MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

AUTOMATIZÁLÁSI ÉS
INFOKOMMUNIKÁCIÓS INTÉZETI
TANSZÉK

ÁLTALÁNOS
INFORMATIKAI TANSZÉK

Robot Operating System távérzékelés periféria

Diplomamunka

Készítette: Tompa Tamás

Neptun-kód: LJQHYK

Cím: Miskolc, Szentgyörgy út 47 8/1

Tartalomjegyzék

1. Bevezetés

A tudomány és a technika folyamatos fejlődése révén egyre újabb eszközök válnak életünk részévé, ezen eszközök egyik csoportja a robotok. A robotika napjainkban egyre népszerűbb tudományág, a csúcstechnológia alkalmazása is egyre inkább ebbe az irányba halad. De mit is értünk robot illetve robotika alatt? Az interneten található számos definíció közül az egyik szerint „*A robotok olyan fizikai ágensek, amelyek a fizikai világ megváltoztatásával oldanak meg feladatokat.*" [1]. Mit is takar ez a definíció? Állítása szerint a robot egy olyan környezetét érzékelő illetve környezetébe beavatkozni tudó gépezet, amely bizonyos feladatok megoldásával megváltoztatja a körülöttünk lévő világot. Tehát egy robotnak konkrét célja, feladata van. Robotika alatt pedig a robotokkal foglalkozó tudományágat értjük. A robotok alkalmazása számos előnnyel jár, az élet több területén is alkalmazzák őket, elterjedtek például az iparban, az orvostudományban, illetve a jövőben alkalmazandóak, mint szociális segítőtársak. Napjainkban viszont a robotok még terjedtek el a háztartásokban, de már vannak ma is megvásárolható robot eszközök, melyek a háztartásbeli feladatok elvégzését hivatottak segíteni, ilyen például Roomba, a porszívórobot, TurlteBot, a felszolgáló robot, vagy a házilag épített fűnyíró robotok.

Dolgozatom célja a robotika tématerületének mélyrehatóbb tanulmányozása, konkrét robotok adott feladatot megvalósító perifériájának tervezése és megvalósítása, illetve a periféria működtetéséhez szükséges számítógépes robotvezérlő keretrendszerbeli szoftverkomponens implementálása céljából.

A robotok, melyekhez a tervezendő, megvalósítandó periféria illesztendő Ethon és TurtleBot. Ethon-t, a Budapesti Műszaki és Gazdaságtudományi Egyetem fejlesztése, célja egy portás robot megtestesítése, amely etológusok által meghatározott feladatokat lát el. Ilyen feladatok például, hogy adott folyosón bolyong, bolyongás közben felismeri a körülötte lévő embereket, felismerés után üdvözli őket, ha akkumulátora lemerülőben van, akkor megkeresi a töltőegységét. Maga a portás robot fogalom, funkció ezen időszakban van kialakulóban, azt, hogy hogyan kell működnie és milyen funkciókkal kell rendelkeznie, milyen feladatotokat kell ellátnia egy portás robotnak, vagyis, hogy mi is a portás robot, azt az Eötvös Lóránd Tudományegyetem etológusai és a Budapesti Műszaki és Gazdaságtudományi Egyetem mérnökeivel, kutatóival együttműködve határozzák meg. Ethon főként etológiai kutatásra, azaz ember-gép interakció vizsgálatára készült, melyet az Eötvös Lóránd Tudományegyetem tanulmányoz. A Miskolci Egyetem Általános

Informatikai tanszékén lévő, a projektben résztvevő kisebb csapat feladata, hogy a fentebb említett funkciókat megvalósító robot rendszert megtervezze, implementálja. Dolgozatom célja e projekt keretében felmerülő konkrét feladat megoldásának, megvalósításának ismertetése, bemutatása.

Diplomamunkám keretében tehát saját feladatom kerül bemutatásra, melynek célja egy olyan robotperiféria tervezése, megvalósítása és illesztése a fentebb említett robotokhoz, amely bináris események fogadására, illetve ezen események alapján történő robotvezérlésre alkalmas. Mivel Ethon célja egy portás funkció betöltése, így alapvető feladat lehet, hogy az általa felügyelt területet ellenőrizze, védje és a terültre csak belépési engedéllyel renelkező személyeket engedjen be. A beléptetés egy lehetséges módja, hogy az illető valamilyen módon jelez a robot számára, hogy be szeretne lépni a terültre, majd a robot megvizsgálja az illetőt és ha az adott személy jogosult a belépésre, akkor a robot ajtót nyit számára. Az illető által a belépési szándék jelzésének lehetséges módja lehet egy csengetés jelzés küldése a robot számára. Ez alapján tervezendő robotperiféria által kiváltott bináris esemény lehet például a „csengetés” esemény megvalósítása, azaz egy, a robotok számára alkalmas vezetéknélküli csengő megvalósítása, vagy továbbá egy a robot távirányítását lehetővé tevő esemény. A bináris események továbbítására, fogadására alkalmas robotperiféria két elektronikus hardver összetevőt takar, egyik eszköz maga az adó, amellyel az esemény generálható, például a csengetés, vagy a távirányítás, illetve a vevő periféria, amely az adó által küldött bináris jelet fogadja, vaalmilyen összeköttetésben áll a robottal, illetve annak számítógépével. A vevő periféria által fogadott jelet egy számítógépes robotvezérlő keretrendszerbeli szoftverkomponens dolgozza fel és valósítja meg a robot vezérlését.

A dolgozat tématerülete tehát kettős, egyrészt villamosmérnöki, hiszen a fejlesztendő robotperiféríák beágyazott rendszereket, áramköröket takarnak, másrészt pedig informatikai, hiszen a fejlesztendő robotperiféríákat működtető, illetve a robotok vezérlését megvalósító robotvezérlő keretrendszerbeli szoftver implementálásra van szükség. A dolgozat felépítését a megvalósítandó feladat alapján logiakilag több részre tagolom. Az első részben a megvalósításhoz szükséges elméleti hátteret tanulmányozom, illetve ismertetem a robotok felépítését, technikai paramétereit. A második részben a bináris események küldésére, fogadsára alkalmas robotperiféríák tervezését, megvalósítását, működését, felépítését, a harmadik részben pedig a robot periféríákat működtető robotvezérlő keretrendszerbeli szoftverkomponens tervzését, műkdését, implementálását ismertetem.

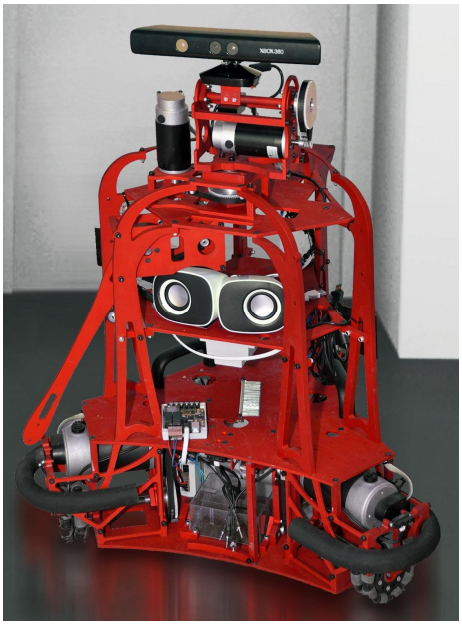
2. Információgyűjtés

Ezen fejezetben a feladat megvalósításához szükséges elméleti hátteret, többek között a robotok célját, felépítését, technikai paramétereit ismertetem, valamint tanulmányozom a különböző robotvezérlő keretrendszereket.

2.1 Ethon bemutatása

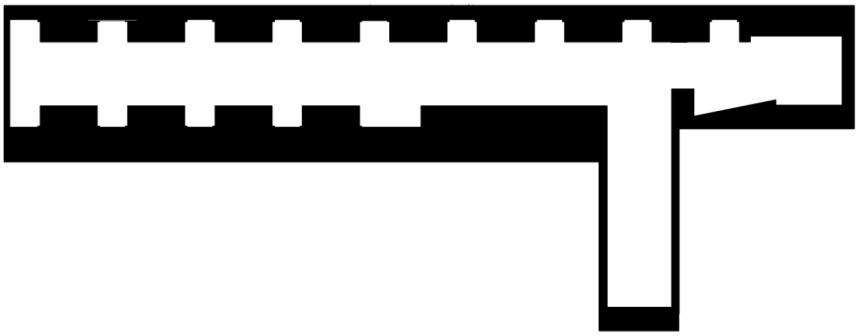
[2]

Ethon (teljes nevén Mogi Ethon Rubens, továbbiakban Ethon) célja egy portás robot megtestesítése, amely egy adott területen (pontosabban az Eötvös Lóránt Tudomány Egyetem Etológia tanszékének folyosóján) bolyongva felismeri a körülötte lévő embereket, a felismerés után üdvözli, követi, megjegyzi az illető arcát, felismeri az őt körülvevő akadályokat (akár dinamikus akadályokat is) majd ha akkumulátora merülőben van, akkor megkeresi a töltő egységét.



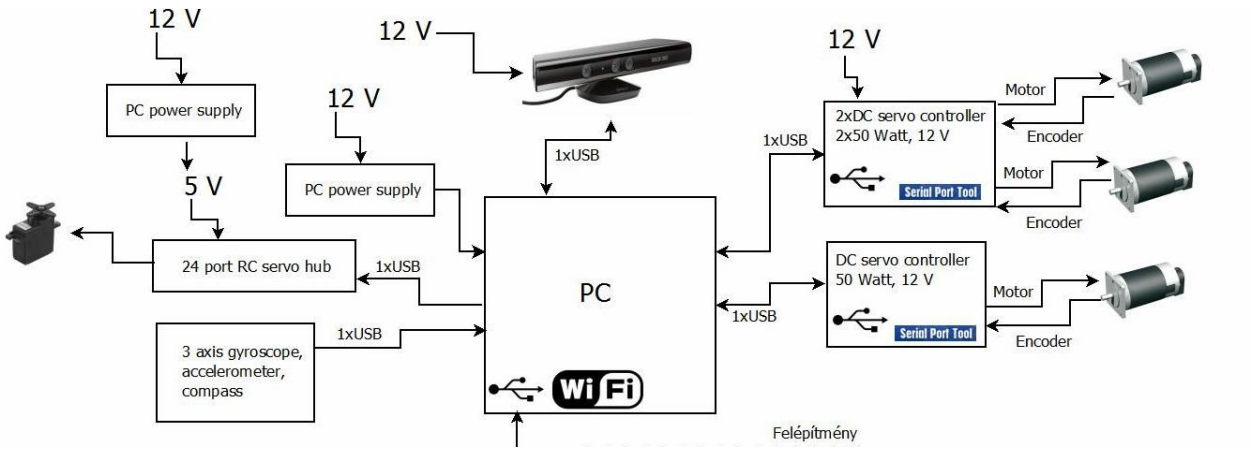
1. ábra Ethon (Etorobot2-ELTE)

Ethon-t a Budapesti Műszaki és Gazdaságtudományi Egyetem (továbbiakban BME) tervezte és valósította meg, a robot fő célja, hogy az Eötvös Lóránt Tudomány Egyetem (továbbiakban ELTE) etológusai segítségével vizsgálják az ember-gép interakciót, azaz, hogy az emberek hogyan reagálnak a környezetükben lévő robotra, robotokra. Ethon célját, funkcióit az ELTE etólugusai határozzák meg illetve ők validáják az elkészült modelleket, megvalósított funkciókat. Az alábbi 2. ábrán az ELTE Etológia tanszékének folyosólya látható, azaz az a terület, melyen a robot a funkcióját betölti majd.

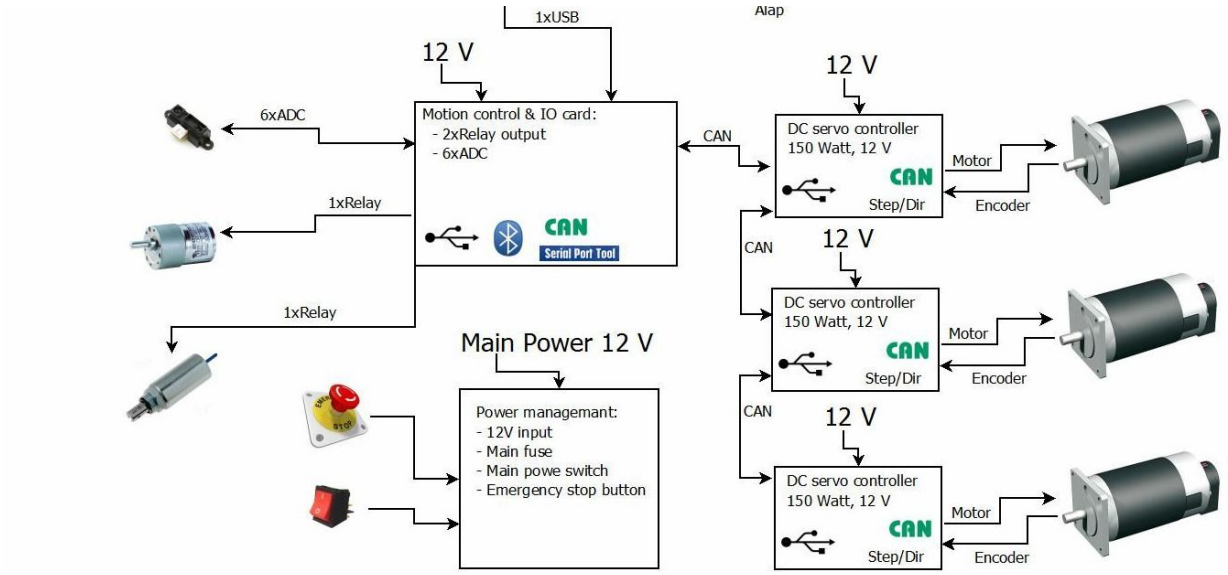


2. ábra Az ELTE Etológia tanszék folyosójának alaprajza

Ethon felépítése két részre különíthető el, ez az alsórész (robotalap) és a felsőrész (felépítmény). Az alsórész, azaz a robotalap felépítése állandó, majd e felett helyezkedik el a felépítmény, amely moduláris kivitelű, azaz a későbbiekben még bővülhet különböző funkciókat, feladatokat megvalósító modulokkal. A robotalap és a felépítmény USB kapcsolaton keresztül van összekötettésben a robot számítógépével. Az alábbi 3. és 4. ábrák a felépítmény illetve a robotalap blokkvázlatát szemléltetik:



3. ábra A felépítmény blokkvázlata



4. ábra A robotalap blokkvázlata

A robotalapon található az akkumulátor, a robot működését irányító mikorvezérlők (AVR-ek), a kereket meghajtó, egyenként 150W-os összesen 3 darab kefés egyenáramú motor, a motorok üzemeléséhez, működtetéséhez szükséges teljesítmény végfokok, 6 darab Sharp távolságérzékelő szenzor, további elektronikák (step up és step down áramkörök, szervó vezérlő), illetve egy takarító egység, amely egy forgó henger alakú kefe kosztárolóval együtt, és egy labdaütögető egység. Ezen egységeket relével kapcsolgatja a központi elektronika. Ezen részen foglal helyet a robot számítógépe is, amely a robot irányítását, vezérlését hívatott megvalósítani. Mivel autonóm robotról lévén szó, így ez a számítógép vezérli magát a robot, annak perifériáit, USB portokon keresztül tartja a kapcsolatot a mikrovezérlőkkel, illetve ez a számítógép hívatott megvalósítani a képfeldolgozási műveletetek is.

Ethon holonómikus hajtással rendelkezik, amely azt jelenti, hogy kerekei kerültén újabb kisebb kerekek (görgők) helyezkednek el. Ennek a hajtásnak az előnye, hogy a robot bármilyen irányban tud mozogni, mert kerekei a forgásiránnyal keresztben is képesek elmozdulni. A motor és a kerekek között szíjáttétel található.

A felső részen 2 darab mikorvezérlő (AVR) helyezkedik el, melyek a robot fejének és karjának a vezérléséért felelősek, itt található a robot fejét megtestesítő Kinect, és itt helyezkedik el a robot karja is. A fej minden irányban mozgatható, a kéz fel-le mozgásra képes. A robot előre meghatározott üzenetekkel (keretekkel) vezérelhető, melyeket különböző soros portokon (alsó rész, fej, kar) szükséges továbbítani.

Ethon technikai paramétereit összefoglaló táblázat:

1. Táblázat Ethon technikai paraméterei [2]

Kerület ütközővel: 757 mm
Kerékátmérő: 101,6 mm
Hasmagasság: 16 mm
Tömeg kb: 40kg
6db távolságérzékelő: 60 fokonként 3db a földtől: 158 mm-re, a középponttól 688/2 mm-re 3db a földtől 201 mm-re, a középponttól 265/2 mm-re
Kar forgástengelye a földtől: 516 mm, a középponttól: 316/2 mm
Kinect magassága a földtől: 805 mm, Kinect függőleges forgástengelye a középponttól 25 mm-re előre Kinect mozgástere: függőleges tengely körül: +/- 90 fok, vízszintes tengely körül a függőlegessel bezárt szög: + - 50 fok
Takarító modul: a robot elejében kb középen, tengelye a robot közepétől 130/2 mm-re takarító tároló térfogata: 325*116 mm^3 összevethető Roomba robottal.

2.2 Turtlebot bemutatása

[3] [4] [5]

Trutlebot egy háromkerekű, több felépítményt, Kinect-et illetve számítógépet tartalmazó, komplett készletként megvásárolható felszolgáló robot. A készlet tartalmazza magát a robotvázat, a Kinect-et, egy számítógépet, akkumulátorokat illetve egy pendrive-ot, melyen előre feltelepített, konfigurált Linux disztibíció (Ubuntu) és a robot működését vezérlő ROS (Robot Operating System) keretrendszer található. A felépítmény alsó részén található maga a robotalap, azaz az a robot működését vezérlő elektronika, a kerekek, az ütközés illetve a távolságérzékelők. A felépítmény középső részén a Kinect található, illetve itt kap helyet a robotot vezérlő számítógép is, a felső rész pedig kialakítása miatt lehetőségeket biztosít újabb perifériák (például robotkar, fejmozgató egység) csatlakoztatására.

A robot 2 darab USB porton keresztül csatlakozik számítógéphez, az egyik USB port a robotalapot, a másik pedig a Kinect-et működteti. A robot vezérlése a robotalaphoz tartózó USB port-ra (valójában sorosportra) küldött utasításkeretek segítségével valósítható meg. A robot számos beépített szenzort, például úthosszmérő szenzort a kerekein, távolságszenzorokat a robot alapon, giroszkópot, ütközésérzékelőket, stb. tartalmaz.

Turtlebot technikai paramétereit összefoglaló táblázat:

2. Táblázat Turtlebot technikai paraméterei

Súly: 2.35 kg
Méretek: 351.5mm x 124.8mm
Maximális sebesség: 70 cm/s
Maximális fordulási sebesség: 180 deg/s
Maximális teherbírás: 5 kg
Üzemidő: 3/7 óra (akkumulátor függő),
Töltési idő: 1.5/2.6 óra (akkumulátor függő)
Szenzorok: giroszkóp, ütközésérzékelő (3 darab), lépcsőérzékelő (3 darab),odométer a kerekeken
Szenzorok frissítési frekvenciája: 50 Hz

Az alábbi 5. ábrán a robot fényképe látható:

5. ábra Turtlebot robot [6]

2.3 Robotvezérlő keretrendszerek

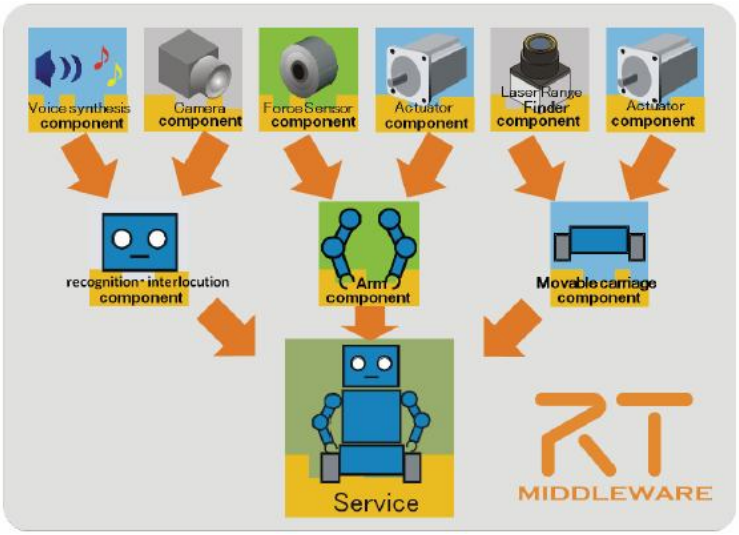
[7]

A robotvezérlő rendszerek célja, hogy egységes felületet biztosítsanak robotok programozására, segítséget nyújtva nagyobb, összetettebb feladatok megvalósításában. Ahogy egyre népszerűbbek lettek a robotokkal kapcsolatos fejlesztések egyre több probléma mutatkozott meg: a robotrendszerek fejlesztése meglehetősen drága, összetett feladat, amely sok időt és specifikus tudást igényel. Ezért terjedtek el a robotvezérlő rendszerek, melyek komponens alapú szemléletükkel lehetőséget biztosítanak ilyen feladatok megvalósításában. A komponens alapú szemlélet szerint minden egyes jól meghatározott funkciót egy adott modul valósít meg, a teljes rendszer pedig több ilyen modul összekapcsolása révén valósul meg.

A robotvezérlő rendszerek főbb funkciói:

- érzékelők, beavatkozók kezelése, integrációja
- vezérlési architektúra létrehozása
- kommunikációs csatornák definiálása
- működés közben adatok gyűjtése
- tesztelés meglévő adatmintákkal

A 6. ábra egy robotvezérlő keretrendszerbeli szoftver moduláris felépítését szemlélteti:



6. ábra Robotvezérlő rendszer lehetséges komponensei RT-Middleware rendszer esetén [8]

2.3.1 ROS - Robot Operating System

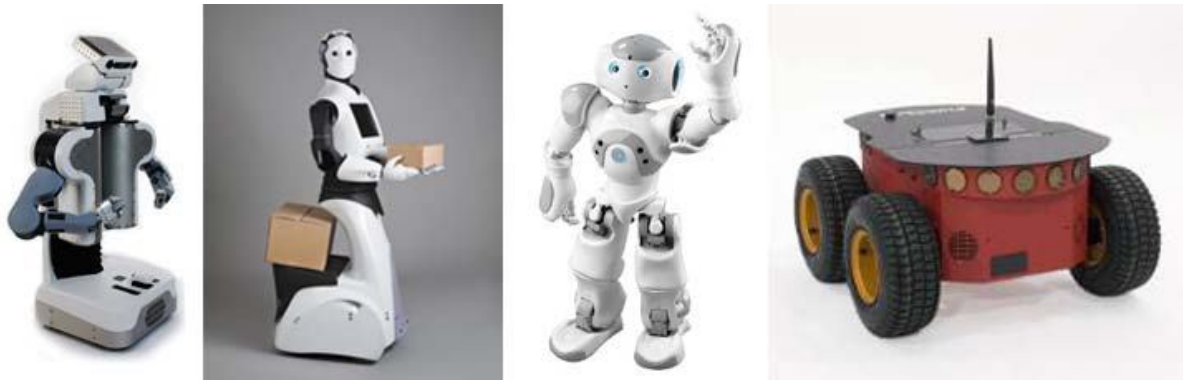
[9][10][11]

A Robot Operating System, vagyis a ROS egy robot operációs rendszer, melyet a robotikában széles körben alkalmaznak. Amerikai fejlesztésű, nyílt forráskódú, rendszer, amely könyvtárai segítségével lehetővé teszi komplexebb robotvezérlő szoftverek fejlesztését. E rendszer fejlesztését 2007-ben kezdte a Stanford Artificial Intelligence Laboratory (SAIL) a Stanford AI Robot projekt keretében, majd 2008-ban csatlakozott a fejlesztéshez a Willow Garage és még számos kutatócsoport. Előnye, hogy sok előre beépített funkciót tartalmaz, ilyenek például az arcfelismerés, mozgáskövetés, sztereo látás (több kamera segítségével), robotvezérlés, útvonaltervezés, SLAM, stb. A ROS főként Linux operációs rendszert (de Mac OS X-et, illetve már Windows-t is) és a robotvezérlő szoftverek fejlesztéséhez C++ illetve Python programozási nyelveket támogat.

