# Project: Making Music with Code



PROJECTS WITH THE TI-INNOVATOR™ SYSTEM - TI-NSPIRE CX

## Overview:

Students will write a TI-BASIC program on their calculator to create sounds using the built-in speaker on the TI-Innovator Hub. They will also experiment with changing the length of time in which each sound plays. Once they have some experience, they will write a program to play a song.

## Goals:

Students will:

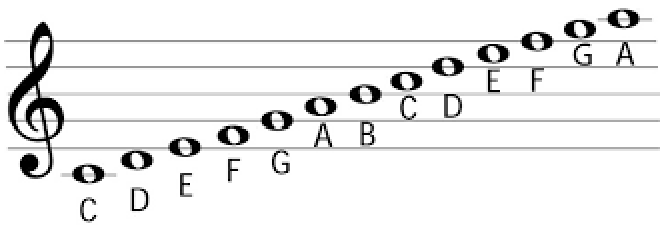
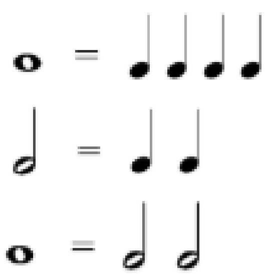
- learn to begin, open, close, edit and run programs from the editor.
- write a TI BASIC program that controls the built-in speaker on the TI-Innovator Hub
- write a program that controls how long a sound will play.

**Note** – A good foundation before beginning this project can be found at http://education.ti.com/en/activities/ti-codes/nspire/10-minutes-innovator.
Go to Unit 1: Getting Started with TI-Innovator Hub and then “Skill Builder 3 - Request and SOUND”. Also, Unit 2: For Loops with TI-Innovator Hub and then “Skill Builder 3 Loop through the musical notes” has related concepts that may be helpful.

## Background:

| Command | Example | Behavior |
|---|---|---|
| SET SOUND <frequency> TIME <duration> | Send “SET SOUND 440 TIME 4” | Plays a 440 hertz tone on the speaker for 4 seconds |
| DispAt <line #> , <”text”> , <variable name> | DispAt 3, “Note = 440 Hz” | Displays the message “Note = 440 Hz” on calculator. A variable value may also be displayed with the text message. For example DispAt 3,” Note =“,n will display the same output if the value of 440 is stored in the variable named n. |
| For <counter variable>,<start value>,<end value>,[<step value>]<br><statements><br>EndFor | For n,1,10<br><br>EndFor | Runs the statements within the For/EndFor loop 10 times |
| eval(<variable name>) | n:=440<br>Send “SET SOUND eval(n)” | When eval is used within a Hub command, the variable value is inserted into the Hub command. In the example, the value for variable n is 440. The next command then is sent as Send “SET SOUND 440” and a tone is played on the Hub. |
| Wait <duration> | Wait 2 | Causes program execution to halt and wait for two seconds. |
| ©<Information that you want to note but does not affect the program> | ©D5 | The © symbol is used as a “comment”. A comment is meant to leave notes within the program for easier navigation later or to let the user know what the following line of code is used for. Anytime there is a © in a program, that line of information is NOT part of the program. It is ignored by the system and is only there for reference. |

See TI-Innovator Technology eGuide for more background https://education.ti.com/html/webhelp/EG_Innovator/EN/index.html

