



Elektrotehnički fakultet Univerziteta u Sarajevu

Odsjek za računarstvo i informatiku



*Primjena Arduino platforme za
pomoć osobama oštećenog vida*

Mentor:

Vanr. prof. dr Samir Ribić

Student:

Timur Čerimagić

Sarajevo, 2016



Elektrotehnički fakultet Univerziteta u Sarajevu

Odsjek za računarstvo i informatiku



*Primjena Arduino platforme za
pomoć osobama oštećenog vida*

Mentor:

Vanr. prof. dr Samir Ribić

Student:

Timur Čerimagić

Sarajevo, 2016

Sažetak

Arduino predstavlja *open-source* platformu za razvoj elektroničkih projekata čija popularnost od 2005. ide isključivo uzlaznom putanjom i tek treba da doživi svoj vrhunac. Kako raste broj ljudi koji doprinose Arduino zajednici, platforma se ubrzano razvija i postaje sve pristupačnija u smislu dostupnih materijala i tutorijala na Internetu. Druga platforma, koja je prije nešto više od godinu dana ugledala svjetlost dana, a sa Windows 10 operativnim sistemom doživjela procvat, nudi niz benefita, a jedan od njih je Cortana personalni asistent. Naziv joj je Universal Windows Platform i intuitivno se da zaključiti ideja koja stoji iza naziva - univerzalnost. Stoga, razvijeno je rješenje koje koristi ove dvije tehnologije zajedno sa Bluetooth komunikacijom za prijenos podataka između njih. Podatke prikupljaju senzori na Arduino strani, a obradu informacija obavlja UWP aplikacija. Obrada podrazumijeva njihovu interpretaciju i konvertovanje u 3D zvuk. Sve ovo je implementirano u svrhu simulacije osjećaja za orijentaciju u prostoru kod slijepih i slabovidnih osoba.

Cilj ovog rada je uvezati sve navedene tehnologije i proizvesti bazni model uređaja koji bi mogao doprinijeti zajednici.

Abstract

Arduino represents an open-source platform for the development of electronic projects, whose popularity since 2005. was going only upwards and have yet to experience its peak. As the number of people who contribute to the Arduino community rises, the platform develops rapidly and becomes more accessible in terms of available materials and tutorials on the Internet. Another platform that a little over a year ago saw the daylight, and with Windows 10 operating system flourished, offers a range of benefits, one of which is Cortana personal assistant. That platform is called Universal Windows Platform and one can intuitively infer idea behind the title - universality. Therefore, a solution that uses these two technologies along with Bluetooth communication to transfer data between them was developed. The data is collected by sensors on the Arduino side, and information processing is performed by UWP application. Processing involves their interpretation and conversion to 3D sound. All this has been implemented for the purpose of simulating a sense of spatial orientation for blind and visually impaired people.

The aim of this thesis is to bind all of these technologies and produce base model of the device that could contribute to the community.

Postavka rada

<i>Naslov:</i>	Primjena Arduino platforme za pomoć osobama oštećenog vida
<i>Kurs:</i>	Operativni sistemi
<i>Student:</i>	Timur Ćerimagić
<i>Cilj:</i>	Dizajn i implementacija sistema za navigaciju slijepih i slabovidnih lica uz pomoć senzora na Arduino platformi, UWP aplikacije, Bluetooth tehnologije i 3D zvuka
<i>Opis:</i>	Potrebno je projektovati sistem na bazi Arduino mikrokontrolera koji sa Universal App aplikacijom komunicira koristeći Bluetooth tehnologiju u cilju slanja očitanih i procesiranih vrijednosti sa senzora na strani mikrokontrolera, te analize dobijenih rezultata i aktivacije određenih zvukova na strani UWP aplikacije
<i>Očekivani rezultati:</i>	Funkcionalan sistem za kretanje u prostoru

Mentor: Vanr. prof. dr Samir Ribić

Akronimi

- **AREF** – AnalogREference
- **BT** - Bluetooth
- **DIY** – Do It Yourself
- **EEPROM** - Electrically Erasable Programmable Read-Only Memory
- **GND** - Ground
- **HRTFS** – Masking and Head-Related Transfer Functions
- **ICSP** - In-Circuit Serial Programming
- **IDE** - Integrated Development Environment
- **ILD** – Inter-aural Level Difference
- **ITD** – Inter-aural Time Difference
- **LED** – Light Emitting Diode
- **MISO** – Master In Slave Out
- **MOSI** – Master Out Slave In
- **PCB** - Printed circuit board
- **PWM** – Pulse Width Modulation
- **RX** – Recieve pin
- **SCK** – Serial Clock
- **SCL** – Clock line
- **SDA** – Data line
- **SDK** – Software Development Kit
- **SPI** - Serial Peripheral Interface
- **SPP** - Serial Port Protocol
- **SRAM** – Static random-access-memory
- **SS** – Slave Select
- **TTL** - Transistor–transistor logic
- **TX** – Transfer pin
- **UART** - Universal asynchronous receiver/transmitter
- **UI** – User Interface
- **USB** – Universal Serial Bus
- **UWP** – Universal Windows Platform
- **XAML** - Extensible Application Markup Language

Sadržaj

Sažetak	i
Abstract.....	i
Postavka rada.....	ii
Akronimi.....	iii
Spisak slika	1
Spisak tabela	2
1. Uvod.....	3
2. Teorijski uvod.....	4
2.1. Šta je Arduino platforma.....	4
2.1.1. Arduino platforma kroz historiju [5] [6] [7]	5
2.1.2. Verzije Arduino uređaja	7
2.1.3. Arduino Uno [11]	9
2.1.4. Digitalni ulazi/izlazi i analogni ulazi.....	12
2.1.5. Senzori.....	17
2.1.6. Cross development za Arduino i Arduino IDE [21].....	19
2.2. Općenito o Universal App (UWP) platformi.....	25
2.2.1. Universal Windows Application	25
2.2.2. Visual Studio 2015 i UWP plugin [27]	31
2.3. Bluetooth [28].....	32
2.3.1. Upotreba Bluetooth tehnologije.....	33
2.4. 3D (Binaural – dva uha) zvuk [29]	34
2.4.1. Tehnika snimanja [30].....	35
2.4.2. Reproduciranje zvuka [30]	35
3. Praktični dio	36
3.1. Funkcionalni opis sistema	36
3.2. Uspostavljanje platforme.....	37
3.2.1. HC-SR04 ultrazvučni senzor udaljenosti	37
Senzori [31]	38
3.2.2. Arduino Uno	38
3.2.3. HC-05 Bluetooth modul [28]	39
3.2.4. UWP aplikacija	40
3.2.5. 3D zvuk.....	42
3.3. Arduino aplikacija.....	43
3.3.1. Biblioteka New Ping [32] [33]	43

3.3.2. Arduino kod.....	44
3.4. UWP aplikacija	47
3.4.1. UWP kod	48
4. Zaključak.....	55
5. Bibliografija	56
Dodatak A.....	59
A.1. Shema sistema	59
B.1. Slike sistema.....	60

Spisak slika

Slika 2-1 Arduino logo [4].....	4
Slika 2-2 Osnivači Arduino-a	5
Slika 2-3 BASIC Stamp mikrokontroler [8]	6
Slika 2-4 Arduino Mega [10].....	8
Slika 2-5 Arduino LilyPad [35]	9
Slika 2-6 Arduino MKR1000 [36]	9
Slika 2-7 Arduino Uno shema.....	10
Slika 2-8 Primjeri analognog signala [13]	13
Slika 2-9 Analogne i diskretne vrijednosti.....	14
Slika 2-10 Poruka kodirana jedinicama i nulama [15]	14
Slika 2-11 Digitalni izlaz sa par tranzistora [16]	15
Slika 2-12 Digitalni ulaz koji možemo direktno povezati sa napajanjem [16].....	15
Slika 2-13 PWM duty-cycle [17]	16
Slika 2-14 HC-SR04 ultrazvučni senzor udaljenosti [18].....	17
Slika 2-15 Lociranje objekta pomoću ultrazbučnog senzora [19]	18
Slika 2-16 LM35 temperaturni senzor [20]	18
Slika 2-17 Arduino IDE 1.6.8.....	21
Slika 2-18 Universal Windows Apps na različitim uređajima [23]	25
Slika 2-19 Algoritam skaliranja 24px fonta na različitim uređajima [26]	27
Slika 2-20 Repozicija elemenata na UI-u [26]	28
Slika 2-21 Promjena veličine elemenata na UI-u [26].....	28
Slika 2-22 Reflow [26]	29
Slika 2-23 Prikazivanje i sakrivanje dijelova UI-a [26]	29
Slika 2-24 Zamjena elemenata [26].....	30
Slika 2-25 Potpuna reorganizacija [26]	30
Slika 2-26 Mogući uređaji koji mogu biti emulirani	31
Slika 2-27 Pokretanje aplikacije u različitim okruženjima.....	31
Slika 2-28 ITD [29]	34
Slika 2-29 ILD [29]	34
Slika 2-30 HRTFS [29]	34
Slika 3-1 FeelTheSpace shema	37
Slika 3-2 Bluetooth modul HC-05 [28]	40
Slika 3-3 Mobilna verzija FeelTheSpace aplikacije	41
Slika 3-4 Desktop verzija FeelTheSpace aplikacije	41
Slika 3-5 Udaljenosti i područja očitavanja senzora.....	42
Slika A-1 Shema sistema	59
Slika B-1 Sistem u cjelosti.....	60
Slika B-2 Pojas sa senzorima i mobilni telefon.....	60
Slika B-3 Slušalice	60

Spisak tabela

Tabela 2-1 Prikaz aktuelnih Arduino uređaja [9]	7
Tabela 2-2 Arduino Uno tehnički detalji	11
Tabela 2-3 Usporedba Programino i Visual Micro razvojnih okruženja.....	20

1. Uvod

Arduino platforma je trenutno najzastupljenija platforma za razvijanje elektroničkih projekata i prototipova u svijetu ugradbenih sistema i sve popularnijeg interneta stvari (*Internet of Things*). Ovaj status je postignut s razlogom. Odlikuju je modularnost, univerzalnost, dostupnost, kao i pristupačnost obzirom na *open-source* zajednicu koja joj doprinosi i sve diže na viši nivo. Na taj način se ruše predrasude i barijere između ljudi neinženjerskih i inženjerskih pozadina. Inovatori, studenti, amateri, profesionalci, programeri i ostali zainteresovani imaju mogućnost kreiranja inovativnih elektroničkih projekata koji doprinose zajednici.

U svrhe ovog rada razvijeno je rješenje za pomoć slijepim i slabovidnim osobama pod nazivom „FeelTheSpace“. Ovo rješenje predstavlja pomoć pri kretanju kroz prikupljanje informacija o udaljenosti objekata, odnosno potencijalnih prepreka putem odgovarajućih senzora. Prikupljeni podaci se interpretiraju i predstavljaju korisniku u vidu 3D zvuka koji simulira osjećaj za orijentaciju u prostoru. Kako bi izvedba rješenja bila jasnija, kroz teorijski dio biće prikazana generalna slika 4 tehnologije koje su korištene u praktičnom dijelu. To su Arduino, UWP, Bluetooth i 3D zvuk.

Kako je praktični dio hardversko-softverskog karaktera, najprije je objašnjeno samo jezgro hardvera - Arduino platforma. Dat je historijski osvrt na razvoj platforme i detaljno pojašnjenje trenutno najpopularnijeg modela – Uno. Posebna pažnja je posvećena senzorima i Arduino razvojnom okruženju. Osnovu softverske strane čini UWP aplikacija. Predstavljene su prednosti razvoja rješenja za navedenu platformu, te njene mogućnosti. S obzirom da je Visual Studio 2015 jedino razvojno okruženje koje podržava UWP, date su smjernice za njegovo korištenje. Spona između prethodne dvije cjeline je Bluetooth tehnologija, odnosno modul za Arduino kojim se ostvaruje komunikacija između njih. Navedeni su ključni parametri komunikacije i generalna upotreba Bluetooth prijenosa podataka u svakodnevnicu. Krajnji produkt i rezultat interakcije sva tri sistema je 3D zvuk. Radi se o metodi snimanja zvuka koja je upotrebljena u svrhe ovog rješenja, te su prezentovani njeni koncepti, tehnike snimanja i reprodukcija. U okviru praktičnog dijela dat je funkcionalni opis sistema kao i detalji o korištenim komponentama. Na kraju rada priloženi su ključni dijelovi Arduino koda i UWP koda.

2. Teorijski uvod

2.1. Šta je Arduino platforma

Izvorno Arduino predstavlja *open-source* platformu namijenjenu za pravljenje elektroničkih prototipova i projekata. Bazirana je na fleksibilnom hardveru i softveru koji u kombinaciji sa dobro dizajniranim korisničkim interfejsom čini korištenje veoma jednostavnim. Arduino je kao takav namijenjen širokom spektru publike različitih tehničkih pozadina; inovatorima, dizajnerima, studentima, ljudima koji to rade iz zabave i generalno bilo kome ko je zainteresovan za kreiranje interaktivnih uređaja i samih okruženja. Sastoji se iz programibilne fizičke matične ploče (PCB) koja se često naziva mikrokontroler i dijela softvera pod nazivom IDE (*Integrated Development Environment*) koji predstavlja okruženje za razvoj aplikacija i pomoću kojeg se napisani kod sa kompjutera preko USB komunikacije prebacuje na prethodno navedeni mikrokontroler. U biti se osnova ove platforme bazira na mogućnosti očitavanja različitih tipova ulaza, obradi tih ulaza, te pretvaranju i slanju rezultata na izlaze. Na strani ulaza mogu biti različiti senzori, kao na primjer senzor za očitavanje temperature, vlage, osvjetljenja ili pak dugme, kao jedan tip senzora, dok se na izlaznoj strani obično nalaze LED lampice, motori, ekrani i slično, koji se jednim imenom nazivaju aktuatori. Više o detaljima u nastavku. Da bi mikrokontroleru rekli šta da radi, potrebno mu je poslati set instrukcija. Kod koji će biti preveden u mikrokontroleru poznate instrukcije se piše u Arduino programskom jeziku koji je mješavina C i C++ jezika, sa određenim dodacima za korištenje ulaza i izlaza. [1] [2] [3]

Sljedeća rečenica predstavlja citat sa arduino.cc stranice o tome šta je Arduino i u ovom radu će biti naveden u originalnoj formi na engleskom jeziku i u slobodnom prijevodu:

“Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.”
[1]

Slobodni prijevod:

“Tijekom godina Arduino je bio “mozak” hiljadama projekata, od svakodnevnih objekata do kompleksnih naučnih instrumenata. Svjetska zajednica stvaralaca- studenata, amatera, umjetnika, programera i profesionalaca- se okupila oko ove *open-source* platforme i njihovi doprinosi su proizveli nevjerovatnu količinu dostupnog znanja koje može biti od pomoći podjednako početnicima i ekspertima.”



Slika 2-1 Arduino logo [4]

2.1.1. Arduino platforma kroz historiju [5] [6] [7]

Kolumbijski student Hernando Barragán je 2004. godine u svrhe svog magistarskog rada kreirao razvojnu platformu pod nazivom “*Wiring*”¹ na *Interaction Design Institute Ivera (IDII)* univerzitetu u Ivrei, Italija. Mentori su mu bili Massimo Banzi i Casey Reas (duo poznat po svom radu na programskom jeziku *Processing*²). Cilj je bio kreirati jeftine i jednostavne alate namijenjene ljudima neinženjerskih pozadina za pravljenje digitalnih projekata. *Wiring* platforma se sastojala od matične ploče na kojoj je bio ATmega128 mikrokontroler, IDE-a baziranog na *Processing* programskom jeziku i bibliotečnim funkcijama za jednostavno programiranje mikrokontrolera.

2005. godine Massimo Banzi, zajedno sa David-om Mellis-om (tadašnjim IDII studentom) i David-om Cuartielles-om, je dodao podršku *Wiring* platformi za jeftiniji ATmega8 mikrokontroler. Umjesto da su nastavili raditi na *Wiring* platformi, oni su preuzeli njen izvorni kod, odvojili se i počeli ga koristiti kao odvojen projekat, nazvan Arduino. Interesantno, Arduino potiče od naziva bara u Ivrei (izvorni naziv: 'Bar Di Re Arduino') koji je dobio naziv po kralju Arduinu, a gdje su se neki od osnivača znali družiti.

Prije nego što prođem detaljnije kroz historiju Arduino firme i same platforme, spomenut ću osnivače i članove razvojnog tima. To bi bili : Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, i David Mellis



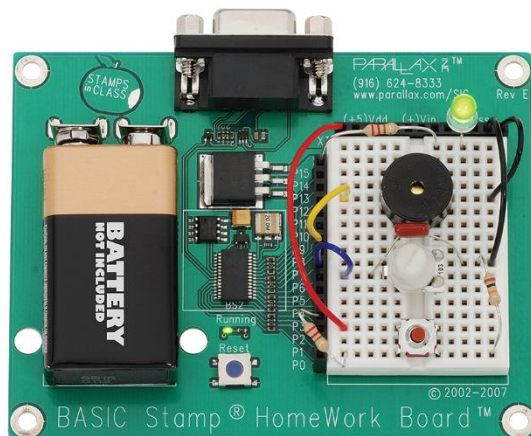
Slika 2-2 Osnivači Arduino-a

Banzi je 2002. regrutovan od strane IDII-a kao vanredni profesor kako bi promovisao nove načine interaktivnog dizajna. Došao je sa velikim brojem ideja, ali s obzirom na ograničen budžet i manjak časova realizacija mu nije pošla za rukom. Kao i većina njegovih kolega, Banzi se oslanjao na BASIC Stamp mikrokontroler koji je u to vrijeme bio u upotrebi već nekih desetak godina. Programirao se BASIC programskim jezikom i sastojao od jednostavne ploče sa mjestom za napajanje, memorijom, mikrokontrolerom i ulaznim/izlaznim portovima na koje se vezao hardver. Po Banzi-u bila su tri problema u vezi sa tim mikrokontrolerom. Prvo, bio je previše skup za potrebe studenata; koštao je

¹ Wiring – *open-source cross-platform* razvojna platforma za pravljenje elektroničkih projekata koja se sastoji iz programskog jezika, IDE-a i mikrokontrolera; Hernando Barragán ju je razvio 2003. godine

² Processing – *open-source* programski jezik i IDE primarno namijenjen za programiranje elektroničkih uređaja

nekih 100 dolara sa osnovnim setom dijelova. Drugi problem je bio u procesorskoj snazi. Banzi je smatrao da nema dovoljnu snagu da podrži projekte na kojim bi on sa svojim studentima radio i treći problem je bio u podršci za Macintosh kompjutere koji su se u velikoj mjeri koristili na IDII univerzitetu.



Slika 2-3 BASIC Stamp mikrokontroler [8]

U to vrijeme Banzi-ev kolega na MIT-u (Massachusetts Institute of Technology) je razvijao programski jezik pod nazivom „Processing“. *Processing* je brzo postao popularan među dizajnerima, programerima po profesiji, pa čak i amaterima programerima jer je omogućavao kreiranje kompleksnih vizualizacija podataka na vrlo jednostavan način. Banzi-u se dopao koncept *Processing* jezika i to ga je navelo na razmišljanje kako bi on sa svojim timom mogao napraviti nešto slično, samo da osim vizualizacije podataka mogu kodirati mikrokontroler. Prvi korak ka tom cilju je ostvario Hernando Barragán sa svojim magistarskim radom. On je uz pomoć svojih mentora Banzi-a i Reas-a razvio novu platformu pod nazivom „Wiring“ koja se sastojala iz korisnički prihvatljivog (*user-friendly*) IDE-a i spremne štampane pločice. Banzi je imao veće i ambicioznije ciljeve, tako da je htio napraviti platformu koja je još jeftinija, jednostavnija i lakša za korištenje. Zacrtni cilj je ostvario 2005. i te godine je napravio prvi prototip štampane ploče. Na prvi mah nije nazvana Arduino; to se desilo tek kasnije.

Sljedeća velika odluka se ticala licence platforme i pratećeg softvera. Cijeli tim je duboko vjerovao u *open-source* licencu. Da bi što brže kreirali lako dostupnu platformu, bili su mišljenja da bi bilo bolje omogućiti rad na projektima što većem broju ljudi. Veliki faktor u tom trenutku bila je i finansijska situacija na univerzitetu. Univerzitet nije imao više novca i zatvaranje je bilo relativno neizbježno. Među studentima i osobljem fakulteta je nastupio veliki strah da će projekti propasti ili da će doći u pogrešne ruke, tako da je Banzi odlučio učiniti Arduino platformom otvorenog koda.

Da bi to sve ispalo kako treba morali su naći pogodnu licencu, što nije bio baš jednostavan zadatak, obzirom da se u to vrijeme pretežno licencirao softver. Odlučili su na cijelu stvar pogledati iz drugog ugla i projekat su licencirali koristeći licencu firme „Creative Commons“, koja je pretežno licencirala umjetnička i muzička djela. Prema Banzi-u „hardver je dio kulture koji mora biti podijeljen sa drugim ljudima!“.

Naredni korak je bio napraviti pločicu. Tim se odlučio za cijenu od 30 dolara, jer je primarno bila namijenjena studentima i samim tim pristupačnost je igrala veliku ulogu. Dvije stvari koje su doprinijele cjelokupnom *DIY (Do It Yourself)* izgledu su plava boja pločice i karta Italije odštampana na

poledini. Proizvod se sastojao od jeftinih dijelova koji su se mogli lako naći i kupiti za slučaj da neko od korisnika želi napraviti vlastitu pločicu, a shemu su mogli besplatno preuzeti na arduino web stranici. Sistem je bio spreman za korištenje bez dodatnih ulaganja i kupovanja dijelova. U drugu ruku, mnogo drugih sistema je zahtijevalo kupovinu dijelova koji su povećavali troškove.

Posljednji korak je bio čisti test koliko će ljudi biti zainteresovano za rad na toj platformi. Tim je podijelio 300 praznih štampanih ploča studentima IDII-a sa jednom jednostavnom uputom: „Pogledajte upute za sastavljanje dostupne na Internet-u, napravite vlastitu ploču i iskoristite je da napravite nešto.“ Kreirano je mnogo projekata, među kojima je i sat koji je služio kao alarm koji visi iznad glave i kako mu se približavate da ga odgodite ili ugasite, on se podizao prema plafonu i tjerao vas da ustanete. Vrlo brzo su ljudi čuli za dotičnu platformu i poželjeli da je imaju.

Projekat je već odmakao, a nije imao adekvatno ime i tada je rođen Arduino.

