

Designing A Christmas Novelty

Contents:

- Timescales
- Requirements
- Modes Of Operation
- Coding
- Playing Music
- Bringing It Together
- Test & Demonstration



Timescales: Example Programme

STEM 1:

- Project launch
- Initial design - Fascia

STEM 2:

- Physical design
- Circuit design
- Parts List

STEM 3:

- Coding:
 - Switches
 - Functions
 - If / Else or Switch
 - Controlling LEDs

STEM 4:

- Playing Music
- Choosing melodies

STEM 5:

- Bringing it together

STEM 6:

- Test & Demonstration

Making It



Coding



365	September 2015						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
36		1	2	3	4	5	6
37	7	8	9	10	11	12	13
38	14	15	16	17	18	19	20
39	21	22	23	24	25	26	27
40	28	29	30				

STEM 1

365	October 2015						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
40				1	2	3	4
41	5	6	7	8	9	10	11
42	12	13	14	15	16	17	18
43	19	20	21	22	23	24	25
44	26	27	28	29	30	31	

STEM 2

STEM 3

365	November 2015						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
44							1
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29
49	30						

STEM 4

STEM 5

365	December 2015						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
49		1	2	3	4	5	6
50	7	8	9	10	11	12	13
51	14	15	16	17	18	19	20
52	21	22	23	24	25	26	27
53	28	29	30	31			

STEM 6

Consider Requirements:

- Who is it for?
- How big will it be?
- How will it look?
- What will it do?
 - Play tunes - How many?
 - Flash Lamps - ...
 - Respond to user input
 -
- How will it be powered?
 - Battery pack
 - Mains adapter
- How will it be controlled?
 - Simple power up
 - Push button start
 - Remote controlled
 - Sound activated (clap!)
 - ...
-
- How will I test it?
- What are my constraints?
 - What will it cost?



**ENGINEERS WORK TO
REQUIREMENTS**

MANY THINGS TO CONSIDER!

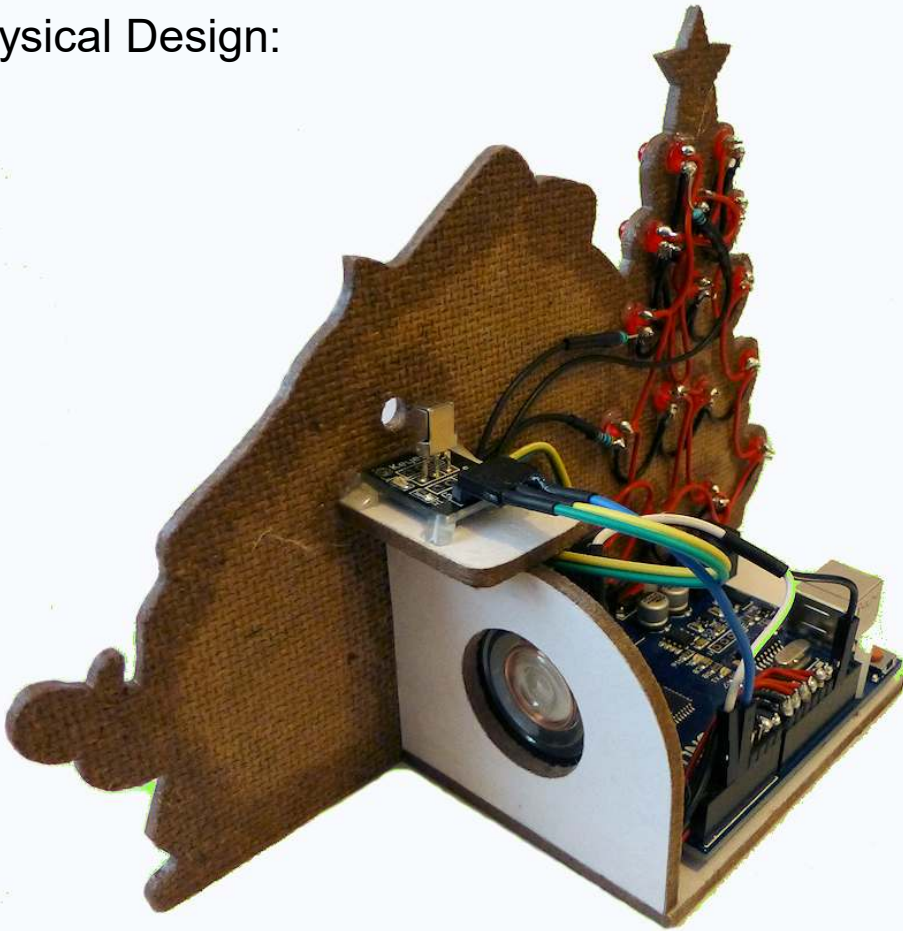
DESIGN == MAKING DECISIONS

- Who is it for?
- How big will it be?
- How will it look?

Merry Christmas!

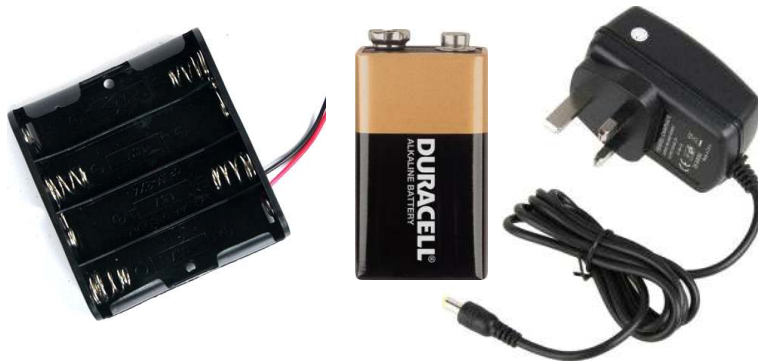


Physical Design:



How will it be powered?

- Battery pack
- Mains adapter



How will it be controlled?



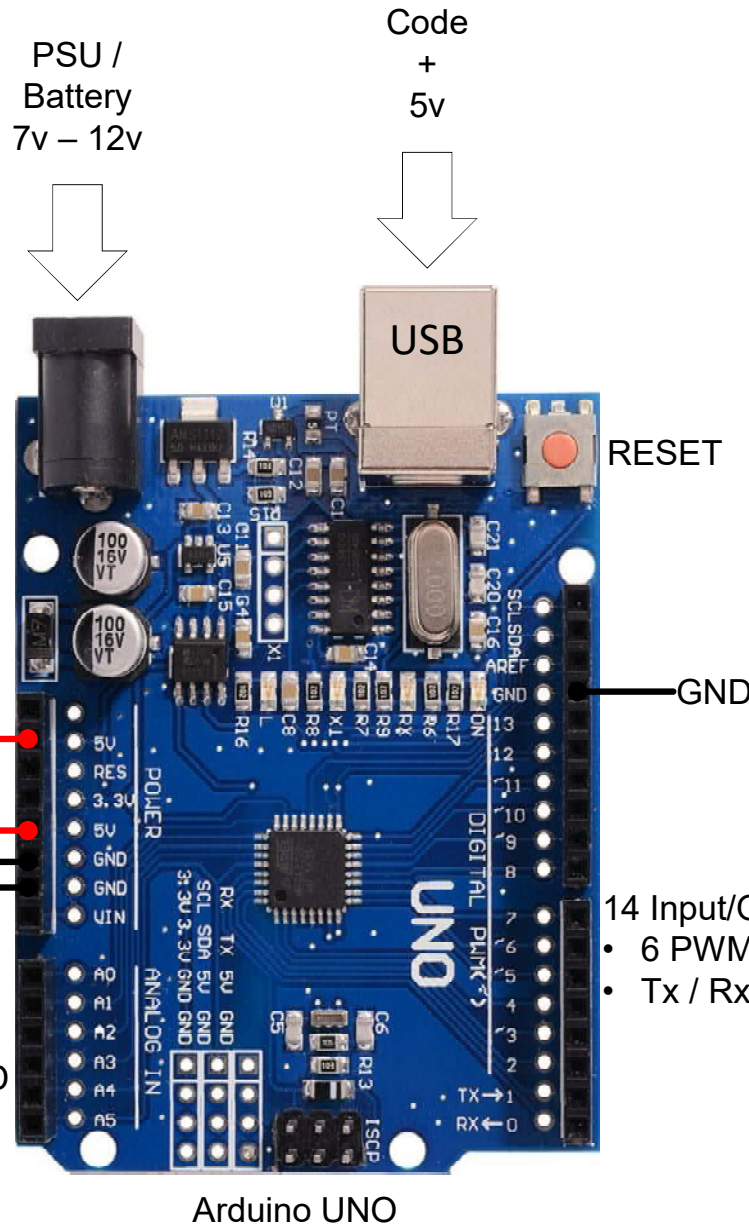
Switches



IR Sensor



6 Inputs
• 10-bit A/D



Arduino UNO

What will it do?

