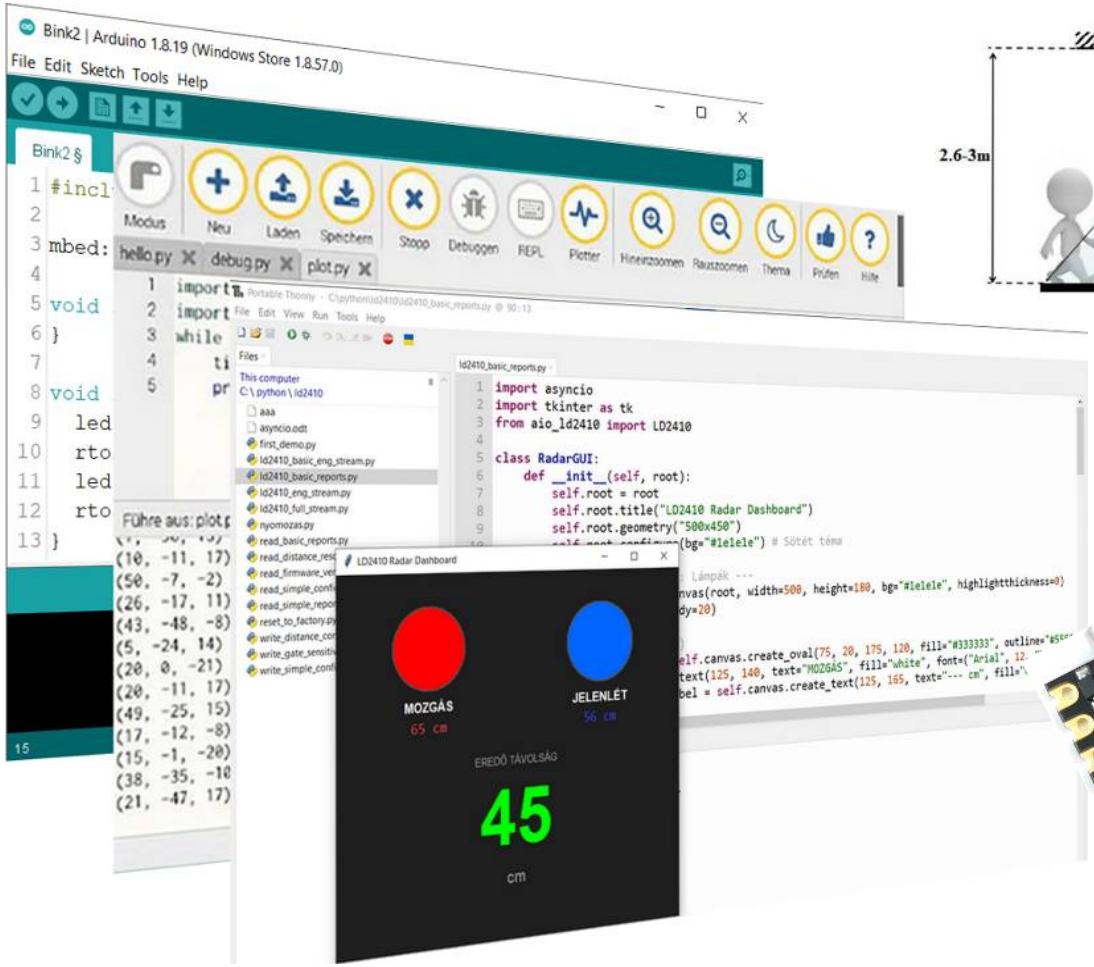
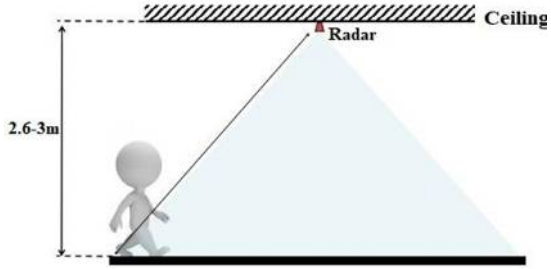
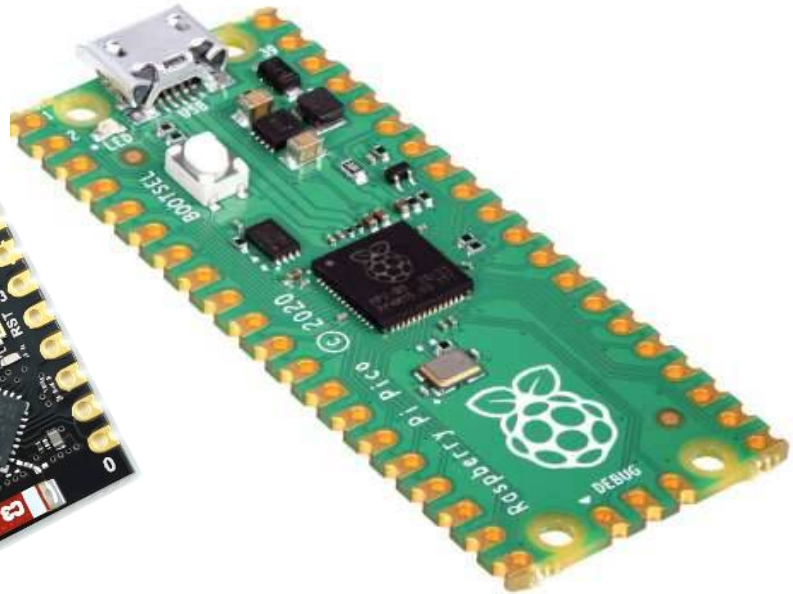
Emberi jelenlét-érzékelés mmWave radarral

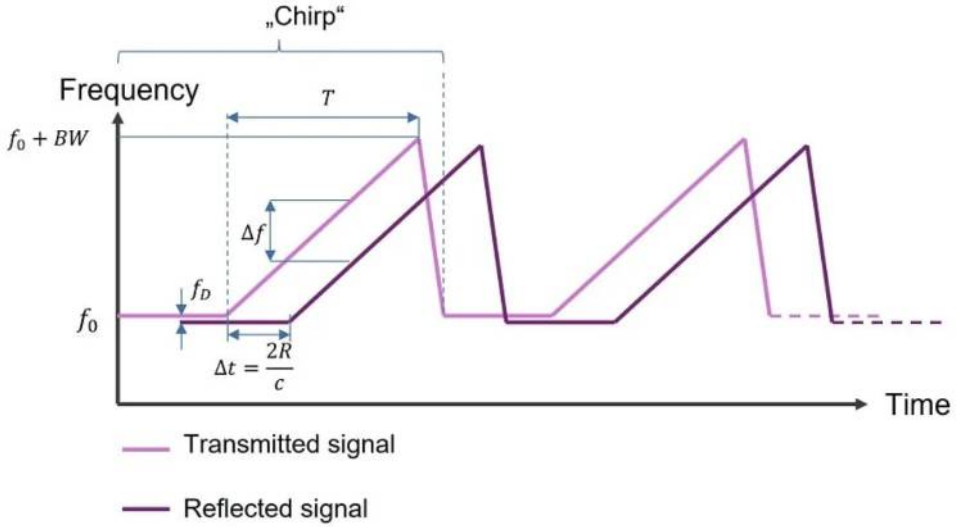






FMCW Radar

- Az autóipari radarrendszerek Frequency Modulated Continuous Wave (FMCW) segítségével működnek. A rendszer folyamatos hullámot sugároz egy bizonyos frekvencián, amelyet T időtartam alatt modulál. Ez „időbélyeget” ad a továbbított jelnek
- A jel eljut a célponthoz, és egy része visszaverődik. A radar érzékeli a visszavert jelet, és összehasonlítja az eredetivel, összekeverve és feldolgozva a kapott jelet
- A visszavert és az eredeti jel **Δf** frekvenciaeltolódásából meghatározható a távolság, míg a több frekvenciasöprés során észlelt **fD** eltolódás a sebességet jelzi (Doppler-effektus)



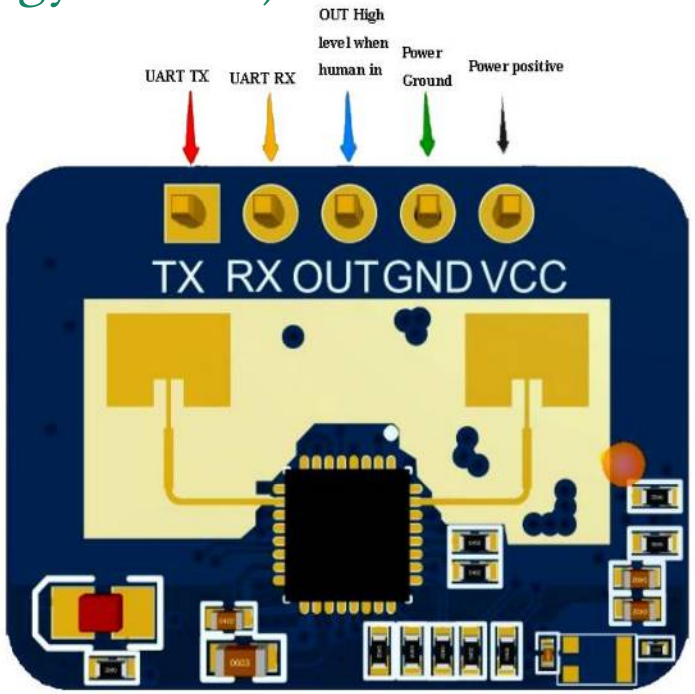
A célpont R távolságát az FMCW radarban a következő képlet adja meg:

$$R = \frac{c \cdot \Delta f}{2 \cdot S}$$

Ahol **S** a frekvenciasöprés sebessége [Hz/s], **c** a fénysebesség (kb. 3 x 10^8 m/s)

