Izrada uređaja i programske podrške za mjerenje parametara treninga za ugradnju u hokej pak

Mustapić, Sandro

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij

IZRADA UREĐAJA I PROGRAMSKE PODRŠKE ZA MJERENJE PARAMETARA TRENINGA ZA UGRADNJU U HOKEJ PAK

Diplomski rad

Sandro Mustapić

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 11.07.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Sandro Mustapić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 723 R, 14.10.2014.
OIB studenta:	28647131524
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	Dino Kurtagić
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Nenadić
Član Povjerenstva:	Dino Kurtagić
Naslov diplomskog rada:	Izrada uređaja i programske podrške za mjerenje parametara treninga za ugradnju u hokej pak
Znanstvena grana rada:	**Arhitektura računalnih sustava (zn. polje računarstvo)**
Zadatak diplomskog rada:	U radu je potrebno osmisliti i izraditi IoT uređaj koji se može integrirati u hokej pak, a može mjeriti brzinu udarca, silu udaranja i putanju pomaka. Podatke treba sinkronizirati na mobilni uređaj uz pomoć bluetootha ili testirati na računalu serijskom vezom. Podatke je potrebno spremiti u formatu prikladnom za daljnju analitiku i izradu mobilne aplikacije za detaljno praćenje treninga i davanje savjeta. (Dino Kurtagić)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	11.07.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 19.07.2017.

Ime i prezime studenta:	Sandro Mustapić
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 723 R, 14.10.2014.
Ephorus podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada uređaja i programske podrške za mjerenje parametara treninga za ugradnju u hokej pak**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora Dino Kurtagić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

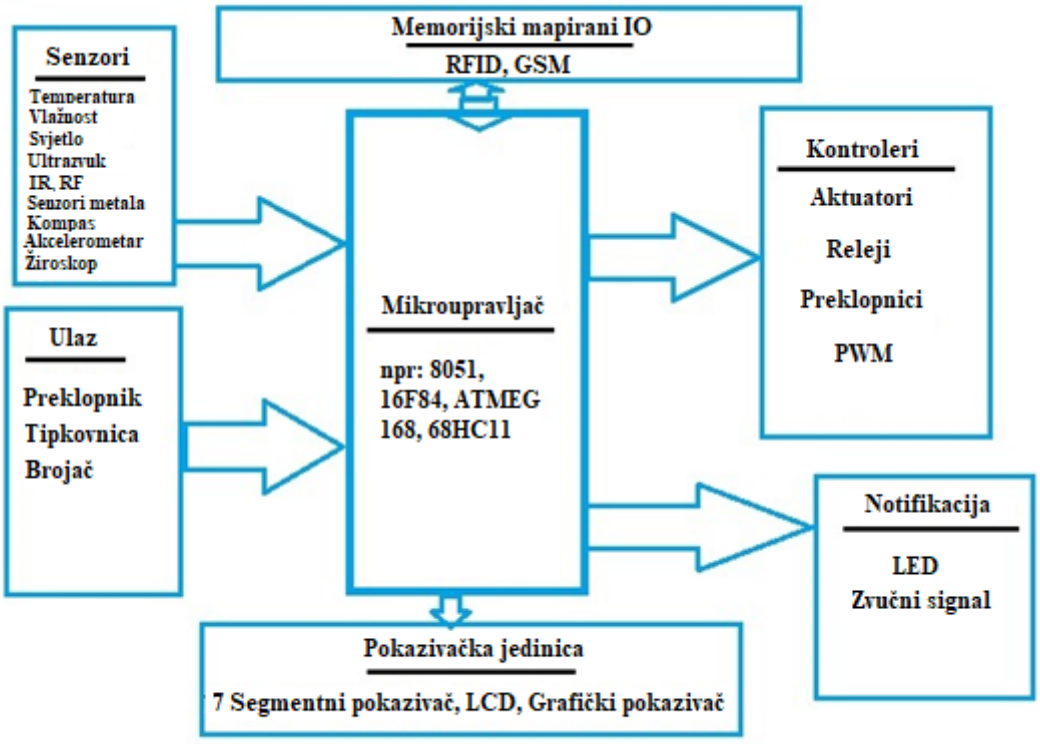
Današnja tehnologija, posebno elektronika je drastično napredovala te se projekti koji su nekada bili kompliciranі sada se mogu dizajnirati i implementirati kod kuće. Već ogromna količina podataka koja se dnevno procesira postaje još veća razvojem uređaja u Internetu stvari (engl. *Internet of things, IoT*). Koncept IoT-a nije nov, ali postaje sve popularniji zbog tri glavna razloga, a to su: cijena senzora koja je drastično pala, performanse računala su se uvelike povećale te uvođenje Internet protokola verzije 6 (Ipv6) koji je povećanjem IP broja s 32 bit-a na 128 bit-a omogućio 2^{128} IP adresa omogučavajući „neograničenom“ broju ljudi, podataka ili uređaja spajanje preko Interneta.

Cilj je ovog diplomskog rada osmisliti i izraditi IoT uređaj za ugradnju u hokej pak koji može mjeriti brzinu udarca, silu udaranja i putanju pomaka. Dobivene podatke treba sinkronizirati na mobilni uređaj te upotrijebiti za izradu mobilne aplikacije kojom bi se detaljno pratili treninzi i davali savjeti.

U poglavlju 2 bit će opisani ugradbeni računalni sustavi u sportu, navest će se postojeći sustavi. U poglavlju 3 opisano je idejno rješenje IoT sustava za ugradnju u hokej pak kao i razlozi mjerenja u sportu te pripadajuća problematika. U poglavlju 4 detaljno će biti razrađeno sklopovsko i programsko rješenje sustava gdje će se prikazati i opisati korišteno sklopovlje, programske tehnologije i pripadajuća programska rješenja. Na kraju slijedi testiranje razvijenog sustava i upute za korištenje aplikacije.

2. UGRADBENI RAČUNALNI SUSTAVI U SPORTU

Sustav koji se sastoji od računalnog sklopovlja (engl. *hardware*), programske podrške (engl. *software*) te potrebnih dodatnih dijelova u svrhu obavljanja neke zadaće [1]. Središnja komponenta ugradbenog sustava je mikroupravljač , što će u ovom radu biti ATMega 2560. Za razliku od stolnog računala, mikrokoupravljač u ugradbenom sustavu izvodi jedan program u beskonačnoj petlji. Prema Sl. 2.1 vidi se da je moguće spojiti i senzore kao što su npr. senzori za toplinu, akcelerometar, žiroskop, itd.



Sl. 2.1. *Shematski prikaz osnovnog ugradbenog sustava[2]*

Područje primjene ugradbenih računalnih sustava je vrlo veliko što se može vidjeti iz Tab. 2.2.

Tablica 2.2. *Primjeri primjene ugradbenih sustava[1]*

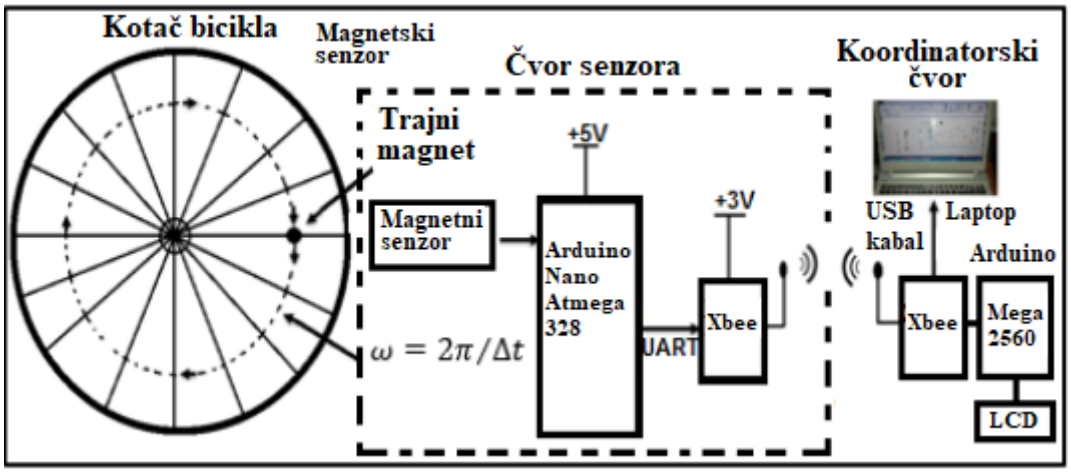
PODRUČJE PRIMJENE	PRIMJER
Avionska i vojna industrija	Automatski sustavi navođenja, navigacijski sustavi, automatski sustavi za slijetanje, upravljanje motorima
Medicina	X-zrake, MRI, uređaji za praćenje (monitoring) stanja pacijenta
Automobilska industrija	Upravljanje motorom, sustav protiv blokiranja kotača prilikom kočenja (ABS), sustav protiv prklizavanja kotača, GPS navigacija…
Komunikacije	Komnikacijski sateliti, routeri, switchevi, hubovi…

Ugradbeni računalni sustavi veliku primjenu nalaze i u sportu. Tehnologije za nagledanje daju veliki uvid u izvedbu sportaša te omogućuju bolji i brži napredak jer lakše i preciznije otkrivaju greške. Današnji sustavi i aplikacije nisu ograničeni samo na profesionalne sportaše nego omogućuju i rekreativcima praćenje i poboljšanje.

U sljedećem dijelu bit će dani primjeri nekih ugradbenih sustava u sportu te njihovi opisi. Na kraju će se uz *miCoachSoccerBall* spomenuti i ugradbeni računalni sustav kojeg ovaj rad rješava jer je iz njega nastala i ideja za dizajniranjem sličnog sustava.

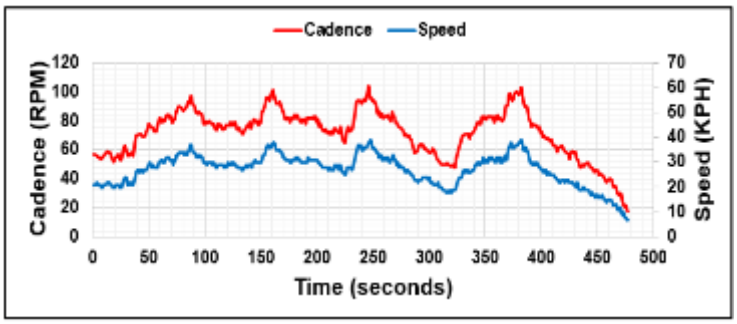
2.1. Arduino bežični senzor za praćenje vožnje biciklom po stazi

U Maleziji, vožnja biciklom po stazi je jedan od sportova koji je dobio velik interes u zadnjih nekoliko godina. Iz tog razloga Malezijsko vijeće sporta je odlučilo napraviti ovaj projekt zbog razvitka izvedbe i talenta lokalnih biciklista. Na Sl. 2.2 prikazan je korišteni sustav za realizaciju koji se može razdijeliti na senzorski i koordinatorski dio. Senzorski dio mjeri bitne parametre,a to su broj okretaja u minuti (engl. *revolutions per minute, RPM*) i brzinu. Senzorski dio se sastoji od „Zigbee“ primopredajnika, ATmega 328 mikroupravljača i magnetometra. Koordinacijski dio prima podatke o broju okretaja i brzini. Sastoji se od „Zigbee“ primopredajnika, AT Mega 2560 mikroupravljača te LCD-a na kojem se prikazuju podatci. [3]



Sl. 2.2. *Shematski prikaz sustava za praćenje bicikla[3]*

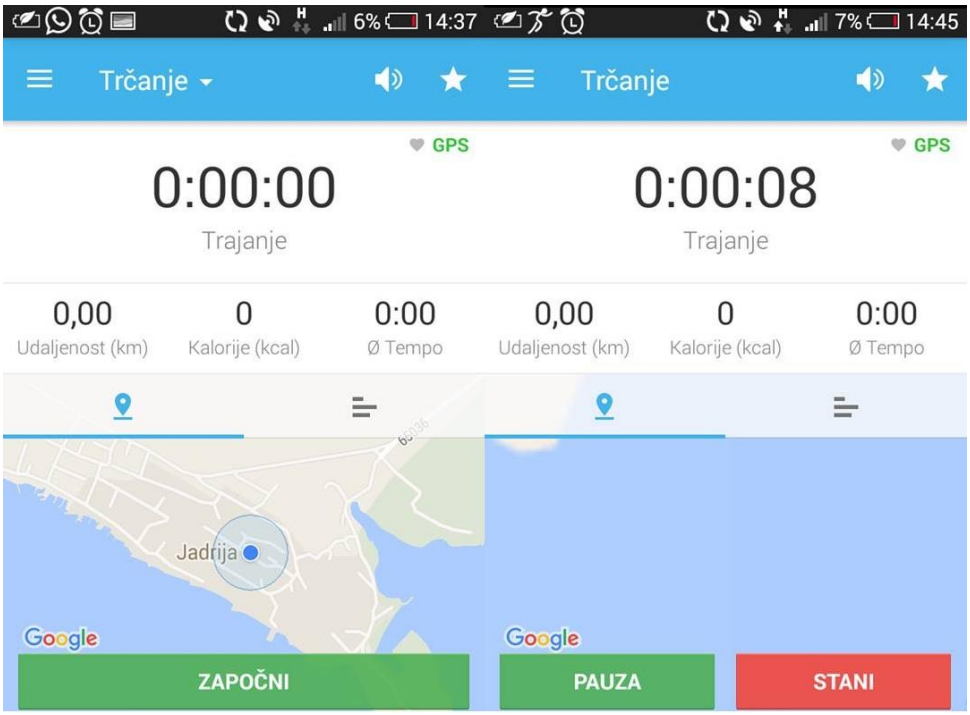
Izmjereni podatci prenose se bežično od senzorskog dijela do koordinatorskog koji je spojen s glavnim mikroupravljačem AT Mega 2560. Mikroupravljač je smješten ispod sjedala bicikla kako bi smanjio aerodinamični otpor. Kada podatci dođu do mikroupravljača oni se obrađuju po unaprijed napisanim algoritmima u C programskom jeziku te prikazuju na *LCD-u*. Podatci se mogu preko sustava poslati trenerskom timu kako bi mjerili broj okretaja i brzinu u stvarnom vremenu čiji se prikaz može vidjeti na Sl. 2.3 [3].



Sl. 2.3. *Prikaz brzine i broja okretaja bicikla u stvarnom vremenu[3]*

2.2. Aplikacija FITAPP GPS

FITAPP GPS je aplikacija koja putem GPS-a prati aktivnosti korisnika te ima mogućnost čuvanja radi kasnije usporedbe. Kompatibilna je s raznim sportskim aktivnostima kao što su trčanje, biciklizam, skijanje i brojne druge. Posjeduje glasovno navođenje ovisno o postavkama tako da može npr. obavještavati korisnika svakih 500 metara o vremenu obavljanja aktivnosti, potrošenim kalorijama, udaljenosti, tempu. Ima ugrađeni BMI (indeks tjelesne mase) kalkulator koji ovisno o unesenoj težini i visini računa BMI indeks i daje povratnu informaciju [4].



