The prelab portion of this lab has to be completed at home and presented to a TA at the BEGINNING of your lab section. The prelab may contain written assignments and/or assignments to implement on your own Arduino kit. Before starting each lab, present your work papers and/or your working implementations to a TA and get their signature and keep this sheet for your records. Prelabs are INDIVIDUAL assignments. The lab portions may be worked on in groups of two.

Lab #4 – Prelab/Home Portion

1) Look up the tone() function (https://www.arduino.cc/en/Reference/Tone) and answer the following:

    a. What is the minimum frequency the function can produce **on the Arduino UNO**?

    b. What is the maximum frequency the function can produce **on the Arduino UNO**?

    c. What kind of wave does the function produce?

2) Attach the piezo buzzer to your Arduino (one pin should be attached to a PWM pin, the other to GND). Load Listing 5-1 and insure you are hearing the melody. If the volume is low, consider using a transistor switch to allow for an external voltage (9V) to drive the buzzer.

3) **Read the lab part 1 and part 2 assignments.** Draw a flowchart that describes how the program should work (the approach of reading characters from the serial port, getting the correct pattern for each incoming character, and then play the correct Morse code tones). Your flowchart should consist of a series of goals, the decisions involved in reaching each goal, and the actions taken after each decision.

Lab #4 Assignment

Part 1.Write a program that will read a single human readable character from the serial monitor and send back a string that represents the Morse code pattern for that character.

Morse Code Table (*=dit, _=dah)

Character	Pattern	Character	Pattern
A/a	* _	S/s	* * *
B/b	_ * * *	T/t	_
C/c	_ * _ *	U/u	* * _
D/d	_ * *	V/v	* * * _
E/e	*	W/w	* _ _
F/f	* * _ *	X/x	_ * * _
G/g	_ _ *	Y/y	_ * _ _
H/h	* * * *	Z/z	_ _ * *
I/i	* *		
J/j	* _ _ _		
K/k	_ * _		
L/l	* _ * *		
M/m	_ _		
N/n	_ *		
O/o	_ _ _		
P/p	* _ _ *		
Q/q	_ _ * _		
R/r	* _ *		

Some notes of how to design your program:

- Patterns can be represented as an array of characters like this:

```
char pattern[26][7] = {"*_",//A,a
                       "_***",//B,b
                          .
                          .
                          .
                       etc
```

- In order to save some typing time, we've included the declarations for the pattern array on the next page.
- The pattern array first index goes from 0 up to 25 (26 element). For example, pattern[2][]="_*_*", where pattern[2][1]= '*'.
- From the ASCII table on Page 6, you can notice the following:
  - The characters A,B,C,D,E,…,Z have the following decimal representation 65,66,67,68,69,…,90. [**Increment of 1**]
  - By subtracting 65 from each character (if it was between A and Z), the result is a number between 0 and 25 (26 character), this number can be used as the first index for the array. For the second index, use a for loop.

```
char pattern[26][7]={
"*_",  //A,a
"_***",//B,b
"_*_*",//C,c
"_**", //D,d
"*",   //E,e
"**_*",//F,f
"__*", //G,g
"****",//H,h
"**",  //I,i
"*___",//J,j
"_*_", //K,k
"*_**",//L,l
"__",  //M,m
"_*",  //N,n
"___", //O,o
"*__*",//P,p
"__*_",//Q,q
"*_*", //R,r
"***", //S,s
"_",   //T,t
"**_", //U,u
"***_",//V,v
"*__", //W,w
"_**_",//X,x
"_*__",//Y,y
"__**" //Z,z
};

const int piezoPin = 9;
char c;
int index1=0,index2=0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(100);
  Serial.println("Ready!");
  pinMode(piezoPin,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available())
  {

  // Cont on Nxt page
```

```
    c=Serial.read();
    index1=-1;
    //between A and Z
    if(c>='A'&&c<='Z')
    {
      index1=c-65;
    }
    else
    {
      //between a and z
      /*code goes here*/

      //space
      /*code goes here*/

      //new line (Enter)
      //code goes here*/

      // Not a char
      /*code goes here*/

    }

    if(index1>=0&&index1<=25)//if it's a valid character
    {
      for(index2=0;index2</*code goes here*/;/*code goes here*/)
      {
        //dit
        if(pattern[index1][index2]==/*code goes here*/)
        {
          dit();
        }
        //dah
        /*code goes here*/

        //null [Each string must end with a null -check ASCII table-]
        /*code goes here*/

        delay(300);//space between Dit and Dah
      }
     delay(100);//space between Letters
    }
  }
}
void dit()
{
  /*code goes here*/
}
void dah()
{
  /*code goes here*/
}
```

