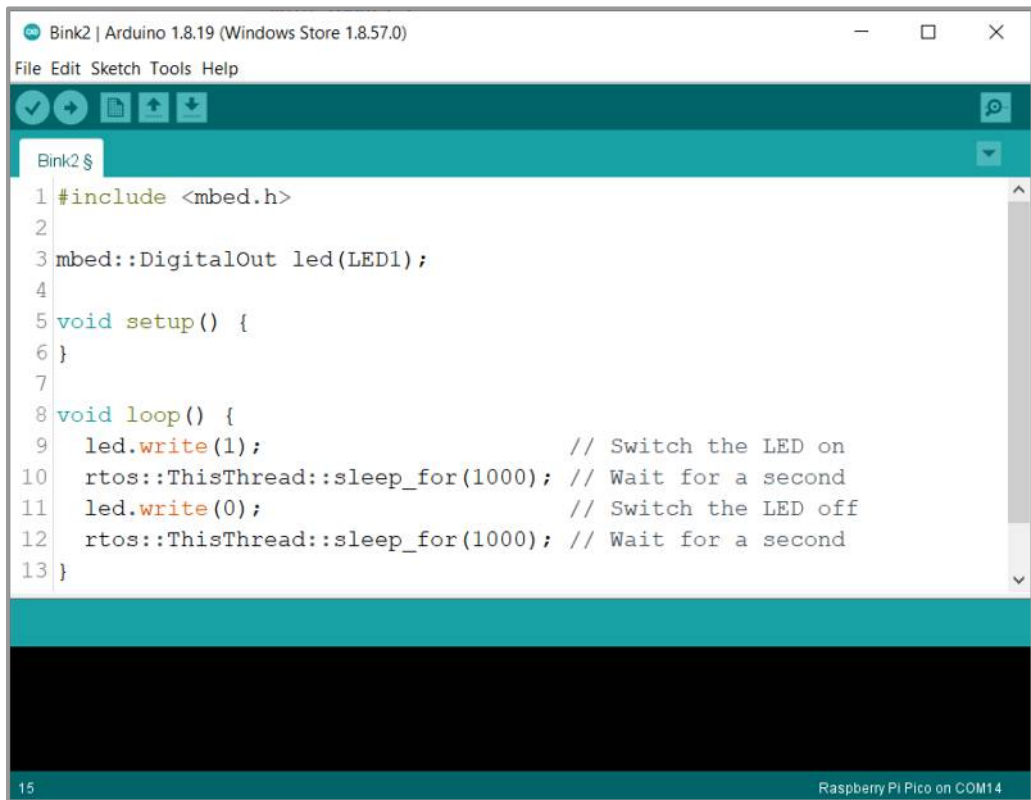
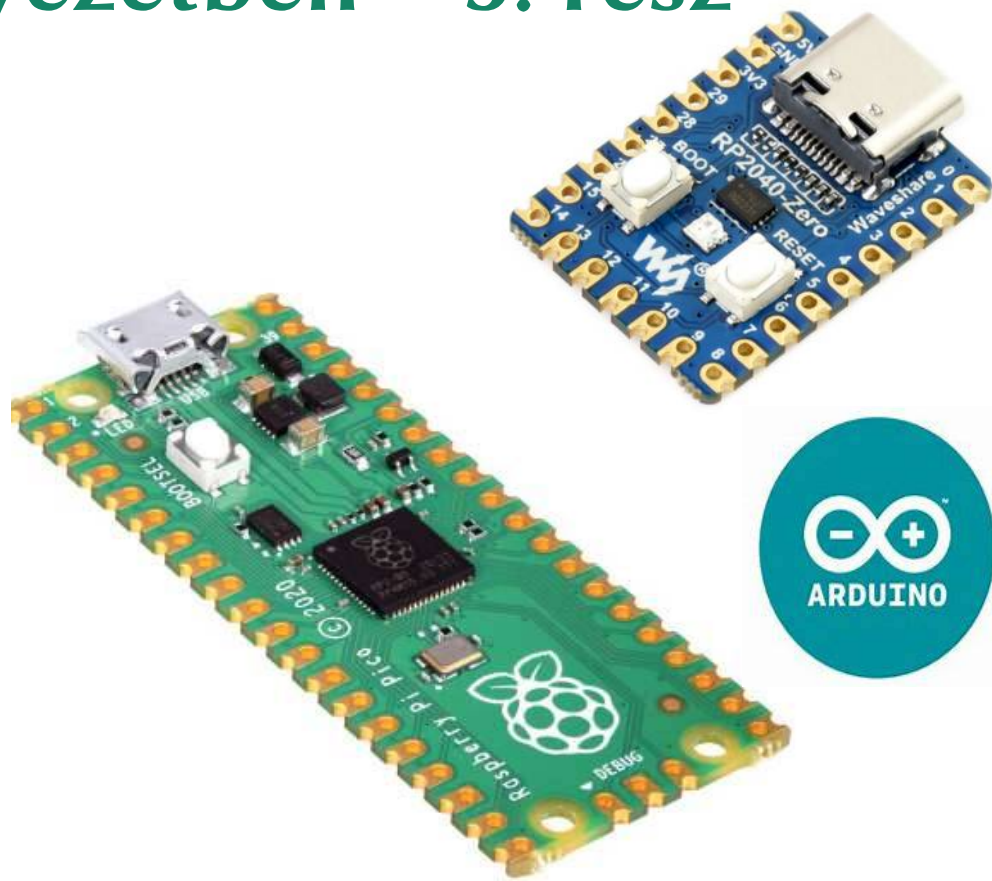


