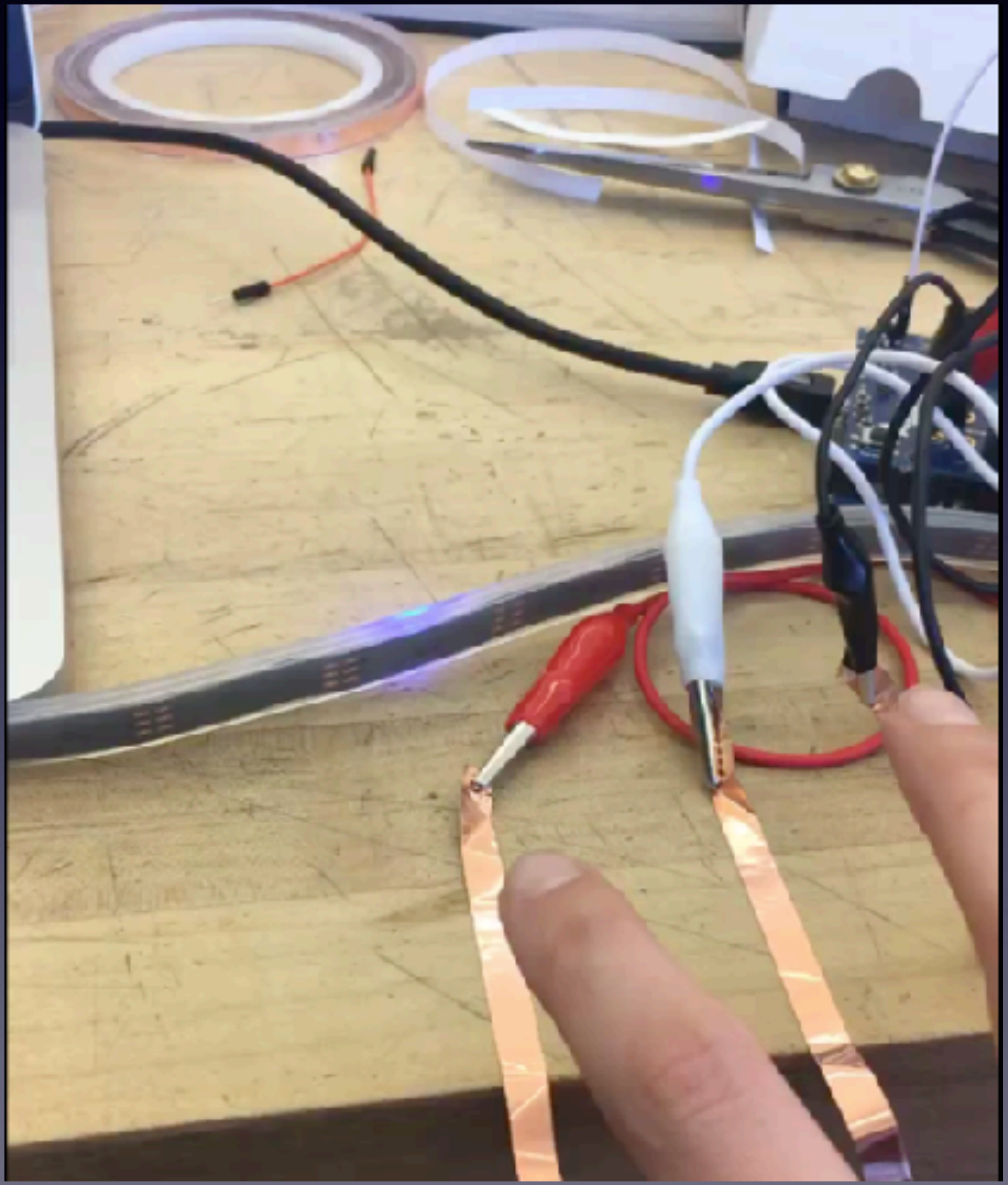
Capacitive touch keyboard

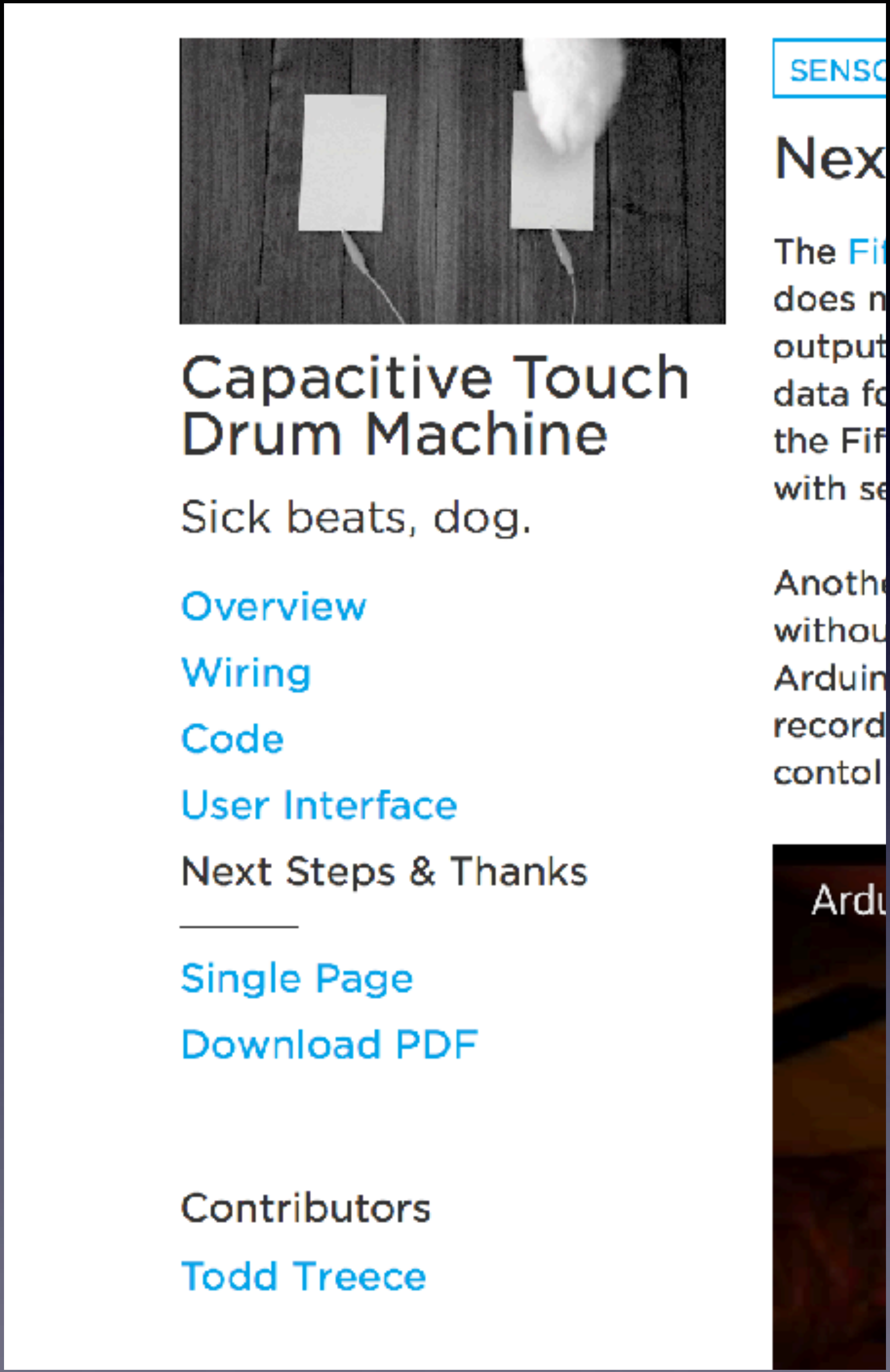
Final Project
Programming and Electronics
Michelle Macis
Spring 2017

Look What I Made!

- I transformed a capacitive touch synth drum kit and tweaked it to make a keyboard I could play a song to- while the a strip of neopixel lights would react to each note with a different color



On the ada fruit site, I found this:



Capacitive Touch Drum Machine

Sick beats, dog.

