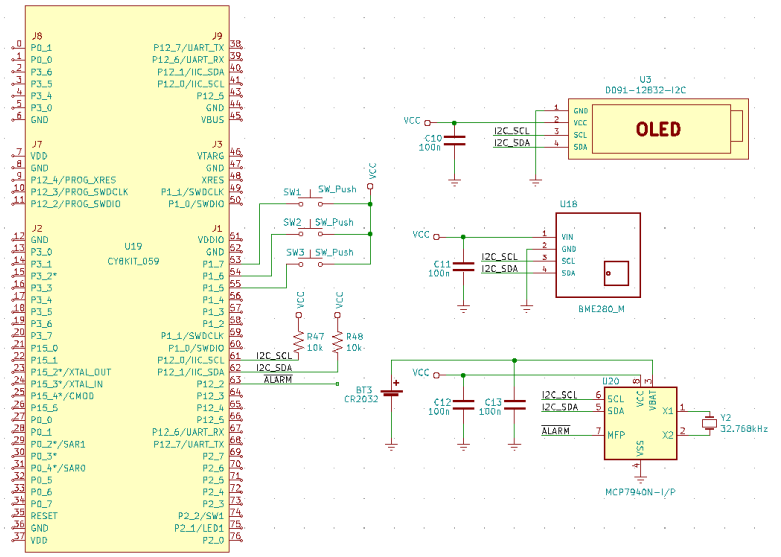
Az MSc Diplomaterv végére:

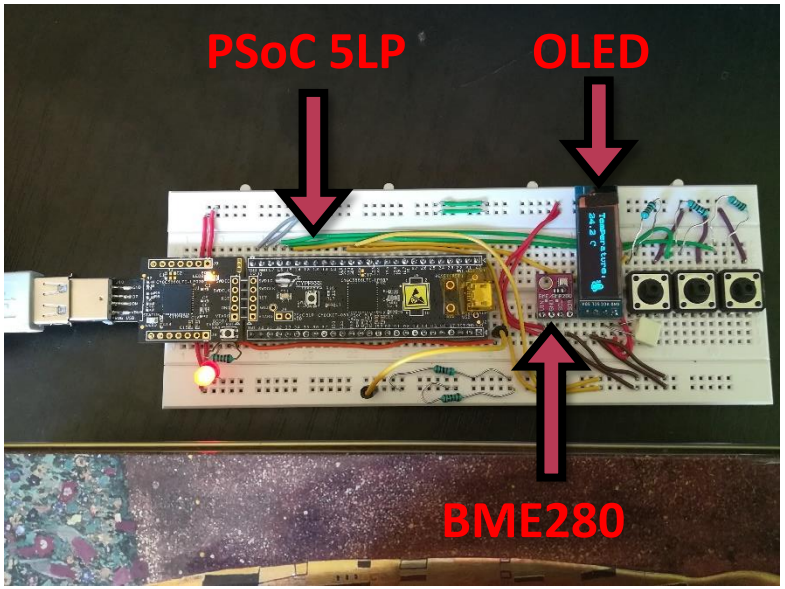
- ESP32 alapú
- Moduláris felépítésű
- Plug&Play bővíthetőség
- Wifi & Bluetooth LE
- Over-the-Air updates
- Helyi adatmentés (µ-SD)
- Önálló tápellátás (Li-ion)



Feladatok a félévben

- OLED kijelző illesztése
- BME280 szenzor illesztése
- Realtime-clock (RTC) illesztése
- UNIX-szerű terminál UART-on
- Breadboard prototípus megépítése
- Magasszintű szoftver API előkészítése
- Tápfeszültség ellátás megtervezése (a későbbi feladatokhoz)
- Felhasznált mikrokontroller: Cypress PSoC 5LP





Alapelvek a fejlesztés során

- Moduláris szoftver
- Újrafelhasználható
- Cross-platform
- Egyszerűen cserélhető szoftvermodulok
- Hierarchikus felépítés
- Open-source (Github)
- Dokumentáció (Doxygen)

Functions and datastructures. More...

```
#include <stdint.h>
#include "SSD1306_Defs.h"
```

Go to the source code of this file.

