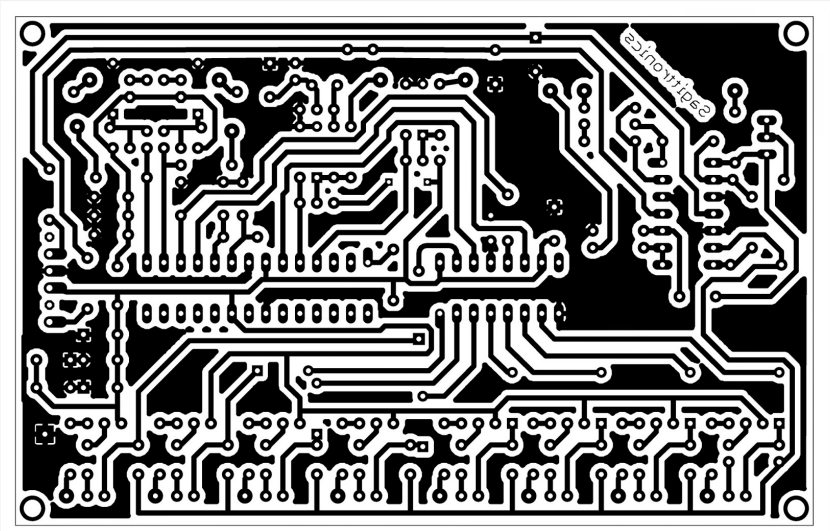
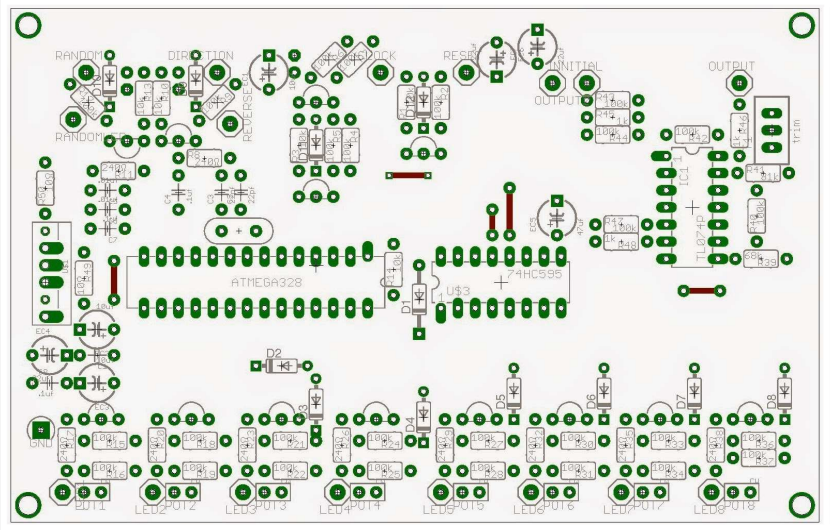
Simple Arduino baby 8 sequencer with reverse and random mode

UPDATE!!!!! 9/18/14 I did some modifications to the PCB and here it is for those who would like to etch it.





UPDATE!!!! 8/27/14 -- I apologize for the sloppy coding below. It was written before I'd learned any standards for C++. From now on, I'll name variables and functions in a way to indicate how they are used (not after my favorite movie shit). I'll probably do a follow up to this sequencer fairly soon but as I've been using it now for many months, I can say that it has performed flawlessly.

I've been getting into coding lately and so I thought it would be cool to come up with a modular project in which I needed to code. This is a very simple sequencer similar to a baby 10 but instead of using a CD4017, I used an arduino programmed atmega chip. I feel like this project was in the spirit of the modular.

Getting the atmega chip to respond both to clock inputs and reset inputs that are triggered at the same time was the most difficult part of this project. Unlike CMOS chips which respond instantaneously to changes at inputs, the atmega must read each input in cycles. The cycle happens very fast but for things like clock signals in which timing is important, getting the arduino to reliable check the clock input at the correct time is difficult.

In order to "catch" the clock inputs rising edge, I initially put and if statement inside of a while statement like....

```
While(clock pin is low)
{
    if (clock pin goes high)
    { x = x +1;
    }
}
```

The above general code worked ok but every once in a while, like when the cycles matched up with the speed of the clock, the sequencer would lose a step. I ultimately solved this by using an attachInterrupt();

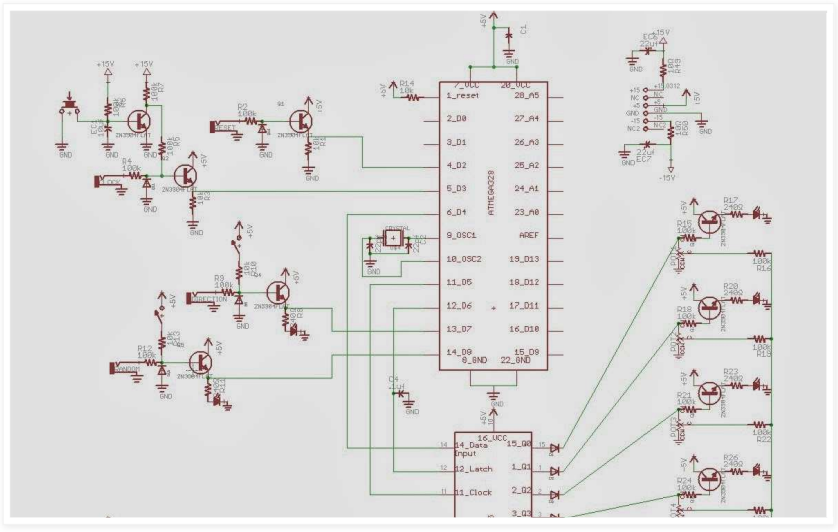
The interrupt will interrupt the main code to execute code specific to a change to one of the two interrupt pins. I set it to interrupt on the rising edge. I did the same for the external reset input. unfortunately when the external reset and the clock input go high at the same time (which tends to happen often), it is somewhat unpredictable which code will take precedent,