Overview

Wiring

Code

User Interface

Next Steps & Thanks

Single Page

Download PDF

Contributors

Todd Treece

SENSC

Nex

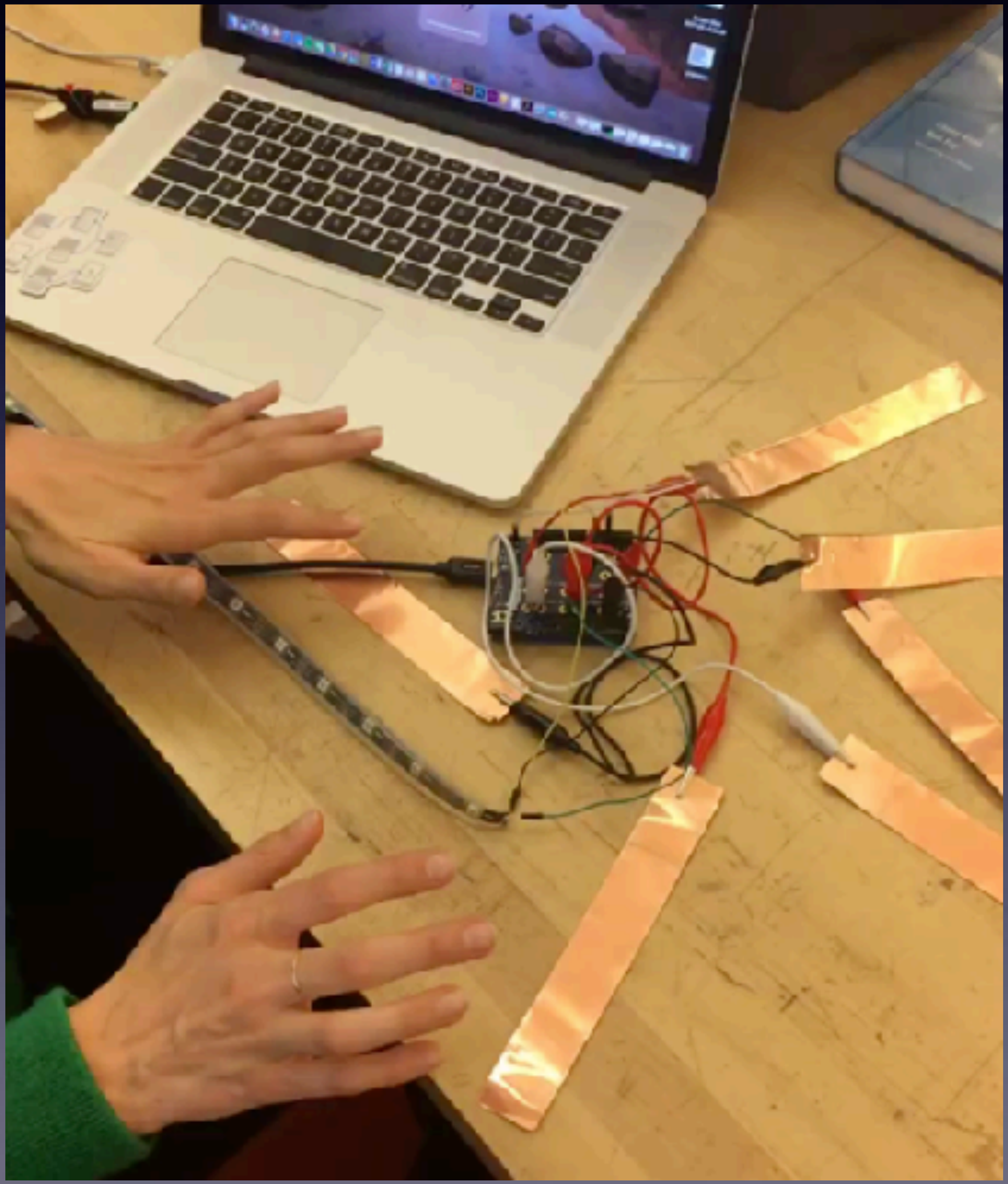
The Fi
does n
output
data fo
the Fif
with se

