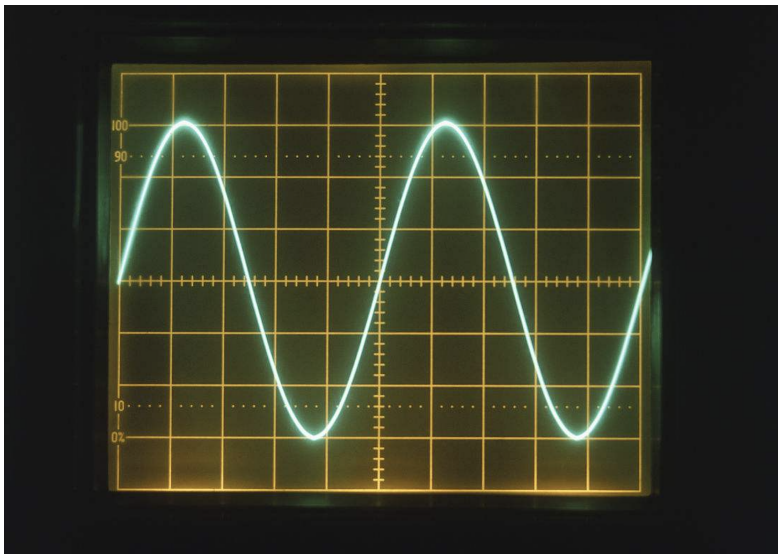
Electric Circuits (Fall 2017)

Project: Arduino Oscilloscope

Final Report



Song Xiaoyu and Qiu Jiacong

TA: LI Yike

January 14, 2018

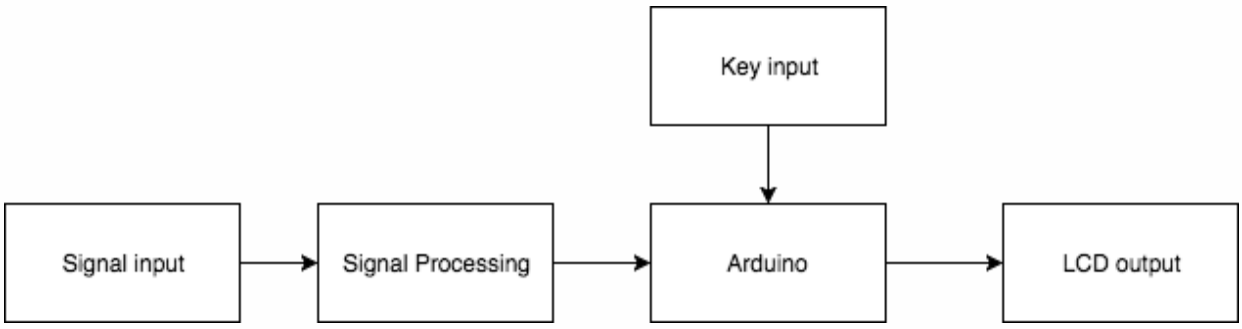
Design Requirements

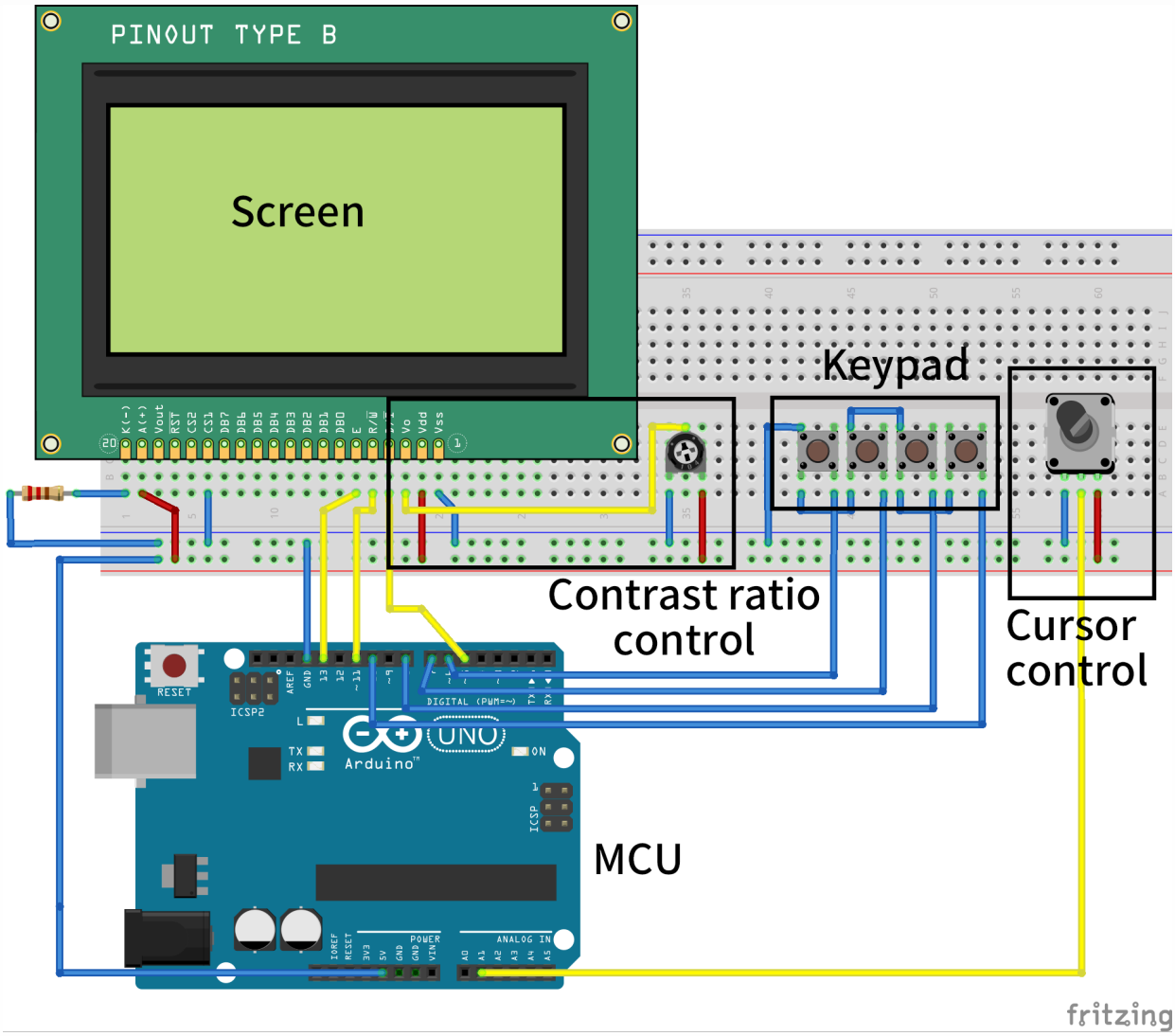
The goal of this project is to build an oscilloscope based on Arduino, that:

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

Actually, the original requirement was to build an 8-channel digital voltmeter. But our TA decided that this is too easy. So we and some other students came up with this instead.

Schematic





Get a detailed explanation below.

Overview

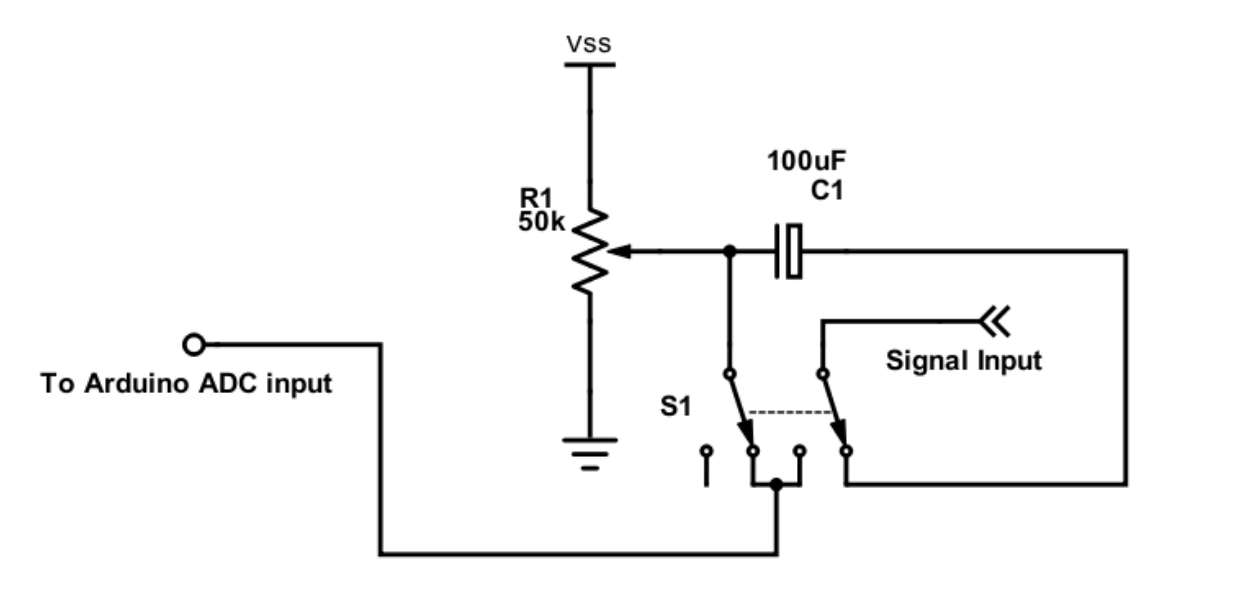
The purpose of an oscilloscope is sampling and processing signals.

Our goal is to build an oscilloscope that is capable of dealing with voltage signals between 0V to 5V, with a frequency up to 1000Hz. Due to the limitation of Arduino, any signal beyond 1000 Hz will be hard to see. Theoretically speaking, we can display any frequency below that. All we need to do is adjust sample rate. But sometimes it can be tiresome. We would need to push the `DivT +` button 100 times if we want to measure a 5 Hz signal.

