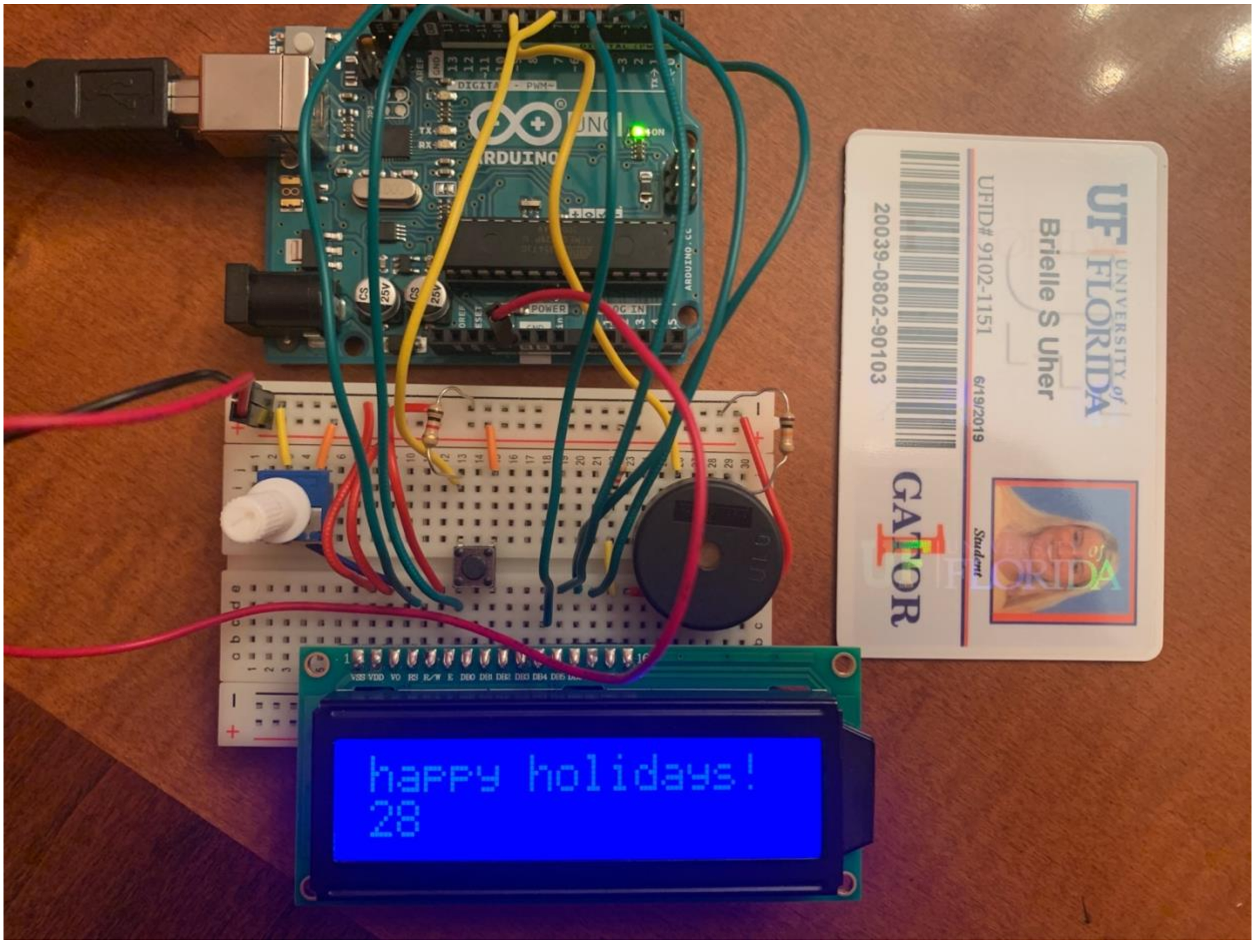
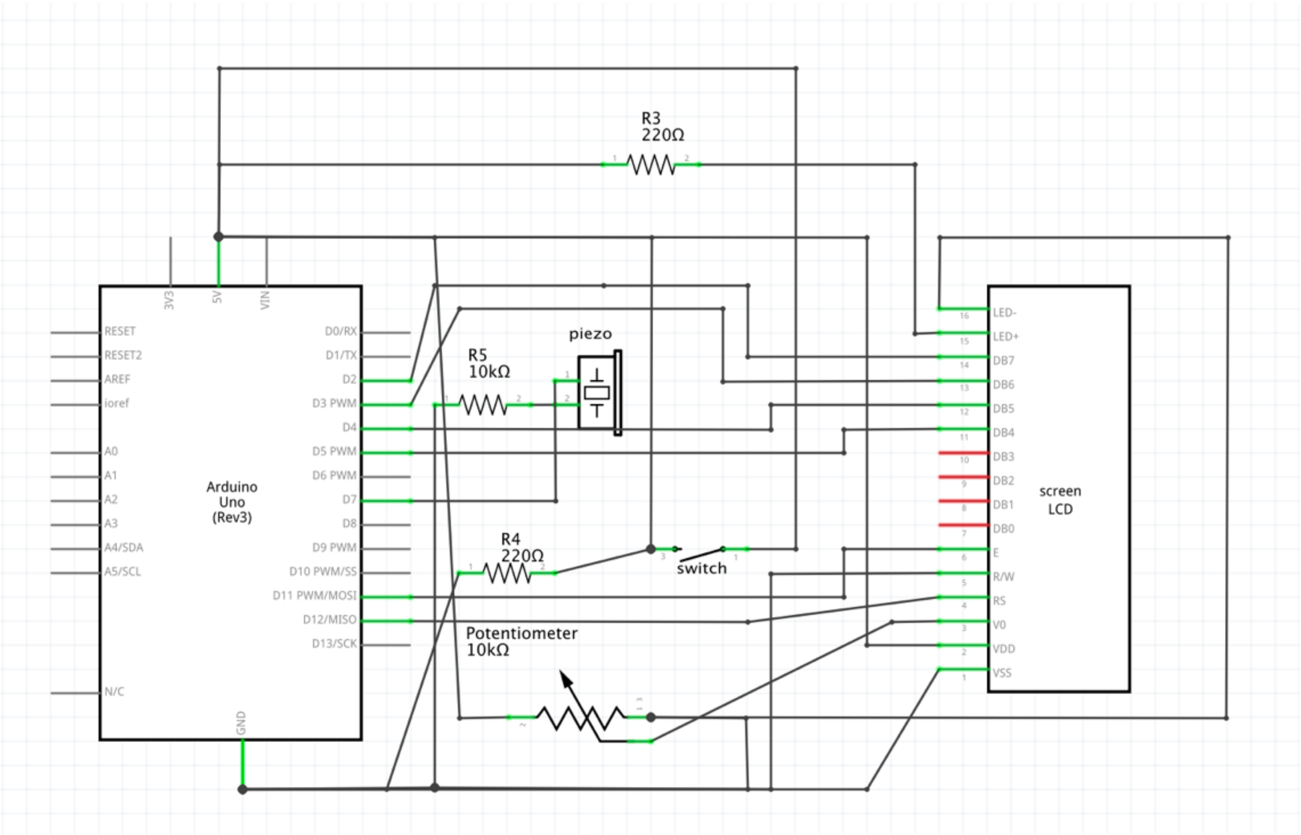
First Name: Brielle
Last Name: Uher
UF-ID Number: 91021151



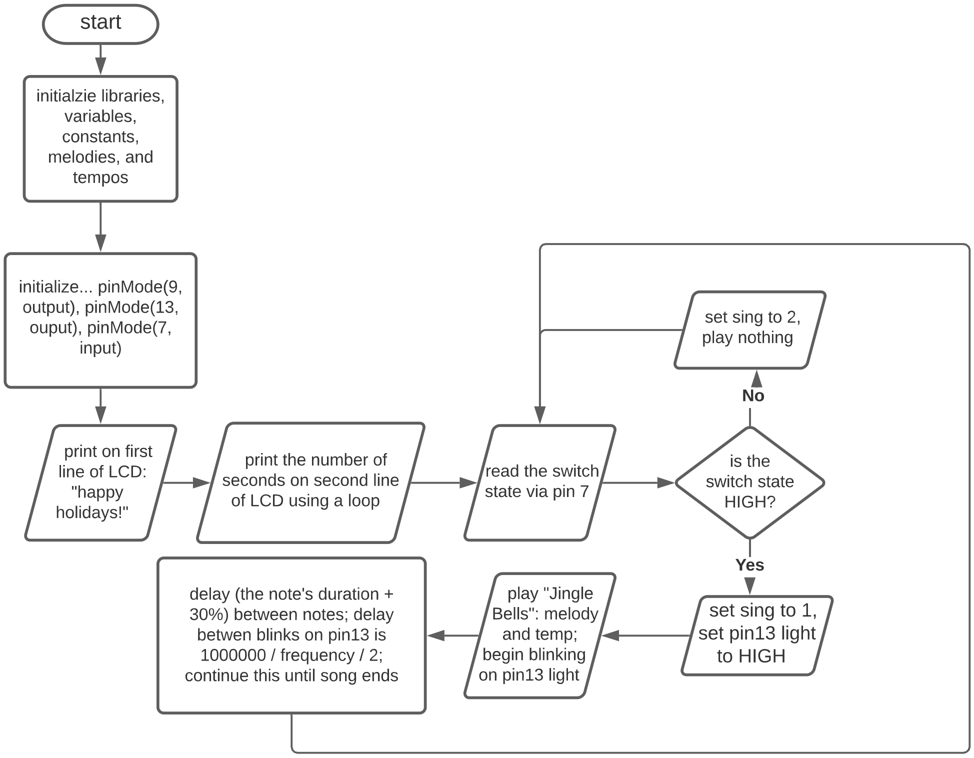
My circuit uses a switch, piezo and ‘pitches.h’ library to play the song “Jingle Bells” (getting in the holiday spirit!). The song begins to play when the switch is pressed. I adapted this Arduino Uno circuit and code from the linked website: https://create.arduino.cc/projecthub/joshi/piezo-christmas-songs-fd1ae9?ref=platform&ref_id=424_trending_part___&offset=85 . This code includes the ‘pitches.h’ library which is used to create the pitches of different notes by reading a value between 0-1023. In addition, the little yellow light near digital port 13 lights up to the beat of the