Az eddig megjelent ROS disztribúciók:

- Box Turtle (2010 Február)
- C Turtle (2010 Augusztus)
- Diamondback (2011 Február)
- Electric Emys (2011 Augusztus)
- Fuerte Turtle (2012 Március)
- Groovy Galapagos (2012 Október)
- Hydro (2013 szeptember)

Az alábbi 7. ábrán néhány robot látható, melyet a ROS rendszer működtet:

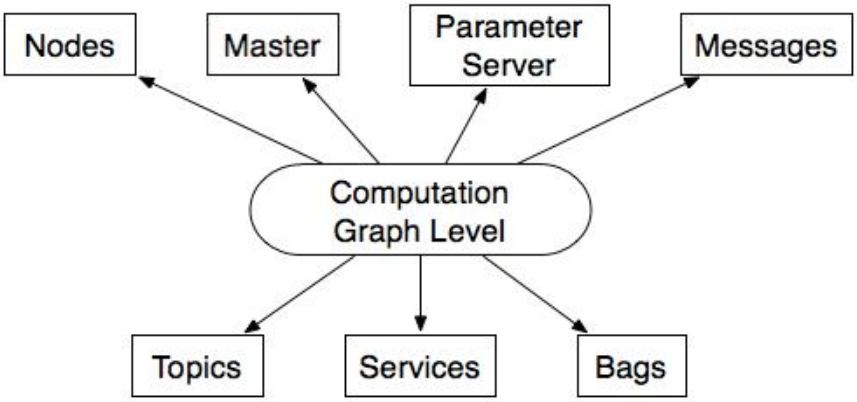


7. ábra ROS rendszer által működtetett robotrendszerek [11]

2.3.1.1 A rendszer felépítése

[11] [12]

A következő 8. ábra a rendszer elemeit, felépítését szemlélteti:



8. ábra A rendszer elemei, felépítése [11]

A rendszer az ábra alapján az alábbi egységekből épül fel:

- Nodes (csomópontok)
- Master (központi vezérlő)
- Parameter server (paraméter szerver)
- Messages (üzenetek)
- Topics (topik-ok, üzenetsorok)
- Services (szervizek, szolgáltatások)
- Bags

Csomópontok (nodes)

Node-oknak, csomópontoknak a rendszer futtatható alkalmazásait nevezik, melyek jól meghatározott feladatot hajtanak végre. A rendszerben több ilyen csomópont lehetséges, melyek kapcsolatban is állhatnak egymással adatcsere, kommunikáció céljából. Egy komplexebb robotvezérlő szoftver több ilyen csomópontból épül fel, ezen csomópontok több, különböző számítógépen is futhatnak. Node-ok implementálásra a ROS beépített könyvtárai segítségével biztosít lehetőségeket c++ illetve python programozási nyelveken.

Központi vezérlő (master)

A master a rendszer fontos eleme, ő valósítja meg az egyes csomópontok, illetve a rendszer elemi közötti kommunikációt, futtatókörnyezetet biztosítva azoknak. Névszervízt biztosít a

rendszernek, segít a node-oknak, hogy tudjanak egymás létezéséről. A rendszer működéséhez alapvető feltétel, hogy a master mindig jelen legyen.

Paraméter szerver (parameter server)
A csomópontoknak lehet segítségével paramétereket átadni, akár a csomópontok futása közben is.

Üzenetek (messages)
A csomópontok kommunikációja üzeneteken keresztül történik, ezen üzenetek segítségével továbbítanak adatot egymásnak. Az üzenet tartalmazza a továbbítandó adatot. A rendszer tartalmaz előre beépített üzenettípusokat, de lehetőséget biztosít a fejlesztő számára, hogy saját üzenettípust is definiálhasson.

Üzenetsorok, topik-ok (topics)
A rendszer az üzenetek továbbítására üzenetsorokat, úgynevezett topikokat biztosít. Mikor egy csomópont üzenetet küld egy másik csomópont számára, akkor ezt egy üzenetsoron keresztül valósítja meg. A publikáló csomópont közzéteszi üzenetét a topikban, majd a fogadó csomópont pedig kiolvassa onnan. A rendszer lehetőséget biztosít saját, új topikok definiálásra, mindegyik topik saját névvel ellátva él a rendszerben.

Szervizek, szolgáltatások (services)
A csomópontok közötti kommunikáció megvalósításának másik módja. Mikor a node-ok közötti kommunikáció a fentebb említett üzenetsorokon keresztül történik, akkor az adott csomópont adott időközönként írja üzentet az üzenetsorba, egy másik csomópont pedig olvassa az onnan. A szervizek a csomópontok közötti kérés-válasz típusú kommunikációt hívatottak megvalósítani. Ebben az esetben az egyes node-ok csak akkor kommunikálnak, mikor azt egy másik node kéri tőle. A fejlesztő általi szervizek definiálására a ROS könyvtárai segítségével biztosít lehetőséget, minden szerviz adott néven él a rendszerben.

Bags
A „bag”-ek a rendszerbeli üzenetek, adatok mentésére és azok visszajátszására biztosítanak lehetőséget. Fejlesztésnél különösen hasznos lehet például az egyes szenzorok által küldött adatok mentésére, majd azok visszajátszására tesztelés céljából.

Az alábbi 9. ábra a ROS jegyzékszerkezetének felépítését szemlélteti:

```
workspace_folder/ -- WORKSPACE
        src/ -- SOURCE SPACE
                CMakeLists.txt -- 'Toplevel' CMake file, provided by catkin
                package_1/
                        CMakeLists.txt -- CMakeLists.txt file for package_1
                        package.xml -- Package manifest for package_1
                        ...
                package_n/
                        CMakeLists.txt -- CMakeLists.txt file for package_n
                        package.xml -- Package manifest for package_n
```

9. ábra A ROS jegyzékszerkezete [11]

A workspace_folder a jegyzék a rendszer munkakönyvátra, itt helyezkedik el minden, a fejlesztő által létrehozott komponens, szolgáltatás. Ezen jegyzékszerkezetet a rendszer hozza létre a megfelelő utasítás (catkin_init_workspace) végrehajtásával.
Az src-ben található CMakeList.txt állomány a rendszer fordítójának (catkin_make) szóló információkat tartalmaz, a package_1, package_n jegyzékek pedig az egyes, fejlesztő által létrehozott csomagokat, komponenseket jelölik. Mindegy egyes, a fejlesztő által létrehozott csomag tartalmaz egy CMakeList.txt és egy package.xml állományt. A csomagokban elhelyezkedő CMakeList.txt a fordítónak szóló információkat tartalmaz melyek az adott csomópontra vonatkoznak, például a csomópont fordításához, futtatásához szükséges függőségeket. A csomagokon belül elhelyezkedő package.xml a csomagról, annak céljáról, funkciójáról, illetve a csomag létrehozójáról tartalmaz információkat.

Egy, a 10. ábrán látható csomag (package_1,.., package_n) szerkezetének részletesebb felépítését a következő ábra tartalmazza:

```
chapter3_tutorials/
├── CMakeLists.txt
├── config
│   └── chapter3_tutorials.config
├── launch
│   ├── example1_gdb.launch
│   ├── example1.launch
│   ├── example1_valgrind.launch
│   ├── example2.launch
│   └── example3.launch
├── mainpage.dox
├── Makefile
├── manifest.xml
├── output
│   └── gdb_run_node_example1.txt
└── src
    ├── example1.cpp
    ├── example2.cpp
    └── example3.cpp

4 directories, 14 files
```

10. ábra Egy csomag jegyzékszerkezete [11]

A rendszerben minden egyes csomag, csomópont a 10. ábrán látható módon épül fel. A chapter3_tutorials nevű jegyzék tartalmazza a komponens állományait, a jegyzék neve pedig csomópont nevét jelöli, az adott komponens ezen a néven fog létezni a rendszerben. Ezen jegyzék további aljegyzékeket tartalmaz, melyek a csomópontra vonatkozó beállításokat, illetve forráskódokat tartalmaznak. A forráskódok az src nevű jegyzékben foglalnak helyet, az ábra esetében cpp kiterjesztésű állományok, tehát c++ forráskódok szerepelnek.

Az alábbi 3. táblázat a rendszer fontosabb utasításait tartalmazza [13]:

3. Táblázat A rendszer fontosabb utasításai

Utasítás	Jelentés
roscore	A központi master elindítása
catkin_init_workspace	ROS munkaterület létrehozása
catkin_make	ROS csomagok fordítása
rospack find [package_name]	A package_name nevű csomag keresése
roscd [locationname[/subdir]]	A [locationname[/subdir]] ROS csomagra váltás
catkin_create_pkg <package_name> [depend1] [depend2] [depend3]	A package_name nevű csomag létrehozása, a depend1, depend2, depend3 függőségekkel
rosnode list	A rendszerben jelenlévő, éppen futó állapotban lévő csomópontok listázása
rosrun [package_name] [node_name]	A package_name nevezetű csomag node_name nevű csomópontjának elindítása
rostopic echo [topic]	A topic nevezetű üzenetsor üzeneteinek megjelenítése
rosservice list	Aktív szervizek listázása
rosservice call [service] [args]	A service nevű szerviz hívása az args argumentumokkal

2.3.2 Az OpenRTM-aist

[14] [15] [16] [17]

Az OpenRTM egy japán fejlesztésű, nyílt forráskódú middleware (köztes réteg) robotrendszer. Ezen middleware az operációs rendszer és az alkalmazás között helyezkedik el, biztosítja a futtató környezetet, futtatható kódot állít elő.

Az OpenRTM lehetőséget biztosít robotok számítógépes hálózaton keresztüli összekapcsolására. Előnyei közé tartozik, hogy mind Linux, mind pedig Windows operációs rendszeren is használható, támogatja a Java, C++ és Python programozási nyelveket, a fejlesztők feladatát pedig magas szintű grafikus eszközökkel segíti. Ezen eszközök segítségével lehetősége van a programozónak megadott paraméterek alapján osztályvázat generálni, illetve az elkészült modulokat egymással összekapcsolni a teljes rendszer működtetése érdekében. Ezen szemlélet szerint az adott feladatot megvalósító rendszer több OpenRTM komponensből áll, melyek közötti kommunikációt maga a keretrendszer valósítja meg, így a komponensek lehetnek azonos hálózati csomópontban vagy különbözőekben is. A távoli komponensek együttműködésére távoli eljáráshívásokat használ.

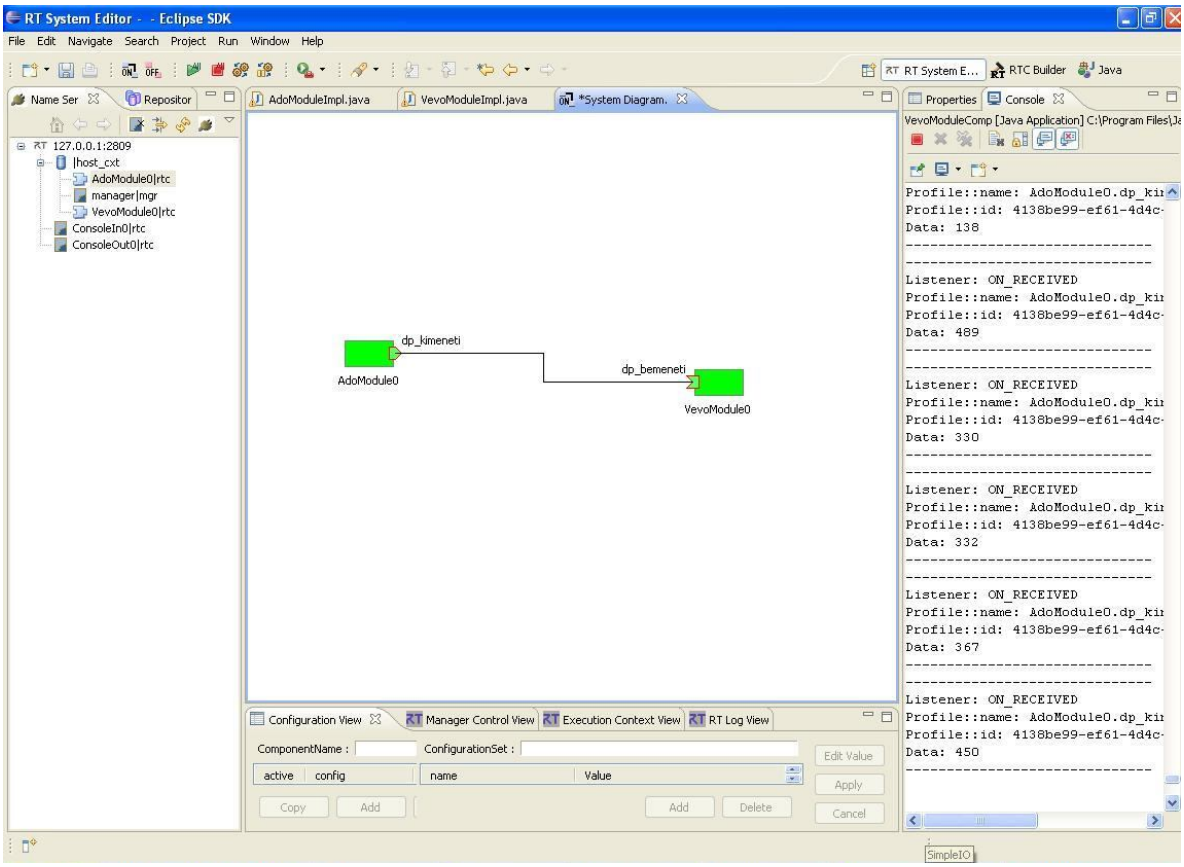
Egy OpenRTM komponens különböző függvényekből, metódusokból épül fel, melyek adott események bekövetkeztekor kerülnek meghívásra. Ilyen függvények pl.: az onInitialize, onStartup, onActivated, onExecute függvények, melyek törzsét a mi feladatunk definiálni az adott feladattól függően. Az onExecute metódusba olyan programkód kerül, amely mindig adott időközönként végrehajtódik, ilyen lehet például a robot távolságérzékelői által szolgáltatott értékek adott időközönkénti lekérdezése.

Néhány segédprogram, amely a fejlesztő feladatát könnyíti:

- *RTC Bulider*: grafikus felületet biztosít komponens létrehozására, a megadott paraméterek alapján projektet, osztályvázat generál. Segítségével megadható a komponens neve, kategóriája, verziója, illetve beállíthatunk adat portokat, szolgáltatás portokat és a használni kívánt programozási nyelvet is. A megadott paraméterek alapján a kiválasztott programozási nyelven ez az eszköz generálja a forráskódot és a szükséges fájlokat, osztályvázakat. Lehetőséget biztosít kódgenerálás utáni paraméter módisításra is, ebben az esetben a módisítások elvégzése után újragenerálja a projekt megfelelő részeit.

- *RT System Editor*: segítségével grafikus felületen irányíthatjuk komponenseink működését, viselkedését, állíthatjuk állapotaikat, illetve összeköthetjük az egyes modulokat egymással. Segítségével grafikus felületen indítható el, állítható le, újraindítható, törölhető az adott modul. Az egyes komponensek állapotainak jelölésére különböző színeket használ, a zöld az aktív állapotot, a piros pedig a hibaállapotot jelöli.

A következő 11. ábrán a szerkesztő felület látható:



11. ábra Az RT System Editor grafikus szerkesztő felülete

Néhány szó a 11. ábrán látható, általam készített demo-ról:

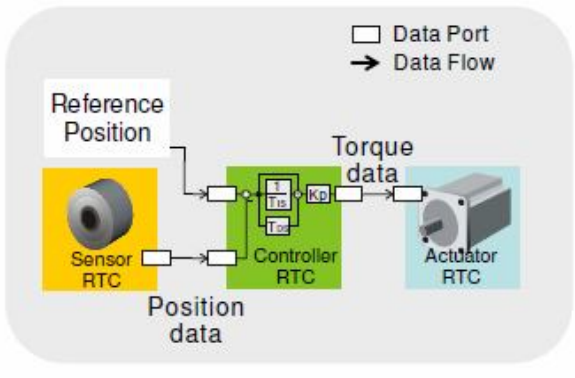
Két komponens látható működés közben (színük zöld --> állapotuk aktív) melyek közötti kommunikációt adatportok valósítják meg. Az egyik komponens az AdoModule0, amely célja, hogy egy véletlen értéket (egész számot 100 és 500 között) generáljon és azt elküldje a VevoModule0-nak. A kommunikáció tehát portok segítségével történik, a dp_kimeneti az adó modul, a dp_bemeneti pedig a vevő modul adatportja. Minkét adatport short típusú, az AdoModule0 a kimeneti portjára kiírja a generált véletlen számot, majd a VevoModule0 a bementi portján fogadja ezt az értéket és kiírja a konzolra. A példa esetében a két komponens helyi gépen fut, de akár a világ két különböző pontján lévő számítógépen is futhatnának.

2.3.2.1 A komponensek között kommunikáció

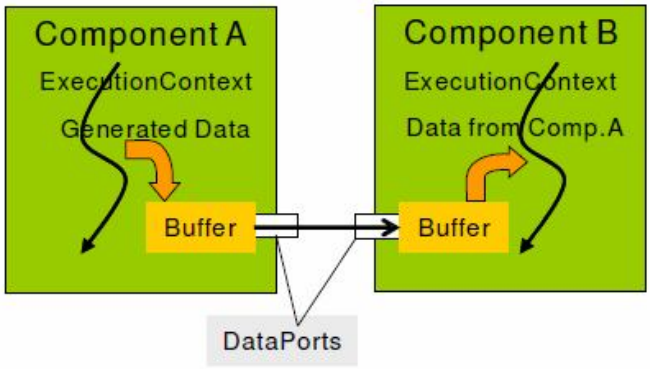
A komponensek közötti kommunikáció megvalósítására a rendszer kétféle lehetőséget, az adat portokat és a szerviz portokat kínálja.

Az adatportok [17]

Az adatportok olyan változókat jelölnek, melyeket egy komponens be- vagy kimenetéül szolgálnak. Ebben az esetében annyi különböző adatportra van szükség, amennyi különböző változó értékét befolyásolni szeretnénk. Az adatportok lehetnek bemeneti és kimeneti portok. A kimeneti adatport egy komponens kimenetéül, a bemeneti adatport pedig egy komponens bemenetéül szolgál. Például ha vezérelnünk kell egy motor fordulatszámát és egy LED ki/bekapcsolását, akkor a felhasználói felület komponensének van két int típusú kimeneti adatport, egy a fordulatszámra (egész szám), egy pedig a led működtetésére (szintén egész szám legyen, 0 mikor kikapcsoljuk, 1 ha bekapcsoljuk). A motor vezérlését és a LED működtetését megvalósító komponensnek pedig kettő, szintén int típusú bemeneti adatportja van. A 12. és 13. ábrák két komponens közötti kommunikáció megvalósítását szemlélteti adatportok segítségével. A két komponens, a szenzor (érzékelő szerv) és a motor (beavatkozó szerv) egy-egy adatporttal, a szenzor kimeneti a motor pedig bemeneti porttal rendelkezik.



13. ábra Példa adatport alkalmazására [17]

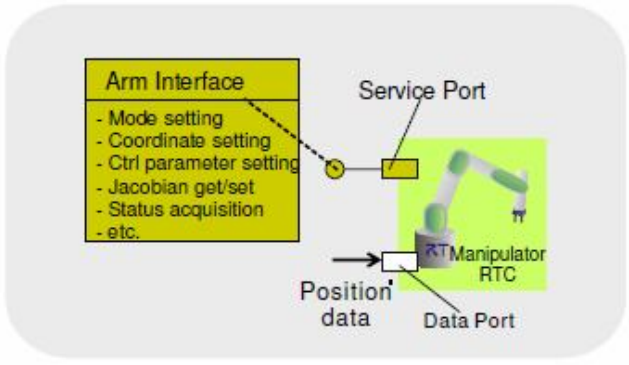


12. ábra Adatport két komponens között [17]

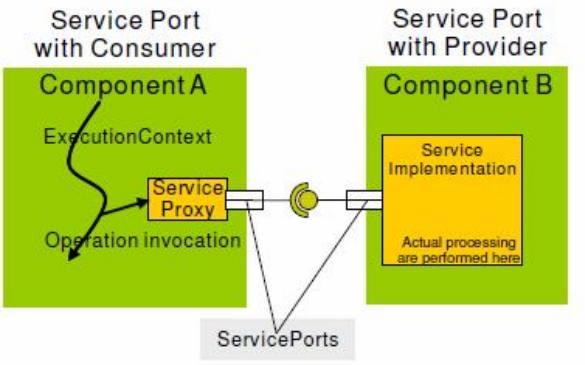
A szervíz portok [17]

A szerviz portok két típusát, a szolgáltatót és a fogyasztót különböztetjük meg. Mindkét esetben interfészeken keresztül szolgáltatnak funkciókat. Nagy különbség az adatportokhoz képest, hogy itt nem ki- és bemeneti változókat valósítanak meg, hanem a távoli osztály metódusait használhatjuk úgy, mint ha azok helyben lennének. Az OpenRTM interfészek definiálásra az IDL (Interface Definition Languange)-t használja. Ebben az IDL kiterjesztésű fájlban megadott metódusok lesznek az a metódusok, melyek távolról hívhatók úgy, mint ha azok helyben lennének. Ezen IDL fájl alapján az OpenRTM szerkesztő felülete generálja a szükséges osztályokat, osztályvázakat.

A 14. és 15. ábrák két komponens közötti kommunikáció megvalósítását szemlélteti szervizportok segítségével.



14. ábra ábra Példa szervízport használatára [17]



15. ábra Szervízport két komponens között [17]

2.4 Mikrokontrollerek

[18]

A mikrovezérlő, vagy más néven mikrokontroller tulajdonképpen egyetlen soklábú integrált áramkör, mely egyetlen áramköri tokban tartalmazza a központi vezérlő egységet, az adatmemóriát, a különböző perifériákat, illetve a perifériaillesztő áramköröket. Különböző beágyazott rendszerekben használatosak, melyekben a vezérlés szerepét töltik be, egy adott feladatot hajtanak végre. Tehát a mikrovezérlő egy adott feladatot megvalósító áramkör, amely küldönböző perifériákkal van ellátva. A vezérlési feladatot megvalósító program a mikorvezérlő flash memóriájában van tárolva, amely könnyedén törölhető illetve szükségesetén újraprogramozható, így rugalmassá téve ezen eszközök használatát. Több féle archtektúrájú mikorvezérlő létezik, mindegyik más és más programozási módót, fejlesztőkörnyezetet igényel, ilyen elterjedtebb mikorvezérlő típusok az AVR, PIC, illetve az ARM.

2.4.1 AVR

[19]

Az AVR az Atmel cég által 1996-ban kifejlesztett módosított Harvard-architektúrájú 8 bites RISC típusú egycsipes mikrovezérlő. Az AVR volt az első mikrovezérlő-család, amelyben csipre integrált flash memóriát kezdtek használni a programtárolásra.
Az AVR készítői nem adnak végleges választ azzal kapcsolatban, hogy az AVR kifejezés mit jelent. Egyes vélemények szerint azonban, két magyarázat is adható a név eredetére. Az egyik megközelítés szerint, az eredeti struktúrát kidolgozó két egyetemista nevéből származtatható. Alf-Egil Bogen és Vegard Wolan neveinek kezdőbetűiből → Alf-Vegard-RISC. A másik elmélet szerint a név a következő szavakból képzett mozaikszó: Advanced Virtual RISC.

Az AVR-ek általában négy széles csoportba vannak osztályozva:

- tinyAVR
- megaAVR
- xMega
- Felhasználásspecifikus AVR

2.4.2 PIC

[20]

A PIC mikrokontroller a Microchip Technology cég által kifejlesztett Harvard architektúrájú programozható eszköz. Ez a mikrokontroller csökkentett utasításkészletű (RISC architektúrájú), ennek köszönhetően az utasításai gyorsan elsajátíthatóak. Ezen mikrovezérlők tervezésénél a minimális bonyolultságra törekedtek.

Jellemzőik:

- Kompakt kivitel. Ha csak áramot kap, már akkor is működik! Nincsen szükség bonyolult hardverelemekre, hogy ellássa a feladatát.
- A kód- és az adatmemória szétválasztása
- Kisszámú és fix hosszúságú utasítás.
- Minden adatmemória rekesz egyben regiszterként is használható. Egyes regiszterek speciális jelentéssel rendelkeznek a külső hardver elemek vezérlésében.
- Hardveres veremkezelés a szubrutinhívásokhoz.
- Mivel az utasításokban kódolva van, hogy melyik memória elemet címezzük, ezért nagyon kicsi a címezhető memóriaterület. Ezen segít a bankkezelési mód.
- Modulszerű perifériakezelés, vagyis ugyanazok a perifériamodulok találhatóak a különböző típusú PIC-ekben feltéve, hogy rendelkeznek vele. A különböző PIC típusok leginkább a perifériák fajtáiban és a kivezetések számában térnek el.

A többi processzorral szemben a PIC mikrovezérlőnél nincs különbség adatmemória és regiszter között, RAM-ként szolgál minden általános célú regiszter.