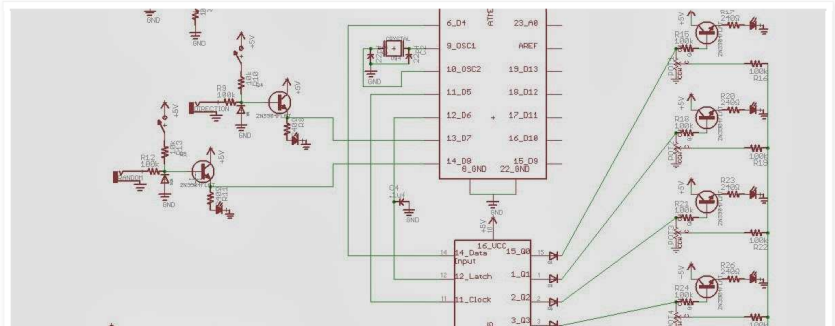
The interrupt will interrupt the main code to execute code specific to a change to one of the two interrupt pins. I set it to interrupt on the rising edge. I did the same for the external reset input. unfortunately when the external reset and the clock input go high at the same time (which tends to happen often), it is somewhat unpredictable which code will take precedent, either the clock code or the reset code, and for my original code, I would get different results depending on the order. I resolved that by rewriting the code so that it didn't matter which happened first.

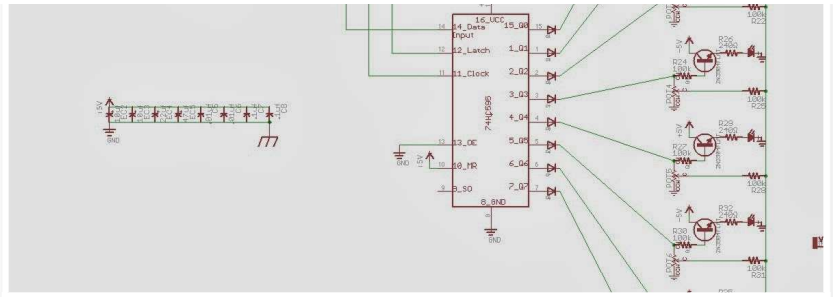
```
int dataPin = 4;
  //data pin for shift register
int latchPin = 6;
  //latch pin for shift register
int clockPin = 5;
  //clock pin for shift register or "shiftout"
int togglePin = 7;
  //toggle pin for direction read
int randomEnable = 8;
  //source enable pin
volatile byte count;
  //4 digit variable
byte countArray[8];
  //4 digit array with 8 spaces
boolean reset = 0;
  //boolean reset toggle variable
boolean toggleDirection = 0;
  //boolean direction toggle variable
volatile int j = 0;
  //variable for counting through count array
int k = 0;
int l = 0;
  //for making random non repeating
volatile int x = 1;
  //variable for counting
volatile boolean bishop = 0;
  //create a variable that can be used for clock
boolean hudson = 0;
  //create variable used in clock function
boolean hicks = 0;

void setup(){
  pinMode(dataPin, OUTPUT);
    //set shift register pin to output
  pinMode(latchPin, OUTPUT);
    //set latch pin to output
  pinMode(clockPin, OUTPUT);
    //set clock pin to output
  pinMode(togglePin, INPUT);
    //set reset pin to input
  pinMode(randomEnable, INPUT);
    //set sourceEnable pin to input
  attachInterrupt(0, sigourney, RISING);
    //when pin 2 changes (rising edge) move to "sigourney" reset loop
  attachInterrupt(1, alienClock, RISING);
    //when pin 3 changes (rishing edge) move to "alienClock" clock loop
  randomSeed(analogRead(0));
     //read analog pin 0 for start of pseudo random sequence

  countArray[0] = 1;   //0000 0001, step 0
  countArray[1] = 2;   //0000 0010, step 1
  countArray[2] = 4;   //0000 0100, step 2
  countArray[3] = 8;   //0000 1000, step 3
  countArray[4] = 16;  //0001 0000, step 4
  countArray[5] = 32;  //0010 0000, step 5
  countArray[6] = 64;  //0100 0000, step 6
  countArray[7] = 128; //1000 0000, step 7
}
void loop(){

  if(bishop != hudson){
    //limites access to this loop, once per external clock cycle until random is enabled
    hicks = digitalRead(randomEnable);
    //check to see if random mode is enabled
    if(hicks == 1){
      //if random mode is enabled then do the following
      l = k;
      k = j;
      //creates a history.  k is j last cycle and l is j from the cycle before
      do {
        j = random(8);
      }
      while (j == k || j == l);
      //generate random value for j and make sure it does not match previous 2 value
    }
  else {
    j = j + x;
    // add 1 to j
    }
```

