Visualisation of Audio using Dot Matrix Display

Submitted by

Roopesh O R
Prabhath C S
Pranav Praveen

Division of Electronics Engineering
School of Engineering
Cochin University of Science and Technology
Kochi - 682022

June 2024

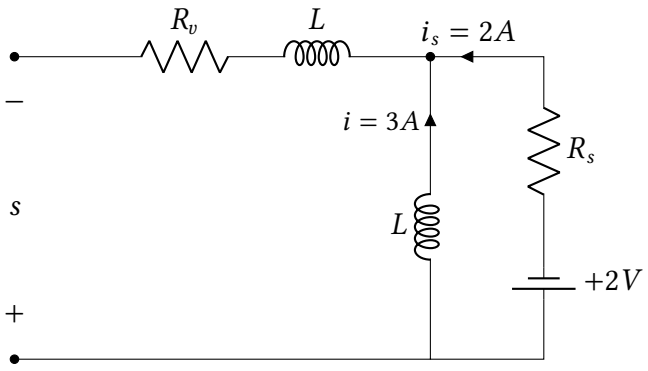
Abstract

From simple music player applications to concerts, audio visualization enhances the ambience and overall experience of the enjoyer. In this project, we replicate a simpler version of such systems using an Arduino UNO development board, 32*8 LED dot matrix display, and a few small components. An audio signal of our interest is preprocessed and fed to one of the analogue pins of Arduino. This signal is then processed using the "ArduinoFFT" library, which splits the audio into discrete chunks and performs FFT on it. The resulting frequency spectrum is then again processed and scaled down to match the width and height of the LED display. Finally, the resulting waveform is displayed. In the end, we try to add more customizations such as different "display modes" and effects.

