Ted Drell, W5HEU

893 Ridgeview Dr, Pineville, LA 71360; **teddy@tldrell.com**

An Experimenter's Variable Voltage Transformer

Here is a modern twist to controlling an old workbench essential.

I am an old antique radio buff, and as anyone playing with such old toys as these will tell you, you must have a variable voltage transformer (autotransformer) to work on them. (One brand name of such an autotransformer is Variac.) I had one that I put together from excess parts; a "Bud" box, a panel mounted autotransformer, fuse holder, switch, and a panel mounted AC receptacle. This was a neat unit that would handle 3 A and fit nicely on my workbench. It did a good job until it was overloaded and I let the smoke out. The problem was that there was no way to monitor the voltage and current without using external meters and that was messy. Oh well. I needed to add meters to the unit, but panel space was at a premium so the unit sat there with the top off taking up space.

Then along came the May 2013 issue of *QST* and the article on page 39 touting the merits of the Arduino Uno processor. I immediately ordered two of these little guys to play with. I got the complete experimenters kit with prototype board, LCD display, and so on. It did not take long to realize that this could be the answer I've been looking to use for a number of projects. I have several going on in parallel, but one is now finished. It is an instrumented version of my Autotransformer, and the details follow.

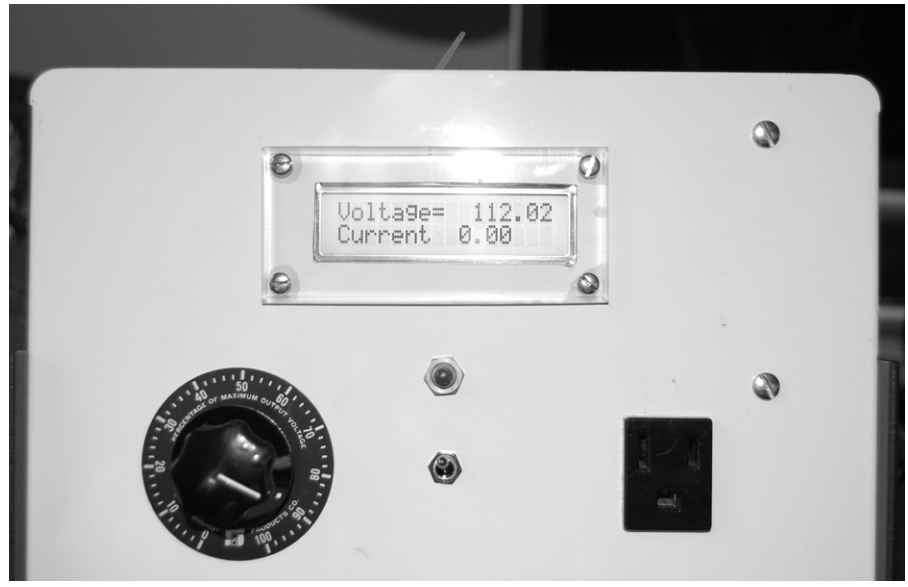

Figure 1 — Front panel layout.

The Circuit

The May 2013 article in *QST* gives the basics to get started with the Arduino, and there are books from ARRL as well as lots of free stuff on the Internet that will give you most of what you need for the Arduino itself, so I won't take up space with that. It became obvious that I need some sort of interface for the processor. More searching revealed multiple prototype "shields" available in the range of $15.00 to $25.00. I chose the Proto-Screwshield (Wingshield) kit from Adafruit ($16.00) because it has screw type terminal blocks for all of the Arduino connections and a large prototyping area.

The first interface was to the LCD readout. For that, I used a dual row, 8 pins per row header. The LCD also has a dual row header, so a 16 conductor ribbon cable with IDC connectors was used to connect the proto board to the LCD. The only component needed for the LCD was the contrast potentiometer.

All that was needed to complete the project was a voltage and current transducer and the project was done. I think my "junk box" is better than the average ham's, because I design and build a lot of small portable test systems. I had a current transducer that was self excited and would output 0 to 5 V DC for 0 to 10 A. Since the autotransformer was a 3 A unit, two turns of wire through the current transducer made it 0 to 5 V DC for 0 to 5 A. No scaling is necessary because the Arduino analog input is 0 to 5 V DC; Perfect! The voltage transducer is also self excited and puts out 0 to 5 V DC for 0 to 600 V AC input. I multiplied the 5 V output by 120 in the software and I now read line voltage perfectly.

[1]The author's software code for the Arduino is available for download from the ARRL *QEX* files web page. Go to **www.arrl.org/qexfiles** and look for the file **3x13_Drell.zip**.