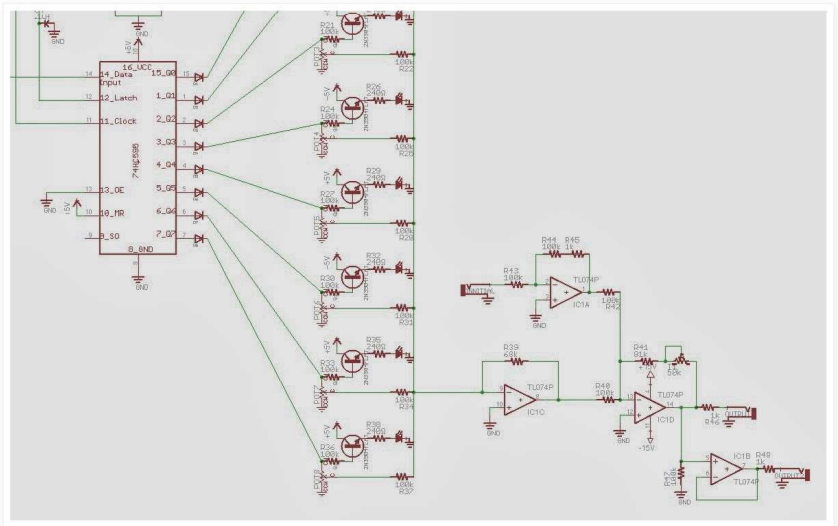
```
  }
  else {
   j = j + x;
    // add 1 to j
    }
  hudson = !hudson;
  //prevents function from returning to this loop before another cycle of the external clock
  }
  //toggle hudson
  if(j > 7) j = 0;
    //when j reachers 8, go to startpoint
  if(j < 0) j = 7;
    //when j is less than 0 go to startpoint

}

void sigourney(){
  j = 0;
  count = countArray[j];
    // set count eaqual to present state of starting point
  digitalWrite(latchPin, 0);
    //ground latchPin and hold low for as long as you are transmitting
  shiftOut(dataPin, clockPin, MSBFIRST, count);
    //serial output count
  digitalWrite(latchPin, 1);
    //bring latch pin high to end transmission
}
void alienClock(){
  count = countArray[j];
    // set count eaqual to present state of j
  digitalWrite(latchPin, 0);
    //ground latchPin and hold low for as long as you are transmitting
  shiftOut(dataPin, clockPin, MSBFIRST, count);
    //serial output count
  digitalWrite(latchPin, 1);
    //bring latch pin high to end transmission
  bishop = !bishop;
  toggleDirection = digitalRead(togglePin);
   if(toggleDirection == 1){
  x = -1;
   }
    else{
  x = 1;
   }
}
```

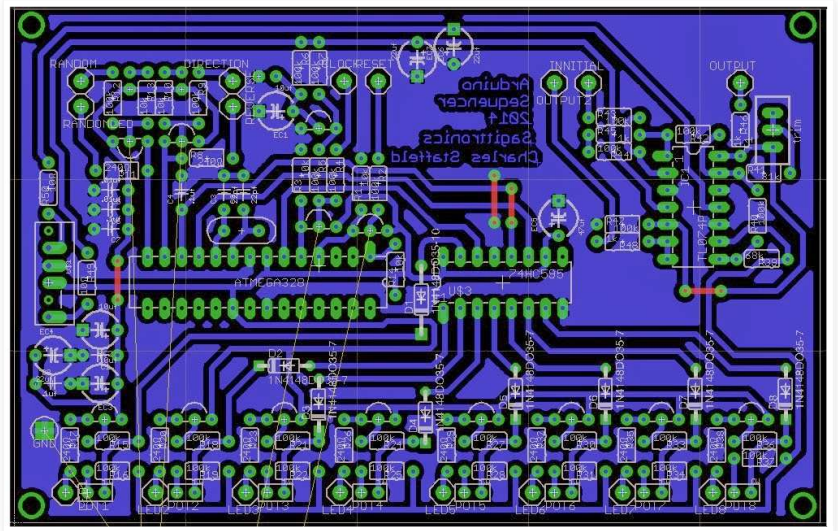
The schematic is large and hard to read as a single pictures so I broke it down to three.

