SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5399

Sustav za upravljanje aktivnim automobilskim ovjesom

Karlo Svetec

Zagreb, lipanj 2018.

SADRŽAJ

1. Uvod

Aktivni automobilski ovjes podrazumijeva ovjes koji promjenom svojih fizičkih karakteristika utječe na karakteristike vožnje, koje pak poboljšavaju upravljivost vozila i udobnost putnika tijekom vožnje. U okviru ovog završnog rada razmatramo implementaciju sustava za upravljanje aktivnim automobilskim ovjesom. Sustav za upravljanje potreban nam je kako bi uopće mogli mijenjati karakteristike zračnog ovjesom, a zatim i za neke od naprednih funkcionalnosti, poput ispravljanja stava vozila tijekom vožnje, zaštitu od nenamjernog pražnjenja sistema i slično. Sustav je implementiran koristeći nekoliko sklopovskih komponenti i mikrokontroler koji predstavlja mozak samog sustava. Napredne funkcionalnosti poput rada sustava u dinamičkim uvjetima nisu implementirane u sklopu ovog završnog rada.

2. Automobilski ovjes

Ovjesni sustav automobila podrazumijeva skup komponenti koje spajaju karoseriju vozila s kotačima kako bi omogućile relativno gibanje između pojedinih dijelova. Prema načinu rada, dijele se na aktivne i pasivne. Neke od bitnijih komponenata ovjesnog sustava su: kotači s pneumaticima, opruge, amortizeri te razne mehaničke poveznice. Svrha ovjesnog sustava, osim već spomenute, je osigurati dobre vozne karakteristike, kao i udobnost putnicima u vozilu. Vozne karakteristike opisuju reakcije vozila na vozačevo upravljanje. Ocjenjujemo ih prema ponašanju vozila u zavojima, prilikom ubrzavanja i usporavanja, te zadržavanja pravca kretanja. Udobnost se opisuje mogućnošću apsorpcije udaraca i vibracija izazvanih neravninama na cestovnoj podlozi. Ovjes također osigurava maksimalan mogući kontakt vozila s podlogom. Sustav za upravljanje razvijen u okviru ovog završnog rada fokusira se na promjenu karakteristika zračnih opruga, koje zamjenjuju konvencionalne čelične opruge.

2.1 Aktivni automobilski ovjes

Aktivni automobilski ovjes sastoji se od aktivnih ovjesnih komponenata (opruge, amortizeri) koje omogućavaju izmjenu karakteristika ovjesa u stvarnom vremenu. Takav način rada omogućava nam upravljanje nad vertikalnim gibanjem kotača prema šasiji, za razliku od pasivnog ovjesa, kod kojeg je to gibanje određeno isključivo cestovnom podlogom. Možemo ga podijeliti na adaptivni i potpuno aktivni ovjes. Upravljanje kod adaptivnog ovjesa ostvareno je mogućnošću promjene tvrdoće amortizera ovisno o različitim tipovima podloge i načinu vožnje. S druge strane, potpuno aktivni ovjes nam nudi i podešavanje visine šasije prema svakom kotaču neovisno. Kao rezultat dobivamo bolje vozne karakteristike i veću udobnost. Prema vrsti, aktivni ovjes možemo podijeliti na hidraulički, elektromagnetski i pneumatski. Ovaj rad bavi se razvojem sustava za upravljanje pneumatskim ovjesom.

2.1.1 Princip rada pneumatskog ovjesa

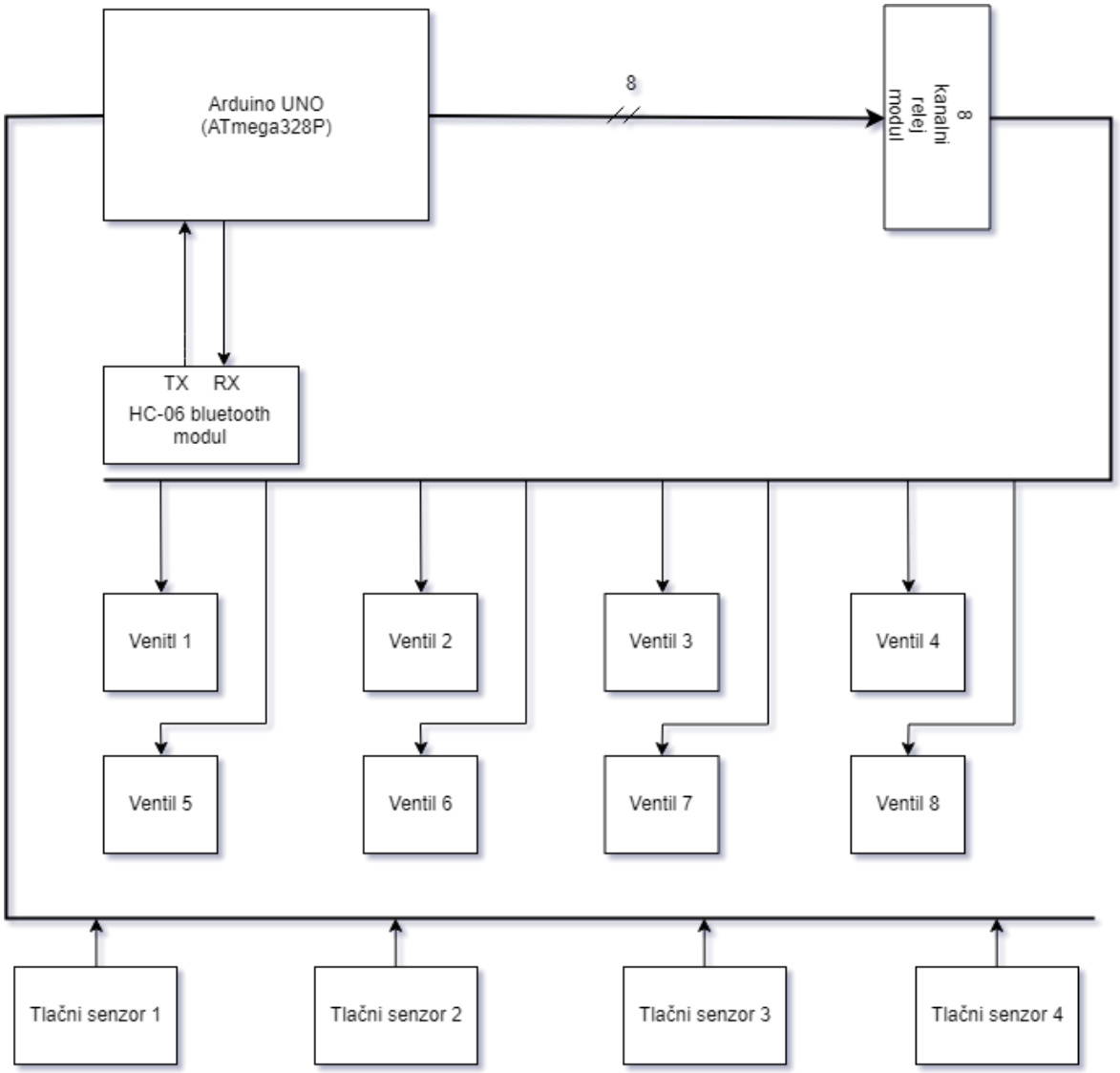
Osnovne komponente pneumatskog ovjesa su: gumene zračne opruge, ventili, spremnik sa stlačenim zrakom, kompresor, tlačna sklopka i plastične cijevi koje spajaju pojedine komponente. Gumene zračne opruge ugrađuju se umjesto konvencionalnih čeličnih opruga. To su ustvari gumeni mjehovi koji mijenjaju svoj oblik ovisno o količini i tlaku zraka koji se nalazi u njima. Što je veći tlak u opruzi, karoserija je više podignuta od kotača, te je sam auto povišen u odnosu na cestovnu podlogu. Kad ispuhujemo zrak iz opruge, karoserija se spušta prema cestovnoj podlozi. Niža razina vožnje omogućuje nam bolje vozne karakteristike. Ventili služe za upuhivanje (ulazni) i ispuhivanje (izlazni) zraka iz zračnih opruga. To su elektromagnetski ventili upravljani naponom iznosa 12 V. Kompresor služi za punjenje spremnika sa stlačenim zrakom. Na spremnik je pričvršćena i tlačna sklopka, koja je zadužena za uključivanje i isključivanje kompresora. Fiksni prag uključenja podešen je ispod 7.5 bara, a prag isključenja na 10.3 bara. Sustav za upravljanje aktivnim ovjesom razvijen u okviru ovog završnog rada omogućava preciznu kontrolu nad podešavanjem karakteristika zračnih opruga, odnosno visine vozila. Ugrađeno je bežično upravljanje sustavom putem Android aplikacije i bluetooth tehnologije, zbog olakšanja dovođenja auta na primjerenu postavku, prije nego uopće sjednemo u auto.

Slika 1. Stlačeni spremnik

Slika 2. Kompresor

3. Sklopovlje

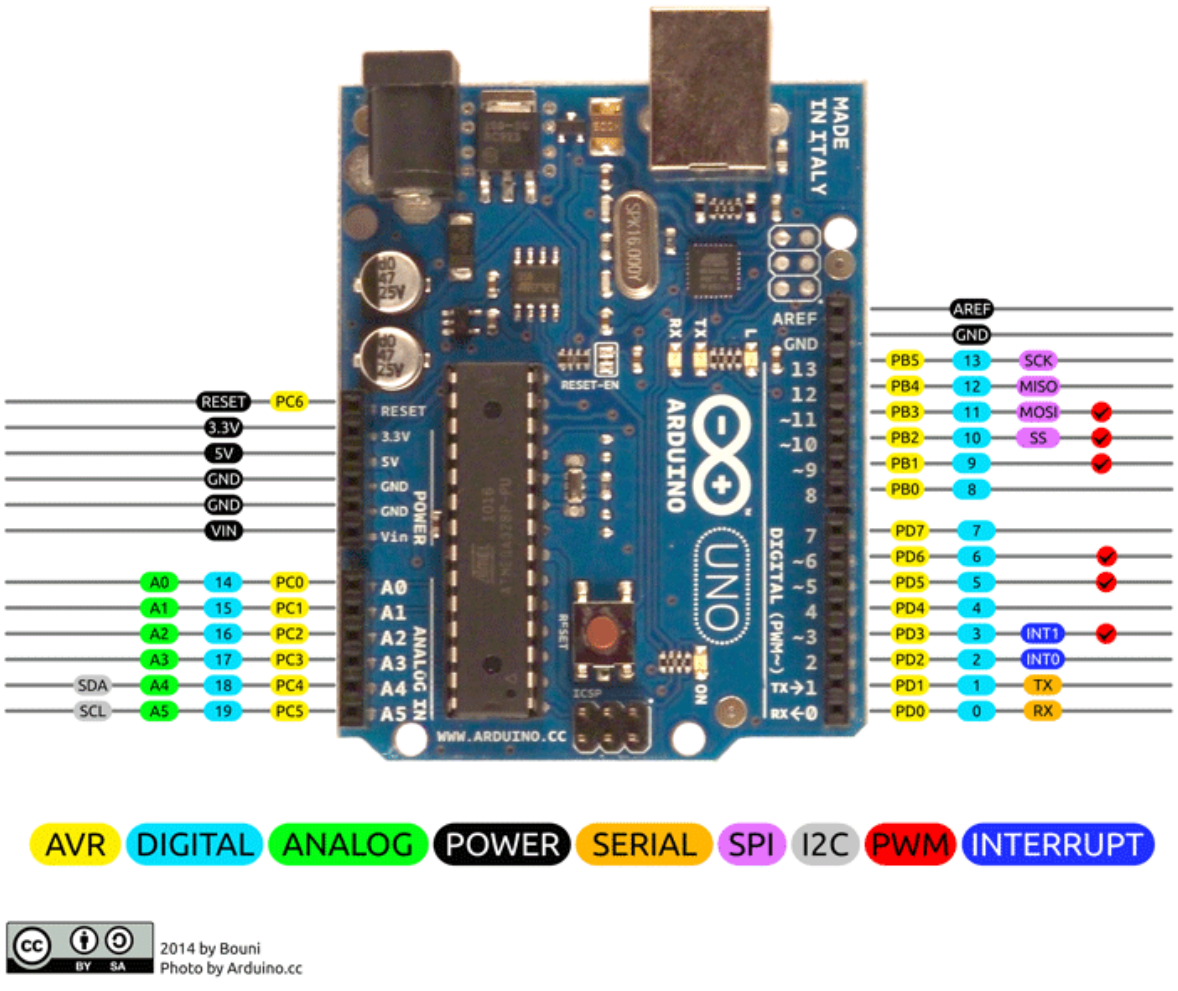
Upravljački dio sustava izveden je pomoću Atmega328P mikrokontrolera u okviru razvojne pločice Arduino. Za komunikaciju s mobilnom aplikacijom korištena je Bluetooth tehnologija implementirana pomoću HC-06 serijskog bluetooth modula. Ventilima za regulaciju tlaka u zračnim oprugama upravljamo pomoću 8 kanalnog relej modula. Napajanje svih komponenata osim ventila ostvareno je s DC-DC step down konverterima temeljenim na LM2596 regulatoru. Tlak u zračnim oprugama mjeri se tlačnim pretvaračima koji na izlazu daju napon između 0.5 i 4.5 V, ovisno o tlaku na ulazu.



