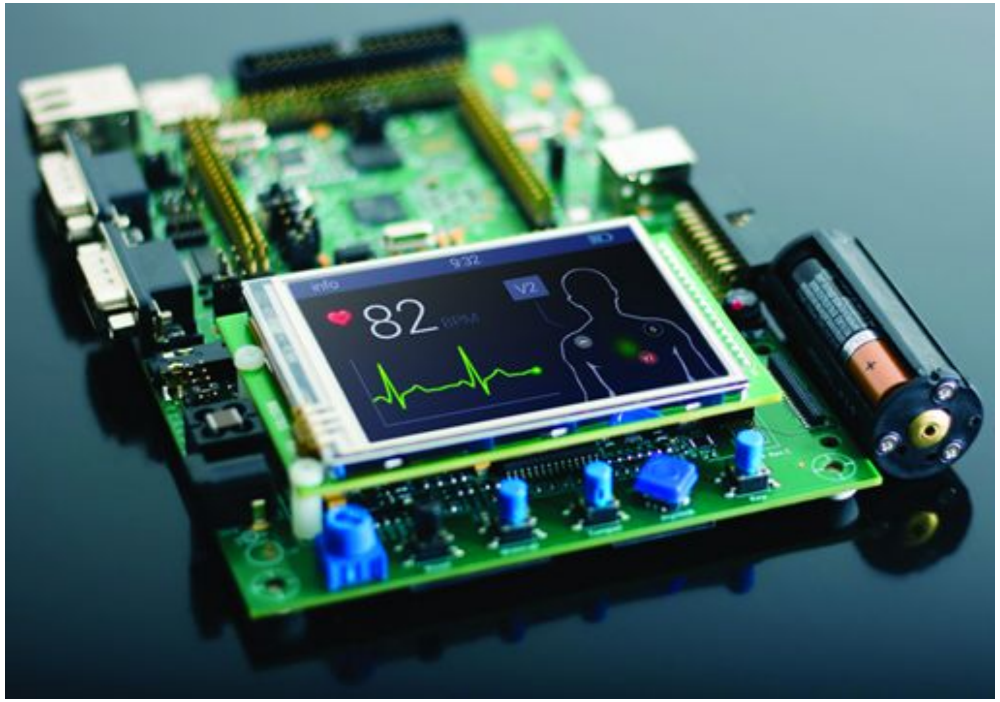
Embedded Software

CS 145/145L



Caio Batista de Melo

Announcements (2022-04-28)

- Project 2 is due tomorrow!
- Mid-quarter course evaluation is due on Saturday for extra credit on Project 3
    - Completely anonymous, please provide your honest feedback :)
    - Will replace the early submission extra credit for P3
    - https://evaluations.eee.uci.edu/takeLanding/WTWYYF

Project 3

Design an embedded computer centered around the ATMega32 microcontroller.
For input: use a keypad;
For output: use an LCD and a speaker.

Write a C program that implements a music player. Your music player should be able to play musical notes stored in its memory.

https://canvas.eee.uci.edu/courses/45047/assignments/929272

Project 3 is short! It’s due next week (2022-05-06)!!

Project 3 - Grading

Basic requirements (100%)

- Plays a sound which is neither croaky nor severely distorted. (We understand this is digital music). Player should support start/stop through button press. (65% for functionality + 30% for quality).
- LCD displays the name of the song currently playing (5%)

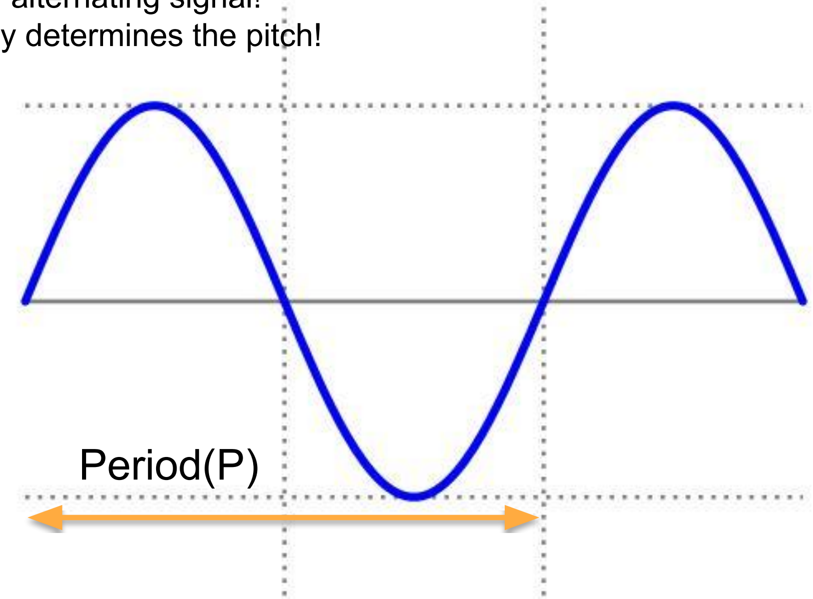
Extra credits (20%)

- Implements pitch control (at least 3 levels) (5%)
- Implements tempo control (at least 3 levels) (5%)
- Plays multiple songs, and supports user selection between each song (5%)
- Complete the Mid-quarter evaluation (5%)

https://canvas.eee.uci.edu/courses/45047/assignments/929272

Intro

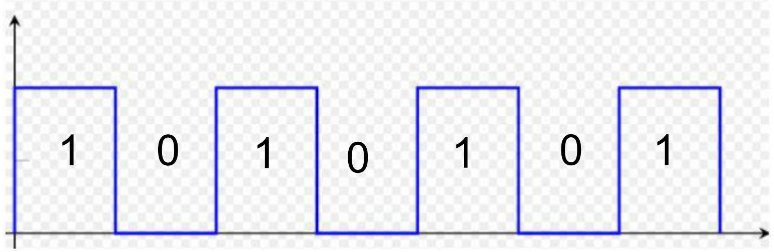
Sound is an alternating signal!
Its frequency determines the pitch!

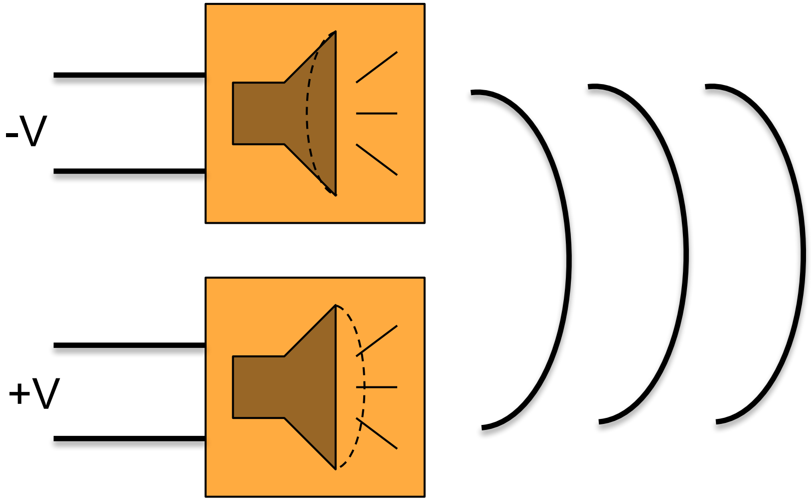


Frequency(F) = 1/P

20Hz to 20 KHz

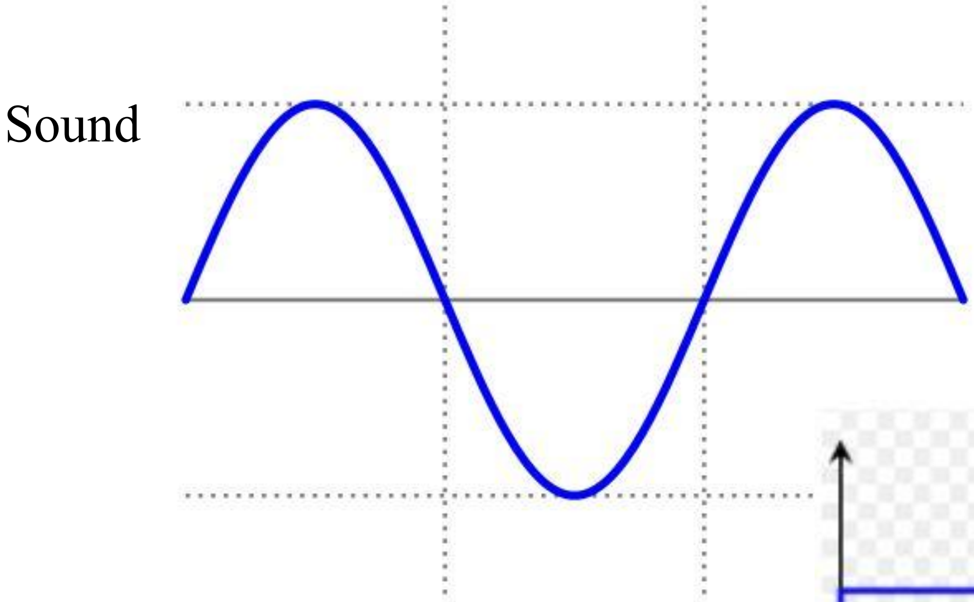
We want to generate a simple sine wave

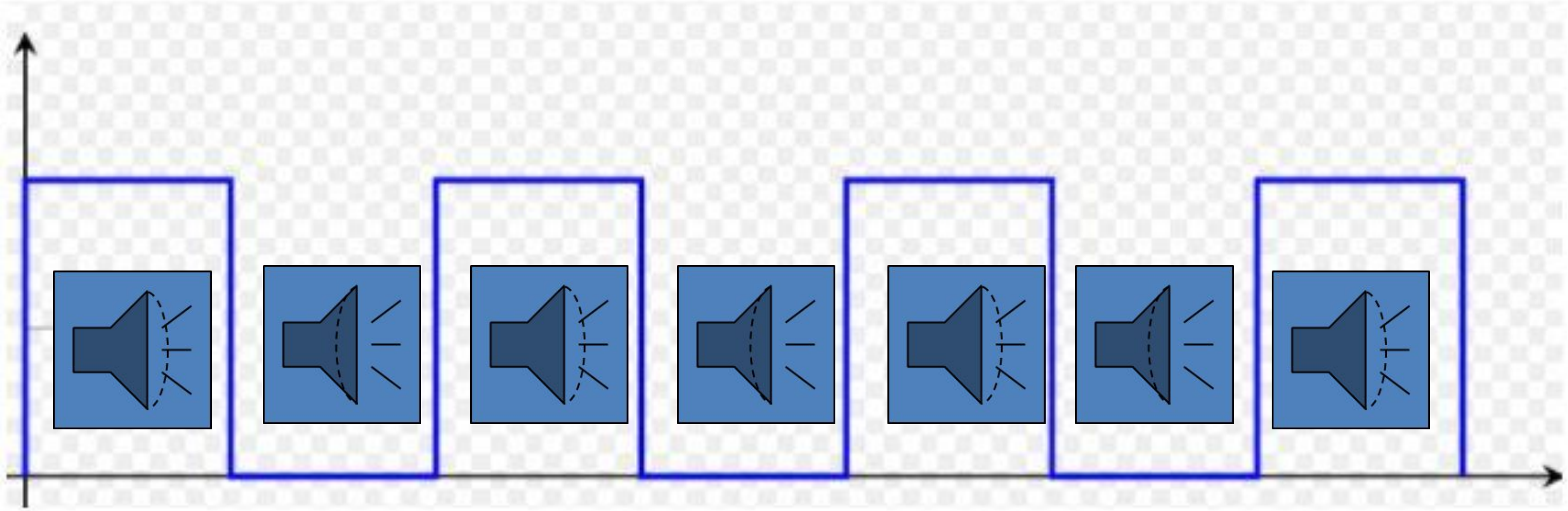




- The speaker's whole diaphragm changes according to the voltage applied.
- Thus vibration of pressure (technically) or sound is generated by alternating this voltage.
- But our AVR cannot provide a purely analog signal as shown
- We have voltages in the digital nature in the form of 0s and 1s

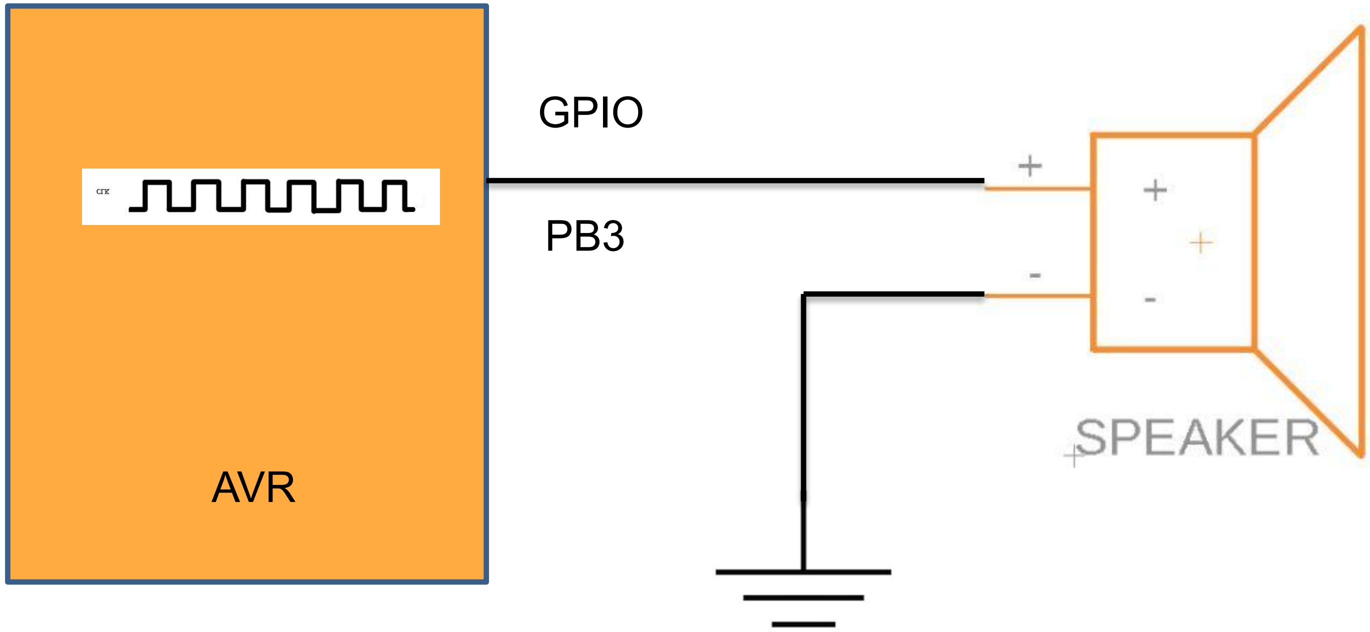
Use of Digital Signal





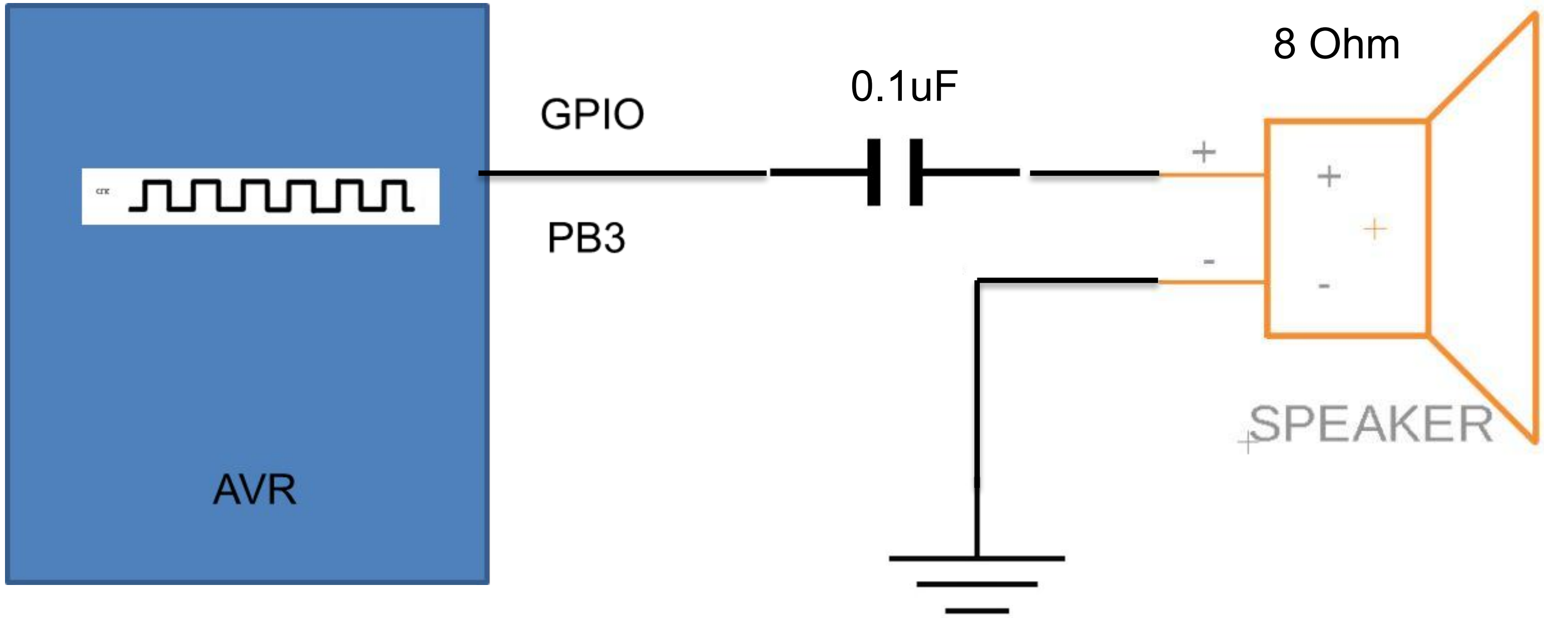
The momentum of diaphragm’s motion will help it oscillate.

Initial Layout



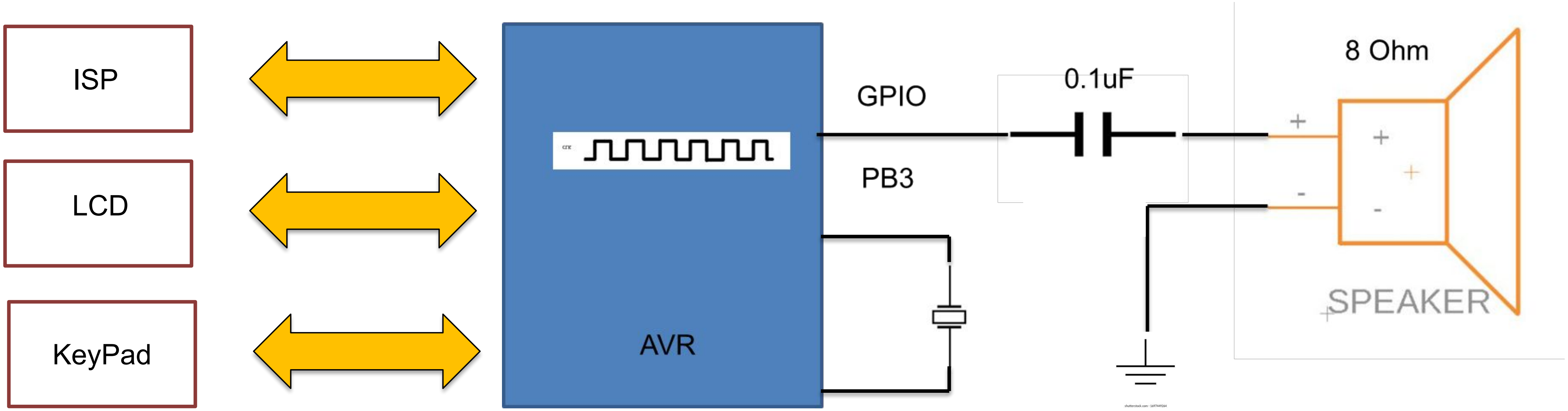
shutterstock.com · 1697449264

Some improvements in the Layout



- The speaker has an impedance of its own
- The capacitor charges during the positive cycle and the charging speed is decided by the R and C combination
- In the negative cycle or 0 cycle in digital terms, the capacitor discharges again as per the R and C network
- Thus smoothening of the square wave takes place

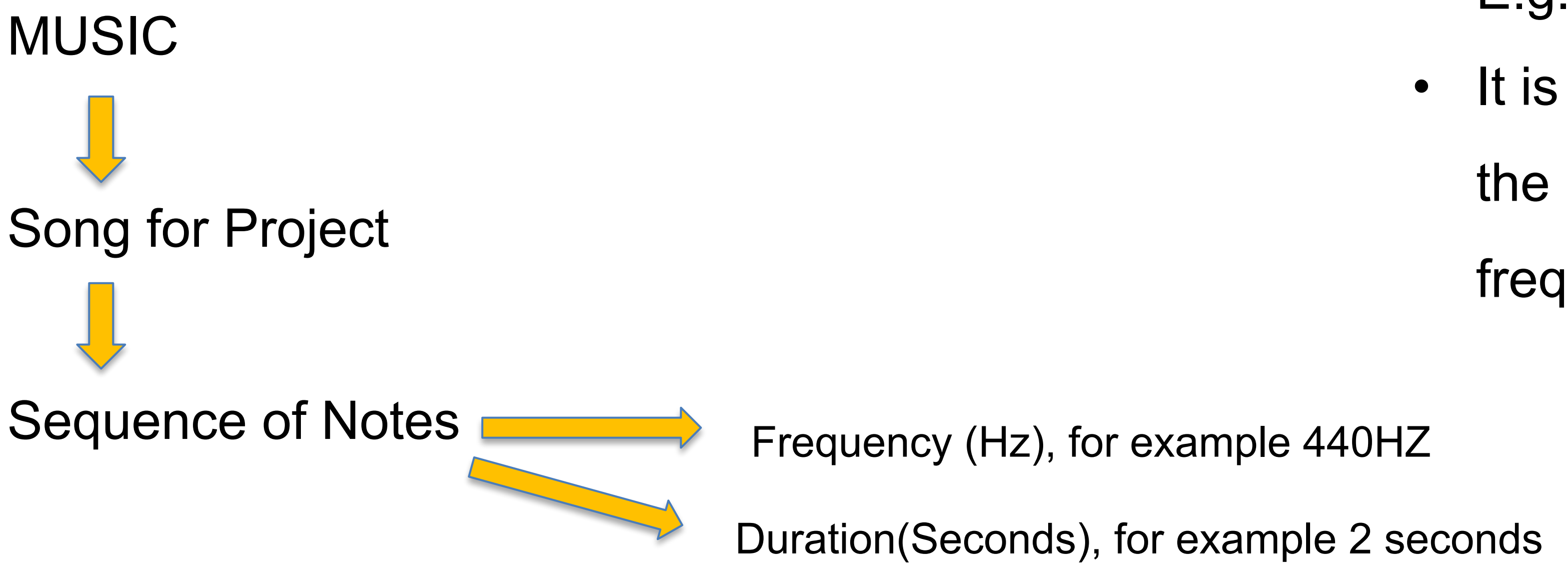
Final Layout



For extra credit you probably need a keypad.
Standard project could use a single button.

Song for Project 3

- Notes can be defined as a combination of frequency and duration
- Musicians abstract this out using symbols.
  - E.g.,: A, ½ Time
  - It is believed that most of the musics on the planet can be played using 12 frequencies and their variations.



MUSIC

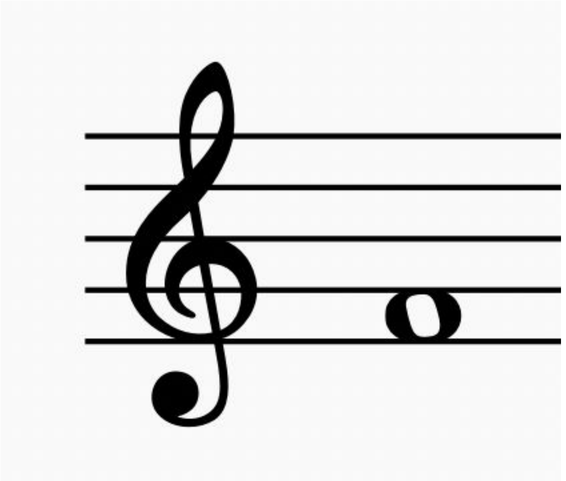
Song for Project

Sequence of Notes

Frequency (Hz), for example 440HZ

Duration(Seconds), for example 2 seconds

Decoding Notes

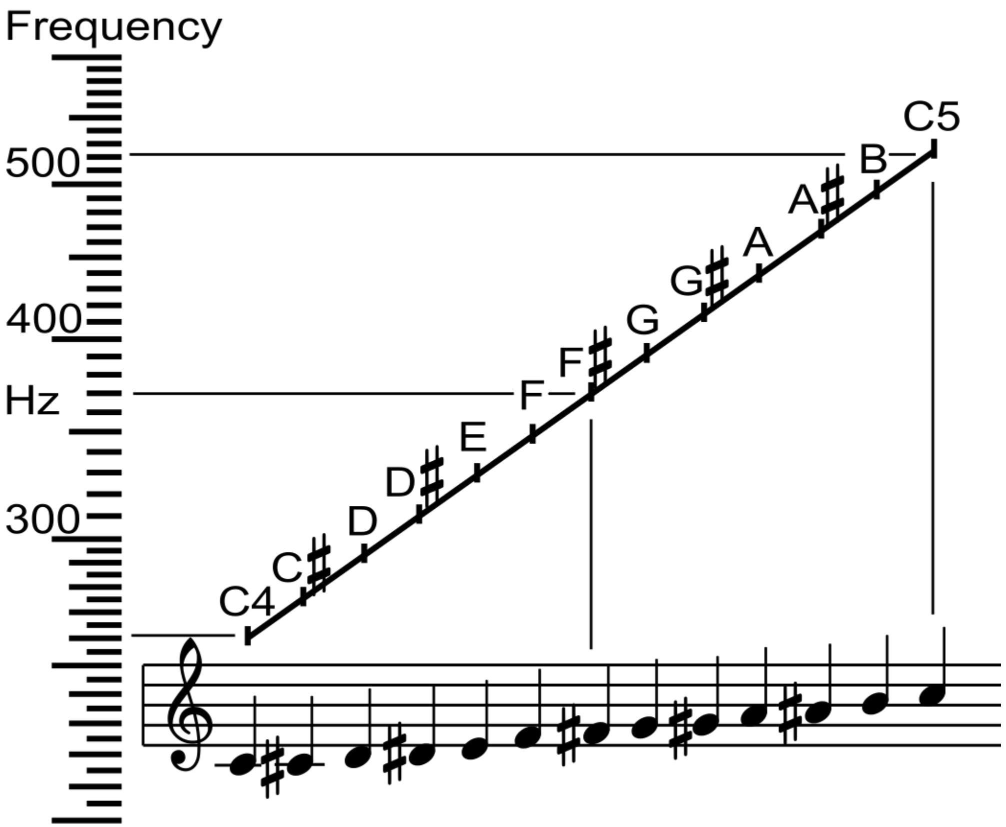


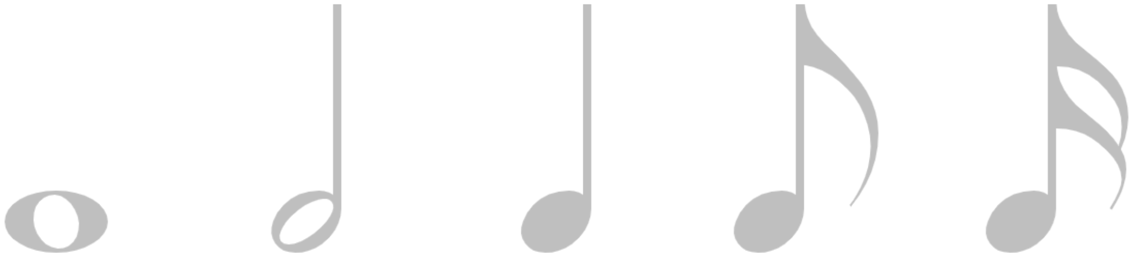
(440 Hz, 1 sec)

(466 Hz, 2 sec)

(490 Hz, 0.5 sec)

Musical Notes Resources

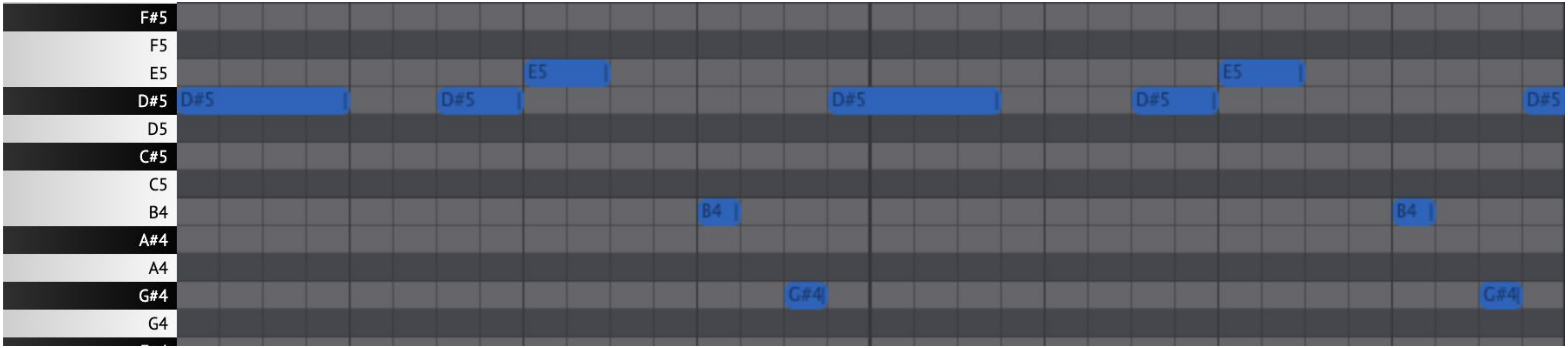




https://www.musictheory.net/lessons/11

https://en.wikipedia.org/wiki/Musical_note
https://www.szynalski.com/tone-generator/

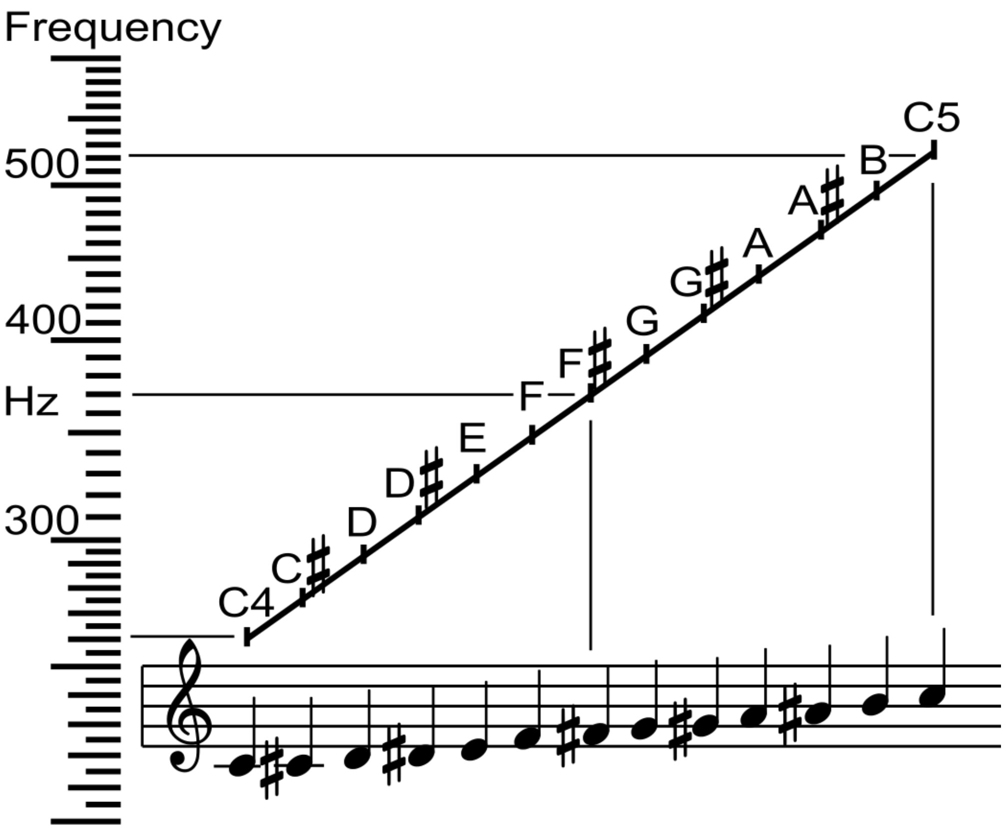
Example Music



https://onlinesequencer.net/433516

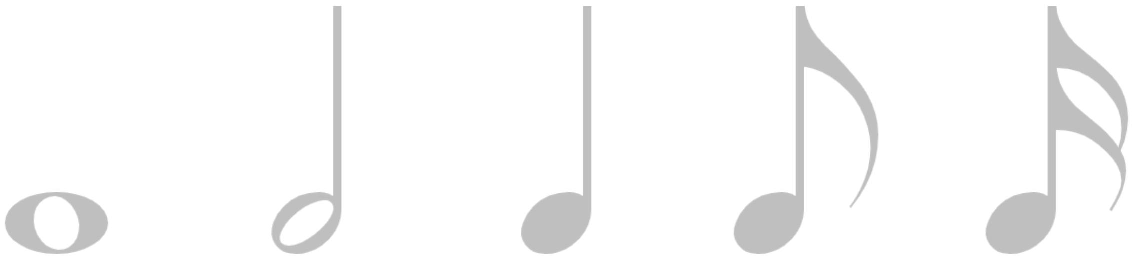
HINT: You can try searching for the song you want + “midi” to try to find a note sequence.
Example: to find the above one, I searched for “shooting stars midi”
https://www.google.com/search?q=shooting+stars+midi

Code Layout



```
typedef enum {
  A, As, B, C, Cs, D, Ds, E, F, Fs, G, Gs
} Note;
```

Code Layout



```
typedef enum {
   W, H, Q, E
} Duration;
```

Code Layout

```c
typedef enum {
  A, As, B, C, Cs, D, Ds, E, F, Fs, G, Gs
} Note;

typedef enum {
   W, H, Q, E
} Duration;
```

```c
typedef struct {
  Note note;
  Duration duration;
} PlayingNote;
```

Anything wrong?
Two enums with the same value!

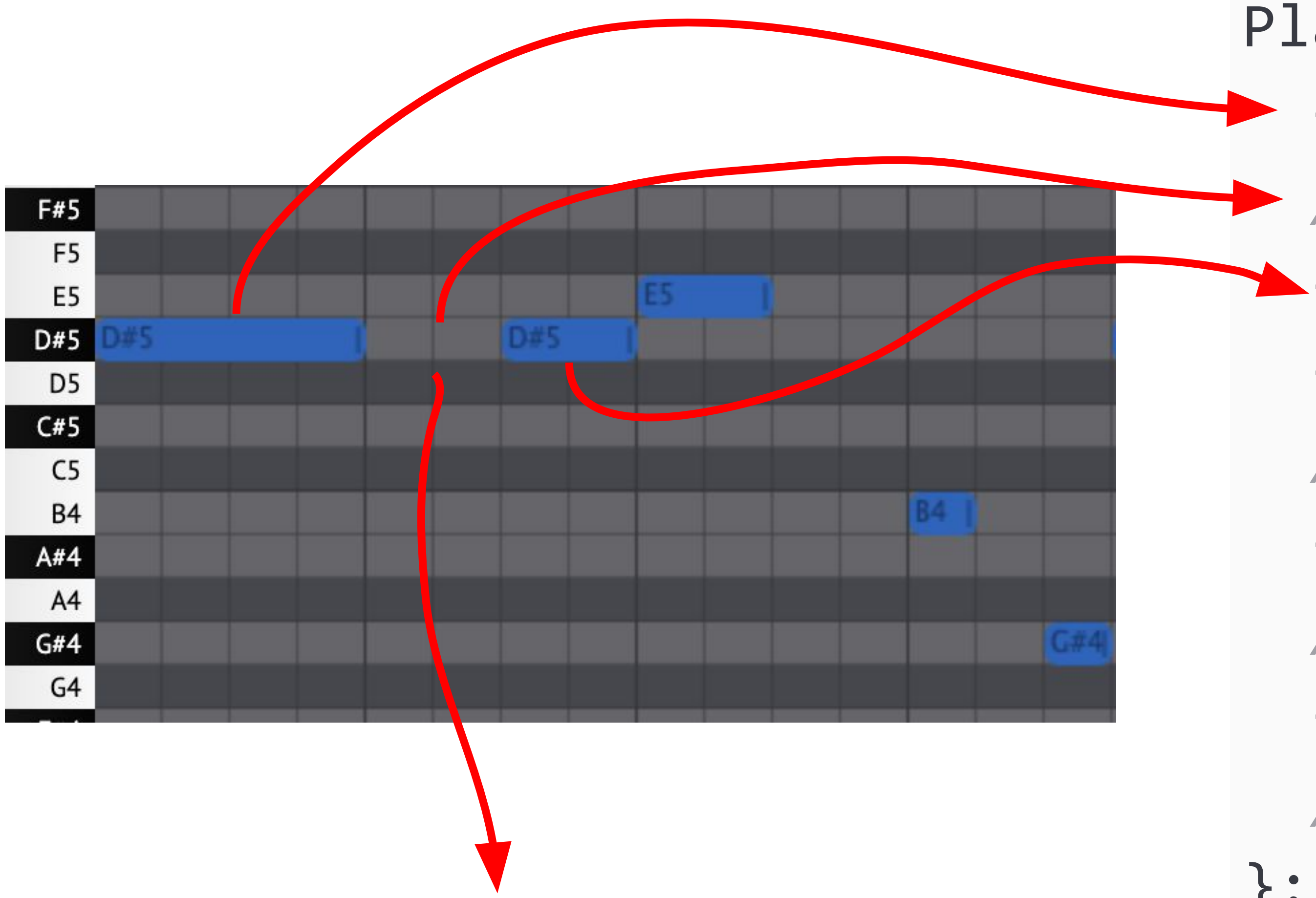
Code Layout

```
typedef enum {
  A, As, B, C, Cs, D, Ds, Ee, F, Fs, G, Gs
} Note;

typedef enum {
   W, H, Q, Ei
} Duration;
```

```
typedef struct {
  Note note;
  Duration duration;
} PlayingNote;
```

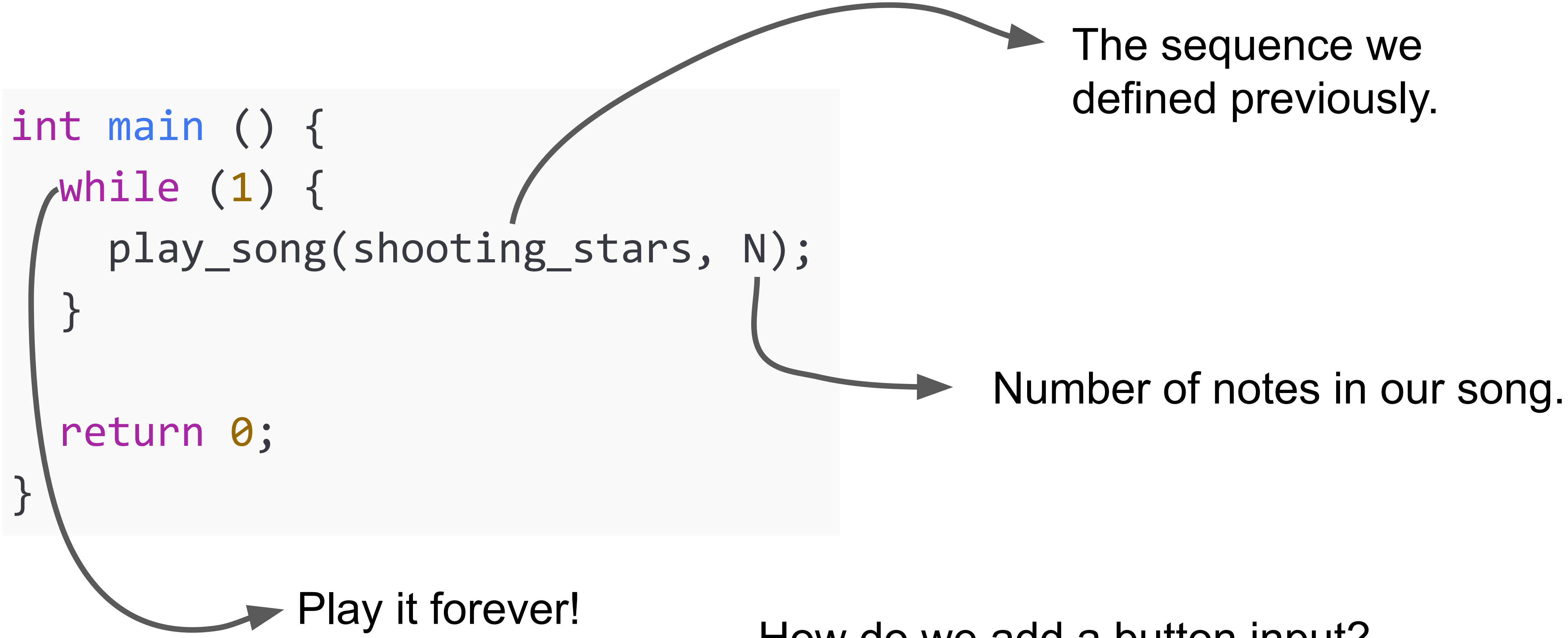
Example



```
PlayingNote shooting_stars[] = {
  {Ds, W},
  /* Wait for half */
  {Ds, H},
  {Ee, H},
  /* Wait for half */
  {B, Q},
  /* Wait for quarter */
  {Gs, Q}
  /* Keep going... */
};
```

How can you wait?

Main Function



```
int main () {
  while (1) {
    play_song(shooting_stars, N);
  }

  return 0;
}
```

The sequence we defined previously.

Number of notes in our song.

Play it forever!

How do we add a button input?
What changes are needed for multiple songs?

play_song function

```
void play_song(const PlayingNote song[], int length) {
  int i;
  for (i = 0; i < length; i++) {
    play_note(&song[i]);
  }
}
```

Can we do a loop like the one for strings?

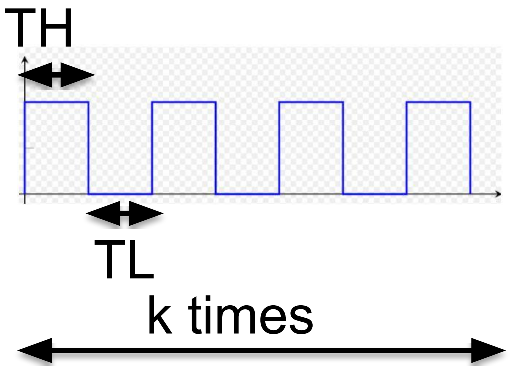
*while (note = *song++)*

Why not?

play_note function

```
void play_note(const PlayingNote* note) {
  int i, k;
  for (i = 0; i < k; i++) {
    SET_BIT(PORTB, 3);
    wait(TH);
    CLR_BIT(PORTB, 3);
    wait(TL);
  }
}
```

Create *k* ups and downs



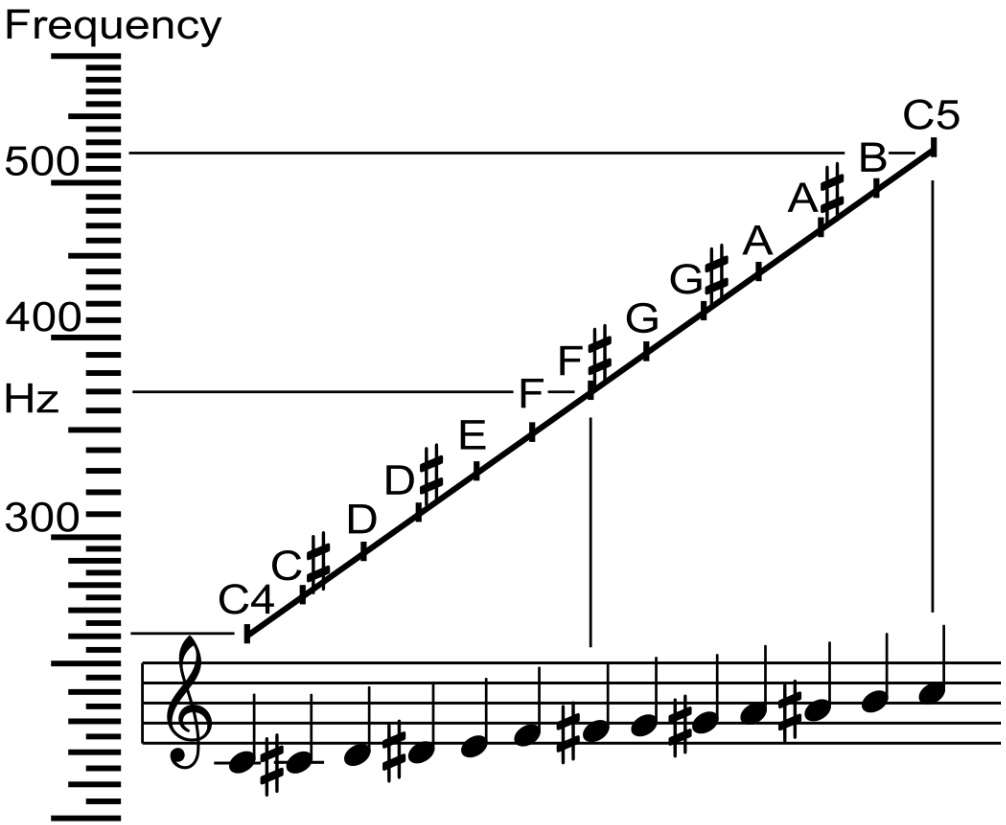
F = 1 / P *(you know F)*

P = TH + TL

TH = TL

k = Duration / P

Notes Frequencies



https://en.wikipedia.org/wiki/Musical_note

Notes Frequencies

Note	Offset	Frequency (Hz)	Period (s)	TH / TL (s)	Wait (1ms resolution)
A	0	440.00	0.002272727273	0.001136363636	1
A#	1	466.16	0.002145168892	0.001072584446	1
B	2	493.88	0.002024769814	0.001012384907	1
C	3	523.25	0.001911128216	0.000955564108	0
C#	4	554.37	0.001803864832	0.000901932415	0
D	5	587.33	0.001702621678	0.000851310839	0
D#	6	622.25	0.001607060866	0.000803530433	0
E	7	659.26	0.001516863471	0.000758431735	0
F	8	698.46	0.001431728466	0.000715864232	0
F#	9	739.99	0.001351371722	0.000675685860	0
G	10	783.99	0.001275525055	0.000637762527	0
G#	11	830.61	0.001203935334	0.000601967667	0

Cannot tell them apart!

Maybe we can have a finer timer?

Notes Frequencies - Finer Timer Resolution

Note	Offset	Frequency (Hz)	Period (s)	TH / TL (s)	Wait (0.1ms resolution)
A	0	440.00	0.002272727273	0.001136363636	11
A#	1	466.16	0.002145168892	0.001072584446	11
B	2	493.88	0.002024769814	0.001012384907	10
C	3	523.25	0.001911128216	0.000955564108	10
C#	4	554.37	0.001803864832	0.000901932415	9
D	5	587.33	0.001702621678	0.000851310839	9
D#	6	622.25	0.001607060866	0.000803530433	8
E	7	659.26	0.001516863471	0.000758431735	8
F	8	698.46	0.001431728466	0.000715864232	7
F#	9	739.99	0.001351371722	0.000675685860	7
G	10	783.99	0.001275525055	0.000637762527	6
G#	11	830.61	0.001203935334	0.000601967667	6

Still can’t tell some apart!

Maybe we can change frequencies?

Notes Frequencies - Down an Octave

Note	Offset	Frequency (Hz)	Period (s)	TH / TL (s)	Wait (0.1ms resolution)
A	0	220.00	0.004545454545	0.002272727273	23
A#	1	233.08	0.004290337785	0.002145168892	21
B	2	246.94	0.004049539628	0.002024769814	20
C	3	261.63	0.003822256433	0.001911128216	19
C#	4	277.18	0.003607729664	0.001803864832	18
D	5	293.66	0.003405243357	0.001702621678	17
D#	6	311.13	0.003214121733	0.001607060866	16
E	7	329.63	0.003033726941	0.001516863471	15
F	8	349.23	0.002863456932	0.001431728466	14
F#	9	369.99	0.002702743443	0.001351371722	14
G	10	392.00	0.00255105011	0.001275525055	13
G#	11	415.30	0.002407870669	0.001203935334	12

Can tell ~~all~~ most of them apart!

For our use-case it's *probably* ok :)

But you could use a finer resolution!

How do you get these values in your code?

Frequency Mapping

Note	Offset	Frequency (Hz)
A	0	220.00
A#	1	233.08
B	2	246.94
C	3	261.63
C#	4	277.18
D	5	293.66
D#	6	311.13
E	7	329.63
F	8	349.23
F#	9	369.99
G	10	392.00
G#	11	415.30

1. Store only the original frequency (220Hz) and use the formula (2^(n/12) * 220);

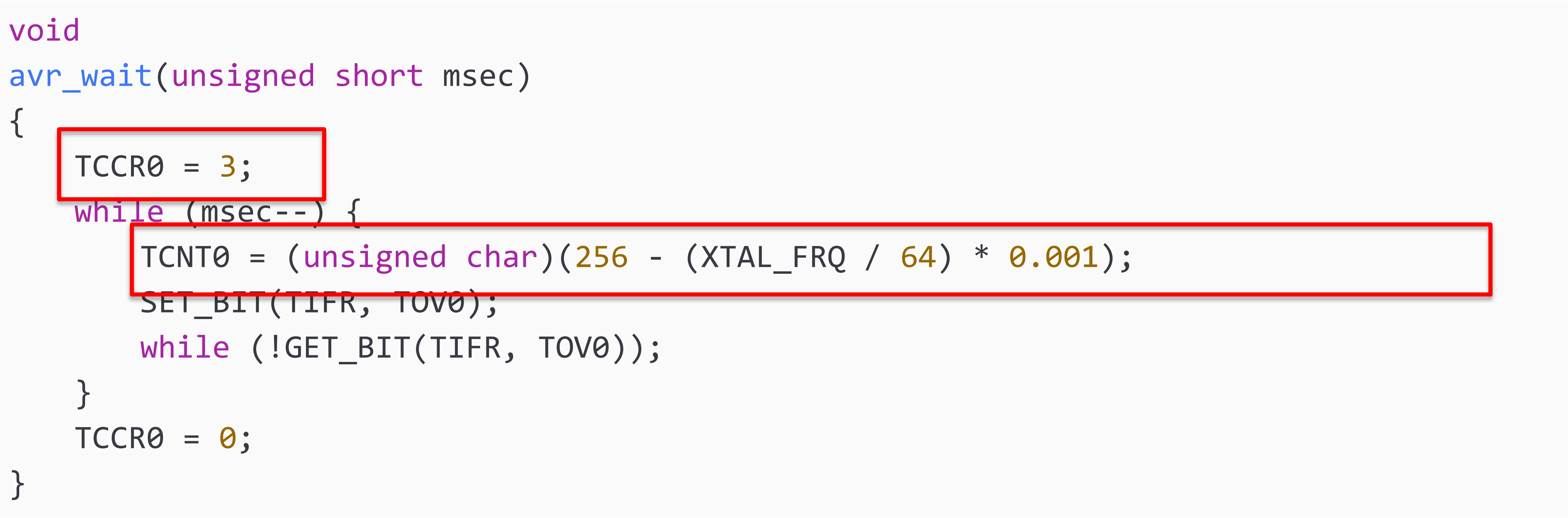
or

2. Store these values as constants and use them as needed
   a. Could also store period, TH, number of waits, etc.

Which approach is better?

It depends on your application!

AVR Resolution



```
void
avr_wait(unsigned short msec)
{
    TCCR0 = 3;
    while (msec--) {
        TCNT0 = (unsigned char)(256 - (XTAL_FRQ / 64) * 0.001);
        SET_BIT(TIFR, TOV0);
        while (!GET_BIT(TIFR, TOV0));
    }
    TCCR0 = 0;
}
```

Check our slides about timers!

Make a new function *or* fix existing code that uses the 1ms resolution (e.g., lcd_init)

See you next time :)

Q & A