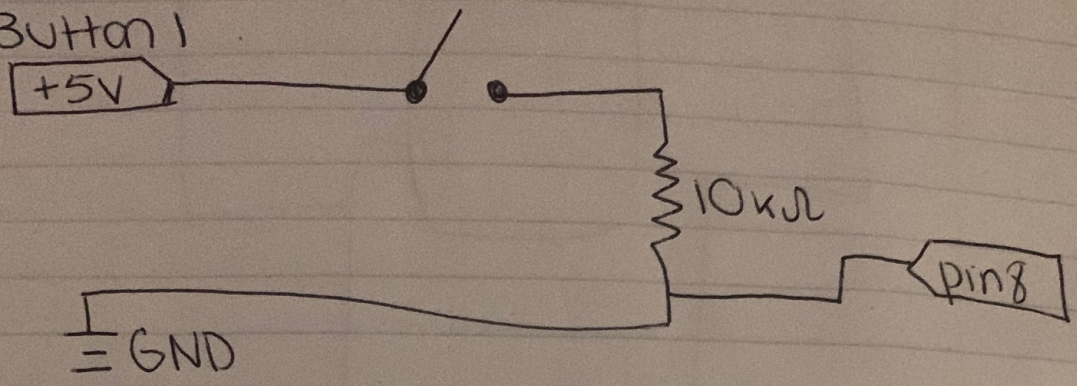
Lab # 5 : The Arduino Music Maker

Introduction:
The purpose of this lab was to build an Arduino controlled music system. The synthesizer is a four note electronic synthesizer (one button for each note) with distortion control (using the potentiometer), and a percussion instrument motor using the servo-motor. In this lab we used an Arduino board, a USB power cable, a breadboard for building external circuts, circut components like jumpers, 4 buttons, resistors, a potentiometer, and a servo-motor. I also used other tools and components like a small screw driver for attaching arm to the servo-motor armature, tape, and cardboard for making a motor mount. Our tasks for this lab were to lay out the components on the board and adding the 4 buttons, the potentiometer, and the speaker. We then added the appropriate resistance for each of the circut elements. Unlike previous labs, in this lab we used the potentiometer by itself, not in series with a fixed resistance. In this lab the total value of our potentiometer is fixed at 10kΩ (between the two legs on the same side), and the leg on the other side represents the intermediate point between two resistors in series. Then, we connected the breadboard to the Arduino to monitor the status of the buttons. After, we developed code to create 3 distinct tones and allow these tones to be distorted by the potentiometer.

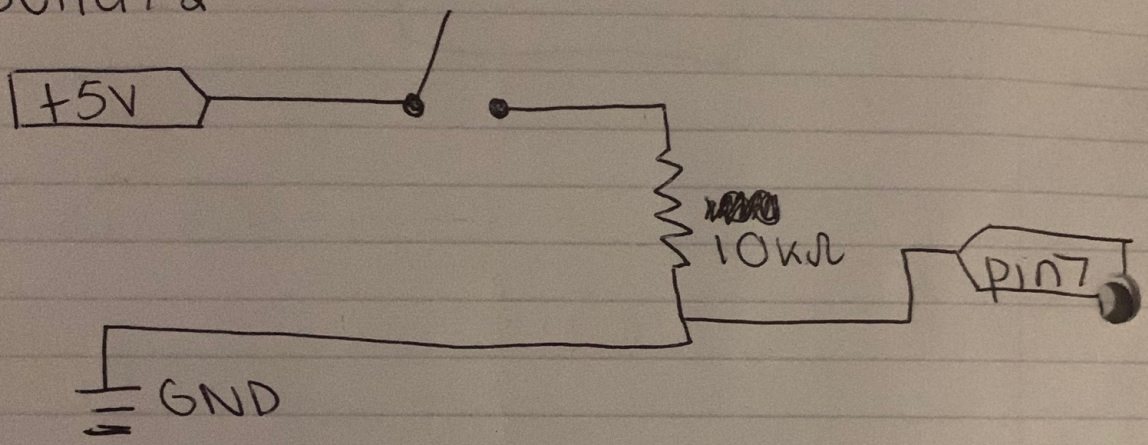
TO do this, we calculated the timing for the tones by using the microsecond delay function and added our tones by putting const int t1 = 400 (example) representing tone 1 is 400. We then connected the servo-motor to the board & red = +5V, black = ground, white = signal from a digital pin, and finished developing code that allows the servo-motor to be controlled as part of the music machine. TO develop this code we used the latch command & #include <Servo.h>. We also have to remember that the servo-motor is a special motor that has a limit to its motion and limits are defined as 0° and 180°. Lastly, we built a structure to mount our motor and made some kind of instrument.