2.1.2. Verzije Arduino uređaja

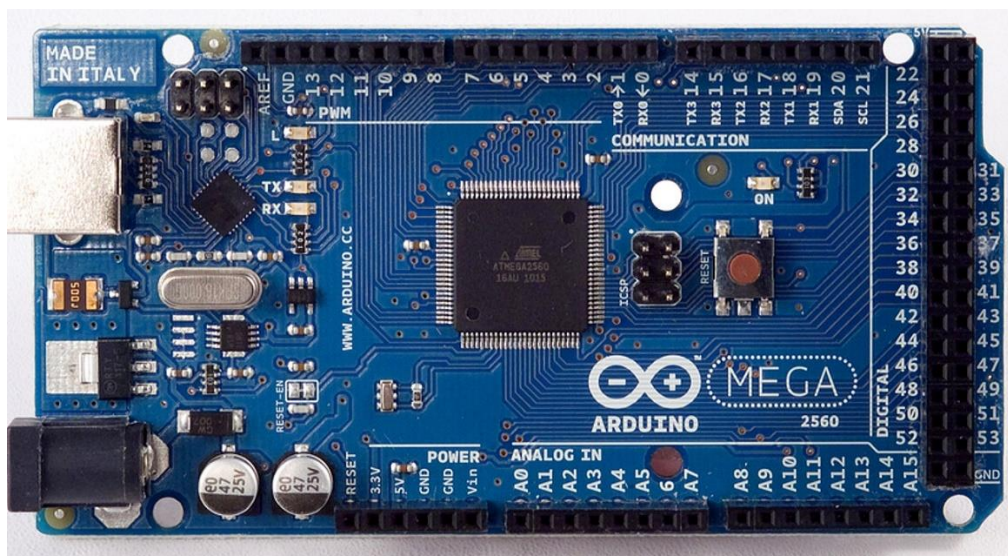
Kroz historiju na Arduino platformi su se mijenjali mikrokontroleri, dodavale nove mogućnosti u vidu različitih tipova komunikacije (sa drugim kontrolerima i uređajima), različite frekvencije rada, količina memorije, naponi koje podržava i slično; što je na kraju dovelo i do kreiranja različitih verzija Arduino uređaja sa različitim karakteristikama. Sljedeća tabela predstavlja spisak verzija i neke od njihovih primarnih karakteristika.

Tabela 2-1 Prikaz aktuelnih Arduino uređaja [9]

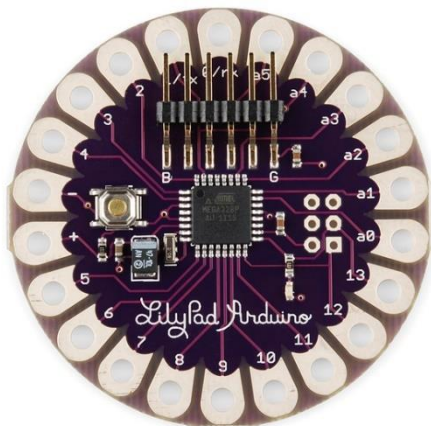
Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
IO1	Intel® Curie	3.3 V / 7-12V	32MHz	6/0	14/4	-	24	196	Regular	-
Gemma	ATtiny85	3.3 V / 4-16 V	8 MHz	1/0	3/2	0.5	0.5	8	Micro	0
LilyPad	ATmega168V ATmega328P	2.7-5.5 V / 2.7-5.5 V	8MHz	6/0	14/6	0.512	1	16	-	-
LilyPad SimpleSnap	ATmega328P	2.7-5.5 V / 2.7-5.5 V	8 MHz	4/0	9/4	1	2	32	-	-
LilyPad USB	ATmega32U4	3.3 V / 3.8-5 V	8 MHz	4/0	9/4	1	2.5	32	Micro	-
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
Micro	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
MKR1000	SAMD21 Cortex-M0+	3.3 V / 5V	48MHz	7/1	8/4	-	32	256	Micro	1
Pro	ATmega168 ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	0.512 1	1 2	16 32	-	1
Pro Mini	ATmega328P	3.3 V / 3.35-12 V 5 V / 5-12 V	8 MHz 16 MHz	6/0	14/6	1	1	32	-	1
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	-	32	256	2 Micro	2

U današnje vrijeme najpopularnije verzije Arduino platforme su Uno, Mega i nekadašnji Yun, koji se više ne proizvodi. Po nazivu ATmega procesora vidimo da preovladavaju 32kB-tni modeli (prva dva broja u nazivu procesora ATmega328P ili ATmega32U4 predstavljaju 32kB integrirane *flash* memorije). Radne voltaže su približno slične; LilyPad - 2.7 V, preko 3.3 V, te Mega i Uno 5V. Što se frekvencije tiče vidimo da je LilyPad rađen sa 8MHz frekvencijom, dok su Mega i Uno rađeni sa procesorom koji radi na 16 MHz, što je sasvim dovoljno za većinu projekata. Mega je u značajnoj prednosti u pogledu analognih ulaza, digitalnih ulaza/izlaza i PWM (*Pulse Width Modulation*) portova. Na primjer, Mega ima 54 digitalna U/I porta od kojih je 15 sa mogućnošću PWM-a, dok Uno ima samo 14, od kojih 6 ima omogućen PWM. Slična situacija je i za analogne ulaze. Razlike su i u UART (*Universal asynchronous receiver/transmitter*) portovima; 4:1 u korist Mega verzije u odnosu na Uno i posljednja bitna razlika se odnosi na memoriju. Mega posjeduje 256kB *flash* memorije, dok Uno ima 32kB.

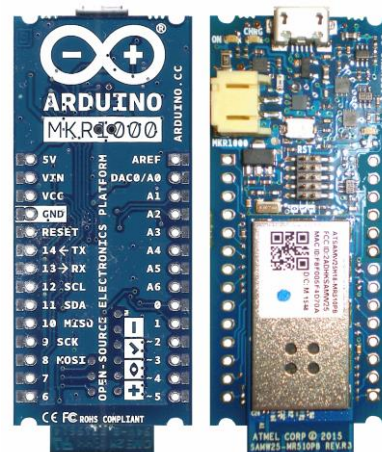
Vidimo da su dizajneri i kreatori Arduino platforme odlučili napraviti cijeli spektar različitih uređaja koji mogu pronaći svoju primjenu u isto tako širokom spektru različitih projekata. Ako je potrebna velika frekvencija procesora, dosta memorije i kompaktan dizajn, ali mali broj pinova, dobro rješenje je MKR1000 mikrokontroler. Za projekte gdje se koristi veći broj ulaznih i izlaznih uređaja sa dosta memorije kako bi se moglo baratati podacima koje proizvode, ali nije pretjerano bitna kompaktnost mikrokontrolera, Mega je idealno rješenje. Ako je pak dovoljan nešto manji broj različitih portova i nešto manja memorija (u odnosu na Mega verziju), ali dobra procesorska moć, tu je Uno dobro rješenje.



Slika 2-4 Arduino Mega [10]



Slika 2-5 Arduino LilyPad [35]



Slika 2-6 Arduino MKR1000 [36]

S obzirom da sam za potrebe praktičnog dijela ovog rada iskoristio Uno, u narednom poglavlju ću detaljnije obraditi njegove karakteristike.

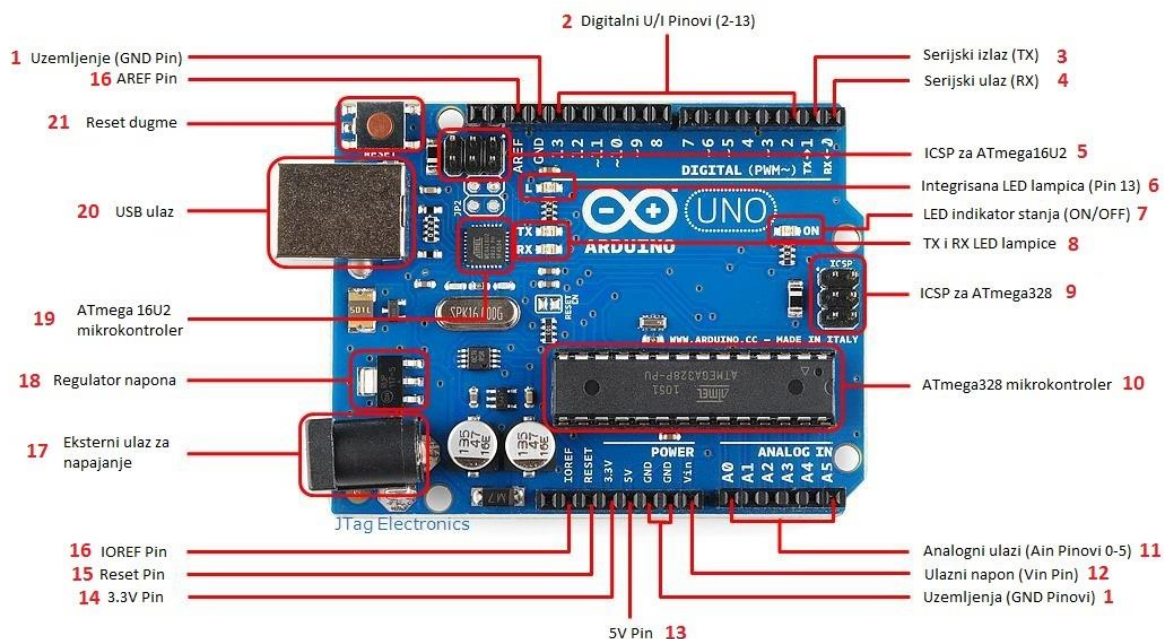
2.1.3. Arduino Uno [11]

2.1.3.1. Opis

Uno je mikrokontroler baziran na ATmega328P čipu. Sadrži 14 digitalnih ulaz/izlaz pinova (od kojih se 6 mogu koristiti kao PWM izlazi), 6 analognih ulaza, kvarcni kristal od 16 MHz, USB port, napojni ulaz, ICSP (*In-Circuit Serial Programming*) pinove i reset dugme. Sadrži sve što je neophodno za podršku mikrokontrolera te je samo nužno priključiti ga na kompjuter preko USB kabela ili ga uključiti u struju preko AC-DC adaptera ili baterije kako bi radio. Moguće je testirati mogućnosti Uno mikrokontrolera bez mnogo brige o posljedicama. Najgori mogući slučaj je da otkáže čip cijene od nekoliko dolara i da ga se treba promijeniti nakon čega je uređaj ponovo upotrebljiv.

“Uno” na italijanskom znači “jedan” te je odabrano za obilježavanje izlaska Arduino Softvera (IDE) 1.0. Uno ploča i verzija 1.0 Arduino Softvera su bili referentna verzija Arduino platforme koja je evoluirala u novija izdanja. Uno ploča je prva u serijalu USB Arduino ploča i referentni model za Arduino platformu.

2.1.3.2. Shema



Slika 2-7 Arduino Uno shema

Legenda:

- 1) Zajedničko uzemljenje
- 2) Digitalni ulazni/izlazni pinovi korišteni za generalnu upotrebu uz pomoć funkcija poput `pinMode()`, `digitalRead()` and `digitalWrite()`. Svaki pin ima ugrađen tzv. pull-up otpornik. Pinovi 3,5,6,9,10,11 se mogu koristiti kao PWM izlazi (detaljnije u narednom poglavlju)
- 3) TX serijski izlaz korišten u serijskoj komunikaciji za slanje podataka (UART protokol)
- 4) RX serijski ulaz korišten u serijskoj komunikaciji za primanje podataka (UART protokol)
- 5) ICSP pinovi za programiranje ATmega16U2 mikrokontrolera (promjenu firmware-a)
- 6) LED lampica spojena na pin 13 koju možemo koristiti za razna testiranja bez da spajamo eksternu lampicu
- 7) LED lampica koja pokazuje da li je Arduino upaljen ili ne
- 8) RX i TX lampice koje svjetlucaju u trenucima kad se šalju ili primaju podaci putem te komunikacije
- 9) ICSP pinovi za programiranje ATmega328 mikrokontrolera
- 10) ATmega328 mikrokontroler na kojem se u biti izvršava sav kod. Mozak Arduino pločice
- 11) Analogni ulazi korišteni za razna očitavanja sa senzora
- 12) Ulazni napon pomoću kojeg se može napajati pločica. Poželjno između 7 i 12V
- 13) 5V izlaz pomoću kojeg se napajaju vanjski uređaji tipa senzora i aktuatora
- 14) 3.3V izlaz pomoću kojeg se napajaju vanjski uređaji tipa senzora i aktuatora
- 15) Pin za softversko resetovanje pločice (obično korišten za dodavanje reset dugmeta kad se koriste *shield*-ovi)

- 16) IOREF - pin za podešavanje referentne voltaže sa kojom radi mikrokontroler i AREF - pin za podešavanje referentne voltaže za analogne ulaze (pomoću analogReference() metode)
- 17) Eksterni ulaz za napajanje. Može se koristiti AC-DC adapter ili adekvatan adapter za bateriju
- 18) Regulator napona koji dovedeni napon regulira i šalje na pinove
- 19) ATmega16U2 mikrokontroler koji se koristi isključivo kao most između kompjutervog USB porta i serijskog porta glavnog mikrokontrolera. U biti, pretvara podatke u pogodan oblik za komunikaciju putem USB-a
- 20) USB ulaz za programiranje pločice (ATmega328 mikrokontrolera)
- 21) Reset dugme za ponovno pokretanje programa spašenog na pločicu

Da ne bih zatrpavao legendu dodatnim tekstom značenja akronima su data na vrhu rada.

2.1.3.3. Tehnički detalji

Tabela 2-2 Arduino Uno tehnički detalji

Mikrokontroler	ATmega328P
Radni napon	5V
Ulazni napon (preporučeni)	7-12V
Ulazni napon (granični)	6-20V
Digitalnih U/I pinova	14 (od kojih 6 imaju PWM izlaz)
PWM digitalnih U/I pinova	6
Analognih ulaznih pinova	6
Količina istosmjerne struje po U/I pinu	20 mA
Količina istosmjerne struje za 3.3V pin	50 mA
Flash memorija	32 KB (ATmega328P), od koje je 0.5 KB rezervisano za <i>bootloader</i>
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Frekvencija sata	16 MHz
Dužina	68.6 mm
Širina	53.4 mm
Težina	25 g

2.1.3.4. Programiranje

Uno se programira koristeći Arduino IDE, odabirući „Arduino/Genuino Uno“ u Tools > Board meniju. ATmega328 čip dolazi preprogramiran sa tzv. *bootloader*-om koji omogućava prebacivanje koda sa računara na mikrokontroler bez korištenja eksternog hardverskog programera. Komunikacija se odvija putem STK500 protokola. *Bootloader* se može zaobići i mikrokontroler se može isprogramirati korištenjem ICSP-a, ako ima potrebe za nekim posebnim *firmware*-om³. Za te svrhe se može koristiti

³ Firmware – permanentni softver programiran unutar memorije namijenjene samo za čitanje; u slučaju ATmega328 i ATmega16U2 čipova moguće ga je reprogramirati jer se koristi EEPROM

Atmel-ov FLIP softver na Windows platformi ili DFU programmer na Mac OS X i Linux platformi. Arduino IDE i drugi alati za *cross development* će biti obrađeni u zasebnom poglavlju

2.1.3.5. Napajanje

Uno može biti napajan na 2 načina

- USB konekcija
- Vanjski izvor napajanja

Izvor napajanja se odabire automatski od strane pločice.

Vanjski izvor napajanja može biti AC-DC adapter ili baterija. Bateriju možemo spojiti putem Vin i GND ulaza ili kao i AC-DC adapter putem 2.1mm muškog konektora u ulaz prikazan na slici: Slika 2-7 Arduino Uno shema, dio pod brojem 17. Za stabilan rad potrebno mu je dovesti između 6 i 20 volti eksternog napona. U slučaju da ga napajamo sa manje od 7V, zbog interne strukture, moguće da izlaz od 5V neće moći obezbijediti punih 5V i samim tim pločica može postati nestabilna. U slučaju korištenja više od 12V, regulator napona koji pretvara tu voltažu u izlaze od 5 i 3.3V na pločici bi se mogao zagrijati i oštetiti pločicu. Tako da je u biti preporučena voltaža između 7 i 12V.

2.1.3.6. Komunikacija

Uno ima nekoliko načina za komunikaciju sa kompjuterom, drugom Uno pločicom i generalno drugim mikrokontrolerima. ATmega328 omogućava komunikaciju putem UART TTL (5V) serijske komunikacije koja se koristi na pinovima 0 (RX) i 1 (TX). ATmega16U2 kanalira ovu serijsku komunikaciju preko USB-a i prikazuje se na kompjuteru kao virtuelni port. 16U2 *firmware* koristi standard USB COM i nisu potrebni nikakvi eksterni drajveri. Na pinovima 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) ATmega328 nam omogućava SPI komunikaciju koja još uvijek nije uvrštena u Arduino jezik, ali I²C komunikacija koja se nalazi na pinovima 4 (SDA) i 5 (SCL) je podržana „Wire“ bibliotekom u Arduino jeziku.

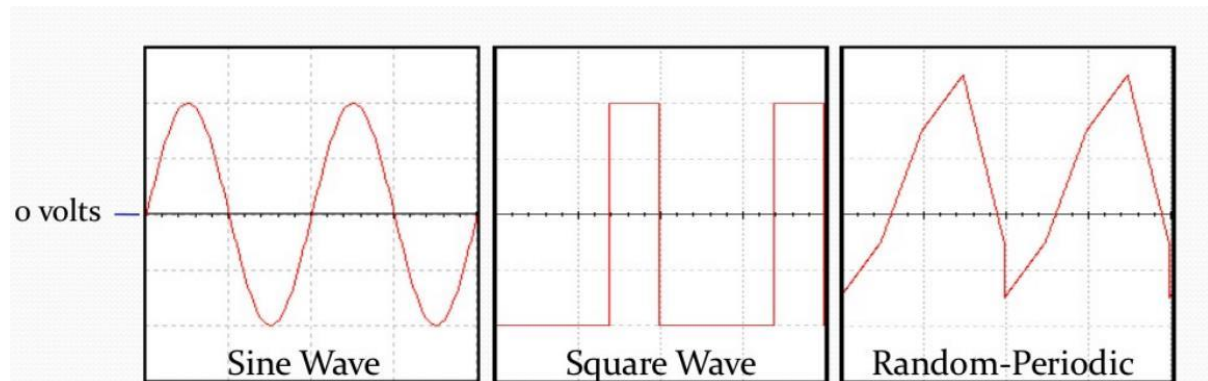
2.1.4. Digitalni ulazi/izlazi i analogni ulazi

U prethodnom poglavlju sam spomenuo digitalne i analogne pinove (portove, ulaze/izlaze) i s obzirom na činjenicu da bi Arduino bez korištenja ovih mogućnosti bio gluh, slijep i nijem, smatram da zaslužuju posebno poglavlje. U ovom poglavlju će biti objašnjene osnovne ideje analogne i digitalne tehnologije, kao i njihova primjena u Arduino svijetu.

2.1.4.1. Analogni signal i analogni ulaz [12]

Analogni signal je vremenski kontinualan signal, što znači da se amplituda signala mijenja kontinuirano u vremenu i ne prekida. Generalno gledajući, postoji beskonačno vrijednosti unutar nekog raspona, samo je pitanje rezolucije sistema koji koristi te analogne vrijednosti. Kontinualne veličine su temperatura, nivo osvjetljenja, zvuk i slično. Postoji više vrsta i tipova analognih signala. Oni mogu biti

periodični ili neperiodični, te maksimalne vrijednosti mogu biti pozitivne ili negativne. Učestali oblici su sinusni i kvadratni oblik funkcije. Na sljedećoj slici vidimo tri tipa analognih signala:



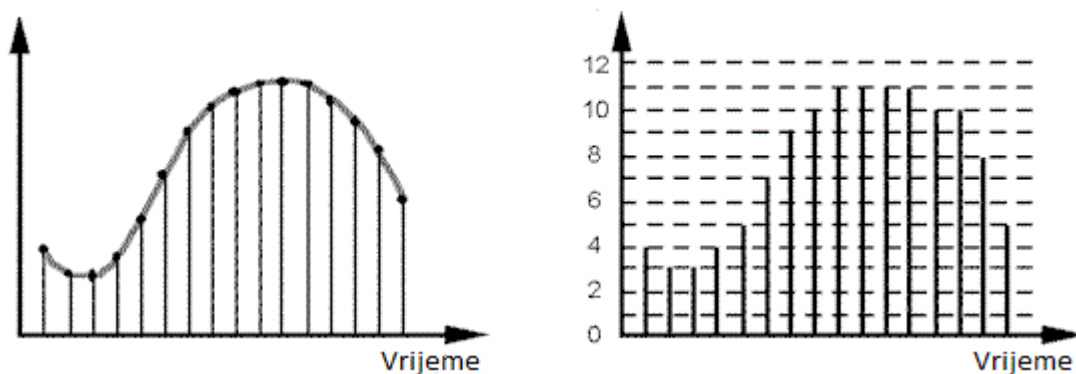
Slika 2-8 Primjeri analognog signala [13]

Kvadratni signal možda izgleda kao digitalni, ali s obzirom da ima i negativnu amplitudu, jasno je da je riječ o analognom signalu. Česta upotreba ovih tipova signala je u video i audio industriji. Vrijednosti su dosta egzaktnije, ali je teže raditi s njima u odnosu na digitalne vrijednosti. S obzirom da se za slanje analognih signala koriste provodnici u kojima zbog različitih faktora (tipa elektromagnetnog polja) može doći do smetnji, distorzija signala nije tako rijetka pojava.

U elektronici analogni signal je kontinualno promjenjivi napon ili struja. Ako uzmemo u obzir konkretno Arduino Uno platformu, ona ima 6 analognih ulaza na koje možemo priključiti senzore koji proizvode analogni signal. Primjer jednog takvog senzora je temperaturni LM35 senzor kojem ću se posvetiti u narednom poglavlju. Kako to sve radi? Senzor koji radi sa analognim vrijednostima minimalno ima 3 pina. Jedan je namijenjen za izvor napajanja, drugi uzemljenje i treći predstavlja analogni izlaz. Struja teče kroz senzor, u ovisnosti od njegove namjene stvara se pad napona koji se preslikava na pin koji je uključen u analogni ulaz mikrokontrolera. Taj pad napona daje neku vrijednost od 0 do 5V (za druge Arduino mikrokontrolere vrijede drugi radni naponi). Senzori su konstruisani tako da ne daju veći izlaz napona nego što im je dovedeno na ulaz. Pomoću funkcije **analogRead()** u sklopu Arduino IDE-a očitavamo tu vrijednost, ali dobivamo vrijednost između 0 i 1023. Šta se tu desilo- Arduino na svakom od svojih analognih ulaza ima 10 bitni analogno-digitalni konverter koji mapira raspon od 0 do 5V u cjelobrojnu vrijednost od 0 do 1023. Ako podijelimo 5V sa 1024 moguće vrijednosti dobijemo rezoluciju od 0.0049V (4.9 mV) po jedinici što je vrlo dobra rezolucija. Arduino treba oko 100 mikrosekundi da pročita vrijednost sa analognog ulaza, što daje oko 10 000 očitavanja u sekundi.