Data Structures

struct **SSD1306_I2CControllerStruct**
Structure for passing the I2C bus controller functions, that are used by the SSD1306 driver. **More...**

Typedefs

typedef struct **SSD1306_I2CControllerStruct** **SSD1306_I2CControllerStruct**
Structure for passing the I2C bus controller functions, that are used by the SSD1306 driver. **More...**

Functions

void **SSD1306_SetI2CController** (**SSD1306_I2CControllerStruct** controller)
Installs the I2C bus controller structure holding function pointers to the bus control methods. **More...**

uint8_t * **SSD1306_GetDisplayBuffer** (void)
Returns the display shadow buffer. **More...**

void **SSD1306_Init** ()
Initializes the display with default values provided in **SSD1306_Defs.h**, and clears the display shadow buffer.

void **SSD1306_Reset** ()
Resets the device settings and clears the shadow buffer & physical display.

void **SSD1306_WriteCommand** (uint8_t command)
Sends a single command byte to the device. **More...**

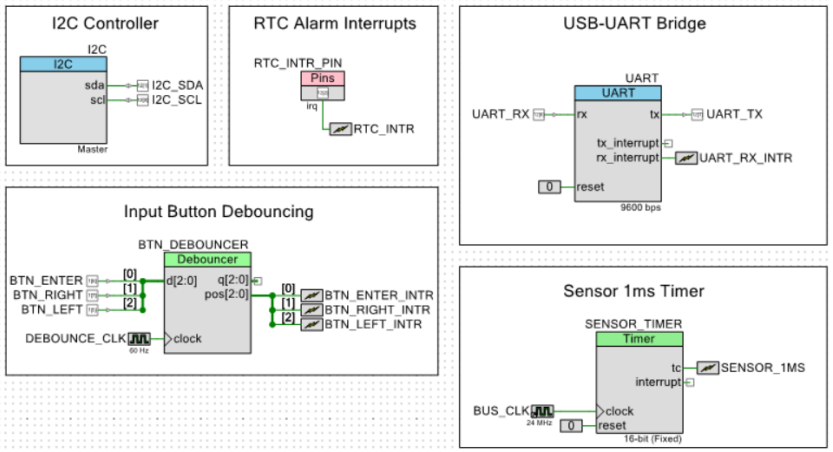
void **SSD1306_WriteCommands** (uint8_t commands[], uint8_t count)
Writes multiple commands to the device. **More...**

void **SSD1306_WriteData** (uint8_t data[], uint16_t count)
Writes the specified stream of data to the device. **More...**

void **SSD1306_WritePixel** (uint8_t x, uint8_t y, uint8_t color)
Writes the specified color to the pixel at [x,y] display coordinates. **More...**

void **SSD1306_ClearDisplay** (void)
Clears the content of the display shadow buffer, by filling it with zeros. **More...**

void **SSD1306_RefreshDisplay** (void)
Writes the content of the display shadow buffer to the device.



https://github.com/Peterskhan/Rhea_software
https://github.com/Peterskhan/Rhea

Az RTC modulok

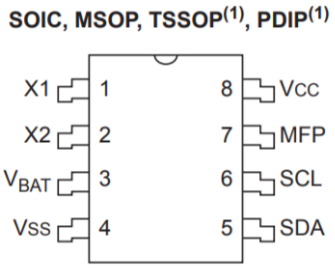
- Integrált áramkör: MCP7940N

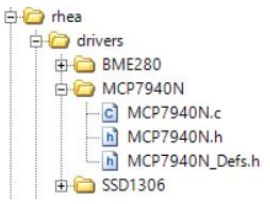
Alacsonyszintű illesztőprogram:

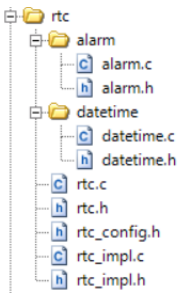
- Önmagában is felhasználható
- Regiszterek másolata RAM-ban tárolva
- I2C hook-up

Magasszintű API:

