

Sean Sands & Max Henderson | CS 351 | February 24, 2019

CS 351 Assignment 2

Keyboarduino

# Get Started Right Away

When you click this placeholder text, just start typing to replace it all. But don’t do that just yet!

This placeholder includes tips to help you quickly format your report and add other elements, such as a chart, diagram, or table of contents. You might be amazed at how easy it is.

# Make It Gorgeous

* Need a heading? On the Home tab, in the Styles gallery, just click the heading style you want. Notice other styles in that gallery as well, such as for a quote or a numbered list.
* You might like the cool, blue ice pond on the cover page as much as we do, but if it’s not ideal for your report, right-click it and then click Change Picture to add your own photo.
* Adding a professional-quality graphic is a snap. In fact, when you add a chart or a SmartArt diagram from the Insert tab, it automatically matches the look of your document.

# Give It That Finishing Touch

Need to add a table of contents or a bibliography? No sweat.

## Add a Table of Contents

It couldn’t be easier to add a table of contents to your report. On the Insert tab, click Cover Page to see cover page designs that include a table of contents page — look for TOC.

Just click to insert one of these and you’ll be prompted to update the TOC. When you do, text you formatted using Heading 1, Heading 2, and Heading 3 styles is automatically added.

## Add a Bibliography

On the References tab, in the Citations & Bibliography group, click Insert Citation for the option to add sources and then place citations in the document.

When you’ve added all the citations you need for your report, on the References tab, click Bibliography to insert a formatted bibliography in your choice of styles.

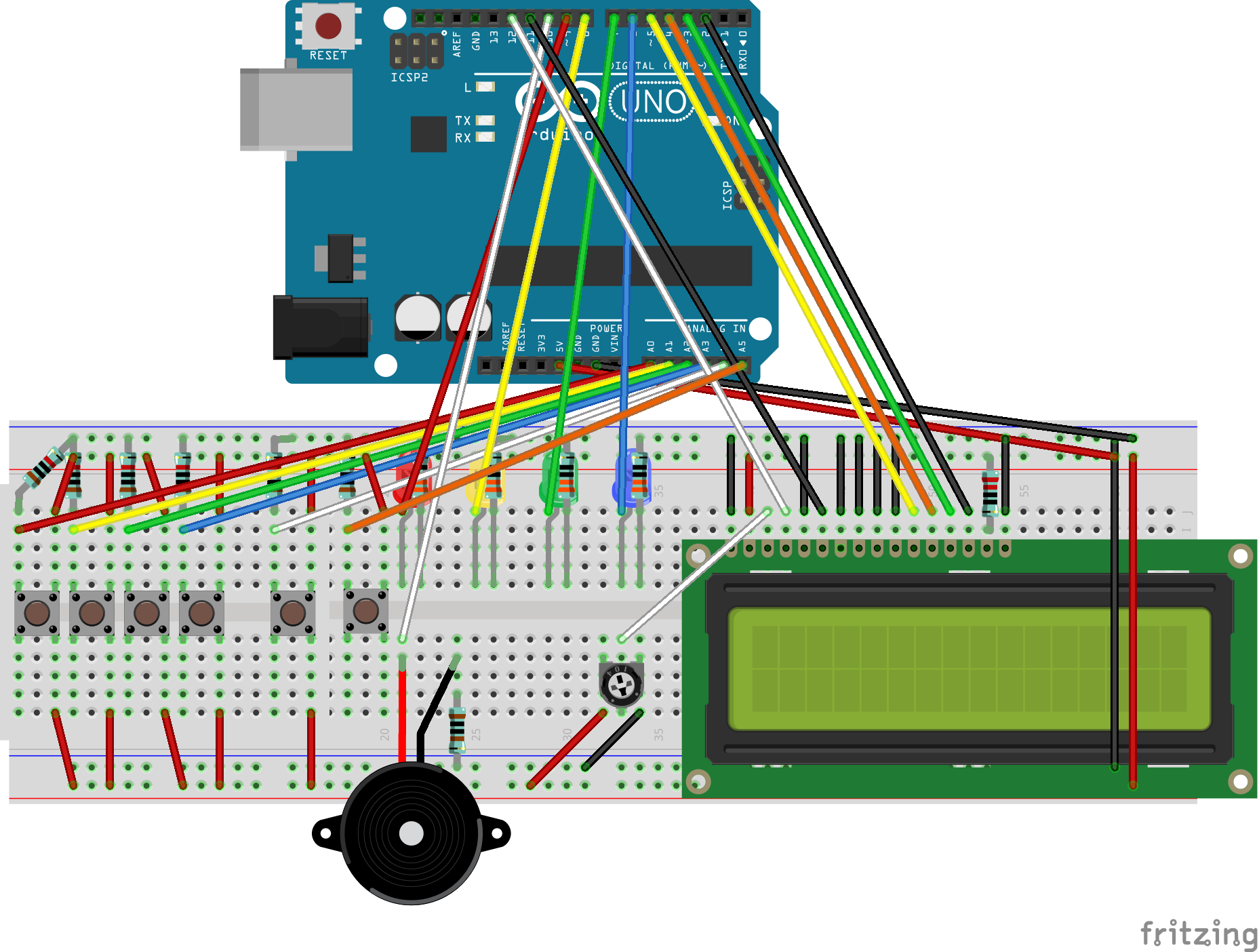
And you’re done. Nice work!

