



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

Ovdje ide neki tekst

## Abstract

Here goes some text

## 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 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
- **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
- **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 Application
- **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
2.1.2. Verzije Arduino uređaja .....	7
2.1.3. Arduino Uno .....	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 .....	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 .....	31
2.3. Bluetooth .....	32
2.3.1. Upotreba Bluetooth tehnologije.....	33
2.4. 3D (Binaural) zvuk .....	34
2.4.1. Tehnika snimanja .....	34
2.4.2. Reproduciranje zvuka.....	35
3. Praktični dio .....	35
3.1. Uspostavljanje platforme.....	35
3.2. Funkcionalni opis sistema .....	35
3.3. UML model.....	35
3.4. Arduino aplikacija.....	35
3.4.1. Biblioteke .....	35
3.4.2. Arduino kod.....	35
3.5. UWP aplikacija .....	35
3.5.1. UWP kod .....	35
4. Zaključak.....	35
5. Dodatak.....	35

6. Reference.....	35
-------------------	----

## Spisak slika

Slika 2-1 Arduino logo .....	4
Slika 2-2 Osnivači Arduino-a .....	5
Slika 2-3 BASIC Stamp mikrokontroler .....	6
Slika 2-4 Arduino Mega .....	8
Slika 2-5 Arduino LilyPad .....	9
Slika 2-6 Arduino MKR1000 .....	9
Slika 2-7 Arduino Uno shema .....	10
Slika 2-8 Primjeri analognog signala .....	13
Slika 2-9 Analogne i diskretne vrijednosti .....	14
Slika 2-10 Poruka kodirana jedinicama i nulama .....	14
Slika 2-11 Digitalni izlaz sa par tranzistora .....	15
Slika 2-12 Digitalni ulaz koji možemo direktno povezati sa napajanjem .....	15
Slika 2-13 PWM duty-cycle .....	16
Slika 2-14 HC-SR04 ultrazvučni senzor udaljenosti .....	17
Slika 2-15 Lociranje objekta pomoću ultrazbučnog senzora .....	18
Slika 2-16 LM35 temperaturni senzor .....	18
Slika 2-17 Arduino IDE 1.6.8 .....	21
Slika 2-18 Universal Windows Apps na različitim uređajima .....	25
Slika 2-19 Algoritam skaliranja 24px fonta na različitim uređajima .....	27
Slika 2-20 Repozicija elemenata na UI-u .....	28
Slika 2-21 Promjena veličine elemenata na UI-u .....	28
Slika 2-22 Reflow .....	29
Slika 2-23 Prikazivanje i sakrivanje dijelova UI-a .....	29
Slika 2-24 Zamjena elemenata .....	30
Slika 2-25 Potpuna reorganizacija .....	30
Slika 2-26 Mogući uređaji koji mogu biti emulirani .....	31
Slika 2-27 Pokretanje aplikacije u različitim okruženjima .....	31
Slika 2-28 ITD .....	34
Slika 2-29 ILD .....	34
Slika 2-30 HRTFS .....	34



## Spisak tabela

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

## 1. Uvod

## 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 tih rezultata na izlaze. Na strani ulaza mogu biti različiti senzori, kao na primjer senzora 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.

Sljedeća rečenica predstavlja citat sa [arduino.cc](http://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.”*

#### 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

### 2.1.1. Arduino platforma kroz historiju

Kolumbijski student Hernando Barragán je 2004. godine u svrhe svog magistarskog rada kreirao razvojnu platformu pod nazivom “*Wiring*”<sup>1</sup> 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*<sup>2</sup>). 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, naziv 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



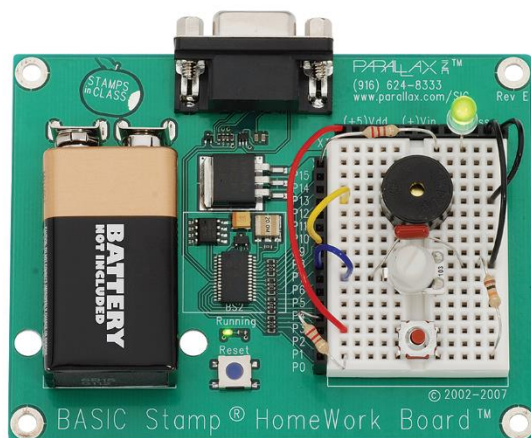
Slika 2-2 Osnivači Arduino-a

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

<sup>1</sup> 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

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

problema u vezi sa tim mikrokontrolerom. Prvo, bio je previše skup za potrebe studenata; koštao je 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

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 ugasisite, 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.