2.4.3 ARM

[21][22]

Az ARM egy 32 bites RISC processzor architektúra, amelyet nagyon sok helyen használnak beágyazott vagy kisteljesítményű rendszerekben. Az ARM architektúra fejlesztése 1983 -ban kezdődött az Acorn Computers Ltd. Cég laboratóriumában, ahonnan a nevét is kapta (Acorn RISC Machine). Energiatakarékosságuk miatt az ARM architektúrájú CPU-k a vezetők a hordozható elektronikai piacon, ahol az alacsony energiafogyasztás fontos tervezési szempont.

Az első eredmények 1985 -ben születtek, ARM1 név alatt de ez a típus nem jutott el a sorozatgyártásig. Ez a típus rendelkezett az alap adatfeldolgozó, byte, word és multi-word load/store utasításokkal.
Az első sorozatgyártásig eljutott típus az 1986-ban megjelent ARM2 volt, amely 32 bites adatbusszal rendelkezett, ugyancsak 26 bites címtartományt tudott kezelni, 16 32 bites regisztere volt. Ez a típus már rendelkezett szorzás utasítással és támogatott segédprocesszort is. Ez volt a világ legegyszerűbb 32 bites mikroprocesszora, csak 30000 tranzisztort tartalmazott.
Az utód ARM3 már rendelkezett 4KB gyorsítótárral ami nagyon sokat javított a teljesítményén és már 32 bites címtartományt tudott kezelni. Az ARM4 bevezetett egy új kiváltságos üzemmódot, amely felhasználó módú regiszetereket használ, valamint a T változatokban megjelent egy új utasítás, amellyel Thumb üzemmódba lehetett állítani a processzort.
Az ARM processzorok fejlődését ma már nagyon nehéz követni hiszen szinte minden chipgyártó rendelkezik valamilyen ARM alapú processzor családdal. A kis órajel ciklus ellenére ezekkel a megoldásokkal az ARM sikeresen felveszi a versenyt más sokkal komplexebb és gyorsabb processzorokkal.

3. A robotperiféria tervezése, megvalósítása

Ebben a fejezetben a bináris események továbbítására, fogadására alkalmas robotperiféria célját, tervezését, megvalósítását, a megvalósítási lehetőségeket illetve a választás szempontjait ismertetem.

3.1 A kitűzött cél

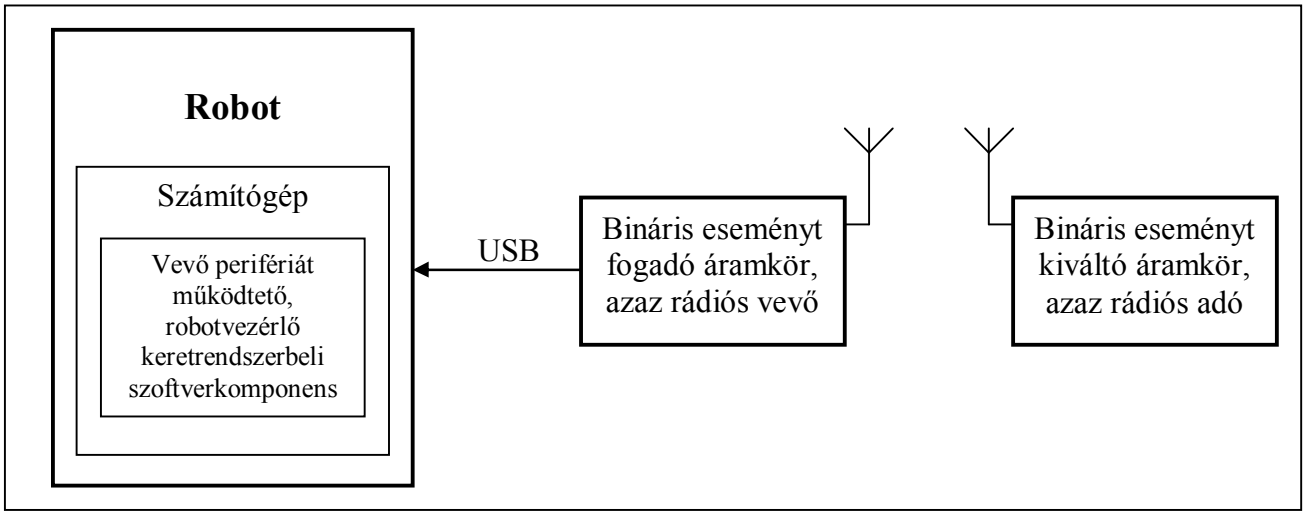
A cél egy bináris események küldésére és fogadására alkalmas robotperiféria tervezése, amely könnyen, minden egyéb, a robotban történő átalakítás nélkül illeszthető ahhoz. Ezen periféria/perifériák célja, hogy segítségükkel valamilyen bináris esemény generálható, előidézhető amely hatására a robot egy adott funkciót, feladatot végre hajt. A bináris eseményt a felhasználó generálja, idézi elő az áramkör segítségével, ilyen esemény lehet például a „csengetés” esemény, amely hatására a robot azon ajtóhoz pozícionál, ahol a „csengő”, vagyis a bináris eseményt kiváltó periféria el van helyezve. Bináris esemény alatt egy kétállapotú információt értek, tehát, hogy az adott esemény létrejött-e vagy sem. Ezen információk alapján a robotperiféria két hardverösszetevőt (beágyazott rendszert) takar, az egyik maga az adó, a másik pedig a vevő áramkör, de az a lehetőség sem kizárandó, mikor a vevő áramkörhöz több bináris eseményt kiváltó adó áramkör csatlakozik.

Az adó áramkör segítségével generálható a bináris esemény, melyet a vevő áramkör fogad, majd a vevő áramkört kezelő számítógép oldali robotvezérlő keretrendszerbeli szoftverkomponens valósítja meg a robot vezérlését, irányítását az adó által elküldött bináris jeltől függően. Az adó és a vevő perifériák (beágyazott rendszerek) között tehát vezeték nélküli kommunikációs kapcsolat van, ezen kapcsolaton keresztül történik a kiváltott bináris események, adatok továbbítása a robot felé, a vevő áramkör pedig valamilyen módon kapcsolódva a robothoz vezetékes összeköttetésben áll azzal.

Mind az adó, mind a vevő áramkör, periféria jól megfogalmazott, konkrét feladatot lát el, így szükség van valamilyen céláramkörre, mellyel a feladat megvalósítható, erre a legmegfelelőbb egy mikrovezérlő áramkör. Mivel az adó és a vevő periféria nincs egymással közvetlen vezetékes összeköttetésben, így a két periféria közötti kommunikációs kapcsolatot egy vezeték nélküli kommunikációs egység fogja megvalósítani, amely lehet bluetooth, wifi, vagy akár 433 Mhz-es rádiós modul is.

További fontos elvárás a perifériával szemben, hogy a robothoz/robotokhoz (Ethon, Turtlebot) azok nagyobb átalakítása nélkül csatlakoztatni, beépíteni lehessen. Mivel mindkét robot számítógépén található USB port, így a legcélszerűbb, ha a vevő periféria USB porton keresztül csatlakozik, így nincs szükség a robotok működtetését megvalósító mikrovezérlős áramkörök módosítására, átalakítására, újratervezésére.

Egy első elgondolás alapján felépített blokkvázlatot az alábbi 16. ábra szemlélteti.



16. ábra A rendszer lehetséges felépítésének blokkvázlata

A 16. ábrán látható módon, a bináris esemény kiváltására és a bináris esemény fogadására képes periféria két különböző áramkör, melyek valamilyen vezetéknélküli kommunikációs összekötettésben vannak egymással. A vezetéknélküli kommunikációs összekötettés miként je függ azon területtől, ahol a perifériát használni, működtetni fogják, azaz az ELTE etológia tanszékének folyosójától. A bináris esemény fogadására képes periféria USB kapcsolaton keresztül csatlakozik a robot/robotok számítógépéhez, a vevőperiféria által fogadott adatokat (binárs eseményt/eseményeket) ezen kapcsolatonkeresztül továbbítja a periféria működtetésést megvalósító robotvezérlő keretrendszerbeli szoftverkomponensnek, amely szoftverkomponens a fogadott bináris esemény alapján vezréli majd a robotot, robotokat.

3.2 Mikrokontroller típusának megválasztása

Mivel a tervezendő robotperiférиák jól meghatározott feladatokat látnak el, bináris jelek továbbítását, fogadását végzik, így szükség van olyasféle áramkörre, mellyel ez a feladat megvalósítható. Erre a célra mikrovezérlő alkalmazása a legmegfelelőbb, amely perifériáit, portjait megfelelően programozva ezen feladatot megvalósítja.
Az előző fejezetben kifejtett mikrovezérlő architektúrák közül az AVR típust fogom alkalmazni a megvalósítás során, az AVR család pontos típusát pedig a robotperiférиával szemben támasztott elvárások, követelmények alapján választom majd ki. Azért az AVR családra eset a választásom, mert tanulmányaim során főként ARM mikrokontrollerekkel foglalotoskodtam, AVR-ekkel csak nagyon keveset, így szeretnék ezen mikrovezérlők működésével, felépítésével, programozásával is megismerkedni.

Elvárások a tervezendő robotperiférиával szemben:

- valamilyen bináris eseményt lehessen vele előidézni, azaz szükség van valamiféle, nyomógombra, kapcsolóra, amellyel az esemény kiváltható
- ezen esemény által valamilyen bináris jel áll elő, melyet vezeték nélküli kommunikációs kapcsolaton keresztül továbbít, azaz szükség van arra, hogy a mikrovezérlő rendelkezzen UART/USART perifériával
- az esemény létrejöttét valamilyen módon jelezze a felhasználó számára, azaz legyen rajta visszajelző LED

A fentebb említett elvárások alapján a mikrokontrollernek az alábbi perifériákkal kell rendelkeznie:

- Bemenet, a nyomógombok kezelésére
- UART/USART az adott esemény alapján létrejött bináris jel továbbítására, melyhez majd egy vezeték nélküli kommunikációs modul csatlakozik
- Kimenet, a LED működtetésére

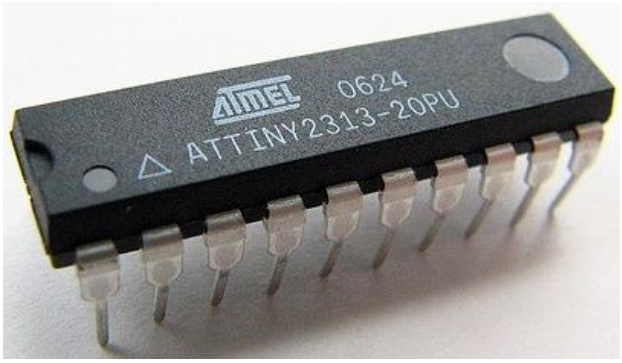
Ezen elvárások alapján az AVR-ek világában a Tiny család az, amely már rendelkezik a megvalósításhoz szükséges, fentebb felsorolt perifériákkal. A Tiny-től léteznek több szolgáltatást, funkciót nyújtó AVR-ek is pl.: a Mega család, de ezen család által kínált funkciókra (pl. ADC) előreláthatólag nincs szükség, így a Tiny család tagjai közül a 2313-

as típust választom, amley már elegendő erőforrással rendelkezik a feladat megvalósításához.

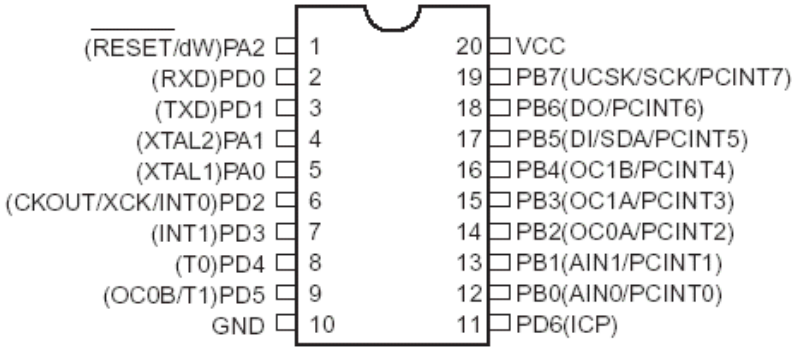
A Tiny2313 típus adatlapja [23] alapján a következő perifériákkal, erőforrásokkal rendelkezik:

- 1 darab 8 bites időzítő/számláló
- 1 darab 16 bites időzítő/számláló
- 4 darab PWM csatorna
- Beépített analóg komparátor
- Programozható watchdog időzítő
- USI, azaz Universal Serial Interface
- Full duplex USART
- 20 Mhz-es órajel

Az alábbi ábrákon a Tiny2313 típusú mikrokontroller fényképe (17. ábra), illetve lábkiosztása (18. ábra) látható:



17. ábra A Tiny2313-as típusú mikrokontroller fényképe [24]



18. ábra A Tiny2313-as típusú mikrokontroller lábkiosztása [25]

3.3 A robothoz történő csatlakoztatási lehetőség megválasztása

A bináris események fogadására alkalmas vevő periféria célja, hogy a fogadott esemény/események alapján vezérelje a robotot, így annak valamilyen módon a robothoz kell csatlakoznia. Az előző fejezetekben taglalt, perifériával szemben támasztott elvárások között fontos szempont, hogy az egyszerű módon, a robotban illetve a robot áramköreiben történő nagyobb átalakítás nélkül csatlakozatható legyen ahhoz.

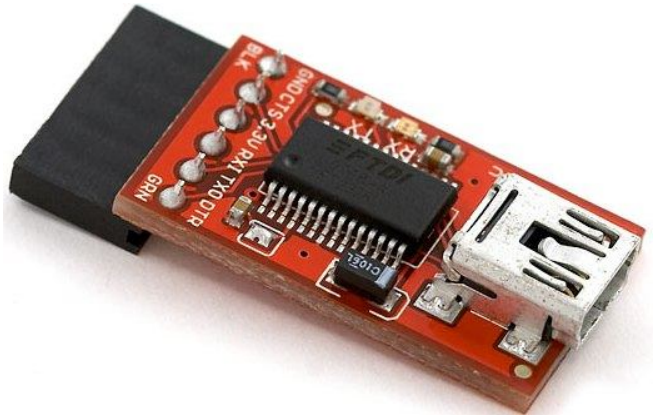
A vevő periféria közvetlenül csatlakozhatna a robot fő áramköréhez, azaz a működtetését megvalósító mikrovezérlőhöz (illetve annak egy portjához), de ebben az esetben szükséges lenne újratervezni a robotok fő paneljeit, amely mind Ethon illetve mind Turtlebot esetén annak gyártója által tervezett, ezért a kapcsolási rajz és a nyomtatott áramköri terv nem feltétlenül elérhető így ezen megoldási lehetőség nem a legmegfelelőbb.

Mivel mind a két robot tartalmaz számítógépet, Ethon esetében egy, a robot feladataihoz méretezett hardvererőforrásokkal rendelkező teljesértékű számítógépet, Turtlebot esetében pedig egy laptopot, így a legcélszerűbb ha ezen számítógépek USB portjához kapcsolódik a bináris események fogadására képes robotperiféria. Ebben az esetben nincs szükség a robotok áramköri paneljeiben történő módosításra, hanem egyszerűen USB eszközként lehet csatlakoztatni a robotperifériát, amely nyomtatott áramköre majd egy műszerdobozban foglal helyet majd ezen műszerdobozt szükséges rögzíteni a robot vázához.

A mikrovezérlő programozása szempontjából kétféle lehetőség kínálkozik a számítógéphez történő USB csatlakoztatás megvalósítására. Az egyik lehetőség, hogy a vevő periféria USB HID eszközként jelentkezik a rendszerben, a másik pedig, hogy egy sorosportként jelenik meg. Mivel mind a két robot saját perifériái sorosportként vannak jelen a rendszerben (Ethon esetén 3 darab, Turtlebot eseténben pedig 1 darab soros port), azokon keresztül van lehetőség a vezérlésükre, így én is ezt a lehetőséget választom.

Ennek a megvalósítására újabb két lehetőség kínálkozik. Az egyik lehetőség, hogy a mikrovezérlő USB perifériája csatlakozik a számítógép USB portjához és az USB-sorosport konverzió szoftveresen valósul meg a mikrovezérlő szoftverében (virtuális soros port), a másik lehetőség pedig, hogy a mikrovezérlő UART perifériája egy az UART-USB konverziót megvalósító áramkör segítségével kapcsolódik a számítógép USB portjához. A piacon több ilyen integrált áramkör is létezik, az egyik ilyen például az FT232 típusnevű integrált áramkör, amely külön nyomtatott áramköri modulként is megvásárolható. Az

utóbbi megvalósítási lehetőséget választom, tehát, hogy az UART-USB konverziót egy külön modulként megvásárolható integrált áramköri elem valósítja meg, amely fényképe az alábbi 19. ábrán látható.



19. ábra Az FT232 modul fényképe [26]

Ezen modulnak több előnye is van, egyrészt nyomtatott áramköri lapra szerelve helyezkedik el rajta az FT232 típusú integrált áramkör (amely az USB-sorosport konverziót, szintillesztést valósítja meg), másrészt pedig minden olyan áramköri alkatrész is helyet foglal rajta, amely az IC megfelelő működéséhez szükséges. Helyet foglalnak rajta többek között az IC lábkivezetései, a kommunikációs kapcsolat működésének helyességét jelző LED-ek, illetve egy USB csatlakozó is. Az eszköz működéséhez a gyártó holnapján található driver szükséges, illetve a modul mind Windows, mind pedig Linux operációs rendszer esetén használható, sőt mi több az Ubuntu disztribúció kernele beépítve tartalmazza az eszköz működéséhez szükséges drivert, így a modul csatlakoztatás után már meg is jelenik sorosportként (általában ttyUSB0 néven).

3.4 Vezetéknélküli kommunikációs kapcsolat megválasztása

Az adó és a vevő robotperiféria közötti kommunikációs kapcsolat kialakításához vezetéknélküli technológia alkalmazására van szükség, ilyen vezetéknélküli kommunikációs technológia lehet a wifi, bluetooh, illetve az analóg rádiós, például a 433 Mhz-es frekvenciasávban működő rádiós kapcsolat. Ezen kommunikációs kapcsolat kialakításához a választott vezetéknélküli technológiát megvalósító hardvermodult fogok alkalmazni, melyek mind a wifi, bluetooh, és mind a 433 Mhz-en működő rádiós kapcsolat esetén külön nyomtatott áramköri lapkán, kész modulként megvásárolhatóak. Ezen vezeték nélküli kommunikációt megvalósít áramkör, egy kis modulként kapcsolódik a mikrovezérlők megfelelő portjaihoz (azaz az adó és a vevő periféria áramköréhez).

A megvalósítás során a 433 Mhz-es frekvenciasávban üzemelő rádiós adó-vevőt fogom alkalmazni, amely előnye a wifi, illetve a bluetooth technológiával szemben, hogy minden egyéb konfigurálás nélkül használható. Bluetooh és wifi modul esetében az eszköz megfelelő konfigurálása szükséges, melyet AT utasítások segítségével lehet megvalósítani (sorosporton keresztül), konfigurálni kell, hogy az eszköz milyen módban üzemeljen (például wifi modul esetében, hogy AP vagy host módban), be kell állítani, hogy mely másik eszközhöz csatlakozzon. Wifi modul esetén, hogy milyen nevű hálózathoz kapcsolódjon és milyen IP címre/címekre küldjön adatot, bluetooth modul esetén pedig, hogy mely másik bluetooth modulhoz kapcsolódjon, annak a másiknak mi az elérhetősége (neve, MAC címe).

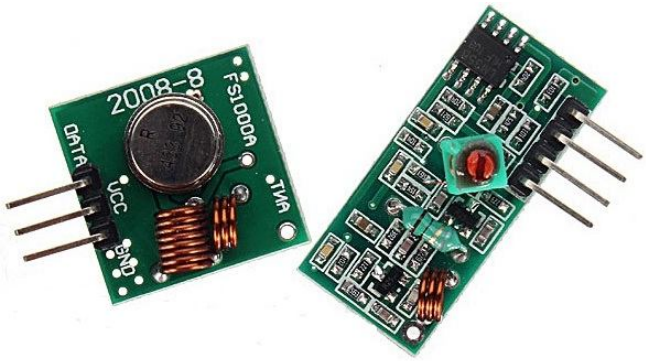
A 433Mhz-es sávban működő rádiós adó-vevő esetén ilyen konfigurálási beállításokra nincs szükség, elegendő a két eszközt feszültség alá helyezni és rögtön el a két eszköz közötti kommunikációs kapcsolat. További előnye, hogy egy vevő modullal akár több adó modul is kommunikálhat, hiszen csak az a lényeg, hogy mindegyik a 433 Mhz és a közeli frekvencián sugározzon. Hátrányai közé tartozik, hogy a jelerősség nagyban az alkalmazott tápfeszültség és a távolság függvénye, illetve a továbbítandó adatok többszöri továbbítása szükséges (távolság függő), általában 20-200 méter közötti az a távolság, amely ezen modulok segítségével áthidalható.

Mivel a tervezendő bináris események kiváltására és fogadására alkalmas robotperifériát elsősorban az ELTE etológia tanszékének folyosólyán fogják majd működtetni (a folyosó alaprajza, elrendezése a 2. ábrán látható), így több szempontból is előnyös a 433 MHz-es rádiós kapcsolat alkalmazása. Ezen folyosón több wifi-s és bluetooth-os rendszer is

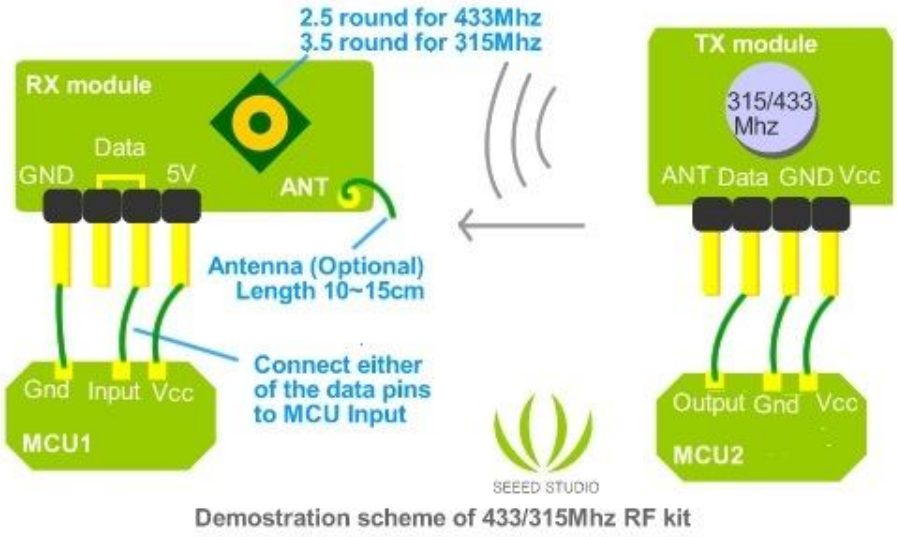
működőben van, illetve kiépítés alatt van egy IP kamerás rendszer is, így előnyösnek tartom ezen rendszerek által használt rádiófrekvenciás sávtól teljesen eltérő frekvenciasáv használtatát, így nem lesz interferencia ezen rendszerek és az általam tervezendő robotperiféria között .

3.4.1 Az alkalmazott vezetéknélküli kommunikációs modul

Az előző, 3.4 fejezetben leírtak alapján 433 Mhz-es vezetéknélküli kommunikációs modult fogok alkalmazni a megvalósítás során. A piacon több ilyen, kész modulként megvásárolható áramkör ismegtalálható, melyek elsősorban az általuk áthadalható távolságban, illetve a kommunikűciós során alkalmazott kódolásban térnek el.
A megvalósítás során az elterjedtebb FS1000A típusú 433 Mhz-es frekvenciasávban üzemelő rádiós adót és vevőt fogom alkalmazni, melyek fényképe és bekötési rajza az alábbi 20. és 21. ábrákon látható:



20. ábra A 433 Mhz-es rádiós adó (bal oldal) és vevő (jobb oldal) fényképe [27]



21. ábra ábra A 433 MHz-es rádiós adó-vevő bekötési rajza [28]

Az alábbi 4. és 5. táblázatok az adó és a vevő modul fontosabb technikai paramétereit tartalmazzák, azok adatlapja [29] alapján:

4. Táblázat Az adó modul technikai paraméterei

Üzemi tápfeszültség:	3V-12V DC
Áramfelvétel:	20-28 mA
Működési frekvenciasáv:	315 MHz - 433.92 MHz
Moduláció:	ASK/OOK
Érzékenység:	-103 dB
Ajánlott antenna típus:	17 cm botantenna, vagy 32 cm spirál antenna
Méretek:	19 * 19 * 4 mm

5. Táblázat A vevő modul technikai paraméterei

Üzemi tápfeszültség:	5V DC
Áramfelvétel:	4 mA
Működési frekvenciasáv:	315 MHz - 433.92 MHz
Moduláció:	OOK/ASK
Érzékenység:	-105 dB
Ajánlott antenna típus:	17 cm botantenna, vagy 32 cm spirál antenna
Méretek:	30 * 17 * 4 mm

Ezen modulok alkalmazásával az áthidalható távolság, adó modul 12V-os tápfeszültséggel történő megtáplálása esetén körül-belül 200 méter. Ez a távolság sokkal nagyobb, mint az áthidalandó 30 méteres távolság (ELTE Etológia tanszékének folyosója), tehát a modulok által áthidalható távolság nem fog szűk keresztmetszetet jelenti a kommunikáció során. Az így elérhető adatátviteli sebesség körül-belül 9.6 KB/s, amely egyszerű adatkeretek továbbítására több mint elegendő.

