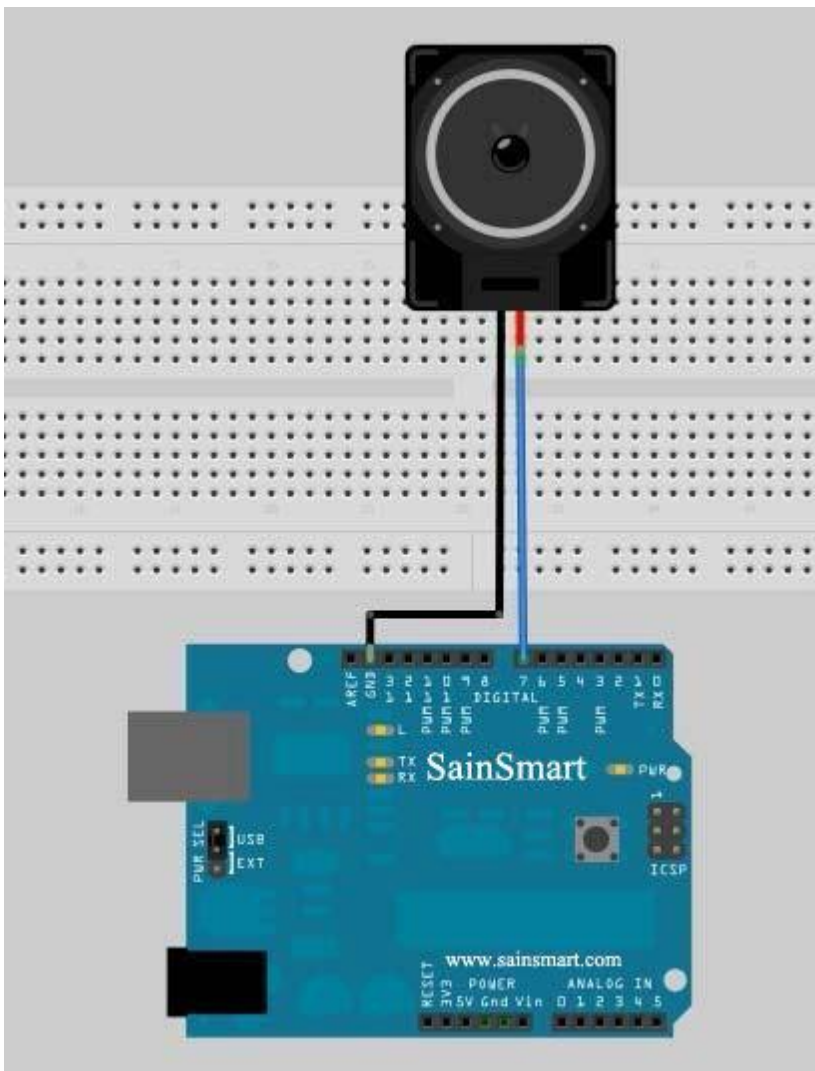


**Homework 9 – Multi-purpose Infrared Remote Control**

**Infrared remote controls** are commonly used to control TVs, DVD players, stereos, cable boxes, and so on... Infrared remote controls send a series of binary pulse code using infrared light signals. The signal between a remote control handset and the device it controls consists of pulses of infrared light, which is invisible to the human eye, but can be seen through a digital camera, video camera, or a phone camera.

