Audio Visualizer

By: Tran Dao & Jason Lee

---Table of Contents---

Introduction

What we wanted to do for this project was to create a audio visualizer that reacts to sounds/music. We also wanted to create something that was useful and could be used past the life -span of this project. Since audio visualizers are legitimate features that are embedded into products sold in a variety of stores, we thought it would be fun to see exactly how easy, or hard it would be to create this type of feature. Choosing this specific project was something we felt was still in our feasible range without limiting us from being able to explore more possibilities of what we could add to the device. Because we were not sure exactly how much time it would take to complete the project, we had a wide range of ambitious ideas that could make the device more complex than a typical audio visualizer. Since we are using two arduinos for the project, one will implement the audio processing and relay the recorded values to the the second arduino that will then display it on the two arduinos.

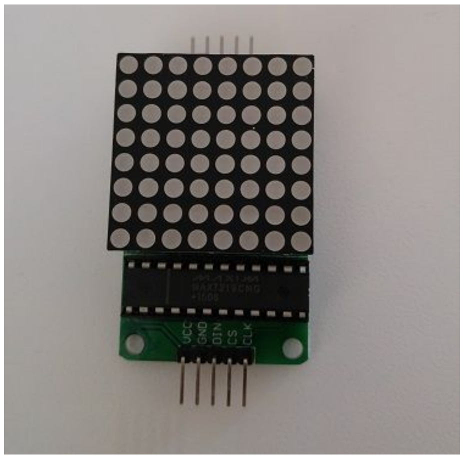
Led Matrix

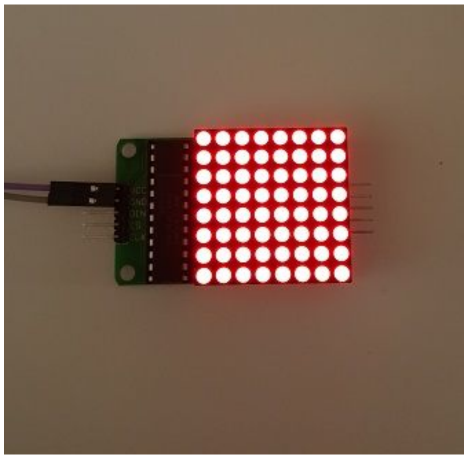
Ideas

For the Led Matrix, we wanted to be able to mimic the functionality of an audio visualizer that you could find at stores where the sound or frequencies of the music being played would modify how the visualizer reacts. We plan to create several patterns that will act as our visualizer when playing music with our sound sensor. Some ideas are the basic jumping bars visualizer, and others that we will create later. We also hope to have a photoresistor o dim or brighten the led display based on the amount of light, add in a motion sensor to allow the arduino to change patterns, and possibly include some type of voice recognition to allow for voice commands to change patterns.