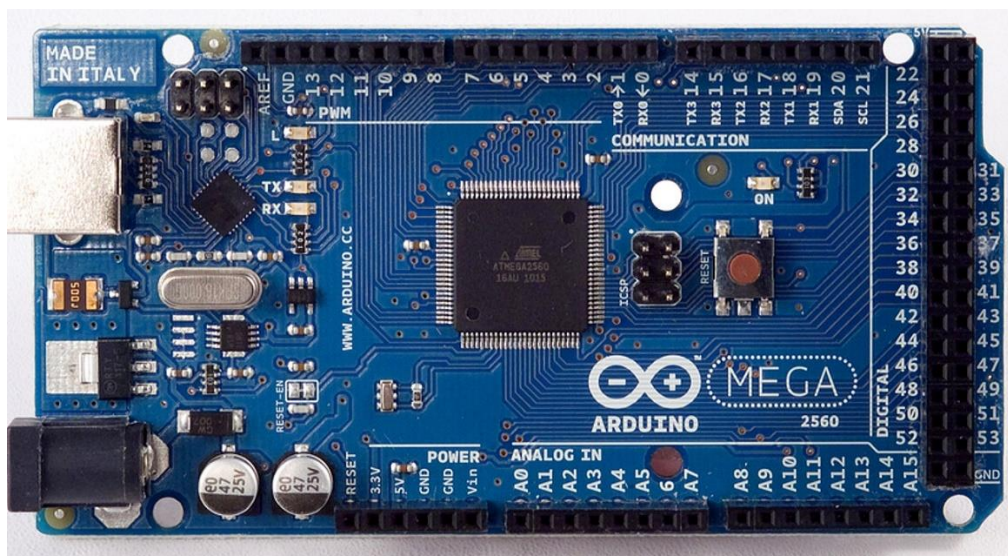
Tabela 2-1 Prikaz aktuelnih Arduino uređaja

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
<b>101</b>	Intel® Curie	3.3 V / 7-12V	32MHz	6/0	14/4	-	24	196	Regular	-
<b>Gemma</b>	ATtiny85	3.3 V / 4-16 V	8 MHz	1/0	3/2	0.5	0.5	8	Micro	0
<b>LilyPad</b>	ATmega168V ATmega328P	2.7-5.5 V / 2.7-5.5 V	8MHz	6/0	14/6	0.512	1	16	-	-
<b>LilyPad SimpleSnap</b>	ATmega328P	2.7-5.5 V / 2.7-5.5 V	8 MHz	4/0	9/4	1	2	32	-	-
<b>LilyPad USB</b>	ATmega32U4	3.3 V / 3.8-5 V	8 MHz	4/0	9/4	1	2.5	32	Micro	-
<b>Mega 2560</b>	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4
<b>Micro</b>	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	Micro	1
<b>MKR1000</b>	SAMD21 Cortex-M0+	3.3 V / 5V	48MHz	7/1	8/4	-	32	256	Micro	1
<b>Pro</b>	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
<b>Pro Mini</b>	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
<b>Uno</b>	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1
<b>Zero</b>	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	-	32	256	2 Micro	2

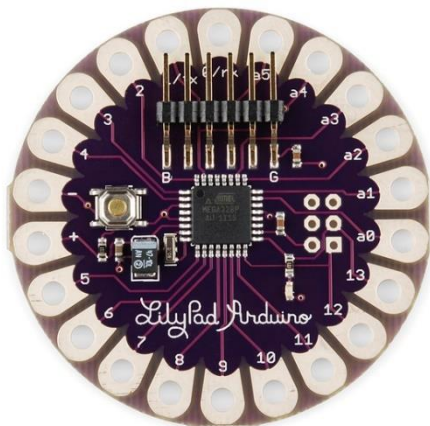


U današnje vrijeme najpopularnije verzije Arduino platforme su Uno, Mega i nekadašnji Yun, koji se više ne proizvodi. Po nazivu ATmega procesora vidimo da preovladavaju 32kB-tni modeli (prva dva broja u nazivu procesora ATmega328P ili ATmega32U4 predstavljaju 32kB integrirane *flash* memorije). Radne 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.