- Absztrakciók, az alacsonyszintű megvalósítás könnyen cserélhető
- Dátum és idő tárolása
- Power-up & Power-down
- Riasztások








```
/** @brief Returns the current date and time. */
rhea_rtc_datetime rhea_rtc_Now(void);

/**
 * @brief Sets the current date and time.
 * @param now The new date and time.
 */
void rhea_rtc_SetDatetime(const rhea_rtc_datetime now);

/** @brief Returns the date and time of the last power-up. */
rhea_rtc_datetime rhea_rtc_PowerUp(void);

/** @brief Returns the date and time of the last power-fail. */
rhea_rtc_datetime rhea_rtc_PowerFail(void);
```

```
/**
 * @brief Constructs and adds a new alarm to the alarm registry.
 * @param datetime A datetime structure filled with the necessary alarm parameters (based on mode).
 * @param recurring Non-zero if the alarm is recurring, zero otherwise.
 * @param mode The alarm mode (one of the #rhea_rtc_alarmMode values).
 * @param persistent Non-zero if the alarm should be persistent (with SRAM back-up), zero otherwise.
 * @param callback A #rhea_rtc_alarmCallback pointer, or NULL if not used.
 * @returns The ID of the created alarm, or -1 if the alarm could not be created.
 */
int16_t rhea_rtc_AddAlarm(rhea_rtc_datetime datetime, uint8_t recurring, uint8_t mode, uint8_t persistent,
                          rhea_rtc_alarmCallback callback);

/**
 * @brief Deletes an alarm from the alarm registry.
 * @param The ID of the alarm to be deleted.
 */
void rhea_rtc_DeleteAlarm(uint16_t id);

/**
 * @brief Handles all alarms by invoking callbacks and deleting
 *        non-recurring alarms.
 */
void rhea_rtc_HandleAlarms(void);
```

A grafikus modulok

- Integrált áramkör: SSD1306

Alacsonyszintű illesztőprogram:

- Önmagában is felhasználható
- Csak pixelek kirajzolása
- Pixel buffer RAM-ban

Magasszintű API:

- Absztrakciók, az alacsonyszintű megvalósítás könnyen cserélhető
- Grafikus primitívek rajzolása
- Szöveges és szimbólumkijelzés
- Saját pixelkódolás (JSON)
- 1 bit/pixel (C-kód generálás)

A szenzor modulok

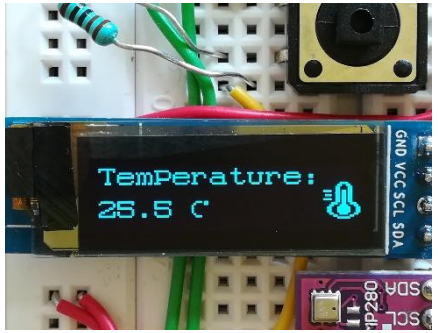
- Illesztett szenzor: BME280

Alacsonyszintű illesztőprogram:

- Önmagában is felhasználható
- Egységes stílus a többi illesztőprogrammal

Magasszintű API:

- Absztrakciók, bármilyen szenzor egyszerű illesztése
- Egységes interface tetszőleges szenzor kezeléséhez
- Measurements, Attributes
- Automatikus periodikus mérések