Sl. 2.4. *Prikaz sučelja FITAPP aplikacije*

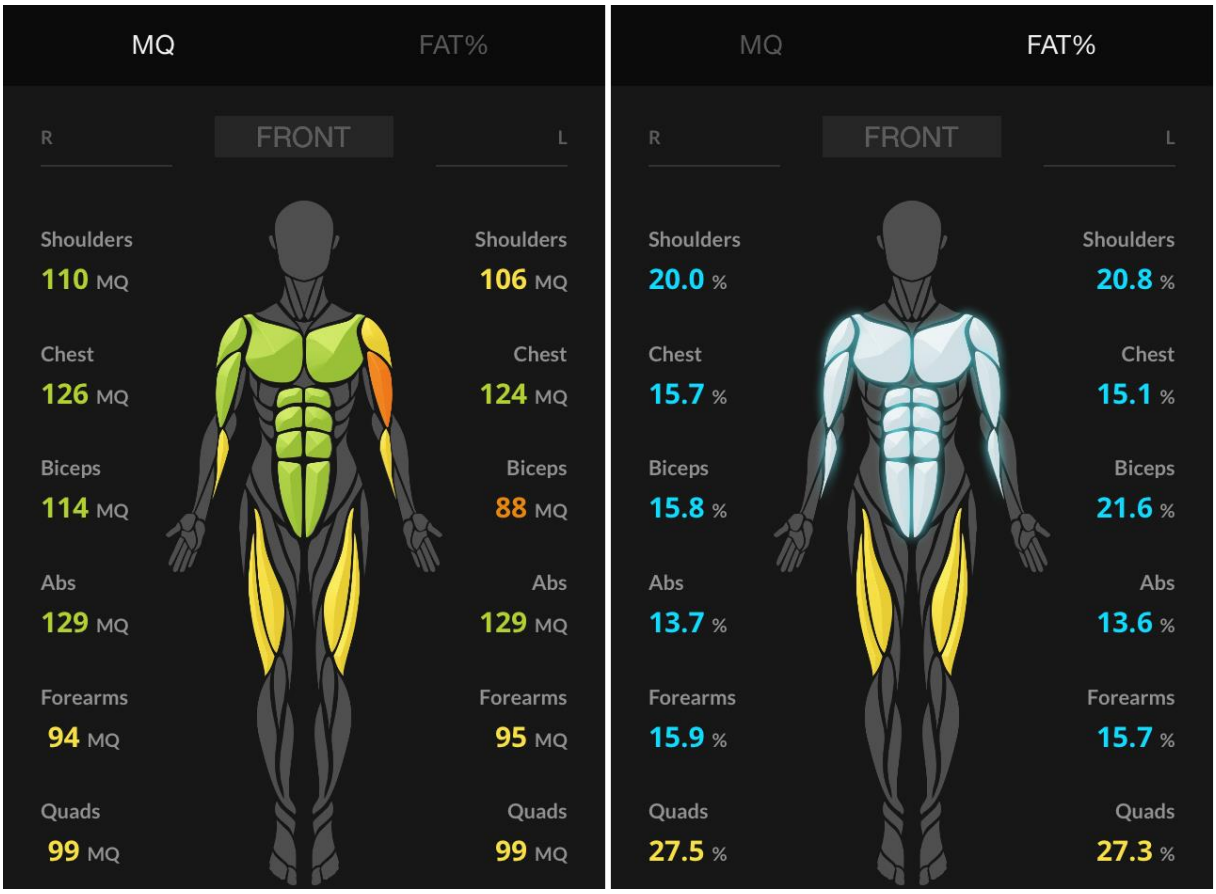
2.3. Aplikacija Athos

Vježbanje može biti vrlo frustrirajuće jer je vrlo teško procijeniti koliko se dobro izvode neke vježbe tijekom treninga. Dan poslije treninga može se osjećati jako veliki umor što bi značilo da je treninga bilo previše, a vrijedi i suprotno. Zbog toga razloga je stvorena tehnologija koja može „zaviriti“ u ljudsko tijelo i analizirati aktivnost te dati točnu povratnu informaciju o broju potrebnih ili nepotrebnih ponavljanja.

Prema [5], Athos je aplikacija koja koristi odjeću s ugrađenim senzorima koji mogu osjetiti aktivnosti unutar mišičnih vlakana te pratiti trening. Athos odjeća koristi tehnologiju koja se zove elektromiografija (engl. *electromyography, EMG*). Pri svakom pokretu ljudskog tijela kontrakcije mišića šalju električni signal koji onda može biti zabilježen i izmjeren. Budući da su uređaji za EMG glomazni i skupi (od 5 do 15 tisuća dolara) kompanija Athos je napravila novi mikro EMG senzor koji je otporan na vodu. Athos nije jedina tvrtka koja je započela raditi s EMG senzorima.

Aplikacija omogućuje sprječavanje ozljeda jer može unaprijed vidjeti koja se mišićna vlakna najviše naprežu i obavijestiti ako se slabost u tom dijelu može odraziti na ostale dijelove. Ima sposobnost mjerenja pulsa, kapaciteta pluća, udaha i izdaha, a sve to u stvarnom vremenu

kako bi na vrijeme spriječili neželjene promjene. Sl. 2.5 prikazuje sučelje mobilne aplikacije Athos.



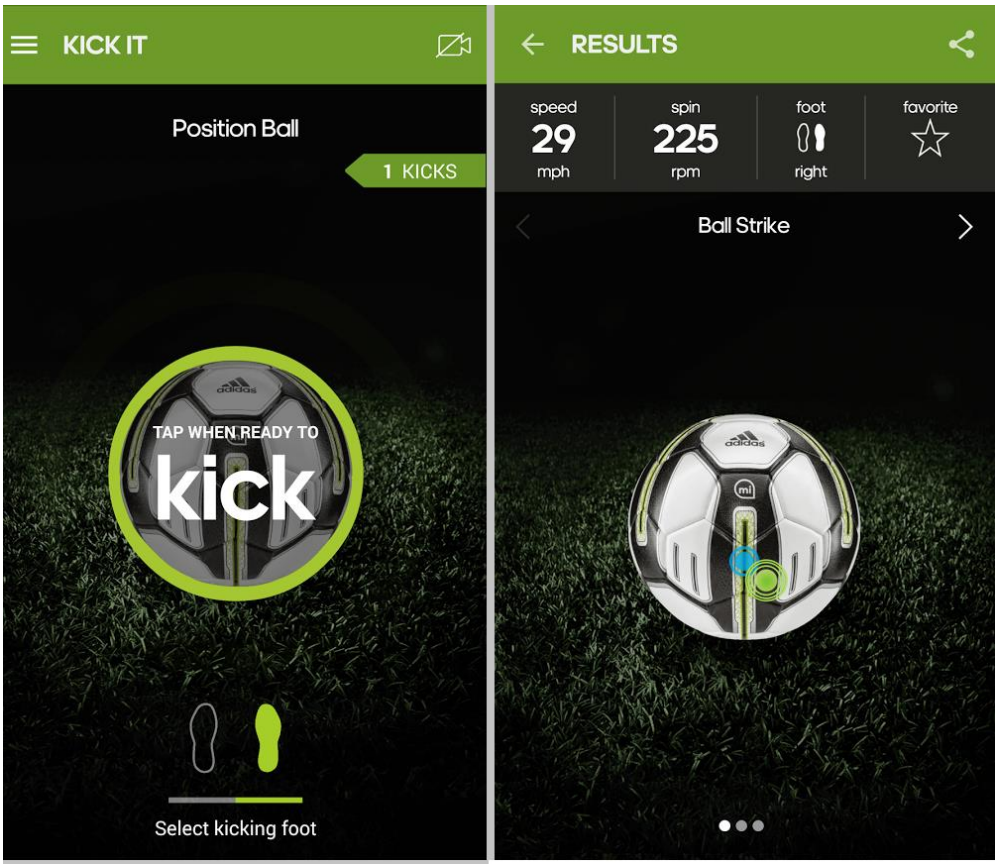
Sl. 2.5. *Prikaz sučelja mobilne aplikacije Athos [5]*

2.4. Adidas miCoach Soccer Ball

Nogomet je najpopularniji sport na svijetu te je bilo pitanje vremena kada će se tehnologija umiješati kako bi ga poboljšala, a upravo to je učinio projekt miCoach Soccer Ball. Iako je elektronika bila ključ za mjerenje pokreta signala, bio je ostao još jedan veliki izazov, a to je zadržavanje elektronike (senzora, mikroupravljača, bežičnog modula, baterije) u lopti. U ovom projektu je odlučeno da će sklop staviti u sredinu lopte te ga učvrstiti iznutra s 12 dijelova kako bi uvijek bio u samom centru (Sl. 2.6 prema [6]).

Sl. 2.6. *Adidas miCoach Smart Ball [7]*

Ugrađeni senzori mjere brzinu, brzinu vrtnje, snagu i putanju lopte te odmah prikazuju statistiku udarca na iOS ili Android aplikaciji koja je prikazana na Sl. 2.7.



Sl. 2.7. *Prikaz sučelja mobilne aplikacije miCoach Smart Ball [6]*

Upravo od ovog projekta je došla ideja dizajniranja sustava za potrebe hokeja. Budući da hokej još uvijek nema razvijene sustave koji bi mogli pratiti treninge i napredovanje sportaša te im na temelju rezultata pružiti upute za poboljšanje perfomansi ovaj rad će se baviti dizajniranjem i implementacijom sličnog sustava gdje će hokej pak preuzeti ulogu lopte.

3. IDEJNO RJEŠENJE IoT SUSTAVA ZA IMPLEMENTACIJU U HOKEJ PAK

Kod izrade svakog projekta bitno je proučiti sve čimbenike koji mogu utjecati pozitivno ili negativno na projekt. U prvom dijelu istražit će se problemi mjerenja u sportu općenito te zatim navesti neki od problema s kojima se susrećemo u ovom radu. Nadalje, opisat će se hokej kao sport, njegova pravila te fizika koja pomaže pri dizajniranju. Na kraju će se iznijeti ideja rješenja kao odgovor na potrebe u hokeju te mogućnosti nadogradnje projekta.

3.1. Problemi mjerenja u sportu

Kako bi se nešto poboljšalo prvo je potrebno shvatiti kako to funkcionira. Najbolji način razumijevanja stvari je da ih izmjerimo, ali za inženjere u sportu to zna biti „škakljivo“. Glavni problem mjerenja u sportu je taj što nije isto promatrati sportaše u kontroliranom okolišu i kontroliranim uvjetima ili ih promatrati „u akciji“ dok su pod najvećim opterećenjem. U isto vrijeme nije izvedivo mjerenje „u akciji“ tijekom važnih događaja jer se riskira promjena ishoda [8].

Objasnit će se neke od tehnologija mjerenja pokreta, a to su IMU (engl. *Intertial measurement unit*), elektromagnetski sustavi i optičko hvatanje pokreta.

3.1.1. IMU tehnologija

IMU se sastoji od mikoromehaničkih uređaja što je u osnovi akcelerometar, ali su se počeli koristiti i žiroskop te magnetometar zbog naglog razvoja. Akcelerometri se koriste za kutne rotacije te rade dobro na statičnim objektima dok na dinamičnim pokretima izlazi postaju sve složeniji te se javlja problem drifta pa ih je teško prikazati. Kako bi se riješio taj problem, koristi se spoj tehnologija akcelerometra, žiroskopa te magnetometra što je i slučaj u današnjim pametnim uređajima. Prednost ovih uređaja je njihova cijena i jednostavnost , ali može doći do problema u sinkroniziranju više uređaja [9].

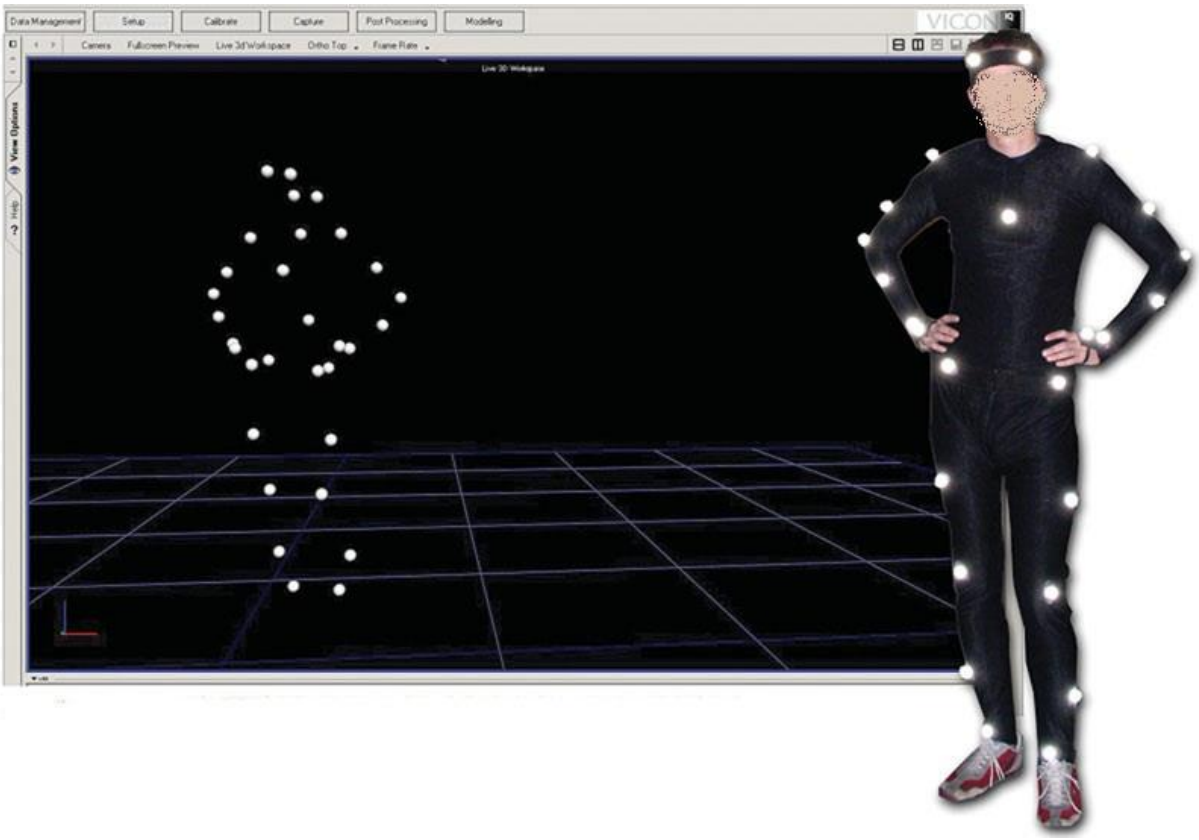
3.1.2. Elektromagnetski sustavi

Elektromagnetski sustavi koriste male senzorske zavojnice, unutar dobro definiranog magnetskog polja, za određivanje položaja i orijentacije. Jasna prednost tih sustava je što iz jednog senzora omogućuju prikaz položaja i orijentacije te imaju mogućnost sinkronizacije više senzora. Loša strana im je što bilo koji metal unutar tog magnetskog polja može izazvati greške u mjerenju pa su zbog toga ti sustavi ograničeni na mali broj senzora [9].

3.1.3. Optičko hvatanje pokreta

Izraz „optičko hvatanje pokreta“ se koristi za opis sustava koji prate aktivne ili pasivne oznake koji su postavljeni po tijelu (Sl. 3.1). Takvi sustavi većinom imaju seriju kamera te svaka od njih snima položaj oznake u 2D-u kao točke koje se kreću kroz prostor. Kroz početnu kalibraciju, sustav ima informaciju o položaju svake kamere te tako može od tih 2D točaka locirati iste u 3D prostoru. Budući da je svaka oznaka predstavljena kao samostalna točka u prostoru, potrebne su tri oznake na svakom segmentu da bi mogli saznati točan položaj u prostoru te iz tog razloga može biti jako veliki broj oznaka.