song. This is controlled using only code, no external wires are connected to this port. The entire code includes three songs that can be changed using three different switches. While I got this code/circuit to work, I decided to only play one song because I added more components to the circuit causing the breadboard to become crowded. The components that I added were the LCD (Liquid Crystal Display) and potentiometer which I adapted from the linked website: https://www.arduino.cc/en/Tutorial/LibraryExamples/HelloWorld . This website displays “hello world!” but I changed this to “happy holidays!” The potentiometer is used to control the contrast of the LCD. The ‘LiquidCrystal.h’ library is needed in the code to allow the LCD to work. Also, under the words “happy holidays!” there is a timer which counts up from 0. I integrated the two sets of code by putting the ‘void setup’ codes together and the ‘void loop’ codes together. Due to merging two circuits, I had to change some pin numbers in the original code.

Part:	**Quantity:**
Arduino Uno	1
Bread Board	1
LCD	1
Potentiometer	1
Piezo	1
Switch	1
220-ohm resistor	2
10-kilohm resistor	1
Wires	19
USB connector	1

Schematic done using Fritzing:



Flowchart done using LucidChart:


start
initialzie libraries, variables, constants, melodies, and tempos
initialize... pinMode(9, output), pinMode(13, ouput), pinMode(7, input)
print on first line of LCD: "happy holidays!"
print the number of seconds on second line of LCD using a loop
read the switch state via pin 7
is the switch state HIGH?