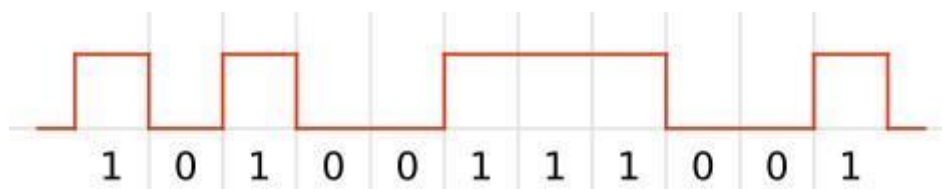
2.1.4.2. Digitalni signal i digitalni ulazi/izlazi

Digitalni signal je signal koji predstavlja sekvencu diskretnih vrijednosti. Za razliku od prethodno objašnjenih analognih veličina koje mogu uzimati vrijednosti međusobno različite za proizvoljno male iznose, kod digitalnih veličina vrijednosti se mogu uzimati u tačno određenim trenucima koji su jasno razdvojeni. Na sljedećoj slici vidimo razliku između kontinualnih (analognih) na lijevoj strani i diskretnih vrijednosti na desnoj strani: [14]



Slika 2-9 Analogne i diskretne vrijednosti

U računarstvu se koristi tzv. logički signal koji je u biti digitalni signal sa samo dvije moguće vrijednosti. U našem slučaju na digitalni signal možemo gledati kao na sekvencu bita koji predstavljaju fizičku veličinu. Za fizičku veličinu se uzima struja ili napon električnog signala, intenzitet svjetla optičkog signala, jačina radio signala itd. Digitalni signal u sklopu elektronike se pretežno koristi za prenošenje podataka i najčešća fizička veličina koju predstavlja je napon. Za komunikaciju između uređaja i unutar njih samih se preferira digitalno kodiranje jer manje prostora zauzimaju jedinice i nule nego cijeli spektar vrijednosti koje može poprimiti neka analogna varijabla. Ako za primjer uzmemo *bluetooth* komunikaciju između dva uređaja, slanje podataka ide isključivo prenosom niza jedinica i nula, gdje se tačno zna (u skladu sa određenim protokolima) šta su podaci, a šta oznake početka i kraja poruke. Primjer jednog takvog niza je dat na sljedećoj slici: [15]



Slika 2-10 Poruka kodirana jedinicama i nulama [15]

Osnovna vrsta ulaza i izlaza koju koriste mikrokontroleri su digitalni ulazi i izlazi. Pomoću njih se očitavaju i postavljaju digitalne vrijednosti, odnosno vrijednosti napona koje odgovaraju logičkoj jedinici ili logičkoj nuli. Digitalni ulaz omogućava očitavanje vrijednosti napona i softversku interpretaciju ove vrijednosti kao logičke nule ili logičke jedinice. To znači da jedan takav ulaz očitava vrijednost napona koji odgovara jednom bitu. Opsezi vrijednosti napona koji će biti ispravno protumačeni ovise o korištenoj tehnologiji:

- Za TTL vrijedi:
 - $0V - 0.8V$ se tumači kao logička **0**
 - $2V - 5V$ se tumači kao logička **1**
- Za CMOS vrijedi:
 - $0V - \frac{1}{3}V_{dd}$ se tumači kao logička **0** (V_{dd} predstavlja voltažu izvora napajanja)

- $\frac{2}{3}V_{dd} - V_{dd}$ se tumači kao logička 1

Digitalni izlaz omogućava da se logička vrijednost u okviru aplikacije koja se izvršava na mikrokontroleru iskoristi za postavljanje vrijednosti napona na nekom od mogućih izlaza mikrokontrolera. Također, i u ovom slučaju digitalni izlaz interpretira jedan bit u formu vrijednosti napona na pinu.

Glavna razlika u internoj strukturi ulaza i izlaza se ogleda u otporu koji se nalazi na ulazu odnosno izlazu. U slučaju digitalnog ulaza otpor je reda $M\Omega$ što u biti ne dozvoljava oštećenje pločice ako je spojimo direktno na veći pozitivni napon ili na 0V. Iz sljedeće jednačine vidimo da će u slučaju spajanja na 5 voltni izvor proteći vrlo mala struja:

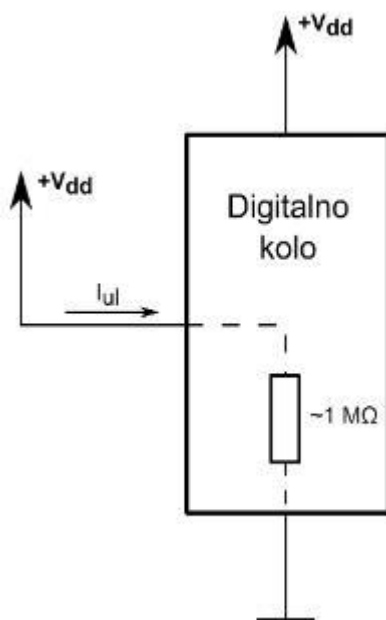
$$I_{ul} = \frac{5V}{1M\Omega} = 5\mu A$$

koja proizvodi snagu:

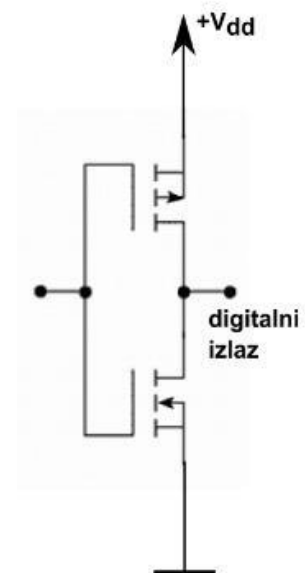
$$P = U * I_{ul} = 25\mu W$$

Ova snaga je dovoljno mala da ne može izazvati bilo kakvo oštećenje poluprovodnika u kolu digitalnog ulaza. [16]

Kako je bitno da dovođenjem 5V ili 0V na ulaz, pločica ostane neoštećena, isto tako je bitno da prilikom postavljanja logičke nule ili jedinice na izlazne pinove, vrijednost napona ostane što približnija idealnih 5V ili 0V, što postizemo malim izlaznim otporom (reda Ω). S obzirom da je generalna ideja smanjiti komponente od kojih se sastoji mikroprocesor, izlazni tranzistori su vrlo mali i ne trpe velike vrijednosti struja. Obično se „podnošljive“ struje kreću u opsegu od 5 mA do nekih 25 mA u ovisnosti od procesora. Naredne slike prikazuju izled digitalnog ulaza i izlaza.



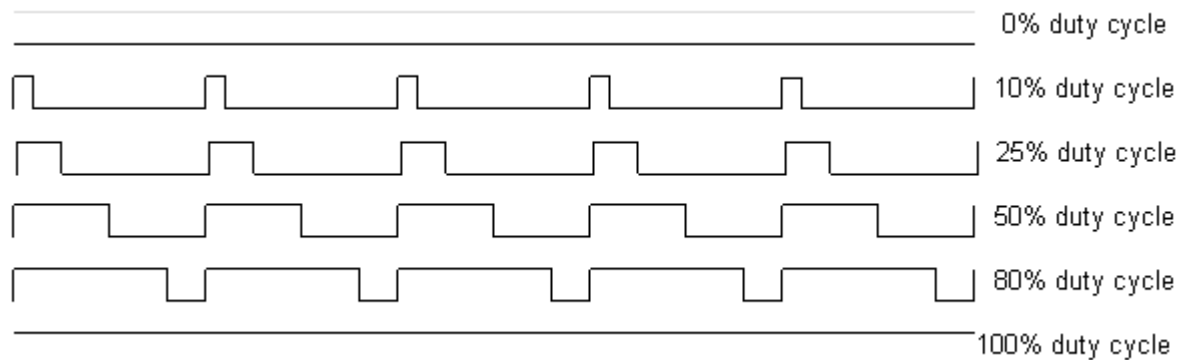
Slika 2-12 Digitalni ulaz koji možemo direktno povezati sa napajanjem [16]



Slika 2-11 Digitalni izlaz sa par tranzistora [16]

Još jedna interesantna stvar koja se tiče digitalnog izlaza je tzv. PWM, što predstavlja simulaciju analognog izlaza putem digitalnog izlaza. Ideja iza toga je da omjerom dužine trajanja jedinice kao

logičke reprezentacije 5V i nule kao logičke reprezentacije 0V mi u biti prividno postizemo neku međuvrijednost. Npr. ako je u nekom vremenskom intervalu 50% vremena aktivna jedinica i 50% nula, s obzirom na 16 megahercni interni sat mikrokontrolera i mogućnost dovoljno brzog uzorkovanja mi prividno imamo 2.5V na izlazu. Taj omjer dužine „ON“ stanja i „OFF“ stanja se naziva *duty-cycle*. Na sljedećoj slici vidimo primjer različitih *duty-cycle*-ova:



Slika 2-13 PWM *duty-cycle* [17]

S obzirom da Arduino Uno nema analogni izlaz, ovo rješenje je vrlo prihvatljivo kao njegova simulacija. Recimo ako želimo postići da LED lampica svijetli na 25%, umjesto regularnih 100% koje dobijemo kada na pin postavimo u stanje “HIGH”, odnosno dovedemo 5V, mi možemo koristiti PWM i sa *duty-cycle*-om od 25% postizemo taj efekat. Izmjene su toliko brze da ih naše oko ne može registrovati, nego imamo osjećaj da lampica svijetli na 25%.

Pošto sad znamo da Arduino posjeduje mogućnosti za očitavanje i postavljanje digitalnih vrijednosti na pinove, na koji način se koriste ta njegova “osjetila”? Sve se postiže kroz kod koji kucamo u IDE-u i šaljemo na mikrokontroler. Postoji par pogodnih funkcija:

- **pinMode(pin, mode)** – koristi se za postavljanje tipa pina
 - pin – broj pina koji želimo postaviti
 - mode – INPUT, OUTPUT ili INPUT_PULLUP (aktivan u 0)
- **digitalRead(pin)** – koristi se za očitavanje vrijednosti sa pina koja može biti HIGH ili LOW
 - pin – broj pina koji želimo očitati
- **digitalWrite(pin, value)** – koristi se za postavljanje vrijednosti pina proslijeđenog kao parametar na vrijednost value
 - pin – broj pina koji želimo postaviti na vrijednost value
 - value – HIGH ili LOW (5V ili 0V)
- **analogWrite(pin, value)** – koristi se za postavljanje vrijednosti pina proslijeđenog kao parametar koristeći *duty-cycle* proslijeđen kao value
 - pin – broj pina koji želimo postaviti
 - value – *duty-cycle* između 0 (uvijek ugašeno) i 255 (uvijek upaljeno)

2.1.5. Senzori

U prethodnom poglavlju sam spomenuo da bi Arduino bio gluh, slijep i nijem kako bez analognih ulaza tako i bez digitalnih ulaza/izlaza. To je u potpunosti tačno, ali da bi iskoristili mogućnosti koje ima, potrebno je spojiti nešto u ulaze/izlaze. U ovom poglavlju ću navesti primjere dva tipa senzora koji se razlikuju po signalima koje projiciraju na svoje izlazne pinove, a što se tiče aktuatora u smislu lampica, displeja ili motora, oni neće biti posebno obrađeni u ovom radu jer izlaze iz opsega teme. U sklopu rada se koriste HC – 05 *Bluetooth* modul i HC-SR04 ultrazvučni senzor udaljenosti koji koriste po jedan pin postavljen kao OUTPUT, ali to direktno ne znači da su oni aktuatori.

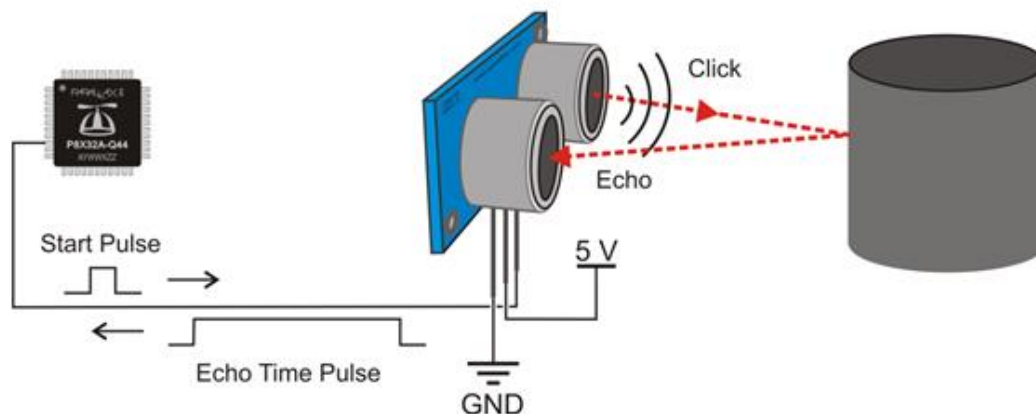
2.1.5.1. Senzori koji koriste digitalne signale

Postoji jako veliki broj senzora koji nakon procesiranja i obavljanja posla za koji su napravljeni šalju digitalne signale. Primjeri takvih senzora su senzor vatre, DHT11 digitalni senzor temperature i vlažnosti zraka, RTC (*Real Time Clock*) ili sistemski sat, digitalni senzor dodira i još mnogo drugih. Reprezentativan primjerak skupine senzora koji koriste digitalni signal je ultrazvučni senzor udaljenosti, model HC-SR04.



Slika 2-14 HC-SR04 ultrazvučni senzor udaljenosti [18]

Na koji način on radi - Jedan od „zvučnika“ služi kao odašiljač, dok drugi predstavlja prijemnik. Na pin Vcc se dovede napon od 5V. Na GND pin se dovede uzemljenje. Srednja dva pina služe za aktivaciju „zvučnika“. Trig pin je OUTPUT pin i predstavlja okidač (skraćeno od *trigger*), odnosno pin čije postavljanje na HIGH uzrokuje pomjeranje membrane i iniciranje ultrazvučnih valova. Potrebno je u kratkom vremenskom periodu (reda 2 mikrosekunde) postaviti trig na HIGH, sačekati, pa ga postaviti na LOW. Nakon toga Echo pin postaviti u INPUT mod te oslušivati kad će se vratiti poslani ultrazvučni val. Na ovaj način se dobije vremenska razlika između slanja i primanja signala, tako da ako želimo prikazati centimetre ili neku drugu mjernu jedinicu na ekranu, potrebno je konvertovati dobivenu vrijednost. Naredna slika prikazuje situaciju lociranja udaljenosti predmeta:



Slika 2-15 Lociranje objekta pomoću ultrazvučnog senzora [19]

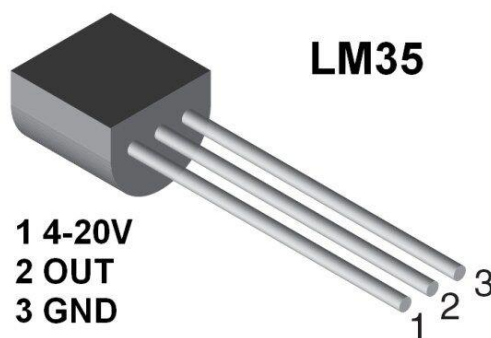
Na Internetu postoje biblioteke koje nam olakšavaju mjerenje udaljenosti na način da su već implementirane funkcije koje aktiviraju slanje i primanje signala. Na nama je samo da ih pozovemo i njihov rezultat dodijelimo nekoj varijabli. Jedna takva biblioteka je **NewPing**. Više o njoj u praktičnom dijelu. Iz ovog primjera vidimo kako digitalni signali mogu inicirati određene aktivnosti i kako pomoću njih možemo očitavati vrijednosti.

2.1.5.2. Senzori koji koriste analogne signale

Prethodno sam objasnio šta predstavlja analogni signal i u kojim situacijama je koristan. S obzirom da se u sklopu praktičnog dijela rada ne koriste analogni signali i analogni senzori, u ovom poglavlju ću samo navest neke senzore i prikazati na koji način radi LM35 temperaturni senzor.

Kako sam spomenuo da se analogni signali dosta koriste u audio i video industriji, reprezentativni primjerci takvih senzora su senzor zvuka, svjetlosti, magnetnog polja i slično. Vidimo da se u biti mjere sve one konitinalne veličine koje sam naveo kao primjere u prethodnom poglavlju.

LM35 temperaturni senzor ima 3 pina. Jedan za napajanje, drugi za uzemljenje i treći pin služi za slanje analognih vrijednosti na mikrokontroler. Na sljedećoj slici vidimo kako izgleda taj senzor:



Slika 2-16 LM35 temperaturni senzor [20]

Vidimo da je mogući raspon ulaznog napona između 4 i 20 volti i samim tim izlazni napon o tome ovisi. Ako nam je poznato da je rezolucija senzora koju daje na izlazu 10mV/°C, da bi dobili temperaturu u °C potrebno je podijeliti očitane vrijednosti sa 1023 (zbog AD konverzije i mapiranja vrijednosti u tom intervalu) i rezultat pomnožiti sa umnoškom ulaznog napona i broja 100 (100 jer je rezolucija 10mV). Ta formula bi izgledala ovako:

$$temperatura^{\circ C} = \left(\frac{vrijednostSenzora}{1023} \right) * ulazniNapon * 100$$

Slična logika bi bila i za druge senzore. Bitno je znati u koje vrijednosti se mapiraju očitavanja i šta iz toga želimo dobiti.

2.1.6. Cross development za Arduino i Arduino IDE [21]

Cross development kao pojam predstavlja razvoj i kreiranje egzekutabilnog koda za platformu različitu od one na kojoj se pravi taj kod. U našem slučaju *cross development* je neizbježan iz razloga što ATmega328 čip nema dovoljno veliku procesorsku moć da brzo i efikasno kompajlira i izvršava kod pisan za njega. Drugi razlog nemogućnosti kompajliranja je što Arduino, za razliku recimo od Raspberry Pi mikrokontrolera nema operativni sistem, što opet proizilazi iz manjka procesorske moći i memorijskih resursa. Tu se uviđa potreba za korištenjem različitih softverskih paketa i IDE okruženja pomoću kojih pišemo kod, te ga kompajliramo posebno za svaku platformu na koju se prebacuje. U biti koristimo mogućnosti današnjih desktop i mobilnih računara kako bi napravili egzekutabilne verzije koda za više različitih platformi koje nisu u mogućnosti same sve obaviti. Ako bolje razmislimo, primarni zadatak Arduino uređaja je da „osjeti“ svijet oko sebe i to prezentira na neki način, a ne da obavlja neke velike, procesorske i memorijske zahtjevne operacije. U ovoj situaciji vidimo čari IDE-ova i svega što je nastalo iz *Wiring platforme* i *Processing* jezika.

Najpopularnije razvojno okruženje za pisanje i kompajliranje koda za Arduino je Arduino IDE napravljen od strane istoimene kompanije. Pored njega postoji još 15-ak popularnih razvojnih alata poput:

- **PlatformIO IDE** - dio Atom razvojnog okruženja (Windows, Mac, Linux)
- **Programino IDE**
- **Arduino for Visual Studio** - *plugin* za Visual Studio koji za razliku od prethodna dva alata ima ugrađen *debugger* sa kojim čak postoji mogućnost inspekcije vrijednosti upisanih u registre (Windows)
- **Arduino Eclipse IDE** – *plugin* za Eclipse razvojno okruženje (Windows, Mac, Linux)
- **emedXcode** – Arduino na Xcode-u (Mac)
- **Arduino za Atmel Studio** – također razvojno okruženje pomoću kojeg se može debugirati
- **Zerynth Studio** – Koji dolazi u paketu sa istoimenom aplikacijom za mobilne uređaje koju koristimo kao interfejs za projekte kreirane u Zerynth studiu.
- **Atmel Studio**
- **biicode** - (Windows, Linux, Mac)
- **Pluto**- IDE za programiranje Arduino platforme u Python jeziku

Postoji još par alata, ali ovo su najčešće korišteni alati sa najpotpunijim interfejsom za rad. Naredna tabela daje detaljniji uvid u mogućnosti 2 najpopularnija okruženja, pored samog Arduino IDE-a:

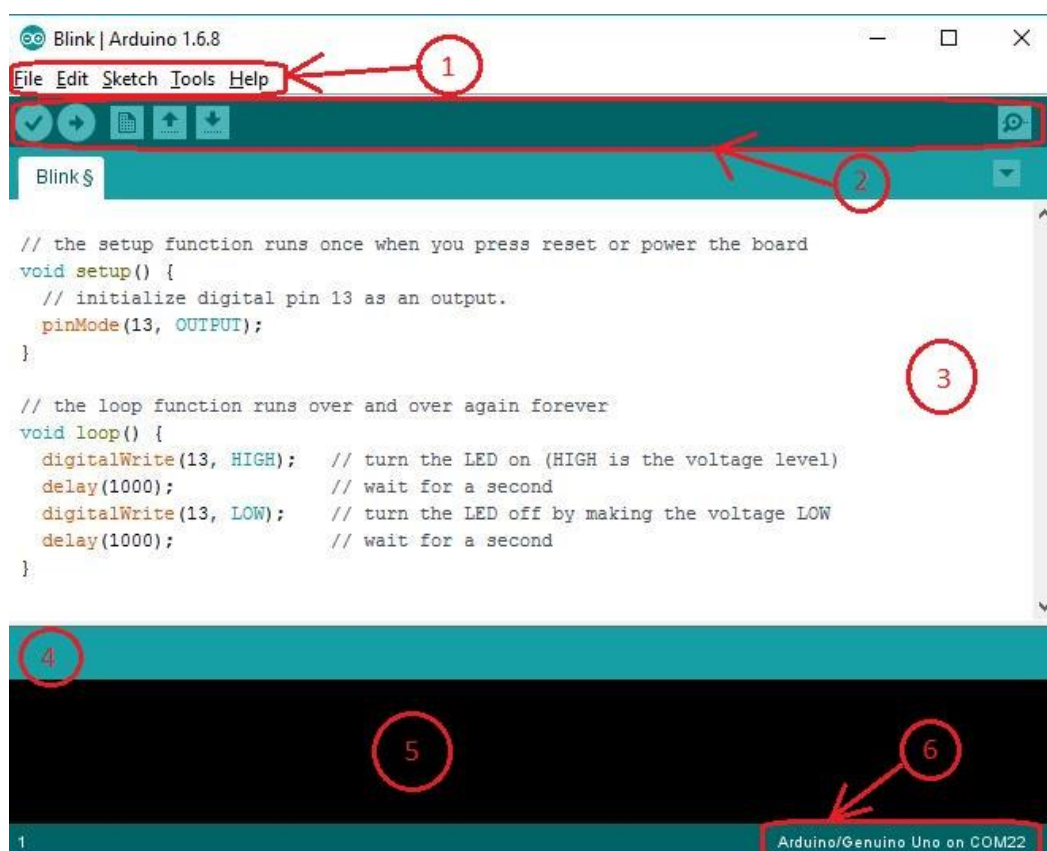
Tabela 2-3 Usporedba Programino i Visual Micro razvojnih okruženja

Programino IDE	Arduino za Visual Studio (Visual Micro)
Lagan za korištenje	U potpunosti kompatibilan sa projektima iz Arduino IDE-a
Kompatibilan sa svim verzijama Ardina	Maksimalno podesiv korisnički interfejs
Jezici: Arduino, C, C++, Header, HTML	<i>Intellisense</i> - sam predlaže, sam dopunjava imena funkcija i klasa kako kucate
Podržava sve Arduino Biblioteke	Provjera sintakse – pronalazi, označava i objašnjava sintaksne greške bez prethodne kompilacije
Pretraživač funkcija i objekata	Podržava širok spektar pločica
Code autocompletion	Unikatan <i>debugger</i> – (<i>breakpoints, tracepoints, watch and change variables</i>)
Hintovi i informacije o Arduino komandama	Moćan alat za upravljanje bibliotekama (<i>download</i> i pretraga online biblioteka)
Hardware viewer	Upravljanje pločicom kroz IDE
2 serijska terminala	Rad na više pločica simultano
Dizajner za 8x8 matrični displej	Podrška za Git i TFS alate za verzioniranje
RGB-LED izbornik boja	
Konverter vrijednosti (DEC,HEX,BIN,ASCII)	
Korisničke pločice (možete dodati svoju pločicu)	
Podešavanje naglašavanja sintaksnih dijelova (<i>syntax highlighting</i>)	
Automatsko uvlačenje koda	

Za izradu praktičnog dijela rada je korišten Arduino IDE, tako da će u narednom dijelu biti govora o mogućnostima i načinu korištenja tog okruženja.

