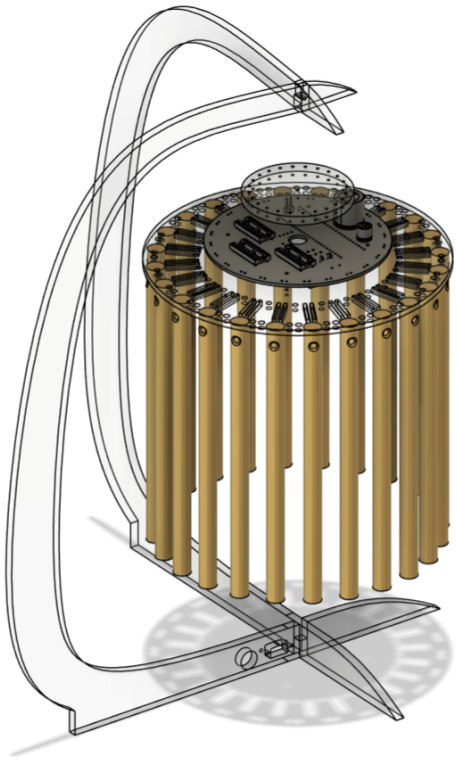

New York University

Electronic Product Design: Music and Audio,
Final Report

eChimes

Kyle Six

supervised by

Prof. Steven Litt

December 20th, 2024

Contents

Section 1: Concept

Introducing **eChimes**, a novel, digitally-enhanced mark tree that connects acoustic chime sounds with modern computer control. By decoding MIDI input, eChimes uses an array of solenoids to translate digital notes into precise, percussive strikes- creating music across the individual hanging chimes. With easy integration into digital audio workstations and electronic keyboards, it allows musicians, composers and sound designers to explore chime textures in fast, rhythmic polyphony. This blend of acoustic and electronic elements enables a new layer of expression across a variety of musical communities that otherwise would have been impossible.

Figure 1: Mark Tree (left) and eChimes 5.0 prototype (right)

Instrument

A **mark tree**, often called "wind chimes" or "rack chimes" in a musical setting, is a percussion instrument made up of a series of thin metal rods of ascending lengths, suspended in a row. When brushed or struck, these bars create a cascading, shimmering sound that adds a unique, ethereal quality to music. Typically, mark trees are played by sweeping a hand or stick across the bars, producing a gentle, unpredictable wash of tones that decay naturally, making them popular in jazz, film scoring, and other atmospheric music genres.

Electronic Control

eChimes builds on this traditional instrument design but adds electronic control. Instead of needing to physically sweep across the chimes, eChimes uses MIDI signals to activate solenoids attached to each rod. Each solenoid functions like a small mallet, striking an individual chime in response to a specific MIDI note. This allows precise control over which chime sounds and when, something that's nearly impossible with traditional mark trees. By using MIDI, eChimes can sequence complex patterns, play specific notes in harmony with other instruments, sync with a digital audio workstation, and much more.

Section 2: Demo

Video

Electronic Product Design Midterm Update | eChimes
(https://youtu.be/HzyVbaWNGPg)

Section 3: Documentation

This section provides a comprehensive guide to the various considerations in prototyping eChimes. Here, you'll find everything from an overview of its components and wiring diagrams to code snippets and CAD models.

Mechanism

To hit each individual chime, a push-pull micro solenoid is mounted on the underside of piece of the laser cut acrylic stand. These mounts are a pair of slots pointing towards the hanging chime such that the distance between the solenoid and the chime may be finely adjusted by screws on top. When given power, the solenoid's striker extends to make contact with the upper edge of the chime and retracts when power is cut.

