Önálló laboratórium beszámoló — BME–TMIT

Készítette: Fekete Ákos, N6Z4HP
Szak: villamosmérnök
Szakirány: Infokommunikációs hálózatok és alkalmazások
Email cím: akosroland25@gmail.com

Konzulens: Dr. Fehér Gábor
Email cím: fehergabor@tmit.bme.hu

Tanév: 2021/22. tanév, 2. félév

Téma címe: RFID kártyaolvasó rendszer RS485-tel

Feladat:

Egy RFID-ra alapuló beléptető rendszer megvalósítása RS485-re alapuló kommunikációval.

A feladataim közé tartozott az RS485 megismerése, alkalmazásainak közelebbi megismerése, RFID kártyaolvasó rendszer megismerése, Arduino-hoz illesztése, RS485-tel kommunikáció megvalósítása, saját kommunikációs protokoll kidolgozása, vagy egy ismert felhasználása.

Ezenfelül a szabványok által előírt kritériumok ellenőrzése.

Tartalomjegyzék

1. A laboratóriumi munka környezetének ismertetése

A következő részben munkához szükséges elméleti tudás és a projekt kiindulási állapotát mutatom be.

1.1. Bevezetés és elméleti összefoglaló

Az RS485 egy buszrendszer alapú, vezetékes kommunikációra készült rendszer, amelyet 1983-ban dolgoztak ki. Külön kommunikációs protokoll nincs hozzárendelve, ez egy elektronikai standard, amely a korábbi RS422-őt váltotta fel.

Közöttük a legnagyobb különbség a meghajtó/szintillesztőikben van, lehetővé téve a megbízhatóbb több pontos kommunikációt ugyanazon vonalon. Szabványosítva EIA/TIA-485-ként vezettek be.

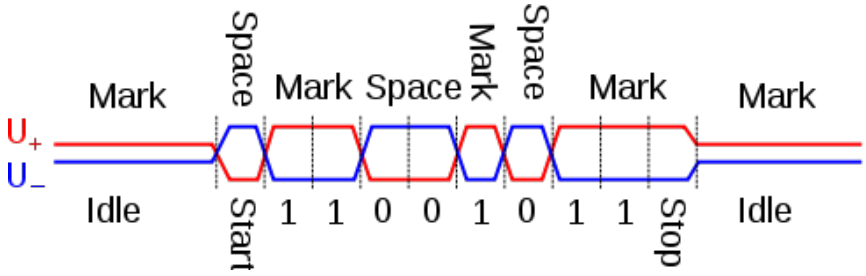
A RS485-re több további technológia is fejlődött ki, ilyen például az autókban használt CAN-busz [1] és egy hasonló univerzális ipari kommunikációs rendszer a Profibus [2].

1. táblázat. RS422 és RS485 fontosabb jellemzők összehasonlítása [3]

	RS422	RS485
Üzemmód	Egyirányú differenciális többpontú	Full duplex differenciális többpontú
Megengedett Tx és Rx [1]	1 Tx, 10 Rx	32 Tx, 32 Rx
Max kábel hossz	1250 m	1250 m
Max adatsebesség	akár 10 Mbps (4)	akár 10 Mbps (4)
Minimum és a Maximum driver kimeneti tartománya	± 2 V - ± 5 V	± 1.5 V - ± 5 V

[1] Tx - Transmitter, adó/ Rx - Receiver, vevő

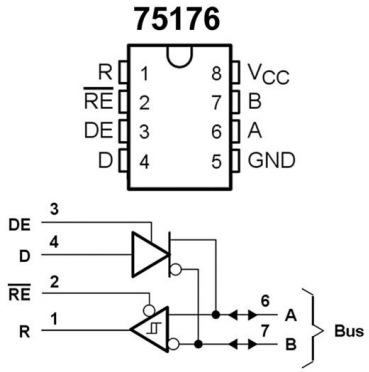
Az 1. táblázatban olvashatóak a főbb különbségek, ebben láthatjuk, hogy a rendszer felépítését tekintve a buszrendszer egy differenciális érpárból áll, illetve hogy kifejeztetetten egy többpontú oda-vissza kommunikáció megvalósítása volt a cél. A buszrendszer két vonalát "A" és "B" síneknek nevezzük, differenciális mivoltukat tekintve működésük ellentétes, amikor az egyik a "magas"/High állapotban van, a másik "alacsony"/Low állapotba kerül. Tipikusan az "A" szokott High állapotban logikai 1-et adni. Az 1. ábra ezt jól szemlélteti, ahol az U+ felel meg az "A"-nak, illetve értelemszerűen az U- a "B"-nek.s