U poglavlju 2.1.1 sam spomenuo da je Banzi htio napraviti platformu koja je spremna za korištenje odmah nakon kupovine bez potrebe za dodatnim ulaganjima i doradama. Na engleskom jeziku postoji fraza koja kaže: „Ready out of the box“, što bi u bukvalnom prijevodu značilo „Spremno iz kutije“. Da bi u potpunosti ispunio taj cilj, bilo je potrebno predefinisati jedan osnovni projekat koji se može pokrenuti čim izvadimo Arduino iz kutije i tako je kreiran „Blink“, projekat za koji ne trebaju nikvi dodatni senzori, aktuatori, žice i slično. Arduino dolazi spreman sa jednom LED lampicom integrisanom na pločicu i nakon instalacije i pokretanja IDE softvera prvi projekat koji se otvara nije prazan, nego baš navedeni „Blink“. Kreatori Arduino platforme su o svemu razmišljali, tako da je ta lampica spojena na pin 13 i ako ništa drugo ne spojimo na taj pin, njegovom aktivacijom ćemo upaliti lampicu. To je upravo ono što radi „Blink“ aplikacija; definiše pin 13 kao izlazni pin i u ponavljajućoj beskonačnoj petlji šalje signal HIGH na pin, čeka jednu sekundu, nakon čega šalje LOW signal i opet čeka jednu sekundu. Mislim da je sad u potpunosti jasno preklapanje naziva projekta i onoga što se dobiva kao rezultat koda.

Programi koji su napisani uz pomoć Arduino IDE okruženja se nazivaju **sketches (skice)**. Ove skice se pišu u tekst editoru i spašavaju se pod ekstenzijom **.ino**. Editor ima standardne mogućnosti za obradu teksta poput *cut/paste* i *search/replace*. Unutar okruženja, pored editora postoji i prostor za poruke koji obavještava korisnika o statusu spašavanja, kompajliranja i eksportovanja koda na pločicu, kao i o greškama. Ispod tog prostora nalazi se konzola koja pruža informacije o zauzetoj memoriji kad se kod prebaci na pločicu, preostaloj memoriji i još par nekih informacija. U donjem desnom uglu se nalazi labela koja prikazuje na kojem serijskom COM portu se nalazi naša pločica. Da bi cijeli postupak bio još jasniji neophodno je proći kroz razvojno okruženje prikazano na narednoj slici:



Slika 2-17 Arduino IDE 1.6.8

Legenda:

1. Generalni meni sa funkcionalnostima koje ovise o kontekstu u kojem se otvore.
2. Traka za verifikovanje, *upload* na pločicu, kreiranje nove skice, otvaranje postojećih projekata, spašavanje trenutne skice i otvaranje *Serial monitor-a*
3. Editor
4. Traka za poruke za upload, spašavanje i greške
5. Konzola
6. Informacije o portu







Dio pod brojem 1:

- **File**
 - **New** – Kreira novu instancu editora sa funkcijama **setup()** (dio koda koji se izvrši samo jednom) i **loop()** (beskonačna petlja u kojoj se izvršava kod)
 - **Open** – Omogućava učitavanje skica iz foldera na kompjuteru
 - **Open Recent** – Otvara kratku listu nedavno uređivanih skica spremnih za korištenje
 - **Sketchbook** – Prikazuje trenutnu skicu unutar menija koji predstavlja mjesto na računaru (folder) gdje se nalaze skice i klikom na bilo koje ime otvaramo tu skicu u novom editoru
 - **Examples** – Razni primjeri koji su predefinisani i mogu se koristiti kao takvi; Obično se sa instalacijom nove biblioteke dobiva i popratni primjer korištenja iste koji je smješten u ovoj listi
 - **Close** – zatvara instancu Arduino softvera u kojoj je kliknuta opcija
 - **Save** – Spašava skicu sa trenutnim imenom; ako prethodno nije imenovana, potrebno je ukucati naziv u „Save As..“ prozoru
 - **Save as..** – Omogućava spašavanje trenutne skice sa drugim imenom
 - **Page Setup** – Prikazuje podešavanje stranice za printanje
 - **Print** – Šalje trenutnu skicu na printer prema podešavanjima iz Page Setup menija
 - **Preferences** – Otvara podešavanja okruženja
 - **Quit** – Zatvara sve instance Arduino IDE prozora
- **Edit**
 - **Undo/Redo**
 - **Cut**
 - **Copy**
 - **Copy as HTML** – kopira kod *clipboard* kao HTML i tako je pogodan za ugrađivanje u web stranice
 - **Paste**
 - **Select All**
 - **Comment/Uncomment** – Ako je selektovani kod odkomentarisan, stavlja oznake komentara na njega, u suprotnom uklanja iste oznake iz koda
 - **Increase/Decrease Indent** – Uvlači dodatno ili „izvlači“ kod
 - **Find**
 - **Find Next**

- **Find Previous**
- **Sketch**
 - **Verify/Compile** – Provjerava greške u kodu kompajlirajući ga i prezentuje rezultate o memoriji koju kod zauzima i varijablama u konzoli
 - **Upload** – Kompajlira i učitava binarni fajl na postavljenu pločicu kroz postavljenu port
 - **Upload Using Programmer** - Ovo prepisuje predefinisani *bootloader*. Uz pomoć Tools > Burn Bootloader vraćamo stari *bootloader*
 - **Export Compiled Binary** – Spašava .hex fajl koji se može koristiti za upload na pločicu ili od strane drugih alata
 - **Show Sketch Folder** – Otvara trenutni folder u kojem se nalazi skica
 - **Include Library** – Uključuje biblioteku u skicu dodavanjem #include iskaza na vrhu koda; putem ove opcije možemo pristupiti Library Manager-u koji služi za importovanje novih biblioteka iz .zip datoteka
 - **Add file** – Dodaje izvorni fajl u skicu
- **Tools**
 - **Auto Format** – Formatira kod tako da se poravnaju otvorene i zatvorene zagrade, uvlači kod unutar vitičastih zagrada i slično
 - **Archive Sketch** – Arhivira trenutnu skicu u .zip fajl
 - **Fix Encoding & Reload** – Popravlja moguće razlike između enkodiranja unutar koda
 - **Serial Monitor** - Otvara prozor *serial monitor-a* i inicira razmjenu podataka sa priključenom pločicom kroz odabrani port.
 - **Board** – Spisak pločica koje ovaj IDE podržava; tu odabiremo pločicu koju koristimo
 - **Port** – Meni koji sadrži sve serijske uređaje (fizičke ili virtuelne) koji su spojeni na kompjuter; služi da odaberemo port na kojem se nalazi Arduino
 - **Programmer** – Opcija koju koristimo kad ne želimo programirati pločicu na standardni način pomoću USB serijske konekcije
 - **Burn Bootloader** – Opcija za vraćanje standardnog *bootloader-a* na mikrokontroler
- **Help**
 - Generalna pomoć u vezi sa platformom i korisni linkovi

Dio pod brojem 2:

U biti se kroz prethodno objašnjene menije može pristupiti svim opcijama iz ovog dijela, samo što je ovaj način kraći. S obzirom da sam već objasnio šta koja akcija radi i postiže, ovdje ću samo povezati ikonice sa njihovim nazivima:

-  - **Verify/Compile**
-  - **Upload**
-  - **New**
-  - **Open**
-  - **Save**
-  - **Serial Monitor**

Dio pod brojem 3:

Ovaj dio predstavlja Editor. U Editoru pišemo kod koji će biti kompajliran i nakon toga prebačen na mikrokontroler. Arduino kod možemo podijeliti na neka 3 dijela.

- biblioteke
- setup()
- loop()

Biblioteke omogućavaju dodatne funkcionalnosti za korištenje unutar skica. One se nalaze na vrhu koda i uključuju sa **#include NazivBiblioteke**, ili pomoću menija **Sketch > Import Library**. Kroz isti meni imamo mogućnost importovati nove biblioteke iz .zip fajlova

Setup dio je u biti funkcija u koju pišemo kod za koji želimo da se samo jednom izvrši. Primjer bi mogao biti kreiranje serijske komunikacije za *Serial monitor* (`Serial.begin()`) ili deklarisanje pinova kao ulaza ili izlaza.

Loop dio predstavlja funkciju koja se izvršava u beskonačnoj petlji (što samo ime govori) i dok god ima napajanja na uređaju prolazi se kroz taj dio koda. Tu pišemo glavni kod.

Dio pod brojem 4:

U ovoj traci se prikazuju poruke obavještenja u vezi sa akcijama tipa **upload**, **compile** i slično.

Dio pod brojem 5:

Konzola koja služi za davanje detaljnijih informacija o greškama koje su se desile prilikom kompilacije koda, u kojoj liniji se desila greška, šta je pošlo po zlu prilikom prebacivanja koda na pločicu, koliko je memorije zauzeto na pločici i koliko je ostalo, te par nekih drugih informacija sa kojima se još nisam susreo.

Dio pod brojem 6:

Labela koja nam govori na kojem virtuelnom portu se nalazi naš Arduino uređaj i koji je to uređaj.

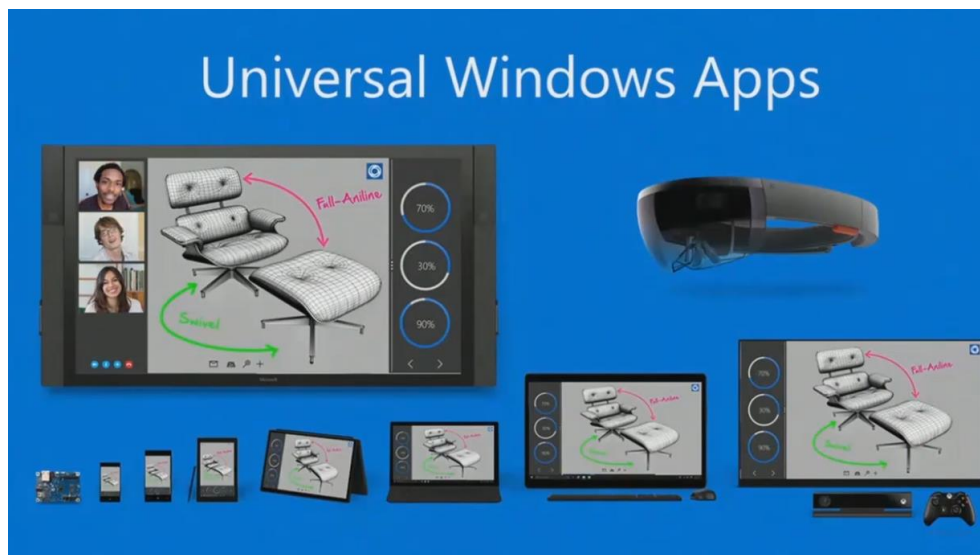
Na kraju ovog poglavlja bih jednom prošao kroz cijeli proces kreiranja koda za Arduino. Pokrenemo Arduino IDE koji otvara „Blink“ kod. Možemo obrisati taj kod i kad završimo sa pisanjem našeg koda sve spasiti pod novim imenom ili jednostavno kreirati novu instancu IDE-a. U novoj instanci imamo već napisane funkcije `setup()` i `loop()`. Na vrh dodamo željene biblioteke, u `setup()` dio napišemo kod koji želimo da se jednom izvrši i glavni kod u `loop()` funkciju. Nakon što smo završili sa pisanjem koda, poželjno je kompajlirati kod prije samog priključivanja Arduino uređaja putem USB-a. Ako je kompilacija uredno prošla, priključimo Arduino. U Tools meniju odaberemo pločicu za koju želimo da se kompajlira kod i na koju će kod biti prebačen, te odaberemo Port na kojem se nalazi naš Arduino. Nakon toga možemo kliknuti dugme koje aktivira kompilaciju i **upload** koda. Pratimo situaciju na traci za informacije i u konzoli. U slučaju da se desila neka greška, bićemo uredno obavješteni, dok u suprotnom dobijamo obavještenje o uspješnom prebacivanju. U trenucima dok se kod šalje na Arduino nije moguće koristiti *Serial monitor* jer je port koji Arduino koristi virtuelni port i već je zauzet za **upload**. Tek po okončanju te akcije možemo koristiti *monitor*. To bi otprilike bio proces razvoja koda za najpopularniju *DIY* platformu na svijetu. Po mom mišljenju, vrlo jednostavno.

2.2. Općenito o Universal App (UWP) platformi

U prethodnom dijelu se govorilo o teoriji koja stoji iza prikupljanja informacija iz vanjskog svijeta, a u ovom dijelu će biti govora o teoriji potrebnoj za shvatanje mehanizama sve popularnije Universal App platforme.

2.2.1. Universal Windows Application

Osnovna i najbitnija tehnološka prednost UWP aplikacija u odnosu na druge tipove aplikacija je izvršavanje na bilo kojem Windows uređaju. Ovo je relativno nova tehnologija koja je u kratkom vremenu postala vrlo popularna. Iako je postojala kao standardni dio biblioteka za operativne sisteme Windows 8 i 8.1, svoju popularnost doživjela je nastankom Windows 10 operativnog sistema. Windows 10 operativni sistem donio je jedinstveno Windows jezgro koje se nalazi na svim uređajima iz trenutne Windows familije, kako kod desktop i mobilnih uređaja, tako i kod Xbox-a, HoloLens-a, Surface Hub-a, Raspberry Pi 2 itd. [22]



Slika 2-18 Universal Windows Apps na različitim uređajima [23]

Na slici 2-18 vidimo kako se jedna aplikacija sa adaptivnim izgledom pokreće na različitim hardverima i veličinama ekrana. Sa desna na lijevo vidimo veliki LCD ekran na koji je priključena Xbox konzola, desktop računar, laptop, Surfacebook, *phablet*, tablet, mobitel i Raspberry Pi, dok je iznad HoloLense. Upravo univerzalnost ovih aplikacija, koja je prethodno spomenuta, je ono što izdvaja ovu platformu od ostalih. Zbog jedinstvenog jezgra operativnog sistema, kod za aplikacije se piše samo jednom, sa jednim setom poslovne logike i jednim grafičkim korisničkim interfejsom, nakon čega se pravi paket za aplikaciju koji se objavljuje na Windows Store. Aplikacija je na taj način pogodna za korištenje na svakoj platformi za koju je njen kreator želio da je učini dostupnom. To donosi veliki napredak i uštedu vremena kada je u pitanju razvoj *multiplatform* aplikacija. Ovome doprinosi mnoštvo mogućnosti koje Windows 10 platforma nudi, koje su namijenjene da inteligentno izvršavaju većinu adaptacije, čime je na programeru ostavljeno više vremena da se fokusira na krajnji produkt i na korisnika. [24]

Neke od karakteristika ovih aplikacija su:

- **Aplikacija se pravi za porodicu uređaja, ne operativni sistem**

Porodica uređaja predstavlja API-je, sistemske karakteristike i ponašanja koja su očekivana od uređaja unutar jedne porodice; Također određuje set uređaja na koje se aplikacija može instalirati sa Store-a.

- **Aplikacije su pakovane i distribuirane koristeći .AppX format pakovanja**

Sve UWP aplikacije su distribuirane kao AppX paket. Ovo omogućava pouzdan mehanizam instalacije i certifikacije aplikacije.

- **Samo je jedan Store za sve aplikacije**

Nakon što se registrujete kao developer možete objaviti aplikaciju na Store i odabrati za koje familije uređaja je pogodna.

- **Adaptivni korisnički doživljaj**

Fluidno prilagođavanje i *rendering* u trenutku izvršavanja u odnosu na korisnikovu interakciju sa uređajem.

- **Prirodni korisnički unosi (*natural user inputs*)**

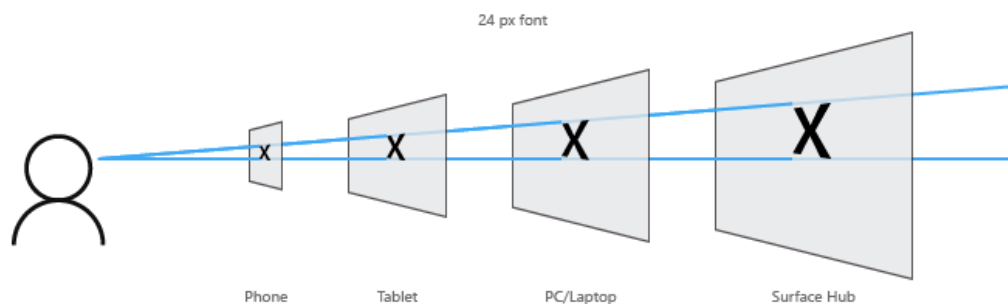
Omogućen je input na prirodan način, u vidu glasovnih naredbi, pokreta, pisanja po ekranu, različitih gestikulacija očima i sl., te je sve to vrlo lako implementirati u vlastita rješenja

- **Cloud bazirni servisi i API**

Dostupan je jako veliki broj servisa koji su postali dostupni developerima za upotrebu unutar aplikacija, a najpopularniji su Cortana AI, OneDrive i Application Insights. Tu su također Windows Notification Services, Windows Credential Locker i slično.

Brojne su pogodnosti uključene i za implementaciju ponašanja aplikacije kada je korisnik ne koristi, na primjer Cortana *integration* – mogućnost pokretanja aplikacija direktno kroz Cortana pretraživač. Cortana, prema Windows definiciji, predstavlja inteligentnog personalnog asistenta, što je u biti vid vještačke inteligencije. Ovaj korak u budućnost je olakšao pretraživanje računara, Interneta i mnogo drugih stvari. Dovoljno je samo da aktiviramo Cortana asistenta na Windows uređaju i prirodnim govorom/glasovnim naredbama, pitamo za savjete tipa šta obući danas i dobivamo odgovor kolika će temperatura biti, da li će padati kiša i šta bi mogli obući. UWP aplikacija nam omogućava da sami implementiramo komande/naredbe, kao i reakcije koje bi mogli dobiti na njih. Zašto ih onda izbjegavati?! Jedino ograničenje koje je postavljeno tiče se razvojnog okruženja u kojem se razvija ovaj tip aplikacija. Neophodno je imati Visual Studio 2015 kako bi mogli biti instalirani dodatni *plugin*-i potrebni za razvoj UWP aplikacija. Više o kreiranju i potrebnim *plugin*-ima u narednom poglavlju. [25]

Najjednostavnije bi bilo objasniti realizaciju ovog modela kroz par slika:



Slika 2-19 Algoritam skaliranja 24px fonta na različitim uređajima [26]

Prethodna slika prikazuje korištenje tzv. efektivnih pixel-a. UWP aplikacija automatski prilagođava veličinu kontrola, fontova i drugih UI elemenata tako da prirodno izgledaju na svim uređajima. Kad je aplikacija pokrenuta na uređaju, sistem koristi algoritme za normalizaciju prikaza UI elemenata na ekranu. Ovaj algoritam uzima u obzir udaljenost sa koje se gleda i gustinu ekrana (broj pixela po inču) da optimizira percipiranu veličinu, radije nego stvarnu fizičku veličinu. Zbog načina na koji ovaj algoritam radi mi dizajniramo UWP aplikaciju u efektivnim pixel-ima. To se postiže skaliranjem objekata u koji čine UI množeći ih sa brojem 4, jer u ovisnosti od veličine ekrana, jedan pixel u biti predstavlja matricu od 4*4 pixel-a pomnoženu faktorom skaliranja. UWP aplikacije se prave koristeći programske jezike tipa C# ili VisualBasic u kombinaciji sa XAML-om (za UI), JavaScript sa HTML-om ili C++ sa DirectX-om. Visual Studio 2015 omogućava predloške za bilo koji odabrani jezik.

Došli smo do dijela kreiranja samog koda. Uredno složen i adaptivan dizajn se kroz kod postiže **RelativePanel** klasom i **Grid (matrica,mreža)** notacijom unutar XAML koda. Na početku, ekran je sastavljen od matrice dimenzija 1x1. Nakon toga mi definišemo redove i kolone u koje ćemo smještati elemente. Da bi se iskoristio puni potencijal *grid* elemenata najbolje je definisati širinu i visinu redova i kolona u procentima, da bi nakon pokretanja aplikacije na različitim ekranima omjer elemenata ostao isti. **RelativePanel** nam služi da definišemo gdje i na koji način će se elementi presložiti kad okrenemo telefon iz vertikalnog u horizontalni položaj ili otvorimo aplikaciju na nekom drugom uređaju. U biti sa tom klasom specificiramo gdje će se relativno nalaziti objekat u odnosu na neki drugi objekat ili sam pozadinski panel uređaja (lijevo, desno, ispod, iznad).

