PROGRAMMING YOU CAN HEAR

AUDIO SIGNAL MANIPULATION IN THE C/C++ LANGUAGE

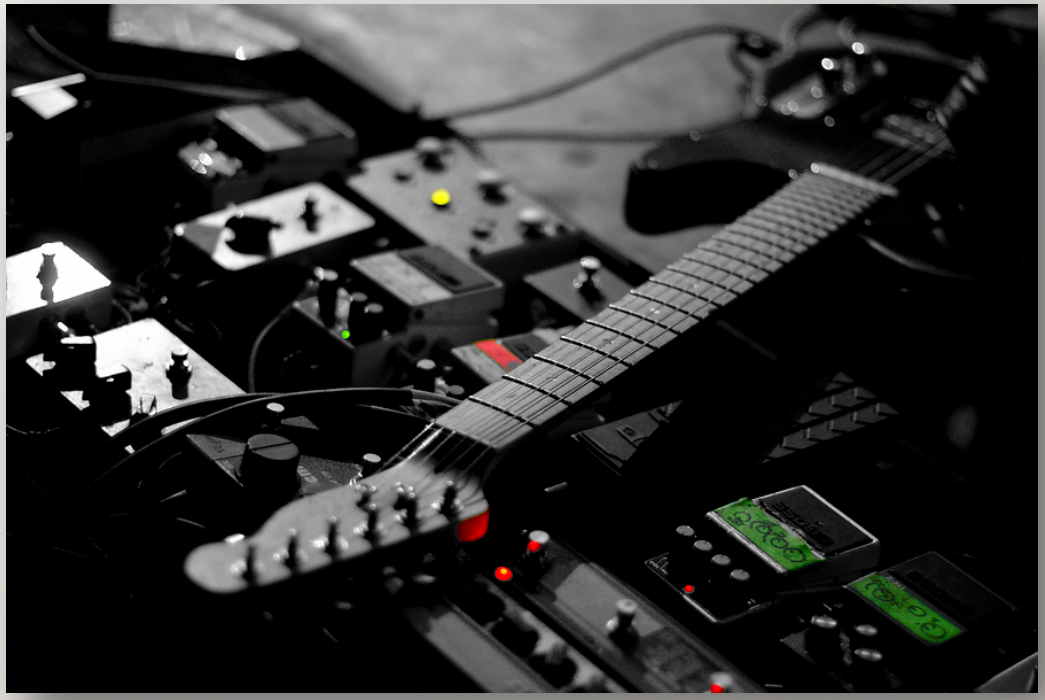
BY TANNER CHERRY

OBJECTIVES:

- To do hands-on work in creating, wiring, and programming a guitar pedal.
- To develop a greater understanding and learn how to send, receive, and manipulate an audio signal using the C/C++ language in the Arduino IDE.

WHAT IS A GUITAR PEDAL?

- Also called an effects pedal.
- Modifies the tone of an electric guitar.
- Takes an audio signal input and returns a modified audio output.
- Only a multi-effects pedal will hold more than one effect, but tend to be large with many buttons.



THE GAME PLAN

PLAN A:

- Gather the materials and build the physical structure of the pedal.
- Program an Arduino Uno Rev 3 chip to manipulate the signal.
- Develop an even deeper understanding by creating my own unique code to manipulate the audio signal with overdrive, reverb, etc.

Did it work out as I planned?

Sadly not quite ☹

PROBLEMS DURING PLAN A:

- I am not as skilled in electrical engineering as I wish I was.
- Though the audio signal can run through the circuit, the circuit and the chip had problems working together due to a voltage issue involving the signal input.

PLAN B:

- Assemble the body of the guitar pedal and wire to the best of my ability.
- Study on how to manipulate the audio signal through C/C++ Language and develop a complete understanding in order to utilize once the electrical parts of the pedal are complete.