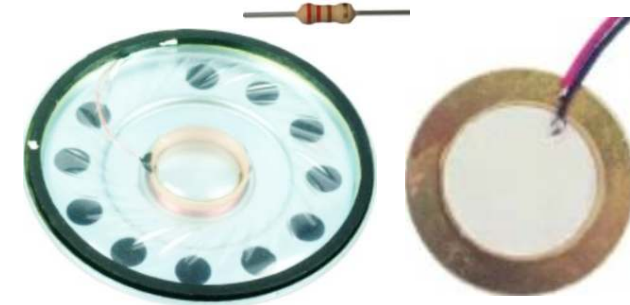
- Play tunes - How many?
- Flash Lamps - ...
- Respond to user input



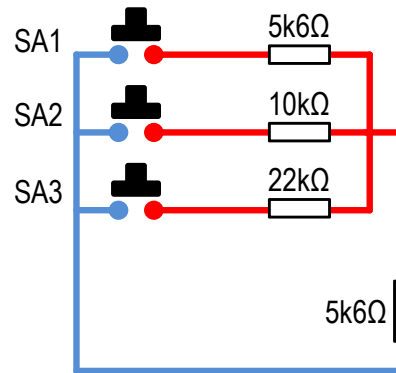
**RESISTORS MUST BE
USED WITH LEDs
AND SPEAKERS!**

14 Input/Outputs:

- 6 PWM
- Tx / Rx

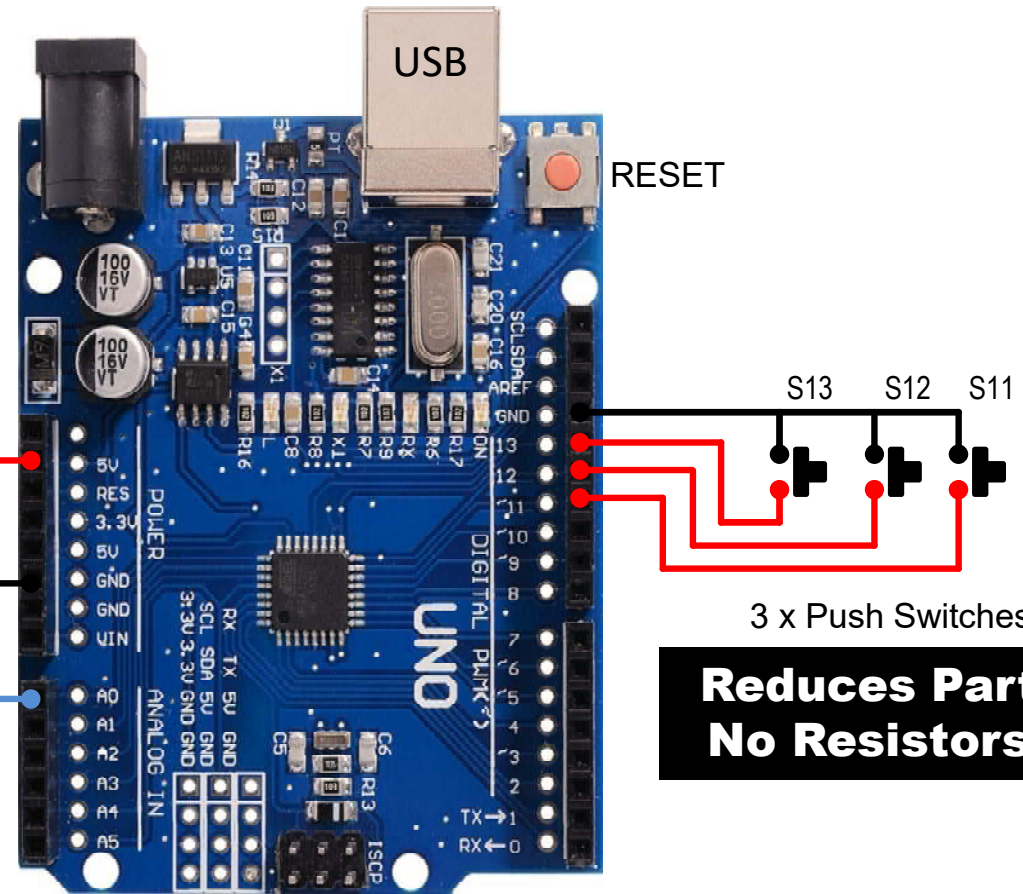


Using Switches:



SA1	SA2	SA3	I/P
0	0	0	0
1	0	0	512
0	1	0	367
0	0	1	208
1	1	0	623
1	0	1	569
0	1	1	459
1	1	1	660

E12 STANDARD RESISTOR SERIES		
1.0	1.2	1.5
1.8	2.2	2.7
3.3	3.9	4.7
5.6	6.8	8.2



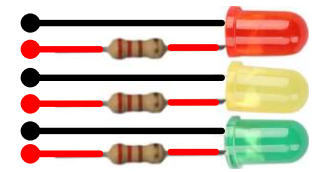
Arduino UNO

3 x Push Switches

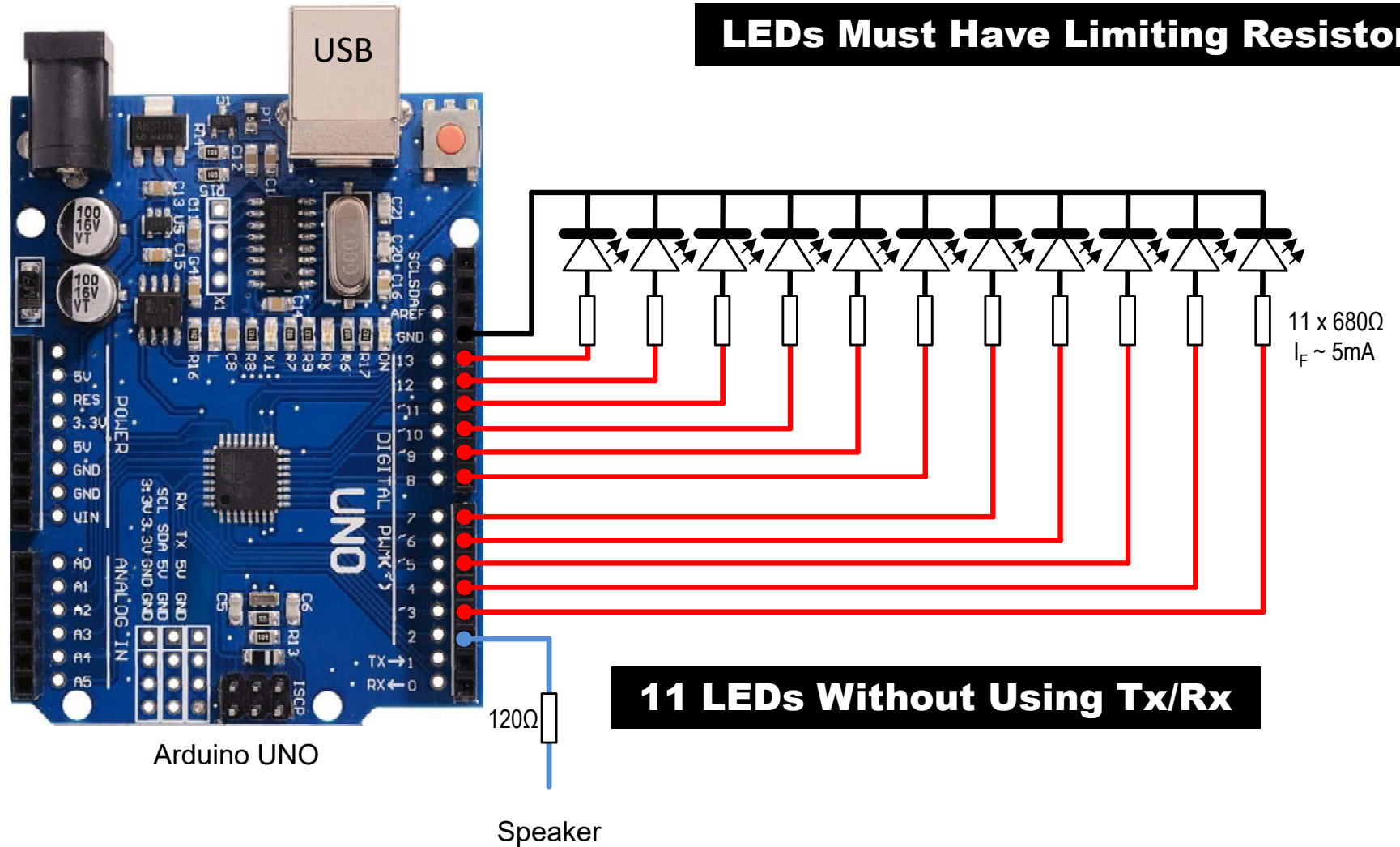
**Reduces Parts Count!
No Resistors Needed**

3 Switches Gives 7 Combinations!

Using LEDs:



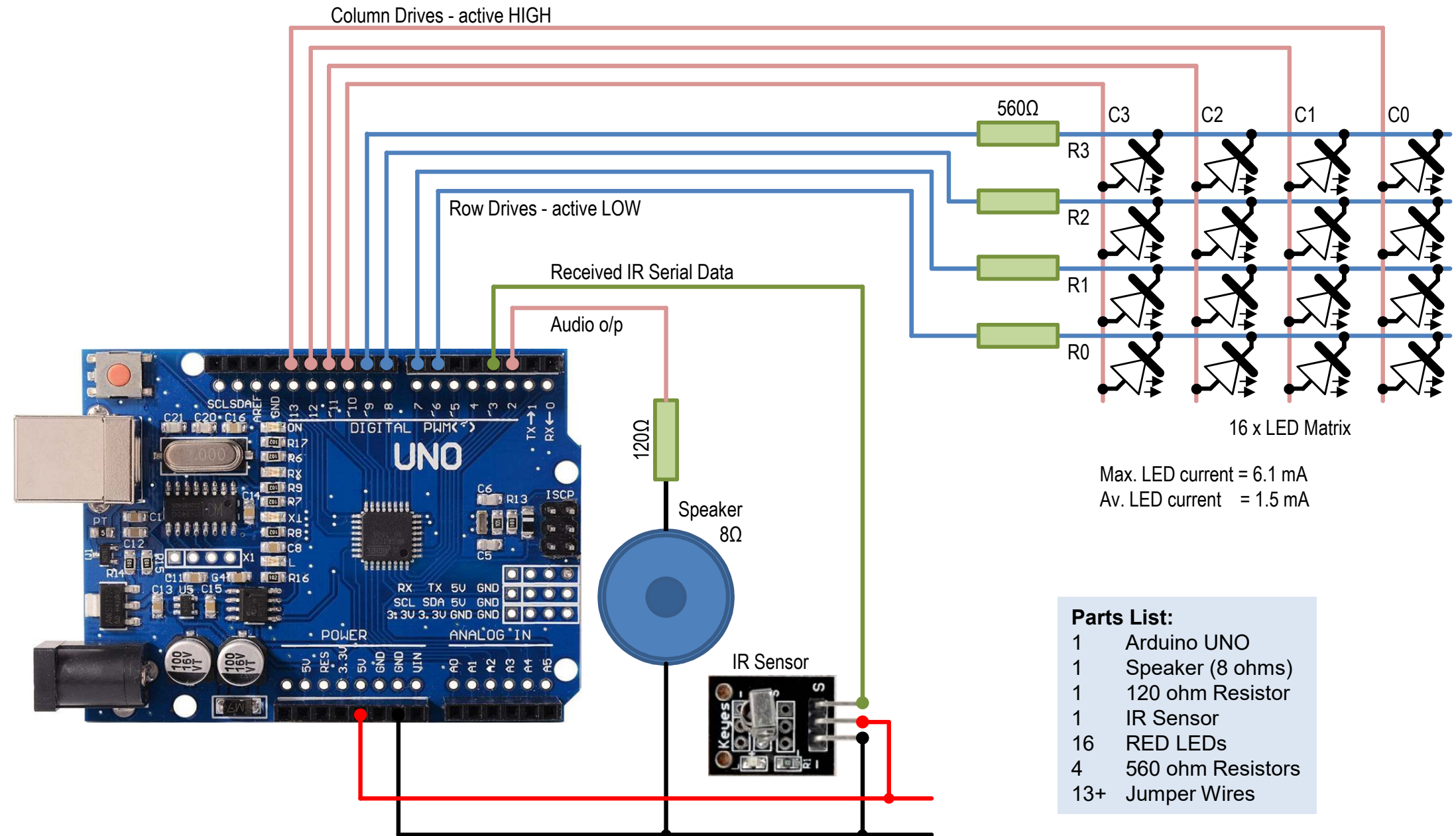
LEDs Must Have Limiting Resistors



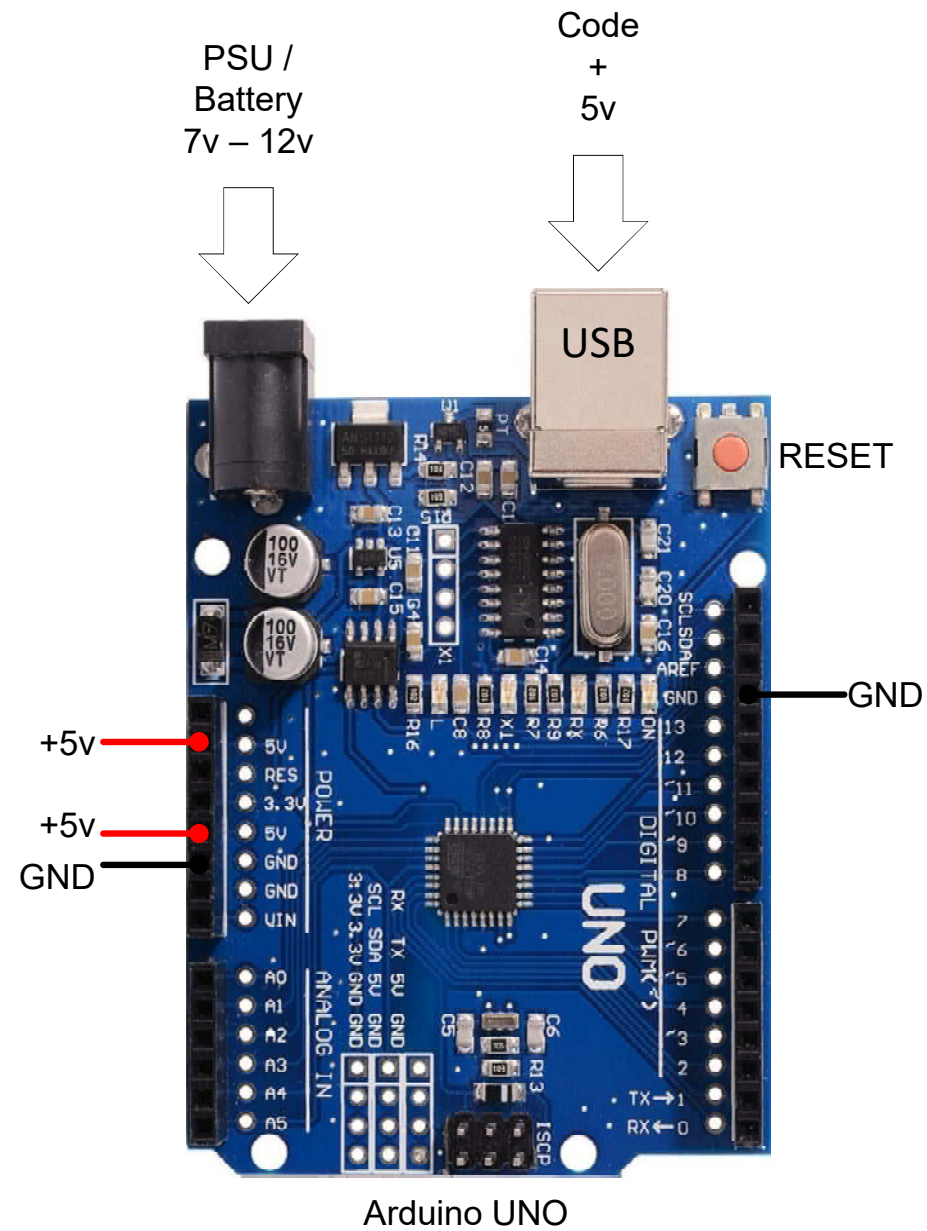
11 LEDs Without Using Tx/Rx

3 x Push Switches Using Ladder Network

My Solution: Multiplexed LED's & Infrared Remote



Draw Your Circuit Diagram:



Why Use Multiplexing?