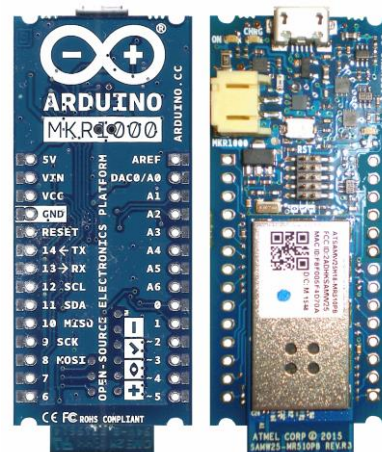
Vidimo da su dizajneri i kreatori Arduino platforme odlučili napraviti cijeli spektar različitih uređaja koji mogu pronaći svoju primjenu u isto tako širokom spektru različitih projekata. Ako je potrebna velika frekvencija procesora, dosta memorije i kompaktan dizajn, ali mali broj pinova, dobro rješenje 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



Slika 2-5 Arduino LilyPad



Slika 2-6 Arduino MKR1000

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

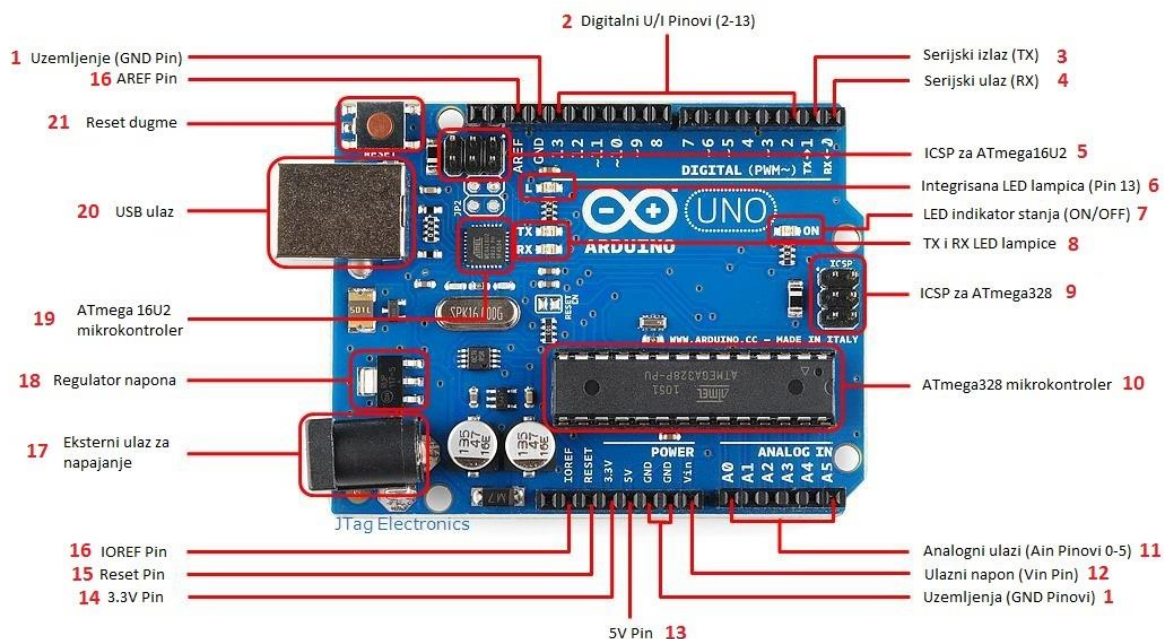
#### 2.1.3.1. Opis

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

<b>Mikrokontroler</b>	ATmega328P
<b>Radni napon</b>	5V
<b>Ulazni napon (preporučeni)</b>	7-12V
<b>Ulazni napon (granični)</b>	6-20V
<b>Digitalnih U/I pinova</b>	14 (od kojih 6 imaju PWM izlaz)
<b>PWM digitalnih U/I pinova</b>	6
<b>Analognih ulaznih pinova</b>	6
<b>Količina istosmjerne struje po U/I pinu</b>	20 mA
<b>Količina istosmjerne struje za 3.3V pin</b>	50 mA
<b>Flash memorija</b>	32 KB (ATmega328P), od koje je 0.5 KB rezervisano za bootloader
<b>SRAM</b>	2 KB (ATmega328P)
<b>EEPROM</b>	1 KB (ATmega328P)
<b>Frekvencija sata</b>	16 MHz
<b>Dužina</b>	68.6 mm
<b>Širina</b>	53.4 mm
<b>Težina</b>	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*-om<sup>3</sup>. Za te svrhe se može koristiti

<sup>3</sup> 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 omogućuje 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 I<sup>2</sup>C komunikacija koja se nalazi na pinovima 4 (SDA) i 5 (SCL) je podržana „Wire“ bibliotekom u Arduino jeziku.