Postoji par tehnika na koje treba obratiti pažnju, a to su [26]:

- **Repozicija**

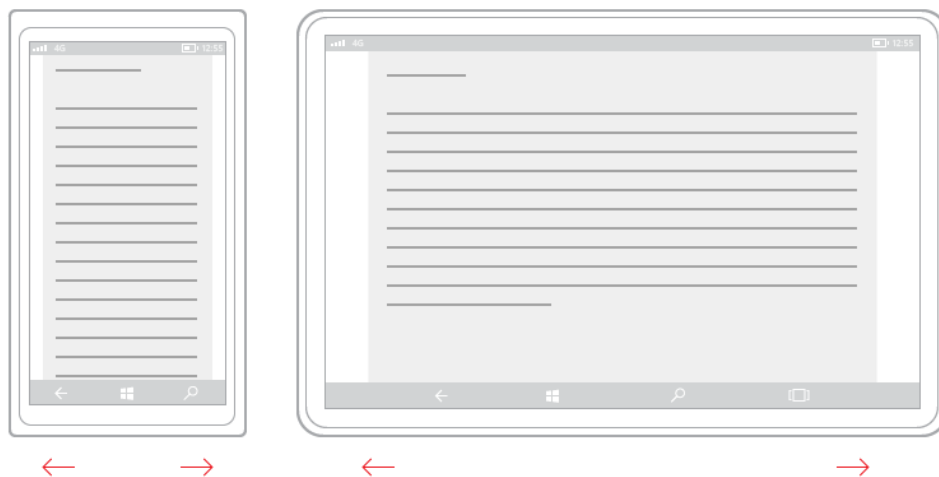
Da bi dobili najviše od našeg uređaja, pogodno je podesiti pozicije elemenata. U narednom primjeru vidimo dobro podešene elemente u ovisnosti od veličine ekrana.



Slika 2-20 Repoziacija elemenata na UI-u [26]

- **Promjena veličine**

Moguće je optimizirati veličinu okvira podešavajući margine i veličine UI elemenata. Pogodna situacija za takvo nešto je data na narednoj slici, gdje se se neki tekst širi i skuplja u ovisnosti od veličine ekrana.



Slika 2-21 Promjena veličine elemenata na UI-u [26]

- **Podešavanje količine elemenata iste veličine (*Reflow*)**

Podešavanjem toka (en. flow) možemo namjestiti optimalan prikaz sadržaja. Naredna slika prikazuje adaptaciju:



Slika 2-22 Reflow [26]

- **Sakrivanje i otkrivanje**

Pogodan sistem za popunjavanje prostora ako je neophodno za postizanje intuitivnijeg interfejsa. Ovom tehnikom prikazujemo ili sakrivamo željene metapodatke. Pogodno je na manjim uređajima ostaviti što čišći ekran u smislu zatrpanosti detaljima, dok na većim ekranima tipa računara nije loše zadržati neke detalje.



Slika 2-23 Prikazivanje i sakrivanje dijelova UI-a [26]

- Zamjena

Ova tehnika omogućava da za različite veličine ekrana imamo potpuno različite kako veličine, tako i oblike elemenata. Sljedeća slika demonstrira takvu situaciju:



Slika 2-24 Zamjena elemenata [26]

- Potpuna promjena

U nekim situacijama pogodno je u potpunosti zamijeniti dizajn aplikacije u ovisnosti od uređaja na kojem se koristi. Ovo možda izgleda kao zahtjevan i naporan posao, ali obično uz par novih klikova i podešavanja se dođe do željenog rezultata. Naredna slika prikazuje kako na tabletu imamo dvije kolone i u prvoj je npr. lista mailova sa njihovim pošiljaocima i odabirom nekog maila se u drugoj koloni prikazuje sadržaj istog. Na mobilnom uređaju je to organizovano da se klikom na određeni mail prikazuje njegov sadržaj, ali u novom prozoru jer zajedno bi bilo previše zgusnuto.

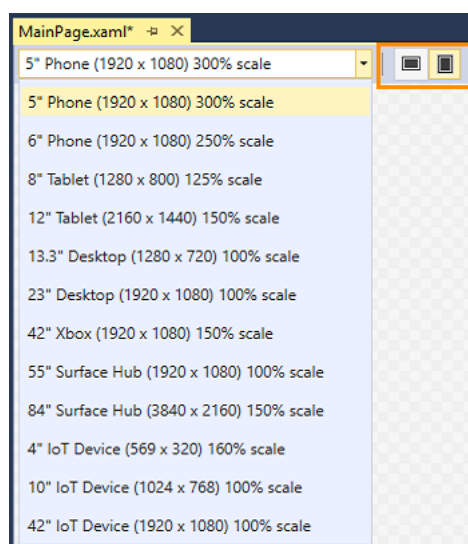


Slika 2-25 Potpuna reorganizacija [26]

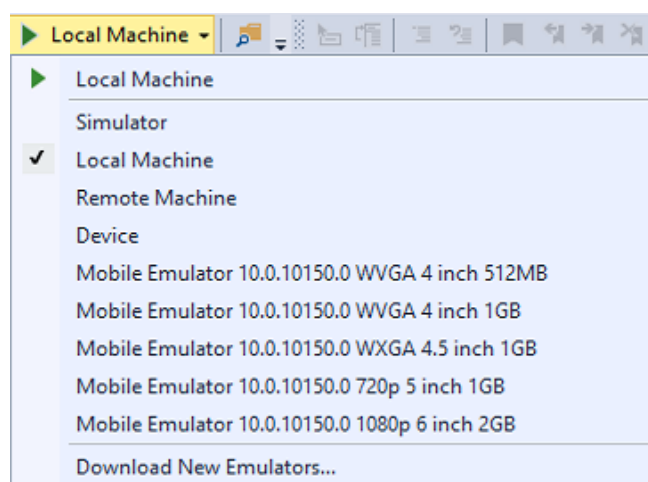
2.2.2. Visual Studio 2015 i UWP plugin [27]

Visual Studio 2015 je jedini IDE iz franšize koji omogućava razvoj UWP aplikacija. Kako sam naveo u prethodnom poglavlju, Visual Studio podržava dosta popularnih jezika koji se već duži period koriste u sklopu ovog okruženja, ali je neophodno instalirati neke dodatne *plugine* za kompajliranje i pokretanje UWP aplikacija. Prilikom instalacije ili *update*-a Visual Studio 2015 okruženja potrebno je izabrati sa liste ponuđenih instalacija „**Universal Windows App Development Tools**“. Unutar tog paketa se nalaze alati za Windows SDK (Software Development Kit) ⁴ i Emulator Windows 10 uređaja. Emulator je hardver ili softver koji omogućava jednom kompjuterskom sistemu (nazvanom *host*) da se ponaša kao drugi kompjuterski sistem (nazvan *guest*). Emulator omogućava *host* sistemu da pokreće programe ili koristi periferne uređaje dizajnirane za *guest* sisteme. Konkretno u našem slučaju emulator služi za simuliranje mobilnih uređaja različitih veličina ekrana i memorije. Windows koristi Hyper-X virtualne mašine za podizanje tih uređaja. Ako imamo dovoljno različitih fizičkih uređaja, nema potrebe za instalacijom emulatora, dok nam je SDK neophodan. Kada smo instalirali neophodni softver, trebamo kreirati našu UWP aplikaciju. Najbolje je započeti sa „Blank App (Universal Windows)“ predloškom koji nam je ponuđen prilikom kreiranja novog projekta. Sa desne strane unutar Solution Explorer prozora vidimo različite foldere i fajlove koji su kreirani, ali nas najviše interesuju **MainPage.xaml** i ako kliknemo strelicu pored imena, u hijerarhijskoj strukturi se pojavi još fajl pod nazivom **MainPage.xaml.cs**. Iz samih ekstenzija možemo pretpostaviti koja im je namjena. Prvi fajl predstavlja **xaml** fajl u koji stavljamo sve naše UI kontrole i u kojem kreiramo izgled aplikacije, a **.cs** ekstenzija je ekstenzija za klasu u C# programskom jeziku i služi nam za pisanje funkcija i evenata koji će se pokretati prilikom korisnikove interakcije sa aplikacijom. Kako sam naziv kaže, MainPage, predstavlja glavnu stranicu koja se prikazuje kada pokrenemo aplikaciju, a mi naknadno možemo dodavati još stranica i kreirati navigaciju kroz njih. Načini realizacije tih stvari izlaze iz opsega ovog rada, tako da im neće biti posvećena velika pažnja.

Prethodno sam spomenuo u koje svrhe koristimo emulator i na naredne dvije slike možemo vidjeti koji se sve uređaji mogu emulirati i na koje se sve načine aplikacija može pokrenuti.



Slika 2-26 Mogući uređaji koji mogu biti emulirani



Slika 2-27 Pokretanje aplikacije u različitim okruženjima

⁴ SDK- set razvojnih alata koji omogućavaju pravljenje aplikacija za određene softverske pakete, okruženja, hardverske platforme i sl.

2.3. Bluetooth [28]

Bluetooth tehnologija je sistem bežične, full duplex (dvosmjerne), komunikacije na kraćim udaljenostima, dizajnirana kako bi zamijenila kablove koji povezuju uređaje u komunikaciji. Pored ovog osnovnog postulata, Bluetooth je zamišljen kao siguran sistem za transfer podataka, a da je pri tome jeftin i ekonomičan. Da bi dva uređaja komunicirala preko Bluetootha, moraju proći kroz proces tzv. “uparivanja”, koji se sastoji u tome da pronađu jedan drugog na prethodno ustanovljenoj frekvenciji, te pomoću nekog vida sigurnosne provjere dozvole jedan drugom pristup, u smislu da mogu slobodno razmjenjivati podatke. Svi Bluetooth uređaji rade na istom principu, tako da se ovdje ponajviše iskazuje univerzalnost ove tehnologije.

Bluetooth komunikacija je neovisna o drugim mrežama. Kada se uređaji upare, između njih je uspostavljena samo njihova, privatna mreža, nazvana piconet. Bluetooth komunikacija je *master-slave* strukture, tako da je jedan uređaj *master* a drugi *slave*. Svaki uređaj u piconet-u može istovremeno vršiti razmjenu podataka sa do 7 drugih uređaja unutar tog piconet-a. U ovoj situaciji jedan je uređaj *master* a ostalih 7 su *slave* uređaji. Sat po kojem se odvija komunikacija je sat *master* uređaja, i *slave* uređaji moraju da prate master clock. Dva okidanja master clock-a čine jedan vremenski prozor koji traje 625 mikrosekundi. Dvosmjerna komunikacija se ostvaruje tako što master šalje podatke u parnim vremenskim prozorima a prima podatke u neparnim.

Na nivou radio-talasa, Bluetooth koristi frekventni opseg umjesto jedne fiksne frekvencije. Ovakav pristup osigurava bolji kvalitet prenosa kao i mnogo brži transfer podataka. Taj frekventni opseg leži između 2400 i 2483.5 MHz. Ovo je globalno rezervisani i neregulirani ISM opseg. Radio tehnologija koja se koristi za ovakav pristup se svodi na prebacivanje između kanala unutar datog opsega i slanje paketa na tim odvojenim kanalima. Ovakvih odvojenih kanala za Bluetooth postoji 79, svaki sa propusnosti od 1 MHz. Najnoviji standard Bluetootha, 4.0 koristi 40 kanala sa po 2 MHz. Ovakav pristup daje mogućnost brzog prebacivanja i paralelnog slanja podataka razdvojenih u pakete. Ukoliko je na komunikaciju primjenjeno adaptivno prebacivanje između frekvencija - AFH, gdje se biraju samo “dobri” kanali, tj. oni koji su trenutno najmanje zagušeni saobraćajem, može se desiti i do 1600 prebacivanja između kanala u sekundi. AFH omogućuje nesmetanu komunikaciju čak i kada se istovremeno koriste i druge tehnologije između uređaja u komunikaciji, čime se pruža potrebni kvalitet.

Udaljenost koja mora biti zadovoljena da bi uređaji mogli da se povežu zavisi od uređaja do uređaja, uzimajući u obzir da se minimalna udaljenost od 10 metara, propisana Bluetooth Core specifikacijom, mora zadovoljiti. Svaki proizvođač može da dalje povećava udaljenost koliko želi, ovisno od tehnologije koju koristi kao i kvalitete transceiver-a.

Potrošnja Bluetooth uređaja, proizvedenih po standardnoj Bluetooth specifikaciji, iznosi 2.5mW. Kako bi se ovo minimiziralo, uređaji su dizajnirani da se gase kada nisu aktivni. Također, noviji standardi Bluetooth-a su donijeli i Bluetooth LE – Bluetooth sa veoma niskom potrošnjom, od 1/2 do 1/100 potrošnje klasičnog Bluetooth uređaja. Ovakav dizajn je posljedica mnogobrojnih zahtjeva da inkorporiranjem Bluetooth-a u uređaje od kojih se zahtijeva veoma dugo trajanje baterije.

2.3.1. Upotreba Bluetooth tehnologije

Uzimajući u obzir da je za Bluetooth komunikaciju potrebno biti u neposrednoj blizini uređaja sa kojim se komunicira, upotreba se uglavnom zadržava na nivou PAN-a – Personal Area Network, mreže koja opisuje komunikaciju na nivou kompjutera, mobilnog telefona i sličnih uređaja. Da bi koristio Bluetooth, uređaj mora biti u mogućnosti da interpretira određeni Bluetooth profil, koji je u biti specifikacija ponašanja određenih Bluetooth uređaja. Profili Bluetooth uređaja mogu dati grubu podjelu upotrebe ove tehnologije u današnjem svijetu.

Jedna od najraširenijih upotreba Bluetooth tehnologije jeste u segmentu automobila. Jedna od prvih i najpopularnijih upotreba Bluetootha jeste bežična slušalica sa mikrofonom za mobilne telefone. Ovo omogućava vrlo laku kontrolu nad pozivima kada osoba koja koristi mobilni telefon nije u mogućnosti da drži sam uređaj u ruci za vrijeme razgovora. Ovakav način razgovora je postao posebno popularan pri vožnji, jer omogućava punu kontrolu nad vozilom u toku razgovora. Također, bitno je spomenuti povezivanje mobilnih telefona i zvučnih sistema koji su ugrađeni u automobile. Nakon što se mobilni uređaj i automobil povežu, muzika se dobavlja sa telefona a kontroliše u automobilu. Tu je i GPS navigacija. Pomoću bežične slušalice je moguće kontrolisati uređaj za navigaciju, bez potrebe za distrakcijama tokom same vožnje.

Bluetooth tehnologije je veoma raširena i u segmentu kućne tehnologije. Sa pojavom pametnih televizora koji koriste posljednje verzije Android operativnog sistema, širi se potreba za naprednim kontrolerima koji koriste bežičnu tehnologiju, kako bi se televizori mogli upravljati iz bilo kojeg dijela prostorije u kojoj se nalaze. Razlika između naprednih kontrolera i običnih daljinskih upravljača jeste što napredni kontroleri posjeduju čitav spektar funkcija koje ga dovode mnogo bliže klasičnoj tastaturi i mišu, nego samom upravljaču za televizor. Kontroleri koji koriste Bluetooth nisu zaobišli ni igraće konzole, pa posljednje verzije Sony Playstation konzole, kao i Microsoft Xbox-a koriste kontrolere koji sa sistemom komuniciraju zahvaljujući Bluetooth tehnologiji. Vrijedi spomenuti i bežične zvučnike, koji nalaze sve šire aplikacije, kako u oblasti tehnologije doma, tako i u sportskim aktivnostima. Bitno je spomenuti i moderne brave, termostate, nadzorne sisteme i sisteme rasvjete koji se unutar vlastitog doma mogu kontrolisati sa jednog centralnog uređaja. Ovo omogućava bolju sigurnost, lakše upravljanje, kao i nadzor potrošnje energije, smanjujući troškove. U segmentu poslovnih prostora, raširena je upotreba Bluetooth-a za bežično slanje podataka na printere, tako da se jedan printer može koristiti u čitavoj prostoriji, iz ugodnosti vlastitog stola, bez potrebe za fizičkom vezom.

Još jedna, u posljednje vrijeme sve popularnije primjena Bluetooth-a se nalazi u sportu, posebno u fitness-u. Pametne vage, monitori rada srca i potrošnje kalorija, pružaju detaljniji i dublji uvid u životne navike i korist koju daje fizička aktivnost. Ovo omogućava korisniku postavljanje ciljeva i praćenje napretka tokom same sportske aktivnosti. Ovu mogućnost koriste i profesionalni sportisti i sportski timovi, koji detaljno prate rad članova tokom treninga ili neke slične aktivnosti. Ovakvi podaci mogu rezultovati u promjenama pri samom treningu ili personaliziranju treninga prema nekom od članova, kako bi se lakše adaptirao ili postigao bolje rezultate.

2.4. 3D (Binaural – dva uha) zvuk [29]

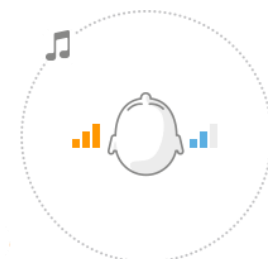
Binaural snimanje je metod snimanja zvuka koji koristi dva mikrofona postavljena s namjerom kreiranja osjećaja 3D stereo zvuka za onoga ko sluša. Efekat je često postignut koristeći tehniku poznatu kao „snimanje uz pomoć lutke“ gdje se lutki u oba uha stave mikrofoni i na taj način se snimaju zvukovi iz okoline. Tim metodom snimljen zvuk je namijenjen za reproduciranje putem slušalica, jer putem zvučnika ne dobijamo potpun efekat. Ljudi često miješaju pojmove stereo i binaural. Glavna razlika je u tome što stereo zvuk ne uzima u obzir fizički razmak ušiju i pojavu pod nazivom „*head shadow*“. *Head shadow* u bukvalnom prijevodu znači „sjenka glave“ i predstavlja prepreku koju glava pravi zbog koje smanjuje amplitudu dolazećih zvukova. S obzirom da su uši u različitim položajima, svaki zvuk koji čujemo putuje drugačijom putanjom do uha. Osnova 3D percepcije zvuka, je naša sposobnost da lociramo odakle zvuk dolazi i koliko je blizu ili daleko. U biti, postoje 3 glavna faktora tehnologije 3D zvuka koji zajedno kreiraju ovu senzaciju.

- **ITD** („*Inter-aural Time Difference*“ - vremenska razlika unutar uha) - U ovisnosti od smjera odakle zvuk dolazi, sigurno će doći u jedno uho prije nego u drugo



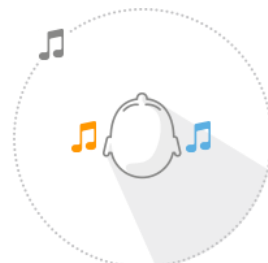
Slika 2-28 ITD [29]

- **ILD** („*Inter-aural Level Difference*“ - razlika u glasnoći unutar uha) – Jedno uho će percipirati određeni zvuk glasnije nego drugo, u ovisnosti opet od pozicije i udaljenosti od izvora zvuka



Slika 2-29 ILD [29]

- **HRTFS** („*masking and Head-Related Transfer Functions*“ – funkcije prenosa zvuka ovisne o glavi) – Uho koje je dalje od izvora zvuka će čuti malo drugačiji zvuk; ovo se dešava jer zvuk mora proći oko glave i biće odbijen od strane površina oko glave i dodatno je izmijenjen od strane ulaza u vanjsko i unutrašnje uho



Slika 2-30 HRTFS [29]

2.4.1. Tehnika snimanja [30]

Snimanje se vrši pomoću dva mikrofona postavljena na udaljenost oko 18cm jedan od drugog usmjereni u suprotnim smjerovima. Da bi maksimizirali efekat osjećaja položaja predmeta u prostoru bio potpun, pored ovako postavljenih mikrofona potrebno je dizajnirati model uha i pozicionirati mikrofone unutar tog modela. Tu lutka igra važnu ulogu. Pored samog modela uha, rješavamo problem i *head shadow* pojave sa glavom lutke. Što je kvalitetnije napravljena lutka, realističniji zvuk dobijamo (bitno je kako se zvuk odbija od vanjsko uho, pa unutrašnje uho i sve do opne). Najčešće upotrebljavani mikrofoni su mikrofoni kompanije Neumann, a konkretni modeli su KU-81 i KU-100, posebno od strane muzičara.

2.4.2. Reproduciranje zvuka [30]

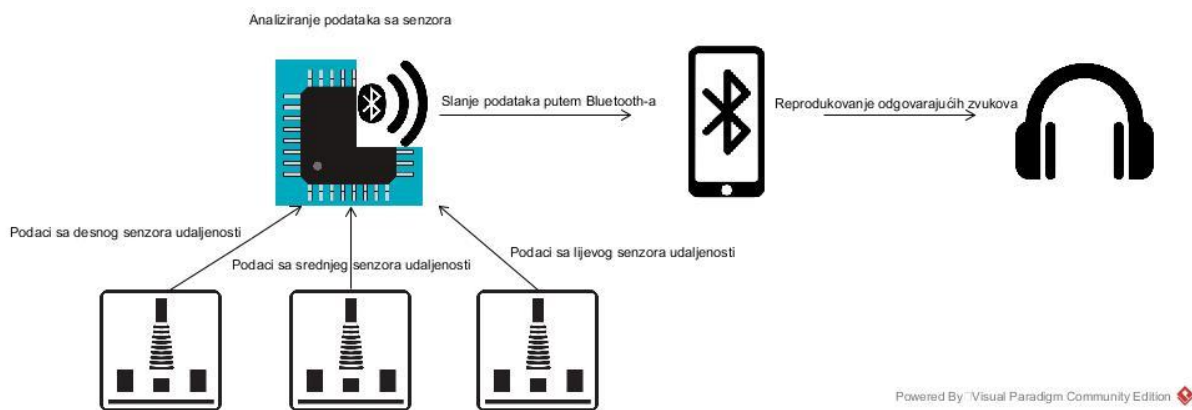
Kada su snimljeni, *binaural* efekti se mogu reproducirati koristeći slušalice ili tzv. dvopolni stereo. Reprodukcijska ne radi sa mono zvukom ili sa velikim zvučnicima zbog akustičnih kutija u koje su ugrađeni, koje stvaraju distorziju zvuka i efekti se gube. Bilo koji set slušalica koji omogućava dobru izolaciju lijevog i desnog kanala je dovoljno dobar za reprodukciju 3D zvukova. Čak i jeftiniji modeli bez većih problema podržavaju ove zvukove jer generalno nisu frekvencijski zahtjevni. Neke kompanije su napravile i posebne serije slušalica posebno za puštanje 3D zvukova. Čak su neke kompanije koje razvijaju pojačala za slušalice kreirale posebne hardverske strukture kako bi maksimalno iskoristili potencijal zvukova. Pokazalo se da *in-ear* (male slušalice koje idu direktno u ušni kanal) slušalice stvaraju lošiji doživljaj s obzirom da se gubi efekat odbijanja zvuka od vanjsko i unutrašnje uho, tako da su dosta pogodnije otvorene *over-ear* (velike slušalice koje idu preko cijelog uha). Hipoteza je da kada je ušni kanal potpuno blokiran impedansa zvuka koja dolazi do bubnjića bude promijenjena, što negativno utiče na sveukupni dojam. Postoje neke komplikacije u vezi sa reprodukcijom zvuka na običnim slušalicama koje se tiču same izrade. Obzirom da su pretežno pravljene za puštanje stereo zvukova na frekvenciji od oko 5Hz imaju takozvani usjek ili prelom. U situacijama reproduciranja 3D zvuka, neophodno je da su slušalice „ravne“ i da nema tih preloma. U teoriji je moguće ispraviti sve te prelome i napraviti „ravni“ odziv, ali je puno bolje koristiti slušalice koje su originalno dizajnirane da to rade. Ima tu još par faktora koji utiču na potpuni doživljaj, a tiču se vanskog okruženja, veličine uha (teško je napraviti slušalice koje idealno odgovaraju svim veličinama ušnih školjki), oštećenja same opne ili kanala i sl.

