Izrada biblioteke s Arduino melodijama

Turkalj, Krešimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA, OSIJEK

Sveučilišni studij računarstva

IZRADA BIBLIOTEKE S ARDUINO MELODIJAMA

Završni rad

Krešimir Turkalj

Osijek, 2018

SADRŽAJ

1.UVOD 2

Izrada biblioteke s Arduino melodijama 2

2.OPIS KORIŠTENIH UREĐAJA I ALATA 3

2.1. Arduino razvojno okruženje 3
2.2. ATMega328P razvojni sustav 4
2.3. Pasivni piezo buzzer 5
2.4. EEPROM 24C256 5
2.5. Sklop s četiri tipkala 6

3. REALIZACIJA SUSTAVA 7

3.1. Zapisivanje nota u digitalni oblik 7
3.2. Spremanje nota u EEPROM 7
3.3. Programsko rješenje sustava 7

4. ZAKLJUČAK 19
LITERATURA 20
SAŽETAK 21
ABSTRACT 22
ŽIVOTOPIS 23
PRILOG 24

1.UVOD

U ovom završnom radu je potrebno napraviti bazu pjesama za izvođenje na arduino platformi. Potrebno je napraviti jednu arduino biblioteku te ju postaviti na javno dostupan GitHub. U dokumentaciji treba uz pojedinu pjesmu dodijeliti sliku melodija na klavijaturi, sliku notnog zapisa, riječi povezane uz melodiju te arduino program koji izvodi tu melodiju. U obzir dolaze samo poznate kratke melodije.

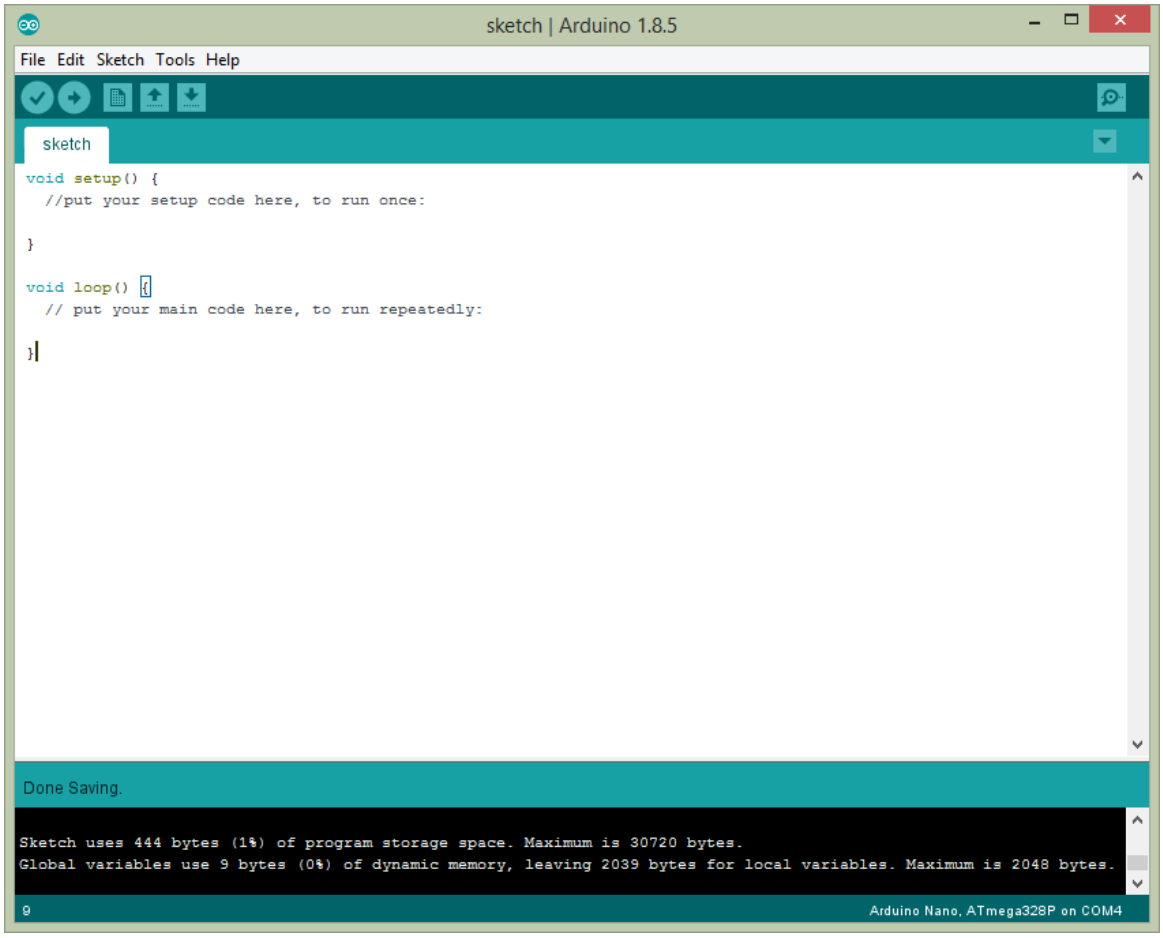
Izrada biblioteke s Arduino melodijama

U radu je napravljeno 8 pjesama s engleskim tekstom i 10 pjesama s hrvatskim tekstom. Pjesme su spremljene na EEPROM komponentu kojima pristupamo uz pomoć programskog rješenja, te ih sviramo na piezo buzzer komponenti.

2.OPIS KORIŠTENIH UREĐAJA I ALATA

2.1. Arduino razvojno okruženje

Arduino razvojno okruženje je program uz pomoć kojeg pišemo kod i zapisujemo ga na odabrani Arduino uređaj. Može se koristiti C i C++ jezik, no većina napisanih knjižnica koristi se C++ jezikom.Kod se sastoji od dijela koji se pokreće samo jednom kroz *setup()* funkciju i dijela koji se stalno ponavlja unutar *loop()* funkcije. Program napisan unutar Arduino razvojnog okruženja se zove *sketch* [1]. Njegova najveća veličina ovisi o mikroupravljaču koji određuje veličinu prostora za programski kod i varijable. Sučelje se može vidjeti na slici 2.1.



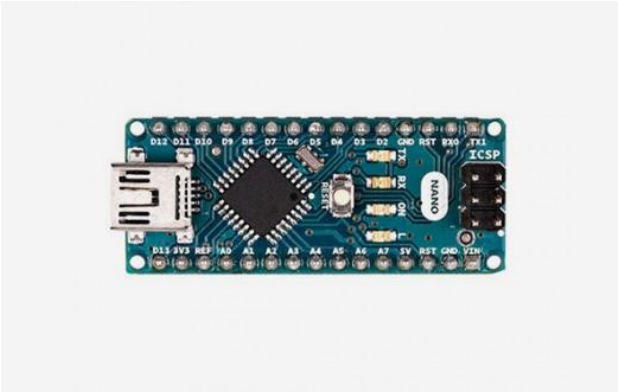
Sl. 2.1. Prikaz sučelja Arduino razvojnog okruženja.

2.2. ATMega328P razvojni sustav

ATMega328P je mikroupravljač koju koristi Arduino Nano sklopovlje. Koristi 8-bitni registar, ima 32 pina, od kojih su 11 digitalni, a 8 analogni. Sadrži 32 KB flash memorije od kojih 2 KB se koristi za SRAM. Unutarnji EEPROM sadrži 1 KB memorije. Ima unutarnje oscilatore koji rade na 1 MHz i 8 MHz s mogućnošću najveće frekvencije od 20 MHz. Treba napon između 1.8 od 5.5 V, te ima maksimalnu dopuštenu struju od 20 mA pri 5 V po pinu [2]. Navedeni podaci se nalaze u tablici 2.1., a izgled Arduino Nano sklopovlja na slici 2.1.

