PROIECT

REȚELE DE SENZORI ȘI AD-HOC

Preluare si afisare date GPS

Îndrumător,

Ș.l. dr. ing. Lavic Alexandru

Studenți,

Mușină Silviu
Oprea Alexandru
Carează Denisa-Dumitrela

Cuprins

1. Introducere

Acest proiect propune dezvoltarea unui GPS-tracker, format dintr-un dispozitiv bazat pe Arduino pentru colectarea datelor GPS și o aplicație mobilă modernă pentru afișarea acestora în timp real.

Soluția utilizează biblioteca TinyGPS++ pentru interpretarea eficientă a datelor GPS la nivelul microcontrolerului, iar interfața aplicației mobile este realizată cu ajutorul React Native Expo, asigurând o experiență intuitivă și accesibilă utilizatorilor.

2. Programare Arduino si conectarea modulelor

2.1. Descriere

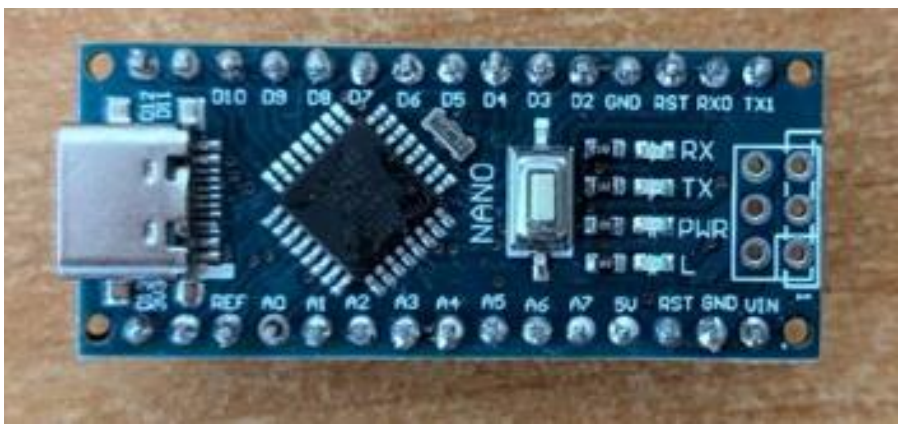
Sistemul integrează un microcontroler **Arduino Nano**, care coordonează funcționarea modulelor hardware, un modul GPS NEO-6M pentru recepționarea coordonatelor GPS și un modul Bluetooth HC-05 pentru transmiterea datelor către aplicația mobilă. Biblioteca TinyGPS++ este utilizată pentru procesarea datelor GPS, permițând extragerea precisă a coordonatelor de latitudine și longitudine.

2.2. Funcționalități

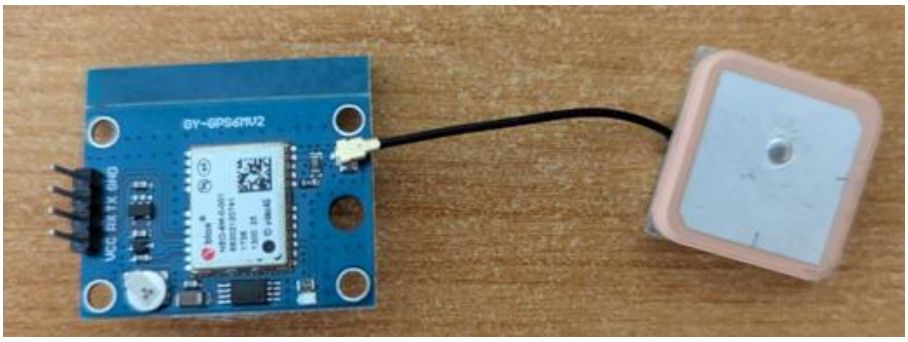
- *Recepționare date GPS*
  Arduino primește coordonatele GPS (latitudine și longitudine) prin intermediul modulului NEO-6M.
- *Transmitere date Bluetooth*
  Coordonatele GPS sunt trimise prin Bluetooth către aplicația mobilă.