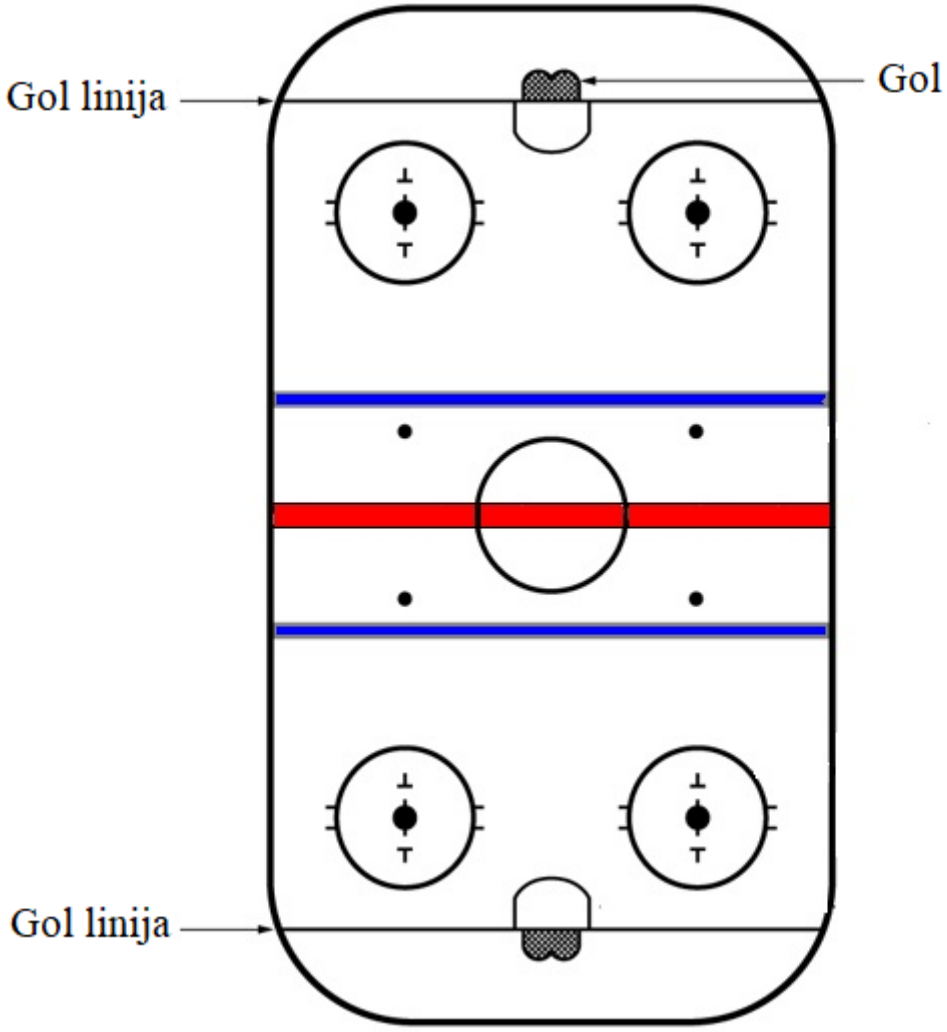
Ovakvi sustavi su vrlo skupi, ali točnost koja se dobije korištenjem oznaka i prilagodljivost čine ju najboljom tehnologijom za hvatanje pokreta. Tu se javlja problem korištenja u stvarnom scenariju jer je puno optičkih sustava osjetljivo na sunce pa mjerenje na otvorenom može biti problem. Voda, snijeg, led ili bilo kakvi blistavi predmeti mogu stvoriti odsjaj pa su ispitivanja u takvim uvjetima isključena. Također neki dijelovi sustava koji se koriste u zatvorenom laboratoriju nisu prenosivi. Takvo okruženje može biti znatno drugačije od sportaševa okruženja u kojemu trenira te ograničiti dobivene podatke [9].



Sl. 3.1. *Prikaz optičkog hvatanja pokreta [10]*

3.2. Hokej

Hokej je najpopularniji ekipni zimski sport, a najpopularniji je u Rusiji, Kanadi, SAD-u i Švedskoj. Igra se s dvije momčadi po šest igrača na klizaljkama. Hokej se igra na velikom komadu leda koji je dugačak 61 metar, a širok 26 metara. Golovi su široki 1.8 metara , a visoki 1.2 metar. Hokej se igra s pakom koji se pravi od topljene gume te je 2.5 centimetara debljine i promjera 7.6 centimetara. Težina mu je oko 170 grama. Pakovi se prije utakmice smrzavaju kako bi se smanjilo odskakivanje dok su u igri. Utakmica se dijeli na tri trećine od po dvadeset minuta [11]. Na Sl. 3.2 može se vidjeti prikaz hokej igrališta. Hokej igralište je raspodijeljeno na dva dijela crvenom linijom. Plava linija označava granicu obrambene zone svakog tima.



Sl. 3.2. *Prikaz hokej igrališta [12]*

3.2.1. Utjecaj trenja na hokej pak

Većina prvih susreta ljudi s ledom je vrlo vjerojatno završila s padom na led. Tim „eksperimentom“ zaključeno je da je led tvrd i da je klizak. Skliskost leda je povezana s svojstvom koje nazivamo trenje. Iako se trenje proučava već dugo vremena, još uvijek nije jasno kako točno ono funkcionira, barem ne na molekulskoj razini. Kada se dva čvrsta objekta dodiruju, mikroskopske razlike se trljaju jedna od drugu i stvaraju otpor pokretu – trenje. Eksperimentalno je dokazano da je sila trenja proporcionalna sa silom kontakta. Matematički zapisano to bi izgledalo ovako:

$$f = \mu * N \qquad (1)$$

gdje je:

f – sila trenja

μ - koeficijent trenja

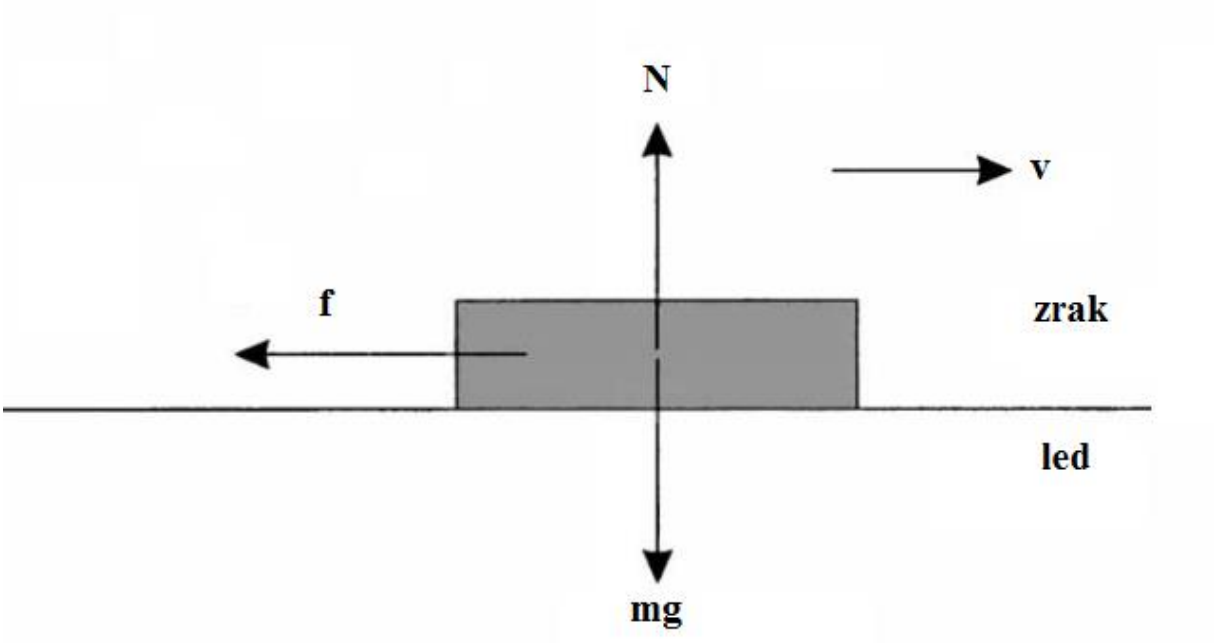
N – „normalna sila“

Koeficijent trenja ovisi materijalima koji su u kontaktu. Npr. guma na ledu će imati manji koeficijent trenja nego guma na betonu. Na Tab. 3.1 prikazano je nekoliko materijala s njihovim koeficijentima.

Tab. 3.1. *Prikaz tablice koeficijenta trenja ovisno o materijalu [13]*

Materijal	Dinamički koeficijent trenja
Guma na betonu	0.8
Željezo na željezu	0.6
Drvo na drvu	0.2
Vosak na snijegu(npr. skijanje)	0.1
Željezo na ledu (npr. klizanje na ledu)	0.005
Led na ledu	0.003
Zglobovi (koljeno, lakat…)	0.003

Jednostavan primjer trenja je hokej pak koji se kreće vodoravno po površini. U tom slučaju, sila trenja ovisi samo o težini objekta. Prema Sl. 3.3 primjećuje se da gravitacija stvara silu koja vuče pak prema dolje, težina, označena kao $m*g$, gdje m predstavlja masu paka u kilogramima, a g konstantu koja se određuje kao akceleracija u odnosu na gravitaciju koja iznosi 9.81 m/s^2. Normalna sila N je također jednaka $m*g$ jer se pak ne miče gore - dolje. Budući da su sada te dvije sile jednake rezultanta sila je samo sila trenja koja će uzrokovati smanjenje brzine paka tijekom vremena.



Sl. 3.3. *Prikaz sila koje djeluju na pak koji se kreće po ledu [13]*

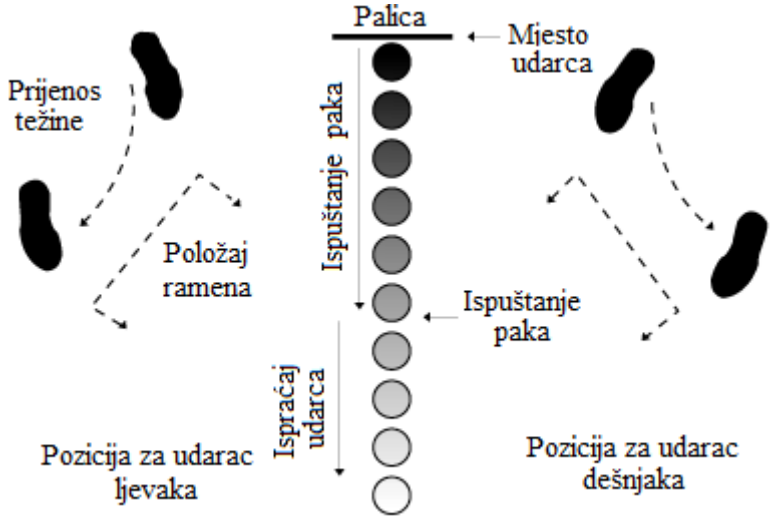
Budući da se hokej pakovi zamrznu prije utakmice kako bi što manje odskakivali od površinu i smanjilo se trenje, u ovom radu će se trenje zanemariti u proračunima, jer je npr. na udaljenosti od 25 metara pogreška u mjerenju, ako zanemarimo trenje, oko 8 centimetara što je zanemarivo.

3.2.2. Udarac u hokeju

Svi igrači hokeja sanjaju o savršenom udarcu koji bi donio pobjedu njihovoj ekipi. Imati dobar udarac jako je bitno u hokeju, jer se tako postižu tri četvrtine golova dok se jedna četvrtina postiže kruženjem oko golmana. Kako bi postali dobri, hokejaši moraju usavršiti puno stvari od kojih su najbitnije preciznost te brzina udarca koja će biti jedna od mjerenih vrijednosti u ovom radu.

Kako bi igrač postigao udarac potrebno je da hokejaškom palicom primjeni silu veću od sile trenja koja se opire kretanju hokej paka. Sila koju je potrebno proizvesti nije velika, jer je koeficijent trenja između paka i leda vrlo malen što se vidi na Tab. 3.1. Igrači imaju mogućnost podizanja paka u zrak tijekom udarca, jer sve palice imaju određeni kut nagiba. Zakrivljenost palice pomaže stvaranju spina koji poboljšava stabilnost i preciznost paka. Postoje dvije glavne vrste udaraca u hokeju , a to su: *wrist shot* (udarac iz zgloba) i *slap shot* (udarac iz zraka).

Wrist shot je tip udarca za čije izvođenje igrači najviše koriste zglob ruke i rame. To je vjerojatno najefikasniji udarac u hokeju, jer je najprecizniji i može se izvesti vrlo brzo uz minimalnu pripremu za razliku od *slap shot-a* što stvara element iznenađenja. Kod *wrist shot-a* pomiče se donja ruka ispod polovine drške palice kako bi se dodala snagu u udarac. Tijelo treba biti pod kutom od 45° u odnosu na gol. Pak se dovodi iza ili u ravninu sa stražnjom nogom spuštajući pritom ramena dok pak dolazi u tu poziciju. Pozicija paka je cijelo vrijeme na sredini donjeg dijela palice koji je nagnut na pak. U ovoj poziciji, težina tijela treba biti na stražnjoj nozi.

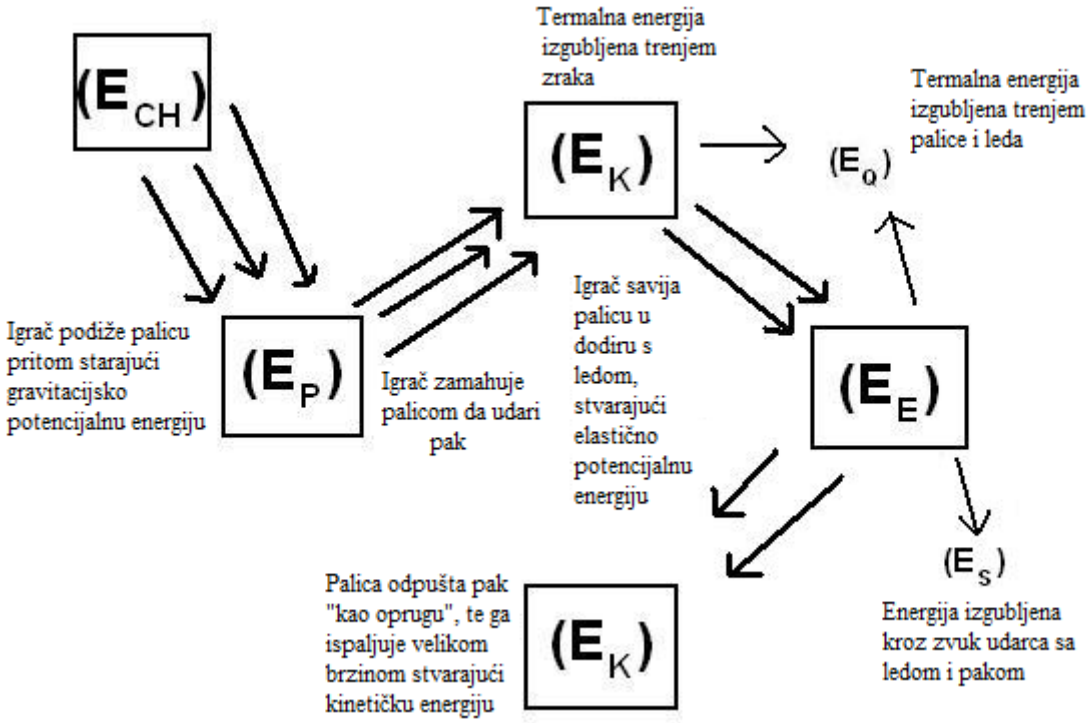


Sl. 3.4. *Prikaz pravilnog udarca iz zgloba [14]*

Povlačeći pak naprijed potrebno je prebaciti težinu prema prednjoj nozi te rotirati tijelo naprijed. Kada palica prijeđe liniju tijela prebacuje se težina na palicu dok donja ruka gura pak prema naprijed ,a gornja prema nazad. Pak je izbačen kada dosegne prednju noge te ramena igrača budu okrenuta prema mreži tvoreći kvadar. U trenutku izbačaja zglobovi se zarotiraju što uzrokuje podizanje paka u zrak. Nakon izbačaja paka prati se njegova putanja dok vrh palice ne bude pokazivao prema cilju. Snaga udarca ovisi o količini rotacije zgloba i koliko daleko se „isprati“ hokej pak palicom.

Slap shot je najsnažniji udarac u hokeju. Igrač podiže palicu u ravnini ramena ili više te zamahom udara u led nekoliko centimetara iza hokej paka koristeći pritom svoju težinu kako bi savio palicu i „napunio“ je energijom kao oprugu. Kada palica dotakne pak, igrač rotira zglobove i prebacuje težinu kako bi se sakupljena energija otpustila kroz hokej pak. Na kraju, kao i kod udarca iz zgloba pak se „isprati“ prema cilju. Ovi udarci stvaraju veliku količinu sile, ali su teški za kontrolirati pa se njima gubi preciznost [15].

Na Sl. 3.5 prikazan je dijagram stanja energija tijekom svakog koraka prethodno opisanog udarca.