1. ábra. Tipikus RS485-ön alapuló adatátvitel elméleti hullámformája [1]

A 2 . ábrán egy tipikus szintillesztőt, úgynevezett driver-nek [4] láthatjuk a blokk diagramját. Egy ilyen driver be és kimenetei a következőképpen alakulnak:

- RO - Receiver out, a vevő kimenete
- RE - Receiver enable, vevő engedélyező bemenete, Low állapotban érvényes
- DE - Driver enable, adó engedélyező bemenete High állapotban érvényes
- DI - Driver in, adó bemenete
- GND - Ground (0V), föld
- A - Az "A" sín
- B - A "B" sín
- Vcc - Táp (Vcc/5 V)



2. ábra. RS485-ös szintillesztő/driver blokk diagramja

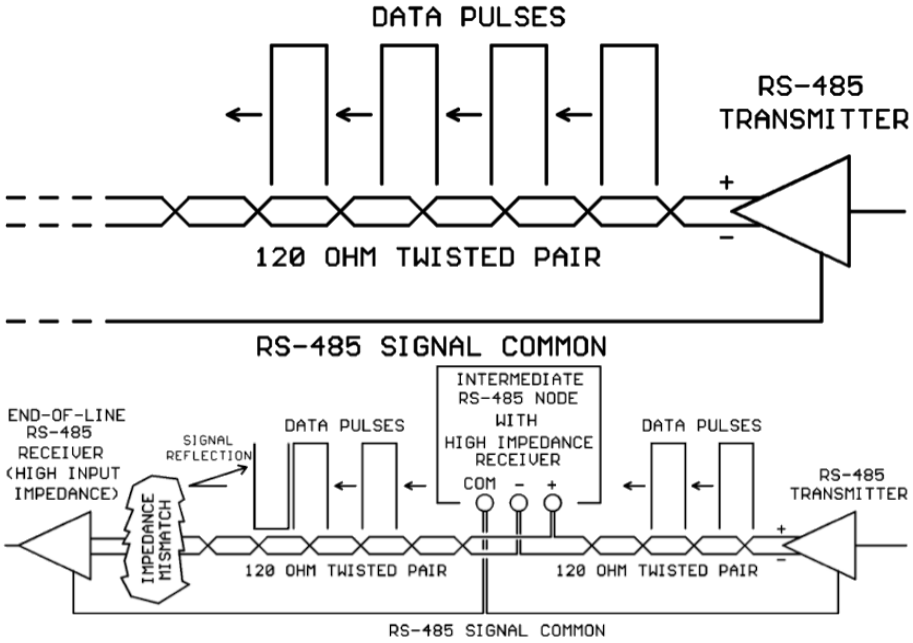
Az 1. táblázatban olvashatjuk, hogy 32 adó-vevő készüléket lehet maximálisan elhelyezni a vonalon, azokat a helyeket, ahol terminálszerűen lehet csatlakozni a buszrendszerre, node-oknak [5] nevezzük. A node-okra a terminálokat célszerű nagy impedanciájúra (12 kΩ - 96 kΩ) választani, mivel különben az adót túlzottan leterhelné a kis impedanciával rendelkező vevők sokasága.

Az RS485 kommunikációs vonal végein tipikusan egy 120 Ω-os lezáró ellenállást szokás elhelyezni, a vonalvégeken történő adatimpulzusjelek visszaverődéseit, reflexióinak a küszöbölésének érdekében.

[1] Forrás: `https://en.wikipedia.org/wiki/File:RS-485_waveform.svg`

Fontos hogy mindkét végén legyen elhelyezve, hiszen a feljebb írtakkal összeegyeztetve, ha az utolsó terminál lenne a vonal lezárása, akkor az nagy impedanciás lezárásnak minősülne, amit impedancia eltérésnek szokás az nevezni, ami a vonal reflexiók kialakulását nagyban növeli. Egy ilyen visszaverődés a nagyságát és sokaságát tekintve az továbbított adat értelmezésének nehézségét okozhatja, vagy akár a teljes értelmetlenséget.

