Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Háda Tamás Menyhért

HORDOZHATÓ LOPÁSGÁTLÓ

Mikrokontroller és androidos készülék kapcsolata Bluetooth Smart-on keresztül

KONZULENS

Dr. Benesóczky Zoltán

BUDAPEST, 2015

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Háda Tamás Menyhért**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2015. 12. 11.

.....
Háda Tamás Menyhért

Összefoglaló

Napjainkban a különböző vezeték nélküli elektronikai készülékek és használati tárgyak egyre inkább elterjednek az élet minden területén. Ez fontos fejlődési folyamat, hiszen a mobilitás elengedhetetlen a mai modern, rohanó világban. A fitnesz, egészségügyi, telekommunikációs, szórakoztatóipari és sok egyéb területen használt eszközök egyre inkább támogatják a Bluetooth Smart technológiát. A klasszikussal szemben ez a kommunikációs forma jóval kisebb energiafogyasztással jár, ami rendkívül versenyképessé teszi a piacon. Egyre több vállalat integrálja eszközeibe, ezáltal rengeteg fejlesztési lehetőséget biztosít.

Manapság már az okostelefonok számos lehetőséget biztosítanak a különböző napi feladataink elvégzésére, ami fontos kényelmi szempont. Szakdolgozatom célja egy olyan Bluetooth Smart alapú rendszer elkészítése, mely az androidos készülékek által nyújtott lehetőségeket bővíti ki azáltal, hogy az említett technológiát felhasználva egy ultrahang elvű lopásgátló eszközhöz csatlakozik.

Jelen szakdolgozatban a Bluetooth Smart technológiát fogom bemutatni és ismertetem a lopásgátló hardveres felépítését és egyes részei közötti kommunikációt, illetve tárgyalni fogom a mikrokontroller és Android oldali szoftverek működését. A megtervezett, végleges eszköz központi eleme az említett Bluetooth Smart modul, fő funkciója pedig, hogy az értéktárgyunk mellé helyezve azt, az androidos eszközünkön riasztást kapjunk az esetleges eltulajdonításáról.

Abstract

These days wireless electronic devices increasingly spreading all over across in every aspect of our life. This is an important progress because mobility became an essential element in our modern, rushing world. Just to mention a few: fitness, healthcare, telecommunication, entertainment industry and much other areas are those which starting to support Bluetooth Smart technologies. Opposing the classical form of communication this form consume much less energy which makes it competitive in the market. More and more company willing to integrate this technology in their product which offers a huge potential for development.

Nowadays even smarthphones provide a chance to complete our everyday tasks which is an important part in our comfort. With my thesis my aim is to build a Bluetooth Smart based Android system extension which is capable of connecting to an ultrasound based anti theft device.

In this thesis I'll present the Bluetooth Smart technology and also the hardware structure of the anti theft system and the communication between It's segments. In addition I'll review how microcontrollers and android softwares works. The fully designed and completed device's main task is to being able to protect our valuables if we put the mechanism next to it, and It's core is Bluetooth Smart based. As soon as the act of thiefery will occur we will get an alert to our Android device.

1 Bevezetés

A szakdolgozat megírása során a Bluetooth Low Energy vagy más néven Bluetooth Smart eszközök alaposabb megismerése és használatának elsajátítása volt a kitűzött cél. Manapság már rengeteg hordozható eszköz tartalmaz ilyen modult, ezért egy ilyen egységet tartalmazó eszköz elkészítése hasznos tapasztalatokat adhat a fejlesztő számára. A Bluetooth Smart és az androidos alkalmazások terén szerzett tudás versenyképes lehet, hiszen mindkét irány az elmúlt években hatalmas fejlődésnek indult. Egymást kölcsönösen támogatják egyre jobban, az újabb és újabb verziók megjelenésével. Az androidos oldalon rengeteg beépített függvény nyújt segítséget a fejlesztőnek egy Bluetooth Smart alapú alkalmazás elkészítésében. A Bluetooth modulok terén is igen nagy a választék, így a feladatunknak legmegfelelőbbet választhatjuk ki. Ennek a két iránynak a piaca egyre bővül és az élet minden terére lassan eljut. Az ilyen eszközök alacsony fogyasztása pedig a környezet kímélő hatás és az energiatakarékosság miatt is fontos, egyszerű működése pedig lehetővé teszi, hogy akár saját otthonunkat felszereljük ilyen modulokkal és a mobiltelefonunkról elérve azokat különböző feladatokat elvégezzünk egy érintéssel, ezzel kényelmesebbé téve a mindennapokat. Ezen szempontok alapján választottam a szakdolgozat témámat, vagyis a Bluetooth Smart alapú eszközöket.

2 Irodalomkutatás

A XXI. századra, az újabb és újabb technológiai újításoknak köszönhetően, az elektronikai eszközök eljutottak arra a szintre, hogy már nem csak vezetékeken keresztül, és ennek hátrányaként helyhez kötve, kommunikálhatnak egymással, hanem vezeték nélkül is. Ez a rádiós összeköttetések rohamos fejlődésének köszönhető, amely rádióhullámokon keresztüli kommunikációval váltotta fel az addig megkerülhetetlen kábelen keresztüli kommunikációt a mobilitás kényelmét adva a felhasználók számára. A mobiltelefonok jelentették az első lépést a hordozhatóság, mint személyi kényelem, eléréséhez. Az első eszköz piacra kerülése után néhány éven belül teljesen elterjedtek, mert használatuk praktikus, kényelmes és szükséges is volt. Hamar felvetődött azonban a kérdés, hogy milyen módon lehetne a mobiltelefont rövid távolságon belül is vezeték nélkül összekötni egy másik mobiltelefonnal, vagy akár más eszközzel (például PDA vagy fejhallgató). Ennek szükségességét legelőször 1994-ben, az egyik legnagyobb telekommunikációs vállalat, az Ericsson ismerte fel [13] . Néhány másik vállalattal együtt megalapították a SIG-et (Special Interest Group), melynek feladatául egy olyan vezeték nélküli, univerzális szabvány kidolgozását tűzték ki, amely lehetővé teszi a különböző számítástechnikai, kommunikációs és kiegészítő eszközök rövid távú, kis teljesítményű rádiós összeköttetését, olcsó rádiós adó-vevők segítségével. Ezek lettek a Bluetooth szabvány alapjai. Az első szabványt (Bluetooth 1.1) már 2001-ben elfogadta az IEEE, és ettől kezdve gyors fejlődésnek indult a technológia. Néhány évvel később, az első okostelefonok megjelenésével, egyre nagyobb szerephez jutott, és már nem csak a telefonok tartalmaztak ilyen modult. Ez a technológia az egészségügyi, biztonsági, fitnesz és sok más eszközcsalád körében is elterjedt. A Bluetooth képes eszközök robbanásszerű elterjedése során figyelt fel a SIG és más vállalatok is arra, hogy a klasszikus Bluetooth túl sokat fogyaszt. A cél tehát az lett, hogy egy jelentősen kisebb fogyasztású, Bluetooth technológiát fejlesszenek ki, mellyel a hordozható eszközök is tovább bírják töltés vagy elemcsere nélkül. Ez kulcsfontosságú szempont ugyanúgy az egészségügyi vagy a fitnesz termékeknél, mint a mobiltelefon esetében. Az energiatakarékosság jegyében tehát a Nokia elkezdte egy minden addiginál alacsonyabb energiaigényű adatátviteli technológia kidolgozását, melynek első változata 2006-ban Wibree néven vált ismertté [4] . Egy évvel később már a Bluetooth SIG támogatását

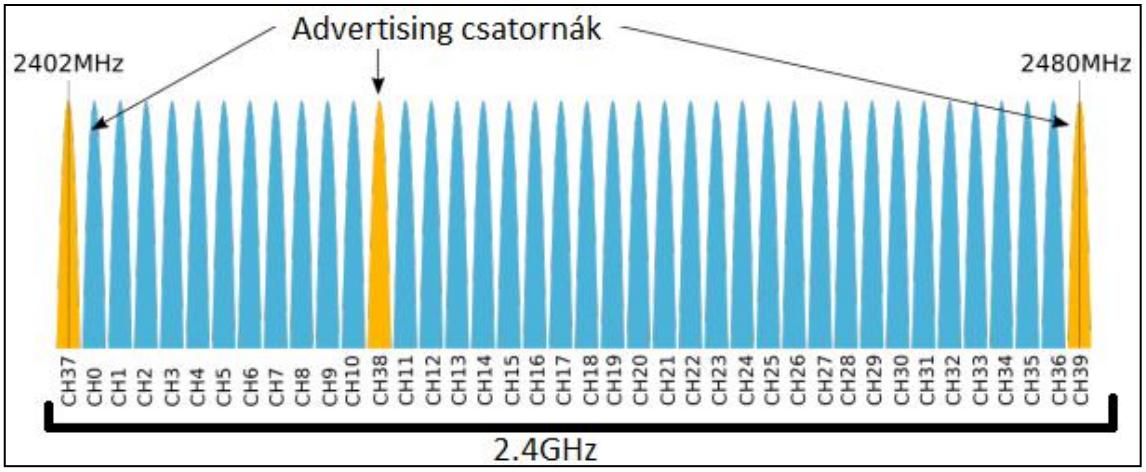
élvezte, és Bluetooth 4.0 LE (Low Energy) néven élt tovább. A szabvány ezután még egy névváltoztatáson átesett. Manapság már Bluetooth Smart illetve Bluetooth Smart Ready néven ismerhető. A kettő közötti különbség, hogy míg a Smart csak a kizárólag alacsony fogyasztású és emiatt limitált funkcionalitású eszközök számára létrehozott, lassabb, egymódú átviteli technológia, addig a Smart Ready már az alacsony fogyasztás mellett nagysebességű adatátvitelre is képes, kétmódú megoldás. Ezekkel a Bluetooth modulokkal megoldhatóak olyan feladatok, mint a futás közbeni pulzusmérés, és a mért adatok átküldése az okosórának, zenehallgatás vezeték nélküli headset-tel, otthonunkban az ajtózár nyitása, a lámpák, a televízió vagy egyéb más hasonló modullal rendelkező eszköz vezérlése az okostelefon vagy a táblagép segítségével.

2.1 A Bluetooth Smart

A Bluetooth Smart tehát egy vezeték nélküli, személyi hálózati technológia, mely használata jelentősen kisebb energiafogyasztással jár, mint a hagyományos Bluetooth modul esetében. A 4.0 Bluetooth szabványba integrálták, és az újabb mobil operációs rendszerek (mint például az Android 4.3 és magasabb, vagy az iOS és Windows phone) is széles körben támogatják már, a korábbiakkal ellentétben, ez a verzió viszont az előzőekkel nem kompatibilis. Ennek ellenére léteznek olyan eszközök, melyekben mindkét technológia megtalálható. Legfőképpen az okostelefonok és táblagépek nyújtotta támogatás miatt olyan népszerű a Bluetooth Smart. Használatával egyszerűen fejleszthetőek eszközök, melyek könnyedén kommunikálhatnak mobiltelefonunkkal. A telefonra szintén egyszerűen fejleszthetünk Smart modullal kapcsolatos alkalmazást. Az Apple is jelentős támogatásokat nyújt a Smart technológiának, melynek következtében számos kis méretű, kis fogyasztású elemes termék került a cégtől a piacra. Az Android hasonlóképpen segíti a fejlesztőt beépített függvényeket tartalmazó könyvtárakkal, melyek az újabb és újabb API verziók kiadásával folyamatosan bővülnek [1] . Ilyen előnyök mellett tehát kézenfekvő lehet a vezeték nélküli eszközök fejlesztésénél a Bluetooth Smart választás.

2.1.1 Fizikai Réteg

A fizikai réteg alatt azt értjük, hogy a Smart modul, hogyan küldi és fogadja az adatokat vezeték nélkül. A Bluetooth Smart a megszokott 2.4 GHz-es ISM sávot használja, ugyanúgy mint a klasszikus Bluetooth. Ezért is lehetséges olyan eszközt készíteni, amely mindkettőt tartalmazza, mert így osztozhatnak a rádióantennán. A sáv a 2400 MHz-től a 2483.5 MHz-ig terjed. A Bluetooth v4.0 (és feljebb) ezt a spektrumot osztja fel 40 csatornára. Ebből hármat használ az úgynevezett Advertising (hirdető) módban.



2.1. ábra: Bluetooth Smart működési spektrum[2]

A két technológia főleg azért nem kompatibilis, mert más a modulációs indexük, és különböző számú csatornát használnak. A Smart 40 darab 2 MHz-es csatornájához képest a klasszikus modul 79 darab 1 MHz-es csatornát használ és ezek ráadásul máshogy is helyezkednek el a spektrumban.

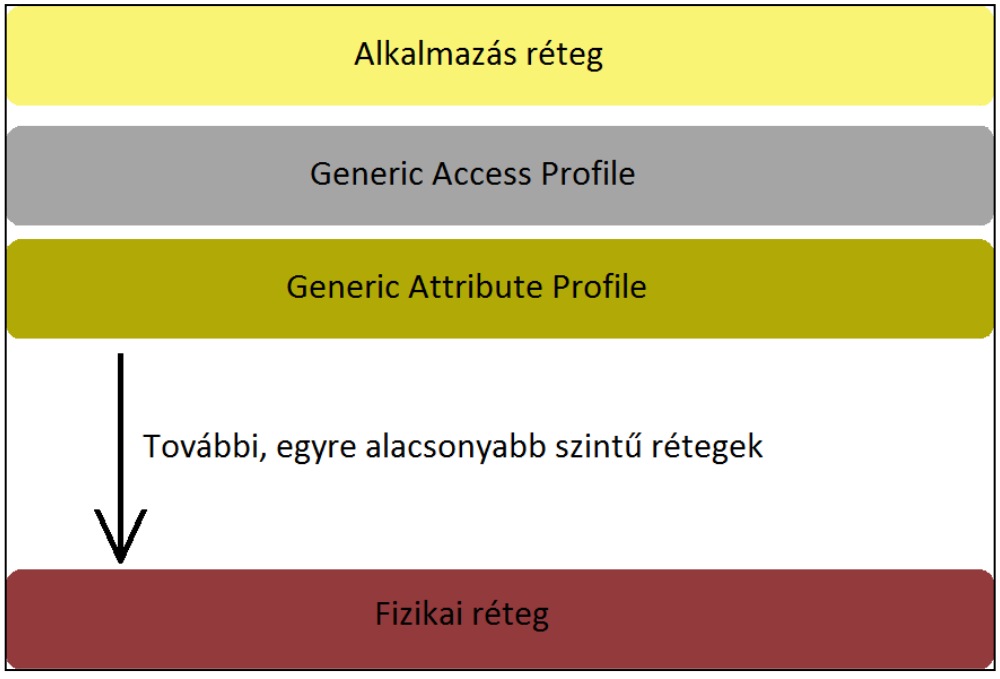
Technikai tulajd.	Klasszikus BT	Bluetooth Smart	WiFi
Adatsug. távolsága	100 m	< 100 m	< 300 m
Moduláció	GFSK	GFSK	OFDM
Fogyasztás	< 10 mW	< 50 mW	> 200mW
Adatsebesség	1 - 2 Mbit/s	1 Mbit/s	11, 54, 150+ Mbit/s
Felhasználás	Játék, PC...	Fitnesz, headset...	Internet elérés...

2.1. táblázat: Klasszikus BT, BT Smart és WiFi összehasonlítás [2] , [12] , [14]

A táblázat alapján megfigyelhető a különbség a klasszikus és a Smart Bluetooth között. Míg az előző nagyobb távolságban, gyorsabb adatátvitelre képes ugyan, addig az utóbbi jelentősen kisebb áramot vesz fel, ezáltal, sokkal kisebb a fogyasztása is. Érdekes az összehasonlítás a WiFi-vel kapcsolatban is. Egy Smart modul képes akár hónapokig működni egy gombelemről, egy WiFi modul viszont körülbelül 15-ször nagyobb áramfelvételével erre képtelen [2] . Fontos még megemlíteni, hogy a Smart kevesebb mint 2µA áramot vesz fel készenléti állapotban, így tehát használaton kívül szinte elhanyagolható a fogyasztása.

2.1.2 Profilok

A Bluetooth Smart tehát egy alacsony fogyasztású technológia, melynek használatával egyszerűen kezelhetünk szenzorokat, különböző perifériákat. A kommunikáció, a klasszikus Bluetooth eszközhöz hasonlóan, profilokra épül. Egy profil definiál egy adatstruktúrát, mely meghatározza, hogy a periféria milyen módon szolgáltatja (láttatja) az adatot, amelyet küld. A két általános (generic) profil a GATT és a GAP. A GATT két Bluetooth eszköz között létrehozott kapcsolat során az adatcserét határozza meg, tehát gyakorlatilag a legfelsőbb adatréteg, míg a GAP a kapcsolat felvételét, és a Broadcast/Advertise folyamatát, írja le, vagyis a legfelsőbb kapcsolati réteg. Ezek együttesen az alsóbb protokoll rétegek működését is meghatározzák [6] .



2.2. ábra: Bluetooth Smart rétegszerkezet (Stack) [3]

2.1.2.1 GAP

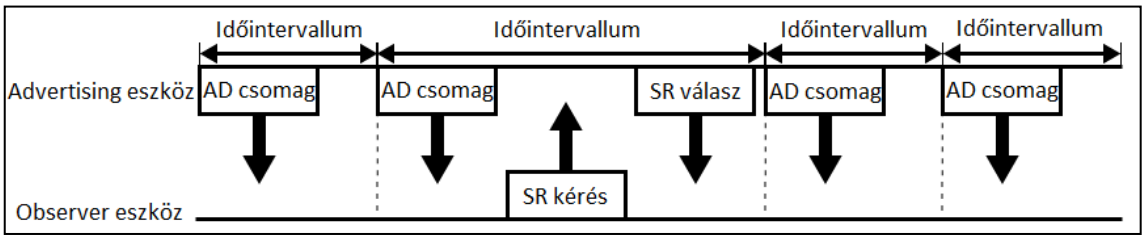
A GAP (Generic Access Profile) tehát biztosítja az egyes Smart eszközök együttműködését. Egy keretrendszert ad, amit minden Smart eszköznek követni kell ahhoz, hogy egymást megtalálják, adatokat sugározzanak, biztonságos kapcsolatokat létesítsenek sztenderd, univerzális módon. A GAP négy alapvető állapotot definiál az eszközöknek. Az egyik a Peripherial (periféria), amelyben általában kis méretű, kis fogyasztású, energiatakarékos eszközök működnek, mint például egy pulzusmérő. A Central (központi) állapotú egység egy robosztusabb, nagyobb számítási teljesítményű és több memóriával rendelkező eszköz, például mobiltelefon vagy táblagép. A Central tipikusan használja a Peripherial által szolgáltatott adatot. Ezeken kívül Broadcaster (adat sugárzó) illetve Observer (megfigyelő) lehet a BT egység .

2.1.2.2 Advertising és Broadcasting

Egy Bluetooth Smart eszköz esetében alapvetően két kommunikációs módszer határozható meg. Az egyik során a modul adatcsomagokat sugároz ki a körülötte lévő összes eszköznek. Erre egy tetszőleges fogadó eszköz vagy reagálni tud, vagy pedig kérést küldeni az információt sugárzó perifériához további adatokért. A kommunikáció második módja a tényleges kapcsolat kialakítása, mely során a periféria eszköz és a központi egység is küld adatot egymásnak. Az adatok kisugárzása, vagyis az első kommunikációs forma két módon valósulhat meg, melyek abban különböznek egymástól, hogy a modul saját adatokat küld, vagy más eszköztől, például egy mikrokontrollertől, kapott adatokat sugároz ki. Előbbi esetében Advertising (hirdető), utóbbi esetében pedig a már említett Broadcasting (adat szóró) üzemmódról beszélünk [5] .