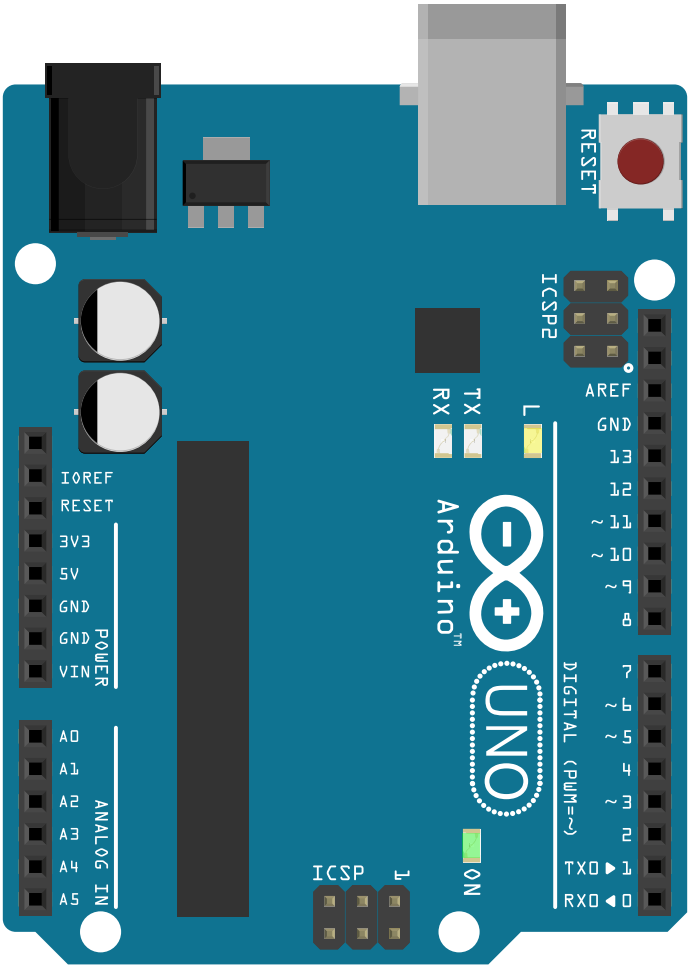
Implementation

Introduction

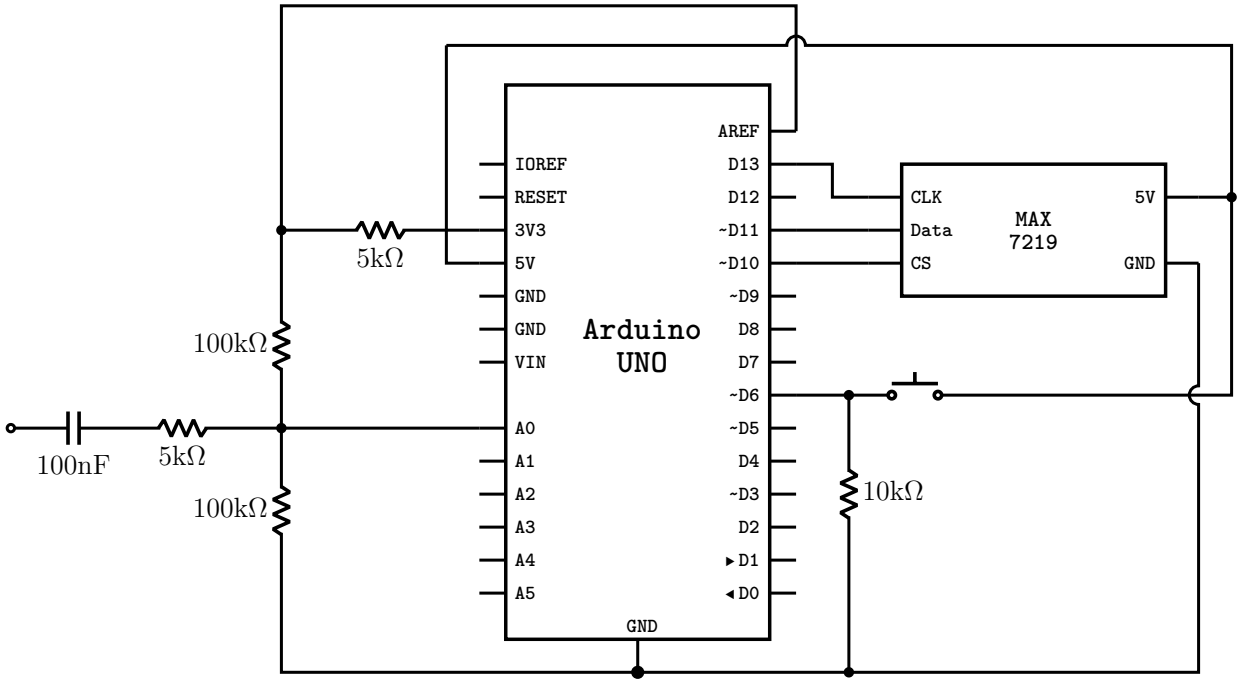
Block Diagram



Arduino UNO


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
TX
RX
L
ON
ICSP
1
Arduino™
UNO
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

Hardware



Program

```
#include <arduinoFFT.h>
#include <MD_MAX72xx.h>

#define SAMPLES 64
#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define CLK_PIN 13
#define DATA_PIN 11
#define CS_PIN 10
#define xres 32
#define yres 8

const int buttonPin = 5;
int displaymode = 1;
int previousState = LOW;

double vReal[SAMPLES];
double vImag[SAMPLES];

char data_avgs[xres];
```

```
21
22  int yvalue;
23  int displaycolumn, displayvalue;
24  int peaks[xres];
25
26  unsigned long lastDebounceTime = 0;
27  unsigned long debounceDelay = 200;
28  const float samplingFrequency = 8000;
29
30  int CURRENT_PATTERN[] = { 0, 1, 3, 7, 15, 31, 63, 127, 255 };
31  const int MODES = 2;
32  int PATTERNS[MODES][9] = {
33    { 0, 1, 3, 7, 15, 31, 63, 127, 255 },// standard pattern
34    { 0, 1, 2, 4, 8, 16, 32, 64, 128},    // peak pattern
35  };
36
37
38  MD_MAX72XX mx = MD_MAX72XX(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);
39  ArduinoFFT<double> FFT = ArduinoFFT<double>(vReal, vImag, SAMPLES, samplingFrequency);
40
41  void setup() {
42    // set ADC to free running mode and set pre-scalar to 32 (0xe5)
43    ADCSRA = 0b11100101;
44    ADMUX = 0b00000000;// A0
45
46    pinMode(buttonPin, INPUT);
47    mx.begin();
48    // wait to get reference voltage stabilized
49    delay(50);
50    mx.setColumn(0, 63);
51    mx.setColumn(31, 7);
52    delay(900);
53  }
54
55  void loop() {
56    // Sample collection
57    for (int i = 0; i < SAMPLES; i++) {
58      // wait for ADC to complete current conversion
59      while (!(ADCSRA & 0x10));
60      // clear ADIF bit so that ADC can do next operation (0xf5)
61      ADCSRA = 0b11110101;
62      int value = ADC - 512;
63      vReal[i] = value / 8;
64      vImag[i] = 0;
65    }
66
67    FFT.windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
68    FFT.compute(vReal, vImag, SAMPLES, FFT_FORWARD);
69    FFT.complexToMagnitude(vReal, vImag, SAMPLES);
70    int step = (SAMPLES / 2) / xres;
71
72    // shift averages
73    for (int i = 0; i < xres; i += step) {
74      data_avgs[i] = 0;
75      for (int k = 0; k < step; k++) {
76        data_avgs[i] = data_avgs[i] + vReal[i + k];
```

```
      }
      data_avgs[i] = data_avgs[i] / step;
    }

    // Display
    for (int i = 0; i < xres; i++) {
      data_avgs[i] = constrain(data_avgs[i], 0, 80);
      data_avgs[i] = map(data_avgs[i], 0, 80, 0, yres);
      yvalue = data_avgs[i];

      peaks[i] = peaks[i] - 1;
      if (yvalue > peaks[i])
        peaks[i] = yvalue;
      yvalue = peaks[i];
      displayvalue = CURRENT_PATTERN[yvalue];
      displaycolumn = i;
      mx.setColumn(displaycolumn, displayvalue);
    }
    // check if button pressed to change display mode
    displayModeChange();
  }

  void displayModeChange() {
    int reading = digitalRead(buttonPin);
    if (
      reading ^ previousState &&
      (millis() - lastDebounceTime) > debounceDelay
    ) {
      displaymode = (displaymode + 1) % MODES;
      for (int i = 0; i <= 8; i++) {
        CURRENT_PATTERN[i] = PATTERNS[displaymode][i];
      }
      lastDebounceTime = millis();
    }
    previousState = reading;
  }
```

Conclusion

References