THE PROJECT

MATERIALS:

- Metal Guitar pedal enclosure



- 3 Linear Taper 100 K Ohm Potentiometers

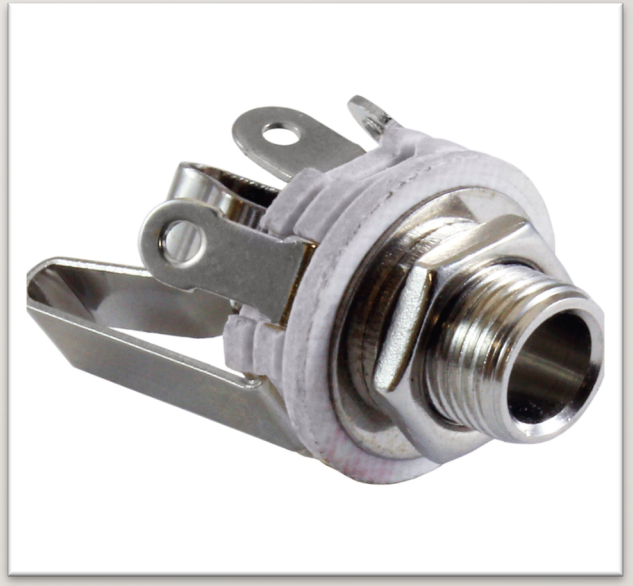


MATERIALS:

- 6-Pin, DPDT Stomp Switch

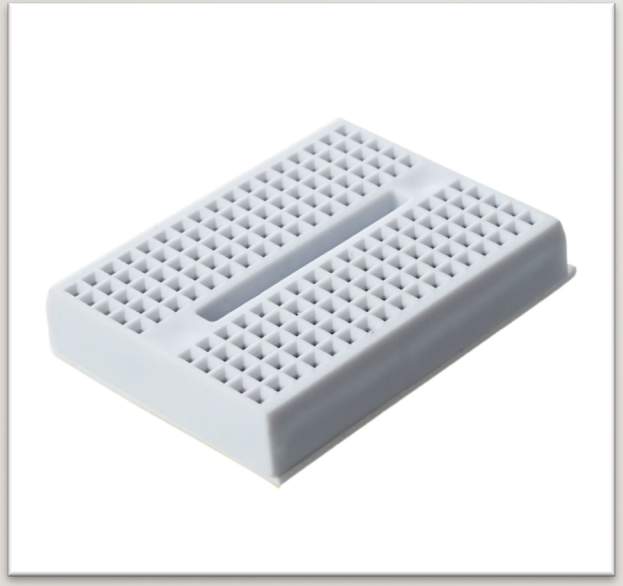


- 2 GLS Audio ¼” Jacks (Female, Stereo)



MATERIALS:

- 2-3 Mini Breadboards



- Arduino UNO Rev 3



MATERIALS:

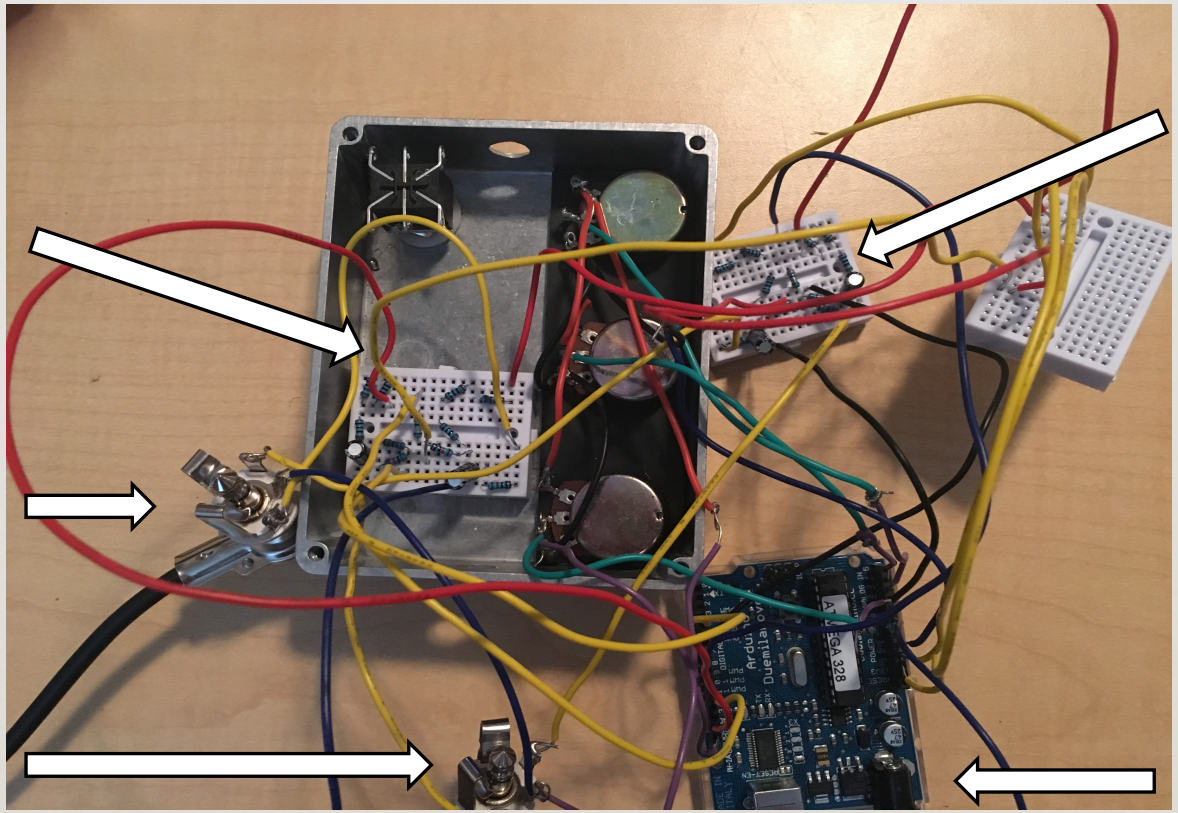
- Assorted stranded and single coil wire
- Resistors (1k, 10k, 1.2k, 1.5k, 390k)
- Capacitors (2.2 uF)
- 3 rotary knobs for potentiometers
- Arduino IDE Software

THE PEDAL:

LEFT CIRCUIT

AUDIO INPUT

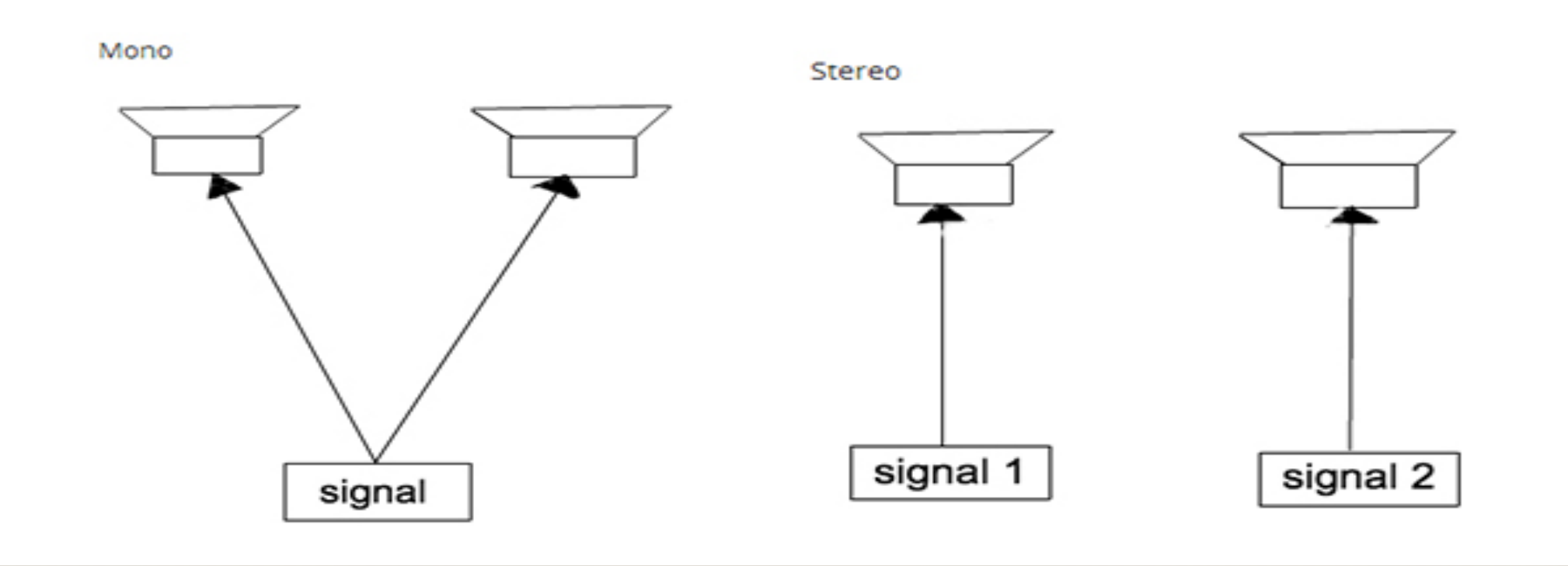
AUDIO OUTPUT



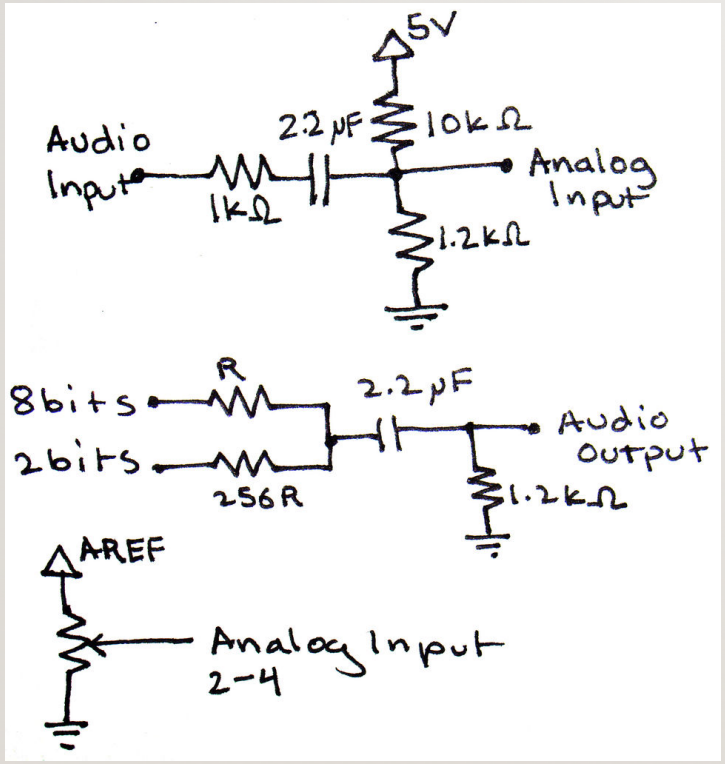
RIGHT CIRCUIT