Anoth
withou
Arduin
record
contol

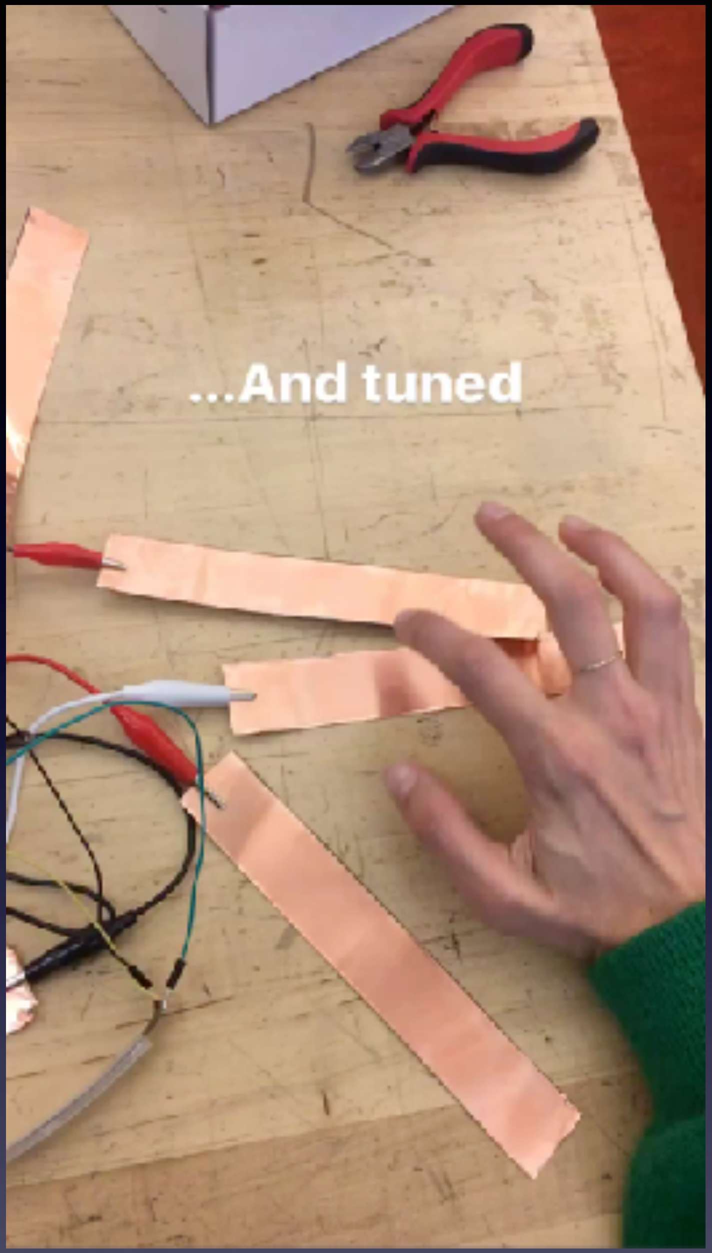
Ardu

I learned so much!

- I learned that finding the right code makes all the difference
- I learned that making your own alligator clips is a pain (true story)
- I learned if you don't record it, it didn't happen
- I learned how to tune by adjusting the pitch number and find middle C
- I learned how to make a keyboard I could play a song to!!