Az RP2040 mikrovezérlő programozása Arduino IDE környezetben – 5. rész



```
Bink2 §  
1 #include <mbed.h>  
2  
3 mbed::DigitalOut led(LED1);  
4  
5 void setup() {  
6 }  
7  
8 void loop() {  
9   led.write(1);           // Switch the LED on  
10  rtos::ThisThread::sleep_for(1000); // Wait for a second  
11  led.write(0);           // Switch the LED off  
12  rtos::ThisThread::sleep_for(1000); // Wait for a second  
13 }
```

15
Raspberry Pi Pico on COM14



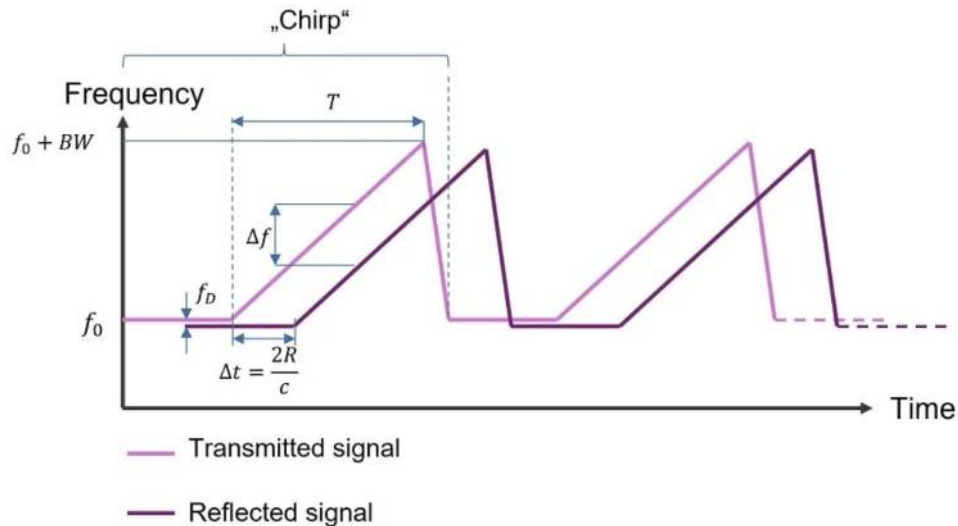
Felhasznált és ajánlott irodalom

- ❖ Raspberry Pi: [Pico-series Microcontrollers](#)
- ❖ Raspberry Pi: [RP2040 adatlap \(PDF\)](#)
- ❖ Raspberry Pi: [Raspberry Pi Pico Datasheet](#)
- ❖ Raspberry Pi: [Raspberry Pi Pico-series C/C++ SDK](#)
- ❖ Raspberry Pi: [Getting started with Raspberry Pi Pico-series Microcontrollers](#)
- ❖ Waveshare: [RP2040-Zero, a Pico-like MCU Board](#)

- ❖ Ralphjy: [Program RPi Pico using Mbed library with Arduino IDE](#)
- ❖ Learn Embedded Systems: [Basic Multicore Pico Project](#)
- ❖ BigG: [Four Multicore C programs for Raspberry Pi Pico using Arduino IDE](#)
- ❖ Valentin Milea: [DHT sensor library for the Raspberry Pi Pico](#)
- ❖ Alan Yorinks: [NeoPixelConnect](#)

FMCW Radar

- ❖ Az autóiipari radarendszerek 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 frekvenciasöpítés során észlelt f_D 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 frekvenciasöpítés sebessége [Hz/s],
 c a fénysebesség (kb. 3×10^8 m/s)