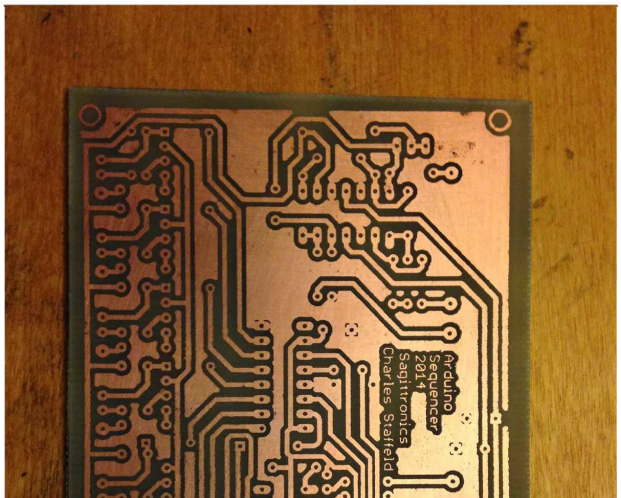


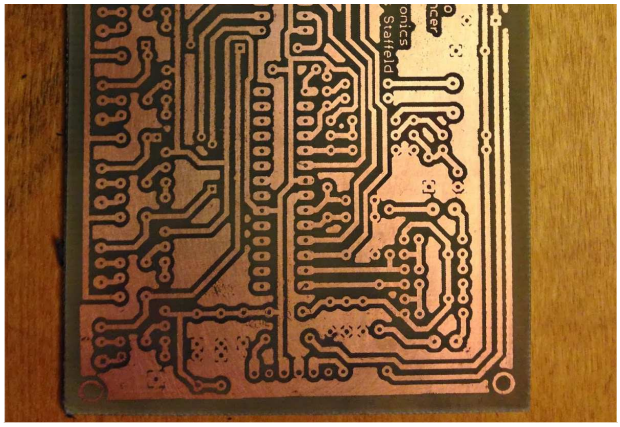


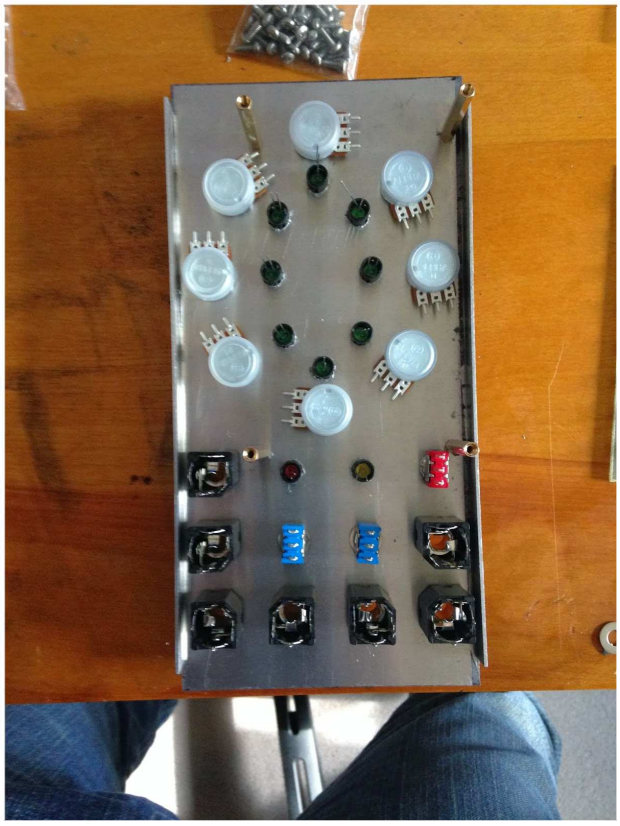


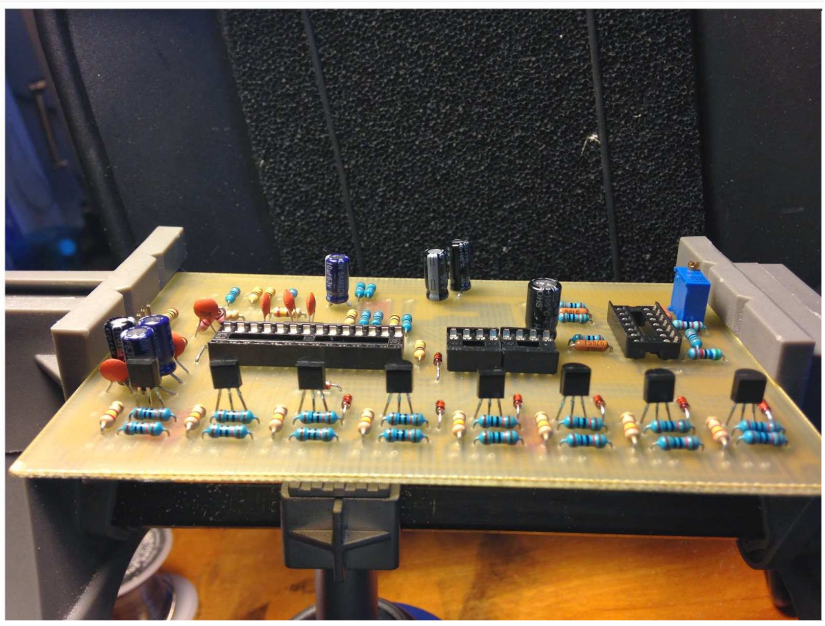


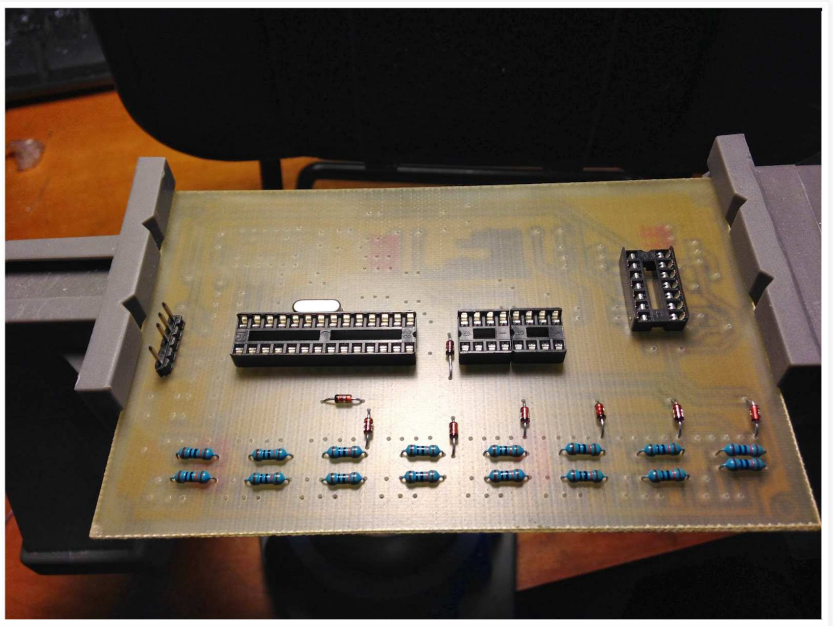

RANDOM
DIRECTION
CLOCKRESET
INNITIAL
OUTPUT
RANDOMISED
Arduino
Sequencer
2014
Sagittronics
Charles Satfield
ATMEGA328P
74HC595