Slika 3. Blok shema sklopa

3.1 Arduino UNO

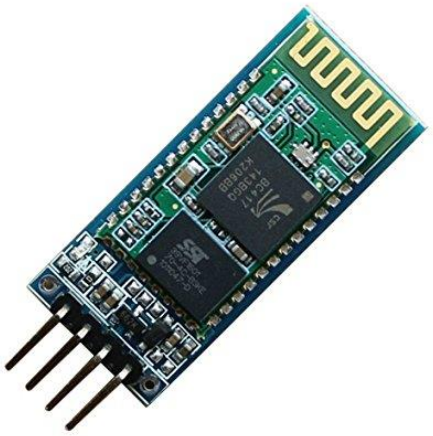
Arduino UNO je razvojna platforma koja se temelji na ATmega328P mikrokontroleru. Programira se kroz razvojno okruženje Arduino IDE, u programskom jeziku C. Takt procesora iznosi 16 MHz. Mikrokontroler sadrži 2 kB RAM memorije. Od ostalih tehničkih karakteristika ističe se 14 digitalnih ulazno-izlaznih priključaka, 6 analognih ulaznih priključaka, flash memorija veličine 32 kB, serijski USB-TTL čip, te operacijski napon od 5 V. Ova razvojna platforma odabrana je zbog niske cijene i relativno jednostavnog programiranja mikrokontrolera. Na slici 4 vidimo Arduino UNO pločicu s dijagramom priključaka. U okviru ovog projekta korišteni su analogni ulazni priključci A0 – A4 za očitanje vrijednosti senzora tlaka u zračnim oprugama i u spremniku sa stlačenim zrakom. Oni se oslanjaju na 10-bitni 6-kanalni A/D konverter koji je sastavni dio mikrokontrolera. Digitalni priključni 0 – 7 korišteni su za upravljanje relejima. Digitalni priključci 12 i 13 služe za komunikaciju s bluetooth modulom.



Slika 4. Arduino UNO sa označenim priključcima

3.2 HC-06 Bluetooth modul

Bluetooth je bežična tehnologija koja služi za prijenos podataka na male udaljenosti. U okviru ovog sustava bluetooth je korišten za izmjenu informacija između Android aplikacije i Arduina. Arduino mobilnom uređaju šalje podatke o trenutnim parametrima ovjesa, dok aplikacija šalje upravljačke naredbe prema Arduinu. Odabrani bluetooth modul nudi serijsko sučelje koje spajamo na Arduino. Postoje dva načina rada, slave i master. Kod master načina rada modul uspostavlja konekciju prema drugom uređaju, dok u slave načinu rada modul čeka na konekcije od strane drugih uređaja. Ovdje je korišten slave način rada. Neke od karakteristika modula su: osjetljivost -80 dBm, ugrađena antena, 8 MB flash memorije.

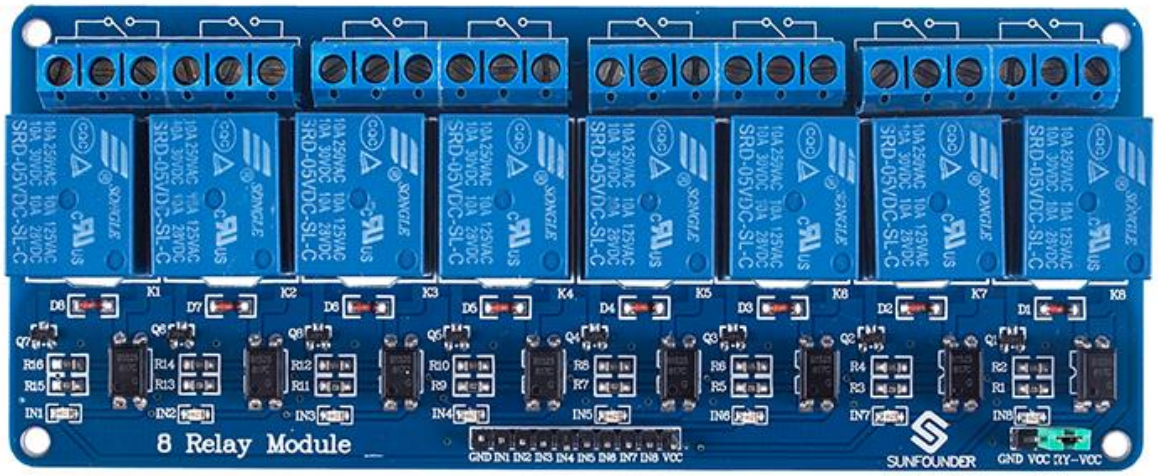


Slika 5. HC-06 bluetooth modul

3.3 Relej modul

Kao što je već spomenuto, Arduino upravlja ventilima putem svojih digitalnih izlaza, naponom visoke razine iznosa 5 V, niske razine 0 V, te malim iznosima struje. S druge strane, ventilima je na ulazu za upravljanje potreban napon iznosa 12 V i struja nekoliko redova veličina veća od one koju Arduino daje na svojim izlazima. Kako bi se ostvarila takva vrsta upravljanja, izabran je 8 kanalni relej modul upravljan s naponom iznosa 5 V. Relej je elektromehanička sklopka. Sastavljen je od elektromagneta kojeg pobuđujemo malom ulaznom strujom, kako bi mehanički upravljali sklopkom. Relej korišten u ovom sustavu podržava istosmjerne napone do 30 V i 10 A, kao i izmjenične napone do 250 V,

10 A. Na slici vidimo dizajn modula. Svaki od releja opremljen je zaštitnim sklopom koji štiti upravljačku elektroniku (Arduino) od velikih napona induciranih kod prestanka rada releja (isključenja sklopke).



Slika 6. 8 kanalni relej modul

3.4 DC-DC step down konverter (LM2596)

Kod implementacije razvijanog sustava u automobil dolazimo do zapreke. Glavni izvor napajanja u automobilu je akumulator, koji nam daje izlazni napon 12 V. Još jedan problem kod akumulatora su varijacije napona. Kada je auto ugašen, napon akumulatora trebao bi iznositi otprilike deklariranih 12 V. Međutim, kad se auto upali, alternator koji služi za punjenje akumulatora stvara napon između 14 i 15 V kako bi uopće mogao puniti akumulator. S obzirom na to da većina komponenata u sustavu radi na naponu 5 V, bilo je potrebno pronaći pouzdan izvor napajanja koji na ulazu prima takav varijabilni napon, a na izlazu daje stabilnih 5 V. Nakon razmatranja, izabran je podesivi DC-DC step down konverter temeljen na LM2596 (Buck switching regulator) koji na ulazu prima 3.2 – 40 V, a na izlazu daje 1.25 – 35 V. Najveća podržana izlazna struja iznosi 3 A. Korištena su dva takva izvora kako bi se odvojio napon na upravljačkom dijelu relej modula i napon u ostatku sklopa.

Slika 7. DC-DC step down konverter

3.5 Tlačni pretvarač

Tlačni pretvarač koristi se kako bi izmjerenu fizikalnu veličinu (vrijednost tlaka u zračnoj opruzi) dovedenu na ulaz pretvorili u električnu veličinu pogodnu za obradu u Arduinu. Izabran je tlačni pretvarač koji na ulaz dobiva tlak između 0 i 10.34 bara, a na izlazu daje napon linearno ovisan o ulaznom tlaku, između 0.5 i 4.5 V. Odstupanje izmjerene vrijednosti od stvarne vrijednosti tlaka deklarirano je između +/- 0.5%. Uzorkovanje ulaznog signala provedeno je na sljedeći način: Arduino na svojim analognim ulazima prima napon između 0 i 5 V, te ga pretvara u cijeli 10-bitni broj. To znači da se ulazne vrijednosti preslikavaju u brojeve između 0 i 1023 (2^{10} = 1024). Iz specifikacije tlačnog pretvarača znamo da je napon na izlazu za 0 bara dovedenih na ulaz 0.5 V. Prvo je potrebno tu vrijednost preslikati u 10-bitnu digitalnu vrijednost. Najmanji uzorak na izlazu A/D pretvarača jednak je:

$$\frac{5\ V}{1024\ uzoraka} = 4.88 * \ 10^{-3}\ V/uzorak \qquad (1)$$

Ako sada podijelimo 0.5 V s gore dobivenom vrijednosti, dobivamo:

$$\frac{0.5\ V}{4.88*10^{-3}\ V/uzorak} = 102.46\ uzorak \qquad (2)$$

Znamo da su nam vrijednosti uzoraka isključivo cijeli brojevi između 0 i 1023 pa gore dobivenu vrijednost zaokružujemo na najbliži cijeli broj. Dakle, 0.5 V na ulazu A/D pretvarača uzorkujemo s vrijednosti 102. Slično, za maksimalnu vrijednost na izlazu tlačnog pretvarača imamo:

$$\frac{4.5\ V}{4.88*10^{-3}\ V/uzorak} = 922.13\ uzoraka \qquad (3)$$

Zaokruženo na najbliži cijeli broj, 4.5 V uzorkujemo s 922 uzorka. Naš uzorkovani napon kreće se dakle između 102 i 922 uzorka. Skala je veličine 922 – 102 = 820. Konačno, formula za izračunavanje vrijednosti tlaka na temelju uzorkovanog napona je:

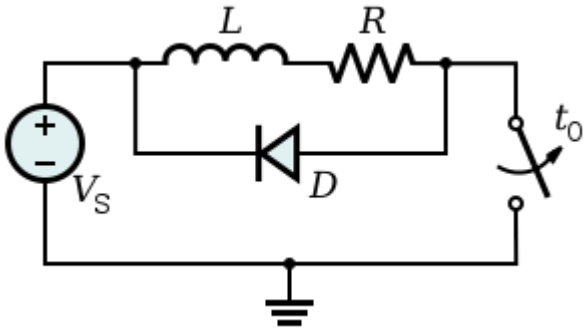
$$p = \frac{i-102}{820} * 10.34\ [bar] \qquad (4)$$

Gdje je i uzorkovana vrijednost napona.

Slika 8. Tlačni pretvarač

3.6 Ventili

Ključan dio sustava koji omogućava dovod i odvod zraka u zračne opruge, temelji se na elektromagnetskim ventilima. Korišteni su jednosmjerni normalno zatvoreni ventili (otvaraju se dovođenjem napona na upravljačku zavojnicu). Izabrani ventili podržavaju plinove i tekućine, a mi ih koristimo za zrak. Napon upravljanja zavojnice iznosi 12 V, a snaga zavojnice 6.5 W. Operacijski pritisak im je između 0 i 12 bara. Kod implementacije ventila pojavio se problem koji se manifestirao resetiranjem Arduina prilikom prekida napajanja na zavojnici ventila. Nakon razmatranja, zaključeno je da se radi o interferenciji uzrokovanoj pojavom naponskih šiljaka induciranih na upravljačkoj zavojnici ventila. Problem je riješen dodavanjem diode između dva pola zavojnice koja disipira taj naponski šiljak. Na slici x. vidimo shemu diode u induktivnom krugu.

