Lab #8
TECH 3233

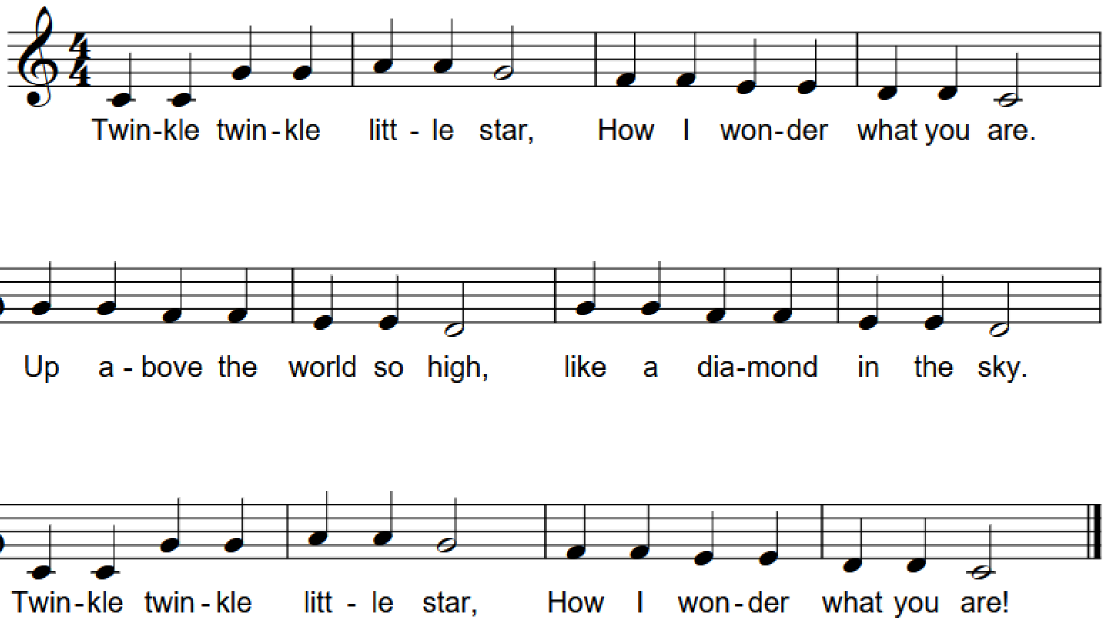
Ver 1.7

Overview of Lab:

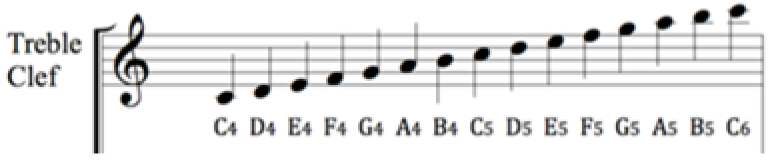
We are going to use Timer Output Compare (OCR1A) and its associated pin (PB1) to generate the song "Twinkle Twinkle Little Star". The program will adjust the frequency for each note and the time that note is played.

Here is the sheet music for the song:

Twinkle Twinkle Little Star

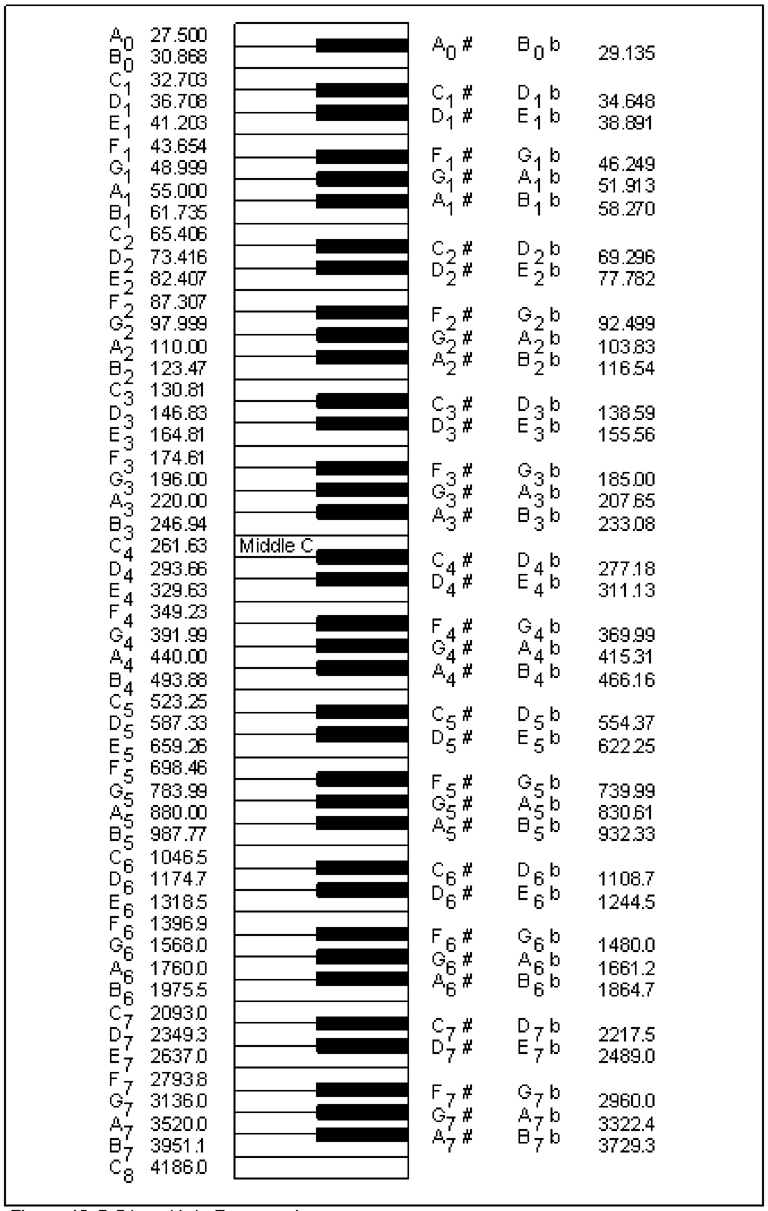

Twin-kle twin-kle litt - le star, How I won-der what you are.
Up a - bove the world so high, like a dia-mond in the sky.
Twin-kle twin - kle litt - le star, How I won-der what you are!

First we need to determine what notes are used in the song. Use the table below to mark each note in the sheet music above with the note.



So in the case of the first note (for Jeopardy) it would be an C5.

Once that is done, now you can determine the frequency (in Hz) of the signal you need to generate is for each one of the notes using the table below (all Frequencies are in Hz):

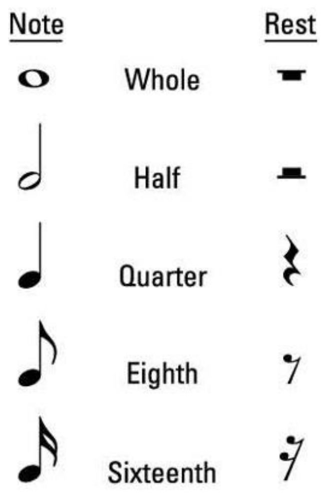


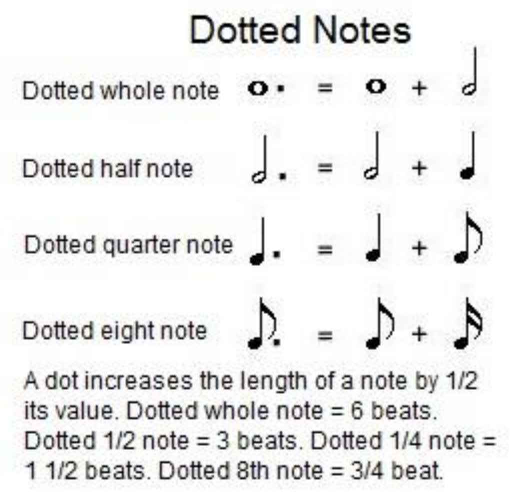
A_0	27.500
B_0	30.868
C_1	32.703
D_1	36.708
E_1	41.203
F_1	43.654
G_1	48.999
A_1	55.000
B_1	61.735
C_2	65.406
D_2	73.416
E_2	82.407
F_2	87.307
G_2	97.999
A_2	110.00
B_2	123.47
C_3	130.81
D_3	146.83
E_3	164.81
F_3	174.61
G_3	196.00
A_3	220.00
B_3	246.94
C_4	261.63
D_4	293.66
E_4	329.63
F_4	349.23
G_4	391.99
A_4	440.00
B_4	493.88
C_5	523.25
D_5	587.33
E_5	659.26
F_5	698.46
G_5	783.99
A_5	880.00
B_5	987.77
C_6	1046.5
D_6	1174.7
E_6	1318.5
F_6	1396.9
G_6	1568.0
A_6	1760.0
B_6	1975.5
C_7	2093.0
D_7	2349.3
E_7	2637.0
F_7	2793.8
G_7	3136.0
A_7	3520.0
B_7	3951.1
C_8	4186.0