A modulok által alkalmazott modulációs technika jelentése ASK esetén, Amplitude Shift Key, OOK esetén pedig On/Off Key. A modulációk célja, hogy egy tisztán szinuszos jelre (vivő) információt hordozó jelet ültessenek. ASK, vagyis amplitúdóbillentyűzés a

moduláló jeltől függően két amplitúdó között kapcsolgat, ennek az egyik lehetséges módja az OOK, vagyis mikor ki-be kapcsolgatja a vivőt, tehát logikai 1-nél a vivő létezik, logikai 0-nál pedig eltűnik. [30] [31]

Ezen modulok nem rendelkeznek beépített antennával, így a megfelelő antennatípust a tervezőnek szükséges megválasztania. Az adatlap [29] ajánlása szerint vagy 17 cm hosszú botantennát vagy 32 cm hosszú, feltekercselt, spirál alakú antennát szükséges alkalmazni. A modulok pontosan 433.92 Mhz-en üzemelnek, amely hullámhossza az alábbiak alapján határoható meg:

Fénysebesség: 299 792,452 km/s => 299 792 452 m/s
Rádiónk frekvenciája: 433,92 MHz => 433 920 000 Hz
Vagyis: 299 792 452 / 433 920 000 = 0,6908 méter

22. ábra A 433 Mhz-en üzemelő rádiósmodul hullámhosszának számítása

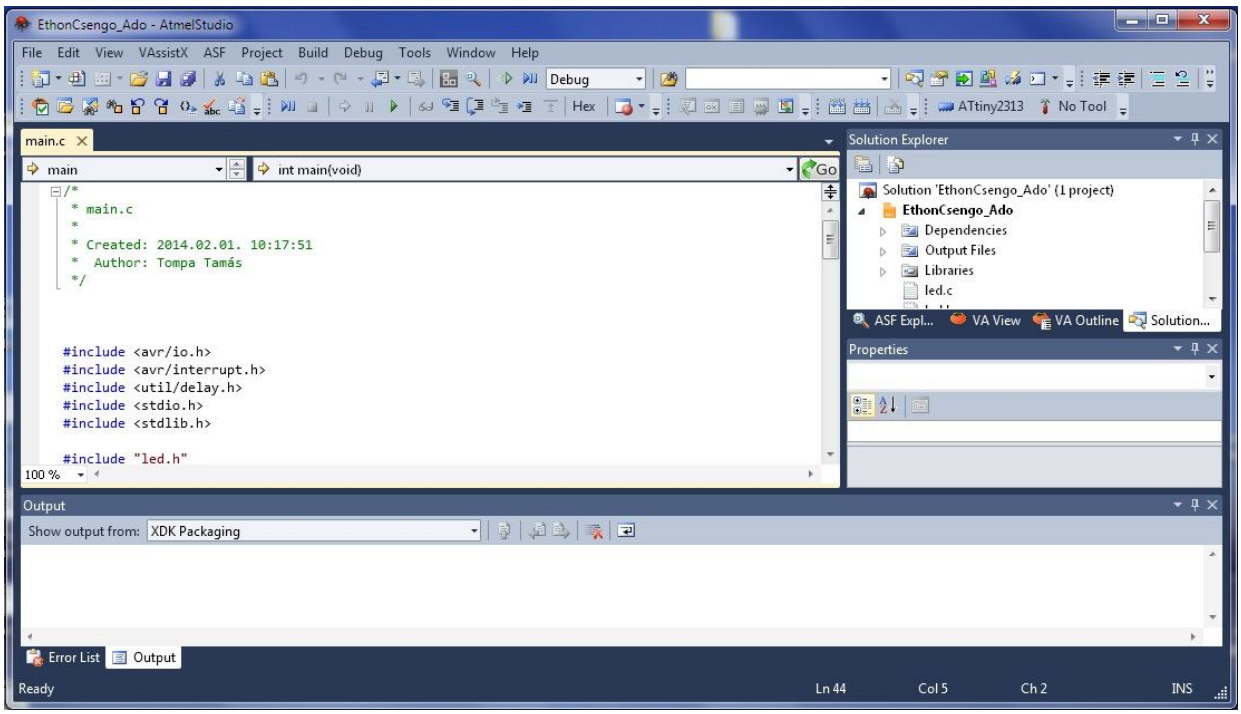
A megfelelő antenna alkalmazása kulcsfontosságú, hiszen a kommunikációs sikerességét nagymértékben befolyásolja, meghatározza. Nem megfelelő antenna alkalmazása esetén előfordulhat, hogy gyengébb jelet vesz a modul, mint egy pont akkora antennával mint ami az előírás szerint szükséges. Ezen a rádiós modulokhoz 1/4-ed hullámhosszúságú antennát ajánlanak, tehát 0,6908 méter / 4 = 0,1727 méter, vagyis körül-belűl egy 173 miliméteres antennát kell alkalmazni a legjobb vétel / adás eléréséhez. Az adatlap janálása egy 17 cm hosszú botantenna, tehát az ajnálás tényleg megfelelő, így ennek megfelelően egy tömör rézvezetékből fogok kialakítani egy ilyen antennát. [32]

3.5 Az alkalmazott fejlesztőkörnyezet

A tervezendő robotperiféríákat működtető mikrovezérlő szoftverének fejlesztéséhez hasznos lehet fejlesztői környezet használata, amely funkciói, szolgáltatásai által számos hasznos eszközt biztosít a fejlesztőnek. Segítségével a mikorvezérlő perifériái paraméterezhetőek, konfigurálhatóak, majd a mikrokontroller konfigurációs beállításai alapján forráskód generálásra biztosít lehetősgét. A fejlesztendő robotperifériában AVR architektúrájú Atmel márkajelzésű mikorvezérlő foghelyet foglalni, a gyártó ezen mikorkontrollerek szoftverének fejlesztéséhez egy ingyenesen letölthető, AtmelStudio

nevezetű fejlesztőkörnyezetet javasol. A fejlesztés során az Atmel Studio 6-os verzióját használtam, melyet a [33]-as forrás által megjelölt hivatkozásról töltöttem le.
Az Atmel Studio sok hasznos funkcióval rendelkezik, amely a fejlesztő munkáját könnyíti, a legtöbb Atmel márkajelzésű AVR mikorvezérlő-családot támogatja, új projekt létrehozásánál a fejlesztéhez alkalmazott mikrokontroller típusa beállítható. Lehetőséget biztosít az elkészült szoftver AVR-be történő „égetéséhez", a megfelelő programozóegység megadásával, annak paramétereinek beállításával, illetve segítségével az AVR biztosítékbitjei beállíthatóak, annak értékei kiolvashatóak.

A szoftver felülete az alábbi árbrán látható:



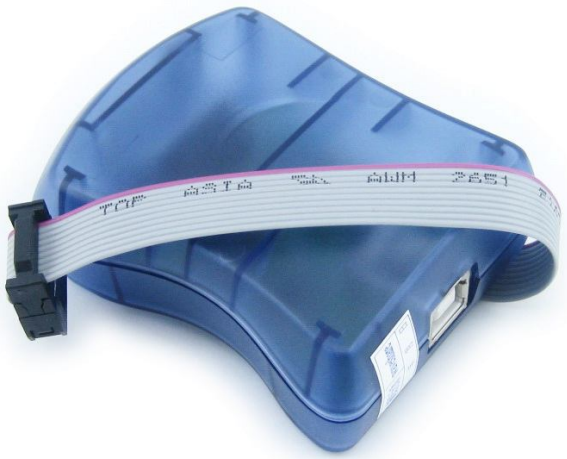
23. ábra Az AtmelStudio 6 fejlesztőkörnyezet felülete

3.6 A fejlesztéshez használt mikorkontroller programozó

Az elkészült mikorvezérlőszoftver mikrokontrollerbe való letöltéséhez az AtmelStudio által támogatott AVRISP mkII típusjelzésű programozót használtam. Ezen eszköz az AVR család legtöbb tagját támogatja, többek között a megvalósítás során általam alkalmazott ATtiny2313-as típust is. Az eszköz ISP 6-os (In-System-Progammer) csatlakozóval van ellátva, amely lehetőséget biztosít a mikorkontroller áramkörben való programozására, azaz a mikorvezérlőt nem szükséges eltávolítani a funkcióját betöltő áramkörből, hanem lehetővé teszi annak a rendzerben történő programozását. A programozó USB-n keresztül

kapcsolódik a számítógéphez, rendelkezik beépített rövidzárvédelemmel, használatához az AVR tápfeszültség alá történő helyezése szükséges.

A programozó fényképe az alábbi ábrán látható:



24. ábra Az AVRIPS mkII progamozó [34]

3.7 Alkalmazott áramkörtervező szoftver

[35][36]

A robotperiféria áramkörének, nyomtatott áramköri rajzának tervezéséhez, megvalósításához elengedhetetlen áramkörtervező szoftver használta. Erre a célra a CadSoft térmékét pontosabban az Eagle nevezetű szoftvert használom majd.

Az Eagle azaz Easly Applicable Graphical Layout Editior, kezdőbetűiból összetevődő mozaikszó, magyarul egyszerűen alkalmazható grafikus nyáktervező. A programot a Német CadSoft GmbH fejlesztett ki, az Eagle támogatja a Windows, Linux és 2004-től Mac operációs rendszereket is, a piaconlévő egyéb más áramkörtervező szoftverekkel szemben gépigénye meglehetősen kicsi. A program használata könnyen elsajátítható és kezelő felülete könnyen átlátható. Alapvetően három fő funkcionális modulból áll: a Schematic Editor-ból – az elvi kapcsolási rajz szerkesztéséhez, a Layout Editor-ból – a kész NYÁK-tervek módosításához, illetve az Autorouter-ből – a nyomatatott áramkör vezetősávjainak automatikus megtervezéséhez. Alkatrészkönytára nagyméretű, számos alkatrésztípus megtalálható benne, illetve lehetőséget biztosít az alkatrészkönyvtár bővítésére is. Az elkészített kapcsolási rajz alapján lehetővé teszi többrétegű NYÁK

tervezését, továbbá rendelkezik automatikus alakatrész elrendezési és automatikus vezetősáv bekötési funkcióval.

3.8 A bináris események

A bináris eseményeket a fejlesztendő robotperifériák fogják megvalósítani, a robotperifériák fő célja, funkciója, hogy eme események generálására, fogadására lehetőséget biztosítson, majd a robotok (Ethon, Turtlebot) vezérlése, adott feladat végrehajtása ezen események bekövetkezése által valósuljon meg. A bináris eseményeket tehát a robotperifériák valósítrják meg, majd a bináris események alapján történő robotvezérlésre pedig a robotvezérlő keretrendszerbeli szoftverkomponens implementálása által valósul meg. Az események a robotperiféria szempontjából konkrét, általam meghatározott adatkereteket, illetve ezen adatkeretek továbbítását jelentik.
A bináris esemény az, melyet a felhasználó az adó periférián lévő nyomógomb segítségével idéz elő, generál. Bináris esemény alatt egy kétértékű információ értendő, tehát a „van esemény”, „nincs esemény” típusú adatok. Ha a felhasználó megnyomja az adó periférián található nyomógombot, akkor a „van esemény” típusú adatkeret jön létre, illetve kerül továbbításra. A bináris esemény, események alapján valósul meg a robot vezérlése, adott feladat végrehajtása. Az ELTE etológusai által meghatározott egyik lehetséges bináris esemény például, mikor a robot az esemény létrejöttének hatására a tanszék ajtajához pozícionál, ha az adó periféria jelzést küldött a robot számára, úgymond egyfajta „csengő” funkciót megvalósítva. Egy másik lehetséges alkalmazás, mikor az adó periféria egyfajta távirányítóként üzemel, azaz nyomógombjai segítségével távvezérli a robtot. Mind a „csengő”, mind perig a távvezérlést megvalósító funkció esetében más és más robotvezérlő keretrendszerbeli szoftverkomponensek implementálásra van szükség.

3.8.1 A „csengetés” esemény

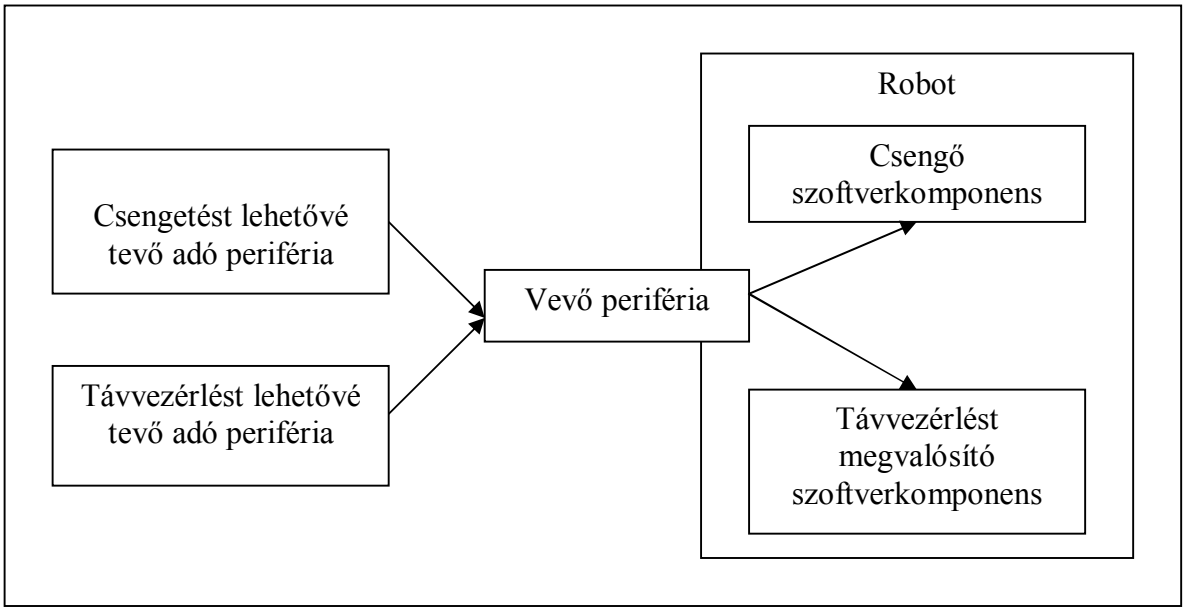
Egy lehetséges bináris esemény például a „csengetés” esemény, amely hatására a robot azon ajtóhoz pozícionál, ahol a csengető (azaz az adó periféria) el van helyezve. Mivel Ethon esetében egy portás robotról van szó, így alapvető funkció lehet az, hogy adott személyt vagy személyeket az Etológia tanszék területére beengedjen. Ha az adott illető a

zárt ajtók mögött lévő tanszékre szeretne például ügyintézés céljából bemenni, akkor az ajtónál felszerelt adó periféria segítségével „csenget” a robotnak, amely ennek hatására az ajtóhoz pozícionál, majd ha az adott személy sikeresen beazonosította, akkor a robot a karjában lévő mágneskártya segítségével kinyitja számára az ajtót, azaz beengedi az illetőt. Tehát ezen funkció (csengő) megvalósítására az adó- és a vevő robotperiféria szolgál: az adón lévő nyomógomb megnyomásának hatására egy a csengetést jelentő adatkeret kerül továbbításra, melyet a robotban elhelyezkedő vevő periféria fogad, ezt az adatkeretet továbbítja USB porton keresztül a robot számítógépének. A robot számítógépén futó, robotvezérlő keretrendszerbeli szoftverkomponens pedig ezen fogadott keret hatására megvalósítja a robot ajtóhoz történő vezérlését, tehát azt, hogy mi is történjen a bináris esemény létrejöttének hatására.

3.8.2 A távvezérlést lehetővé tévő esemény

Egy másik lehetséges bináris esemény, amley a robot távirányítására, távvezérlésére ad lehetőséget. Az adó periféria segítségével, illetve az azon lévő nyomógombok megnyomásával olyan bináris esemény jön létre, amely a robot távvezérlését teszi lehetővé. Ezen feladat, funkció megvalósításához feltehetően egy újabb adó periféria tervezésére, megvalóstására van szükség, hiszen a csengetést megvalósító adó periféria a csengő funkciója lévén fixen van felszerelve az ajtó mellé, illetve funkciója miatt csak egy nyomógomb található majd rajta. Ethon tevvezérlést lehetővé tevő adó periféria szintén egy mikrokontrollert tartalmazó áramkör, amely nyomógombjai megnyomásának segítségével Ethon előre, hátra, jobbra, balra, balra forog és jobbra forog mozgását megvalósító eseményeket továbbít a robotban helyetfoglaló vevő periféria számára. A robot mozgásának vezérlést szintén egy robotvezérlő keretrendszerbeli szoftverkoponens valóítja majd meg. Ezen esetben megoldandó feladat, hogy mind a „csengő” adó periféria és mind a távvezérlést lehetővé tevő adó periféria ugyanzon, a robotban helyetfoglaló vevő perifériához kapcsolódjon.

Az alábbi 24. ábra a „több adó periféria és egy vevő periféria” rendszer felépítésének blokkvázlatát szemlélteti:



25. ábra Több adó periféria, egy vevő periféria esetén a rendszer felépítése

3.9 A bináris esemény kiváltására alkalmas robotperiféria tervezése, megvalósítása

Ezen alfejezet az előző fejezetben ismertetett megvalósítási lehetőségek illetve azok felhasználásával ismerteti a bináris események kiváltásra, küldésére alkalmas robotperiféria célját, tervezését, megvalósítását, működését.

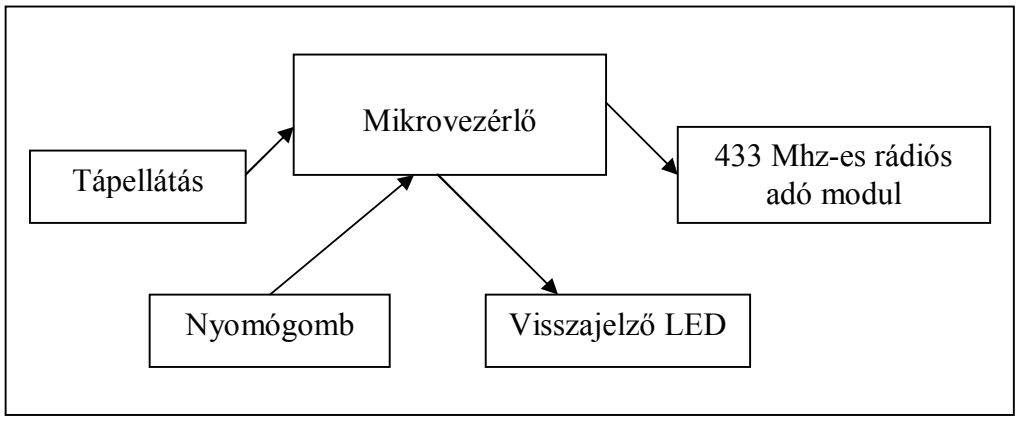
3.9.1 A robotperiféria célja

A bináris események generálására illetve továbbítására alkalmas robotperiféria célja, hogy a felhasználó által valamilyen módon kiváltott bináris eseményt továbbítsa a robot bináris eseményeket fogadó perifériájának számára. Ez a periféria egy mikrovezérlőt (pontosabban egy ATTiny2313 típusú AVR-t) tartalmazó áramkör, amely a felhasználó előidézett bináris eseményt továbbítja a robotban helyet fogalaló, bináris esemény fogadására alkamas robotperiféria felé, melyhez vezetéknélküli kommunikációs modul alkalmazásával kapcsolódik.

A megvalóstás során szükség lesz nyomógombra, amely segítségével az adott esemény a felhasználó által előidézhető, például a nyomógomb megnyomásával, továbbá szükség lesz egy visszajelző LED-re, amely a felhasználó számára ad visszajelzést arról, hogy a bináris esemény létrejötte, az áramkör sikeresen továbbította-e ezt az eseményt vagy sem. Továbbá szükség lesz a 433Mhz-en üzemelő rádiós adó modulra, amely a felhasználó által generált esemény alapján adatkeret továbbító a robotban helyet foglaló vevő periféria számára.

3.9.2 A robotperiféria áramkörének blokkvázlata

Az alábbi ábrán a periféria célja, funkciója alapján felépített áramkör blokkvázlata látható:

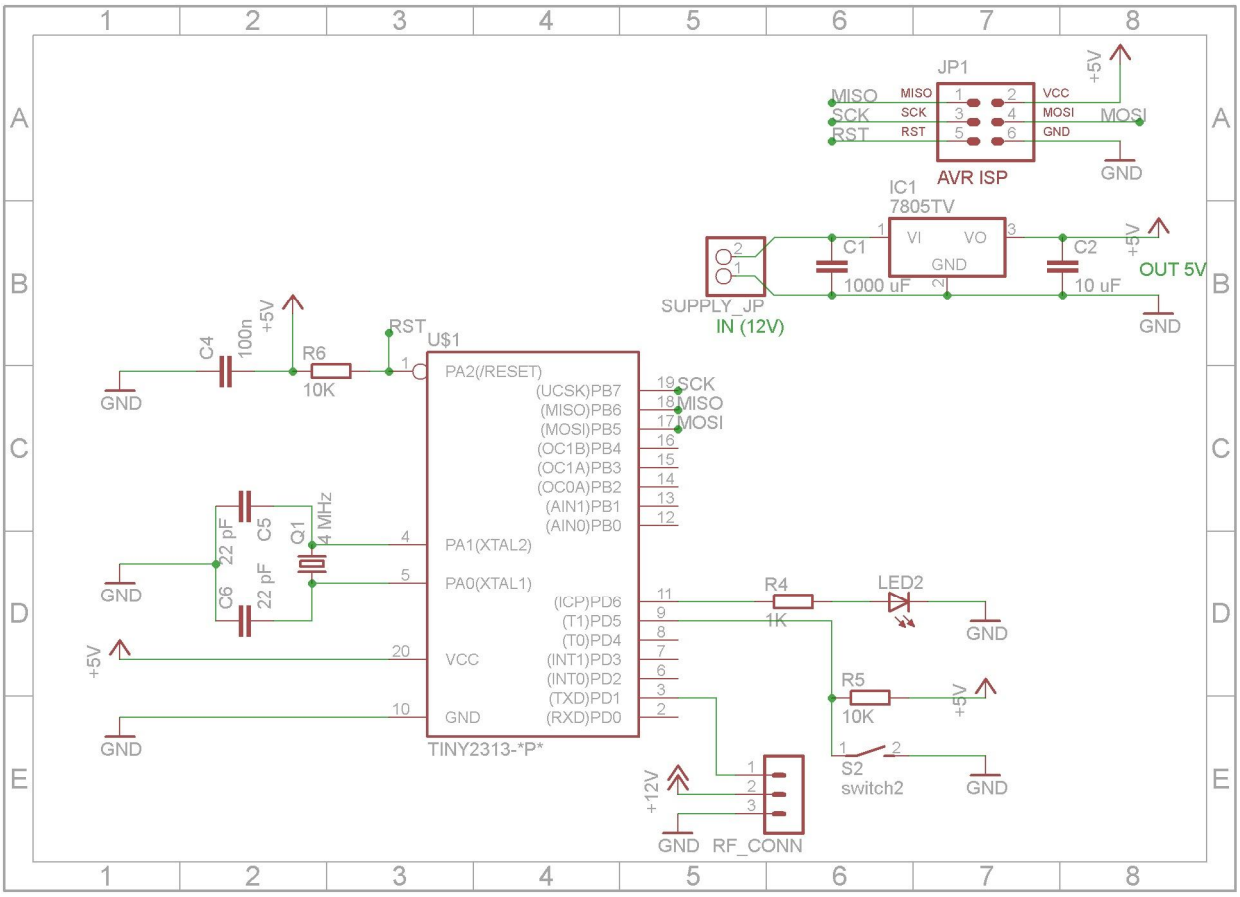


26. ábra A bináris események kiváltására alkalmas periféria áramkörének blokkvázlata

A 25. ábrán látható módon az áramkörben helyet foglal egy mikorkontroller, amely a periféria központi eleme, ezen mikorvezérlő működteti magát a perifériát, valósítja meg a portjaira csatlakoztatott nyomógomb, LED és a 433 Mhz-en üzemelő rádiós adó modul működtetését.