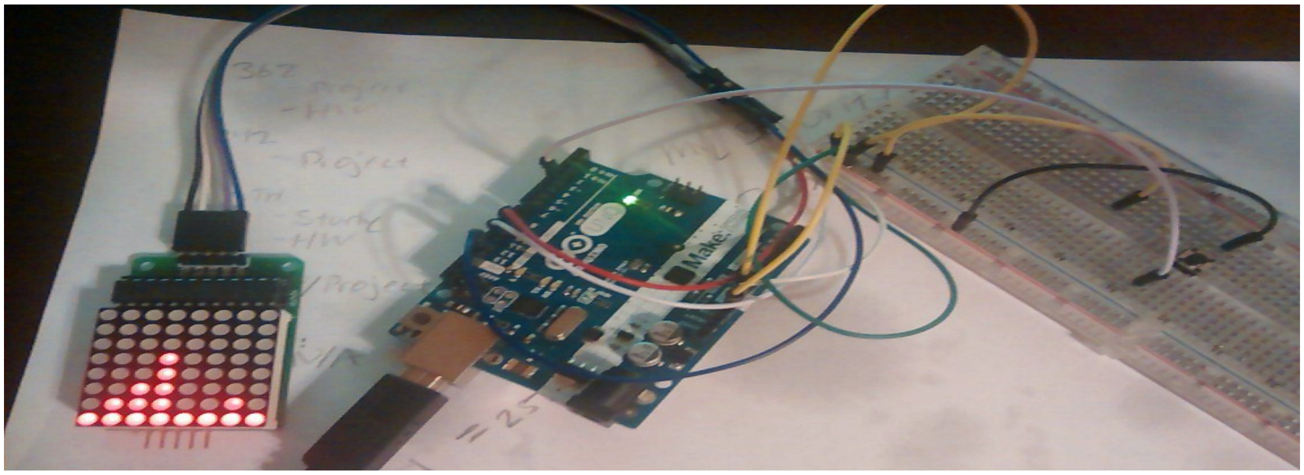
Hardware Progression

The beginning of our project was focused on getting the tools and components that we needed, as well as determining what was necessary to obtain. The most import part of the display is of course the LED matrix which we were lucky enough to have two available without having to purchase them. We were able to work on this part of the project first because of its availability. The LED matrix was quite simple to connect to the arduino itself requiring only a few wires. And since both of us had one of the matrices to work with, we were able to verify that both were working and can be connected together. We used two 8x8 red LED matrices to display the patterns.





The wiring for the LED matrix is quite simple, with 5 pins (VCC, GND, DIN, CS, CLK).
Our wiring was chosen as VCC to 5v/power, GND to ground, DIN to pin 12, CS to pin 11, and CLK to pin 13



The above picture is the led matrix connected to the arduino with a button to switch to different patterns designed below.

Software Progression/Code

Libraries:

At first we were using the library LedControl to control and modify different patterns onto the led matrix. After multiple tests, we manage to get the led matrix to respond to test cases within the code to simulate inputs from the sound sensor since we were still waiting for that part to arrive. Later on we found out that although the library allowed for us to modify different patterns onto the led matrix, it was not quick enough to simulate different inputs from the sound sensor (See obstacles below). We later switched to the Max72xxPanel and Adafruit-GFX library since those libraries were designed for our specific type of led matrix. After using those, the led matrix now reacts fast enough to simulate different inputs from the sound sensor.

<u>**Pattern (Code):**</u>

During the coding section for the pattern, there were several functions used from the Max72xxPanel library that became the benchmark for the entire design. The functions:

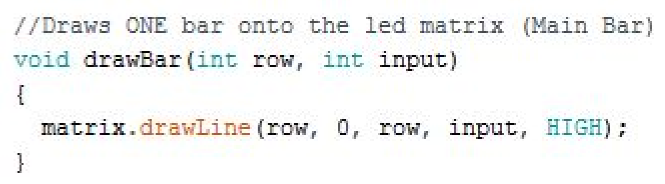
→ **drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t color)**
→ **fillScreen(uint16_t color),**
→ **write()**

were the most widely used function in designing different patterns for the project. The function drawLine(…) takes in the starting x and y position, the ending x and y position, and the “color”, which is either HIGH or LOW. HIGH meaning the led at that coordinate is on and LOW meaning the led is off. This function does not change the actual led matrix, instead it modifies an internal bitmap buffer. This is to increase performance for the led matrix, something that the LedControl library could not do. The function fillScreen(…) fills the entire led matrix with two possible values, HIGH or LOW. By using this function, it can act as a “clear” for the entire led matrix. It’s very useful when creating new patterns and removing them to allow for a newer pattern to take its place. The most important function would be the write() function. The function of write() is to “write” or display the entire led matrix at once by sending the bitmap buffer from the drawLine(…) function to the led matrix itself.

