

Ezt mondtad:

FM rádióvevőt szeretnék csinálni RDA5807M használatával.

A ChatGPT ezt mondta:

Nagyszerű választás! Az RDA5807M egy egyszerűen kezelhető FM rádióvevő chip, amely I2C interfészen keresztül vezérelhető. Van már Arduino alaplapod, vagy az is kérdéses, hogy melyik lenne a legjobb? Esetleg szeretnéd, hogy segítsék a kezdeti kód megírásában, vagy a kapcsolási rajz összeállításában? 😊

Ezt mondtad:

Arduino mini, vagy azzal kompatibilis vezérlővel szeretném

A ChatGPT ezt mondta:

Értem! Az Arduino Mini egy kiváló választás, kompakt méretével ideális kisebb projektekhez. Az RDA5807M modullal való munkához az alábbi lépések lesznek szükségesek:

Kapcsolási rajz:

Csatlakoztasd az RDA5807M modul SDA (adat) és SCL (órajel) lábait az Arduino megfelelő I2C lábaira (A4 és A5 az Arduino Mini esetében).

A modul tápellátását biztosítsd a 3.3V-os vagy 5V-os kimenetről (a modul kompatibilis mindkettővel).

GND-t (föld) kösd össze az Arduino GND lábával.

Hangszóró vagy fejhallgató csatlakoztatásához használd az RDA5807M audio kimeneti csatlakozásait.

Könyvtár hozzáadása:

Az RDA5807M-hez van egy népszerű Arduino könyvtár, amely megkönnyíti a vezérlést. A könyvtár neve például RDA5807. Telepítheted az Arduino IDE-ben a Library Manager segítségével.

Kezdeti kód:

Íme egy egyszerű példa, amely inicializálja a modult és beállítja az FM frekvenciát:

```
#include <RDA5807.h>
RDA5807 radio;

void setup() {
  Serial.begin(9600);          // Soros monitorhoz, hibakereséshez
  radio.init();                // Rádió inicializálása
  radio.setFrequency(101.1);   // FM frekvencia beállítása (pl. 101.1 MHz)
  Serial.println("Rádió indítva!");
}

void loop() {
  // További funkciók, például hangerő-szabályozás vagy csatornaváltás, itt adhatók hozzá
}
```

Tesztelés:

Töltsd fel a kódot az Arduino Mini-re, csatlakoztasd az RDA5807M-et, és ellenőrizd, hogy működik-e a rádióvevő.

Ezt mondtad:

Szeretném az FM rádiót kibővíteni I2C-n 7szegmenses, 4 digitos kijelzővel. Ezen felváltva jelenne meg a vételi frekvencia és a pontos idő.

A ChatGPT ezt mondta:

Nagyon jó ötlet, ez még látványosabbá és praktikusabbá teszi a projektet! Az alábbiakat kell figyelembe venni a bővítéshez:

1. 7 szegmenses kijelző kiválasztása

Az TM1637 vagy HT16K33 chipet használó kijelzők népszerűek, mivel egyszerűen vezérelhetők I2C-n vagy más interfészen keresztül. Az I2C-kompatibilis kijelzőhöz:

Az SDA és SCL vonalakat oszd meg az RDA5807M modullal (I2C támogat több eszközt egy buszon).

Győződj meg arról, hogy az I2C-címek nem ütköznek.

2. Kapcsolási rajz

A kijelző I2C lábait (SDA, SCL) ugyanazokra a lábakra kell csatlakoztatni, mint az RDA5807M esetében (Arduino Mini-nél A4 = SDA, A5 = SCL). Tápellátásként használhatsz 3.3V vagy 5V kimenetet, a kijelző specifikációjától függően.

3. Kódlogika

A kijelzőn felváltva szeretnéd megjeleníteni a frekvenciát és az időt. Ehhez egy időzítőt használsz, amely például 2 másodpercenként vált a két adat között.