No
set sing to 2, play nothing
Yes
set sing to 1, set pin13 light to HIGH
play "Jingle Bells": melody and temp; begin blinking on pin13 light
delay (the note's duration + 30%) between notes; delay betwen blinks on pin13 is 1000000 / frequency / 2; continue this until song ends

Arduino Code:

```
/*
  LiquidCrystal Library - Hello World

 Demonstrates the use a 16x2 LCD display.  The LiquidCrystal
 library works with all LCD displays that are compatible with the
 Hitachi HD44780 driver. There are many of them out there, and you
 can usually tell them by the 16-pin interface.

 This sketch prints "Hello World!" to the LCD
 and shows the time.

  The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)

 Library originally added 18 Apr 2008
 by David A. Mellis
 library modified 5 Jul 2009
 by Limor Fried (http://www.ladyada.net)
 example added 9 Jul 2009
 by Tom Igoe
 modified 22 Nov 2010
 by Tom Igoe
 modified 7 Nov 2016
 by Arturo Guadalupi

 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/LiquidCrystalHelloWorld

*/

// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


/************************************************
 * 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
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
```

```
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978


/*
  Arduino Christmas Songs

  Based on a project and code by Dipto Pratyaksa, updated on 31/3/13

  Modified for Christmas by Joshi, on Dec 17th, 2017.
*/


#define melodyPin 9

// Jingle Bells

int melody[] = {
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_G5, NOTE_C5, NOTE_D5,
  NOTE_E5,
  NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5,
  NOTE_F5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_D5, NOTE_D5, NOTE_E5,
  NOTE_D5, NOTE_G5
};

int tempo[] = {
  8, 8, 4,
  8, 8, 4,
  8, 8, 8, 8,
  2,
  8, 8, 8, 8,
  8, 8, 8, 16, 16,
  8, 8, 8, 8,
  4, 4
};


int switchOne = 0;

void setup(void) {
  pinMode(9, OUTPUT); // Buzzer
  pinMode(13, OUTPUT); // Led indicator when singing a note
  pinMode(7, INPUT);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("happy holidays!");
}

void loop() {
    // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
  //song
  switchOne = digitalRead(7);
  if (switchOne == HIGH) {
    sing(1);
  } else if (switchOne == LOW) {
    sing(2);
  }
}

int song = 0;

void sing(int s) {
  // iterate over the notes of the melody:
  song = s;
  if (song == 1) {
    Serial.println(" 'Jingle Bells'");
    int size = sizeof(melody) / sizeof(int);
    for (int thisNote = 0; thisNote < size; 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 / tempo[thisNote];

      buzz(melodyPin, 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:
      buzz(melodyPin, 0, noteDuration);

    }
  }
}

void buzz(int targetPin, long frequency, long length) {
  digitalWrite(13, HIGH);
  long delayValue = 1000000 / frequency / 2; // calculate the delay value between transitions
  //// 1 second's worth of microseconds, divided by the frequency, then split in half since
  //// there are two phases to each cycle
  long numCycles = frequency * length / 1000; // calculate the number of cycles for proper timing
  //// multiply frequency, which is really cycles per second, by the number of seconds to
  //// get the total number of cycles to produce
  for (long i = 0; i < numCycles; i++) { // for the calculated length of time...
    digitalWrite(targetPin, HIGH); // write the buzzer pin high to push out the diaphram
    delayMicroseconds(delayValue); // wait for the calculated delay value
    digitalWrite(targetPin, LOW); // write the buzzer pin low to pull back the diaphram
    delayMicroseconds(delayValue); // wait again or the calculated delay value
  }
  digitalWrite(13, LOW);

}
```