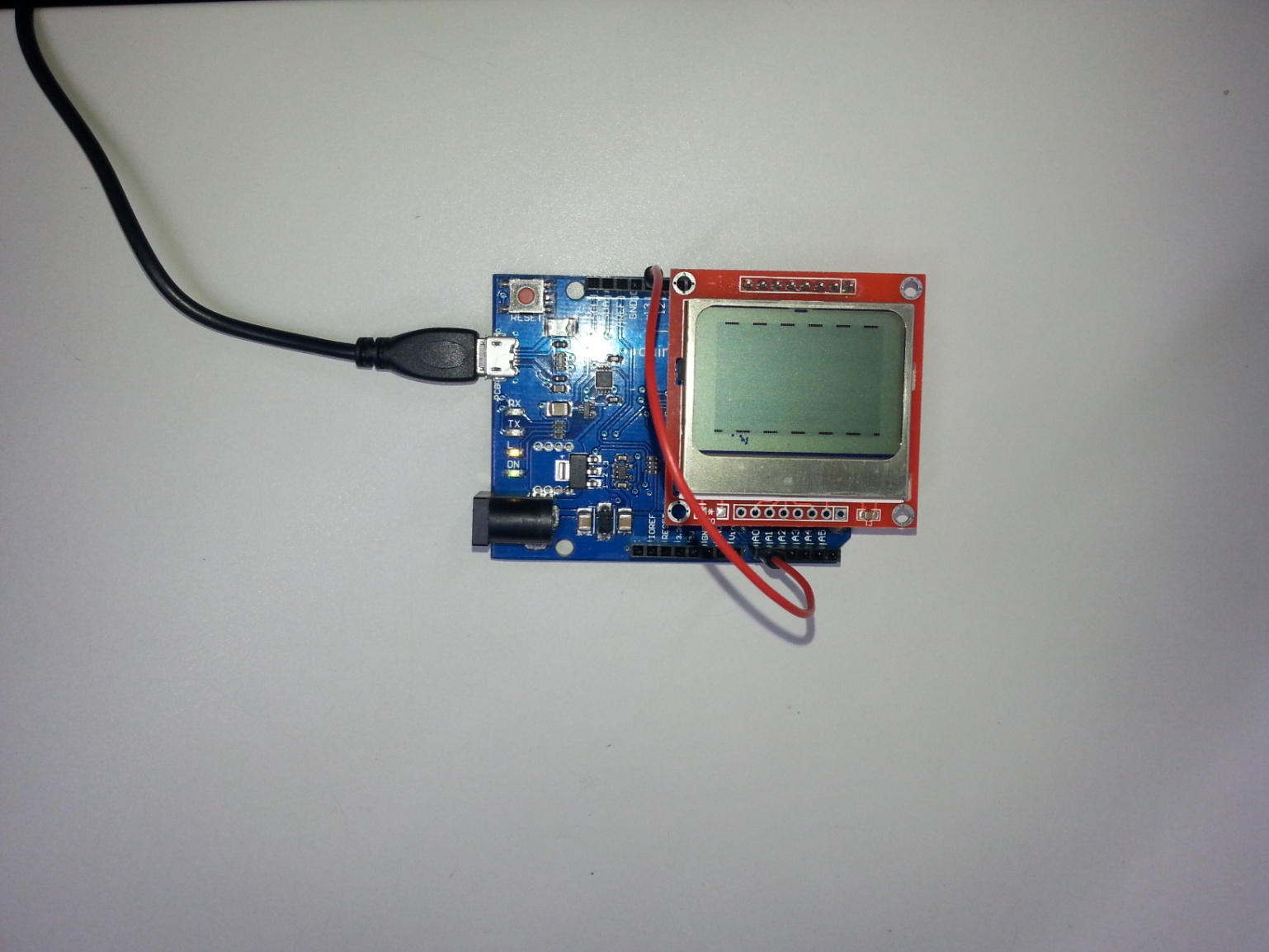
Simple Duinotech Oscilloscope

This little test tool is designed to be easily put together if you need a very basic ‘scope in a hurry. There’s no high end specs here- the maximum sample rate is about 700 samples per second, and it’s limited to the 0-5V that the Arduino analog pins can handle. Still, it’s sensitive enough to pick up the 50Hz noise from mains wiring without making contact. Also includes a tone generator, so you can compare signal frequencies too.