A HLK-2410C szenzor bemutatása

- A **HLK-2410C** egy 24 GHz-es **FMCW** radar alapú mozgásérzékelő, amely képes észlelni álló és mozgó célpontokat, és azok távolságát is meg tudja határozni
  Az OUT kimenet adott feltételek teljesülésekor aktiválható (például jelenlét érzékelésekor kimeneti jel kapcsolható egy relére vagy LED-re)
- **Mire képes?**
  - Mozgó és álló objektumok észlelése
  - Távolságmérés
  - Fényérzékelés (a környezeti fény szintje)
  - Automatikus érzékelési küszöb beállítások
  - Soros kommunikáció (UART, BLE)
- **Hol használható?**
  - Épületautomatizálás
  - Biztonsági rendszerek
  - Érintésmentes érzékelés

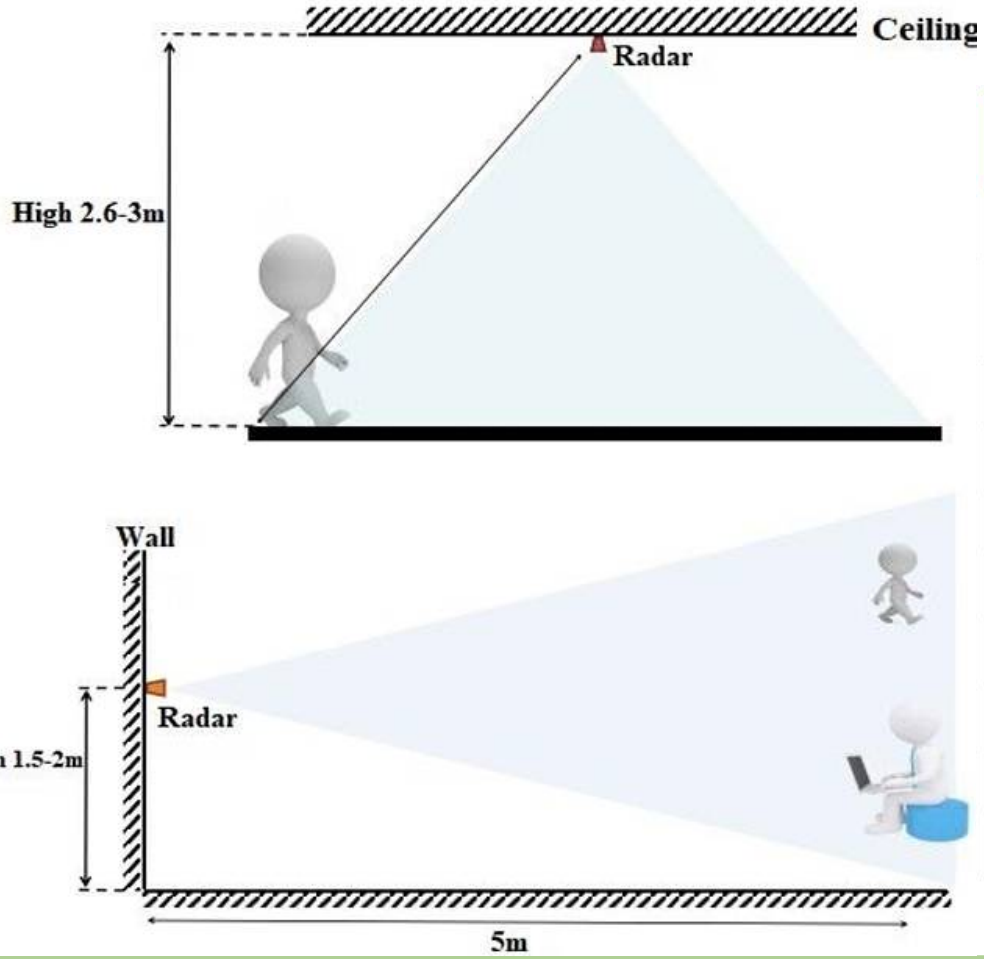


Teljesítményadatok és elektromos paraméterek

Operating frequency	24GHz~ 24.25GHz Compliant with FCC, CE, non-commission certification standards
Operating Voltage	DC 5V, power supply capacity>200mA
Average operating current	79 mA
Modulation	FMCW
Interface	A GPIO, IO level 3.3V A UART
Target application	Human presence sensor
Detection distance	0.75m ~ 6m, adjustable
Detection angle	±60 °
Distance resolution	0.75m
Sweep Bandwidth	250MHz Compliant with FCC, CE, non-commission certification standards
Ambient temperature	-40 ~ 85°C
Dimensions	7mm x 35 mm

A jelszintek azonban 3,3 V-osak!

Tipikus alkalmazások jellemzői

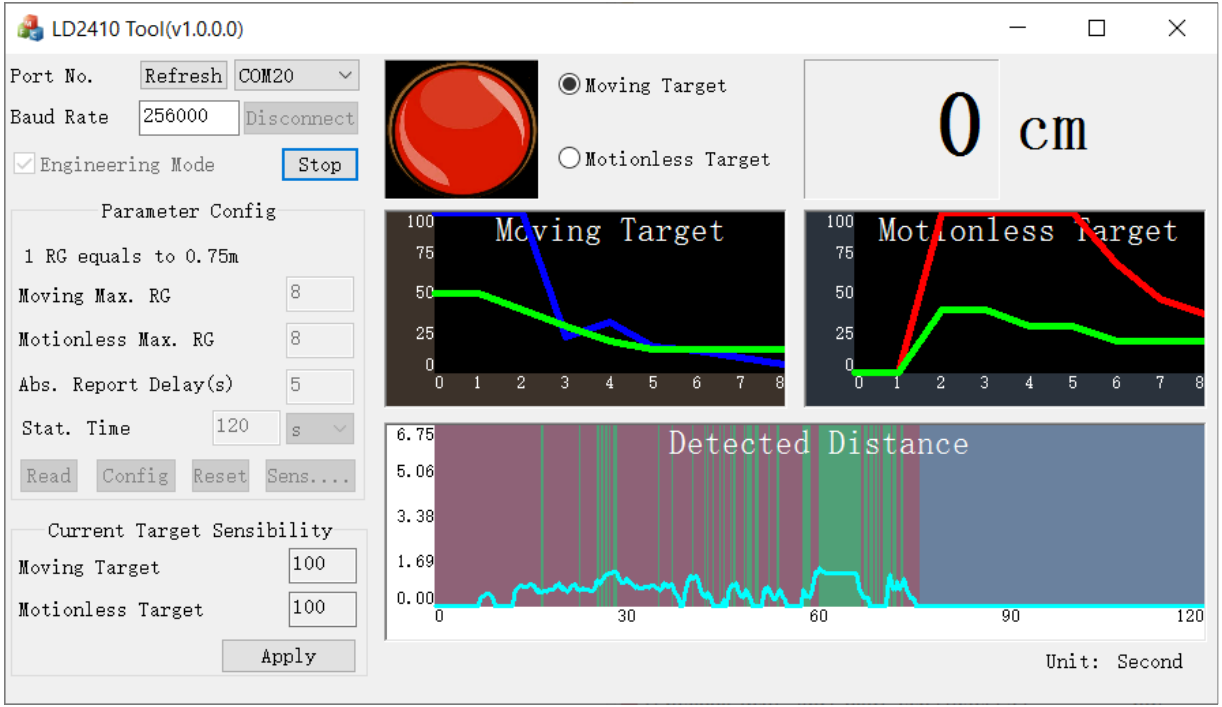


	Infrared solution	Visual solution	Ultrasonic wave	Lidar	Millimeter wave radar
Application flexibility	Weak	Weak	Weak	Weak	Good
Resistance to environmental influences (weather light, etc.)	Weak	Weak	Weak	Common	Good
Detection speed	Weak	Common	Common	Good	Good
Detection accuracy	Weak	Common	Good	Good	Good
Resolution	Weak	Good	Good	Good	Good
Directionality	Weak	Common	Weak	Good	Good
Detection distance	Common	Weak	Weak	Good	Good
Ability to penetrate material	Weak	Weak	Weak	Weak	Good
Dimension	Good	Weak	Good	Weak	Good
Cost	Good	Weak	Good	Weak	Common

Good Common Weak

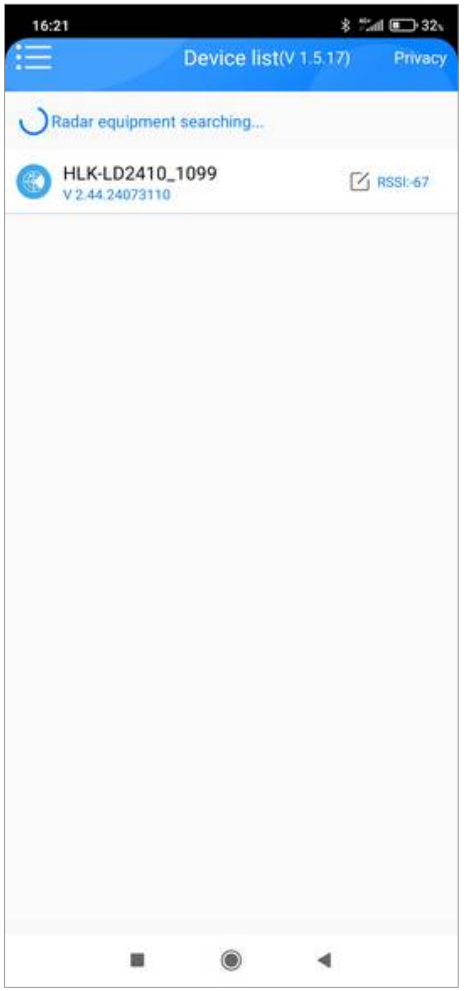
LD2410 Tool – PC alkalmazás

- A **HLK-LD2410 Tool** egy Windows alapú konfigurációs alkalmazás, amely lehetővé teszi a **HLK-2410C** szenzor beállításainak módosítását és a működésének monitorozását **egy USB-UART átalakítón keresztül**
- Valós idejű adatmegjelenítés
- Paraméterek módosítása – Kapuk érzékenységének finomhangolása, maximális hatótáv beállítása.
- Automatikus küszöbérték tanítás: Az érzékelési küszöbök optimalizálása a környezethez.
- A felhasználó grafikus interfészen keresztül vezéreli a beállításokat