2.3. Echipament utilizat

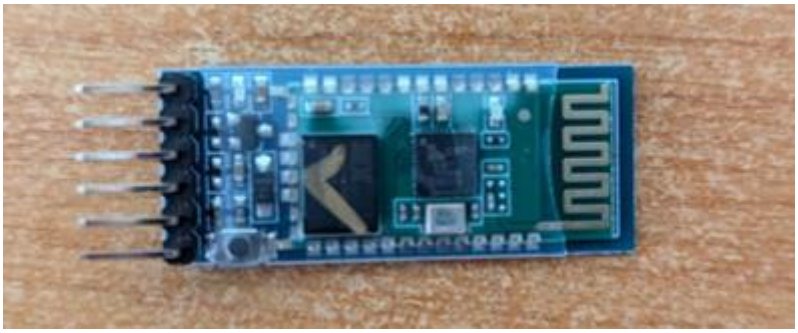
- Arduino NANO.



- Modul GPS NEO-6M este responsabil pentru recepționarea semnalelor GPS și transmiterea datelor către Arduino.



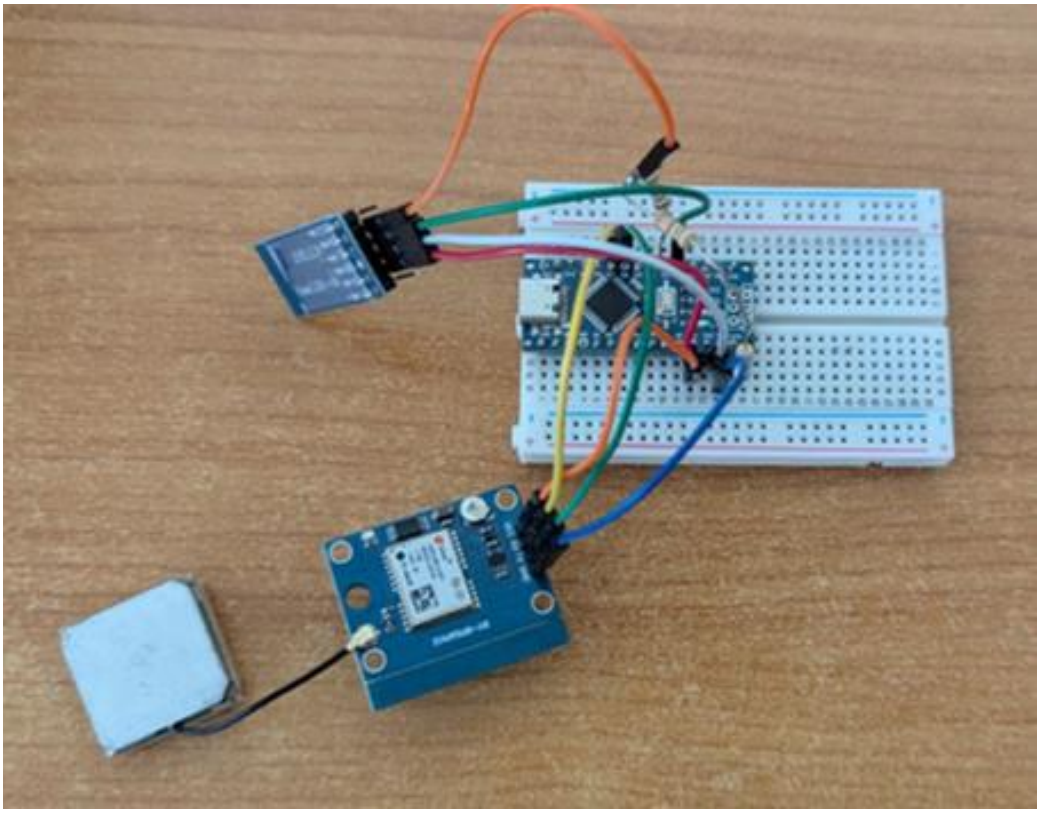
- Modul Bluetooth HC-05 este utilizat pentru comunicarea wireless cu dispozitive mobile.