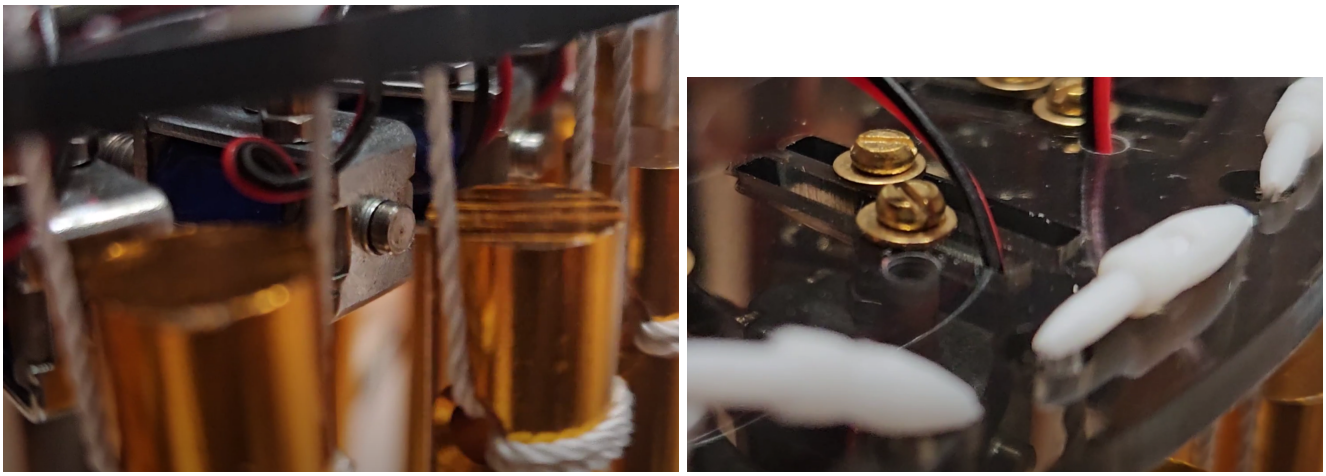

Figure 2: Solenoid and chime mechanism (left) and Adjustable solenoid mount (right)

Electronics

ULN2803APG In order to control the 12V needed by the solenoid, a Darlington transistor array is used to amplify current significantly, allowing small input signals to drive larger loads. This is an integrated circuit consisting of pairs of transistors, enabling each solenoid to strike its corresponding chime with sufficient driving voltage while responding rapidly to the microcontroller's 3.3V logic signals.

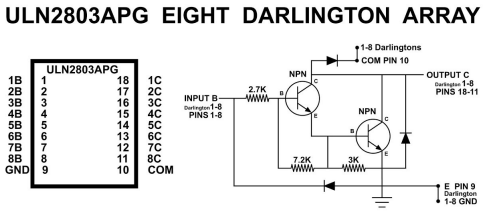

Figure 3: Diagram of ULN2803APG integrated circuit

Teensy 4.1 With high processing speed and extensive I/O capabilities, this microcontroller is ideal for handling complex tasks like MIDI processing. In eChimes, it decodes MIDI input commands and maps each note to the corresponding solenoid, providing precise, real-time control over which chime is struck in response to digital signals.

12V Micro Push-Pull Solenoid An electromagnetic actuator that moves a small rod in and out when powered, ideal for controlled striking actions. In this version, 24 solenoids are mounted to a specific chime, and connected to a specifc darlington transistor output.
Amazon Link

Design

Coming from the long, linear design of the Mark Tree, I set out to create a more compact footprint that could comfortably live on a desktop. For this version, the chime mechanism was modeled in Fusion360 and duplicated 24 times. These were then equally placed around a minimum-sized circle, and the stand to support these followed suit. Note that many design iterations have taken place, as detailed in this video update.