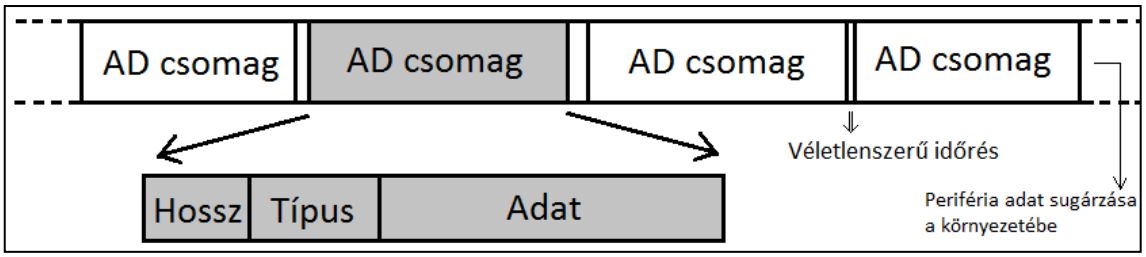
Advertising esetén az eszköz saját maga, periodikusan, adatcsomagokat sugároz ki a körülötte lévő eszközöknek. Ez az elsődleges adatküldési elv az úgynevezett Advertising Data (AD). Ezt kötelezően betartja minden ilyen állapotban lévő Smart eszköz. A meghatározott időintervallum különböző hosszúságúra állítható 20 ms-tól 10.24 s-ig, 0.625 ms-os lépésenként. Ha hosszabb, akkor kisebb az eszköz fogyasztása, viszont kevésbé érzékeny a kérésekre. A Smart robosztusságát azonban jól tükrözi, hogy egyes intervallumok közé véletlenszerű, 0-10 ms közötti késleltetéseket tesz annak érdekében, hogy az eszközt biztosan megtalálja a központi egység. Az Advertising módban sugárzó periféria megtalálása kritikus feladat, ezért sugározza egyszerre három csatornán a csomagokat, így ha esetleg az egyik csatorna interferál, például egy WiFi

eszközzel, akkor a másik kettőn még észlelhető a modul. A másodlagos, opcionális módszer pedig a Scan Response (SR), mellyel ebben az üzemmódban adatokat kaphatunk a perifériától. Ha egy másik hallgatózó (központi) egységnek a fogadotton kívül további adatokra van szüksége, akkor még az adott intervallumon belül, küld egy SR kérést a perifériának, amely az SR adatcsomaggal válaszol erre. Ehhez viszont az AD csomag kisugárzása után a perifériának vételi módba (RX) kell kerülnie, hogy fogadni tudja a kérést.



2.3. ábra: Értesítési folyamat [5]

Fontos azonban megjegyezni, hogy Scan Response esetén sem létesít a két eszköz tényleges kapcsolatot, és az adó eszköz továbbra is adatszóró állomásként működik. Ennek egyik hatalmas előnye, hogy a fejlesztőnek csak az Advertising rétegig (azaz a GAP-ig) kell implementálnia a Bluetooth Stack-et, az alacsonyabb rétegekkel nem kell foglalkoznia, mert az AD csomagokban elhelyezheti a küldeni kívánt adatokat, és ezáltal a fogadó oldalon is elég csak a periodikusan kapott üzenetet kibontani. Ilyen módon elkerülhető a tényleges kapcsolat kialakítása. A periféria által kisugárzott adatcsomagban információk vannak az eszközről, illetve magáról az adatcsomagról, felépítését pedig a GAP határozza meg. Egy-egy adatcsomag egy adatstruktúrának felel meg, mely három részegységből áll.

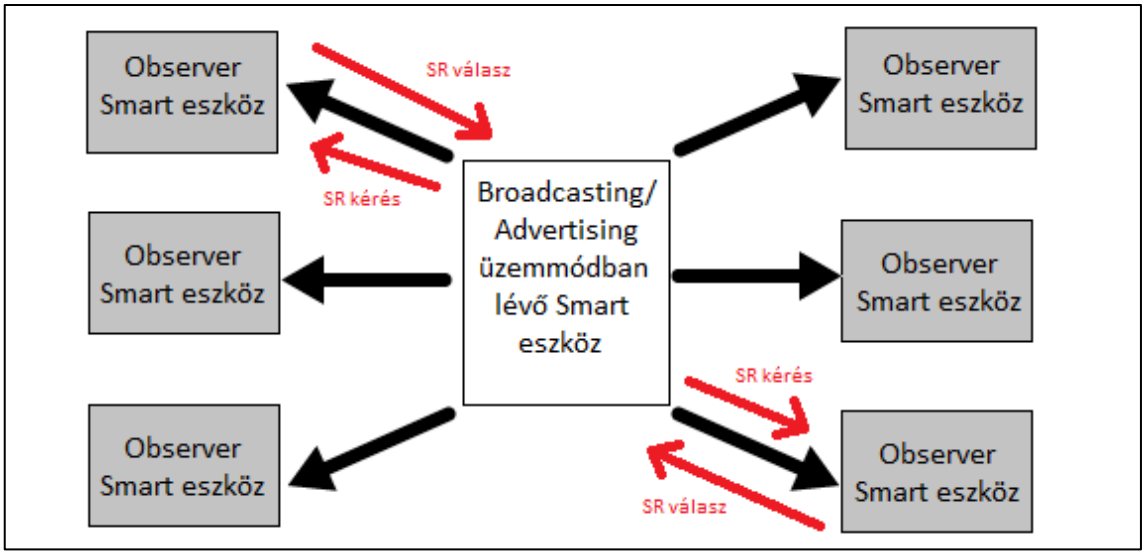


2.4. ábra: Advertising csomag adatstruktúrája

A struktúra első része a hosszt, a második a struktúra típusát, a harmadik pedig a tényleges adatot tartalmazza. A típust erősen meghatározza a GAP. Tartalma lehet a Control Flag-ek, eszköz neve, UUID (Universally Unique Identifier), szerviz adat vagy például a jelerősség. A csomagok feldolgozását a vevő eszköz oldalán kell megoldani,

például a fogadott adat feldarabolásával. A struktúrának köszönhetően lehetőségünk van azonosítani az eszközöket, szűrni azokat bizonyos tulajdonságok, például az eszköz neve alapján.

Az Advertising üzemmódtól a Broadcasting annyiban különbözik, hogy ekkor a modul már nem saját maga sugároz ki saját adatcsomagokat, hanem egy másik eszközhöz van csatlakozva, amely által szolgáltatott adatokat közvetítőként sugározza a körülötte lévő eszközöknek adóállomásként üzemelve. Ilyen módon vizsgálhatjuk meg például egy helyiség hőmérsékletét, vagy a légnyomást szenzorokkal, melyek különböző helyekről, folyamatosan szolgáltatják a pillanatnyi mért adatot.



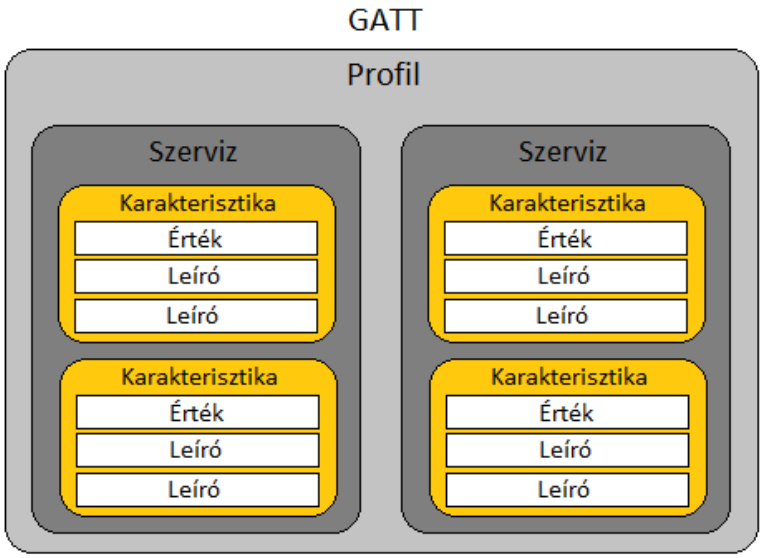
2.5. ábra: Broadcasting / Advertising topológia

Ahhoz, hogy fogadni tudjuk tehát a periodikusan kisugárzott adatokat, szükség van egy olyan Smart eszközre, amely Observer üzemmódban van. Ez azt jelenti, hogy az egység "hallgatózik" a környezetében lévő adatsugárzó eszközök után. Ez az elrendezés tehát alkalmas lehet arra, ha több szenzorunk is van, és azoktól folyamatosan várunk meghatározott információkat. A képen látható topológia alapján, több Observer állapotú Smart eszköz is tudja a szórt adatot fogadni egy időben. Természetesen ebben az esetben is van lehetőség például az UUID-re való szűrésre annak érdekében, hogy csak meghatározott Broadcasting/Advertising állapotban lévő Smart eszközöktől vehessen az Observer egység adatot.

2.1.2.3 GATT

Advertising vagy Broadcasting módban tehát az eszközök nem létesítenek tényleges kapcsolatot, csak a vevő eszköz érzékeli az adó által küldött információt, és dolgozza fel azt, vagyis az adó gyakorlatilag nem is tud az Observer eszközről, amely az általa kiadott adatcsomagok után hallgatózik. Az Advertising módra azonban szükség van, hogy megtaláljuk a modult. Amint két Smart eszköz között kialakul a pont-pont kapcsolat, az egyik működését a Peripherial a másikét pedig a Central mód határozza meg, és csak a Central kezdeményezheti a kapcsolat kialakítását. A kapcsolat felépítése előtt tehát a Peripherial eszköz sugározza a kapcsolati státuszát (az AD csomagok segítségével, Advertising módban), majd a Central ezt megtalálva (Observer módban), elindítja a kapcsolódási folyamatot. Ha már egyszer kapcsolódtak egymáshoz, akkor beállítható az úgynevezett Bond (kötelék), mely következtében a Peripherial eszköz csak a Central párjához tud ezentúl csatlakozni. Ez természetesen meg is szüntethető később. Ezt a folyamatot párosításnak is nevezik. A kulcskezeléssel kiegészítve segítséget nyújt, hogy más eszköz ne vehesse a Peripherial által szolgáltatott adatot.

A klasszikus Bluetooth-hoz hasonlóan a Smart is a profilkoncepciót alkalmazza, a kapcsolat felépítéséhez, azonban a Smart esetében a profilok úgynevezett szervizek gyűjteményei. Minden szerviz a GATT (Generic Attribute Profile) köré épül, mely meghatározza az attribútumok elérhetőségét, és azt, hogy milyen módon láthatja egy eszköz egy másik által küldött adatot. Definiálja tehát a két eszköz közti adatcserét. A szervizek karakterisztikákat foglalnak magukba. A karakterisztika tartalmazhat például adatokat és különböző leírókat a mért adatról. Amelyik eszköz tehát egy karakterisztika adatát szolgáltatja, az a rendszerben Server (szerver) szerepben működik, míg amelyik eszköz pedig a karakterisztika adatát fogadja Cliens (kliens) szerepben van. Példaként említhető egy androidos telefonból és egy Bluetooth Smart alapú szenzorból álló rendszer. A telefon Central, a szenzor pedig Peripherial állapotban van. Szükséges, hogy Central és Peripherial állapotban is legyen egy-egy egység, ellenkező esetben ugyanis csak akkor kommunikálhat két eszköz, ha mindkettő Central állapotú. Ha a kapcsolat felépült, a két eszköz között megkezdődik a GATT alapú adatcsere. Az adat típusától függően az egyik mindig Server, míg a másik Cliens szerepben működik. Ha például a szenzor mérési adatot küldd, akkor Server, a telefon pedig Cliens, amikor pedig frissíteni/konfigurálni akarjuk a szenzort a telefonnal, a szerepek felcserélődnek.



2.6. ábra: GATT profil

Egy profil tehát attribútumokból áll. Egyes attribútumok tartalmazhatnak továbbiakat is. Attribútum a szerviz, a karakterisztika és annak további elemei is, mint a fogadott adat értéke, és a különböző leírók. Minden attribútum rendelkezik saját UUID-vel (azonosítóval), Handle-el (speciális index, amely az adott attribútumra mutat, ezen keresztül érhető el az attribútum), illetve saját Value-val (az attribútum saját értéke). Az attribútumoknak létezhetnek különböző tulajdonságai is. Ezeket Property-nek nevezzük, és meghatározzák, hogy milyen művelet végezhető el az adott attribútumon: lehet-e írni, olvasni, vagy úgy írni, hogy arra visszajelzést ne kapjunk, kapjunk-e értesítést az attribútum változásáról. Egy profil több szervizt és egy szerviz tetszőleges számú karakterisztikát tartalmazhat. A hőmérséklet érzékelő szervize például tartalmazhat egy karakterisztikát, mely a mért hőmérséklet értékét adja meg, de tartalmazhat egy másikat is, amely például a mért hőmérséklet mértékegységét határozza meg. A hőmérséklet érzékelő szenzor mértékegységét átírva más egységben kaphatjuk az információt. Az alábbi táblázatban felsorolásra kerültek az attribútumok lehetséges tulajdonságai.

Tulajdonság	Érték	Leírás
Extended Property(1)	0 'b10000000	Additional property available.
Authenticated Write(1)	0 'b01000000	Write characteristic with authentication from client to server.
Indicate	0 'b00100000	Indicate value of characteristic with acknowledgment from server to client.
Notify	0 'b00010000	Notify value of characteristic without acknowledgment from server to client.
Write	0 'b00001000	Write value of characteristic with acknowledgment from client to server.
Write Without Response	0 'b00000100	Write value of characteristic without acknowledgment from client to server.
Read	0 'b00000010	Read value of characteristic. Value is sent from server to client.
Broadcast(1)	0 'b00000001	Broadcast value of characteristic.

2.2. táblázat: Attribútumok tulajdonságai [7]

Minden szerviz és karakterisztikái rendelkeznek egy azonosítóval, az UUID-vel. Ennek két típusa van. A rövidebb 16 bites verziót az előre definiált profiloknál használják, valójában viszont ezek is a hosszabb 128 bites UUID formátumot képviselik, azzal az eltéréssel, hogy a maradék bitek minden Bluetooth SIG által beépített profilnál megegyeznek. Különbség csak akkor van, ha a fejlesztő definiál egy új profilt, mert ebben az esetben meg kell adnia a teljes 128 bites UUID-t. A 16 bites változattal tudjuk tehát a beépített profilok szervizeit megkülönböztetni. A Health Thermometer Service 16 bites ID-ja például a 0x1809, a teljes UUID pedig 00001809-0000-1000-8000-00805F9B34FB. Ez lehetőséget ad arra, hogy szűrni tudjon a fejlesztő az egyes profilok között, ezáltal pedig választhasson, hogy konkrétan melyik szenzor által sugárzott adatot fogadja. Ha tehát egy szobában elhelyezünk hőmérséklet érzékelő szenzoros Smart eszközöket, melyek valamilyen Thermo profilt használnak, akkor ismerve ezen szervizek UUID-jét, ki tudjuk szűrni az általuk szolgáltatott adatot, más eszközök által sugárzott adatok közül, illetve arra is lehetőség van, hogy egy adott profil szervizei vagy karakterisztikái közül választhasson a fejlesztő.

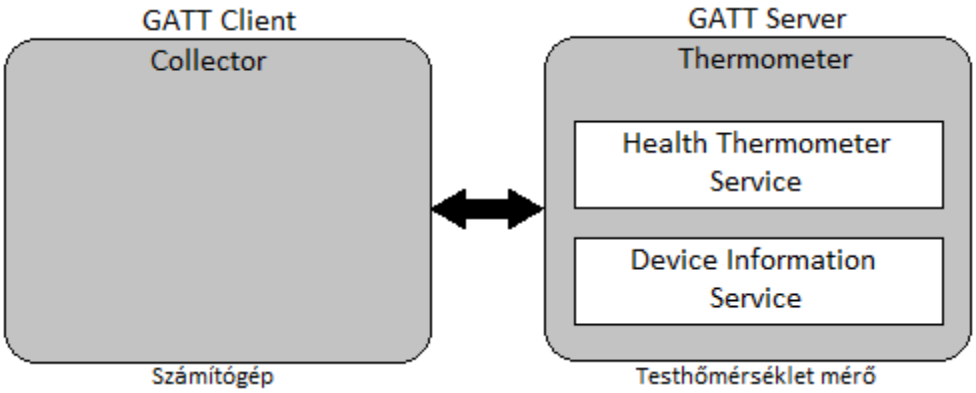
2.1.2.4 Beépített profilok

A Bluetooth SIG tehát előre definiál nekünk profilokat, melyek más-más helyzetekben lehetnek előnyösek. Ezek tartalmaznak bizonyos szervizeket a hozzájuk tartozó UUID-vel, és ezen belül bizonyos karakterisztikákat, attól függően, hogy éppen melyik profilt használjuk. Ezen profilok listája folyamatosan bővül, melyek közül felhasználhatunk a feladatunknak megfelelően bármilyen profilt. Mindegyik rendelkezik valamilyen speciális szolgáltatással, a feladatának megfelelően:

- BLP: Blood Pressure Profile. Engedélyezi egy Smart eszköznek, hogy kommunikáljon egy vérnyomásmérő szenzorral.
- FMP: Find Me Profile. Definiál egy működést, mely szerint egy gomb megnyomása egy eszközön, figyelmeztető jelzést vált ki egy periféria eszközből.
- HRP: Hearth Rate Profile. Engedélyezi egy Smart eszköz számára, hogy csatlakozzon egy pulzusmérő szenzorhoz, és kommunikáljon vele. Fitnesz területen alkalmazható.

- TIP: Time Profile. Segítségével lekérdezhető információ a dátumról, időről, időzónáról és eltolódásról, és engedélyt ad az idővel kapcsolatos funkciók irányításához.
- PXP: Proximity Profile. Engedélyezi két eszköz között a távolság meghatározást.

Egyik legjobb példa a profilokra a HTP (Health Thermometer Profile). Ez a profil két szerepet definiál: a Thermometer-t (hőmérő) és a Collector-t (adatgyűjtő). A Thermometer megméri a hőmérsékletet, a Collector pedig az adatokat kapja meg és dolgozza fel. A Thermometer a GATT szerver, míg a Collector a GATT kliens.



2.7. ábra: HTP szerkezete [3]

A Thermometer-nek a GAP Peripherial szerepét, a Collector-nak pedig a GAP Central szerepét kell betölteni, vagyis a Collector kezdeményezheti a kapcsolat felvételét a Thermometer eszközzel. A Health Thermometer Service UUID-t az AD csomagnak tartalmaznia kell, hogy a Collector azonosítani tudja a kapcsolat kialakítása előtt. A Thermometer profilja tartalmaz egy Device Name karakterisztikát, amelyen bizonyos esetekben támogatva van az írás művelet, ezáltal engedélyt adva a Collector-nak, hogy nevet adjon az eszköznek. Az eszköz információs szervize tartalmazza a gyártó nevét, a sorozatszámot, és egy rendszer azonosítót.

A Microchip által gyártott, és a szakdolgozat egyik elemeként is használt, Microchip RN4020 Bluetooth Smart modul is tartalmaz egy, a gyártó által meghatározott profilt, az MLDP-t (Microchip Low Energy Data Profile). Ez tartalmaz például egy saját MLDP szervizt. Ebben találhatóak az MLDP Data és MLDP Control karakterisztikák. Az első a notify (jelzés) és write (írás) tulajdonságokkal rendelkezik, míg a másik a read (olvasás) és a write tulajdonságokkal. Ezeket a karakterisztikákat az androidos alkalmazás megírásánál használhatja a fejlesztő, amennyiben a Microchip

saját profiljával kívánja megvalósítani a kapcsolatot a telefon és a periféria között. Az MLDP használatához szükség van az egyes szerviz és karakterisztika UUID-kra, melyek segítségével azonosíthatjuk az attribútumait. Az MLDP Data fogja tartalmazni a számunkra fontos, távolságérzékelőtől érkező adatot.