No. LEDs	Without Multiplexing	With Multiplexing
1	1 Resistor + 2 Wires	1 Resistor + 2 Wires
2	2 Resistors + 3 Wires	1 Resistor + 3 Wires
4	4 Resistors + 5 Wires	2 Resistors + 4 Wires
9	9 Resistors + 10 Wires	3 Resistors + 6 Wires
16	16 Resistors + 17 Wires	4 Resistors + 8 Wires
25	25 Resistors + 26 Wires	5 Resistors + 10 Wires
36	36 Resistors + 37 Wires	6 Resistors + 12 Wires
...	...	...
921,600	921,600 Resistors + 921,601 Wires	720 Resistors + 1,280 Wires

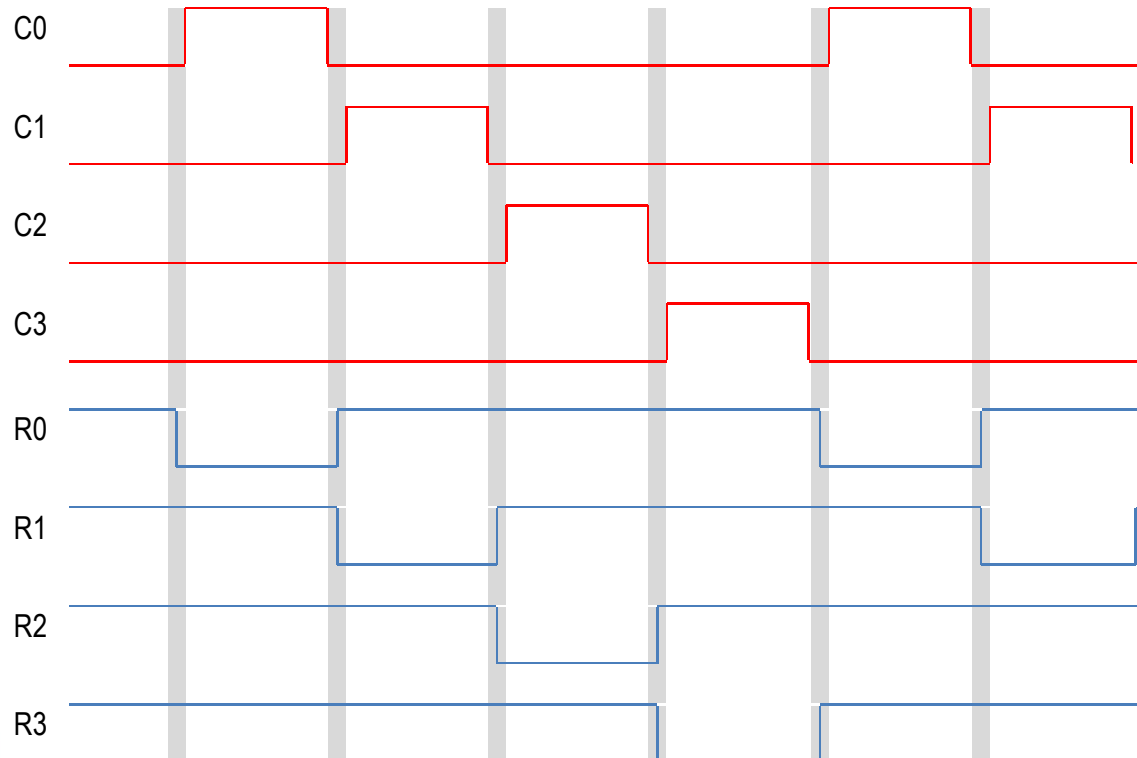
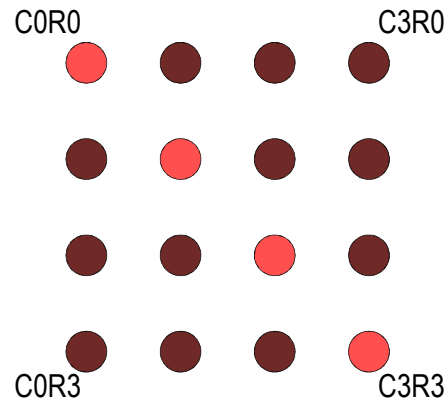
1280 x 720 HD



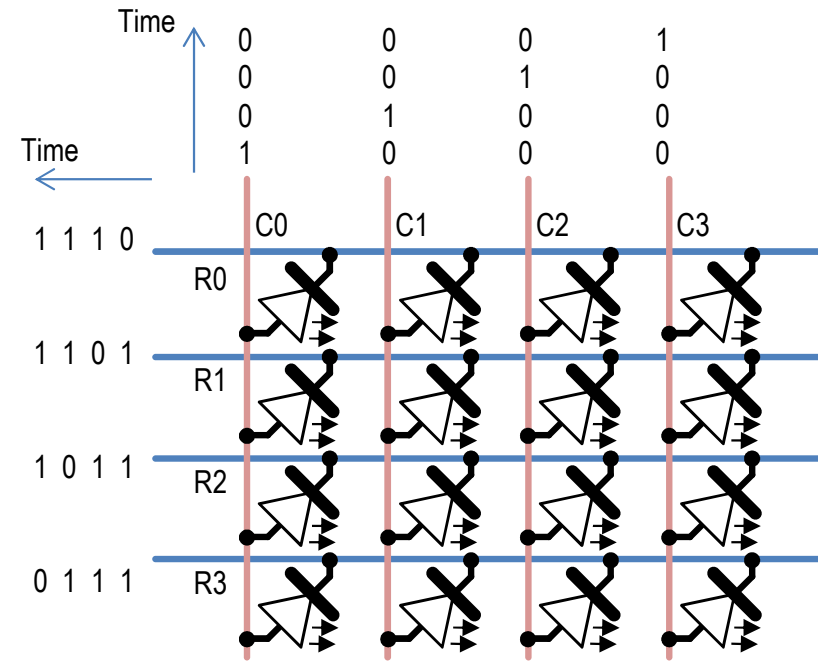
Reduces Component Count!

What Is Multiplexing?

Consider a 4 x 4 display on which we want to draw a diagonal line:



...Generalised Solution...



16 x LED Matrix

We use a 4 x 4 array to store the LED ON/OFF states, and a counter to drive the column lines and address the array to retrieve the LED states as the count progresses.

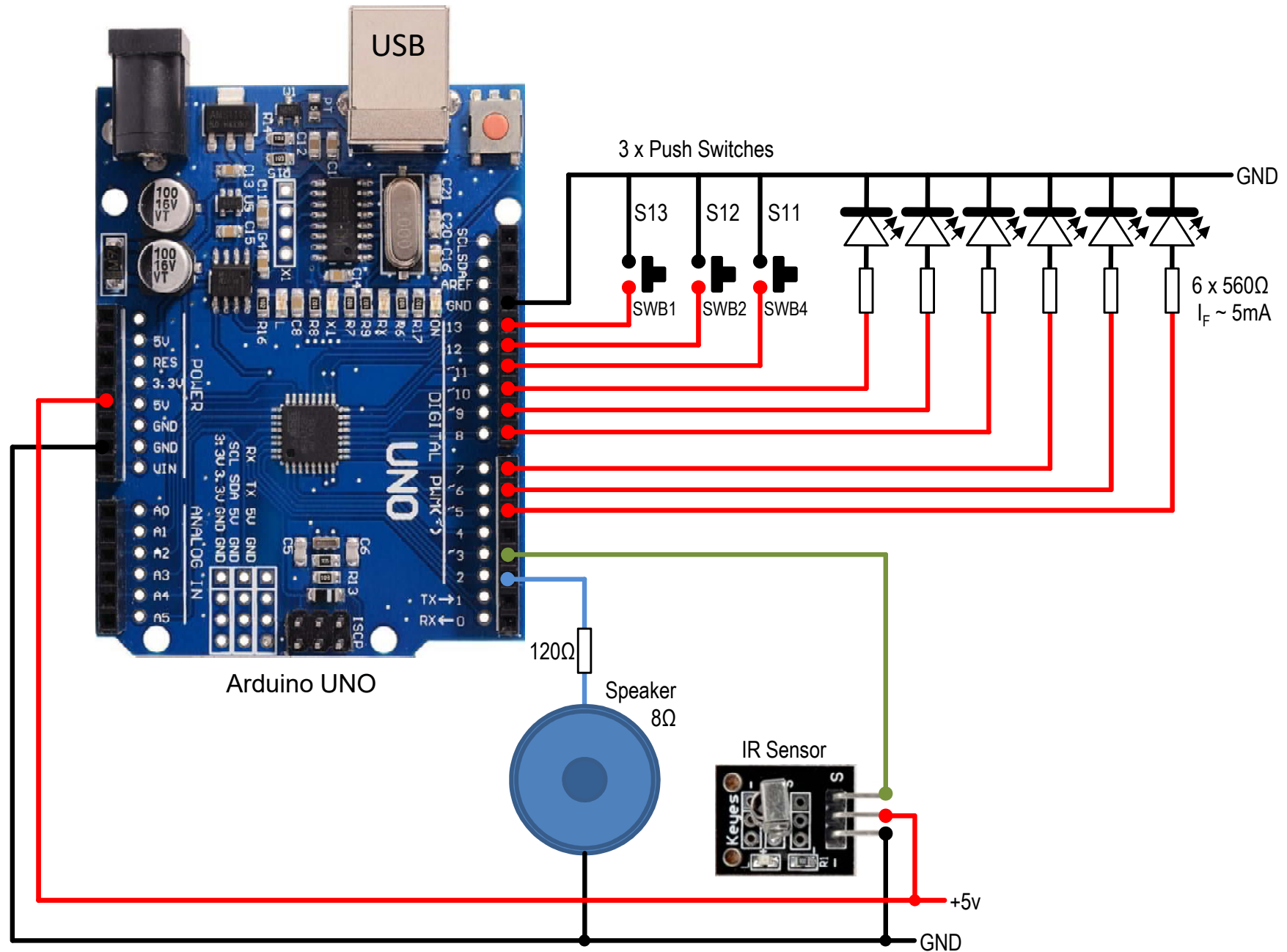
Counter frequency is high to avoid visible flickering of the LEDs.