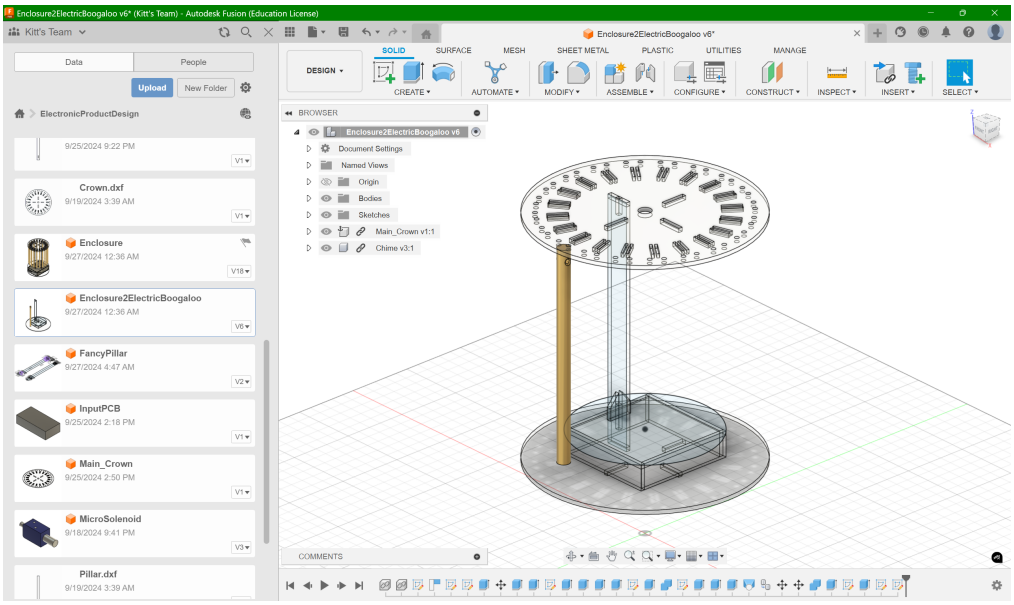

Figure 4: CAD of early design

Code

Written in C++, with Visual Studio Code and the PlatformIO plugin, the main code files for this prototype are below.

settings.h

```
#ifndef SETTINGS
#define SETTINGS
#include <Darlington.h>

namespace SETTINGS
{
    // Software Configuration
    static const int MIDI_CABLE = 0;
    static const int MIDI_CHANNEL = 1;

    static bool USE_CLOSEST_OCTAVE = true; // Adjusts out of bounds MIDI notes to the the closest playable octave

    // Hardware Configuration
    static const int NUM_SOLENOIDS = 24;         // The count of connected darlington transistors -> solenoids
    static const int HIGHEST_MIDI_NOTE = 128 + 1;   // The highest MIDI note supported by the note/pitch decoding
    /*********************************************
     *** Solenoids
    *** - Immutable array declaring the darlington
    *** transistor object on the specified pin.
    *********************************************/
    // Darlinton Transistor Array, assigning pins to each [0-23]
    static Darlington solenoids[NUM_SOLENOIDS] =
    {// Darlington(Pin#)···
    /*********************************************
    *** NOTE MAPPING, MIDI -> Solenoid
    *** - Immutable array using the MIDI note number
    *** as an index to a pointer to the correct solenoid
    *********************************************/
    static Darlington* NOTE_MAPPING[HIGHEST_MIDI_NOTE] = {···

    // static const float NOTE_PITCHES[HIGHEST_MIDI_NOTE] = {···
}

#endif
```

Figure 5: settings.h

main.pp

The complete code for this version can be found here.

eChimesMidtermCode.zip

```ino
// Outside Libraries
#include <Arduino.h>
#include <USB-MIDI.h>
#include <StevesAwesomeButton.h>

// Project Libraries
#include <Darlington.h>
#include <Settings.h>
#include <MyMIDI.h>

int PITCH_SHIFT = 12;

// Creates GLOBAL instance named "MIDI" over usb transport protocol, on specified cable
USBMIDI_CREATE_INSTANCE(MIDI_CABLE, MIDI);

// Handles Note ON MIDI command
void myNoteOn(byte channel, byte note, byte velocity) {
    Serial.printf("Channel: %d, Note: %d \n", channel, note);
    if (channel != SETTINGS::MIDI_CHANNEL)
        return;
    note += PITCH_SHIFT;

    Darlington* solenoid = SETTINGS::NOTE_MAPPING[note];
    if (solenoid == nullptr && SETTINGS::USE_CLOSEST_OCTAVE == true) {
        for(int i = 12; i + note < SETTINGS::HIGHEST_MIDI_NOTE; i += 12) {
            if (SETTINGS::NOTE_MAPPING[note + i] != nullptr){
                solenoid = SETTINGS::NOTE_MAPPING[note + i];
                note = note + i;
                break;
            }
        }
    }

    if (solenoid == nullptr)
        return;
    else {
        // play note
        Serial.printf("Playing note: %d, Pin: %d\n", note, solenoid->getPin());
        solenoid->set(DarlingtonState::ON, true);
        delay(10);
        solenoid->set(DarlingtonState::OFF, true);
    }
}

void setup()
{
    int starting_note = 92;
    for (int i = 0; i < 24; i++){
        SETTINGS::NOTE_MAPPING[starting_note + i] = &SETTINGS::solenoids[i];
    }
    SETTINGS::USE_CLOSEST_OCTAVE = false;
    Serial.begin(9600);
    // Listen for incoming midi commands
    //MY_MIDI::attachMIDIHandlers(MIDI);
    MIDI.setHandleNoteOn(myNoteOn);
    MIDI.begin(MIDI_CHANNEL);
}

void loop()
{
    MIDI.read();
}
```