Mikroupravljač	ATMega328P
Preporučeni ulazni napon	1.8- 5.5 V
Digitalni U/I pinovi	11 (6 može davati PWM izlaz)
Analogni U/I pinovi	8
DC struja po U/I pinu	20 mA
Flash memorija	32 KB
SRAM	2 KB
Unutarnji EEPROM	1 KB
Takt procesora	16 MHz

Tab 2.1. Tablica s tehničkim karakteristikama ATMega328P.



Sl. 2.2. Izgled Arduino Nano sklopa s ATMega328P mikroupravljačem.

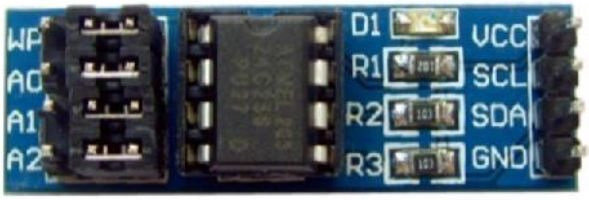
2.3. Pasivni piezo buzzer

Piezo buzzer ili zujalica je uređaj kod kojeg materijal unutar kućišta vibrira kada mu se dovede struja. Mogu biti pasivni i aktivni, razlika je što aktivni ima unutarnji oscilator, te mu treba samo DC napon, dok pasivni treba AC signal. Pasivna piezo zujalica ima X oznaku na kućištu [3]. Izgled zujalice i navedene oznake se može vidjeti na slici 2.3.

Sl. 2.3. Izgled pasivne piezo zujalice.

2.4. EEPROM 24C256

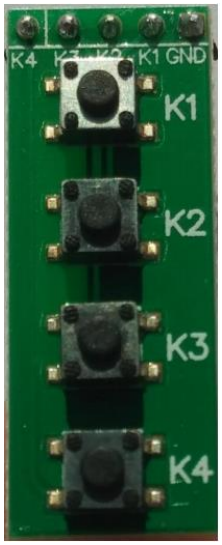
Za potrebe spremanje podataka koji ne stanu na Arduino Nano koristi se EEPROM vanjska memorija koja ima sveukupni kapacitet od 256 KB, podijeljeno na 8 dijelova po 32 KB. Sklop koristi 8 pinova, VCC, GND, SDA, SCL, WP i 3 analogna pina.WP pin se koristi za zaštitu od pisanja, SDA i SCL pinovi se koriste za slanje i uzimanje podataka s EEPROM-a, i 3 analogna pina se koriste za pristupanje dijelovima EEPROM-a [4]. Za potrebe završnog rada bilo je potrebno 32 KB, zbog čega nisu bile korišteni pinovi od A0 do A2. Izgled EEPROM-a s dodatnim pomoćnim okvirom se može vidjeti na slici 2.4.



Slika 2.4. Izgled EEPROM uređaja s označenim pinovima.

2.5. Sklop s četiri tipkala

Tipkalo je uređaj koji daje signal ovisno o tome je li pritisnut ili nije i u kojem načinu korištenja je. Postoje INPUT_PULLUP i INPUT. Kod INPUT_PULLUP koristi se unutarnji otpornik i otpušteno tipkalo daje HIGH signal. Kod INPUT se mora koristiti vanjski otpornik i daje LOW i opuštenom stanju [5]. Korišteni sklop ima 4 tipkala od kojih se koriste 3. Postoji 5 pinova, 4 za izlaz tipkala i peti za GND. Na slici 2.5. se nalazi izgled sklop s četiri tipkala



Slika. 2.6. Sklop s 4 tipkala.

3. REALIZACIJA SUSTAVA

3.1. Zapisivanje nota u digitalni oblik

Kako bi se ostvario sustav, prvo je potrebno napisati pjesme u digitalni oblik. Korištene pjesme se nalaze na slikama od broja 1 do 18 redom kojim su navedene: *Jingle Bells, Old McDonald had a farm, Yankee Doodle, Row Row Row Your Boat, We Wish You A Merry Christmas, Itsy Bitsy Spider, Black Sheep, Twinkle Twinkle Little Star, Sveti Nikola, Jeste L'Ikad Čuli To?, Hopa Cupa, Igra Kolo, Pile, Maca, Čista Cica, Moj Konjić, Jesenska* i *Djeca i Maca*. Svaka nota ima svoju frekvenciju i duljinu trajanja. U programskom rješenje se koristi jednodimenzionalno polje kod kojega u prvoj polovici stoje frekvencije nota, a u drugoj polovici stoje duljine nota.

3.2. Spremanje nota u EEPROM

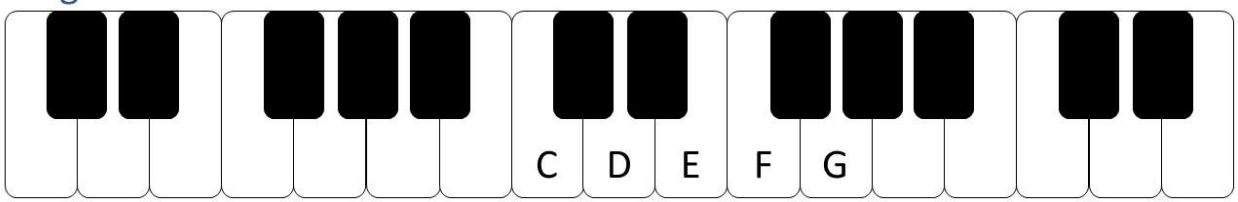
Arduino Nano sklopovlje nema mogućnost spremanja svih pjesama, zbog čega se mora koristiti vanjski EEPROM sklop. Jednom kad se postave podaci na EEPROM u prosjeku ostanu zapamćeni 10-15 godina, poslije tog vremena postoji mogućnost gubljenja podataka [6].

3.3. Programsko rješenje sustava

Funkcija Arduino programa su postaviti note na EEPROM, odabrati jednu pjesmu, iščitati podatke odabrane pjesme s EEPROM-a i zatim odsvirati ih preko uređaja. Prvi program je posvećen spremanju nota na EEPROM i treba se samo jednom izvršiti. EEPROM ima svoja ograničenja. Jedino može obraditi podatke u byte obliku, int oblik se prvo mora razdvojiti na byte-a. Programski kod za pretvorbu je prikazan pod 3.1. Kod zapisivanja više podataka odjednom mora se obratiti pozornost na granice stranice, inače će doći pisanja preko prijašnjih podataka. Također, kod čitanja se odjednom može najviše preuzeti 32 bajta odjednom [7]. Cijeli kod se nalazi unutar *setup* jer se treba izvršiti samo jednom. Prvo se otvara veza prema EEPROM-u, a zatim se poziva funkcija koja zapisuje podatke na EEPROM jedan iza drugoga. Varijable zauzimaju više nego što ima mogućeg mjesta, zbog čega se kod mora pokrenuti u dijelovima. Drugi program treba odabrati pjesmu, uzeti odabrane podatke s EEPROM i odsvirati ih preko uređaja.Uzimanje je izvedeno preko *switch case* kontrole toka s točnim određenim adresama za početak i koliko podataka će se uzeti, jer je poznata veličina svih polja. Nakon odabira pjesme, moraju se preuzeti podaci koji su zapisani u byte obliku, zbog čega se prvo moraju vratiti nazad u int oblik. To je izvedeno preko funkcije prikazane u 3.2. Nakon toga se podaci predaju petlji koji daje frekvenciju i duljinu trajanja funkciji. U *setup* dijelu se postavljaju ulazni pinovi za tipkala na INPUT_PULLUP, izlazni pin za zujalicu i otvara se veza prema EEPROM-u.U *loop* dijelu se provjeravaju stanja na tipkalima i zatim radi odabrana radnja,

ovisno o tipkalu. Prva dva tipkala su za odabiranje pjesme od 1 do 18, a treće tipka za pokretanje i zaustavljanje pjesama. Na slici 3.19. se nalazi izgled Arduino Nano sklopovlja sa spojenim komponentama Nakon završetka izrade programa, potrebno je postaviti programski kod na GitHub. To se može obaviti sa učitavanjem datoteka u kojima se nalazi kod na navedenu stranicu.