We run the counter continuously and simply load HIGH or LOW states into the array to turn the respective LED ON or OFF.

Row data changes only when column drives are LOW to avoid ghosting effects.

Column ON period widths can be varied to provide a dimming effect for the whole display.

A Generalised Solution:



Modes Of Operation – After setup

Mode 1 – Twinkle Lamps:

- From 'Start' or from Mode 2 'tune' wait 2 seconds delay
- While in Mode 1:

- Randomly set lamp pattern every 700ms
 - Multiplex lamps
 - Check for serial keyboard
 - Check for infrared codes
 - Change brightness up/down
 - Loop every 1ms
- Loop

Respond to key code

- 0 - 9 - Play a tune → **Mode 2**
- * - toggle lamps ON/OFF

Mode 2 – Play Music:

- From Mode 1
- While in Mode 2:
 - Play the first/next note
 - Change lamp pattern in sequence
 - Wait for note to end:

- Multiplex lamps
- Check for serial keyboard
- Check for infrared codes
- Loop every 1ms

Respond to key code

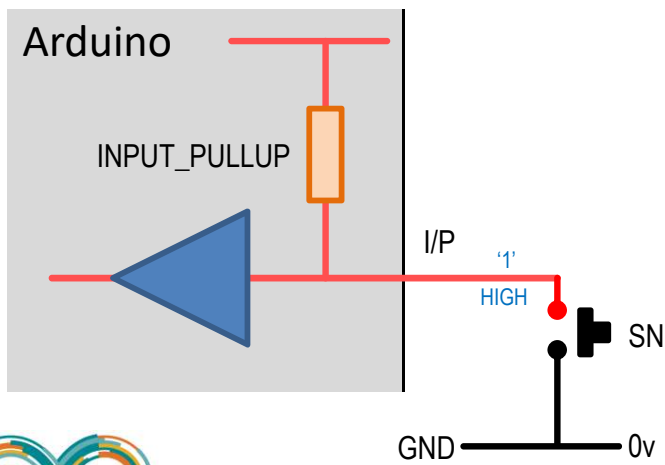
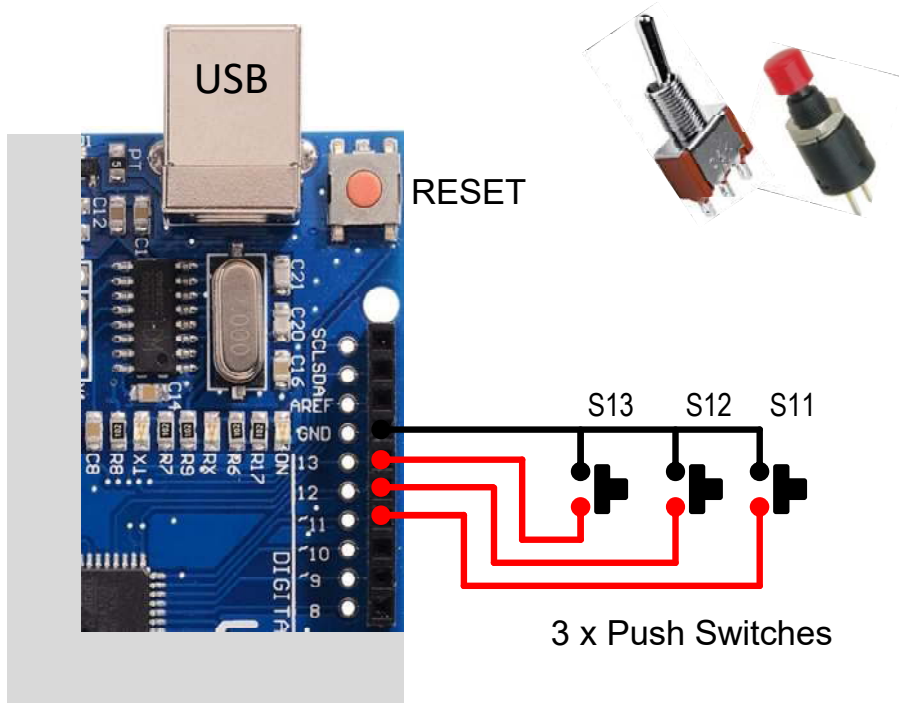
- 0 - 9 - Play a different tune
- # - STOP playing current tune → **Mode 1**
- <> - change pattern sequence

- If more notes to play Loop
- Switch to **Mode 1**

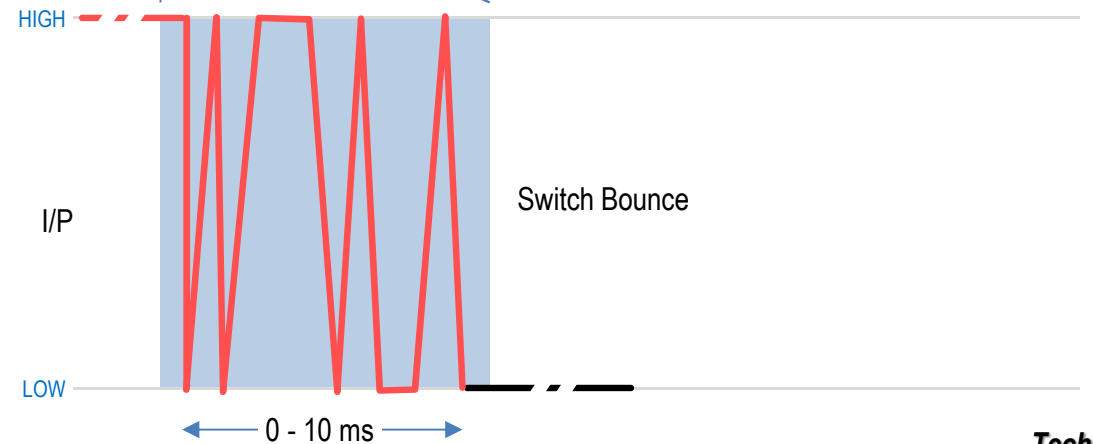
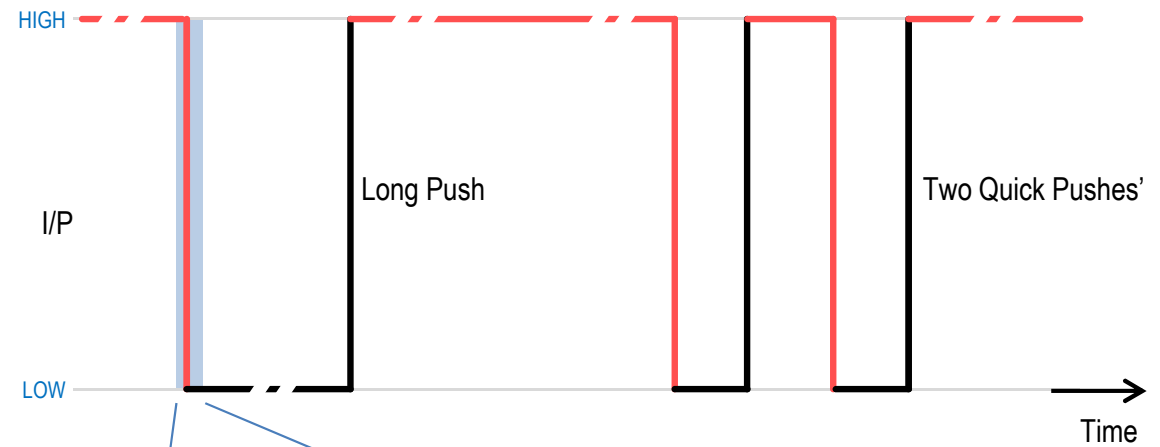
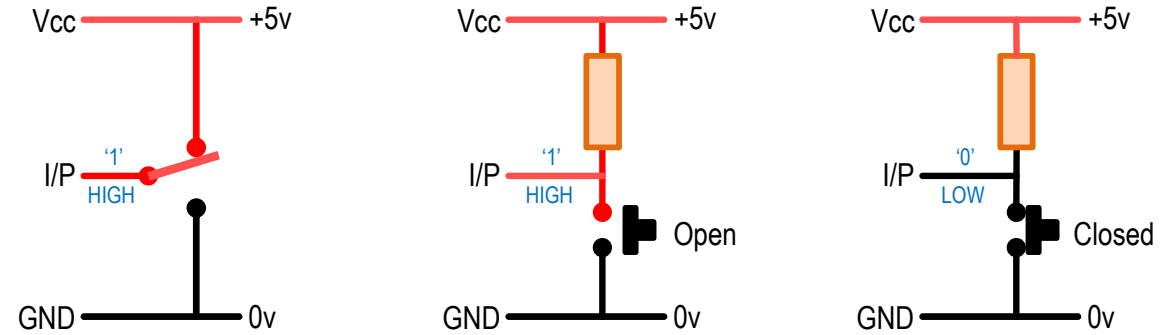
Coding – Essentials:

- Switches
 - If... Else...
- Counters:
 - Inc++; Dec--;
 - Timing delays
 - Flags
- Functions
 - Passing Variables
 - For...
- Random Numbers
- Switch... Case...
- Using Libraries
 - Music
 - Infrared ?
- Arrays []
- Large programs

Using Switches:



...Switches...