2.4. Conectare module

Ambele module vor fi alimentate la 5V prin conectarea pinului VCC al fiecărui modul la pinul de 5V al Arduino Nano, iar pinul GND va fi conectat la masă (GND). Modulul HC-05 funcționează la o tensiune de 3.3V, dar dispune de un regulator de tensiune integrat pe pinul VCC, astfel încât poate fi alimentat în siguranță la 5V.

Pentru modulul GPS NEO-6M, conectăm pinul TX la pinul 6 și pinul RX la pinul 7 de pe Arduino. În cazul modulului HC-05, conectăm pinul RX la pinul 3 al Arduino, iar pinul TX la pinul 2, trecând printr-un divizor de tensiune. Divizorul de tensiune este necesar deoarece pinul TX al Arduino operează la 5V, iar pinul RX al modulului HC-05 suportă doar până la 3.3V.



2.5. Programare microcontroler

Pentru comunicare între modulele conectate este utilizata librăria *<SoftwareSerial.h>*, care va permite comunicare seriala pe alți pini decât cei standard (RX și TX pe Arduino). Decodarea si prelucrarea datelor GPS va fi efectuata cu ajutorul librăriei *<TinyGPS++.h>* pentru a afla informația despre longitudine si latitudine.

```
#include <SoftwareSerial.h>
#include <TinyGPS++.h>

// Pinii pentru conectarea Bluetooth HC-05
const int bluetoothTx = 2; // Conectat la RX pe HC-05
const int bluetoothRx = 3; // Conectat la TX pe HC-05

// Pinii pentru conectarea modulului GPS NEO-6M
const int gpsTx = 6;       // Conectat la TX pe GPS
const int gpsRx = 7;       // Conectat la RX pe GPS

SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);  // Configurare modul Bluetooth
SoftwareSerial gpsSerial(gpsTx, gpsRx);              // Configurare modul GPS
TinyGPSPlus gps;                                     // Obiect pentru TinyGPS++

float latitude, longitude;

void setup() {
  Serial.begin(9600);         // Conectare la serial monitor
  bluetooth.begin(9600);      // Initializare Bluetooth HC-05
  gpsSerial.begin(9600);      // Initializare GPS NEO-6M
```

```
  Serial.println("Așteptare semnal GPS...");
}

void loop() {
  // Citirea datelor de la GPS și actualizarea TinyGPS++
  while (gpsSerial.available() > 0) {
    gps.encode(gpsSerial.read());  // Transmiterea fiecărui caracter la TinyGPS++
  }

  // Verificăm dacă locația a fost actualizată
  if (gps.location.isUpdated()) {
    latitude = gps.location.lat();   // Latitudine
    longitude = gps.location.lng();  // Longitudine

    // Transmiterea datelor prin Bluetooth
    bluetooth.print(latitude, 6);
    bluetooth.print(",");
    bluetooth.print(longitude, 6);
    bluetooth.print("\n");

    // Opțional: Afișarea datelor pe serial monitor
    Serial.print("Latitudine: ");
    Serial.print(latitude, 6);
    Serial.print(", Longitudine: ");
    Serial.println(longitude, 6);

    delay(1000);  // Pauză între transmisii
  }
}
```

Creăm două obiecte SoftwareSerial pentru a comunica cu modulele Bluetooth și GPS și un obiect TinyGPSPlus pentru decodarea și procesarea datelor GPS. În funcția setup(), inițializăm comunicarea cu modulele la o viteză de 9600 band. Dacă dorim să transmitem datele și prin Serial Monitor, activăm și Serial.begin(9600).

În funcția loop(), verificăm constant dacă sunt disponibile date noi prin gpsSerial. Atunci când sunt disponibile, transmitem aceste date obiectului gps pentru procesare. Dacă locația a fost actualizată cu succes, extragem coordonatele de latitudine și longitudine și le transmitem prin Bluetooth. De asemenea, putem afișa datele în Serial Monitor, dacă dorim. La final, adăugăm o pauză de o secundă înainte de următoarea transmisie.