3.9.3 A robotperiféria kapcsolási rajza, működése

Az alábbi 26. ábra a 25. ábrán látható blokkvázlat alapján tervezett bináris esemény előállítására, továbbítására alkalmas áramkör, az Eagle tervező szoftver segítségével szerkesztett kapcsolási rajzát tartalmazza:



27. ábra A bináris események kiváltására, küldésére alkalmas periféria áramkörének kapcsolási rajza

A periféria központi eleme az ATTiny2313-as típusú AVR mikrovezérlő. Ezen mikrovezérlő megfelelő portjaira (illetve perifériáira) kapcsolódnak a további áramkörök, alkatrészek.

A 9. portjára (PD5) egy nyomógomb kapcsolódik egy felhúzó ellenálláson keresztül, a nyomógomb célja, hogy a felhasználó ezen gomb megnyomásával generálhatja az adott bináris eseményt.

A 11. ortjára (PD6) egy ellenálláson keresztül kapcsolódik egy LED, amely funkciója hogy a felhasználó számára jelezze, ha az adott esemény létrejött, és az áramkör a vezetéknélküli modul segítségével továbbította azt, tehát egy rövid időre felvillan ha az esemény létrejött.

A 3. portjára (TX, PD1) a 433 MHz-es rádiós adó modul kapcsolódik.

A 4. és 5. portokon található kvarcoszcillátor és kondenzátor hármas a mikorvezérlő órajelét hívatott előállítani, amely megválasztása az alábbiak alapján történt: ezen mikrovezérlőt alapbeállításban a belső órajele működteti, amely 1 Mhz. A nagyobb pontosság érdekében külső órajel generátort szokás alkalmazni, amely a kapcsolási rajzon látható kvarc és kondenzátorok segítségével áll elő. Az órajelet elsősorban az USART periféria adattovábbítási sebességének megválasztása alapján állítottam be. Adatátviteli sebességnek a 9600 bps-t választottam, amely a bináris események, vagyis az azok alapján létrejött adatkeretek továbbítására elegendő. A mikrovezérlő adatlapja alapján az USART periféria 9600 bps sebességéhez 4 MHz-es órajel esetén 0.2%-os hiba tartozik, így a külső órajelet 4 Mhz-re állítottam, melyet a 4 Mhz-es kvarc és a 2 darab 22 pF-os kondenzátor állítja elő.

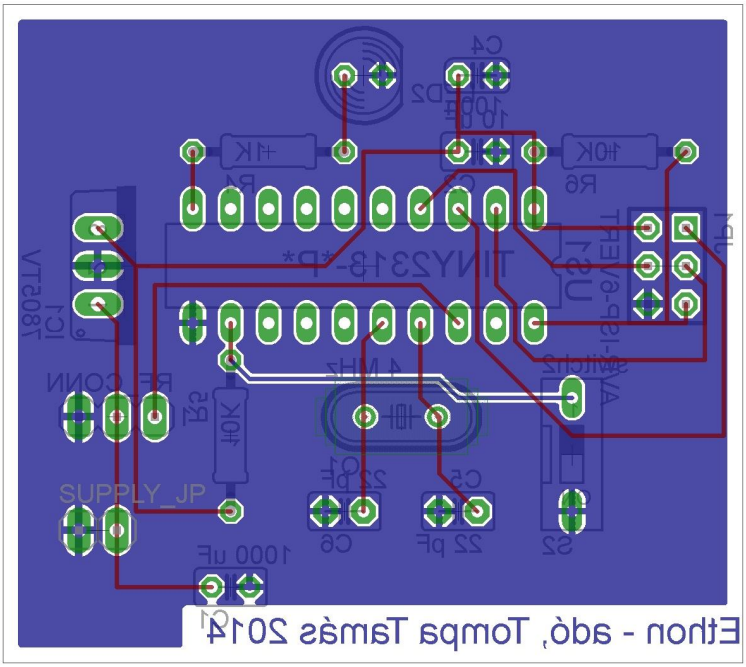
A mikrovezérlő órajelének, működési paramétereinek beállításához a biztosítékbitek megfelelő konfigurálására is szükség van.

Mivel a 433 Mhz-es rádiós adó modul hatótávolsága 12V-os tápfeszültséggel történő megtáplálás esetén a legnagyobb, így a periféria a működtetéséhez külső, 12V-os egyenfeszültségű megtáplálást igényel. A mikrovezérlő működtetéséhez 5V-os tápfeszültség szükséges, így egy 7805-ös típusú DC-DC konverter került alkalmazásra a kapcsolásban, amely a 12V-os tápfeszültségből a mikrovezérlő megfelelő működtetéséhez szükséges 5V-os tápfeszültséget állítja elő.

A kapcsolásban (majd a nyomtatott áramköri lapon) helyet kapott egy ISP 6-os (In-System Programmer) aljzat is, amely mikrovezérlő szoftverének módosítása esetén annak könnyebb újraprogramozását teszi lehetővé.

3.9.4 A robotperiféria nyomtatott áramköri rajza

A következő ábrán a kapcsolási rajz alapján tervezett kétoldalas nyomtatott áramköri rajz látható, az egyes alkatrészek beültetési helyét is szemléltetve:



28. ábra A bináris esemény generálására és továbbítására alkalmas robotperiféria nyomtatott áramköri rajza

3.9.5 A robotperiféria mikorvezérlőjének szoftvere

A mikrovezérlő szoftverét az ingyenesen letölthető AtmelStudio nevezetű fejlesztőkörnyezet segítségével implementáltam C programozási nyelven. A szoftver célja, hogy a mikrokontrollert illetve annak perifériáit működtesse.

A szoftver működése az alábbi:
Figyeli a mikrovezérlő PD5-ös portján lévő nyomógomb állapotát, ha logikai 1-es állapotban van, azaz megnyomták, akkor egy adatkeretet ír az USART perifériára (pontosabban az USART periféria TX portjára csatlakoztatott 433 Mhz-es adó modulra) és közben egy rövid időre bekapcsolja a PD2-es porton lévő LED-et. A LED funkciója, hogy jelezze, ha a nyomógomb logikai 1-es állapotú, azaz éppen megnyomták, illetve, hogy jelezze az adatkeret küldésének sikerességét.

A szoftver az alábbi állományokat tartalmazza:

- led.c: a PD6-os poton lévő LED működtetését valósítja meg
- led.h: a led.c állomány függvény definícióit tartalmazza
- usart.c: az USART perifériát működteti, adatok küldését, és fogadását valósítja meg
- usart.h: az usart.c állomány függvény definícióit tartalmazza
- main.c: főprogram, main-t tartalmazza, a fentebb felsorolt állományok függvényeinek felhasználásával megvósítja az áramkör működtetését

A következő táblázatok az egyes állományok függvényeit és azok funkcióit tartalmazzák:

6. Táblázat A led.c állomány függvényei és azok funkciói

Függvény	Funkció
void LEDon()	Bekapcsolja a PD6-os porton lévő LED-et (logikai 1-es állapotba kerül a PD6-os port)
void LEDoff()	Kikapcsolja PD6-os porton lévő LED-et (logikai 0 állapotba kerül a PD6-os port)

7. Táblázat Az usart.c állomány függvényei és azok funkciói

Függvény	Funkció
void USARTInit(unsigned int ubrr_value)	A paraméterében megadott UBRR értékkel inicializálja az USART perifériát
char USARTReadChar()	Az RX pin-en fogadott adatot adja vissza char típusként
void USARTWriteChar(char data)	A paraméterében megadott char típusú adatot írja a TX pin-re
void USARTWriteCharArray(char* array)	A paraméterében megadott char típusú tömböt írja a TX pin-re
void USART_TransmitChar(unsigned char data)	A paraméterében megadott előjel nélküli char típusú adatot írja a TX pin-re
void USART_TransmitInt(unsigned int data)	A paraméterében megadott előjel nélküli int típusú adatot írja a TX pin-re
unsigned char USART_Receive(void)	Előjel nélküli char típusként adja vissza az RX pin-en fogadott adatot
unsigned int USART_ReceiveInt(void)	Előjel nélküli int típusként adja vissza az RX pin-en fogadott adatot
void USART_Flush(void)	Üríti az USART periféria bufferét

8. Táblázat Az main.c állomány függvényei és azok funkciói

Függvény	Funkció
int main(void)	Főprogram, a fentebb felsorolt állományok függvényeit felhasználva megvalósítja a robotperiféria működtetését
void test()	Az áramkör tesztelése során használt függvény, amely adott időközönként (1 sec-ként) adatkeretet küld a 433 Mhz-es rádiós adó modul segítségével

3.9.6 A robotperiféria által megvalósított bináris esemény

Ezen robotperiféra segítségével a „csengetés” esemény megvalósítása lehetséges. Az áramkör egy nyomógombot tartalmaz, amely a felhasználó általi megnyomásával egy adatkeretet, jelzést továbbít a robotban helyetfoglaló, annak számítógépéhez csatlakoztatott vevő periféria számára. A „csengetés” esemény jelzése tehát az alábbi adatkeret küldését jelenti:

9. Táblázat A „csengetés” esemény adatkeretének felépítése

0x01	0x55	0x10

A 0x01 a kerefej, a 0x10 a keretvég, a 0x55 pedig az adat, amley a magát a „csengetés” eseményt azonosítja. A robot a „csengetés” esemény hatására megvalósuló feladatot a robotvezérlő keretrendszerbeli szfotverkomponens által valósítja majd meg, vagyis a robot ajtóhoz történő pozícionálását a későbbi fejezetkeben kifejtett szoftverkomponensek valósítják majd meg.

A kommunikáció során adatkeret alkalmazása a kommunikáció sikersesége érdekében fontos lépés. Egyrészt az éterből bárhonnan érkezhetnek olyan elektromágneses zajok, jelek, melyek aktiválhatják a periféria működését, azaz pont olyan adatot jelentő zaj érkezhet, amely a periféria működését aktiválhatja. Adatkeret alkalmazása esetén, amley több, egymásután elhelyezkedő adatból áll, kisebb a valószínűsége, hogy pont olyan, több egymásutáni adat (zaj) fog érkezni a külvilág felől. Másrészt pedig a vezetéknélküli kommunikáció megvalósítására 433 Mhz-es rádiós adó és vevő modult alkalmazok, amelyek által használt frekvenciát más eszközök, berndezések is használják. Ezen frekvencián üzemelnek például a vezetéknélküli kapucsengők és a kapunyitót működtető

távirányítók is. Ezen eszközök jelét a robotban helyetfoglaló vevő periféria fogadja, de az általuk sugárzott adatot már feldolgzn nem fogja, mert az az adat terljesen másként épül fel, mint az általam alkalmazott adatkeret. Tehát ezen eszközök jelét fogadja a vevő periféria, de feldolgzoni már nem fogja, mert nem éppen az a keret érkezik, amelyet az adó robotperiférától vár. Ezen mechanizmus segítségével oroslom, hogy más ugyenezen a frekvencián üzemelő eszközök ne zavarják az adó- és vevőperiféria működését.

3.9.7 A robotperiféria tesztelése

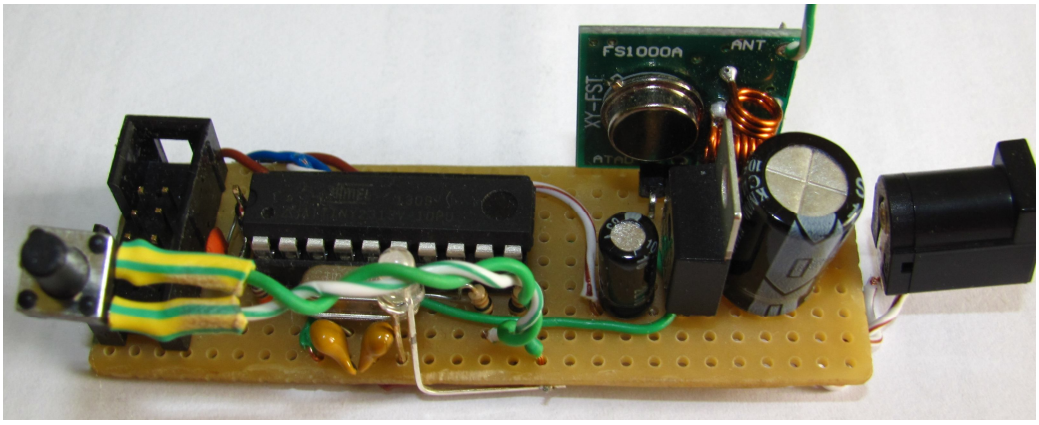
Ezen robotperiféria tesztelése az általa betöltendő funkció helyes működésének az ellenőrzését jelenti, azaz hogy a nyomógomb megnyomásának hatására továbbítódik-e az „csengetés” esemény létrejöttét jelező adatkeret, illetve, hogy helyes adatkeret kerül-e továbbításra. A működés helyességének ellenőrzése a RealTerm nevű terminálprogramot használtam. Ezen terminálprogram lehetővé teszi a számítógép sorosportjára érkező adatok, adatkeretek karakteres formában történő megjelenítést. A robotperifériát egy USB-soroport konverter segítségével csatlakoztattam a számítógép USB portjára, majd az USB-sorosport konverter által létrehozott virtuális sorosportra érkező adatokat a terminálprogram segítségével ellenőriztem. A megfigyelés azt mutatta, hogy a nyomógomb megnyomásra helyes adatkeretet továbbít az áramkör, de nem minden egyes nyomógomb megnyomás alkalmával. Tehát ha továbbított adatkeretet akkor a helyes adatkeretet továbbította, de az adat küldése nem minden egyes gombnyomás esetén valósult meg. Ennek oka vélhetően a 433 Mhz-en üzemel rádiós modul távolság és környezetfüggősége, tehát, hogy milyen távolságra kell éppen az adatot továbbítani illetve, hogy azt milyen környezetben, vannak-e betonfalak, vasrácsok, vagy egyéb nagy fémes felülettel rendelkező akadályok. Ezen problámára megoldást jelentett az, hogy nyomógomb egyszeri megnyomásának hatására az adatkeret többször (50-szer) kerül továbbításra. A mikrokontroller szoftverét ennek megfelelően módosítottam, azaz a keretküldés egy for ciklusban valósul meg, amely ciklus 50-szer fut le. A kódrészletet az alábbi szövegdoboz tartalmazza:

```
for(i=0;i<=50;i++) {
        USART_Flush();
        LEDon();
        USARTWriteCharArray(frame);
}
```

29. ábra Az adatkeret egymásutáni többszöri küldését megvalósító for ciklus a kommunikáció sikerességének növelése érdekében

3.9.8 A prototípus fényképe

Az alábbi ábrán a próbanyákon elkészített prototípus fényképe látható:



30. ábra A binárs "csengő" esemény küldésére alkalmas robotperiféria próbanyákon megépített prototípusa

3.10 A bináris esemény fogadására alkalmas robotperiféria tervezése, megvalósítása

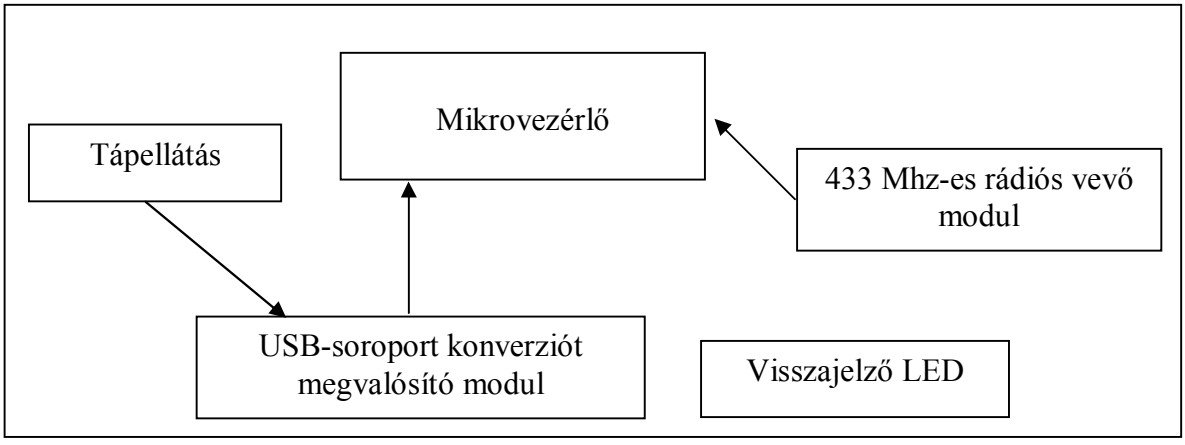
Ezen alfejezet az előző fejezetben ismertetett megvalósítási lehetőségek illetve azok felhasználásával ismerteti a bináris események fogadására alkalmas robotperiféria célját, tervezését, megvalósítását, működését.

3.10.1 A robotperiféria célja

A bináris események vételére, fogadására alkalmas robotperiféria célja, hogy az adó periféria által a vezetéknélküli kommunikációs kapcsolaton keresztül továbbított bináris eseményt, eseményeket fogadja, majd ezen esemény alapján egy adatkeretet továbbítson a robotban található számítógép USB portjára. Ez a periféria is egy mikrovezérlőt, pontosabban egy ATTiny2313 típusú AVR-t tartalmazó áramkör, amely a fogadott bináris eseményt továbbítja a robot számítógépe felé annak USB portján, vagyis pontosabban sorosportján keresztül. A megvalósítás során a mikrovezérlő adott portjaihoz csatlakoznak a funkció megvalósítását lehetővé tevő áramkörök, modulok. A mikrokontroller megfelelő portjára csatlakozik egy 433 Mhz-es rádiós vevő modul, továbbá egy LED, amely az esemény, pontosabban adatkeret vételének helyességét jelzi, illetve egy USB-sorosport konverziót, szintillesztést megvalósító áramköri modul.

3.10.2 A robotperiféria áramkörének blokkvázlata

Az alábbi 29. ábrán a periféria célja, funkciója alapján felépített áramkör blokkvázlata látható:

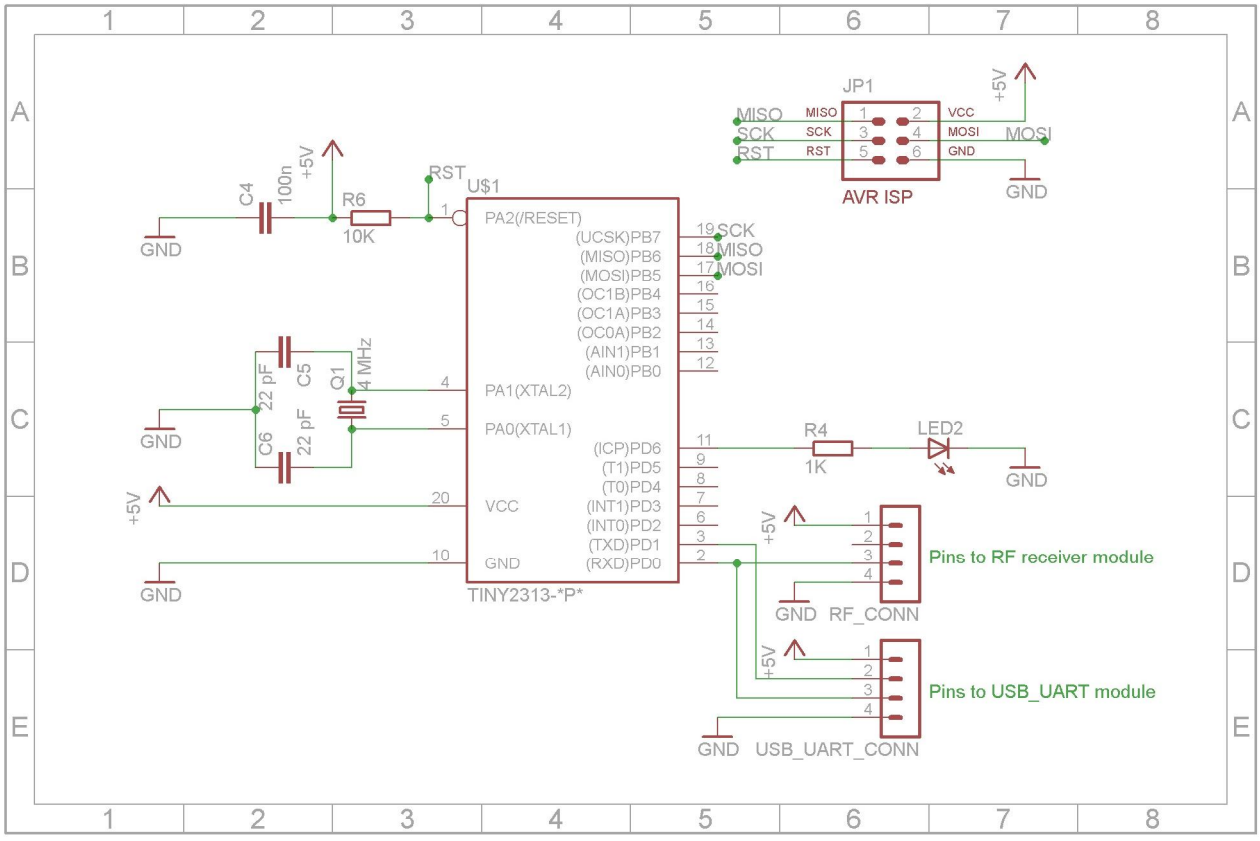


31. ábra A bináris események fogadására alkalmas periféria áramkörének blokkvázlata

A 29. ábrán látható módon az áramkörben helyet foglal egy mikorkontroller, amely a periféria központi eleme, ezen mikorvezérlő működteti magát a perifériát, valósítja meg a portjaira csatlakoztatott, LED és a 433 Mhz-en üzemelő rádiós adó modul működtetését, illetve az adó periféria által elküldött adatkeretek fogadását, azok helyességének ellenőrzését.

3.10.3 A robotperiféria kapcsolási rajza, működése

Az alábbi 30. ábra a 29. ábrán látható blokkvázlat alapján tervezett bináris esemény előállítására, továbbítására alkalmas áramkör, az Eagle tervező szoftver segítségével szerkesztett kapcsolási rajzát tartalmazza:



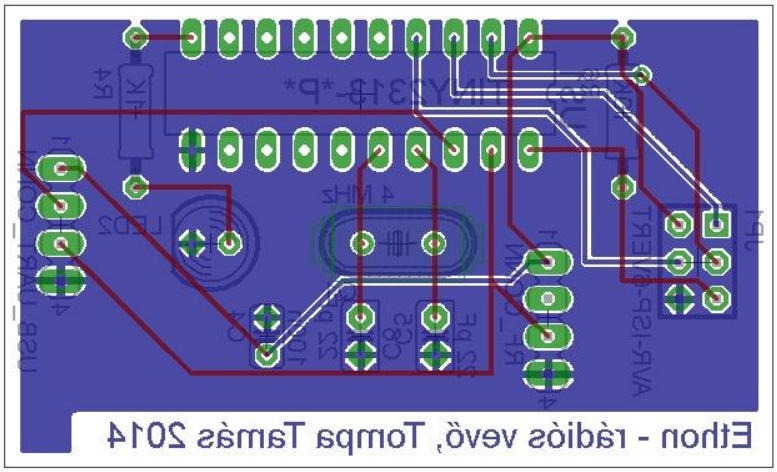
32. ábra A bináris események fogadására alkalmas periféria áramkörének kapcsolási rajza

A periféria központi eleme – az adó perifériához hasonlóan - az ATTiny2313-as típusú AVR mikrokontroller. Ezen mikrovezérlő megfelelő portjaira, perifériáira kapcsolódnak a további áramkörök, áramköri modulok.

A mikorvezérlő PD6-os portjára (11. láb) ellenálláson keresztül kapcsolódik egy LED, amely funkciója annak jelzése a LED rövid felvillanásának segítségével, hogy az adó periféria által elküldött adatkeretet az áramkör a 433 Mhz-es rádiós vevő modul segítségével fogadta és sikersenen feldolgozta. A PD0-ás portra az az USART periféria RX tüskéjére a 433 MHz-es rádiós vevő modul kapcsolódik. Az USART periféria adattovábbítási és adatfagadási sebessége az adó áramkörhöz hasonlóan 9600 bps.
A PA0 és PA1 (4. és 5. pin) portokon található kvarcoszcillátor és kondenzátor hármas szerepe az mikorvezérlő órajelének előállítása, az órajel beállítása (4 MHz) megegyezik az adó perifériánál is alkalmazott órajellel.
A mikrovezérlő illetve a rádiós vevő modul is 5V-os tápfeszültséget igényel működéséhez, melyet az USB-sorosport konverter modul állít elő, azaz a szükséges tápfeszültségeket az áramkör a robot számítógépén található USB portról kapja, így nincs szükség egyéb külső tápfeszültségforrás alkalmazására.
A kapcsolásban (pontosabban majd a nyomtatott áramköri lapon) helyet kapott egy ISP (In-System Programmer) 6-os aljzat is, amely a mikrovezérlő szoftverének módosítása esetén annak könnyebb újraprogramozást teszi lehetővé.