Sl. 3.5. *Prikaz dijagrama stanja energije tijekom slap shot-a [15]*

3.2.3. Sudari igrača

Iako su igrači dobro opremljeni igra zna biti surova te često dolazi do kontakta i izravnog zabijanja u protivnike. Koristeći nekoliko jednostavnih matematičkh formula, može se izračunati koliko se energije oslobodi prilikom sudara dvaju igrača. Prema [16], energija svakog igrača može se računati po formuli**:**

$$E = 1/2\ (m * a\text{^}2\ ) \qquad (2)$$

gdje je:

m – masa igrača

a – akceleracija igrača

Onda je konačna energija u sudaru jednaka:

$$E = \frac{1}{2}(m_u * {a_k}^2) \tag{3}$$

gdje je:
m_u – ukupna masa igrača u sudaru
a_k – konačna akceleracija

Budući da je za izračun potrebna konačna akceleracija koristimo još jednu formulu za izračun, a to je:

$$a_k = \frac{(m_1 * a_1) + (m_2 * a_2)}{m_u} \tag{4}$$

gdje su:
m_1, m_2 – mase igrača 1 i 2
a_1, a_2 – akceleracije igrača 1 i 2
m_u – ukupna masa

3.3. Ideja rješenja kao odgovor na potrebe

Budući da tehnologija brzo napreduje, posebno u IoT sektoru, pojavile su se i nove mogućnost rješavanja određenih problema. Problem statistike i mjerenja u hokeju pronalazi rješenje u tom području. Ideja rješenja je da se vrijednosti sa senzora koje obrađuje razvojna pločica šalju putem *bluetootha* na Android aplikaciju koja ima mogućnost daljnje analize podataka.

U ovom radu za izradu prototipa koristit će se Arduino Mega razvojna pločica napajanja s baterijom od 9V. Na nju će biti spojen akcelerometar i žiroskop u jednom modulu MPU-6050 čije vrijednosti Arduino sakuplja te *bluetooth* modulom šalje na Android pametni telefon , tj. u aplikaciju. Svaka od komponenti kao i njihovi međusobni spojevi i komunikacija, bit će detaljnije objašnjeni u poglavlju broj 4.

Android aplikacija ima mogućnost biranja spajanja na određeni hokej pak. Sučelje omogućuje slanje signala spremnosti za udarac, čime se nadolazeće vrijednosti upisuju u bazu podataka, kao i signala za prekid. Također nudi prikaz statistike svih udaraca te upute o pravilnom izvođenju udaraca, tj. tehnici.

3.4. Mogućnosti nadogradnje

U ovom dijelu opisat će se neke od mogućnosti nadogradnje sustava izrade IoT uređaja koje bi ga poboljšali kako funkcionalno tako i vizualno.

Prototip koji je napravljen u ovom radu većih je dimenzija te bi bilo potrebno smanjiti dimenzije navedenog kako bi svi potrebni senzori imali mogućnost ugradnje u hokej pak koji je poprilično malen (opisan u potpoglavlju 3.2). Kako bi se to postiglo, potrebno je smanjiti sklopovlje tako što će se koristiti samo čip Atmega koji je isprogramiran preko Arduina te manje inačice ostalih i integrirati u jedan sklop na tiskanoj pločici malih dimenzija. Nadalje, integriranje rfid prijemnika u pak i rfid čipa u palicu omogučila bi praćenje treninga cijele momčadi.

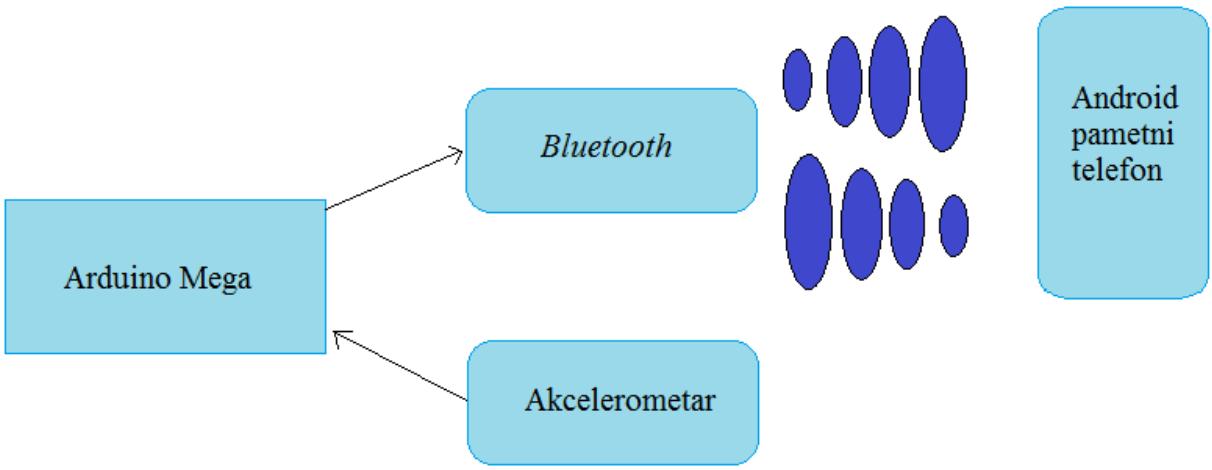
Aplikacija se može nadograditi boljim korisničkim sučeljem u smislu dizajna budući da je izgled aplikacije u radu na osnovnoj razini. Budući da je aplikacija napravljena za treninge zahtijeva pripremu svakog udarca tako što se sustavu šalje signal spremnosti za početak mjerenja te signal za uspješno izvršen udarac. Implementacija automatske detekcije udarca omogućila bi zapisivanje podataka tijekom utakmice, jer bi se izbjegla potreba za signaliziranjem svakog udarca te bi se omogućila dostupna statistika svih udaraca koji su se dogodili tijekom utakmice. Također je moguća implementacija iscrtavanja putanje udarca u tri dimenzije.

4. SKLOPOVSKO I PROGRAMSKO RJEŠENJE

Budući da je u ovom radu korištena kombinacija sklopovlja i mobilne aplikacije, sljedeći dio rada biti će podijeljen na sklopovsko i programsko rješenje. U sklopovskom dijelu prikazuje se realizacija sklopovlja programskim alatom *Fritzing*, međusobna komunikacija (odnos) dijelova sustava te će se uz njihov slikovni prikaz opisat nožice, radni naponi, frekvencije te ostale bitne karakteristike.. U programskom dijelu objasnit će se korištene programske tehnologije za izradu te će biti prikazani i objašnjeni ključni dijelovi koda.

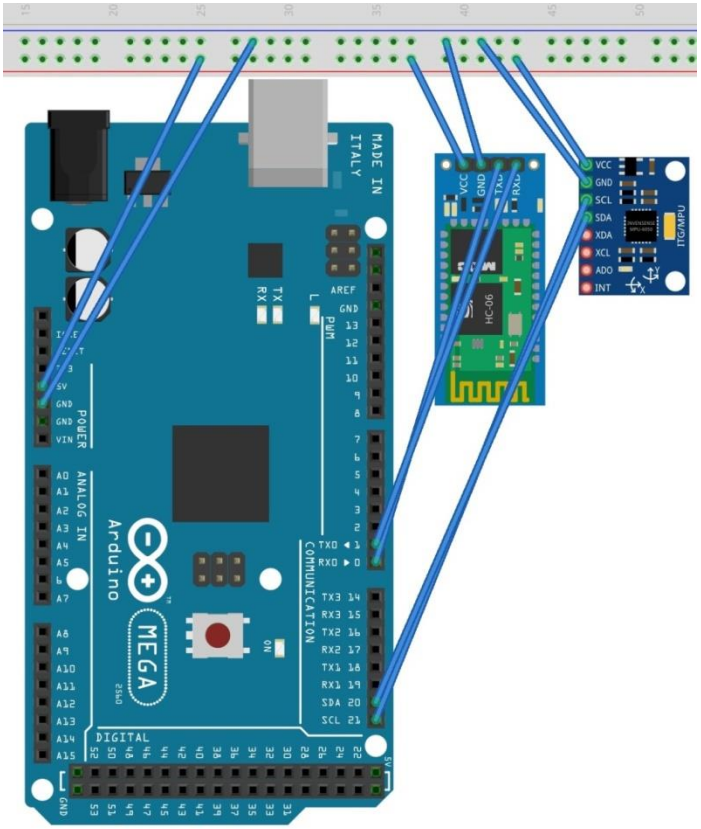
4.1. Sklopovsko rješenje i korišteno sklopovlje

Sklopovlje se sastoji od razvojne pločice Arduino Mega, akcelerometra, *bluetooth* modula i Android pametnog telefona. Na razvojnoj pločici Arduino Mega su spojeni akcelerometar i *bluetooth* modul pomoću kojih podatci dolaze do pametnog telefona i koriste se za rad aplikacije. Akcelerometar mjeri promjene položaja i ubrzanje te se podatci putem *bluetooth-a* šalju na pametni telefon prema Sl. 4.1.

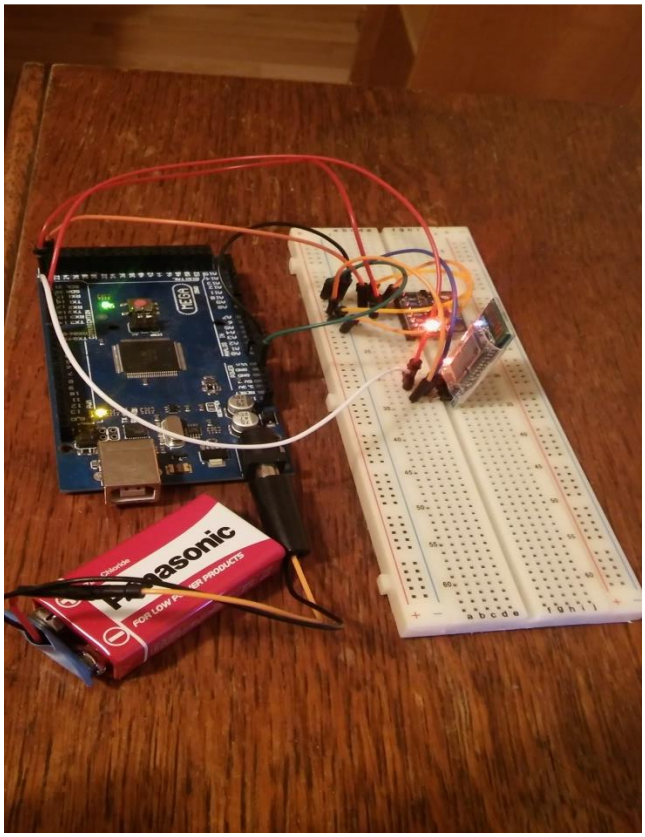


Sl. 4.1. *Prikaz i interakcija sklopovlja*

Na Sl. 4.2 prikazana je razvojna pločica Arduino Mega, akelerometar i *bluetooth* u *Fritzingu* te način na koji su povezani (nožice), a na Sl. 4.3 u stvarnom svijetu. Nožica TXD je serijski izlaz bluetooth modula i spojen je na nožicu RX0 od Arduina koji je serijski ulaz. Nožica RXD je serijski ulaz bluetooth modula te je analogno tome spojen na serijski izlaz Arduina TX0.



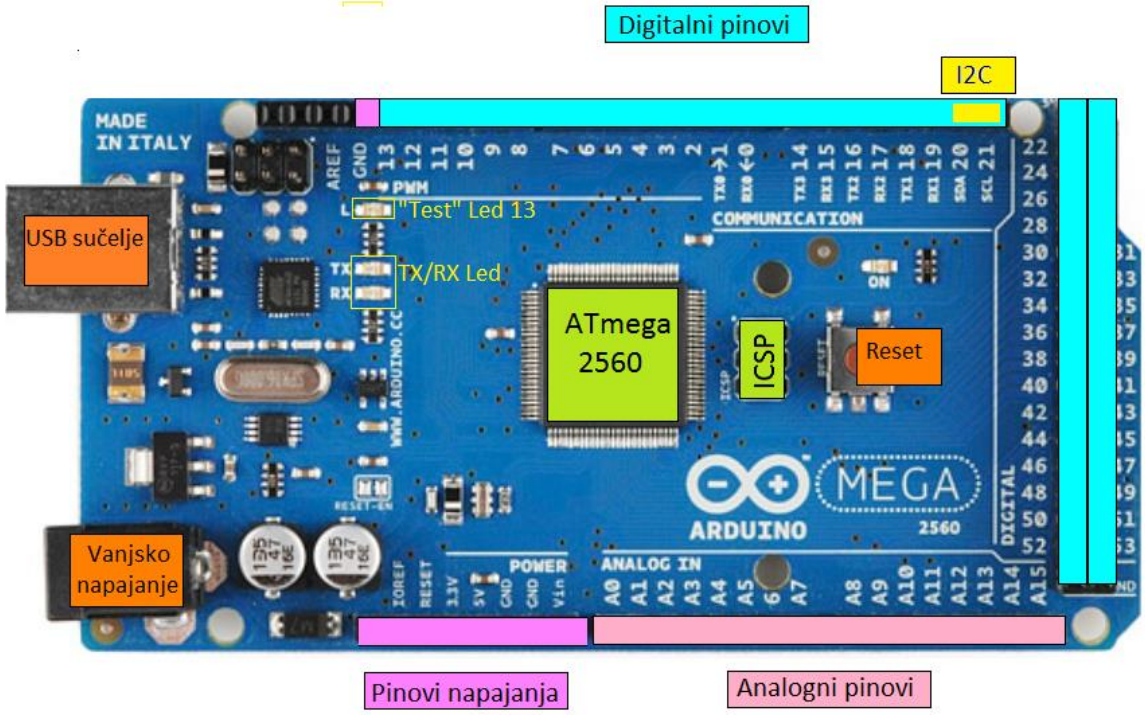
Sl. 4.2. *Prikaz sklopovlja u Fritzingu*



Sl. 4.3. *Prikaz sklopovlja u stvarnom svijetu*

4.1.1. Arduino Mega

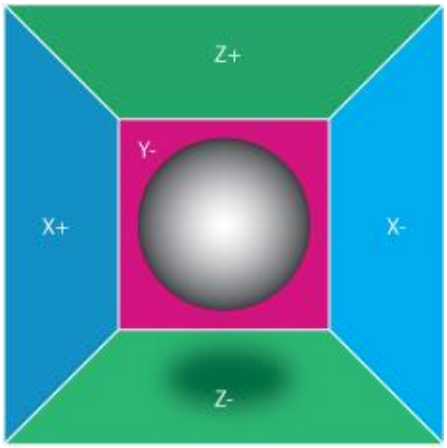
Arduino Mega je razvojna pločica zasnovana na ATmega2560 mikroupravljaču. Ima 54 digitalnih izlazno/ulaznih nožica od kojih se se 15 može koristit kao PWM (engl. *Pulse Width Modulation*) izlazi, 16 analognih ulaza, 4 UART-a (engl. *universal asynchronous receiver/transmitter*), 16MHz keramički otpornik, USB (engl. *Universal Serial Bus*) priključak, priključak za napajanje, ICSP (engl. *in-circuit serial programming*) zaglavlje i tipku reset. Sadrži sve što je potrebno kao podrška mikroupravljaču. PWM signal je vrsta digitalnog izlaza koja ima određene strukture u vremenskom ponašanju. Točnije, signal ima konstantan period. UART je sklopovlje za asinkronu serijsku komunikaciju. ICSP je „sposobnost“ nekih programibilnih uređaja, u ovom slučaju mikroupravljača, da se programira nakon što je cijeli sustav instaliran. USB je standardno sučelje za spajanje uređaja na računalno. Korištenje Arduina Mega je vrlo jednostavno, a spaja se USB kabelom te ima i mogućnost napajanja preko AC/DC adaptera ili baterijama. Kompatibilan je s većinom štitova koji su dizajnirani za Arduino Uno, Duemilanove i Diecimila. ATmega2560 ima 128 KB kapaciteta memorije od kojih se 4KB koriste pri podizanju operacijskog sustava, 8KB SRAM memorije te 4KB EEPROM memorije. SRAM je memorija koja se koristi za pričuvnu memoriju (engl. *cache*) ili za računala gdje je brzina osnovni cilj. EEPROM je vrsta memorije koja se koristi za trajno pohranjivanje podataka, ali ima i mogućnost brisanja upisanih podataka samo električnim putem. Ključna razlika u navedenim vrstama memorije je ta što se SRAM memorija briše svaki put kada integrirani krug izgubi napajanje dok kod EEPROM podatci ostaju. Pločica može raditi na vanjskom napajanju od 6 do 20 volti, ali je preporučeno da se napaja s 7 do 12 V jer manje od 7 V može prouzročiti nestabilnosti, a više od 12 V može pregrijavanjem regulatora oštetiti ploču. Na Sl. 4.4 prikazana je Arduino pločica te njenii ulazi i izlazi [17].



