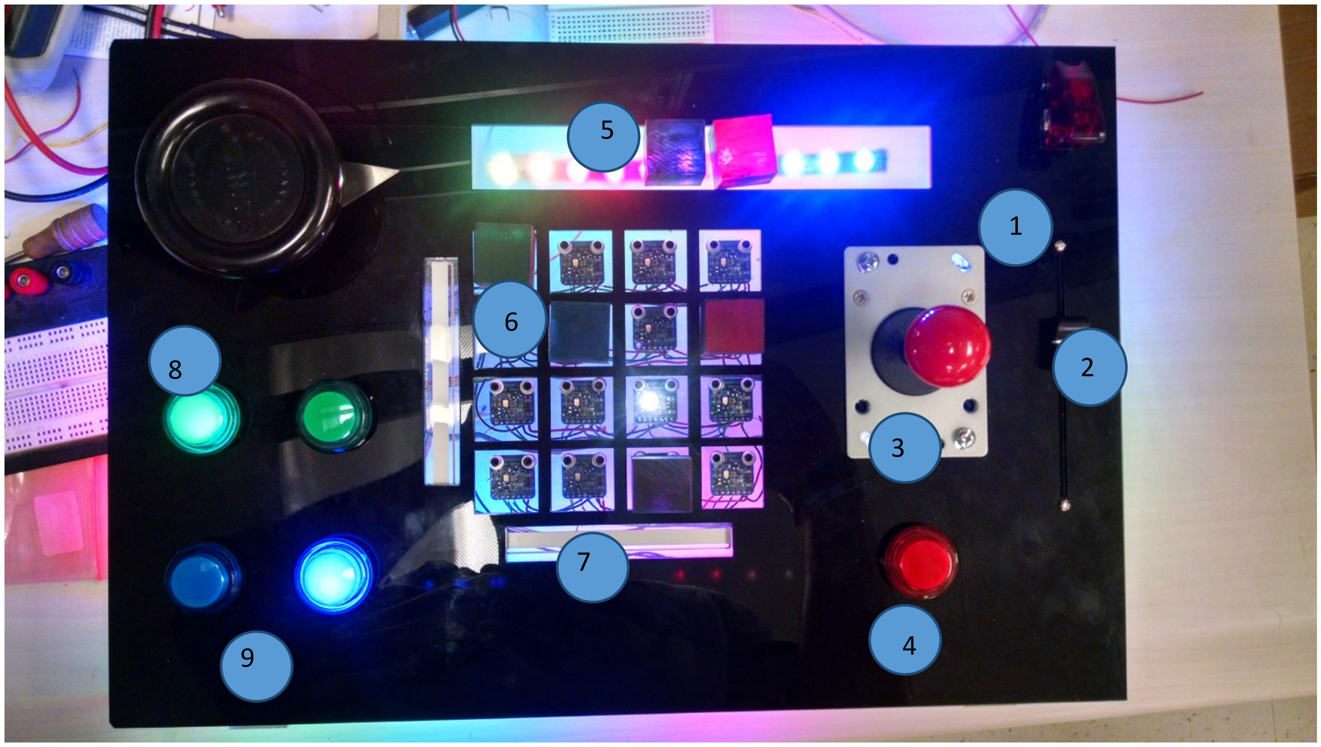
Kurt Oleson
EMID Project 1
7 May 2015

The Light Stick/Cubes of Whatever a.k.a. Cube-a Gooding Jr.

Introduction

This is the second and final version of our EMID project; a few names have been added to the title. The purpose of this project was to follow up and refine our earlier design. We also wanted to create added functionality to expand the musical repertoire of the instrument. We were able to reuse many of the same parts and a lot of the same of software. A diagram of the final instrument is shown below.

The instrument is still more or less a tactile, visual sequencer controlled by a joystick and cubes and various buttons and parameter controls. The power switch (1) is what initializes the instrument and turns the sounds on or off. The tempo slider (2) controls the overall tempo of the instrument including



bass note tempo, melodic tempo, and the speed of the lights on the grid. The joystick (3) controls the bass note of the music being produced based on the direction that it moves. For the final design, we utilized all 8 directions of the joystick instead of only 4 like the last project. This way the user can play the I, II, III, IV, V, VI, and VI note of a preset key signature, as well as a rest. Once a direction is chosen the bass note (or chord) will repeat at a given tempo, set by the slider, playing whatever note is linked to that direction. Aside from just controlling the notes, the joystick also controls the direction of the lights that move on the grid.

The red button (4) activates record mode. This mode will allow the user to record some pattern of joystick directions and then have the pattern loop without being affected by joystick movements. This allows the user to record some bassline or chord progression and have it loop while synced to a tempo and also control the lights without changing the bass. The grid (6) is a 4x4 set of slots for cubes. In each slot is a color sensor with an LED light. Using the joystick, one can “move” a light around the grid by cycling through different color sensors when some direction is chosen. So this means that every time the light changes direction, the bass note (or chord) of the sequence changes.