Alapvető kódvázlat:

```
#include <Wire.h>
#include <RDA5807.h>
#include <TM1637Display.h> // Ha TM1637 kijelzőt használsz

RDA5807 radio;
TM1637Display display(2, 3); // TM1637 CLK és DIO lábak
unsigned long lastUpdate = 0;
bool showFrequency = true;

void setup() {
  radio.init();
  radio.setFrequency(101.1); // Példa frekvencia
  display.setBrightness(7); // Maximális fényerő
}

void loop() {
  unsigned long currentMillis = millis();
  // 2 másodpercenként váltás
  if (currentMillis - lastUpdate >= 2000) {
    lastUpdate = currentMillis;
    showFrequency = !showFrequency;

    if (showFrequency) {
      int frequency = (int)(radio.getFrequency() * 10); // Pl. 101.1 MHz → 1011
      display.showNumberDecEx(frequency, 0b01000000, true); // Pont a megfelelő helyre
    } else {
      // Példa: az időt mutatjuk
      int hours = 12; // Cseréld a valódi időre!
      int minutes = 34;
      display.showNumberDecEx(hours * 100 + minutes, 0b01000000, true);
    }
  }
}
```

4. Pontosság és valós idejű óra (RTC)

Ha az idő megjelenítését pontosabbá szeretnéd tenni, érdemes egy RTC modult (például DS3231 vagy DS1307) hozzáadni a projekthez. Ez szintén I2C-n kommunikál, így könnyen integrálható.

Ezt mondtad:

Add hozzá a DS1307 modult.

A ChatGPT ezt mondta:

A DS1307 RTC modul egyszerű kiegészítés, mivel pontos időt biztosít, és könnyen integrálható az I2C-buszhoz. Íme, hogyan illesztheted a rendszerbe:

Kapcsolási rajz bővítése

DS1307 I2C csatlakoztatása:

SDA (adatvonal): Csatlakoztasd az Arduino Mini A4-es lábához.

SCL (órjel): Csatlakoztasd az Arduino Mini A5-ös lábához.

VCC: Csatlakoztasd a 5V-os kimenethez.

GND: Csatlakoztasd a földhöz (GND).

Megosztott I2C-busz: Az RDA5807M, a 7 szegmenses kijelző, és a DS1307 mind ugyanazon a buszon lesz, ezért figyelj az I2C-címekre:

DS1307 alapértelmezett címe: 0x68.

Az RDA5807M-hez és a kijelzőhöz az adatlap alapján állítsd be a címeket, ha szükséges.

Kód frissítése

A DS1307-hez használd a RTCLib könyvtárat, amely könnyen kezelhető, és kompatibilis a legtöbb Arduino-val.

Kódvázlat:

```
#include <Wire.h>
#include <RDA5807.h>
#include <TM1637Display.h>
#include <RTCLib.h> // DS1307 könyvtár

RDA5807 radio;
TM1637Display display(2, 3); // TM1637 CLK és DIO lábak
RTC_DS1307 rtc;           // DS1307 óra
unsigned long lastUpdate = 0;
bool showFrequency = true;

void setup() {
  Serial.begin(9600);
  // Rádió inicializálás
  radio.init();
  radio.setFrequency(101.1); // Példa frekvencia
  // Kijelző inicializálás
  display.setBrightness(7); // Maximális fényerő
  // RTC inicializálás
  if (!rtc.begin()) {
    Serial.println("Nem található az RTC modul!");
    while (1); // Állj meg, ha nincs RTC
  }
  if (!rtc.isrunning()) {
    Serial.println("Az RTC nem fut! Indítás...");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); // Állítsd be az időt a fordítás időpontjára
  }
}

void loop() {
  unsigned long currentMillis = millis();
  // 2 másodpercenként váltás
  if (currentMillis - lastUpdate >= 2000) {
    lastUpdate = currentMillis;
    showFrequency = !showFrequency;
  }

  if (showFrequency) {
    // Frekvencia megjelenítése
    int frequency = (int)(radio.getFrequency() * 10); // Példa: 101.1 MHz → 1011
  }
}
```

```
display.showNumberDecEx(frequency, 0b01000000, true);
} else {
    // Idő megjelenítése
    DateTime now = rtc.now();
    int timeToDisplay = now.hour() * 100 + now.minute(); // Óra:Perc formátum
    display.showNumberDecEx(timeToDisplay, 0b01000000, true); // Pont az óra és perc között
}
}
}
```

További funkciók

Idő manuális beállítása: Ha egyedi időt szeretnél megadni, az `rtc.adjust(DateTime(év, hónap, nap, óra, perc, másodperc))` metódussal megteheted.