A_0#	B_0b	29.135
C_1#	D_1b	34.648
D_1#	E_1b	38.891
F_1#	G_1b	46.249
G_1#	A_1b	51.913
A_1#	B_1b	58.270
C_2#	D_2b	69.296
D_2#	E_2b	77.782
F_2#	G_2b	92.499
G_2#	A_2b	103.83
A_2#	B_2b	116.54
C_3#	D_3b	138.59
D_3#	E_3b	155.56
F_3#	G_3b	185.00
G_3#	A_3b	207.65
A_3#	B_3b	233.08
C_4#	D_4b	277.18
D_4#	E_4b	311.13
F_4#	G_4b	369.99
G_4#	A_4b	415.31
A_4#	B_4b	466.16
C_5#	D_5b	554.37
D_5#	E_5b	622.25
F_5#	G_5b	739.99
G_5#	A_5b	830.61
A_5#	B_5b	932.33
C_6#	D_6b	1108.7
D_6#	E_6b	1244.5
F_6#	G_6b	1480.0
G_6#	A_6b	1661.2
A_6#	B_6b	1864.7
C_7#	D_7b	2217.5
D_7#	E_7b	2489.0
F_7#	G_7b	2960.0
G_7#	A_7b	3322.4
A_7#	B_7b	3729.3

Figure 13-5. Piano Note Frequencies

The last piece of information is how long each note needs to be held for. This is determined by the type of note as shown in the sheet music as per the note types below:




Dotted Notes
Dotted whole note
Dotted half note
Dotted quarter note
Dotted eight note
A dot increases the length of a note by 1/2 its value. Dotted whole note = 6 beats. Dotted 1/2 note = 3 beats. Dotted 1/4 note = 1 1/2 beats. Dotted 8th note = 3/4 beat.

Note since a whole note is 8x as long as an eighth note, and an eighth note is the shortest note in the piece, it is a good idea to set a constant for the time of a the quarter note, so if the song is going too fast or too slow, you can change it in one place to change the speed of the song.

Now, knowing the frequencies you need to generate, figure out what pre scaler you need for the Atmel238p timer.

Calculate the number of counts needed to generate the 50% duty cycle square wave necessary to generate the frequency of each note.

Create two arrays, one for the timer counts and one for the length (1=eighth note, 2=quarter note, 4=half note and 8 being full note). It is a good idea, for readability of the code, to define the counts for each note and use that name when filling the array:

```
#define c6 1568
#define d6 1174

freq[4]={c6,d6,c6,c6};
time[4]={2,4,4,1};
```

NOTE – for the above example the notes in the above were defined as Hz not as counts!

Lastly, since the note to play (ie the frequency to generate) automatically gets generated by the Output Compare of the Atmel IC, we just have to pause the program for how long the note needs to play. It would be easy to just do a _delay_ms(tempo*time) to do this, but UNFORTUNATELY, the way Atmel Studio does the _delay_ms() function, it must be a constant in the ( ). So you need to make your own function that does the base (tempo) time delay for a number of times then returns to solve this issue. So create a function that does this and name it t_delay and pass it an int.

To test, connect the Arduino output to a set of computer speakers (Gnd on Arduino to Ground on the 3.5mm Jack and PB1 to the Right or Left Input on the 3.5mm jack).

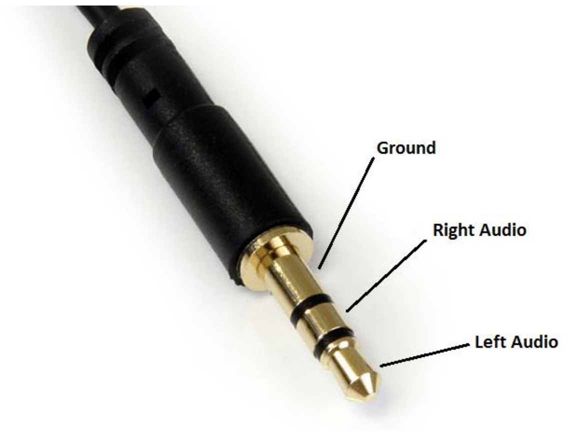


Figure 1 - 3.5mm Jack Connections