3. Aplicația mobilă (React Native Expo)

3.1. Descriere

Aplicația mobilă dezvoltată în React Native Expo permite utilizatorului să caute, să împerecheze și să se conecteze la un dispozitiv Bluetooth (Arduino) pentru a recepționa și afișa datele GPS. Aceasta folosește biblioteca react-native-bluetooth-classic pentru gestionarea comunicației Bluetooth.

3.2. Funcționalități

- *Căutare și împerechere cu dispozitive*
  Aplicația oferă utilizatorului posibilitatea de a descoperi dispozitive Bluetooth din apropiere și de a le împerechea pentru conectare ulterioară.
- *Conectarea la dispozitive împerecheate*
  Utilizatorul poate vizualiza lista dispozitivelor deja împerecheate și se poate conecta la acestea pentru a primi date în timp real.
- *Afișarea datelor GPS*
  După conectarea cu succes la Arduino, aplicația recepționează date GPS (latitudine și longitudine) și le afișează în interfață.

3.3. Tehnologii utilizate

- React Native Expo: pentru crearea interfeței de utilizare a aplicației mobile.
- react-native-bluetooth-classic: pentru gestionarea conexiunilor Bluetooth și recepționarea datelor de la dispozitive externe.

3.4. Componente principale de cod și explicații

1. **Gestionarea permisiunilor și deconectarea dispozitivului la închiderea aplicației:**

```
useEffect(() => {
  const requestPermissions = async () => {
    if (Platform.OS === "android") {
      await PermissionsAndroid.requestMultiple([
        PermissionsAndroid.PERMISSIONS.BLUETOOTH_SCAN,
        PermissionsAndroid.PERMISSIONS.BLUETOOTH_CONNECT,
        PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION,
      ]);
    }
  };
  requestPermissions();
  return () => {
    try {
      if (selectedDevice?.isConnected()) selectedDevice.disconnect();
```

```
      setSelectedDevice(null);
    } catch (error) {
      console.log(error);
    }
  };
}, []);
```

Aplicația gestionează automat permisiunile necesare pentru utilizarea Bluetooth și a locației pe dispozitivele Android, precum și deconectarea dispozitivului Bluetooth la închiderea aplicației. Aceste operațiuni sunt realizate în cadrul unui hook useEffect, care solicită permisiunile necesare la pornirea aplicației și curăță conexiunile la demontarea componentei.

2. **Căutarea dispozitivelor disponibile:**

```
const discoverDevices = async () => {
  if (isSearching) return;
  try {
    setIsSearching(true);
    const availableDevices = await RNBluetoothClassic.startDiscovery();
    const filteredDevices = availableDevices.filter(
      (item) => item.address !== item.name,
    );
    setDevices(filteredDevices);
    setIsSearching(false);
  } catch (error) {
    setIsSearching(false);
    console.error("Error discovering devices:", error);
  }
};
```

Funcția discoverDevices inițiază căutarea dispozitivelor Bluetooth disponibile din apropiere. La apelarea acestei funcții, aplicația verifică mai întâi dacă este deja în desfășurare o căutare de dispozitive, verificând starea isSearching. Dacă nu este deja activă o căutare, funcția pornește descoperirea dispozitivelor disponibile prin RNBluetoothClassic.startDiscovery().

Dispozitivele găsite sunt apoi filtrate pentru a elimina intrările fără un nume asociat, iar lista este actualizată în interfața aplicației, permițând utilizatorului să vadă dispozitivele disponibile. La finalul căutării, isSearching este setat înapoi la false, iar erorile sunt capturate și afișate în consolă.

