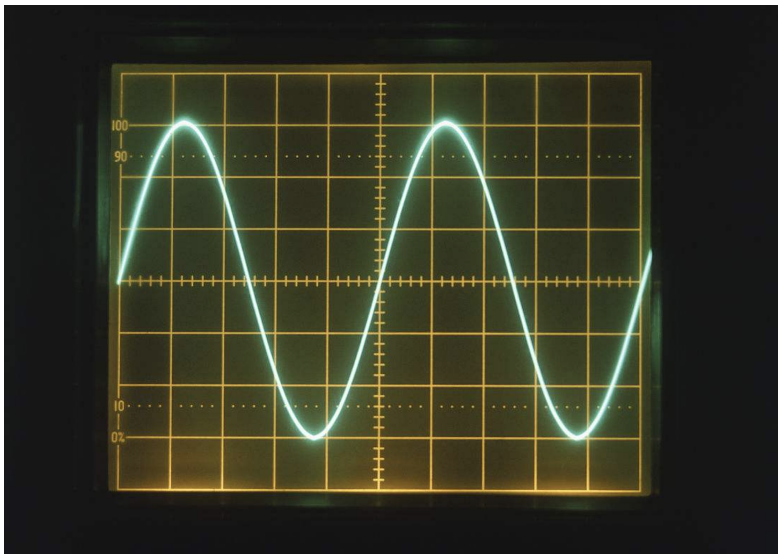
Electric Circuits (Fall 2017)

Project: Arduino Oscilloscope

Midterm Report



Song Xiaoyu and Qiu Jiacong

TA: LI Yike

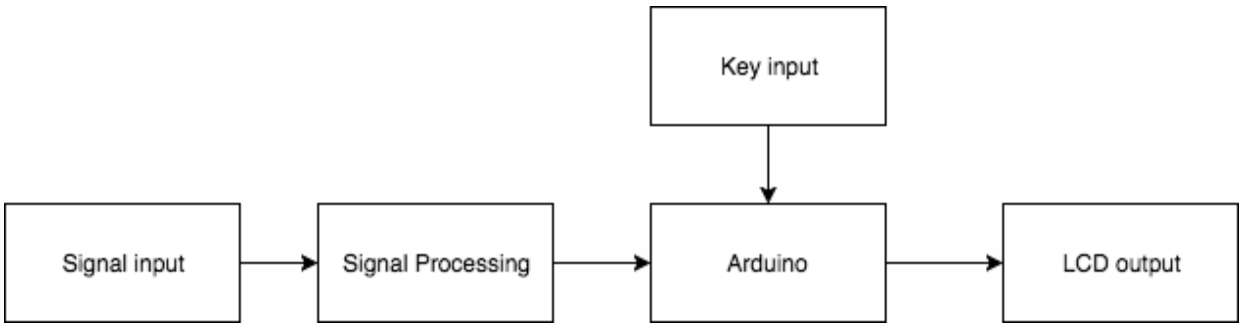
December 28, 2017

Introduction

The goal of this project is to build an oscilloscope based on Arduino, that:

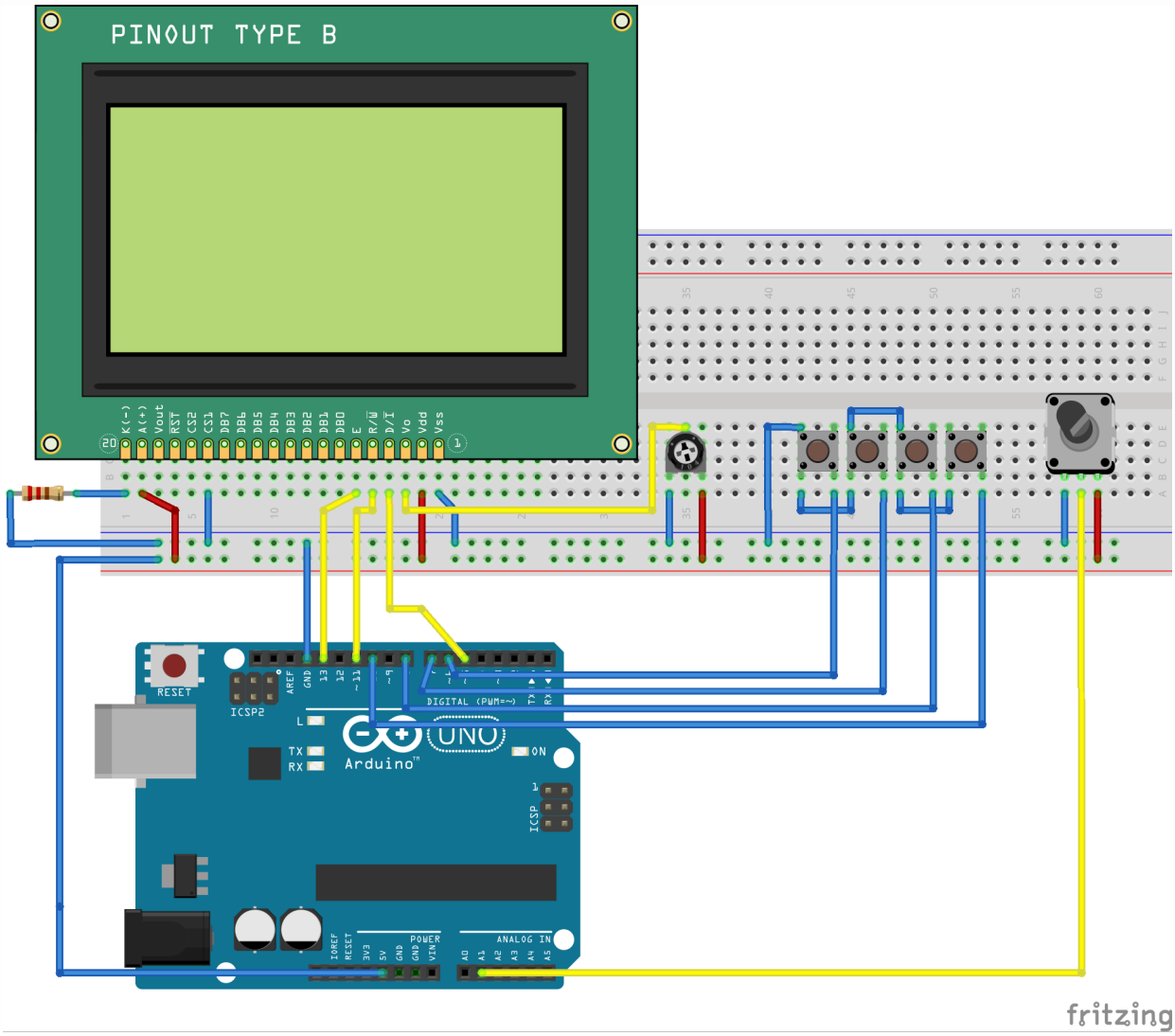
1. Display voltage signals on LCD (duh)
2. Rising edge triggering
3. Vpp measuring
4. Cursor mode
5. Stop mode
6. Looks cool

Block Schematic



Schematic

Since we are using Arduino (and a cheap, unbranded, manual-less LCD module) for our project, it's impossible to run a simulation of the whole project. Actually, we decided to go straight to breadboarding, get our hands dirty and have tons of fun. So instead of some simulation results, we will present an overview of our project and follow up with a detailed explanation and breakdown of all parts, along with some test run footages.