A HLK-2410C szenzor bemutatása

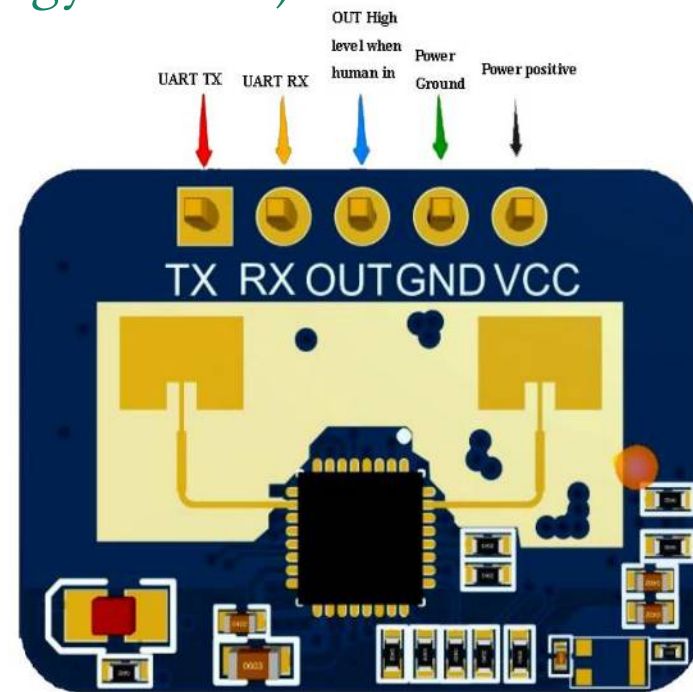
- ❖ A **HLK-2410C** egy 24 GHz-es **FMCW** radar alapú mozgásérzékelő, amely képes észlelni álló és mozgó célpontokat, és azok távolságát is meg tudja határozni. Az **OUT** kimenet adott feltételek teljesülésekor aktiválható (például jelenlét érzékelésekor kimeneti jel kapcsolható egy relére vagy LED-re)

- ❖ **Mire képes?**

- Mozgó és álló objektumok észlelése
- Távolságmérés
- Fényérzékelés (a környezeti fény szintje)
- Automatikus érzékelési küszöb beállítások
- Sorosan kommunikáció (UART, BLE)

- ❖ **Hol használható?**

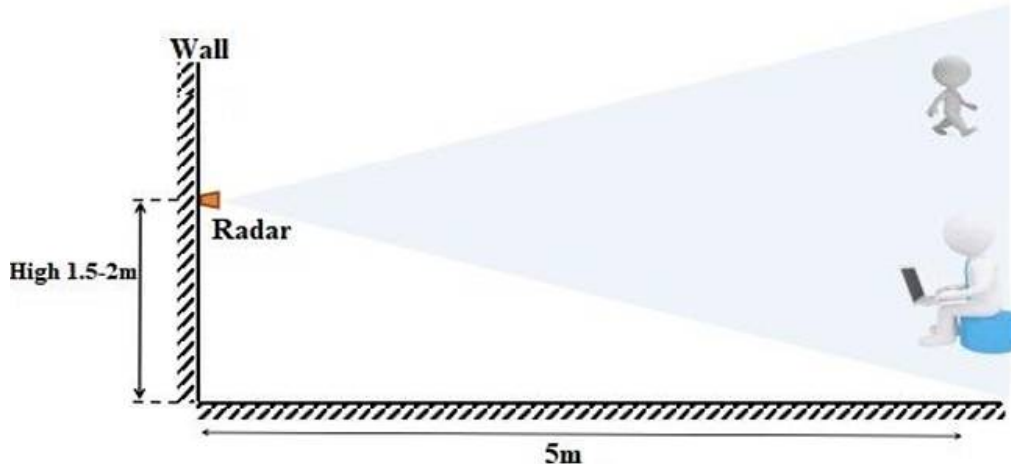
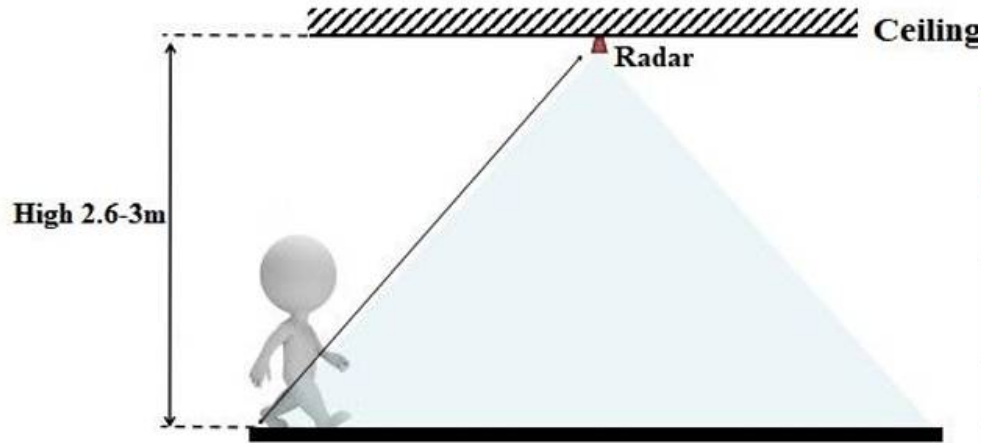
- Épületautomatizálás
- Biztonsági rendszerek
- Érintésmentes érzékelés