# Function

Description of interactive function here

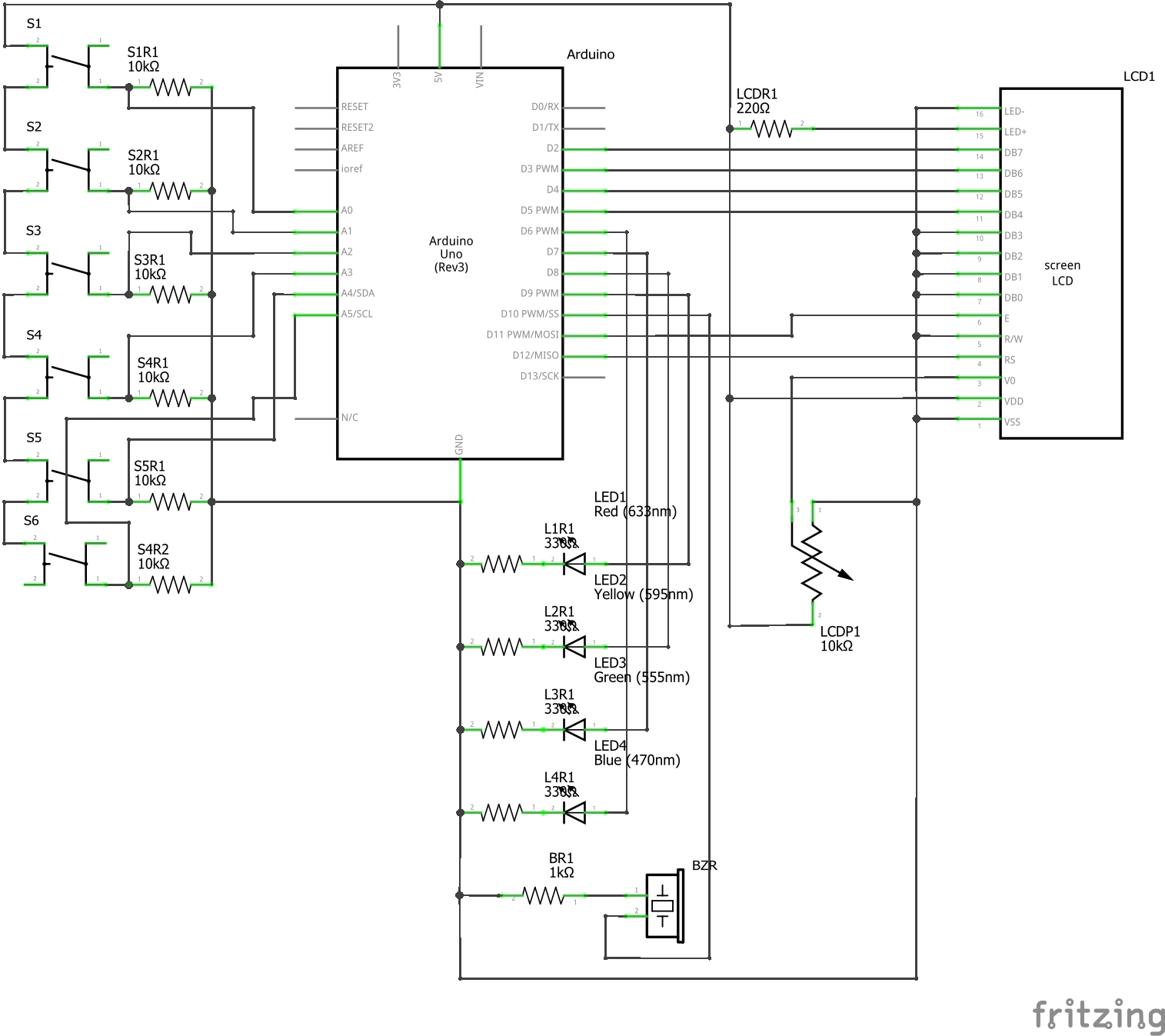
# Circuit

Intro to description of circuit heredfsd



Textd

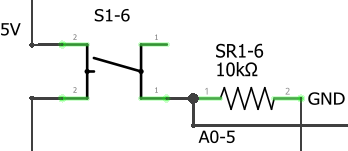
## SCHEMATIC



The sub circuits of this assignment implementation are as follows:

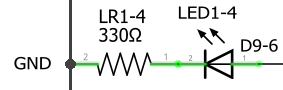
* 6 Button Circuits
* 4 LED Circuits
* 1 Buzzer Circuit
* 1 LCD Circuit

## Button Circuits



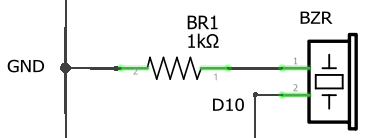