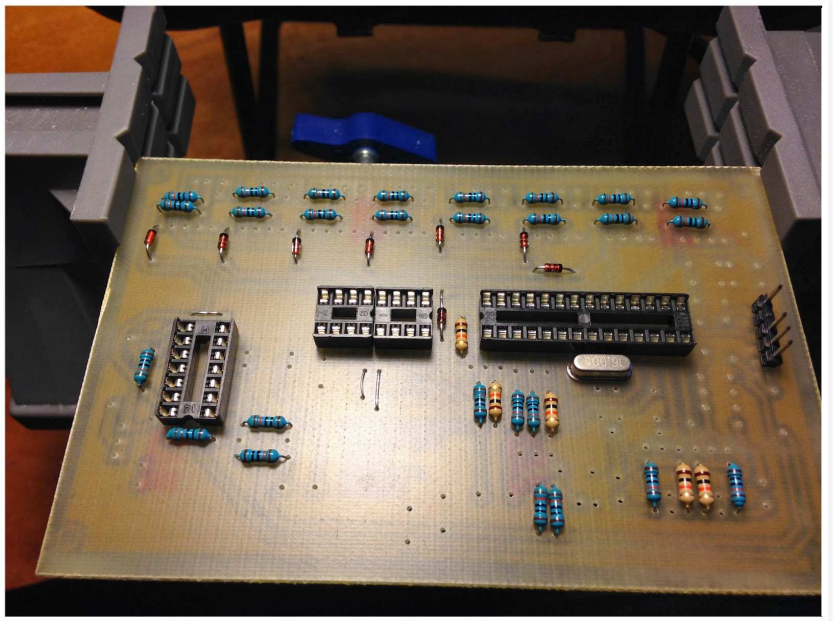

Arduino
Sequencer
2014
Sagittronics
Charles Satfield

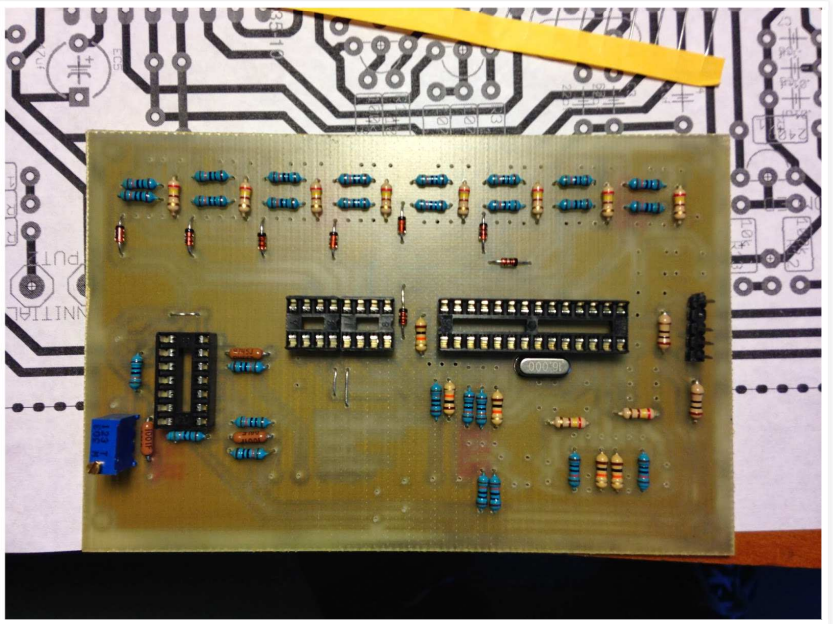

onics
Staffeld



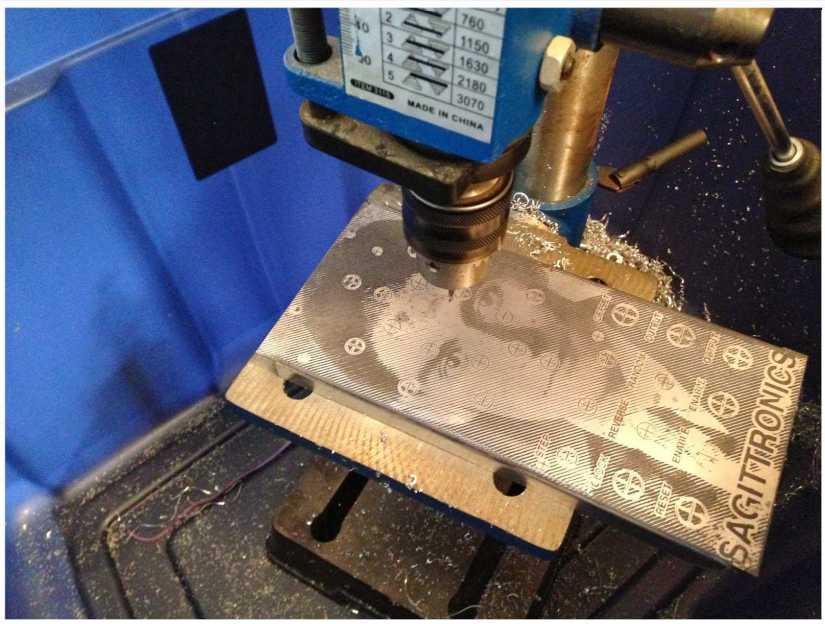




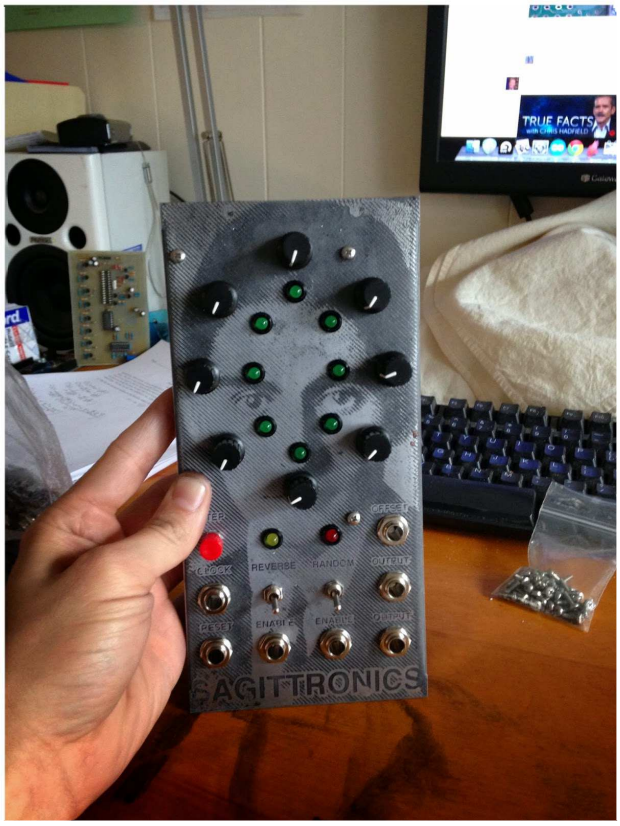





STEP
OFFSET
CLOCK
REVERSE
RANDOM
OUTPUT
RESET
ENABLE
ENABLE
OUTPUT
SAGITTRONICS


760
1150
1630
2180
3070
MADE IN CHINA
SAGITTRONICS


STEP
OFFSET
CLOCK
REVERSE
RANDOM
OUTPUT
RESET
ENABLE
ENABLE
OUTPUT


TRUE FACTS
SAGITTRONICS


STEP
OFFSET

...

Posted by Charlie at 11:03 PM

Labels: 8 step, analog, arduino, atmega, baby 10, cd4017, charlie slick, diy, dotcom, etching, how to, modular, moog, motm, sagittronics, schematic, sequencer, step, synthesizer, synthesizers.com

6 comments:

MIKZED53 September 18, 2014 at 10:14 AM

HI CHARLIES , PLEASE CAN YOU ADD PCB in black and white of your cv sequencer arduino based ,great works great blog,,,, merci ,,, friendly,,, mik

Reply

Replies

Charlie September 18, 2014 at 11:54 AM

done!

condor May 6, 2015 at 1:47 AM

Hello Charlie, will this work in eurorack +/-12V ? If not with what modifications. Thanks.

Reply

Unknown October 13, 2015 at 7:37 AM

Hello, what is the size of the PCB? Thanks.

Reply

sergio_ June 1, 2016 at 4:19 AM

Hello! The big request to share the hex file..

Reply

snaper August 15, 2016 at 10:28 PM

Oh Gosh, just saw this.
Amazing!
Is it possible to mod this somehow to 16 steps?

snaper August 15, 2016 at 10:28 PM

Oh Gosh, just saw this.
Amazing!