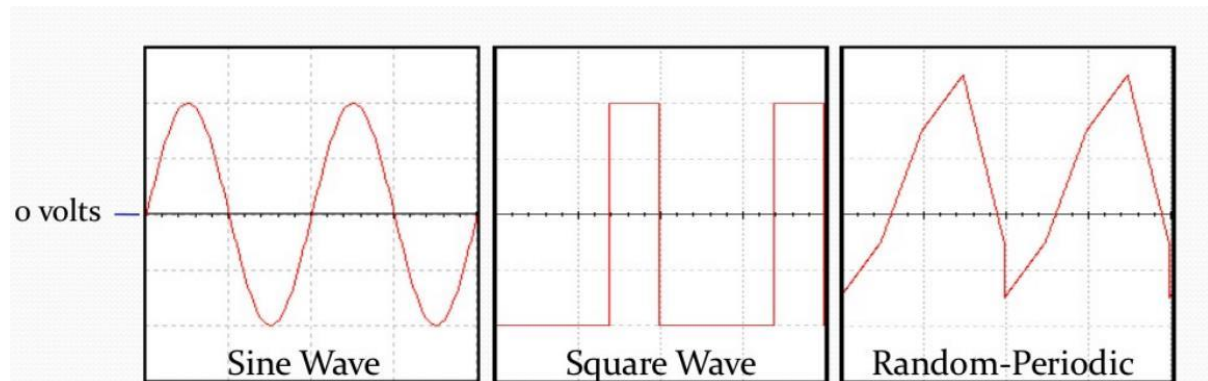
## 2.1.4. Digitalni ulazi/izlazi i analogni ulazi

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

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



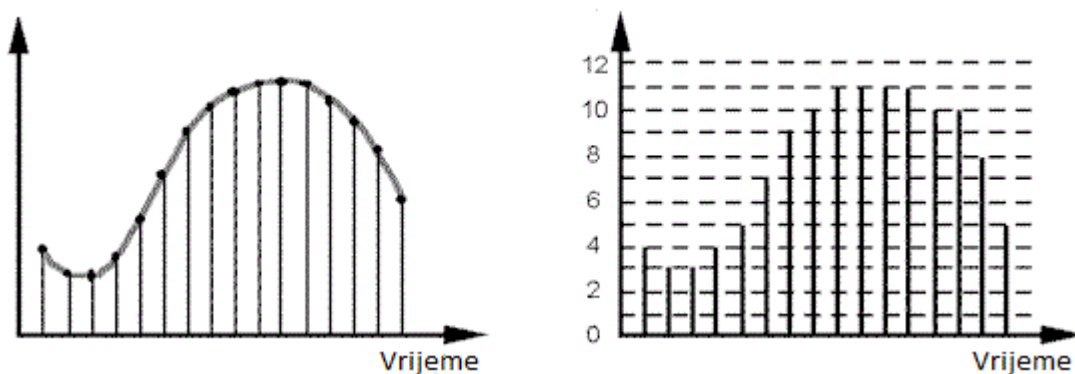
Slika 2-8 Primjeri analognog signala

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.

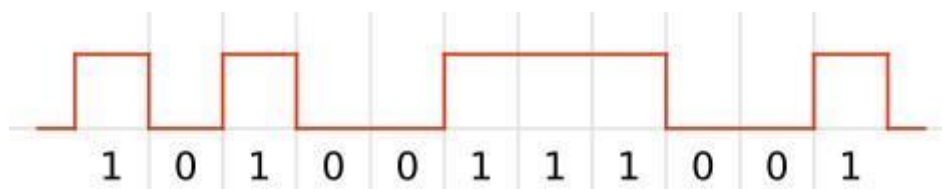
### 2.1.4.2. Digitalni signal i digitalni ulazi/izlazi

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:



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:



Slika 2-10 Poruka kodirana jedinicama i nulama

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:

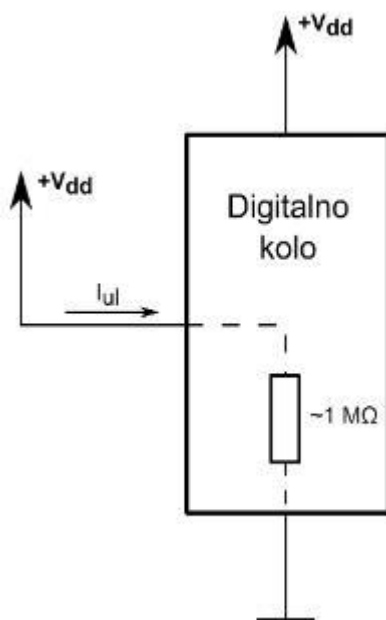
$$I_{ul} = \frac{5V}{1M\Omega} = 5\mu A$$

koja proizvodi snagu:

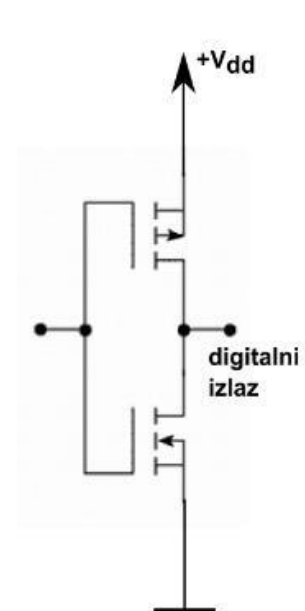
$$P = U * I_{ul} = 25\mu W$$

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

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  $\Omega$ ). 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.



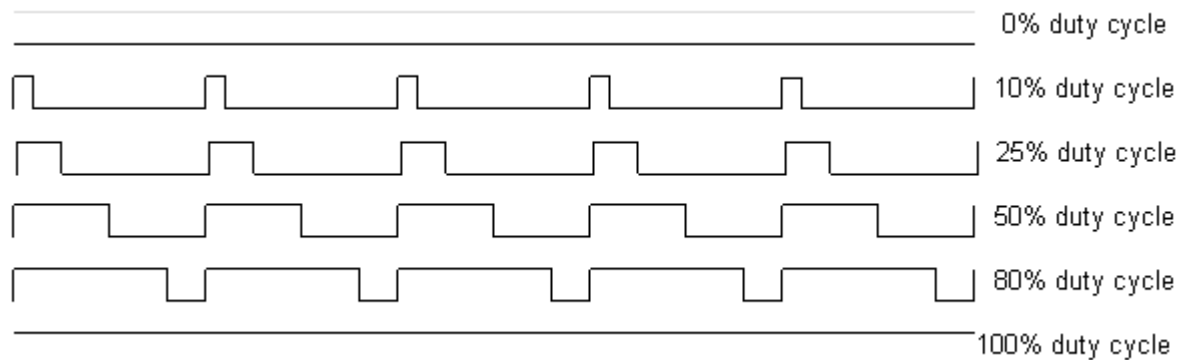
Slika 2-12 Digitalni ulaz koji možemo direktno povezati sa napajanjem



Slika 2-11 Digitalni izlaz sa par tranzistora

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*

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

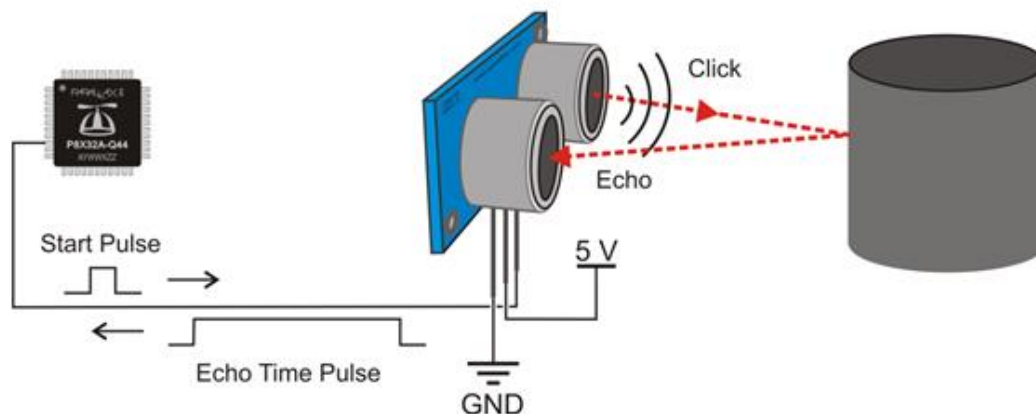
Postoji jako veliki broj senzora koji nakon procesiranja i obavljanja posla za koji su napravljeni šalju digitalne signale. Primjeri takvih senzora su senzor vatre, DHT11 digitalni senzor temperature i vlažnosti zraka, RTC (*Real Time Clock*) ili sistemski sat, digitalni senzor dodira i još mnogo drugih.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



