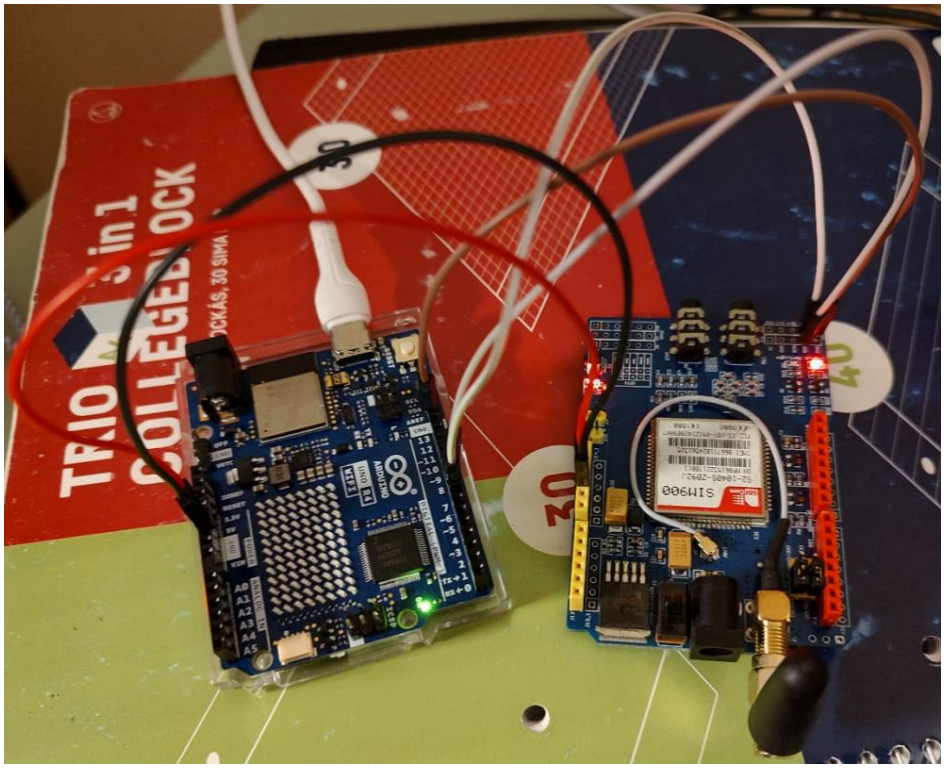
Fejlesztői kézikönyv és dokumentáció MEMR_Kapunyito programhoz



Előfeltételek

Kedves fejlesztő, ez a kézikönyv feltételezi, hogy ismeri a C és a C++ programozási nyelveket, valamit már programozott Arduino specifikus programot. Megismerte és értelmezte a Felhasználói kézikönyv tartalmát.

A feladat tervezése

Szempontok

A tervezésnél az egyik legfontosabb szempont a költségek minimalizálása, mivel akkor van értelme ennek a modulnak, ha olcsóbb, mint a jelenleg piacon kapható egyéb modulok.

A fejlesztést vegül egy univerzális boardon valósítottam meg.

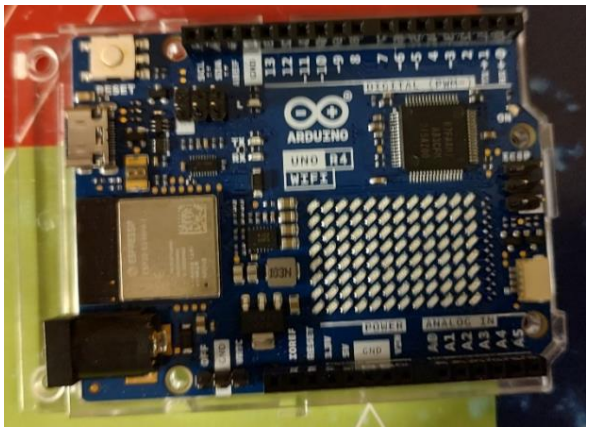
További fontos szempont volt még a kompakt méret és a kis fogyasztás is, mivel már egy meglévő szárnyaskapu vezérlése fölött kapna helyet a saját dobozában.

Adat gyűjtés

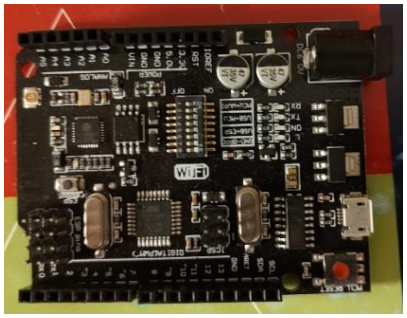
A hardver eszközök kiválasztásánál, már volt egy alap elképzelés, hogy mire lesz szükség, csak a pontosítás maradt mind megoldandó feladat.

Ezt egy internetes áruházban sikerült kivitelezni, ahol a korábbi pozitív tapasztalataim miatt otthonosan mozgok.

A megvásárolt eszközök:



Felmerült problémák



A költségek minimalizálása miatt először egy egyszerűbb fejlesztő platformot szereztem be, ami sajnos nem felelt meg a tervezett felhasználásnak.