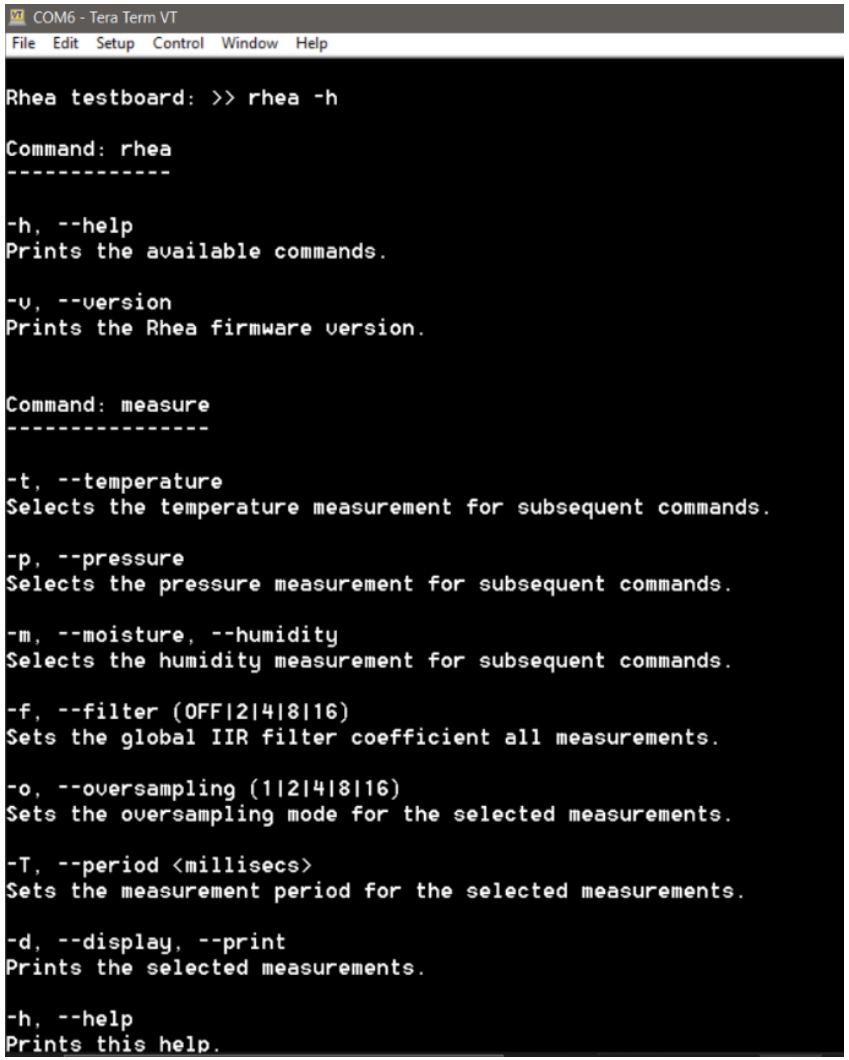



```
/**
 * @brief Registers a new measurement to the measurement registry.
 * @param [in] device The name of the sensor device performing the measurement.
 * @param [in] parameter The name of the parameter being measured by the sensor.
 * @param [in] period The period of the measurement in milliseconds.
 * @param [in] function The function performing the measurement.
 */
void rhea_sensor_AddMeasurement(const char *device,
                                const char *parameter,
                                uint64_t    period,
                                rhea_sensor_measurementFunction function);


/**
 * @brief Registers an attribute to the specified device.
 * @param [in] device The name of the sensor device having the attribute.
 * @param [in] attribute The name of the attribute.
 * @param [in] getter Optional getter for the attribute if it can change on it's own.
 * @param [in] setter Optional setter for the attribute if it is an external value.
 */
void rhea_sensor_AddAttribute(const char *device,
                              const char *attribute,
                              rhea_sensor_attributeGetter getter,
                              rhea_sensor_attributeSetter setter);


/**
 * @brief Returns the latest measured value of the specified measurement.
 * @param [in] device The name of the sensor device performing the measurement.
 * @param [in] parameter The name of the parameter being measured by the sensor.
 * @returns The latest measured value of the specified measurement.
 */
RHEA_SENSOR_MEASUREMENT_TYPE rhea_sensor_GetMeasurement(const char *device,
                                                        const char *parameter);
```

Az terminál modul

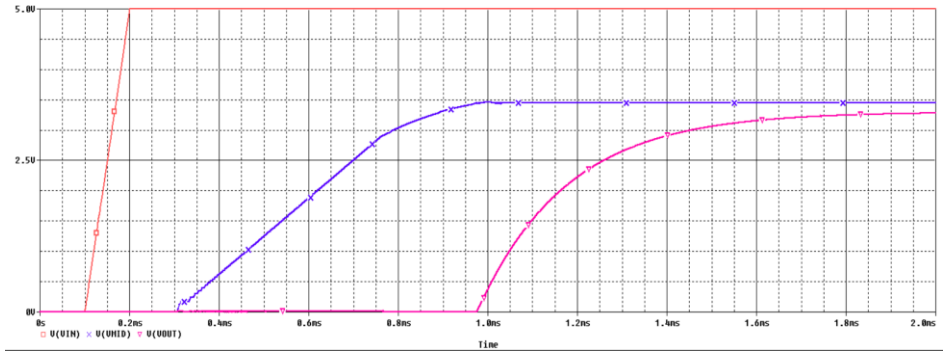
- UART segítségével megvalósított
- USB-UART átalakító a mikrokontrolleren
- Unix-szerű terminál
- Emulátorral PC-ről kezelhető
- Mérések kiolvasása
- Paraméterek beállítása