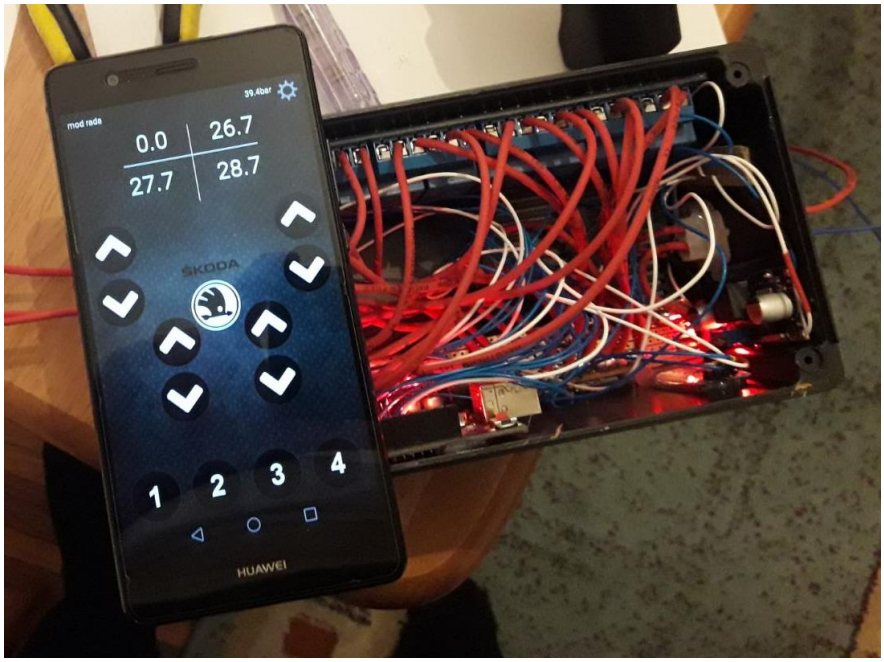


Slika 9. Dioda u induktivnom krugu

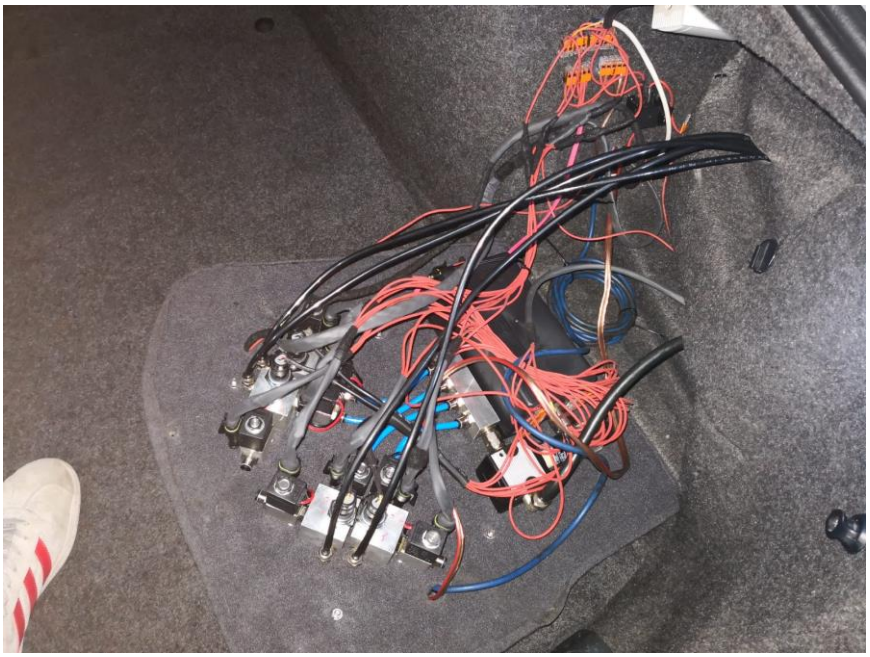
Slika 10. Elektromagnetski ventil

3.7 Konačan sklop

Slika 11 prikazuje izgled konačnog sklopa sastavljenog u plastično kučište i pokrenutu aplikaciju na mobilnom uređaju. Na slici 12 vidimo ventile spojene na upravljački sustav.



Slika 11. Izgled konačnog sklopa



Slika 12. Upravljački dio zračnog ovjesa

4. Programska podrška

U okviru sustava razvijenog u ovom završnom radu bilo je potrebno razviti programsku podršku za Arduino i Android uređaje. Aplikacija za Arduino ostvaruje logiku iza upravljanja ventilima, očitanja senzora i komunikacije s Android aplikacijom putem bluetooth tehnologije. S druge strane, aplikacija za Android uređaje se brine o ostvarivanju konekcije prema Arduinu, prikazu podataka o zračnom ovjesu na ekranu uređaja, te slanju naredbi Arduinu.

4.1 Programska podrška za Arduino

Za potrebe razvoja programske podrške za Arduino, odnosno za mikrokontroler Atmel328P, korišteno je razvojno okruženje Arduino IDE. Kod je napisan u programskom jeziku C, uz korištenje raznih biblioteka koje podržavaju funkcionalnosti mikrokontrolera. Svaki Arduino kod karakteristično se sastoji od dva dijela: funkcije setup() koja se poziva kod pokretanja ili resetiranja Arduina, te funkcije loop() koja se nakon izvršenja funkcije setup vrti u beskonačnoj petlji.

Za testiranje programa iskorišten je Serial monitor, dio razvojnog okruženja Arduino IDE koji služi za serijsku komunikaciju ugrađenu u sam mikrokontroler. Unutra koda možemo ispisivati na proizvoljnim pozicijama poruke koje nam mogu poslužiti za debugiranje, a vidimo ih na ekranu računala unutar Serial monitora. Na ovaj način su otkriveni naponski skokovi kod prekida napajanja na zavojnici elektromagnetskog ventila, jer je uočeno resetiranje Arduina prilikom zatvaranja ventila.

4.1.1 Struktura Arduino aplikacije

Program sadrži dvije vanjske biblioteke, SoftwareSerial.h i EEPROM.h. Arduino podržava serijsku komunikaciju putem UART-a, posebnog sklopa koji omogućava prijenos podataka paralelno s procesorskom aktivnosti. Budući da se taj isti sklop koristi za serijsku komunikaciju s računalom putem USB-a, tijekom procesa debugiranja, sklop nije bio iskoristiv za povezivanje bluetooth modula. Stoga smo bluetooth modul povezali na obične digitalne ulazno/izlazne priključke, te iskoristili biblioteku SoftwareSerial.h, koja programski emulira UART. Jedini nedostatak ovog pristupa je trošenje procesorskog

vremena na prijenos podataka. Biblioteka EEPROM.h korištena je za pristup trajnoj memoriji mikrokontrolera, za spremanje korisnički definiranih konfiguracija parametara zračnog ovjesa. Nakon deklariranja korištenih biblioteka, u programu su definirane konstante koje pridjeljuju simbolička imena fizičkim priključcima Arduina, prema komponentama koje su spojene na pojedini priključak. Definirane su i globalne varijable koje koristimo za spremanje vrijednosti očitanih tlakova, kao i ostale pomoćne varijable. U funkciji setup() postavljamo vrstu priključaka kojima upravljamo ventilima na izlazne, i postavljamo ih na visoku naponsku razinu (relej modul je upravljan niskom naponskom razinom). Zatim određujemo brzine sklopovskog i programskog serijskog priključka.
U funkciji loop() čekamo na dolazak naredbe s mobitela putem bluetootha, i izvršavamo akcije određene s kodom naredbe. Također periodički svaku sekundu čitamo vrijednosti svih tlačnih pretvornika, i šaljemo ih na mobilni uređaj. Funkcija presetSetup() služi za spremanje trenutne vrijednosti tlaka očitanih s tlačnih pretvornika u trajnu memoriju mikrokontrolera, kako bi ih kasnije mogli učitati i postaviti tlak u zračnim oprugama na te vrijednosti, pomoću funkcije toPreset(). Funkcija toPreset() prvo učita vrijednosti tlaka iz memorije u lokalne varijable, pošalje te vrijednosti mobilnom uređaju, te čeka na potvrdu korisnika. Ako ne dobije potvrdu unutar 10 sekundi, mobilnom uređaju šaljemo poruku o odustanku i vraćamo se u normalan režim rada (funkcija loop()). Ako pak dobijemo potvrdu, pozivamo funkciju toPressure() koja šalje upravljačke signale prema relej modulu, a oni dalje prema ventilima, kako bi doveli zračne opruge na određeni tlak. toPressure() sadrži petlju koja na početku očita vrijednosti tlakova, izračunava razliku između očitanog i ciljnog tlaka u svakoj od zračnih opruga, te u ovisnosti o predznaku šalje upravljački signal na ulazne ili izlazne ventile određene opruge. Ako je ta razlika manja od nule, tlak u opruzi je manji od željenog tlaka, te šaljemo upravljački signal na ulazni ventil koji puni oprugu sa zrakom iz stlačenog spremnika. Suprotno tome, ako je predznak pozitivan, odnosno razlika veća od nule, upravljački signal šaljemo na izlazni ventil kako bi izvršili pražnjenje zraka iz opruge. Izračunate razlike također koristimo za određivanje vremenske konstante koja određuje trajanje niske razine pojedinih upravljačkih signala, prema formuli:

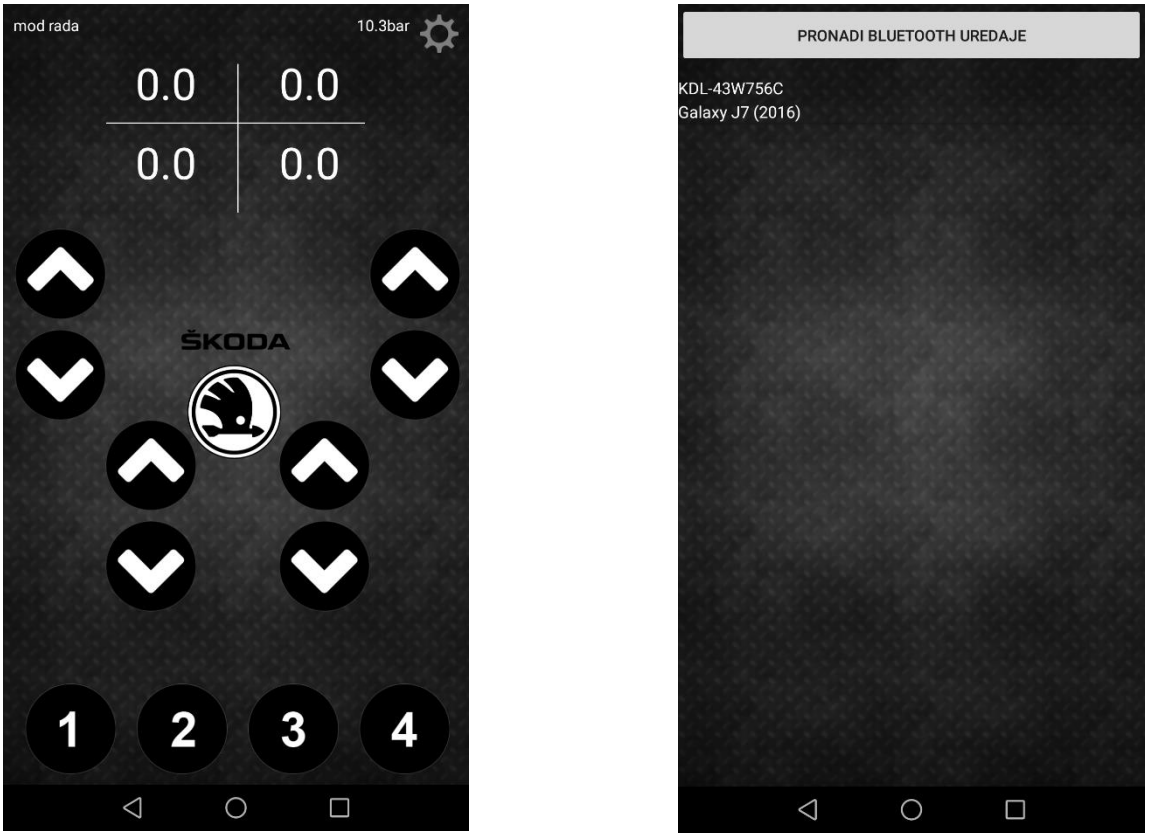
$$T = \left(p_{očitano} - p_{ciljno}\right) * 400\ [ms] \qquad (5)$$

U praksi se vrijednosti tlakova otprilike kreću između 0 i 5 bara, a konstanta od 400 ms određena je eksperimentalno, metodom pokušaja i pogrešaka. Takvo računanje vremenskih koraka osigurava nam duže trajanje niskog signala, što je očitana vrijednost tlaka dalje od ciljne vrijednosti. Kada se približimo ciljnoj vrijednosti, vremenski koraci postaju sve manji, kako bi postigli veću preciznost podešavanja na određen tlak. Na kraju svakog vremenskog koraka ponovno očitavamo vrijednosti senzora i obavještavamo korisnika o tlakovima putem mobilne aplikacije, kako bi vidio napredak izabrane akcije. Na kraju funkcije toPressure() računamo apsolutnu vrijednost trenutne razlike u tlakovima za svaku oprugu. Ako je ona manja od vrijednosti tolerancije, koja iznosi 0.1, dostigli smo željeni tlak i prekidamo upravljanje nad tom oprugom. Funkcija readSensors() čita vrijednosti analognih ulaza, odnosno digitalne diskretizirane vrijednosti očitane s tlačnih pretvornika, preračunava ih u vrijednosti izražene u barima, i sprema u globalne varijable dodijeljene svakom od tlačnih pretvornika. Naposljetku, funkcija publishSensors() očitane vrijednosti slaže u formatirani oblik i šalje ih mobilnoj aplikaciji.