Fabrication

As a graduate assistant for the Tandon MakerSpace, I have much experience in laser cutting, 3D printing and other rapid prototyping workflows. Because of this, I was able to use this advantage to rapidly iterate on my acrylic instrument design as well as breadboard my circuit.

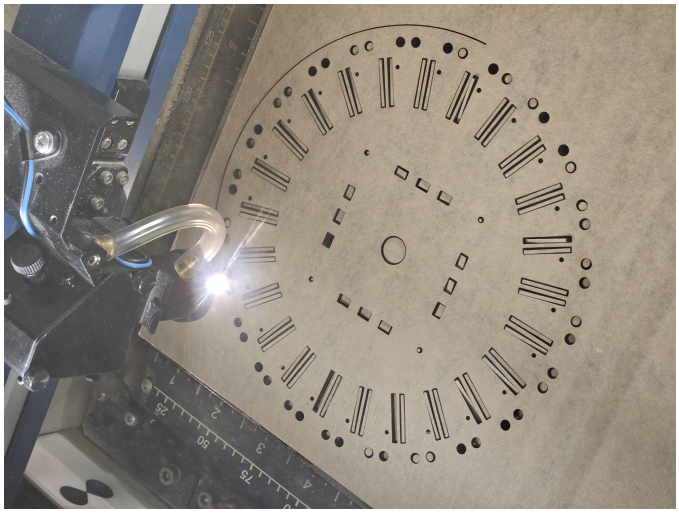

Figure 7: Laser cutting with acrylic

Schematic and PCB

To keep things neat, and also order materials in advance, I began creating the schematic and PCB design in KiCad. Featuring a DC-DC power regulator, usb type C conversion, and the solenoid driver circuitry, this design has been fully breadboarded and is ready to order.