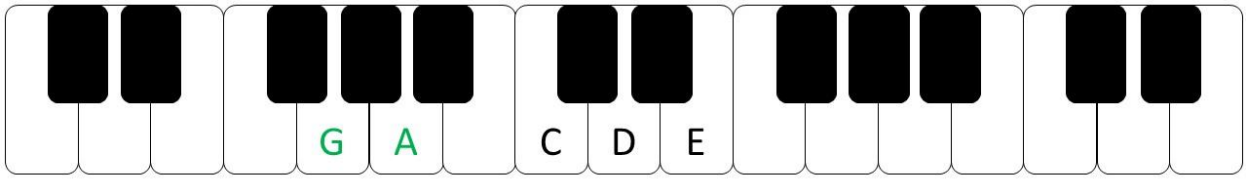
Jingle Bells



E	E	E	-	E	E	E	-	E	G	C	D	E	-	-	-
Jin-	gle	bells,		jin-	gle	bells		jin-	gle	all	The	way			
F	F	F	F	F	E	E	E E	E	D	D	E	D -	G -		
Oh	what	fun	it	is	to	ride	in a	one	horse	op	En	sle	igh		
E	E	E	-	E	E	E	-	E	G	C	D	E	-	-	-
Jin-	gle	bells,		jin	gle	bells,		jin-	gle	all	the	way			
F	F	F	F	F	E	E	E E	G	G	F	D	C			
Oh	what	fun	it	is	to	ride	in a	one	horse	op	en	sleigh			

Sl. 3.1. Pjesma Jingle Bells.

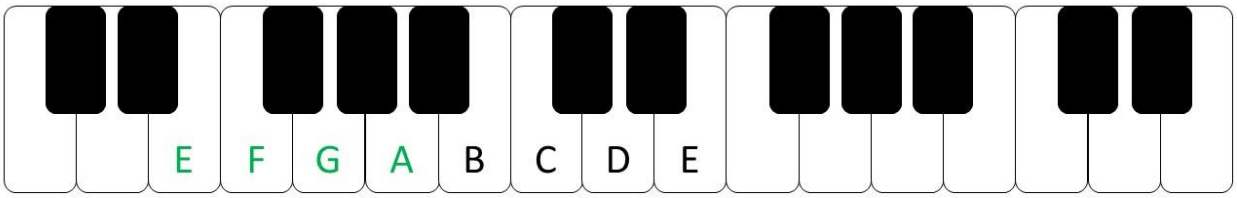
Old MacDonald had a farm



C	C	C	G	A	A	G	E	E	D	D	C				
Old	Mac	Don-	ald	had	A	farm,	E-	I-	E-	I-	O.				
G	C	C	C	G	A	A	G	E	E	D	D	C			
And	on	his	farm	he	had	some	chicks	E-	I-	E-	I-	O			
G	G	C	C	C	G	G	C	C	C						
With	a	chick,	chick	here	and	a	chick,	chick	there						
C	C	C	C	C	C	C	C	C	G	A	A	G			
Eve-	rywh-	ere	a	chick,	Chick,	Old	Mac	Don-	ald	had	a	farm,			
E	E	D	D	C											
E-	I-	E-	I-	O.											

Sl. 3.2. Pjesma Old MacDonald had a farm.

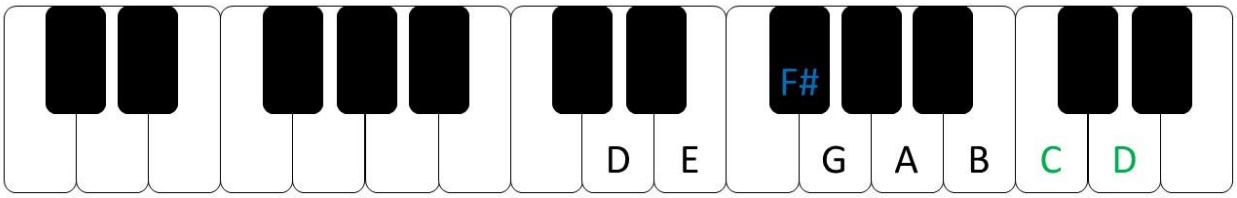
Yankee Doodle



C	C	D	E	C	E	D	G	C	C	D	E	C	B		
Yan	kee	Doo	dle	went	to	town	A-	ri	ding	on	a	po	ny		
G	C	C	D	E	F	E	D	C	B	G	A	B	C	C	
He	stuck	a	fea	ther	in	his	hat	and	called	it	ma	ca	ro	ni	
A	B	A	G	A	B	C	G	A	G	F	E	G			
Yan	kee	Doo	dle	keep	it	up	Yan	kee	Doo	dle	dan	dy			
A	B	A	G	A	B	C	A	G	C	B	D	C			
Mind	the	music	and	the	step	and	with	the	girls	be	han	dy!			

Sl. 3.3. Pjesma Yankee Doodle.

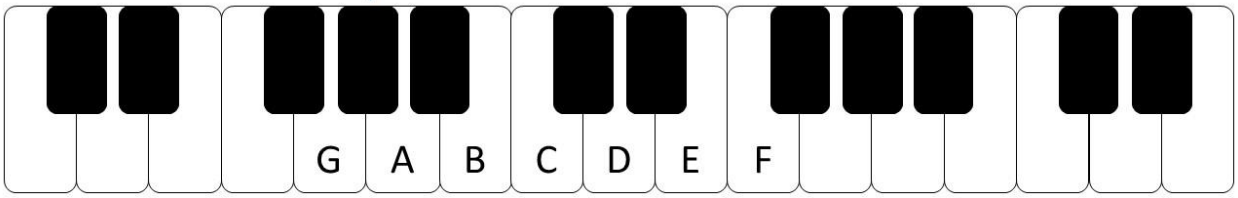
Row Row Row Your Boat



D	D	D	E	F#	F#	E	F#	G	A		
Row,	row,	row	your	boat,	gen	tly	down	the	stream.		
D	D	D	A	A	A	F#	F#	F#	D	D	D
Me	rri	ly,	me	rri	ly,	me	rri	ly,	me	rri	ly,
A	G	F#	E	D							
Life	is	but	a	dream.							

Sl. 3.4. Pjesma Row Row Row Your Boat.