Dátum megjelenítése: Ha szeretnéd, hogy időnként a dátum is megjelenjen, a kódhoz adhatsz egy harmadik állapotot.

Ezt mondtad:

Bővítsük automatikus kereséssel. Ha kellően erős, sztereó állomást talál, a frekvenciát mentse el EEPROMba.

A ChatGPT ezt mondta:

Az automatikus keresés és a frekvencia mentése az EEPROM-ba remek funkciók egy FM rádióhoz.

Az alábbiakban bemutatom, hogyan integrálhatod ezt a funkciót a meglévő rendszerbe:

Kapcsolási és kódmódosítások

Automatikus keresés beállítása: Az RDA5807M modul támogatja az automatikus keresést, amely a megfelelő könyvtárban egyszerűen meghívható.

Sztereó állomás ellenőrzése: Az RDA5807M jelezheti, ha sztereó vétel érhető el. Ezt az állapotát a könyvtár segítségével lekérheted.

Frekvencia mentése EEPROM-ba: Az Arduino beépített EEPROM.h könyvtára lehetővé teszi adatok tárolását.

Példakód az automatikus kereséshez

```
#include <Wire.h>
#include <RDA5807.h>
#include <TM1637Display.h>
#include <RTCLib.h>
#include <EEPROM.h>

RDA5807 radio;
TM1637Display display(2, 3); // TM1637 CLK és DIO lábak
RTC_DS1307 rtc;           // DS1307 óra
unsigned long lastUpdate = 0;
bool showFrequency = true;

void setup() {
  Serial.begin(9600);
  // Rádió inicializálása
  radio.init();
  radio.setVolume(8); // Hangerő alapérték
  display.setBrightness(7);
  // RTC inicializálása

  if (!rtc.begin() || !rtc.isrunning()) {
    Serial.println("RTC nem elérhető vagy nem fut!");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); // Beállítás fordítás idejére
  }
  // Automatikus keresés indítása
  findAndSaveStation();
}

void loop() {
  unsigned long currentMillis = millis();

  // Felváltott kijelzés frekvencia és idő között
  if (currentMillis - lastUpdate >= 2000) {
    lastUpdate = currentMillis;
    showFrequency = !showFrequency;
    if (showFrequency) {
      // Frekvencia megjelenítése
      int savedFreq = EEPROM.read(0) * 10 + EEPROM.read(1); // Példa: 101.1 MHz → 1011
      display.showNumberDecEx(savedFreq, 0b01000000, true);
    }
  }
}
```

```
} else {  
    // Idő megjelenítése  
    DateTime now = rtc.now();  
    int timeToDisplay = now.hour() * 100 + now.minute();  
    display.showNumberDecEx(timeToDisplay, 0b01000000, true);  
}  
}  
}
```

```
void findAndSaveStation() {  
    int startFreq = 870; // Kezdő frekvencia (87.0 MHz)  
    int endFreq = 1080; // Végző frekvencia (108.0 MHz)  
    for (int freq = startFreq; freq <= endFreq; freq += 1) {  
        radio.setFrequency(freq / 10.0); // Frekvencia MHz-ben  
        delay(200); // Várakozás az állomás stabilizálásához  
  
        if (radio.isStereo() && radio.getRssi() > 20) { // Sztereó és erős jel  
            Serial.print("Állomás megtalálva: ");  
            Serial.println(freq / 10.0);  
            // Frekvencia mentése EEPROM-ba  
            EEPROM.write(0, freq / 10); // Tízes érték  
            EEPROM.write(1, freq % 10); // Egység érték  
            break; // Állomás megtalálva, kilépés a ciklusból  
        }  
    }  
}
```

Kód működése:

Automatikus keresés:

A `findAndSaveStation()` függvény végigmegy az FM sávban (87.0 MHz–108.0 MHz), és megkeresi az első sztereó, erős állomást.