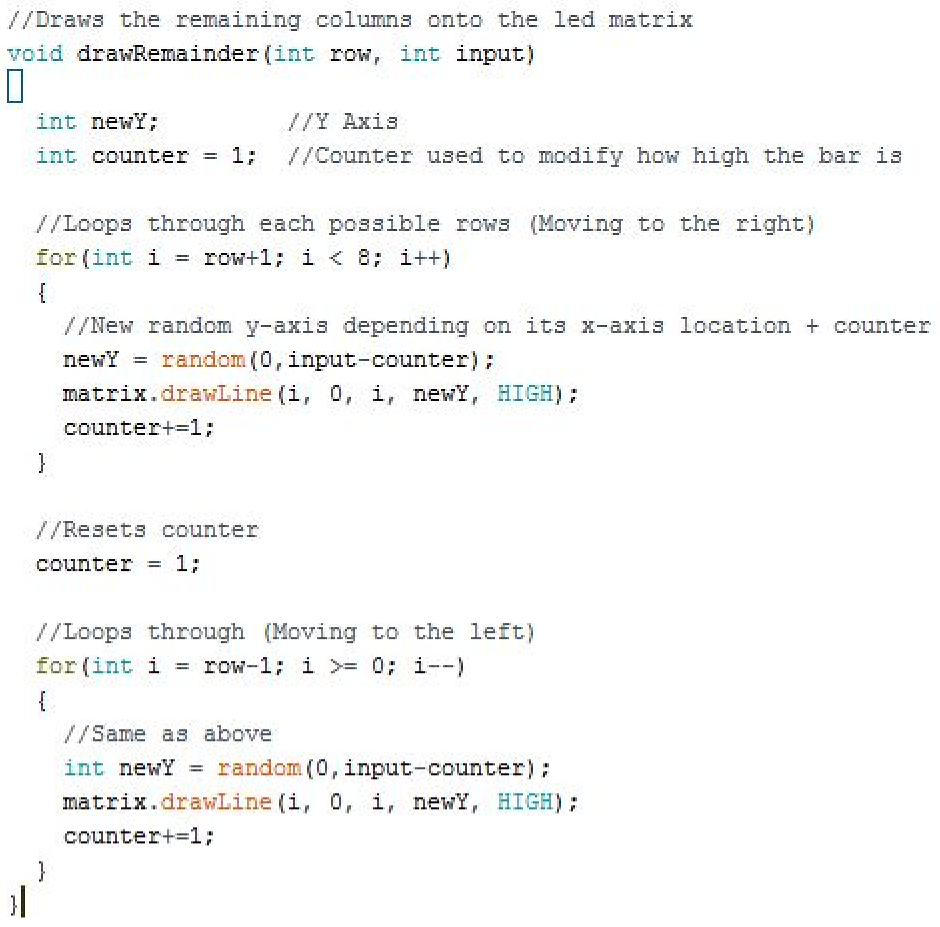
By using a combination between these three function, it allows for different patterns to be created. The Adafruit-GFX library also offers various different functions to create various shapes using the same idea as drawLine(…).

<u>**Pattern (Jumping Bars)**</u>

The code for creating a “jumping” bar visualizer consist of three different functions. The first function is the main function that ties in the remaining two functions. It checks input from the sound sensor that ranges from 0-7. After that, two functions are called, one function draws the column line into the matrix. The other function then draws the remaining lines on the rest of the led matrix. The following is the code are the two helper functions in order to draw jumping bars onto the led matrix:



```
//Draws ONE bar onto the led matrix (Main Bar)
void drawBar(int row, int input)
{
  matrix.drawLine(row, 0, row, input, HIGH);
}
```

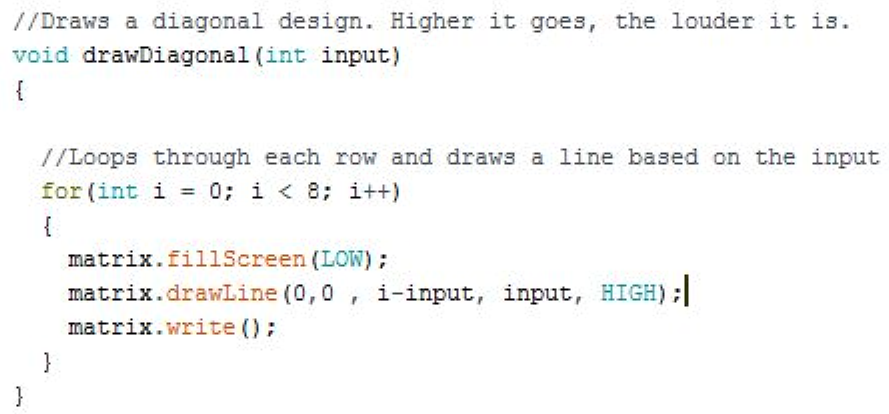


```
//Draws the remaining columns onto the led matrix
void drawRemainder(int row, int input)
{
  int newY;          //Y Axis
  int counter = 1;   //Counter used to modify how high the bar is

  //Loops through each possible rows (Moving to the right)
  for(int i = row+1; i < 8; i++)
  {
    //New random y-axis depending on its x-axis location + counter
    newY = random(0,input-counter);
    matrix.drawLine(i, 0, i, newY, HIGH);
    counter+=1;
  }

  //Resets counter
  counter = 1;

  //Loops through (Moving to the left)
  for(int i = row-1; i >= 0; i--)
  {
    //Same as above
    int newY = random(0,input-counter);
    matrix.drawLine(i, 0, i, newY, HIGH);
    counter+=1;
  }
}
```