Coding_Tech_Ex_00

```
// Switches

// Read switch state on Pin 13.

#define swPin 13 // declare switch i/p pin

void setup() {
  // put your setup code here, to run once:
  pinMode(swPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(swPin) == HIGH) {Serial.println("HIGH");}
  if (digitalRead(swPin) == LOW) {Serial.println("LOW");}
  delay(100);
}
```

Requirement:

- Read switch state on Pin 13.

Coding_Tech_Ex_01

```
// Switches

// Read switch state on Pin 13.
// With internal pull-up resistor

#define swPin 13 // declare switch i/p pin

void setup() {
  // put your setup code here, to run once:
  pinMode(swPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(swPin) == HIGH) {Serial.println("HIGH");}
  if (digitalRead(swPin) == LOW) {Serial.println("LOW");}
  delay(100);
}
```

Requirement:

- Use internal pull-up resistor for stability.

Coding_Tech_Ex_02

```
// Switches

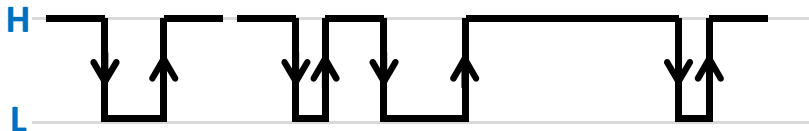
// Read switch transitions on Pin 13.

#define swPin 13 // declare switch i/p pin

int swRead = HIGH; // latest input value
int swState = HIGH; //previous switch state

void setup() {
    // put your setup code here, to run once:
    pinMode(swPin, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop() {
    // put your main code here, to run repeatedly:
    swRead = digitalRead(swPin);
    if (swRead != swState) {
        // a transition has occurred!
        if (swRead == HIGH) {Serial.println("HIGH");}
        else {Serial.println("LOW");}
    }
    swState = digitalRead(swPin);
    delay(100);
}
```



Requirement:

- Read switch transitions on Pin 12.

Coding_Tech_Ex_03

```
// Switches

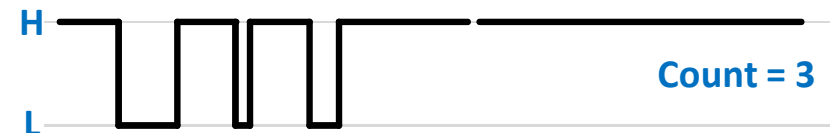
// Read and count switch transitions on Pin 13.
// display count after 2 second pause

#define swPin 13 // declare switch i/p pin

int swCnt = 0; // number of switch presses
int swRead = HIGH; // latest input value
int swState = HIGH; //previous switch state
int timeout = 0; // inactivity timeout timer

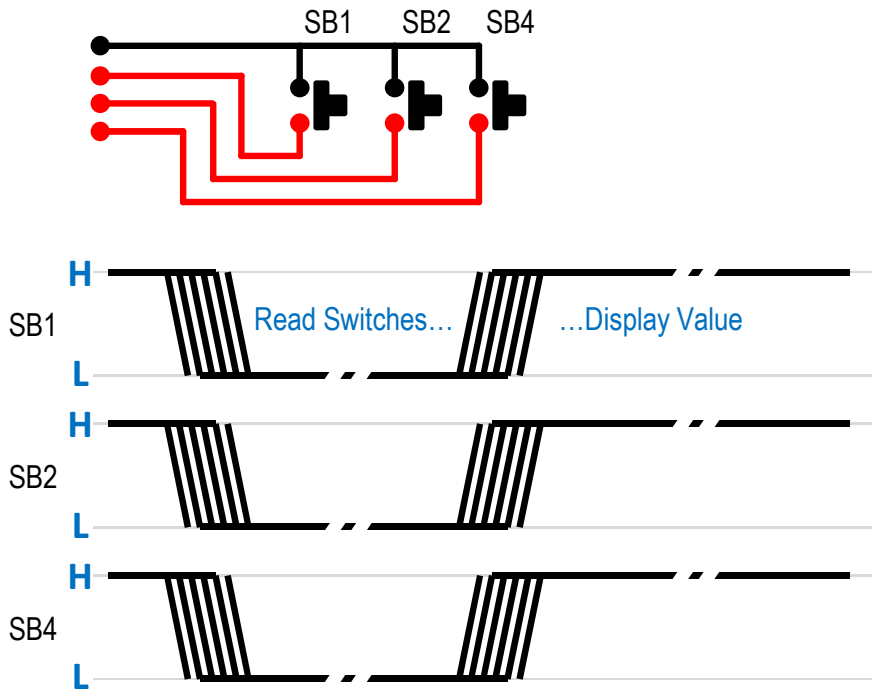
void setup() {
    // put your setup code here, to run once:
    pinMode(swPin, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop() {
    // put your main code here, to run repeatedly:
    swRead = digitalRead(swPin);
    if (swRead != swState) {
        // a transition has occurred!
        if (swRead == LOW) {swCnt++; timeout = 0;}
    }
    swState = digitalRead(swPin);
    timeout++; // increment timeout delay
    if (timeout == 100) {Serial.println(swCnt); swCnt = 0;}
    delay(20);
}
```



Requirement:

- Count and display switch pushes.



SB4	SB2	SB1	
HIGH	HIGH	HIGH	0
HIGH	HIGH	LOW	1
HIGH	LOW	HIGH	2
HIGH	LOW	LOW	3
LOW	HIGH	HIGH	4
LOW	HIGH	LOW	5
LOW	LOW	HIGH	6
LOW	LOW	LOW	7

Coding_Tech_Ex_04

```
// Switches

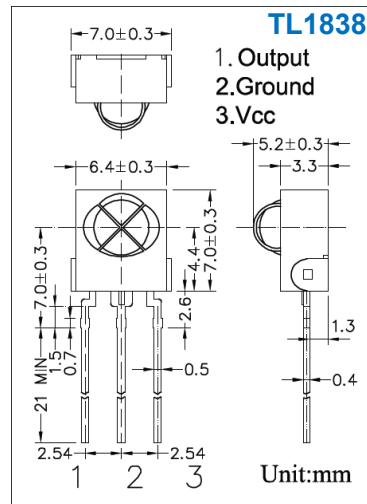
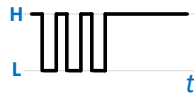
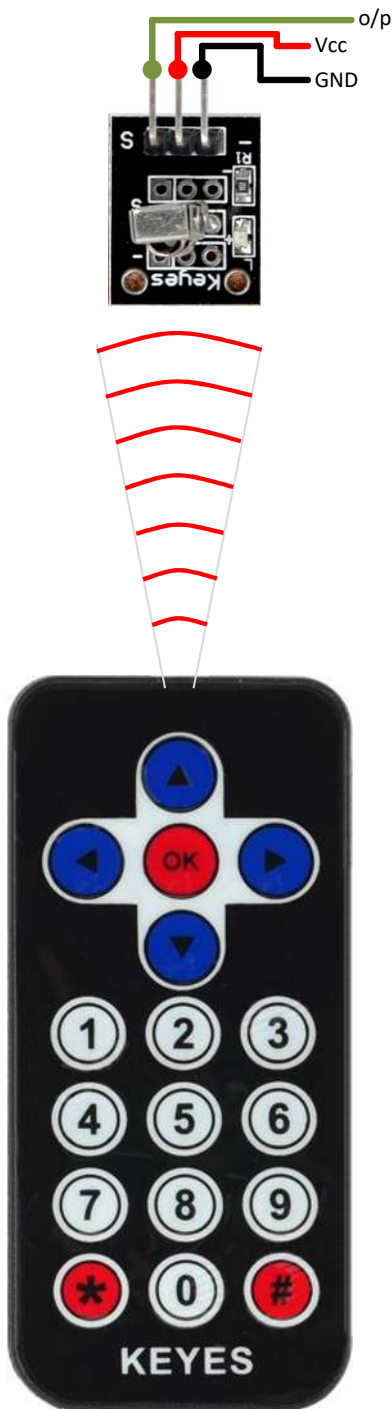
// Read and count switch transitions on Pin 13.
// display count after 2 second pause

#define swB1 13 // declare switch i/p pin
#define swB2 12 // declare switch i/p pin
#define swB4 11 // declare switch i/p pin

int swState = HIGH; //previous switch state
int swVal = 0; // combined value of switches

void setup() {
  // put your setup code here, to run once:
  pinMode(swB1, INPUT_PULLUP);
  pinMode(swB2, INPUT_PULLUP);
  pinMode(swB4, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(swB1) != HIGH || digitalRead(swB2) != HIGH || digitalRead(swB4) != HIGH) {
    // a transition has occurred!
    if (swState == LOW) {
      if (digitalRead(swB1) == LOW) {swVal = (swVal | 1); }
      if (digitalRead(swB2) == LOW) {swVal = (swVal | 2); }
      if (digitalRead(swB4) == LOW) {swVal = (swVal | 4); }
    }
    else {swVal = 0;} // clear the previous old value
    swState = LOW;
  }
  else {
    if (swState == LOW) {Serial.println(swVal);}
    swState = HIGH;
  }
  delay(20);
}
```