## Shopping list:

XC4430 Duinotech Leonardo Board

XC4616 Arduino Compatible 84x48 Dot Matrix LCD Display Module

WC6024 Plug-Plug Jumper Leads

## Connections:

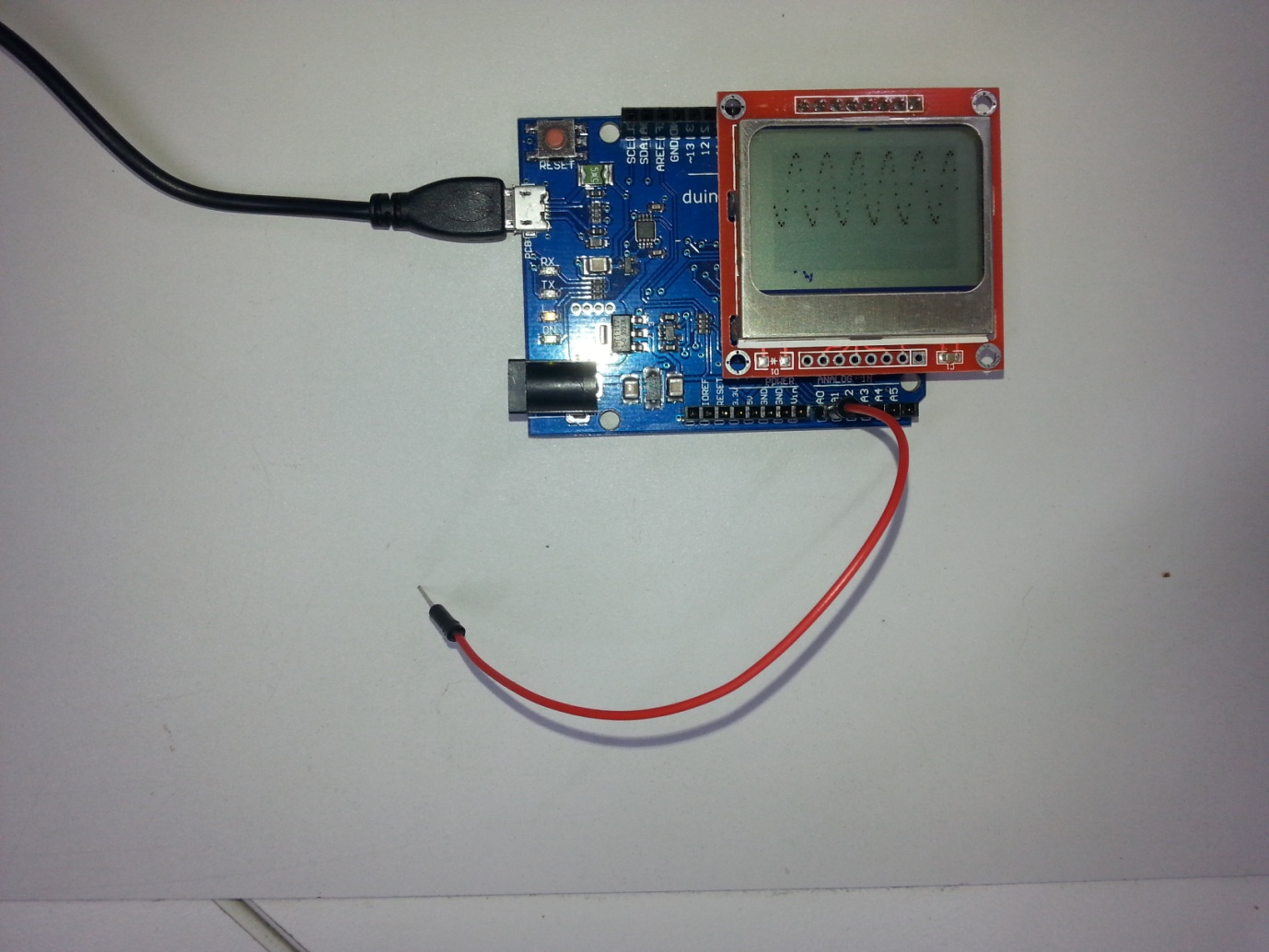
|  |  |  |  |
| --- | --- | --- | --- |
| Leonardo | Display | Jumper Wire | Function |
| D0 | VCC |  | Power |
| D1 | GND |  | Ground |
| D2 | SCE |  | Chip Select |
| D3 | RST |  | Reset |
| D4 | D/C |  | Data/Command Select |
| D5 | DN(MOSI) |  | Serial Data from Leonardo |
| D6 | SCLK |  | Serial Clock from Leonardo |
| D7 | LED |  | LED power |
| A0 |  | One end | Scope input |
| D13 |  |  | Output for tone generator |