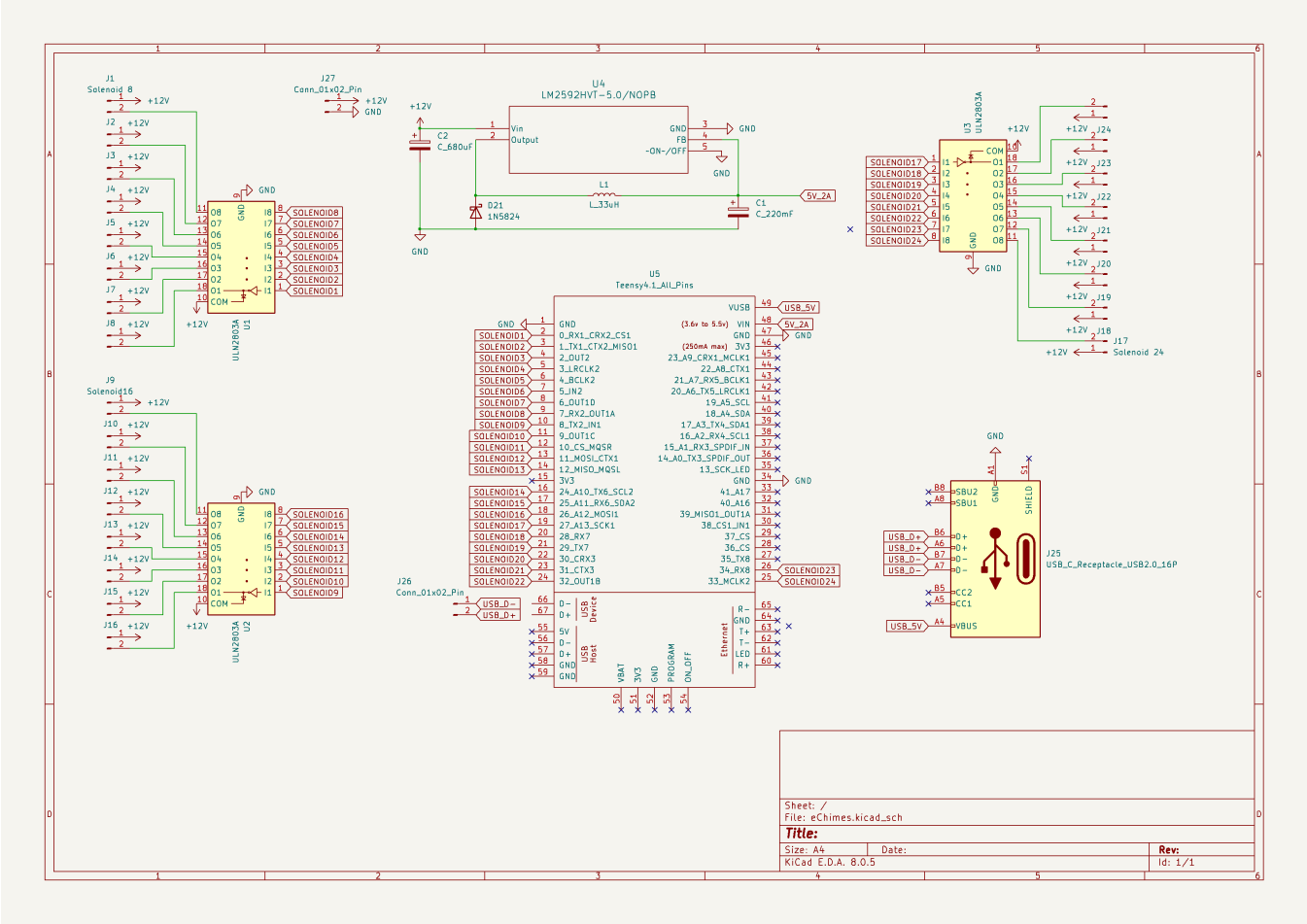

Figure 8: eChimes schematic with power converter and USB C connector

Section 4: Roadmap

For the second half of the semeseter, the remaining tasks concerning productizing eChimes should be completed while the existing components are refined. Here is a schedule detailing each week's projected. progress, followed by a breakdown of each task.

Week	Goals
7	Version 1.0, Midterm Presentation
8	Update Instrument Design
9	Complete PCB Layout Finalize Design
10	Order or Mill PCBs Fabricate Enclosure
11	Assemble Enclosure Solder Circuit Boards
12	Integrate Components Refine Codebase
13	Finished Product

Design

With the original instrument design, I was dissatisfied by the simplistic cynlindrical profile and expecially the gaps left by smaller chimes. To push myself to create a more beautiful, almost sculptural design, I wanted to create a spiralized ascent of chimes from the base to reflect the descending lengths of chimes as their pitch increases.

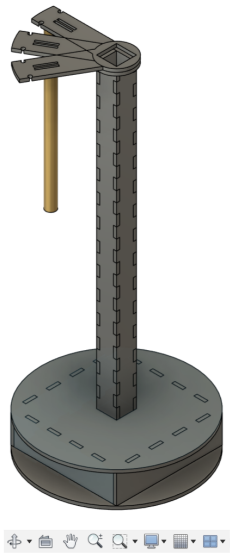

Figure 9: Fourth design iteration featuring chime mounts that form a spiral when stacked

PCB

Following the completed schematic, and the receipt of my electrical components, I assigned footprints and began the PCB layout process. Notably, I imported the drawings from the CAD directly so that I can align mounting holes and other important geometry. Currently, only connections need to be run before this PCB is ready to be ordered. As an interesting tangent, I also would like to try fabricating the board myself with a CNC mill at the MakerSpace.