Teljesítményadatok és elektromos paraméterek

Operating frequency	24GHz~ 24.25GHz Compliant with FCC, CE, non-commission certification standards
Operating Voltage	<u>DC 5V, power supply capacity>200mA</u>
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

Tipikus alkalmazások jellemzői

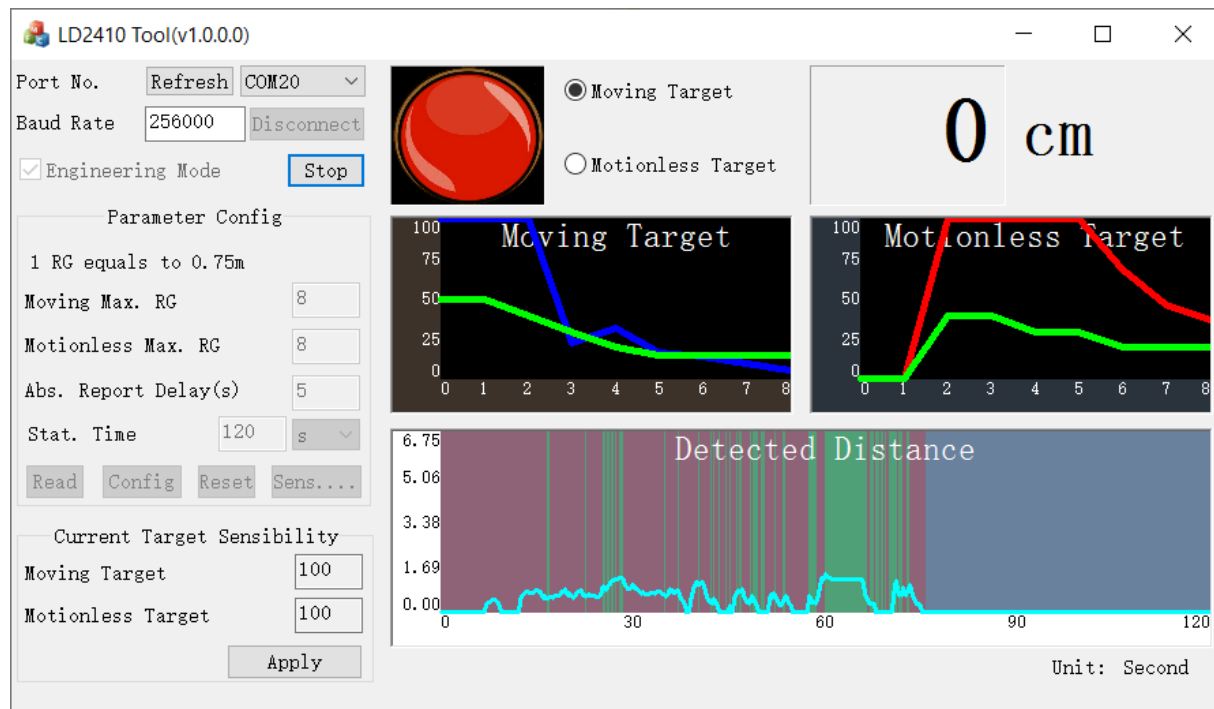


	Infrared solution	Visual solution	Ultrasonic wave	Lidar	Millimeter wave radar
Application flexibility	●	●	●	●	●
Resistance to environmental influences (weather light, etc.)	●	●	●	●	●
Detection speed	●	●	●	●	●
Detection accuracy	●	●	●	●	●
Resolution	●	●	●	●	●
Directionality	●	●	●	●	●
Detection distance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