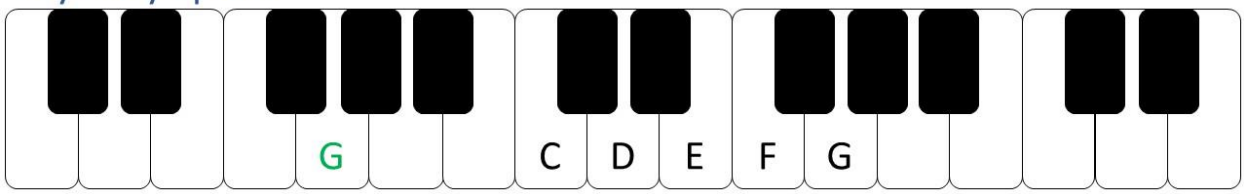
We Wish You A Merry Christmas



G	C	C	D	C	B	A	A			
We	wish	you	a	me	rry	Christ	mas,			
A	D	D	E	D	C	B	G			
we	wish	you	a	me	rry	Christ	mas,			
G	E	E	F	E	D	C	A			
we	wish	you	a	me	rry	Christ	mas			
G	G	A	D	B	C					
and	a	Hap	py	New	Year!					

Sl. 3.5. Pjesma We wish you a Merry Christmas.

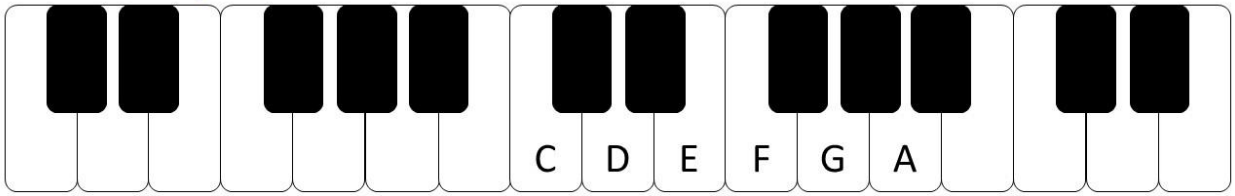
Itsy Bitsy Spider



G	C	C	C	D	E	E	E	D	C	D	E	C	
The	its	sy-	bit	sy	spi	der	climbed	up	the	wa	ter	spout	
E	E	F	G	G	F	E	F	G	E	C	C	D	E
Down	came	the	rain	and	washed	the	spi	der	out	out	came	the	sun
E	D	C	D	E	C	G	G	C	C	C	D	E	E
and	dried	up	all	the	rain	and	the	it	sy-	bit	sy	spi	der
E	D	C	D	E									
Climbed	up	the	spout	again									

Sl. 3.6. Pjesma Itsy Bitsy Spider.

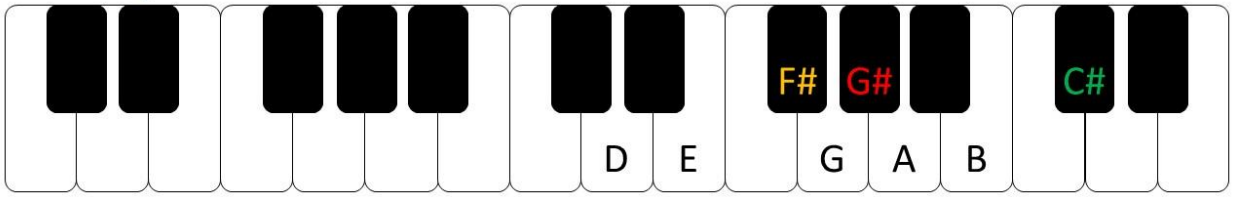
Black Sheep



C	C	G	G	A	A	A	A	G	F	F	E	E	D	D	C
Baa,	baa,	black	sheep,	have	you	a	ny	wool?	Yes	sir,	yes	sir,	three	bags	full.
G	G	G	F	F	E	E	E	D	C	G	G	G	F	F	F
One	for	the	mas	ter,	one	for	the	dame,	and	one	for	the	lit	tle	boy
F	E	E	E	D											
who	lives	down	the	lane.											
C	C	G	G	A	A	A	A	G	F	F	E	E	D	D	C
Baa,	baa,	black	sheep,	have	you	a	ny	wool?	Yes	sir,	yes	sir,	three	bags	full.

Sl. 3.7. Pjesma Black Sheep.

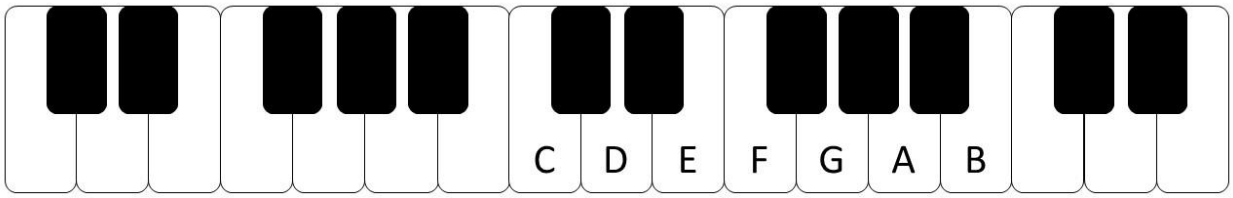
Twinkle, Twinkle Little Star



E	E	B	B	C#	C#	B	A	A	G#	G#	F#	F#	E
Twin	kle,	twin	kle,	lit	tle	star,	how	I	won	der	what	you	are.
B	B	A	A	G#	G#	F#	B	B	A	A	G#	G#	F#
Up	a	bove	the	world	so	high,	like	a	dia	mond	in	the	sky
E	E	B	B	C#	C#	B	A	A	G#	G#	F#	F#	E
Twin	kle,	twin	kle,	lit	tle	star,	how	I	won	der	what	you	are.

Sl. 3.8. Pjesma Twinkle Twinkle Little Star.

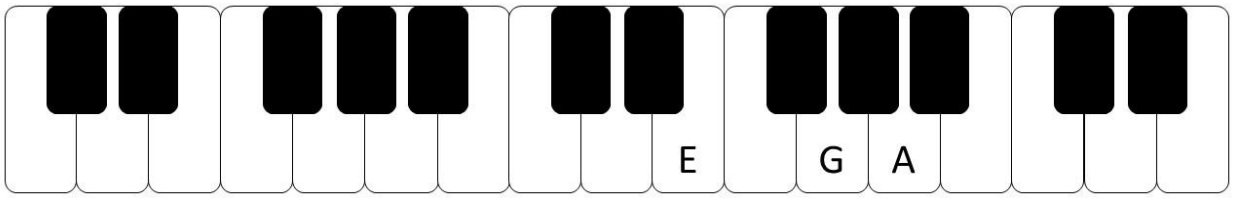
Sveti Nikola



D	D	F	F	A	A	F	F	B	B	G	B	A	A	-	
Sve	ti	Ni	ko	svije	tom	še	ta	tra	ži	dje	cu	svo	ju		
G	G	A	G	F	A	D	D	E	F	G	E	D	D	-	
jer	za	sva	ko	do	bro	dije	te	i	ma	stvar	cu	ko	ju		
D	D	F	F	A	A	F	F	B	B	G	B	A	A	-	
I	pred	na	šom	stat’	će	ku	ćom	jer	smo	do	bri	bi	li		
G	G	A	G	F	A	D	D	E	F	G	E	D	D	-	
pa	smo	dra	gom	Sve	tom	Ni	ki	da	nas	vr	lo	mi	li		

Sl. 3.9. Pjesma Sveti Nikola.

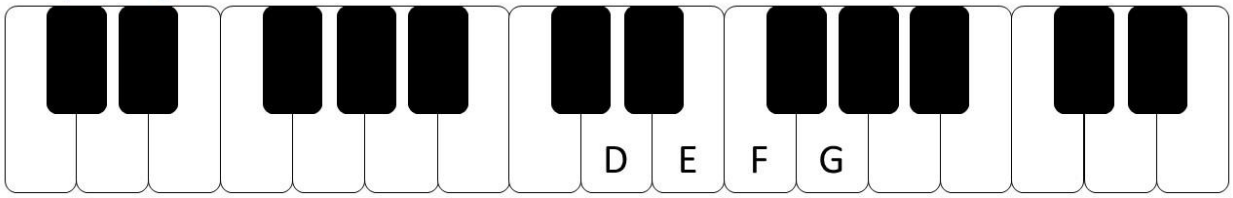
Jeste L’i Kad Čuli To?