The cubes (5) rest near the top of the instrument as shown and can be placed in any of the slots. The cubes have different colored sides – red, green, blue or purple – that trigger different events. When the light reaches a cube, the color sensor detects the color and based on that value it does one of 4 things. The melodic events that occur differ in arpeggio directions as opposed to the musical content. This way the cubes can be used to control the contour of the piece which is better for sequencing. We have 4 different colors where each color triggers a different direction of an arpeggio; there are ascending, descending, ascending step-wise, and descending step-wise patterns.

The softpots on the axes of the grid (7) control a filter frequency parameter for the chords/bass notes and a modulation wheel that alters different parameters of the chords/bass notes. The green buttons (8) control the toggle between chords or walking bass notes that are played by the joystick. With the bass notes selected (left button) the joystick will control a walking bass line, and the chords (right button) activated control for sustained chords. With the current design, the joystick only controls either the chords or the bass at any one time so after switching to a different control, the last note or chord will sustain. The blue buttons (9) control the toggle between major and minor modalities. The modality change only effected the chords and bass notes. So with these features we were able to create a modular sequencer with visual feedback and that can be manipulated with physical objects.

Max Patch and Arduino Code

The Max patch created for this instrument was fairly simple but cluttered. One part of the patch was the joystick control. In the Arduino code we set the joystick controller to send a different value (0 through 8) for each direction of the joystick so we had less data parsing to do in Max. Once this direction command was received in Max it was converted into a note. The loadbang() of the Max patch initialized it so that the tonic was playing and from there the 8 directions toggled between notes. The tonic is manually set and the other notes are created based on adding specific intervals to the predetermined tonic. The bass notes are synced to a metro() object that is controlled by a slider in Max. The joystick, when controlling the chords, worked in much the same way. However, the dirrection would trigger a triad, not a single note, and instead of repeating notes for every beat of the metro() object, the chords were sustained. These chords would sustain until a new chord value came into the system. This was done by only triggering a note off message when a change in joystick direction was detected.

As for the color sensors, they were connected using a multiplexer so in the Arduino code we had to initialize each of the 16 sensors and poll to see which one was currently active. The joystick commands that change the position of the lights are also in the Arduino code and tell the multiplexer to switch inputs to the next closest color sensor in the direction that the joystick was pointed. The sensors were switched between using an array in the code. In Max, we received RGB values from the currently activated color sensor. We then parsed this RGB value to recognize the 4 possible colors with some simple logic. When a color was detected, an arpeggio was played in the direction corresponding to the color of the cube. There was also some sudo-randomness built into the code so that the melodies would not always be exactly the same. There were 2 possible sequences of notes – a triad with 7th added and a chord with a 5th, 6th, and 9th – and they could be traversed in four different ways based on the colors. They could also be delayed by a beat so there was a chance they played on beat or in syncopation with the bass.

Another big section of the Max code was record mode. When the record button triggered the signal that the user was going to record a sequence, the joystick positions that followed were dynamically recorded into a list using a multislider object until the record button was triggered again. After this second trigger was hit, the list of directions was reversed and then looped and outputted to the bass/chord control.

