Mauro Ricchiuti s2462168

VGA hybrid Video Synth

The initial idea for this project was to build a VGA Audio/Video synthesizer based on Arduino. The inspiration came from the project CHA/V from Jonas Bers[1] in which he built a VGA video synth based on CD 40106. In that project, the artist used the Hex Schmitt Trigger as a source to drive RGB colors of a VGA Signal generator. My aim was to recreate something similar using the PWM of the Arduino as source. Unfortunately changing the PWM frequency of Arduino was not as simple as I thought, so I decided to do a hybrid analog/digital video synth, in which the RGB colors can be controlled either by Arduino or by a 555 timer. In the initial plan, the idea was also to control the Hsync and Vsync of the VGA signal, but that was really hard to do and I end up not doing it.

In this documentation I will explain first the Basic prototype, which uses just Arduino PWM, and then the more advanced prototype that has also 2 555 timers.

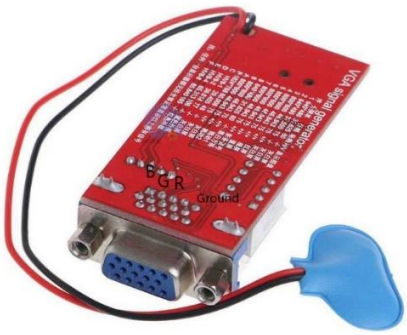
Basic Prototype

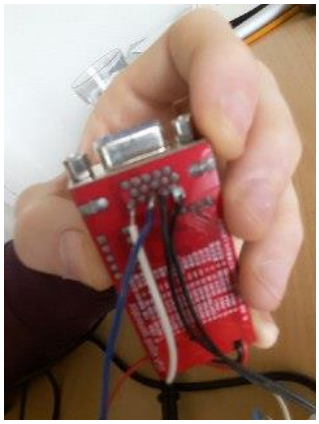
Components:

- VGA Signal generator[2]
- Arduino Uno (other board work aswell)
- 3x 10KOhm linear potentiometers
- 1x push button

The aim of this project is to take advantage of the Hsync and Vsync signals generated by the VGA Signal generator, and change RGB colors value of the same. In this way, we will already have an image on the screen created by the signal generator, and we are going just to change the color values. The colors value will be controlled by PWM of Arduino outputs. RGB values in a VGA signal in fact, accept analog values between 0 and +0.7V. A fake analog value can be created using the PWM outputs of an Arduino, and changing the duty cycle we are able to control the voltage we are sending in. Even if the PWM frequency is fixed in fact, we can change the duty cycle of it, changing the average voltage. For this project, I have to admit that I did not really pay attention to the technical part, but was all a matter of trials and errors. For example, even if the RGB values of a VGA signal should be between 0V and +0.7V, I got very interesting effect in sending values between 0 and 5V, which is the PWM voltage out of the Arduino.

First, we have to solder 4 wires from the Red, Green, Blue and Ground channels of the signal generator. (In the picture below on the right, white = ground, the others are R, G, B).

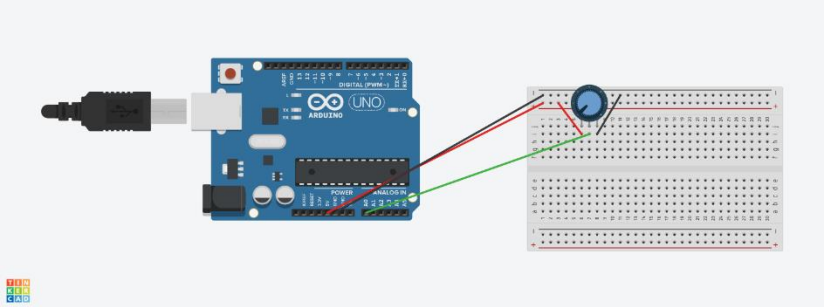




Then we are going to ground together the Arduino and the signal generator to the breadboard.