G	G	A	A	G	G	E	E	G	G	A	A	G	G	E	E
Ko	njić	tje	ra	ko	či	ja	ša,	ze	ko	go	ni	psa	bun	da	ša.
G	G	A	G	G	E	G	G	A	A	G	G	E			
O-	ho-	hoo,	o-	ho-	hoo.	Je	ste	l’i	kad	ču	li	to?			

Sl. 3.10. Pjesma Jeste l'ikad čuli to.

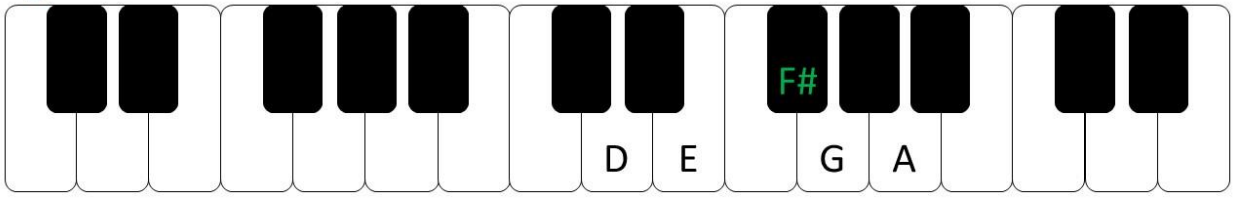
Hopa Cupa



G	G	F	F	G	G	F	F	E	G	F	E	F	E	D	D
Ho	pa	cu	pa	čiž	me	mo	je,	još	kod	ku	će	i	ma m	dvo	je.
G	G	F	F	G	G	F	F	E	G	F	E	F	E	D	D
I	te	dvo	je	ni	su	mo	je,	već	kom	ši	je	kri	vo	ši	je.

Sl. 3.11.Pjesma Hopa Cupa.

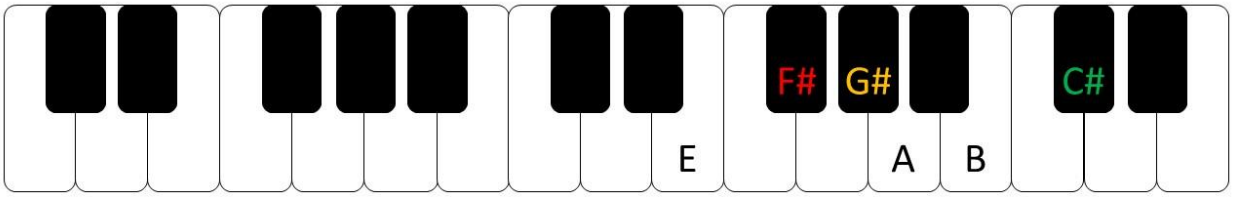
Igra Kolo



A	A	A	F#	A	A	A	F#	A	A	G	G	F#	D		
I	gra	ko	lo,	i	gra	ko	lo	u	dva	de	set	i	dva,		
F#	F#	F#	D	F#	F#	F#	D	F#	F#	E	E	D	D		
u	tom	ko	lu,	u	tom	ko	lu	lije	pa	Ma	ra	i	gra.		
A	A	A	F#	A	A	A	F#	A	A	G	G	F#	D		
U	zmi	Ma	ro,	u	zmi	Ma	ro	ko	ga	ti	je	dra	go,		
F#	F#	F#	D	F#	F#	F#	D	F#	F#	E	E	D	D		
sa	mo	ne	moj,	sa	mo	ne	moj	ko	ga	ne	maš	ra	do.		

Sl. 3.12. Pjesma Igra kolo.

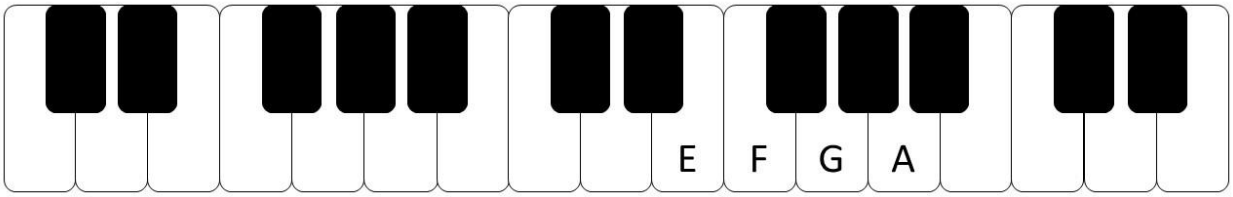
Pile



E	E	E	F#	F#	F#	F#	F#	F#	G#	G#	G#				
Ko	ki	no	dje	ten	ce,	žuč	ka	sto	pi	len	ce				
A	A	A	G#	F#	F#	-	E	F#	G#	F#	E	E	-		
do	ruč	ku	je	pr	ve		u	ži	vo	tu	mr	ve.			
E	E	E	-	C#	E	-	-	E	E	E	-	C#	E	-	-
Piu,	piu,	piu,			piu,			piu,	piu,	piu,			piu.		
-	-	C#													
		Piu.													

Sl. 3.13. Pjesma Pile.

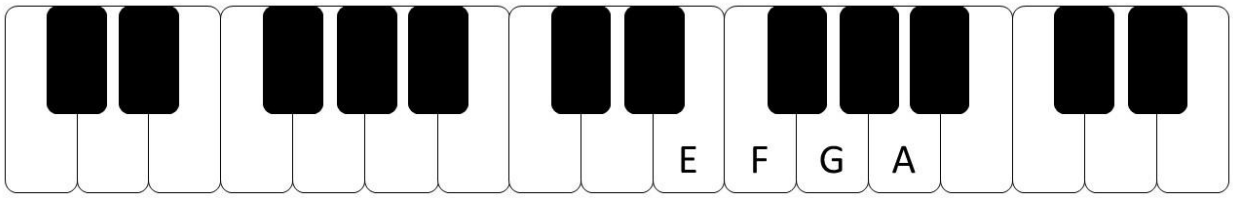
Maca



G	F	E	E	A	A	G	G	F	F	E	F	E	E		
Ma	la	ma	ca	li	ce	re	di	s'pre	dnje	ša	pe	dvi	je.		
G	F	E	E	A	A	G	G	F	F	E	F	E	E		
Ma	lo	dije	te	ru	ke	i	ma	da	si	li	ce	mi	je.		

Sl. 3.14. Pjesma Maca.

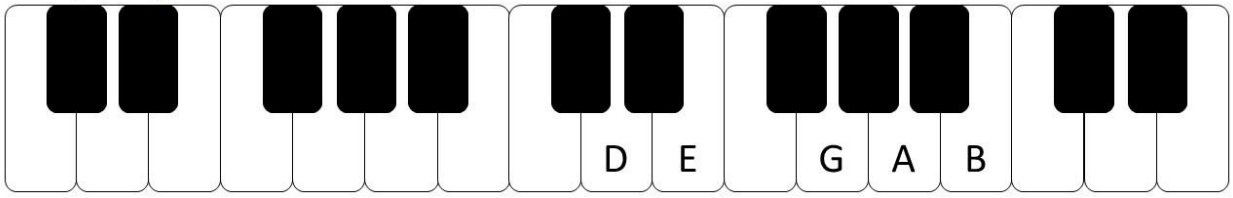
Čista Cica



G	G	G	F	E	E	G	G	G	F	E	E				
Po	gle	daj	te	li	ce	na	še	ma	le	Ci	ce.				
A	A	A	G	F	A	G	G	G	F	E	E				
Ka	ko	sa	mo	bli	sta,	jer	je	Ci	ca	či	sta.				
G	G	G	F	E	E	G	G	G	F	E	E				
O	bje	su	joj	ru	ke	o	pra	ne	bez	mu	ke.				
A	A	A	G	F	A	G	G	G	F	E	E				
ka	o	ro	sa	bli	sta,	na	ša	Ci	ca	či	sta.				