The drawBar(…) method draws one horizontal line with the value of the input from the sound sensor. For example, if the input of the sound sensor is 8 (the loudest), then the matrix will draw a horizontal line with the max led lights at the far end. If the input is 2, then the matrix will draw a line at row 2, with only a height of 2, meaning a softer volume. Now since this function only draws one line onto the matrix, we want to fill in the remaining areas of the matrix, so that's where drawRemainder(…) comes along. This function will take in the specific row where the first horizontal line was drawn, and draws around that line, filling in the remaining area of the matrix. The algorithm used will decrease the original level of volume so that the led matrix will display smaller values the further it goes from the original starting position.

Pattern (Diagonal Line):

This design is similar to the "Jumping Bars" design. One main function and one helper function. The main function, again, takes an input between 0-7 from the sound sensor and calls the helper function given the input and row. Below is the helper function code that "draws" he designed based on the input of the sound sensor:



```
//Draws a diagonal design. Higher it goes, the louder it is.
void drawDiagonal(int input)
{

  //Loops through each row and draws a line based on the input
  for(int i = 0; i < 8; i++)
  {
    matrix.fillScreen(LOW);
    matrix.drawLine(0,0 , i-input, input, HIGH);
    matrix.write();
  }
}
```

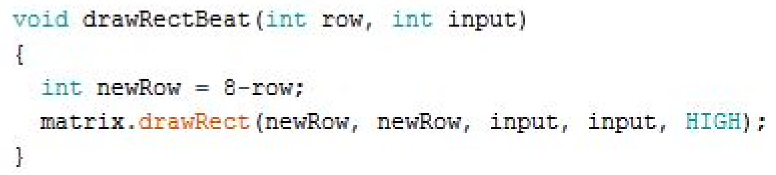
The drawDiagonal(…) draws a diagonal onto the led matrix and will change depending on the input that is given. The higher the input, the higher the diagonal will rise.

<u>Pattern (Rectangle):</u>

This design would be similar to the Diagonal pattern where instead of drawing a single line to one point to another, we are to draw a square where it's size will be determined by the input between 0-7 from the sound sensor. The higher the input, the bigger the square and vice versa. The function, drawRect(…) is given from the Adafruit-GFX library and it does what it says, draws a rectangle onto the led matrix. The function:

→ **drawRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)**

Takes in the starting x and y positions, the width, height, and "color" meaning HIGH or LOW. Below is the function used for drawing the square beat design:



```
void drawRectBeat(int row, int input)
{
  int newRow = 8-row;
  matrix.drawRect(newRow, newRow, input, input, HIGH);
}
```

The function drawRectBeat(…) will draw a square onto the matrix and depending on the volume given from the sound sensor, will grow or shrink the square on the led matrix.

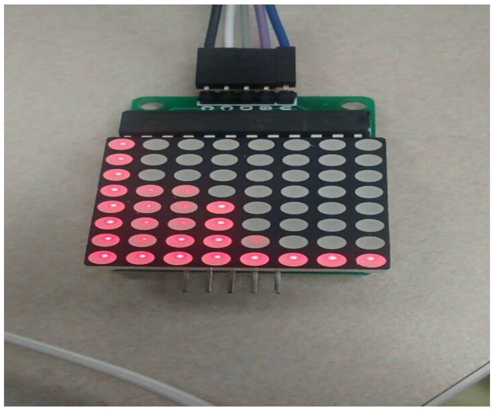
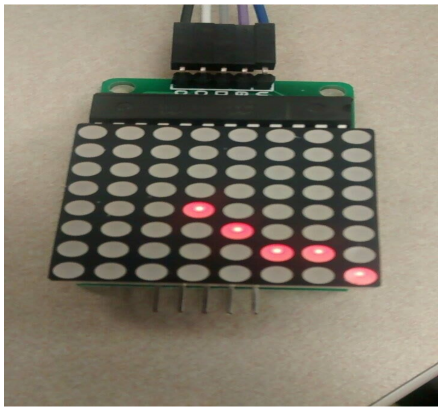
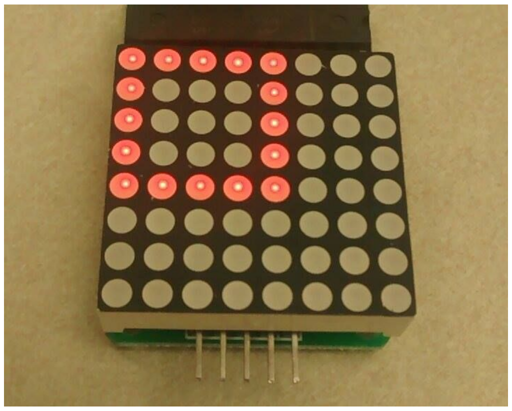
<u>Obstacles</u>