Sl. 4.4. *Prikaz i opis nožica Arduina Mega*

4.1.2. Akcelerometar

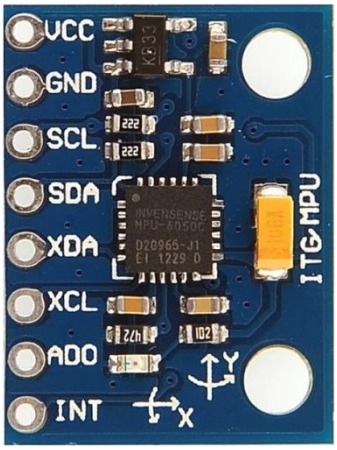
Akcelerometar je senzorski uređaj koji regira na promjenu brzine i položaja. Računajući količinu ubrzanja zbog gravitacije može dobiti informaciju o svom položaju u odnosu na Zemlju, a očitavajući količinu dinamičkog ubrzanja može otkriti koliko brzo i u kojem smjeru se kreće. U današnje vrijeme postoji u raznim uređajima: pametnim telefonima, tabletima, prijenosnim uređajima za glazbenu reprodukciju, itd.



Sl. 4.5. *Prikaz beztežinskog stanja[18]*

Pri objašnjavanju načina rada akcelerometra često se vizualizira kugla koja je zatvorena u kocki (Sl. 4.5). Svaka od koordinatnih osi predstavlja jedan zid, a os Y+ je uklonjena radi bolje vizualizacije. Pomicanjem kocke ulijevo, kugla će udariti u zid X- te se mjeri sila pritiska koju je prouzročila kugla na zid. Prema tome, akcelerometar mjeri akceleraciju indirektno kroz silu koja je prouzročena udarom lopte.

U ovom radu koristit će se MPU-6050 3-osni akcelerometar modul koji je prikazan na Sl. 4.6.



Sl. 4.6. *Akcelerometar MPU-6050[19]*

Značajka MPU-6050 akcelerometra su tri 16-bitna pretvornika analognog u digitalni signal, za žiroskop i tri 16-bitna pretvornika analognog u digitalni signal za akcelerometar što omogućava istovremeno prikupljanje podataka svih osi žiroskopa i akcelerometra. Za mjerenje kretnji, žiroskop i akcelerometar se mogu programirati u određenom rasponu ovisno po potrebi prema Tab. 4.1 MPU-6050 ima međuspremnik koji pomaže u smanjenju potrošnje energije sustava tako što omogućava procesoru da odjednom očita podatke sa senzora te uđe u niskoenergetski način rada dok glavni procesor prikuplja dodatne podatke.

Veličina senzora je 4x4x0.9 mm, radi na napajanju u rasponu od 2.375 V do 3.46 V. Uređaj ima toleranciju na udarce te programabilne filtre niskih frekvencija za žiroskop i akcelerometar [20].

U tablici 4.1 dan je prikaz svih registara korištenih u ovom projektu i njihov opis.

Tablica.4.1. *Registri korišteni u projektu[20]*

Adresa (Hex)	Ime registra	Opis registra
1B	GYRO_CONFIG	Registar za regulaciju osjetljivosti. Ovisno o traženoj osjetljivosti žiroskopa upisuju se hekasdecimalne vrijednosti. Moguće osjetljivosti su ±250°/s, ±500°/s, ±1000°/s i ±2000°/s.
43	GYRO_XOUT_H	Registar koji pohranjuje vrijednosti ubrzanja u osi X.
45	GYRO_YOUT_H	Registar koji pohranjuje vrijednosti ubrzanja u osi Y.
47	GYRO_ZOUT_H	Registar koji pohranjuje vrijednosti ubrzanja u osi Z.
1C	ACCEL_CONFIG	Registar za regulaciju osjetljivosti. Ovisno o traženoj osjetljivosti akcelerometra upisuju se hekasdecimalne vrijednosti. Moguće osjetljivosti su ±2g, ±4g, ±8g i ±16g.
3B	ACCEL_XOUT_H	Registar koji pohranjuje vrijednosti ubrzanja u osi X.
3D	ACCEL_YOUT_H	Registar koji pohranjuje vrijednosti ubrzanja u osi Y.
3F	ACCEL_ZOUT_H	Registar koji pohranjuje vrijednosti ubrzanja u osi Z.
6B	PWR_MGMT_1	Registar za odabir načina rada. (DEVICE_RESET,CYCLE,SLEEP,CLKSEL,TEMP_DIS)

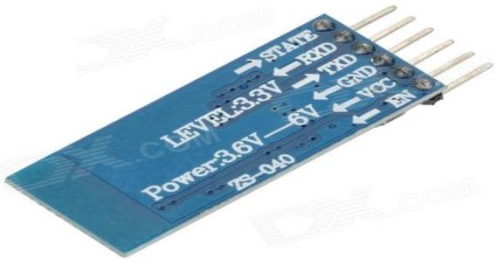
4.1.3. *Bluetooth*

Bluetooth je prema [21] globalni bežični komunikacijski standard koji povezuje uređaje na određenoj udaljenosti. Koristi radio valove umjesto žica ili kabela kako bi se povezao s uređajima, konkretno u ovom radu s pametnim telefonom. Radio valovi putem kojih se prenosi su frekvencije od 2,402 do 2.480GHz. Kako ne bi došlo do kontakta s drugim uređajima koji koriste iste frekvencije, *bluetooth* šalje vrlo slab signal oko 1mW što mu ograničava domet na 10m. Brzina prijenosa ovisi o inačici *bluetooth*a prema tablici 4.2.

Tablica. 4.2. *Brzina prijenosa podataka prema bluetooth verziji[22]*

Verzija	Brzina prijenosa
1.2	1 Mbit/s
2.0 + EDR	3 Mbit/s
3.0 + EDR	24 Mbit/s
4.0	24 Mbit/s

U ovom radu koristit će se *bluetooth* HC-06 modul koji je inicijalno podređen (engl. slave). Zaporka koja se koristi pri spajanju na *bluetooth* je „1234“.



Sl. 4.7. *Bluetooth HC-06[23]*

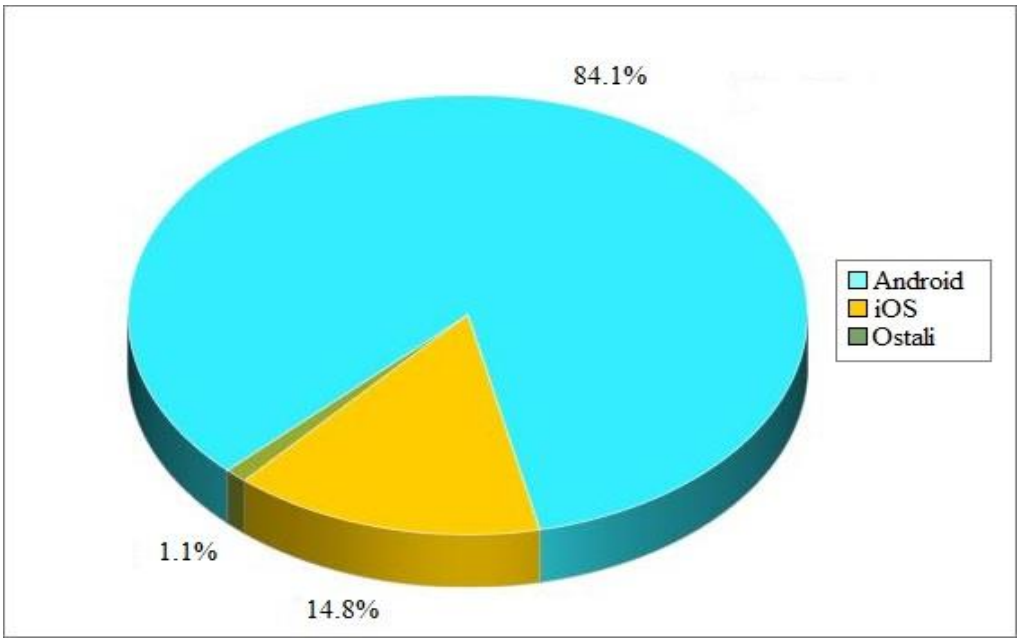
Opis nožica *bluetootha*:

- STATE – nožica koji služi za provjeru statusa komunikacije je postavljena u '0' kada komunikacija ne postoji , tj. u '1' kada postoji
- TXD – serijski izlaz modula
- RXD – serijski ulaz modula
- GND – uzemljenje
- VCC – 3 do 6 V istosmjerne struje
- EN – koristi se za ulazak u postavke

4.2. Programske tehnologije

U ovom dijelu reći će se nešto više o *Arduino* programskoj tehnologiji koja je potrebna za upravljanje sklopovljem. Podatci koji se dobiju sa senzora i obrade s *Arduino* programskim jezikom mogu se koristiti za izradu mobilne aplikacije. Nadalje će se opisati program Fritzing koji se koristi za izradu nacrta sklopovlja.

Mobilne aplikacije su postale strašno popularne u današnje vrijeme te postoji velik broj platformi na kojima se mogu razvijati. Mobilna aplikacija u ovom radu bit će razvijena u razvojnom okruženju *Android Studio* koje je detaljnije opisano u dijelu 4.2.2. Razlog zbog kojega je izabran *Google Android* operacijski sustav je njegova popularnost što se vidi na Sl. 4.8 koja prikazuje udio pametnih telefona u prvom kvartalu 2016. godine.



Sl. 4.8. *Prikaz udjela pametnih telefona u prvom kvartalu 2016.[24]*

4.2.1. Platforma *Arduino*

Arduino je platforma otvorenog koda koja ima mogućnost primati ulaze (svjetlo na senzoru, pristisak sklopke) i pretvoriti ih u izlaze (aktivacija motora, paljenje lampica). Omogućava upravljanje pločom slanjem naredbi mikroupravljaču.

Razvojna pločica *Arduino* programira se putem *Arduino* programskog jezika. Struktura je jednostavna te se sastoji se od dva dijela, a to su funkcije setup() i loop(). Funkcija setup() služi za pripremu i poziva se jednom kada se program pokrene, a koristi se za incijalizaciju nožica i započinjanje serijske veze. Nakon završetka funkcije setup(), loop() funkcija se stalno izvršava omogućujući programu da se mijenja, reagira i kontrolira *Arduino* ploču [25].

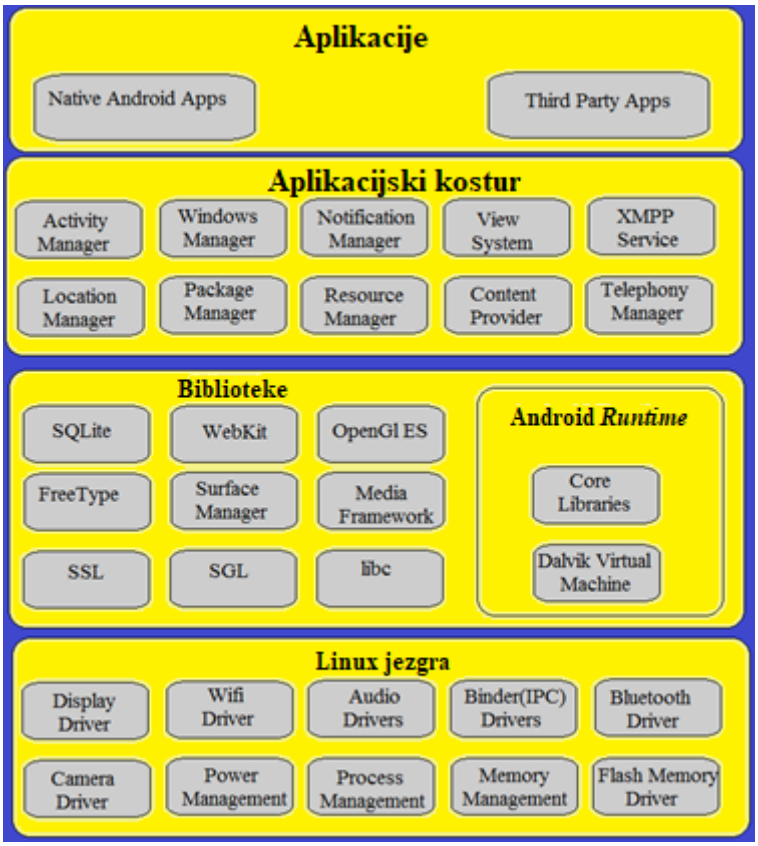
```
void setup() {
  // kod za inicijalizaciju koji se izvršava samo jednom
}

void loop() {
  // glavni kod koji se periodički izvršava
}
```

Sl. 4.9. *Prikaz Arduino strukture[25]*

4.2.2. *Android i Android Studio*

Android je operacijski sustav otvorenog koda zasnovan na jezgri Linux 2.6 i napisan u C/C++ programskom jeziku. S obzirom na otvorenost izvornog programskog koda, aplikacije putem međusloja imaju mogućnost komuniciranja i pokretanja drugih aplikacija, primjerice za ostvarivanje poziva, slanje SMS poruka, pokretanja kamere i slično. Sve Android aplikacije se razvijaju u programskom jeziku Java.

Arhiktetura mu je u obliku stoga. Svaki sloj u stogu odgovara elementima unutar svakog sloja te su elementi usko povezani. Na Sl. 4.10 prikazana je arhiktetura Androida. Na dnu stoga nalazi se Linux jezgra koja pruža određen razinu apstrakcije između sklopovlja uređaja i gornjih slojeva stoga. Sadrži pogonske programe (engl. *driver*) od kojih su najvažniji pogonski programi za međuprocesnu komunikaciju (*Inter-process communication - IPC*) koji služi za izmjenu podataka između različitih procesa ili niti unutar istog procesa te pogonski program za upravljanje napajanjem (eng. *power managment*). Iznad jezgre su biblioteke napisane u programskom jeziku C/C++. Pored njih je sloj *Android Runtime* koji služi za pokretanje aplikacija, a sastoji se od komponente *Core Libraries*, koja sadrži većinu jezgrenih biblioteka Jave, te *Dalvik Virtual Machine (DVM)* koja izvodi aplikacije kao proces direktno na Linux jezgri, kroz svoju instancu na virtualnom stroju. DVM pretvara Java datoteke u svoj vlastiti format (.dex), jer on ostavlja 50% manji memorijski otisak od Java koda. Aplikacijski okvir (eng. *Application Framework*) je sloj koji pruža mnoge usluge više razine aplikacijama u obliku Java klasa. Programerima je dozvoljeno korištenje usluga iz tog sloja u njihovim aplikacijama. Aplikacije su na vrhu arhikteture Android sustava, a sastoje se od ugrađenih aplikacija koje dolaze s uređajem (imenik, web preglednik) i aplikacija instaliranih nakon kupovine uređaja.



Sl. 4.10. *Prikaz arhikteture Android sustava [26]*

Android Studio je službeno integrirano razvojno okruženje za razvoj mobilnih aplikacija. Bazira se na JetBrains programu, te je namijenjen isključivo razvijanju aplikacija za Android mobilnu platformu. Ima zaseban sustav koji predstavlja skup alata za izgradnju, testiranje i pokretanje aplikacija.

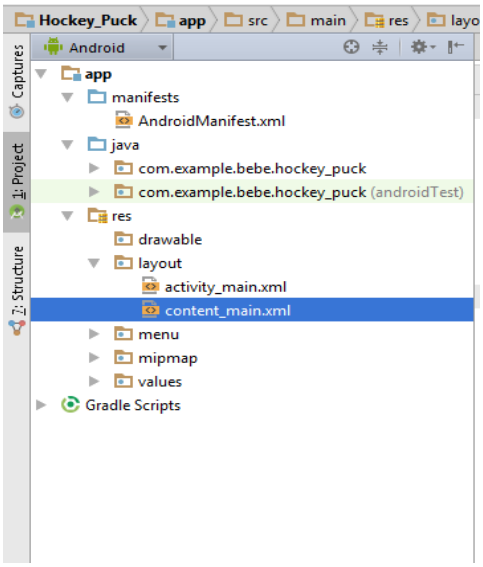
Android studio koristi Android Virtualni Uređaj (engl. *Android Virtual Device*, AVD) kao alat za upravljanje virtualnim uređajima za pokretanje aplikacija. Pomoću AVD-a kreira se vlastiti virtualni uređaj koji se može prilagoditi potrebama korisnika te stvoriti emulator pomoću kojega se mogu testirati aplikacije.