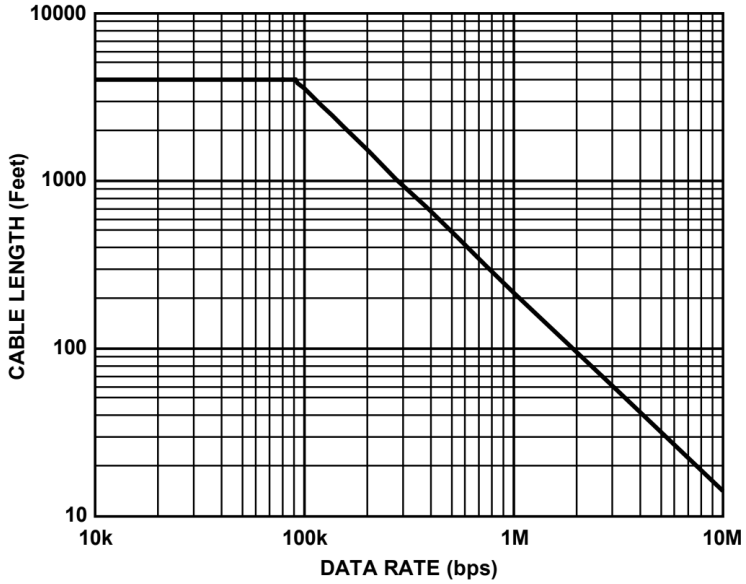
Megjegyezendő, hogy ezeket a terminálszerű csatlakozásokat külön-külön nem szokták ellátni lezáró ellenállással abban az esetben, ha viszonylag a kialakított buszrendszerre közvetlenül csatlakozik fel. Viszont ellenkező esetben, ha egy külön lecsatlakozási ágat szeretnénk létrehozni, akkor célszerűen ott is érdemes a lezáró ellenállások alkalmazása.



3. ábra. Az adattovábbítása a buszrendszeren és a adatimpulzus reflexiója

A 3 . ábrán az adattovábbítás látható. Az alsóbbik képen egy terminál csatlakozása is megfigyelhető. Megállapítható az is, hogy akár broadcast[2] típusú kommunikáció is megvalósítható egy ilyen vonalon.

[2]Broadcast: ez egy adó, és sok vevő típusú kommunikációs forma, ilyen például a TV, rádió



4. ábra. Az átviteli sebesség ábrázolása a kábelhosszúság tekintetében

Az 1. táblázatban olvashatjuk illetve a 4. ábrán látható az átvihető adatsebesség. Az ábrán jól látható a 10 Mbps-es maximális sebesség és azt a következtetést lehet levonni, hogy minél távolabb szeretnénk adatot eljuttatni, az adatfolyam sebessége egyre jobban lelassul.

Az RFID [6] technológiáját tüzetesebben nem vizsgáltam, de egy rövid leírás gyakorlati szempontból érdekes lehet a számunkra. A legegyszerűbben látásmódhoz elég annyit tudni, hogy egy RFID kártyában van egy tároló egység amin adatot lehet tárolni, ilyen például a ID-ja a kártyának, ami elvileg egyedi (a gyakorlatban léteznek olyan kártyák ahol ez is átírható), a tárhelye nem nagy, amiket blokkokra vannak szétosztva. Minden egyes ilyen tagben vagy kártyában van egy antenna (egy tekercsszerű vezető) és kizárólag passzív módon működik, azaz ha közeli elektromágneses térbe kerül, akkor feszültség fog benne indukálódni, a tárolt adat kinyerhető.

1.2. A munka állapota, készültségi foka a félév elején

A féléves munka elején a projektet teljesen az elejéről kezdtem, azaz nem bocsájtottak a rendelkezésemre a témába illő dokumentumot, félkész projektet.

1.3. A munka kezdete

Elméleti tudásom a specifikus témában nem volt, illetve az Arduino programozásában szükséges alapvető tudást témalaboratóriumi foglalkozások keretein belül elsajátítottam, a dokumentációhoz használt LaTeX környezet használatát önállóan kezdtem el, amely a kezdeteiben nagy nehézségeket tapasztaltam. Az Arduiono programozásához nagyban segített az C++-ban szerzett tudásom a második félévben, ahol egy SDL2-es Sudoku-t [7] készítettem el.