HLKRadarTool – Android alkalmazás

- A HLKRadarTool Android alkalmazás, amely lehetővé teszi a **HLK-LD2410** szenzor Bluetooth (BLE) kapcsolaton keresztüli konfigurálását és monitorozását
- **Valós idejű adatmegjelenítés** – A szenzor által érzékelt célpontok és távolságok figyelése
- **Paraméterek módosítása** – Kapuk érzékenységének finomhangolása, maximális hatótáv beállítása
- **Automatikus küszöbérték tanítás** – Az érzékelési küszöbök optimalizálása a környezethez



Config, Basic és Enhanced üzemmód

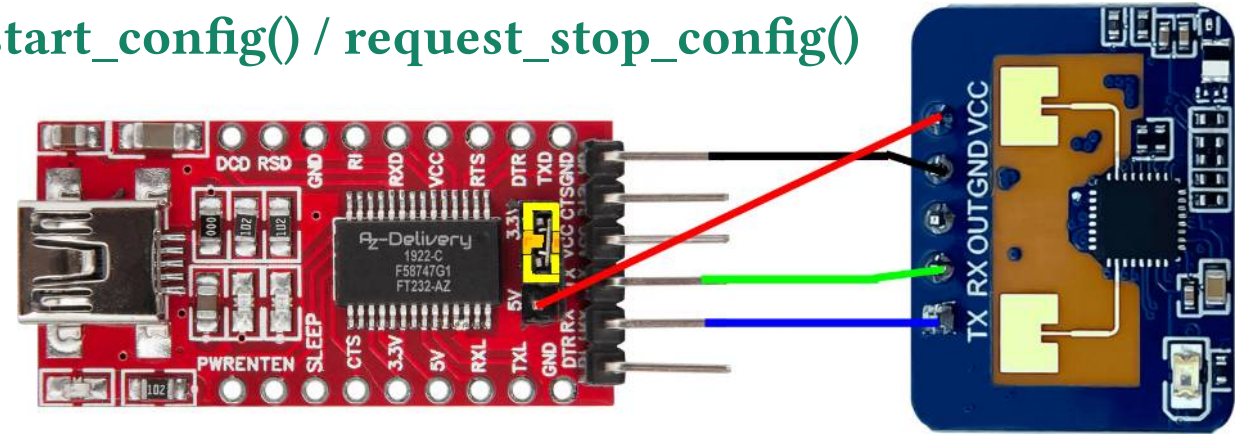
- A **HLK-2410C** szenzor háromféle módban működhet:
  - **Config Mode** – Ebben a módban beállításokat végezhetünk a szenzoron Ebben az üzemmódban a szenzor nem küld érzékelési adatokat, hanem lehetőséget biztosít a beállítások módosítására (Hatótáv és érzékenység beállítása, automatikus küszöbértékek tanítása, Soros kommunikációs paraméterek módosítása)
  - **Basic Mode** – A szenzor egyszerű jelenlét-érzékelést végez és egyszerű adatokat küld (pl. mozgás van/nincs, távolság cm-ben)
  - **Enhanced Mode** - Részletesebb információkat küld a célpontokról (pl. érzékelt mozgó és álló célpontok jelei). Küszöbértékeket és jelek intenzitását is továbbítja. Ideális pontos mozgásérzékeléshez és távolságméréshez

HLK-LD2410C kezelés számítógéppel

PC támogatás: az aio-ld2410 programkönyvtár

- **aio-ld4310** – Python aszinkron programkönyvtár a **HLK-LD2410C** radarhoz
- Vezérlés és adatfolyam (bináris csomagokkal): soros porton keresztül (UART)
- Telepítés: `pip install aio-ld2410` (függőség: **pyserial, asyncio**)
- Fő objektum: **LD2410**(transport) – a protokoll és állapotkezelő mag
- Működési módok: Támogatja a Standard és Engineering (részletes) módokat
- Mozgásadatok: **moving_target_distance, moving_target_energy**
- Statikus adatok: **static_target_distance, static_target_energy**
- Detektálási táv: **detection_distance** (cm-ben kifejezve)
- Konfigurációs mód: **request_start_config() / request_stop_config()**
- Aszinkronitás: Minden I/O művelet **awaitable**

Bekötési vázlat:



A Python asyncio programkönyvtár

- Az **asyncio** egy Python könyvtár, amely lehetővé teszi több feladat futtatását egyetlen szálon, várakozás közbeni váltogatással. Főbb rendszerelemei:
  - **Eseményhurok (Event Loop):** A központi vezérlő, amely ütemezi a feladatokat
  - **Korutinok: `async def`**-fel létrehozott függvények
  - **Vezérlésátadás:** Az **`await`** kulcsszónál a program félreteszi az aktuális feladatot, és amíg az várakozik (pl. adatra a radartól), addig más munkát végez
- Példa: weboldalak letöltése

```python
import time
import requests

urls = [
    "https://example.com",
    "https://httpbin.org/delay/1",
    "https://httpbin.org/delay/2",
]

start = time.time()
for url in urls:
    requests.get(url)  # blokkol
print("Idő:", time.time() - start)
```

```python
import asyncio, aiohttp, time
urls = [   "https://example.com",
    "https://httpbin.org/delay/1",
    "https://httpbin.org/delay/2",
]
async def fetch(session, url):
    async with session.get(url) as resp:
        return await resp.text()
async def main():
    start = time.time()
    async with aiohttp.ClientSession() as session:
        tasks = [fetch(session, url) for url in urls]
        await asyncio.gather(*tasks)  # párhuzamosan futnak
    print("Idő:", time.time() - start)
asyncio.run(main())
```

aio-ld2410 – Lekérdező függvények (Getters)

- **get_firmware_version()**: A modul belső szoftververziójának lekérése
- **get_bluetooth_address()**: A beépített BT modul MAC címének kiolvasása
- **get_distance_resolution()**: Aktuális felbontás (0.25m vagy 0.75m)
- **get_parameters()**: Érzékenységi és távolsági küszöbértékek lekérése
- **get_last_report()**: A legutolsó érvényes állapot lekérése (nem blokkoló)
- **get_next_report()**: Várakozás a következő beérkező adatcsomagra
- **get_reports()**: Aszinkron iterátor a folyamatos adatfolyamhoz
- **get_light_control()**: Fényérzékelőhöz kötött kimeneti paraméterek
- **motion_state**: Property (bool), jelzi ha bármilyen mozgást érzékel a radar
- **static_target_distance**: Property (int), az álló cél távolsága cm-ben
- **moving_target_distance**: Property (int), a mozgó cél távolsága cm-ben
- **static_target_energy**: Property (int), az álló cél visszaverődési energiája (0-100)
- **moving_target_energy**: Property (int), a mozgó cél visszaverődési energiája (0-100)
- **detection_distance**: Property (int), a legközelebbi detektált objektum távolsága
- **max_moving_gate**: A konfigurált maximális mozgási távolság kapu indexe

aio-ld2410 – Beállító függvények (Setters)

- **set_engineering_mode(bool):** Részletes mérési adatok (kapu-energia) bekapcsolása
- **set_distance_resolution(res):** Távolsági felbontás (lépköz) módosítása
- **set_gate_sensitivity(gate, mov, stat):** Adott kapu érzékenységének finomhangolása
- **set_parameters(min, max, timeout):** Detektálási tartomány és tartási idő beállítása
- **set_baud_rate(baud):** Soros sebesség állítása (alapértelmezett: 256 000 bps)
- **set_bluetooth_mode(bool):** Bluetooth rádió távoli engedélyezése vagy tiltása
- **set_bluetooth_password(pwd):** 6 jegyű PIN kód beállítása a BT hozzáféréshez
- **set_light_control(params):** Fényérzékelési küszöbértékek és logika programozása
- **configure():** Speciális blokk, használata: `async with device.configure():`
  Automatikusan indítja és lezárja a konfigurációs módot. Előnye, hogy ha hiba történik a kódban, a radar akkor sem ragad be **config** módban
  Alternatíva: `request_start_config()` – ha kézzel akarjuk nyitni a módot
- **Gate indexek:** 0-tól 8-ig (vagy 15-ig) terjedő távolsági zónák
- **Fontos:** A fenti set metódusok csak a **configure()** blokkon belül működnek

Példaprogramok az aio-ld2410 programkönyvtárhoz

- **open_device.py**: Kapcsolódás a soros porthoz és az eszköz inicializálása
- **read_basic_reports.py:** A célpontok állapotának, távolságának és energiájának folyamatos olvasása és formázott kiírása alapmódban (BASIC MODE)
- **read_distance_resolution.py:** A jelenlegi távolsági lépésköz (felbontás) lekérdezése
- **read_firmware_version.py:** A modul belső szoftververziójának kiolvasása
- **read_simple_configuration.py:** A hatótávolság, az időzítés és a kapunkénti érzékenységi küszöbök lekérdezése.
- **read_simple_reports.py:** mérési adatok beolvasása alapmódban (BASIC MODE)
- **reset_to_factory.py:** Gyári alapbeállítások visszaállítása a modulon
- **write_distance_configuration.py:** A távolsági lépésköz beállítása
- **write_gate_sensitivity.py:** a kapunkénti érzékenységi küszöbök beállítása
- **write_simple_configuration.py:** A hatótávolsági határok és az időtúllépés beállítása