The power of our oscilloscope come from a 12V DC wall adapter, and the voltage signal measured is displayed on a 128x64 LCD screen. Also, it can measure peak-to-peak voltage.

Preprocessing

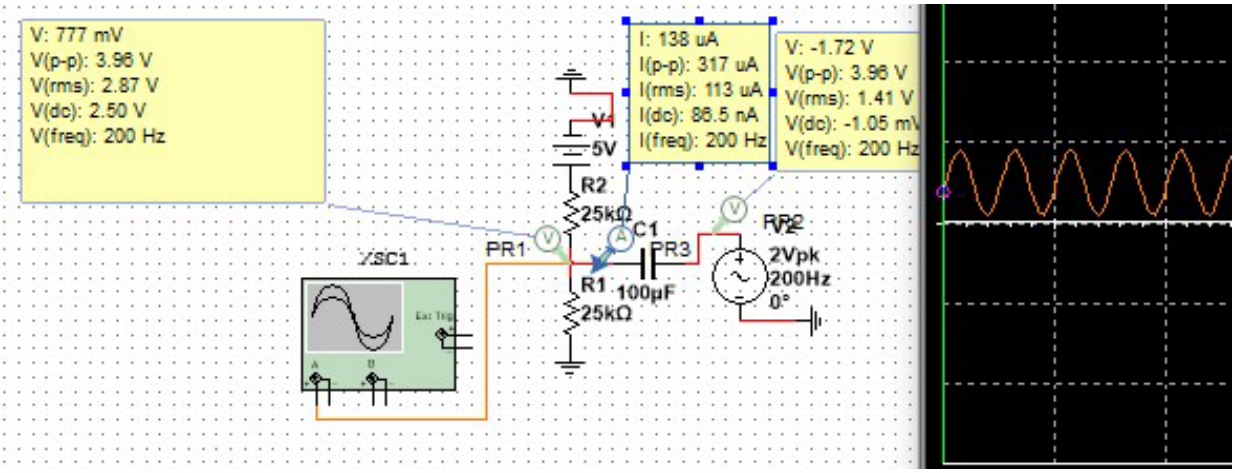
The first problem is that Arduino, like most other devices with an analog-to-digital converter, can not handle negative voltages. To deal with this problem, some pre-processing is needed so that our Oscilloscope can deal with negative voltage.



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.

Note that this circuit worked well when we were still working on the breadboard. But we **did not** implement this part in our final design because we ran out of potentiometers in an unfortunate incident. Later we also found that there isn't enough space in our casing for this part. So ,unfortunately, we had to give up.

Sampling and Rising edge triggering

Basically, we take 192 samples from the input signal.

```
for(x = 0;x < 192;x++)    //sampling input waveform from A0
{
  delayMicroseconds(divT); //this delay function controls sample rate.
  Buffer[x] = (analogRead(A0));
}
```

Then we find max and minimum voltage by iteraing through our sample.

```
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];
}
```

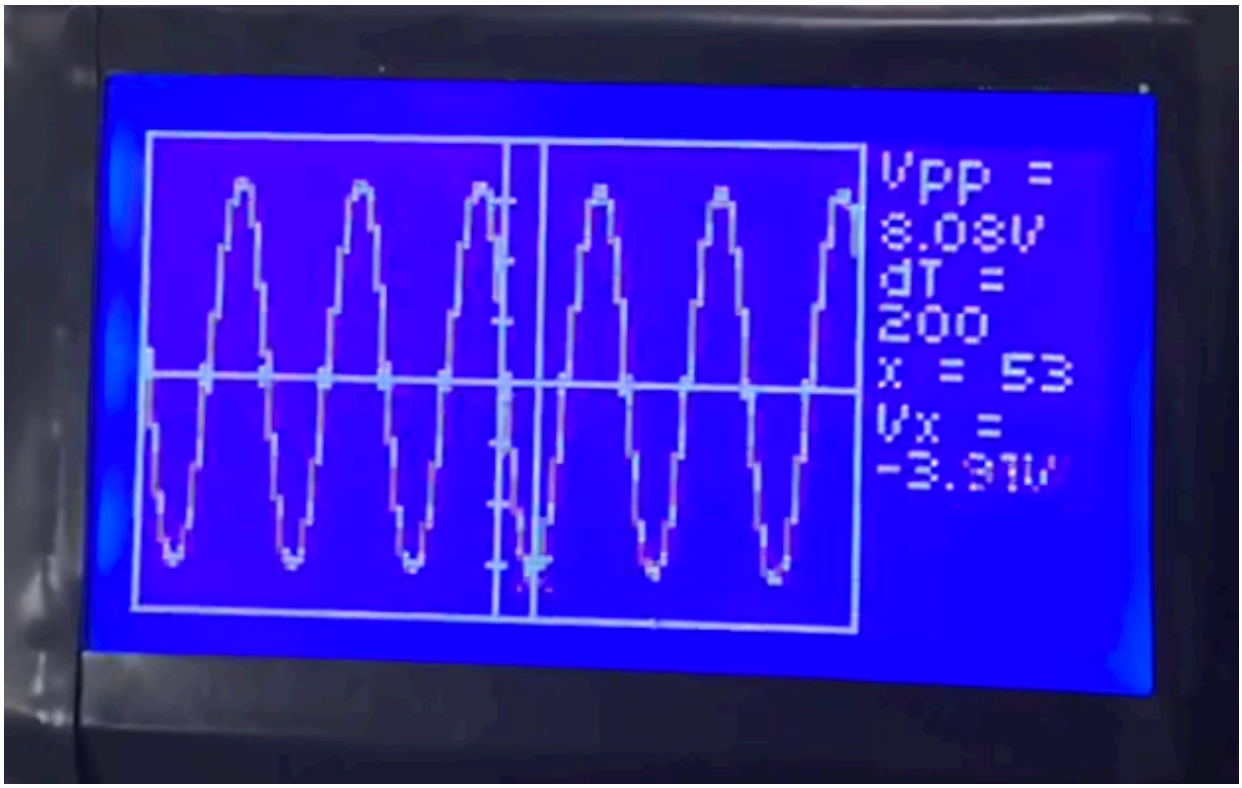
Now `vmax - vmin` is the peak-to-peak voltage.