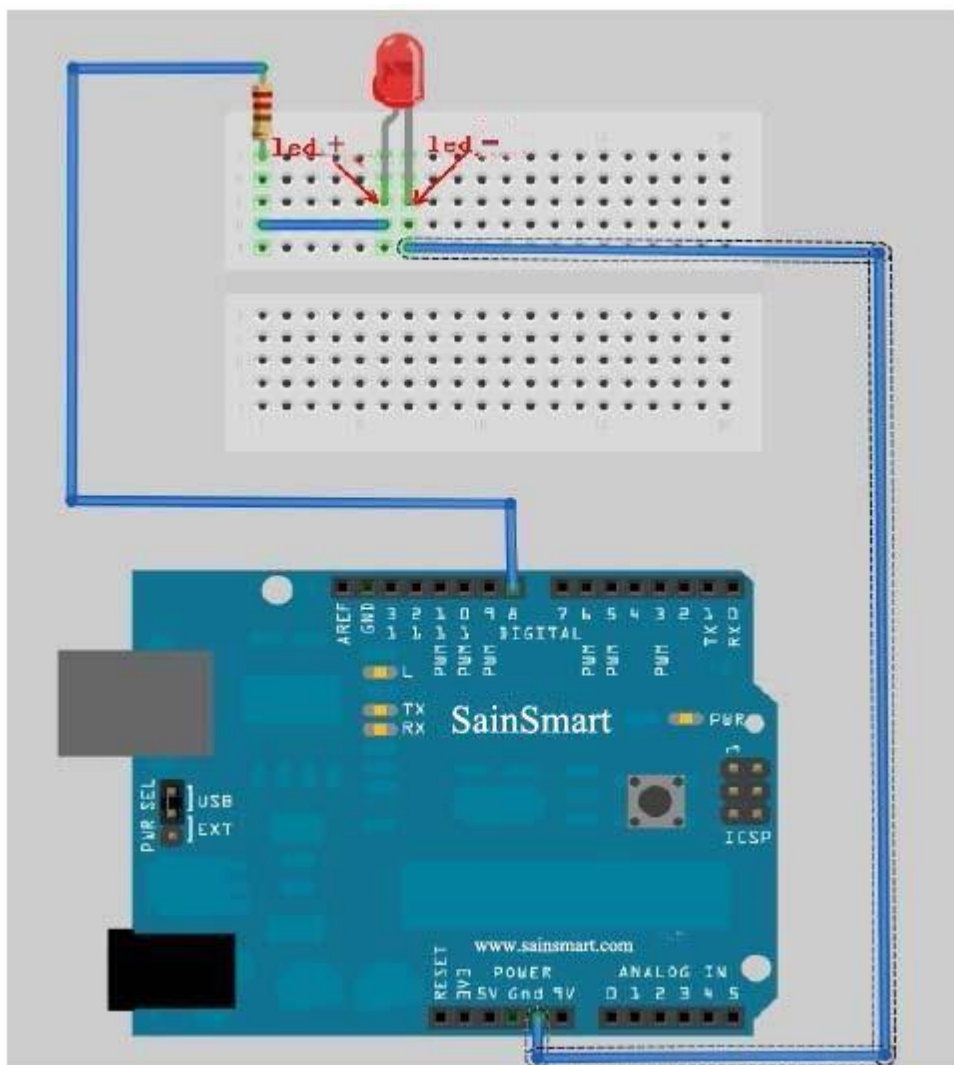
The **transmitter** in the remote control handset sends out a stream of pulses of **infrared light** when the user presses a button on the handset. A transmitter is often a light emitting diode (LED) which is built into the pointing end of the remote control handset. The infrared light pulses form a pattern unique to that button. The **receiver** in the device can be programmed to recognize the pattern and causes the device to respond accordingly.

**Connect the Arduino Uno R3 compatible board with a breadboard and a passive buzzer, using jumper wires.** The following figure shows that the positive terminal of the buzzer is connected to the Uno's digital I/O pin 7 and the negative terminal is connected to the **ground**.



Connect the Arduino Uno R3 compatible board with a breadboard using jumping wires, and serially connect a  $220\Omega$  resistor with one light emitting diode (LED). The following figure shows that the positive terminal of the LED is connected to the  $220\Omega$  resistor, and the  $220\Omega$  resistor is connected to the Uno's digital I/O pin 8 and the negative terminal of the LED is connected to the ground.

Repeat the steps above again and connect a different color LED to the Uno's digital I/O pin 3, 4, 5, 6, 9, 10, and 11. Install a total of 8 LED lights (using I/O Pins 3, 4, 5, 6, 8, 9, 10, and 11) in one line.



Connect the 3 pins of the infrared receiver sensor to the bread board and add jumping wires as follows:







- Connect VOUT to the Uno’s digital I/O pin 2.
- Connect GND to the Uno’s GND.
- Connect VCC to the Uno’s +5 v source. Ensure the power source of the Uno board is set to 5v.

























Write a c program, [using the Arduino’s integrated development environment \(IDE\)](#), to utilize an infrared remote controller to simulate the following:

1. A piano keyboard.
2. Garage door opener.
3. Sound volume controller.
4. Play a song.

The remote control buttons and actions are listed below:

Simulation	Mode #	Button	Action
Piano keyboard	Press  then   Initialize Mode 1 with all the lights turned off.		Play C note (frequency: 1047) for 0.5 sec. Turn on light #1 during the note plays.
			Play C sharp / D flat note (frequency: 1109) for 0.5 sec. Turn on light #1 and #2 during the note plays.
			Play D note (frequency: 1175) for 0.5 sec. Turn on light 2 during the note plays.
			Play D sharp / E flat note (frequency: 1245) for 0.5 sec. Turn on light #2 and #3 during the note plays.