4.1.2 Struktura Android aplikacije

Za pisanje Android aplikacije korišteno je razvojno okruženje Android studio, koje nam nudi jednostavnu izradu korisničkog sučelja po drag and drop principu. Aplikacija je napisana u programskom jeziku Java. Najvažnije korištene biblioteke omogućavaju nam upravljanje s komponentama grafičkog sučelja i komunikaciju s bluetooth adapterom mobilnog uređaja. Program se sastoji od četiri klase: MainActivity i SettingsActivity, klase povezane s dva različita prozora aplikacije, te BluetoothConnectionService i DeviceListAdapter, zaduženih za ostvarivanje bluetooth konekcije i prikaz liste dostupnih bluetooth uređaja. Klasa MainActivity na početku definira simboličke konstatne za naredbe koje se šalju prema Arduinu. Dalje definiramo objekte zadužene za stvaranje bluetooth konekcije, i jednu privatnu pomoćnu klasu Properties koja definira zastavice za prikaz poruke tijekom pozivanja spremljenih korisničkih postavki. Metoda onCreate() poziva se prilikom otvaranja aplikacije. U njoj stvaramo reference na sve komponente grafičkog sučelja, kako bi ih kasnije mogli modificirati. Zatim stvaramo promatrače koji čekaju pritiske na određene komponente grafičkog sučelja, i izvršavaju akcije sukladne namjeni pojedine komponente. Bitno je napomenuti još i metodu startConnection(), koja

stvara konekciju prema bluetooth uređaju izabranom u SettingsActivity. U klasi SettingsActivity ostvarena je funkcionalnost pretraživanja dostupnih bluetooth uređaja. Za formatiranje liste dostupnih uređaja koristimo klasu DeviceListAdapter. BluetoothConnectionService klasa sadrži metode za dohvat, obradu i prikaz podataka poslanih od strane Arduina. Za debugiranje Android aplikacije iskorištena je klasa Log, koja nam omogućava ispis proizvoljnih poruka unutar koda u dnevnik događaja, koji je sastavni dio Android studia.



Slika 13. Zasloni Android aplikacije

Slika lijevo prikazuje glavni prozor aplikacije. U gornjem lijevom kutu predviđeno je mjesto za oznaku načina rada. Postoje dva načina rada: manualni i automatski. Pri manualnom načinu rada korisnik isključivo samostalno upravlja sa zračnim ovjesom, dok u automatskom sustav sam održava razinu tlakova određenu s odabranom spremljenom korisničkom postavkom. U gornjem desnom kutu vidimo vrijednost razine tlaka u stlačenom spremniku. Pokraj te oznake, nalazi se ikona zupčanika koja nas vodi u drugi prozor aplikacije, prikazan na slici desno. Ispod toga vidimo četiri oznake koje

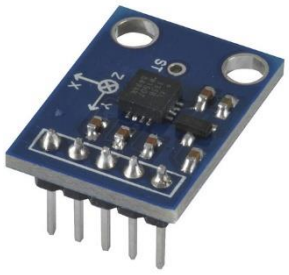
predstavljaju tlak u svakoj od zračnih opruga. Pozicija oznaka jednaka je kao i fizička pozicija opruga na automobilu, npr. u gornjem lijevom kutu, prikazan je tlak u prednjoj lijevoj opruzi. Tipke sa strelicama grupirane u četiri grupe po dvije tipke aktiviraju ventile na pojedinim oprugama, npr. gornja lijeva grupa tipki služi za podešavanje prednje lijeve opruge. Tipka sa strelicom prema gore aktivira ulazni ventil, a tipka sa strelicom prema dolje izlazni ventil. Naposlijetku, imamo četiri tipke numerirane brojevima 1-4 koje služe za memoriranje i pozivanje četiri korisničke postavke. Kratkim pritiskom na bilo koju od tih tipki pozivamo postavku, a dugim pritiskom iniciramo spremanje trenutnih vrijednosti tlaka na memorijsku lokaciju označenu brojem na tipki. Nakon dugog pritiska na odabranu tipku, akciju spremanja potvrđujemo pritiskom na tipku u sredini aplikacije (Škoda logo). Na slici desno vidimo drugi prozor aplikacije do kojeg dolazimo pritiskom na ikonu zupčanika u glavnom prozoru. Na vrhu prozora nalazi se gumb koji inicira postupak pronalaska bluetooth uređaja u dometu mobilnog uređaja. Tijekom postupka, pronađeni uređaji se pojavljuju u listi koja se prikazuje ispod gumba. Pritiskom na uređaj u listi, aplikacija se spaja na odabrani uređaj.

5. Upravljanje sustavom u dinamičkim uvjetima

U okviru ovog završnog rada nije predviđena izvedba upravljanja u dinamičkim uvjetima, međutim, bit će navedeni okvirni zahtjevi sklopovskih komponenata i izmjene u programskoj podršci za izvedbu takvog upravljanja. Dinamički uvjeti podrazumijevaju vozilo u pokretu. Analiza dinamički uvjeta vozila u pokretu je iznimno kompliciranа, uključuje proučavanje sila koje djeluju na vozilo na različitim nagibima cestovnih podloga, kao i prilikom ubrzavanja i kočenja. Konkretna analiza također nije izvršena, ali predložene su komponente koje bi mogle pomoći u provođenju takve analize.

5.1 Sklopovlje za ostvarenje dinamičkog upravljanja

Kako bi mogli dinamički mijenjati stav vozila prilikom vožnje u zavojima, potrebno je kontinuirano tijekom vožnje izvršavati mjerenje stava vozila u zavoju (nagib vozila). Kada imamo dostupne podatke o stavu, možemo izvršiti punjenje određenih zračnih opruga zrakom, kako bi postigli što veću silu koja se odupire izlijetanju vozila iz zavoja, te time postigli veću brzinu prolaska zavoja. To bi bilo posebno važno u sustavima razvijenim za trkače automobile, međutim, i u automobilima za svakodnevnu vožnju doprinijelo bi stabilnosti i udobnosti. Druga situacija u kojoj nam je korisno upravljanje u dinamičkim uvjetima je kod ubrzavanja i usporavanja. Prilikom ubrzavanja prednji kraj vozila podiže se od poda, pa bi ga mogli spustiti i tako dobiti na aerodinamičnosti. Slično kod usporavanja, prednji kraj vozila se spušta, i potrebno ga je podići za poboljšanje učinka kočenja. Komponenta koja omogućava mjerenje zahtijevanih fizikalnih veličina je akcelerometar (mjerač ubrzanja). Akcelerometar je uređaj koji pomoću mehaničkih i elektroničkih komponenti ugrađenih u njegove kućište mjeri ubrzanja na 3 osi.



Slika 14. Akcelerometar za Arduino

6. Zaključak

Razvijen je programski i sklopovski sustav zadužen za upravljanje radom aktivnog automobilskog ovjesa. Sustav je razvijen za rad u statičkim uvjetima, međutim, predložene su promjene te sustav omogućava nadogradnju za rad u dinamičkim uvjetima. Razvoj sustava za rad u dinamički uvjetima bitno je kompliciraniji od statičkih uvjeta zbog promjenjivih sila na sve dijelove vozila tijekom vožnje.

Sustav je testiran u stvarnom okruženju, na stvarnom osobnom vozilu. Karakteristike sustava pokazale su se očekivanima, nisu primjećene nikakve značajne pogreške u radu u statičkim uvjetima.

Za primjenu sustava u stvarnom okruženju potrebno je provesti dodatna testiranja, s obzirom na to da postoji i sigurnosna komponenta koja ograničava puštanje jednog takvog sustava u rad.

Literatura

[1] Reza N. Jazar: Vehicle Dynamics: Theory and Applications, Springer, 2008.

[2] Wikipedia - Suspension (vehicle): https://en.wikipedia.org/wiki/Suspension_(vehicle)

[3] Wikipedia – Active suspension: https://en.wikipedia.org/wiki/Active_suspension

[4] Arduino Software Serial Library: https://www.arduino.cc/en/Reference/softwareSerial

[5] Arduino EEPROM Library: https://www.arduino.cc/en/Reference/EEPROM

[6] HC-06 datasheet: https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf

[7] Atmega328P microcontroller datasheet: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

Sustav za upravljanje aktivnim automobilskim ovjesom

Sažetak

U radu je ukratko opisan princip rada automobilskog ovjesa i podjela na aktivni i pasivni ovjesni sustav. Objašnjen je način rada i osnovne komponente aktivnog automobilskog ovjesa, kao i način realizacije jednog takvog sustava. Navedene su i opisane sklopovske komponente potrebne za izradu sustava za upravljanje i način njihovog spajanja. Rezultat rada je sustav za upravljanje aktivnim automobilskim ovjesom u statičkim uvjetima ostvaren uporabom mikrokontrolera ATmega328P. Na kraju rada predložene su izmjene koje bi omogućile rad sustava u dinamičkim uvjetima.

Ključne riječi: automobilski ovjes, upravljanje, sustav za upravljanje, mikrokontroler, Arduino

An Active Automotive Suspension Control System

Abstract

This paper describes basic working principles of automotive suspension and the divisons on active and passive suspension system. It explains how active suspension works, and the basic components from which it is built. Hardware components for building the control system for automotive suspension are described with connections between them explained. The result of this paper is an active automotive suspension control system for controlling in a static environment, developed on Atmega328P microcontroller. In the end are suggested changes for controlling the system in a dynamic environment.