Sl. 3.15. Pjesma Čista Cica.

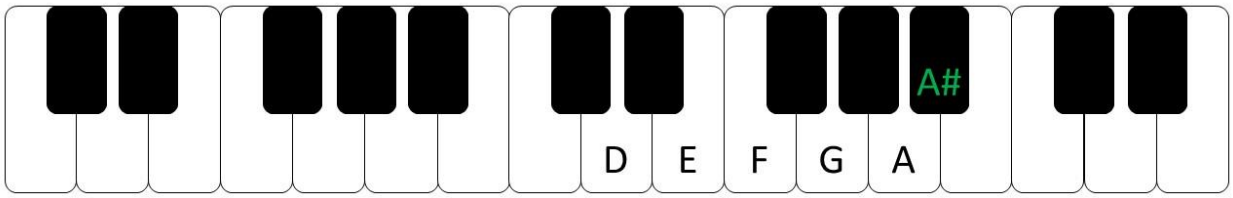
Moj konjić



B	G	A	B	G	E	D	D	A	G	A	B	G	-		
Ja	uz	ja	šem	svo	ga	ko	nja,	je	dan,	dva,	tri,	hop!			
A	G	A	B	G	E	D	D	A	G	A	E	G	-		
A	on	od	mah	vje	tro	go	nja	pre	đe	u	ga	lop!			
D	B	A	E	B	A	D	D	E	D	G	A	B			
Sve	on	ru	ši	što	mu	sme	ta	na	pu	tu	na	tlu.			
D	B	A	E	B	A	D	D	E	G	A	B	G			
O	bi	đe	mo	po	la	svije	ta	i	o	pet	smo	tu.			

Sl. 3.16. Pjesma Moj Konjić.

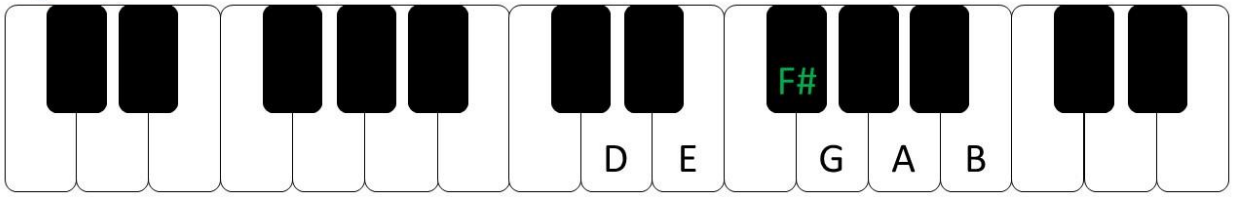
Jesenska



D	E	F	G	A	A	A#	A	G	A#	A					
Ti	ho,	ti	ho	pa	da	list	za	list	om	žut,					
D	E	F	G	A	A	A#	A	G	A#	A					
po	kri	o	je	sta	ze,	ce	ste,	šum	ski	put.					
G	G	A	G	F	D	E	F	G	A#	A					
Vje	tar	gra	nje	lju	lja	nje	žno	k'o	u	snu,					
G	G	A	G	F	D	E	E	F	E	D					
bla	gi	mir	je	šu	mu	o	bu	ze	o	svu.					

Sl. 3.17. Pjesma Jesenska.

Djeca i maca



F#	G	A	A	A	B	A	F#								
Po	kraj	pe	ći	ma	ca	pre	la,								
A	A	G	G	F#	G	F#	E	F#	F#	G	E				
do	nje	do	bra	dje	ca	sje	la,	pa	joj	ta	ko				
F#	F#	G	E	F#	F#	E	E	D	D	D	D				
šap	tat	sta	la,	pre	di	pre	đu	ma	co	ma	la.				

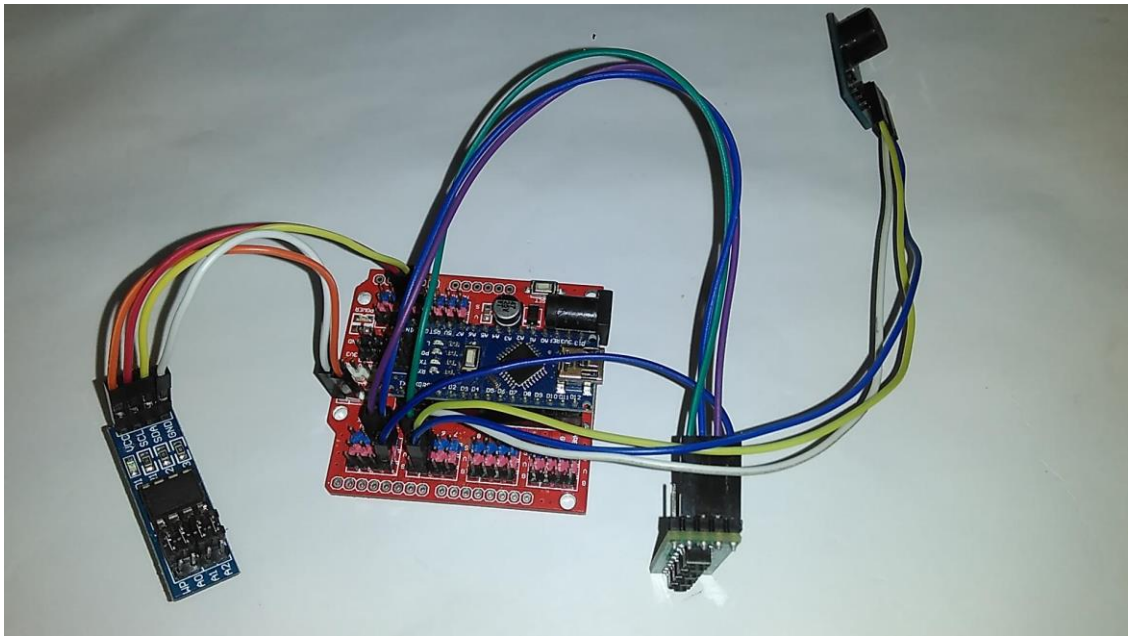
Sl. 3.18. Pjesma Djeca i maca.

```
byte* IntToByteArray(int* intArray, int arraySize) {
  byte high, low;
  int j = 0;
  byte* byteArray = new byte[arraySize];
  for (int i = 0; i < arraySize/2; i++) {
    high = (byte) ((intArray[i] >> 8));
    low = (byte) (intArray[i] & 0xFF);
    byteArray[j] = high;
    byteArray[j + 1] = low;
    j += 2;
  }
  return byteArray;
}
```

Pr. kod 3.1. Pretvorba polja int brojeva u polje byte-ova.

```
int* ByteToIntArray(byte* byteArray, int intSize) {
  int j = 0;
  byte high, low;
  int* intArray = new int[intSize];
  for (int i = 0; i < intSize; i++) {
    high = byteArray[j];
    low = byteArray[j + 1];
    intArray[i] = (((int)high<<8) | (int)low );
    j += 2;
  }
  delete byteArray;
  return intArray;
}
```

Pr. kod 3.2. Pretvorba polja byte-ova u polje int brojeva.