3. Praktični dio

3.1. Funkcionalni opis sistema

S obzirom da nigdje u okviru rada nisam spomenuo naziv aplikacije, ovo je pravo mjesto da to učinim. Aplikacija se naziva „**FeelTheSpace**“ što u bukvalnom prijevodu znači „Osjeti prostor“, a za njenu izradu su korištene prethodno pojašnjene tehnologije i alati. Dakle, u okviru platforme postoje individualne komponente, koje integrisane u jednu cjelinu čine sistem za pomoć slijepim i slabovidnim osobama prilikom kretanja. Sistem je koncipiran tako da se glasovnom komandom pokrene aplikacija na Windows 10 uređaju, nakon čega se putem tri HC-SR04 ultrazvučna senzora postavljen na opasač mjere udaljenosti do predmeta koji se nalaze ispred njih. Pomoću Arduino mikrokontrolera se obrađuju te informacije (filtriraju greške prilikom mjerenja, traže optimalne vrijednosti, pripreme podaci za slanje) i dobijeni rezultati pomoću HC-05 Bluetooth modula proslijede na Windows 10 uređaj koji u ovisnosti od tih rezultata aktivira različite zvučne signale. Ovako na prvu sistem možda izgleda konfuzno ali ako ga rastavimo na pojedinačne dijelove i objasnimo na primjeru, vjerujem da će postati jasniji. Krenimo redom. Slijepa ili slabovidna osoba posjeduje FeelTheSpace aplikaciju na svom/jim Windows 10 uređaju/ima i prateći hardver. Množina je u ovom kontekstu vrlo bitna jer je aplikacija pravljen koristeći UWP platformu i samim tim je podržana od strane svih uređaja sa Windows 10 jezgrom. Uzmimo za primjer da korisnik ima mobilni uređaj i krenuo je u šetnju u obližnji park. Na tom putu postoji dosta prepreka tipa kanti za smeće, klupa, ljudi koji hodaju i slično. Na pojas zakači senzore sa mikrokontrolerom, u mobilni uređaj uključi slušalice i spreman je za polazak. Aplikacija se aktivira glasovnom komandom „**feel, start navigating**“. Postoji određeni nivo tolerancije na greške prilikom pokretanja aplikacije koje mogu biti uzrokovane pozadinskim zvukovima iz okoline ili samim izgovorom riječi na engleskom jeziku. Program također prihvata i naredbe poput „**feel, start navigation**“ ili „**feel, start guiding**“. Detaljnije objašnjenje ove konstrukcije će biti dato u nastavku. Nakon što je aplikacija pokrenuta, Windows 10 uređaj pokušava uz pomoć Bluetooth tehnologije uspostaviti komunikaciju sa Arduino uređajem, konkretno Bluetooth modulom HC-05 spojenim na Arduino platformu. U toj komunikaciji mobitel je *master*, dok je BT modul *slave*. Uređaji se sami pokušavaju upariti i u slučaju greške glasovnom instrukcijom korisnik biva obavješten o neuspjeloj konekciji, te se upućuje da provjeri da li je upaljena BT veza na mobitelu. Kada prođe proces uparivanja, aplikacije su spremne i puštaju se inicijalni zvukovi na lijevoj i desnoj slušalici kako bi korisnik znao da li je dobro stavio slušalice, jer bi njihova zamjena mogla imati kobne posljedice. Ultrazvučni senzori HC-SR04 vrše više uzastopnih mjerenja udaljenosti do najbližeg predmeta u kratkom vremenskom periodu i filtriraju dobre vrijednosti. Ako uzmemo za primjer da se kanta nalazila 1.2m desno od nas, dok ispred i lijevo od nas nema predmeta bliže od 1.2m, desni senzor će imati očitavanje koje je najbliže. Za tu obradu podataka sa senzora i odabir najbližeg je zadužen Arduino. Kada je to obavljeno, odabrana distanca se putem BT veze šalje na mobilni uređaj zajedno sa senzorom koji ju je očitao. U skladu sa dobijenim podacima aplikacija odabire odgovarajući zvuk iz liste zvukova i reprodukuje ga na slušalice.

U ovom poglavlju je predstavljen samo generalni koncept na kojem funkcioniše aplikacija, a detaljniji opis sa tačnim udaljenostima i tipovima zvukova će biti dat u nastavku.



Slika 3-1 FeelTheSpace shema

3.2. Uspostavljanje platforme

Platforma se sastoji iz sljedećih komponenti:

- HC-SR04 ultrazvučni senzor udaljenosti
- Arduino Uno
- HC-05 Bluetooth modul
- UWP aplikacija
- 3D zvuk

te će doprinos i konkretan razlog odabira svake od njih će biti pojedinačno objašnjen. Prikaz sheme spajanja dat je u dodatku A.1.

3.2.1. HC-SR04 ultrazvučni senzor udaljenosti

O ovom senzoru je bilo riječi u poglavlju 2.1.5.1 i njegova izgled je dat na slici: Slika 2-14 HC-SR04 ultrazvučni senzor udaljenosti. Objašnjen je način upotrebe i za koje svrhe se koristi koji pin, tako da nema potrebe ponavljati taj dio. Uloga ultrazvučnih senzora u ovom projektu je da očitavaju stvarne udaljenosti do prepreka tako što su zakačeni na pojas. Postoji 5 faktora koji su razmatrani prilikom odabira senzora. Cijena, performanse, pouzdanost, tehnologija koju koristi i veličina. Naredna tabela prikazuje opcije koje su bile razmotrene.

Senzori [31]					
	Cijena	Performanse	Pouzdanost	Tehnologija	Veličina
<i>hc-sr04</i>	~1\$	srednje	srednja	ultrazvuk	srednja
<i>Maxbotix LV-EZO</i>	35 \$	visoke	visoka	ultrazvuk	mala
<i>Sharp GP2Y0A21YK IR Proximity Sensor</i>	4 \$	srednje-niske	srednja-niska	infrared	srednja
<i>QRD1114 IR emitted / Phototransistor</i>	0,5\$	niske	niska	infrared/fotootpor	ultra mala

S obzirom na podatke iz tabele, smatram da su prva dva senzora najpogodnija za realizaciju projekta, s tim što bi se Maxbotix senzor koristio za prestižniju verziju, a slabija verzija bi išla sa SR04 senzorom. Maxbotix senzor je jedan od najboljih ultrazvučnih senzora na tržištu i s razlogom je ušao u razmatranje. Poprilično je skup, čak skuplji i od samog Arduino uređaja, ali vrijedi svakog dolara. Mjeri udaljenosti do 650 cm bez grešaka. Izlaz je linearan, tako da je vrlo lako očitati stvarnu udaljenost. Jedini problem je što prilikom očitavanja objekata unutar 30cm, može biti malo nestabilan, ali s obzirom na upotrebu u sklopu ovog projekta to apsolutno ne predstavlja nikakav problem. Senzor ima nekoliko izvedbi sa različitim širinama zraka koje emituje i poželjna bi bila neka verzija sa ne toliko širokim talasima, čisto zbog preklapanja sa drugim senzorima. Pored sigurnosti, bitan faktor je i cijena (iako nekih 130\$ nije previsoka cijena za ovakav uređaj), potrebno je napraviti i jeftiniju verziju, ali dovoljno sigurnu i pouzdanu. Jedina stvar koja bi bila zamijenjena u toj verziji je senzor. Za izradu završnog rada sam koristio HC-SR04 senzor koji je cjenovno dosta jeftiniji, a pokazao je zavidne rezultate. Ovo je senzor koji dolazi u dosta Arduino Starter Kit paketa i na taj način je stekao popularnost. Mjeri udaljenosti do 5 metara i registruje predmete koji se nalaze direktno ispred njega. Na ovom radu su korištena 3 senzora tako da karakteristika registrovanja predmeta isključivo ispred njega ne predstavlja veliki problem. Jedini problem je velika osjetljivost. „hvata“ i najmanje pokrete, tako da se kodom mora raditi filtriranje podataka da bi dobili normalna očitavanja, ali postoje implementirane biblioteke koje nam olakšavaju taj posao i funkcije sve same obavljaju. Ovo je razlog zašto sam stavio da su pouzdanost i performanse srednje.

3.2.2. Arduino Uno

Arduino platforma i konkretno Uno model je trenutno najpopularniji *DIY* uređaj za razvoj elektroničkih projekata na svijetu i nije bilo potrebe da bude zaobiđen. S obzirom na *open-source* platformu i veliku zajednicu koja doprinosi njegovom razvoju, nije postojalo puno problema za čije rješenje nije postojao neki prijedlog na internetu. Neophodno je bilo da posjeduje 5V izlaz i uzemljenje, mogućnost serijske komunikacije putem UART interfejsa (RX i TX) i 6 digitalnih pinova. ATmega328 mikrokontroler posjeduje sve navedene stavke i obzirom da radi na frekvenciji od 16MHz sa 32Kb memorije bio je idealan kandidat za projekat. Njegov zadatak je bio da pomoću funkcija iz biblioteke **NewPing** napravi 10 očitavanja na svakom pojedinačnom senzoru, filtrira greške, nađe srednju udaljenost pojedinačno po senzorima i usporedi međusobno te rezultate. Usporedba rezultata podrazumijeva nalaženje najmanje vrijednosti od 3 koje su produkt filtriranja. Nađena vrijednost se potom proslijeđuje serijskoj

komunikaciji, odnosno *bluetooth*-u u fromatu **senzorUdaljenost (npr. c 150-** što znači da je najmanja od 3 udaljenosti ona koju je očitao centralni senzor i to je 150 cm). U slučaju da se očitavanja sa dva susjedna senzora razlikuju za manje od 7 cm (što je po mišljenju autora ovog rada dovoljno da se zna da je objekat ispred senzora dovoljno velik da obuhvata oba senzora) onda se šalje komanda istog formata sa drugim početnim slovom. Obzirom da sva slova nisu toliko intuitivna, detalji će biti objašnjeni u dijelu sa kodom.

3.2.3. HC-05 Bluetooth modul [28]

Za Arduino platformu pored originalnog *shield*-a proizvedenog od strane istoimene kompanije, postoji još par *Bluetooth* modula. Modeli s kojima sam se susreo su HC-05, HC-04 i HC-06 kompanije Linvor. Razlika između njih je što prvi navedeni model može u komunikaciju biti i *master* i *slave*, dok druge dvije verzije mogu biti samo *slave* tipa. HC-05 model koristi SPP (Serial Port Protocol) i 2.0 verziju *Bluetooth-a*, sa povećanom propusnosti od 3Mb u sekundi i radio primopredajnikom brzine 2.4GHz. Na sebi ima CSR Bluecore 04 *Bluetooth* čip sa adaptivnim mijenjanjem kanala u frekventnom opsegu. Navedeni modul ima 2 načina rada, AT komandni mod za konfiguraciju i Data mod za razmjenu podataka sa drugim *Bluetooth* uređajem. Kroz AT mod, između ostalog, HC-05 može biti konfigurisan da bude *master* ili *slave* u *Bluetooth* vezi. Za naš sistem nije bilo potrebe posebno kofigurisati ulogu modula.

Bitne karakteristike:

Hardverske karakteristike	Softverske karakteristike
Radi na naponima od 3.6 do 6 volti	Podržava opsek promjena frekvencije u sekundi od 9600 do 460800
Daje do 4 dBm snage	U komunikacije koristi 8 bita za podatke, a oznaka kraja prenosa je 1
Posjeduje UART interfejs i programiranu UI kontrolu	Automatska konekcija sa posljednjim uparenim uređajem ukoliko je aktivan
Ima integrisanu antenu	Moguće podesiti primarni uređaj za konektovanje
	Može komunicirati samo sa jednim uređajem u jednom trenutku

HC-05 posjeduje dvije LED lampice za signaliziranje stanja u kojem se nalazi. LED1 je indikator *Bluetooth* načina rada. Ukoliko se LED1 gasi i pali brzo (2 Hz) to znači da uređaj nije povezan i da čeka na zahtjev za povezivanje. Ako se LED1 gasi i pali sporo (1 Hz) to znači da je u procesu povezivanja, dok se stanje u kojem je modul povezan očitava tako što se LED1 brzo upali i ugasi dva puta u sekundi. LED2 također služi indikator stanja uparivanja. Ugašena je kada uređaj nije uparen ni sa jednim drugim uređajem, a upaljena kada je uređaj uparen.

Pri samom povezivanju drugog uređaja sa modulom potrebno je na drugom uređaju unijeti šifru putem koje se postiže sigurnosna provjera. U slučaju HC-05 modula, šifra je "1234". Nakon samog povezivanja potrebno je uspostaviti konekciju, o čemu će detaljnije biti govora u samom pojašnjenju koda UWP aplikacije. Specifična situacija u kojoj se naš sistem našao jeste da nije bilo dovoljno brzo slanje zahtjeva za konekcijom, uspostavljanje konekcij, slanje informacija sa senzora i zatvaranje konekcije. Ovo je uzimalo previše vremena, jer je sistem veoma interaktivan i zahtjeva brzu komunikaciju, što nije bilo ostvarivo ukoliko bi se svaki put otvarala i zatvarala konekcija, pogotovo

ukoliko uzmemo u obzir da je potrebna full-duplex komunikacija. Način na koji je premošten ovaj problem jeste da se na početku otvara konekcija i uspostavlja se protok podataka. Nakon zatvaranja aplikacije, zatvara se i konekcija.

Modul je relativno malih dimenzija 28 mm x 15 mm x 2.35 mm. Došao je spojen na TTL ploču sa 6 potrebnih pinova, tako da smo ga mogli direktno spojiti na matador. Funkcije pinova su kako slijedi:

- EN: Ukoliko je poslano 3.3 V na ovaj pin prije nego je sam modul uključen, ulazi se u AT konfiguracijski mod.
- VCC: Pin za napon
- GND: Uzemljenje
- TX: Pin za slanje podataka
- RX: Pin za primanje podataka
- STATE: Očitavanje stanja konekcije



Slika 3-2 Bluetooth modul HC-05 [28]

U okviru projekta, modul je korišten za komunikaciju između dvije fizički odvojene platforme (Arduino i UWP) i prenos podataka u obliku mjerenih udaljenosti od Arduino aplikacije do UWP aplikacije.

3.2.4. UWP aplikacija

Postoje dvije činjenice koje su me oduševile i zbog kojih sam odlučio na korisničkoj strani koristiti UWP aplikaciju. To bi bila njena univerzalnost, odnosno ideja koja na prvo mjesto stavlja familiju uređaja, a ne operativni sistem i druga stvar je mogućnost integrisanja Cortana personalnog asistenta. Karakteristika pokretanja FeelTheSpace aplikacije na raznim platformama sa Windows 10 jezgrom omogućava korisniku kretanje po kući sa laptopom (ako je neophodno), kao i van kuće sa tabletom, mobilnim uređajem i slično. Sve dok je aplikacija instalirana na svim uređajima, nema brige koji će od njih biti upotrebljen. Moje mišljenje je da potencijal Cortana sistema nije dovoljno iskorišten i pomoć slijepim i slabovidnim osobama bi bio jedan smjer u kojem bi se moglo krenuti kako bi iskoristivost bila veća. Smatram da je to vrlo pogodno tlo za razvijanje inovativnih aplikacija koje bi mogle olakšati svakodnevnicu ljudima sa poteškoćama.

Kao što sam već spominjao FeelTheSpace aplikacija se pokreće glasovnom komandama poput „feel, start navigating“, „feel, start navigation“ ili „feel, start guiding“. Vidimo da je to ograničeni skup predefinisanih komandi. Na početku se izgovara riječ „feel“ i tu nema varijacija. „Uvodna“ riječ služi da sistem zna na kojem mjestu, odnosno za koju aplikaciju će vezati ostatak naredbi, tako da u biti „feel“ označava FeelTheSpace aplikaciju i onda ostatak rešenice je set komandi koje ta aplikacija može izvršiti. Definisane su dvije promjenjive, gdje se na mjestu prve promjenjive očekuju naredbe „start“ ili „stop“, dok na mjestu druge promjenjive mogu biti komande poput „navigation“, „navigating“ ili „guiding“. Ovaj set komandi bi mogao biti i duži, ali s obzirom da Windows 10 pretraživač baziran na Cortana sistemu ima dobru toleranciju na greške, naredbe se prepoznaju iako korisnikov izgovor nije baš najbolji ili postoji pozadinska buka. Kada se pokrene aplikacija, aktiviraju se pojedinačno lijevi i desni zvuk kako bi korisnik znao da li je dobro stavio slušalice. Nakon toga se inicira *Bluetooth* konekcija i primanje podataka počinje. Poruke pristižu a funkcije ih tumače i puštaju odgovarajuće zvukove. Konkretno postavke će biti objašnjene u dijelu koji je posvećen kodu. Podešeno je da se zvukovi ne preklapaju, tako da tek po završetku jednog zvuka može biti pušten drugi. Što se samog korisničkog interfejsa tiče, smatram da nije bio neophodan neki dizajn sa mnogo grafičkih elementa obzirom na ciljanu skupinu korisnika. Uprkos tome, napravljen je adaptivni dizajn koji se prilikom okretanja ekrana, povećavanja ili smanjivanja uredno širi i skuplja. Na narednim slikama je prikazan izgled aplikacije na mobilnom uređaju i nekom desktop/tablet uređaju horizontalne orijentacije:



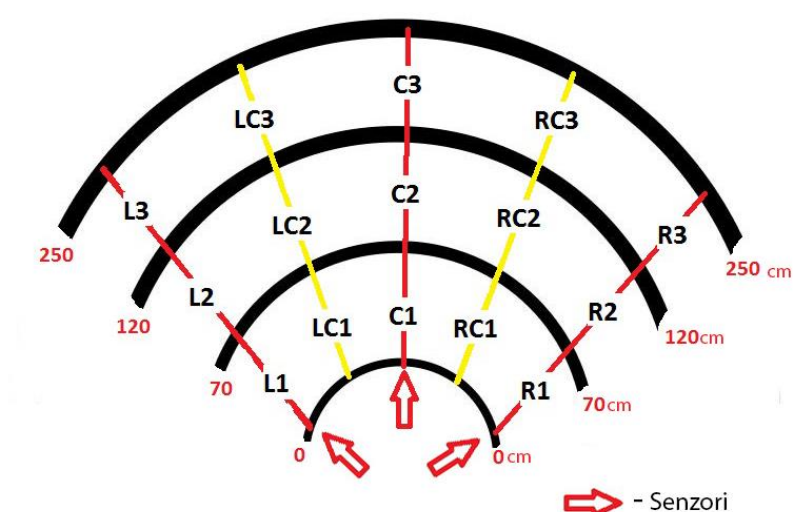
Slika 3-3 Mobilna verzija
FeelTheSpace aplikacije



Slika 3-4 Desktop verzija FeelTheSpace aplikacije

3.2.5. 3D zvuk

Osnove 3D zvuka smo objasnili, još je ostalo da se kaže nešto o njegovoj primjeni u ovom projektu. 3D zvuk je finalni produkt cijelog procesa; od prikupljanja informacija, obrade, slanja bežičnim putem, još jedne obrade i tek na kraju dolazi reprodukcija. Neophodno je da se dobro poslože svi prethodni elementi da bi i zvuk bio korektan. Biće reprodukovani zvuk koji odgovara onom senzoru koji je od 10 mjerenja u odnosu na 10 mjerenja druga dva senzora imao najmanje očitavanje.



Slika 3-5 Udaljenosti i područja očitavanja senzora

Prethodna slika može izgledati malo konfuzno, ali šta ona u biti predstavlja. Kao što piše u legendi crvene strelice predstavljaju senzore i njihove položaje. Crvene linije ukazuju na smjer u kojem je poslan zvuk sa svakog senzora i iz istog smjera se očekuje eho, dok žute linije označavaju kombinacije očitavanja dva susjedna senzora. Crnim lukovima su označene određene udaljenosti. Kako to sad sve zajedno funkcionira; L je oznaka lijevog senzora (od *left*- lijevo), C oznaka centralnog senzora (od *center*- centar) i R je oznaka desnog senzora (od *right* – desno). Intuitivno se da zaključiti da LC predstavlja kombinaciju lijevog i centralnog senzora, dok RC predstavlja kombinaciju desnog i centralnog senzora. Brojevi pored slovnih oznaka predstavljaju interne oznake za region u kojem se nalazi prepreka, pa vidimo da je **od 0 do 70 cm region 1, od 70 do 120 region 2 i od 120 do 250 region 3**. Senzori su podešeni da sve objekte koji su dalje od 250 cm uopšte ne registruju. Ovo je urađeno da se smanji broj pogrešnih očitavanja, a i realno nema potrebe za uočavanjem objekata na 5 m. Čak ni komercijalni štamp koji masovno koriste slijepe osobe nema domet veći od 2,5 m. Dakle, u ovisnosti od informacije koja dođe do UWP aplikacije pustit će se neki od 15 predefinisanih zvukova. Zvukovi su unaprijed snimljeni i podešeni tako da po glasnoći i smjeru odgovaraju shemi datoj na slici 3-5. FeelTheSpace aplikacija i cjelokupan sistem bi bio samo obični Arduino projekat za eholokaciju objekata u prostoru kakvih ima pregršt na internetu da nema 3D zvuka. Bluetooth komunikacijom, UWP aplikacijom i 3D zvukom je projekat dignut na viši nivo na kojem se kombinuju mogućnosti Cortana personalnog asistenta, univerzalne aplikacije koja radi na različitim uređajima i osjećaja prostora koji se postiže 3D zvukom. Kako je Arduino platformi omogućeno da osjeti svijet oko sebe, istu senzaciju bi trebao dobiti i čovjek.