Is it possible to mod this somehow to 16 steps?
Is there any chance to get a complete documentation?

Reply

Monday, December 16, 2013

quad ADSR simplified yu synth ADSR.

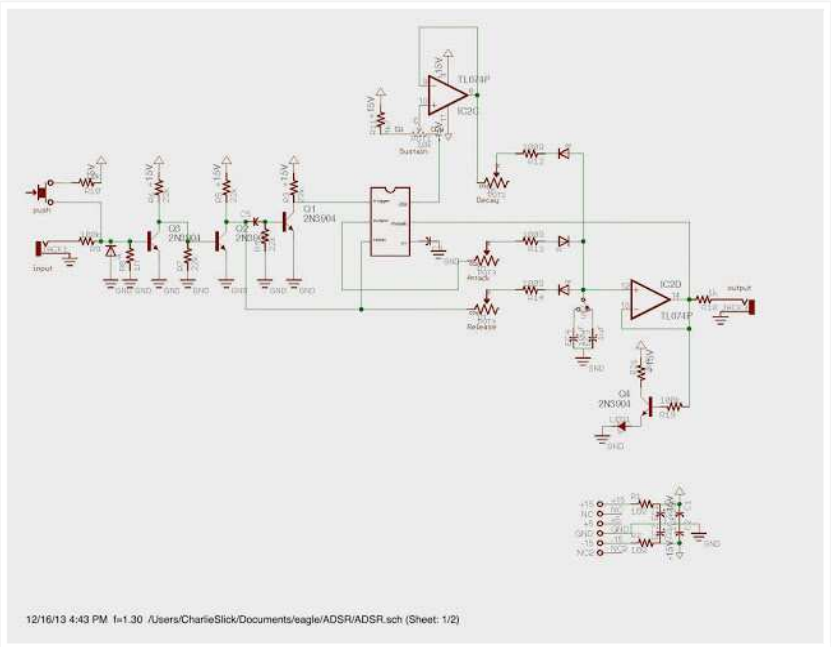
I am an aspiring electrical engineer. Circuit design inspires me in the same way that songwriting once inspired me. In electronics as in songwriting, one often imitates until he can create on his own. When modifying another person's circuit, it's hard to say exactly when one can claim a circuit. How much needs to be different? What kind of changes are considered more than arbitrary and therefore justify claiming ownership (my circuit)?

A majority of the circuits I've worked on for my synth started as either yu synth designs or MFOS designs. The designers of these circuits have far more electronics knowledge than myself and clearly execute a higher level of intention in their circuits. Though I've learned a lot since I began my modular project, I still often find myself in the "shit, lets try this resistor" process. The designs available on these sites are sometimes built upon older designs in which the newer design only varies in peripherals (like input/output buffer impedance, trigger method, resistor value, LED indicator), keeping the core function identical. In all cases, these designs are based on simple electronic processes which were discovered by someone else years ago-- a fact to which these sites openly acknowledge. As participants, we cannot ignore this chain of influence.