Sl 3.19. Prikaz spajanja komponenti na Arduino Nano.

4. ZAKLJUČAK

Cilj ovog rada bio je napraviti sustav uz pomoć Arduino Nano sklopovlja i piezo zujalice koji će svirati tonove u ritmu pjesama. Prvo je trebalo zapisati note u programskom kodu, a zatim ih pretvoriti u byte oblik i zapisati na EEPROM radi daljnjeg korištenja. Iduće je trebalo preuzeti i pretvoriti podatke nazad u int oblik. Zadnji korak je bilo napraviti način kako upravljati izborom i pokretanjem uz pomoć tipkala. Na kraju je trebalo staviti cijeli kod na GitHub stranicu.

LITERATURA

[1] Arduino Guide - https://www.arduino.cc/en/Guide/Environment (23.9.2018)

[2] E-radionica - https://e-radionica.com/hr/atmega328p-smd.html (23.9.2018)

[3] Microbuzzer - https://web.archive.org/web/20150720220454/http://www.microbuzzer.com/active-buzzer-and-passive-buzzer-difference.shtml (23.9.2018)

[4] Hobbytronics external EEPROM – http://www.hobbytronics.co.uk/tutorials-code/arduino-tutorials/arduino-external-eeprom (23.9.2018)

[5] Arduino pinmode – https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/ (23.9.2018)

[6] The Museum of HP Calculators – http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/archv021.cgi?read=237068 (23.9.2018)

[7] Hobbytronics page write - http://www.hobbytronics.co.uk/eeprom-page-write (23.9.2018)

SAŽETAK

Naslov: Izrada biblioteke s Arduino melodijama

U ovom završnom radu je napravljen sustav uz pomoć Arduino Nano sklopovlja koji će svirati pjesme preko piezo zujalice. Note pjesme su zapamćene na EEPROM i po potrebi se preuzimaju i koriste za sviranje. Odabir pjesme i upravljanje se izvodi preko tipkala, prva dva za promjenu pjesme i treći za pokretanje i zaustavljanje sviranja pjesme.

Ključne riječi: Arduino, pasivni piezo buzzer, EEPROM

ABSTRACT

Title: Making of library with Arduino melodies

In this final assignment system was made with Arduino Nano hardware that plays songs via passive piezo buzzer. Notes for songs are stored on EEPROM and are taken when required and used for playing. Selection of songs and starting is done via buttons, first two are used for selection and third is used as start and stop.

Keywords: Arduino, passive piezo buzzer, EEPROM

ŽIVOTOPIS

Krešimir Turkalj rođen je u Vinkovcima, Republika Hrvatska 7. veljače 1996. godine. Pohađao je osnovnu školu “Ivan Mažuranić” u Vinkovcima. U sedmom i osmom razredu sudjeluje na državnom natjecanju iz kemije. Nakon osnovne škole, 2011. godine upisuje Gimnaziju “Matije Antuna Reljkovića”. U prvom razredu sudjeluje na državnom natjecanju iz kemije. Završava sva četiri razreda sa vrlo dobrim uspjehom. Maturirao je 2015. godine. 2015. godine upisuje Elektrotehnički fakultet u Osijeku, smjer računarstvo. Nakon završenog preddiplomskog studija planira upisati diplomski studij na istome fakultetu.