Keywords: car suspension, controlling, control system, microcontroller, Arduino

Dodatak A

A.1. Arduino kod

```
#include <SoftwareSerial.h>
#include <EEPROM.h>

#define FRONT_LEFT_UP 2
#define FRONT_LEFT_DOWN 5
#define FRONT_RIGHT_UP 4
#define FRONT_RIGHT_DOWN 3
#define REAR_LEFT_UP 6
#define REAR_LEFT_DOWN 7
#define REAR_RIGHT_UP 8
#define REAR_RIGHT_DOWN 9

#define FRONT_LEFT_SENSOR 16
#define FRONT_RIGHT_SENSOR 17
#define REAR_LEFT_SENSOR 14
#define REAR_RIGHT_SENSOR 15
#define TANK_SENSOR 18

SoftwareSerial BT(10, 11);

byte a;
int i;

float frontLeftPressure;
float frontRightPressure;
float rearLeftPressure;
float rearRightPressure;
float tankPressure;

int floatSize = sizeof(float);

char pressureBuffer[28];
char potvrda[5]="";
char odustanak[5]="";

unsigned long previousMillis = 0;
const long interval = 1000;

void setup()
{
  pinMode(FRONT_LEFT_UP, OUTPUT);
  digitalWrite(FRONT_LEFT_UP, HIGH);

  pinMode(FRONT_LEFT_DOWN, OUTPUT);
  digitalWrite(FRONT_LEFT_DOWN, HIGH);

  pinMode(FRONT_RIGHT_UP, OUTPUT);
  digitalWrite(FRONT_RIGHT_UP, HIGH);

  pinMode(FRONT_RIGHT_DOWN, OUTPUT);
  digitalWrite(FRONT_RIGHT_DOWN, HIGH);

  pinMode(REAR_LEFT_UP, OUTPUT);
  digitalWrite(REAR_LEFT_UP, HIGH);

  pinMode(REAR_LEFT_DOWN, OUTPUT);
  digitalWrite(REAR_LEFT_DOWN, HIGH);
```

```
58.
59.   pinMode(REAR_RIGHT_UP, OUTPUT);
60.   digitalWrite(REAR_RIGHT_UP, HIGH);
61.
62.   pinMode(REAR_RIGHT_DOWN, OUTPUT);
63.   digitalWrite(REAR_RIGHT_DOWN, HIGH);
64.
65.   BT.begin(115200);
66.   Serial.begin(115200);
67.   Serial.println("Setup");
68. }
69.
70.
71. void loop()
72. {
73.     if(BT.available()){
74.          a = BT.read();
75.
76.          if(a == 0) digitalWrite(FRONT_LEFT_UP, LOW);
77.          if(a == 1) digitalWrite(FRONT_LEFT_UP, HIGH);
78.
79.          if(a == 2) digitalWrite(FRONT_LEFT_DOWN, LOW);
80.          if(a == 3) digitalWrite(FRONT_LEFT_DOWN, HIGH);
81.
82.          if(a == 4) digitalWrite(FRONT_RIGHT_UP, LOW);
83.          if(a == 5) digitalWrite(FRONT_RIGHT_UP, HIGH);
84.
85.          if(a == 6) digitalWrite(FRONT_RIGHT_DOWN, LOW);
86.          if(a == 7) digitalWrite(FRONT_RIGHT_DOWN, HIGH);
87.
88.          if(a == 8) digitalWrite(REAR_LEFT_UP, LOW);
89.          if(a == 9) digitalWrite(REAR_LEFT_UP, HIGH);
90.
91.          if(a == 10) digitalWrite(REAR_LEFT_DOWN, LOW);
92.          if(a == 11) digitalWrite(REAR_LEFT_DOWN, HIGH);
93.
94.          if(a == 12) digitalWrite(REAR_RIGHT_UP, LOW);
95.          if(a == 13) digitalWrite(REAR_RIGHT_UP, HIGH);
96.
97.          if(a == 14) digitalWrite(REAR_RIGHT_DOWN, LOW);
98.          if(a == 15) digitalWrite(REAR_RIGHT_DOWN, HIGH);
99.
100.          if(a == 16) toPreset(0);
101.          if(a == 17) toPreset(1);
102.          if(a == 18) toPreset(2);
103.          if(a == 19) toPreset(3);
104.
105.          if(a == 20) presetSetup();
106.       }
107.
108.        unsigned long currentMillis = millis();
109.        if (currentMillis - previousMillis >= interval) {
110.             previousMillis = currentMillis;
111.             readSensors();
112.             publishSensors(frontLeftPressure, frontRightPressure, rearLeftPressure,
                                rearRightPressure, tankPressure);
113.             delay(100);
114.     }
115.   }
116.
117.   void presetSetup(){
118.     boolean set = false;
119.
120.     unsigned long currentMillis = millis();
121.     unsigned long staro = currentMillis;
122.
```

```
123.     while(true){
124.       if(set == true) break;
125.
126.       if(BT.available()){
127.         a = BT.read();
128.
129.         if(a == 255){
130.           currentMillis = millis();
131.           if (currentMillis - staro >= interval) {
132.             staro = currentMillis;
133.             readSensors();
134.             publishSensors(frontLeftPressure, frontRightPressure, rearLeftPressure,
                                rearRightPressure, tankPressure);
135.             delay(150);
136.           }
137.         }
138.
139.         if(a == 21){
140.           int address = 0;
141.
142.           EEPROM.put(address, frontLeftPressure);
143.           address += floatSize;
144.           EEPROM.put(address, frontRightPressure);
145.           address += floatSize;
146.           EEPROM.put(address, rearLeftPressure);
147.           address += floatSize;
148.           EEPROM.put(address, rearRightPressure);
149.           set = true;
150.         }
151.
152.         if(a == 22){
153.           int address = 4 * floatSize;
154.
155.           EEPROM.put(address, frontLeftPressure);
156.           address += floatSize;
157.           EEPROM.put(address, frontRightPressure);
158.           address += floatSize;
159.           EEPROM.put(address, rearLeftPressure);
160.           address += floatSize;
161.           EEPROM.put(address, rearRightPressure);
162.           set = true;
163.         }
164.
165.         if(a == 23){
166.           int address = 8 * floatSize;
167.
168.           EEPROM.put(address, frontLeftPressure);
169.           address += floatSize;
170.           EEPROM.put(address, frontRightPressure);
171.           address += floatSize;
172.           EEPROM.put(address, rearLeftPressure);
173.           address += floatSize;
174.           EEPROM.put(address, rearRightPressure);
175.           set = true;
176.         }
177.
178.         if(a == 24){
179.           int address = 12 * floatSize;
180.
181.           EEPROM.put(address, frontLeftPressure);
182.           address += floatSize;
183.           EEPROM.put(address, frontRightPressure);
184.           address += floatSize;
185.           EEPROM.put(address, rearLeftPressure);
186.           address += floatSize;
187.           EEPROM.put(address, rearRightPressure);
```

```
            set = true;
          }
        }
      }
    }

    void toPreset(int n){
        unsigned long currentMillis = millis();
        unsigned long staro = currentMillis;
        int address = n*4*floatSize;
        boolean set = false;
        float frontLeft, frontRight, rearLeft, rearRight;

        if(n == 0){
          strcpy(potvrda,"01/1");
          strcpy(odustanak, "02/1");
        }

        if(n == 1){
          strcpy(potvrda,"01/2");
          strcpy(odustanak, "02/2");
        }

        if(n == 2){
          strcpy(potvrda,"01/3");
          strcpy(odustanak, "02/3");
        }

        if(n == 3){
          strcpy(potvrda,"01/4");
          strcpy(odustanak, "02/4");
        }


        EEPROM.get(address, frontLeft);
        address += floatSize;
        EEPROM.get(address, frontRight);
        address += floatSize;
        EEPROM.get(address, rearLeft);
        address += floatSize;
        EEPROM.get(address, rearRight);
        publishSensors(frontLeft, frontRight, rearLeft, rearRight, tankPressure);
        delay(100);

        while(true){
          if(set == true) break;
          currentMillis = millis();
          if (currentMillis - staro >= 10000) {
                BT.print(odustanak);
                delay(500);
                set = true;
          }

          if(BT.available()){
            a = BT.read();
            if(a == 25){
              toPressure(frontLeft, frontRight, rearLeft, rearRight);
              BT.print(potvrda);
              delay(500);
              set = true;
            }
          }
        }

    }
```

```
254.   void toPressure(float frontLeft, float frontRight, float rearLeft,
                       float rearRight){
255.       double diff;
256.       boolean flagFL, flagFR, flagRL, flagRR;
257.       int timeF, timeR;
258.       float F, R;
259.
260.       flagFL = false; flagFR = false; flagRL = false; flagRR=false;
261.
262.       while(flagFL == false || flagFR == false || flagRL == false ||
                flagRR == false){
263.           readSensors();
264.
265.           if(!flagRL || !flagRR){
266.             diff = rearLeftPressure - rearLeft;
267.             R = absolute(diff)*400.0;
268.             timeR = (int) R;
269.             if(diff < 0.0 && !flagRL){
270.               digitalWrite(REAR_LEFT_UP, LOW);
271.             }
272.
273.             if(diff > 0.0 && !flagRL){
274.               digitalWrite(REAR_LEFT_DOWN, LOW);
275.             }
276.
277.             diff = rearRightPressure - rearRight;
278.             if(diff < 0.0 && !flagRR){
279.               digitalWrite(REAR_RIGHT_UP, LOW);
280.             }
281.
282.             if(diff > 0.0 && !flagRR){
283.               digitalWrite(REAR_RIGHT_DOWN, LOW);
284.             }
285.           }
286.
287.           if(!flagFL || !flagFR){
288.             diff = frontLeftPressure - frontLeft;
289.             F = absolute(diff)*400.0;
290.             timeF = (int) F;
291.                 if(diff < 0.0 && !flagFL){
292.                    digitalWrite(FRONT_LEFT_UP, LOW);
293.             }
294.
295.             if(diff > 0.0 && !flagFL){
296.               digitalWrite(FRONT_LEFT_DOWN, LOW);
297.             }
298.
299.             diff = frontRightPressure - frontRight;
300.             if(diff < 0.0 && !flagFR){
301.               digitalWrite(FRONT_RIGHT_UP, LOW);
302.             }
303.
304.             if(diff > 0.0 && !flagFR){
305.               digitalWrite(FRONT_RIGHT_DOWN, LOW);
306.             }
307.           }
308.
309.          if(timeF <= timeR){
310.             delay(timeF);
311.             digitalWrite(FRONT_LEFT_UP, HIGH);
312.             digitalWrite(FRONT_LEFT_DOWN, HIGH);
313.             digitalWrite(FRONT_RIGHT_UP, HIGH);
314.             digitalWrite(FRONT_RIGHT_DOWN, HIGH);
315.
316.             delay(timeR-timeF);
317.             digitalWrite(REAR_LEFT_UP, HIGH);
```

```
318.              digitalWrite(REAR_LEFT_DOWN, HIGH);
319.              digitalWrite(REAR_RIGHT_UP, HIGH);
320.              digitalWrite(REAR_RIGHT_DOWN, HIGH);
321.           }
322.
323.           if(timeF > timeR){
324.              delay(timeR);
325.              digitalWrite(REAR_LEFT_UP, HIGH);
326.              digitalWrite(REAR_LEFT_DOWN, HIGH);
327.              digitalWrite(REAR_RIGHT_UP, HIGH);
328.              digitalWrite(REAR_RIGHT_DOWN, HIGH);
329.
330.              delay(timeF-timeR);
331.              digitalWrite(FRONT_LEFT_UP, HIGH);
332.              digitalWrite(FRONT_LEFT_DOWN, HIGH);
333.              digitalWrite(FRONT_RIGHT_UP, HIGH);
334.              digitalWrite(FRONT_RIGHT_DOWN, HIGH);
335.           }
336.
337.            delay(100);
338.            readSensors();
339.            publishSensors(frontLeftPressure, frontRightPressure, rearLeftPressure,
                              rearRightPressure, tankPressure);
340.            delay(100);
341.
342.                diff = absolute(frontLeftPressure - frontLeft);
343.                if(diff < 0.1) {
344.                  flagFL = true;
345.                }
346.
347.                diff = absolute(frontRightPressure - frontRight);
348.                if(diff < 0.1) {
349.                  flagFR = true;
350.                }
351.
352.                diff = absolute(rearLeftPressure - rearLeft);
353.                if(diff < 0.1) {
354.                  flagRL = true;
355.                }
356.
357.                diff = absolute(rearRightPressure - rearRight);
358.                if(diff < 0.1) {
359.                  flagRR = true;
360.                }
361.
362.        }
363.
364.
365.        flagFL = false; flagFR = false; flagRL = false; flagRR=false;
366.
367.        while(flagFL == false || flagFR == false || flagRL == false ||
                flagRR == false){
368.            readSensors();
369.
370.            if(!flagRL || !flagRR){
371.              diff = rearLeftPressure - rearLeft;
372.              R = absolute(diff)*400.0;
373.              timeR = (int) R;
374.              if(diff < 0.0 && !flagRL){
375.                digitalWrite(REAR_LEFT_UP, LOW);
376.              }
377.
378.              if(diff > 0.0 && !flagRL){
379.                digitalWrite(REAR_LEFT_DOWN, LOW);
380.              }
381.
```

```
382.              diff = rearRightPressure - rearRight;
383.              if(diff < 0.0 && !flagRR){
384.                digitalWrite(REAR_RIGHT_UP, LOW);
385.              }
386.
387.              if(diff > 0.0 && !flagRR){
388.                digitalWrite(REAR_RIGHT_DOWN, LOW);
389.              }
390.            }
391.
392.            if(!flagFL || !flagFR){
393.              diff = frontLeftPressure - frontLeft;
394.              F = absolute(diff)*400.0;
395.              timeF = (int) F;
396.              if(diff < 0.0 && !flagFL){
397.               digitalWrite(FRONT_LEFT_UP, LOW);
398.              }
399.
400.              if(diff > 0.0 && !flagFL){
401.                digitalWrite(FRONT_LEFT_DOWN, LOW);
402.              }
403.
404.              diff = frontRightPressure - frontRight;
405.              if(diff < 0.0 && !flagFR){
406.                digitalWrite(FRONT_RIGHT_UP, LOW);
407.              }
408.
409.              if(diff > 0.0 && !flagFR){
410.                digitalWrite(FRONT_RIGHT_DOWN, LOW);
411.              }
412.            }
413.
414.           if(timeF <= timeR){
415.              delay(timeF);
416.              digitalWrite(FRONT_LEFT_UP, HIGH);
417.              digitalWrite(FRONT_LEFT_DOWN, HIGH);
418.              digitalWrite(FRONT_RIGHT_UP, HIGH);
419.              digitalWrite(FRONT_RIGHT_DOWN, HIGH);
420.
421.              delay(timeR-timeF);
422.              digitalWrite(REAR_LEFT_UP, HIGH);
423.              digitalWrite(REAR_LEFT_DOWN, HIGH);
424.              digitalWrite(REAR_RIGHT_UP, HIGH);
425.              digitalWrite(REAR_RIGHT_DOWN, HIGH);
426.           }
427.
428.           if(timeF > timeR){
429.              delay(timeR);
430.              digitalWrite(REAR_LEFT_UP, HIGH);
431.              digitalWrite(REAR_LEFT_DOWN, HIGH);
432.              digitalWrite(REAR_RIGHT_UP, HIGH);
433.              digitalWrite(REAR_RIGHT_DOWN, HIGH);
434.
435.              delay(timeF-timeR);
436.              digitalWrite(FRONT_LEFT_UP, HIGH);
437.              digitalWrite(FRONT_LEFT_DOWN, HIGH);
438.              digitalWrite(FRONT_RIGHT_UP, HIGH);
439.              digitalWrite(FRONT_RIGHT_DOWN, HIGH);
440.           }
441.
442.            delay(100);
443.            readSensors();
444.            publishSensors(frontLeftPressure, frontRightPressure, rearLeftPressure,
                              rearRightPressure, tankPressure);
445.            delay(100);
446.
```

```
447.            diff = absolute(frontLeftPressure - frontLeft);
448.            if(diff < 0.1) {
449.              flagFL = true;
450.            }
451.
452.            diff = absolute(frontRightPressure - frontRight);
453.            if(diff < 0.1) {
454.              flagFR = true;
455.            }
456.
457.            diff = absolute(rearLeftPressure - rearLeft);
458.            if(diff < 0.1) {
459.              flagRL = true;
460.            }
461.
462.            diff = absolute(rearRightPressure - rearRight);
463.            if(diff < 0.1) {
464.              flagRR = true;
465.            }
466.
467.        }
468.    }
469.
470.    int readSensors(){
471.      int fl, fr, rl, rr, t;
472.
473.      fl = analogRead(FRONT_LEFT_SENSOR);
474.      delay(5);
475.      fr = analogRead(FRONT_RIGHT_SENSOR);
476.      delay(5);
477.      rl = analogRead(REAR_LEFT_SENSOR);
478.      delay(5);
479.      rr = analogRead(REAR_RIGHT_SENSOR);
480.      delay(5);
481.      t = analogRead(TANK_SENSOR);
482.
483.      frontLeftPressure = (float)((fl-96)*12.0/819);
484.      frontRightPressure = (float)((fr-96)*12.0/819);
485.      rearLeftPressure = (float)((rl-96)*12.0/819);
486.      rearRightPressure = (float)((rr-96)*12.0/819);
487.      tankPressure = (float)((t-96)*12.0/819);
488.
489.    }
490.
491.    void publishSensors(float frontLeft, float frontRight, float rearLeft,
                           float rearRight, float tank){
492.      char buffFrontLeft[5], buffFrontRight[5], buffRearLeft[5], buffRearRight[5],
               buffTank[5];
493.      dtostrf(frontLeft,4,1,buffFrontLeft);
494.      dtostrf(frontRight,4,1,buffFrontRight);
495.      dtostrf(rearLeft,4,1,buffRearLeft);
496.      dtostrf(rearRight,4,1,buffRearRight);
497.      dtostrf(tank,4,1,buffTank);
498.
499.      pressureBuffer[0] = '0';
500.      pressureBuffer[1] = '0';
501.      pressureBuffer[2] = '/';
502.
503.      for(int i=0; i<4; i++){
504.        pressureBuffer[3+i] = buffFrontLeft[i];
505.      }
506.
507.      pressureBuffer[7]='/';
508.
509.      for(int i=0; i<4; i++){
510.        pressureBuffer[8+i] = buffFrontRight[i];
```

```
    }

    pressureBuffer[12]='/';

    for(int i=0; i<4; i++){
      pressureBuffer[13+i] = buffRearLeft[i];
    }

    pressureBuffer[17]='/';

    for(int i=0; i<4; i++){
      pressureBuffer[18+i] = buffRearRight[i];
    }

    pressureBuffer[22]='/';

    for(int i=0; i<4; i++){
      pressureBuffer[23+i] = buffTank[i];
    }

    BT.print(pressureBuffer);
  }

  double absolute(double num){
    if(num < 0.0) return num *= -1.0;
    else return num;
  }
```