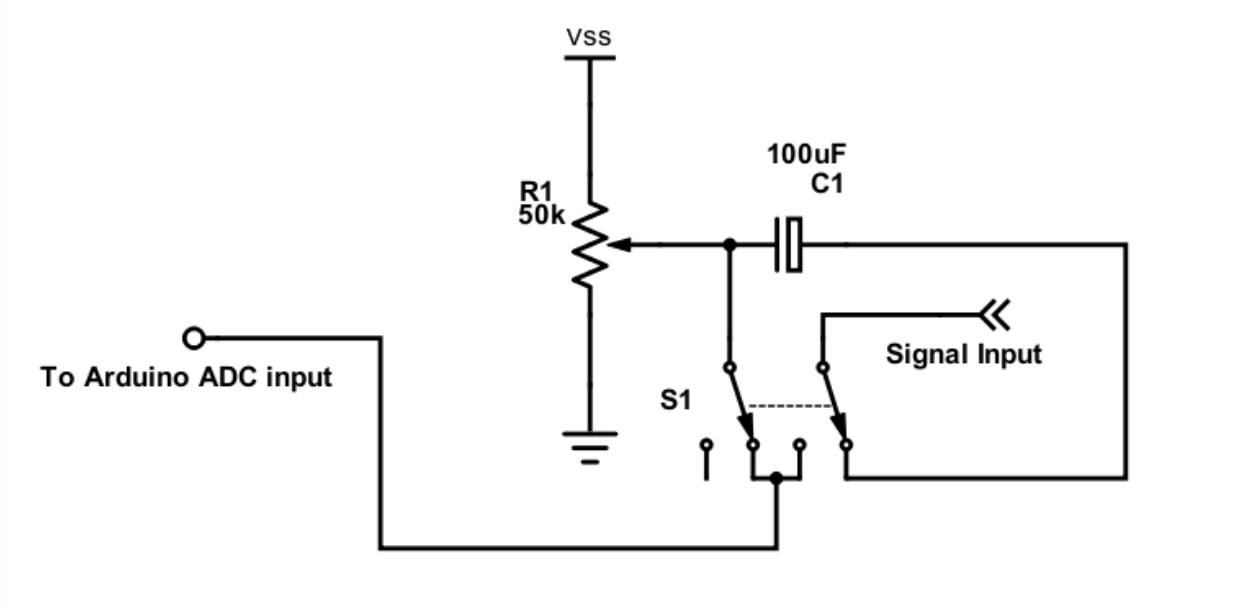


Explanation

The purpose of an Oscilloscope is sampling and processing signals. Our goal is to build an oscilloscope that is capable of dealing with voltage signals between +-2.5V, with a frequency up to 200Hz.

Preprocessing

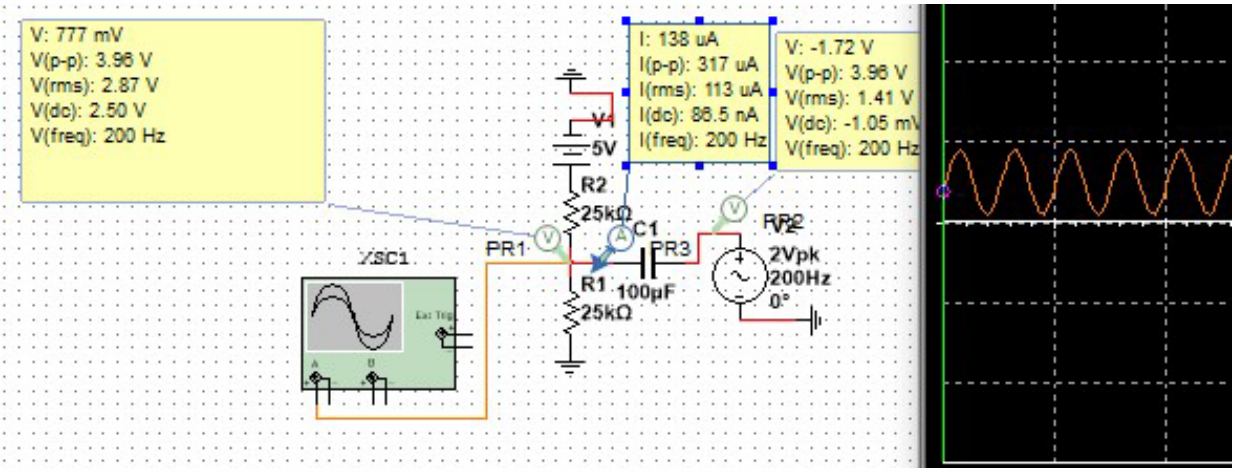
The first problem is that Arduino, like most other devices with an analog-to-digital converter, can not handle negative voltages. To deal with this problem, some pre-processing is needed so that our Oscilloscope can deal with negative voltage.



To deal with this problem, some pre-processing is needed. In the figure above, input signal goes through **C1**, getting rid of its DC component, then we feed a 2.5V DC component to it. In this way our signal goes from -2.5 ~ 2.5V to 0 ~ 5V. Arduino can read the signal and assume that `2.5V` is ground.

Of course, if we ever need DC coupling, we can always flip **S1** to the left and send the input signal directly to Arduino input.

We also ran a simulation to make sure this solution works, and the result looks great.



