IOT projecteszköz személtetése

Gilányi Adrián, Simon Bence, Sinka Tibor

GPRS-es helymeghatározó kutyanyakörv:

A projektünk kiindulópontja egy személyes esemény volt, amikor Sinka Tibor kutyája eltűnt. Szerencsére a kutya meglett, de ez az eset rávilágított egy fontos problémára: a háziállatok könnyű elvesztésére. Ez az élmény indította el azt a gondolatmenetet, amely végül egy olyan helymeghatározós kutyanyakörv ötletéhez vezetett. Célunk egy olyan eszköz megalkotása volt, amely segítségével a kutyatulajdonosok megelőzhetik kedvenceik elvesztését vagy legalábbis biztosíthat egy módszert a megtalálásra. Az ötlet megvalósításához alaposan megterveztük a szükséges erőforrások beszerzését és a projekt megvalósításának lépéseit.

A nyomkövető rendszer megvalósítása:

Az eszköz részeit Szlovák techcégtől importáltuk magyar futárcég által akik Debreceni székhellyel vannak bejegyezve (TechFunSK)

Az eszköz részeit egyben rendeltük meg és a borháló csomagpont segített abban hogy eljusson hozzánk, akik az összeszerelést és a kód megírását végezték. A mellékletben részletesen elemezzük és szemléltetjük a kód sorait és az eszköz működését nagyvonalakban.

A GPRS-es helyzetmeghatározó alkatrészei:

1. 18650 Li-ion akkumulátor 1C 3,7V (2400mAh)
2. Tartó egy 18650-es Li-Ion akkumulátorhoz
3. TTGO T-Call V1.3 LilyGO WiFi / Bluetooth / GSM modul

0.Ábra: Működés közben az eszköz

1) Miután a kód sikeresen fel lett töltve a készülékre, a behelyezett SIM kártya segítségével (GPRS) lekérhetjük a legközelebbi adótorony koordinátáit hosszúsági és szélességi fokok alapján

1. Ábra: Serial Monitor Output

```
12:26:54.224 -> [50118] ### Modem: SIMCOM SIM800H
12:26:54.224 -> AT+CLTS=1
12:26:54.224 ->
12:26:54.224 -> RDY
12:26:54.224 ->
12:26:54.224 -> +CFUN: 1
12:26:54.224 ->
12:26:54.224 -> OK
12:26:54.224 -> AT+CBATCHK=1
12:26:54.224 ->
12:26:54.224 -> OK
12:26:54.224 -> AT+CPIN?
12:26:54.224 ->
12:26:54.224 -> +CME ERROR: SIM busy
12:26:55.204 -> AT+CPIN?
12:26:55.204 ->
12:26:55.204 -> +CPIN: READY
12:26:55.204 ->
12:26:55.204 -> +CPIN: READY
12:26:55.204 ->
12:26:55.204 -> OK
12:26:58.680 ->
12:26:58.680 -> Statuscode: 200
12:26:58.680 -> {"lat":47.973502,"lon":21.719972,"mcc":216,"mnc":30,"lac":65,"cellid":408,"averageSignalStrength":0,"range":1182,"samples":1,"char
12:26:58.680 -> Send: https://www.google.com/maps/@47.973502,21.719972,17z?entry=ttu
12:27:00.702 -> Timer Reached
```

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

lat:	47.973502
lon:	21.719972
mcc:	216
mnc:	30
lac:	65
cellid:	408
averageSignalStrength:	0
range:	1182
samples:	1
changeable:	1
radio:	"GSM"
rnc:	0
cid:	0
tac:	0
sid:	0
nid:	0
bid:	0
message:	null