Setup and procedure

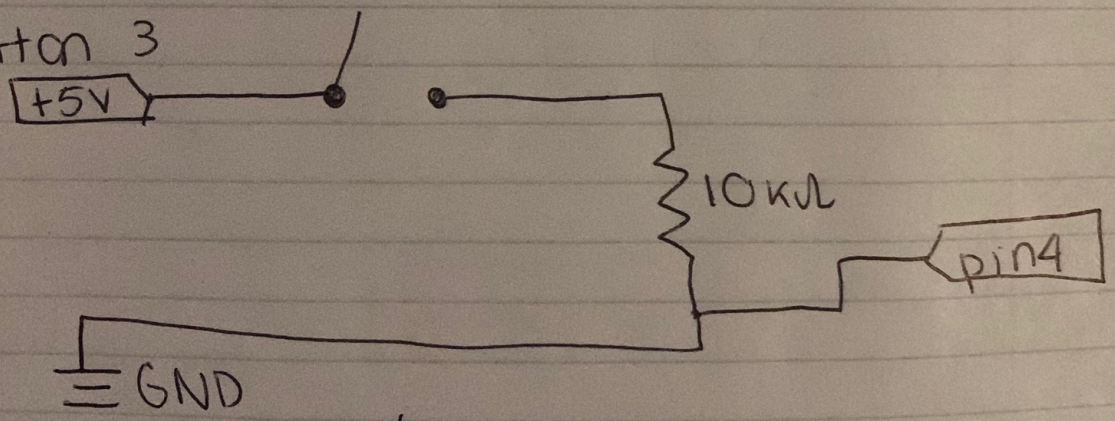
1) Button 1



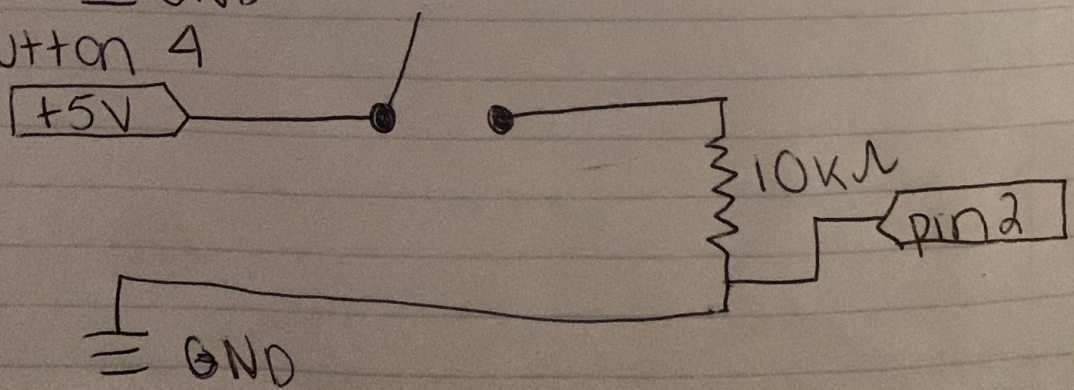
Button 2



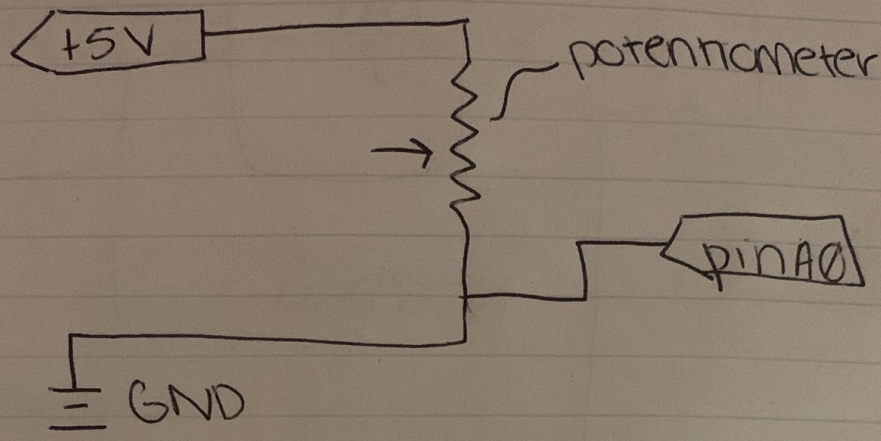
Button 3



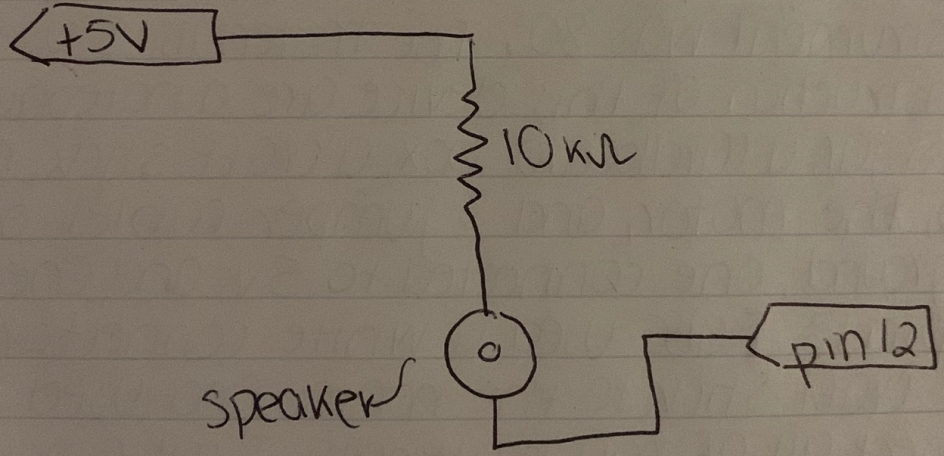
Button 4



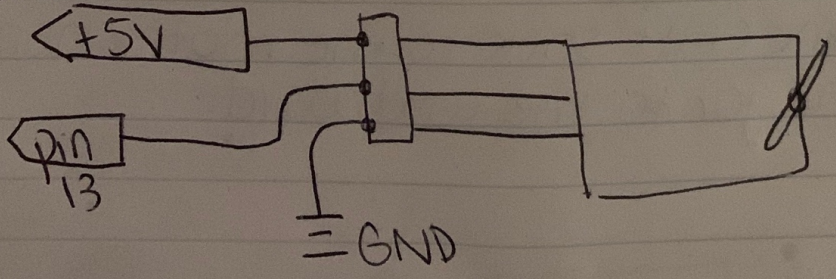
potentiometer



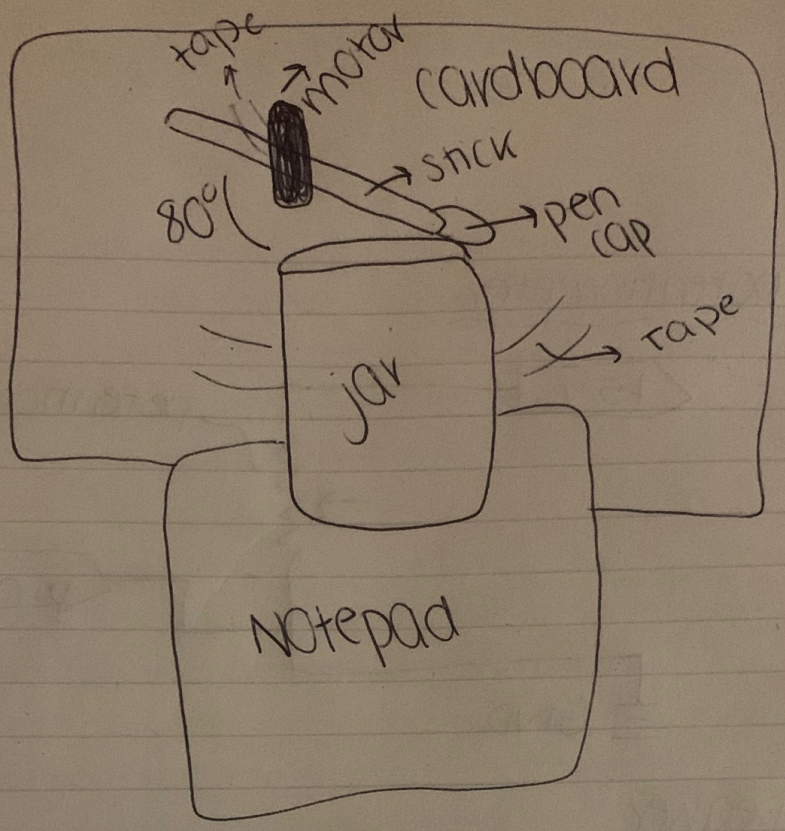
speaker



motor



2)



The range of motion needed to activate this instrument was 80°. The materials I used for the construction of this device are a notepad, a glass jar, a cardboard box, a craft stick, a pen cap, the motor, and 3 jumper cables one connected to ground, one connected to 5v, and one connected to pin 13. I also used a white connector to connect the craft stick to the motor and used tape to connect all of my materials. I cut a hole into the cardboard to ~~fit~~ fit the motor in. I used tape to connect the craft stick to the motor handle. I also taped the jar to the cardboard.