Each button is connected on one end to an input pin and to ground through a 10k resistor, and to 5V on the other end. When open, the input pin is pulled to ground. When closed, the input pin is pulled to 5V.

## LED Circuits



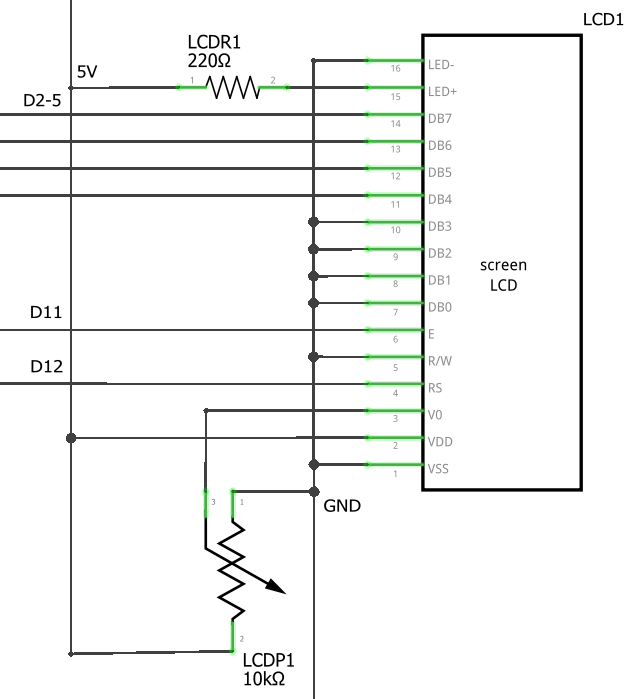
Each LED is connected on the positive end to an output pin and to ground through a 330 resistor on the negative end. When the output pin is on (5V), the LED will be on. When the output pin is off (GND), the LED will be off.