● Good ● Common ● Weak

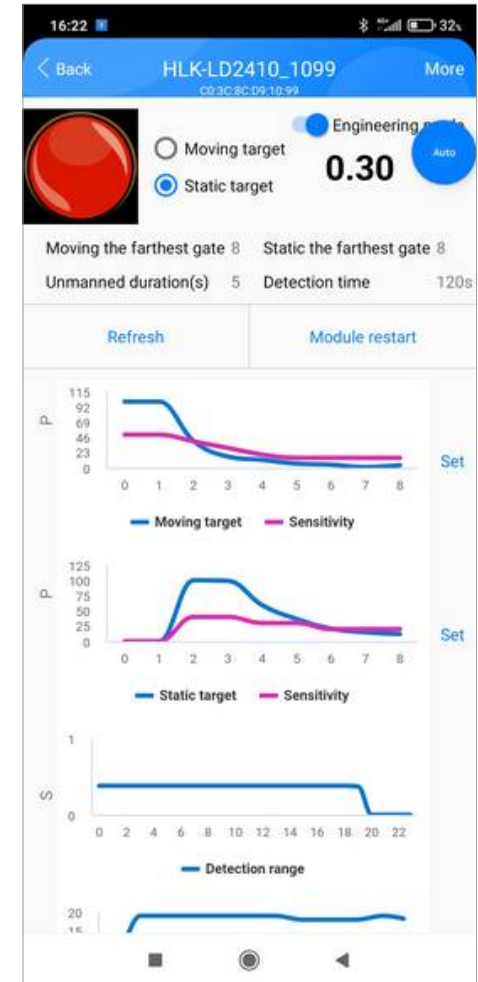
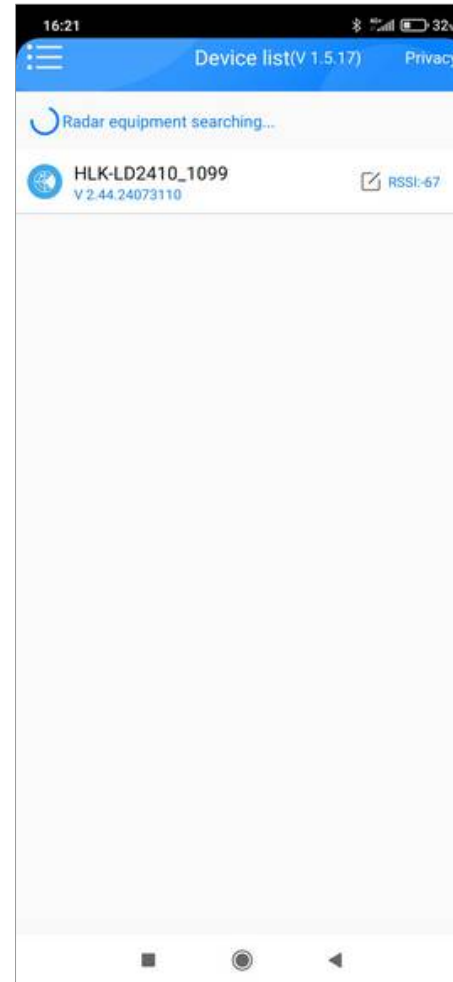
LD2410 Tool – PC alkalmazás

- ❖ 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 vezérel a beállításokat



HLK Radar Tool – 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

A MyLD2410 programkönyvtár

- ❖ Arduino környezetben a [MyLD2410 könyvtár](#) segítségével kezelhetjük a **HLK-2410C** szenzort (a szűkszavú dokumentáció [itt található](#))
- ❖ 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 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 mozgá célpontok közt 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