Saját mintapéldák:

- **ld2410_full_stream.py** - lekérdezi a szenzor beállításait, majd mérnöki módba (Engineering Mode) kapcsolva folyamatosan listázza a részletes jelentéseket
- **ld2410_basic_demo.py** – A BASIC módú adatok megjelenítése grafikus felületen

Két egyszerű mintapélda

- **read_distance_resolution.py** az aktuális távolsági lépésköz (75 cm, vagy 20 cm) lekérdezése és kiíratása
- **read_simple_reports.py:** mérési adatok beolvasása alapmódban (BASIC MODE) és kiíratása formázatlanul

```
ReportStatus (obj)
├── .basic (obj)
│   ├── .target_status      (enum)
│   ├── .moving_distance    (int)
│   ├── .moving_energy      (int)
│   ├── .static_distance    (int)
│   ├── .static_energy      (int)
│   └── .detection_distance (int)
│
└── .engineering (obj/None)
```

```python
import asyncio
from aio_ld2410 import LD2410

async def main():
    async with LD2410('COM4') as device:
        async with device.configure():
            res = await device.get_distance_resolution()
    print(f'Each gate covers {res} centimeters')

if __name__ == '__main__':
    asyncio.run(main())
```

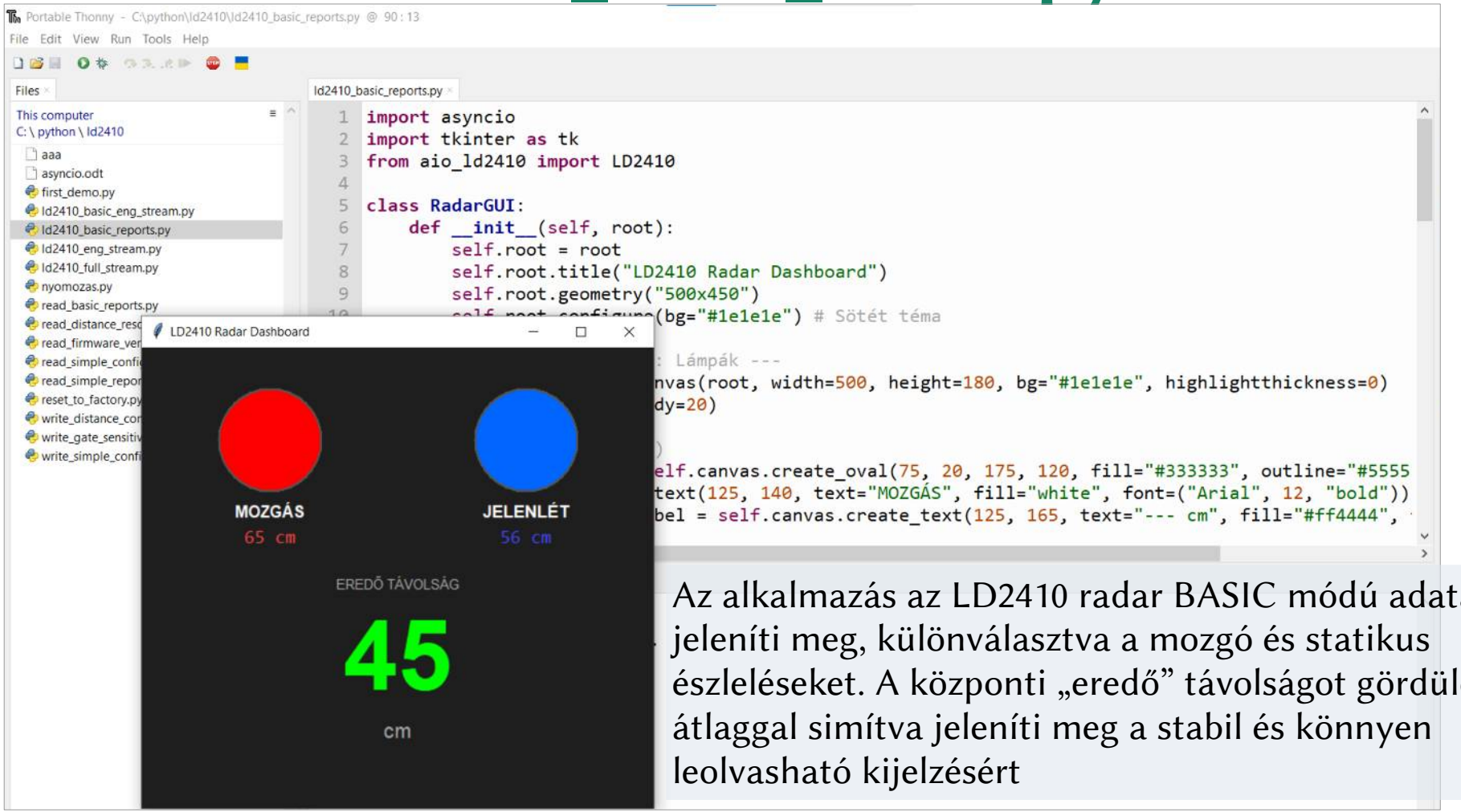
```python
import asyncio
from aio_ld2410 import LD2410

async def main():
    async with LD2410('COM4') as device:
        async with device.configure():
            ver = await device.get_firmware_version()
            print(f'[+] Running with firmware v{ver}')
        # Reports are typically generated every 100ms.
        async for report in device.get_reports():
            print(report)

if __name__ == '__main__':
    asyncio.run(main())
```

A kiterjesztett módú riport szerkezete

- A kiterjesztett üzemmódban (engineering mode) részletesebb információt kapunk a szenzortól
- Megjegyzés: fényérzékelő csak az **LD2410C** típusban van

```
report (objektum)
├── .basic (ReportBasicStatus objektum)
│   ├── .target_status        -> <TargetStatus.MOVING|STATIC: 3>
│   ├── .moving_distance      -> 30
│   ├── .moving_energy        -> 100
│   ├── .static_distance      -> 60
│   ├── .static_energy        -> 100
│   └── .detection_distance   -> 43
│
└── .engineering (ReportEngineeringStatus objektum)
    ├── .moving_max_distance_gate -> 8
    ├── .static_max_distance_gate -> 8
    ├── .moving_gate_energy       -> [100, 100, 72, 9, 10, 5, 4, 7, 5]
    ├── .static_gate_energy       -> [0, 0, 100, 100, 100, 68, 42, 16, 16]
    ├── .photosensitive_value     -> 30
    └── .out_pin_status           -> <OutPinLevel.HIGH: 1>
```

ld2410_basic_demo.py



Az alkalmazás az LD2410 radar BASIC módú adatait jeleníti meg, különválasztva a mozgó és statikus észleléseket. A központi „eredő” távolságot gördülő átlaggal simítva jeleníti meg a stabil és könnyen leolvasható kijelzésért