Ha talál egy megfelelő állomást, a frekvenciát elmenti az EEPROM-ba.

Mentett frekvencia betöltése:

A mentett frekvencia a loop során jelenik meg a kijelzőn.

EEPROM kezelés:

Az EEPROM 0. és 1. címein tároljuk a frekvencia tizedes és egység értékeit.

Bővítési lehetőségek

Több állomás mentése: Az EEPROM több címére is tárolhatsz frekvenciákat, például csatornaváltás funkcióhoz.

Felhasználói vezérlés: Nyomógombbal indítható automatikus keresés vagy mentett állomások közötti léptetés.

Ezt mondtad:

Több állomás, több címre és a felhasználói vezérlés. kérem

A ChatGPT ezt mondta:

A bővítés során beépítjük a több állomás mentését, valamint a felhasználói vezérlést a mentett állomások közötti váltáshoz. Íme, hogyan valósíthatod meg:

Kapcsolási rajz frissítése

Gombok hozzáadása:

Keresés indítása: Egy gomb a frekvenciák automatikus kereséséhez és mentéséhez.

Állomásváltás: Egy másik gomb a mentett állomások közötti váltáshoz.

Csatlakoztasd ezeket az Arduino két digitális lábára (például D4 és D5), és mindkettőt kösd 10kΩ-os felhúzóellenállással.

Frissített kód

Főbb funkciók:

Több állomás tárolása (pl. 10 állomás, külön EEPROM címeken).

Nyomógombbal indítható keresés és váltás a mentett állomások között.

```

#include <Wire.h>
#include <RDA5807.h>
#include <TM1637Display.h>
#include <RTCLib.h>
#include <EEPROM.h>

RDA5807 radio;
TM1637Display display(2, 3); // TM1637 CLK és DIO lábak
RTC_DS1307 rtc;
#define SEARCH_BUTTON 4 // Gomb az automatikus kereséshez
#define NEXT_BUTTON 5 // Gomb a következő állomásra lépéshez
#define MAX_STATIONS 10 // Maximális tárolható állomások száma
int currentStation = 0; // Az aktuálisan kiválasztott állomás indexe
int savedStations = 0; // Mentett állomások száma

void setup() {
  Serial.begin(9600);
  pinMode(SEARCH_BUTTON, INPUT_PULLUP);
  pinMode(NEXT_BUTTON, INPUT_PULLUP);
  // Rádió és kijelző inicializálása
  radio.init();
  display.setBrightness(7);
  // RTC inicializálása

  if (!rtc.begin() || !rtc.isrunning()) {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
  // Betöltött állomások száma
  savedStations = EEPROM.read(0); // Az első cím tartalmazza a mentett állomások számát
}

void loop() {
  // Automatikus keresés indítása
  if (digitalRead(SEARCH_BUTTON) == LOW) {
    delay(200); // Gombzaj szűrés
    findAndSaveStations();
  }
  // Következő állomásra váltás
  if (digitalRead(NEXT_BUTTON) == LOW) {
    delay(200); // Gombzaj szűrés
  }
}