3 Specifikáció

A hordozható lopásgátló rendszer két fő részből épül fel. Az egyik rész az adatgyűjtő és adatfeldolgozó oldal, mely a mikrokontrollert, az ultrahangos szenzort, a Bluetooth Smart modult és az esetleges fejlesztéshez szükséges segédeszközöket (mint az LCD kijelző, LED-ek, USB összeköttetés a PC-vel) tartalmaz. Ezen az oldalon keletkezik az esetleges lopást jelző figyelmeztető üzenet, akár egy hangjelzés kíséretében. A rendszer másik részét az androidos készülék adja. Ez lehet egy mobiltelefon vagy táblagép is, ha megfelelő szoftveres illetve hardveres paraméterekkel rendelkezik. Ennek az oldalnak a feladata, hogy a mikrokontroller felől a Smart modulon keresztül érkező figyelmeztető jelzés hatására értesítse a tulajdonost a tárgy esetleges elmozdításáról. A lopásgátló vezeték nélküli kommunikációs részénél tényleges Bluetooth kapcsolatot kell kialakítani a GATT-ra alapozva. Debuggolásra egy LCD kijelzőt illesztettem a mikrokontroller oldalon, illetve egy soros port-ot használó PC szoftvert, a Hyper Serial Port-ot (HSP) használtam, mert nem rendelkeztem működőképes debugger egységgel. Az előbbi nagy segítséget jelent, amikor a Bluetooth modult konfiguráljuk a megfelelő működés eléréséért, ugyanis ekkor kiírathatjuk a kijelzőre az elküldött parancsunkat és az arra kapott válaszunkat is, ezzel megvizsgálva, hogy a modul megfelelő módba lett-e állítva. A fogadott és az elküldött parancsok is ASCII karakterekből állnak. A HSP használata a mikrokontrolleren konfigurált UART kommunikáció tesztelése esetén nyújt segítséget, mert nyomon követhető a rajta keresztüli adatforgalom. A Bluetooth modul állapotainak ellenőrzésére a megfelelő kimeneteire kötött LED-ek a legalkalmasabbak. Az androidos készülékre megírt program ellenőrzésére a legmegfelelőbb mód, az adott készüléken való telepítés és futtatást debug módban, amit az Android Studio támogat is.

Ha a rendszer összeállt, és a kapcsolatok kialakításra kerültek az egyes részei között, akkor működése szerint a lopásgátlónak a védendő tárgy elmozdításakor először feljegyzést kell küldenie a telefonnak, majd adott számú mérés után, ha a tárgy pozíciója még mindig eltér a kezdeti állapottól, hangjelzést kell adnia és jelzést kell küldenie az androidos eszköznek, melyre a mobiltelefon a felhasználó figyelmeztetésével és egyéb tetszőleges lépéssel reagál. Minden megvalósítandó

funkciót egy egyszerű GUI-n (Graphical User Interface) keresztül érhessen el a felhasználó, a könnyű kezelhetőség érdekében.

3.1 A rendszer összetevői

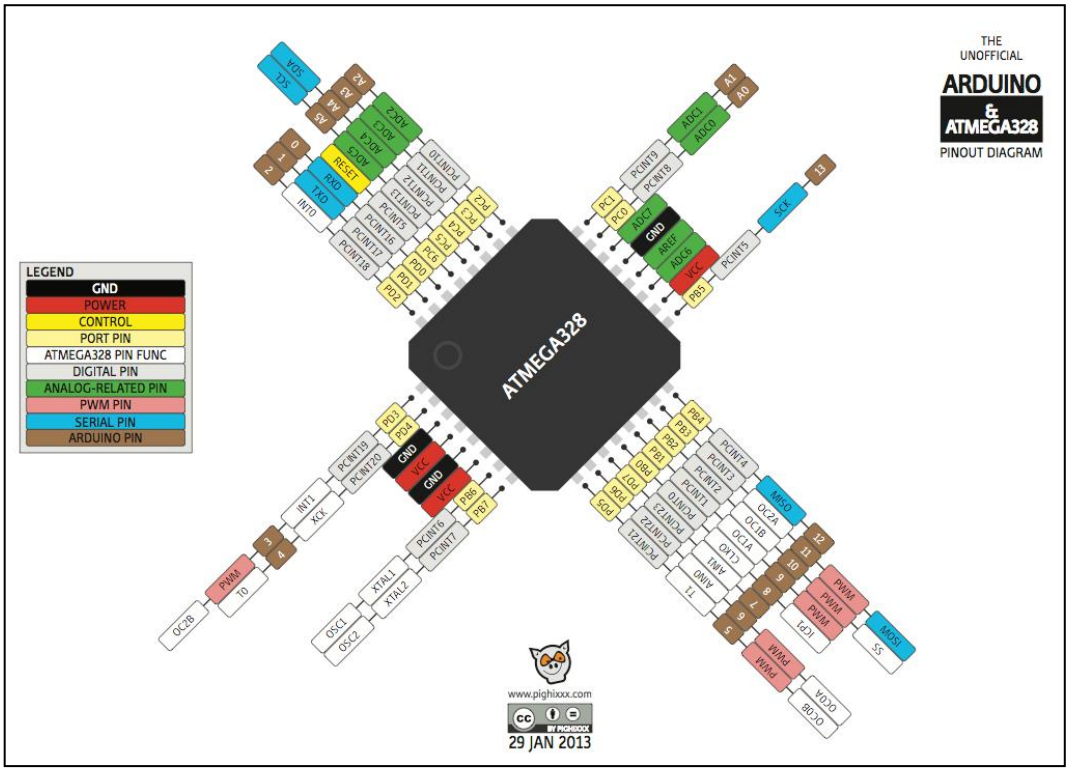
A specifikációkban foglaltak megvalósításához célszerű olyan összetevőket választani, melyek viszonylag keveset fogyasztanak. Az energiatakarékosság fontos szempont a hordozható eszközök esetében, egy Bluetooth Smart-ot tartalmazó rendszernél pedig ez elvárható, hiszen az a cél egy ilyen modul használatakor, hogy a kész eszköz a lehető legtovább működjön mielőtt elemet kell cserélni, vagy újra kell tölteni az akkumulátort. Másik szempont a részegységek kiválasztásánál, hogy a méretük lehetőleg kicsi legyen. Ez azért fontos, mert ebben az esetben az egész szerkezet elfér kis helyen, és kevésbé szembetűnő. Ilyen paraméterekkel rendelkező rendszer létrehozásához, a korábban említett szempontok figyelembe vételével, egy mikrokontroller szükséges, mint feldolgozó egység, vagy esetleg egy olyan BT modul, amelyben lévő mikrokontrollert a fejlesztő saját feladataira is tudja használni. Ehhez egy, a távolságérzékelők közül ár szempontjából legoptimálisabb, szenzort illeszthetünk a védett tárgy elmozdításának érzékeléséhez. További komponens a Bluetooth Smart modul, amely a kapcsolatot, és az adattovábbítást biztosítja, és az androidos mobiltelefon, mely a dokumentálást és a konkrét figyelmeztetést végzi el.

3.1.1 Mikrokontroller

A mikrokontroller megválasztásakor érdemes utána nézni az általa biztosított lehetőségeknek, és fel kell mérni, hogy azok megfelelőek-e a feladat megvalósításához. Ez azt jelenti, hogy ellenőriznünk kell, milyen hardver támogatásokat biztosít a mikrokontroller, mert ez kritikus lehet a lopásgátló megtervezésekor. A hardver komponensek mellett egy másik jelentős tényező lehet a választáskor a mikrokontroller támogatottsága, amely alatt a megfelelő dokumentáltságot, a megfelelő fejlesztői környezet rendelkezésre állását, illetve az internetes fórumokat, forrásokat értjük. Ezek megléte jelentősen megkönnyítheti, és meggyorsíthatja a fejlesztés folyamatát. Megfelelő választás lehet erre a célra az Atmel ATmega328p típusú mikrokontrollere, amely az egyszerű kezelhetősége, felépítése illetve kellően alapos dokumentáltsága és támogatottsága révén kedvelt a beágyazott-szoftver fejlesztők között.

3.1.1.1 Hardver támogatások

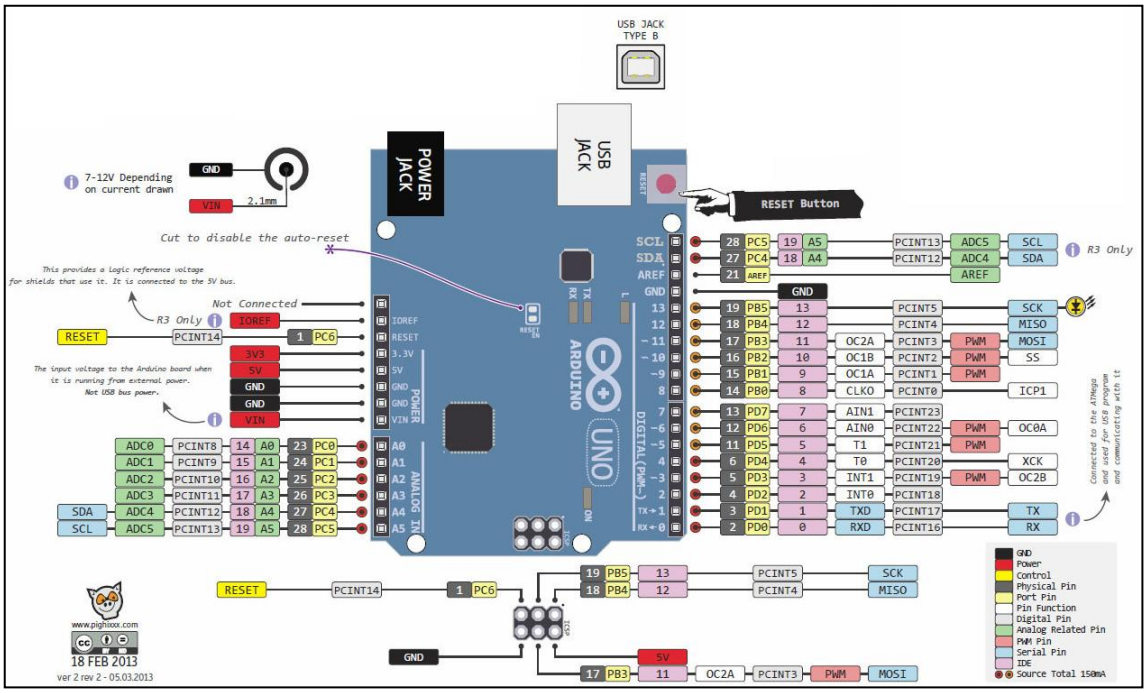
Az ATmega328p tehát egy viszonylag egyszerű felépítésű, 5 V tápellátáson működő, 8 bites RISC architektúrájú mikrokontroller. A nevéből is adódóan 32 kByte-os Flash típusú programmemóriával rendelkezik, emellett pedig egy 1 kByte-os EEPROM, illetve egy 2 kByte-os SRAM is található benne. Perifériaként lehetőséget biztosít két darab 8 bites és egy darab 16 bites Timer/Counter használatára. Rendelkezik hat darab PWM és nyolc darab ADC csatornával. Programozható USART, SPI és I2C interfész is található benne. Lehetőség van különböző interruptok (megszakítások) használatára. Ezen kívül hat darab PWM kivezetéssel együtt összesen tizenkét darab általános célú digitális programozható lábat használhat a fejlesztő.[8]



3.1. ábra: SMD ATmega328p kivezetések [15]

Manapság azonban könnyű hozzájutni úgynevezett fejlesztőkártyákhoz, melyek a felhasználót segítik olyan értelemben, hogy a mikrokontroller egyes lábait kivezetik, ezáltal jobban hozzáférhetővé téve, illetve ezeken a nyomtatott áramköri kártyákon illesztik a kontrollerhez a megfelelő ellenállásokat, kondenzátorokat, biztosítanak rajtuk USB hozzáférést és külső tápellátásnak is csatlakozót, a különböző LED-ek pedig visszajelzést adhatnak a program működéséről. Mindemellett általában külső oszcillátor is helyet kap a kártyákon, amire azért van szükség, mert a mikrokontroller belső oszcillátora nem kellően pontos. Ebben az esetben ez egy 16MHz-es kvarc oszcillátor,

tehát a rendszer erről az órajelről fog működni, ami fontos adat az időzítők és számlálók konfigurálásakor. Számos cég gyárt már ilyen fejlesztőkártyákat, melyek közül az egyik legelterjedtebb az Arduino UNO [15], [9] .



3.2. ábra: Arduino UNO fejlesztőkártya

3.1.1.2 Dokumentáció és fejlesztői környezet

Az Atmel mikrokontrollerei alaposan dokumentáltak, ezért lehet alkalmas választás. A cég honlapján keresztül elérhetőek az egyes termékek részletes adatlapja, mely sok esetben példakódot is tartalmaz az egyes fejezetekben, segítve ezzel is a fejlesztő munkáját [8] . Ezen kívül az interneten is számos lehetőség nyílik a felhasználó előtt a különböző fórumokon, és weblapokon, sok más fejlesztőnek köszönhetően. Az Atmel Studio 7 fejlesztői környezet, egyszerű, könnyen kezelhető felületet biztosít.

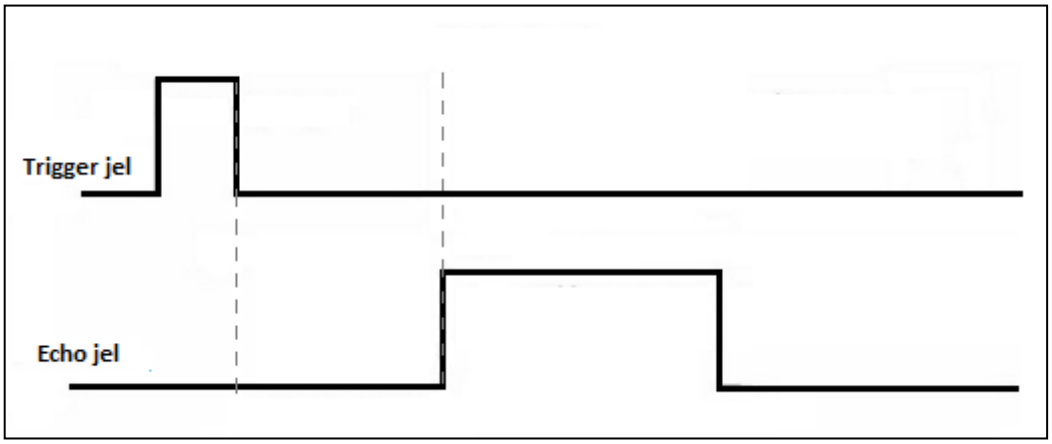
3.1.2 Ultrahang elvű távolságérzékelő

A rendszer ezen elemének a feladata, a védendő tárgy esetleges elmozdításának jelzése a mikrokontroller felé. A szenzor segítségével az elé rakott tárgyról visszavert ultrahang alapján meg tudjuk állapítani, körülbelül fél centiméteres pontossággal, hogy milyen messze van maga a tárgy a szenzortól. Az interneten számos ultrahangos távolságérzékelő található. Ezek közül megfelelő választás a HC-SR04 vagy az US-100 is. Az előbbi 5 V az utóbbi pedig 3.3 V tápfeszültségről működik, ami a későbbiekben még fontos szempont lesz a végleges hardver megtervezésénél. Én a szakdolgozat

elkészítésénél a HC-SR04-et választottam a korlátozott lehetőségek miatt, azonban az US-100, hasonló árával, jobb választás lehet a fogyasztás csökkentése miatt.

3.3. ábra: HC-SR04 ultrahangos távolságérzékelő szenzor

A szenzor működési elve tehát az ultrahangon alapszik. Négy kivezetéssel rendelkezik: Vcc, amin a tápfeszültséget kapja, GND, ami földet biztosítja, és két jelkivezetés, Trig illetve Echo. A Trig láb szolgál a szenzor triggerelésére (aktiválására), ezáltal működésbe hozva a szenzort, míg az Echo lábon kapjuk a trigger jelre érkezett választ. A működés diagramon:



3.4. ábra: HC-SR04 idődiagram [10]

A Trig lábra egy legalább 10 µs-os logikai 1 értékű impulzust kell adni, melynek hatására a szenzor 8 periódusnyi, 40 kHz frekvenciájú impulzussorozatot bocsát ki. A kiadott ultrahangos jel az útjában álló tárgyról visszaverődik, és a szenzor vevő része érzékeli azt. Ennek hatására az Echo lábat logikai 1 szintre emeli, és a tárgy távolságával arányos ideig ott is tartja, majd logikai 0-ra húzza. Az Echo impulzus szélességének, és az ultrahang terjedési sebességének ismeretében már kiszámolható a tárgy és a szenzor közti távolság. Ez persze a nyomástól, közegtől, hőmérséklettől egyaránt függhet, azonban szobahőmérsékletet feltételezve és közegellenállást

elhanyagolva a hang sebességét vehetjük körülbelül $343\frac{m}{s}$-nak. Ez a sebesség megegyezik az ultrahang által megtett út (oda-vissza) és a megtételhez szükséges idő hányadosával. A megtett út tehát a szenzor és a tárgy távolságának a kétszerese, vagyis:

$$sebesség = \frac{2*távolság}{eltelt\ idő}, \text{ átrendezve: } távolság = eltelt\ idő * \frac{sebesség}{2}$$

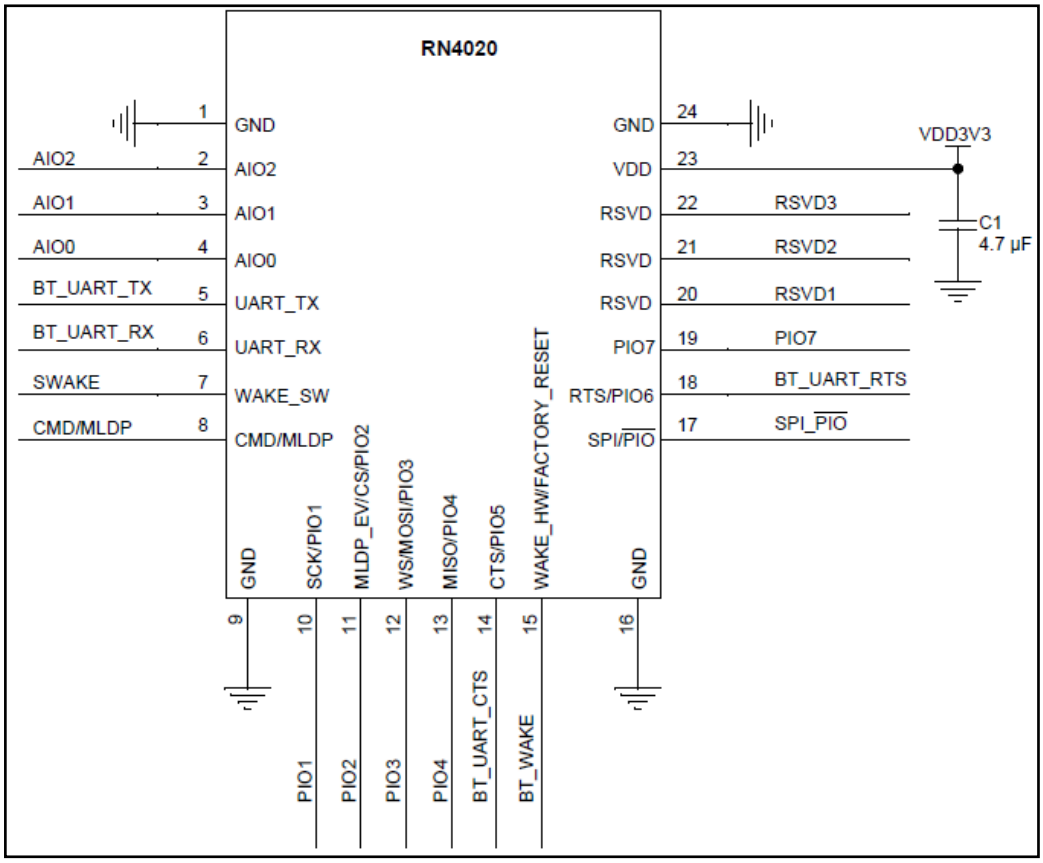
Célszerű a mértékegységeket µs-ban illetve cm-ben megadni. Az eltelt idő tehát az Echo impulzus szélessége µs-ban, míg a sebesség pedig az ultrahang sebessége, azaz 0,0343 $\frac{cm}{\mu s}$. A kettő szorzatának fele adja a távolságot cm-ben. A megoldandó feladat a projekt ezen részén tehát, az Echo impulzus szélességének mérése lesz. Ez az impulzushossz körülbelül 230 ms lehet maximálisan, mert a szenzor legfeljebb 4 méterig mér.