2.. Ábra: generált JSON fájl

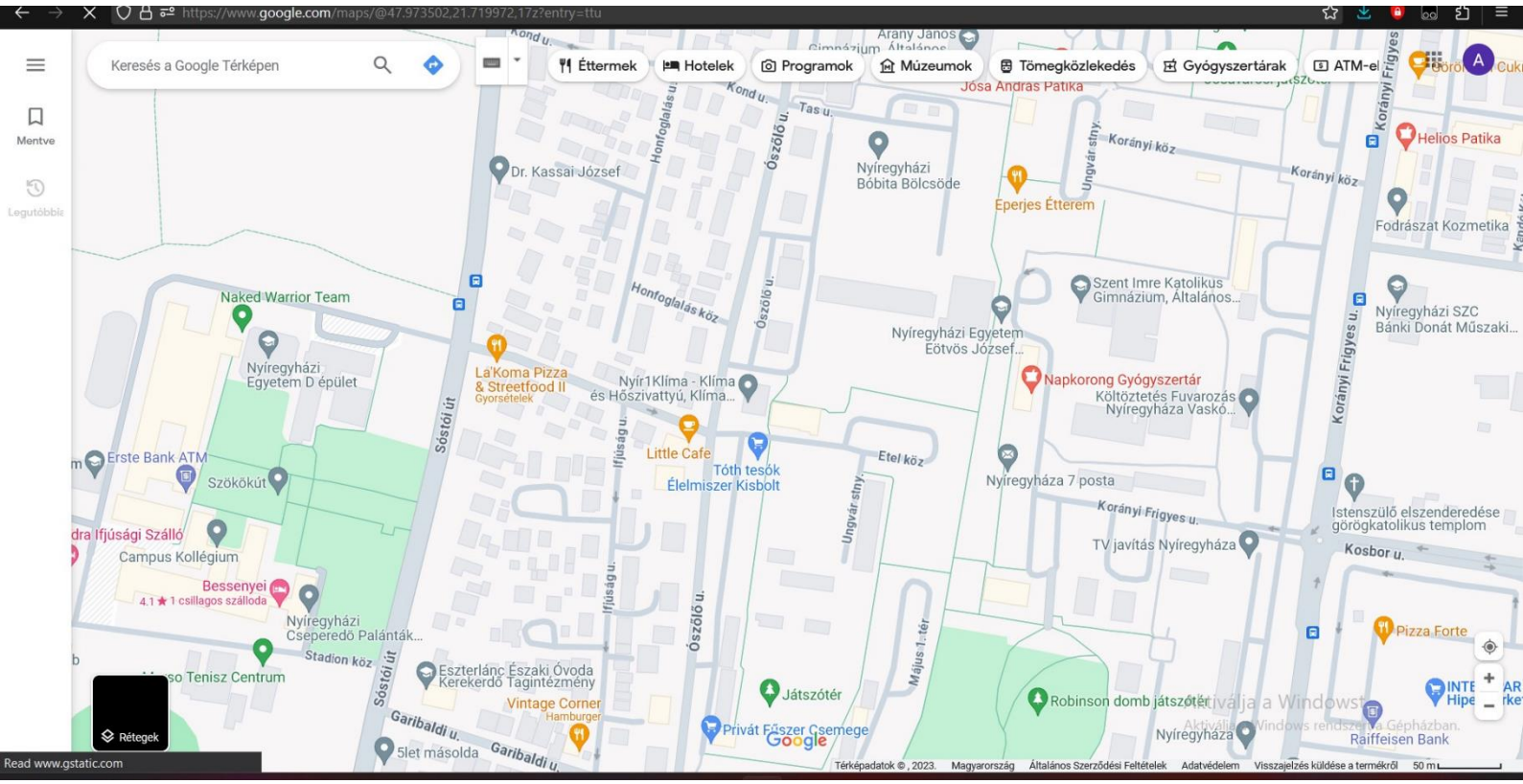
2) A GSM modul által megtalált torony adatainak beküldésével kapjuk meg a JSON formátumot, mely tartalmazza a szélességi és hosszúsági koordináták adatait, melyeket később feldolgozunk.

3. Ábra Blynk szerverre felkerült koordináták

3) A 3-as ábrán jól látható, hogy az automatizált folyamat sikeresen lefutott, és a megfelelő koordinációs adatok felkerültek a Blynk szerverre. Ezek az adatok lehetővé teszik a Google Maps számára, hogy visszaadja a legközelebbi adótornyot, ami alapján a hozzávetőleges helyzet megállapítható.

4) Tesztelés: A helyzetmeghatározás működik, az alábbi pontot kaptuk vissza a tesztfuttatás végeredményeképpen.