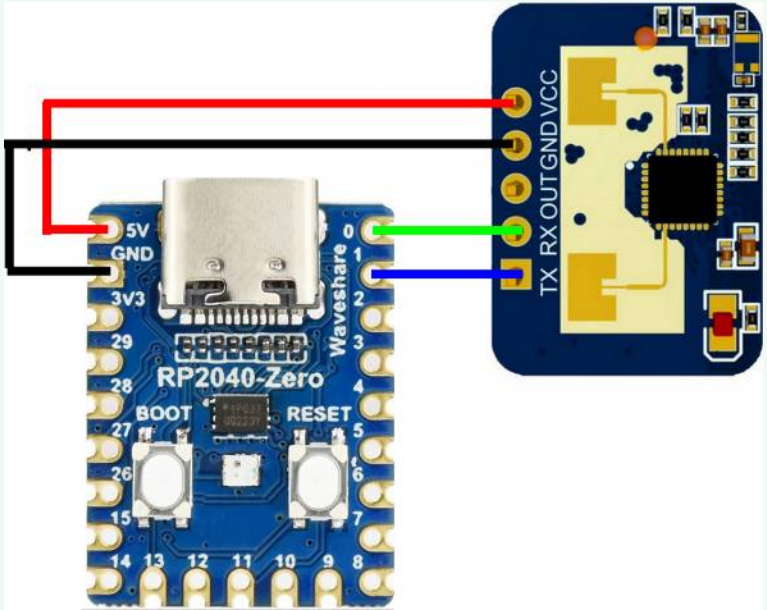
ld2410_full_stream.py

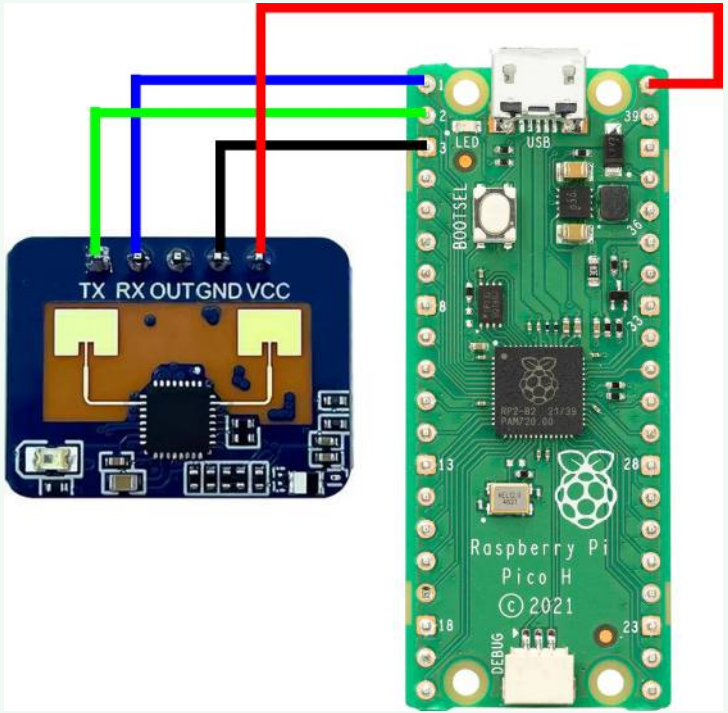
```
==========================================
      LD2410 RENDSZER-DIAGNOSZTIKA (v2.44)
==========================================
[+] Firmware: v2.44.24073110
[+] Kapu felbontás: 75 cm
[+] Max észlelés:  600 cm
[+] Max mozgó táv: 600 cm
[+] Max álló táv:  600 cm
[+] Időtúllépés:   5 mp

[+] ÉRZÉKELÉSI KÜSZÖBÖK (Gate Thresholds):
Kapu | Mozgó Küszöb | Álló Küszöb
-----|--------------|-------------
 0.  |       50     |        5
 1.  |       50     |        5
 2.  |       40     |       40
 3.  |       30     |       40
 4.  |       20     |       30
 5.  |       15     |       30
 6.  |       15     |       20
 7.  |       15     |       20
 8.  |       15     |       20
==========================================
Indul az élő adatfolyam...
```

```
=== TELJES ADATCSOMAG ===
ÁLLAPOT: Mozgó + Álló
DETEKTÁLT TÁV: 40 cm
MOZGÓ CÉL -> Táv:  62 cm | Energia: 100
ÁLLÓ CÉL  -> Táv:  62 cm | Energia: 100
MOZGÓ ENERGIÁK (0-8): [100, 100, 100, 18, 7, 11, 2, 10, 3]
ÁLLÓ ENERGIÁK  (0-8): [0, 0, 100, 100, 100, 63, 37, 21, 12]
MAX TÁV KAPU -> Mozgó: 8 | Álló: 8
FÉNYERŐ ÉRTÉK: 29
OUT PIN ÁLLAPOT: 1
----------------------------------------------------------
=== TELJES ADATCSOMAG ===
ÁLLAPOT: Mozgó + Álló
DETEKTÁLT TÁV: 39 cm
MOZGÓ CÉL -> Táv:  54 cm | Energia: 100
ÁLLÓ CÉL  -> Táv:  62 cm | Energia: 100
MOZGÓ ENERGIÁK (0-8): [100, 100, 87, 29, 14, 5, 1, 9, 4]
ÁLLÓ ENERGIÁK  (0-8): [0, 0, 100, 100, 100, 64, 38, 21, 12]
MAX TÁV KAPU -> Mozgó: 8 | Álló: 8
FÉNYERŐ ÉRTÉK: 29
OUT PIN ÁLLAPOT: 1
----------------------------------------------------------
```

HLK-LD2410C kezelés Arduino alatt

- Kísérleteinket Raspberry Pi Pico kártyával végeztük, melynek ARDUINO környezetben történő programozásához az **Arduino Mbed OS RP2040 boards** nevű bővítményt használtuk





A MyLD2410 programkönyvtár

- Arduino környezetben a MyLD2410 könyvtár segítségével kezelhetjük a **HLK-2410C** szenzort (a szűkszavú dokumentáció itt található)
- Olyan kártyával célszerű használni, amelynek van szabad hardveres soros portja, mivel a radar modul nagy sebességgel kommunikál (256 000 baud a default)
- Telepítés Arduino IDE-ben:
  - Nyisd meg az Arduino IDE-t
  - Menj a Tools menüben a Library Manager-be
  - Kereső kifejezésként ezt írd be: LD2410
  - Válaszd a **MyLD2410** könyvtárat
  - Kattints az INSTALL (Telepítés) gombra
- **Megjegyzés:** A **Library Manager**-ben látni fogunk egy **LD2410** nevű másik könyvtárat is, ami ESP32-höz jó, de az **Arduino MbedOS** környezetben átalakítás nélkül nem használható, mert **FreeRTOS** környezetet feltételez

A MyLD2410 osztály fontosabb tagfüggvényei

- **Beállítás és üzemmódok**
  - **begin()** – Elkezd kommunikálni az eszközzel
  - **end()** – Lezárja az érzékelőt (pl. alvó mód)
  - **configMode**(bool enable=true) – Konfigurációs módba lépteti az eszközt
  - **enhancedMode**(bool enable=true) – Aktiválja az "Enhanced Mode"-ot
  - **setResolution**(bool fine=false) – Beállítja az érzékelő felbontását (75cm/20cm)
  - **SetNoOneWindow**(byte dly) – a távollét állapotba lépés késleltetési idejét állítja be
  - **setMaxGate**(byte movingGate, byte stationaryGate, byte noOneWindow=5) – Beállítja az érzékelési tartományt mozgó és álló célpontokhoz
- **Állapotkezelés**
  - **check()** – Ellenőrzi, hogy van-e új adat a szenzorból.
  - **getStatus**() – Érzékelő státusza (0–6, 255 = érvénytelen)
  - **statusString**() – Státusz szövegesen
  - **requestReset**() – Gyári beállítások visszaállítása.
  - **requestReboot**() – Szenzor újraindítása

A MyLD2410 osztály fontosabb tagfüggvényei

- **Jelenlét és célpontok érzékelése**
  - **presenceDetected**() – Megállapítja, hogy van-e jelenlévő objektum
  - **detectedDistance**() – Az észlelt távolság lekérése [cm]
  - **movingTargetDetected**() – Mozgó célpont érzékelése
  - **movingTargetDistance**() – Mozgó célpont távolságának lekérése
  - **movingTargetSignal**() – Mozgó célpont jelének lekérése
  - **stationaryTargetDetected**() – Álló célpont érzékelése
  - **stationaryTargetDistance**() – Álló célpont távolságának lekérése
  - **stationaryTargetSignal**() – Álló célpont jelének lekérése
- **Fejlett mód jelek és küszöbértékek**
  - **getMovingSignals**() – Mozgó célpont jeleinek lekérése
  - **getMovingThresholds**() – Mozgó célpont érzékelési küszöbértékeinek lekérése
  - **getStationarySignals**() – Álló célpont jeleinek lekérése
  - **getStationaryThresholds**() – Álló célpont érzékelési küszöbértékeinek lekérése

A MyLD2410 osztály fontosabb tagfüggvényei

- **Extra adatok és kimeneti vezérlés**
  - **getLightLevel()** – A környezeti fényerősség lekérése
  - **getLightControl()** – A fényvezérlés állapotának lekérése
  - **getOutputControl()** – Kimeneti vezérlés állapotának lekérése
  - **resetAuxControl()** – Kimeneti vezérlés visszaállítása alapértelmezett értékekre
  - **setAuxControl()** – az **OUT** kimenet viselkedését és a fényérzékelés küszöbértékét állíthatjuk be vele
- **A MyLD2410::SensorData adatstruktúra szerkezete**
  - byte **status**
  - unsigned long **timestamp**
  - unsigned long **mTargetDistance**
  - byte **mTargetSignal**
  - unsigned long **sTargetDistance**
  - byte **sTargetSignal**
  - unsigned long **distance**
  - ValuesArray **mTargetSignals**
  - ValuesArray **sTargetSignals**

A setAuxControl() függvény

- A **setAuxControl()** függvény segítségével az **OUT** kimenet viselkedését és a fényérzékelés küszöbértékét állíthatjuk be

```
bool MyLD2410::setAuxControl(LightControl light_control,    // A fényérzékelés beállításai
                             byte light_threshold,          // A fényerőszint küszöbértéke
                             OutputControl output_control); // OUT aktiválásának feltételei
```

- **light_control:**
  - `LightControl::DISABLED` A fényérzékelés kikapcsolása
  - `LightControl::ENABLED` A fényérzékelés aktiválása
  - `LightControl::SMART` Automatikus küszöb a környezet fényereje alapján
- **light_threshold**: 0–100 között, megadja, milyen fényerőszint felett legyen aktív az OUT kimenet
- **output_control:**
  - `OutputControl::ALWAYS_ON` Az OUT kimenet mindig aktív
  - `OutputControl::MOTION` Az OUT kimenet mozgás érzékelésekor aktív
  - `OutputControl::MOVING_ONLY` Az OUT kimenet csak mozgó célpontok érzékelésekor aktív
  - `OutputControl::STATIONARY_ONLY` Az OUT kimenet csak álló célpontok esetén aktív