Reason

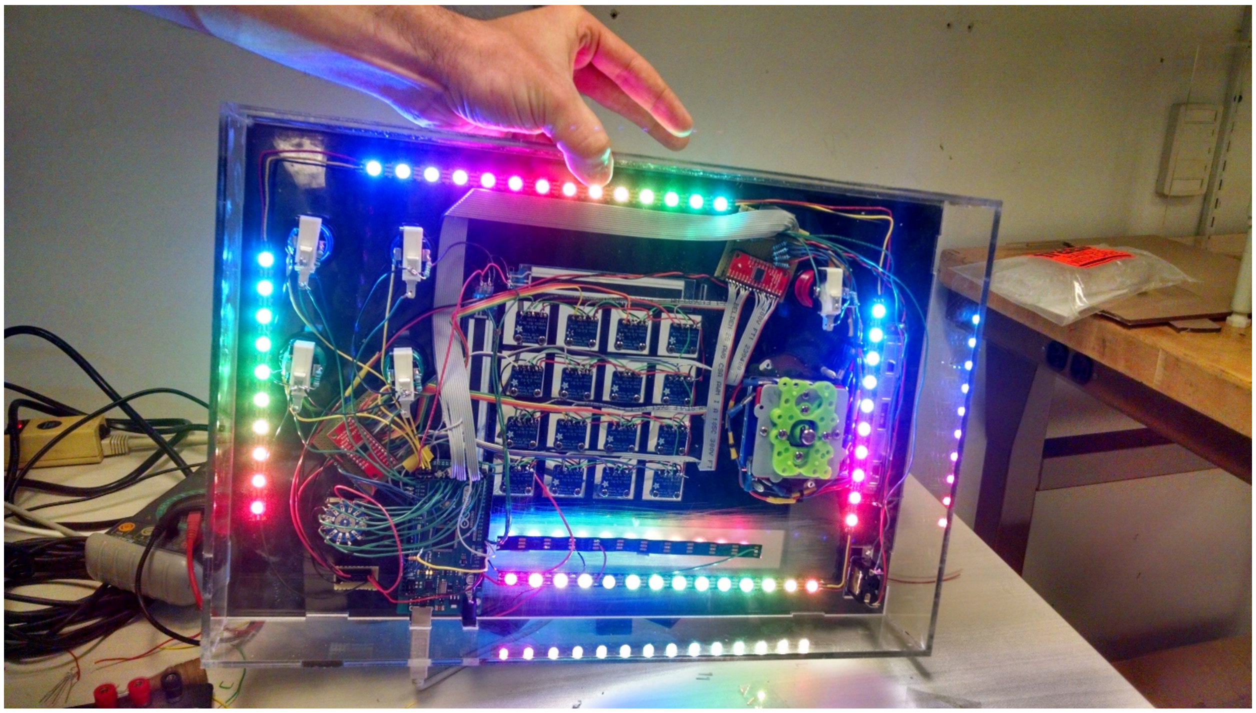
The patches in reason were made from scratch. There was a Subtractor synth for the bass notes that used a slow LFO to change the filter frequency in order to provide some variation so that not every time the bass note was played it would sound the same. The lead synth was made with a Thor synthesizer. The chords were played from a Maelstrom granular synthesizer. Each patch in reason had a different MIDI channel for input. The filter frequency of the Subtractor and Maelstrom was controlled by one of the softpots and the modulation wheel of the same two synths were controlled by the other softpots.

Construction and Wiring

The physical components of the instrument making up the frame and surface were modeled in SolidWorks based on the dimensions of the components (joystick, cubes, etc.) and then broken up into 2-dimensional objects to be cut. These objects were then transferred to an Adobe Illustrator file and loaded onto the machine in the Makers' Lab in the CEEO where they were laser cut. Then, the color sensors were mounted onto the bottom layer of the top surface to align with the grid.

The joystick was wired using a bread board and the color sensors were done using a soldered circuit board. The joystick had 4 outputs going to the Arduino; diagonal directions were measured by a combination of active outputs. The color sensors were all attached to power, ground and data output wiring. The outputs of the color sensor were attached to a 16-channel digital multiplexer which had the common pins attached to the Arduino so that all color sensors could be read and receive data from fewer pins on the Arduino.

The buttons sent binary information to the max patch based on when they were pressed or released. There was logic in the Arduino code that controlled when the buttons would be lit up and how the toggle worked. The tempo slider was just sending raw data from ~150 to ~1100 which we used as a measurement of milliseconds for the gap between notes in order to control the tempo for the entire instrument. There are a number of LED strips attached to the underside of the instrument for aesthetic effect. These strips were controlled by a separate power source and take serial communication from the Arduino. The Arduino talked to Max via serial communication, sending binary values for each sensor. A picture of the wiring is shown below.



<u>Problems and Future Design Considerations</u>

There were many problems that came up when creating the project but we were very satisfied with the way that we handled and solved these issues. One of the biggest problems we faced came right near the end of our project's completion. When troubleshooting an issue with the color sensors, we sent them 5V, what was listed as an approved voltage, but this voltage shorted out the sensors so we lost our color sensing capabilities the night before our project presentation.

Another issue was having the desired time to dedicate to this project. The time constraint ultimately led us to leave out one of our final features. In the upper-left corner of the instrument diagram there is a large rotary. Our intentions were to include this rotary in this final version of the instrument as a way to select the root key signature so that the instrument would have even more flexibility and modularity. The 12-position switch that we installed in the interface was never completely wired up and so this was a feature that we could not include.

We also faced an issue with the tempo slider sending data to the softpots due to a wiring issue that we did not have time to sort out. So depending on where the tempo slider was positioned, there was a higher or lower amount of noise coming from the sensors attached to the two softpots.

In the future we would like to add a more flexible record mode so that the user could record two different patterns on the chords patch and bassline patch respectively. We would also obviously like to add a more robust color sensing feature to recognize more colors more accurately.

<u>Conclusion</u>

Overall we were very satisfied with the final product as this project has gone through many different versions and design considerations. Although we can't necessarily decide on a name, we are confident that we have designed a useful, novel interface for musical expression. Besides some obvious setbacks we accomplished our goals and learned a lot in the process.