A.2. Android kod

A.2.1. MainActivity.java

```
package com.air.karlo.airmanagement;

import android.annotation.SuppressLint;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Build;
import android.os.SystemClock;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    private static final byte FRONT_LEFT_UP_START = 0;
    private static final byte FRONT_LEFT_UP_STOP = 1;
```

```
28.     private static final byte FRONT_LEFT_DOWN_START = 2;
29.     private static final byte FRONT_LEFT_DOWN_STOP = 3;
30.     private static final byte FRONT_RIGHT_UP_START = 4;
31.     private static final byte FRONT_RIGHT_UP_STOP = 5;
32.     private static final byte FRONT_RIGHT_DOWN_START = 6;
33.     private static final byte FRONT_RIGHT_DOWN_STOP = 7;
34.
35.
36.     private static final byte REAR_LEFT_UP_START = 8;
37.     private static final byte REAR_LEFT_UP_STOP = 9;
38.     private static final byte REAR_LEFT_DOWN_START = 10;
39.     private static final byte REAR_LEFT_DOWN_STOP = 11;
40.     private static final byte REAR_RIGHT_UP_START = 12;
41.     private static final byte REAR_RIGHT_UP_STOP = 13;
42.     private static final byte REAR_RIGHT_DOWN_START = 14;
43.     private static final byte REAR_RIGHT_DOWN_STOP = 15;
44.
45.     private static final byte PRESET_1 = 16;
46.     private static final byte PRESET_2 = 17;
47.     private static final byte PRESET_3 = 18;
48.     private static final byte PRESET_4 = 19;
49.
50.     private static final byte PRESET_SETUP = 20;
51.
52.     private static final byte PRESET_1_SETPRESSURE = 21;
53.     private static final byte PRESET_2_SETPRESSURE = 22;
54.     private static final byte PRESET_3_SETPRESSURE = 23;
55.     private static final byte PRESET_4_SETPRESSURE = 24;
56.
57.     private static final byte SET_TO_PRESET_1 = 25;
58.     private static final byte SET_TO_PRESET_2 = 26;
59.     private static final byte SET_TO_PRESET_3 = 27;
60.     private static final byte SET_TO_PRESET_4= 28;
61.
62.
63.     final Properties myProperties = new Properties();
64.     BluetoothConnectionService mBluetoothConnection;
65.     private static final UUID MY_UUID_INSECURE =
66.             UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
67.     BluetoothDevice mBTDevice;
68.     BluetoothAdapter mBluetoothAdapter;
69.     String device = "00:18:E4:00:0D:63";
70.
71.
72.
73.     @SuppressLint("ClickableViewAccessibility")
74.     @Override
75.     protected void onCreate(Bundle savedInstanceState) {
76.         super.onCreate(savedInstanceState);
77.
78.         requestWindowFeature(Window.FEATURE_NO_TITLE);
79.         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManag
                                    er.LayoutParams.FLAG_FULLSCREEN);
80.
81.         setContentView(R.layout.activity_main);
82.
83.         ImageView settingsButton = (ImageView) findViewById(R.id.settingsButton);
84.         ImageView okButton = (ImageView) findViewById(R.id.logo);
85.
86.         ImageView frontLeftUp = (ImageView) findViewById(R.id.frontLeftUp);
87.         ImageView frontLeftDown = (ImageView) findViewById(R.id.frontLeftDown);
88.         ImageView frontRightUp = (ImageView) findViewById((R.id.frontRightUp));
89.         ImageView frontRightDown = (ImageView) findViewById(R.id.frontRightDown);
90.
91.         ImageView rearLeftUp = (ImageView) findViewById(R.id.rearLeftUp);
92.         ImageView rearLeftDown = (ImageView) findViewById(R.id.rearLeftDown);
```

```
93.         ImageView rearRightUp = (ImageView) findViewById(R.id.rearRightUp);
94.         ImageView rearRightDown = (ImageView) findViewById(R.id.rearRightDown);
95.
96.         ImageView preset1 = (ImageView) findViewById(R.id.preset1);
97.         ImageView preset2 = (ImageView) findViewById(R.id.preset2);
98.         ImageView preset3 = (ImageView) findViewById(R.id.preset3);
99.         ImageView preset4 = (ImageView) findViewById(R.id.preset4);
100.
101.            TextView modRada = (TextView) findViewById(R.id.modRada);
102.            TextView status = (TextView) findViewById(R.id.status);
103.
104.            mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
105.
106.            if (!mBluetoothAdapter.isEnabled()) {
107.                mBluetoothAdapter.enable();
108.            }
109.
110.            startConnection();
111.
112.            Log.i("mojTag", "onCreateMain");
113.
114.            settingsButton.setOnClickListener(v -> {
115.                Intent i = new Intent(this, SettingsActivity.class);
116.                startActivityForResult(i, 0);
117.            });
118.
119.            frontLeftUp.setOnTouchListener(new View.OnTouchListener() {
120.                @SuppressLint("ClickableViewAccessibility")
121.                @Override
122.                public boolean onTouch(View v, MotionEvent event) {
123.                    switch(event.getAction()) {
124.                        case MotionEvent.ACTION_DOWN:
125.                            mBluetoothConnection.writeByte(FRONT_LEFT_UP_START);
126.                            return true;
127.                            //return true; // if you want to handle the touch event
128.                        case MotionEvent.ACTION_UP:
129.                            mBluetoothConnection.writeByte(FRONT_LEFT_UP_STOP);
130.                            return true;
131.                            //return true; // if you want to handle the touch event
132.                        default:
133.                            break;
134.                    }
135.                    return false;
136.                }
137.            });
138.
139.            frontLeftDown.setOnTouchListener((v, e) ->{
140.                switch(e.getAction()) {
141.                    case MotionEvent.ACTION_DOWN:
142.                        mBluetoothConnection.writeByte(FRONT_LEFT_DOWN_START);
143.                        return true;
144.                    //return true; // if you want to handle the touch event
145.                    case MotionEvent.ACTION_UP:
146.                        mBluetoothConnection.writeByte(FRONT_LEFT_DOWN_STOP);
147.                        return true;
148.                    //return true; // if you want to handle the touch event
149.                    default:
150.                        break;
151.                }
152.                return false;
153.            });
154.
155.            frontRightUp.setOnTouchListener((v, e) -> {
156.                switch(e.getAction()) {
157.                    case MotionEvent.ACTION_DOWN:
158.                        mBluetoothConnection.writeByte(FRONT_RIGHT_UP_START);
```

```
159.                     return true;
160.                 //return true; // if you want to handle the touch event
161.                 case MotionEvent.ACTION_UP:
162.                     mBluetoothConnection.writeByte(FRONT_RIGHT_UP_STOP);
163.                     return true;
164.                 //return true; // if you want to handle the touch event
165.                 default:
166.                     break;
167.             }
168.             return false;
169.         });
170.
171.         frontRightDown.setOnTouchListener((v, e) -> {
172.             switch(e.getAction()) {
173.                 case MotionEvent.ACTION_DOWN:
174.                     mBluetoothConnection.writeByte(FRONT_RIGHT_DOWN_START);
175.                     return true;
176.                 //return true; // if you want to handle the touch event
177.                 case MotionEvent.ACTION_UP:
178.                     mBluetoothConnection.writeByte(FRONT_RIGHT_DOWN_STOP);
179.                     return true;
180.                 //return true; // if you want to handle the touch event
181.                 default:
182.                     break;
183.             }
184.             return false;
185.         });
186.
187.         rearLeftUp.setOnTouchListener((v, e) -> {
188.             switch(e.getAction()) {
189.                 case MotionEvent.ACTION_DOWN:
190.                     mBluetoothConnection.writeByte(REAR_LEFT_UP_START);
191.                     return true;
192.                 //return true; // if you want to handle the touch event
193.                 case MotionEvent.ACTION_UP:
194.                     mBluetoothConnection.writeByte(REAR_LEFT_UP_STOP);
195.                     return true;
196.                 //return true; // if you want to handle the touch event
197.                 default:
198.                     break;
199.             }
200.             return false;
201.         });
202.
203.         rearLeftDown.setOnTouchListener((v, e) -> {
204.             switch(e.getAction()) {
205.                 case MotionEvent.ACTION_DOWN:
206.                     mBluetoothConnection.writeByte(REAR_LEFT_DOWN_START);
207.                     return true;
208.                 //return true; // if you want to handle the touch event
209.                 case MotionEvent.ACTION_UP:
210.                     mBluetoothConnection.writeByte(REAR_LEFT_DOWN_STOP);
211.                     return true;
212.                 //return true; // if you want to handle the touch event
213.                 default:
214.                     break;
215.             }
216.             return false;
217.         });
218.
219.         rearRightUp.setOnTouchListener((v, e) -> {
220.             switch(e.getAction()) {
221.                 case MotionEvent.ACTION_DOWN:
222.                     mBluetoothConnection.writeByte(REAR_RIGHT_UP_START);
223.                     return true;
224.                 //return true; // if you want to handle the touch event
```

```
                    case MotionEvent.ACTION_UP:
                        mBluetoothConnection.writeByte(REAR_RIGHT_UP_STOP);
                        return true;
                    //return true; // if you want to handle the touch event
                    default:
                        break;
                }
                return false;
            });

            rearRightDown.setOnTouchListener((v, e) -> {
                switch(e.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                        mBluetoothConnection.writeByte(REAR_RIGHT_DOWN_START);
                        return true;
                    //return true; // if you want to handle the touch event
                    case MotionEvent.ACTION_UP:
                        mBluetoothConnection.writeByte(REAR_RIGHT_DOWN_STOP);
                        return true;
                    //return true; // if you want to handle the touch event
                    default:
                        break;
                }
                return false;
            });

            preset1.setOnClickListener((v) -> {
                if(!myProperties.isPreset1Flag()) {
                    status.setText("Želite li učitati postavku 1?");
                    if (mBluetoothConnection != null) mBluetoothConnection.writeByte(
                                                                    PRESET_1);
                    myProperties.setPreset1Flag(true);
                }

                else if(myProperties.preset1Flag){
                    status.setText("");
                    if(mBluetoothConnection != null) mBluetoothConnection.writeByte(
                                                                    SET_TO_PRESET_1);
                    myProperties.setPreset1Flag(false);
                }
            });

            preset2.setOnClickListener((v) -> {
                if(!myProperties.isPreset2Flag()) {
                    status.setText("Želite li učitati postavku 2?");
                    if (mBluetoothConnection != null) mBluetoothConnection.writeByte(
                                                                    PRESET_2);
                    myProperties.setPreset2Flag(true);
                }

                else if(myProperties.preset2Flag){
                    status.setText("");
                    if(mBluetoothConnection != null) mBluetoothConnection.writeByte(
                                                                    SET_TO_PRESET_2);
                    myProperties.setPreset2Flag(false);
                }
            });

            preset3.setOnClickListener((v) -> {
                if(!myProperties.isPreset3Flag()) {
                    status.setText("Želite li učitati postavku 3?");
                    if (mBluetoothConnection != null) mBluetoothConnection.writeByte(
                                                                    PRESET_3);
                    myProperties.setPreset3Flag(true);
                }