We will also send the 5V from the Arduino to the breadboard, connecting the 5V pin of Arduino with the (+) pin of the breadboard. In this way, we have a breadboard that has a 5V voltage running through, and both the signal generator and the Arduino grounded together.

In order to control the colors, we are going to use 3x 10 KOhm linear potentiometers. A potentiometer is a variable resistor, in this case his value ranges from 0 to 10 KOhm. We wire the potentiometer to the Arduino board as in the picture below. In this way, we are able to read in the A0 pin of the board the analog value of the resistor, and this will be a value between 0 and 1023.



Now we are ready to see the first colors change in the video. We should wire the pin 9 of the Arduino with the Red wire of the signal generator and run the following code in Arduino. Then attach a battery to the signal generator and connect it to a VGA screen.

```
int port = 9;
int analogPin = 0;
int val = 0;

void setup() {
  pinMode(port, OUTPUT);
}

void loop() {
  val = map(analogRead(analogPin0), 0, 1023, 0, 255);
  analogWrite(port, val);
}
```

Rotating the potentiometer, we will change the PWM voltage output at the pin 9, in this way we will control the red stripes that will appear in the screen, we can make the whole screen turn red, or have very small red lines moving vertically. The speed of which the lines move on the screen is dictated by the PWM frequency of the pin 9 in Arduino, that, for an Arduino Uno is 490Hz. My initial plan was to change this frequency, in order to change the lines speed, but I was not able to do so in the time I had for this project.

If we want to control all the Red, Green and Blue values of the signal, we have to do the same process 3 times. I also added push button which randomize the values for the Red channel when it is pressed.

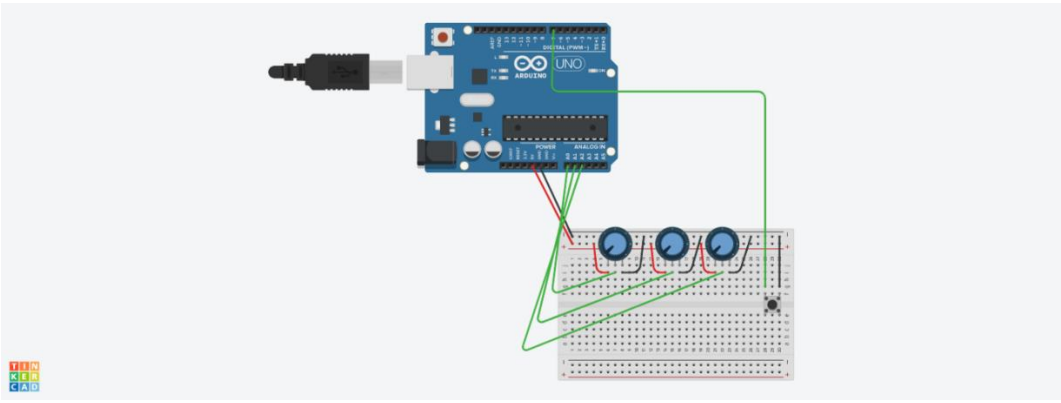


Figure 1: Arduino Wiring of the 3 Potentiometers

If we follow the schematics in the picture above, we connect the pins 9, 10 and 5 to the R, G and B wires of the signal generator and we upload the following code, we will be able to control the 3 colors stripes and to randomize the red value when pressing the button.

```
int port = 9;
int port1 = 10;
int port2 = 5;// LED connected to digital pin 9
int analogPin = 0;
int analogPin1 = 1;
int analogPin2 = 2;// potentiometer connected to analog pin 3
const int buttonPin = 7;

int val = 0;
int val1 = 0;// variable to store the read value
int val2 = 0;
int buttonState = 0;
void setup() {

  pinMode(port, OUTPUT);
  pinMode(port1, OUTPUT);// sets the pin as output
  pinMode(port2, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  Serial.print(buttonState);
  if (buttonState == LOW) {
    val = random(0, 255);
  } else {
    val = map(analogRead(analogPin), 0, 1023, 0, 255);
  }
  val1 = map(analogRead(analogPin1), 0, 1023, 0, 255);// read the input pin
  val2 = map(analogRead(analogPin2), 0, 1023, 0, 255);
  analogWrite(port, val);
  analogWrite(port1, val1);// analogRead values go from 0 to 1023, analogWrite values from 0 to 255
  analogWrite(port2, val2);
}
```