Up	= FF629D / 511DBB
Left	= FF22DD / 52A3D41F
OK	= FF02FD / D7E84B1B
Right	= FFC23D / 20FE4DBB
Down	= FFA857 / A3C8EDDB
1	= FF6897 / C101E57B
2	= FF9867 / 97483BFB
3	= FFB04F / F0C41643
4	= FF30CF / 9716BE3F
5	= FF18E7 / 3D9AE3F7
6	= FF7A85 / 6182021B
7	= FF10EF / 8C22657B
8	= FF38C7 / 488F3CBB
9	= FF5AA5 / 449E79F
0	= FF4AB5 / 1BC0157B
*	= FF42BD / 32C6FDF7
#	= FF52AD / 3EC3F1B
Repeat	= FFFFFFFF

Coding_Tech_Ex_05

// Infrared Receiver

// Read an IR code from the receiver on Pin 3.

// Declare libraries

#include "IRremote.h"

#define receiver 3 // pin 3 of IR receiver to pin 3

// Declare objects

IRrecv irrecv(receiver); // create instance of 'irrecv'

decode_results results; // create instance of 'decode_results'

// Declare and initialise global variables

long irResult; //value received

void setup() {

// put your setup code here, to run once:

irrecv.enableIRIn(); // initialise IR receiver

Serial.begin(9600);

}

void loop() {

// put your main code here, to run repeatedly:

readIR();

delay(120);

}

void readIR() {

// read the IR sensor

if (irrecv.decode(&results)) {

// we have received an IR signal

irResult = results.value;

Serial.print("Rx Val: 0x");

Serial.println(irResult, HEX);

irrecv.resume(); // ready to receive the next value

}

}

You will need to install the [Arduino-IRremote-master](https://github.com/z3t0/Arduino-IRremote) library. Download it from here: <https://github.com/z3t0/Arduino-IRremote>

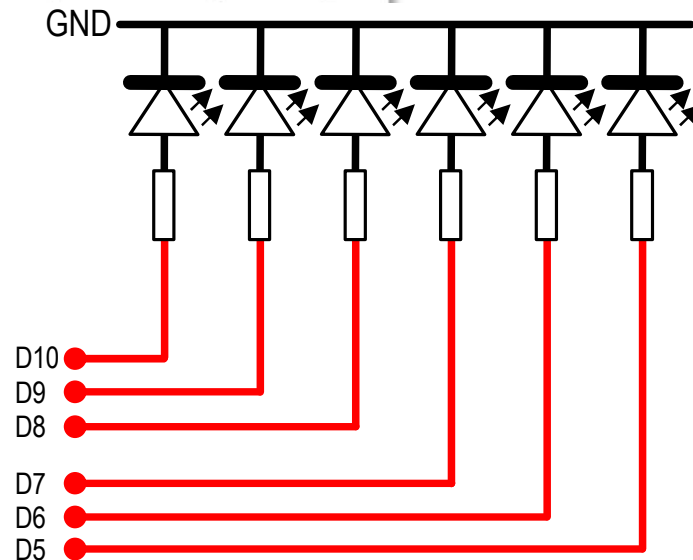
```
// LED's

// Define arrays
#define arrayMax 5 // max array element = size -1
int ledPins[arrayMax + 1] = {10,9,8,7,6,5};
int ledVals[arrayMax + 1] = {0,0,0,0,0,0};

// Declare and initialise global variables
int ledPnt = 0; // LED task pointer

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPins[0], OUTPUT);
  pinMode(ledPins[1], OUTPUT);
  pinMode(ledPins[2], OUTPUT);
  pinMode(ledPins[3], OUTPUT);
  pinMode(ledPins[4], OUTPUT);
  pinMode(ledPins[5], OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  playLights();
  scanLeds();
  delay(500);
}
```



```
void playLights() {
  // illuminate a row of lights sequentially
  // walking lamps
  switch (ledPnt) {
    case 0: setArray(1,0,0,0,0,0); break;
    case 1: setArray(0,1,0,0,0,0); break;
    case 2: setArray(0,0,1,0,0,0); break;
    case 3: setArray(0,0,0,1,0,0); break;
    case 4: setArray(0,0,0,0,1,0); break;
    case 5: setArray(0,0,0,0,0,1); break;
  }
  ledPnt++; if (ledPnt > 5) {ledPnt = 0;}
}

void scanLeds() {
  // performs 1 scan of the LED matrix
  for (int zL = 0; zL <= arrayMax; zL++) {
    if (ledVals[zL] > 0) {digitalWrite(ledPins[zL], HIGH);}
    else {digitalWrite(ledPins[zL], LOW);}
  }
}

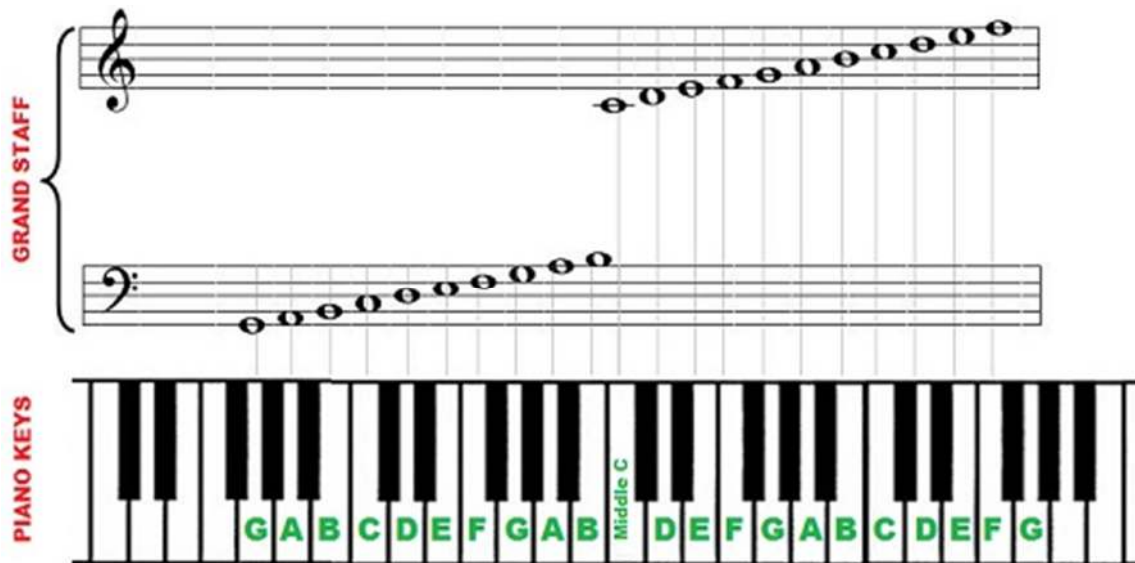
void setArray(int zL0, int zL1, int zL2, int zL3, int zL4, int zL5) {
  // set individual array values by calling one function
  ledVals[0] = zL0; ledVals[1] = zL1; ledVals[2] = zL2;
  ledVals[3] = zL3; ledVals[4] = zL4; ledVals[5] = zL5;
}
```



MUSIC



<http://www.makingmusicfun.net/htm/printit.htm> - Free printable sheet music



Notes:

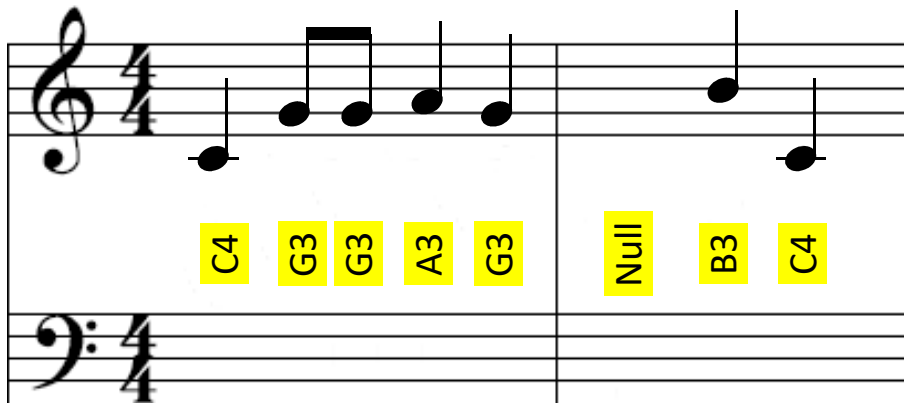
This reference sheet is intended for use with the Arduino 'NewTone' Library - v1.0 dated 01/20/2013
Created by Tim Eckel

Include the '[pitches.h](#)' reference file in your workspace folder, which contains all of the note definitions.