3.10.4 A robotperiféria nyomtatott áramköri rajza

A következő ábrán a kapcsolási rajz alapján tervezett kétoldalas nyomtatott áramköri rajz látható, az egyes alkatrészek beültetési helyét is szemléltetve:



33. ábra A bináris esemény fogadására alkalmas robotperiféria nyomtatott áramköri rajza

3.10.5 A robotperiféria mikorvezérlőjének szoftvere

A mikrovezérlő szoftverét szintén az ingyenesen letölthető AtmelStudio nevezetű fejlesztőkörnyezet segítségével implementáltam C programozási nyelven.

A szoftver működése az alábbi:
Folyamatosan figyeli az USART perifériát (pontosabban annak RX portját), ellenőrzi, hogy érkezett-e az adó periféria által küldött adatkeret, az adatekretet a 433 Mhz-es rádiós vevő modul segítségével fogadja. Ha helyes, vagyis ténylegesen az adó periféria által küldött adatkeret érkezett és azt sikeresen fogadta, feldolgozat akkor egy rövid időre bekapcsolja a PD6-os porton lévő LED-et, a keretfeldolgozás sikerességének jelzésére. Helyes adatkeret fogadása esetén, a fogadott adatkeretet változatlanul továbbítja az USART periféria TX portjára, vagyis pontosabban USB-sorosport konverziót megvalósító modul számára. Az USB-sorosport konverziót megvalósító modul segítségével az adatkeret a vevő perifériát szimblizáló, a koncverziót megvalósító modul által létrehozott sorosportra továbbítódik. Teháta vevő periféria egy sosrportként jelentekzik a robot számítógépén.

A szoftvert az alábbi állományok alkotják:

- led.c: a PD6-os poton lévő LED működtetését valósítja meg
- led.h: a led.c fájl függvény definícióit tartalmazza
- usart.c: az UART perifériát működteti, az adó periféria által elküldött adatok fogadását, illetve a fogadott adatok tömbben való tárolását valósítja meg
- usart.h: az usart.c állomány függvénydefinícióit tartalmazza
- main.c: főprogram

A következő táblázatok az egyes állományok függvényeit és azok funkcióit tartalmazzák:

10. Táblázat A led.céllomány függvényei ésazok funkciói

Függvény	Funkció
void LEDon()	Bekapcsolja a PD6-os porton lévő LED-et (logikai 1-es állapotba kerül a PD6-os port)
void LEDoff()	Kikapcsolja PD6-os porton lévő LED-et (logikai 0 állapotba kerül a PD6-os port)

11. Táblázat Az usart.c állomány függvényei és azok funkciói

Függvény	Funkció
void USARTInit(unsigned int ubrr_value)	A paraméterében megadott UBRR értékkel inicializálja az USART perifériát
char USARTReadChar()	Az RX pin-en fogadott adatot adja vissza char típusként
void USARTWriteChar(char data)	A paraméterében megadott char típusú adatot írja a TX pin-re
void USARTWriteCharArray(char* array)	A paraméterében megadott char típusú tömböt írja a TX pin-re
char* USARTReadCharArray(int arraySize)	A paraméterében megadott számú bájtot olvas be és adja vissza egy char típusú tömbben
void USART_TransmitChar(unsigned char data)	A paraméterében megadott előjel nélküli char típusú adatot írja a TX pin-re
void USART_TransmitInt(unsigned int data)	A paraméterében megadott előjel nélküli int típusú adatot írja a TX pin-re
unsigned char USART_Receive(void)	Előjel nélküli char típusként adja vissza az RX pin-en fogadott adatot
unsigned int USART_ReceiveInt(void)	Előjel nélküli int típusként adja vissza az RX pin-en fogadott adatot
void USART_Flush(void)	Üríti az USART periféria bufferét

12. Táblázat A main.c állomány függvényei és azok funkciói

Függvény	Funkció
int main(void)	Főprogram, a fentebb felsorolt állományok függvényeit felhasználva megvalósítja a robotperiféria működtetését. Inicializálja a mikorvezérlő perifériáit, olvas az USART perifériáról, ellenőrzi, hogy azadó által küldött adatkeret érkezett-e és ha igen, akkor a keretet változatlanul továbbítja az USART periféria TX pin-jére.

3.10.6 A robotperiféria által megvalósított bináris esemény

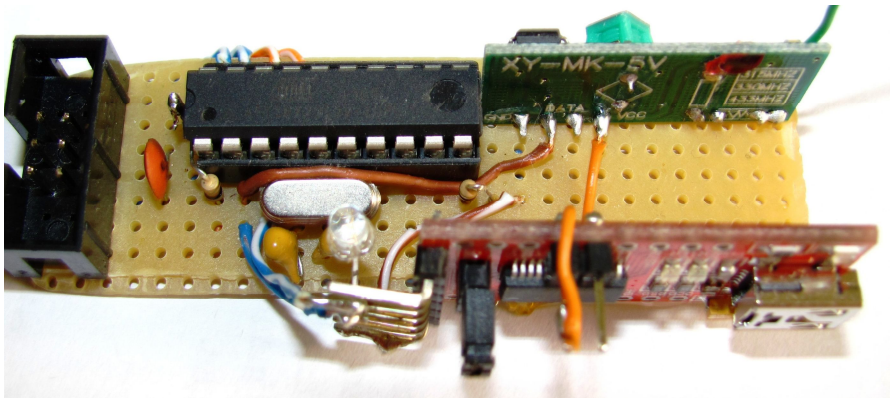
Ezen robotperiféira nem valósít meg bináris eseményt, hanem az adó robotperiféria által továbbított események fogadását valósítja meg. A fogadott, bináris eseményeket szimbolizáló adatkeretek helyességének ellenőrzését végzi, majd helyes adatcsomag érkezése esetén továbbítja az a számítógép USB portjára, vagyis pontosabban az USB porton jelentkező sorosportra (virtuális sorosport).

3.10.7 A robotperiféria tesztelése

Ezen robotperiféria tesztelése is a helyes működés ellenőrzését jelenti, ezen periféria esetében annak ellenőrzése szükséges, hogy megbízható módon fogadja-e az adatkeretek, és sikersen adatcsomag feldolgozás esetén valóban a fogadott adatkeretet továbbítja-e. Ezen működés ellenőrzését szintén a RealTerm nevezetű terminálprogram segítségével valósítottam meg. A teszteléshez az előző lépésben validált adó robotperifériát használtam fel, ezen periféria segítségével továbbítottam bináris eseményeket, vagyis pontosabban az eseményeket szimbolizáló adatcsomagokat. A megfigyeléseim azt mutatták, hogy helyes adatkeret érkezéseesetén a vevő periféria ténylegesen feldolgozta az adatot, majd annak hatására a megfelelő keretet továbbította a robot számítógépének USB portjára, pontosabban az azt jelképező sororportra. Hibás adatkeret küldése esetén a vevő periféria fogadta ugyan az adatkeretet, de mivel annak felépítése nem egyezett meg az adó periféria által küldött, az adott bináris eseményt szimbolizáló adatcsomaggal, így azt a vevő robotperiféria nem dolgozta fel.

3.10.8 A prototípus fényképe

Az alábbi ábrán a próbanyákon elkészített prototípus fényképe látható:



34. ábra A binárs események fogadására alkalmas robotperiféria próbanyákon megépített prototípusa

3.11 A robot távvezérlését lehetővé tevő robotperiféria tervezése

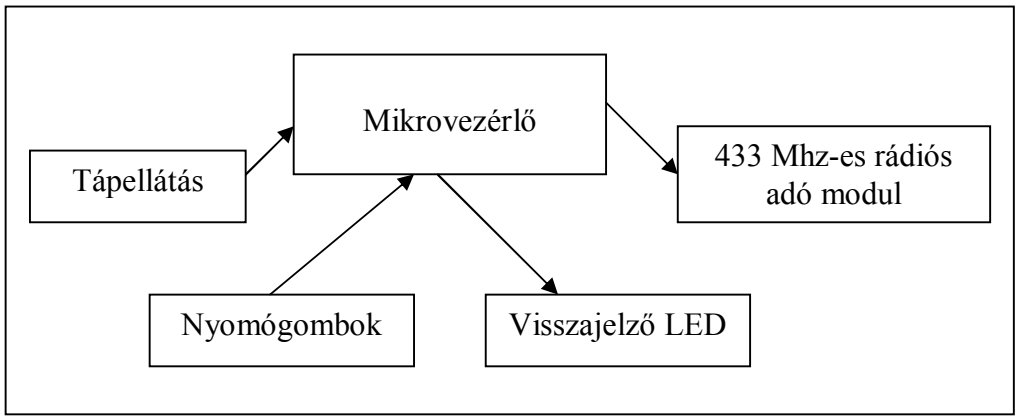
Ezen alfejezet a robot távvezérlését lehetővé tevő binárs események küldését megvalósító robotperiféria tervezését, célját, funkcióját, működését ismerteti.

3.11.1 A robotperiféria célja

Ezen robotperiféria célja, hogy a robot (Ethon) távvezérlését megvalósítsa, azaz egyfajta távirányítóként funkcionálva vezérelje a robot működését. Ethon távvezérlése alatt a robot pozícióváltoztatása, azaz adott irányba történő elmozdulása értendő. Ezen robotperiféria célja, hogy a távvezérlési funkciót megvalósító bináris eseményeket felhasználó által történő módon létrehozza, illetve továbbítsa a robot számítógépéhez USB porton keresztül csatlakoztatott vevőperiféria számára.
Ezen áramkör megvalósítása is egy mikorkontroller, pontosabban az előző fejezetekben ismertetett adó- és vevőperiféria esetében is alkalmazott ATTiny2313-as típusú AVR segítségével fog megvalósulni. A mikorvezérlő megfelelő perifériához csatlakoznak majd a távirányítás bináris eseményét kiváltó nyomógombok, az adatküldés sikerességét jelző LED, illetve a vezetéknélküli kommunikációt megvalósító 433 Mhz-es rádiós adó modul. Tehát ezen robotperiféria is egyfajta rádiós adóként működik, felépítése annyiban tér el a 3.9-es fejezetben ismertett adó perifériáétől, hogy több nyomógombbal rendelkezik. Mivela 3.9-es fejezetben ismertetett periféria célja a „csengetés” funkció megvalósítása, amley által az áramkör fixen van elhelyelzve a funkcióját betöltő területen, így a távvezérlést megvalósító bináris események kiváltására egy újabb periféria tervezése szükségeltetik.

3.11.2 A robotperiféria áramkörének blokkvázlata

Az alábbi 32. ábrán a periféria célja, funkciója alapján felépített áramkör blokkvázlata látható:

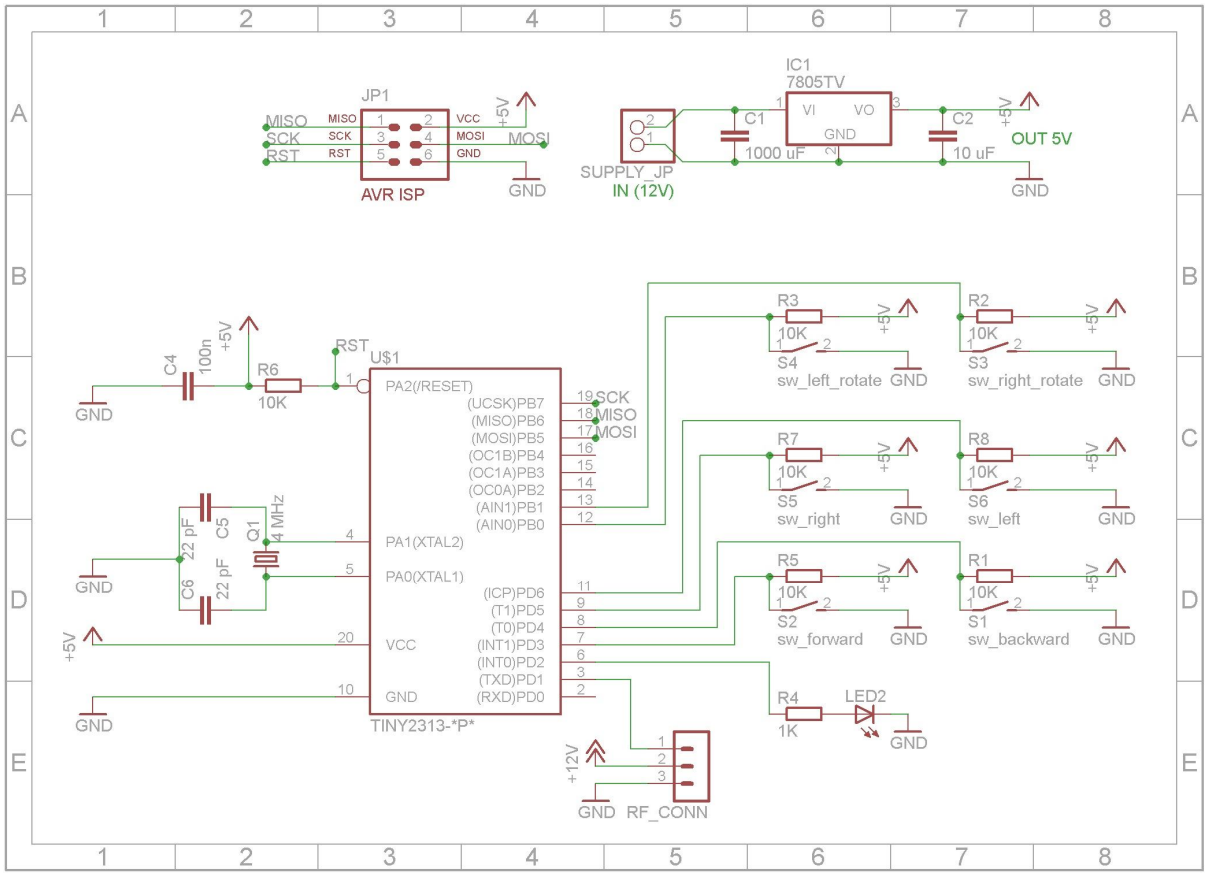


35. ábra A távvezérlést megvalósító bináris események küldésére alkalmas periféria áramkörének blokkvázlata

A 32. ábrán látható módon az áramkörben helyet foglal egy mikorkontroller, amely a periféria központi eleme, ezen mikorvezérlő működteti magát a perifériát, valósítja meg a portjaira csatlakoztatott, LED és a 433 Mhz-en üzemelő rádiós adó modul működtetését, illetve a portjaira csatlakoztatott nyomógombok állapotának olvasását. Mivel Ethon holonómikus hajtással rendelkezik, így összesen 6 irány, azaz az előre, hátra, jobbra, balra, balra forgás és a jobbra forgás áll rendelkezésre. Ennek megfelelően, az adott irányba történő mozgatáshoz különböző nyomógombok alkalmazására van szükség, amely összesen 6 darab nyomógombot jelent.

Ezen robotperiféria is a robot számítógépében helyet foglaló, ahhoz USB porton keresztül csatlakoztatott vevő perifériához csaltakozik, mint a „csengetés” funkciót megvalósító adó áramkör is. Mivel a vezetéknélküli kommunikáció megvalósítására 433 Mhz-es rádiós adó- és vevő modulokat alklamazok, így a „több adó – egy vevő” koncepció nehézségek nélkül megvalósítható. Ezen esetben az a követlemény, hogy az adó modulok ugyanazon a frekvencián sugározzanak adatot, melyen a vevő modul is üzemel. Az általam alkalmazott rádiós vevő modul a 433 Mhz-es frekvencián üzemel, így ha ahhoz töb adó modult szeretnék csatlakoztatni, akkor az adó moduloknak is ezen a 433Mhz-es frekvencián kell üzemelniük. Jelen esetben mind a két adó funkciót megvalósító robotperifériában a 433 MHz-es rádiós adó modul foglal helyet, így azok automatikusan a 433 Mhz-es rádiós vevő modulhoz fognak csatlakozni (pontosabban azon a frekvencián fognak sugározni adatot, melyen a vevő modul is üzemel).

3.11.3 A robotperiféria kapcsolási rajza, működése



36. ábra A távvezérlést megvalósító bináris események küldésére alkalmas periféria áramkörének kapcsolási rajza

A 33. ábra a 32. ábrán látható blokkvázlat alapján tervezett, a távvezérlést megvalósító bináris esemény előállítására, továbbítására alkalmas áramkör, Eagle tervező szoftver segítségével szerkesztett kapcsolási rajza látható.

A mikorvezérlő PD2-es portjára (6. láb) ellenálláson keresztül kapcsolódik egy LED, amely funkciója, hogy a bináris esemény létrejöttét, illetve annak sikeres továbbítását egy rövid felvillanással jelezze.

A PD1-es portra az az USART periféria TX tüskéjére a 433 MHz-es rádiós adó modul kapcsolódik. Az USART periféria adattovábbítási és adatfagadási sebessége az eddig megvalósított áramkörhöz hasonlóan 9600 bps.

A PA0 és PA1 (4. és 5. pin) portokon található kvarcoszcillátor és kondenzátor hármas szerepe az mikorvezérlő órajelének előállítása, az órajel beállítása (4 MHz) megegyezik a „csengetés” funkciót megvalósító adó perifériánál is alkalmazott órajellel.
A kapcsolásban szintén helyet kapott egy ISP (In-System Programmer) 6-os aljzat is, amely a mikrovezérlő szoftverének módosítása esetén annak könnyebb újraprogramozást teszi lehetővé.
A PD3, PD4, PD5, PD6 valamint a PB0, PB1-es portokra az adott irányba történő pozícióváltoztatást lehetővé tevő nyomógmbok kapcsolódnak. Ezen nyomógombok az előre, hátra, jobbra, balra vaalmint a jobbra forgás, balra forgás funkció működtetését valósítják meg.
Az áramkör működtetéséhez 12V-os egyenfeszültség szükséges. A 12V közvetlenül a rádiós adó modul tápellátást biztosítja, amely esetében az általa áthídalható távolság körülbelűl 200 méter. A mikorvezérlő illetve a további áramköreinek működtetéséhez szükséges 5V-os tápfeszültséget pedig a 7805-ös feszültségstabilizátor IC állítja elő.

3.11.4 A robotperiféria mikorvezérlőjének szoftvere

A mikrovezérlő szoftverét szintén az ingyenesen letölthető AtmelStudio nevezetű fejlesztőkörnyezet segítségével implementáltam C programozási nyelven.

A szoftver működése az alábbi:
A 6 darab nyomógomb állapotától függően az adott bináris eseményt szimbolizáló adatkeret küldését valósítja meg. Attól függően, hogy mely nyomógomb van éppen logikai „1”-es azaz megnyomott állapotban, annak a nyomógomb funkciójának megfelelő adatcsomagot (bináris eseményt) továbbítja. Az adatcsomagot az USART periféria TX pinjére továbbítja, majd küldi a 433 Mhz-en üzemelő rádiós adó modul számára. Az esemény létrejöttét, illetve a sikeres adattovábbítást a PD2-es porton lévő LED rövid idejű felvillanásával jelzi.

A szoftver állományai, azok funkciói:

- led.c: a PD2-os poton lévő LED működtetését valósítja meg
- led.h: a led.c fájl függvény definícióit tartalmazza
- usart.c: az UART perifériát működteti, inicializálja a megfelelő paraméterekkel, függvényeket biztosít adatkeretek küldéséhez

- usart.h: az usart.c állomány függvénydefinícióit tartalmazza
- main.c: főprogram

A következő táblázatok az egyes állományok függvényeit és azok funkcióit tartalmazzák:

13. Táblázat A led.c állomány függvényei és azok funkciói

Függvény	Funkció
void LEDon()	Bekapcsolja a PD2-es porton lévő LED-et (logikai 1-es állapotba kerül a PD2-es port)
void LEDoff()	Kikapcsolja PD2-es porton lévő LED-et (logikai 0 állapotba kerül a PD2-es port)

14. Táblázat Az usart.c állomány függvényei és azok funkciói

Függvény	Funkció
void USARTInit(unsigned int ubrr_value)	A paraméterében megadott UBRR értékkel inicializálja az USART perifériát
char USARTReadChar()	Az RX pin-en fogadott adatot adja vissza char típusként
void USARTWriteChar(char data)	A paraméterében megadott char típusú adatot írja a TX pin-re
void USARTWriteCharArray(char* array)	A paraméterében megadott char típusú tömböt írja a TX pin-re
void USART_TransmitChar(unsigned char data)	A paraméterében megadott előjel nélküli char típusú adatot írja a TX pin-re
void USART_TransmitInt(unsigned int data)	A paraméterében megadott előjel nélküli int típusú adatot írja a TX pin-re
unsigned char USART_Receive(void)	Előjel nélküli char típusként adja vissza az RX pin-en fogadott adatot
unsigned int USART_ReceiveInt(void)	Előjel nélküli int típusként adja vissza az RX pin-en fogadott adatot
void USART_Flush(void)	Üríti az USART periféria bufferét

15. Táblázat A main.c állomány függvényei, azok funkciói

Függvény	Funkció
int main(void)	Főprogram, a fentebb felsorolt állományok függvényeit felhasználva megvalósítja a robotperiféria működtetését, azaz az adott távvezérlést megvalósító bináris események megfelelő adatcsomagot továbbítja

3.11.5 A robotperiféria által megvalósított bináris esemény

Ezen robotperiféira Ethon távvezérlését lehetővé tevő binárs eseményeket, vagyis pontosabban az azoknak megfelelő adatcsomag küldését valósítja meg. A bináris esemény szintén egy kétállapotú információt jelent, azaz attól függően, hogy a felhasználó megnyomta-e a robot adott irányba történő vezérlését lehetővé tevő nyomógombot, vagy küldésre kerül az eseményt szimbolizáló adatkeret (csak a nyomógomb megnyomása esetén) vagy sem (ha a nyomógomb nincs megnyomott állapotban).

Az eseményeknek megfelelő adatkeretek felépítését a következőtáblázat tartalmazza:

16. Táblázat A távvezérlést megvalósító binárs eseményeknek megfelelő adatkeretek felépítése

Funkció	Keretfej	Adat	Keretvég
Előre	0x01	0x12	0x10
Hátra	0x01	0x13	0x10
Balra	0x01	0x14	0x10
Jobbra	0x01	0x15	0x10
Balra forgás	0x01	0x16	0x10
Jobbra forgás	0x01	0x17	0x10

3.11.6 A robotperiféria tesztelése

Ezen robotperiféria tesztelése szintén az általa betöltendő funkció helyes működésének az ellenőrzését jelenti, azaz hogy a nyomógombok megnyomásának hatására továbbítódnak-e a távvezérlés esemény létrejöttét jelező adatkeretek, illetve, hogy a helyes adatkeretek kerül-e továbbításra. A működés helyességének ellenőrzése szintén a RealTerm nevű terminálprogramot használtam, amely lehetővé tette, a küldött adatkeretek felépítésének megjelenítését. A megfelelő nyomógomb megnyomásának hatására a megfelelő adatkeretet továbbította az áramkör, viszont ugyanaz a jelenség jelenkezett, amit a „csengetés” eseményt megvalósító adóperiféria esetében, hogy nem minden gombnyomás hatására továbbította az adatkeretet. Ezen problámának a megoldását ugyanazon módon valósítottam meg, mint a fentebb említett „csengetés” esemény küldését megvalósító robotperiféria esetében, tehát a nyomógomb egyszeri megnyomásának hatására az adatkeret többszöri továbbítására kerül sor.

4. A robotperiféríák illesztése robotvezérlő keretrendszerhez

Ezen fejezet a bináris események továbbítására és fogadására alkalmas robotperiféríákat működtető, illetve a bináris események alapaján történő robotvezérlést megvalósító számítógépes, robotvezérlő keretrendszerbeli szoftvermodul célját, felépítését, implementálást ismerteti.

4.1 A szoftverkomponensek célja

A robotvezérlő keretrendszerbeli több komponensből álló szoftvermodul célja, hogy a robot számítógépéhez csatlakoztatott bináris események fogadására alkalmas robotperiféria által fogadott bináris esenények alapján történő robot ezérlést megvalósítsa. Az események fogadására képes robotperiféria a robot számítógépéhez annak USB portján keresztül csatlakozik, illetve ez által egy darab sorosportként van jelen a rendszerben.
A szoftverkomponensek célja, hogy megvalósítsa a robotban helyetfoglaló vevő periféria által létrehozott sorosport olvasását, az azon fogadott utasításkeretek feldolgozását, illetve a fogadott adatcsomagok alapján megvalósítsa a robot vezérlését, adott feladat végrehajtását. A fogadott bináris eseményeket az adó perifériák („csengő” periféria, távirányító periféria) hozzák létre. A szoftvermodul tehát a robot vezérlését, annak működtetését valósítja meg, a fejlesztett robotperiféríák által léterhozott események („csengetés”, távirányítás) alapján.