```

```

if (savedStations > 0) {
    currentStation = (currentStation + 1) % savedStations;
    int freq = readStationFromEEPROM(currentStation);
    radio.setFrequency(freq / 10.0); // Állítsd be az új frekvenciát
    display.showNumberDecEx(freq, 0b01000000, true);
}
}
}
// Automatikus állomáskeresés és mentés
void findAndSaveStations() {
    int startFreq = 870; // Kezdő frekvencia
    int endFreq = 1080; // Végző frekvencia
    int stationCount = 0;

    for (int freq = startFreq; freq <= endFreq && stationCount < MAX_STATIONS; freq += 1) {
        radio.setFrequency(freq / 10.0);
        delay(200);

        if (radio.isStereo() && radio.getRssi() > 20) {
            Serial.print("Állomás megtalálva: ");
            Serial.println(freq / 10.0);
            saveStationToEEPROM(stationCount, freq);
            stationCount++;
        }
    }
    // Állomások számának mentése
    EEPROM.write(0, stationCount);
    savedStations = stationCount;

    if (stationCount > 0) {
        Serial.println("Állomások mentve!");
        currentStation = 0;
        radio.setFrequency(readStationFromEEPROM(0) / 10.0);
    } else {
        Serial.println("Nem találtunk állomást!");
    }
}
// Frekvencia mentése EEPROM-ba
void saveStationToEEPROM(int index, int freq) {
    int addr = 1 + index * 2; // Címzés: 1. címtől kezdve, 2 bájtot használva állomásonként

```

```
EEPROM.write(addr, freq / 10);    // Tízes helyiérték
EEPROM.write(addr + 1, freq % 10); // Egység helyiérték
}
// Állomás betöltése EEPROM-ból
int readStationFromEEPROM(int index) {
    int addr = 1 + index * 2;
    int tens = EEPROM.read(addr);
    int units = EEPROM.read(addr + 1);
    return tens * 10 + units;
}
```

Funkciók részletezése

Automatikus keresés:

Az állomásokat 87.0–108.0 MHz tartományban keresi.

Az első 10 erős, sztereó állomást menti.

Állomás mentése és betöltése:

Az EEPROM első címén tárolja a mentett állomások számát.

A mentett állomások frekvenciáit páros címeken (2 bájtos formátumban) tárolja.

Felhasználói vezérlés:

Keresés gomb: Automatikusan keres és ment állomásokat.

Következő állomás gomb: Váltás a mentett állomások között.

Bővítési lehetőségek

Mentett állomások törlése: Gombbal vagy menüből inicializálhatod az EEPROM-t.

Kijelzőn állomásszám megjelenítése: Jelezheted az aktuális állomás sorszámát.

További állomások kezelése: Ha több mint 10 állomást szeretnél tárolni, az EEPROM használatát bővítheted.

Ezt mondtad:

adjunk hozzá egy 2/20 LCD kijelzőt. Ezen mutatja az állomás nevét és az RDS üzenetet.

A ChatGPT ezt mondta:

A 2x16 karakteres LCD kijelzővel könnyen megjelenítheted az FM állomás nevét és az RDS üzenetet.

Az alábbiakban bemutatom a hardveres és szoftveres integráció lépéseit.

Kapcsolási rajz frissítése

I2C LCD modul csatlakoztatása:

SDA: Arduino Mini A4 lábához csatlakoztasd.

SCL: Arduino Mini A5 lábához csatlakoztasd.

VCC: 5V-hoz.

GND: Földhöz.

I2C-cím figyelembevétele:

A kijelző általában 0x27 vagy 0x3F címen érhető el. Ellenőrizheted egy I2C scanner kóddal.

Szükséges könyvtárak

LiquidCrystal_I2C: Az LCD vezérléséhez.

RDA5807: Az FM rádió és RDS adatok kezeléséhez.

Telepítsd ezeket az Arduino IDE Library Manager segítségével.