During our trials with the led matrix, we have ran into several issues on the way. One of the issue was related to the hardware of the actual led matrix. We found out that the led matrix we have only supports the color red and thus creating colorful patterns became impossible for us. We decided to scratch that idea and to keep the color red as it would not matter in the overall project.

Another issue we had was during the code development. The library LedControl we were using to control what the matrix displays was not quick enough to react correctly to the values our sound sensor was sending. For example, when trying to change the display for the led matrix to simulate sound inputs being sent in, the led would work fine; however, it would take about a second for it to completely change to the new pattern. So because of this, we did a little bit more research into the specific type of led matrix and found that the library, Max72xxPanel, was made specifically for our led matrix and reacted quickly enough for it to accurately display the sound values being sent onto the led matrix. The library also uses the Adafruit-GFX library to assist in creating and displaying patterns onto the led matrix.

Result

Using the Max72xxPanel and Adafruit-GFX library, we designed several patterns to simulate sound inputs from the sound sensor. The first pattern consist of the traditional jumping bars visualizer. In summary, the louder the sound, the higher the bar jumps and the softer the sound, the lower the bar jumps. Our second pattern would consist of a diagonal line that jumps higher the louder the sound. Our third pattern is a box like pattern that grows bigger the louder the sound is. Below are static pictures of how it should look like (Jumping Bars, Diagonal Line, Growing Square respectively):

References

Most if not all of our research for the Led Matrix was from websites and videos.

Audio Processing

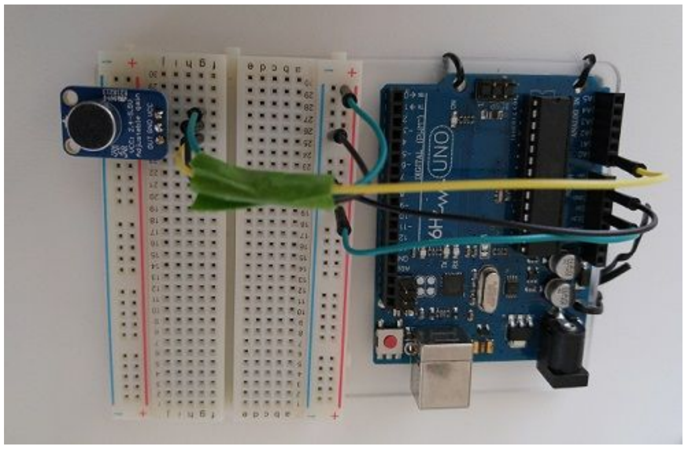
<u>Ideas</u>

For the audio capture, we wanted the led matrix to use the frequency or volume recorded by the device to display the patterns accordingly. Another option was to have the pattern change depending on the values recorded by the microphone. The values from the analog input would be set into ranges and assigned to each unique pattern. We are aiming to have enough patterns to cycle through that will be able to utilize more than 3 frequency ranges recorded by the sound sensor.