3.1.3 Bluetooth modul

A rendszer harmadik fő eleme a Bluetooth modul, ez szolgál ugyanis az androidos készülék és a mikrokontroller közötti adattovábbításra. Az RN4020 típusú, Microchip által gyártott eszköz alkalmas a specifikációban leírt feladat megoldására és a konzulensemtől ezt a típust kaptam. Bár ez a modul még viszonylag újnak számít és éppen ezért nem annyira részletesen dokumentált mint más chip, a döntés mögött főleg az ár/érték arány állhat, illetve az, hogy ez egy BTLE (Bluetooth Low Energy) chip, ami az energiatakarékosság szempontjából célszerű választás. Ilyen típusú Bluetooth hardver komponens található meg manapság már a legújabb telefonokban, ezért ennek használata korszerű megoldás. A gyártó cég a hivatalos honlapján megosztja a fejlesztőkkel a chip felhasználói kézikönyvét és adatlapját is, illetve az interneten is egyre több fórum foglalkozik az eszközzel [7] .

3.5. ábra: Microchip RN4020 Bluetooth Smart modul

A modul 3.3 V tápfeszültségről működik és állítható baud rate-en működő UART-on keresztül tud kommunikálni a mikrokontrollerrel. A 3.3 V-os működési feszültség miatt a rendszer összeállításánál ügyelni kell arra, hogy a választott fejlesztői kártya mikrokontrollerje 5 V tápfeszültségen üzemel (a végleges rendszerben 3.3 V-on), és a kontroller kimenetei és bemenetei is ezen a feszültség szinten működnek, ezért védenünk kell a Bluetooth egységet a túlzottan magas feszültségű jelektől. Ennek kiküszöbölésére egy egyszerű konverziós áramkört kell beiktatni a két egység közé. A Bluetooth modul különböző állapotokban képes működni, illetve támogatja a Deep Sleep üzemmódot is. Konfigurálása során ASCII karakterekből álló parancsokat tudunk küldeni a modulnak, melyekkel a számunkra megfelelő beállításokra konfigurálhatjuk. A két legfontosabb üzemmód a parancs illetve a továbbítás. Mielőtt kommunikálunk a chippel el kell döntenünk, hogy az elküldött adatot a modul továbbítsa vagy parancsként értelmezze, és feldolgozza azt. Ez az úgynevezett CMD illetve MLDP mód. CMD módban parancsokat küldhetünk, MLDP módban pedig adatot továbbíthatunk vele egy másik Bluetooth-os készüléknek. A módok között mind szoftveres, mind pedig hardveres úton választhatunk. A modul egyes kivezetései kimenetként szolgálnak és a fejlesztőnek szolgálhatnak visszajelzéssel, hogy éppen milyen állapotban van az eszköz, ezért ezeket célszerű egy-egy LED-en keresztül kivezetni. A modul rendelkezik szoftveres és hardveres "wake up"-pal is, amely Deep Sleep illetve Hibernated állapotból kelti fel az eszközt. A kettő között a különbség, hogy hibernált állapotban a modul már mindössze nA-es nagyságrendű áramot vesz fel. Az RN4020 Bluetooth Smart modul tartalmaz egy gyártó által meghatározott profilt, az MLDP-t (Microchip Low Energy Data Profile). Ez tartalmaz például egy privát MLDP szervizt. Ebben találhatóak az MLDP Data és MLDP Control karakterisztikák. Az első a notify (jelzés) és write (írás) tulajdonságokkal rendelkezik, míg a másik a read (olvasás) és a write tulajdonságokkal. Ezeket a karakterisztikákat az androidos alkalmazás megírásánál használhatja a fejlesztő, amennyiben a Microchip saját profiljával kívánja megvalósítani a kapcsolatot a telefon és a periféria között. Az MLDP használatához szükség van az egyes szerviz és karakterisztika UUID-kra, melyek segítségével azonosíthatjuk az attribútumait. Az MLDP Data fogja tartalmazni a számunkra fontos, távolságérzékelőtől érkező adatot.



3.6. ábra: RN4020 kivezetések

Ha szükségünk van rá, mert sok adatot akarunk folyamatosan átküldeni a modul segítségével, akkor használhatjuk a CTS, RTS vonalakat is "flow control"-ként az adatvesztés elkerülése érdekében. Ezeken kívül elérhető még három darab programozható analóg láb és egy SPI interfész is.

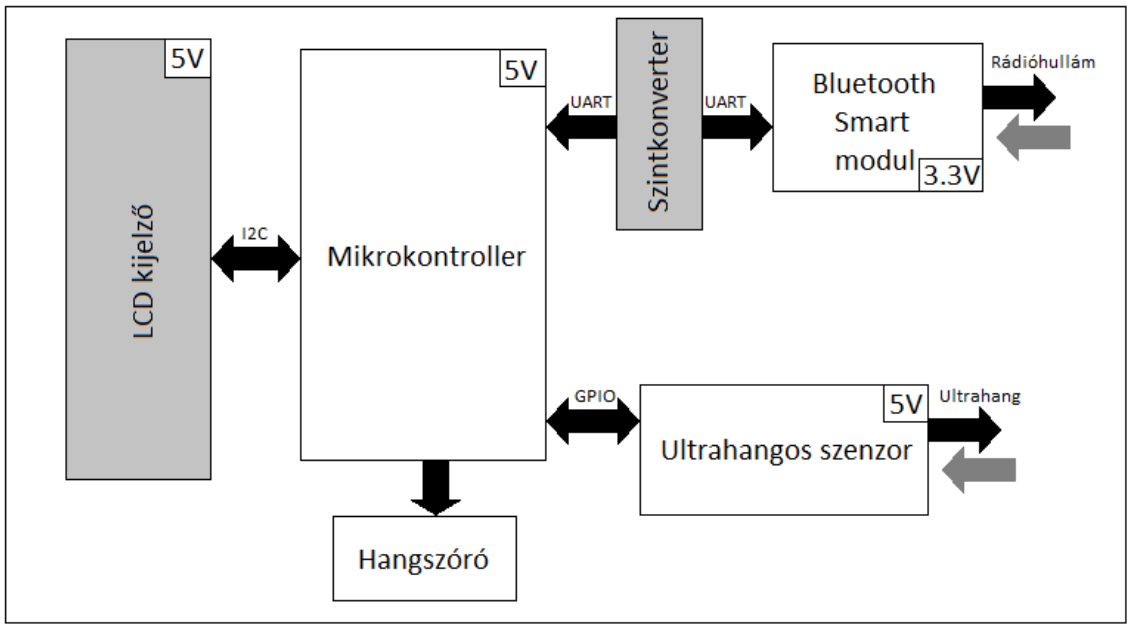
3.1.4 Androidos készülék

A mai telefonok már szinte kivétel nélkül a Bluetooth Low Energy (vagy Bluetooth Smart) technológiát alkalmazzák, ezért a Bluetooth modul kiválasztása után olyan mobiltelefont választottam, ami szintén ilyen chippel rendelkezik, mégpedig egy Samsung Galaxy S4-et. Ez a készülék kapja az értesítést a mikrokontrollertől a tárgy elmozdulásáról. A feladat megoldásának ezen részénél erre a készülékre kell egy egyszerű grafikus felületet elkészíteni a felhasználó számára, amelyen értesítést kaphat a tárgy elmozdításáról. A fejlesztéshez szükséges környezetet ebben az esetben az Android Studio biztosítja.

4 Hardver rendszerterv

A megfelelő alkatrészek kiválasztása és beszerzése után, az eszköz hardver rendszertervét kell kidolgozni, vagyis meg kell határozni a szerkezet egyes elemeinek összeköttetését. A lopásgátló központját maga a mikrokontroller alkotja, és ehhez kapcsolódnak perifériaként az egyes hardver elemek az UART, I2C és GPIO interfészeken keresztül. Ebben az esetben az UART-on a BT modul, GPIO-n a hangszóró és a szenzor, I2C-n pedig a kijelző. Ezek együttesen alkotják a lopásgátló deszkamodelljét, amelyhez a Bluetooth Smart technológia segítségével kapcsolódik az androidos készülék.

4.1 Deszkamodell



4.1. ábra: Lopásgátló deszkamodell blokkdiagram

A blokkdiagramon szürkével jelölt elemek a fejlesztést segítik elő. Ezeket a kész eszköz ténylegesen nem tartalmazza, csak visszajelzést adnak a fejlesztőnek az eszköz elkészítése során és áthidalják az alkatrész szabta megkötéseket. Az LCD kijelzőre a szakdolgozat megírása közben a debuggolás szempontjából volt szükségem, mert nem rendelkeztem debugger egységgel. A szintkonverzió a működési feszültségek különbségéből adódó problémákat oldja meg. A mikrokontroller felől a BT modul felé érkező jeleket letranszformálja. A másik irányba nincs szükség konverzióra, mert a

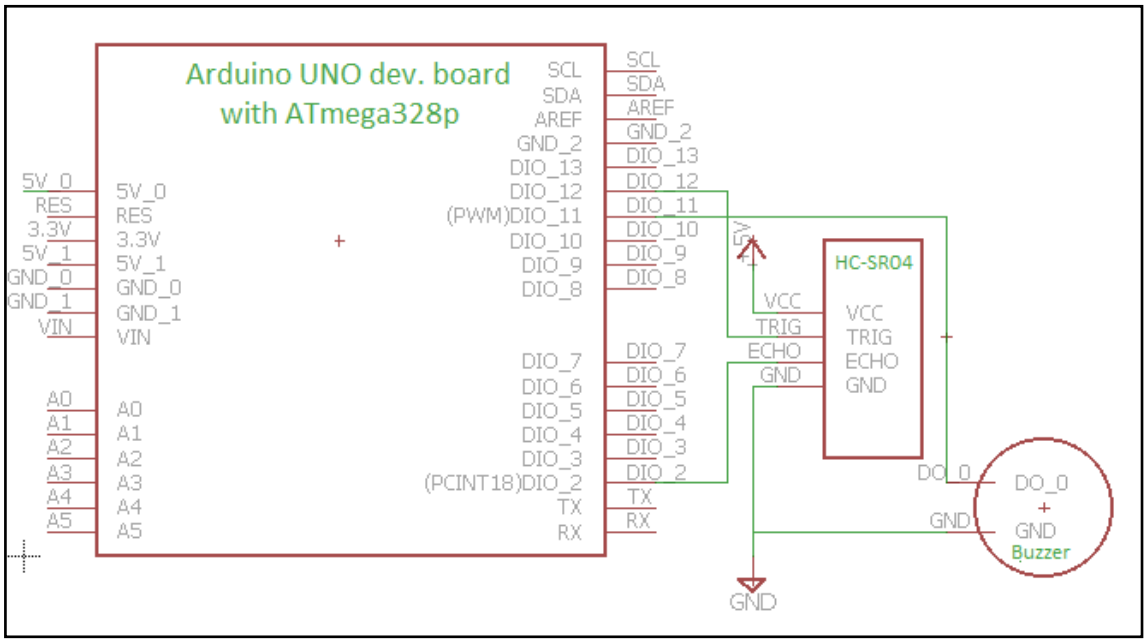
mikrokontroller 5 V-os döntési küszöbe a 3.3 V-os Bluetooth modul logikai 1 feszültségértéke alatt van. A szakdolgozat elkészítéséhez célszerű választás lehet egy 3.3V-os mikrokontroller például az Atmel-től. A feladat megoldása során ugyan 5 V tápfeszültségről működtetjük a fejlesztőkártyán lévő ATmega328p-t, azonban az 3.3 V-ról is képes üzemelni, ezért a végleges hardverben, megfelelő választás lehet. Figyelni kell viszont arra, hogy ebben az esetben legfeljebb 10 MHz-ről képes megfelelően működni. A 3.3 V-os működési feszültség esetén a Smart modul és a mikrokontroller közötti konverzió elhagyható, és csak az 5V-os szenzortól érkező Echo jeltől kell védeni a feldolgozó egységet. Amennyiben 3.3V tápfeszültségről működő, ultrahangos szenzor is kéznél van (US-100), a rendszer még tovább egyszerűsödik. Ezek fontos szempontok egy esetleges sorozatgyártás költségeinél.

A diagramon jól látható az információ áramlásának útvonala. A távolságérzékelő szenzor ultrahangos jeleket bocsát ki, melyeket visszaverődés után érzékel. Ezt periodikusan végeztetjük vele azáltal, hogy a digitális Trig bemenetét aktiválva, az Echo kimenetét meghatározott ideig tápfeszültségre emeli. Ez az időintervallum arányos a távolsággal. A mikrokontroller ezt az impulzust fogadja, és számítja ki belőle a távolságot. A tárgy elmozdítása esetén ez a kalkulált távolság megváltozik, és ha a változás értéke a megadott tűréshatáron kívül esik, a mikrokontroller jelzést küld a Bluetooth modulon keresztül az androidos mobiltelefonnak.

4.1.1 Mikrokontroller és ultrahangos szenzor

A rendszer ezen részénél a kommunikáció, egyszerűen, digitális kimenetek és bemenetek (GPIO) között játszódik le. A tápfeszültséget és a földet biztosítani kell a szenzor számára. Ha mindkét egység azonos tápfeszültségről üzemel, vagy a mikrokontroller nagyobbról mint a szenzor, akkor a táplálás egyszerűen megoldható a fejlesztői kártyáról, egyéb esetben vagy külső tápforrás, vagy a tápfeszültség feltranszformálása szükséges. A tápellátás biztosítása után a szenzor periodikus aktiválását kell megoldani. A mikrokontroller valamelyik digitális kimenete a Trig bemenetre kötve ezt biztosítja, így tetszőleges időpontokban triggerelhető a szenzor. Az aktiválás után a szenzor High szintre emeli az Echo kimenetét, majd adott idő elteltével újra földre húzza azt. A távolságot ennek az időintervallumnak a szélességéből kaphatjuk, ezért szükségszerű az Echo kimenet felfutó illetve lefutó élét detektálni. Ez

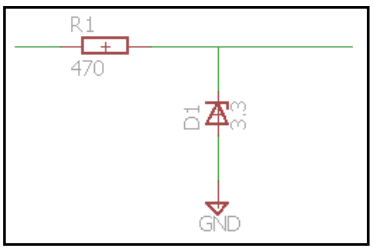
egy Pin Change Interrupt, azaz pin változási interrupt segítségével kivitelezhető. A szenzor Echo kimenetét egy adott digitális bemenetre kötve a mikrokontroller érzékelheti annak megváltozását, és egy Timer segítségével mérhetjük az impulzus szélességét. A rendszer ezen részéhez vehető még a hangszóró is, amely figyelmeztető hangjelzést adhat a tárgy elmozdításakor. Ezt PWM kimenetre érdemes kötni, hogy hardverileg kényelmesebben állíthassunk elő megfelelő frekvenciájú négyszögjelet a hangszórónak.



4.2. ábra: Fejlesztőkártya, szenzor és hangszóró kapcsolata

4.1.2 Mikrokontroller és Bluetooth modul

A mikrokontroller és a Bluetooth modul összekötésénél problémát okoz, hogy eltérő tápfeszültségről működnek. Míg a fejlesztőkártya mikrokontrollere 5 V tápfeszültségről dolgozik, addig a Bluetooth Smart modul 3.3 V-ról. Ez azért probléma, mert így a mikrokontroller logikai 1 értékű adat küldése során 5 V-ra húzza a modul lábát, melynek következtében a Bluetooth egység megsérülhet. Védelmet kell beiktatni a két eszköz közé, amelyet egy konverziós fokozat valósíthat meg.



4.3. ábra: Védelem a mikrokontroller és a Bluetooth modul között

A konverziós fokozatban egy 3.3 V-os Zener dióda korlátozza a mikrokontroller felől érkező 5 V-os jelet 3.3 V-ra. Ezzel megoldottuk a Bluetooth modul védelmét, azonban a végleges rendszernél erre nem lesz szükség, mert annak minden eleme 3.3 V-ról fog működni.