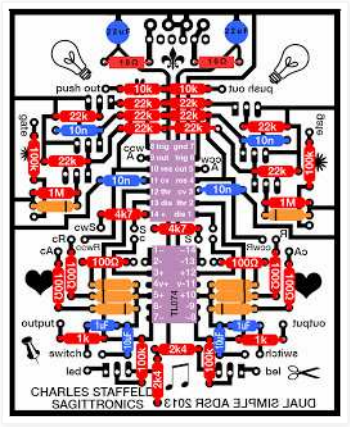
I take the postmodern view when it comes to circuit design. No one owns the function in which these circuits are designed to perform. No one owns the function of a filter or the function of an oscillator. These simple electronic functions can be performed in a variety of ways and as we get further away from the tree trunk of an idea, the branches become thinner and closer together. The things I change about these circuits--however arbitrary--make them more suited to my purpose and therefore are as "mine" as anyone's claim to a recipe for homemade nachos.

That being said, I would never directly clone a person's circuit or disregard the chain of influence. I don't believe that is in the spirit of sharing knowledge or creative engineering. In circuit design as in songwriting, my goal is to participate, not "steal" (whatever that word even means anymore).

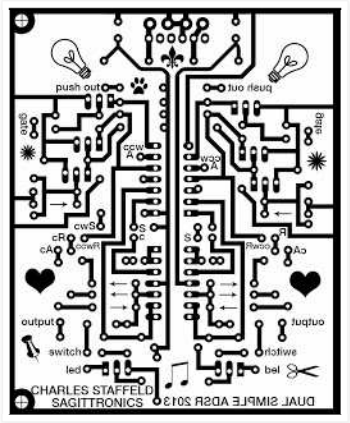
That diatribe was meant to address the fact that "my" quad ADSR circuit is really nothing more than a simplified version of the yu synth ADSR. By reducing the number of functions it performed and therefore reducing the number of components, I was able to squeeze 2 simple circuits onto a PCB using only 2 chips (quad op amp and LM556). I have no need for an inverted ADSR output because most of the cv inputs on my synth were designed with attenuverters. This ADSR is a sort of a simpler compromise between the two yu synth designs.

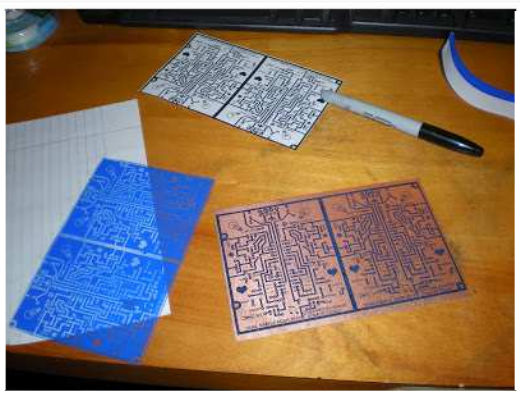


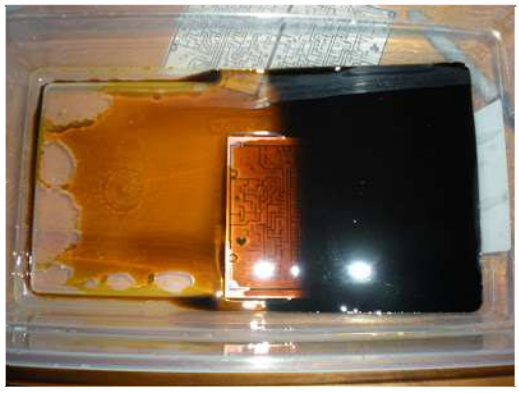
This is likely one of the last PCB I will layout in photoshop. The circuit repeats 4 times and so it made sense to do it in photoshop as eagle cad will not allow you to simply copy a section of your PCB layout and stamp it around.

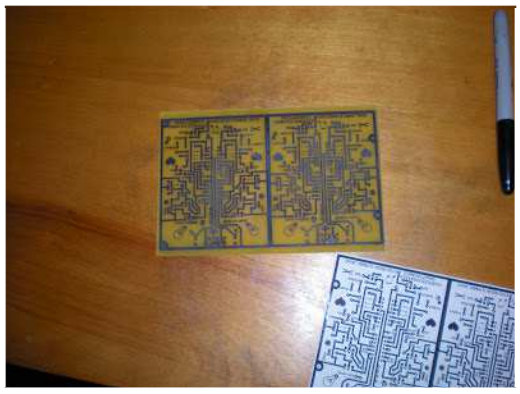


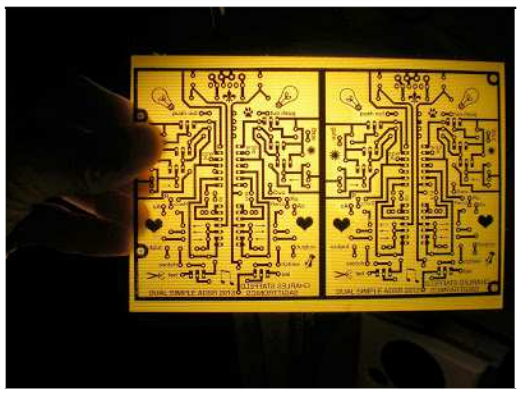
The transistors are 3904's and the collector is the square pad. ha ha. except the very bottom one that controls the led. it's backwards and so the emitter is the square pad. That kind of mistake is easier to make in photoshop. the arrows point to the negative side of the diodes. I doubt anyone will try and build this based on my minimal instruction but posted the PCB just in case. comment any questions, I respond pretty fast.