Slika 2-14 HC-SR04 ultrazvučni senzor udaljenosti

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 osluškivati 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

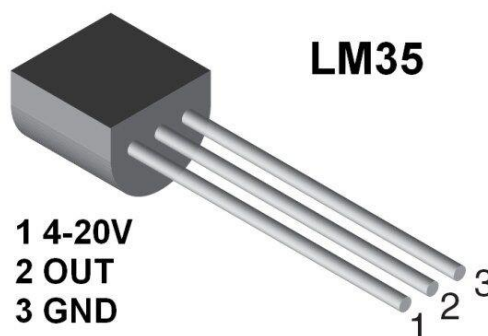
Na Internetu postoje biblioteke koje nam olakšavaju mjerenje udaljenosti na način da su već implementirane funkcije koje aktiviraju slanje i primanje signala. Na nama je samo da ih pozovemo i njihov rezultat dodijelimo nekoj varijabli. Jedna takva biblioteka je **NewPing**. Više o njoj u praktičnom dijelu. 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

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

*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:

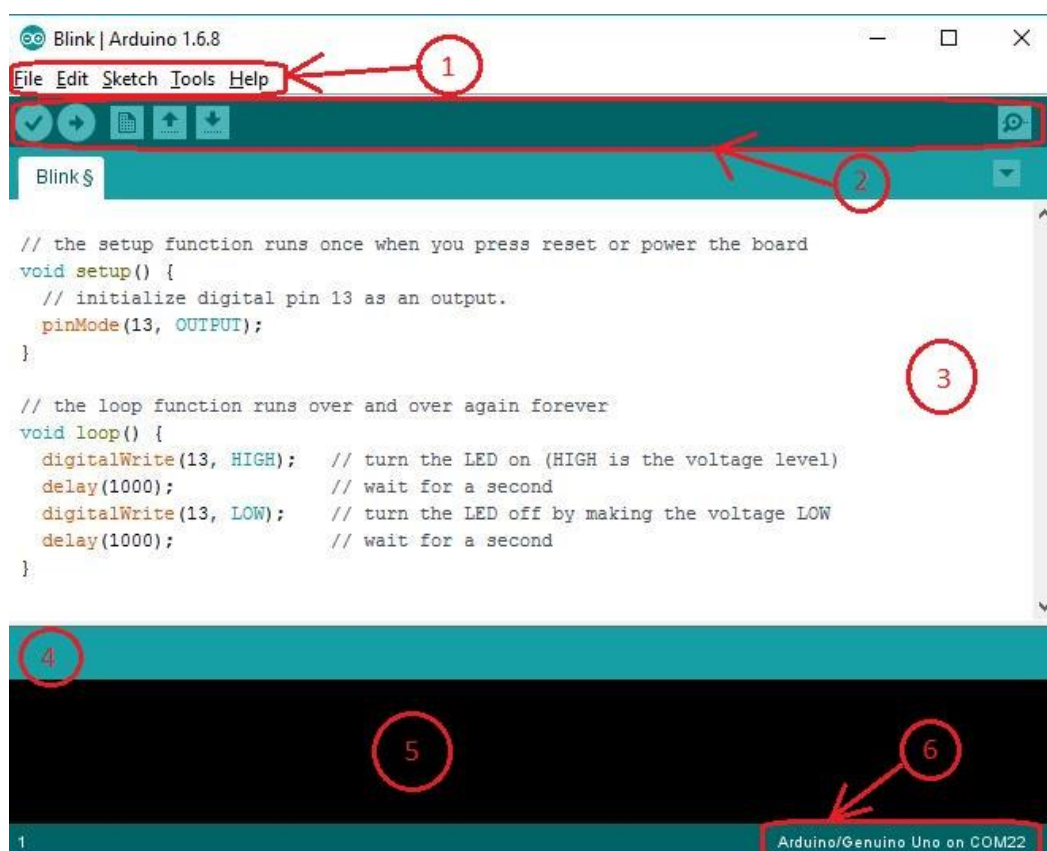
Tabela 2-3 Usporedba Programino i Visual Micro razvojnih okruženja

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 htio jednom proći 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 u **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** i 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.