```

```
286.                else if(myProperties.preset3Flag){
287.                    status.setText("");
288.                    if(mBluetoothConnection != null) mBluetoothConnection.writeByte(
                                                                SET_TO_PRESET_3);
289.                    myProperties.setPreset3Flag(false);
290.                }
291.            });
292.
293.            preset4.setOnClickListener((v) -> {
294.                if(!myProperties.isPreset4Flag()) {
295.                    status.setText("Želite li učitati postavku 4?");
296.                    if (mBluetoothConnection != null) mBluetoothConnection.writeByte(
                                                                PRESET_4);
297.                    myProperties.setPreset4Flag(true);
298.                }
299.
300.                else if(myProperties.preset4Flag){
301.                    status.setText("");
302.                    if(mBluetoothConnection != null) mBluetoothConnection.writeByte(
                                                                SET_TO_PRESET_4);
303.                    myProperties.setPreset4Flag(false);
304.                }
305.            });
306.
307.            preset1.setOnLongClickListener((v) -> {
308.                status.setText("Postavljanje postavke 1");
309.                if(myProperties.isBluetoothConnected()) mBluetoothConnection.
                                                        writeByte(PRESET_SETUP);
310.                myProperties.setAction(PRESET_1_SETPRESSURE);
311.                return true;
312.            });
313.
314.            preset2.setOnLongClickListener((v) -> {
315.                status.setText("Postavljanje postavke 2");
316.                if(myProperties.isBluetoothConnected()) mBluetoothConnection.
                                                        writeByte(PRESET_SETUP);
317.                myProperties.setAction(PRESET_2_SETPRESSURE);
318.                return true;
319.            });
320.
321.            preset3.setOnLongClickListener((v) -> {
322.                status.setText("Postavljanje postavke 3");
323.                if(myProperties.isBluetoothConnected()) mBluetoothConnection.
                                                        writeByte(PRESET_SETUP);
324.                myProperties.setAction(PRESET_3_SETPRESSURE);
325.                return true;
326.            });
327.
328.            preset4.setOnLongClickListener((v) -> {
329.                status.setText("Postavljanje postavke 4");
330.                if(myProperties.isBluetoothConnected()) mBluetoothConnection.
                                                        writeByte(PRESET_SETUP);
331.                myProperties.setAction(PRESET_4_SETPRESSURE);
332.                return true;
333.            });
334.
335.            okButton.setOnClickListener((v) -> {
336.                status.setText("");
337.                if(myProperties.isBluetoothConnected()) mBluetoothConnection.
                                        writeByte(myProperties.getAction());
338.            });
339.
340.
341.        }
342.
343.        @Override
```

```
344.        protected void onResume() {
345.            super.onResume();
346.
347.        }
348.
349.        @Override
350.        protected void onDestroy() {
351.            super.onDestroy();
352.        }
353.
354.        public void onActivityResult(int requestCode, int resultCode, Intent data) {
355.
356.            if(resultCode == RESULT_OK) {
357.                mBTDevice = mBluetoothAdapter.getRemoteDevice(data.getStringExtra(
                                                                    "device_address"));
358.                mBluetoothConnection = new BluetoothConnectionService(MainActivity.
                                                                        this, this);
359.                mBluetoothConnection.startClient(mBTDevice, MY_UUID_INSECURE);
360.                myProperties.setBluetoothConnected(true);
361.            }
362.        }
363.
364.        private void startConnection(){
365.            mBTDevice = mBluetoothAdapter.getRemoteDevice(device);
366.            if(Build.VERSION.SDK_INT > Build.VERSION_CODES.JELLY_BEAN_MR2) {
367.                mBTDevice.createBond();
368.            }
369.            mBluetoothConnection = new BluetoothConnectionService(MainActivity.this,
                                                                                this);
370.            mBluetoothConnection.startClient(mBTDevice, MY_UUID_INSECURE);
371.            myProperties.setBluetoothConnected(true);
372.        }
373.
374.        public void setFlag1(boolean flag){
375.            myProperties.setPreset1Flag(flag);
376.        }
377.
378.        public void setFlag2(boolean flag){
379.            myProperties.setPreset2Flag(flag);
380.        }
381.
382.        public void setFlag3(boolean flag){
383.            myProperties.setPreset3Flag(flag);
384.        }
385.
386.        public void setFlag4(boolean flag){
387.            myProperties.setPreset4Flag(flag);
388.        }
389.
390.        private class Properties{
391.            byte action;
392.            boolean bluetoothConnected;
393.            boolean preset1Flag;
394.            boolean preset2Flag;
395.            boolean preset3Flag;
396.            boolean preset4Flag;
397.
398.            public Properties(){
399.                action = (byte)255;
400.                bluetoothConnected = false;
401.                preset1Flag = false;
402.                preset2Flag = false;
403.                preset3Flag = false;
404.                preset4Flag = false;
405.            }
```

```
406.
407.            public void setAction(byte action) {
408.                this.action = action;
409.            }
410.
411.            public void setBluetoothConnected(boolean bluetoothConnected) {
412.                this.bluetoothConnected = bluetoothConnected;
413.            }
414.
415.            public void setPreset1Flag(boolean preset1Flag) {
416.                this.preset1Flag = preset1Flag;
417.            }
418.
419.            public void setPreset2Flag(boolean preset2Flag) {
420.                this.preset2Flag = preset2Flag;
421.            }
422.
423.            public void setPreset3Flag(boolean preset3Flag) {
424.                this.preset3Flag = preset3Flag;
425.            }
426.
427.            public void setPreset4Flag(boolean preset4Flag) {
428.                this.preset4Flag = preset4Flag;
429.            }
430.
431.            public byte getAction() {
432.                return action;
433.            }
434.
435.            public boolean isBluetoothConnected() {
436.                return bluetoothConnected;
437.            }
438.
439.            public boolean isPreset1Flag() {
440.                return preset1Flag;
441.            }
442.
443.            public boolean isPreset2Flag() {
444.                return preset2Flag;
445.            }
446.
447.            public boolean isPreset3Flag() {
448.                return preset3Flag;
449.            }
450.
451.            public boolean isPreset4Flag() {
452.                return preset4Flag;
453.            }
454.        }
455.    }
```

A.2.2. SettingsActivity.java

```
1.  package com.air.karlo.airmanagement;
2.
3.  import android.Manifest;
4.  import android.app.Activity;
5.  import android.bluetooth.BluetoothAdapter;
6.  import android.bluetooth.BluetoothDevice;
7.  import android.content.BroadcastReceiver;
8.  import android.content.Context;
9.  import android.content.Intent;
```

```
10. import android.content.IntentFilter;
11. import android.os.Build;
12. import android.support.v7.app.AppCompatActivity;
13. import android.os.Bundle;
14. import android.util.Log;
15. import android.view.View;
16. import android.view.Window;
17. import android.view.WindowManager;
18. import android.widget.AdapterView;
19. import android.widget.Button;
20. import android.widget.ListView;
21.
22. import java.util.ArrayList;
23. import java.util.UUID;
24.
25. public class SettingsActivity extends AppCompatActivity implements AdapterView.
    OnItemClickListener{
26.     final String TAG = "mojTag";
27.
28.     BluetoothAdapter mBluetoothAdapter;
29.     BluetoothDevice mBTDevice;
30.
31.     public ArrayList<BluetoothDevice> mBTDevices = new ArrayList<>();
32.     public DeviceListAdapter mDeviceListAdapter;
33.     ListView lvNewDevices;
34.
35.     private BroadcastReceiver mBroadcastReciever = new BroadcastReceiver() {
36.         @Override
37.         public void onReceive(Context context, Intent intent) {
38.             final String action = intent.getAction();
39.
40.             if(action.equals(BluetoothDevice.ACTION_FOUND)){
41.                 BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.
                                                                        EXTRA_DEVICE);
42.                 mBTDevices.add(device);
43.                 mDeviceListAdapter = new DeviceListAdapter(context, R.layout.
                                                    device_adapter_view, mBTDevices);
44.                 lvNewDevices.setAdapter(mDeviceListAdapter);
45.             }
46.         }
47.     };
48.
49.     private final BroadcastReceiver mBroadcastReceiver1 = new BroadcastReceiver() {

50.         public void onReceive(Context context, Intent intent) {
51.             String action = intent.getAction();
52.             if (action.equals(mBluetoothAdapter.ACTION_STATE_CHANGED)) {
53.                 final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
                                                         mBluetoothAdapter.ERROR);
54.
55.                 switch(state){
56.                     case BluetoothAdapter.STATE_OFF:
57.                         Log.d(TAG, "onReceive: STATE OFF");
58.                         break;
59.                     case BluetoothAdapter.STATE_TURNING_OFF:
60.                         Log.d(TAG, "mBroadcastReceiver1: STATE TURNING OFF");
61.                         break;
62.                     case BluetoothAdapter.STATE_ON:
63.                         Log.d(TAG, "mBroadcastReceiver1: STATE ON");
64.                         break;
65.                     case BluetoothAdapter.STATE_TURNING_ON:
66.                         Log.d(TAG, "mBroadcastReceiver1: STATE TURNING ON");
67.                         break;
68.                 }
69.             }
70.         }
```

```
71.     };
72.
73.     private final BroadcastReceiver mBroadcastReceiver2 = new BroadcastReceiver() {

74.
75.         @Override
76.         public void onReceive(Context context, Intent intent) {
77.             final String action = intent.getAction();
78.
79.             if (action.equals(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED)) {
80.
81.                 int mode = intent.getIntExtra(BluetoothAdapter.EXTRA_SCAN_MODE,
                                                  BluetoothAdapter.ERROR);
82.
83.                 switch (mode) {
84.                     case BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE:
85.                         Log.d(TAG, "mBroadcastReceiver2: Discoverability Enabled.");
86.                         break;
87.                     case BluetoothAdapter.SCAN_MODE_CONNECTABLE:
88.                         Log.d(TAG, "mBroadcastReceiver2: Discoverability Disabled.
                                        Unable to receive connections.");
89.                         break;
90.                     case BluetoothAdapter.SCAN_MODE_NONE:
91.                         Log.d(TAG, "mBroadcastReceiver2: Discoverability Disabled.
                                        Not able to receive connections.");
92.                         break;
93.                     case BluetoothAdapter.STATE_CONNECTING:
94.                         Log.d(TAG, "mBroadcastReceiver2: Connecting....");
95.                         break;
96.                     case BluetoothAdapter.STATE_CONNECTED:
97.                         Log.d(TAG, "mBroadcastReceiver2: Connected.");
98.                         break;
99.                 }
100.
101.                }
102.            }
103.        };
104.
105.        private final BroadcastReceiver mBroadcastReceiver3 = new BroadcastReceiver()
    {
106.            @Override
107.            public void onReceive(Context context, Intent intent) {
108.                final String action = intent.getAction();
109.
110.                if(action.equals(BluetoothDevice.ACTION_BOND_STATE_CHANGED)){
111.                    BluetoothDevice mDevice = intent.getParcelableExtra(
                                                  BluetoothDevice.EXTRA_DEVICE);
112.                    if (mDevice.getBondState() == BluetoothDevice.BOND_BONDED){
113.                        Log.d(TAG, "BroadcastReceiver: BOND_BONDED.");
114.                        mBTDevice = mDevice;
115.                    }
116.
117.                    if (mDevice.getBondState() == BluetoothDevice.BOND_BONDING) {
118.                        Log.d(TAG, "BroadcastReceiver: BOND_BONDING.");
119.                    }
120.
121.                    if (mDevice.getBondState() == BluetoothDevice.BOND_NONE) {
122.                        Log.d(TAG, "BroadcastReceiver: BOND_NONE.");
123.                    }
124.                }
125.            }
126.        };
127.
128.        @Override
129.        protected void onCreate(Bundle savedInstanceState) {
130.            super.onCreate(savedInstanceState);
```

```
131.
132.           requestWindowFeature(Window.FEATURE_NO_TITLE);
133.           getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                                    WindowManager.LayoutParams.FLAG_FULLSCREEN);
134.
135.           setContentView(R.layout.activity_settings);
136.           Button buttonDiscover = (Button) findViewById(R.id.find_devices);
137.
138.           buttonDiscover.setOnClickListener((v) ->{
139.               btnDiscover();
140.           });
141.
142.           lvNewDevices = (ListView) findViewById(R.id.listDevices);
143.
144.           IntentFilter filter = new IntentFilter(BluetoothDevice.
                                                      ACTION_BOND_STATE_CHANGED);
145.           registerReceiver(mBroadcastReceiver3, filter);
146.
147.           mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
148.           lvNewDevices.setOnItemClickListener(SettingsActivity.this);
149.
150.       }
151.
152.       public void btnDiscover(){
153.           if(mBluetoothAdapter.isDiscovering()){
154.               mBluetoothAdapter.cancelDiscovery();
155.               mBluetoothAdapter.startDiscovery();
156.               IntentFilter discoverDevicesIntent = new IntentFilter(BluetoothDevice
                                                                        .ACTION_FOUND);
157.               registerReceiver(mBroadcastReciever, discoverDevicesIntent);
158.           }
159.           if(!mBluetoothAdapter.isDiscovering()) {
160.               mBluetoothAdapter.startDiscovery();
161.               IntentFilter discoverDevicesIntent = new IntentFilter(BluetoothDevice
                                                                        .ACTION_FOUND);
162.               registerReceiver(mBroadcastReciever, discoverDevicesIntent);
163.           }
164.       }
165.
166.       @Override
167.       protected void onDestroy() {
168.           super.onDestroy();
169.           unregisterReceiver(mBroadcastReciever);
170.           unregisterReceiver(mBroadcastReceiver3);
171.       }
172.
173.       @Override
174.       public void onItemClick(AdapterView<?> parent, View view, int i, long id) {
175.           mBluetoothAdapter.cancelDiscovery();
176.
177.           String deviceName = mBTDevices.get(i).getName();
178.           String deviceAddress = mBTDevices.get(i).getAddress();
179.           Log.i(TAG, deviceAddress);
180.
181.           if(Build.VERSION.SDK_INT > Build.VERSION_CODES.JELLY_BEAN_MR2){
182.               mBTDevices.get(i).createBond();
183.
184.               Intent intent = new Intent();
185.               intent.putExtra("device_address", deviceAddress);
186.
187.               setResult(Activity.RESULT_OK, intent);
188.               finish();
189.           }
190.       }
191.   }
```