Data and analysis:

1) $\boxed{\Delta V = IR \quad I = 1}$

$\Delta V = IR$

$+5\,V = IR = 0$

When the button is open (not pressed), the resistance is very high because the circuit is not connected/complete. When the button is closed (pressed), there is no resistance. Therefore the button should move from 1 to 0.

2) $\Delta V = 5V$

$R = 10{,}000\Omega$

$I = \frac{5V}{10{,}000\Omega} = 0.0005A$

$V = I \cdot R_{tot}$

$V_{out} = IR_2 = 0.0005A(10{,}000\Omega) = 5V$

$R_2 = 10{,}000\Omega$

$R_1 = 0\Omega$

$V_{in} = V_1 + V_{out} = 5V + 5V = \boxed{10V}$

The further the knob of the potentiometer is turned, the higher or lesser resistance is put on R_2 which lowers or raises the resistance on R_1. Ohms law is used to calculate the voltage put on the potentiometer.

3) $t_1 = (400ms)(2)\left(\frac{1ms}{1000ms}\right) = 0.8$

$\frac{1}{0.8ms}(1000) = \boxed{1250Hz}$ Note: B

dist = (analogRead(potentiometer) - 512/4)

delayMicroseconds $\left(\begin{matrix} t1 + dist \\ t2 + dist \\ t3 + dist \end{matrix}\right)$