Mivel semmilyen Arduino board nem állt rendelkezésre, sajáttal nem rendelkeztem, így a Schönherz Elektronika Műhely által fejlesztett, úgynevezett SEMDuino-t [8] használtam, ami

egy Arduino UNO alapú kártya, működésüket tekintve számomra nincs bennük különbség.

Az Arduino IDE kisebb programok írására alkalmas, viszont egy nagyobb lélegzetű téma feldolgozására már alkalmatlan, a programnyelv szintaxisát nem emeli ki, illetve a debugolásra sincsenek megfelelő eszközök beleépítve. Ezért úgy döntöttem, hogy a Microsoft Visual Code fejlesztőkörnyezetét használom, PlatformIO [9] kiegészítővel, amely segítségével az Code-ból közvetlenül lehetséges a soros portok kezelése, a programkód feltöltése a kártyákra és az Arduino specifikus könyvtárak elérése, importálása.

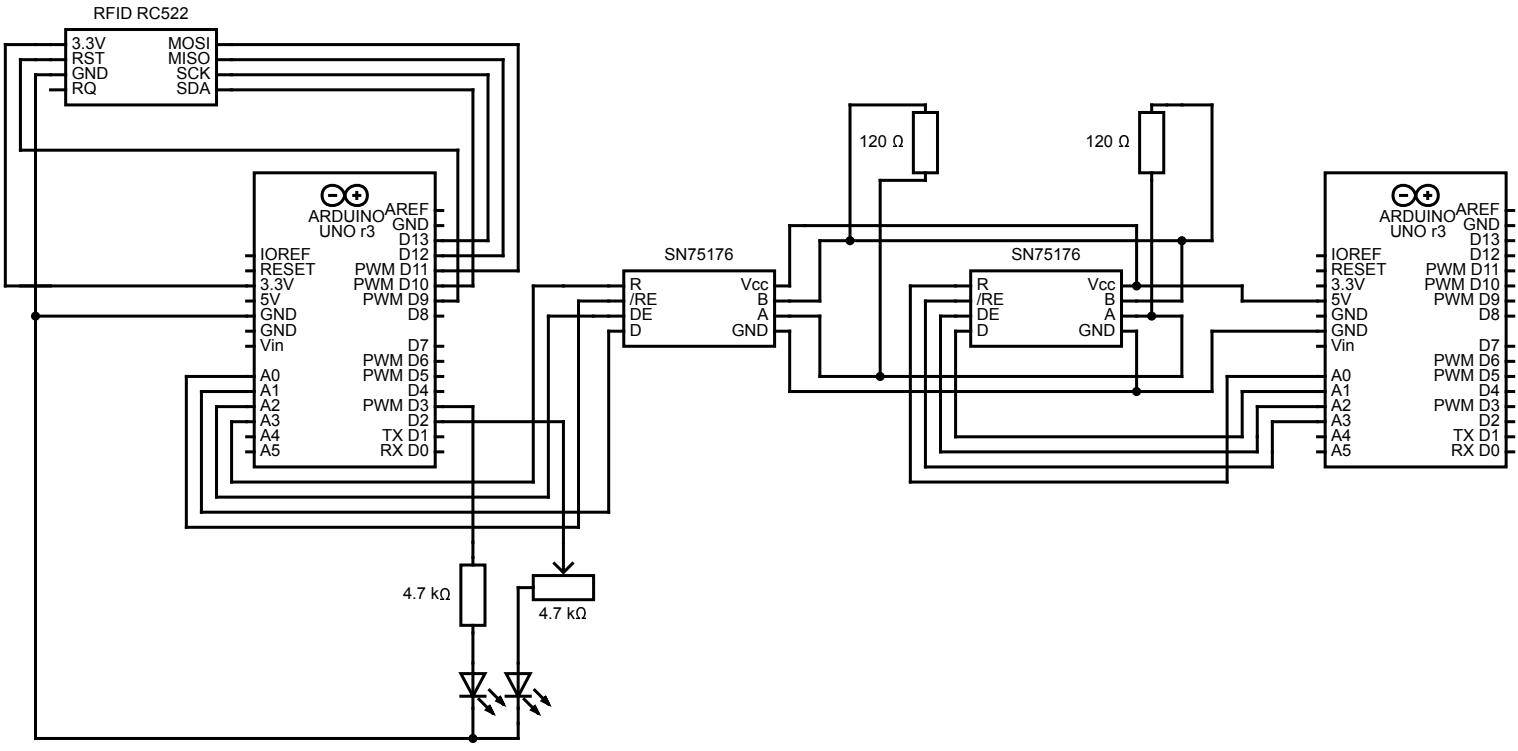
A PlatformIO [9] használatához nem kötelező a VS Code használata, CLion, Atom vagy akár linux terminálból is működtethető, így egy nagyon erős eszköz válik elérhetővé a jól követhető kód elkészítéséhez.

