Su22-ENGR-40M-01 Lab 3b

Jannah Sabic El-Rayess

TOTAL POINTS

70 / 70

QUESTION 1

1 Complexity 25 / 25

+ **0 pts** Choose hardware/software complexity below
+ **5 pts** No additional hardware
+ **10 pts** Minor additional hardware
✓ **+ 15 pts** **Moderate additional hardware**
+ **20 pts** Complex additional hardware
+ **5 pts** Simple hardware response
+ **5 pts** Simple software response
✓ **+ 10 pts** **Minor Software complexity**
+ **15 pts** Moderate software complexity
+ **20 pts** Impressive software complexity
- **12.5 pts** Late

QUESTION 2

2 Lab Questions / Report 10 / 10

✓ **- 0 pts** **Correct**
- **1 pts** 1 question missing/incorrect
- **2 pts** 2 questions missing/incorrect
- **3 pts** 3 questions missing/incorrect
- **4 pts** 4 questions missing/incorrect
- **7 pts** several/majority questions missing/incorrect

QUESTION 3

3 Coding Style 15 / 15

✓ **- 0 pts** **Correct**
- **15 pts** Incomplete or missing

QUESTION 4

4 Build Quality 15 / 15

+ **5 pts** plus
✓ **- 0 pts** **check plus**
- **4 pts** check
- **8 pts** check -
- **11 pts** minus

QUESTION 5

5 Cleanup 5 / 5

✓ **- 0 pts** **spotless**
- **2 pts** missed a spot
- **5 pts** did not clean up

1 Complexity 25 / 25

+ **0 pts** Choose hardware/software complexity below
+ **5 pts** No additional hardware
+ **10 pts** Minor additional hardware
✓ **+ 15 pts** **Moderate additional hardware**
+ **20 pts** Complex additional hardware
+ **5 pts** Simple hardware response
+ **5 pts** Simple software response
✓ **+ 10 pts** **Minor Software complexity**
+ **15 pts** Moderate software complexity
+ **20 pts** Impressive software complexity
- **12.5 pts** Late

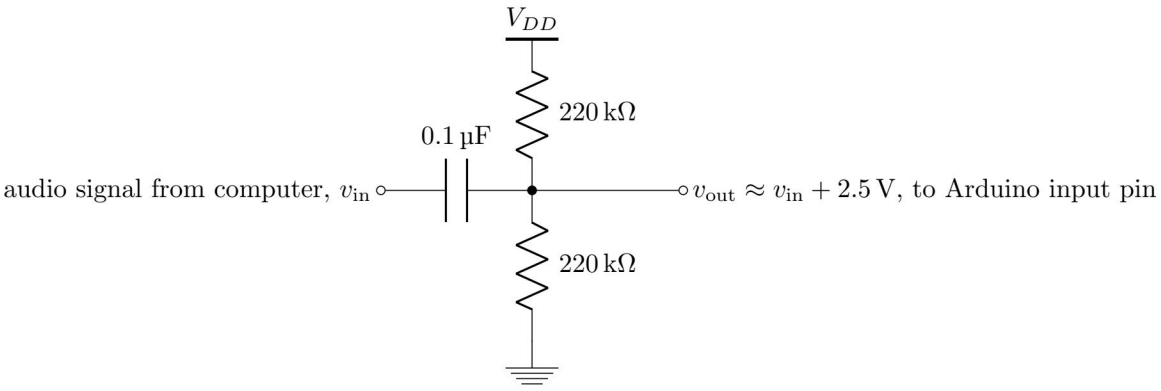
1. Put an Arduino into a breadboard. (You might need to pull the one currently on your LED board.)
2. Open up a power supply window in the WaveForm Software
3. Connect the power supply pin to one of the analog input pins on your Arduino (`A0` through `A5`). The ground lead of the AD2 connects to the ground of the arduino, and the AD2 power supply lead connects to your input pin.
4. Download the starter file adc_test.ino from Canvas. This file continually reads an analog input pin and prints its value to the Serial Monitor. Change the first line to the analog input pin you're using.

L1: For each voltage in the left column, set the PSU to approximately that voltage. Record the *actual* voltage you set, and what digital value is returned by `analogRead()`.