$t2 = (500\,ms)(2)\left(\frac{1\,ms}{1000}\right) = 1\,ms$

$\frac{1}{1\,ms}(1000) =$ **1000 Hz** Note: D

$t3 = (700\,ms)(2)\left(\frac{1\,ms}{1000}\right) = 1.4\,ms$

$\frac{1}{1.4}(1000) =$ **714.29 Hz** note: G

$t4 = (100\,ms)(2)\left(\frac{1\,ms}{1000}\right) = 0.2\,ms$

$\frac{1}{0.2}(1000) =$ [redacted] **5000 Hz**

4) Range: 0 to 1000 → +500 - 500

± 100

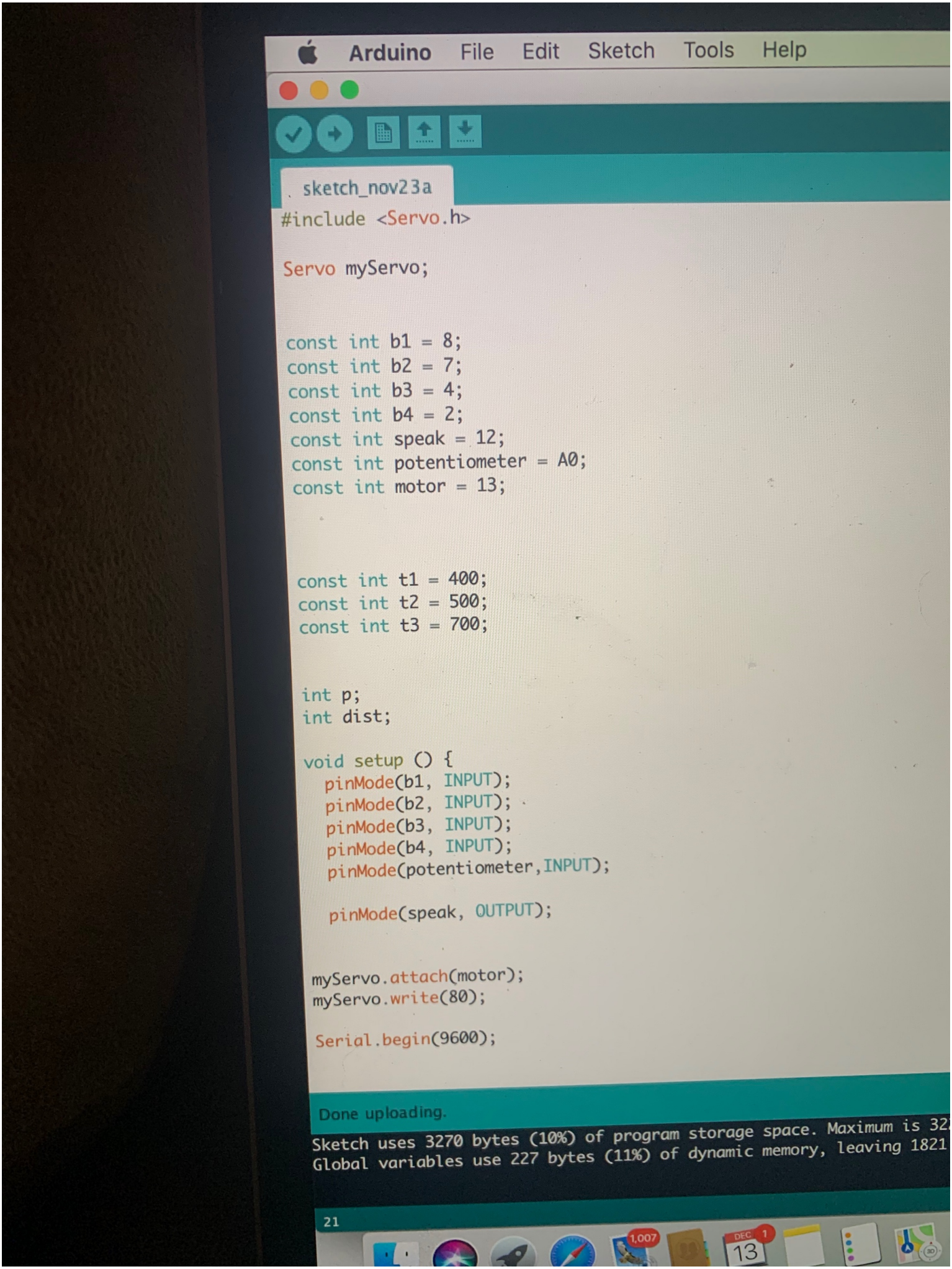
$\frac{100}{500} = \frac{1}{5}$

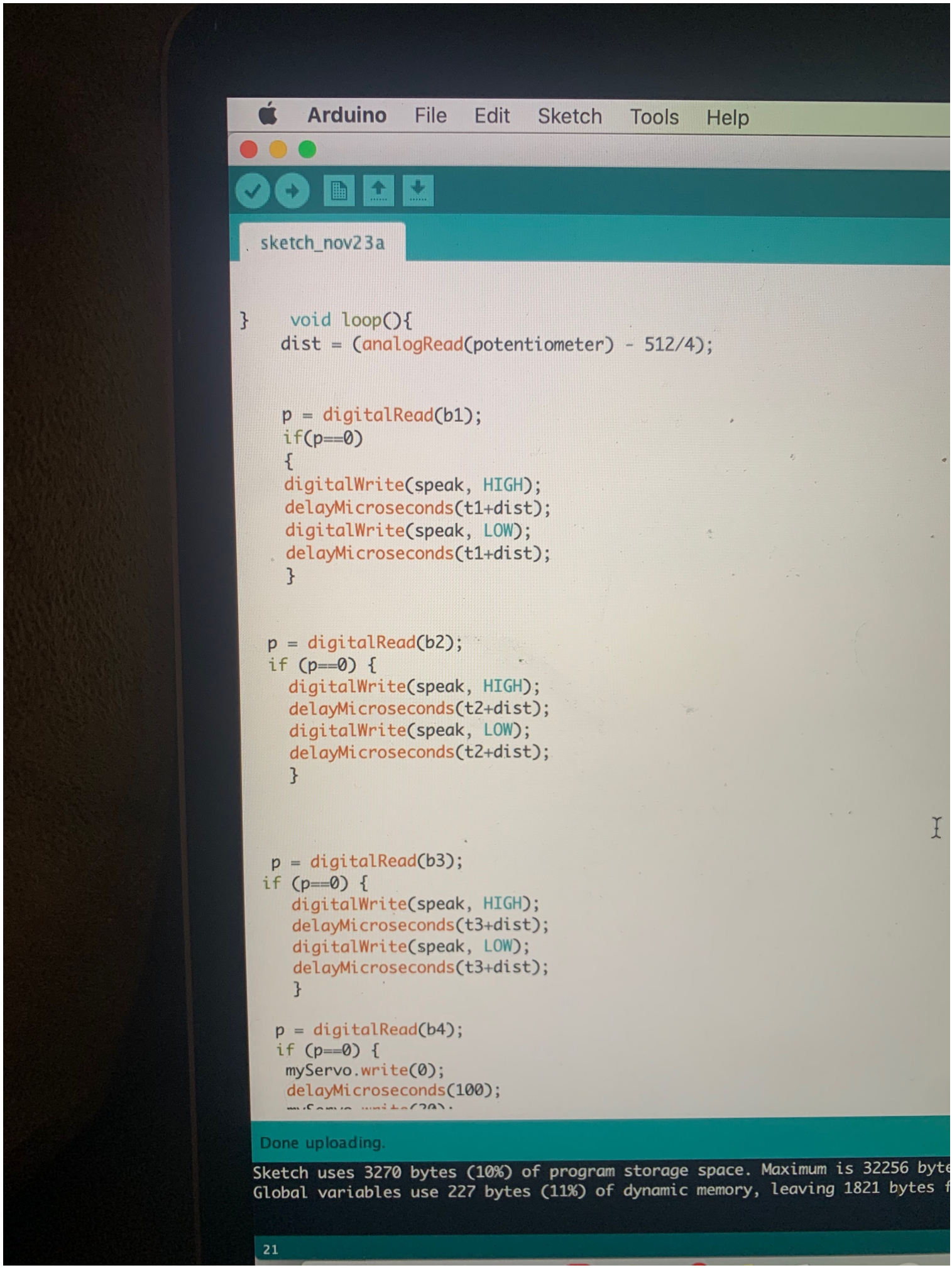
X = ((analogRead(pin) - 500) • 25)/64

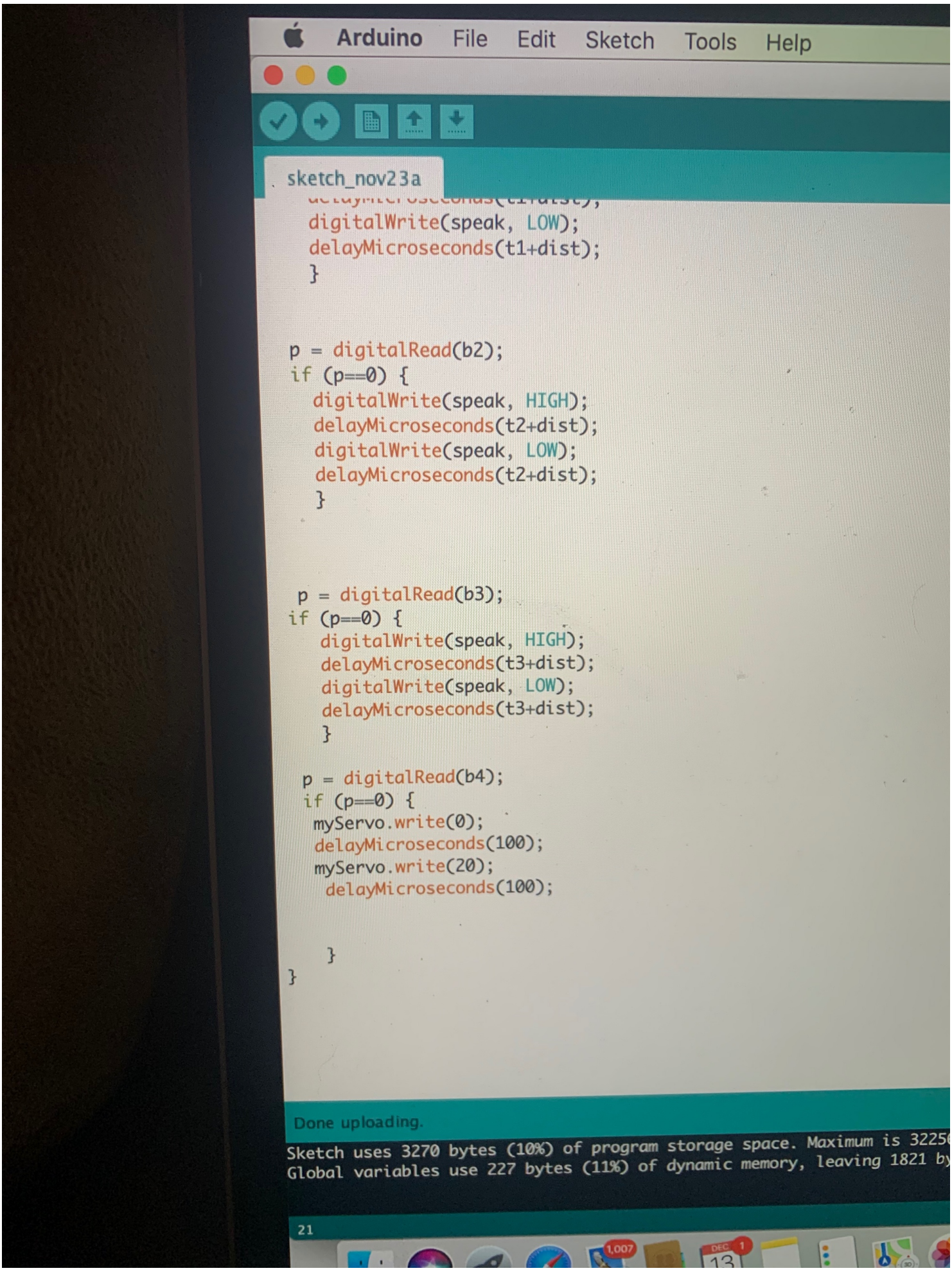
Conclusions

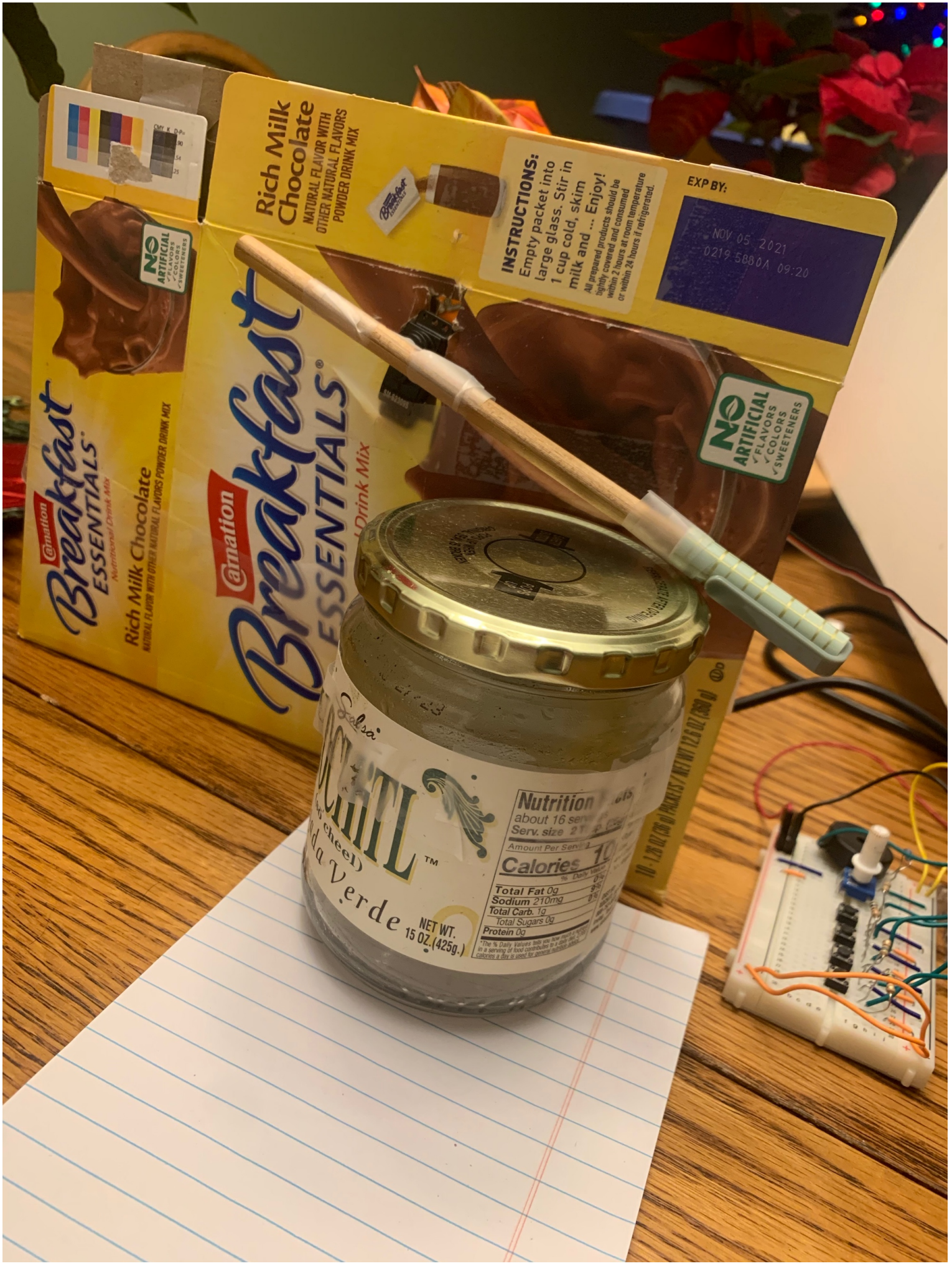
The notes that we get out of our synthesizer when two buttons are pressed is not a smooth combination of tones that we would get if we were playing a real life musical instrument or in a professional synthesizer. This is because we are using very simple code and the computer functions kind of want to interfere with one another and wants either one or the other. In our lab, when button one was pressed, the speaker turns on for some amount of time, about half a period, and then turns off for about a half a period. Then when the second button is pressed, it turns the speaker on for a short amount of time and then off for a short amount of time and then it goes through a loop and starts again and repeats. Since button one turns on for a short amount of time and off for a short amount of time, and button two turns on for a short amount of time and off for a short amount of time, they are staged sequentially. In order to get a smooth combination of tones when two buttons are pressed, when thinking about sin waves, we would want to add the two sin waves together to play simultaneously. Since we are only getting an approximation of adding the two sin waves of the buttons together, we do not get a clean sound and can't play simultaneously, only sequentially, so it doesn't work so well and we don't get a