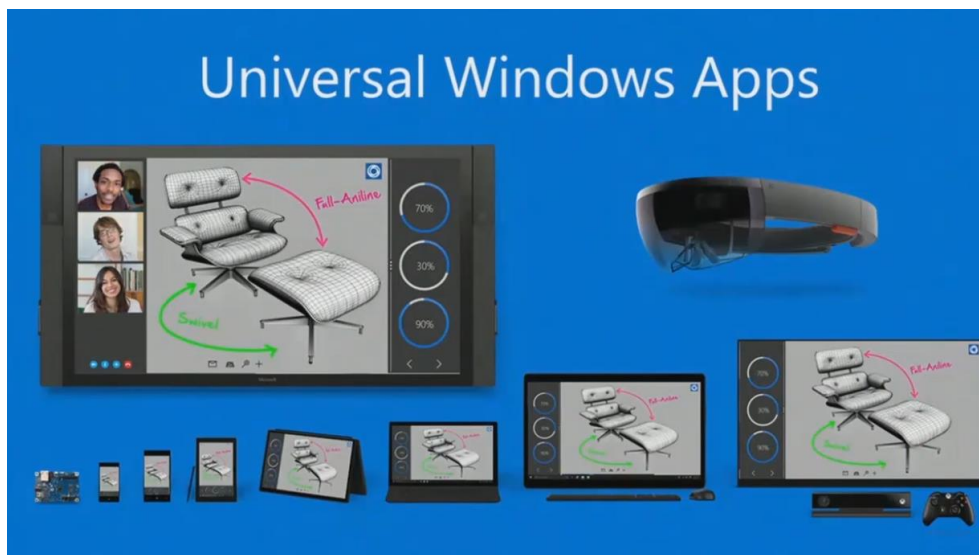


### 2.2. Općenito o Universal App (UWP) platformi

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.



Slika 2-18 Universal Windows Apps na različitim uređajima

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.



Neke od karakteristika ovih aplikacija su:

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

Porodica uređaja predstavlja API-e, 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*)**

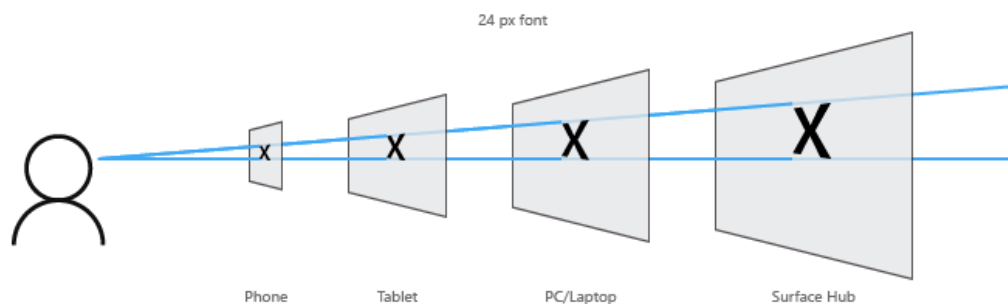
Omogućen je input na prirodan način, u vidu glasovnih naredbi, pokreta, pisanja po ekranu, različitih gestikulacija očima i sl., te je sve to vrlo lako implementirati u vlastita rješenja

- **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.

Najjednostavnije bi bilo objasniti realizaciju ovog modela kroz par slika:



*Slika 2-19 Algoritam skaliranja 24px fonta na različitim uređajima*

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 pomnožena 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:

- **Repozicija**

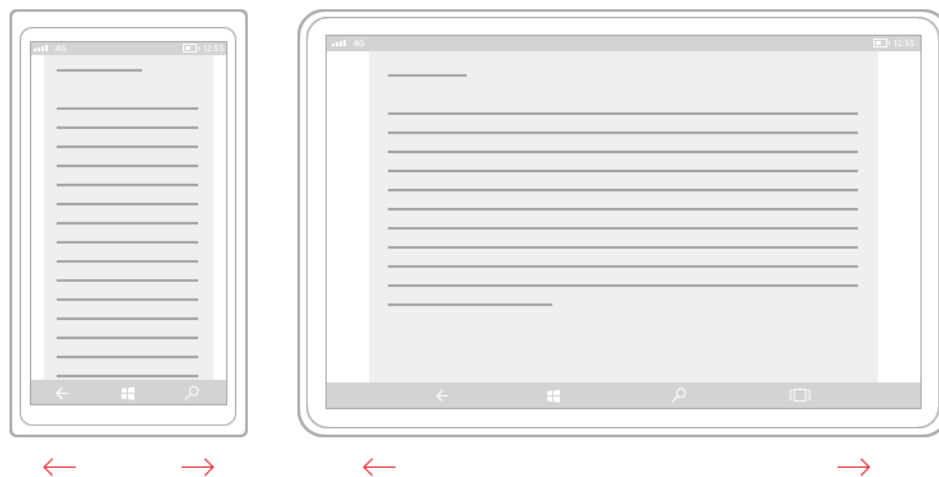
Da bi dobili najviše od našeg uređaja, pogodno je podesiti pozicije elemenata. U narednom primjeru vidimo dobro podešene elemente u ovisnosti od veličine ekrana.