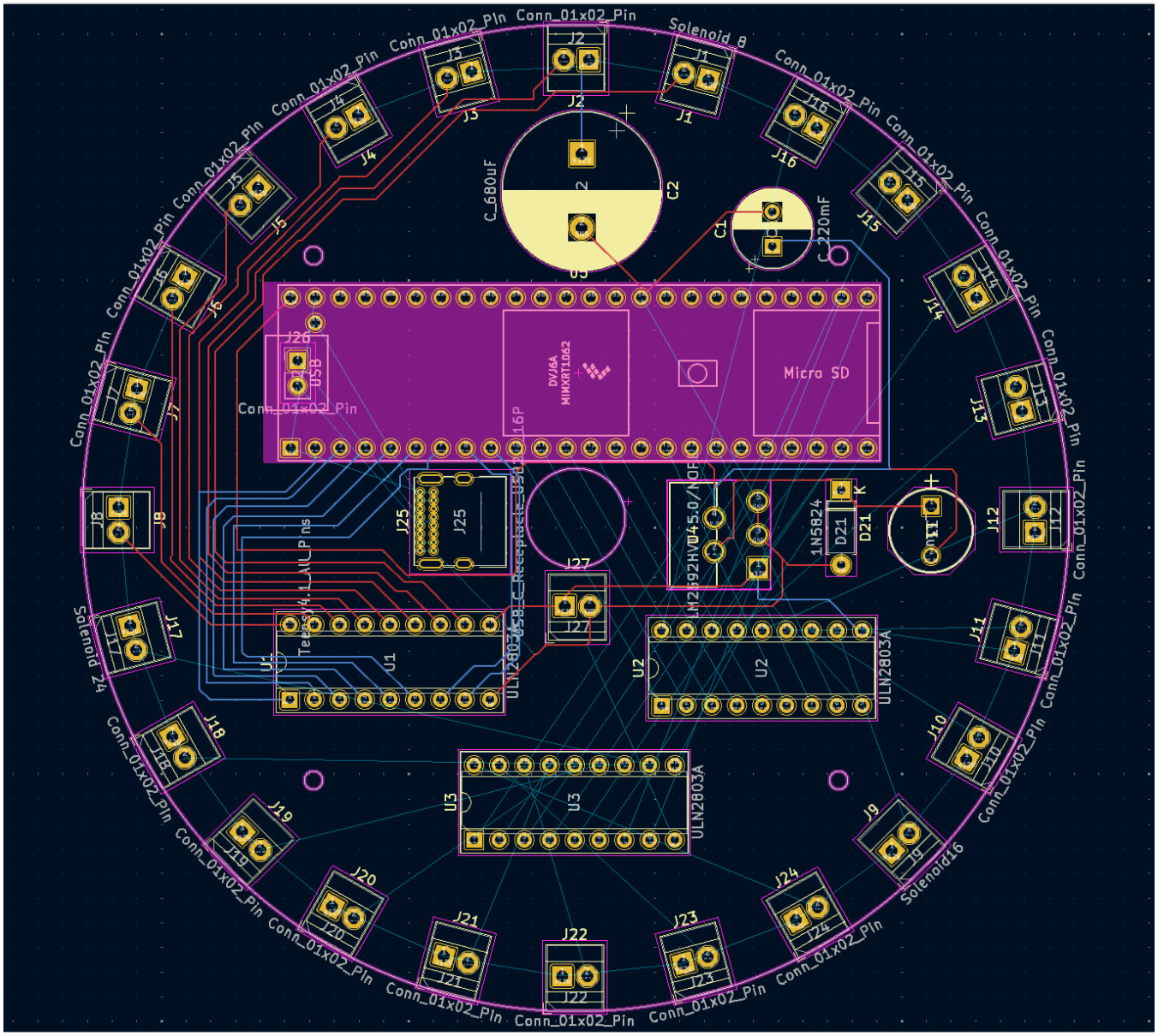

Figure 10: Current state of PCB layout, wiring phase

Bill of Materials

BOM