smooth combination of tones. If I were to expand this project and work towards making a more complicated or higher quality music maker, I would try to get the instrument/ motor to move while a tone is being sound. I would also add more buttons to increase my tone range, attempt to create smoother combinations when more than one button is pressed, and I would create an instrument that incorporates more sounds and beats to create a better rythym.


Arduino File Edit Sketch Tools Help
sketch_nov23a
#include <Servo.h>
Servo myServo;
const int b1 = 8;
const int b2 = 7;
const int b3 = 4;
const int b4 = 2;
const int speak = 12;
const int potentiometer = A0;
const int motor = 13;
const int t1 = 400;
const int t2 = 500;
const int t3 = 700;
int p;
int dist;
void setup () {
pinMode(b1, INPUT);
pinMode(b2, INPUT);
pinMode(b3, INPUT);
pinMode(b4, INPUT);
pinMode(potentiometer,INPUT);
pinMode(speak, OUTPUT);
myServo.attach(motor);
myServo.write(80);
Serial.begin(9600);
Done uploading.
Sketch uses 3270 bytes (10%) of program storage space. Maximum is 32
Global variables use 227 bytes (11%) of dynamic memory, leaving 1821
21


Arduino File Edit Sketch Tools Help
sketch_nov23a
} void loop(){
dist = (analogRead(potentiometer) - 512/4);
p = digitalRead(b1);
if(p==0)
{
digitalWrite(speak, HIGH);
delayMicroseconds(t1+dist);
digitalWrite(speak, LOW);
delayMicroseconds(t1+dist);
}
p = digitalRead(b2);
if (p==0) {
digitalWrite(speak, HIGH);
delayMicroseconds(t2+dist);
digitalWrite(speak, LOW);
delayMicroseconds(t2+dist);
}
p = digitalRead(b3);
if (p==0) {
digitalWrite(speak, HIGH);
delayMicroseconds(t3+dist);
digitalWrite(speak, LOW);
delayMicroseconds(t3+dist);
}
p = digitalRead(b4);
if (p==0) {
myServo.write(0);
delayMicroseconds(100);
Done uploading.
Sketch uses 3270 bytes (10%) of program storage space. Maximum is 32256 byte
Global variables use 227 bytes (11%) of dynamic memory, leaving 1821 bytes f
21


Arduino File Edit Sketch Tools Help
sketch_nov23a
digitalWrite(speak, LOW);
delayMicroseconds(t1+dist);
}
p = digitalRead(b2);
if (p==0) {
digitalWrite(speak, HIGH);
delayMicroseconds(t2+dist);
digitalWrite(speak, LOW);
delayMicroseconds(t2+dist);
}
p = digitalRead(b3);
if (p==0) {
digitalWrite(speak, HIGH);
delayMicroseconds(t3+dist);
digitalWrite(speak, LOW);
delayMicroseconds(t3+dist);
}
p = digitalRead(b4);
if (p==0) {
myServo.write(0);
delayMicroseconds(100);
myServo.write(20);
delayMicroseconds(100);
}
}
Done uploading.
Sketch uses 3270 bytes (10%) of program storage space. Maximum is 3225
Global variables use 227 bytes (11%) of dynamic memory, leaving 1821 by
21


Carnation
Breakfast
ESSENTIALS
Rich Milk Chocolate
NATURAL FLAVOR WITH OTHER NATURAL FLAVORS
POWDER DRINK MIX
INSTRUCTIONS:
Empty packet into large glass. Stir in 1 cup cold, skim milk and ... Enjoy!
EXP BY:
NOV 05 2021
NO ARTIFICIAL
FLAVORS
COLORS
SWEETENERS
Salsa
Verde
NET WT. 15 OZ.(425g.)
Nutrition
Calories
Total Fat 0g
Sodium 210mg
Total Carb. 1g
Total Sugars 0g
Protein 0g