3. **Împerecherea cu un dispozitiv:**

```
const pairDevice = async (device: BluetoothDevice) => {
  try {
    const isBonded = await RNBluetoothClassic.pairDevice(device.id);
    if (isBonded) {
```

```
    console.log(`Device ${device.name} paired successfully`);
   } else {
    console.log(`Failed to pair device ${device.name}`);
   }
  } catch (error) {
   console.error("Error pairing device:", error);
  }
};
```

Funcția pairDevice este utilizată pentru a împerechea aplicația cu un dispozitiv Bluetooth selectat de utilizator. Când funcția este apelată, încearcă să realizeze împerecherea cu dispozitivul indicat prin device.id folosind metoda RNBluetoothClassic.pairDevice(). În cazul în care împerecherea are succes, aplicația afișează un mesaj în consolă pentru a confirma reușita, altfel, un mesaj de eroare este afișat.

Împerecherea dispozitivului asigură o conexiune stabilă și securizată, permițând transferul de date între aplicație și Arduino, iar funcția gestionează eventualele erori pentru a evita întreruperea experienței utilizatorului.

4. **Obtinerea listei dispozitivelor împerecheate:**

```
const getBondedDevices = async () => {
 try {
  const pairedDevices = await RNBluetoothClassic.getBondedDevices();
  setBondedDevices(pairedDevices);
  setBondedModalVisible(true);
 } catch (error) {
  console.error("Error retrieving paired devices:", error);
 }
};
```

Funcția getBondedDevices obține lista de dispozitive Bluetooth care au fost împerecheate anterior cu aplicația. Atunci când această funcție este apelată, ea folosește RNBluetoothClassic.getBondedDevices() pentru a returna toate dispozitivele Bluetooth deja împerecheate și le afișează utilizatorului într-o fereastră modală.

Această funcție permite utilizatorului să selecteze rapid un dispozitiv deja împerecheat pentru a stabili o conexiune, oferind o experiență de utilizare simplificată fără a fi necesară o nouă împerechere. În cazul unei erori, aplicația gestionează situația afișând un mesaj corespunzător în consolă.

5. **Conectarea la un dispozitiv împerecheat:**

```
const connectToDevice = async (device: BluetoothDevice) => {
 try {
  let connection: boolean;
```

```
    if (!(await device.isConnected())) connection = await device.connect();
    else connection = true;
    if (connection) {
      console.log("Device connected successfully");
      setSelectedDevice(device);
      startReading(device);
      setBondedModalVisible(false);
    }
  } catch (error) {
    console.error("Error connecting:", error);
  }
};
```

Funcția connectToDevice stabilește o conexiune între aplicație și un dispozitiv Bluetooth împerecheat. La apelare, funcția verifică dacă dispozitivul specificat este deja conectat folosind device.isConnected(). Dacă dispozitivul nu este conectat, încearcă să realizeze conexiunea folosind device.connect() și stabilește legătura de date dacă aceasta are succes.

După conectarea cu succes, aplicația stetează dispozitivul conectat ca selectedDevice și închide fereastra modală de selecție a dispozitivelor împerecheate. De asemenea, apelarea funcției startReading imediat după conectare permite aplicației să înceapă imediat recepționarea datelor de la Arduino.

6. Citirea datelor GPS:

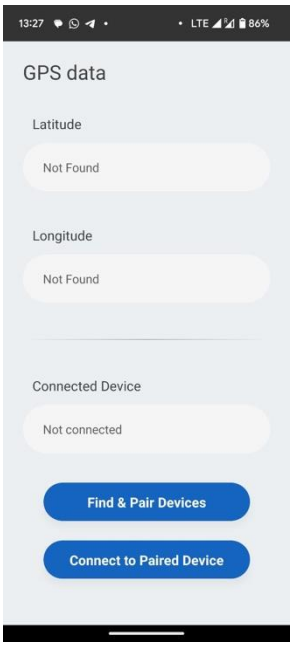
```
const startReading = async (device: BluetoothDevice) => {
  device.onDataReceived((data) => {
    try {
      const receivedData = data.data.trim(); // reduce spațiul suplimentar
      console.log("Received data:", receivedData);

      // desparte latitudinea si longitudinea, asumând că sunt separate de o virgulă
      const [latitude, longitude] = receivedData.split(",").map(Number);

      // Verifică funcția NaN pentru a gestiona corect datele
      if (!isNaN(latitude) && !isNaN(longitude)) {
        setData({ x: latitude.toString(), y: longitude.toString(), z: "0" });
      } else {
        console.log("Error: invalid data format");
      }
    } catch (error: any) {
      console.log("Error processing data:", error.message);
    }
  });
};
```