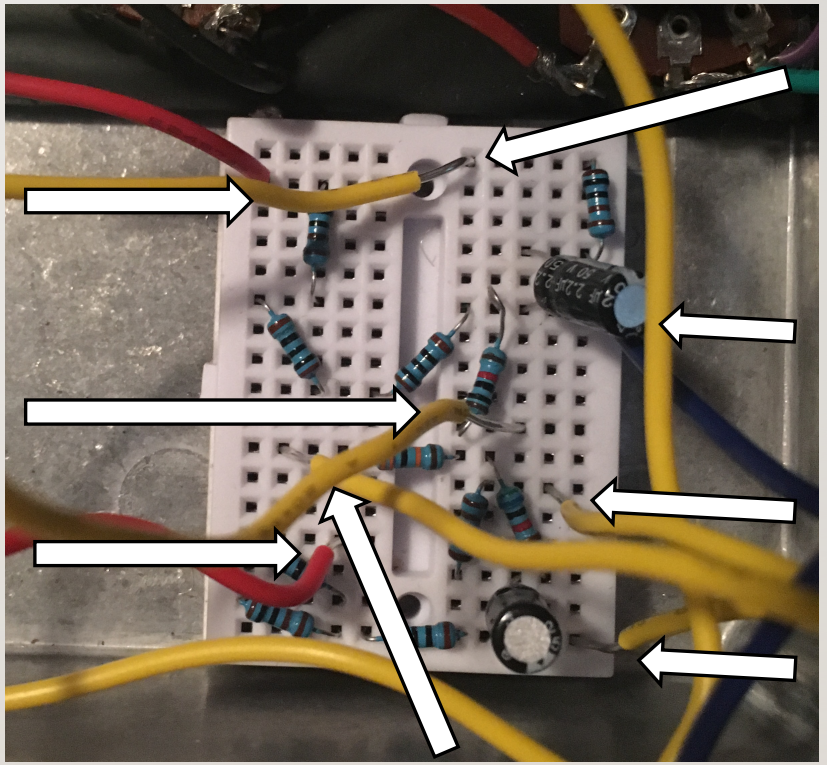
ARDUINO CHIP

MONO VS STEREO:

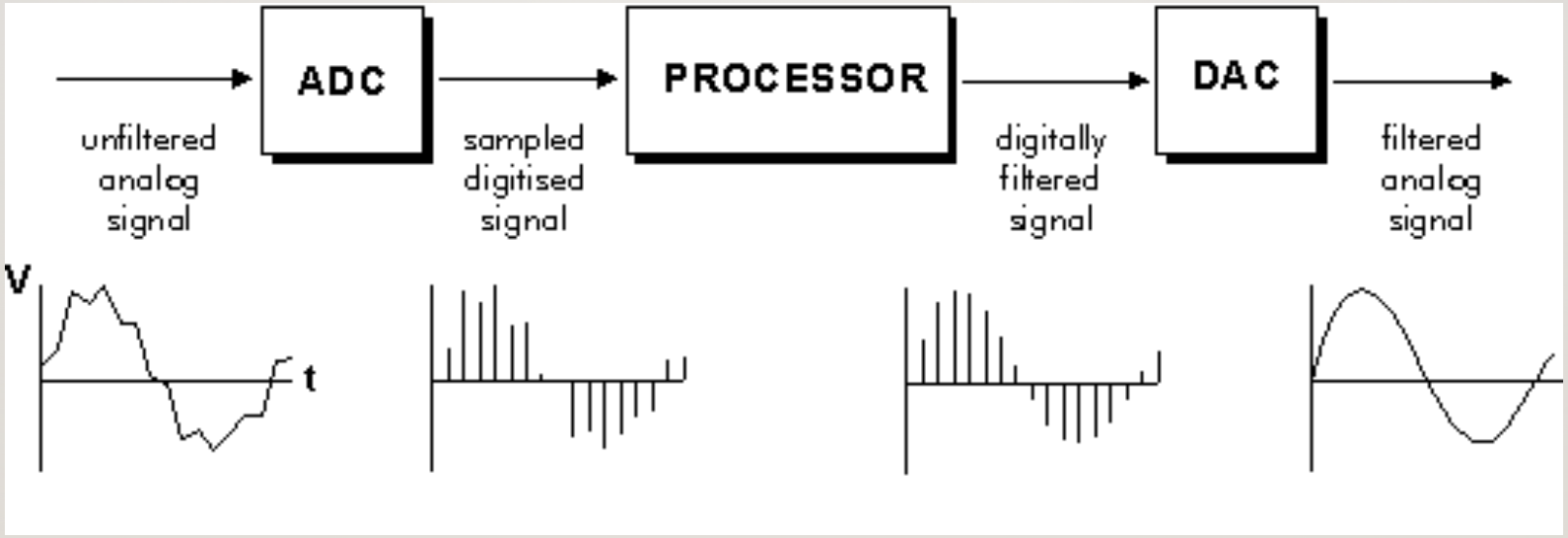


THE CIRCUIT:


5V
Audio Input
2.2 μF
10kΩ
1kΩ
Analog Input
1.2kΩ
8bits
2bits
R
256R
2.2 μF
Audio Output
1.2kΩ
AREF
Analog Input 2-4


G
5V
G
AUDIO INPUT
ANALOG INPUT
8 BITS (DAC)
AUDIO OUTPUT
2 BITS (DAC)

DIGITAL SIGNAL PROCESSING:



THE CODE

SKELETON FILES:

main

dsp.h

dsp.cpp

timers.h

timers.cpp

MAIN:

```
#include "dsp.h"

void setup() {                                  //function in dsp.h and dsp.cpp
  setupIO();
}

void loop() {
  short input = analogRead(left);               //left declared as A0 in dsp.h
  output(left, input);                          //function in dsp.h and dsp.cpp
  output(right, input);                         //function in dsp.h and dsp.cpp
}
```

DSP.H:

```
#define left 0
#define right 1

void setupIO();
void output(int channel, short value);
```

DSP.CPP (P. 1):

```
#include "dsp.h"
#include "timers.h"
#include "Arduino.h"                          //standard types/constants of Arduino

void setupIO() {
// prepare left
  waveformGenerationMode(3, fastPWM);         //from timers.cpp and timers.h
  timerPrescale(3, 1);                        //from timers.cpp and timers.h
  analogWrite(3, 0);

  // prepare right
  waveformGenerationMode(5, fastPWM);         //from timers.cpp and timers.h
  timerPrescale(5, 1);
  analogWrite(5, 0);
```

DSP.CPP (P. 2):

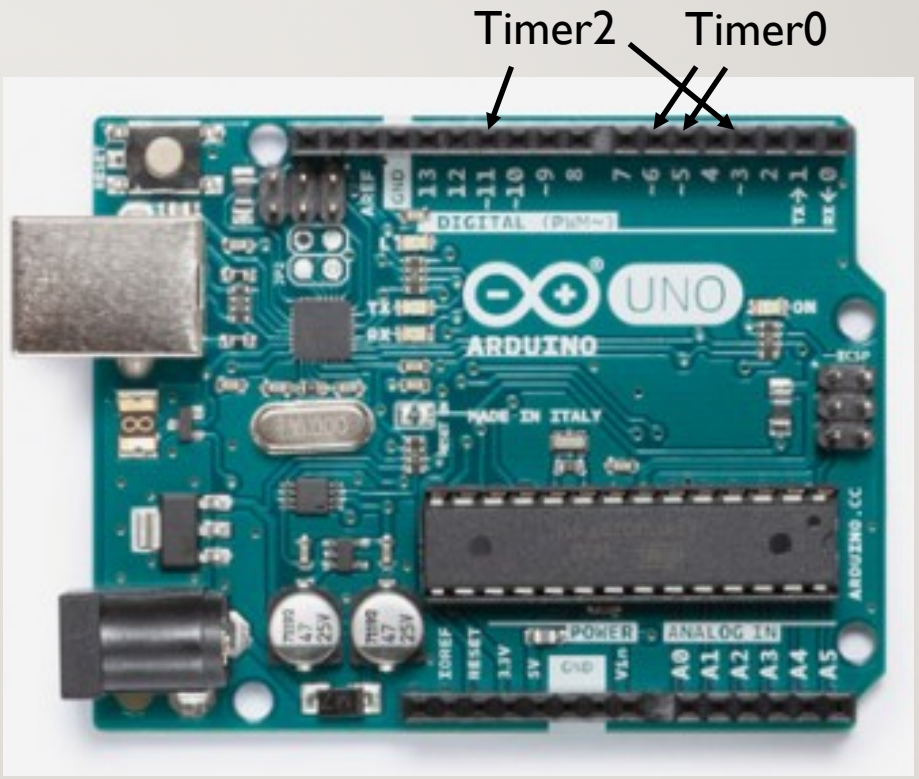
```
.
.
.

void output(int channel, short value) {
  if(channel == left) {
    pwm3 = value >> 2;
    pwm11 = (value & B11) << 6;
  } else if(channel == right) {
    pwm5 = value >> 2;
    pwm6 = (value & B11) << 6;
  }
}
```