A probléma, amivel ma is küzdök, az hogy nem rendeltem külön tápegységet mivel a kapuvezérlés rendelkezik még szabad 5V-os kapacitással.

A program működése nagy vonalakban

A teljes program 3 külső szubrutinon alapszik: getSerialString(), extractPhoneNumber(const String &iString) és parseSMS(const String &receivedMessage).

A loopban a program a modul soros portjáról olvassa be a modul által közölt üzeneteket, amiket a fentebb említett szubrutinok segítségével értékel ki.

A következőkben részletesen megismerheti a szubrutinok működését.

getSerialString()

```
String getSerialString()
{
  delay(500);

  while(sshield.available())
  {
    return(sshield.readString());                          //A szoftveres soros portról olvasott adat visszaadása
  }
  return "";
}
```

A szubrutin feladata a SIM 900 modul UART felületéről érkező adatok beolvasása majd, mint vissza térési érték, a fő kódnak visszaadni.

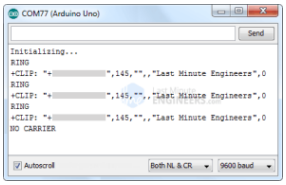
A szubrutin csak abban az esetben fog visszatérni nem üres String-el, ha elérhető a modul és azon van olvasható adat.

extractPhoneNumber(const String &iString)

```
String extractPhoneNumber(const String &iString)
{
    int sPos = iString.indexOf("+CLIP: \"+") + 13;              // A telefonszám kezdete

    int ePos = iString.indexOf("\",", sPos);                   // A telefonszám vége

    if (sPos >= 0 && ePos >= 0) {
        return iString.substring(sPos, ePos);                  // A konkrét szám visszaadása
    }
    return "";                                                 // Üres string vissza, ha nincs szám
}
```



A szubrutin feladata, hogy egy kapott az itt látható formátumú sztringből kiszedje a telefonszámot és ezt vissza adja.

A szubrutin először megkeresi a telefonszám kezdetét, majd a telefonszám végét. A substring eljárás segítségével kivágja a bemeneti stringből a vissza adható telefonszámot.

Amennyiben a kapott string nem felel meg ezeknek a feltételeknek, üres string kerül vissza adásra.

parseSMS(const String &receivedMessage)

```
String parseSMS(const String &receivedMessage)
{
    int mdIndex = receivedMessage.indexOf("MDnDGaTe01!");       // Megnézzük, hogy az sms tartalmazza-e a "MDnDGaTe01!" sztringet
    if (mdIndex != -1)
    {
      int plusIndex = receivedMessage.indexOf("+", mdIndex);    // Kiszedjük a telefonszámot az üzenetből
      if (plusIndex != -1)
      {
        String senderNumber = receivedMessage.substring(plusIndex, plusIndex + 13); // feltételezzük, hogy a telefonszám 13 karakter
        return senderNumber;
      }
    }
  return "";                                                    // Üres sztringet adunk vissza, ha nincs találat.
}
```

A szubrutin feladata, egy kapott stringről megállapítani, hogy az egy, a jelszót tartalmazó sms és ha igen akkor a küldő telefonszámának vissza adása.

A szubrutin legelőször megvizsgálja, hogy megtalálható-e a jelszó a megkapott stringben. Ha igen, akkor a telefonszámot keresi meg, amit aztán visszatérési érték ként visszaad.

Ha nem lenne benne a jelszó, vagy hibás lenne, a vissza térési érték egy üres string.

A program fő eleminek felépítése

Globális beállítások

```
#include <SoftwareSerial.h>

String knownNumber;
SoftwareSerial sshield(7, 8);
```

A program futása során a knownNumber fogja eltárolni az összes ismert telefonszámot.

Az sshield globális változó az ahol megkapja a board, hogy mely pineken tud kommunikálni a SIM modullal.

void setup

```
void setup() {
    sshield.begin(9600);

    delay(1000);
    sshield.println("AT+CMGF=1");
    sshield.println("AT+CNMI=1,2,0,0,0");

}
```

Az Arduinonál megszokott, hogy a soros portokat meg kell nyitni a setup fázisban.

Az alatta lévő két sorban megmondjuk a modulnak, hogy milyen módon kezelje a bejövő sms-eket.

void loop

```
void loop() {
    String buffer, number;
    buffer = getSerialString();                         //Olvasás a bufferba
    if(!buffer.equals(""))                              //Ha van tartalom a bufferban akkor van miverl foglalkozni
    {
      number = extractPhoneNumber(buffer);              //Kiszedjük a számot a bufferből, ha van benne
      if(!number.equals(""))
      {
        if(!number.equals(knownNumber))                 //Megnézzük hogy olyan számról hívtak-e amit ismerünk
        {
          digitalWrite(4, HIGH);
          delay(1000);
          digitalWrite(4, LOW);
          delay(300000);
          digitalWrite(4, HIGH);
          delay(1000);
          digitalWrite(4, LOW);
        }
      }
      number = parseSMS(buffer);
      if(!number.equals(""))
      {
        knownNumber.append(number);
      }
    }
}
```

Ezaz a program részlet, amely folyamatosan futni fog ameddig a board bekapcsolva van. Minden futás alkalmával megkérdezzük a modult, hogy van-e üzenete, majd kiértékeljük azt.