Svaki projekt u Android Studiju sadrži jedan ili više modula sa datotekama izvornih kodova i resursa. Tipovi modula su:

- *Android app* moduli
- Bibliotečni moduli
- *Google App Engine* moduli

Prikaz projekta je organiziran po modulima kako bi omogućio brz pristup ključnim izvornim datotekama. Svaki modul sadrži sljedeće mape:

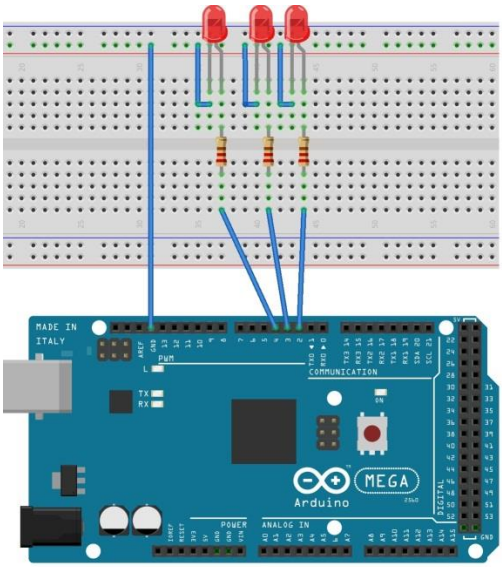
- *manifests*: Sadrži AndroidManifest.xml datoteku
- *java*: Sadrži java datoteku izvornog koda
- *res*: Sadrži sve resurse koji nisu vezani za kod kao XML, UI stringove, *bitmap* slike



Sl. 4.11. *Prikaz strukture Android projekta [27]*

4.2.3. *Fritzing*

Fritzing je prema [28] inicijativa otvorenog koda za razvoj programske podrške koja se koristi za dizajniranje sklopovlja. Idealan je za dokumentaciju *Arduino* sklopovlja te se zbog toga i koristi u radu. Ne nudi mogućnosti simulacije projekata nego samo njihovo crtanje. Izvorni kod za *Fritzing* je pisan u programskom jeziku C++.



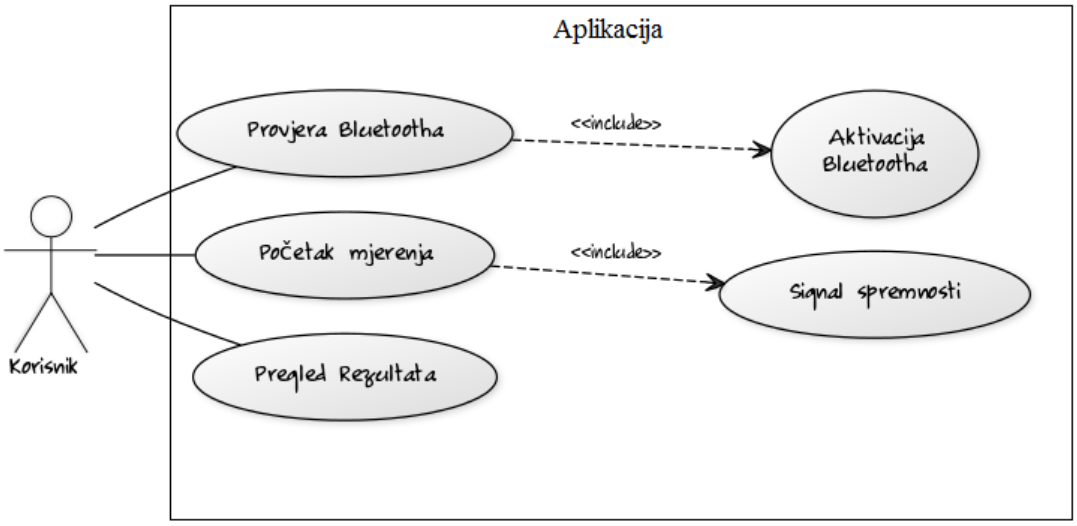
Sl. 4.12. *Prikaz LED dioda i Arduina Mega u Fritzingu*

4.3. Funkcionalnost aplikacije

Funkcionalnost sustava s gledišta korisnika opisana je UML dijagramom slučajeva korištenja (engl. *use case diagram*) i slijednim dijagramom (engl. *sequence diagram)* za nekoliko slučajeva. Osoba koja koristi aplikaciju je sudionik ovog sustava.

4.3.1. Dijagram slučajeva korištenja

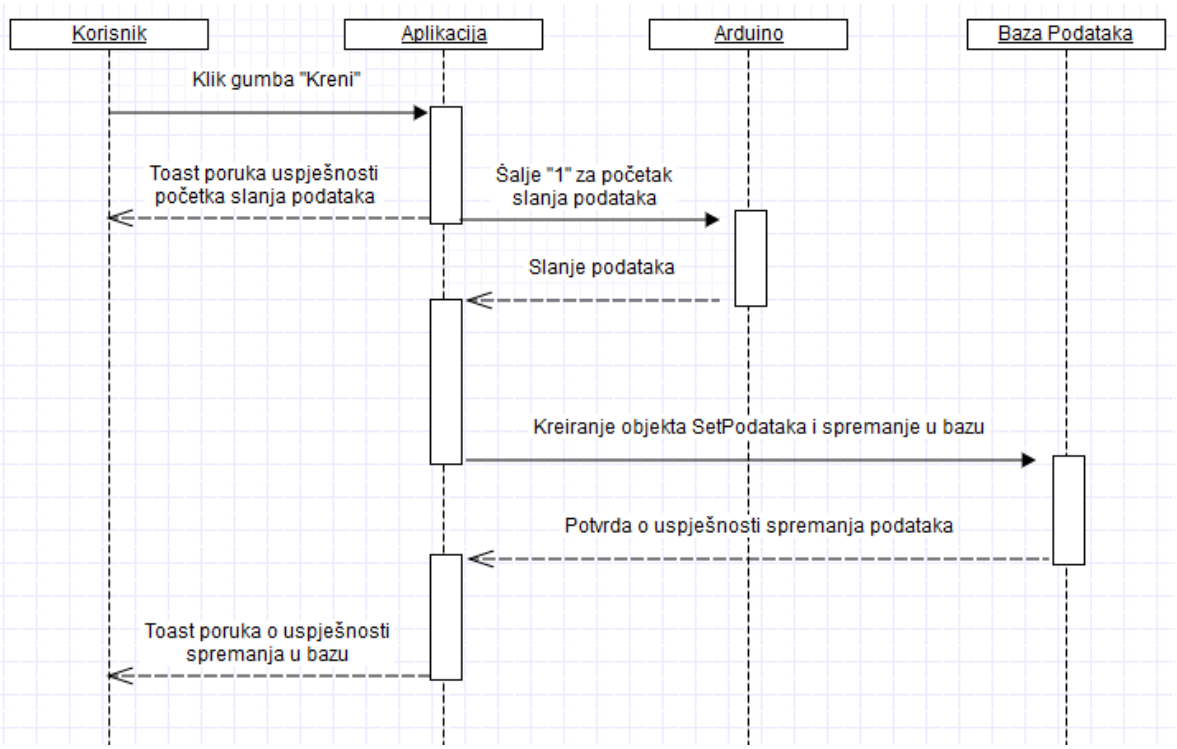
Dijagram slučajeva korištenja prikazuje odnos između korisnika (izvođača, sudionika) i načina korištenja. Opisuje sustav s gledišta vanjskog promatrača, odnosno funkcionalnosti koje vidi korisnik. Dijagram slučajeva korištenja za ovaj rad je prikazan na Sl. 4.13.



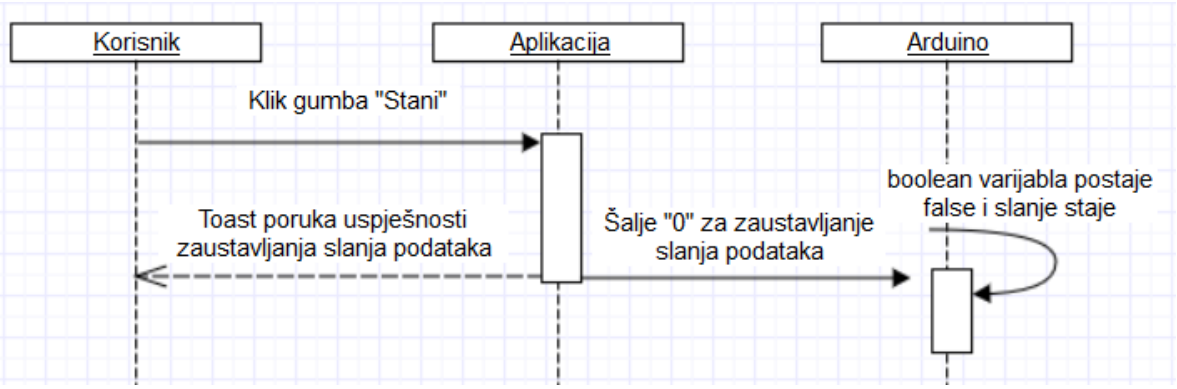
Sl. 4.13. *Dijagram slučajeva korištenja hokej aplikacije*

4.3.2. Slijedni dijagram

Za razliku od use-case dijagrama, kod kojih vremenska komponenta nije važna, sekvencijski dijagram daje naglasak na vremenskom radoslijedu kojim se odvija međudjelovanje sudionika u sustavu, pa ga se svrstava u dinamičke UML dijgrame. U ovoj aplikaciji protok podataka počinje na gumb „Kreni“ te se zaustavlja na gumb „Stani“ što će se detaljnije objasniti u dijelu 4.4 gdje će biti priložen programski kod dok će se ovdje ti slučajevi opisati sekvencijalnim dijagramima prikazanim na Sl.ma 4.14 i 4.15.



Sl. 4.14. *slijedni dijagram klika na gumb „Kreni“*



Sl. 4.15. *Slijedni dijagram klika na gumb „Stani*

4.4. Programsko rješenje

Uređaj koji može mjeriti određene parametre nema svrhu ako te parametre ne može proslijediti i iskoristiti. Za prijenos podataka s uređaja na mobilnu aplikaciju koja će biti opisana u ovom dijelu kao što je prethodno navedeno se koristi *bluetooth*. Upravo iz tog razloga aplikacija kao početnu točku ima aktivnost za provjeru, aktivaciju te spajanje *bluetootha*. Nakon spajanja, sljedeća aktivnost ima mogućnost kontroliranja početka i kraja udarca, popisa udaraca sa njihovim statistikama te savjete za izvođenje udaraca. U ovom dijelu detaljno će se opisati svaka od aktivnosti s priloženim kodom i objašnjenjem istih.

4.4.1. *DeviceListActivity*

Svaka aplikacija mora imati svoju početnu točku (aktivnost) koja je definirana u *Android Manifest* datoteci. U ovom radu početna aktivnost je *DeviceListActivity* što se može vidjeti na Sl. 4.16.

```
<activity
    android:name=".DeviceListActivity"
    android:label="@string/app_name"
    android:screenOrientation="portrait">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

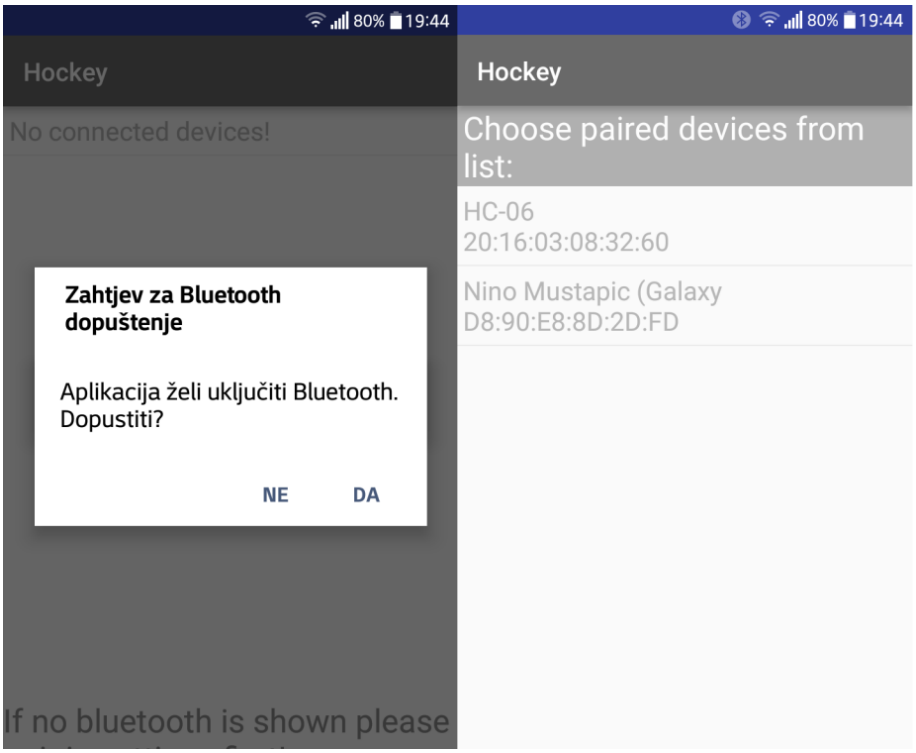
Programski kod 4.1. *Prikaz definiranja početne aktivnosti u Android Manifest datoteci*

Nakon što se postavi sučelje u aktivnosti, prva metoda koja se poziva je *checkBTState()* metoda čija je poslovna logika prikazana na programskom kodu 4.2.

```
private void checkBTState() {

        BA =BluetoothAdapter.getDefaultAdapter();
        if(BA ==null) {
            Toast.makeText(getApplicationContex(), "Bluetooth
not supported! Buy new phone please!",
Toast.LENGTH_SHORT).show();
        } else {
            if (BA.isEnabled()) {
                Log.d(TAG, "...Bluetooth ON...");
            } else {

                Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, 1);

            }
        }
```

Programski kod 4.2. *Prikaz koda metode checkBTState()*

Metoda prvo dohvaća *bluetooth adapter* od uređaja te ako uređaj ne posjeduje *bluetooth* izvješćuje korisnika o tome. Ako je *bluetooth* uključen šalje korisniku poruku da *bluetooth* radi, a u suprotnom nudi korisniku aktivaciju *bluetooth-a* kako bi mogao nastaviti s radom. Na Sl. 4.16 prikazano je početno sučelje nakon prihvaćanja aktivacije *bluetooth-a*.



Sl. 4.16. *Prikaz sučelja prve aktivnosti za povezivanje putem bluetooth veze*

Nakon što je *bluetooth* uključen, slijedi dio koda koji inicijalizira *ArrayAdapter* te ga povezuje s *ListView*. U *ListView* prikazana su imena svih spojenih uređaja s njihovim MAC (engl. *MAC, Media Access Control*) adresama. Ako nijedan uređaj nije povezan prikazat će se tekst koji javlja da nema povezanih uređaja.

```
mPairedDevicesArrayAdapter = new ArrayAdapter<>(this,
R.layout.ime);

        ListView pairedListView = (ListView)
findViewById(R.id.paired_devices);

        pairedListView.setAdapter(mPairedDevicesArrayAdapter);


pairedListView.setOnItemClickListener(mDeviceClickListener);

        BA = BluetoothAdapter.getDefaultAdapter();

        Set<BluetoothDevice> pairedDevices = BA.getBondedDevices();

        if (pairedDevices.size() > 0) {

findViewById(R.id.name_paired_devices).setVisibility(View.VISIBLE);
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() +
"\n" + device.getAddress());
            }
        } else {
           mPairedDevicesArrayAdapter.add("No connected devices!");
        }
    }
```

Programski kod 4.3. *Prikaz koda za dohvaćanje spojenih uređaja*

Kada je *bluetooth* upaljen te lista ispunjenja dostupnim uređajima sljedeći korak je odabir uređaja s kojim je potrebno komunicirati. Za to je implementiran osluškivač događaja (engl. *OnClickListener*) čiji je kod prikazan u programskom kodu 4.4.