You’ll notice that the connection between the Leonardo and display is achieved by simply plugging the Display header into D0-D7 on the Leonardo. One end of a jumper can be plugged into A0 (the other end is your ‘scope probe). There is only one probe, as the signal is referenced to the Arduino’s ground- you could add a return probe here, just be careful that what you’re connecting to doesn’t have another connection back to the Leonardo board. You can use the tone generator output by plugging the other end of your probe into D13- this is what’s happening in the picture at the top of the page.

## Code:

The code is designed to be fast and simple, to achieve a high sample rate- the analog input is read, converted to a pixel value, then he display automatically moves a column to the right (so the scanning happens automatically). There are no libraries needing to be installed, as all the necessary functions are built into the code. The construction can be simplified to the simple plug-in arrangement because the display uses very little power, and even a digital output can provide enough power to run it. We’re using a Leonardo because an Uno would need D0 and D1 for uploading the sketch, and having something plugged in would interfere with uploading. You could use an Uno, just don’t have the screen plugged in while you’re uploading.

## Use:



Here we can see the probe picking up ‘50Hz hum’ from the surroundings. It looks like a lot, but even a very high value resistor would dissipate it and make the value much lower. The scope could be used to quickly visualise whether a signal is present in another project, or even to see if it is at the right voltage levels (there’s no scale, but the bottom is 0V and the top is 5V).

## Improvements:

There’s some notes inline in the sketch as to how it can be upgraded- for example the tone frequency can be changed by changing the definition of TONE\_FREQ. Gridlines can be added by changing the initial declaration of the dispdata array. The actual sample rate can be changed by changing the delay at the end of the loop function- this can be handy if you’re trying to lock onto a certain frequency. If you’re changing some of these settings frequently, perhaps they could be tied to an analog input attached to a potentiometer. The sample rate cannot be easily increased without totally rewriting the code, but is not impossible. Other scope features such as triggering or different vertical scales are certainly possible, but would require more input devices to set these values, which would take away from the simplicity.

## Sketch:

// Simple Duinotech Oscilloscope

// using Leonardo and XC4616 display

// max sample rate of 726 columns/sec (column takes 1377 us) or about 8.6 frames/second (84 columns/screen)

// generates a reference tone on D13 for timebase checking

// plug the 8 pin header straight into d0-d7, LED on pin D0, VCCon pin D7

// VCC GND SCE RST DC MOSI SCLK LED

// 0 1 2 3 4 5 6 7

// LED must be off, as it needs 3.3V not 5V

#define PIN\_VCC 0

#define PIN\_GND 1

#define PIN\_SCE 2

#define PIN\_RESET 3

#define PIN\_DC 4

#define PIN\_SDIN 5

#define PIN\_SCLK 6

//analog input pin

#define PIN\_INPUT A0

//tone output reference pin and frequency

#define PIN\_TONE 13

#define TONE\_FREQ 50

#define LCD\_C LOW

#define LCD\_D HIGH

byte bitdata[8]={128,64,32,16,8,4,2,1}; //array to speed up conversion of value to pixel position;

void setup(void)