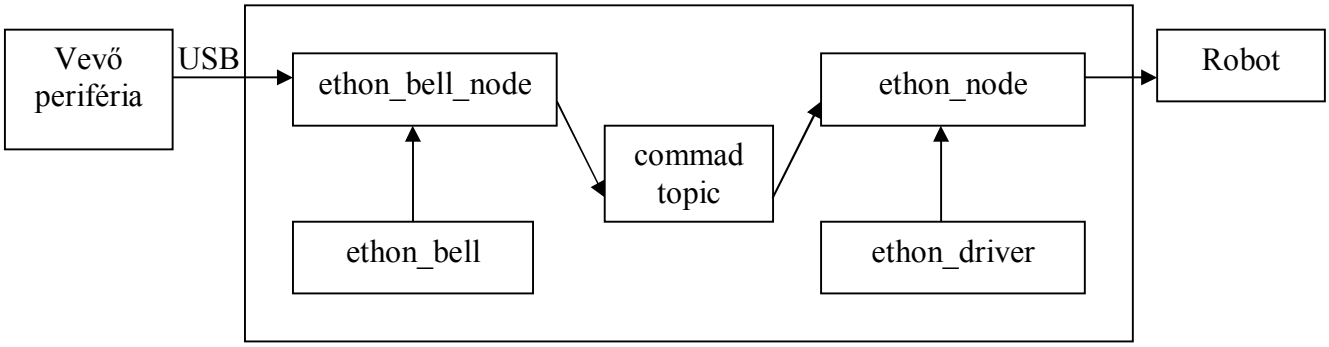
4.2 A választott keretrendszer

A szoftvermodult a ROS, vagyis a Robot Operating System keretrendszerben implementálom C++ programozási nyelven. A választás oka, hogy ezen keretrendszer számos, előre megírt és beépített funkciót, modult tartalmaz illetve jól dokumentált, dokumentációja számos mintapéldán keresztül szemlélteti a rendszer működését, felépítését.

A C++ programozási nyelven történő implementáció megvalósítására QTCreator fejlesztőkörnyezertet alkalmazok majd.

4.3 A rendszer felépítése, működése

A ROS rendszer filozófiája alapján az e rendszerben implementált adott feladatot megvalósító robotvezérlő szoftver több, adott feladatot megvalósító csomagból, csomópontból épül fel. Mindegyik csomagnak, csomópontnak jól meghatározott feladata, funkciója van, ezen csomópontok topik-ok és szervizek segítségével cserélnek egymás között információt, adatot. Ezen szemlélet alapján terveztem meg a bináris események fogadására alkalmas periféria működtetéséhez és a fogadott események alapján történő robotvezérléshez szükséges csomagok, csomópontok felépítését, funkcióját.
Az események alapján történő robotvezérlést megvalósító szoftvermodul több, a ROS rendszerben implementált csomagból, csomópontból épül fel, melyek felépítését az alábbi ábra szemlélteti:



37. ábra A fogadott bináris események alapján történő robotvezérlést megvalósító ROS szoftvermodul felépítése

A rendszer működése az alábbi: a bináris eseményeket fogadó vevő robotperiféria, amely a robot számítógépéhez annak USB portján (pontosabban sorosportján) keresztül kapcsolódik, az adó perifériák által továbbított adatkereteket fogadja, feldolgozza majd sikeres feldolgozás esetén továbbítja az USB portra. A vevő periféria a rendszerben sorosportként van jelen, amely olvasását az ethon_bell csomag valósítja meg. Az ethon_bell_node az ethon_bell csomag felhasználásával folyamatosan (másodpercenként 100 alkalommal) olvas a sorosportról és figyeli, hogy érkezett-e adatkeret és ha igen, akkor helyes adatkeret érkezett-e. Helyes adatcsomag fogadása esetén üzentetet publikál a command nevű topikba. Az ethon_node csomópont figyeli ezt a command nevű topikot, majd

ha az ethon_bell_node által publikált üzenet érkezik, akkor az ethon_driver csomag felhasználásával megvalósítja a robot vezérlését. Mind az ethon_bell_node, mind az ethon_node csomópont futtatása a roboton, pontosabban annak számítógépén szükséges.

4.4 Az egyes csomópontok, csomagok és azok feladatai

Ezen alfejezet a ROS rendszerben megvalósított csomagok, csomópontok céljait, funkcióit, felépítését ismerteti.

4.4.1. Az ethon_bell csomag

Ezen csomag a fejlesztett vevő robotperiféria által, a robotban lévő számítógép USB portjára, vagyis pontosabban annak sorosportra továbbított adatkeretek fogadását, feldolgozását valósítja meg. Tulajdonképpen egy osztály, amely metódusai a sorosport olvasására, megnyitására, lezárására, a fogadott adatcsomagok feldolgozására adnak lehetőséget. Ezen csomag függvénykészletként szolgál egy további csomópont megvalósítására.

A csomag felépítése az alábbi:

```
ethon_bell
        include
                ethon_bell
                        ethonBellDriver.h
        src
                ethonBellDriver.cpp
        CMakeLists.txt
        package.xml
```

38. ábra Az ethon_bell csomag jegyzékszerkezete

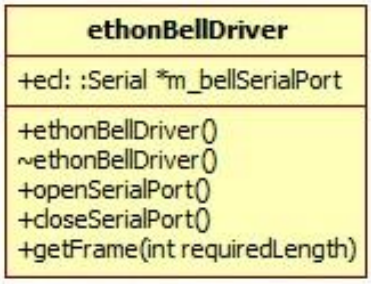
Az include jegyzékben található ethonBellDriver.h az ethonBellDriver.cpp állomány függvénydefinícióit tartalmazza. A package.xml magáról a csomagról tartalmaz információkat, a CMakeLists.txt pedig a csomag fordításához szükséges adatokat, beállításokat tartalmazza. A csomag függősége az ecl csomag, amely a sorosport kezelését

megvalósító függvények gyűjteménye. A fordítási és futtatási függőségek a package.xml állományban találhatóak, a következő formátumban:

```
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend>std_msgs</build_depend>
<run_depend>roscpp</run_depend>
<run_depend>rospy</run_depend>
<run_depend>std_msgs</run_depend>
<build_depend>ecl_devices</build_depend>
<run_depend>ecl_devices</run_depend>
```

39. ábra A függőségek package.xml állománybeli megadási formátuma

Az alábbi ábra az osztály felépítését szemlélteti:



40. ábra Az ethonBellDriver osztály felépítése

Az alábbi táblázat az ethonBellDriver.cpp állomány fontosabb metódusait és azok funkcióit tartalmazza:

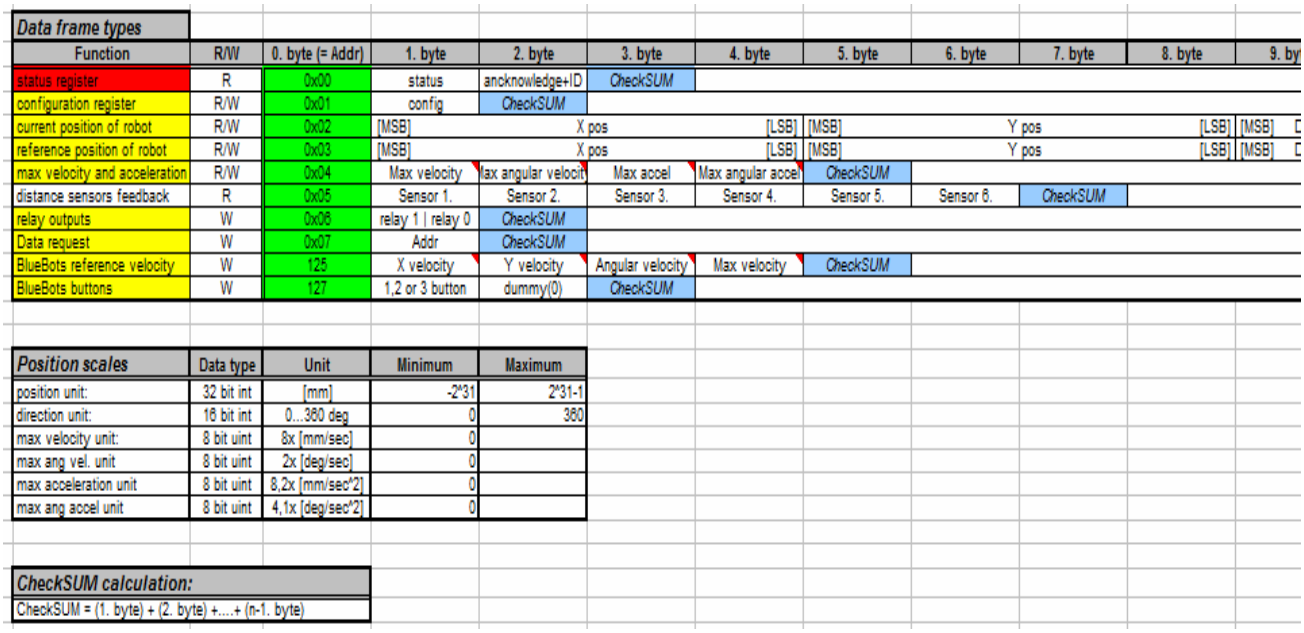
17. Táblázat Az ethonBellDriver.cpp állomány fontosabb metódusai és azok funkciói

Metódus	Funkció
ethonBellDriver::ethonBellDriver()	Konstruktor, a sorosportot szimbolizáló változó példányosítását valósítja meg
void ethonBellDriver::openSerialPort()	Sorosportot nyit, a megfelelő paraméterekkel
void ethonBellDriver::closeSerialPort()	Bezárja a sorosportot
char* ethonBellDriver::getFrame(int requiredLength)	A paraméterében megadott hosszúságú bájtot olvas a sorosportról, és az olvasott adatokat egy char típusú tömbben adja vissza

4.4.2. Az ethon_driver csomag

Ezen csomag feladata, hogy a robot (azaz Ethon) vezéréléséhez szükséges adatkereteket képezze és robot megfelelelő sorosportjára továbbítsa.

Ethon vezérlése előre definiált keretekkel, illetve azok, a robot megfelelő sorosportjára való írásával valósítható meg. A keretek felépítésének részlete az alábbi ábrán látható:



Data frame types

Function	R/W	0. byte (= Addr)	1. byte	2. byte	3. byte	4. byte	5. byte	6. byte	7. byte	8. byte	9. by
status register	R	0x00	status	ancknowledge+ID	CheckSUM						
configuration register	R/W	0x01	config	CheckSUM							
current position of robot	R/W	0x02	[MSB]	X pos	[LSB]	[MSB]	Y pos	[LSB]	[MSB]	[LSB]	[MSB] D
reference position of robot	R/W	0x03	[MSB]	X pos	[LSB]	[MSB]	Y pos	[LSB]	[MSB]	[LSB]	[MSB] D
max velocity and acceleration	R/W	0x04	Max velocity	Max angular velocit	Max accel	Max angular accel	CheckSUM				
distance sensors feedback	R	0x05	Sensor 1.	Sensor 2.	Sensor 3.	Sensor 4.	Sensor 5.	Sensor 6.	CheckSUM		
relay outputs	W	0x06	relay 1 | relay 0	CheckSUM							
Data request	W	0x07	Addr	CheckSUM							
BlueBots reference velocity	W	125	X velocity	Y velocity	Angular velocity	Max velocity	CheckSUM				
BlueBots buttons	W	127	1,2 or 3 button	dummy(0)	CheckSUM						

Position scales	Data type	Unit	Minimum	Maximum
position unit:	32 bit int	[mm]	-2^31	2^31-1
direction unit:	16 bit int	0...360 deg	0	360
max velocity unit:	8 bit uint	8x [mm/sec]	0	
max ang vel. unit	8 bit uint	2x [deg/sec]	0	
max acceleration unit	8 bit uint	8,2x [mm/sec^2]	0	
max ang accel unit	8 bit uint	4,1x [deg/sec^2]	0	

CheckSUM calculation:

CheckSUM = (1. byte) + (2. byte) +....+ (n-1. byte)

41. ábra Ethon vezérlését megvalósító utasításkeretek felépítésének részlete

Ethon vezérlésére összesen 3 darab sorosport áll rendelkezésre, melyek a robotfejnek, a robotkarnak, és az úgynevezett alsórésznek (robotalapnak) a megfelelő adatcsomag küldését és az az általi robotvezérlést teszik lehetővé.

Az alsórészhez egyetlen soros port tartozik, tehát ezen kereteket arra szükséges küldeni. Vezérlés esetében cím+adatbyte+checksum, regiszter lekérdezés pedig 0x07+regiszter cím+checksum a keretformátum. Regiszter lekérdezés esetén tehát először egy data request keretet szükséges küldeni (adott címmel), melyre azt követően küldi vissza a választ a táblázatban lévő formában. A checksum a keretben lévő bájtok összegét jelenti, ha a checksum helyes, akkor végrehajtja az adott utasítást, ha helytelen akkor nem hajtja végre az utasítást, hanem egy „Frame lost the byte” választ küld.

Például ha a státusz regiszter szeretném lekérdezni, akkor a következő keretet kell kiküldeni a soros portra: 0x07 0x00 0x07. Erre a választ 0x07 status ancknowledge+ID checksum formában adja vissza.

A robot helyváltoztatására kétféle lehetőség kínálkozik:

- „BlueBots”-os vezérlés
- Pozícióvezérlés

BlueBots irányítás a táblázatban található „BlueBots reference velocity” nevezetű keret küldésével valósítható meg. Ez azt jelenti, hogy a robot kap x, y pozíciót, szöggyorsulást, és sebesség referenciát, majd ennek hatására az adott x, y pozícióba, az adott szögelfordulással és sebességgel elmozdul.
Másik lehetőség a referencia és a célpozíció megadása. Ez a táblázatban található „reference posotion of robot” és a „current position of robot” keretek segítségével adható meg. Ezt a működési módot előbb engedélyezni kell, melyhez a 0x01 0x01 checksum utasítás tartozik. Ezt követően az aktuális pozíció megadása szükséges (általában 0), melyhez képest a robot elmozduljon. Koordináta, szögelfordulás, gyorsulás és sebesség megadása után Ethon az adott x, y pozícióba indul, úgy, hogy az éppen x, y koordinátába éréskor fejezi be az elfordulást is, tehát egyszerre halad az adott irányba és közben fordul az adott szögben. Ez a mikorvezérlő programjában található trajektória tervezésnek köszönhetően működik.

A csomag felépítése az alábbi:

```
ethon_driver
        include
                ethon_driver
                        EthonDriver.h
        src
                EthonDriver.cpp
        CMakeLists.txt
        package.xml
```

42. ábra Az ethon_driver csomag jegyzékszerkezete

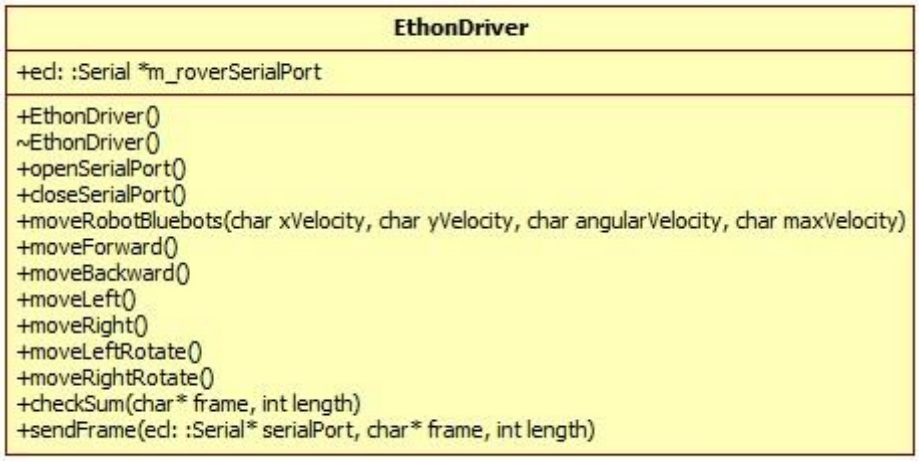
Az include jegyzékben található EthonDriver.h az EthonDriver.cpp állomány függvénydefinícióit tartalmazza. Az előző ethon_bell csomaghoz hasonló módon a package.xml magáról a csomagról tartalmaz információkat, a CMakeLists.txt pedig a csomag fordításához szükséges adatokat tartalmazza.

Az „EthonDriver” osztály feladata, hogy magasabb szintű utasításokat (pl.: menj előre) átfordítsa a robot nyelvére, azaz a magas szintű utasításnak megfelelő keretet/kereteket összeállítsa és elküldje a robot asló részének szóló soros portra.

Az osztálybeli soros portok beállításai:

- Baudrate: 115 200
- Adatbitek: 8
- Stop bitek: 1
- Paritás: 0
- Flow control: nincs

Az alábbi ábra az osztály felépítését szemlélteti:



43. ábra Az EthonDriver osztály felépítése

Az alábbi táblázat az EthonDriver osztály fontosabb metódusait és azok funkcióit tartalmazza:

18. Táblázat Az EthonDriver osztály metódusai és azok funkciói

Metódus	Funkció
EthonDriver::EthonDriver()	Konstruktor, a sorosport példányosítását valósítja meg
void EthonDriver::openSerialPort()	Ethon alsó részéhez tartozó sorosportot nyitja meg a megfelelő paraméterekkel
void EthonDriver::closeSerialPort()	Ethon alsó részéhez tartozó sorosportot

	zárja be
void EthonDriver::moveRobotBluebots(char xVelocity, char yVelocity, char angularVelocity, char maxVelocity)	A robot úgynevezett „BlueBots”-os pozícióváltoztatásához szükséges adatkeretek összeállítását valósítja meg a paraméterében megadott adatok alapján, majd azt az alsórészhez tartozó sorosportra írja
void EthonDriver::moveForward()	A moveRobotBluebots metódust megfelelően paraméterzve a robot előrehaladását valósítja meg
void EthonDriver::moveBackward()	A moveRobotBluebots metódust megfelelően paraméterzve a robot hátrahaladását valósítja meg
void EthonDriver::moveLeft()	A moveRobotBluebots metódust megfelelően paraméterzve a robot bal irányban történő elmozdulását valósítja meg
void EthonDriver::moveRight()	A moveRobotBluebots metódust megfelelően paraméterzve a robot jobb irányban történő elmozdulását valósítja meg
void EthonDriver::moveLeftRotate()	A moveRobotBluebots metódust megfelelően paraméterzve a robot saját tengelye körüli balra forgását valósítja meg
void EthonDriver::moveRightRotate()	A moveRobotBluebots metódust megfelelően paraméterzve a robot saját tengelye körüli jobbra forgását valósítja meg
char EthonDriver::checkSum(char* frame, int length)	A paraméterében megadott hosszúságú keretre kiszámolja az ellenőrzőösszeget
void EthonDriver::sendFrame(ecl::Serial* serialPort, char* frame, int length)	A paraméterében megadott sorosportra, a megadott hoszzúságú adatkeret küldését valósítja meg

4.4.3. Az ethon_bell_node csomópont

Ezen csomópont (node) a ROS rendszerbeli futtatható alkalmazás, amely célja, hogy az ethon_bell csomag felhasználásával olyan adatokat továbbítson üzenetsoron keresztül, amelyek következtében a robot vezérlése megvalósul.
Az ethon_bell csomag metódusai felhasználásával adott frissítési rátával (100 Hz-el) meghívja a csomag sorosport olvasó metódusát, amely visszaadja a sorosportra érkezett, arról olvasott adatokat. Ha a feljesztett robotperifériák által küldött adatcsomag érkezett, akkor azokat értelmezi, feldolgozza, az egyéb adatcsomagokat pedig figyelmen kívül hagyja, azaz eldobja. Ha ténylegesen a robotperifériáktól származó adatcsomag érkezett, akkor annak feldolgozása után üzentet publikál a command nevű topikba.

A csomag felépítése az alábbi:

```
ethon_bell_node
        include
                ethon_bell_node
        src
                ethon_bell_node.cpp
        CMakeLists.txt
        package.xml
```

44. ábra Az ethon_bell_node csomópont jegyzékszerkezete

Az ethon_bell_node.cpp állomány olyan ROS specifikus utasíátsokat tartalmaz, amelyek által a csomópont működése megvalósítható. Az előző csomagokhoz hasonló módon a package.xml és a CMakeLists.txt állományok felépítése, funkciói megegyeznek az előbbi csomagok esetében is ismeretett funkciókkal. A CMakeLists.txt állomány az előző csomagok CMakeLists.txt állományához képest kiegészül a csomópont futtatásához, fordításához szükséges függőségek megadásával. Mivel ezen node az ethon_bell csomag metódusait felhasználva valósítja meg feladtát, így a fordításához és futtatásához is szükséges függőség az etohn_bell csomag. Az ethon_bell csomag importálása az alábbi CMakeLists.txt-beli sorok segítségével valósul meg:

```
catkin_package(
  INCLUDE_DIRS include
  LIBRARIES ethon_bell_node
  CATKIN_DEPENDS roscpp rospy std_msgs ethon_bell
#  DEPENDS system_lib
)
```

45. ábra A CMakeLists.txt tartalmának azon részlete, amley a függőségeket beállítja illetve az ethon_bell csomag importálását megvalóstja

Tehát ezen node az eseméyek fogadását megvalósító robotperiféria által fogadott adatok feldolgozását, ellenőrzését és helyes adatcsomag fogadása esetén üzenet topikba való publikálását valósítja meg. Topik használtához, illetve üzenet topikba való publikálásához azért van szükség, mert a robot (azaz Ethon) tényleges működtetését, vezérlését egy másik csomópont fogja megvalósítani.
A node a rosrun ethon_bell_node ethon_bell_node parancs kiadásával indítható el.

4.4.4. Az ethon_node csomópont

Ezen csomópont (node) szintén a ROS rendszerbeli futtatható alkalmazás, amely célja, hogy az ethon_driver csomag felhasználásával megvalósítsa a robot tényleges vezérlését, működtetését. A robot vezérlésének alapja a fejlesztett adó robotperiférиák által elküldött, majd az elküldött események vételére alkalmas robotperiféria által fogadott bináris események. Ezen csomópont teszi lehetővé, hogy a „csengetés” esemény hatására a robot az ajtóhoz pozícionáljon, illetve a „távirányítás” esemény hatására pedig a megfelelő irányba elmozduljon.
A node a commad topikra feliratkozva figyeli az abba, az ethon_bell_node által publikált üzeneteket, majd a megfelelő üzenet érkezése esetén az ethon_driver csomag megfelelő metódusainak felhasználsával megvalósítja a robot vezérlését, vagyis annak működtetéséhez szükséges adatkeretek sorosportra való írását.

A csomag felépítése az alábbi:

```
ethon_node
        include
                ethon_node
        src
                ethon_node.cpp
        CMakeLists.txt
        package.xml
```

46. ábra Az ethon_node csomópont jegyzékszerkezete

Az ethon_node.cpp állomány szintén, az ethon_bell_node csomópont ethon_bell_node.cpp állományához hasonlóan ROS specifikus utasíátsokat tartalmaz, amelyek által a csomópont működése megvalósítható. Az előző csomagokhoz hasonló módon a package.xml és a CMakeLists.txt állományok felépítése, funkciói megegyeznek az előbbi csomagok esetében is ismeretett funkciókkal. Mivel ezen node az ethon_driver csomag metódusait felhasználva valósítja meg feladtát, így a fordításához és futtatásához is szükséges függőség az etohn_driver csomag.

A node a rosrun ethon_node ethon_node parancs kiadásával indítható el.

5. Összefoglalás

Egy robotvezérlő rendszer tervezésére és megvalósítására alakult a Miskolci Egyetem mérnök informatikus és villamosmérnök hallgatóiból álló néhány fős csapat, amely csapat célja, hogy egy konkrét robotot működtető robotvezérlő rendszer részeként megvalósuló szoftvert tervezzen és implementáljon. Ezen rendszer célja, hogy a BME által tervezett, fejlesztett Ethon nevezetű robot portás funkcióját megvalósítsa, Ethon a portás robot funkcióját az ELTE Etológia tanszékének folyosóján hivatott betölteni. Mivel ezen rendszer tervezése, implementálása egy ember számára túl nagy feladat, így azt több részre osztottuk. A felosztás következtében mindenki a saját tématerületével foglakozott, amely következtében a teljes rendszert alkotó szoftvermodulok kerültek megvalósításra.