```
private OnItemClickListener mDeviceClickListener = new
OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int
arg2, long arg3) {

            tvConnect.setText("Connecting...");
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

          Intent myintent = new Intent(DeviceListActivity.this,
MainActivity.class);
            myintent.putExtra(EXTRA_DEVICE_ADDRESS, address);
                    startActivity(myintent);
        }
    };
```

Programski kod 4.4. *Prikaz koda za osluškivač događaja*

Odabirom jednog od ponuđenih spojenih uređaja pokreće se osluškivač događaja te dohvaća MAC adresu uređaja što je zadnjih 17 znakova iz *View-a*. Nakon toga kreira se naredba eksplicitne namjere (engl. *explicit intent*) klase *Intent* sa dodatkom *putExtra* koji prenosi adresu uređaja u sljedeću aktivnosti koja se poziva intentom, a to je *MainActivity*.

4.4.2. *MainActivity*

U aktivnosti *MainActivity* se upravlja cijelom aplikacijom preko „izbornika“ koji se sastoji od 3 obična gumba (engl. *Button*) i jednog slikovnog gumba (engl. *ImageButton)* koji ima dva moguća stanja, a to su „Kreni“ i „Stani“. Programskim kodom 4.4. prikazana je metoda *onResume MainActivity*-a.

```
public void onResume() {
        super.onResume();

        Intent intent = getIntent();
        String address =
intent.getStringExtra(DeviceListActivity.EXTRA_DEVICE_ADDRESS)
;
                    BluetoothDevice device =
          mBluetoothAdapter.getRemoteDevice(address);

        try {
            btSocket = createBluetoothSocket(device);
        } catch (IOException e) {
            Toast.makeText(getBaseContext(),
getString(R.string.bluetooth_socket_error),
Toast.LENGTH_LONG).show();
        }
        try {
            btSocket.connect();
        } catch (IOException e) {
            try {
                btSocket.close();
            } catch (IOException e2) {
                Toast.makeText(getBaseContext(),
getString(R.string.bluetooth_socket_closing),
Toast.LENGTH_LONG).show();
            }
        }

       if (btSocket.isConnected()) {
        mConnectedThread = new ConnectedThread(btSocket);
        mConnectedThread.start();
        mConnectedThread.write("a");
      } else {
           Toast.makeText(getBaseContext(),
getString(R.string.bluetooth_socket_closed),
Toast.LENGTH_LONG).show();
       }
    }
```

Programski kod 4.4. *Prikaz onResume koda za MainActivity*

U metodi *onResume()* dohvaća se naredba eksplicitne namjere poslana iz *DeviceListActivity*-a s pripadajućom MAC adresom te se pravi objekt *bluetooth* uređaja pomoću adrese. Nakon toga pokušava se stvoriti *BluetoothSocket* preko kojega se izvršava komunikacija između pametnog telefona i Arduino sustava. Ako je uspješno stvorena pokušava se uspostaviti veza povezivanjem. Ako je *BluetoothSocket* uspješno povezan, slijedi stvaranje nove niti (engl. *thread*), pokretanje niti te slanje testnog podatka u ovom slučaju literala "a" da bi se provjerilo je li veza uspostavljena. Prethodno je deklarirana klasa *ConnectedThread* koja nasljeđuje klasu *Thread* i pomoću ulaznog i izlaznog toka (engl. *stream)* upravlja podatcima. Klasa *ConnectedThread* prikazan je programskim kodom 4.5 i 4.6.

```
private class ConnectedThread extends Thread {
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;
        ConnectedThread(BluetoothSocket socket) {
            InputStream tmpIn = null;
            OutputStream tmpOut = null;
            try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
updateOnUi(getString(R.string.error_input_output_streams));
            }
            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }
        public void run() {
            byte[] buffer = new byte[256];
            int bytes;
            while (true) {
                try {
                    bytes = mmInStream.read(buffer);
                    String readMessage = new String(buffer, 0,
bytes);               mBluetoothHandlerIn.obtainMessage(handlerState,
bytes, -1, readMessage).sendToTarget();
                } catch (IOException e) {

updateOnUi(getString(R.string.problem_receiving_data));
                    break;
                }
            }
        }
```

Programski kod 4.5. *Prikaz prvog dijela klase ConnectedThread*

```
void write(String input) {
            byte[] msgBuffer = input.getBytes();
            try {
                mmOutStream.write(msgBuffer);
            } catch (IOException e) {

updateOnUi(getString(R.string.transmission_failed));
                finish();
            }
        }
        public void cancel() {
            try {
                btSocket.close();
            } catch (IOException e) {

updateOnUi(getString(R.string.error_closing_socket));
            }
        }
    }
```

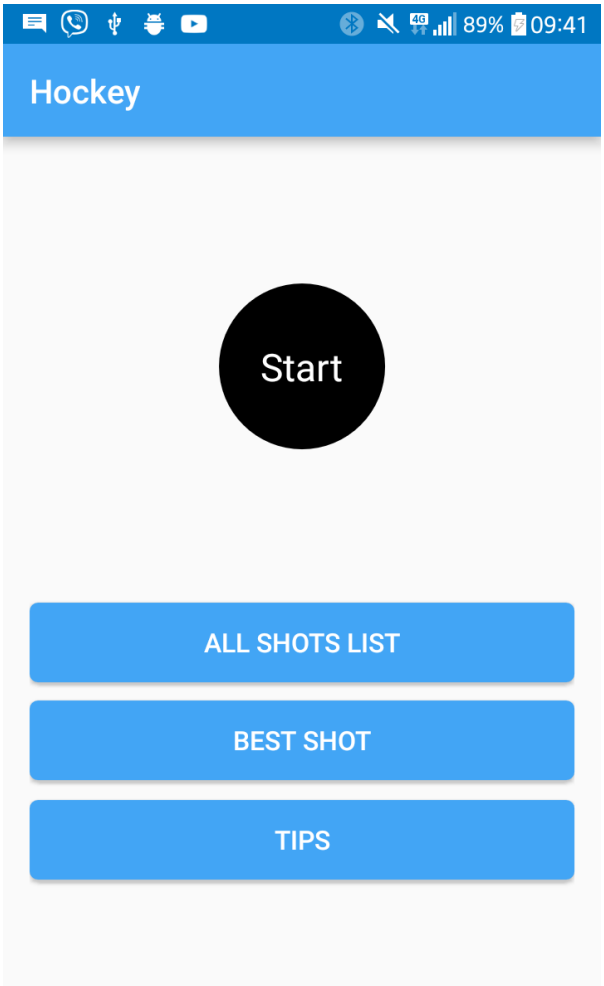
Programski kod 4.6. *Prikaz drugog dijela klase ConnectedThread*

Metoda *updateOnUi()* koristi se za ispis *Toast* poruka na glavnoj niti, a prikazana je u programskom kodu 4.7. Kao argument prima poruku objekta *String* koju ispisuje.

```
   private void updateOnUi(final String message) {
         runOnUiThread(new Runnable() {
             @Override
             public void run() {
                 Toast.makeText(MainActivity.this,
message, Toast.LENGTH_SHORT).show();
             }
         });
    }
```

Programski kod 4.7. *Prikaz koda metode updateOnUi()*

Nakon što je veza uspostavljena i primljena *Toast* poruka o uspješnom kreiranju *BluetoothSocket*-a, može se započeti korištenje gumbova u *MainActivity*-ju čiji je izgled (engl. *layout*) prikazan na Sl. 4.17.



Sl. 4.17. *Prikaz MainActivity layouta*

Budući da se protok podataka i spremanje u bazu želi ograničiti samo na određene promjene vrijednosti, jer nam podatci u stacionarnom stanju ne znače mnogo, koristi se gumb „Kreni“ koji pokreće protok podataka kroz vezu. Nakon pritiska na gumb „Kreni“, njegov se naziv mijenja u „Stani“ te se poziva metoda *startCollectingData()* što je predstavljeno programskim kodom 4.8.

```
private void startCollectingData() {
        mConnectedThread.write("1");
        startHandler();
        mTempId++;
        Toast.makeText(getApplicationContext(),
getString(R.string.collecting_data_starting),
Toast.LENGTH_SHORT).show();
    }
```

Programski kod 4.8. *Prikaz koda metode startCollectingData()*

Metoda *startCollectingData()* šalje na Arduino preko niti vrijednost „1“ te mijenja vrijednost od *boolean* varijable na Arduinu u „true“ što pokreće slanje s Arduina na pametni telefon. Nakon toga, poziva se metoda *startHandler()* koja je prikazana na programskom kodu 4.9.

```
private void startHandler() {

        mBluetoothHandlerIn = new Handler() {
            public void handleMessage(Message msg) {
                if (msg.what == handlerState) {
                    String readMessage = (String) msg.obj;
                    mRecievedDataString.append(readMessage);
                    int endOfLineIndex =
mRecievedDataString.indexOf("*");
                    if (endOfLineIndex > 0) {
                        if (mRecievedDataString.charAt(0) == '#') {
                            String[] readings =
mRecievedDataString.toString().split("\\+");

                            readings[0] = readings[0].substring(1);

                            x_gyro = Double.parseDouble(readings[0]);
                            y_gyro = Double.parseDouble(readings[1]);
                            z_gyro = Double.parseDouble(readings[2]);
                            x_accel =
Double.parseDouble(readings[3]);
                            y_accel =
Double.parseDouble(readings[4]);
                            z_accel =
Double.parseDouble(readings[5]);
                            acc = Math.sqrt((x_accel * x_accel) +
(y_accel * y_accel) + (z_accel * z_accel)); //  2g
                            force = acc * GFORCE * WEIGHT;
                            spin = Math.sqrt((x_gyro * x_gyro) +
(y_gyro * y_gyro) + (z_gyro * z_gyro)); //  250°/s
                            if ((acc > 1) || (acc < -1)) {

                                SetPodataka setPodataka = new
SetPodataka(x_gyro, y_gyro, z_gyro, x_accel, y_accel, z_accel, force,
spin, acc, mTempId);
                                boolean isAdded =
mDatabase.addData(setPodataka);
                                if (isAdded) {
                                    Toast.makeText(MainActivity.this,
getString(R.string.data_added_to_database),
Toast.LENGTH_LONG).show();
                                } else {
                                    Toast.makeText(MainActivity.this,
getString(R.string.data_not_added_to_database),
Toast.LENGTH_LONG).show();
                                }
                            }
                        }
                        mRecievedDataString.delete(0,
mRecievedDataString.length());
                    }
                }
            }
        };
    }
```

Programski kod 4.9. *Prikaz koda metode startHandler()*

Metoda *startHandler()* kreira novi *Handler* objekt koji omogućuje slanje i obradu objekta *Message*. Podaci koje Arduino šalje na pametni telefon dolaze u obliku prikazanom u programskom kodu 4.10.

```
void sendAndroidValues(){
// # zbog android app početak traženog stringa
//running = true ako se pošalje 1, false ako 0
if(running  == true){
  Serial.print('#');
  //podatci s senzora
  for(int k=0; k<6; k++)
  {
    Serial.print(sensorValue[k]);
    Serial.print('+');
  }
 Serial.print('*'); //kraj stringa
 delay(10); //delay da ne propusti  podatke
 //PRIMJER PODATKA: #PODATAK+PODATAK...+PODATAK+*
}
}
```

Programski kod 4.10. *Prikaz Arduino koda metode sendAndroidValues()*

Traženi podatci se dohvaćaju tako da se uzimaju podatci u nizu koji počinje znakom '#' i završava znakom '*'. Ti se podatci još dalje rasčlanjuju (engl. *parse)* po operatoru '+', jer on u ovom slučaju služi kao razdvojnik između traženih vrijednosti unutar ciljanog niza podataka. Nakon što su podatci odvojeni dodjeljuju se u pripadajuće varijable, vrši se računanje potrebnih informacija poput sile, akceleracije te se pravi novi objekt *SetPodataka* u kojeg se upisuju sve vrijednosti te spremaju u bazu (*SqLite*). Svi podatci koji se spreme u bazu su spremljeni pod jednim *ID*-jem, a to je *mTempId* koji se povećava svaki put kada se stisne gumb „Kreni“, te tako osigurava da se svi nadolazeći podatci spremaju pod novim *ID*-jem što označava novi udarac.

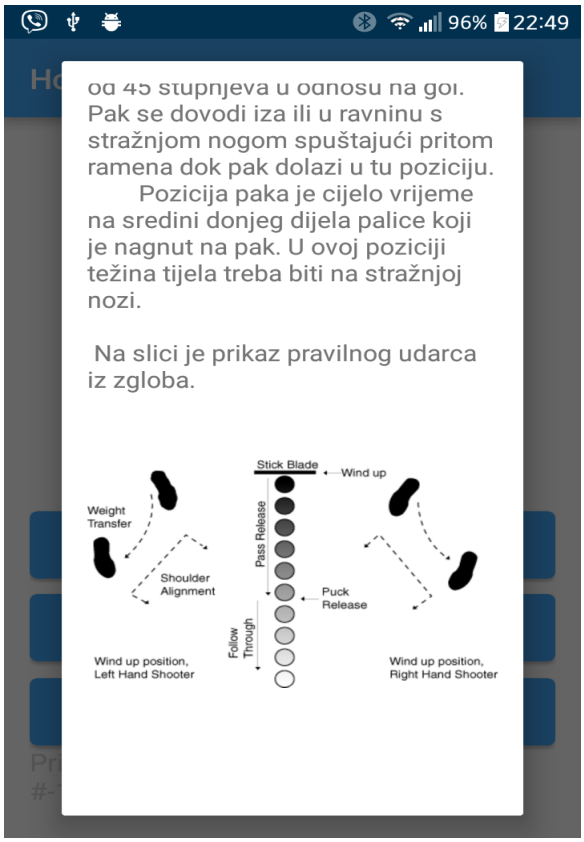
Pritiskom na gumb „Stani“ poziva se metoda *stopCollectingData()* koja je prikazana u programskom kodu 4.11.