Part 2.Write a function that processes the Morse code pattern by playing the correct tones for each character and also printing the Morse code pattern to the serial monitor.

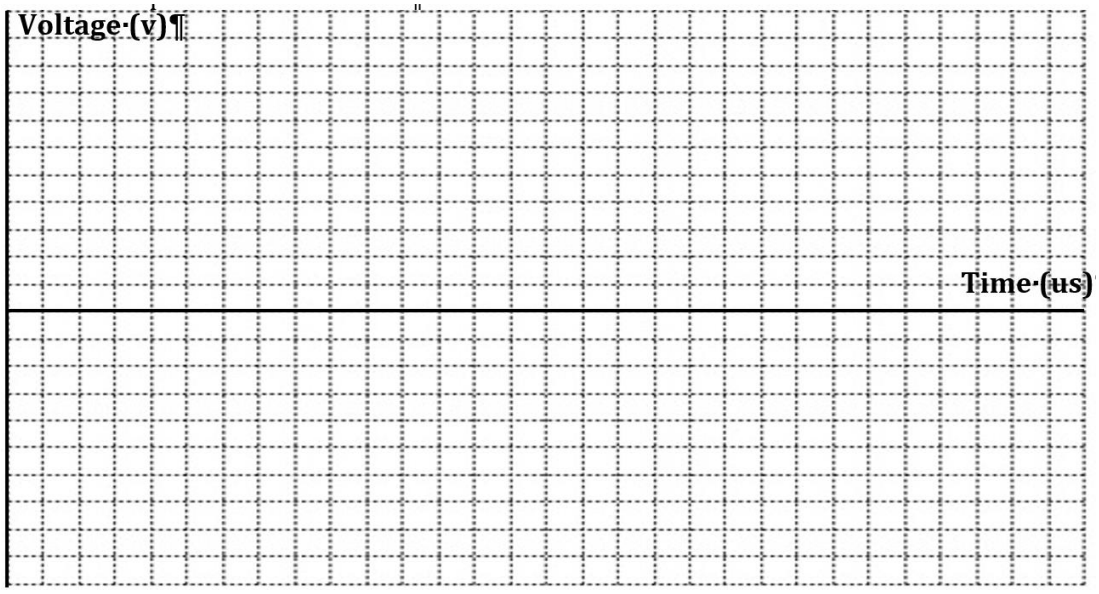
- Your program should use these values for the lengths of the delays you'll encounter in this assignment

Type	Duration
Dit (short)	40 milliseconds
Dah (long)	200 milliseconds
Space between dits and dahs	300 milliseconds
Space between letters	100 milliseconds
Space between words	700 milliseconds

- Demonstrate your code using these test strings and compare them to the results produced by this website (http://morsecode.scphillips.com/translator.html)

  a. ABC DEF GHI JKL MNO PQR STU VWX YZ
  b. SMS
  c. SOS SOS SOS SOS SOS

Part 3.Write a program that captures the transmission of a character from the Arduino over serial communication (UART) on an oscilloscope and explain the different parts of the waveform. Explain to the TA how you used the "trigger" function on the scope to capture the waveform.



Voltage Division: ……..(v) Time Division: ……..(us)

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	Escape	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	File Separator	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	Group Separator	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	Record Separator	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	Unit Separator	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	Del

asciichars.com