A.2.3. BluetoothConnectionService.java

```
1.  package com.air.karlo.airmanagement;
2.
3.  import android.annotation.SuppressLint;
4.  import android.app.ProgressDialog;
5.  import android.bluetooth.BluetoothAdapter;
6.  import android.bluetooth.BluetoothDevice;
7.  import android.bluetooth.BluetoothServerSocket;
8.  import android.bluetooth.BluetoothSocket;
9.  import android.content.Context;
10. import android.os.Handler;
11. import android.os.SystemClock;
12. import android.renderscript.ScriptGroup;
13. import android.util.Log;
14. import android.widget.TextView;
15.
16. import org.w3c.dom.Text;
17.
18. import java.io.IOException;
19. import java.io.InputStream;
20. import java.io.OutputStream;
21. import java.nio.charset.Charset;
22. import java.util.UUID;
23.
24. public class BluetoothConnectionService {
25.
26.     private static final String TAG = "mojTag";
27.     private static final String appName = "AirManagement";
28.     private static final UUID MY_UUID_INSECURE =
29.             UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
30.
31.     byte[] bufferNew;
32.
33.     private final BluetoothAdapter mBluetoothAdapter;
34.     Context mContext;
35.     private ConnectThread mConnectThread;
36.     private BluetoothDevice mmDevice;
37.     private UUID deviceUUID;
38.     ProgressDialog mProgressDialog;
39.     private ConnectedThread mConnectedThread;
40.     private AcceptThread mInsecureAcceptThread;
41.
42.     protected MainActivity context;
43.
44.     private final static int MESSAGE_READ = 2;
45.
46.
47.     @SuppressLint("HandlerLeak")
48.      private Handler mHandler = new Handler(){
49.         public void handleMessage(android.os.Message msg){
50.             if(msg.what == MESSAGE_READ){
51.                 String readMessage;
52.                 readMessage = new String((byte[]) msg.obj);
53.                 readMessage.replaceAll("\\s","");
54.                 if(readMessage.substring(0,2).equals("00")) displayMsg(readMessage);

55.                 if(readMessage.substring(0,2).equals("02")) odustanak(readMessage);

56.
57.             }
58.         }
```

```
59.     };
60.
61.     public void odustanak(String msg){
62.         String[] data = msg.split("/");
63.         TextView status = (TextView) context.findViewById(R.id.status);
64.
65.         if(data[1].equals("1")) {
66.             context.setFlag1(false);
67.             status.setText("");
68.         }
69.         if(data[1].equals("2")) {
70.             context.setFlag2(false);
71.             status.setText("");
72.         }
73.         if(data[1].equals("3")) {
74.             context.setFlag3(false);
75.             status.setText("");
76.         }
77.         if(data[1].equals("4")) {
78.             context.setFlag4(false);
79.             status.setText("");
80.         }
81.     }
82.
83.     public void displayMsg( final String msg){
84.         context.runOnUiThread(new Runnable() {
85.
86.             @Override
87.             public void run() {
88.                 String[] data = msg.split("/");
89.
90.                 TextView frontLeft = (TextView) context.findViewById(R.id.
                                                                    pressureFrontLeft);
91.                 TextView frontRight = (TextView) context.findViewById(R.id.
                                                                    pressureFrontRight);
92.                 TextView rearLeft = (TextView) context.findViewById(R.id.
                                                                    pressureRearLeft);
93.                 TextView rearRight = (TextView) context.findViewById(R.id.
                                                                    pressureRearRight);
94.                 TextView tank = (TextView) context.findViewById(R.id.tankPressure);

95.
96.                 frontLeft.setText(data[1]);
97.                 frontRight.setText(data[2]);
98.                 rearLeft.setText(data[3]);
99.                 rearRight.setText(data[4]);
100.                tank.setText(data[5] + "bar");
101.               }
102.            });
103.        }
104.
105.        public BluetoothConnectionService(Context context, MainActivity contextMain){

106.            this.mContext = context;
107.            this.context = contextMain;
108.            this.mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
109.            start();
110.        }
111.
112.        private class AcceptThread extends Thread {
113.            private final BluetoothServerSocket mmServerSocket;
114.
115.            public AcceptThread(){
116.                BluetoothServerSocket tmp = null;
117.
118.                try{
```

```
119.                 tmp = mBluetoothAdapter.
     listenUsingInsecureRfcommWithServiceRecord(appName, MY_UUID_INSECURE);
120.
121.                 Log.d(TAG, "AcceptThread: Setting up Server using: " +
                                                        MY_UUID_INSECURE);
122.             }catch (IOException e){
123.                 Log.e(TAG, "AcceptThread: IOException: " + e.getMessage() );
124.             }
125.
126.             mmServerSocket = tmp;
127.         }
128.
129.         public void run(){
130.             Log.d(TAG, "run: AcceptThread Running.");
131.
132.             BluetoothSocket socket = null;
133.
134.             try{
135.                 Log.d(TAG, "run: RFCOM server socket start.....");
136.                 socket = mmServerSocket.accept();
137.                 Log.d(TAG, "run: RFCOM server socket accepted connection.");
138.
139.             }catch (IOException e){
140.                 Log.e(TAG, "AcceptThread: IOException: " + e.getMessage() );
141.             }
142.
143.             if(socket != null){
144.                 connected(socket,mmDevice);
145.             }
146.
147.             Log.i(TAG, "END mAcceptThread ");
148.         }
149.
150.         public void cancel() {
151.             Log.d(TAG, "cancel: Canceling AcceptThread.");
152.             try {
153.                 mmServerSocket.close();
154.             } catch (IOException e) {
155.                 Log.e(TAG, "cancel: Close of AcceptThread ServerSocket failed. "
                                                            + e.getMessage() );
156.             }
157.         }
158.
159.     }
160.
161.
162.     private class ConnectThread extends Thread {
163.         private BluetoothSocket mmSocket;
164.
165.         public ConnectThread(BluetoothDevice device, UUID uuid) {
166.             Log.d(TAG, "Bluetooth Thread: started.");
167.             mmDevice = device;
168.             deviceUUID = uuid;
169.         }
170.
171.         public void run() {
172.             BluetoothSocket tmp = null;
173.             Log.i(TAG, "RUN mConnectThread");
174.             try {
175.                 Log.d(TAG, "ConnectThread: Trying to create InsecureRfcommSocket
                                            using UUID: " + MY_UUID_INSECURE);
176.                 tmp = mmDevice.createRfcommSocketToServiceRecord(deviceUUID);
177.             } catch (IOException e) {
178.                 Log.e(TAG, "ConnectThread: could not create insecure
                                            RfCommSocket " + e.getMessage());
179.             }
```

```
180.
181.                mmSocket = tmp;
182.
183.                mBluetoothAdapter.cancelDiscovery();
184.
185.                try {
186.                    mmSocket.connect();
187.                    Log.d(TAG, "run: Conenction established");
188.                } catch (IOException e) {
189.                    try {
190.                        mmSocket.close();
191.                        Log.d(TAG, "run: closed socket");
192.                    } catch (IOException e1) {
193.                        Log.e(TAG, "unable to close connection in socket " + e1.
                                                                        getMessage());
194.                    }
195.                }
196.
197.                connected(mmSocket,mmDevice);
198.            }
199.
200.            public void cancel(){
201.                try{
202.                    Log.d(TAG,"cancel: Closing Client Socket");
203.                    mmSocket.close();
204.                } catch(IOException e) {
205.                    Log.e(TAG, "cancel: close() failed " + e.getMessage());
206.                }
207.            }
208.        }
209.
210.        public synchronized void start() {
211.            Log.d(TAG, "start");
212.
213.            if (mConnectThread != null) {
214.                mConnectThread.cancel();
215.                mConnectThread = null;
216.            }
217.            if (mInsecureAcceptThread == null) {
218.                mInsecureAcceptThread = new AcceptThread();
219.                mInsecureAcceptThread.start();
220.            }
221.        }
222.
223.
224.        public void startClient(BluetoothDevice device, UUID uuid){
225.            Log.d(TAG,"startClient: started");
226.
227.            mProgressDialog = ProgressDialog.show(mContext,"Connecting Bluetooth",
                                                        "Please Wait...",true);
228.            mConnectThread = new ConnectThread(device, uuid);
229.            mConnectThread.start();
230.        }
231.
232.        private class ConnectedThread extends Thread{
233.            private BluetoothSocket mmSocket;
234.            private InputStream inputStream;
235.            private OutputStream outputStream;
236.
237.            public ConnectedThread(BluetoothSocket socket) {
238.                Log.d(TAG,"ConnectedThread: starting");
239.
240.                mmSocket = socket;
241.                InputStream tmpIn = null;
242.                OutputStream tmpOut = null;
243.
```

```
244.                mProgressDialog.dismiss();
245.
246.                try {
247.                    tmpIn = mmSocket.getInputStream();
248.                    tmpOut = mmSocket.getOutputStream();
249.                } catch (IOException e) {
250.                    e.printStackTrace();
251.                }
252.
253.                inputStream = tmpIn;
254.                outputStream = tmpOut;
255.
256.            }
257.
258.            public void run(){
259.                byte[] buffer = new byte[1024];
260.                int bytes;
261.
262.                new Thread(
263.                        new Runnable() {
264.                            @Override
265.                            public void run() {
266.                                while(true) {
267.                                    SystemClock.sleep(1000);
268.                                    writeByte((byte) 255);
269.                                }
270.                            }
271.                        }
272.                ).start();
273.
274.
275.                while(true){
276.                    try {
277.                        bytes = inputStream.available();
278.                        if(bytes != 0) {
279.                            SystemClock.sleep(100);
280.
281.                            bytes = inputStream.available();
282.                            bytes = inputStream.read(buffer, 0, bytes);
283.                            String incomingMessage = new String(buffer, 0, bytes);
284.                            Log.i(TAG, "InputStream: " + incomingMessage);
285.                            mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer)
286.                                    .sendToTarget();
287.                        }
288.                    } catch (IOException e) {
289.                        e.printStackTrace();
290.                    }
291.
292.                }
293.
294.            }
295.
296.            public void write(byte[] bytes){
297.                String text = new String(bytes, Charset.defaultCharset());
298.                Log.d(TAG,"Writing to output: " + text);
299.                try {
300.                    outputStream.write(bytes);
301.                } catch (IOException e) {
302.                    e.printStackTrace();
303.                }
304.            }
305.
306.            public void cancel(){
307.                try {
308.                    mmSocket.close();
309.                } catch (IOException e) {
```

```
                    e.printStackTrace();
                }
            }
        }

        private void connected(BluetoothSocket mmSocket, BluetoothDevice mmDevice){
            Log.d(TAG,"Connected: starting");

            mConnectedThread = new ConnectedThread(mmSocket);
            mConnectedThread.start();
        }

        public void write(byte[] out){
            mConnectedThread.write(out);
        }

        public void writeByte(byte out){
            byte[] tmp = new byte[1];
            tmp[0] = out;
            mConnectedThread.write(tmp);
        }

    }
```

A.2.4. DeviceListAdapter.java

```
package com.air.karlo.airmanagement;

import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.ArrayList;

public class DeviceListAdapter extends ArrayAdapter<BluetoothDevice> {

    private LayoutInflater mLayoutInflater;
    private ArrayList<BluetoothDevice> mDevices;
    private int mViewResourceId;

    public DeviceListAdapter(Context context, int resourceId,
                                            ArrayList<BluetoothDevice> devices) {
        super(context, resourceId, devices);
        this.mDevices = devices;
        mLayoutInflater = (LayoutInflater) context.getSystemService(Context.
                                                        LAYOUT_INFLATER_SERVICE);
        mViewResourceId = resourceId;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        convertView = mLayoutInflater.inflate(mViewResourceId, null);

        BluetoothDevice device = mDevices.get(position);

        if (device != null) {
            TextView deviceName = (TextView) convertView.findViewById(R.id.deviceName
    );

            if (deviceName != null) {
```

```
35.                 deviceName.setText(device.getName());
36.             }
37.         }
38.
39.         return convertView;
40.
41.     }
42. }
```