The input I_{rms} is 113 uA, suggesting a decent input impedance. Although a voltage follower would be nice, at this point it won't be necessary.

Sampling and Rising edge triggering

Here's the code we used:

```
//==============//
//This is just part of our code. The entrie program isn't finished, so
we'll submit our source code in the final report, or show you at the
presentation.
//==============//

int Buffer[256]; // buffer for input signal
int Y[128]; //Data for display

for(x = 0;x < 192;x++)    //take 192 sample input signal from A0
{
  delayMicroseconds(divT); //dT control
  Buffer[x] = (analogRead(A0)); //Read voltage from A0
}

//Find Vmax and Vmin.
int vmax = Buffer[0];
int vmin = vmax;
for(i=0;i<192;i++)
{
  if (Buffer[i] > vmax) vmax = Buffer[i];
  if (Buffer[i] < vmin) vmin = Buffer[i];
}

int Thr = (vmax + vmin)/2; //Thresold voltage, this helps to find the
rising edge
Vpp = (vmax-vmin)/1023.0*10; // peak-to-peak voltage

for(sta=0;sta<96;sta++) // this loop finds the position of the rising edge.
{
  if(Buffer[sta]<Thr && Buffer[sta+2]>Thr)
    break;
}

for(i = 0;i < 96;i++)  //feed the signal to display.
  Y[i] =  map((Buffer[i+sta]),0,1023,0,64);
```

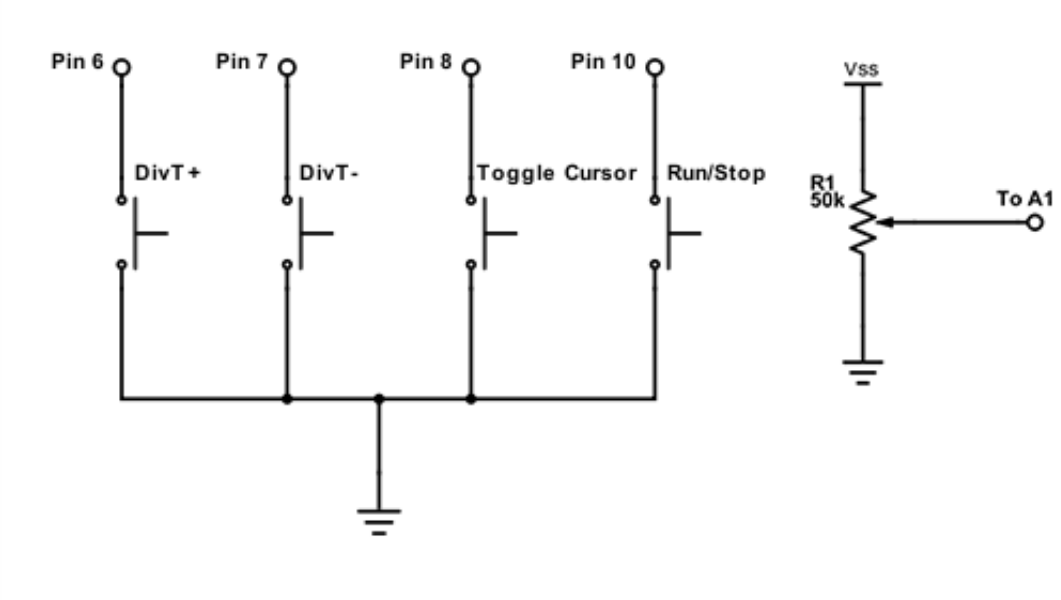
Basically, we take the input signal from A0, find max and minimum voltage. Then we find the rising edge. At the start of the first rising edge, we feed the data needed from buffer to Y, and now the signal is ready for display.

Signal Display

```
void drawWave(void){
       for(x = 0;x < 96;x++)
       u8g.drawLine(x,Y[x],x,Y[x+1]);  //draw the waveform itself.
```

Drawing the waveform itself is easy. We only used 96 columns of pixels on the left, since we need some space to display other data, such as Vpp or Cursor. Here's a screenshot from one of our test runs.