<u>Hardware Progression</u>

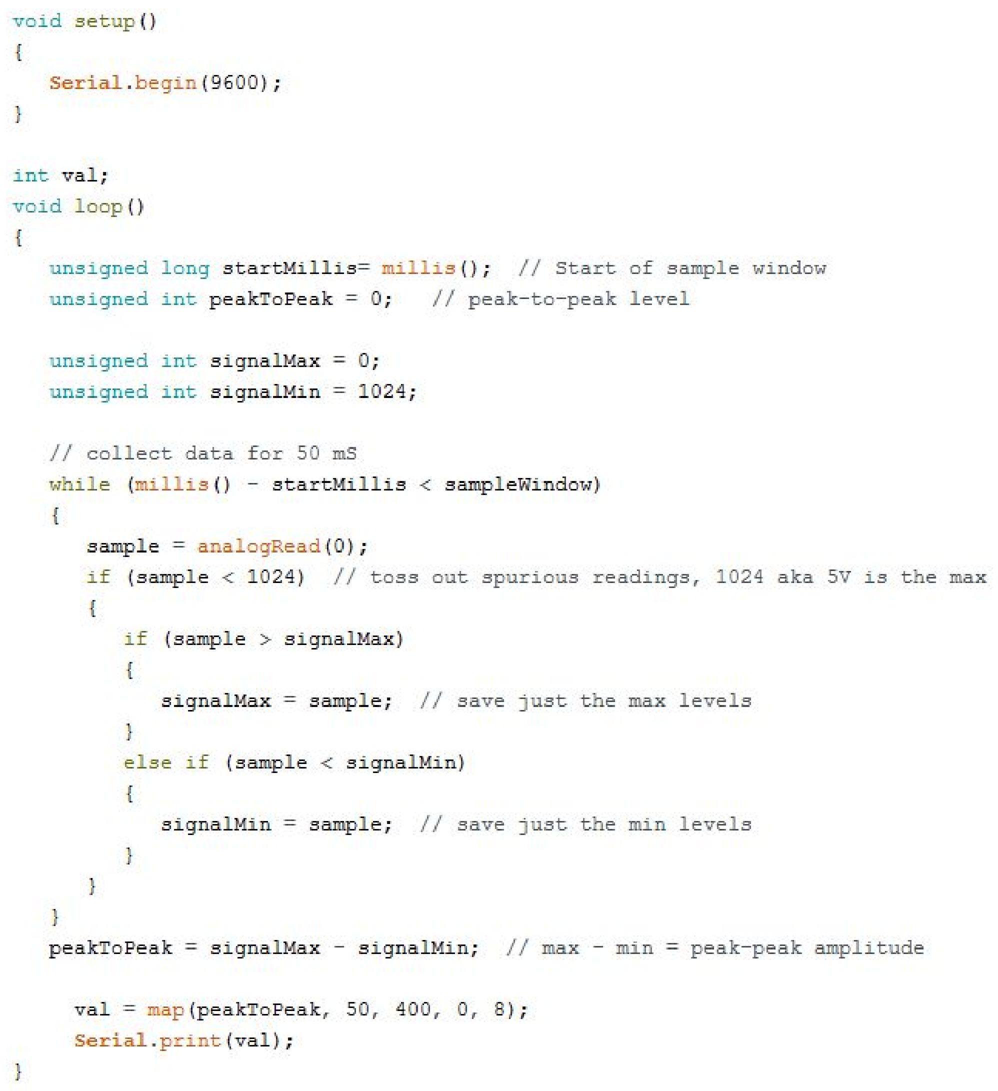
At the beginning we were able to acquire an electret microphone that we hoped would be enough to recording audio values. Unfortunately we could not use only the electret microphone to record sound, so we decided on ordering a sound sensor that was also easier to wire to the arduino. The amplifier that we ordered is the adafruit Electret Microphone Amplifier MAX4466 with adjustable gain.

From the picture when we ordered it we assumed the pins would already be soldered on, but they weren't so we had to do that before working on the audio recording. The wiring for the amplifier is also simple with VCC to 5v, GND to ground, and OUT to an analog pin, which in our case was 0.



Software Progression/Code

The code that we used was found on adafruit's website and is listed in bibliography below. Most of the computation happens in the looping.



```
void setup()
{
   Serial.begin(9600);
}

int val;
void loop()
{
   unsigned long startMillis= millis();  // Start of sample window
   unsigned int peakToPeak = 0;   // peak-to-peak level

   unsigned int signalMax = 0;
   unsigned int signalMin = 1024;

   // collect data for 50 mS
   while (millis() - startMillis < sampleWindow)
   {
      sample = analogRead(0);
      if (sample < 1024)  // toss out spurious readings, 1024 aka 5V is the max
      {
         if (sample > signalMax)
         {
            signalMax = sample;  // save just the max levels
         }
         else if (sample < signalMin)
         {
            signalMin = sample;  // save just the min levels
         }
      }
   }
   peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude

     val = map(peakToPeak, 50, 400, 0, 8);
     Serial.print(val);
}
```

We used a sample window of 50 which allows the sound captured to be as low as 20Hz. The peakToPeak value is the amplitude from one wave to the other. Within the while loop, the data from the sound sensor is recorded. The if statement will filter out readings that are not within the correct range (0 -1023) and set them to be the new range.

The code that was added is the mapping function at the bottom of the loop function. This function will map out the values recorded by the sound sensor and map them to be 8 values. This allows us to assign them to the 8x8 led matrix by giving us a range using the 1023 values. The final serial print will send that value over to the receiving arduino to display the patterns.