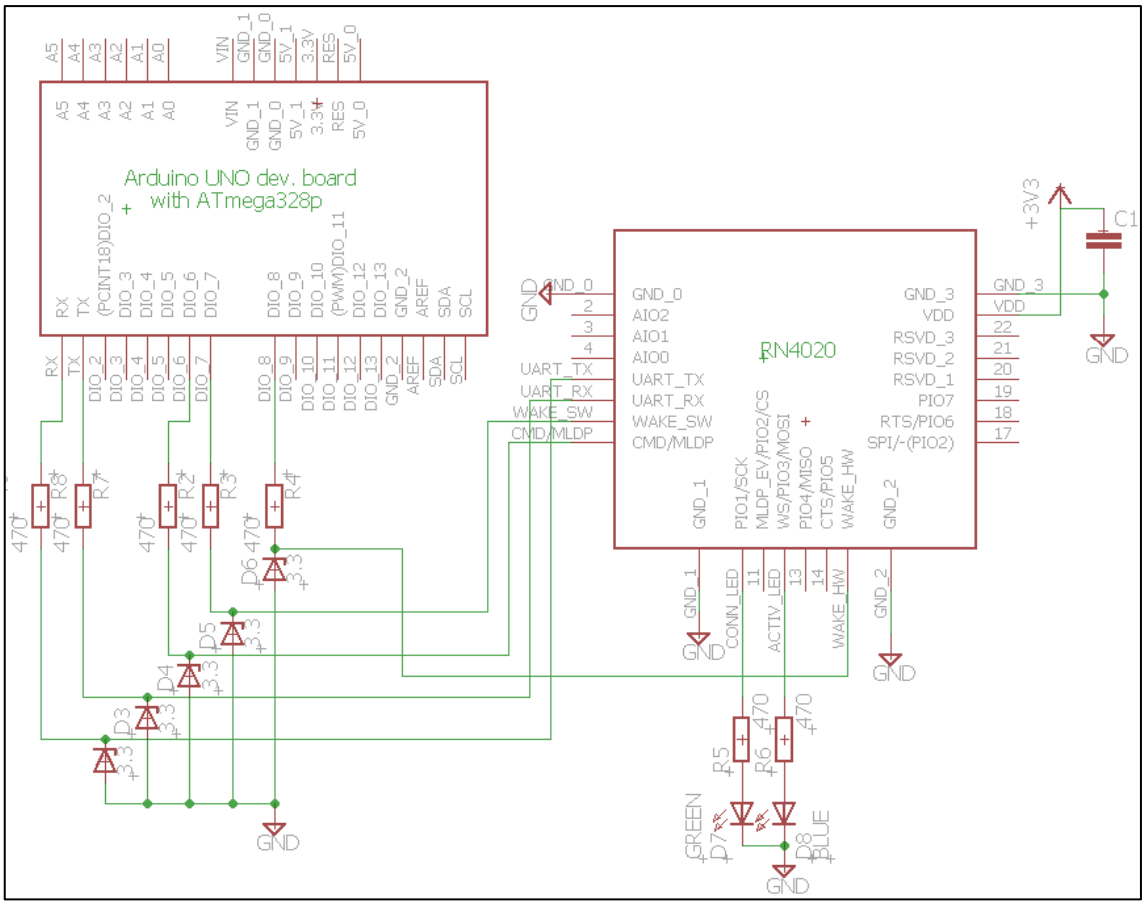
A megfelelő jelszintek, és a Bluetooth modul védelmének biztosítása után a két eszköz ezen a fokozaton keresztül már biztonságosan összeköthető. A fejlesztői kártya rendelkezik 3.3 V-os tápfeszültség kimenettel. Erről táplálható a Bluetooth modul és UART interfészen keresztül érhető el illetve programozható fel, ezért a fejlesztői kártya TX kimenetét kell a Bluetooth RX bemenetére, illetve a Bluetooth modul TX kimenetét a kontroller RX bemenetére. A feladat megvalósításához nincs szükség a "flow control" (adatfolyam szabályozás) használatára, mert a rendszer egyik eleme sem küld folyamatosan nagy mennyiségű adatot, amely az esetleges adatvesztéshez vezetne, ezért a BT modul CTS és RTS lábait most nem kell használni. Fontos azonban a modult az esetleges hibernált vagy alvó módból felébreszteni, vagy másik módba helyezni, illetve az egyes pillanatnyi állapotokat kijelezni. A táp, föld és UART vonalakon kívül tehát még öt darab kivezetésre van szükség a modul vezérléséhez. Az egyik a WAKE_SW, amely ahhoz kell, hogy Deep Sleep módból felébresszük a modult. Ez azért fontos, mert a modullal történő kommunikáció csak akkor lehetséges ha az nincs Sleep módban. Ez alól egy kivétel van, ha az UART baud rate 2400 bps-re van állítva. Ebben az esetben a modul Sleep módban is elérhető és nincs szükség a felébresztésére. Amennyiben éber állapotban van, azt egy másik kivezetésén jelzi a Bluetooth egység, mégpedig a WS-en. Erre a lábra célszerű egy LED-et kötni, ezáltal látható, hogy a modul aktív-e. Ha a WS pin értéke logikai 1, akkor aktív, ha pedig logikai 0, akkor Sleep módban van. Szükségszerűen, ha a modul hibernált állapotban van, akkor a WAKE_HW kivezetés tápfeszültségre húzása felébresztheti azt. A Sleep és Dormant mód között az a különbség, hogy az előbbiben körülbelül 4-5 μA a felvett áram, míg az utóbbiban már csak 700 nA körüli. Van egy másik funkciója is ennek a kivezetésnek. A modul feszültség alá helyezése után ha öt másodpercen belül háromszor fel-le (High-Low) húzzuk ezt a lábat, akkor az egy gyári reset-et okoz. Amennyiben a WAKE_SW logikai 1 értékű a reset bekövetkezésekor, akkor teljes gyári reset történik, ellenkező esetben pedig csak részleges, amikor is az eszköz neve, a privát szerviz illetve a szkriptek megmaradnak. Ha aktív állapotban van a Bluetooth modul, akkor választhatunk két üzemmód közül. Az egyik a CMD. Ebben az állapotban a mikrokontroller parancsokat

tud küldeni a BT egységnek, ezáltal konfigurálva azt. Az MLDP (Microchip Low energy Data Profile) mód, egy GATT alapú profilt biztosít, mely alapján a Bluetooth modul és egy másik Smart eszköz közötti kommunikáció valósítható meg az MLDP által definiált szervizekkel és karakterisztikákkal, vagyis adatcsere bonyolítható le például egy mobiltelefonnal. Ha logikai 1 az értéke ennek a kivezetésnek akkor CMD módban van a modul, ha logikai 0, akkor pedig MLDP módban. Az utolsó lényeges kivezetés pedig az SCK vagy más néven Connecion LED, amely azt jelzi, ha a modul egy másik Bluetooth egységhez csatlakozott. Ezt is célszerű láthatóvá tenni, és kivezetni egy másik LED-re. Az alábbi kilenc kivezetés szükséges tehát a modul használatához és szakdolgozatban megadott feladat megoldásához. Ezeken kívül a modul rendelkezik különböző analóg és SPI interfész kivezetésekkel is, de azokra most nincs szükség.

Pin	Név	Leírás	Funkcionalitás
1	GND	Föld	Föld
5	UART_TX	UART adat küldés	UART kimenet az RN4020-tól.
6	UART_RX	UART adat fogadás	UART bemenet az RN4020-hoz.
7	WAKE_SW	Deep Sleep ébresztés	Aktív magas felébreszti Deep Sleep-ből.
8	CMD/MLDP	Command vagy MLDP	CMD vagy MLDP mód: 1-CMD, 0-MLDP
10	SCK	Kapcsolódás jelzése	1-kapcsolódva egy eszközhöz
12	WS	Aktív állapot jelzése	1-a modul felébredt és aktív
15	WAKE_HW	Hibernált ébresztés	1-felébreszti hibernált állapotból
23	VDD	Táp	Táp

4.1. táblázat: Bluetooth modul felhasznált kivezetései

Az állapotokat jelző WS illetve SCK lábakat célszerű tehát egy-egy ellenálláson és LED-en keresztül földre kötni, hogy szemmel is látható legyen az állapot megváltozása. A Dormant (hibernált) vagy Sleep módból történő ébresztéséhez a megfelelő WAKE_SW és WAKE_HW lábakat a mikrokontroller egy-egy digitális kimenetére kell kötni, ahogy a CMD/MLDP lábat is, hogy egyszerűen kezelhetőek és változtathatóak legyenek.



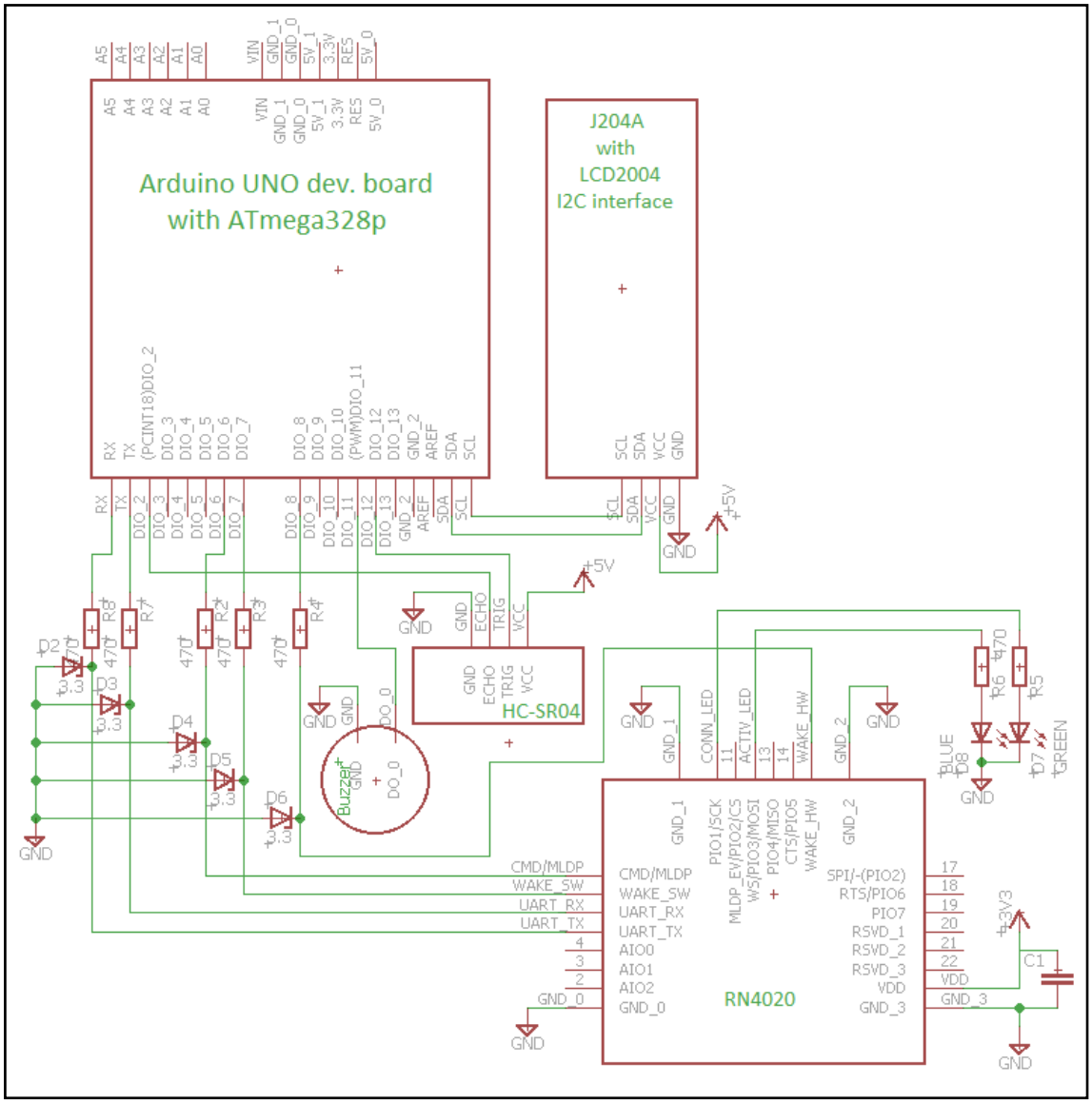
4.4. ábra: Fejlesztőkártya és a Bluetooth Smart modul kapcsolata

A deszkamodell során az 5 V tápfeszültségen működő mikrokontroller miatt kellett tehát konvertálni a jelszinteket, azonban a végleges prototípusban, amely esetleg később sorozatgyártásba kerülne, fontos a költséghatékonyság, vagyis az, hogy minél kevesebb alkatrészből tudjuk megvalósítani. Célszerűbb ezért 3.3 V-os tápfeszültségről működtetni az ATmega328p-t, így elhagyható a szintkonverter és a fogyasztás is csökken.

Arduino UNO	RN4020
RX	UART_TX
TX	UART_RX
PD6	CMD/MLDP
PD7	WAKE_SW
PB0	WAKE_HW

4.2. táblázat: Fejlesztőkártya és BT modul összerendelések

A szakdolgozatban megadott feladat elvégzése közben LCD kijelzőt is használtam debug-olás céljából, mert nem rendelkeztem debugger egységgel, a rendszer ezen eleme azonban szervesen nem a lopásgátló része, mivel az LCD kijelző sokat fogyaszt és fölösleges is a projekt szempontjából. A felhasználó nem akar semmilyen adatot nyomon követni egy LCD kijelzőn, mert azt pont az androidos mobiltelefonja vagy táblagépe helyettesíti. Minden fontos információt a rajta futó alkalmazás egy grafikus interfészen megjelenít. A deszkamodell végleges kapcsolási rajza tehát a következőképpen néz ki:



4.5. ábra: A teljes deszkamodell kapcsolási rajza

A deszkamodell tehát a fejlesztésre és a szakdolgozat elkészítésére szolgál, és segítségével általános kép kapható egy ilyen rendszer elvi működéséről, és főbb összetevőinek kapcsolódásáról, illetve ezen részek helyes konfigurálásáról. A

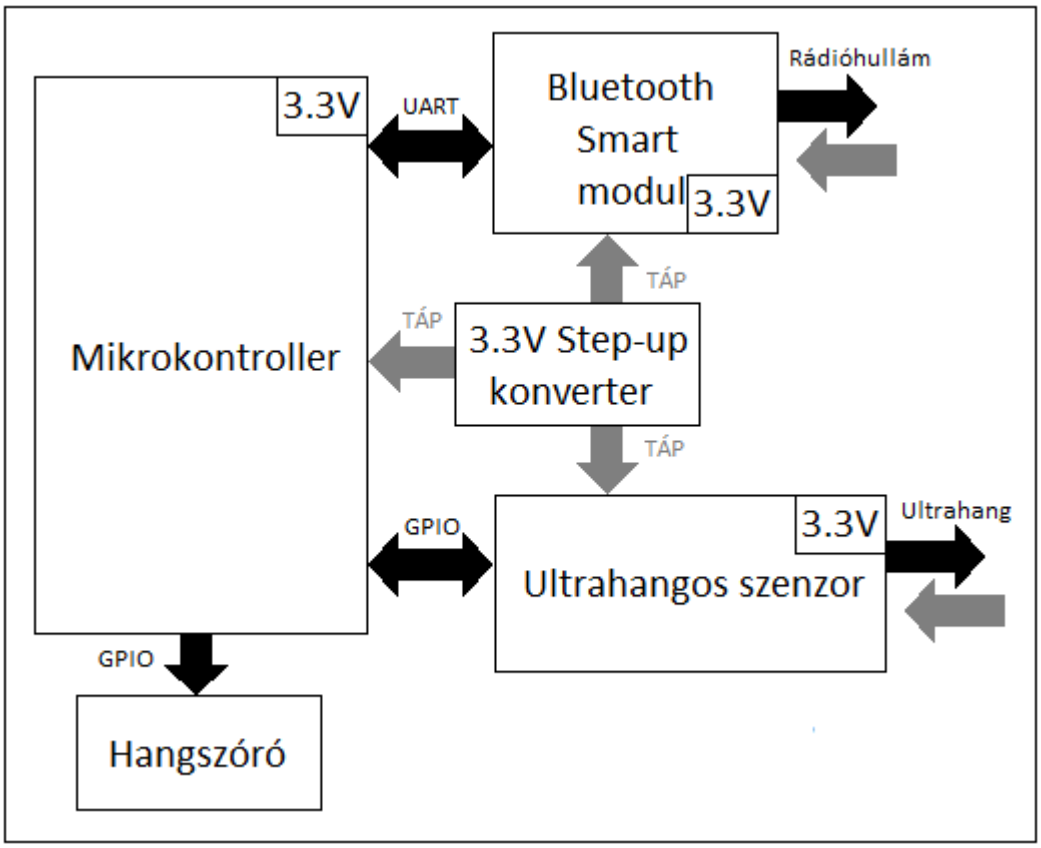
megtervezett rendszer előzetes kipróbálására szolgál. Segítségével meg lehet írni a közel végleges szoftvert. Ennek tapasztalatai alapján lehet módosítani a terven, ha az szükséges. A végleges modellnek, amennyiben lehetséges, célszerű ennél egyszerűbbnek lennie a költségek csökkentése céljából, ugyanis egy esetleges sorozatgyártásnál a felesleges alkatrészek megnövelhetik a gyártás költségét.

4.2 Végleges hardver modell

A végleges modellnél tehát a hardver rendszerterv több, már említett esetben is módosulhat. Ezeket a módosításokat meghatározza az alkatrészek költsége, fogyasztása, praktikussága. Első módosításként érdemes a mikrokontrollerből vagy esetleg egy teljes fejlesztői kártyából 3.3 V tápforrásról működő típust választani, mert ebben az esetben az említett jelkonverzió elhagyható a BT modul és a mikrokontroller között, és csak a szenzor Echo jelétől kell védeni magát a mikrokontrollert, a már ismertetett ellenállás-Zener dióda védőkapcsolással. Ez jelen esetben nem jelent gondot, mert az ATmega328p képes 3.3 V-ról is működni. A felhasznált ultrahangos szenzort is érdemes egy 3.3 V-osra lecserélni. A már említett US-100 típusú távolságérzékelő tökéletes választás, mert működése teljesen megegyezik az előző szenzoréval, használatával pedig még a mikrokontroller és a szenzor között is elhagyható a védelem. Ekkor tehát a mikrokontroller, a Bluetooth modul és a szenzor is 3.3V tápfeszültségről üzemel. Mivel a készülék hordozható, elemről működtethetőnek kell lenni, ezért kell egy tápegység. Alkalmas lehet egy DC-DC konverter erre a feladatra, amely előállítja az elem feszültségéből a szükséges 3.3V-ot. A megfelelő DC-DC konverter kiválasztása a rendszer elemei által felvett összes áram alapján történik. Mivel elemről működtetjük az eszközt célszerű egy Step-up konvertert használni, így a lopásgátló két darab AAA elemről is működhet. A fogyasztás csökkentése érdekében a mikrokontrollert alapvetően Sleep módban kell tartani, hogy az erőforrásoktól elvéve a tápellátást, kevesebbet fogyasszon. Ebből a módból kell felébreszteni adott időközönként, majd elvégezni az ultrahangos szenzorral a mérést. Ebben az esetben a szenzor tápról való lekapcsolását is meg kell oldani, hogy az se fogyasszon a Sleep mód alatt. Ez egy FET-tel megoldható, melynek gate kivezetését a mikrokontroller egyik GPIO port-jára vezetve vezérelhetjük azt. A Bluetooth modult most nem tesszük alvó állapotba, mert a telefonnal való tényleges kapcsolat kialakításához, aktívnak kell lennie, ez azonban még így sem jelent nagy problémát, ugyanis a BT modul még aktív állapotban is keveset fogyaszt. Ennek oka, hogy aktív állapot esetén ha a modul Idle, vagyis éppen nem

kommunikál UART-on keresztül akkor is csak 1.5 mA áramot vesz fel. Amint tehát jelzést kaptunk a mikrokontroller felől, hogy az ultrahanggal mért tárgytávolság megváltozott a legutóbbi mérésnél, akkor Idle módból UART aktív módba lépve továbbítjuk az üzenetet a mobiltelefonra.

A végleges rendszer elemei alapvetően megegyezhetnek a deszkamodellel, a tápegységgel való kiegészítést kivéve, ugyanis a hordozhatóság szempontjából az elengedhetetlen. A LED-ek opcionálisan megtarthatóak, az esetleges hibaelhárítás céljából, hogy könnyebben tájékozódhasson a fejlesztő az eszköz állapotáról vizuálisan is. A fejlesztői kártyát is érdemes lecserélni csak magára a mikrokontrollerre, és csak a legszükségesebb perifériákat meghagyni hozzá, mint a kvarc oszcillátor, az esetleges szűrő kondenzátorok, az ISP csatlakozás és a reset gomb. Ezáltal elég csak a ténylegesen használt lábakat kivezetni, és az USB csatlakozó illetve a hozzá illesztett felprogramozó chip is elhagyható, hiszen a mikrokontroller ISP-n keresztül is programozhatjuk ha később szükség van rá.



4.6. ábra: Végleges eszköz blokkvázlat

Hozzáadott elemként tehát a tápegységet kell megválasztanunk. Ahogy korábban említettük, egy Step-up konverter megfelelő választás lehet, a pontos típusról

azonban a rendszer által felvett teljes áram kiszámítása után dönthetünk, hiszen az meghatározza, hogy a tápegységnek mekkora áramot kell szolgáltatni ezen eszközök felé. A HC-SR04-et lecseréltük az US-100-ra. Ennek átlagos áramfelvétele körülbelül 3.5 mA [16] . A rendszer másik két eleme is 3.3V-ról működik. A maximálisan felvett áramokat ezeknél az egységeknél is meg kell határozni. A Bluetooth Smart modul Idle módban 1.5 mA-t, míg UART kommunikáció közben 16 mA-t vesz fel. Itt feltételezzük a 16 mA-t. A mikrokontroller áramfelvételét méréssel határoztam meg. Ennek során 40 mA-t kaptam eredményül. Összesen tehát a rendszer maximálisan körülbelül 60-70 mA áramot vesz fel. Ezen adatok alapján már meghatározható a szükséges tápegység [11] . A választás az L6920DB-re esett, amely 3.3V és 5V kimenetet is elő tud állítani. Ebből számunkra elegendő most a 3.3 V-os lehetőség. Ezzel a tápegységgel tehát a rendszer két AAA elemről táplálható. A végleges teljes kapcsolási rajzot lásd: A függelék.