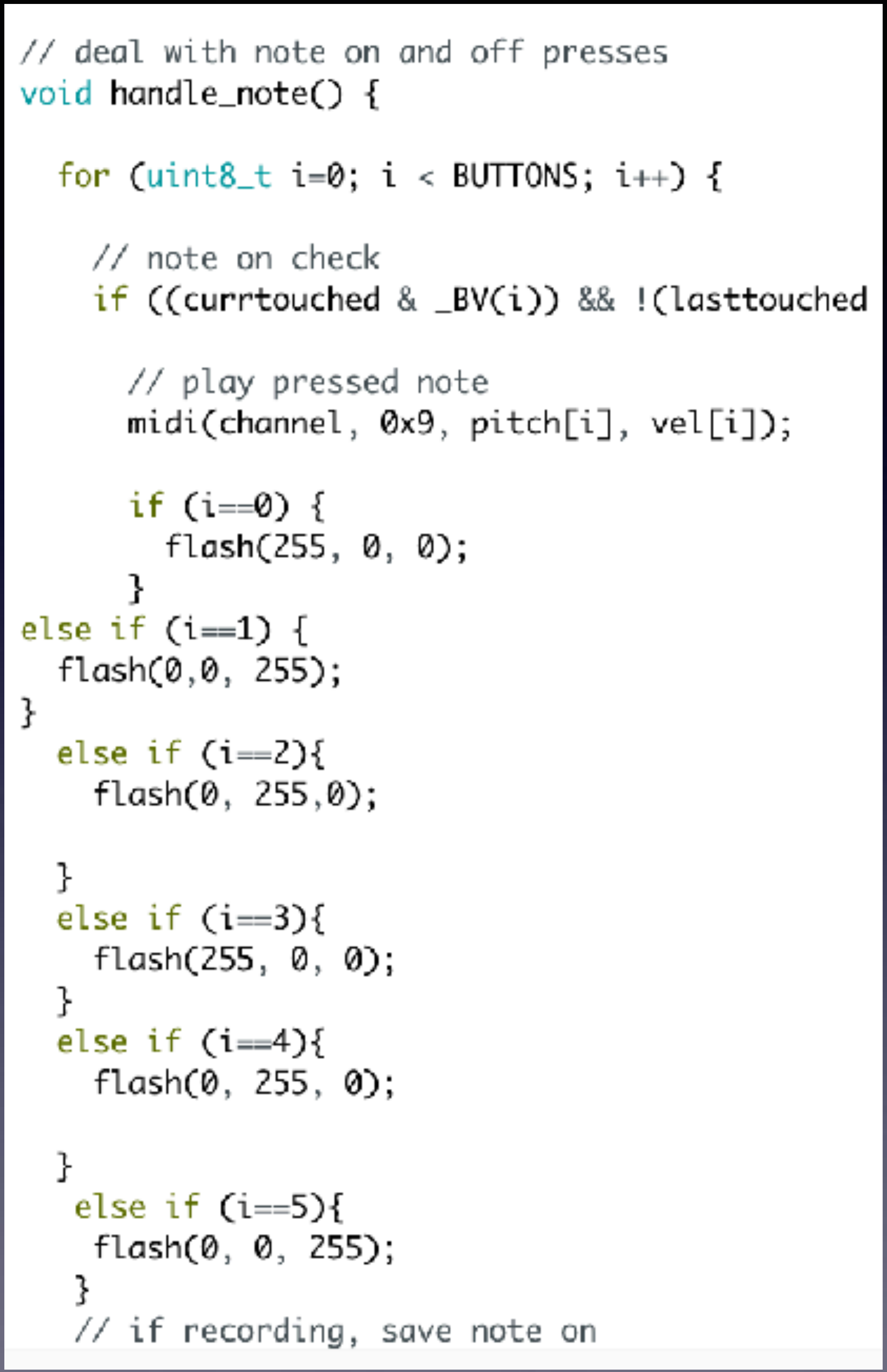
- Next time, I would use different code
- I wouldn't have lights the next go around, I would stick to just the music