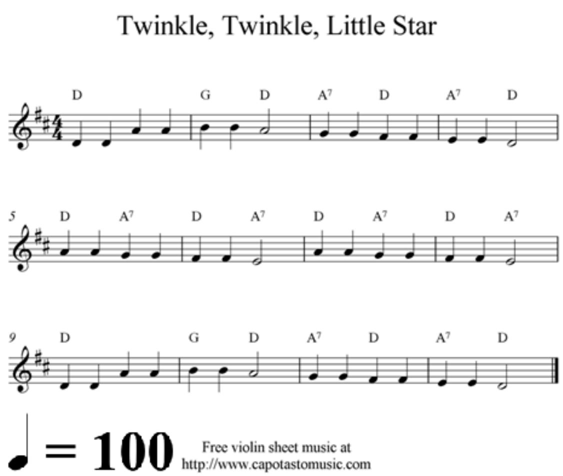
| Music Concepts | Explanation |
|---|---|
| Musical Scale   | Music is arranged with symbols on a “scale”. We use a scale since the notes are divided by “octaves”. An octave represents eight steps between the equivalent note that is at either double or half the frequency. For example, A4 is 440 hertz (Hz), while A5 is 880 Hz (twice the value of A4), while A3 is 220 Hz (half of A4). |
| Beats per minute/Tempo | Have you ever noticed how some songs are slow while others are fast? Every song has its own “tempo” or speed. The speed is determined by the number of “beats per minute”. A slow song may have 50 beats per minute while a faster song may have 100 beats per minute. |
| Quarter, half, and whole notes  | Notes can be shorter or longer depending on the song. One way for a musician to write a song with notes of different durations is to use whole notes, half notes, quarter notes, eighth notes, and sixteenth notes. One whole note equals four quarter notes or two half notes. One half note is equal to two quarter notes and so on. For a song that has 4-4 measures, it means that each measure of the song has 4 beats. A whole note would be equal to 4 beats (hence, one measure). How many half notes could fit into a 4-4 measure? If you said two, you are correct! Since each half note is equal to two beats and there are four total beats per measure, two half notes is equal to four beats. So now you can see how a song with 100 beats per minute will have faster whole notes than a song with 50 beats per minute! Not all whole notes equal the same time duration! |

# Project: Making Music with Code



**PROJECTS WITH THE TI-INNOVATOR™ SYSTEM - TI-NSPIRE CX**

Application of notes and beats per minute




Twinkle, Twinkle, Little Star is a song that has multiple notes and uses 4-4 time. Try to code this song with your calculator.

**Note** - Another useful resource for understanding musical concepts, frequencies and notes can be found

http://www.sengpielaudio.com/calculator-notenames.htm

| **Setup Project:** | **Supplies:** |
|---|---|
| Students may work in groups of two or three. | • Calculator<br>• Unit to Unit Cable<br>• Hub |

| **The Student Activity** | **The Teacher Activity** |
|---|---|
| Sit in small groups with your calculator and supplies for this activity. Practice the guidance modeled by your teacher. | Review and introduce the calculator and Hub commands needed for this activity.<br>• Start a new document<br>• Insert a program<br>• Attach Hub |
| **Challenge 1:** Create a program called “**c1**”. Set the speaker on the Hub to play a tone of 440 Hz. Try several different tones. What is the lowest tone you can hear? | **Guidance during challenge 1:**<br>• Use the DispAt command (Program I/O menu). DispAt takes at least two arguments. The first argument is the row number on which to display. The second argument is the text string, value, variable, etc. to display. |

What is the highest?

- Example program

```
Define c1()=
Prgm
Send "SET SOUND 440 TIME 2"
DispAt 3,"This is 440Hz for 2 Seconds”
EndPrgm
```

**Challenge 2:** Create a program called “**c2**”. Make a door bell using any key press on the calculator to ring the chime.

**Guidance during challenge 2:**

- By the way, most common door bells sound two tones that are a perfect 4th apart. An example is D#5 and G5.
- Example program

```
Define c2()=
Prgm
© G5
Send “SET SOUND 784”
Wait 1
© D#5
Send “SET SOUND 622”
EndPrgm
```

**Challenge 3:** Write a program called “**c3**” that plays the Major C scale from C4 to C5. Each note should be played for a variable length of time using a variable named “duration”.