Frissített kód LCD támogatással

Kódvázlat:

```
#include <Wire.h>
#include <RDA5807.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
#include <RTCLib.h>

RDA5807 radio;
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C cím: 0x27, méret: 20x2
RTC_DS1307 rtc;

#define SEARCH_BUTTON 4
#define NEXT_BUTTON 5
#define MAX_STATIONS 10

int currentStation = 0;
int savedStations = 0;

void setup() {
  Serial.begin(9600);
  pinMode(SEARCH_BUTTON, INPUT_PULLUP);
  pinMode(NEXT_BUTTON, INPUT_PULLUP);

  // Inicializációk
  radio.init();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("FM Radio Indul...");

  if (!rtc.begin() || !rtc.isrunning()) {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  // EEPROM betöltés
  savedStations = EEPROM.read(0);

  delay(2000);
}
```

```

void loop() {
  // Gombnyomások kezelése
  if (digitalRead(SEARCH_BUTTON) == LOW) {
    delay(200);
    findAndSaveStations();
  }

  if (digitalRead(NEXT_BUTTON) == LOW) {
    delay(200);
    nextStation();
  }

  // LCD frissítése RDS adatokkal
  updateLCD();
}

// Automatikus állomáskeresés és mentés
void findAndSaveStations() {
  int startFreq = 870;
  int endFreq = 1080;
  int stationCount = 0;

  for (int freq = startFreq; freq <= endFreq && stationCount < MAX_STATIONS; freq += 1) {
    radio.setFrequency(freq / 10.0);
    delay(200);

    if (radio.isStereo() && radio.getRssi() > 20) {
      saveStationToEEPROM(stationCount, freq);
      stationCount++;
    }
  }

  EEPROM.write(0, stationCount);
  savedStations = stationCount;
  currentStation = 0;

  if (stationCount > 0) {
    int firstFreq = readStationFromEEPROM(0);
    radio.setFrequency(firstFreq / 10.0);
  }
}

```

```

}

// Következő állomás
void nextStation() {
    if (savedStations > 0) {
        currentStation = (currentStation + 1) % savedStations;
        int freq = readStationFromEEPROM(currentStation);
        radio.setFrequency(freq / 10.0);
    }
}

```

```

// Állomás mentése EEPROM-ba
void saveStationToEEPROM(int index, int freq) {
    int addr = 1 + index * 2;
    EEPROM.write(addr, freq / 10);
    EEPROM.write(addr + 1, freq % 10);
}

```

```

// Állomás betöltése EEPROM-ból
int readStationFromEEPROM(int index) {
    int addr = 1 + index * 2;
    int tens = EEPROM.read(addr);
    int units = EEPROM.read(addr + 1);
    return tens * 10 + units;
}

```

```

// LCD frissítése RDS adatokkal
void updateLCD() {
    char rdsText[9];
    radio.getRdsText(rdsText, 8);
    rdsText[8] = '\0'; // Null-terminálás

```

```

    lcd.setCursor(0, 0);
    lcd.print("Frekvencia: ");
    lcd.print(radio.getFrequency());
    lcd.print(" MHz");

```

```

    lcd.setCursor(0, 1);
    if (strlen(rdsText) > 0) {
        lcd.print("RDS: ");

```

```
    lcd.print(rdsText);  
  } else {  
    lcd.print("RDS nem elerhető");  
  }  
}
```

Funkciók részletezése

LCD kijelző frissítése:

A felső soron a frekvencia jelenik meg.

Az alsó soron az RDS szöveg vagy üzenet látható (pl. állomás neve, műsor címe).

RDS támogatás:

A RDA5807 modul képes az RDS szöveg fogadására, amelyet a `getRdsText()` függvénnyel lehet lekérni.

Felhasználói vezérlés:

Gombokkal indítható az automatikus keresés és állomásváltás.

További bővítési lehetőségek

Dátum és idő megjelenítése: Időnként válthatsz az RDS és az aktuális idő között.

Hangolás finomítása: Forgó enkóder vagy további gombok segítségével kézi hangolást is hozzáadhatsz.

Ha további segítségre van szükséged a kijelző vagy az RDS implementálásában, jelezd bátran! 😊

Ezt mondtad:

Finomhangolás forgó enkóderrel.

A ChatGPT ezt mondta:

A forgó enkóder használatával pontosan és intuitívan állíthatod be a rádió frekvenciáját. Az alábbiakban bemutatom, hogyan adhatod hozzá a forgó enkódert a rendszerhez.

Kapcsolási rajz frissítése

Forgó enkóder csatlakoztatása:

CLK: Arduino egyik digitális lába, például D6.

DT: Arduino másik digitális lába, például D7.

SW (nyomógomb): Egy további digitális láb, például D8.