Dolgozatom keretében saját munkám került bemutatásra, amely során mélyebb bepillantást nyertem a robotika világába, megismerkedtem a BME által fejlesztett Ethon nevezetű robot felépítésével, illetve a különböző robotvezérlő keretrendszerek felépítésével, működésével, részletesebben tanulmányoztam az OpenRTM és a ROS rendszer működését, felépítését. Terveztem és megvalósítottam bináris események továbbítására és a továbbított bináris események fogadására alkalmas robotperifériákat, illetve a robotperifériákat működtető, a ROS rendszerben implementált, több komponensből álló szoftvermodult.

A fejlesztett robotperifériák beágyazott rendszereket, áramköröket takarnak, amelyek segítségével a robot adott funkciója a ROS rendszerben implementált szoftverkomponensek által működtethető. A fejlesztett robotperifériák tulajdonképpen adatkereteket továbbító és azokat fogadó adó és vevő áramkörök, melyek vezetéknélküli kapcsolaton keresztül kommunikálnak. Az adó periféria segítségével bináris események válthatóak ki, mely bináris eseményeket a robot számítógépében helyet foglaló, annak USB portján keresztül csatlakoztatott vevő periféria fogadja. A vevő áramkör által fogadott bináris eseményeket a ROS rendszerben implementált szoftverkomponensek dolgozzák fel, illetve valósítják meg a robot bináris események alapján történő vezérlését. A bináris események az adó periférián lévő nyomógombok segítségével válthatóak ki. Az egyik esemény célja, hogy a robot számára egyfajta „csengető” jelzésként funkcionáljon. Mivel Ethon célja portás funkció megvalósítása az ELTE etológia tanszékének folyosóján, így hasznos funkció, hogy a tanszék zárt ajtaját csak engedéllyel rendelkező személyek léphessék át. A megvalósított „csengetés” funkció egy tényleges csengőként funkcionál a robot számára, azaz a zárt ajtót átlépni kívánó személy a fejlesztett adó periféria

segítségével egy „csengetés” eseményt küld a robot számára, majd az, az esemény hatására az ajtóhoz pozícionál és mágneskártyája segítségével ajtót nyit az illető számára.
Egy másik megvalósított bináris esemény a robot távvezérlésére ad lehetőséget. Ezen funkció által a robot pozíciója adott irányba változtatható, azaz a mozgása távvezérelhető. Ezen funkció megvalósításához egy újabb adó periféria került tervezésre, amely nyomógombjai segítségével távirányítóként funkciónál és szintén vezetéknélküli kapcsolaton keresztül kommunikál a robot számítógépében helyet foglaló, USB-n keresztül csatlakoztatott vevő perifériával. Ethon távvezérlését a fogadott bináris események által a ROS rendszerben implementált szoftverkomponens valósítja meg.
A feladat megvalósítása során rengeteg tapasztalatot szereztem, amely által értékes útravalót kaptam életem további állomásaira.

6. Summary

A robot controller system onto the planning of a system and his realisation University of Miskolc was formed engineer some people teams consisting of information engineers and electrical engineers listeners, that a team of goal, that an actual robot controller software plan and implement. The goal of this system, that by way of BME planned, developed Ethon let a robot accomplish a doorkeeper's function, Ethon the doorkeeper the function of a robot on the corridor of ELTE Ethology department meant to fill. We divided it into more parts so since the planning of this system, his implementing are too big tasks for a man. Everybody is tooth lodge with his own topic area, as a result of which the software modules forming the full system were realize, as a result of the partition.

My own work, in the course of which I gained deeper insight, was presented in the framework of my paper into the world of the robotics, got acquainted Ethon developed by BME with the construction of a robot, concerned the different robot controller framework I studied it with the construction of framework systems, with his function, in detail OpenRTM and ROS the function of a system, his construction. I planned it and realized onto the transmission of binary events and the reception of the binary events sent on suitable robot peripheries, concerning the robot peripheries operator, ROS software module consisting of more components implemented in a system. The developed robot peripheries as embedded systems, electronic circuits, that his function of the robot ROS by way of software components implemented in a system operable. The developed robot peripheries tax receiving them sending on data frames actually and customer electronic circuits, which ones wirelles they communicate through a contact. The tax with the help of a periphery binary events can be changed who, which is binary, events in the computer of the robot make way deposit, it through USB port connected customer a periphery receives it. The customer ROS implemented binary events received by an electronic circuit in a system software components digest it, they accomplish the control of the robot happening based on binary events concerned. The binary events the tax with the help of push buttons on a periphery can be ransomed. The aim of one of the events, that for the robot uniform ringing let him serve as a mark. Since Ethon aim is a doorkeeper the realisation of a function ELTE on the corridor of the department of ethology, like this useful function, that let persons at who a permit is only be allowed to step over the department's closed door. The realized ringing a function serves as an actual bell for the robot, that is to step over the closed door wishing person the developed tax a ringing sends an event for the robot with

the help of a periphery, then it, due to the event to the door calibrates and it opens a door for a person with of his magnetic card. An other realized binary event provides an opportunity for the remote control of the robot. By way of this function the position of the robot into a given direction can be made change in, that is his motion remote control. To the realisation of this function a newer tax a periphery got to planning, that with the help of his push buttons as remote control function and likewise wirelles a deposit communicates place through a contact in the computer of the robot, through USB connected customer with a periphery. Ethon remote control by way of the adopted binary events ROS a software component implemented in a system accomplishes it.

I obtained many experiences, by way of which I got a valuable sendoff for the additional stations of my life, in the course of the realisation of the task.

7. Irodalomjegyzék

[1]
http://project.mit.bme.hu/mi_almanach/books/aima/ch25s01

[2]
Haláchy Nóra, Schmidt Dorottya - Esztétikus markerek robot lokalizációhoz (TDK dolgozat)

[3]
http://www.turtlebot.com/

[4]
http://wiki.ros.org/Robots/TurtleBot

[5]
http://kobuki.yujinrobot.com/home-en/about/specifications/

[6]
http://turtlebot.com/assets/templates/turtlebot/img/turtlebot_2_lg.png

[7]
http://www.mk.unideb.hu/userdir/vmt2/images/tantargyak/eloadasanyag/intelligens_robot_szerk.pdf

[8]
http://www.openrtm.org/openrtm/sites/default/files/440/rtsystem_integration_en.png

[9]
http://en.wikipedia.org/wiki/Robot_Operating_System

[10]
http://www.sze.hu/~herno/NGB_IN039_1/docs_ros/ros_02.pdf

[11]
Aaron Martinez, Enrique Fernández - Learning ROS for Robotics Programming

[12]
http://rs1.sze.hu/~herno/NGB_IN039_1/docs_ros/ros_02.pdf

[13]
http://wiki.ros.org/ROS/Tutorials

[14]
http://robotmodell3d.hu/TDK2.pdf

[15]
http://openrtm.org/

[16]
http://openrtm.org/openrtm/en/content/rtsystemeditor-100

[17]
Noriaki Ando, Takashi Suehiro, and Tetsuo Kotoku - A Software Platform for Component Based RT-System Development: OpenRTM-Aist

[18]
http://www.mit.bme.hu/system/files/oktatas/targyak/8607/IA08BeagyazottRendszer.pdf

[19]
http://hu.wikipedia.org/wiki/Atmel_AVR

[20]
http://hu.wikipedia.org/wiki/PIC_mikrokontroller

[21]
http://www.inf.u-szeged.hu/projectdirs/bohusoktat/regi/szganyagok/esszek/ARM_Architektura.pdf

[22]
http://hu.wikipedia.org/wiki/ARM_architekt%C3%BAra

[23]
http://www.atmel.com/images/doc2543.pdf

[24]
http://www.ventor.co.in/images/categories/attiny2313-20PU.jpg

[25]
http://www.gaw.ru/im/atmel/avr/tiny2313_p.gif

[26]
http://arduino.cc/en/uploads/Main/FTDI_adaptor.jpg

[27]
http://www.pocketmagic.net/wp-content/uploads/2013/07/fa1000a_2_avr.jpg

[28]
http://unsuspectingbit.com/category/projects/

[29]
http://www.elcesiyejn.com/IC/All%20data%20modules/1402.zip

[30]
http://www.arek.uni-obuda.hu/opto/3_Modulacio.htm#A%20modul%C3%A1ci%C3%B3

[31]
www.mazsola.iit.uni-miskolc.hu/DATA/segedletek/digjelf/fejezet2.doc

[32]
http://www.hobbielektronika.hu/cikkek/434mhz_wireless.html

[33]
http://www.atmel.com/tools/atmelstudio.aspx

[34]
http://www.wvshare.com/img/preview/USB-AVRISP_1.jpg

[35]
http://plc.mechatronika.hu/elektro/pdf/eagle_6_magyar.pdf

[36]
http://www.element14.com/community/servlet/JiveServlet/downloadBody/23825-102-1-70962/CadSoft%20EAGLE%20%5Bmagyar%20nyelv%C5%B1%20v%C3%A1ltozat%5D.pdf

Hivatkozások ellenőrzésének dátuma: 2014.05.05.

Köszönetnyilvánítás

Köszönöm *Dr. Kovács Szilveszter* konulensemnek, hogy lehetőséget biztosított dolgozatom elkészítéséhez, köszönöm segítőkész támogatását, hasznos tanácsait, illetve, hogy tanácsaival és iránymutatásaival hozzájárult szakmai fejlődésemhez. Köszönöm *Dr. Krizsán Zoltánnak* és *Dr. Vincze Dávidnak*, hogy hasznos tanácsaikkal segítették munkámat.

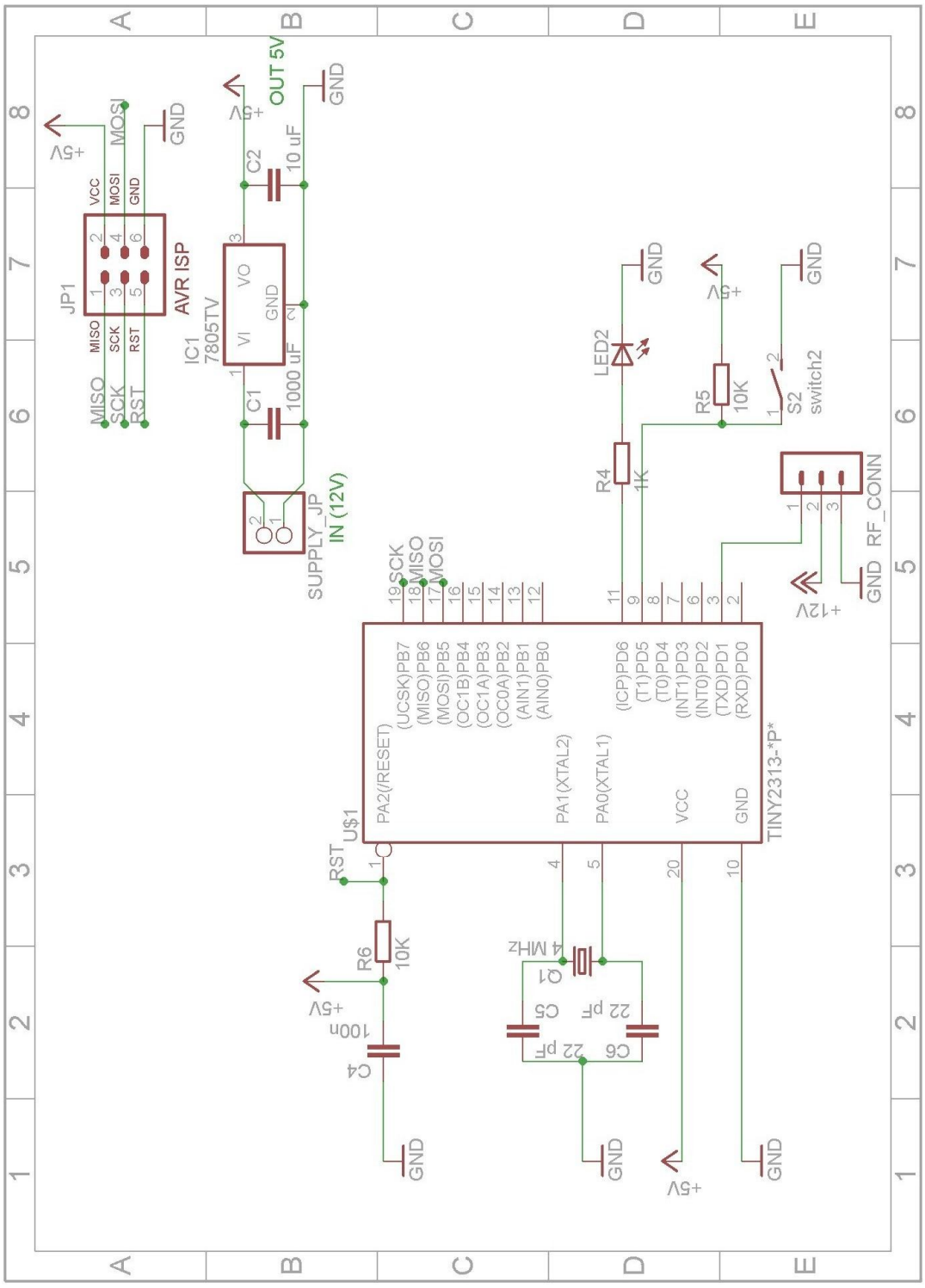
Köszönöm barátaimnak, *Vikinek*, *Fruzsinak*, *Nikinek*, *Csillának*, *Andrásnak* és *Rolandnak*, hogy bátorításaikkal, tanácsaikkal végig segítségül voltak utamon.

Köszönöm!

8. Nyomtatott mellékeltek

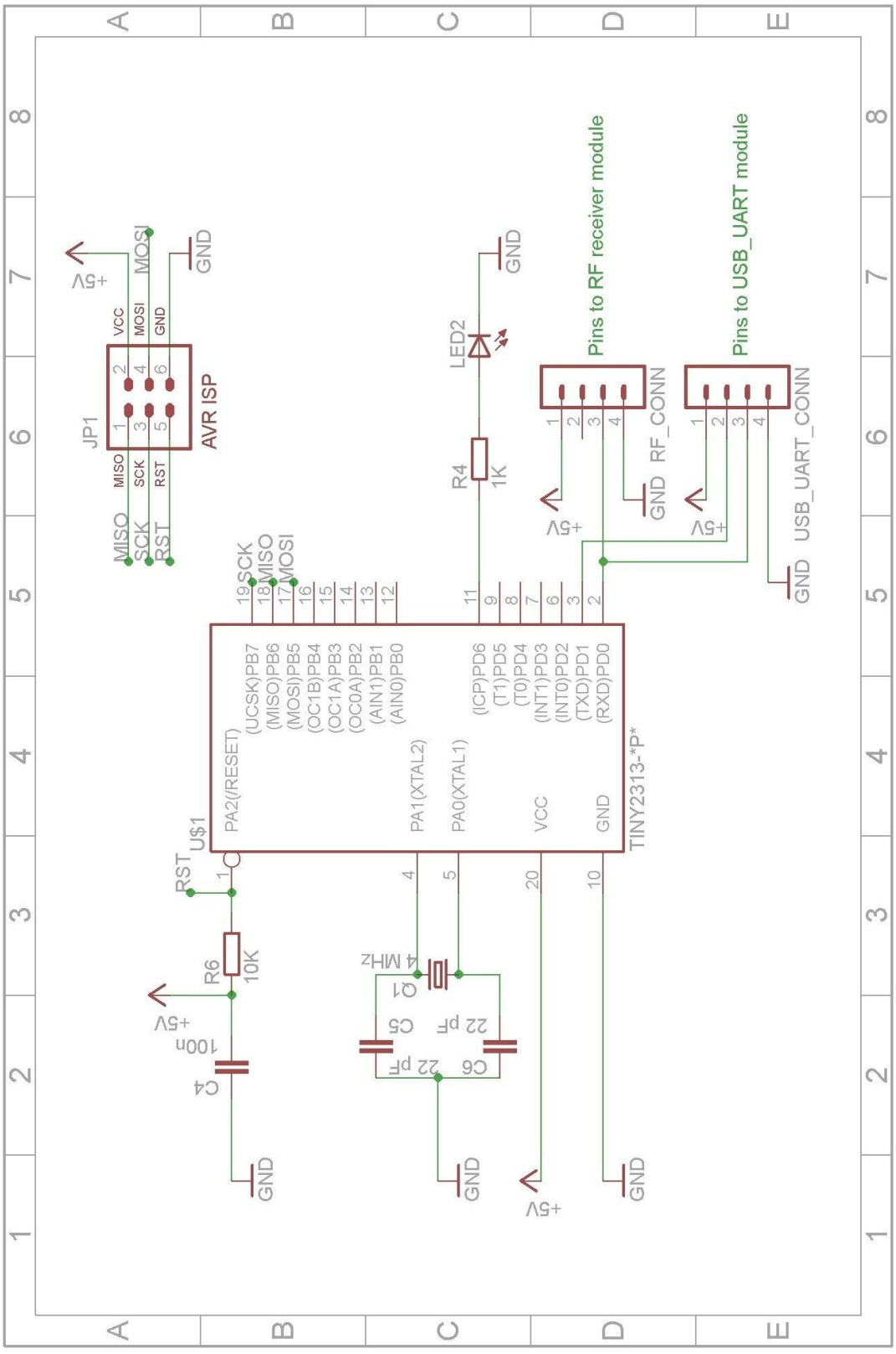
8.1 1. Melléklet

A bináris események küldésére alkalmas robotperiféria kapcsolási rajza:



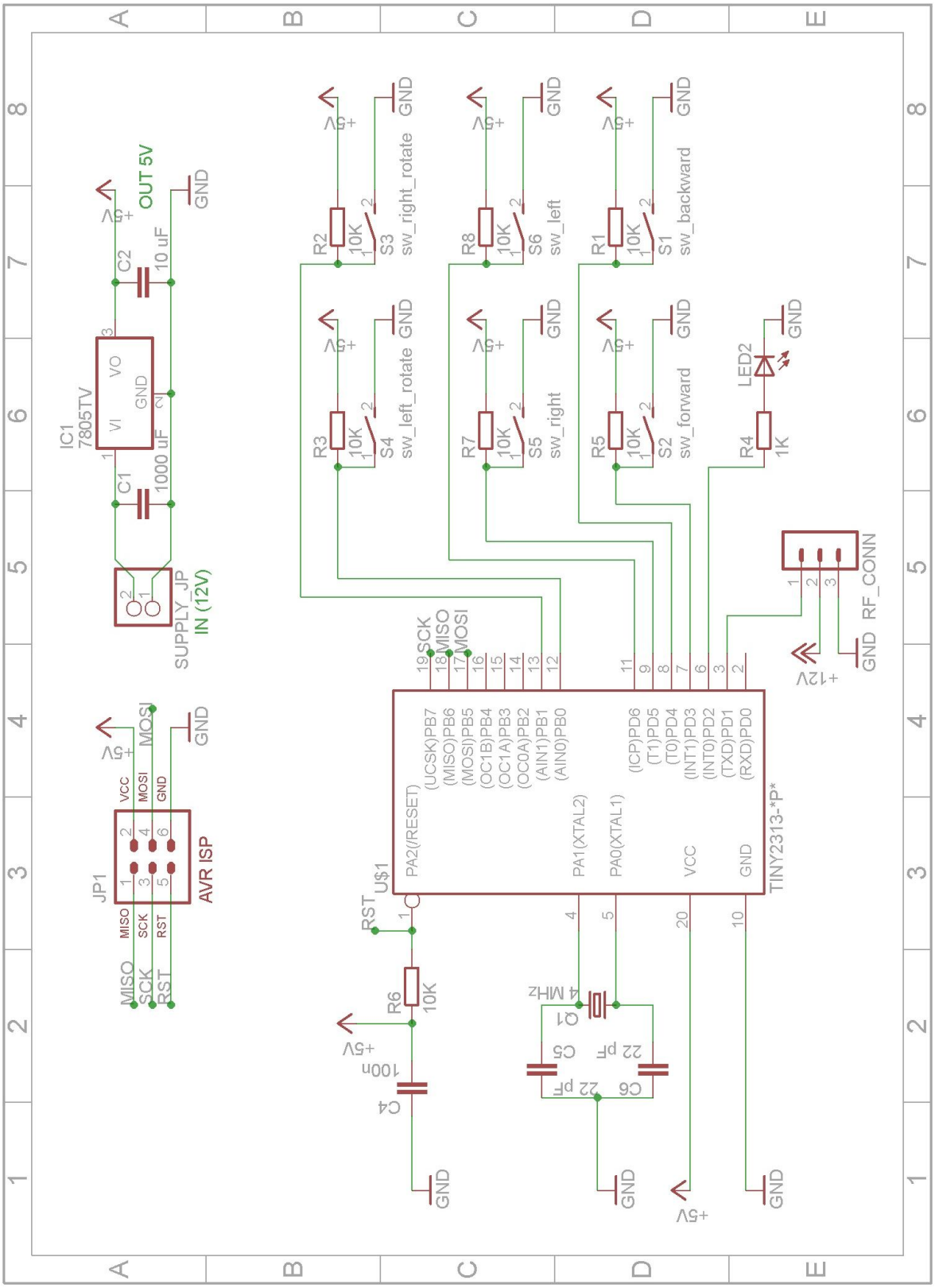
8.2 2. Melléklet

A bináris események fogadására alkalmas robotperiféria kapcsolási rajza:



8.3 3. Melléklet

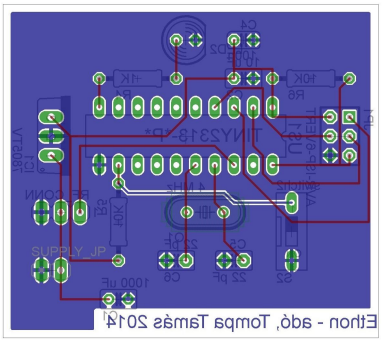
A robot távvezérlését lehetővé tevő robotperiféria kapcsolási rajza:



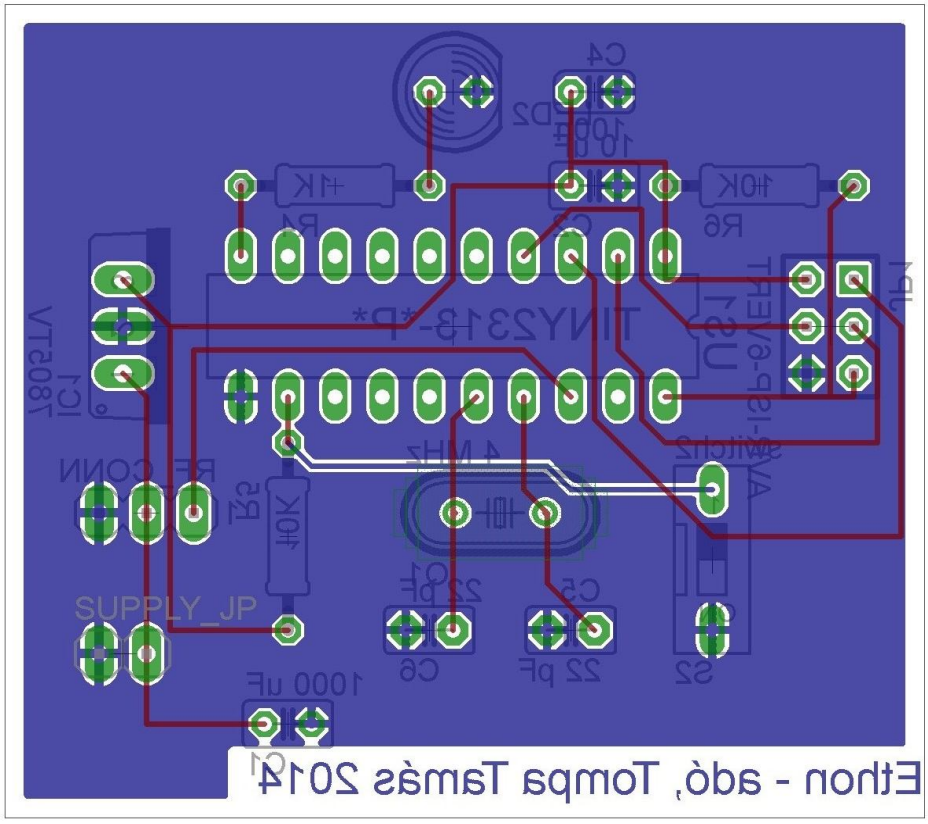
8.4 4. Melléklet

A bináris események küldésére alkalmas robotperiféria nyomtatott áramköri terve:

Méretarányos:



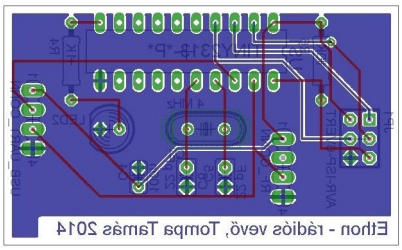
Felnagyított:



8.5 5. Melléklet

A bináris események fogadására alkalmas robotperiféria nyomtatott áramköri terve:

Méretarányos:



Felnagyított: