Electronics 1

Electronics for Interactive Media Design

Les 3

Emma Pareschi

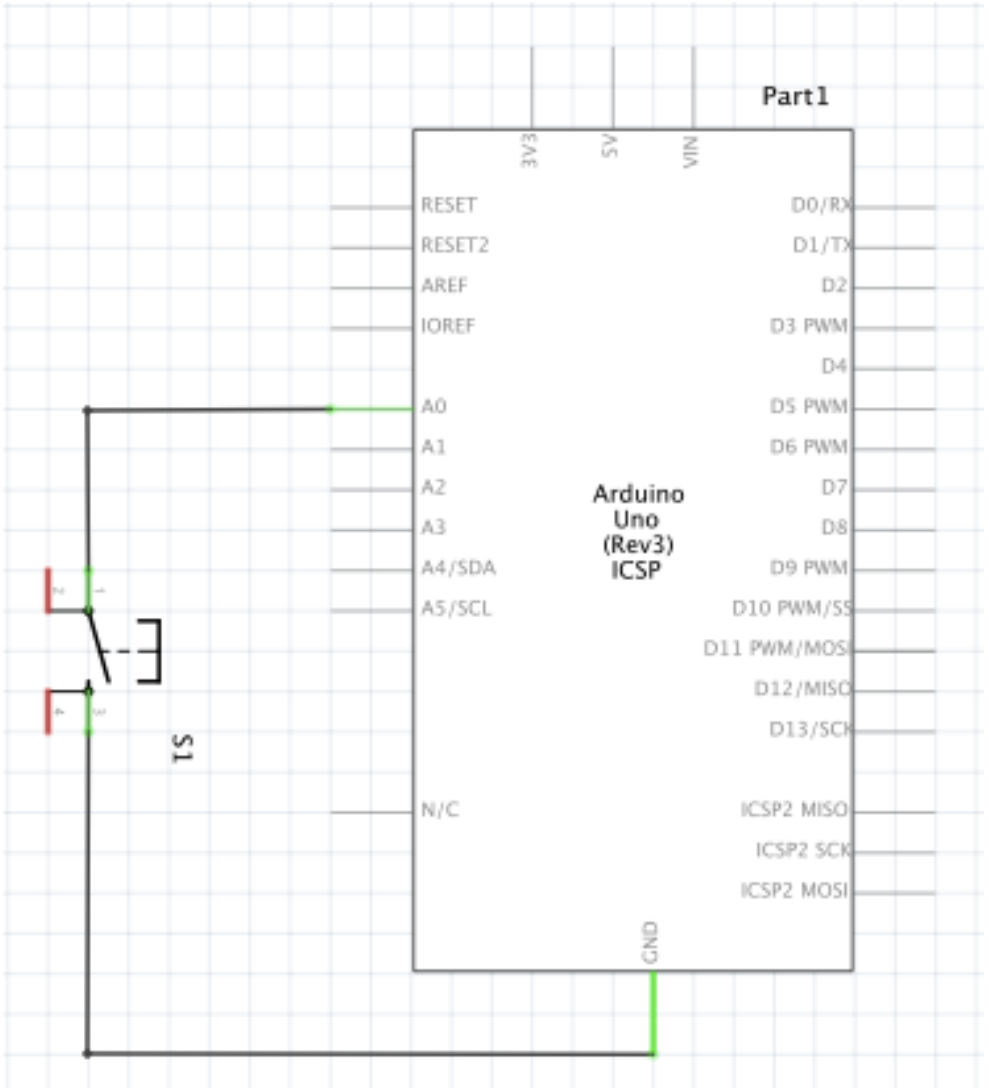
From the last time

From the last time:
- general structor of an Arduino sketch (setup/loop)
- how to read a digital/analog sensor (digitalWrite/analogWrite)
- How to print the values of the sensor in Serial Monitor.
- how to control a digital/analog output device (digitalWrite/ analogWrite)

Today
* Make a light dice
    - How to count a switch
    - For statement
    - Array structure

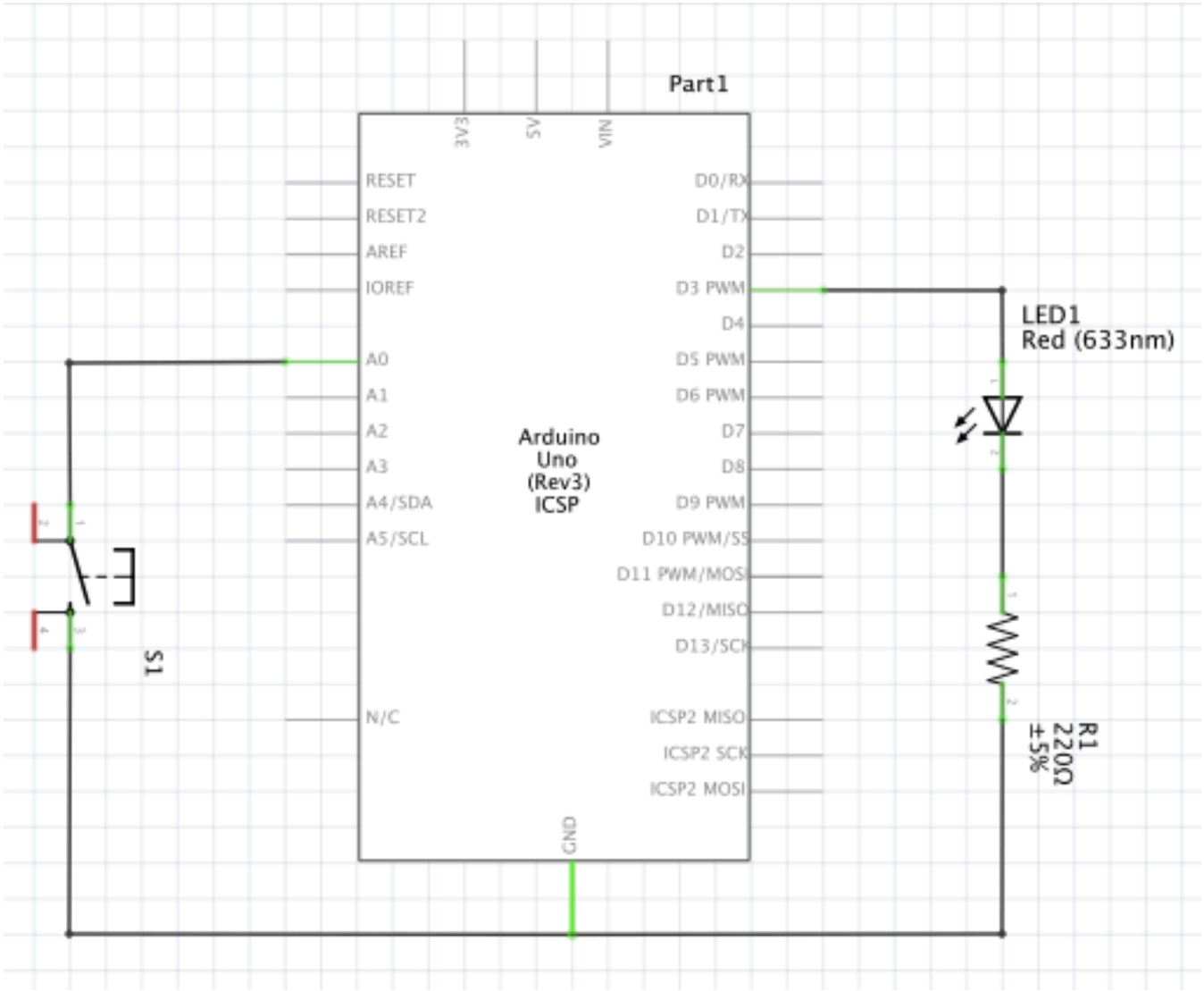
* Make a simple theremin
    - Generate sound: Buzzer
    - Light sensor

Count with a Switch

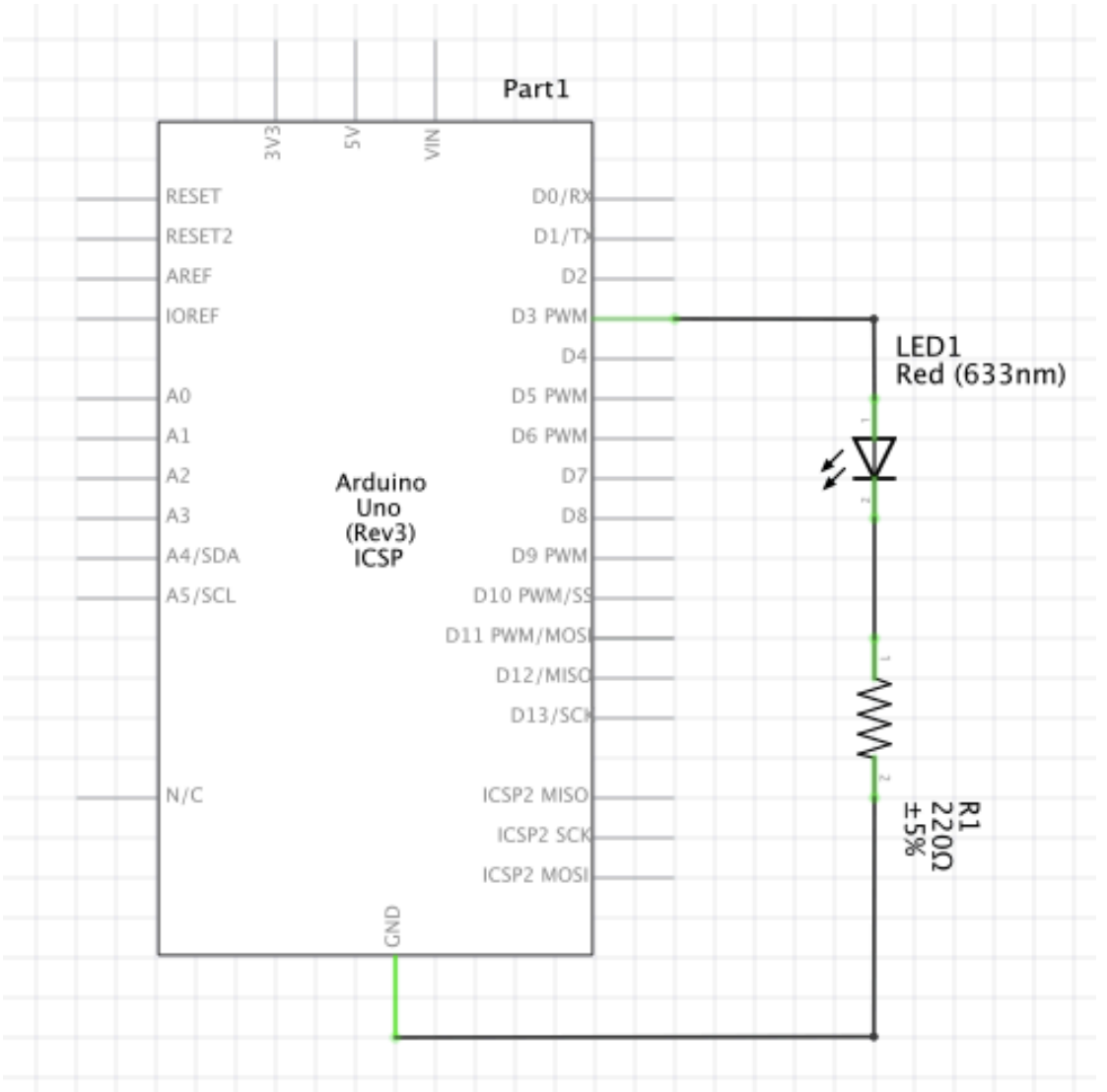


Codes:
00_COUNTER

Led On-Off / Led Level



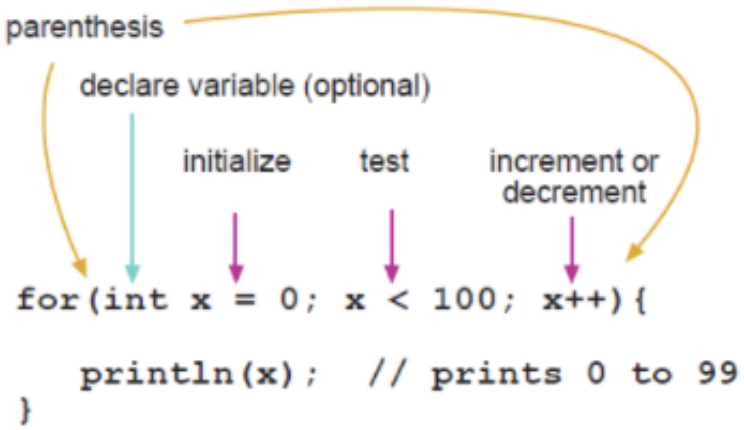
For Statement



FOR

The FOR statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The FOR statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

```
for (initialisation; condition; increment) {
//statement(s);
}
```



Implementation of incremental: FOR

To understand how FOR statement works we simply it as much as possible and we print on Serial Monitor the value of the counter

```
void setup(){
  Serial.print(9600);
}

void loop()
{
  for (int i=0; i <= 255; i++){
    Serial.print(i);
    delay(10);
  }
}
```

Variable SCOPE
Global and Local Variable

Variables in the C programming language, which Arduino uses, have a property called scope.
Scopes: GLOBAL and LOCAL

Global variable is one that can be seen by every function in a program.

Local variables are only visible to the function in which they are declared.

```
int led_pin = 3;  // any function will see this variable
int sensor_value; // any function will see this variable

void setup(){
  // ...
}

void loop(){
  int i;    // "i" is only "visible" inside of "loop"
  float f;  // "f" is only "visible" inside of "loop"
  // ...
}
```

Implementation of incremental: FOR

You want to increment the brightness of a LED.

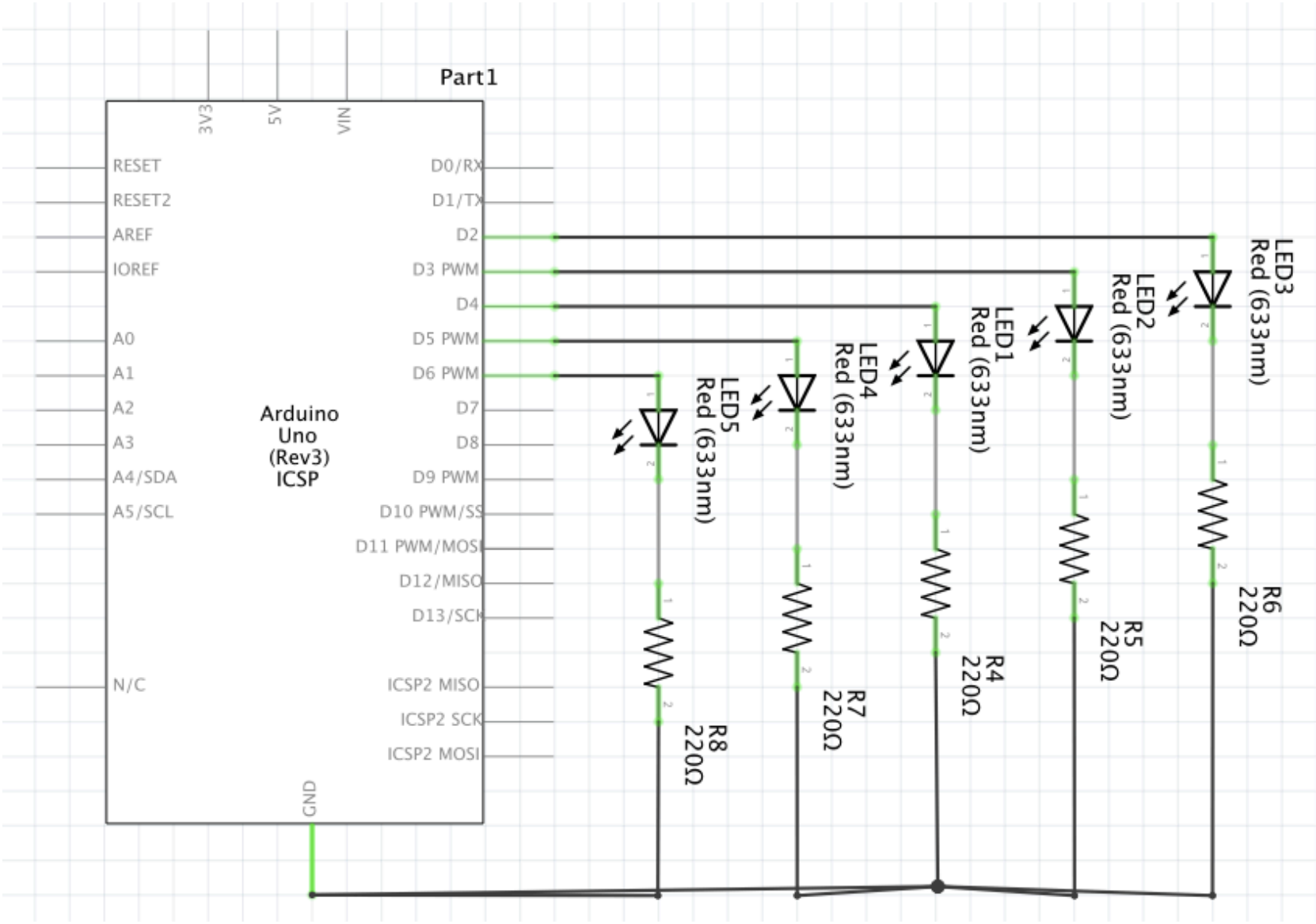
```
int PWMpin = 10; // LED in series with 470 ohm resistor on pin 10

void setup(){
  pinMode(PWMpin, OUTPUT);
}

void loop()
{
  for (int i=0; i <= 255; i++){
    analogWrite(PWMpin, i);
    delay(10);
  }
}
```

Increment

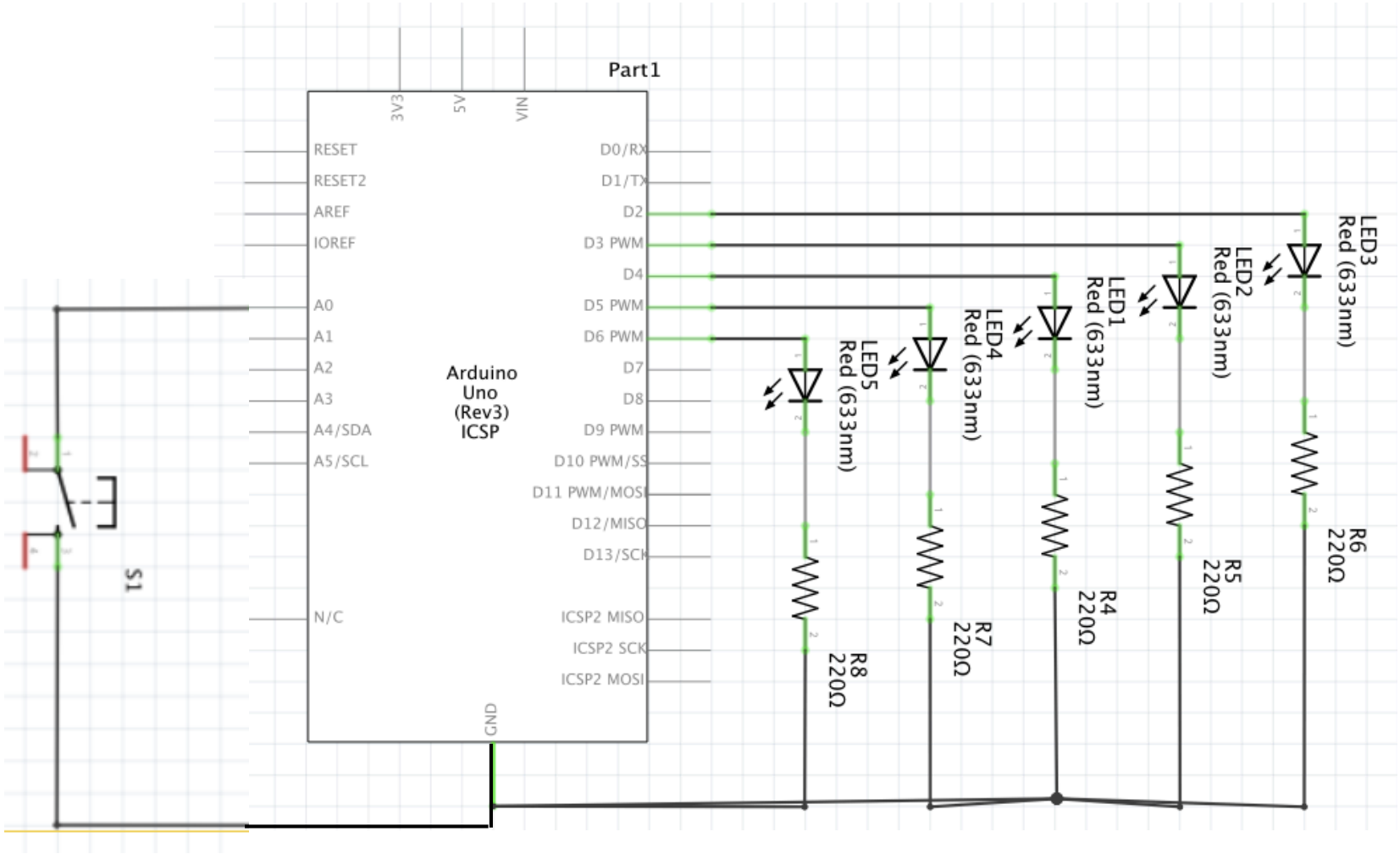
You want to increase or decrease the value of a variable
+=
-=

```
int myValue = 0;

myValue = myValue + 1;   // this adds one to the variable myValue
myValue += 1;            // this does the same as the above
myValue++                // this does the same as the above

myValue = myValue - 1;   // this subtracts one from the variable myValue
myValue -= 1;            // this does the same as the above
myValue - -              // this does the same as the above

myValue = myValue + 5;   // this adds five to the variable myValue
myValue += 5;            // this does the same as the above
```

LED ARRAY (AND FOR LOOP)



Dice


Part1
3V3
5V
VIN
RESET
RESET2
AREF
IOREF
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
Arduino
Uno
(Rev3)
ICSP
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
ICSP2 MISO
ICSP2 SCK
ICSP2 MOSI
GND
S1
LED5
Red (633nm)
LED4
Red (633nm)
LED1
Red (633nm)
LED2
Red (633nm)
LED3
Red (633nm)
R8
220Ω
R7
220Ω
R4
220Ω
R5
220Ω
R6
220Ω

ARRAY

An array is a collection of variables that are accessed with an index number.

To create (or declare) an array

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
```

Accessing an Array
Arrays are zero indexed, that is, referring to the array initialization above, the first element of the array is at index 0,

```
int myArray[10]={9,3,2,4,3,2,7,8,9,11};
    // myArray[9]    contains 11
    // myArray[10]   is invalid and contains random information (other memory address)
```

Generating sounds: Transducer

It converts electrical energy into sound.

The term transducer here describes a noise-creating device that is driven by external electronics.
(By comparison, an audio indicator contains its own internal electronics and only requires a DC power supply.)

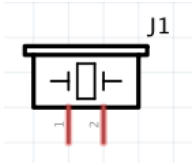
An audio transducer is a device that can create an alert. It requires an AC signal that is supplied by external electronics, and in its simplest form may be referred to as a buzzer or a beeper.

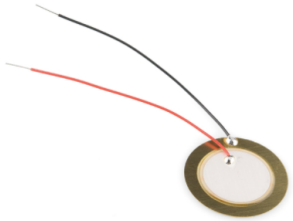
Audio alerts are used in microwave ovens, wash- er/dryers, automobiles, gasoline pumps, security devices, toys, phones, and many other consumer products. They are often used in conjunction with touch pads, to provide audio confirmation that a tactile switch has been pressed.

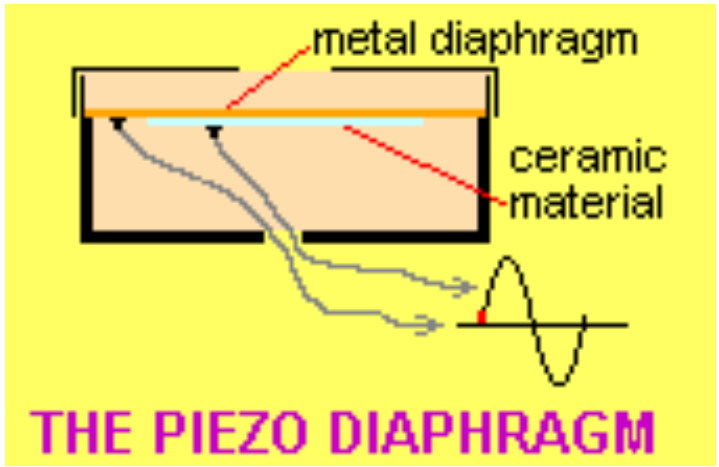
Variants: Electromagnetic, Piezoelectric , Ultrasonic

Transducer - Piezoelectric

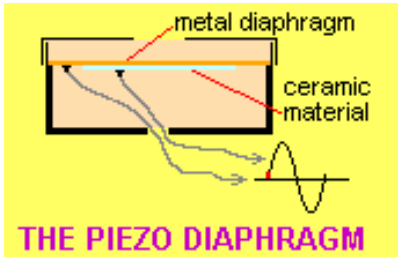
A piezoelectric transducer contains a diaphragm consisting of a thin brass disc on which is mounted a ceramic wafer. When an AC signal is applied between the piezoelectric wafer and the disc, the disc flexes at that frequency.







Frequency



Frequency is the number of occurrences of a repeating event per unit of time.[1]

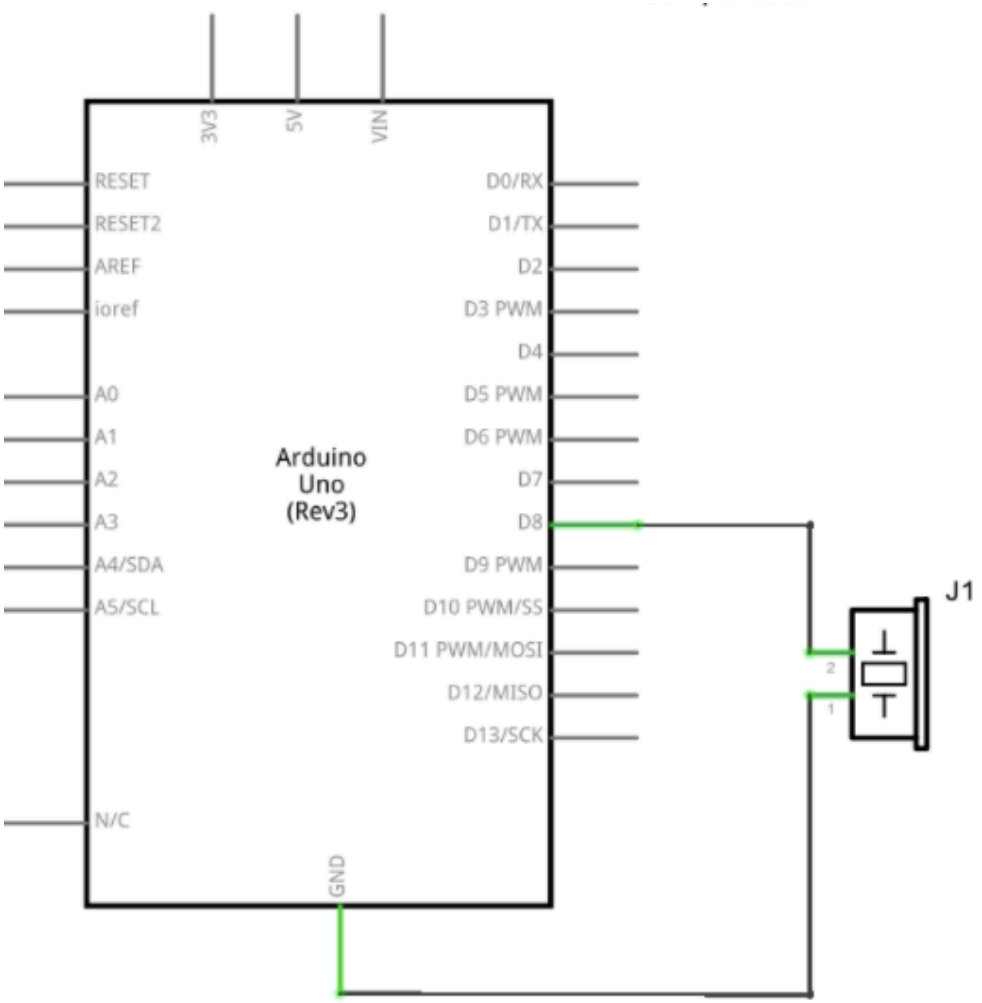
Audio frequency is measured in Hertz, abbreviated Hz, named after Heinrich Rudolf Hertz, the first scientist to prove the existence of electromagnetic waves.

The H in Hz is capitalised because it refers to a real name. One thousand Hertz can be written as 1 kiloHertz, almost always abbreviated as 1kHz (note that the k is lowercase).

The human ear is often described as being able to detect sounds between 20Hz and 20kHz, although the ability to hear sounds above 15kHz is relatively unusual and diminishes naturally with age. Sensitivity to all frequencies can be impaired by long-term exposure to loud noise.

The most common frequencies applied to audio transducers range between 3kHz and 3.5kHz. Piezoelectric elements are inefficient for generating sounds below 1kHz.

Piezo schematic



The piezo is connected between pin 8 and ground

piezo §

```
/*Emma pareschi
 * October 2017
 * I generate a sound with a piezo element
 * and I apply a frequenzy of 1000Hz
 */


const int buzzer = 8; //buzzer to arduino pin 9


void setup(){

  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output

}

void loop(){

  tone(buzzer, 1000); // Send 1KHz sound signal

}
```

Function:
- Tone();

Function TONE

tone(pin number, frequency in hertz);

1. The pin number that you will use on the Arduino.
2. The frequency specified in hertz

tone(pin number, frequency in hertz, duration in milliseconds);

1. The pin number that you will use on the Arduino.
2. The frequency specified in hertz
3. The duration of the tone in milliseconds (optional) - unsigned long

```
const int buzzer = 8;

tone(buzzer, 1000);
tone(buzzer, 1000, 5000);
```

Notes:

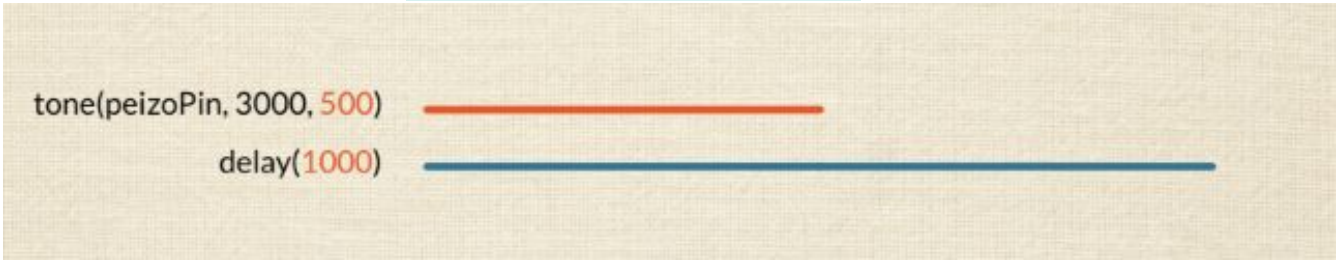
- min frequency: 31Hz
- Max frequency: 65535

Function TONE // noTone

```
const int buzzer = 8;

tone(buzzer, 1000, 1000);
delay(1000);
```

I want to generate distinct beats long 1sec every 1sec.
I try to use delay.
But it doesn't work.
Why?

```
tone(buzzer, 3000, 500);
delay(1000);
```




```
tone(buzzer, 1000); // Send 1KHz sound signal...
delay(1000);        // ...for 1 sec
noTone(buzzer);     // Stop sound...
delay(1000);        // ...for 1sec
```

noTone(pin number);

Stops the generation of a square wave triggered by `tone()`

Piezo sketch

piezo

```
/*Emma pareschi
 * October 2017
 * I generate a sound with a piezo element
 * and I apply a frequenzy of 1000Hz
 */


const int buzzer = 8; //buzzer to arduino pin 9


void setup(){

  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output

}

void loop(){

  tone(buzzer, 1000); // Send 1KHz sound signal...
  delay(1000);        // ...for 1 sec
  noTone(buzzer);     // Stop sound...
  delay(1000);        // ...for 1sec

}
```

Limits of the Tone Function

1. You can't use tone() while also using analogWrite() on pins 3 or 11. If you do – you get some whacky results – neither will work like you expect. That's because the tone() function uses the same built in timer that analogWrite() does for pins 3 and 11. It's worth trying just hear the weird noises.

2. You cannot generate a tone lower than 31 HZ. You can pass values 31 and less to the tone() function, but it doesn't mean you will get a good representation of it.

3. The tone() function cannot be used by two separate pins at the same time. Let's say you have two separate piezo speakers, each on a different pin. You can't have them both play at the same time. One has to be on, and then the other. Furthermore, before you can have the other pin use the tone() function, you must call the noTone() function and "turn off" the tone from the previous pin.

Melody

To create a melody you need notes. Each note is defined by a frequency.

Note Frequency

	C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B
0	16.35	17.32	18.35	19.45	20.60	21.83	23.12	24.50	25.96	27.50	29.14	30.87
1	32.70	34.65	36.71	38.89	41.20	43.65	46.25	49.00	51.91	55.00	58.27	61.74
2	65.41	69.30	73.42	77.78	82.41	87.31	92.50	98.00	103.8	110.0	116.5	123.5
3	130.8	138.6	146.8	155.6	164.8	174.6	185.0	196.0	207.7	220.0	233.1	246.9
4	261.6	277.2	293.7	311.1	329.6	349.2	370.0	392.0	415.3	440.0	466.2	493.9
5	523.3	554.4	587.3	622.3	659.3	698.5	740.0	784.0	830.6	880.0	932.3	987.8
6	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976
7	2093	2217	2349	2489	2637	2794	2960	3136	3322	3520	3729	3951
8	4186	4435	4699	4978	5274	5588	5920	6272	6645	7040	7459	7902

```
#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247
```

Melody

piezo_simple_melody

```
#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247

const int buzzer = 8;

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(buzzer, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(buzzer);
  }
}
void loop() {
  // no need to repeat the melody.
}
```

Melody

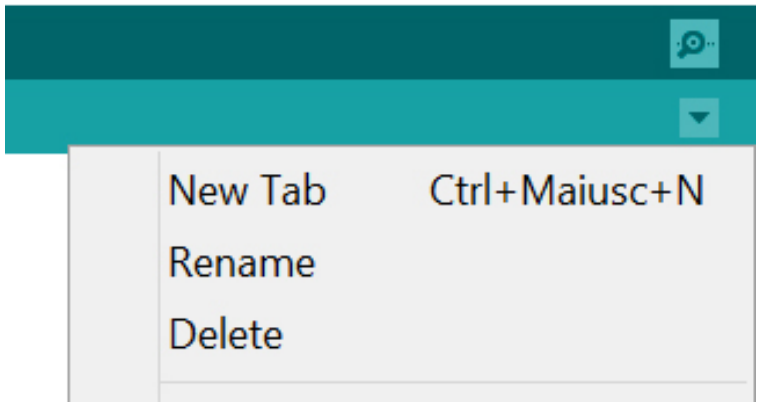
```
#include "pitches.h"

const int buzzer = 8;

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second divided by the no
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(buzzer, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(buzzer);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

piezo_melody | pitches.h

```
/*************************************************
 * 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
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
```

Melody

To create the file pitches,
Select new tab:

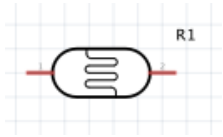


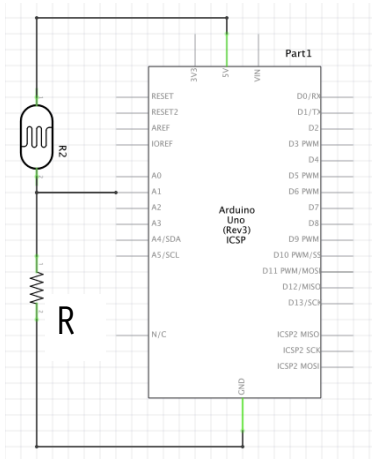
piezo_melody | pitches.h

```
/*************************************************
 * 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
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
```

LIGHT SENSOR

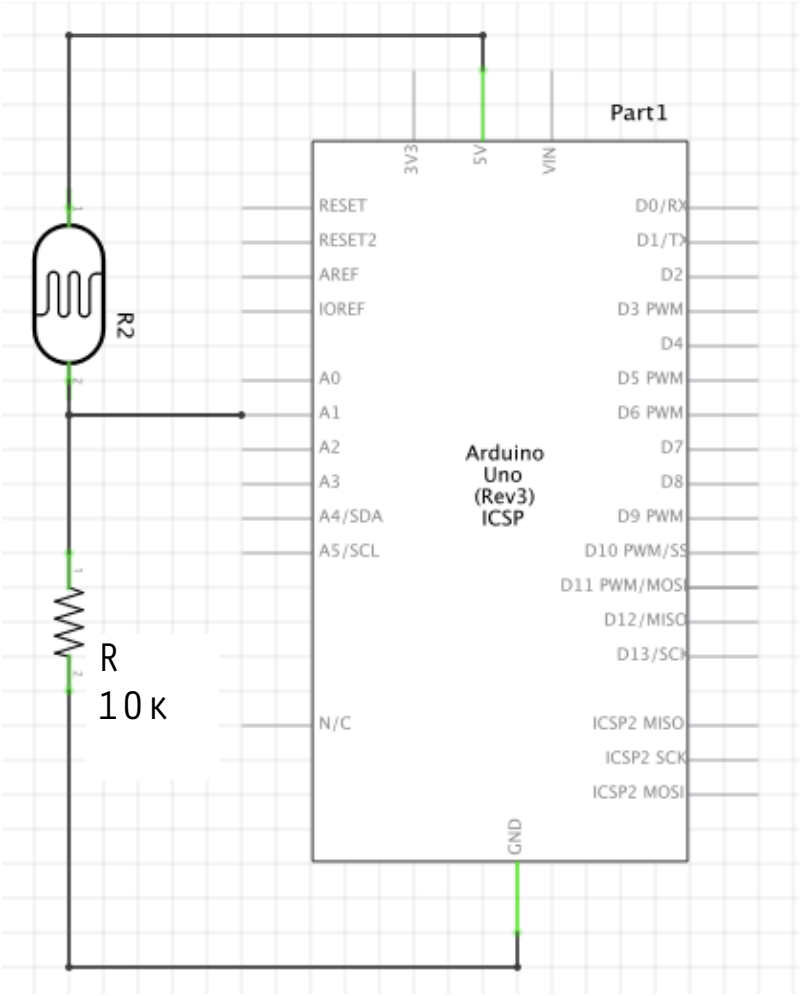
A photoresistor (or light-dependent resistor, LDR, or photo-conductive cell) is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity



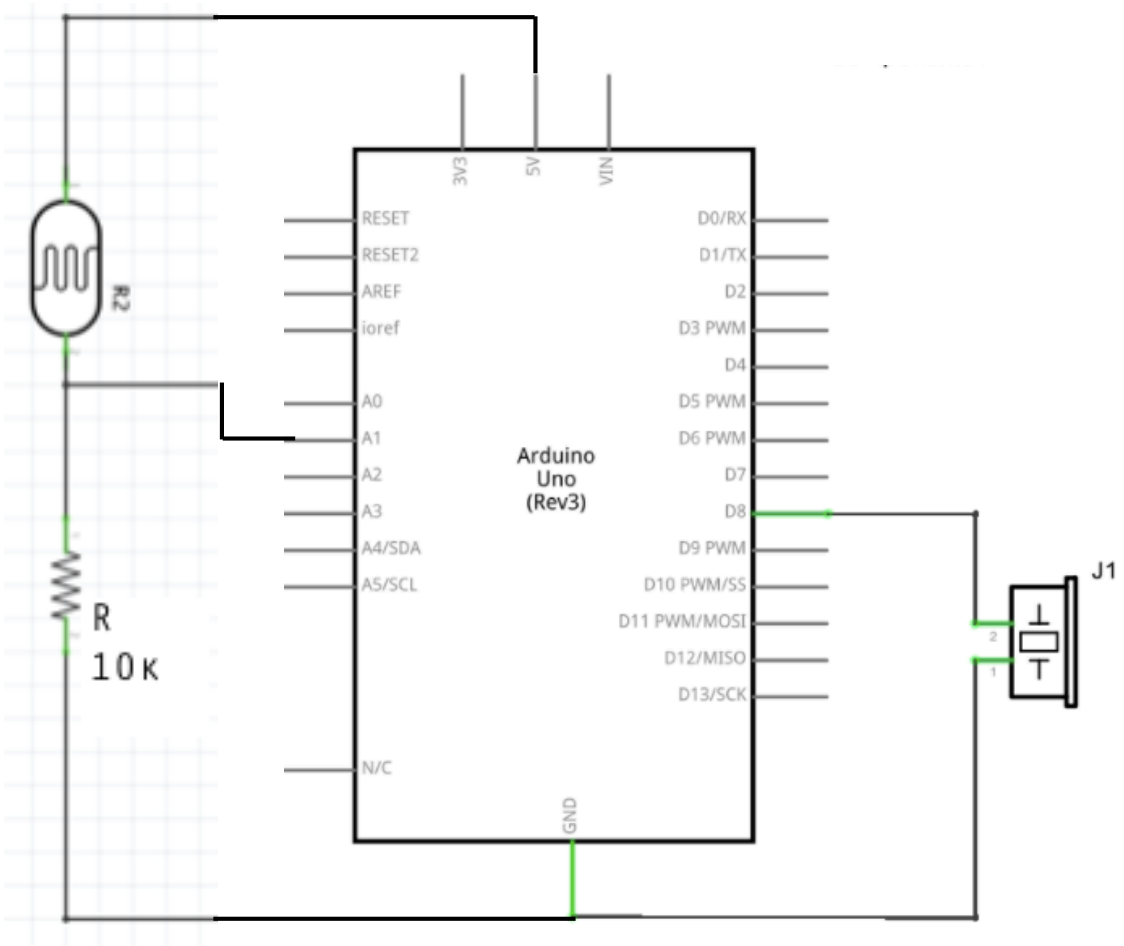


LIGHT SENSOR

What resistor to use?
Measure the resistance of the LDR in the dark and in the light.
The average of the two values is the value of the pull-down resistor.



Theremin


3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
GND
Arduino
Uno
(Rev3)
R2
R
10K
J1
2
1

Weekly Assignments

- Study the codes we used in class.
- Project 1
  - Finish the dice. Combine the counter code with the random sequence code to reproduce the behaviour of a Dice. The dice will work in three steps:
    1- you launch the dice -> press the switch
    2- get the random number
    3- reset the game for the next player -> press the switch again

- Project 2
  - Input device: select a sensor from the Arduino and Sensor kits. The input device must be used to sense the presence of a person.
  - Output device: passive buzzer
  - Functionality: trigger a melody or sound using the input device.
  - Decide the story of your project (where would you use it? What is the circumstance in which you detect the person), make the schematic, make the circuit on breadboard, define the functionality, write the code, document it.

References

Commands or structures explained and used in class:
- const / int
- void setup / void loop
- pinMode
- analogWrite / digitalWrite
- analogRead / digitalRead
- delay
- Serial.begin / Serial.print / Serial.println
- if / if..else
- for
- array
- tone / noTone

description: https://www.arduino.cc/reference/en/

Long list of explained examples with schematic and codes:
https://juniorfall.files.wordpress.com/2011/11/arduino-cookbook.pdf