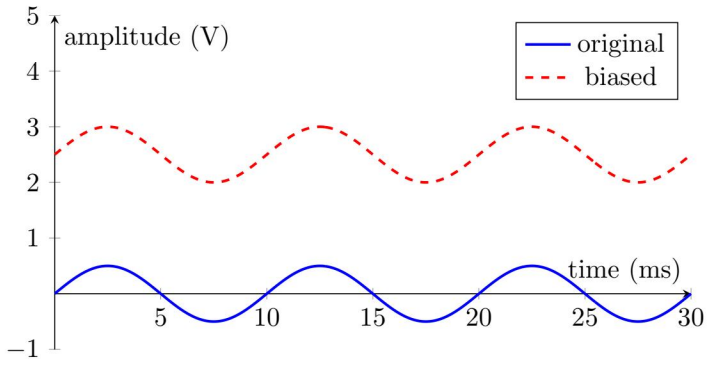
Intended voltage	Actual voltage	Value from `analogRead()`
0 V		7
1 V		211
2 V		417
2.5 V		521
3 V		624
4 V		830
5 V		1023

5.2 Connecting a biasing circuit

We can't connect the audio signal straight to the ADC input. Audio signals are centred around zero, so are negative around half the time, and the ADC can't detect voltages below 0 V. So we need a circuit that will add a 2.5 V DC offset to the signal, so that it is centered in the middle of the 0 V to 5 V input range. This circuit is called a *biasing circuit*, and one such circuit looks like this:



We will analyze this circuit in this week's homework. For now, here's the idea: the capacitor "blocks" the DC component, so that the DC bias of the output is determined solely by the two resistors. At the same time, the capacitor "allows" the AC component (the audio signal) to pass through. The plot below shows what the Arduino's ADC will see from a 1 Vpp, 100 Hz sine wave before (blue, solid) and after (red, dashed) biasing.



1. Put an audio connector into your breadboard and use an audio cable to connect your phone's or computer's audio output to it. Play some music, and max out your device's volume. (Be sure to do this, we want the range to be as big as it can be!)

   On the audio connector, the middle pin is ground, and the other two are the left and right (stereo) signals respectively. The ground pin should connect to ground. It doesn't matter which of the other two you use, but make sure the left and right pins aren't connected together in the same row. Use the scope window on the AD2 to probe either the left or the right pin. (Don't forget to connect ground.)

   L2: What is the voltage range that it covers, roughly?

   -0.5V to 0.5V

2. Add the *biasing circuit* above to your breadboard, connecting it to the Arduino V_{DD} and ground where necessary, but *don't connect it to the Arduino audio input pin* (yet). Have a look at the output voltage of the biasing circuit (which will become the input voltage to the ADC). Verify that it looks just like the input signal, except raised by 2.5 V.

   L3: What is the voltage range that this signal covers?

   1.5V to 2.8V

3. Set up a waveform generator on the AD2 to emulate this voltage range. The amplitude should match your audio signal's range, with an offset of 2.5 V. Set the frequency to 1 Hz. This signal emulates your voltage range *after* the biasing circuit. Connect the signal generator to your Arduino's analog input. (Your biasing circuit should **not** be connected to the Arduino.) Have a look at the ADC values being reported in the Serial Monitor.

   L4: What are the minimum, maximum and midpoint of values you see in the Serial Monitor?

   min = 323, max = 591, mid = 457

The fork in the road: If you're doing the raw signal option, move on to section 5.3. If you're doing the frequency spectrum option, skip to section 5.4.