After that, we find the rising edge. At the start of the first rising edge, we copy the data needed from the buffer to `Y`, and now the signal is ready for display.

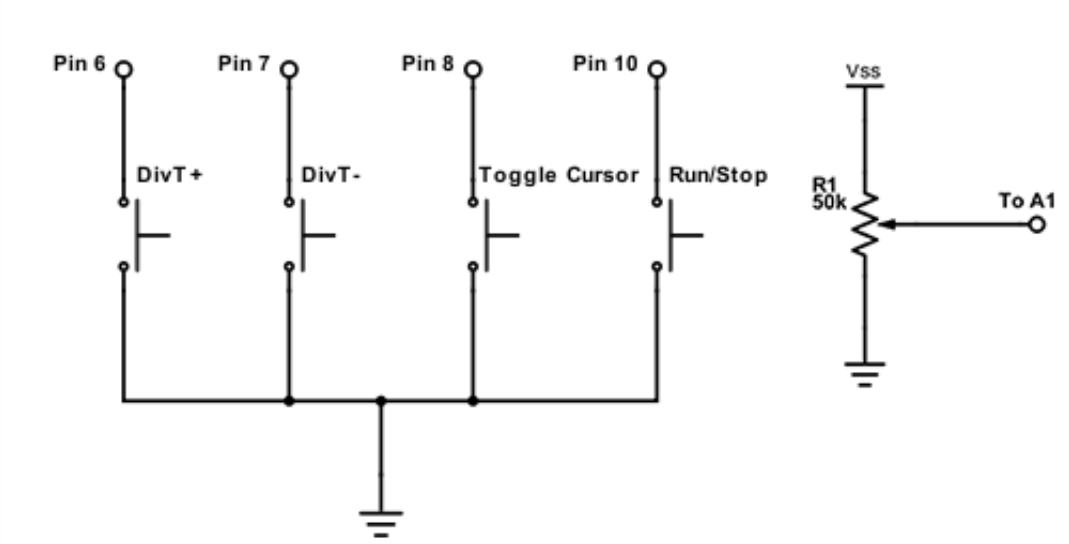
Signal Display

```
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.



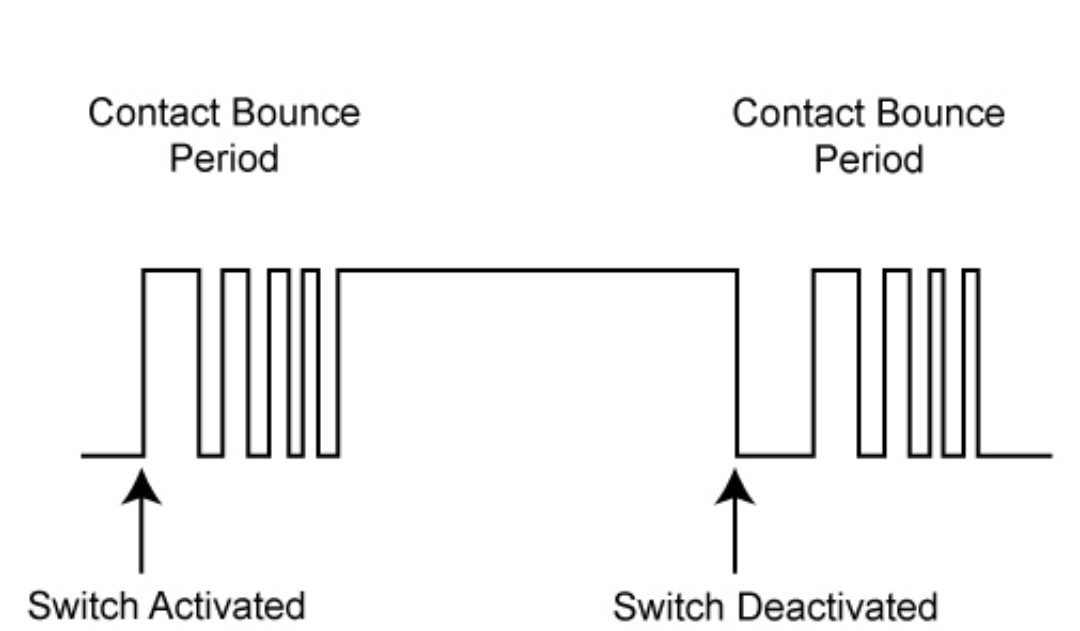
Keypad



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.
 }
```