PRILOG

Cijeli kod se može naći na https://github.com/KresimirTurkalj/ArduinoBuzzer

```
#include <Wire.h>
#include "songs.h"
#define ADDRESS 0x50//Address of 24LC256 ADDRESS, eeprom chip


void setup() {
 Wire.begin();
 unsigned int ee = 0, writePos = 0, arraySize = 0;
 for (int i = 0; i < N_SONGS; i++) {
   switch (i) {
      case 0: arraySize = sizeof(JingleBells);
      writeEEPROM(ee + writePos, IntToByteArray(JingleBells, arraySize), arraySize);
      break;
      case 1: arraySize = sizeof(OldMcDonald);
      writeEEPROM(ee + writePos, IntToByteArray(OldMcDonald, arraySize), arraySize);
      break;
      case 2: arraySize = sizeof(YankeeDoodle);
      writeEEPROM(ee + writePos, IntToByteArray(YankeeDoodle, arraySize), arraySize);
      break;
      case 3: arraySize = sizeof(RowYourBoat);
      writeEEPROM(ee + writePos, IntToByteArray(RowYourBoat, arraySize), arraySize);
      break;
      case 4: arraySize = sizeof(WeWishYou);
      writeEEPROM(ee + writePos, IntToByteArray(WeWishYou, arraySize), arraySize);
      break;
      case 5: arraySize = sizeof(ItsyBitsy);
      writeEEPROM(ee + writePos, IntToByteArray(ItsyBitsy, arraySize), arraySize);
      break;
      case 6: arraySize = sizeof(BlackSheep);
      writeEEPROM(ee + writePos, IntToByteArray(BlackSheep, arraySize), arraySize);
      break;
      case 7: arraySize = sizeof(TwinkleTwinkle);
      writeEEPROM(ee + writePos, IntToByteArray(TwinkleTwinkle, arraySize), arraySize);
      break;
      case 8: arraySize = sizeof(SvetiNikola);
      writeEEPROM(ee + writePos, IntToByteArray(SvetiNikola, arraySize), arraySize);
      break;
      case 9: arraySize = sizeof(JesteLIkad);
      writeEEPROM(ee + writePos, IntToByteArray(JesteLIkad, arraySize), arraySize);
      break;
      case 10: arraySize = sizeof(HopaCupa);
      writeEEPROM(ee + writePos, IntToByteArray(HopaCupa, arraySize), arraySize);
      break;
```

```
      case 11: arraySize = sizeof(IgraKolo);
      writeEEPROM(ee + writePos, IntToByteArray(IgraKolo, arraySize), arraySize);
      break;
      case 12: arraySize = sizeof(Pile);
      writeEEPROM(ee + writePos, IntToByteArray(Pile, arraySize), arraySize);
      break;
      case 13: arraySize = sizeof(Maca);
      writeEEPROM(ee + writePos, IntToByteArray(Maca, arraySize), arraySize);
      break;
      case 14: arraySize = sizeof(CistaCica);
      writeEEPROM(ee + writePos, IntToByteArray(CistaCica, arraySize), arraySize);
      break;
      case 15: arraySize = sizeof(MojKonjic);
      writeEEPROM(ee + writePos, IntToByteArray(MojKonjic, arraySize), arraySize);
      break;
      case 16: arraySize = sizeof(Jesenska);
      writeEEPROM(ee + writePos, IntToByteArray(Jesenska, arraySize), arraySize);
      break;
      case 17: arraySize = sizeof(DjecaMaca);
      writeEEPROM(ee + writePos, IntToByteArray(DjecaMaca, arraySize), arraySize);
      break;
      }
   writePos += arraySize;
   }
}

void loop() {}
byte* IntToByteArray(int* intArray, int arraySize) {
 byte high, low;
 int j = 0;
 byte* byteArray = new byte[arraySize];
 for (int i = 0; i < arraySize/2; i++) {
   high = (byte) ((intArray[i] >> 8));
   low = (byte) (intArray[i] & 0xFF);
   byteArray[j] = high;
   byteArray[j + 1] = low;
   j += 2;
   }
 return byteArray;
}

void writeEEPROM(unsigned int eeaddress, byte* data, unsigned int data_len){
 byte i = 0, counter = 0;
 unsigned int  address;
```

```
 unsigned int  page_space;
 unsigned int  page = 0;
 unsigned int  num_writes;
 byte first_write_size;
 byte last_write_size;
 byte write_size;

 page_space = int(((eeaddress / 64) + 1) * 64) - eeaddress;

// Calculate first write size
 if (page_space > 16) {
   first_write_size = page_space - ((page_space / 16) * 16);
      if (first_write_size == 0) first_write_size = 16;
   }
 else first_write_size = page_space;

// calculate size of last write
 if (data_len > first_write_size)  last_write_size = (data_len - first_write_size) % 16;

// Calculate how many writes we need
 if (data_len > first_write_size) num_writes = ((data_len - first_write_size) / 16) + 2;
 else num_writes = 1;
   i = 0;
   address = eeaddress;
   for (page = 0; page < num_writes; page++){
   if (page == 0) write_size = first_write_size;
   else if (page == (num_writes - 1)) write_size = last_write_size;
      else write_size = 16;
   Wire.beginTransmission(ADDRESS);
   Wire.write((int)((address) >> 8));   // MSB
   Wire.write((int)((address) & 0xFF)); // LSB
   counter = 0;
   do {
   Wire.write((byte)data[i]);
   i++;
   counter++;
   } while ((i<data_len) && (counter < write_size));
   Wire.endTransmission();
   address += write_size; // Increment address for next write

   delay(6);  // needs 5ms for page write
   }
 delete data;
}
```

```
#define DEBOUNCE_DELAY 100
#define TONE_PIN 5
#define N_SONGS 18
#define GO_PIN 4
#define UP_PIN 2
#define DOWN_PIN 3
#define ADDRESS 0x50

#include <Wire.h>

byte index = 0;
bool goState = false;

unsigned long lastGoDebounceTime = 0;
unsigned long lastUpDebounceTime = 0;
unsigned long lastDownDebounceTime = 0;

short int buttonGoState;
short int buttonUpState;
short int buttonDownState;

short int readingGo;
short int readingUp;
short int readingDown;

short int lastGoButtonState = HIGH;
short int lastUpButtonState = HIGH;
short int lastDownButtonState = HIGH;

void setup()
{
 Wire.begin();
 pinMode(UP_PIN, INPUT_PULLUP);
 pinMode(DOWN_PIN, INPUT_PULLUP);
 pinMode(GO_PIN, INPUT_PULLUP);
 pinMode(TONE_PIN, OUTPUT);
}

void loop()
{
 if (CheckGo()) goState = true;
 if (CheckUp()) IncrementIndex();
 if (CheckDown()) DecrementIndex();
 if(goState) PlaySong();
```

```
}

bool CheckGo(){
 readingGo = digitalRead(GO_PIN);
 if (readingGo != lastGoButtonState) {
   lastGoDebounceTime = millis();
   lastGoButtonState = readingGo;
}
 if (millis() - lastGoDebounceTime > DEBOUNCE_DELAY){
   if (readingGo != buttonGoState){
      buttonGoState = readingGo;
      if (buttonGoState == LOW) return true;
   }
 }
 return false;
}

bool CheckDown(){
 readingDown = digitalRead(DOWN_PIN);
 if (readingDown != lastDownButtonState) {
   lastDownDebounceTime = millis();
   lastDownButtonState = readingDown;
}
 if (millis() - lastDownDebounceTime > DEBOUNCE_DELAY){
   if (readingDown != buttonDownState){
      buttonDownState = readingDown;
      if (buttonDownState == LOW) return true;
   }
 }
 return false;
}
bool CheckUp(){
 readingUp = digitalRead(UP_PIN);
 if (readingUp != lastUpButtonState) {
   lastUpDebounceTime = millis();
   lastUpButtonState = readingUp;
}
 if (millis() - lastUpDebounceTime > DEBOUNCE_DELAY){
   if (readingUp != buttonUpState){
      buttonUpState = readingUp;
      if (buttonUpState == LOW) return true;
   }
 }
 return false;
```

```
}

void IncrementIndex(){
 if (index == N_SONGS - 1) index = 0;
 else index++;
}

void DecrementIndex(){
 if (index == 0) index = N_SONGS - 1;
 else index--;
}

void PlaySong(){
 int* song;
 int songLength, frequency, duration;
 song = GetSongData(&songLength);
 delay(750);
 for (int i = 0; i < songLength; i++)
 {
   if(digitalRead(GO_PIN) == LOW){
      goState = false;
      break;
   }
   if(digitalRead(UP_PIN) == LOW){
      IncrementIndex();
      break;
   }
   if(digitalRead(DOWN_PIN) == LOW){
      DecrementIndex();
      break;
   }
   frequency = song[i];
   duration = song[i + songLength];
   tone(TONE_PIN, frequency, duration);
   delay(duration * 1.1);
   noTone(TONE_PIN);
 }
 delete song;
}

int* GetSongData(int* songLength) {
 byte* song;
 int startEE, remainChars, pageLoad, passedChars = 0;
 switch (index) {
```

```
case 0 : *songLength = 51;
   startEE = 0;
   break;
case 1 : *songLength = 61;
   startEE = 204;
   break;
case 2 : *songLength = 55;
   startEE = 448;
   break;
case 3 : *songLength = 27;
   startEE = 668;
   break;
case 4 : *songLength = 30;
   startEE = 776;
   break;
case 5 : *songLength = 47;
   startEE = 896;
   break;
case 6 : *songLength = 52;
   startEE = 1084;
   break;
case 7 : *songLength = 42;
   startEE = 1292;
   break;
case 8 : *songLength = 60;
   startEE = 1460;
   break;
case 9 : *songLength = 29;
   startEE = 1700;
   break;
case 10: *songLength = 32;
   startEE = 1816;
   break;
case 11: *songLength = 56;
   startEE = 1944;
   break;
case 12: *songLength = 42;
   startEE = 2168;
   break;
case 13: *songLength = 28;
   startEE = 2336;
   break;
case 14: *songLength = 48;
   startEE = 2448;
```

```
      break;
    case 15: *songLength = 54;
      startEE = 2640;
      break;
    case 16: *songLength = 44;
      startEE = 2856;
      break;
    case 17: *songLength = 32;
      startEE = 3032;
      break;
  }
  song = new byte[*songLength * 4];
  remainChars = *songLength * 4;
  while(remainChars > 0){
    if(remainChars > 32){
      remainChars -= 32;
      pageLoad = 32;
    }
    else{
      pageLoad = remainChars;
      remainChars = 0;
    }
    readEEPROM(startEE, song, passedChars, pageLoad);
    passedChars += pageLoad;
    startEE += pageLoad;
  }
  return ByteToIntArray(song, *songLength * 2);
}

int* ByteToIntArray(byte* byteArray, int intSize) {
  int j = 0;
  byte high, low;
  int* intArray = new int[intSize];
  for (int i = 0; i < intSize; i++) {
    high = byteArray[j];
    low = byteArray[j + 1];
    intArray[i] = (((int)high<<8) | (int)low );
    j += 2;
  }
  delete byteArray;
  return intArray;
}

void readEEPROM(int eeaddress, byte* data, byte i, int num_chars){
```

```
 Wire.beginTransmission(ADDRESS);

 Wire.write((int)(eeaddress >> 8));   // MSB
 Wire.write((int)(eeaddress & 0xFF)); // LSB
 Wire.endTransmission();

 Wire.requestFrom(ADDRESS, num_chars);

 while (Wire.available()) data[i++] = Wire.read();
}
```