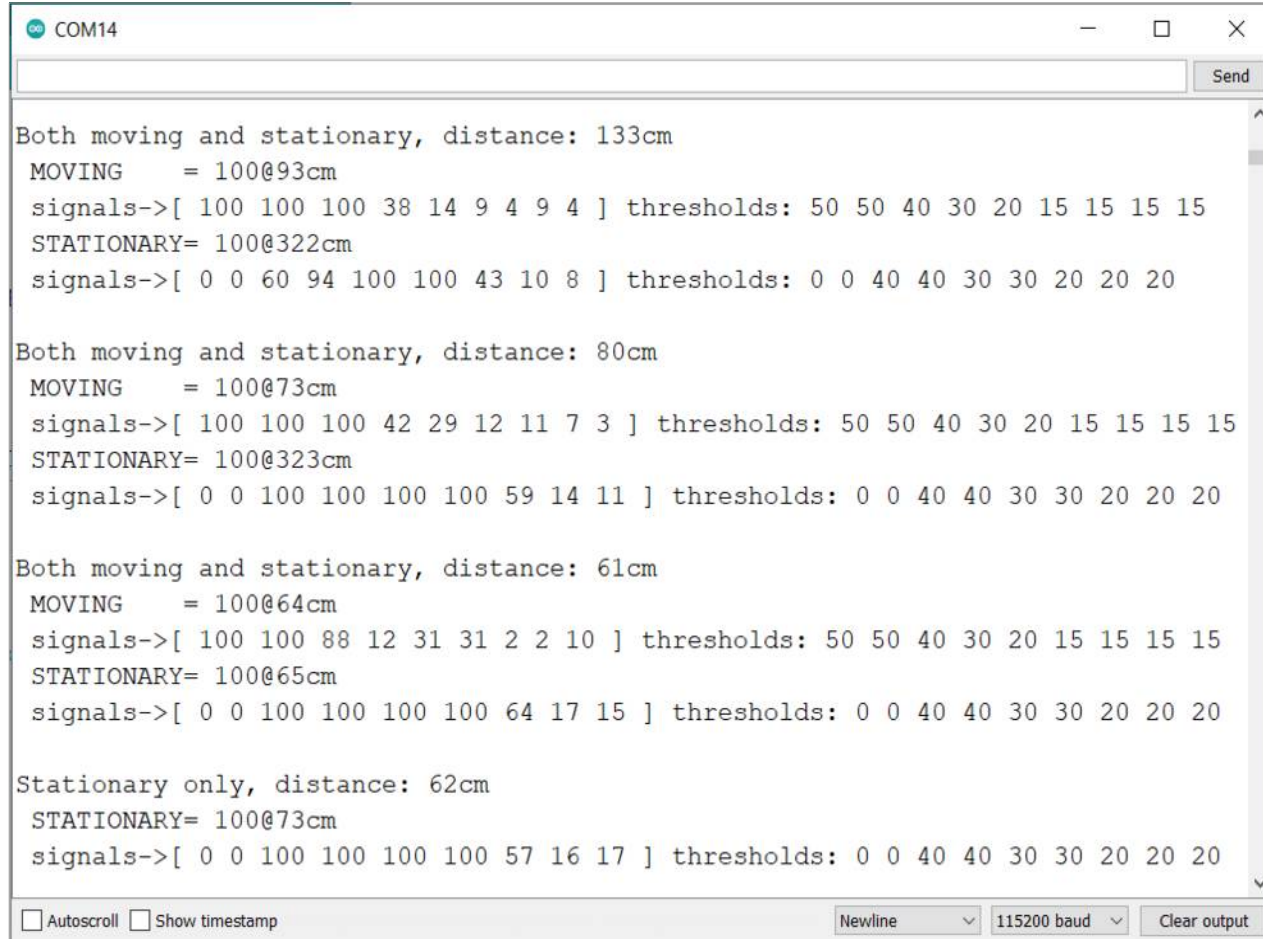
```
#include <MyLD2410.h>
MyLD2410 sensor(Serial1);
#define CONTROL_PIN 25 // Vezérlő kimenet

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



```
COM14
Both moving and stationary, distance: 133cm
MOVING    = 100@93cm
signals->[ 100 100 100 38 14 9 4 9 4 ] thresholds: 50 50 40 30 20 15 15 15 15
STATIONARY= 100@322cm
signals->[ 0 0 60 94 100 100 43 10 8 ] thresholds: 0 0 40 40 30 30 20 20 20

Both moving and stationary, distance: 80cm
MOVING    = 100@73cm
signals->[ 100 100 100 42 29 12 11 7 3 ] thresholds: 50 50 40 30 20 15 15 15 15
STATIONARY= 100@323cm
signals->[ 0 0 100 100 100 100 59 14 11 ] thresholds: 0 0 40 40 30 30 20 20 20

Both moving and stationary, distance: 61cm
MOVING    = 100@64cm
signals->[ 100 100 88 12 31 31 2 2 10 ] thresholds: 50 50 40 30 20 15 15 15 15
STATIONARY= 100@65cm
signals->[ 0 0 100 100 100 100 64 17 15 ] thresholds: 0 0 40 40 30 30 20 20 20

Stationary only, distance: 62cm
STATIONARY= 100@73cm
signals->[ 0 0 100 100 100 100 57 16 17 ] thresholds: 0 0 40 40 30 30 20 20 20