2. Az elvégzett munka és az eredmények ismertetése

2.1. Az általam konkrétan elvégzett munka bemutatása

A félév során következőket olvastam el, programoztam, készítettem el, teszteltem, dokumentáltam és néztem át:

- Ebben a beszámolóban ábrák készítése és szerkesztése
- Elkészítettem egy slave (250 sor) és egy master (184 sor) kódját
- Teszteltem a helyes működését a megvalósított áramkör működését oszcilloszkóppal
- A kódok helyes működéséének tesztelése
- Arduino-s könyvtárak és példakódok olvasása, értelmezése
- A felhasznált eszközök adatlapjainak tanulmányozása
- Áramkör kapcsolási rajzának megrajzolása
- Ezen dokumentum elkészítése, az ahhoz használt LaTeX nyelv elsajátítása



5. ábra. A megvalósított áramkör kapcsolási rajza

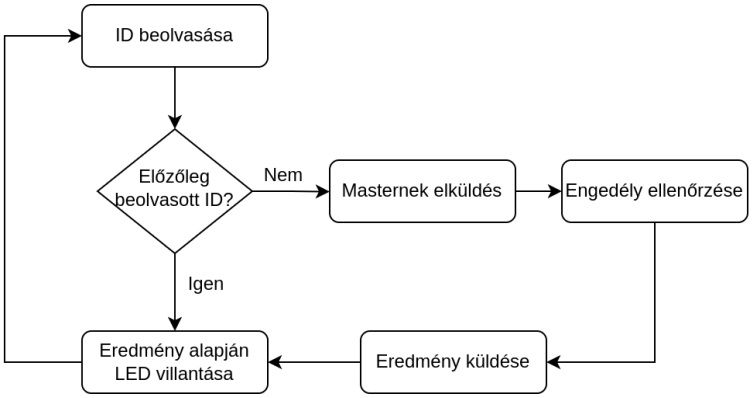
Az 5. ábrán látható a kész, megvalósított áramkör kapcsolási rajza. Az áramkörnek a megvalósításához két Arduino alapú fejleszőkártya (esetemben ez SEMDuino), egy RFID-RC522-es

RFID olvasó, 2 SN75176-os szintillesztő, 2 fele méretű próbanyák, 1 piros LED, 1 zöld LED, trimmer és sima ellenállás a LED-ekhez, jumper kábelek és 2 darab 120 Ω-os lezáró ellenállás szükséges.

2. táblázat. Az RFID RC522-es ajánlott bekötése különböző típusú Arduino kártyákhoz

Signal	MFRC522 Reader/PCD Pin	Arduino Uno/101 Pin	Arduino Mega Pin	Arduino Nano v3 Pin	Arduino Leonardo/Micro Pin	Arduino Pro Micro Pin
RST/Reset	RST	9	5	D9	RESET/ICSP-5	RST
SPI SS	SDA(SS)	10	53	D10	10	10
SPI MOSI	MOSI	11 / ICSP-4	51	D11	ICSP-4	16
SPI MISO	MISO	12 / ICSP-1	50	D12	ICSP-1	14
SPI SCK	SCK	13 / ICSP-3	52	D13	ICSP-3	15

A 2. táblázatból ki lehet olvasni, hogy a gyártó mit javasol az olvasókártya lábkiosztásnak egy adott fejlesztőkártya estén a bekötéséről. Mivel a különböző fejlesztőkártyák működésükben eltérnek. Az általam használt SEMDuino-k UNO alapúak, ezért az alapján kötöttem be, ez a 2. táblázatnak a sárgával kijelölt oszlopa alapján történt meg. Az RC522-nek a RQ, azaz az interrupt vagy más néven megszakítást lábát nem használnám fel amúgy se semmire, mivel az én céljaim elérésében csak a bonyolultsági faktort növelné, így azt az áramkörbe bekötni felesleges. Ez az RC522-es kártya SPI-t [10] használ az Arduinoval való adatküldésre. Az projekt jobb átláthatóságának kedvéért készítettem egy megfelelő hosszúságú szalagkábelt, így letisztultabb lett kinézete.



