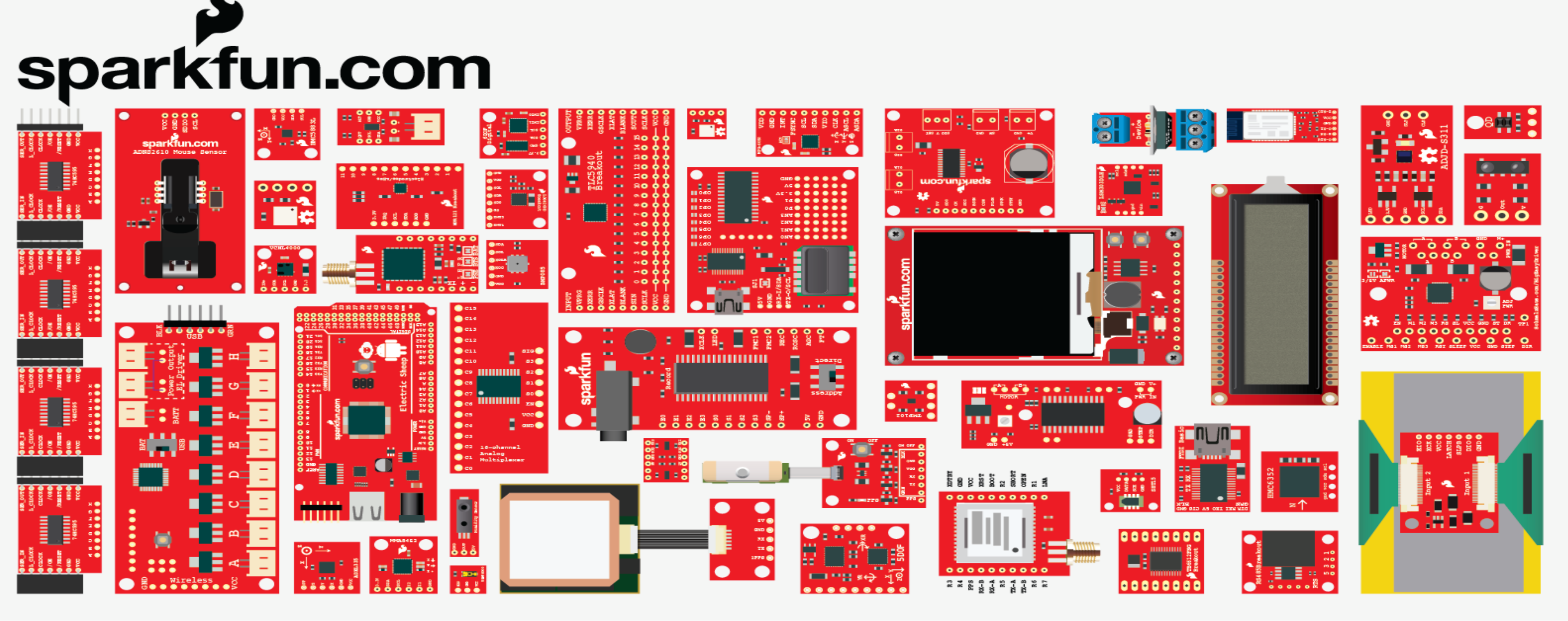
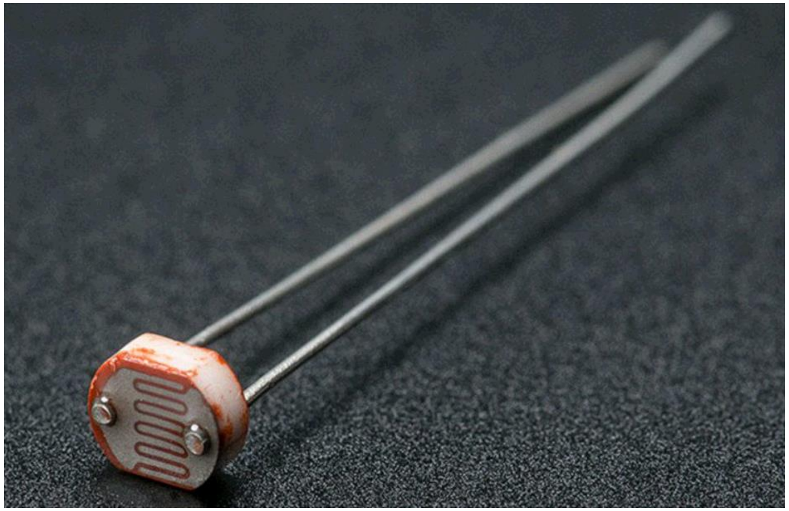


Sparkfun Intermediate

Sparkfun Inventor’s Kits [SIK]

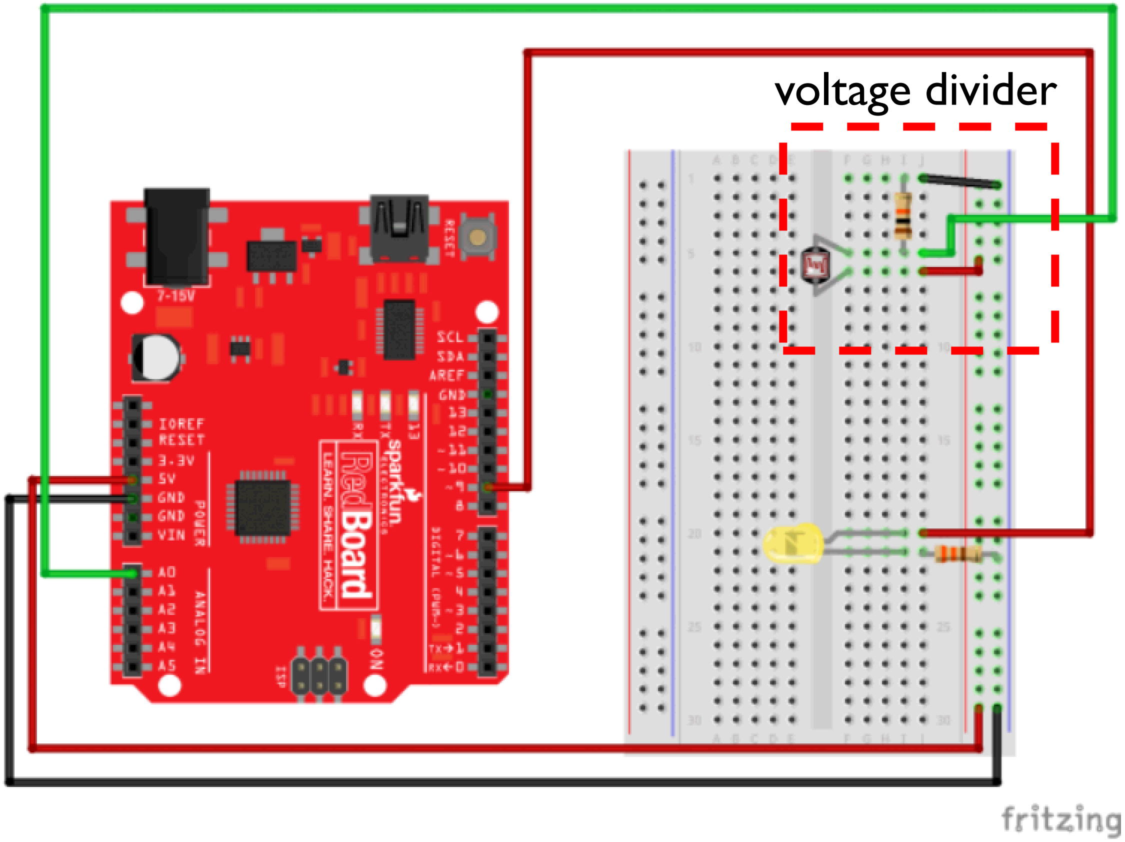
Reading a Photoresistor

- You will need the following parts:
- **1x** Breadboard
- **1x** RedBoard
- **1x** LED
- **1x** 330Ω Resistor
- **6x** Jumper Wires
- **1x** Photoresistor
- **1x** 10k Resistor



Circuit #6-1: Photoresistor wiring diagram

- Build the following circuit.
- Download circuit #6 from Canvas
- Open in Arduino, compile and upload to your board



voltage divider

Circuit #6: photo resistor script

- create constants to name the pins:
  - const int sensorPin = 0; → analog voltage, use pin A0
  - const int ledPin = 9; → This is a digital pin, we want to output voltage to vary the brightness of LED between LOW and HIGH. This pin must support PWM, which is indicated by "~“.
- Repeatedly, read analog voltage (Vout) from photoresistor:
  - No need to use pinMode! A0-A5 are always input pins
  - Use: analogRead(pinNumber)
  - pinNumber: the analog pin number, returns a int (0-1023)
  - → reads analog voltage (0-5V) → ADC coverts analog to a digit ranging from 0-1023
  - lightLevel = analogRead(sensorPin);
  - When the intensity of light is high, lightLevel return a number closer to 1023

Circuit #6: photo resistor script

- Controls the brightness of LED using lightLevel variable:
  - pinMode (ledPin, OUTPUT)
  - Use PWM to dim LED (0-5V): analogWrite(ledPin,lighLevel)
- issue:
  - lightLevel=analogRead() returns values between 0 and 1023
  - analogWrite() gets duty cycle value between 0-255
- Solution: map() and constrain()
  - lightLevel = map(lightLevel, 0, 1023, 0, 255); → "squeeze" the larger range into the smaller range
  - lightLevel = constrain(lightLevel, 0, 255); → only values 0-255 allowed

Circuit #6-1: photo resistor script

- issue:
  - Our voltage divider circuit for photo resistor will not have 0-1023 (0-5V) range!
  - It will be a smaller range, such as 300 (dark) to 800 (light).
  - The LED will not turn on and off completely!
- Let's use serial Monitor to read lightLevel values right after analogRead():
  - Set communication Baud Rate:

```
Void setup()
{
Serial.begin(9600); //9600 bits per second
}
```

Display values on the serial monitor:

```
void loop()
{
lightLevel = analogRead(sensorPin);
Serial.print("lighLevel ="); //print at the same line
Serial.println(lightLevel); // make a new line
}
```

Circuit #6-1: photo resistor script

- Solution:
  - Use the displayed values from the serial monitor
  - Change the min and max range in map() function
  - lightLevel = map(lightLevel, min, max, 0, 255);
  - min= minimum value from the photoresistor displayed on serial monitor;
  - max= maximum value from photoresistor displayed on the serial monitor

- Example:
  - min=300;
  - max= 800;
  - lightLevel = map(lightLevel, 300, 800, 0, 255);

Circuit #6-1: photo resistor script

- Functions that adjust the lightLevel manually: manualTune();

```
void manualTune()
{ //change the 0, 1023 in the line below!
  lightLevel = map(lightLevel, 0, 1023, 0, 255);
  lightLevel = constrain(lightLevel, 0, 255);
}
```

- Functions that adjust the lightLevel automatically: autoTune();
  - Arduino takes care of the alteration of the range
  - Let's use autoTune()
  - Comment out map(); and constrain(); inside the void loop()
  - Uncomment autoTune()

SF-3 Challenge: Create a night light

- Turn off the LED when there is light
- Turn on the LED when it is dark

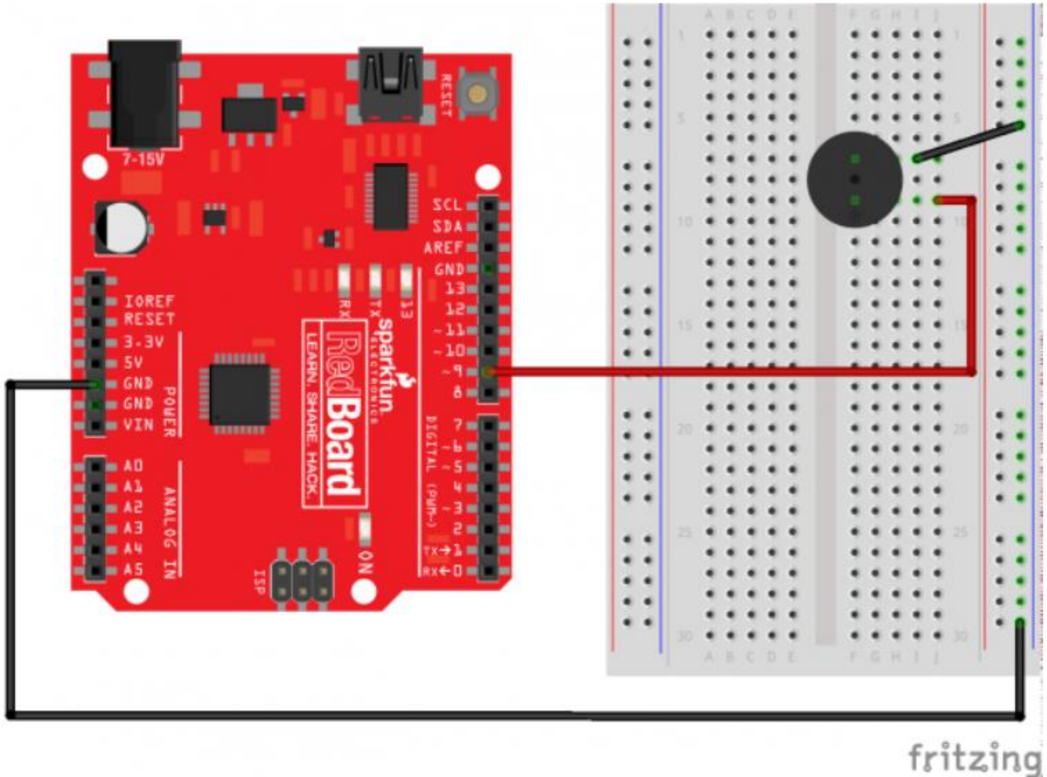
Record a short video and upload to Canvas through SF-3. Do not forge to include your Husky ID in the video.

Using a Piezo Buzzer

- You will need the following parts:
- **1x** Breadboard
- **1x** RedBoard
- **1x** Piezo Buzzer
- **3x** Jumper Wires

Circuit #11: Piezo Buzzer wiring diagram

- Build the following circuit.
- On the buzzer, pin with '+' sign connects to pin 9
- Download circuit #11 from BB
- Open in Arduino, compile and upload to your board



Circuit #11: Piezo Buzzer script

- Create constant variables to declare the buzzer pin
  - const int buzzerPin = 9;   // connect the buzzer to pin 9
- pinMode() for the buzzer is OUTPUT (uses PWM)
  - pinMode(buzzerPin, OUTPUT)

- Arduino can also work with Characters!

Circuit #11-1: Piezo Buzzer script

- Built-in function: tone(pin, frequency, duration);
- tone: drives an output pin at a certain frequency and duration
- duration:
  - If you give it a duration (in milliseconds), it will play the tone then stop
  - If you don't give it a duration: tone(pin, frequency) it will keep playing the tone forever, then you need to use noTone() to stop it!
- frequency: 262 Hz (note 'c')

Circuit #11: custom function: frequency()

- tone(pin, frequency(‘c’),duration)

```
int frequency(char note)
{
  int i;
  const int numNotes = 8;  // number of notes we're storing
  char names[numNotes] = {  'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C‘   };
  int frequencies[numNotes] = {262, 294, 330, 349, 392, 440, 494, 523};

  for (i = 0; i < numNotes; i++)  // Step through the notes
  {
    if (names[i] == note)         // Is this the one?
    {
      return(frequencies[i]);     // Yes! Return the frequency and exit function.
    }
  }
  return(0);}
```

note	frequency
c	262 Hz
d	294 Hz
e	330 Hz
f	349 Hz
g	392 Hz
a	440 Hz
b	494 Hz
C	523 Hz

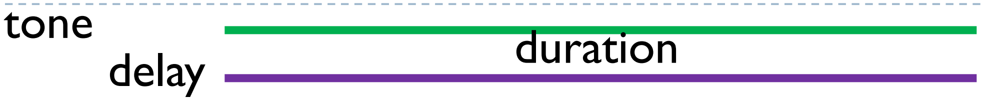
Circuit #11: create a song and loop

- What to play?

```
char notes[18] = {'c', 'd', 'f', 'd', 'a', ' ', 'a', 'g', ' ', 'c', 'd', 'f', 'd',
'g', ' ', 'g', 'f', ' '};
 tone(buzzerPin, frequency(notes[i]), duration)
```

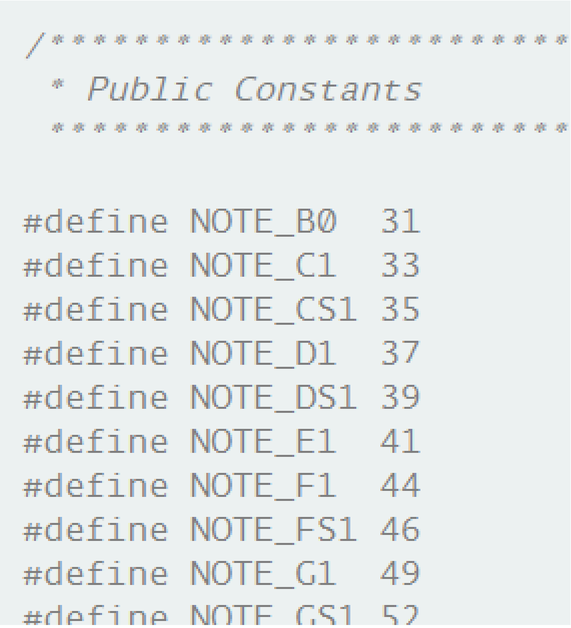
- How long to play?

```
int beats[18] = {1, 1, 1, 1, 1, 1, 4, 4, 2, 1, 1, 1, 1, 1, 1, 4, 4, 2};
int tempo = 113; // beats per second
duration = beats[i] * tempo; //in millisecond
 tone(buzzerPin, frequency(notes[i]), duration)
delay(duration); //if you want to create distinct beats
```



Adding libraries

- Define the notes and the frequencies associated in a new file and save it as a new library
- Open pitches.h from Canvas

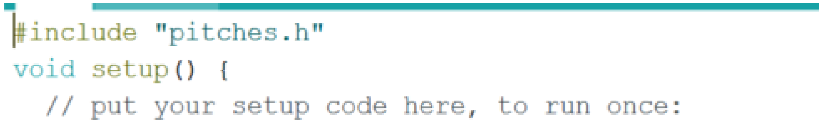


```
/*************************
 * Public Constants
 *************************

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
```

Adding libraries

- Define the notes and the frequencies associated in a new file and save it as a new library
- Open pitches.h from BB
- Save the library in the same folder as your Arduino sketch
- Use #include "pitches.h" library to use these notes and frequencies associated with them



```
#include "pitches.h"
void setup() {
  // put your setup code here, to run once:
```

Adding libraries

- Use #include "pitches.h" library to use these notes and frequencies associated with them
- Update the code

```
 char notes [numNotes] = {'c', 'd', 'f', 'd', 'a', ' ', 'a', 'g', ' ',
'c', 'd', 'f', 'd', 'g', ' ', 'g', 'f', ' '};
int frequencies[numNotes] = {262, 294, 330, 349, 392,
440, 494, 523};
tone(pin, frequency, duration);
```

```
int notes[numNotes] = {  NOTE_C4, NOTE_D4,
NOTE_F4 , NOTE_D4 , NOTE_A4 , 0, NOTE_A4 ,
NOTE_G4 , 0, NOTE_C4, NOTE_D4, NOTE_F4,
NOTE_D4, NOTE_G4, 0, NOTE_G4, NOTE_F4, 0};

tone(pin, notes[i],duration)
```

SF-3 Challenge #2 twinkle twinkle

- Modify Circuit 11 and use the following values to play twinkle twinkle song:
- Notes:"ccggaagffeeddc " //a rest at the end
- Beats: { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 }
- Tempo: 300
- If you want to play forever: // while(true){};