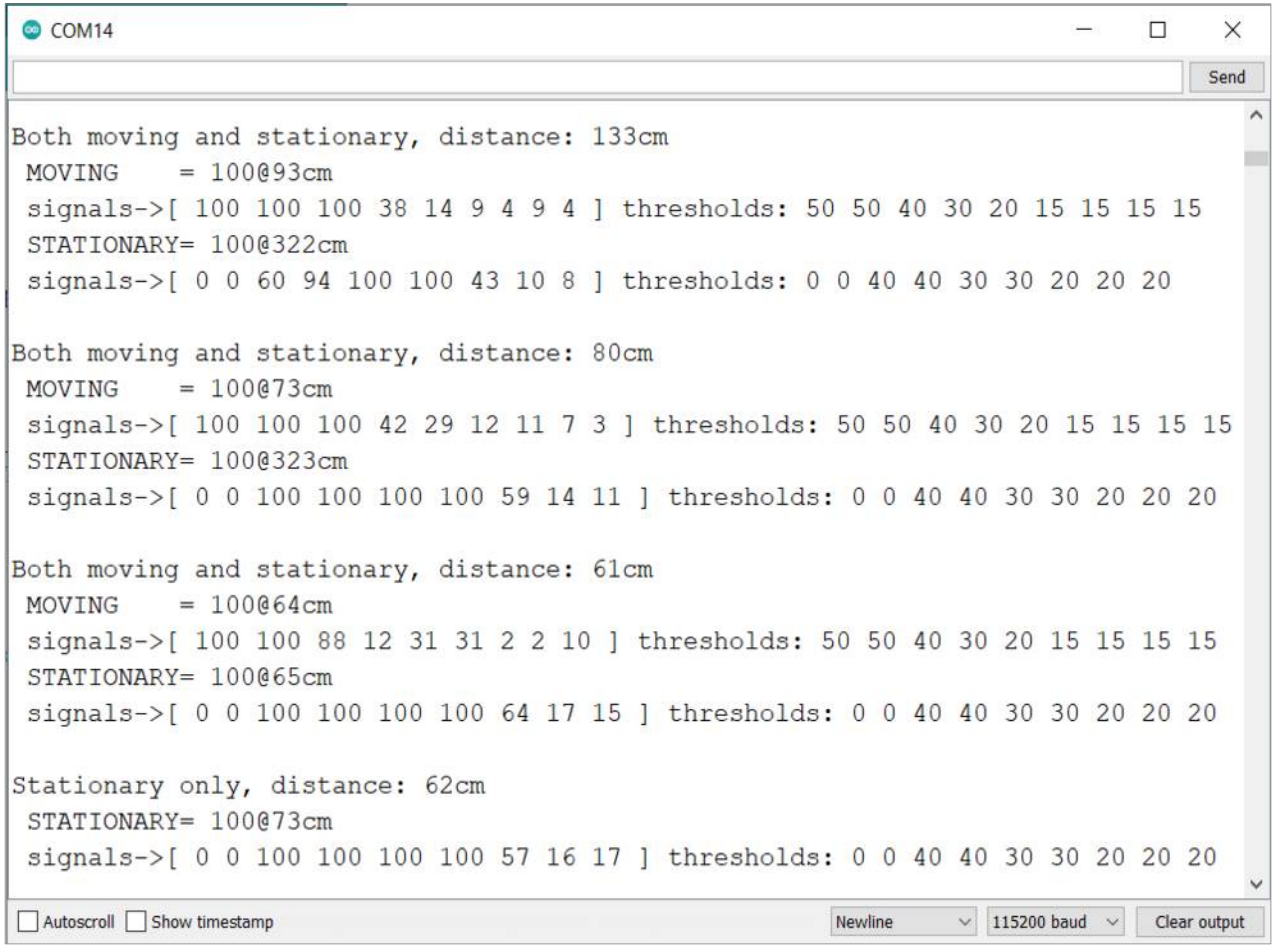
Egyszerű Basic Mode mintapélda

- Ha a szenzor 3 méteren belüli jelenlétet érzékel, felkapcsoljuk a vezérlő kimenetet

```
#include <MyLD2410.h>
MyLD2410 sensor(Serial1);
#define CONTROL_PIN 25                  // Vezérlő kimenet

void setup() {
    Serial.begin(115200);
    Serial.begin(256000);
    pinMode(CONTROL_PIN, OUTPUT);
    sensor.configMode(false);           // Kilépés Config Mode-ból
    sensor.enhancedMode(false);         // Kilépés Enhanced  Mode-ból
}

void loop() {
    if (sensor.check() == MyLD2410::DATA) {
        if (sensor.presenceDetected() && sensor.detectedDistance() <= 300) {
            digitalWrite(CONTROL_PIN, HIGH);
        } else { digitalWrite(CONTROL_PIN, LOW); }
    }
}
```

Egyszerű Enhanced Mode mintapélda

- Ez a program a **HLK-LD2410** szenzor **Enhanced Mode** működését mutatja be, amely részletes információt szolgáltat az érzékelt mozgó és álló célpontokról
- **Fő funkciók:**
  - Valós idejű jelenlét-érzékelés (mozgó és álló célpontok)
  - Távolságadatok megjelenítése
  - A jelek és küszöbértékek kiírása minden kapuhoz
- A szenzor külön adja meg a mozgó és álló objektumok távolságát, valamint azok jelintenzitását

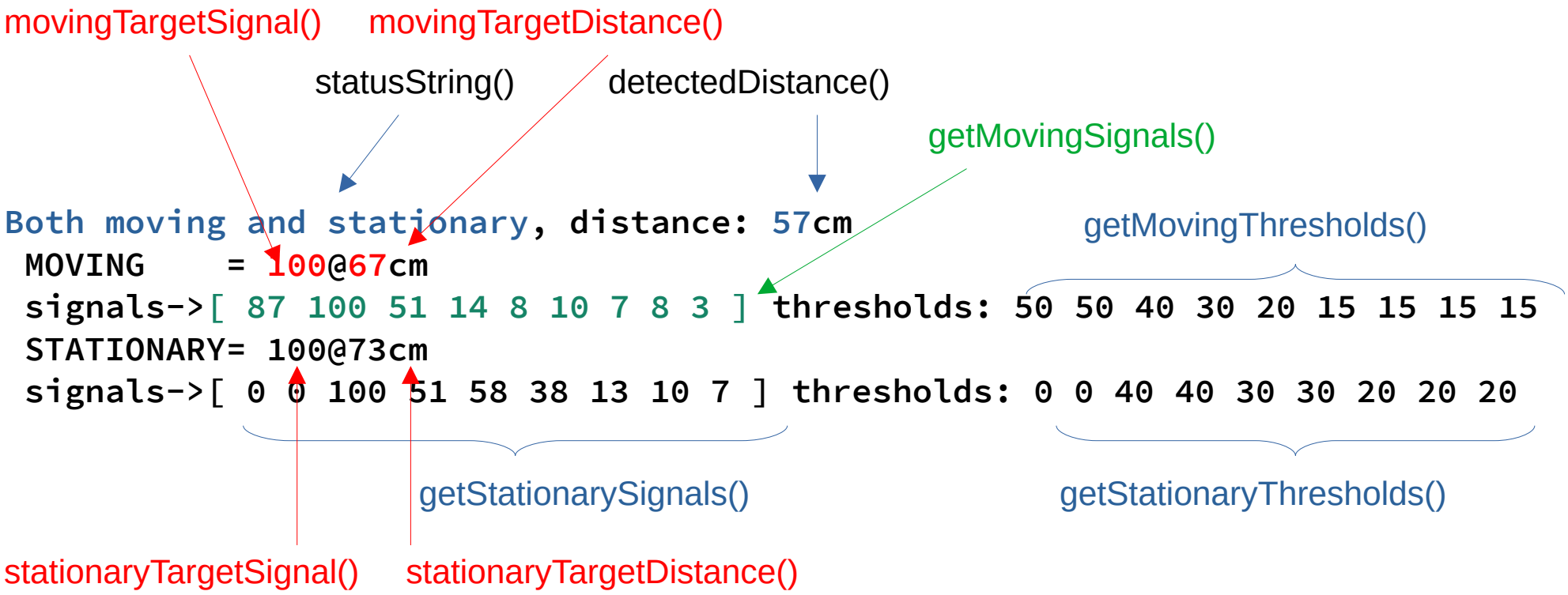


enhanced_demo.ino (részlet)

```
#include "MyLD2410.h"
MyLD2410 sensor(Serial1);
unsigned long nextPrint = 0, printEvery = 1000;

void setup() {
    Serial.begin(115200);
    Serial1.begin(LD2410_BAUD_RATE);
    if (!sensor.begin()) {
        Serial.println("Failed to communicate with the sensor.");
        while (true);
    }
    sensor.enhancedMode();
    delay(nextPrint);
}

void loop() {
    if ((sensor.check() == MyLD2410::Response::DATA) && (millis() > nextPrint)) {
        nextPrint = millis() + printEvery;
        printData();
    }
}
```

A printData függvényt itt nem részletezzük

Enhanced Mode mintapélda



Ez az ábra azt mutatja be, hogy a **printData()** függvényben a kiírt adatokat mely függvények segítségével olvastuk be

Automatikus küszöbbeállítás

- Az automatikus küszöbbeállítás azt jelenti, hogy a **HLK-LD2410C** jelenlétérzékelő maga határozza meg az optimális érzékelési küszöbértékeket egy kalibrációs folyamat során

A kalibrációs folyamat a következő lépésekből áll:

- **Kalibráció indítása** – Az **autoThresholds()** függvény meghívásakor az érzékelő konfigurációs módba lép, majd elindítja az automatikus küszöbérték beállítási folyamatot
- **Környezet elemzése** – Egy 10 másodperces időablak alatt a felhasználónak el kell hagynia a helyiséget, hogy az érzékelő az "üres tér" állapotát mérhesse fel
- **Érzékelési küszöbértékek meghatározása** – Az érzékelő ezen időszak alatt figyeli a háttérzajt és a környezeti interferenciákat, majd ennek alapján optimalizálja a mozgó és álló célok érzékelési küszöbeit
- **Beállítás véglegesítése** – Ha sikeres volt a folyamat, az új küszöbértékeket alkalmazza, így az érzékelő pontosabban és hatékonyabban tudja érzékelni a jelenlétet
- Az automatikus küszöbbeállítás pillanatnyi állapotáról a **getAutoStatus()** függvénnyel kérhetünk információt (NOT_SET, NOT_IN_PROGRESS , IN_PROGRESS , COMPLETED)