6. ábra. Folyamat ábra a megvalósított algoritmusról

A 6. ábrán látható az általam elképzelt és megvalósított algoritmus folyamat ábrája, a folyamat meglehetősen egyszerű, az adatküldése és maga a szintillesztő lekezelése volt nagyobb kihívás, RFID tag beolvasása és elmentése a kártya működéséhez szükséges könyvtárban található példakód átgondolása után a saját igényeimre szabtam. A kommunikációs protokollnak végül

sajátot készítettem, ami a következő logika alapján működik:

3. táblázat. A felépített kommunikáció

Master	Slave	
[	[	Start
Portnum	Master_ID	Hova
Ok/No	RFID	Végeredmény/Beolvasott érték
]	]	End

Az adatküldéshez szükséges úgynevezett portok használok, ezt fel lehet úgy is fogni, hogy minden egyes terminálnak és Masternek egy külön egyedi ID-ja van, ami alapján be lehet azonosítani. Mivel 32 adó-vevő egység lehet, a Masternak választottam a legnagyobbat, a 32-őt, a többi szabad címre, ami az 1-től 31-ig lévő tartomány, pedig a beolvasó terminálokat lehet megcímezni. Az általam elkészített példámban ezt 31-re választottam. Az 1.1.-es pontban taglaltam, hogy akár broadcast típusú adatküldés is lehetséges megvalósítani, ez a portok felhasználásával nagyon könnyen elkészíthető, például az összes Slave figyeli a Master által küldött forgalmat.

Az általam felépített kommunikáció tipikusan így fog kinézni:

A Slave beolvassa a kártya ID-ját és elküldi a Masternek: [3231xx xx xx xx]

A hozzá tartozó kódrészlet:

```
void␣RFID_send(String␣slave_port,␣String␣ID) {
lastestCard␣=␣ID;
digitalWrite(DE,␣HIGH);
rs485.println("["␣+␣MASTER_PORT␣+␣slave_port␣+␣ID␣+␣"]");
rs485.flush();␣//␣wait␣for␣complete␣the␣message
digitalWrite(DE,␣LOW);
return;
}
```

Ezek után a Master leellenőrzi hogy van-e jogosultsága a kártyának a belépéshez, dönt, majd visszaküldi a válasz: [31x]

Master kódrészlete:

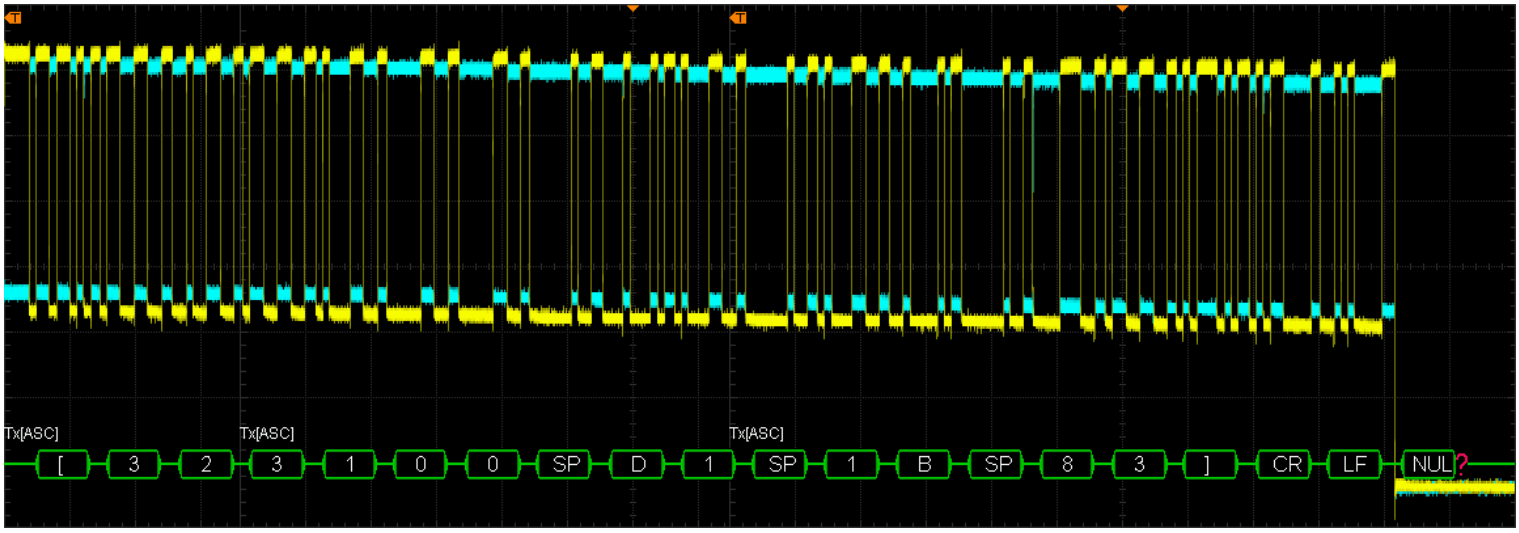
```
void␣Approve_send(String␣Port,int␣result) {
digitalWrite(DE,␣HIGH);
rs485.println("["␣+␣Port␣+␣result␣+"]");
rs485.flush();␣//wait␣for␣complete␣the␣message
digitalWrite(DE,␣LOW);
return;
}
```