Obstacles

With the electret microphone, we had to research many possibilities on how we could use it without the need for purchasing extra components. From reading many of the wiring diagrams as well as reading different forums we discovered that it seems, in order to use the microphone on its own, a amplifier was required. In order for the microphone to record significant changes in the audio it is necessary to have the amplifier within the circuit. Neither of our kits supplied amplifiers so we would either have to buy an amplifier and work with the type that we have, or buy a sound sensor that would have all parts to support the electret microphone.

During our research we also ran into an issue with one of our ideas which was to have a voice command recognition to change the patterns on the display. We discovered that most people recommended the use of a VR shield to recognize voice, but that feature comes with a high price that we definitely could not afford for the project. Because of this, the idea of implementing voice command had to be scraped.

The sound sensor became one of the last components we worked on because we were trying to research anything we could that would allow us to use just the electret microphone by itself. Our goal to was to avoid purchasing anything unnecessary since we are college students and the struggle is real. Unfortunately in the end we chose to purchase the sensor and had to wait for it to arrive before being able to move forward with the project because at that point, most of the Led Matrix designs were already done.

Result

The sound sensor works as expected and is able to capture sound and display in real time. The sound sensor can only capture music if the speaker is very close to the sensor itself, if not it ends up capturing ambient noises and doesn't display the correct pattern onto the matrix. We tested it by blasting music on a pair of headphones so that the sound could funnel onto the sensor as well as using our phone speakers and playing on to top of the sound sensor.

References

Most if not all of our research for the audio processing was from websites and videos.

Arduino Communication

Now that we had our separate components working, we had to find a way to join the two arduinos together to create one cohesive device. We decided to go with what we know, which was serial communication. It requires no extra hardware and we were familiar with it after doing it in lab.

Hardware Progression

We are using two arduino Uno's for this communication/ Setting up the communication, we used the arduino with the amplifier (we will call this arduino 1) as the transmitter and set the other arduino as a receiver (we will call this arduino 2. We connected both arduinos with a common ground and set arduino 1's TX to arduino 2's RX using a jumper cable.

Software Progression

For the software, each arduino would have to have different code since each had a different role. The transmitter code was simply a serial print statement with the mapped value of the audio that was captured. This statement then sends that message to the receiving arduino. On the receiver side, the arduino would need to be listening at all times for any incoming bytes. So we used serial.available to listen on the serial port and capture the bytes. We then read the values using Serial.read and converted the bytes read into an integer value so that we could manipulate and used for displaying the patterns.

Obstacles

Our biggest issue was at the beginning. We were unable to find out why the values sent from one arduino was different when it was read by the other. Not only that, the values on the receiving arduino did not change with the surround sounds inputted into the amplifier. When we tried to reference our previous lab, we could not find an issue with the code that we were using as it has worked before with mutual communication between the two arduinos. The issue was fixed by using the mapping function and sending over a smaller amount of inputs, in this case the numbers 0 to 8.

Additional Components

We wanted to add additional components to the main project and some ideas such as adding a photoresistor and a motion sensor were attempted. Because our motion sensor was not successful, we instead chose to do the pattern switching with a button instead.

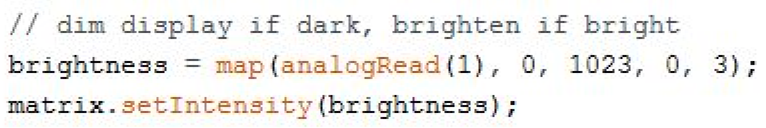
Light Control

Our idea was to have the matrix react to different light values in the environment by dimming or getting brighter which is something that is common in modern phones and devices. We used a photoresistor to do so.

Implementation

The wiring for the photoresistor consisted of a 10K ohm resistor and 3 wires. One end of the photoresistor goes to ground, the other end goes to analog pin 1 along with one end of the 10K resistor. The other end of the 10K resistor connects to power.

Code:



```
// dim display if dark, brighten if bright
brightness = map(analogRead(1), 0, 1023, 0, 3);
matrix.setIntensity(brightness);
```

Brightness is some value from 0 -1023 that we receive from the photoresistor. We then map this value to 3 values which will allow 3 settings for brightness of the matrix. The last line then sets the matrix brightness based on the value that is given back from the map function.