Ábra 4. Tesztelés



Az eszköz működéséhez szükséges programkód, amelyet az Arduino fejlesztői környezet implementációs szabályai szerint készítettünk.

```
#define SIM800L_IP5306_VERSION_20190610

#define DUMP_AT_COMMANDS
#define TINY_GSM_DEBUG          SerialMon

#include "utilities.h"

#define SerialMon Serial
#define SerialAT  Serial1

#define TINY_GSM_MODEM_SIM800
#define TINY_GSM_RX_BUFFER      1024

#define BLYNK_TEMPLATE_ID "TMPL4xqcKbKpA"
#define BLYNK_TEMPLATE_NAME "LocationDataBase"
#define BLYNK_AUTH_TOKEN "9AgXyWEd3LuBjsyVVJJJ-ch5AVnDPktd"

#define factor 1000000
#define sleep_time 1800

#define BLYNK_PRINT Serial
```

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <TinyGsmClient.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

#ifdef DUMP_AT_COMMANDS
#include <StreamDebugger.h>
StreamDebugger debugger(SerialAT, SerialMon);
TinyGsm modem(debugger);
#else
TinyGsm modem(SerialAT);
#endif

#define SMS_TARGET  "+36307807579"

//char ssid[] = "Telekom-2FFD79";
//char pass[] = "8d7bnha6cmv2v21d";
//char ssid[] = "Adrian";
//char pass[] = "asd12345";
char ssid[] = "Mamama internete";
char pass[] = "Dzsordzsi7";

String send = "";

BlynkTimer timer;

void setup() {
  SerialMon.begin(115200);
  delay(10);

  if (setupPMU() == false) {
    Serial.println("Setting power error");
  }

  setupModem();

  SerialAT.begin(115200, SERIAL_8N1, MODEM_RX, MODEM_TX);

  delay(6000);

  WiFi.begin(ssid, pass);
  Serial.print("Connecting to WiFi");

  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(500);
  }
  Serial.println("\nConnected to the WiFi network");
```

```
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  delay(10000);

  timer.setInterval(30000L, myTimerEvent);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

  esp_sleep_enable_timer_wakeup (sleep_time * factor);

}

void loop() {

  SerialMon.println("Initializing modem...");

  modem.sendAT(GF("+CNETSCAN=1"));
  modem.sendAT(GF("+CNETSCAN"));

  // GSM module collects the data from the nearest GSM tower.
  if (modem.waitResponse(10000L, GF("Operator:")) == 1){
    String res1 = SerialAT.readStringUntil('\n');
    modem.waitResponse();
    SerialMon.print("Data:");
    SerialMon.println(res1);

    modem.restart();
    delay(1000);

    String response = stringProcessor(res1);

    delay(1000);

    if ((WiFi.status() == WL_CONNECTED)){
      // Get Json file from the server, and store the longitude, latitude
      HTTPClient client;

      client.begin(response);
      int httpCode = client.GET();

      if(httpCode > 0){
        String payload = client.getString();
        Serial.println("\nStatuscode: " + String(httpCode));
        Serial.println(payload);

        StaticJsonDocument<200> doc;
        deserializeJson(doc, payload);

        String lat = doc["lat"];
        String lon = doc["lon"];
```

```
        // String that we send to the Blynk server
        send = "https://www.google.com/maps/@" + lat + "," + lon +
",17z?entry=ttu";
      }else{
         Serial.println("Error on HTTP request");
        send = "Cannot reach CellId Database";
      }
    }


    SerialMon.println("Send: " + send);
    delay(2000);

    Blynk.run();
    timer.run();

    delay(10000);

    // Esp hibarnation for 30 minutes.
    esp_deep_sleep_start();
  }

}

void myTimerEvent()
{
  Serial.println("Timer Reached");
  Blynk.virtualWrite(V0, send);
}

// Process data that the esp GSM module get from the nearest GSM tower.
String stringProcessor(String myString){
  String networkName;
  String mcc;
  String mnc;
  String rxlev;
  String cellId;
  String arfcn;
  String lac;
  String bsic;

  String url = "https://opencellid.org/cell/get?key=";
  String urlToken = "pk.9cd978cf808614a36b883b148590f5b2";
  String radio = "&radio=GSM";
  String format = "&format=json";

  int stringCounter = 0;
  int previousComma = 0;

  for(int i = 0; i < myString.length(); i++){
    if(myString.charAt(i) == ','){
```