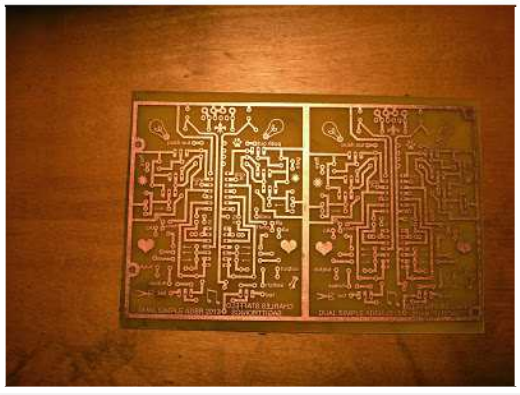




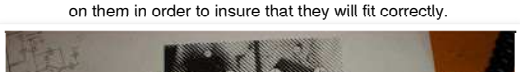






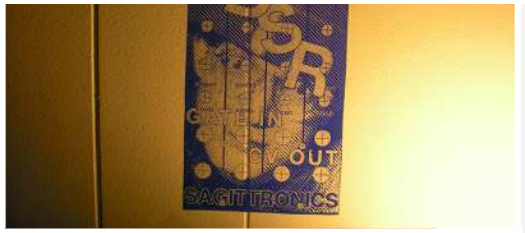


since I began building modules, I find myself planning more. I like to print version of the panels and lay the components out on them in order to insure that they will fit correctly.



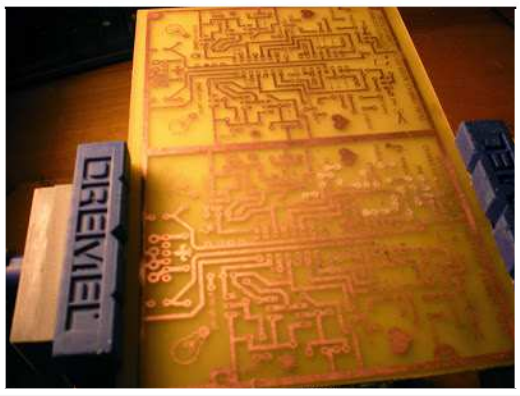
since I began building modules, I find myself planning more. I like to print version of the panels and lay the components out on them in order to insure that they will fit correctly.

a little DEVO joke for you.


ADSR
GATE IN
CV OUT
SAGITTRONICS


ADSR
SAGITTRONICS

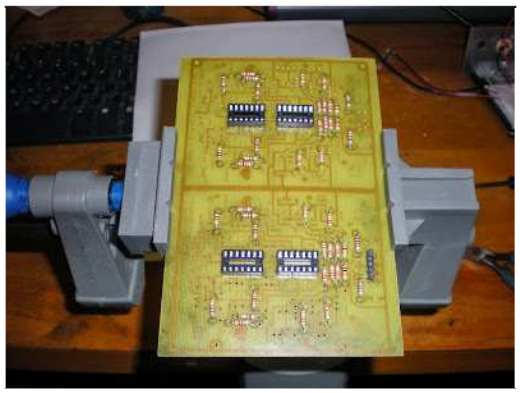

ADSR

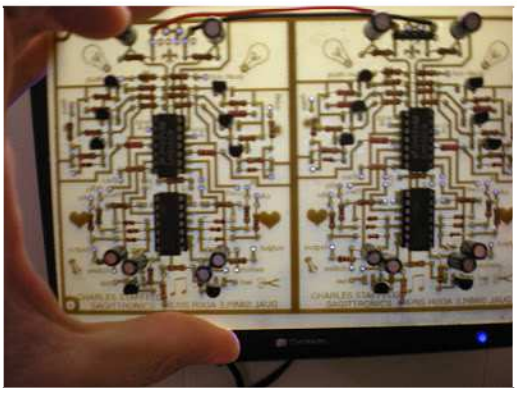

DREMEL

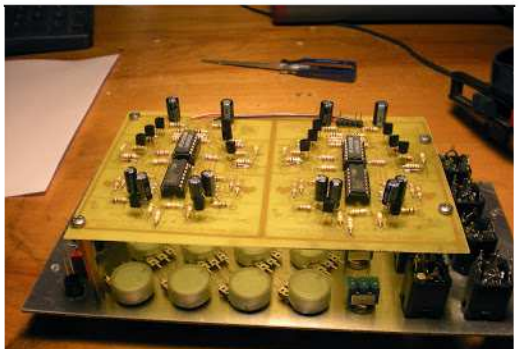

ADSR
GATE IN
CV OUT
SAGITTRONICS


SAGITTRONICS

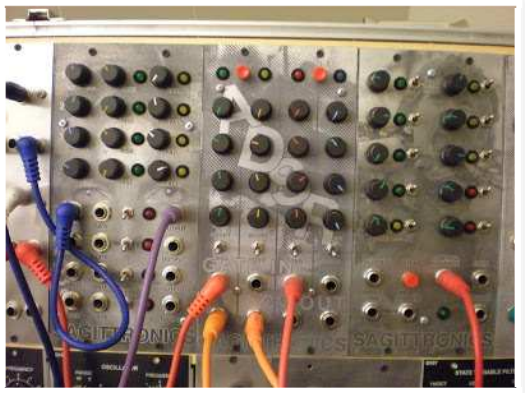

SAGITTRONICS







I managed to take a million pictures of my build, except for one showing all the panel wires.



...