Autoscroll Show timestamp Newline 115200 baud Clear output
```


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

movingTargetSignal() movingTargetDistance()

statusString()

detectedDistance()

getMovingSignals()

Both moving and stationary, distance: 57cm

getMovingThresholds()

MOVING = 100@67cm

signals->[87 100 51 14 8 10 7 8 3] thresholds: 50 50 40 30 20 15 15 15 15

STATIONARY= 100@73cm

signals->[0 0 100 51 58 38 13 10 7] thresholds: 0 0 40 40 30 30 20 20 20

getStationarySignals()

getStationaryThresholds()

movingTargetSignal() movingTargetDistance()

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`)

MyLD2040 mintapéldák

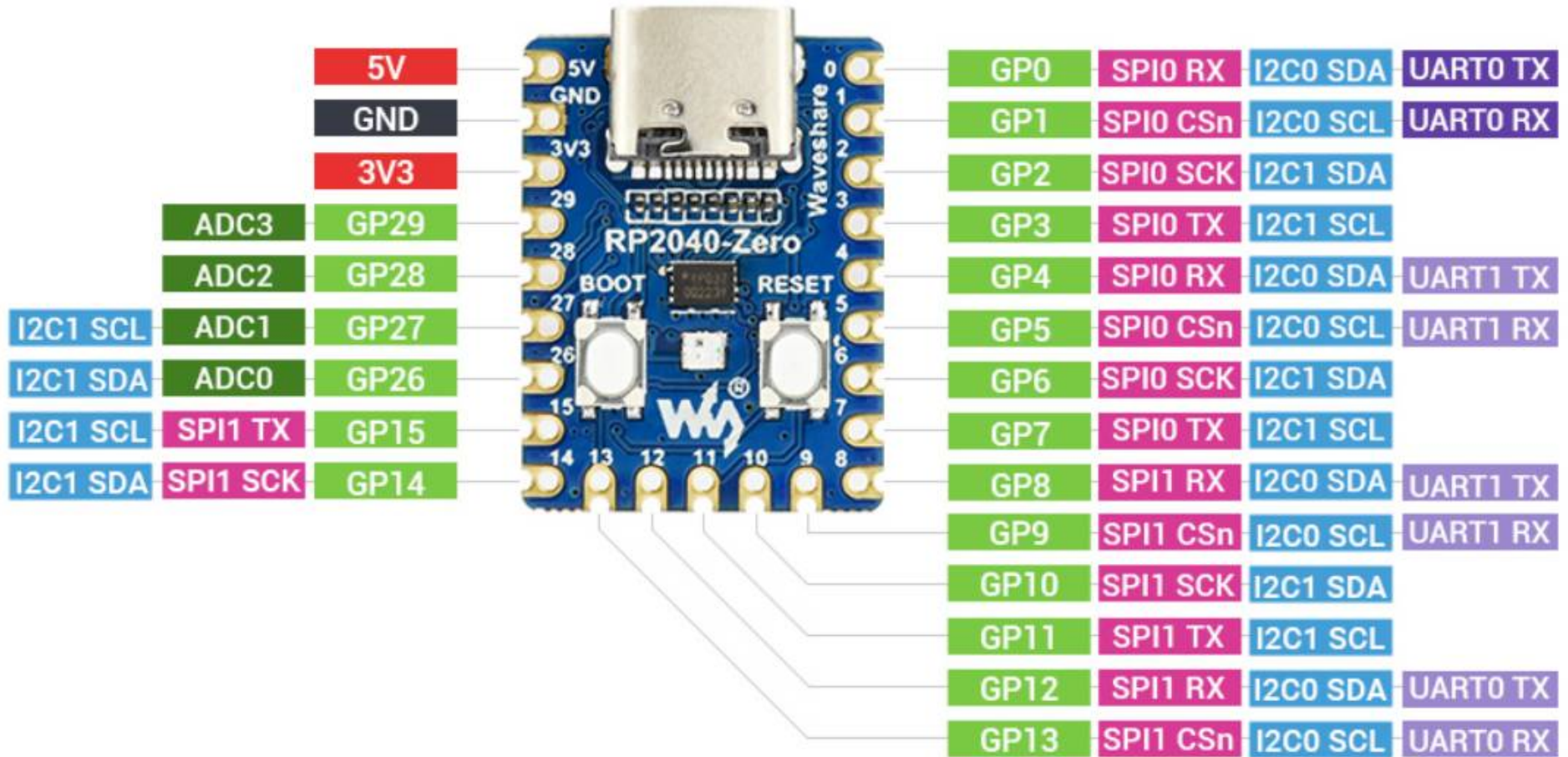
- ❖ A **MyLD2040** programkönyvtár az alábbi mintapéldákat tartalmazza:
 - **auto_thresholds** – az automatikus küszöbbeállítást mutatja be
 - **factory_reset** – gyári alapállapotba állítja a szenzor paramétereit
 - **modify_parameters** – az érzékelő paramétereinek beállítása/módosítása
 - **print_parameters** – kilistázza a szenzor paramétereit
 - **sensor_data** – a mért adatok kiírása (lényegében ez az **enhanced_demo.ino**)
 - **set_baud_rate** – a szenzor soros port sebességének beállítása
 - **set_bt_password** – a Bluetooth jelszavána beállítása/törlése
- ❖ A fenti mintaprogramokat az **RP2040** mikrovezérlőre csak akkor tudjuk lefordítani, ha az alábbi változtatásokat elvégezzük a programokban:

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