Mivel az RFID olvasó minden egyes olvasási ciklusában beolvassa az hozzá közelkerülő kártyák ID-ját, így viszonylag sokszor, egy másodpercben többször is elküldte ellenőrzésre az ID-t, így kialakulhatott az, hogy ha egy új kártyát tartottunk oda, akkor a terminál még az előző kártyának a válaszát kapta meg a Slave, így úgy tűnhetett mintha belépési jogosultsága lenne. Ez

ellen úgy védekezhetünk, hogy a előző kártya ID-ját és a hozzá tartozó belépési kódot tároljuk, így csak akkor szükséges elküldeni újra az adatot, ha egy másik ID-val rendelkező kártyát érzékel az olvasó. A megfelelő működéshez érdemes belassítani a folyamatot, én ezt $300ms$-es várakozással értem el.

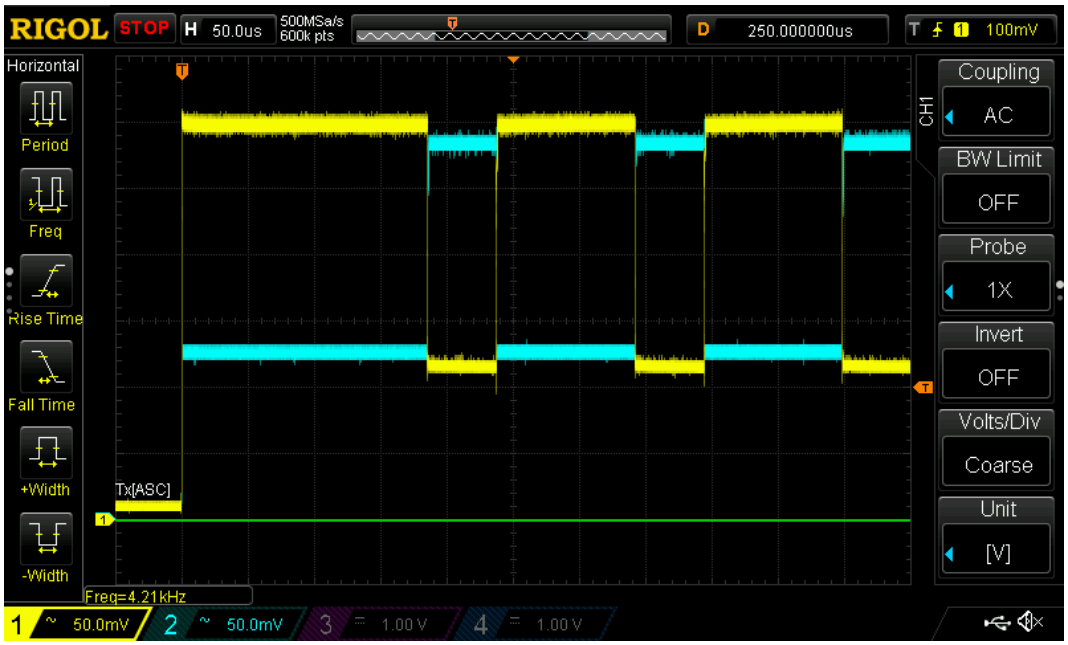
Az ehhez tartozó kódrészlet:

```
...
if(to_check_id␣!=␣lastestCard){
to_check_id.trim();
RFID_send(PORT,␣to_check_id);
delay(300);
}
//Or␣blink␣the␣correct␣led/decide␣again
else{
deceider(lastestResult);
}
...
```