TIMERS:

- Hardware inside Arduino chip.
- Timer0, Timer2 (8-bit)
- Timer1, Timer3, Timer4, Timer5 (16-bit)
- Pin controller





TIMERS.H:

```
#define pwm3 OCR2B
#define pwm5 OCR0B
#define pwm6 OCR0A
#define pwm11 OCR2A

int getTimer(int pin);
int getChannel(int pin);

#define phaseCorrect 0
#define fastPWM 1
.
.
.
```

TIMERS.CPP:

```
#include "timers.h"
#include "Arduino.h"

int getTimer(int pin) {
  switch(pin) {
    case 5: case 6: return 0;
}
  return 2;
}

int getChannel(int pin) {
  switch(pin) {
    case 6: case 11: return 0;
  } return 1 ……
```

HEARING THE CODE

Overdrive and Delay Examples

OVERDRIVE:

```
value = 1 + ((float) 1  / (float) 20);
    byte input = analogRead(left);
    input = (input * value);
    output(left, input);
    output(right, input);
```



*Audio sample from T1M The Pearl Overdrive Pedal

DELAY:

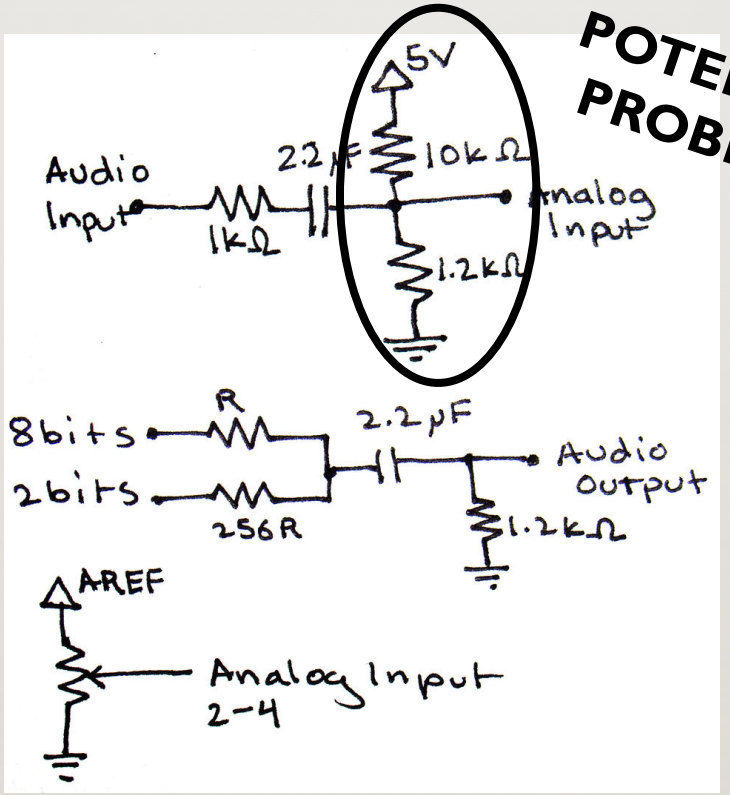
```
byte input = analogRead(left);
int wait;  //points to pot A2
int time; //points to pot A3
int vol = time; //points to pot A4
for(int a = 0; a < time; a++){
  output(left, input);
  output(right, input);
  delay(wait*100); //milliseconds*100
  vol = vol – a; //decrease volume slowly
}
```