Switch bouncing is another real-world problem that happens too quickly for human perception but which can doom an electronics project. When a switch is toggled, contacts have to physically move from one position to another. As the components of the switch settle into their new position, they mechanically bounce, causing the underlying circuit to be opened and closed several times, like this:



So one push of the button could be recognized as several pushes by our MCU. There are lots of ways of debouncing, and we have chosen the simplest of ways, which is wait 20ms until the switch is stable.

Cursor

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

X position of the cursor is controlled by the voltage from potentiometer R1.

Component List

Name	Type/Value	Desc	Qty
Arduino	UNO V3	MCU	1
Push button		For keypad	4
LCD module	ST7920	Display module	
Potentiometer	B10K(WH148 packaging)		2
Wall Adapter Power Supply	12V DC		1

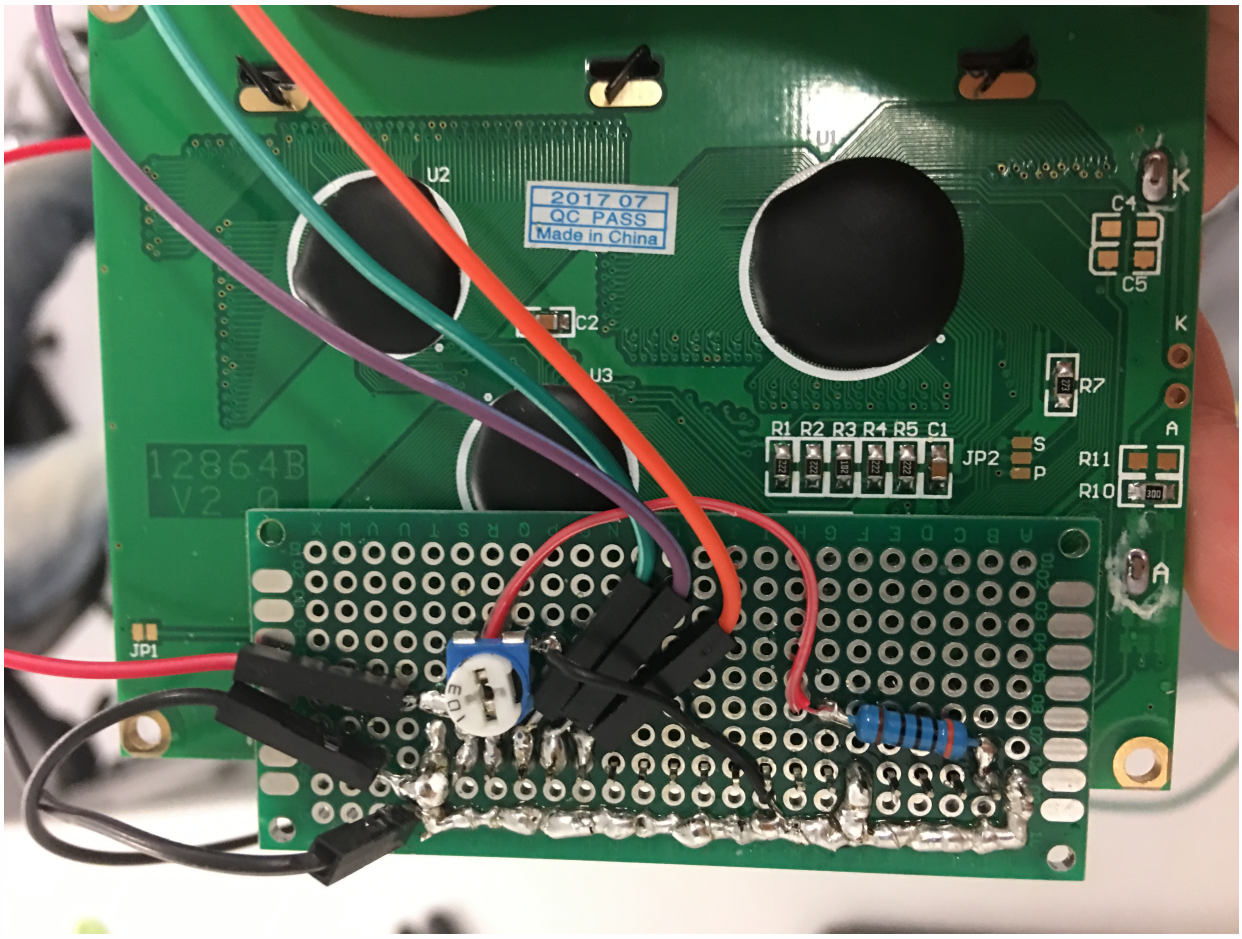
Also, there are wires, switches, etc.