{

delay(100);

LcdInitialise();

tone(PIN\_TONE,TONE\_FREQ);

}

void loop(void)

{

//LcdSety(0);

int data;

byte dispdata[6]={0,0,0,0,0,0}; //pixels to display per row (could change this to something like (2,0,0,1,0,128) to give rudimentary gridlines

data =analogRead(PIN\_INPUT)/22; //read input and convert from 0-1023 to 0-47

int bt;

bt=5-data/8;

// dispdata[bt]=dispdata[bt]|bitdata[data&7]; //set pixel (use this version if pixels are non zero)

dispdata[bt]=bitdata[data&7]; //set pixel

digitalWrite(PIN\_DC, LCD\_D); //unroll display function to speed it up

digitalWrite(PIN\_SCE, LOW);

shiftOut(PIN\_SDIN, PIN\_SCLK, MSBFIRST, dispdata[0]);

shiftOut(PIN\_SDIN, PIN\_SCLK, MSBFIRST, dispdata[1]);

shiftOut(PIN\_SDIN, PIN\_SCLK, MSBFIRST, dispdata[2]);

shiftOut(PIN\_SDIN, PIN\_SCLK, MSBFIRST, dispdata[3]);

shiftOut(PIN\_SDIN, PIN\_SCLK, MSBFIRST, dispdata[4]);

shiftOut(PIN\_SDIN, PIN\_SCLK, MSBFIRST, dispdata[5]);

digitalWrite(PIN\_SCE, HIGH);

delayMicroseconds(51); //change timebase by changing delay between cycles- 51=> 700Hz sample rate, gives a stable display of 50Hz wave

}

void LcdClear(void)

{

for (int index = 0; index < 504; index++){LcdWrite(LCD\_D, 0x00);}

}

void LcdInitialise(void)

{

pinMode(PIN\_VCC, OUTPUT); //VCC

digitalWrite(PIN\_VCC,HIGH);

pinMode(PIN\_GND, OUTPUT); //GND

digitalWrite(PIN\_GND,LOW);

//Leave LED floating (off)

pinMode(PIN\_SCE, OUTPUT);

pinMode(PIN\_RESET, OUTPUT);

pinMode(PIN\_DC, OUTPUT);

pinMode(PIN\_SDIN, OUTPUT);

pinMode(PIN\_SCLK, OUTPUT);

digitalWrite(PIN\_RESET, LOW);

delay(100);

digitalWrite(PIN\_RESET, HIGH);

LcdWrite( LCD\_C, 0x21 ); // LCD Extended Commands.

LcdWrite( LCD\_C, 0xB0 ); // Set LCD Vop (Contrast). //A8-B1 TRY ADJUSTING THIS IF IT'S NOT WORKING 80-FF are valid

LcdWrite( LCD\_C, 0x04 ); // Set Temp coefficent. //0x04

LcdWrite( LCD\_C, 0x14 ); // LCD bias mode 1:40. //0x14

LcdWrite(LCD\_C, 0x20);

LcdWrite(LCD\_C, 0x22); //display on, vertical scan (0x20 is horizontal scan)

LcdWrite(LCD\_C, 0x0C); //display on and normal

// LcdWrite(LCD\_C, 0x0D); //display on and inverted

LcdClear();

LcdSetx(0);

LcdSety(0);

}

void LcdWrite(byte dc, byte data)

{

digitalWrite(PIN\_DC, dc);

digitalWrite(PIN\_SCE, LOW);

shiftOut(PIN\_SDIN, PIN\_SCLK, MSBFIRST, data);

digitalWrite(PIN\_SCE, HIGH);

}

void LcdSetx(byte x)

{

if(x<84)

{LcdWrite(LCD\_C,0x80 | x);

}

}

void LcdSety(byte y)

{

if(y<6)

{LcdWrite(LCD\_C,0x40 | y);

}

}