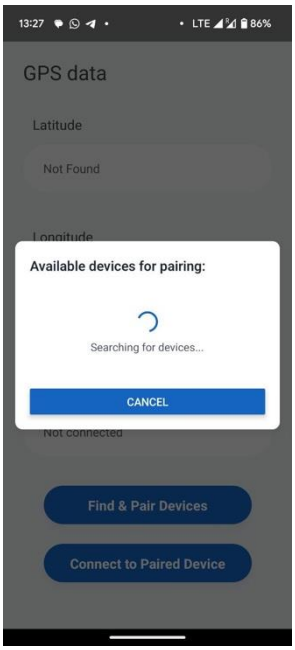
Funcția startReading activează un ascultător (listener) pentru recepția datelor de la dispozitivul Bluetooth conectat. Odată ce funcția este apelată, metoda device.onDataReceived monitorizează continuu datele transmise de la dispozitivul Arduino și declanșează o acțiune de fiecare dată când sunt primite date noi.

Ascultătorul preia aceste date și le procesează pentru a extrage valorile de latitudine și longitudine, asumându-se că acestea sunt separate prin virgulă. Dacă datele sunt valide (verificate pentru a nu fi NaN – Not-a-Number), acestea sunt actualizate în starea aplicației și afișate utilizatorului. Astfel, startReading asigură recepția în timp real a datelor GPS de la dispozitivul Arduino, iar în cazul în care formatul datelor este incorect, un mesaj de eroare este afișat în consolă pentru diagnosticare.

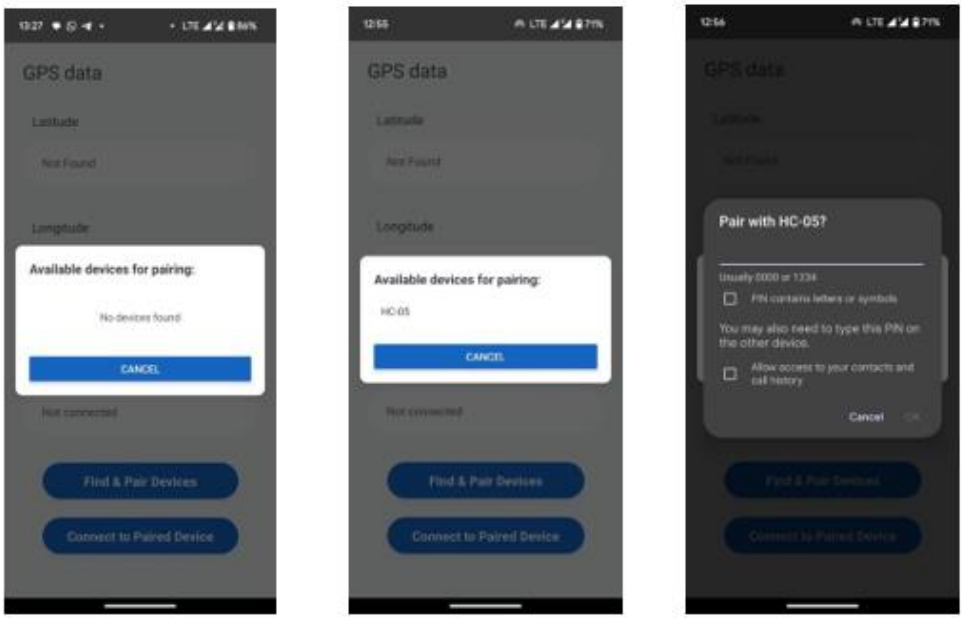
3.5. Flux de utilizare

La deschiderea aplicației, sunt afișate două butoane și trei câmpuri cu denumirile: Latitude, Longitude și Connected Device. Primele două câmpuri vor afișa informațiile despre coordonatele GPS, iar câmpul Connected Device va indica denumirea dispozitivului conectat la aplicație, dacă acesta este prezent.