5 Szoftver rendszerterv

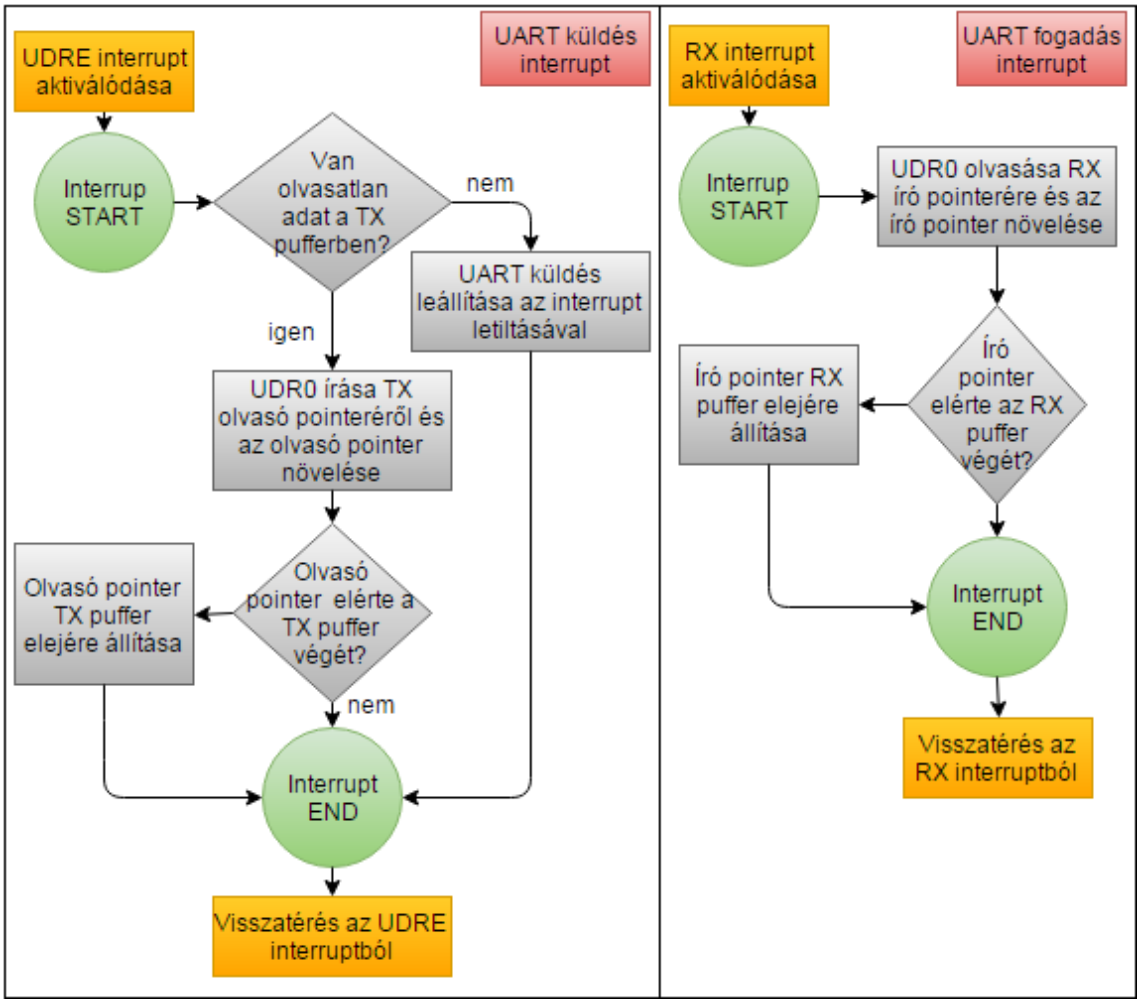
A szoftver megírása két részre bontható. Az egyik a mikrokontrolleren futó program, amely irányítja az ultrahangos szenzort és a Bluetooth modult, a másik pedig az Android-os készüléken futó alkalmazás, amely a Bluetooth modul felől érkező jelzést fogadja, és egy felhasználói interfészt biztosít. A szakdolgozat során használt deszkamodellre írt program ebben az esetben is a későbbi, végleges modellen futó program kipróbálását szolgálja, de ez a szoftver még nem a végleges program, ahogyan a deszkamodell sem a végleges hardver terv.

5.1 Mikrokontroller oldali szoftver

A szoftver a rendszer egyes részeit konfigurálja be, majd ez alapján a szakdolgozatban kiírt feladatot valósítja meg. A végleges szoftver verzió ennél valamelyest egyszerűbb lehet, és néhány helyen másképp is működhet, az alapvető gondolatmenete azonban hasonló. Természetesen rengeteg továbbfejlesztési lehetőség van. A mikrokontroller oldalon kezdve a szoftver írását több részfeladatot kell megoldani. Illeszteni kell a mikrokontrollerhez az ultrahangos távolságérzékelőt, és a Bluetooth Smart modult.

5.1.1 Mikrokontroller

A mikrokontrollerhez tehát két egység csatlakozik, a Bluetooth Smart modul és az ultrahangos szenzor. Ezen alkatrészek illesztéséhez a mikrokontroller belső perifériái közül a feladat megoldásához szükségeseket inicializálni és konfigurálni kell. Az ultrahangos szenzor illesztéséhez például szükség van Timer-re, a Bluetooth modulnál pedig az UART-ra. Három darab Timer/Counter egységgel és egy UART interfésszel rendelkezik az ATmega328p. Az UART beállításánál arra kell figyelni, hogy a Bluetooth modul gyárilag 115200-as baud rate-en kommunikál, ezért célszerű erre inicializálni. Az UART egységet a mikrokontrollernél egy ciklikus puffer támogatja az adatvesztés elkerülése miatt. A TX és az RX is külön-külön egy ilyen puffert használ, azonban pont ellentétesen. A TX puffert mindig a program írja és az interrupt során az UART adó olvassa, míg az RX puffert mindig az UART receiver írja az interrupton belül és a program olvassa ki a következő olvasatlan adatot a ciklikus pufferből.



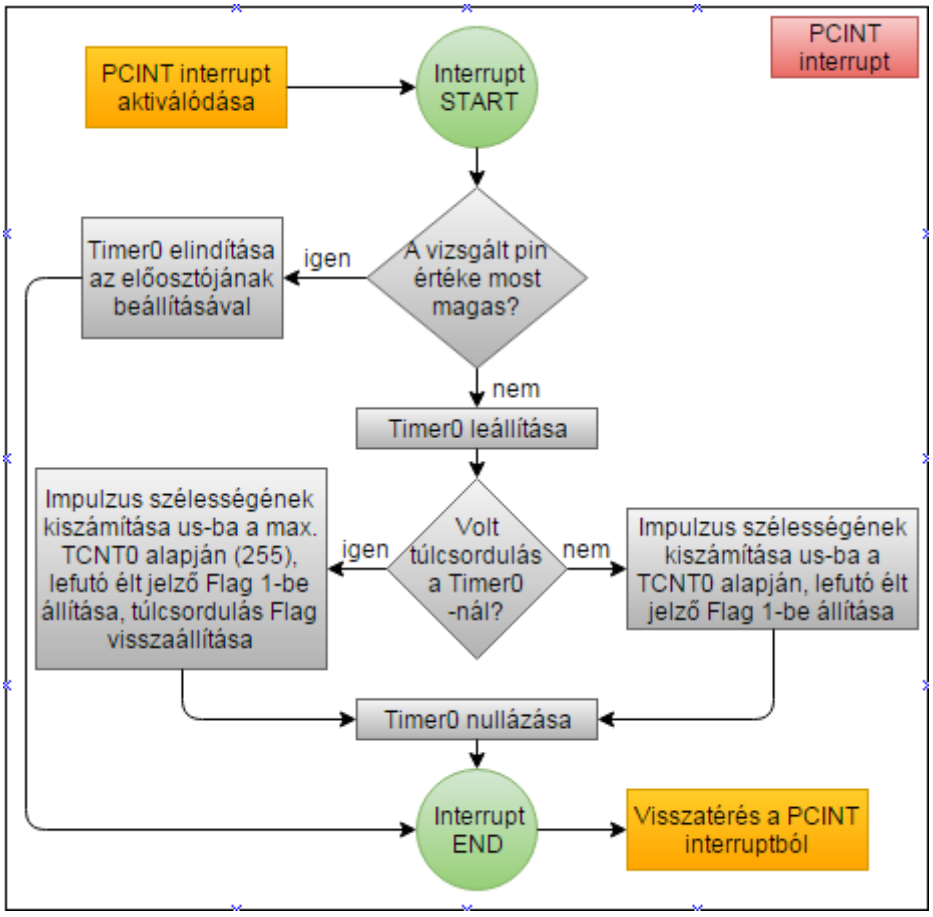
5.1. ábra: UART interruptok folyamatábrája

Az UART küldést úgy valósítottam meg, hogy feltöltöm a TX puffert az író pointer pillanatnyi helyétől kezdve, majd engedélyezem az interruptot, ami végigmegy a TX olvasó pointerével a pufferen egészen az író pointer helyéig és elküldi egyesével a karaktereket.

5.1.2 Mikrokontroller és ultrahangos szenzor

Az ultrahangos szenzor illesztésénél a mikrokontroller digitális kimeneteit kell használni. Szükségesek még a mikrokontroller Timer/Counter egységei. Ezek az idő mérése miatt fontosak. A feladat megoldása során a szenzorhoz a Timer1 és Timer0 perifériákat használtam. A Timer1 egy 16 bites számláló/időzítő. Ennek segítségével és 256-os órajel előosztással 1000 * (65535 / (16000000 / 256)) vagyis 1048 ms-ig számlálhatunk. Ehhez még a Timer-t CTC módba kell állítani és meg kell adni, hogy mekkora értéknél indítsa újra a számlálót. Én a 31250-et választottam, vagyis fél másodpercet (eddig tud elszámolni fél másodperc alatt). Ez a működés során a szenzor

Trig bementének aktiválásához megfelelő időköz. Így tehát fél másodpercenként tudjuk aktiválni a szenzort, hogy mérje meg a tárgy távolságát. A Trig bemenetet a főprogram végtelen ciklusában egy digitális kimeneti port-tal aktiválom 10 µs-ra. Ennek hatására küldi ki az ultrahangos jelet és méri meg a visszaérkezéséig eltelt időt. Ezzel az idővel arányosan Echo impulzust kell a mikrokontrollernek érzékelni, vagyis először az Echo felfutó, majd lefutó élét. Én ehhez egy Pin Change interruptot használtam, illetve egy Timer-t, ami a felfutó és lefutó él között eltelt időt méri. Az idő mérésére a Timer0 8 bites időzítő/számláló-t használtam. Azt feltételezzük, hogy a lopásgátlót a védendő tárgy mellé rakjuk, vagyis maximum fél méterre. Ekkor a szenzor képletéből visszaszámolva megkapjuk (például 50 cm-t feltételezve), hogy ez egy maximum 3 ms-os impulzust generál. 256-os előosztással a 8 bites Timer 4 ms-ig képes elszámolni. A maximálisan mérhető távolság 70 cm, a felbontás pedig 0.27 cm. Ezen eredmények alapján a 8 bites Timer elegendő a feladat végrehajtásához.



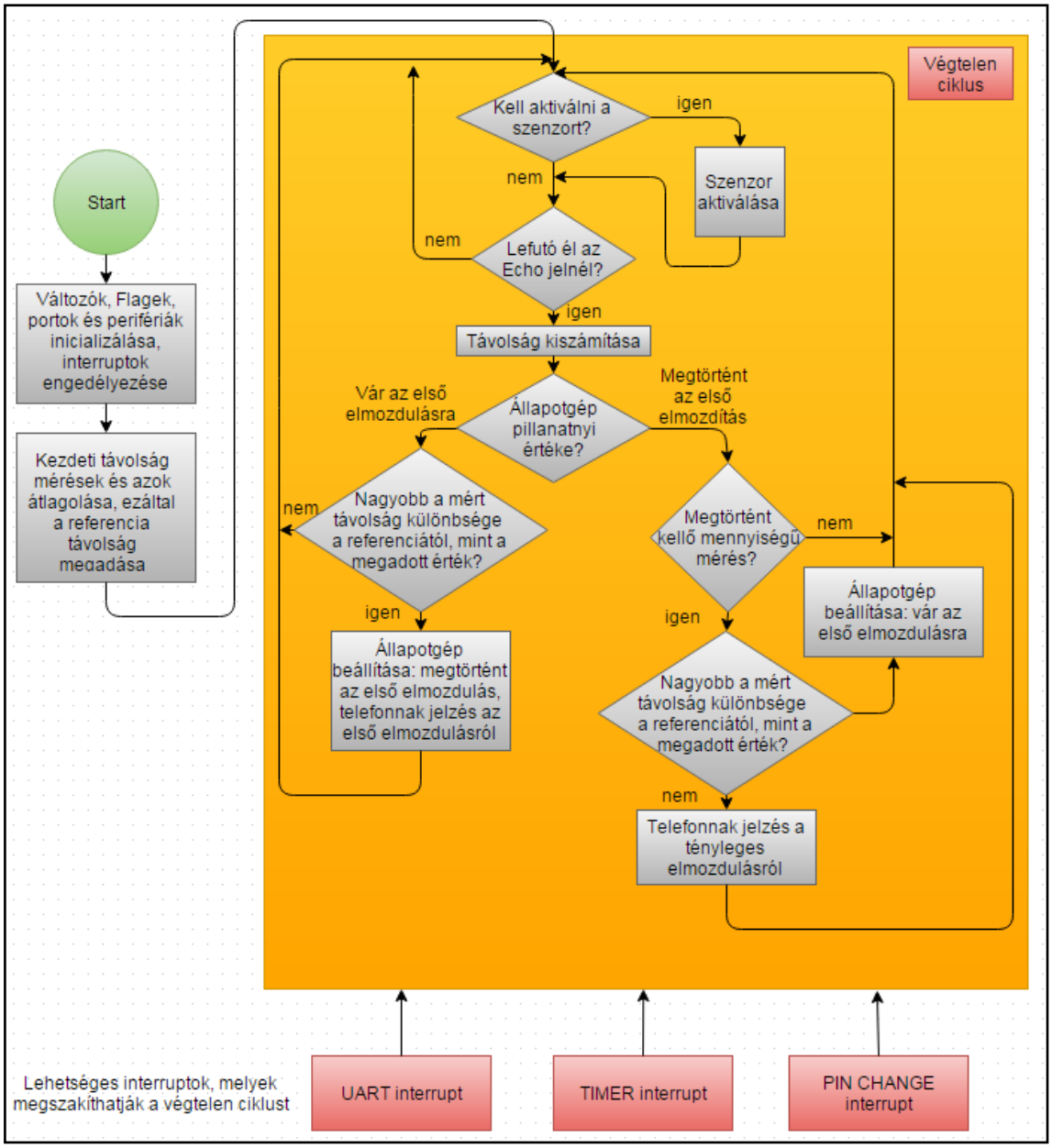
5.2. ábra: PCINT interrupt folyamatábrája

Az ultrahangos szenzor működésének lényege, hogy amikor az Echo jel felfutó éle megérkezik a mikrokontroller adott bemenetére, ott egy Pin Change interrupt

következik be. Az interrupton belül eldöntjük a pin pillanatnyi értékével, hogy az felfutó vagy lefutó él, és ha felfutó akkor elindítjuk az idő mérésére szolgáló Timer0-t, ha lefutó akkor megállítjuk, és a TCNT0 számláló segítségével kiszámítjuk µs-ban az eltelt időt az élek között. Ha túlcsordulás volt, akkor a lehető legnagyobb értéket vesszük, amit mérhetünk, mert ennél messzebb volt a tárgy a méréskor, és nem a túlcsordult értéket akarjuk megkapni. Végül nullázzuk a számlálót és jelezzük, hogy a lefutó él bekövetkezett és érvényes mérési eredményünk van.

A főprogramban vizsgáljuk, hogy mikor van új mérési eredmény, és amint kapunk egy újat abból kiszámoljuk a távolságot cm-ben. Egy előre megadott konstansban tároljuk a tűréshatárt, ami azt adja meg, hogy mennyivel térhet el a tárgy kezdeti távolságától, amit éppen mérünk. A kezdeti távolságot a főprogram végtelen ciklusa előtt megadott számú távolságmérés eredményeinek átlagából kapjuk. Ha tehát új mérési eredmény van, akkor azt összevetjük a kezdeti értékkel és ha az a tűréshatáron kívül esik, akkor egy állapotgép következő állapotába lépünk. Ez az állapotgép úgy működik, hogy egy, az előbb említett eltérési eseményre vár az első állapotában, ha ez megtörténik, akkor mér még párat (ezt is megadhatjuk előre), és ezen mérések közül az utolsóra vár. Az első állapotban, vagyis amikor legelőször történt elmozdulás, először csak egy jelzést küld a Bluetooth modul felé az UART-on. Amint a második állapotában az utolsó mérésig elértünk, megvizsgálja, hogy még mindig el van e mozdítva a tárgy. Ha igen akkor már egy "WARNING!" stringet küldd a Bluetooth felé. Ellenkező esetben, ha már nincs elmozdítva a tárgy (ez lehet akkor, ha téves volt az első jelzés), akkor visszalépteti az állapotgépet az első állapotba és újra vár az első elmozdításra.

A szakdolgozatom elkészítésénél egy kis hangszórót is illesztettem a mikrokontrollerhez, amely a "WARNING!" eseménynél periodikusan ad hangjelzést. Ehhez felhasználtam a harmadik időzítő/számlálót, vagyis a Timer2-t. A megfelelő 440Hz-es négyszögjelet az egyik PWM kimenetén állítottam elő. Ehhez arra volt szükség, hogy a hangszórót vezérlő port lábat megfelelő időközönként invertáljuk. Ezeket az időpontokat lehet a Timer2-vel meghatározni. Be kell állítani, hogy adott érték elérésekor a számláló induljon előről, és a megadott port értéke invertálódjon. Ehhez CTC módba állítottam a Timer-t, és 30-at állítottam be komparálási értéknek.



5.3. ábra: Főprogram folyamatábrája

5.1.3 Mikrokontroller és Bluetooth modul

Ez a két egység UART interfészen kommunikál egymással, illetve néhány általános digitális lábon keresztül kapcsolódnak. A Bluetooth modult használata előtt fel kell konfigurálni, hogy a megfelelő állapotba kerüljön és létre tudja hozni a kapcsolatot a mobiltelefonnal. A modult Idle állapotban tartjuk, és amint jelzés érkezik a mikrokontroller felől, továbbítjuk az adott üzenetet a telefonra, amellyel jelzünk a felhasználónak, hogy a tárgyat elmozdították. A konfigurálást először azzal kell kezdeni, hogy a BT egységet felélesztjük Sleep módból. A hardveres leírásból kiderül, hogy az RN4020 Bluetooth egység WAKE_SW lábát logikai 1-re húzva ez megtehető.