- Next time I would really explore the synthesizing capacity- that was a little over my head



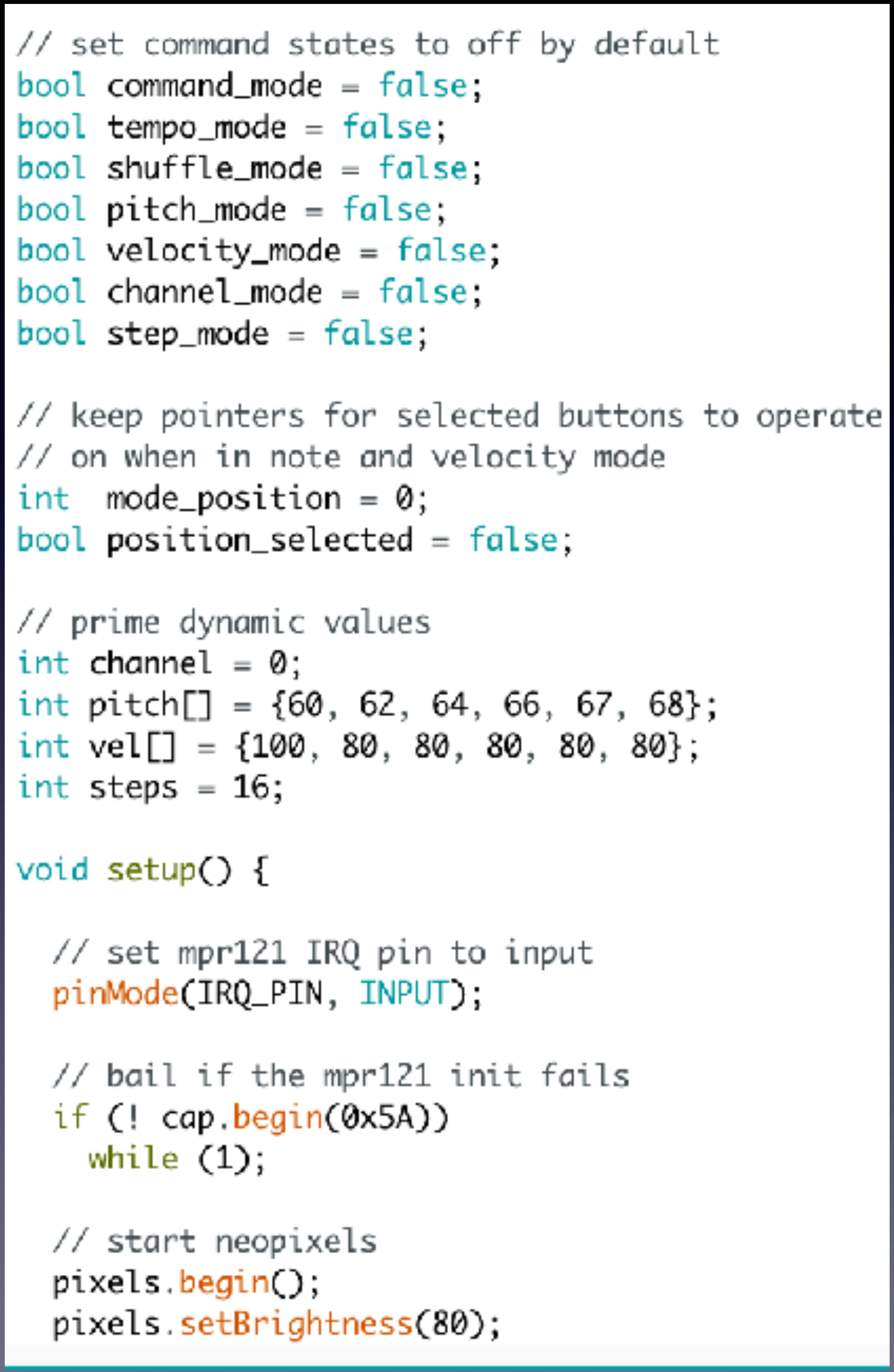
My Demo!