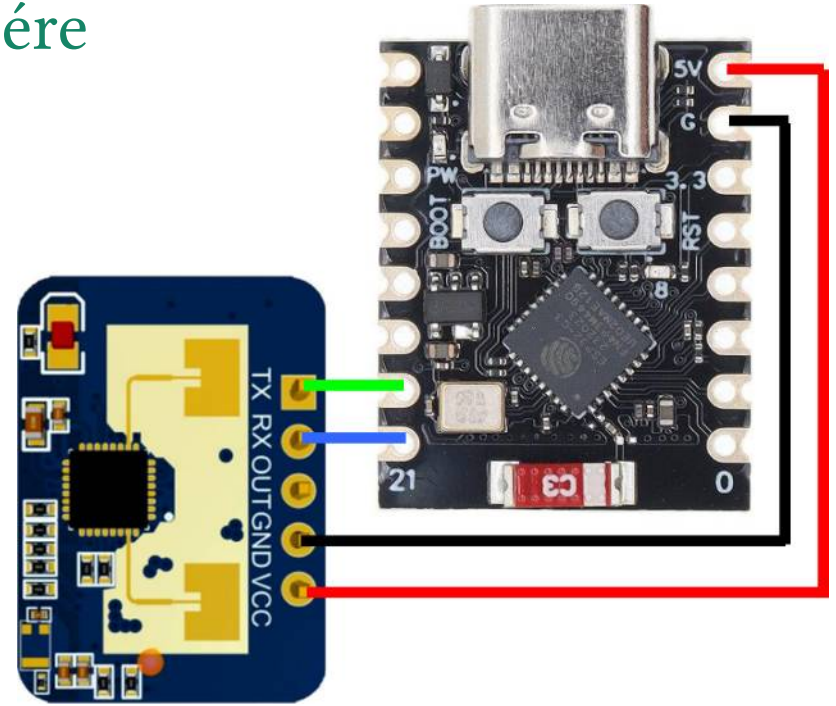
MyLD2410 mintapéldák

- A **MyLD2410** programkönyvtár az alábbi mintapéldákat tartalmazza:
  - **auto_thresholds** – az automatikus küszöbbeállítást mutatja be
  - **factory_reset** – gyári alapállapotba állítja a szenzor paramétereit
  - **modify_parameters** – az érzékelő paramétereinek beállítása/módosítása
  - **print_parameters** – kilistázza a szenzor paramétereit
  - **sensor_data** – a mért adatok kiírása (lényegében ez az **enhanced_demo.ino**)
  - **set_baud_rate** – a szenzor soros porti sebességének beállítása
  - **set_bt_password** – a BlueTooth jelszavána beállítása/törlése
- A fenti mintaprogramokat az **RP2040** mikrovezérlőre csak akkor tudjuk lefordítani, ha az alábbi változtatásokat (betoldás) elvégezzük a programokban:

```
#if defined(ARDUINO_SAMD_NANO_33_IOT) || defined(ARDUINO_AVR_LEONARDO) || defined(ARDUINO_ARCH_RP2040)

...

#if defined(ARDUINO_XIAO_ESP32C3) || defined(ARDUINO_XIAO_ESP32C6) || defined(ARDUINO_SAMD_NANO_33_IOT) ||
defined(ARDUINO_AVR_LEONARDO) || defined(ARDUINO_ARCH_RP2040)
```

HLK-LD2410C kezelés CircuitPython alatt

- Kísérleteinket **ESP32-C3** kártyával végeztük, CircuitPython 10.1.1 firmware és az általunk átírt **MyLD2410.py** programkönyvtár felhasználásával. Az adaptáció során törekedtünk a kompatibilitás megőrzésére így a könyvtár ismertetésére nem is térünk ki
- A bekötésnél mi az **ESP32-C3** kártya előre definiált **board.TX** (IO21) és **board.RX** (IO20) kivezetéseit használtuk, de az **UART** port lábai bármelyik lábra átkonfigurálhatók



Egyszerű Basic Mode mintapélda

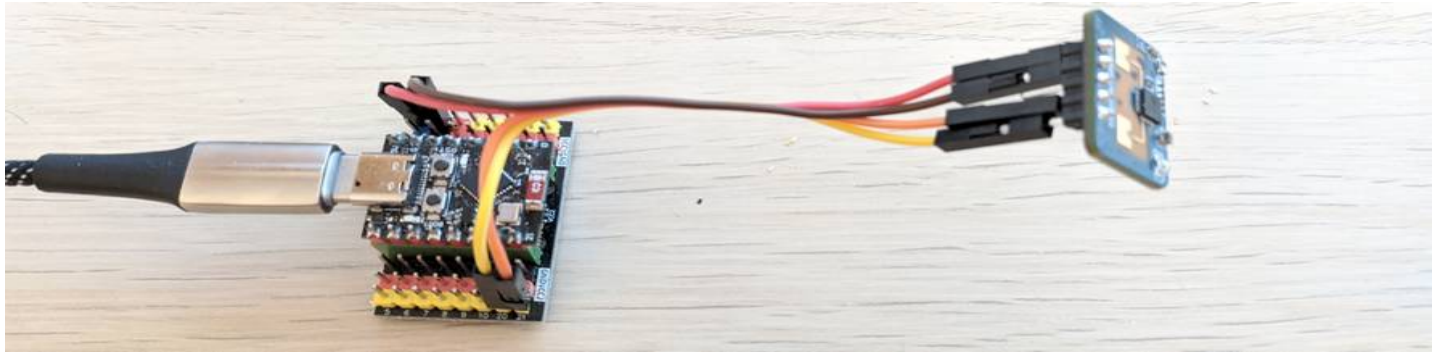
- Ha a szenzor 3 méteren belüli jelenlétet érzékel, felkapcsoljuk a vezérlő kimenetet

myld2410_mintapelda.py

```
import time, board, digitalio, busio
from MyLD2410 import MyLD2410

uart = busio.UART(tx=board.TX, rx=board.RX, baudrate=256000, timeout=0.01)
sensor = MyLD2410(uart) # Radar objektum
led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

sensor.configMode(False)       # Arduino: configMode(false)
sensor.enhancedMode(False)     # Arduino: enhancedMode(false)

while True:
    result = sensor.check()
    if result == MyLD2410.DATA:
        if sensor.presenceDetected() and sensor.detectedDistance() <= 300:
            led.value = False   # LED ON (katódvezérelt)
        else:
            led.value = True    # LED OFF
    time.sleep(0.01)
```

MyLD2410 mintapéldák

- A **MyLD2410** programkönyvtár mellek annak mintapéldáit is átírtuk CircuitPython nyelvere. A gyűjtemény az alábbi mintapéldákat tartalmazza:
  - **auto_thresholds** – az automatikus küszöbbeállítást mutatja be
  - **factory_reset** – gyári alapállapotba állítja a szenzor paramétereit
  - **modify_parameters** – az érzékelő paramétereinek beállítása/módosítása
  - **print_parameters** – kilistázza a szenzor paramétereit
  - **sensor_data** – a mért adatok kiírása (lényegében ez az **enhanced_demo.ino**)
  - **set_baud_rate** – a szenzor soros porti sebességének beállítása
  - **set_bt_password** – a BlueTooth jelszavának beállítása/törlése