*Slika 2-20 Repozicija elemenata na UI-u*

- **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 neki tekst siri u skuplja u ovisnosti od veličine ekrana.



*Slika 2-21 Promjena veličine elemenata na UI-u*

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

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



*Slika 2-22 Reflow*

- **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*

### - 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

### - 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.

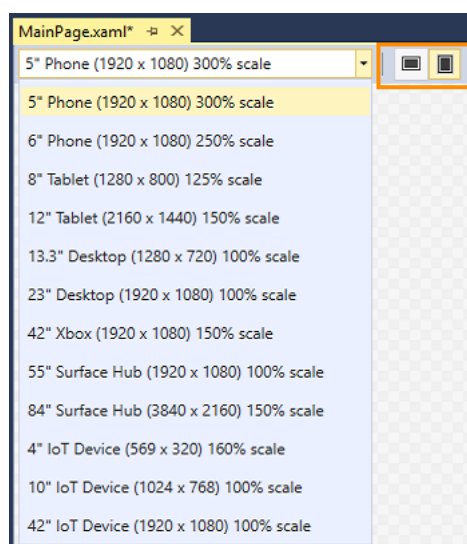


Slika 2-25 Potpuna reorganizacija

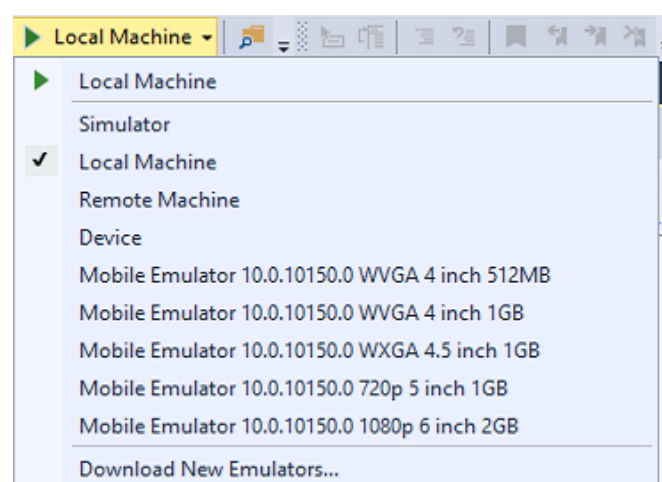
## 2.2.2. Visual Studio 2015 i UWP plugin

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) <sup>4</sup> 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 virtuelne 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

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

### 2.3. Bluetooth

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 potrebiti 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) zvuk

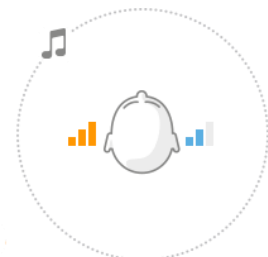
*Binaural* snimanje je metod snimanja zvuka koji koristi dva mikrofona postavljena s namjerom kreiranja osjećaja 3D stereo zvuka za onoga ko sluša. Efekat je često postignut koristeći tehniku poznatu kao „snimanje uz pomoć lutke“ gdje se lutki u oba uha stave mikrofoni i na taj način se snimaju zvukovi 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 u biti predstavlja prepreku koju glava pravi i 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



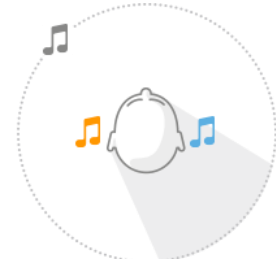
Slika 2-28 ITD

- **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

- **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

### 2.4.1. Tehnika snimanja

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

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 frekventijski 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. S obzirom da su pretežno pravljenе 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. Uspostavljanje platforme

### 3.2. Funkcionalni opis sistema

### 3.3. UML model

### 3.4. Arduino aplikacija

#### 3.4.1. Biblioteke

##### 3.4.1.1. *Arduino biblioteka New Ping*

#### 3.4.2. Arduino kod

### 3.5. UWP aplikacija

#### 3.5.1. UWP kod

## 4. Zaključak

## 5. Dodatak

## 6. Reference

**There are no sources in the current document.**