Yeah!!!

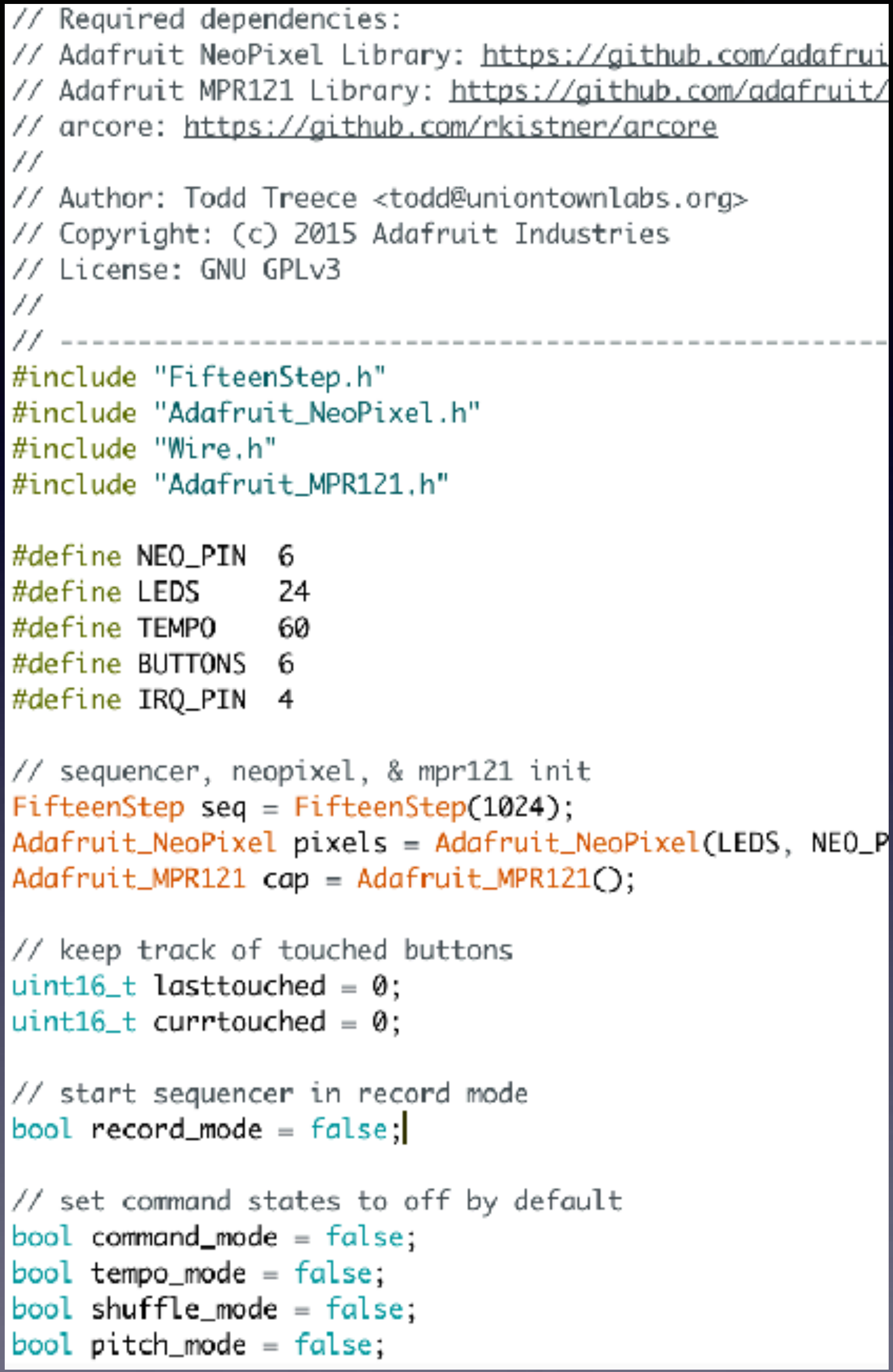
This is code for making the light strip light up a different color every time a new key was pressed:


```
// deal with note on and off presses
void handle_note() {

  for (uint8_t i=0; i < BUTTONS; i++) {

    // note on check
    if ((currtouched & _BV(i)) && !(lasttouched

      // play pressed note
      midi(channel, 0x9, pitch[i], vel[i]);

      if (i==0) {
        flash(255, 0, 0);
      }
else if (i==1) {
  flash(0,0, 255);
}
  else if (i==2){
    flash(0, 255,0);

  }
  else if (i==3){
    flash(255, 0, 0);
  }
  else if (i==4){
    flash(0, 255, 0);

  }
   else if (i==5){
    flash(0, 0, 255);
   }
   // if recording, save note on
```