+ és GND: 5V és föld.

10kΩ-os felhúzó ellenállások:

Az enkóder kapcsoló lábaira felhúzó ellenállást kell csatlakoztatni, ha a modulon nincs beépített.

Kód frissítése forgó enkóder támogatással

A forgó enkóder jeleit kell feldolgozni a frekvencia növelésére vagy csökkentésére.

Kód:

```
#include <Wire.h>
#include <RDA5807.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
#include <RTClib.h>

RDA5807 radio;
LiquidCrystal_I2C lcd(0x27, 20, 2);
RTC_DS1307 rtc;

#define ENCODER_CLK 6
#define ENCODER_DT 7
#define ENCODER_SW 8

int currentFrequency = 875; // Kezdő frekvencia (87.5 MHz)
const int minFrequency = 870; // Minimum frekvencia (87.0 MHz)
const int maxFrequency = 1080; // Maximum frekvencia (108.0 MHz)

int lastEncoderState = HIGH;

void setup() {
  Serial.begin(9600);

  pinMode(ENCODER_CLK, INPUT);
  pinMode(ENCODER_DT, INPUT);
  pinMode(ENCODER_SW, INPUT_PULLUP);

  radio.init();
  lcd.init();
  lcd.backlight();

  lcd.setCursor(0, 0);
  lcd.print("FM Radio Indul...");
  delay(2000);

  radio.setFrequency(currentFrequency / 10.0);
}

void loop() {
```

```

handleEncoder();
updateLCD();
}

// Forgó enkóder kezelés
void handleEncoder() {
  int currentState = digitalRead(ENCODER_CLK);

  if (currentState != lastEncoderState) {
    if (digitalRead(ENCODER_DT) != currentState) {
      currentFrequency += 1; // Növelés
    } else {
      currentFrequency -= 1; // Csökkentés
    }

    // Frekvencia határok között tartása
    if (currentFrequency < minFrequency) currentFrequency = minFrequency;
    if (currentFrequency > maxFrequency) currentFrequency = maxFrequency;

    radio.setFrequency(currentFrequency / 10.0);
  }
  lastEncoderState = currentState;

  // Gombnyomás ellenőrzése
  if (digitalRead(ENCODER_SW) == LOW) {
    delay(200); // Gombzaj szűrés
    saveCurrentFrequency();
  }
}

// Frekvencia mentése az EEPROM-ba
void saveCurrentFrequency() {
  EEPROM.write(1, currentFrequency / 10); // Tízes helyiérték
  EEPROM.write(2, currentFrequency % 10); // Egység helyiérték
  lcd.setCursor(0, 1);
  lcd.print("Frekvencia mentve!");
  delay(2000);
}

// LCD frissítése

```

```
void updateLCD() {
  lcd.setCursor(0, 0);
  lcd.print("Frekvencia: ");
  lcd.print(currentFrequency / 10.0, 1);
  lcd.print(" MHz");

  char rdsText[9];
  radio.getRdsText(rdsText, 8);
  rdsText[8] = '\0'; // Null-terminálás

  lcd.setCursor(0, 1);
  if (strlen(rdsText) > 0) {
    lcd.print("RDS: ");
    lcd.print(rdsText);
  } else {
    lcd.print("RDS nincs");
  }
}
```

Funkciók részletezése

Forgó enkóder működése:

Az enkóder CLK és DT lábait figyeli a program, és a forgás irányától függően növeli vagy csökkenti a frekvenciát.

A frekvencia lépésköze: 0.1 MHz.

Nyomógomb funkció:

Az enkóder beépített gombját megnyomva a jelenlegi frekvenciát menti az EEPROM-ba.

LCD kijelző:

Folyamatosan frissül a frekvencia és az RDS szöveg.

További bővítési lehetőségek

Lépésköz állítása: A gomb nyomva tartásával választhatsz nagyobb frekvencialépést (pl. 1 MHz).

Állomás mentése gyorsgombbal: Külön gombbal is bővítheted az állomások mentését.