## Speaker Circuit



The buzzer is connected on one end to an output pin and to ground through a 1k resistor on the other end to ensure the pin is not damaged. When a waveform is passed by the output pin, the buzzer emits a tone with the frequency of that waveform.

## LCD Circuit



5V connects to VDD(2) directly, and to LED+(15) through a 220 resistor. Power is provided to the unit through VDD, and the backlight is powered through LED+. 5V also connects to one end of the 10k potentiometer.

GND connects to VSS(1), R/W(5), DB0-3(7-10), and LED-(16). GND also connects to the other end of the 10k potentiometer. VSS completes the power circuit with VDD. LED- completes the circuit with LED+. DB0-3 are unused in 4 bit mode, so are tied to ground to avoid spurious signals. The LCD unit is only used in write mode in this project, so R/W is tied to ground.

D2-5 connect to DB7-4(14-11). These 4 pins are used by the Arduino to communicate with the LCD unit.

D11 connects to E(6). It is a falling edge triggered clock.

D12 connects to RS(4). It determines whether the LCD unit will parse D2-5 as commands or data.

The middle connection of the 10k potentiometer is connected V0(3). As a potentiometer, the voltage can range between the voltages of the other two connections: in this case, 0V and 5V. The potentiometer can be adjusted to alter the contrast on the LCD between text and screen.

# Code

Intro to description of code here

## Preprocessor Commands

The frequencies of the four notes used by the assignment are defined during preprocessing.

The LiquidCrystal library is included to drive the LCD Unit.

## Global Values

### Constants

int notes[]: array of note frequencies used for tones  
int noteDuration: duration of notes when played by replay(), in 2 ms ticks (125, so 250ms)  
int buttonPins[]: array of pins associated with the six buttons (A0-A5)  
int noteLedPins[]: array of pins associated with the four LEDs (9,8,7,6)  
int speakerPin: pin associated with the speaker (10)  
int numButtons: number of buttons, used for sanity checks by some functions  
int rs,en,d4,d5,d6,d7: pins used by the lcd (12,11,5,4,3,2)

### Variables

int buttonDebounceStates[]: array of integers used as rolling buffers by debounceBtn(), volatile  
bool buttonPressedStates[]: array of bools written by debounceBtn() and read by readBtn(), volatile  
bool lastButtonPressedStates[]: array of bools used by readBtn(), compared to the above array  
pressing: current actively pressed button, determined by press() when called by readBtn()  
playing: index of the next note to play when replaying, used by replay(), modified by press()  
playTime: time playing the current replayed note, incremented by the ISR, read by replay(), volatile  
played[]: array of characters representing recorded notes,  
 written by record(), read by replay(), cleared by reset()  
playedSize: number of elements in above array  
playedCap: max number of elements in above array

### Objects

lcd: a LiquidCrystal object that drives the LCD Unit

## Functions

rawBtnPressed(int btn): wrapper for digitalRead()s of buttonPins, called by debounceBtn()  
debounceBtn(int btn):   
noPlay(): stops tone() and turns off all LEDs  
play(char note): plays a note and lights the associated LED on a valid character, else noPlay()  
record (char n): adds a char to played(if there’s room), adds a char to the LCD (if there’s room)  
replay(): plays the next note in played if playTime>noteDuration,  
 called by loop() when pressing is 4, or press() (which also sets playTime>noteDuration)  
reset(): clears played  
press(): does the appropriate action based on a change in button state  
readBtn(int btn): compares buttonPressedState to lastButtonPressedState,  
 assigns and calls press() when they aren’t equal

## Setup

## Loop

## Interrupt Service Routine