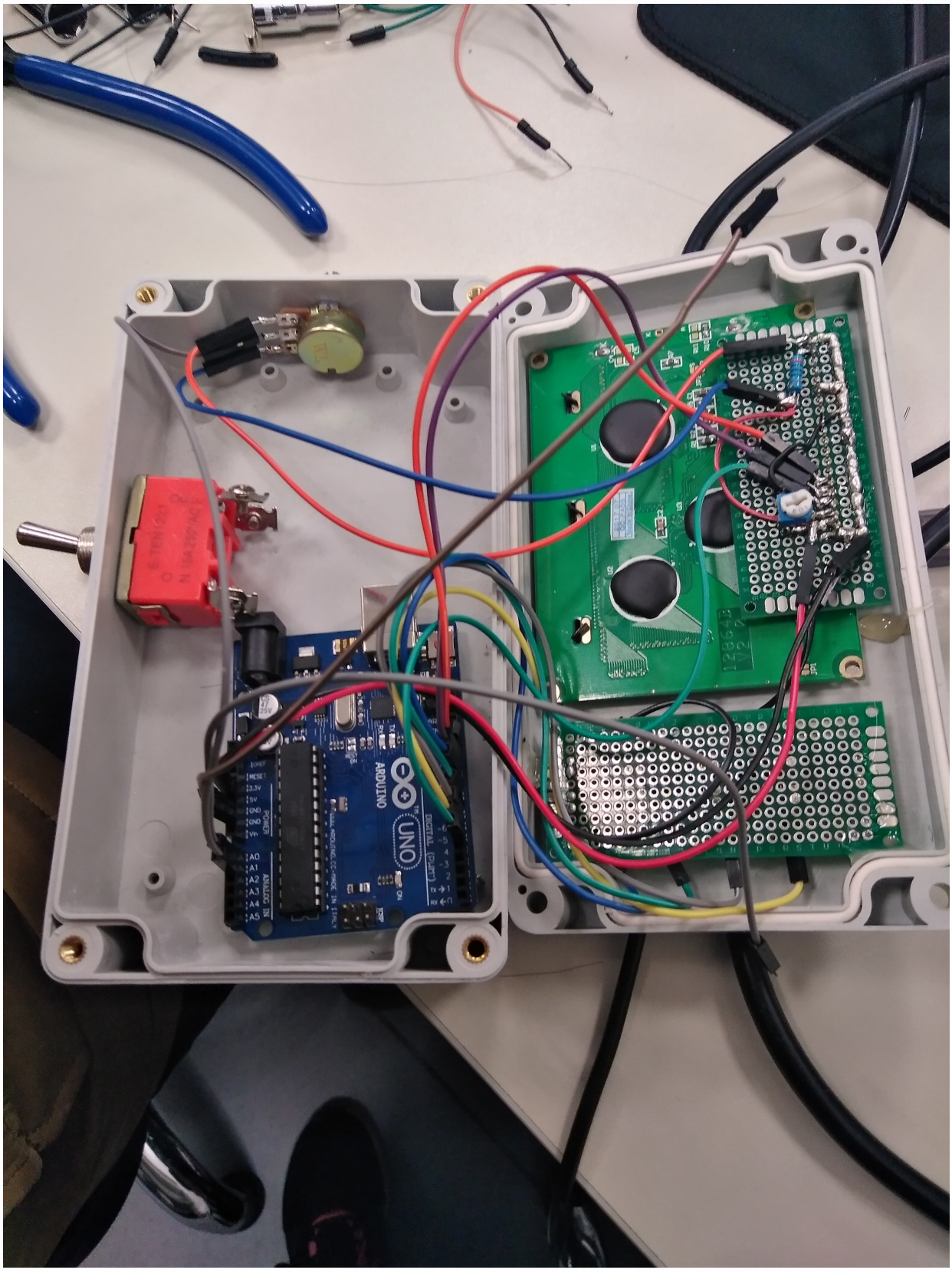
Simulation

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. Like what we have reported back in our midterm report, we skipped the simulation and straight went to building.

Building

The building process went (mostly) without a hitch. So here are some shots we've taken during our work.


U2
2017 07
QC PASS
Made in China
U1
C4
C5
K
C2
U3
R7
R1 R2 R3 R4 R5 C1
JP2
S
P
A
R11
R10
12864B
V2.0
JP1



Final Result

Here's an overview of our result.

We also have this cute welcome screen!

Don't forget to check out our video so that you can see this baby in action!