```
private void stopCollectingData() {
        mConnectedThread.write("0");
        stopHandler();
        Toast.makeText(getApplicationContext(),
getString(R.string.collecting_data_stopping),
Toast.LENGTH_SHORT).show();
    }
```

Programski kod 4.11. *Prikaz koda metode stopCollectingData()*

Metoda *stopCollectingData()* zaustavlja prikupljanje podataka tako što šalje vrijednost „0“ na Arduino te mijenja vrijednost od *boolean* varijable na Arduinu u „false“ što zaustavlja slanje s Arduina na pametni telefon. Nakon toga, poziva se metoda *stopHandler()* koja uklanja sve poruke u *MessageQueue* handlera.

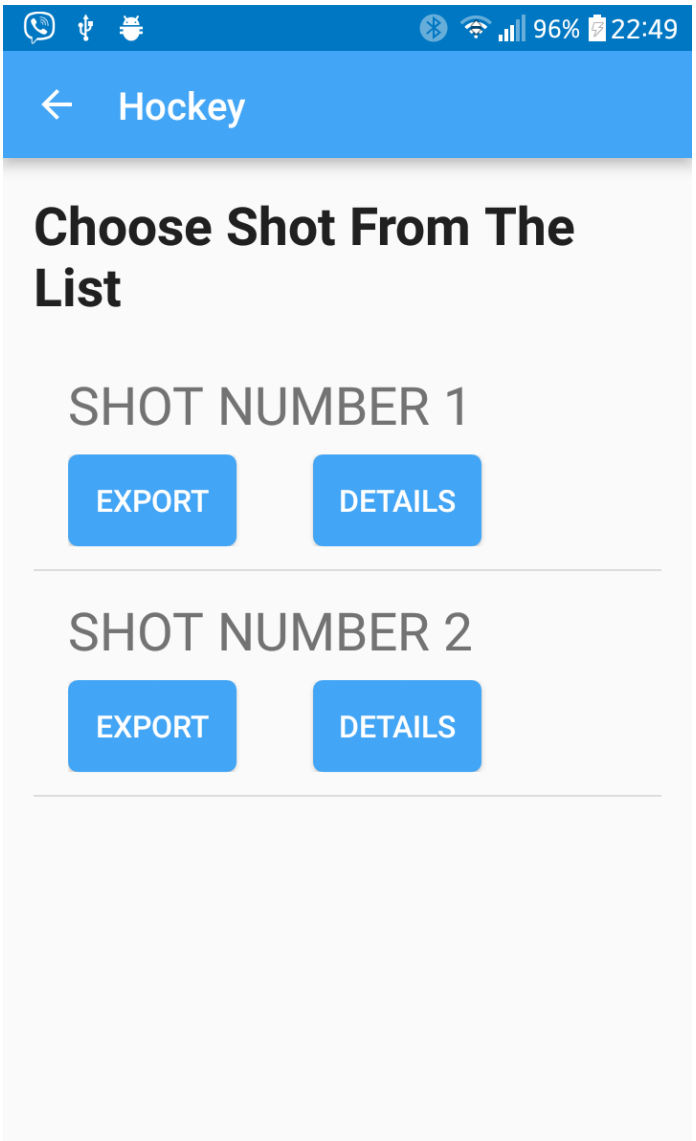
Pritiskom na button „Tips“ poziva se dijalog koji prikazuje osnovne tehnike u hokeju za izvođenje udaraca koje mogu služiti kao smjernice za poboljšanje izvedbe. Dijalog je prikazan na Sl. 4.18.



Sl. 4.18. *Prikaz dijalog gumba „Tips“*

Pritiskom na gumb „All Shots List“ poziva se nova aktivnost koja sadrži listu svih udaraca sortiranu prema *ID*-ju. Svaki od udaraca sadrži dva gumba, a to su „Details“ i „Export“. Klikom na gumb „Details“ dobiva se detaljnija statistika udarca kao što su sve vrijednosti sa senzora (može se isključiti postavljanjem varijable „showDetailed“ na false) te brzinu, silu, putanju, spin i ukupnu akceleraciju. Klikom na gumb „Export“, dobiva se mogućnost slanja podataka tog udarca u obliku .csv tablice nekim od ponuđenih načina (e-mail, viber, ...).

Na Sl. 4.19 prikazan je izgled aktivnosti *AllShotsActivity*.

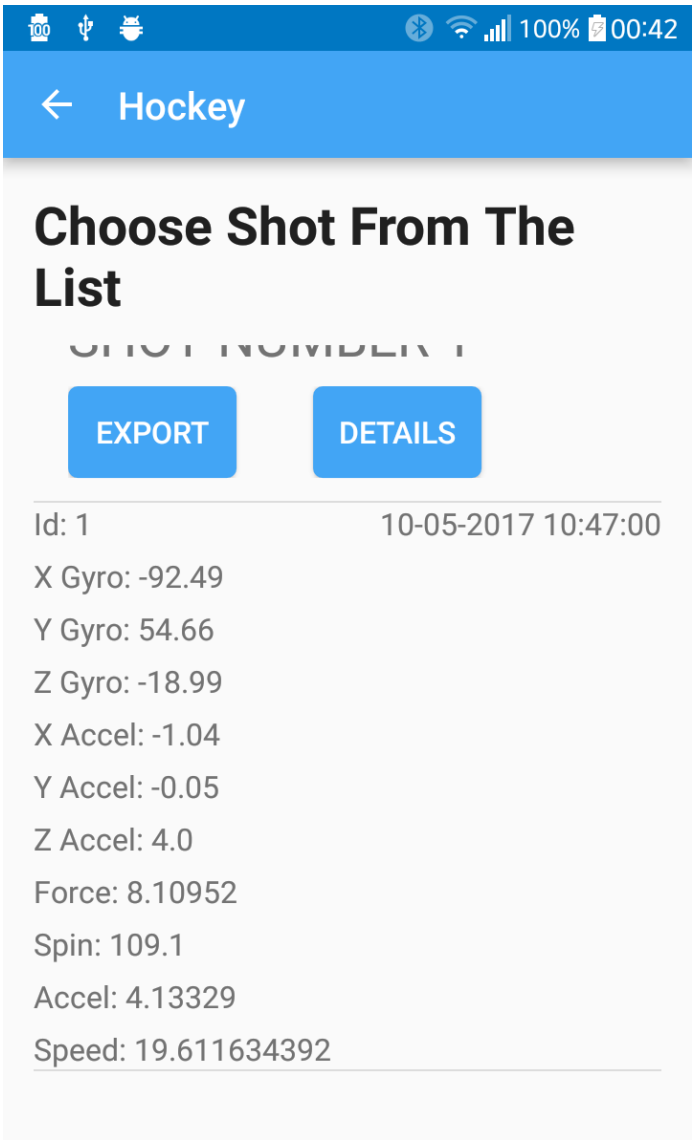


Sl. 4.19. *Prikaz AllShotsActivity aktivnosti*

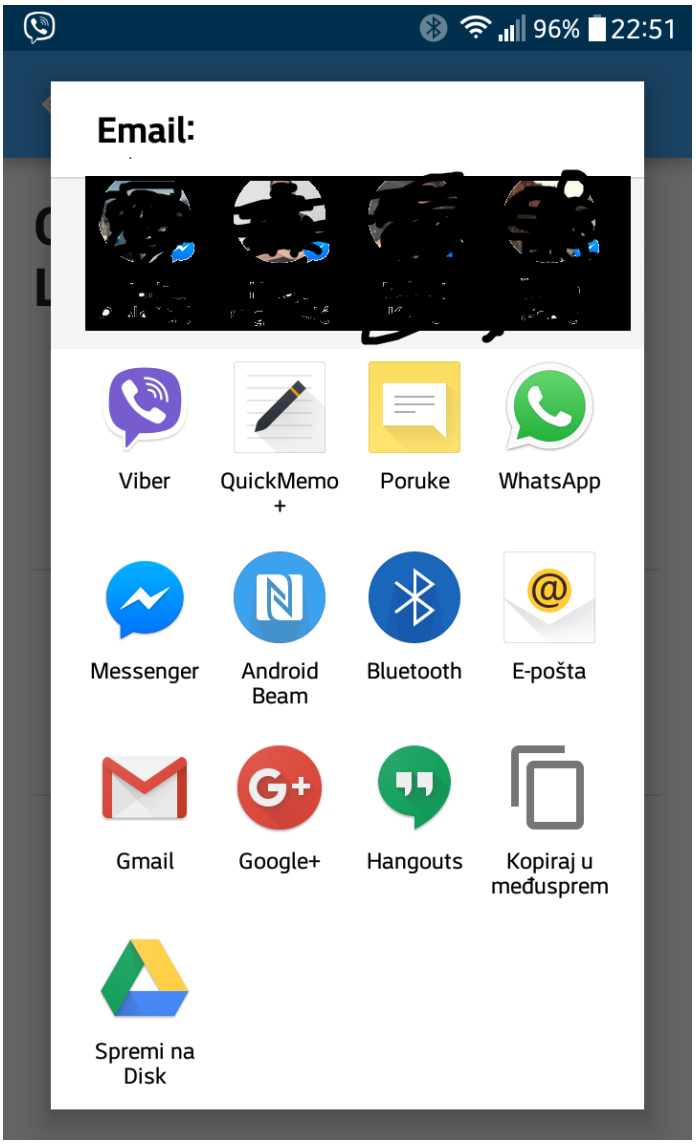
Pritiskom na gumb „Best Shot“ iz baze se dohvaća najbolji udarac po ukupnoj akceleraciji te se za njega računa sila, prijeđeni put, brzina i spin, a izgled će biti prikazan prilikom testiranja aplikacije.

4.5. Testiranje aplikacije

Aplikacija je testirana tako što su se simulirala dva udarca koja su zapisana u bazu te su prikazani rezultati. Na Sl. 4.20 prikazani su detalji udarca pod *ID*-jem broj 1 dok Sl. 4.21 prikazuje mogućnost slanja podataka određenog udarca.



Sl. 4.20. *Prikaz rezultata pristiska gumba „Details“*



Sl. 4.21. *Prikaz rezultata pristiska gumba „Export“*

Na Sl. 4.22 prikazani su detalji najboljeg udarca, odnosno sila, spin, brzina te putanja pomaka najboljeg udarca što je ujedno udarac broj 1 koji je prikazan na Sl. 4.20.

Sl. 4.20. *Prikaz rezultata pristiska gumba „Best Shot “*

5. ZAKLJUČAK

U ovom je radu objašnjena važnost Interneta stvari (IoT) u sportu, jer njegovim napretkom napreduje i sport. Na tome zasnovana rješenja omogućuju bolju analizu treninga kao i detaljniju statistiku sportskih događaja. Objašnjeni su problemi mjerenja u sportu te je dano nekoliko primjera ugradbenih računalnih sustava. Obrađene su neke od metoda koje se koriste pri mjerenju od kojih je jedna IMU tehnologija korištena u ovom radu. Nadalje su opisane sve sklopovske komponente korištene u radu koje su u suradnji s mobilnom aplikacijom stvorile jednu funkcionalnu cjelinu. Rad je pokazao koliko je truda potrebno uložiti da bi se realizirao naizgled jednostavan sustav.

LITERATURA

[1] Sveučilište u Rijeci, *Ugradbeni Računalni Sustavi.* Posjećeno 2.2.2017. na mrežnoj stranici: http://www.riteh.uniri.hr/zav_katd_sluz/zvd_elek/elektrotehnika/nastava/svel/urs/materijali/urs_uvod.pdf

[2] Grasshopper.iics (2014), *Introduction to the Internet of Things: What, Why and How.* Posjećeno 22.12.2016. na mrežnoj stranici CodeProject: https://www.codeproject.com/Articles/832492/Stage-Introduction-to-the-Internet-of-Things-Wha

[3] Movement, Health & Exercise Conference (2014) , str. 55. Posjećeno 11.9.2016. na mrežnoj stranici: http://www.acrm.org.my/ned/documents/journals/MoHE2014_Conference_Proceedings.pdf

[4] FITAPP, *Fitness Walking GPS & Running App Company*. Posjećeno 10.7.2016. na mrežnoj stranici: https://play.google.com/store/apps/details?id=com.fitapp&hl=hr

[5] Ben Popper (2014), *These high-tech gym clothes look inside your muscles to analyze your workout*. Posjećeno 17.11.2016. na mrežnoj stranici: http://www.theverge.com/2014/8/12/5992297/athos-fitness-wearable-electromyography,

[6] Adidas Smart Ball. Posjećeno 7.10.2016. na stranici: https://play.google.com/store/apps/details?id=com.adidas.smartball&hl=hr,

[7] Sam Raudabaugh (2015), *Adidas miCoach Soccer Ball*. Posjećeno 11.10.2016. na stranici: https://github.com/cornelltech/device-library/wiki/Adidas-miCoach-Soccer-Ball

[8] Steve Haake (2012), *Measuring sport in the field*. Posjećeno 2.2.2017. na mrežnoj stranici: http://richannel.org/the-sports-measurement-problem

[9] Dominic F. L. Southgate & Peter R. N. Childs & Anthony M. J. Bull (2016), *Sports Innovation, Technology and Research*. Posjećeno 17.7.2016. na mrežnoj stranici: https://books.google.hr/books?id=poj4DAAAQBAJ&printsec=frontcover&hl=hr#v=onepage&q&f=false

[10] Digital Studio Middle East Staff (2009), *Tracking motion capture*. Posjećeno 18.7.2016. na mrežnoj stranici: *http://www.digitalproductionme.com/article-1936-tracking-motion-capture/*

[11] Wikipedia, *Hokej na ledu*. Posjećeno 23.1.2017. na mrežnoj stranici: https://hr.wikipedia.org/wiki/Hokej_na_ledu

[12] A Wiley Brand dummies, *Hockey for dummies cheat sheet.* Posjećeno 12.1.2017. na mrežnoj stranici: http://www.dummies.com/sports/hockey/hockey-for-dummies-cheat-sheet/ pristupljeno 12.1.2017.

[13] A. Hache (2002), *The Physics of Hockey*. Posjećeno 2.7.2016. na mrežnoj stranici: http://klmfk.ru/books/Alain%20Hache.%20The%20Physics%20of%20Hockey.pdf,

[14] HockeyShot.com (2017), *Wrist Shot - Hockey Shooting Tips & Instructions*. Posjećeno na mrežnoj stranici: https://www.hockeyshot.com/shooting-tips/wrist-shot

[15] *The Physics Behind hockey*. Posjećeno 11.3.2017. na mrežnoj stranici: http://physicsofhockeyproject.weebly.com/shooting.html,

[16] Exploratorium (2010), *Science of Hockey, Collisions on the Ice*. Posjećeno 30.11.2016. na mrežnoj stranici: https://www.exploratorium.edu/hockey/checking2.html,

[17] Arduino (2017), *Arduino Mega*. Posjećeno 15.8.2016. na mrežnoj stranici: https://www.arduino.cc/en/Main/arduinoBoardMega

[18] Gadget Gangster (2010), *Accelerometer & Gyroscope Tutorial*. Posjećeno 19.9.2016. na mrežnoj stranici: http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/

[19] Jsumo Ultimate Robot Parts , *MPU – 6050 GY521 Cheap 6DOF IMU (Accelerometer & Gyro).* Posjećeno 5.7.2016. na mrežnoj stranici: http://www.jsumo.com//mpu6050-gy521-cheap-6dof-imu-accelerometer-gyro-127-82-O.jpg

[20] InvenSense Inc (2013), *MPU – 6000 and MPU – 6050 Product Specification Revision 3.4.* Posjećeno 2.7.2016. na mrežnoj stranici: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf,

[21] Wikipedia, *Bluetooth.* Posjećeno 17.7.2016. na mrežnoj stranici: https://hr.wikipedia.org/wiki/Bluetooth,

[22] E-radionica, *KKM: BLUETOOTH*. Posjećeno 19.8.2016. na mrežnoj stranici: https://e-radionica.com/hr/blog/2016/04/06/bluetooth/

[23] Shah Saifur Rahman (2014), *AT Command Mode of HC-05 and HC-06 Bluetooth Module.* Posjećeno 22.7.2016. na mrežnoj stranici: http://www.instructables.com/id/AT-command-mode-of-HC-05-Bluetooth-module/

[24] Statista (2016), *Global mobile OS market share in sales to end users from 1st quarter 2009 to 1st quarter 2016*. Posjećeno 26.8.2016. na mrežnoj stranici: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/ ,

[25] Brian W. Evans (2007), *Arduino Programming Notebook.* Posjećeno 17.8.2016. na mrežnoj stranici: http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf

[26] Ed Burnette (2010), *Android System Architecture*. Posjećeno 28.8.2016. na mrežnoj stranici: https://kebomix.wordpress.com/2010/08/17/android-system-architecture/

[27] Android Studio, *Meet Android Studio.* Posjećeno 30.8.2016. na mrežnoj stranici: https://developer.android.com/studio/intro/index.html

[28] *Fritzing, electronics made easy.* Posjećeno 11.8.2016. na mrežnoj stranici: http://fritzing.org/home/

SAŽETAK

U radu je detaljno opisan nastanak mobilne aplikacije za praćenje treninga u hokeju. Dana je teorijska podloga problema mjerenja u sportu te opis pojedinih elektroničkih komponenata korištenih u radu. Opisane su neke od uspješnih metoda mjerenja u sportu te su dani primjeri nekoliko ugradbenih sustava u sportu od kojih i AdidasMiCoach koji se bio osnova ovoga rada. U praktičnom dijelu implementirano je rješenje koje pomoću navedenih elektroničkih komponenti komunicira s aplikacijom te daje tražene rezultate. Na kraju je aplikacija testirana te je prikazano nekoliko dobivenih rezultata.

Ključne riječi: akceleracija, *Arduino*, hokej, *IoT*, *IoT* u sportu

ABSTRACT

Making device and software for hockey puck for measuring training parameters

This thesis describes the process of developing a mobile application for hockey training monitoring. It gives a theoretical background of measurements problems in the sport as well as a description of all electronic components used. Some of the successful methods of measuring in sports and few embedded systems where one of them is AdidasMiCoach which was base of this project, were described. In the practical part, the solution has been implemented using electronic components to communicate with application to produce desired results. In the end, application was tested and few given results were shown.

Keywords: acceleration , *Arduino*, hockey, *IoT*, *IoT* in sports

ŽIVOTOPIS

Sandro Mustapić rođen je 23.10.1992. godine u Slavonskom Brodu. Osnovnu školu završava u Slavonskom Brodu. U 2007. godini upisuje opću gimnaziju koju završava s vrlo dobrim uspjehom. U 2011. godini upisuje Sveučilišni preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku. Nakon završetka preddiplomskog studija 2013. godine upisuje Sveučilišni diplomski studij procesnog računarstva.

PRILOZI

- Diplomski rad u .docx formatu
- Diplomski rad u pdf formatu
- Programsko rješenje projekta