For example to play [Middle C](#) you would use the reference [NOTE_C4](#) in your code.

	A0# 29.135																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			</
--	------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

A Simple Tune:



```
Coding_Tech_Ex_07$ pitches.h

// Simple Melody

// Declare libraries and initialise global variables
#include <NewTone.h>
#include "pitches.h"

// 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(2, 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(8);
  }
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Multiple Tunes:

```
Coding_Tech_Ex_08 NewTone.h pitches.h
// Multiple Melodies

// Declare libraries
#include <NewTone.h>
#include "pitches.h"

// Declare and initialise global variables
#define swB1 13 // B1 switch input
#define speakerPin 2 // speaker is connected to digital pin 2

// Define arrays and variables
#define arrayMax 5 // max LED array element = size -1
#define arraySize 80 // max no. notes in any tune
int melody[arraySize]; // notes in teh melody
int noteDurations[arraySize]; // 4 = quarter, 8 = eighth, etc.:

int keyVal; // input value read from serial port
int loopCnt; // main loop delay counter
int loadPnt = 0; // pointer used to load arrays
int noteDuration = 0; // duration of a note being played in milliseconds
int numNotes = 0; // number of notes. Must be less than arraySize
int pauseBetweenNotes = 10; // delay between a note
int playPnt = -1; // set >= 0 to play melody sequence
int swDel = 20; // relay between switch reads
int swCnt = 0; // number of switch presses
int swRead = HIGH; // latest input value
int swState = HIGH; //previous switch state
int swVal = 0; // temporary switch value or inactivity timer

// -----

void setup() {
  // put your setup code here, to run once:
  pinMode(swB1, INPUT_PULLUP);
  pinMode(speakerPin, OUTPUT);
  digitalWrite(speakerPin, LOW);
  Serial.begin(9600);
}

// -----

void loop() {
  // put your main code here, to run repeatedly:
```

```
Coding_Tech_Ex_08 NewTone.h pitches.h
if (playPnt != -1) {
  // playing a note sequence
  if (playPnt > -1) {playANote();} // load the next note
  loopCnt = pauseBetweenNotes;
  while (loopCnt > 0) {
    // this is the main loop with a 1ms loop time
    swDel--; if (swDel <1) {readSwitch1();} // single switch
    loopCnt--; delay(1);
  }
  if (playPnt == -2) {
    // last note reached so turn off sound
    noNewTone(speakerPin); digitalWrite(speakerPin, LOW);
    playPnt = -1;
  }
} else {
  // not playing a note sequence so just scan keys
  swDel--; if (swDel <1) {readSwitch1();} // single switch
  delay(1);
}

// -----

void keyAction() {
  // called when a switch has been pressed
  switch (keyVal) {
    case 1: loadMelody0(); break;
    case 2: loadMelody1(); break;
  }
}

// -----

void loadArray(int zN, int zD) {
  // load values into arrays at position zP
  melody[loadPnt] = zN;
  noteDurations[loadPnt] = zD;
  loadPnt++;
}

// -----
```

```
Coding_Tech_Ex_08 NewTone.h pitches.h
void loadMelody0() {
  // load a 1 note melody
  noNewTone(speakerPin); // stop current melody
  loadPnt = 0;
  loadArray(NOTE_C4, 4); // 1st note..

  numNotes = loadPnt; // length of melody addressed, numNotes - 1
  playPnt = 0; // point to first note to start playing melody
}

// -----

void loadMelody1() {
  // load a 2 note melody
  noNewTone(speakerPin); // stop current melody
  loadPnt = 0;
  loadArray(NOTE_C4, 4); // 1st note..
  loadArray(NOTE_C4, 4); // 2nd note..

  numNotes = loadPnt; // length of melody addressed, numNotes - 1
  playPnt = 0; // point to first note to start playing melody
}

// -----

void playANote() {
  // play a note at the current pointer then increment pointer

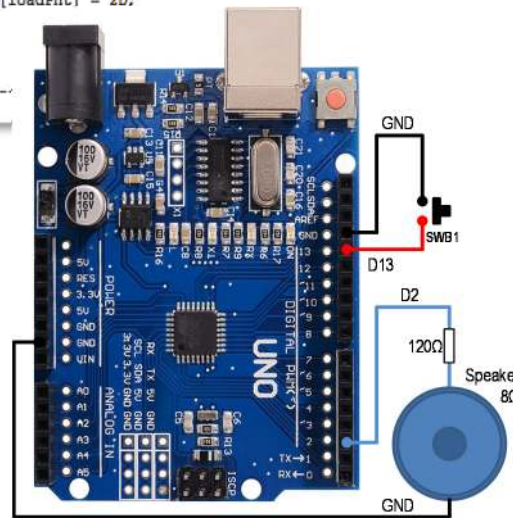
  // calculate note duration by taking 1 second divided by note type.
  //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
  noteDuration = 1000 / noteDurations[playPnt];
  NewTone(speakerPin, melody[playPnt], noteDuration);
  // to distinguish the notes, set a minimum time between them.
  // the note's duration + 10% seems to work well:
  pauseBetweenNotes = noteDuration * 1.1;
  playPnt++;
  if (playPnt >= numNotes) {playPnt = -2;} // reached final note
}

// -----

void readSwitch1() {
  // reads a single switch and counts the number of presses
  swDel = 20; // come here every 20ms
  swRead = digitalRead(swB1);
  if (swRead != swState) {
    // a transition has occurred!
    if (swRead == LOW) {swCnt++; swVal = 0;}
  }
  swState = digitalRead(swB1);
  swVal++; // increment timeout delay
  if (swVal == 50) {
    // no more presses after 1 seconds
    keyVal = swCnt; swCnt = 0;
    keyAction();
  }
}
```

Can you:

- Add more notes to a melody?
- Add more melodies?



Modes Of Operation – After setup

Mode 1 – Twinkle Lamps:

- From 'Start' or from Mode 2 'tune' wait 2 seconds delay
- While in Mode 1:

- Randomly set lamp pattern every 700ms
- Light lamps
- Check for serial keyboard
- Check for infrared codes
- Change brightness up/down
- Loop every 1ms
- Loop

Respond to key code

- 0 - 9 - Play a tune → **Mode 2**
- * - toggle lamps ON/OFF

Mode 2 – Play Music:

- From Mode 1
- While in Mode 2:
- Play the first/next note
- Change lamp pattern in sequence
- Wait for note to end:
- Light lamps
- Check for serial keyboard
- Check for infrared codes
- Loop every 1ms
- If more notes to play Loop
- Switch to **Mode 1**

Respond to key code

- 0 - 9 - Play a different tune
- # - STOP playing current tune → **Mode 1**
- <> - change pattern sequence

Novelty Source Code:

Features:

- Uses libraries which must be installed
- Pre-loaded with 10 Christmas carols
- Includes a 'Tempo' variable
- Reads inputs from:
 - Serial port keyboard characters
 - IR detector
 - 1 - 3 or more switches
- Drives 6 LEDs
- Dims LEDs for a 'twinkle' effect

Adjust the code to:

- Match your switch and LED count
- Add to or change the 10 melodies

```
Xmas_Novelty_3SW_NoMux_v00  NewTone.h  pitches.h
// #####
//
// Christmas Novelty File v1.0
//
// Released: 18/10/2015
//
// Author: TechKnowTone
//
// #####
/*
   This software is furnished "as is", without technical support, and
   with no warranty, expressed or implied, as to its usefulness for any
   purpose.

   Elements of this software were taken from the public domain. It uses
   libraries for tone generation and infrared reception which are readily
   available and must be installed on your system. It plays several
   melodies which are selected from a remote control. Updated with latest
   IRremote library codes.

   This version reads 3 switch inputs and drives 6 LEDs. It does not use
   multiplexed LEDs as in the original design.

Key  Function:
1    play melody 0 - Jingle Bells
2    play melody 1 - God Rest Ye Merry Gentlement
3    play melody 2 - Hark The Herald Angels Sing
4    play melody 3 - Old King Cole
5    play melody 4 - Silent Night
6    play melody 5 - The First Noel
*    toggle LED twinkle on/off when silent to save power
#    STOP the current musical score from playing
>    next LED pattern
<    previous LED pattern
*/
// Declare libraries
#include "IRremote.h"
#include <NewTone.h>
#include "pitches.h"

// Declare and initialise global variables
#define arraySize 80 // max no. notes in a tune
```



Enjoy!