*Audio sample from MXR Carbon Copy Delay Pedal

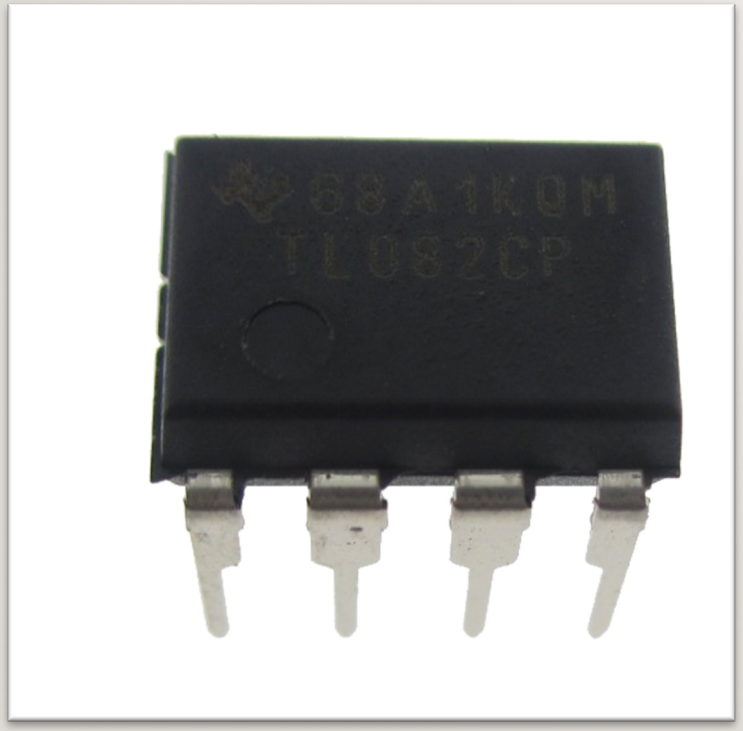
NEXT STEPS

FIX THE CIRCUIT:



POTENTIAL PROBLEM?

ADD OP-AMP AND 9V POWER SUPPLY:





TEST NEW CODE:

```
short input = analogRead(left);

value = input*2;  //also input/2 for lower
octave

output(left, input);

output(right, input);

output(left, value);

output(right, value);
```

- Adds octave of input signal.
- Remove "output(____, input)" to play only octave.

TEST NEW CODE:

```
int val, lr;
byte input = analogRead(left);
if(lr == 0)
  input = input << val;  //shift bits to the left
else
  input = input >> val;  //shift bits to the right
output(left, input);
output(right, input);
```

- Also bit rotation, bit crushing, or other bit manipulation.
- Creates creative, weird, but typically unused effects.

APP FOR CONTROL:

- Tune Pedal for Personal Taste:
  - Currently fairly expensive (H9, $400-$700)
  - Gives power to musician.
  - Flexibility and versatility.
- Multi-effect Load-In:
  - Not done well before (Digitech iStomp)
  - Provides mobility for venue.

SOURCES:

http://www.instructables.com/id/Lo-fi-Arduino-Guitar-Pedal/

https://github.com/telamon/ArduinoDSP

https://playground.arduino.cc/Main/HomePage

https://pdfs.semanticscholar.org/5a37/ae9cd256cbdd4a382a0f007bde6e10e0d66f.pdf

http://www.instructables.com/id/Arduino-Guitar-Pedal/

https://www.cs.indiana.edu/~port/teach/541/sig.proc.html

https://digitech.com/en/products/istomp

https://www.aiseesoft.com/resource/what-is-mono-audio.html

THANK YOU