```
      stringCounter++;
      switch(stringCounter){
        case 1:
          networkName = myString.substring(previousComma, i);
        case 2:
          mcc = myString.substring(previousComma, i);
        case 3:
          mnc = myString.substring(previousComma, i);
        case 4:
          rxlev = myString.substring(previousComma, i);
        case 5:
          cellId = myString.substring(previousComma, i);
        case 6:
          arfcn = myString.substring(previousComma, i);
        case 7:
          lac = myString.substring(previousComma, i);
        default:
          break;
      }
      previousComma = i + 1;
    }
    if(stringCounter == 7){
      bsic = myString.substring(previousComma);
    }
  }

  // Data fields for json string elements.
  mcc = stringCutter(mcc);
  mnc = stringCutter(mnc);
  lac = stringCutter(lac);
  cellId = stringCutter(cellId);

  //Json we send to the cellId server.
  String resultString = url + urlToken + mcc + mnc + lac + cellId + radio +
format;
  return resultString;
}

// Build the string elements that we use to communicate with the cellId server.
String stringCutter(String myString){
  String name;
  String valueString;
  char andSign = '&';
   int value;

  for(int i = 0; i < myString.length(); i++){
    if(myString.charAt(i) == ':'){
      name = myString.substring(0, i);
      valueString =  myString.substring(i + 1);
    }
  }
```

```
  name.toLowerCase();
  if(name != "mcc" && name != "mnc"){
    value = hexToDecimal(valueString);
    valueString = andSign + name + "=" + value;
  } else {
    valueString = andSign + name + "=" + valueString;
  }

  return valueString;
}

// Return decimal number from hexadecimal
int hexToDecimal(String num){
  int digit;
  int counter = num.length();
  int result = 0;

  for(int i = 0; i < num.length(); i++){
    if(num.charAt(i) >= 'A'){
      digit = num.charAt(i) - 'A' + 10;
    }else{
      digit = num.charAt(i) - '0';
    }
    counter--;
    result += pow(16, counter) * digit;
  }

  return result;
}
```

+++ HEADER fájl <utilities.h> +++

```
#include <Wire.h>

#if defined(SIM800L_IP5306_VERSION_20190610)

#define MODEM_RST            5
#define MODEM_PWRKEY         4
#define MODEM_POWER_ON      23
#define MODEM_TX            27
#define MODEM_RX            26

#define I2C_SDA             21
#define I2C_SCL             22
#define LED_GPIO            13
#define LED_ON              HIGH
#define LED_OFF             LOW

#define IP5306_ADDR         0x75
#define IP5306_REG_SYS_CTL0 0x00
```

```
bool setupPMU()
{
    bool en = true;
    Wire.begin(I2C_SDA, I2C_SCL);
    Wire.beginTransmission(IP5306_ADDR);
    Wire.write(IP5306_REG_SYS_CTL0);
    if (en) {
        Wire.write(0x37);
    } else {
        Wire.write(0x35);
    }
    return Wire.endTransmission() == 0;
}

#else

#error "Please select the corresponding model"

#endif

void setupModem()
{

    // Start power management
    if (setupPMU() == false) {
        Serial.println("Setting power error");
    }

#ifdef MODEM_RST
    // Keep reset high
    pinMode(MODEM_RST, OUTPUT);
    digitalWrite(MODEM_RST, HIGH);
#endif

    pinMode(MODEM_PWRKEY, OUTPUT);
    pinMode(MODEM_POWER_ON, OUTPUT);

    // Turn on the Modem power first
    digitalWrite(MODEM_POWER_ON, HIGH);

    // Pull down PWRKEY for more than 1 second according to manual requirements
    digitalWrite(MODEM_PWRKEY, HIGH);
    delay(100);
    digitalWrite(MODEM_PWRKEY, LOW);
    delay(1000);
    digitalWrite(MODEM_PWRKEY, HIGH);

    // Initialize the indicator as an output
    pinMode(LED_GPIO, OUTPUT);
    digitalWrite(LED_GPIO, LED_OFF);
}
```