3.3. Arduino aplikacija

Cjelokupan programski kod sistema je podijeljen na kod aplikacije Arduino mikrokontrolera i UWP aplikacije. U ovom dijelu su obrađene implementirane funkcije Arduino koda i korištene eksterne biblioteke

3.3.1. Biblioteka New Ping [32] [33]

Biblioteka New Ping pruža mogućnosti korištenja HC-SR04 ultrazvučnog senzora bez potrebe za manuelnom aktivacijom pojedinačnih pinova i čekanjem odziva. Osnovni cilj ove biblioteke je pojednostavljenje upotrebe navedenog senzora uz par osnovnih funkcija. Prednost u odnosu na ručno pisanje funkcija i druge biblioteke je odsustvo kašnjenja od jedne sekunde u slučaju da se eho nije vratio. Uz pomoć ove biblioteke, senzori mogu pouzdano vršiti očitavanja čak do 30 puta u sekundi. Ostale prednosti su: lagana upotreba više senzora, preciznija očitavanja u vidu centimetara, inča ili mikrosekundi, ugrađen digitalni filter vrijednosti za jednostavnu korekciju grešaka itd.

Arduino kod je koncipiran tako da se pojedinačno proziva svaki od 3 senzora i s obzirom na benefit od 30 očitavanja u sekundi iskorištena je funkcija **ping_median(uint8_t readings)** koja vrši *readings* mjerenja i traži srednju vrijednost. Da bi uopšte mogli koristiti NewPing biblioteku, potrebno ju je prvo uvesti pomoću include iskaza na vrhu koda:

```
#include <NewPing.h>
```

Nakon toga možemo iz bilo kojeg dijela koda pozvati neku od funkcija iz ove biblioteke. Kreiranje objekta tipa NewPing se obavlja uz pomoć konstruktora u obliku:

```
NewPing(12, 13, MAX_DISTANCE);
```

gdje prvi parametar predstavlja broj pina na Arduino mikrokontroleru na koji je uključen Trig pin senzora, drugi parametar služi za Echo pin i treći parametar je maksimalna udaljenost do koje će senzor očitavati vrijednosti, sve preko toga će smatrati greškom.

Najbitnija funkcija unutar NewPing biblioteke koja je korištena za ovaj projekat je **ping_median()** i njena implementacija je data u narednim redovima:

```
unsigned int NewPing::ping_median(uint8_t it) {
    unsigned int uS[it], last;
    uint8_t j, i = 0;
    uS[0] = NO_ECHO;
    while (i < it) {
        last = ping();
        if (last == NO_ECHO) {
            it--;
            last = _maxEchoTime;
        }
        uS[i++] = last;
    }
    sort(uS, uS + it);
    return uS[it/2];
}
```

```

    } else {
        if (i > 0) {
            for (j = i; j > 0 && uS[j - 1] < last; j--)
                uS[j] = uS[j - 1];
            } else j = 0;
            uS[j] = last;
            i++;
        }
        if (i < it) delay(PING_MEDIAN_DELAY - (last >> 10));
    }
    return (uS[it >> 1]);
}

```

Ako krenemo redom, vidimo da funkcija prima nepredznačeni cijeli broj, što apsolutno ima smisla jer ne može biti negativan broj očitavanja. Nakon toga se šalje ping. U slučaju da je ping izvan dometa preskoči se to očitavanje i postavlja posljednje očitavanje na maksimalno vrijeme koliko treba da se eho vrati za zadanu udaljenost. U suprotnom, ako je očitana vrijednost unutar zadane udaljenosti, provjerava se da li je prvo očitavanje. Ako nije, sortira se niz očitavanja po *insertion sort* metodi. Postavi se očitana vrijednost na korektnu poziciju za *insertion sort*. Dodaje se posljednji ping u niz na sortiranu poziciju i ide dalje na sljedeći ping. Napravi se mala pauza do sljedećeg očitavanja kako bi se svi ultrazvučni valovi vratili, da ne bi došlo do interferencije. Na kraju uz pomoć binarnog *shift* operatora pomjeri se binarna reprezentaciju broja očitavanja desno za jedan bit što daje lokaciju srednjeg člana niza i iz funkcije se vraća očitana udaljenost sa te lokacije.

U okviru prethodnog koda koristi se funkcija ping(), čija je implementacija data u nastavku:

```

unsigned int NewPing::ping() {
    if (!ping_trigger()) return NO_ECHO;
    while (*_echoInput & _echoBit)
        if (micros() > _max_time) return NO_ECHO;
    return (micros() - (_max_time - _maxEchoTime) - 5);
}

```

Vidimo na početku da se provjerava da li je uopšte iniciran (trigerovan) ultrazvučni val. Ako nije, vraća se NO_ECHO vrijednost. Ako je sve uredno prošlo s te strane, provjerava se da li postoji neki eho tako što se ispituje _echoBit u _echoInput registru. Za to vrijeme se u pozadini mjeri vrijeme koje protiče i ako je proteklo vrijeme veće od maksimalnog, vrati da nema eha u suprotnom izračunaj ping po datoj funkciji.

3.3.2. Arduino kod

Prilikom demonstracije NewPing biblioteke, prikazan je samo način očitavanja sa ping_median funkcijom, ali s obzirom da mi imamo 3 senzora to malo drugačije izgleda. Da bi došli do tog dijela prethodno je potrebno definisati neophodne **varijable**.

```
#define SONAR_NUM      3 // Broj senzora
#define MAX_DISTANCE 250 // Maksimalna udaljenost (u cm) koju senzor doseže
#define PING_INTERVAL 33 // Milisekundi između svakog prozivanja senzora.

unsigned long pingTimer[SONAR_NUM]; // Niz vremena kad bi se prozivanje za
svaki senzor trebalo desiti
unsigned int cm[SONAR_NUM];          // Niz pinganih udaljenosti
uint8_t currentSensor = 0;           // Prati aktivni senzor

String sides [5] = {"l ", "c ", "r ", "a ", "b "};
int value = 999; //udaljenost očitana na nekom od senzora
String side = "x "; //inicijalna oznaka strane

int globalCounter = 9; //globalni brojač "prozvanog" senzora

NewPing sonar[SONAR_NUM] = { // Niz senzora
    NewPing(12, 13, MAX_DISTANCE), //Lijevi
    NewPing(8, 9, MAX_DISTANCE), //Srednji
    NewPing(4, 5, MAX_DISTANCE) //Desni
};
```

Sve deklaracije su objašnjene u liniji gdje su i napisane, osim niza od 5 stringovnih elemenata pod nazivom „side“. On predstavlja oznaku senzora koji se u tom trenutku očitava i vidjet ćemo u nastavku kako se ovaj niz koristi. Cjelobrojna vrijednost „value“ je postavljena na 999 jer tu vrijednost nije moguće očitati sa senzora. Isto važi za vrijednost osnovne strane „side“, koje nema u nizu strana (na ovaj način imamo mogućnost detekcije grešaka). S obzirom da su samo 3 senzora, „globalCounter“ je postavljen na 9 da se i na tom mjestu može detektovati greška.

Naredni korak je implementacija zadane **setup()** funkcije koja se pokreće samo jednom, na početku programa:

```
void setup() {
    Serial.begin(9600); // Iniciraj serijsku komunikaciju
    pingTimer[0] = millis() + 75; // Prvi ping počinje na 75 ms
    for (uint8_t i = 1; i < SONAR_NUM; i++) // Postavi početke za sve senzore
        pingTimer[i] = pingTimer[i - 1] + PING_INTERVAL;
}
```

Nakon toga slijedi **loop()** funkcija. Unutar nje je kod koji proziva svaki senzor pojedinačno, provjerava da li taj vremenski trenutak odgovara senzoru koji se proziva (vrijeme se mjeri od pokretanja programa) i ako odgovara postavlja se trenutak kad će biti moguće prozvati naredni senzor. Vršiti se provjera da li je završen jedan ciklus pinganja i ako jeste poziva se funkcija za slanje rezultata. Ako nije, poziva se funkcija ping_median nad tim senzorom i vrši se 10 očitavanja. Rezultat ping_median funkcije se spašava u globalni niz udaljenosti na mjesto koje odgovara trenutnom senzoru. Kod je dat u nastavku:

```

void loop() {
    for (uint8_t i = 0; i < SONAR_NUM; i++) { // Prođi kroz sve senzore
        if (millis() >= pingTimer[i]) {        // Da li je ovo senzor koji se
        treba pingat
            pingTimer[i] += PING_INTERVAL * SONAR_NUM; // Postavi sljedeći
            senzor koji treba pingat
            if (i == 0 && currentSensor == SONAR_NUM - 1) sendResults(); // Jedan
            ciklus pinganja gotov, pozovi funkciju za slanje najmanjeg
            currentSensor = i;
            cm[currentSensor] = sonar[currentSensor].ping_median(10) /
            US_ROUNDTRIP_CM;
        }
    }
}

```

Funkcija **sendResults()** služi da pozove 3 funkcije čiji su zadaci pronalaženje najmanje udaljenosti u nizu udaljenosti, provjera da li se udaljenosti senzora sa lijeve ili desne strane od „najmanjeg“ senzora razlikuju za manje od 7 cm i slanje putem serijske komunikacije.

```

void sendResults()
{
    findSmallest();

    checkNeighbours();

    printToSerial();
}

```

findSmallest() pretražuje niz očitanih vrijednosti i traži najmanju. Postavlja globalne varijable **value**, **side** i **globalCounter** na odgovarajuće vrijednosti. Side odgovara senzoru sa kojeg je očitana ta najkraća udaljenost, value njegovoj vrijednosti, a globalCounter služi za narednu funkciju pod nazivom **checkNeighbours()** koja provjerava „komšije“ najmanjeg senzora. Ako je razlika u udaljenosti između najmanjeg člana niza i njegovog susjeda manja od 7 cm, to znači da se neki veći predmet približava i neophodno je pustiti zvuk koji odgovara položaju između ta dva senzora, tako da se postavlja strana na slovo „a“ iz niza strana ili slovo „b“ što interno označava lijevi i centralni senzor ili desni i centralni senzor.

```

void checkNeighbours()
{
    if (globalCounter == 0 && abs(cm[1] - value) <= 7 || globalCounter == 1
    && abs(cm[0] - value) <= 7)
    {
        side = sides[3];
    }
    else if (globalCounter == 1 && abs(cm[2] - value) <= 7 || globalCounter
    == 2 && abs(cm[1] - value) <= 7)
    {
        side = sides[4];
    }
}

```

Nakon što smo našli vrijednost koju planiramo poslati i senzor sa kojeg je očitana, potrebno je to „spakovati“ u neki prikladan format koji će UWP aplikacija moći očitati. Obzirom da je najveća moguća udaljenost koju senzor može očitati 250 cm, što je trocifreni broj, dvocifrenim i jednocifrenim očitanjima se dodaju jedan i dva karaktera respektivno kako bi uvijek bio isti broj bajta. Tako da u zbiru imamo: strana + prazno mjesto + 3 karaktera, što ukupno daje 5 bajta u svakoj poruci. U slučaju da ima greška pri mjerenju, šalju se podrazumijevane vrijednosti. Podrazumijevano slovo nije iz skupa strana i ono se neće tumačiti. Taj dio će biti objašnjen u sklopu UWP aplikacije. Serial.print funkcija šalje podatke putem Bluetooth komunikacije obzirom da je BT modul podešen na 9600 *baud rate*, isto kao i serijski port. Kod za funkciju **printToSerial()** je dat u nastavku:

```
void printToSerial()
{
    if (value <= 99 && value >= 10)
    {
        Serial.print(side + value + "x");
        restartValues();
    }
    else if (value <= 250 && value >= 100)
    {
        Serial.print(side + value);
        restartValues();
    }
    else if (value <= 9 && value >= 1)
    {
        Serial.print(side + value + "xx");
        restartValues();
    }
}
```

Funkcija **restartValues()** samo vraća globalne varijable na inicijalne vrijednosti i sprema za novi krug očitavanja:

```
void restartValues()
{
    value = 999;
    side = "x ";
    globalCounter = 9;
}
```

3.4. UWP aplikacija

UWP aplikacija je dosta obimnija, tako da će biti priloženi samo najbitniji dijelovi koda za razumijevanje koncepta aplikacije.

3.4.1. UWP kod

UWP aplikaciju možemo podijeliti na 3 osnovna dijela. Dio koji koristi Cortana personalnog asistenta, dio koji implementira grafički korisnički interfejs i kod u pozadini.

- **Cortana**

Sljedeći kod se nalazi unutar zasebnog **.xml** fajla pod nazivom „FeelTheSpaceVoice.xml“ i podešeno je da se prilikom instalacije softvera na Windows 10 uređaju ovaj set komandi učitava u već postojeći set komandi koje Cortana prepoznaje.

```
<VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.2">
  <CommandSet xml:lang="en-us" Name="AndroidApp_en-us">
    <CommandPrefix> feel </CommandPrefix>
    <Example> Start navigating/navigation </Example>
    <Command Name="feelSpace">
      <Example> Start navigation </Example>
      <ListenFor> {option} {action} </ListenFor>
      <Feedback> feel the space around you </Feedback>
      <Navigate/>
    </Command>

    <PhraseList Label="option">
      <Item> start </Item>
      <Item> stop </Item>
    </PhraseList>
    <PhraseList Label="action">
      <Item> navigation </Item>
      <Item> navigating </Item>
      <Item> guiding </Item>
    </PhraseList>
  </CommandSet>
</VoiceCommands>
```

Vidimo da je definisan link s kojeg se pakuje postavke za glasovne komande, nakon čega se definiše jezik komandi. Prethodno sam spomenuo da postoji uvodna riječ i da je ona u našem slučaju „feel“. Tag „CommandPrefix“ predstavlja uvodnu riječ, kao što samo ime kaže. Nakon toga je dat primjer i definisana jedna konkretna komanda pod nazivom „feelSpace“. Nju čine dvije varijable: **option** i **action**. Vidimo da je *feedback* ili odgovor od strane kompjutera definisan unutar taga „Feedback“. Fraze koje se mogu upotrijebiti umjesto *option* varijable su date kao „Item“ tagovi unutar vanjskog „PhraseList“ taga. Slično je i za varijablu *action*. Postoji dio koda koji ugrađuje ovaj xml fajl u predefinisanu listu komandi, ali zbog njegove strukture i složenosti neće biti predstavljen u ovom radu.

- **Korisnički interfejs**

Korisnički interfejs se definiše unutar **MainPage.xaml** fajla čiji sadržaj je dat u nastavku:

```

    <Grid x:Name="Layout" Background="{ThemeResource
ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="2*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>

        <RelativePanel Grid.Row="0" HorizontalAlignment="Center"
VerticalAlignment="Center">
            <Image Source="./Assets/FeelTheSpaceLogo.jpg" />
        </RelativePanel>
        <RelativePanel Grid.Row="1" HorizontalAlignment="Center"
VerticalAlignment="Center">
            <TextBlock TextAlignment="Center" TextWrapping="Wrap" Text="Završni rad
Prvog ciklusa studija &#x0a; Student: Timur Ćerimagić "/>
        </RelativePanel>

        <MediaElement Name="CheckBluetooth" Source="Assets/CheckBluetooth.mp3"
Volume="1" AutoPlay="False"/>
        <MediaElement Name="C1_beep" Source="Assets/Center/C1_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="C2_beep" Source="Assets/Center/C2_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="C3_beep" Source="Assets/Center/C3_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="L1_beep" Source="Assets/Left/L1_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="L2_beep" Source="Assets/Left/L2_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="L3_beep" Source="Assets/Left/L3_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="R1_beep" Source="Assets/Right/R1_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="R2_beep" Source="Assets/Right/R2_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="R3_beep" Source="Assets/Right/R3_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="RC1_beep" Source="Assets/RC/RC1_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="RC2_beep" Source="Assets/RC/RC2_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="RC3_beep" Source="Assets/RC/RC3_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="LC1_beep" Source="Assets/LC/LC1_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="LC2_beep" Source="Assets/LC/LC2_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="LC3_beep" Source="Assets/LC/LC3_beep.mp3" Volume="1"
AutoPlay="False"/>
        <MediaElement Name="left_intro" Source="Assets/Intro/left_intro.mp3"
Volume="1" AutoPlay="False" Balance="-1"/>
        <MediaElement Name="right_intro" Source="Assets/Intro/right_intro.mp3"
Volume="1" AutoPlay="False" Balance="1"/>
    </Grid>

```

Tag „Grid.RowDefinitions“ definiše okvir mreže i onda smo sa dva taga „RowDefinition“ čije su visine 2^* i 1^* postigli omjer 2:1, što znači da će 2/3 ekrana zauzimati jedan red i 1/3 drugi red. „RelativePanel“ postavlja podlogu koja se prilagođava različitim veličinama ekrana i sa „Grid.Row“ odredimo u koji red smještamo tu podlogu. Vidimo da smo u prvi red, koji je 2 puta veći od drugog, smjestili sliku koja će se povećavati i smanjivati ovisno o veličini ekrana i korisnikovom širenju prozora, a u drugi textBlock. Ostatak koda učitava snimljene zvukove i deklarira objekte tipa MediaElement koji su spremni za upotrebu.

- **Pozadinski kod**

Nije baš dobra praksa, ali za potrebe ovog rada sav kod se nalazi u pozadini MainPage.xaml u klasi koja se naziva **MainPage.xaml.cs**. Cijeli proces možemo podijeliti u nekoliko koraka. Prvo se puštaju lijevi i desni zvuk, nakon čega se napravi spisak svih dostupnih konekcija, nakon čega pokušavamo uspostaviti komunikaciju s našim Bluetooth modulom. Ako je komunikacija uspješna započinje osluškivanje dolaznih informacija. Po prijemu se parsiraju udaljenosti i puštaju zvukovi koji im odgovaraju.

Dakle, prva stvar na listi je puštanje inicijalnih zvukova kako bi korisnik znao da li je pravilno stavio slušalice.

```
private async void InitializeSounds()
{
    await PlayIntro();
}

public async Task PlayIntro()
{
    await Task.Delay(2000);
    right_intro.Play();
    while (right_intro.CurrentState == MediaElementState.Playing)
        continue;
    await Task.Delay(1000);
    left_intro.Play();

    await Task.Delay(1500);
}
```

Asinhrona metoda „InitializeSounds“ se poziva iz konstruktora i poziva „PlayIntro“ asinhronu metodu koju čeka da završi da bi dalje nastavila sa radom. Unutar PlayIntro metode imamo početnu pauzu čisto da se aplikacija stabilizuje, puštanje lijevog zvuka, provjeru da li je završio desni zvuk, ako jeste pusti lijevi, a ako nije čekaj pomoću *while* petlje.

Nakon toga imamo funkciju koja pretražuje sve bluetooth module i stavlja ih u listu potencijalnih modula za uparivanje. [34]

```

private async void InitializeRfcommDeviceService()
{
    await Task.Delay(3000);
    try
    {
        DeviceInformationCollection DeviceInfoCollection = await
DeviceInformation.FindAllAsync(RfcommDeviceService.GetDeviceSelector(RfcommServiceId.SerialPort));

        var numDevices = DeviceInfoCollection.Count();

        _pairedDevices = new ObservableCollection<PairedDeviceInfo>();
        _pairedDevices.Clear();

        if (numDevices == 0)
        {
            MessageDialog md = new
MessageDialog("InitializeRfcommDeviceService: No paired devices found.", "No
devices");
        }
        else
        {
            foreach (var deviceInfo in DeviceInfoCollection)
            {
                _pairedDevices.Add(new PairedDeviceInfo(deviceInfo));
            }
        }
        PairedDevices.Source = _pairedDevices;

        InitializeHC05Connection();
    }
    catch (Exception ex)
    {
        Debug.WriteLine("InitializeRfcommDeviceService: " + ex.Message);
    }
}

```

Prvo se pronalaze svi uređaji koji su dostupni putem Bluetooth-a. Ako nema takvih uređaja prikaži poruku, u suprotnom sve nađene uređaje dodaj u kolekciju uparenih uređaja. Poslije toga se poziva funkcija „InitializeHC05Connection“ koja se povezuje s našim modulom. Njen kod je dat u nastavku: [34]

```

private async void InitializeHC05Connection()
{
    DeviceInformation DeviceInfo = null;
    PairedDeviceInfo pairedDevice = null;

    //Prodji kroz uparene uredjaje i probaj naći HC-05 bluetooth modul
    for (int i = 0; i < _pairedDevices.Count(); i++)
    {
        if (_pairedDevices[i].Name == "HC-05")
        {
            pairedDevice = _pairedDevices[i];
            DeviceInfo = pairedDevice.DeviceInfo;
        }
    }
}

```

```

        break;
    }
}

bool success = true;
try
{
    //Kreiraj servis koristeći ID HC-05 modula (ako je uspješno nadjen)
    _service = await RfcommDeviceService.FromIdAsync(DeviceInfo.Id);

    if (_socket != null)
    {
        // Ako već postoji nekih stvari u socket-u, prvo očisti sve resurse
        _socket.Dispose();
    }

    _socket = new StreamSocket();
    try
    {
        // Ako je bilo koji od dva parametra null ili prazan, ovaj poziv
        ce baciti izuzetak
        await _socket.ConnectAsync(_service.ConnectionHostName,
        _service.ConnectionServiceName);
    }
    catch (Exception ex)
    {
        success = false;
        BluetoothConnection(); // U slučaju greške pozovi metodu koja
        reprodukuje zvuk Bluetooth greške
    }
    if (success)
    {
        //Metoda koja aktivira slušanje
        Listen();
    }
}
catch (NullReferenceException ex)
{
    BluetoothConnection();
}
catch (Exception ex)
{
    _socket.Dispose();
    _socket = null;
}
}

```

Ovu funkciju je lakše bilo objasniti direktno pored koda, što je i urađeno. Vidimo da ako je uspješno obavljena konekcija imamo funkciju „Listen“ koja počinje osluškivati podatke koji dolaze putem Bluetooth *stream-a*. Njen zadatak je da kreira objekat tipa *DataReader* i pozove funkciju „ReadAsync“ koja čita podatke iz *stream-a* podataka. [34]

```

private async Task ReadAsync(CancellationTokencancellationTokencancellationToken)
{
    Task<UInt32> loadAsyncTask;

    // Veličina buffer-a 5 bajta
    uint ReadBufferLength = 5;

    // Ako je poslan zahtjev za prekid, prekini
    cancellationToken.ThrowIfCancellationRequested();

    // Postavi InputStreamOptions da asihnrone čitaju kad je jedan ili više
    bajta dostupno
    dataReaderObject.InputStreamOptions = InputStreamOptions.ReadAhead;

    // Kreiraj task objekat da čeka podatke na serijskom portu
    loadAsyncTask =
dataReaderObject.LoadAsync(ReadBufferLength).AsTask(cancellationToken);

    // Pokreni task i čekaj
    UInt32 bytesRead = await loadAsyncTask;

    string[] value;
    char side;

    if (bytesRead > 0) // Ako ima pročitanih bajta
    {
        try
        { // Pročitane bajte iz dataReader objekta pretvori u string i
        dodijeli varijabli tipa string
            string recvdtxt = dataReaderObject.ReadString(bytesRead);

            // Primjena Regex-a za parsiranje vrijednosti
            value = Regex.Split(recvdtxt, @"\D+");

            side = recvdtxt[0]; // Na prvom mjestu je uvijek slovo koje
            označava stranu
            await playSound(side, value[1]); // Pusti zvuk koji odgovara
            strani i udaljenosti
        }
        catch (Exception ex)
        {
            Debug.WriteLine("ReadAsync: " + ex.Message);
        }
    }
}

```

I posljednja funkcija koja je bitna za ovu aplikaciju je „playSound“ funkcija koja pušta odgovarajuće zvukove. S obzirom da je repetativna, biće priložen samo jedan njen dio, ostatak se svodi na provjeravanje isti uslova sa drugim vrijednostima.

```
private async Task playSound(char side, string v)
{
    if (C1_beep.CurrentState == MediaElementState.Playing ||
        C2_beep.CurrentState == MediaElementState.Playing ||
        C3_beep.CurrentState == MediaElementState.Playing ||
        R1_beep.CurrentState == MediaElementState.Playing
// Provjera da li su trenutno pušteni ide za sve zvukove iz liste, kako se neki novi
// zvuk ne bi pustio dok prethodni ne završi
    )
        return;

// Ako je strana 'l' -lijeva i ako je udaljenost manja od 70 pusti lijevi zvuk 1,
// između 70 i 120 cm pusti zvuk 2 i u suprotnom pusti zvuk 3... Za sve ostale moguće
// strane je isti kod
    if (side == 'l')
    {
        if (Convert.ToInt16(v) <= 70)
            L1_beep.Play();
        else if (Convert.ToInt16(v) > 70 && Convert.ToInt16(v) <= 120)
            L2_beep.Play();
        else
            L3_beep.Play();
    }
    else if (side == 'c')
    {
        if (Convert.ToInt16(v) <= 70)
            C1_beep.Play();
        else if (Convert.ToInt16(v) > 70 && Convert.ToInt16(v) <= 120)
            C2_beep.Play();
        else
            C3_beep.Play();
    }
}
```

Rezultat je puštanje predefinisano odgovarajućeg zvuka za svaku poslatu poruku putem Bluetooth komunikacije.