Keypad and cursor control



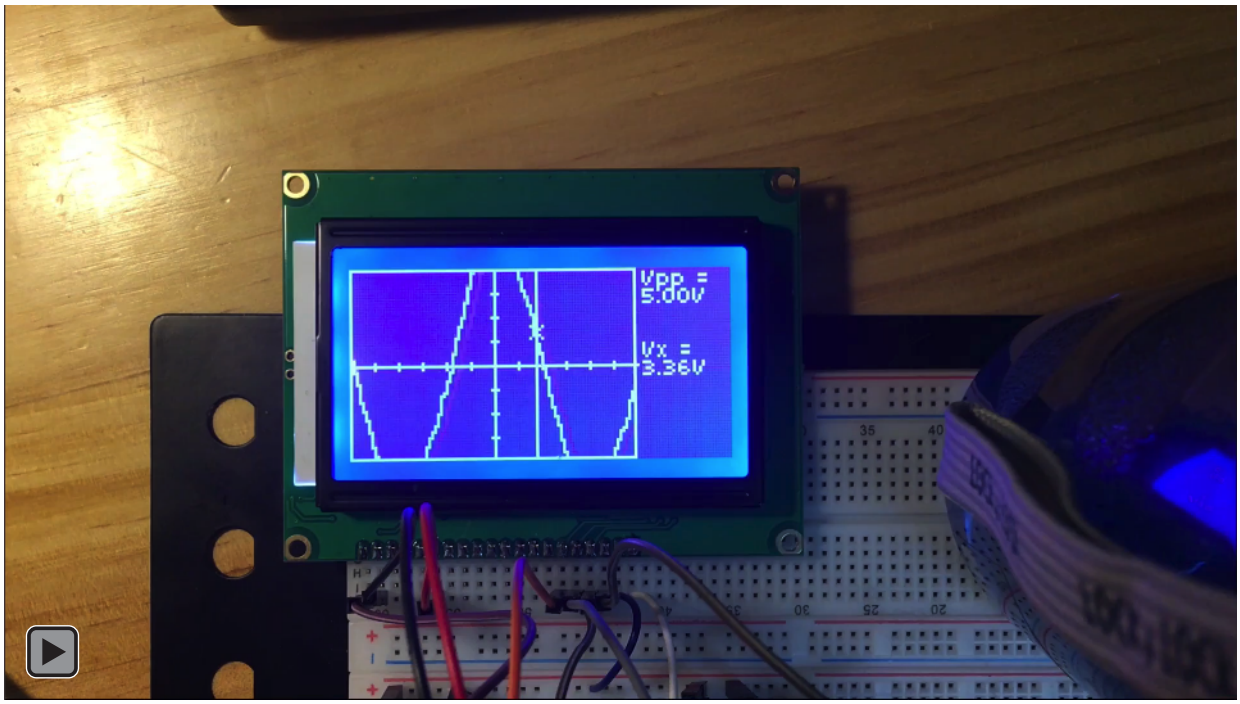
Like most MCUs, Arduino comes with built-in pull-up resistors. This saves a lot of trouble when we are trying to do key scans and stuff. Here's the code we used:

```
//keyscan
if (digitalRead(10) == 0){
    delay(20);                      //wait 20ms to get rid of switch
bouncing.
    if (digitalRead(10) == 0)stopMode = !stopMode;
    while(digitalRead(10) == 0);   //wait until the button is released.
 }

//============================//

//read Cursor position.
int cPos = constrain(map(analogRead(A1),400,800,0,95),0,95);
```

As you can see from the code, the voltage on `A1` controls the position of the cursor. Here's a footage from one of the test runs. The signal comes from a phone charger. I connected a short wire to A0 as an antenna. Then I used my hand to touch the phone charger. This way I become a human transmitter that transfer the electric leakage(which is a 50Hz signal) from my phone charger to A0. This is the closet thing to a function generator I could get. Anyway, if your device can not handle PDF files with embedded videos, you can check out the whole thing Here.(提取密码：dtk2)



Component List

Name	Type/Value	Desc	Qty
Arduino	UNO V3	MCU	1
Push button		For keypad	4
LCD module	ST7920	Display module	
Capacitor	100uf	filter	4
Potentiometer	50k		3

Also there's wires, switches, etc.

What's next

- Pre-Processer circuit and the potentiometer for cursor position control is yet to be implemented.
- Still need to find a good power solution.
- Also we plan to put the entire thing in a nice little casing.