2 Lab Questions / Report **10 / 10**

✓ **- 0 pts Correct**

- 1 pts 1 question missing/incorrect

- 2 pts 2 questions missing/incorrect

- 3 pts 3 questions missing/incorrect

- 4 pts 4 questions missing/incorrect

- 7 pts several/majority questions missing/incorrect

```
/* LED plane responding to audio by amplitude (starter code)
 *
 * Lights an LED array to respond to audio.
 *
 * == Setting up the Serial Monitor ==
 * The Serial Monitor must be configured (bottom-right corner of the screen) as
 * baud rate 115200.
 *
 * ENGR 40M
 * Stanford University
 * July 2018
 */

const byte ANODE_PINS[8] = {13, 12, 11, 10, 9, 8, 7, 6};
const byte CATHODE_PINS[8] = {A3, A2, A1, A0, 5, 4, 3, 2};
const byte AUDIO_INPUT_PIN = A5;

byte pattern1[8][8] = { {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 1, 0, 1, 0, 1, 0, 0},
                        {1, 1, 1, 1, 1, 1, 1, 1} };

byte pattern2[8][8] = { {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 1, 0, 1, 0, 0, 1},
                        {1, 0, 1, 1, 1, 0, 1, 1},
                        {1, 1, 1, 1, 1, 1, 1, 1} };

byte pattern3[8][8] = { {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 1, 0, 0, 1, 0},
                        {1, 1, 0, 1, 0, 1, 1, 0},
                        {1, 1, 1, 1, 1, 1, 1, 1},
                        {1, 1, 1, 1, 1, 1, 1, 1} };

byte pattern4[8][8] = { {0, 0, 0, 0, 0, 0, 0, 0},
```

```
                          {0, 0, 0, 0, 0, 0, 0, 0},
                          {0, 0, 0, 0, 0, 0, 0, 0},
                          {0, 0, 1, 0, 1, 0, 0, 1},
                          {0, 0, 1, 1, 1, 1, 0, 1},
                          {0, 0, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1} };

byte pattern5[8][8] = { {0, 0, 0, 0, 0, 0, 0, 0},
                          {0, 0, 0, 0, 0, 0, 0, 0},
                          {0, 0, 0, 1, 0, 0, 0, 0},
                          {0, 0, 0, 1, 0, 0, 1, 0},
                          {1, 0, 1, 1, 0, 1, 1, 0},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1} };

byte pattern6[8][8] = { {0, 0, 0, 0, 0, 0, 0, 0},
                          {0, 0, 0, 1, 0, 0, 0, 0},
                          {0, 0, 0, 1, 1, 0, 0, 1},
                          {1, 1, 0, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1} };

byte pattern7[8][8] = { {1, 0, 0, 1, 0, 0, 0, 0},
                          {1, 0, 0, 1, 0, 0, 0, 0},
                          {1, 0, 1, 1, 1, 1, 1, 0},
                          {1, 1, 1, 1, 1, 1, 1, 0},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1} };

byte pattern8[8][8] = { {0, 1, 0, 1, 1, 0, 0, 1},
                          {1, 1, 1, 1, 1, 0, 0, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1},
                          {1, 1, 1, 1, 1, 1, 1, 1} };

void setup() {
```

```
  for (byte i = 0; i < 8; i++) {
    pinMode(ANODE_PINS[i], OUTPUT);
    pinMode(CATHODE_PINS[i], OUTPUT);
  }

  for (byte i = 0; i < 8; i++) {
    digitalWrite(ANODE_PINS[i], HIGH);
    digitalWrite(CATHODE_PINS[i], HIGH);
  }

  pinMode(AUDIO_INPUT_PIN, INPUT);
  Serial.begin(115200);
}

/* Function: display
 * -----------------
 * Runs through one multiplexing cycle of the LEDs, controlling which LEDs are
 * on.
 *
 * During this function, LEDs that should be on will be turned on momentarily,
 * one row at a time. When this function returns, all the LEDs will be off
 * again, so it needs to be called continuously for LEDs to be on.
 */
void display(byte pattern[8][8]) {
  // TODO: You need to fill in this function.
  // Here's some suggested pseudocode:
  //   for each anode (+/row) wire
  //     for each cathode (-/column) wire
  //       look up in pattern whether this LED should be on or off
  //       if LED should be on, activate cathode (-) wire, else deactivate it
  //     end for
  //     activate anode (+) wire
  //     wait a short time (hint: try delayMicroseconds())
  //     deactivate anode (+) wire
  //   end for
  for (byte i = 0; i < 8; i++) {
    for (byte j = 0; j < 8; j++) {
      if (pattern[i][j] == 1) {
        digitalWrite(CATHODE_PINS[j], LOW);
      }
      else {
        digitalWrite(CATHODE_PINS[j], HIGH);
      }
    }
    digitalWrite(ANODE_PINS[i], LOW);
```

```
    delayMicroseconds(50);
    digitalWrite(ANODE_PINS[i], HIGH);
  }
}

/* Function: getSample
 * -------------------
 * Returns an unbiased sample of the audio signal. The returned value will be
 * "centered around zero", i.e., sometimes positive and sometimes negative.
 */
int getSample() {
  // TODO: Fill this in!
  int sample = analogRead(AUDIO_INPUT_PIN);

  // You might want to do some processing here, to unbias the sample.
  Serial.println(sample); // to serial monitor
  return sample;
}



/* Function: setLEDs
 * -----------------
 * Sets the LEDs array based on the given sample. The sample is assumed to
 * be returned by getSample() (and have the consequent properties).
 */
void setLEDs(byte pattern[8][8], int sample) {
  for (byte i = 0; i < 8; i++) {
    for (byte j = 0; j < 8; j++) {
      if (sample < 470) {
        pattern[i][j] = pattern1[i][j];
      }
      else if (sample < 485) {
        pattern[i][j] = pattern2[i][j];
      }
      else if (sample < 500) {
        pattern[i][j] = pattern3[i][j];
      }
      else if (sample < 515) {
        pattern[i][j] = pattern4[i][j];
      }
      else if (sample < 530) {
        pattern[i][j] = pattern5[i][j];
      }
      else if (sample < 545) {
        pattern[i][j] = pattern6[i][j];
```

```
      }
      else if (sample < 560) {
        pattern[i][j] = pattern7[i][j];
      }
      else if (sample < 575) {
        pattern[i][j] = pattern8[i][j];
      }
    }
  }
}

void loop() {
  static byte pattern[8][8];
  int sample = getSample();
  setLEDs(pattern, sample);
  display(pattern);
}
```

3 Coding Style 15 / 15

✓ **- 0 pts** **Correct**

- 15 pts Incomplete or missing

4 Build Quality **15 / 15**

+ **5 pts** plus

✓ **- 0 pts check plus**

- 4 pts check

- 8 pts check -

- 11 pts minus

5 Cleanup 5 / 5

✓ **- 0 pts** spotless

- 2 pts missed a spot

- 5 pts did not clean up