		 2	Play E note (frequency: 1319) for 0.5 sec. Turn on light #3 during the note plays.
		 3	Play F note (frequency: 1397) for 0.5 sec. Turn on light #4 during the note plays.
		 4	Play F sharp / G flat note (frequency: 1480) for 0.5 sec. Turn on light #4 and #5 during the note plays.
		 5	Play G note (frequency: 1568) for 0.5 sec. Turn on light #5 during the note plays.
		 6	Play G sharp / A flat note (frequency: 1661) for 0.5 sec. Turn on light #5 and #6 during the note plays.
		 7	Play A note (frequency: 1760) for 0.5 sec. Turn on light #6 during the note plays.
		 8	Play A sharp / B flat note (frequency: 1865) for 0.5 sec. Turn on light #6 and #7 during the note plays.
		 9	Play B note (frequency: 1976) for 0.5 sec. Turn on light #7 during the note plays.
Garage Door Opener	Press  then  2   The  button simulates the garage door opener button.  Initialize Mode 2 with all the lights turned off to simulate a closed garage door.		Press this button the first time should <b>sequentially turn on an additional light per second, starting from light #1. Play frequency 466 continuously to simulate the garage door opening, until all the lights are on.</b>
			Press this button when all the lights are on <b>should sequentially turn off one light per second, starting from light #8. Play frequency 156 continuously to simulate the garage door closing until all the lights are off.</b>
			Press this button during the garage door opening up <b>should pause the garage door.</b> The buzzer should stop humming and the number of lights turned on should stop increasing. <b>The next time when this button is pressed, the garage door should move down.</b>
			Press this button during the garage door closing down <b>should pause the garage door.</b> The buzzer should stop humming and the number of lights turned off should stop increasing. <b>The next time when this button is pressed, the garage door should move up.</b>
Sound Volume Control	Press  then  3  Initialize Mode 3 with all the lights turned off.		Press this button each time will <b>sequentially increment the number of lights on by 1, starting from light #1.</b> This button has no effect after all 8 lights are on.
			Press this button each time will <b>sequentially decrement the number of lights on by 1, starting from light #8.</b> This button has no effect after all 8 lights are off.
Play music and light show	Press  then  4	n/a	<b>Welcome to reuse the music and light show from Homework 7.</b>

The `millis()` function in the Arduino Library (<http://arduino.cc/en/reference/millis>) can be used to track time in milliseconds after the program starts. The `millis()` function returns the number of milliseconds since the Arduino board began running the current program. This number will go back to zero after approximately 50 days. To calculate the actual game time in milliseconds, the value returned by the `millis()` function call should be compared to a value returned by a previous call to the `millis()` function. The `millis()` function call output is a **unsigned long** data type.

### IMPORTANT:

**The Arduino IRremote library functions and the Arduino built-in `tone()` and `noTone()` functions can NOT be used in the same program!** Functions to send/receive infrared light frequency and the functions to produce sound frequency are both trying to control the same hardware interrupt **timer 2** on the Arduino board.

**The workaround is to use the NewTone library functions to produce sound.** The NewTone library functions use the hardware interrupt **timer 1** to produce the desired frequency.

To use the `<NewTone.h>` library functions, include the **NewTone.h** header file in the program. The syntax of the `NewTone()` function call is:

**`NewTone(pin#, frequency, duration);`**

**Using the `<IRremote.h>` library functions, we can capture and decode the signals send from each infrared remote control buttons (for example):**


```
#include <IRremote.h>

int RECV_PIN = 2;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn();
}

void loop()
{
  if (irrecv.decode(&results))
  {
    Serial.println(results.value, HEX);
    irrecv.resume();
  }
  delay(100);
}
```

The IR decoded value of each button from the SainSmart remote control is listed below. The button IR decode results need to be defined in your program for your program to recognize the IR signal.

Infrared Remote Control Button	Decoded Value in Hexadecimal Format
Mode	0xFF629D
+	0xFF906F
-	0xFFA857
	0xFF9867
U/SD	0xFFB04F
0	0xFF6897
1	0xFF30CF
2	0xFF18E7
3	0xFF7A85
4	0xFF10EF
5	0xFF38C7
6	0xFF5AA5
7	0xFF42BD
8	0xFF4AB5
9	0xFF52AD

**Homework 9 will also be presented in class.** You are welcome to use any remote control to do this project, however, if the SainSmart remote control is not used, you **MUST** demo this project in class.

**If you are not able to present Homework 9 during the class time, we will grade your Homework 9 with SainSmart remote controller and our circuit board using the pin numbers specified in this assignment.**