```
...
```

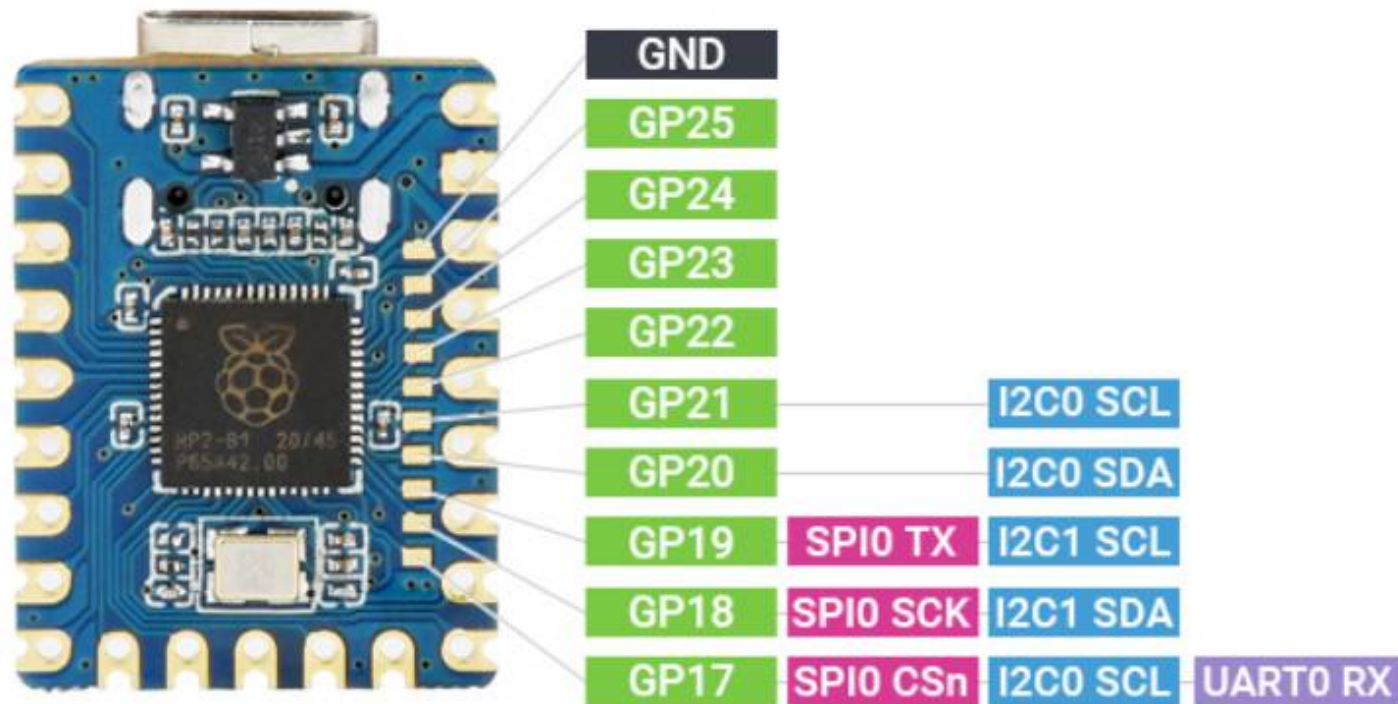
```
#if defined(ARDUINO_XIAO_ESP32C3) || defined(ARDUINO_XIAO_ESP32C6) || defined(ARDUINO_SAMD_NANO_33_IOT) ||  
defined(ARDUINO_AVR_LEONARDO) || defined(ARDUINO_ARCH_RP2040)
```


A kivezetések lábkiosztása



További kivezetések

- ❖ Néhány kivezetés a kártya hátoldalán van kivezelve



WS2812 RGB LED
used pin

GP16

DIN