Appendix A: Code

```
#include "U8glib.h"

U8GLIB_ST7920_128X64_4X u8g(5); // Hardware SPI configuration. Pin 5 = CS.

// The logo used in welcome screen.
const unsigned char logo[512] U8G_PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xc0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xc0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf8, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xfc, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xfe, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xfe, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xf0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xe0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x77, 0xff, 0xe0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xf0, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xf8, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xfe, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xfe, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xfe, 0x00,
  0x00, 0x00, 0x3c, 0x00, 0x1f, 0xff, 0xfc, 0x00,
  0x00, 0x01, 0xff, 0x80, 0x0f, 0xff, 0xfc, 0x00,
  0x00, 0x07, 0xff, 0xc0, 0x0f, 0xff, 0xf8, 0x00,
  0x00, 0x0f, 0xff, 0xe0, 0x07, 0xff, 0xf0, 0x00,
  0x00, 0x1f, 0xff, 0xf0, 0x07, 0xff, 0xe0, 0x00,
  0x00, 0x3f, 0xff, 0xf8, 0x07, 0xff, 0x80, 0x00,
  0x00, 0xff, 0xff, 0xf8, 0x07, 0xff, 0x00, 0x00,
  0x00, 0xff, 0xff, 0xfc, 0x07, 0xff, 0x00, 0x00,
  0x01, 0xff, 0xff, 0xfc, 0x03, 0xff, 0x00, 0x00,
  0x03, 0xff, 0xff, 0xfc, 0x03, 0xff, 0x00, 0x00,
  0x07, 0xff, 0xfc, 0x7f, 0xe3, 0xff, 0x00, 0x00,
  0x0f, 0xff, 0xf0, 0x1f, 0xff, 0xff, 0x00, 0x00,
  0x0f, 0xff, 0xe0, 0x0f, 0xff, 0xff, 0x00, 0x00,
  0x1f, 0xff, 0xc0, 0x1f, 0xff, 0xff, 0x00, 0x00,
  0x1f, 0xff, 0x80, 0x3f, 0xff, 0xff, 0x00, 0x00,
  0x3f, 0xff, 0x80, 0x3f, 0xff, 0xff, 0x00, 0x00,
  0x3f, 0xff, 0x00, 0x3f, 0xff, 0xff, 0x00, 0x00,
  0x3f, 0xff, 0x00, 0x3f, 0xff, 0xfe, 0x00, 0x00,
  0x3f, 0xfe, 0x00, 0x3f, 0xff, 0xfe, 0x00, 0x00,
  0x1f, 0xfe, 0x00, 0x3f, 0xff, 0xfe, 0x00, 0x00,
```

```
  0x0f, 0xfe, 0x00, 0x3f, 0xff, 0xff, 0x00, 0x00,
  0x0f, 0xfc, 0x00, 0x3f, 0xef, 0xff, 0x80, 0x00,
  0x07, 0xfc, 0x01, 0xff, 0xc7, 0xdf, 0xc0, 0x00,
  0x03, 0xfc, 0x03, 0xff, 0xc7, 0xcf, 0xc0, 0x00,
  0x01, 0xfc, 0x07, 0xff, 0x87, 0xcf, 0xe0, 0x00,
  0x00, 0xfc, 0x0f, 0xef, 0x87, 0xe7, 0xf0, 0x00,
  0x00, 0x7c, 0x0f, 0xcf, 0x87, 0xe7, 0xf8, 0x00,
  0x00, 0x78, 0x1f, 0xdf, 0x07, 0xe7, 0xfc, 0x00,
  0x00, 0x38, 0x1f, 0x9f, 0x07, 0xe3, 0xfe, 0x00,
  0x00, 0x18, 0x3f, 0x9f, 0x07, 0xe1, 0xff, 0x00,
  0x00, 0x0c, 0x3f, 0x1f, 0x07, 0xf1, 0xff, 0x80,
  0x00, 0x00, 0x7f, 0x1f, 0x07, 0xf0, 0xff, 0x80,
  0x00, 0x00, 0x7f, 0x1f, 0x87, 0xf0, 0xff, 0x80,
  0x00, 0x00, 0x7f, 0x1f, 0x87, 0xf0, 0x7f, 0x00,
  0x00, 0x00, 0xfe, 0x1f, 0x87, 0xf8, 0x7e, 0x00,
  0x00, 0x00, 0xfe, 0x1f, 0xc7, 0xf8, 0x3e, 0x00,
  0x00, 0x00, 0xff, 0x1f, 0xc7, 0xf8, 0x1c, 0x00,
  0x00, 0x00, 0x3e, 0x1f, 0x87, 0xfc, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x0e, 0x07, 0xfc, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x07, 0xf8, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};

int x,y,sta,i,v;

//Buffer for buffing sample signal, and Y for the data to display
int Buffer[256];
int Y[128];

int divT = 200; // time delay when sampling
int nos = 0 ; //number of samples
float Vpp;

bool stopMode = false;
bool cursorMode = true;




void setup(void) {

  // flip screen, if required
  // u8g.setRot180();


  pinMode(10,INPUT); // Pin 10 reads if stop mode is on
  pinMode(8,INPUT); // Pin 8 reads if cursor mode is on
  pinMode(6,INPUT); // Pin 6 and 7 is for adjusting divT.
  pinMode(7,INPUT);
```

```
  // Turn on the built-in pull-up resistors
  digitalWrite(10,HIGH);
  digitalWrite(8,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  analogRead(A0);