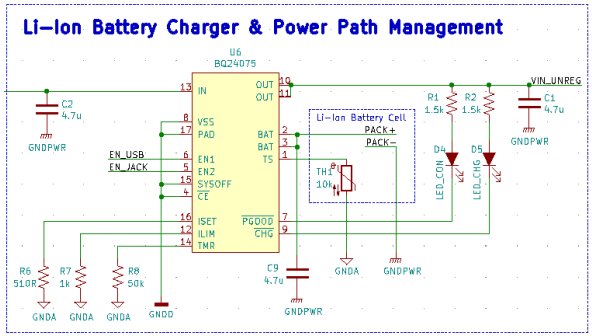
```
Rhea testboard: >> measure -tpm --display
Temperature: 24 Celsius
Pressure: 98774 Pa
Humidity: 33 %

Rhea testboard: >> measure -t --oversampling 9
Wrong oversampling value.
```

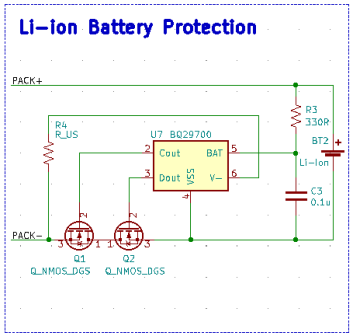


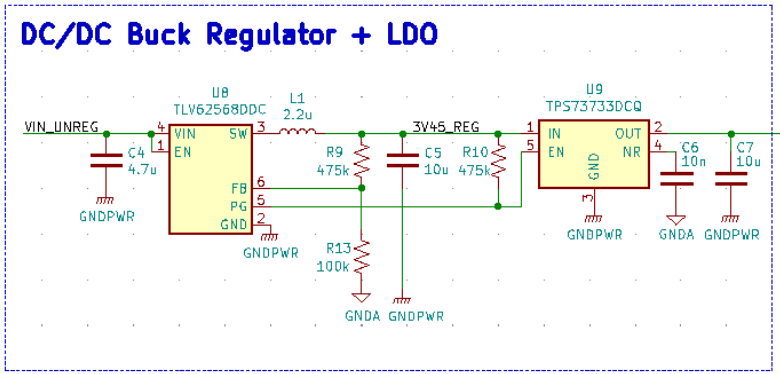
A tápfeszültség előállítása

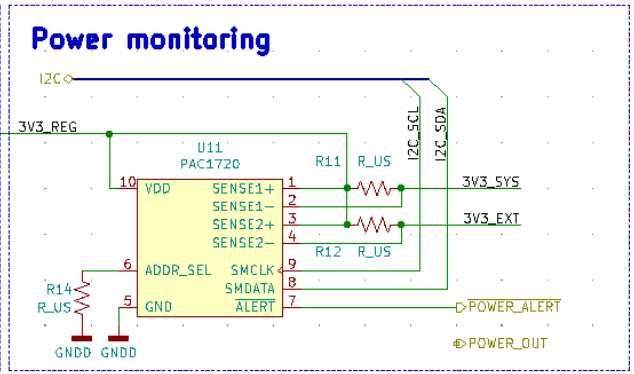
- Előkészület a PCB-hez
- Li-ion akkumulátor töltése
- USB/Barrel jack csatlakozás
- Akkumulátor-védelem
- 2 szintű feszültségszabályozás
- Energia menedzsment
- Veszteség és zajszimulációk

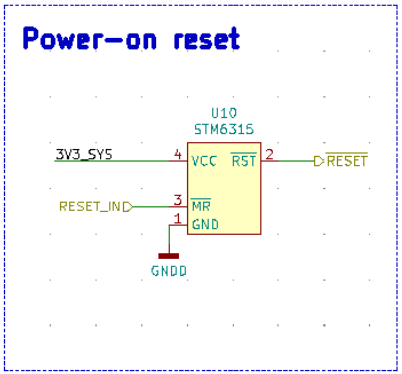












Köszönöm a figyelmet!

- Videók
- Kérdések?