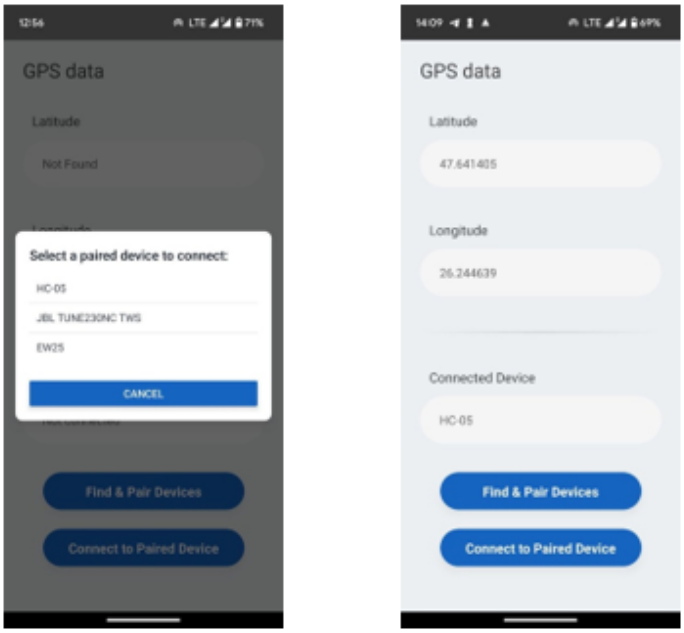


Dacă dispozitivul nu a fost împerecheat anterior, este necesar să realizăm împerecherea înainte de conectare. Pentru aceasta, apăsați butonul **Find & Pair Devices**. La selectarea acestuia, va apărea o fereastră modală și va începe căutarea dispozitivelor disponibile.



La finalul căutării, aplicația va afișa lista dispozitivelor disponibile pentru împerechere sau un mesaj care semnalează că nu au fost găsite dispozitive pregătite. Selectați dispozitivul dorit pentru împerechere, în acest caz **HC-05**. La împerecherea cu HC-05, aplicația va solicita un cod PIN; Introduceți **1234** pentru a finaliza împerecherea.

După împerechere, vom realiza conectarea la acest dispozitiv apăsând butonul **Connect to Paired Device**. Acesta va afișa lista dispozitivelor împerecheate și gata de conectare. Selectați din nou HC-05. După stabilirea conexiunii, în câmpul Connected Device va apărea numele dispozitivului. În același timp, aplicația va începe procesul de preluare a datelor GPS de la microcontrolerul Arduino.



4. Concluzie

Proiectul realizat demonstrează cu succes integrarea tehnologiei IoT prin utilizarea unei soluții accesibile și eficiente pentru monitorizarea coordonatelor GPS. Alegerea componentelor hardware precum Arduino Nano, modulul GPS NEO-6M și modulul Bluetooth HC-05 a asigurat un echilibru între costuri reduse și performanță. În același timp, dezvoltarea aplicației mobile în React Native Expo a facilitat o interfață intuitivă și portabilitatea între platforme.

Implementarea bibliotecilor specializate, cum ar fi TinyGPS++ și react-native-bluetooth-classic, a permis o procesare precisă a datelor și o transmisie rapidă, demonstrând utilitatea integrării eficiente a software-ului cu hardware-ul. Proiectul reflectă, de asemenea, principiile rețelelor de senzori și ad-hoc, utilizând o structură descentralizată pentru colectarea și transmiterea datelor în timp real.

Acest proiect deschide noi perspective pentru aplicații în domenii precum monitorizarea vehiculelor, explorarea în aer liber sau urmărirea obiectelor, subliniind potențialul soluțiilor IoT în viața cotidiană.