I did connect the Blue wire from the signal generator to the pin 5 of the Arduino because this pin has a different PWM frequency than pins 9 and 10. His frequency is 980 Hz in fact, instead of 490 Hz. In this way, the speed with which the lines move vertically will be different from Blue to Red and Green, creating interesting patterns.

This is the basic prototype of my synth, in which, playing with the 3 potentiometers and the button, we can already have interesting patterns coming out. It is also possible to play with the buttons on the signal generator. These buttons, change the pattern shown in the screen, and it is the base of the video synth. The best effects are obtained when this pattern is a black screen, since in that way we have full control of the colors, but using different initial patterns, we can obtain very interesting and aesthetic effects. The patterns which can be obtained with this basic configuration are constrained by the PWM frequency of Arduino and are just composed of lines of different colors moving upwards and overlapping with each other. They can be really aesthetic, but they get boring after a while.

Advanced Prototype

After I built the basic prototype, I wanted to mess around with the Hsync and Vsync channels of the VGA Signal generator, in order to control the vertical movement of the colored lines. To achieve that, I built a small square waves generator using a 555 timer. A 555 timer in fact, is able to generate a similar signal that the PWM output of the Arduino does, a square wave. I choose to use this component because I did have lots of them, and because it is relatively easy to generate and control the square wave output, and to make it work, you just need a potentiometer, a capacitor and a resistor. The square wave output frequency and duty cycle, will be a function of the potentiometer's resistance, the resistor and the capacitor.

I did not manage to control the Hsync and Vsync, so I choose to use the same circuit to control the Red and Green channels of the VGA Signal Generator. I did that using a Toggle Switch, and switching the

signal sent to the VGA Signal Generator Channel between the PWM coming out the Arduino, and a signal made out of a combination of the PWM and the output of the 555 timer.

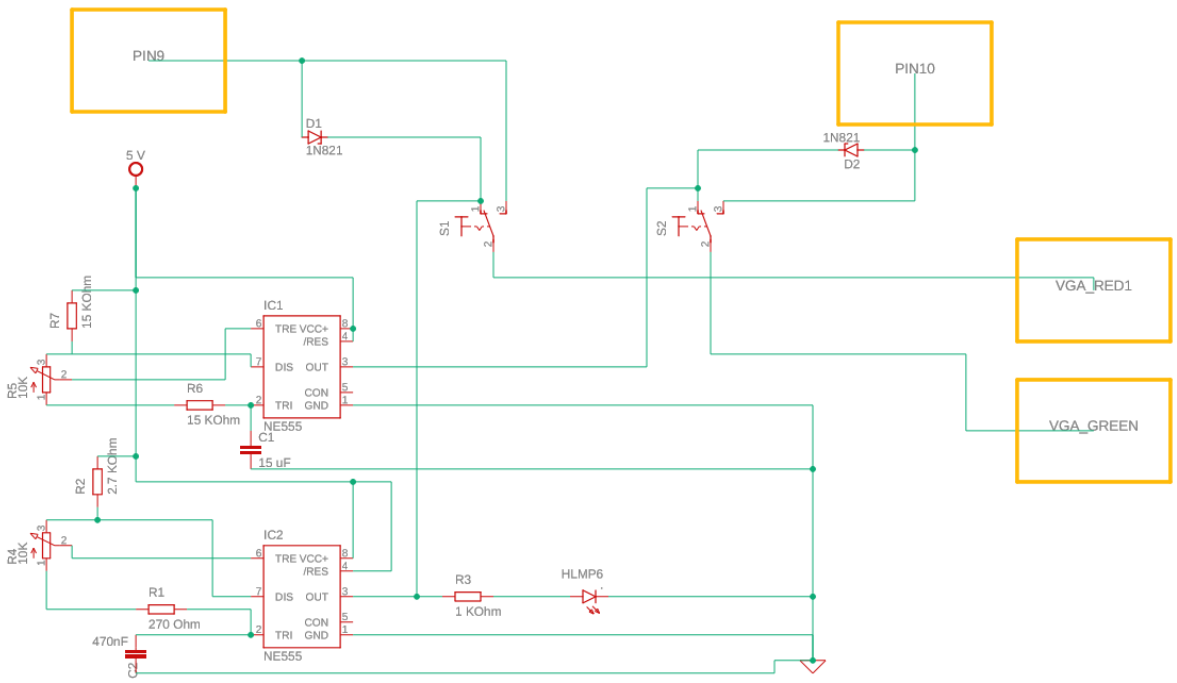
I have to admin that for this process I did not follow many technical rules, but I just achieve a desirable aesthetic with trials and errors. So, I have to say, that probably the electrical calculations for the circuit that I drew are not right, but with this configuration I did manage to get very interesting visual effects, and I was pretty satisfied with them.