Ekkor a WS lábát a modul logikai magas szintre emeli, amit viszont már egy LED-re kapcsoltunk, vagyis ennek a LED-nek kell világítani. Ez jelzi azt, hogy a modul aktív állapotban van. Ezután a CMD/MLDP lábat kell tápra húzni, ezzel a modul belép parancs módba, és innentől kezdve, amíg ez a láb 1-es értékű az UART-on keresztül parancsokat küldhetünk a modulnak konfigurálási célokkal. Ezek a parancsok egyszerű stringek. Rengeteg lehetőséget biztosít az RN4020 a fejlesztő számára a különböző állapotokra való beállításra, és az eszköz paramétereinek konfigurálására. Ebben az esetben a rendszer a Microchip által definiált MLDP profilt fogja használni az üzenetküldéshez, ezért nincs szükség saját profil létrehozásához. A modul gyárilag 115200-as baud rate-en üzemel, ezért az UART-ot is így kell beállítani. Ha az UART-on keresztüli kommunikáció stabil, akkor megkezdődhet a parancsok küldése a modul felé. Ebben a projektben viszonylag kevés parancs elegendő a teljes beállításhoz. Először is célszerű elküldeni két parancsot a Bluetooth modulnak: az "SF,1" illetve "R,1" parancsokat. Az első részleges gyári beállításokra állítja a modult, a második pedig reseteli. Ekkor a modul 115200-as baud rate-en kommunikál. Ez tehát a kiindulási állapot. Az első fontos konfigurációs lépést egy 32 bites paranccsal tehetjük meg. Ez egy "SR, <hex32>" alakú string, ahol a 32 bites beállítást adhatjuk meg. A feladat megoldása szempontjából a következő biteket kell felhasználnunk egy egyszerű vagy kapcsolattal a végleges 32 bites érték előállításához:

Tulajdonság	32 bites érték	Leírás
Automatikus Advertise mód	0x20000000	Periféria eszköznél állítható be. Ha beállítjuk, a táp ráadása után a modul automatikusan belép az Advertising állapotba. Ha nincs beállítva, akkor csak "A" küldése után lép be.
MLDP módba lépés	0x10000000	Beállításával engedélyezzük az MLDP privát szerviz használatát.
Automatikus MLDP módba lépés	0x00000800	Ha az MLDP mód engedélyezve van, akkor ennek a bitnek a beállításával a modul automatikusan MLDP módba lép, ha kapcsolódik egy másik eszközhöz.
MLDP mód státusz nélkül	0x00000400	Ennek beállításával a modul nem küld vissza státuszt jelző üzeneteket az UART-on.

5.1. táblázat: RN4020 bittérkép beállítások

Ha ezeket az értékeket vagy kapcsolatba hozzuk, akkor eredményül a 0x30000C00-t kapunk. Ezt kell elküldeni a modulnak ahhoz, hogy az előbb felsorolt

beállításokat megvalósítsuk. A parancs tehát: "SR,30000C00". Fontos megjegyezni, hogy a modul a fogadott parancsok feldolgozásakor a stringek végét lezáró karaktert is vizsgálja. Ennek pontos leírása nem található meg rendesen a dokumentációban, azonban nekem a "\r" lezárás működött. A végleges string tehát, amit el kell küldeni, a "SR,30000C00\r". Ezzel a főbb működési funkciókat be is állítottuk. Ezen felül beállíthatjuk még például az eszköz nevét az "SN,MyDevice\r" paranccsal. Beállíthatunk másik baud rate-et is. Az "SB,3\r" parancs például 38400-ra állítja. Az eszköz dokumentációja leír minden parancsot, amire szükségünk lehet. Minden S-sel kezdődő parancsnak (vagyis setter-nek) van G-vel kezdődő, lekérdező párja (vagyis getter-e), amit azonban nem követ paraméter. A "GB\r" például lekérdezi a baud rate-et, és azt az UART-on keresztül kapjuk meg. Minden parancs során történő beállítás csak egy reset után jut érvényre. A Bluetooth inicializálását a főprogramunk kezdetekor tehát úgy végezhetjük el, hogy az alábbi kódot alkalmazzuk:

```
void BT_Init()
{
	DDRD |= (1 << DDD7) | (1 << DDD6);
	PORTD |= (1 << PORTD7);
	PORTD |= (1 << PORTD6);

	BT_SendCommand("sf,1\r");
	_delay_ms(10);
	BT_SendCommand("r,1\r");
	_delay_ms(100);

	while(UART0_IsNewDataInRxBuffer()) UART0_ReadCharacter();
	_delay_ms(500);

	BT_SendCommand("gr\r");
	_delay_ms(10);
	if(!BT_CheckResponse("30000C00\r\n"))
	{
		BT_SendCommand("sr,30000C00\r");
		_delay_ms(10);
		BT_SendCommand("r,1\r");
		_delay_ms(100);
	}
}
```

Először tehát adott portok segítségével felélesztjük Sleep módból a modult, majd CMD módba állítjuk. Ezután egy gyári reset parancsot küldünk és újraindítjuk. Az újraindítást követően kiürítjük a bemeneti pufferünket, mert státusz válaszok lehetnek benne, és lekérdezzük a modul bittérkép szerinti beállításait. Ha azok nem egyeznek a várt 0x30000C00 értékkel, akkor ezeket a beállításokat végezzük el, és ismét reseteljük.

Miután sikerült a Bluetooth megfelelően beállítani, megkezdődhet a kapcsolat felvétele az androidos készülékkel. A mobiltelefon ebben az állapotban már megtalálja a csomagokat sugárzó Smart modult. Ettől kezdve a mikrokontroller által a Bluetooth

egység felé UART-on küldött adatok továbbításra kerülnek a telefon felé az MLDP szerint. Küldéshez egyszerűen az UART TX pufferébe kell írnunk, és engedélyeznünk kell a küldés interruptját. A kommunikációhoz definiálhatunk valamilyen egyszerű protokollt. Ilyen protokoll lehet az, ha egy állapotgépen keresztül olvassuk be a mobiltelefontól érkező karaktereket, és keressük az első "<" karaktert. Ha ezt megkaptuk lép az állapotgép és a következő karaktereket már tényleges adatként értelmezi és kimenti egy tömbbe. Az utolsó állapotba akkor ér ha egy lezáró ">" karaktert kap. Ezután ismét a kezdő karakterre fog várni. A "abcd<efgh>ijkl" stringből például a számunkra fontos adat az "efgh" lesz.

5.1.4 Mikrokontrolleres szoftver továbbfejlesztési lehetőségei

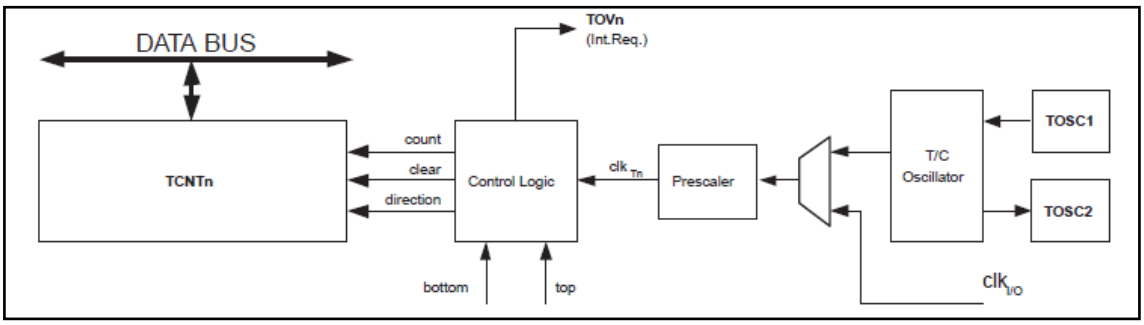
Az eddigi állapotban lévő szoftvert érdemes lehet néhány helyen módosítani, hogy minél inkább alkalmas legyen a megadott feladat elvégzésére, A fogyasztás csökkentése miatt célszerű a mikrokontrollert Sleep módban tartani, majd az egyik Timer egységének segítségével felébreszteni adott időközönként. Ez az időköz nem fontos, hogy rövid legyen, mert azt akarjuk érzékelni, hogy az egyik mérésnél még a tárgy a helyén van, a következőnél azonban megváltozott a pozíciója. Az ATmega328p több fajta Sleep módot is definiál a fejlesztő számára. A legalkalmasabb választás a Power-save mód lehet. Ebben a módban, a Timer/Counter2 hardveres időzítő fut tovább a Sleep módba lépés ellenére. Ez azért fontos, mert így a Timer segítségével felébreszthetjük alvó állapotból a mikrokontrollert. Ehhez az SMCR (Sleep Mode Control Register) megfelelő bitjeit kell beállítanunk. A regiszter alsó 4 SM2, SM1, SM0 és SE bitjét áll módunkban átírni. Ebből az SE a Sleep Enable, ami magát a Sleep módot engedélyezi, az SM0-2-vel pedig a módot választhatjuk ki.

7	6	5	4	3	2	1	0
–	–	–	–	**SM2**	**SM1**	**SM0**	**SE**
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

5.4. ábra: SMCR regiszter

A számunkra megfelelő Power-save módhoz az SM1, SM0 és SE biteket kell 1-be állítani. Ha a Sleep mód beállításra került, akkor konfigurálni kell a Timer2 egységet is. Ez az időzítő 8 bites, vagyis legfeljebb 255-ig tud elszámolni, azonban Power-save módban a fő órajel ki van kapcsolva, vagyis nekünk kell egy oszcillátort illeszteni külsőleg a Timer-hez. Ha ez nem megoldható, akkor célszerűbb az Extended Standby

alvó módot választani, mert ebben az órajel be van kapcsolva, és a Timer arról működhet. Az Extended Stadby-hoz SM0, SM1, SM2 és SE bithez is 1-et kell írni.



5.5. ábra: Timer/Counter2 órajel források és logika

Az órajel forrását természetesen meg kell határozni adott regiszterekben, például az ASSR regiszter AS2 bitjét be kell állítani ahhoz, hogy külső órajelről üzemelhessen a Timer. Ennél a modellnél Extended Standby módot feltételezhetjük, vagyis a fő működési órajelet meghagyhatjuk. A Timer túlcsordulás interruptja felébreszti a mikrokontrollert és az elvégzi a szenzor segítségével a méréseket, majd visszaállítjuk újra Sleep módba. Ezt periodikusan ismételjük. Ha a kezdeti távolságértéktől eltérőt mérünk, akkor az előző verzióhoz hasonlóan a Sleep módba kerülés előtt néhány mérés után újra megvizsgáljuk az értéket, és ha az még mindig eltér, akkor jelzünk a mikrokontroller segítségével. Ha azonban a mért érték egyezik az előzővel, akkor a mikrokontroller újra Sleep módba kerül. Ezzel a fogyasztást nagy mértékben csökkenthetjük. Kiegészítésül, kihasználva a korábban biztosított hardveres lehetőségünket, a Sleep módba lépés előtt, a megfelelő port vezérlésével lekapcsolhatjuk a szenzort a tápról, majd felélesztés után egyből visszakapcsoljuk, és egy kis idő múlva, amint a rendszer felállt, elvégezzük az említett mérést. Ez további fogyasztáscsökkenést eredményez.

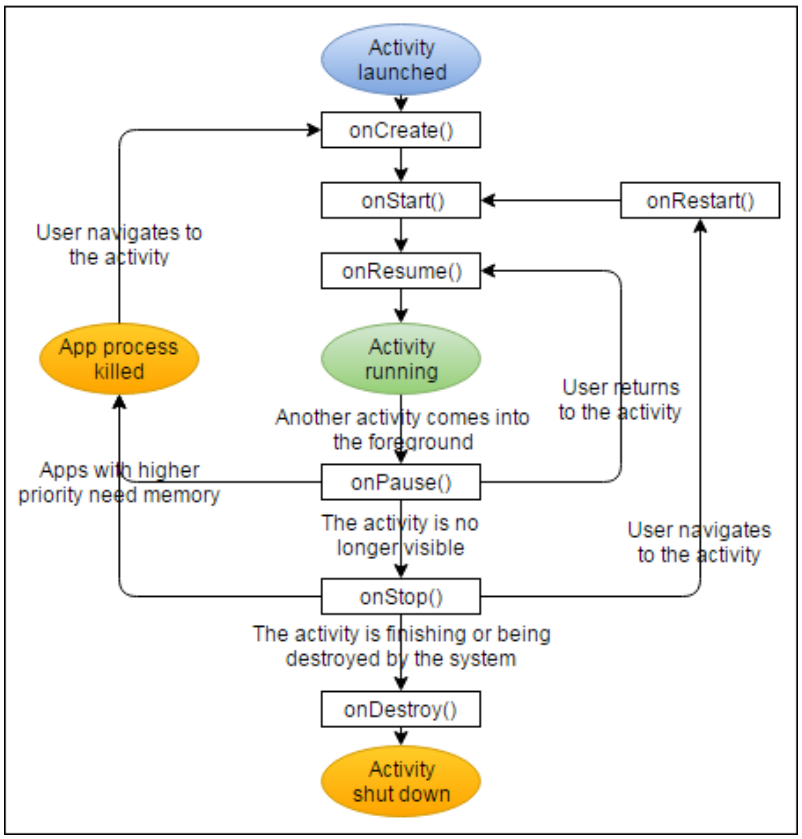
Tovább csökkentheti még a fogyasztást, ha a Bluetooth modult is Sleep módban tartjuk, és csak akkor élesztjük fel, amikor a jelzést kell a telefonnak küldeni. Ekkor be kell állítani azt, hogy a jelzést többször küldje ki egymás után a környezetének, például egy saját szerviz segítségével, amelynek UUID-ját csak a telefon tulajdonosa tudja, így csak ő kaphatja meg. Ekkor a telefon ténylegesen nem kapcsolódna a Bluetooth modulhoz, csak figyelne folyamatosan az esetleges érkező csomagra, vagyis a Smart modul Broadcaster a telefon pedig Observer állapotban lenne. Ekkor kérdés, hogy a telefon, hogyan tudná felkonfigurálni a lopásgátlót.

5.2 Android oldali szotfver

Az Android alkalmazás fejlesztésénél elsősorban az volt a cél, hogy kapcsolódva a Smart modulhoz, a felőle érkező üzeneteket fogadni tudjuk, és meg is jeleníthessük a felhasználó számára, ezzel figyelmeztetve, hogy a tárgyunkat elmozdították. A megjelenítést egy egyszerű grafikai interfész biztosítaná, amelyen egy ablakban kapnánk az üzenetet a mikrokontrollertől, illetve láthatnánk a Bluetooth modul bizonyos adatait, mint például a MAC címét vagy a nevét. Egy menüsorban pedig kiválaszthatnánk, hogy csatlakozni kívánunk hozzá vagy nem.

5.2.1 Az Android alkalmazás működése

Egy Android alkalmazás működése az úgynevezett activity-k köré épül. Egy activity meghatároz egy bizonyos feladatkört és ha egy új indul el, az bekerül az úgynevezett "activity stack" tetejére és elkezd futni. Az előző activity a stack-be marad és csak akkor jön elő a kijelzőn újra és kezd el futni, ha az éppen futó végez, és bezárul. Minden activity-nek van egy életciklusa (Lifecycle), ami leírja, hogyan jön létre, és zárul be.



5.6. ábra: Activity lifecycle

A lifecycle-t fontos ismerni az alkalmazás fejlesztésekor, mert olyan függvényeket tartalmaz, melyeket implementálnunk kell az egyes activity-ken belül. Például az *onCreate()* függvény akkor hívódik meg, amikor az adott activity létrejön, de még nem látjuk. Ekkor érdemes például az egyes változóinkat inicializálni, és a különböző layout-okat hozzárendelni az activity-hez. A fejlesztéskor két dolgot kell elvégezni. Az egyik maga a java program megírása, mely az egyes activity-k feladatát valósítja meg. A másik elvégzendő feladat, az egyes resource (forrás) fájlok létrehozása és szerkesztése. Ezeket használja a programunk forrásként. Innen kapja meg az egyes feliratokat, layout-okat, beállításokat, képeket. A kettő között az összerendelést az automatikusan generált R.java fájl tartalmazza, ebben a fájlban ugyanis benne van minden forrás elérési id-ja, amire hivatkozva elérhetjük az egyes elemeket.

A szakdolgozat két egyszerű activity-t definiál. Az egyik a MainActivity, amely a Bluetooth modul megtalálásáért és a kijelzőn egy listában történő megjelenítéséért felel. A másik pedig a DeviceControlActivity, amely a tényleges kapcsolat felvételét, az adatcserét és egy felhasználói felületet valósít meg. Ezeken kívül az egyes feliratokat a strings.xml fájl, az activity-khez tartozó kinézetet definiáló fájlokat a layout mappa, a menüket meghatározó fájlokat pedig a menu mappa tartalmazza.

5.2.2 MainActivity

A MainActivity layout által meghatározott képet látja először maga előtt a felhasználó. Felül, az úgynevezett *Toolbar*-on, található a menü. Itt kiválaszthatjuk, hogy elindítsunk-e egy scan-t a környező Bluetooth Smart modulok után, hogy megtaláljuk azokat, illetve le is állíthatjuk a keresést. A keresés futása közben egy *ProgressBar* jelzi, hogy a keresés fut. Ha a program megtalál egy Smart eszközt, akkor annak MAC címét illetve nevét berakja a *Toolbar* alatt létrehozott listában. Egy listaelem egy eszközt reprezentál. Ha a kilistázott Smart modulra kattintunk, akkor indul el a DeviceControlActivity.

5.2.2.1 Megjelenítés

Ezen működési mechanizmus eléréséhez először el kell készíteni az activity-hez tartozó layout-ot. A layout-ban definiálni kell egy *ListView*-t, amely az egyes eszközöket tárolja. Létre kell hozni a *Toolbar*-t, amely a menüt tartalmazza. Mivel egymás alá kerül minden komponens ezért egy *LinearLayout* megfelelő választás az egész struktúra magába foglalására. Az eszköz layout-ot külön definiáltam. Ez

önmagában is egy *LinearLayout*, amely két különböző betűméretet megjelenítő *TextView*-t tartalmaz. Ezek jelenítik meg az eszköz címét és nevét. A *Toolbar*-on megjelenő menüt pedig három *Item* alkotja: a scan, stop és refresh *Item*-ek, melyek közül a refresh nem klikkelhető, mert ehhez nem is akarunk funkcionalitást rendelni. Ha a megjelenítési oldal kész, akkor magát a java kódot kell megírni, amely megvalósítja a MainActivity feladatait.