This is code for adjusting the pitch:

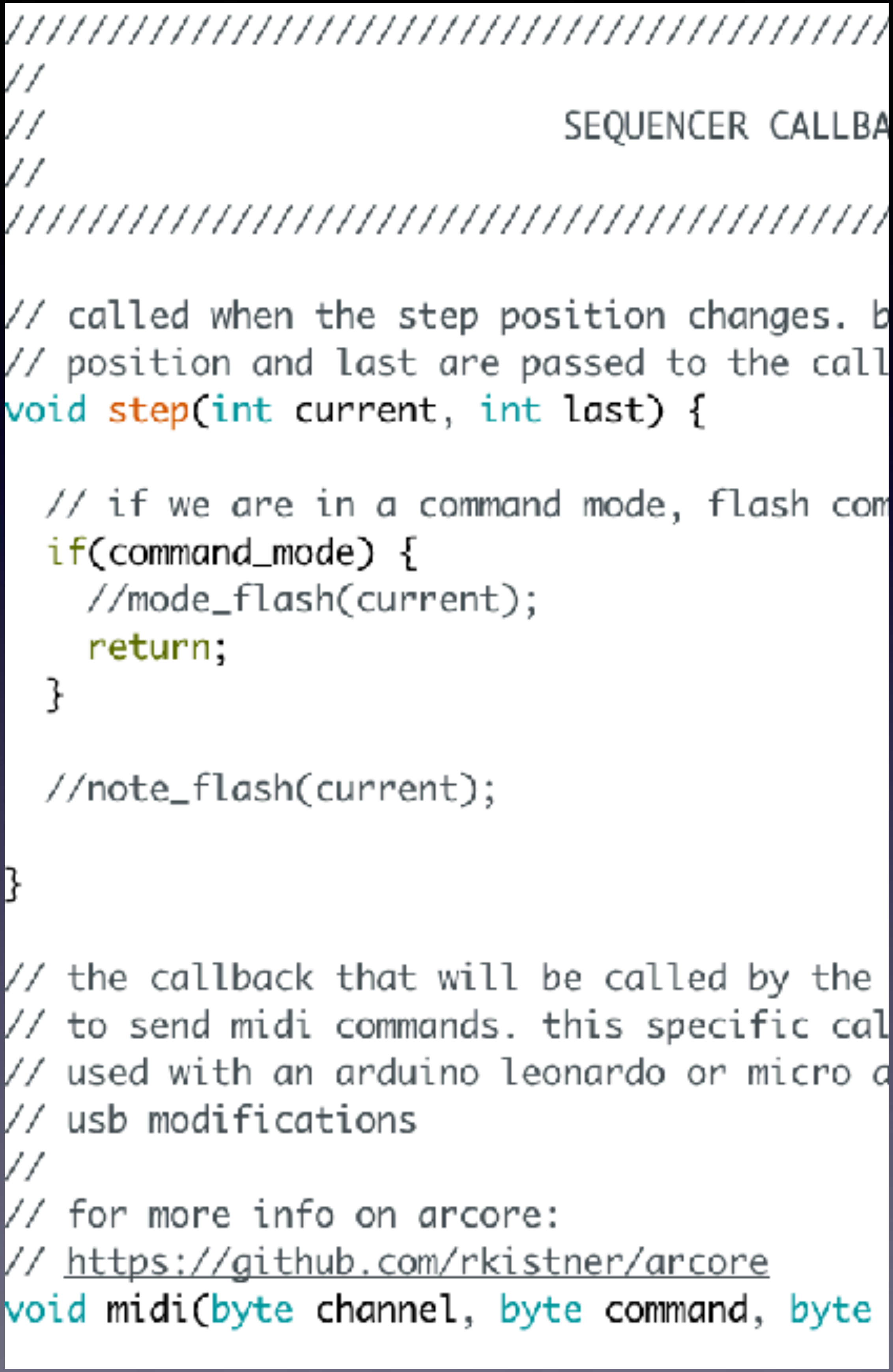

```
// set command states to off by default
bool command_mode = false;
bool tempo_mode = false;
bool shuffle_mode = false;
bool pitch_mode = false;
bool velocity_mode = false;
bool channel_mode = false;
bool step_mode = false;

// keep pointers for selected buttons to operate
// on when in note and velocity mode
int  mode_position = 0;
bool position_selected = false;

// prime dynamic values
int channel = 0;
int pitch[] = {60, 62, 64, 66, 67, 68};
int vel[] = {100, 80, 80, 80, 80, 80};
int steps = 16;

void setup() {

  // set mpr121 IRQ pin to input
  pinMode(IRQ_PIN, INPUT);

  // bail if the mpr121 init fails
  if (! cap.begin(0x5A))
    while (1);

  // start neopixels
  pixels.begin();
  pixels.setBrightness(80);
```

This the code for turning off the synth, so I could play a song:



```
// Required dependencies:
// Adafruit NeoPixel Library: https://github.com/adafrui
// Adafruit MPR121 Library: https://github.com/adafruit/
// arcore: https://github.com/rkistner/arcore
//
// Author: Todd Treece <todd@uniontownlabs.org>
// Copyright: (c) 2015 Adafruit Industries
// License: GNU GPLv3
//
// -----------------------------------------------
#include "FifteenStep.h"
#include "Adafruit_NeoPixel.h"
#include "Wire.h"
#include "Adafruit_MPR121.h"

#define NEO_PIN   6
#define LEDS      24
#define TEMPO     60
#define BUTTONS   6
#define IRQ_PIN   4

// sequencer, neopixel, & mpr121 init
FifteenStep seq = FifteenStep(1024);
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(LEDS, NEO_P
Adafruit_MPR121 cap = Adafruit_MPR121();

// keep track of touched buttons
uint16_t lasttouched = 0;
uint16_t currtouched = 0;

// start sequencer in record mode
bool record_mode = false;

// set command states to off by default
bool command_mode = false;
bool tempo_mode = false;
bool shuffle_mode = false;
bool pitch_mode = false;
```

This is the code for turning off the light while I wasn't playing the keyboard:



```
///////////////////////////////////////////
//
//                           SEQUENCER CALLBA
//
///////////////////////////////////////////

// called when the step position changes. b
// position and last are passed to the call
void step(int current, int last) {

  // if we are in a command mode, flash com
  if(command_mode) {
    //mode_flash(current);
    return;
  }

  //note_flash(current);

}

// the callback that will be called by the
// to send midi commands. this specific cal
// used with an arduino leonardo or micro a
// usb modifications
//
// for more info on arcore:
// https://github.com/rkistner/arcore
void midi(byte channel, byte command, byte
```

That was fun!

Thank you!!