For the advanced prototype you need, in addition to the Basic Prototype components, the following ones:

- 2x NE555P Timer
- 2x 10KOhm Linear Potentiometers
- 1x 470 nF Capacitor
- 1x 15uF Capacitor
- 1x LED
- 2x Diodes (I had some called A7)
- 2x 15 KOhm resistor
- 1x 270 Ohm resistor
- 1x 1.7 KOhm
- 1x 1 KOhm
- 2x 3-Pins Toggle Switch

With the following components, I did follow schematics on how to use the 555 timers in Astable mode. Since I did not have other capacitors, I did not ground the pin 5 through a capacitor, as every schematic was showing, but I still did obtain some signals out.

The following picture is the schematics I drew for the Advanced Prototype.



In the schematics above, the modules ‘PIN9’ and ‘PIN10’ are the output pins of the Arduino, while ‘VGA_RED’ and ‘VGA_GREEN’ are the Red and Green channels of the VGA Signal Generator. The

rest of the configuration, is the same as the Basic Prototype version. The advances in this version are that, we can control R and G channels of the signal generator with the switches S1 and S2, choosing to send the same signals as for the Basic Prototype (PWM from Arduino) or a composite signal made out of the PWM and the output of the 555 timers.

The frequency produced by the two 555 timers circuit should be, for the Red color in the range [132 – 948 Hz), while for the Green color in the range [1.4 – 2.2 Hz]. Since my configuration for the 555 timers is slightly different from the ones find in internet, I am not sure these frequencies are actually right. I get to these values after different trials and errors with different resistor's value. When I did find the most visually aesthetic resistor's values, I stopped trying. Leaving the resistor's values unchanged and decreasing the capacitor's value will result in an increase of output frequency.

The initial goal was to reach a frequency of 30 KHz or a multiple of it, because Hsync and Vsync are around 30 KHz and 60 KHz respectively. Since I did not have capacitors and potentiometers to reach those frequencies, I just used the ones that I had and tried different configurations in order to achieve something 'beautiful'.

This project is for sure not a definitive video synthesizer. It needs still a lot of technical adjustments, but it opens a lot of new ways of experimenting with the VGA medium. Since I had never worked with VGA signals, for me was really interesting to understand them and try to control them. This project could be so, a starting point for future developments.

References

[1] https://jonasbers.com/chav/

[2] https://www.ebay.com/p/1531455367?iid=401172500324&rt=nc

[3] http://www.ti.com/lit/ds/symlink/ne555.pdf

[4] http://www.ohmslawcalculator.com/555-astable-calculator