  //Configure ADC registers.
  ADCSRA |= 00000100;



  // Draw a welcome screen.
  u8g.firstPage();
  do {
     welcomeScreen();
  } while( u8g.nextPage() );

  delay(1000);
}

/***************************/
// this function read the keys.//
void keyScan(void)
{
  // If a key is pressed, delay 20 ms and read again to get rid of any
noise, then wait until the button is released.

  if (digitalRead(10) == 0){
    delay(20);
    if (digitalRead(10) == 0)
    stopMode = !stopMode;
    while(digitalRead(10) == 0);
  }

    if (digitalRead(8) == 0){
    delay(20);
    if (digitalRead(8) == 0)
    cursorMode = !cursorMode;
    while(digitalRead(8) == 0);
  }


    if (digitalRead(6) == 0){
    delay(20);
    if (digitalRead(6) == 0)
    divT+=10;
    while(digitalRead(6) == 0);
    }

    if (digitalRead(7) == 0){
```

```
    delay(20);
    if (digitalRead(7) == 0)
    divT-=10;
    while(digitalRead(7) == 0);
  }

}




/***************************/
// This function samples the
// input signal and process them
// for display.
void sample(void) {



         for(x = 0;x < 192;x++)    //sampling input waveform from A0
        {
        delayMicroseconds(divT);
        Buffer[x] = (analogRead(A0));
        }


        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 for 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);
}

/*****************/
// This function draw the input signal, the axises and Vpp.
```

```
// It should be used inside a picture loop.
void drawWave(void){
       for(x = 0;x < 96;x++)
       u8g.drawLine(x,Y[x],x,Y[x+1]);  //draw the waveform itself.

       u8g.drawLine(48,0,48,63);  // draw axises
       u8g.drawLine(0,32,95,32);
       for(x=0;x < 96;x+=8)
       u8g.drawLine(x,31,x,33);
       for(x=0;x<64;x+=8)
       u8g.drawLine(47,x,49,x);

       u8g.drawFrame(0,0,96,64); //draw the frame

       //print Vpp
       u8g.setPrintPos(98,6);
       u8g.setFont(u8g_font_04b_03);
       u8g.print("Vpp = ");
       u8g.setPrintPos(98,14);
       u8g.print(Vpp,2);u8g.print("V");

       //print dt
       u8g.setPrintPos(98,20);
       u8g.print("dT = ");
       u8g.setPrintPos(98,26);
       u8g.print(divT);

       //print stopmode indicator
       if (stopMode){
        u8g.setPrintPos(98,53);
        u8g.print("STOP");
       }

}

//this function draw the cursor and the corresponding value.
// It should be used inside a picture loop.
void drawCur(void){
  // read cursor position from A9 (which is controlled by a potentiometer)
  // also calculate the corresponding value
  int cPos = constrain(map(analogRead(A9),400,800,0,94),0,95);
  float value = 5-Y[cPos]/64.0*10;

  u8g.drawLine(cPos,0,cPos,63); //draw cursor

  u8g.drawLine(max(cPos-
2,0),max(Y[cPos]-2,0),min(cPos+2,95),min(Y[cPos]+2,64)); //draw a little X
mark
  u8g.drawLine(min(cPos+2,95),max(Y[cPos]-2,0),max(cPos-
2,0),min(Y[cPos]+2,64));

  // show the value of the cursor
```

```
  u8g.setFont(u8g_font_04b_03);
  u8g.setPrintPos(98,32);
  u8g.print("x = ");
  u8g.print(cPos);
  u8g.setPrintPos(98,39);
  u8g.print("Vx = ");
  u8g.setPrintPos(98,45);
  u8g.print(value,2);
  u8g.print("V");
}


void loop(void) {

    keyScan();


    if (!stopMode){
      sample();
    }

    //picture loop
     u8g.firstPage();
     do
     {
      drawWave();
      if(cursorMode)
      drawCur();
    }
   while( u8g.nextPage( ));

}

void welcomeScreen(void) {

  u8g.setColorIndex(1); // draw black pixels
  u8g.setFont(u8g_font_04b_03); //set a font
  u8g.setPrintPos(4, 15);
  u8g.print("The ultimate");
  u8g.drawBitmapP( 32, 0, 8, 64, logo); //draw the logo pony
  u8g.setColorIndex(0);
  u8g.setPrintPos(63, 39);
  u8g.print("volt");
  u8g.setColorIndex(1);
  u8g.setPrintPos(81, 39);
  u8g.print("pony");
}
```