4. Zaključak

Osnovni cilj rada je bio istraživanje i upoznavanje mogućnosti kako Arduino platforme, tako i UWP platforme. Potrebno je bilo pronaći adekvatan način uvezivanja ova dva individualna i neovisna sistema, te pronaći optimalan model prikaza rezultata. Pri samoj izradi sistema bila su potrebna osnovna znanja iz ugradbenih sistema, univerzalne Windows platforme, kao i općenito iz elektronike i kodiranja. Cilj je bio odabrati optimalan hardver, isti povezati, kalibrirati i napisati potreban kod na osnovu kojeg će Arduino mikrokontroler prikupljati informacije sa senzora i obrađivati ih. Pored Arduino dijela, bilo je potrebno napisati UWP aplikaciju koja će iskoristiti potencijal Bluetooth tehnologije u svrhe dovoljno učestalog uzorkovanja, interpretacije i upotrebe interpretiranih vrijednosti za reprodukciju 3D zvukova. Mogućnosti Arduino i UWP platformi su zaista obimne i u ovom radu je prikazan samo jedan mali aspekt primjene. Krajnji produkt rada je autonomni sistem za pomoć slijepim i slabovidnim osobama koji bi uz određene modifikacije mogao imati komercijalnu vrijednost, a uz to doprinijeti društvenoj zajednici.

Glavne prednosti Arduino platforme su njena cijena i prisupačnost. Intuitivan korisnički interfejs doprinosi jednostavnosti i lakoći učenja i razvoja rješenja. Na internetu postoji pregršt primjera, shema, savjeta i generalno materijala koji su dovoljno jasni i osobama koje se tek susreću sa ugradbenim sistemima. U biti, bez velike *open-source* zajednice i doprinosa sviju pojedinačno, rad sa printanim pločicama, mikrokontrolerima i ostalim komponentama bi još uvijek bio tabu tema i tema kojom se bave samo „oni što prave robote“.

Prednost UWP aplikacije se ogleda u njenoj univerzalnosti. Pored toga, činjenica da u okviru ove platforme postoji Cortana personalni asistent je značajno suzila izbor. Uz pomoć tog benefita sam uspješno zamijenio grafički korisnički interfejs sa „audio“ korisničkim interfejsom koji je dosta pogodniji za ciljanu skupinu društva.

Inicijalni pristup rješavanju problema je uključivao slanje svih očitanih vrijednosti na UWP aplikaciju i tek nakon toga filtriranje, obradu i reprodukciju zvukova, što se pokazalo vrlo neefikasno. S obzirom da se na taj način gubilo dragocjeno vrijeme na slanje potencijalno neiskoristivih podataka i s obzirom da format poslanih vrijednosti nije bio unificiran, pristup je promijenjen. Prvo je bilo potrebno smanjiti protok podataka kroz Bluetooth uređaj na način da se iskoristi potencijal Arduino mikrokontrolera za filtriranje osim samog prikupljanja podataka, a drugo, bilo je neophodno predefinisati tačnu veličinu spremnika podataka i njihov format, kako bi ih mogli uspješno parsirati na UWP strani. Prethodno navedeno bi bila najveća poteškoća koju sam imao prilikom izrade rada.

U budućnosti je planirano pravljenje dvije verzije ovog sistema. Razlika između verzija bi se očitovala isključivo u izboru senzora. Jedan od motiva koji je podstakao ovu ideju je nedovoljno iskorišten potencijal Cortana-e i samim tim je planirana njena bolja integracija i implementacija intuitivnijeg seta komandi, kako bi FeelTheSpace aplikacija stvarno omogućila osjećaj prostora onima kojima je to najpotrebnije. Vizija je da se senzori integrišu u odjeću u smislu da zamjenjuju običnu dugmad na košulji, da se interna baterija može puniti pomoću solarnih ćelija i da se ne modificira sam Arduino uređaj kako bi bio kompaktniji.

5. Bibliografija

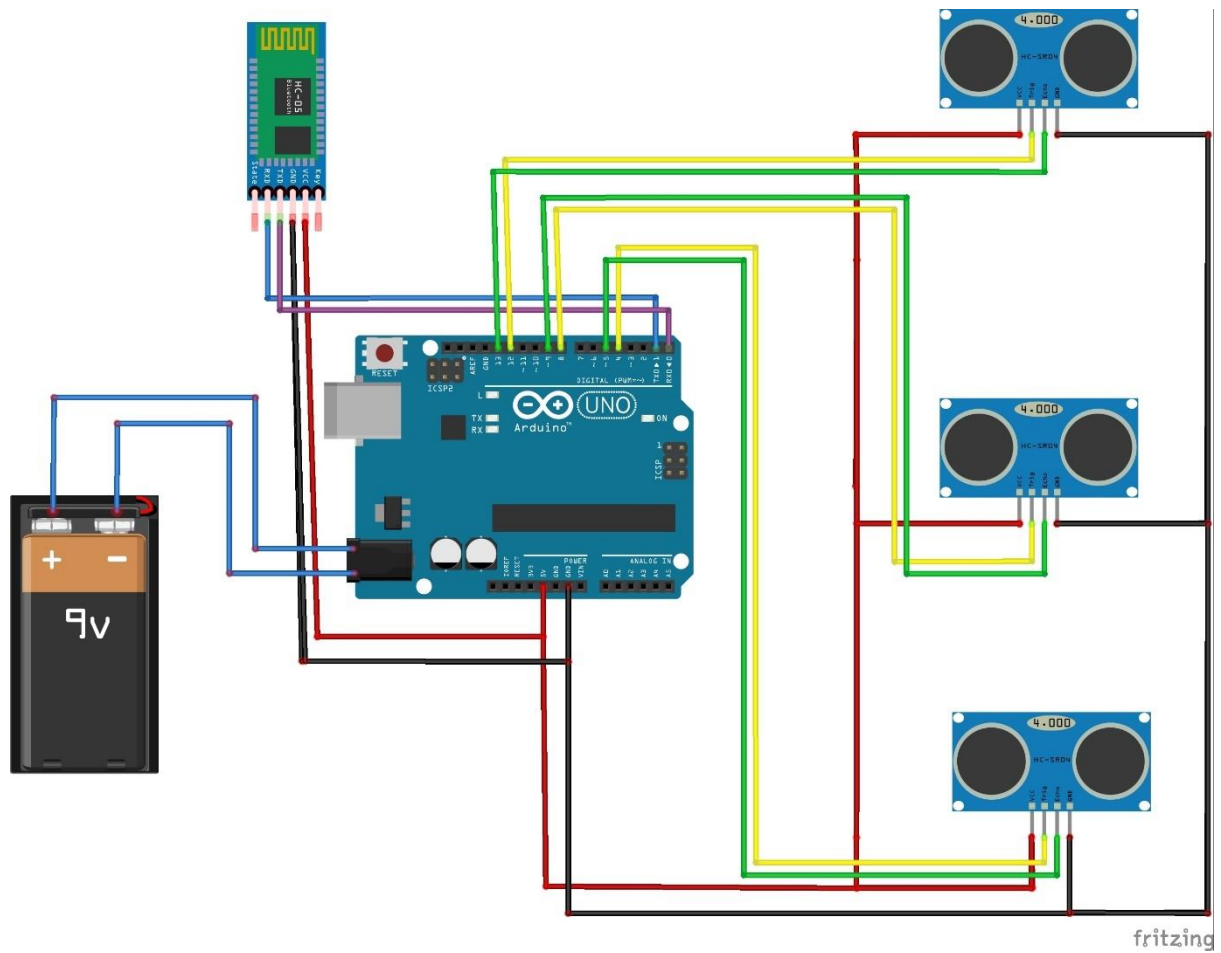
- [1] »Arduino,« 2016. [Mrežno]. Available: <https://www.arduino.cc/en/Guide/Introduction>. [Pokušaj pristupa 15 Juni 2016].
- [2] »Wikipedia,« [Mrežno]. Available: <https://en.wikipedia.org/wiki/Arduino>. [Pokušaj pristupa 15 Juni 2016].
- [3] B_E_N, »sparkfun,« [Mrežno]. Available: <https://learn.sparkfun.com/tutorials/what-is-an-arduino>. [Pokušaj pristupa 15 Juni 2016].
- [4] [Mrežno]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/8/87/Arduino_Logo.svg/1280px-Arduino_Logo.svg.png. [Pokušaj pristupa 15 Juni 2016].
- [5] D. Kushner, »ieee spectrum,« [Mrežno]. Available: <http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino>. [Pokušaj pristupa 15 Juni 2016].
- [6] H. Barragán, »arduinohistory,« [Mrežno]. Available: <https://arduinohistory.github.io/>. [Pokušaj pristupa 15 Juni 2016].
- [7] John, »circuitstoday,« [Mrežno]. Available: <http://www.circuitstoday.com/story-and-history-of-development-of-arduino>. [Pokušaj pristupa 15 Juni 2016].
- [8] »parallax,« [Mrežno]. Available: <https://www.parallax.com/sites/default/files/styles/full-size-product/public/90005a.png?itok=sNiar2QH>. [Pokušaj pristupa 15 Juni 2016].
- [9] »Arduino,« [Mrežno]. Available: <https://www.arduino.cc/en/Products/Compare>. [Pokušaj pristupa 16 Juni 2016].
- [10] »robotics,« [Mrežno]. Available: https://www.robotics.org.za/image/data/Arduino/Arduino%20Boards/arduino_mega_r3_002_hd.jpg. [Pokušaj pristupa 16 Juni 2016].
- [11] »Arduino,« [Mrežno]. Available: <https://www.arduino.cc/en/Main/ArduinoBoardUno>. [Pokušaj pristupa 16 Juni 2016].
- [12] T. Reddy, »slideshare,« [Mrežno]. Available: <http://www.slideshare.net/ReddyTeja/analog-and-digital-signals-15796948>. [Pokušaj pristupa 10 Juni 2016].
- [13] [Mrežno]. Available: <http://image.slidesharecdn.com/analoganddigitalsignals-121229122042-phpapp01/95/analog-and-digital-signals-6-638.jpg?cb=1356784160>. [Pokušaj pristupa 20 Juni 2016].
- [14] d. e. i. Doc. dr. Željko Jurić, Diskretna matematika, Sarajevo: Elektrotehnički fakultet u Sarajevu, 2011.
- [15] »wikipedia,« [Mrežno]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/5/50/Differential_manchester_en

- coding.svg/600px-Differential_manchester_encoding.svg.png. [Pokušaj pristupa 10 Juni 2016].
- [16] S. Konjicija, »c2.etf.unsa.ba,« 2016. [Mrežno]. Available: http://c2.etf.unsa.ba/file.php/117/Predavanja_2016/Ugradbeni_sistemi-poglavlje_2_digitalniIO.pdf. [Pokušaj pristupa 10 Juni 2016].
- [17] »developer.mbed,« [Mrežno]. Available: https://developer.mbed.org/media/uploads/rorydog1/duty_cycle.png. [Pokušaj pristupa 10 Juni 2016].
- [18] »electrosome,« [Mrežno]. Available: <https://electrosome.com/wp-content/uploads/2014/08/Hc-SR04-Ultrasonic-Sensor.jpg>. [Pokušaj pristupa 5 Juni 2016].
- [19] »csharpcorner,« [Mrežno]. Available: <http://csharpcorner.mindcrackerinc.netdna-cdn.com/UploadFile/167ad2/how-to-use-ultrasonic-sensor-hc-sr04-in-arduino/Images/Hc-SR04.jpg>. [Pokušaj pristupa 5 Juni 2016].
- [20] »instructables,« [Mrežno]. Available: <http://cdn.instructables.com/FE0/DHQ4/HV2AIB01/FE0DHQ4HV2AIB01.MEDIUM.jpg>. [Pokušaj pristupa 5 Juni 2016].
- [21] »Arduino,« [Mrežno]. Available: <http://playground.arduino.cc/Main/DevelopmentTools>. [Pokušaj pristupa 1 Juni 2016].
- [22] J. Madden, »searchmobilecomputing,« [Mrežno]. Available: <http://searchmobilecomputing.techtarget.com/tip/How-Universal-Windows-Platform-apps-work>. [Pokušaj pristupa 20 Maj 2016].
- [23] [Mrežno]. Available: <http://image.slidesharecdn.com/20150723windows10uwp20150723-24-150825040539-lva1-app6891/95/20150723-windows-10-uwp-20150723-24-3-638.jpg?cb=1440475793>. [Pokušaj pristupa 20 Maj 2016].
- [24] K. Gallo, »blogs.windows,« [Mrežno]. Available: <https://blogs.windows.com/buildingapps/2015/03/02/a-first-look-at-the-windows-10-universal-app-platform/>. [Pokušaj pristupa 20 Maj 2016].
- [25] MartinEukan, 14 Apr 2016. [Mrežno]. Available: <https://msdn.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp>. [Pokušaj pristupa 20 Maj 2016].
- [26] M. Jacobs, »msdn,« 4 Maj 2016. [Mrežno]. Available: <https://msdn.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp>. [Pokušaj pristupa 20 Maj 2016].
- [27] S.Somasegar, 29 Juli 2015. [Mrežno]. Available: <https://blogs.msdn.microsoft.com/somasegar/2015/07/29/building-apps-for-windows-10-with-visual-studio-2015/>. [Pokušaj pristupa 20 Maj 2016].
- [28] Z. Čilić, »Korištenje Bluetooth standarda za upravljanje igrom na Arduino sistemu,« Sarajevo, 2015.

- [29] »binauric,« [Mrežno]. Available: <https://www.binauric.com/technology/>. [Pokušaj pristupa 1 Juli 2016].
- [30] »wikipedia,« [Mrežno]. Available: https://en.wikipedia.org/wiki/Binaural_recording. [Pokušaj pristupa 1 Juli 2016].
- [31] [Mrežno]. Available: <http://bldr.org/2011/03/various-proximity-sensors-arduino/>. [Pokušaj pristupa 10 Januar 2016].
- [32] T. Eckel, »bitbucket,« [Mrežno]. Available: <http://bldr.org/2011/03/various-proximity-sensors-arduino/>. [Pokušaj pristupa 1 Juli 2016].
- [33] P. Stoffregen, »github,« [Mrežno]. Available: <https://github.com/PaulStoffregen/NewPing/blob/master/NewPing.cpp>. [Pokušaj pristupa 1 Juli 2016].
- [34] D. Jones, »embedded101,« [Mrežno]. Available: <http://embedded101.com/Blogs/David-Jones/entryid/707/Win-10-lot-Core-Bluetooth-Universal-Windows-Serial-App->. [Pokušaj pristupa 12 Februar 2016].
- [35] [Mrežno]. Available: <http://lilypadarduino.org/wp-content/uploads/lilypadMainBoard.jpg>. [Pokušaj pristupa 16 Juni 2016].
- [36] »zoomaviation,« [Mrežno]. Available: <http://zoomaviation.com/temp/mkr1000.png>. [Pokušaj pristupa 16 Juni 2016].

Dodatak A

A.1. Shema sistema



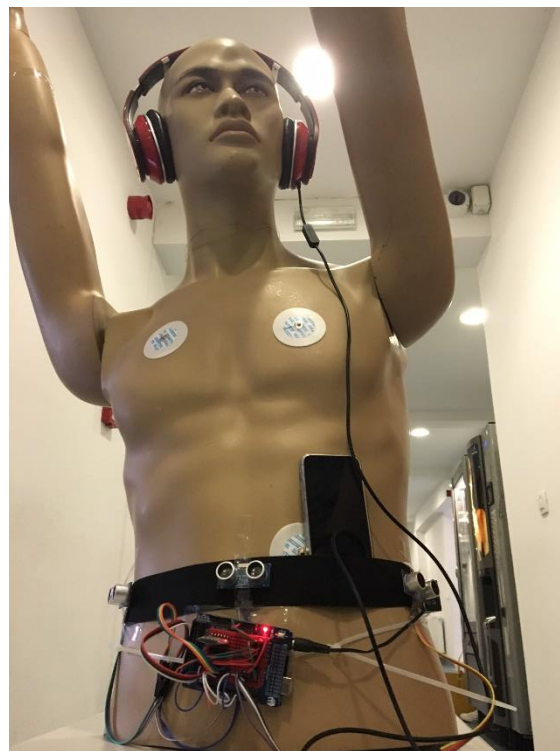
Slika A-1 Shema sistema

Dodatak B

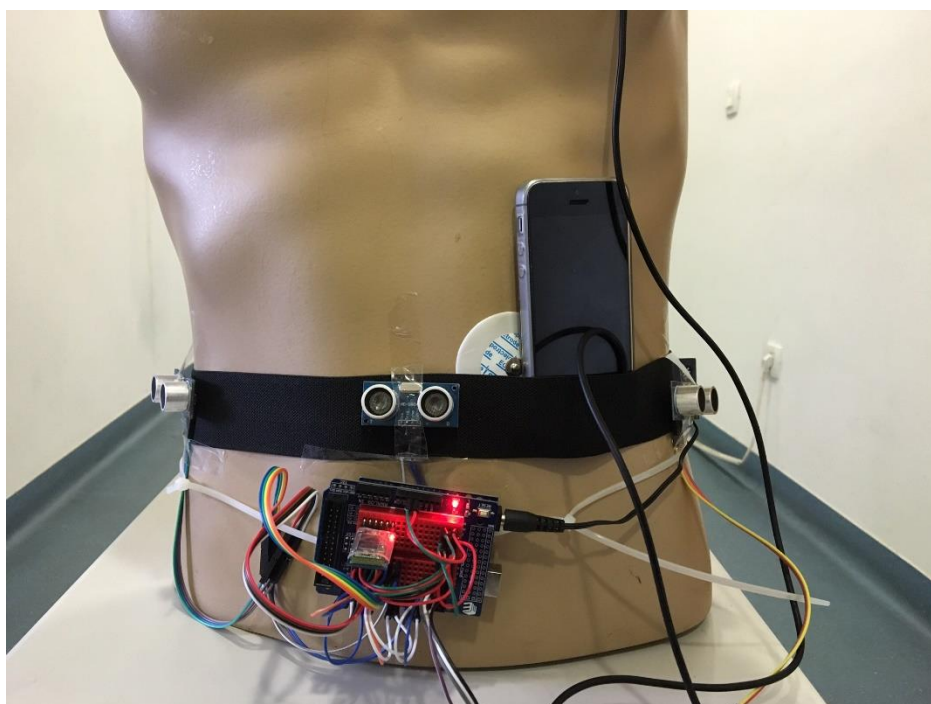
B.1. Slike sistema



Slika B-3 Slušalice



Slika B-1 Sistem u cjelosti



Slika B-2 Pojas sa senzorima i mobilni telefon

Univerzitet u Sarajevu
Naziv fakulteta/akademije _____
Naziv odsjeka i/ili katedre _____
Predmet _____

Izjava o autentičnosti radova

Seminarski rad, završni (diplomski odnosno magistarski) rad za I i II ciklus studija i integrirani studijski program I i II ciklusa studija, magistarski znanstveni rad i doktorska disertacija⁵

Ime i prezime _____
Naslov rada _____
Vrsta rada _____
Broj stranica _____

Potvrđujem:

- da sam pročitao/la dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan/na univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio/la prisustvo citiranog ili parafraziranog materijala i da sam se referirao/la na sve izvore;
- da sam dosljedno naveo/la korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio/la svaku pomoć koju sam dobio/la pored pomoći mentora/ice i akademskih tutora/ica.

Mjesto, datum _____

Potpis _____

⁵ U radu su korišteni slijedeći dokumenti: *Izjava autora* koju koristi Elektrotehnički fakultet u Sarajevu; *Izjava o autentičnosti završnog rada* Centra za interdisciplinarnu studiju – master studij „Evropske studije”; *Izjava o plagijarizmu* koju koristi Fakultet političkih nauka u Sarajevu.