5.2.2.2 Mögöttes programkód

A mögöttes programkód (code behind) végzi el tehát a vezérlést. Ezt a MainActivity.java fájlban készíthetjük el. Az *onCerate()* függvényen belül hozzárendeljük az activity-hez a layout-ot. Ezt a *setContentView(R.layout.<name>)* függvénnyel tehetjük meg, ahol a <name> a layout fájl neve. Az egyes layout elemek az id alapján elérhetőek a *findViewById(R.id.<id>)* függvénnyel, ahol az <id> a keresett *View* id attribútuma. Az *onCreate()*-en belül kell még megvalósítani azt, hogy egy eszközre kattintva, a másik activity induljon el. Ezt úgy érhetjük el, hogy az eszközöket tartalmazó *ListView*-hoz egy *OnItemClickListener*-t implementálunk. Ez az osztály tartalmazza az *OnItemClick()* függvényt, mely akkor hívódik ha az activity egy elemére kattintott a felhasználó. Ebben a függvényben kell tehát meghatározni, hogy ekkor mi történjen. Most azt akarjuk, hogy a kattintás hatására a másik activity-nek átadjuk az eszköz nevét és címét, mint extra információ, leállítsuk a további keresést, és elindítsuk a DeviceControlActivity-t. Az activity-k közötti adatátvitelt az úgynevezett *Intent*-ek segítségével érhetjük el. A new *Intent(MainActivity.this, DeviceControlActivity.class)* konstruktorral meghatározzuk, hogy ebből az activity-ből a másikba akarunk kerülni. a *.puExtra()* segítségével pedig hozzáadott információt is átadhatunk. Létre kell még hoznunk a *BluetoothManager*-t, amelytől a *BluetoothAdapter*-t kapjuk meg. A *BluetoothAdapter* reprezentálja a mobiltelefon Bluetooth modulját. Az *OnResume()* függvény közvetlenül az activity megjelenítése előtt hívódik. Ebben a függvényben ellenőrizhetjük hogy az adapterünk engedélyezve van-e, és létrehozhatjuk az *LeDeviceListAdapter*-t. Ezt fogjuk használni, Smart eszközök tárolására, és a megjelenítésükhöz társítjuk a *ListView*-hoz a *.setAdapter*() segítségével. Az alkalmazásunk felépülésekor rákérdezhetünk a felhasználónál, hogy engedélyezi-e a Bluetooth egységet. Az *onPause()* hívódásakor leállítjuk a keresést, és a kijelzőt reset-eljük. Az *onCreateOptionsMenu()* függvényben a *getMenuInflater()* segítségével tudjuk a létrehozott menu layout-ot megjeleníteni. Itt megvizsgáljuk, hogy éppen keresés

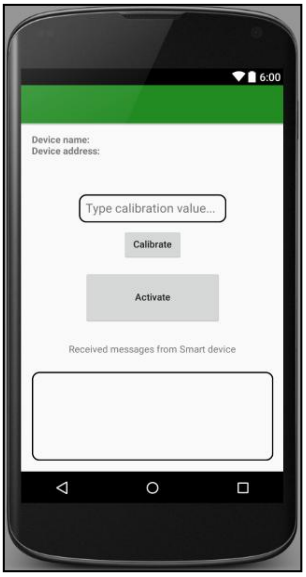
folyik-e, ha igen, akkor a stop menüpont látszik, és a refresh menü helyén egy *ProgressBar* van, ha nem akkor a scan menüpont látszik. Ez a függvény nem csak egyszer fog meghívódni, mert a programban több helyen is használjuk az *invalidateOptionsMenu()* függvényt, amely frissíti a megjelenítést az által, hogy az *onCreateOptionsMenu()*-t újra meghívja. Az *onOptionsItemSelected()* függvény akkor hívódik, ha egy menüpontra rákattintunk. a paraméter *.getItemId()* függvényét meghívva, megtudhatjuk, hogy a felhasználó éppen melyik pontra kattintott, és az alapján indíthatunk keresést vagy leállíthatjuk azt. A Bluetooth egységek utáni keresést a *scanLeDevice()* általunk definiált függvény végzi. Ha engedélyezzük, akkor elindítunk vele egy keresést, és definiálunk mellé a *Handler* (Ez kezeli azt az activity-t, amelyben definiálva van. Most a MainActivity-t.) *.postDelayed()* függvényének segítségével egy "delay"-t, ami adott idő után leállítja a keresést. Ha nem engedélyeztük a keresést, akkor egyszerűen leállítjuk. A *Handler*-re azért van szükség, mert a háttérben akarjuk a delay-t kivárni, és addig nem akarjuk feltartani a program futását, viszont a háttérben futó programrész nem fér hozzá az elől futó részekhez. Ezt hidalja át a *Handler*. A *BluetoothScan* rendelkezik az úgynevezett *LeScanCallback*-kel. Ez ad visszajelzést a keresésről. Itt használhatjuk az *onLeScan()* függvényt, mely akkor hívódik, ha megtaláltunk egy Bluetooth eszközt. A függvényt megvalósításánál a *runOnUiThread()* függvény számára kell egy *Runnable* objektumot megadni. Ebben az objektumban a *run()* függvényt kell implementálni, melyben a létrehozott *ListAdapter*-hez hozzáadjuk, az újonnan megtalált eszközt, illetve a *.notifyDataChanged()* függvénnyel jelezzük, hogy a lista tartalma változott. Az egyes eszközöket tehát egy *LeDeviceListAdapter*-ben tároljuk. Ezt kell még létrehozni a MainActivity-ben. Ez az osztály gyakorlatilag egy listát tartalmaz, amely tárolja az egyes *BluetoothDevice*-okat. Definiálunk hozzá függvényeket, melyekkel ehhez a listához hozzáadhatunk, vagy éppen lekérdezhetünk belőle. Végül pedig megvalósítjuk a *getView()* függvényét, melyben az egyes eszközökhöz tartozó layout-ot töltjük fel.

5.2.3 DeviceControlActivity

Ha tehát rákattintottunk egy listaelemre, akkor meghívódik ez az activity. Ennek feladata, hogy a kiválasztott eszközhöz csatlakozzon, majd az MLDP profil segítségével adatokat fogadjon a mikrokontroller oldali Bluetooth modul felől. Ezek az adatok lesznek a kontroller által küldött jelzések és figyelmeztetések, a tárgy elmozdításáról.

5.2.3.1 Megjelenítés

A megjelenítés hasonlóképpen történik, mint a MainActivity esetén. Ebben az esetben az interfészt egy olyan layout valósítja meg, amely tartalmaz egy *TextView*-t a beérkező adat számára, egy *Toolbar*-t a menünek, és továbbfejlesztési lehetőség gyanánt egy aktiváló gombot, mely a lopásgátlót aktiválja, illetve egy *EditText* elemet, amibe számokkal megadhatnánk a mérések tűréshatárát, ezáltal konfigurálva az eszközt. Az egész kinézetet most is egy *LinearLayout* foglalja magába, amelynek egyik eleme a *Toolbar*, a másik pedig maga az adatokat és gombokat tartalmazó interfész. Ezt az interfészt külön egy *RelativeLayout* foglalja körbe, amelyben az elemeket egymáshoz viszonyítva rendezhetjük, ellentétben a *LinearLayout*-tal, ahol sorosan. Az interfészünk, tetején vagyis a *Toolbar* alatt, két *TextView*-ban még kiírjuk, az előző activity-ből származó címet és nevet, a menüpontjaink, pedig a tényleges kapcsolódásra illetve szétkapcsolásra kellenek.



5.7. ábra: DeviceControlActivity layout

5.2.3.2 Mögöttes kód

Az *onCreate()* függvény megvalósítása hasonló az előbb leírtakhoz néhány változtatással. Az előző activity-ből egy *Intent* segítségével a kiválasztott *BluetoothDevice* címét és nevét is átadtuk. Ezt kell most kinyernünk belőle. Erre a *getIntent()* függvény a megoldás, amellyel először az *Intent*-hez férünk hozzá, majd annak a *getStringExtra()* függvényével az egyes adatokhoz is hozzáférünk. Ezeket be is írjuk a megfelelő *TextView*-ba. A menü létrehozása is hasonló, csak most Connect és Disconnect menüpontok vannak. Ha az egyikre rákattintunk akkor a másik jelenik meg

és fordítva. Az *onOptionsItemSelected()* függvénynél beállítjuk, hogy ha a felhasználó Connect-re kattint, akkor kapcsolódjunk a Bluetooth modulhoz, ha Disconnetre, akkor kapcsolódjunk le róla, ha a Home gombra akkor pedig térjünk vissza a MainActivity-hez. Az *onResume()*-ban az adapter meglétének és az érvényes eszközcímnek ellenőrzése után, az adapter *.getRemoteDevice()* függvényét a címmel meghívva, mint paraméter, megkapjuk a *BluetoothDevice* objektumunkat. Ennek *.connectGatt()* függvényét használva pedig a *BluetoothGatt* objektumot, amely a kommunikációhoz kell. A kapcsolat definiál egy *BluetoothGattCallback*-et, ami különböző visszajelzéseket ad. Olyan függvények hívódnak benne, mint az *OnConnectionStateChange(), onServicesDiscovered(), OnCharacteristicChange()* stb. Nekünk az említett három a legfontosabb most, illetve ha visszafelé is akarunk küldeni adatot akkor még az *OnCharacteristicWrite()*. Az *OnConnectionStateChange()*-ben megvizsgálhatjuk, hogy a kapcsolat éppen mire változott. Ha Connected lett, akkor a *BluetoothGatt .discoverServices()* függvényével a GATT profil szervizeit térképezhetjük fel. Ha ez megtörtént akkor hívódik meg az *onServicesDiscovered()*. Ennél az eseménynél hívjuk meg az általunk írt *findMLDPGattService()* függvényt, amely paraméterül kapja a *BluetoothGatt* objektumból hívott *.getServices()* által visszaadott *BluetoothGattService* listát. Ebben a listában lesznek az MLDP szervizek. A *findMLDPGattService()* feladata, hogy két, egymásba ágyazott, *for* ciklus segítségével, először menjen végig a szervizeken amíg a megadott UUID-jű MLDP privát szervizt meg nem találja, ha ez megvan akkor pedig menjen végig annak az összes karakterisztikáján. A karakterisztikák közül a megfelelő UUID-vel választja ki melyik az adatot tartalmazó és melyik a Control-t tartalmazó karakterisztika, illetve annak egyes tulajdonságait állítsa be. Ha megvannak a kívánt karakterisztikák, akkor ezek írásával és olvasásával létrejön a kommunikáció. A feladat során első sorban a figyelmeztetés fogadását kellett megoldani. Ezt úgy tehettük, hogy megvalósítottuk a már említett *GattCallback* visszajelzésben az *OnCharacteristicChange()* függvényt. Ez azért jó nekünk, mert akkor paraméterül a függvényben megkapjuk a megváltozott MLDPdata karakterisztikát, és abból egy *.getStringValue(0)* hívással ki tudjuk olvasni a megváltozott értéket. Ez minden olyan pillanatban bekövetkezik, amikor a Bluetooth modulon keresztül a mikrokontroller egy karaktert küld, vagyis karakterenként kapjuk az információt. Kell tehát még egy függvény, mely ezekből a karakterekből összefüggő üzenetet hoz létre. Ez a *processIncomingPacket()*, melyben egy globális String változóhoz fűzzük hozzá az újabb és újabb beérkező karaktert, majd ha elértünk a

"\r\n"-hez (ugyanis a mikrokontrollerrel is ilyen formátumban küldtünk stringet, például: "WARNING!\r\n"), akkor egy újabb *runOnUiThread()* hívásával az adatok fogadására létrehozott *TextView* tartalmához hozzáfűzzük ezt a stringet.

5.2.4 Android-os szoftver továbbfejlesztési lehetőségei

Ha tehát szeretnénk a már layout szinten létrehozott kalibrációt, és aktiválást is elvégezni, akkor először az aktiváló gombra kell egy *OnClickListener*-t megvalósítani, melyben elküldünk egy stringet a mikrokontrollerek, amely alapján az tudni fogja, hogy most élesítheti az eszközt. A kalibrációhoz pedig egy újabb *OnClickListener* kell a Calibrate gombhoz, amely kiolvassa a megadott értéket, és továbbítja a mikrokontrollernek, amely ez alapján átállítja a méréshatárt. Továbbfejlesztési lehetőség még, ha például egy arra alkalmas mobiltelefont elhelyezünk úgy, hogy rálátás legyen annak kamerájával a tárgyunkra, és a figyelmeztetés hatására, csinálunk egy képet. Ekkor azonban már az is megvalósítható lenne, hogy ezt a képet a telefon egy megadott telefonszámra küldje el MMS-ben.

6 Értékelés

A szakdolgozat témája, vagyis a Bluetooth Smart és Android alapú eszközök kapcsolata a fejlesztési területek viszonylag nagy részét foglalja magába. Lehetőséget ad, hogy a fejlesztő megismerje alaposabban a Bluetooth Smart technológiát, elmélyítse a mikrokontrollerekről szerzett tudását, és ízelítőt kapjon az Android operációs rendszerre írható alkalmazások fejlesztéséből. Ezek együttesen versenyképes tudást adhatnak egy villamosmérnök számára, melyből később is profitálhat, hiszen ezen technológiák az elmúlt években kezdtek rohamosan fejlődni és terjedni, ezáltal innovatív és modern területek.

A szakdolgozat során elkészült eszköz végleges állapotában praktikus megoldás lehet értéktárgyaink védelmének megőrzésére. Alacsony fogyasztása miatt környezetkímélő, illetve kevés és egyszerű összetevőinek köszönhetően olcsó is. Ez egy esetleges sorozatgyártás során fontos lehet. A hordozhatóság, illetve az androidos interfész nyújtotta kényelem miatt pedig jó választás lehet bárki számára. Akár magunkkal vihetjük nyaralásra, vagy üzleti útra is és este abban a tudatban térhetünk nyugovóra, hogy a személyes tárgyunkat megvédi a lopásgátlónk az illetéktelenektől.

Ábrajegyzék

Irodalomjegyzék

[1] Android Developer: *Android alkalmazás fejlesztéséhez online dokumentáció*
https://developer.android.com/develop/index.html
(2015.dec.)

[2] ARGENOX: *Leírás a Bluetooth Smart-ról*
http://www.argenox.com/bluetooth-low-energy-ble-v4-0-development/
(2015.dec.)

[3] Bluetooth Developer: *Bluetooth-os fejlesztéssel kapcsolatos leírás*
https://developer.bluetooth.org/TechnologyOverview/Pages/BLE.aspx
(2015.dec.)

[4] Mobilaréna: *Rövid cikk a Bluetooth Smart-ról*
http://mobilarena.hu/hir/bluetooth_4_0_le_atnevezve.html
(2015.dec.)

[5] Adafruit: *Rövid leírás a Bluetooth Smart-ról*
https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction
(2015.dec.)

[6] Safari: *Hosszabb leírás a Bluetooth Smart-ról*
https://www.safaribooksonline.com/library/view/getting-started-with/9781491900550/ch01.html
(2015.dec.)

[7] Microchip: *Az alkalmazott RN4020 Bluetooth modul hivatalos oldala*
http://www.microchip.com/wwwproducts/Devices.aspx?product=RN4020
(2015.dec.)

[8] Atmel: *Az alkalmazott ATmega328p mikrokontroller hivatalos oldala*
http://www.atmel.com/devices/atmega328p.aspx
(2015.dec.)

[9] Arduino: *Az alkalmazott Arduino UNO fejlesztői kártya hivatalos oldala*
https://www.arduino.cc/en/Main/ArduinoBoardUno
(2015.dec.)

[10] Ultrahangos szenzor 5V: *Online adatlap az alkalmazott HC-SR04 szenzorhoz*
https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit
(2015.dec.)

[11] Tápegység: *A hardver rendszertervhez kiválasztott tápegység adatlapja*
http://www.farnell.com/datasheets/1690388.pdf
(2015.dec.)

[12] Wikipedia: *Bluetooth Low Energy*
https://en.wikipedia.org/wiki/Bluetooth_low_energy
(2015.dec.)

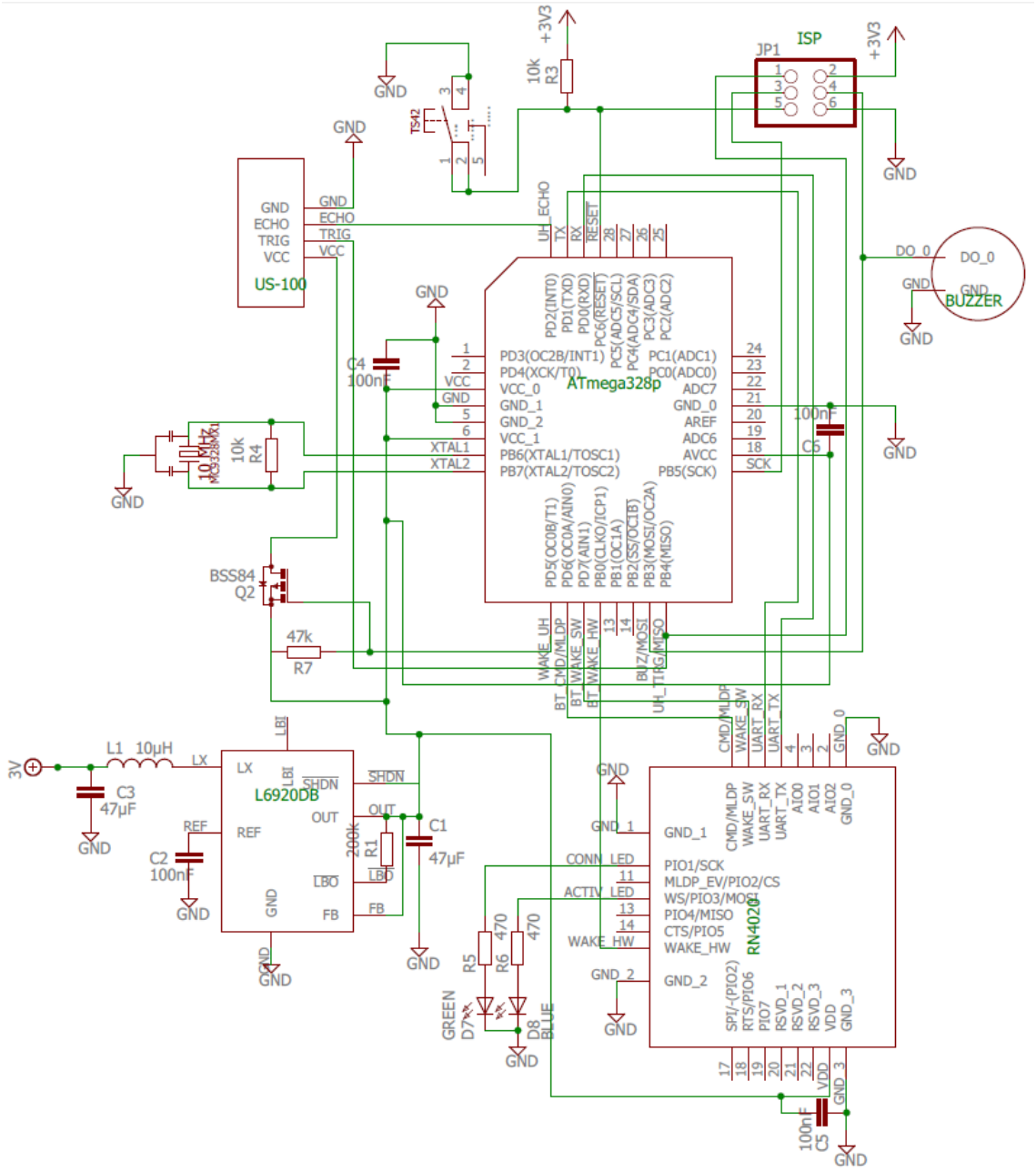
[13] Bluetooth rendszer: *Magyar nyelvű dokumentum a klasszikus Bluetooth-ról*
http://uni-obuda.hu/users/marosd/Bluetooth.pdf
(2015.dec.)

[14] WiFi modulok: *Az összehasonlításhoz kikeresett fogyasztási értékek*
http://eu.mouser.com/Embedded-Solutions/RF-Wireless-Modules/WiFi-80211-Modules/_/N-6l7qa
(2015.dec.)

[15] Arduino Uno és ATmega328p kivezetések ábrái:
https://bigdanzblog.wordpress.com/2015/01/30/cant-get-i2c-to-work-on-an-arduino-nano-pinout-diagrams/ és http://www.arconlab.com/lab/arduino.html
(2015.dec.)

[16] Ultrahangos szenzor 3.3V: *Online adatlap az alkalmazott US-100 szenzorhoz*
http://www.e-gizmo.com/KIT/images/ultrasonicsonar/ultrasonic%20sonar%20module%201r0.pdf (2015.dec.)

Függelék



A függelék: Teljes kapcsolási rajz a végleges hardverről 3.3 V-os szenzorral