Motion Sensor

Our original idea was to use the "motion sensor" in order to detect a hand wave that will then change the pattern displayed on the matrix. The motion sensor we are referring to is the HC-SR04 Ultrasonic Sensor that was included in our kits. It detects objects in front of it by echoing an ultrasound that will bounce off the object.

Implementation Attempt

At first we tried to place the motion sensor on the arduino that also held the sound sensor. Our thought process was that the arduino would be able to send both values across and the receiving arduino would be able to separate these two messages based on their value. Unfortunately we were unable to do so as the values being sent often fluctuate so much so that the the receiving arduino was getting confused as to what values should be displayed and when to change the pattern.

We then tried placing the motion sensor with some test code onto the arduino holding the matrix. This time the matrix was responsive to the motion sensor, but the calculation done by the motion sensor per looping cycle with the photoresistor and the sound sensor slowed down the display on the matrix. This created a lag effect that was no longer synchronizing with the music that was playing. Because of this lag, we chose to not use the motion sensor as a part of our project.

Pattern Switching

For pattern switching we are using a simple button that will change between the 3 patterns when it is pressed. The button will be placed on the arduino that also holds the LED matrix. We want to avoid any extra traffic on the serial communication between the two arduinos.

Wiring for the button requires only two jumper wires and no resistors. We are able to use no resistors since in our code, we are going to use INPUT_PULLUP which utilizes the built in resistor within the arduino. One end of the button will be connected to power/5v, while the other end will be connected to ground.

When the button is pressed, a digital read will read in a 0 (versus a 1 when using regular input instead of input_pullup). The 0 will indicate that the pattern should change, and it will then call the switch_pattern function. The function then changes a testNum global variable that gets checked each time the pattern is printed to the LED matrix. This pattern change will not be randomized, but instead in a certain order. The order in this case will be bars, to diagonal line, to squares.

Hardware List

- 2 x Arduino Uno
- 1 x Led Matrix MAX7219
  - http://www.banggood.com/MAX7219-Dot-Matrix-MCU-LED-Display-Control-Module-Kit-For-Arduino-p-915478.html?p=DQ30066511122014069J
- 1 x Electret Microphone Amplifier MAX4466
  - https://www.adafruit.com/products/1063?gclid=CjwKEAjw9OG4BRDJzY3jrMng4iQSJABddor1nvj6HArG3AosraaB3JP2nxU7SKRagZ8OG9kFfuvqpxoC42vw_wcB
- 1 x Micro Photocell
  - https://www.sparkfun.com/products/9088?gclid=CjwKEAjw9OG4BRDJzY3jrMng4iQSJABddor1UxHk0lg6m32uAD51Rs_esmaqJm13WFVdTVb525OP6xoC3oDw_wcB
- 1 x Switch Button

Bibliography

Earl, Bill. "Adafruit Microphone Amplifier Breakout." *Measuring Sound Levels*. N.p., 13 Jan. 2013. Web. 22 Apr. 2016. <https://learn.adafruit.com/adafruit-microphone-amplifier-breakout/measuring-sound-levels>

Educ8s. "Arduino Tutorial: LED Matrix Red 8x8 64 Led Driven by MAX7219 (or MAX7221) and Arduino Uno." *YouTube*. YouTube, 13 Mar. 2015. Web. 22 Apr. 2016. <https://www.youtube.com/watch?v=TOuKnOG8atk>

"MAX7219 Dot Matrix MCU LED Display Control Module Kit For Arduino With Dupont Cable." *Www.banggood.com*. N.p., n.d. Web. 22 Apr. 2016. <http://www.banggood.com/MAX7219-Dot-Matrix-MCU-LED-Display-Control-Module-Kit-For-Arduino-p-915478.html?p=DQ30066511122014069J>

"Arduino Playground - LedControl." *Arduino Playground - LedControl*. N.p., n.d. Web. 22 Apr. 2016. <http://playground.arduino.cc/Main/LedControl>

"Tiny Arduino Music Visualizer." *Code*. N.p., n.d. Web. 22 Apr. 2016. <https://learn.adafruit.com/piccolo/code>

"Arduino Playground - LedControlDemos." *Arduino Playground - LedControlDemos*. N.p., n.d. Web. 22 Apr. 2016. <http://playground.arduino.cc/Main/LedControlDemos>

"Blog." Shure Blog. N.p., n.d. Web. 22 Apr. 2016. <http://blog.shure.com/how-to-read-a-microphone-frequency-response-chart/>