**Guidance during challenge 3:**

- Example program

```
Define c3()=
Prgm
duration:=0.1
© C4
Send “SET SOUND 262 TIME eval(duration)”
Wait duration
© D4
Send “SET SOUND 294 TIME eval(duration)”
```

```
Wait duration
© E4
Send "SET SOUND 330 TIME eval(duration)"
Wait duration
© F4
Send "SET SOUND 349 TIME eval(duration)"
Wait duration
© G4
Send "SET SOUND 392 TIME eval(duration)"
Wait duration
© A4
Send "SET SOUND 440 TIME eval(duration)"
Wait duration
© B4
Send "SET SOUND 494 TIME eval(duration)"
Wait duration
© C5
Send "SET SOUND 523 TIME eval(duration)"
Wait duration
EndPrgm
```

**Challenge 4:** Create a program called "**c4**". Make a metronome that: plays 10 beats with a tone of middle C4 with a duration of a ¼ note at 100 Beats per minute. To do this you will need to determine the how long a quarter note is (in seconds) based on the information above.

A rest of .1 seconds should be used

**Guidance during challenge 4:**

- Loops are used to repeat a set of commands. A "For loop" repeats a specified number of times. In TI-Basic the programmer defines a For loop with four arguments: a counter variable, a beginning value for the counter variable, an ending value for the counter variable and a step value.
- Students will need to determine the length of time per beat at 100 beats per minute for a quarter note (which is one beat). They can do this by dividing 60 seconds/minute by 100 beats per minute, which equals 0.6 seconds/beat.
- Example program

```
Define c4()=
```

between beats. Try a double time tempo of 200 BPM. Try using a 1/2 or whole note.

*Use the file “88 Piano Keys” utility found in 88 Piano Keys.tns file or create tables to convert notes to duration and frequency.

```
Prgm
© ¼ note at 100BPM
duration:=0.6
For n,1,10
 Send "SET SOUND 262 TIME eval(duration)
 Wait duration
 Wait 0.1
EndFor
EndPrgm
```

**Challenge 5:** Create a program called “**c5**”. Use the sheet music for Twinkle to play the first two bars of the song on your Hub.

*Use the file “88 Piano Keys” utility found in 88 Piano Keys.tns file or create tables to convert notes to duration and frequency.

**Guidance during challenge 5:**

- Example program

```
Define c5()=
Prgm
Send "SET SOUND 261 TIME .6"
Wait 0.6
Wait 0.1
Send "SET SOUND 261 TIME .6"
Wait 0.6
Wait 0.1
Send "SET SOUND 392 TIME .6"
Wait 0.6
Wait 0.1
Send "SET SOUND 392 TIME .6"
Wait 0.6
Wait 0.1
Send "SET SOUND 440 TIME .6"
Wait 0.6
Wait 0.1
Send "SET SOUND 440 TIME .6"
Wait 0.6
```

```
Wait 0.1
Send "SET SOUND 392 TIME 1.2"
Wait 1.2
Wait 0.1
EndPrgm
```

**Challenge 6:** Create a program called "**c6**". Use two lists with frequencies and durations and a loop to play the same Twinkle song.

**Guidance during challenge 6:**

- Lists variables are a set of values. On the TI-Nspire the values are separated by commas and enclosed by braces. The dim(<listname>) function returns the length (dimension) of a list. TI-Nspire list values can be accessed by the listname followed the by the position in the list within brackets starting at 1. For example, *notes*[1] would return the first value of the list variable *notes*.
- Example program

```
Define c6()=
Prgm
© lists with notes and durations
notes:={261,261,392,392,440,440,392}
durations:={0.6,0.6,0.6,0.6,0.6,0.6,1.2}
For n,1,dim(notes)
 Send "SET SOUND eval(notes[n]) TIME eval(durations[n])"
 Wait durations[n]
 Wait 0.1
EndFor
EndPrgm
```

**Final Challenge:** Find the sheet music for a simple song and play it on the Hub.

**Guidance during Final Challenge:**

- Have the students use the skills that they have built from the previous challenges.
- Develop a program that can play a well-known song.
- Example Program

```
Define Final Challenge()=
Prgm
```

```
© make lists with notes and durations
notes:={--,--,--,--,--,--,--}
durations:={--,--,--,--,--,--,--}
For n,1,dim(notes)
 Send "SET SOUND eval(notes[n]) TIME eval(durations[n])"
 Wait durations[n]
 Wait 0.1
EndFor
EndPrgm
```