7. ábra. Pillanatfelvétel egy teljes adatátviteli blokk hullámformájáról

A 7. ábrán pontosan az olvasótól a Masternek küldött teljes adatfolyam oszcilloszkóp ábrája látható. Az oszcilloszkóp (MATH funkciói között) szerencsémre tudott dekódolni, így az ábra alján látható a dekódolt jel. Láthatóan ez megegyezik az általam felépített kommunikációs modellel. Egy kicsit részletesebben belenagyítva az alábbi ábrát kaphatjuk:



8. ábra. Belenagyított felvétel a hullámformáról

A 8. ábrán látszik, hogy az A és B sín jól megkülönböztethető, működésük ellentétes, ahogy azt a 4. ábrához képest elvártuk. A jelszinteket megvizsgálva látható hogy B alacsonyabb valamivel mint A, illetve a feszültség nagyságát tekintve elmaradt az elvártaktól, amit ha az SN75176 dokumentációjában [4] a 7.8-as részben található karakterisztikákat megvizsgáljuk, akkor a legelső ábrából rájöhetünk, hogy az IC tápellátását tekintve nem kap megfelelő nagyságú áramot, így a karakterisztika alján helyezkedik el a munkapontja. Ez magyarázható azzal, hogy az Arduinokat USB-ről hajtottam. Az Arduino 12V-ról működtetése vagy akár külső tápellátást biztosítva ez orvosolható. Az utóbbi szempont a galvanikus leválasztás szempontjából is előnyös lenne, hiszen jelen formájában ez nincs biztosítva, így földáramkörök tudnak kialakulni.

2.2. Összefoglalás

A félévi munka során elért új eredmények ismételt, vázlatos, tömör összefoglalása:

- Tanulmányoztam az RS485, az azt megelőző és ráépülő technológiákat
- Készítettem összesen 434 sor kódot
- Az Arduino segítségével megvalósítottam egy áramkört
- Az írt kódokat teszteltem, debugoltam, törekedtem az átlátható kódolásra, sűrű kommentelésre
- Elkészítettem ezt a beszámolót LaTeX-ben, az idevágó ábrákat szerkesztettem
- Oszcilloszkóppal vizsgáltam az adatfolyam helyességét

3. Irodalom, és csatlakozó dokumentumok jegyzéke

3.1. A tanulmányozott irodalom jegyzéke

[1] CAN-bus `https://ieeexplore.ieee.org/abstract/document/8053160`

[2] Profibus `https://en.wikipedia.org/wiki/Profibus`

[3] Táblázatból nyert adatok `https://www.maximintegrated.com/en/design/technical-documents/tutorials/7/736.html`

[4] A felhasznált szintillesztő adatlapja `https://www.hestore.hu/prod_getfile.php?id=8588`

[5] Csatlakozás és reflexió `https://www.kele.com/content/blog/2014/01/17/the-rs-485-network-terminator`

[6] Az RFID-ről egy bővebb leírás `https://ieeexplore.ieee.org/abstract/document/1593568`

[7] SDL2-es Sudoku forráskódja és dokumentációja `https://github.com/akosblack/Sudoku_cpp`

[8] Régebbi SEMDuino felépítése `http://home.sch.bme.hu/~sasdik/SEMduino.htm`

[9] A PlatformIO-hoz tartozó dokumentáció `https://docs.platformio.org/en/latest/integration/ide/pioide.html`

[10] SPI és I2C bemutatása `https://ieeexplore.ieee.org/abstract/document/4762946`

3.2. A csatlakozó dokumentumok jegyzéke

Csatlakozó két forrásfájl jegyzéke:

- A repository: `https://github.com/akosblack/rs485rfid`

Ezen repositoryban megtalálható a dokumentációhoz használt LaTeX kód és a hozzá szükséges ábrák is.