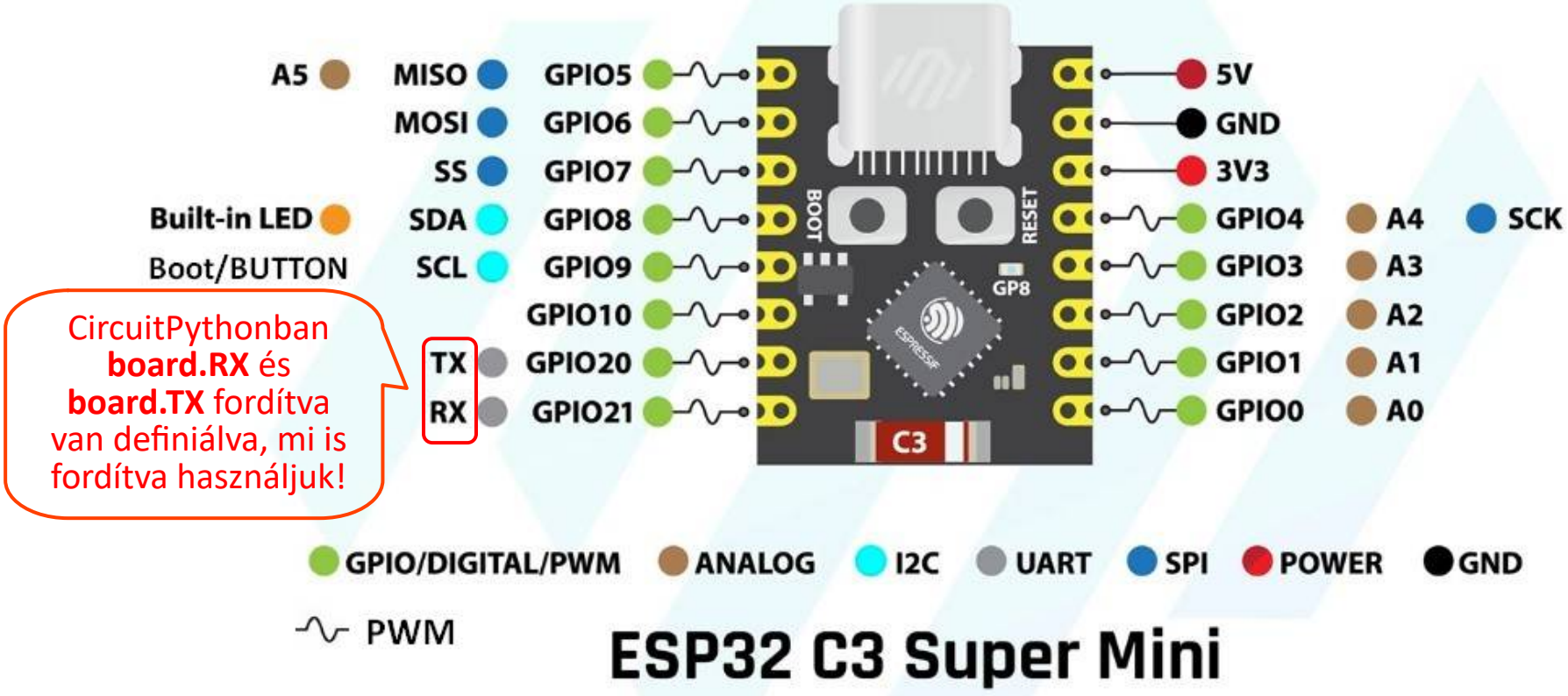


myld2410_sensor_data.py futási eredménye

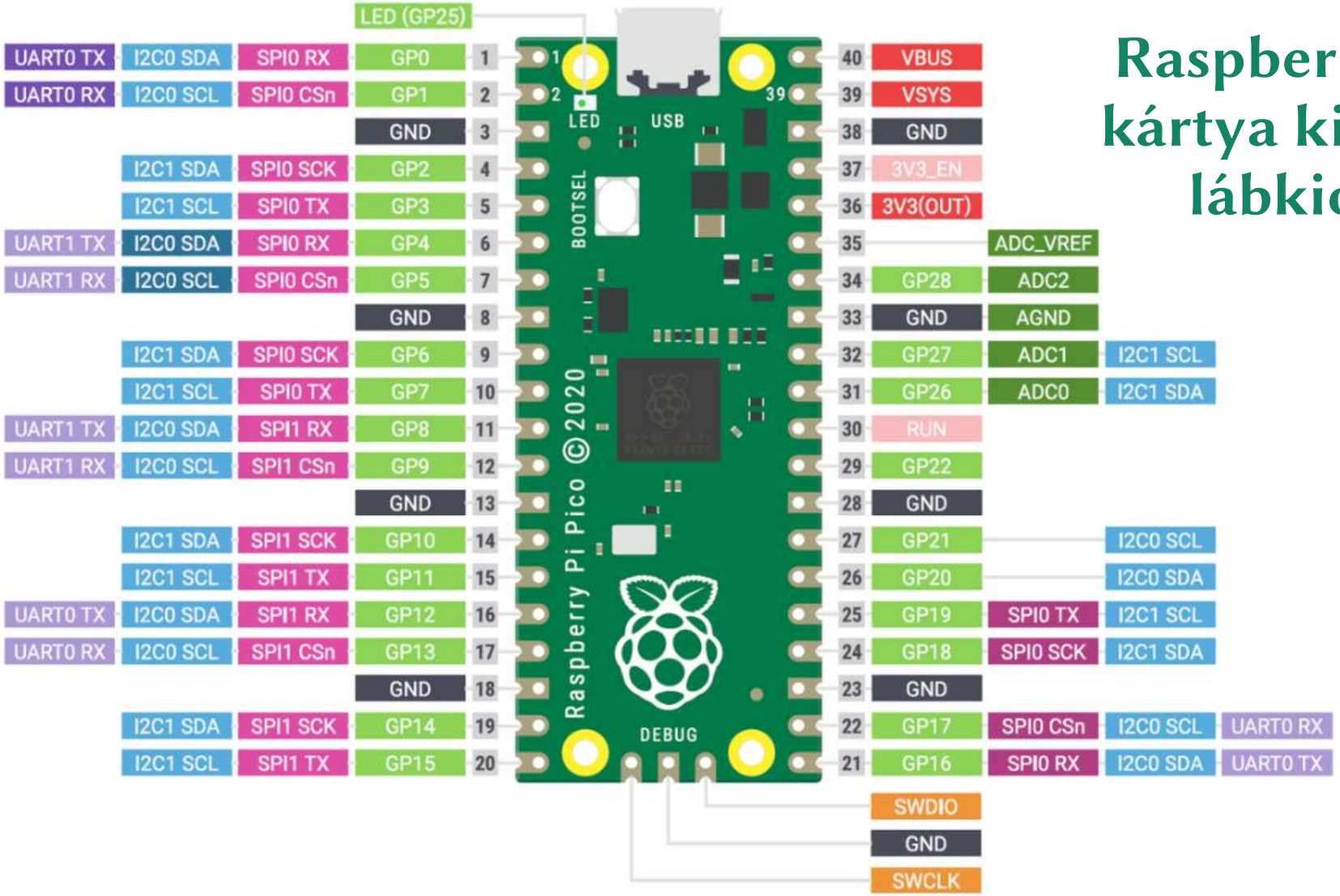
```
Initializing LD2410 sensor...
Checking AUX configuration support...
Running. Press Ctrl+C to stop.

Frame #: 1
Timestamp [ms]: 12130
Stationary only, distance: 0 cm
 STATIONARY = 100 @ 30 cm
  signals     ->   0
  thresholds  ->   5   5  40  40  30  30  20  20  20
 Light level: 0
 Output level: 0

Frame #: 21
Timestamp [ms]: 14195
Stationary only, distance: 31 cm
 STATIONARY = 100 @ 53 cm
  signals     ->   0   0 100  75  55  12  13  11   7
  thresholds  ->   5   5  40  40  30  30  20  20  20
 Light level: 102
 Output level: 1
```

A program bemutatja a HLK-LD2410 radarszenzor teljes funkcionalitását a MyLD2410 könyvtár használatával. Az alapmódú érzékelés mellett a kibővített mód adatait is lekérdezi, beleértve a szektorokra bontott részletes adatokat és a küszöbértékeket.

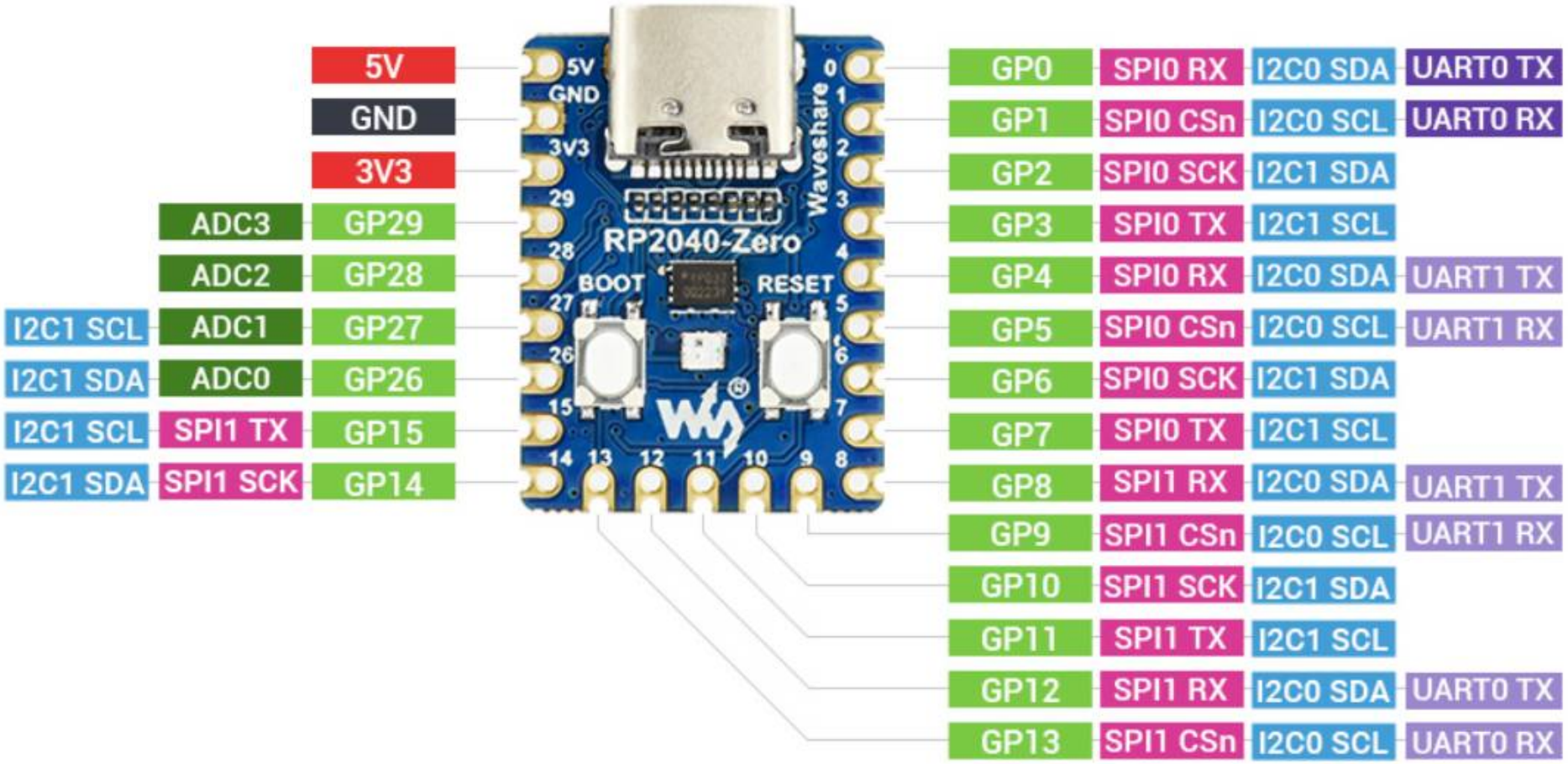
Az ESP32 C3 Super Mini kártya kivezetései



Raspberry Pi Pico kártya kivezetések lábkiosztása



Az RP2040-Zero kártya kivezetéseinek lábkiosztása



További kivezetések

- Néhány kivezetés az RP2040-Zero kártya hátoldalán van kivezetve