Table 1
Software Code Listing for the Arduino

```
/* Variac Voltage and Current Monitor
  Set for a 24x2 LCD display.
  The circuit:
 * LCD RS pin to digital pin 7
 * LCD Enable pin to digital pin 6
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)
  Library originally added 18 Apr 2008 by David A. Mellis  library modified 5 Jul 2009
 by Limor Fried (http://www.ladyada.net)
   */

// include the library code:
#include <LiquidCrystal.h>
//#include <OneWire.h>
//#include <DallasTemperature.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

//const int busPin = 10;
//OneWire bus(busPin);
//DallasTemperature sensors (&bus);
//DeviceAddress sensor;

int voltage=A0; // select the input pin for the voltage sensor
int current1=A1; // select the input pin for the current sensor #1

int voltageValue = 0; // variable to store the value coming from the voltage sensor
int current1Value = 0; // variable to store the value coming from the current sensor #1

float curtrip = 3.00;
int ledPin = 9;
int relayPin = 8;



void setup() {

//  pinMode(busPin, INPUT);
//  sensors.begin();
//  sensors.getAddress(sensor, 0);

  pinMode(ledPin, OUTPUT);
  pinMode(relayPin, OUTPUT);
  // set up the LCD's number of columns and rows:
 lcd.begin(24, 2);  // initialize the lcd
  lcd.clear();  // clear lcd
  lcd.setCursor(3,0);
  lcd.print("TDA LLC");
  lcd.setCursor(1,1);
  lcd.print("Variac Monitor");
```

```
 digitalWrite(ledPin, LOW);
 digitalWrite (relayPin, LOW);

 delay(3000);

}

 void loop() {

 //  sensors.requestTemperatures();

 //  float tempF = sensors.getTempF(sensor);

 voltageValue = analogRead(voltage);
 current1Value = analogRead(current1);

 float vol = (voltageValue * (5.00/1023.0))*30;
float cur1 = current1Value * (5.00/1023.0);

 // set the cursor to column 0, line 0

 if (cur1 < curtrip){

 digitalWrite(ledPin, LOW);
 digitalWrite (relayPin, HIGH);
 lcd.clear();
 // read and display the voltage
 lcd.setCursor(0,0);
 lcd.print("Vol=");
 lcd.setCursor(5,0);
 lcd.print(vol);

// lcd.setCursor(13,0);
// lcd.print("Temp=");
// lcd.setCursor(19,0);
// lcd.print(tempF);

 // read and display the current
 lcd.setCursor(0, 1);
 lcd.print("Cur=");
 lcd.setCursor(7, 1);
 lcd.print(cur1);

 delay(500);

 }
 else{
    while(1){
      digitalWrite(ledPin, HIGH);
      digitalWrite(relayPin, LOW);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Tripped Over Cur");
    lcd.setCursor(0,1);
    lcd.print("Press Reset");
    delay(1000);
    }
 }

 }
```

Since I smoked the first autotransformer, I wanted over-current protection. With a microprocessor now working, adding a little software to protect the unit was no problem. I did have to add a transistor driver and a relay for this to work. The software allows the unit to operate until the current reaches 3.0 A, at which point it disconnects the power, thereby protecting the autotransformer from overload. Once the system is tripped, you have to manually reset it. When tripped, the LCD display says "Tripped on over current; Press reset." If the overload is removed, the system will reset and resume operation.

The low current relay on the prototype board was not large enough to switch line voltage, so I used an external relay. I had an open frame power supply that I used to power the Arduino and relays. Since the recommended power for the Arduino is nominally 12 V DC, I used a neat little DC-DC converter from Murata. It does not get hot and is small so that it fits on the proto board with no heat sink needed. Rounding it out, I included an LED to provide trip indication if desired.

A complete parts list is included as a guide. My sensors are expensive and cheaper ones are available. This article is intended to show the flexibility of the processor and how I applied it. With a little software modification, it can do a multitude of things.

Figure 1 shows the completed project, with the main display showing the adjusted voltage as well as the current. The LCD display is a 24 × 2 (or 2 × 24) which means there are 2 lines each with 24 characters. This limits the dialog. Other displays can be used if defined in the software. I also have another project where I can select multiple pages to display by switch selection, so anything is possible.

Figure 2 is the wiring of the protoshield and Figure 3 is the wiring of the autotransformer.

The software is available for download from the ARRL QEX files web page.[1] It is open source so you are free to use it as is, or change it to suit your needs. The language is basically *C++* so anyone familiar with *C++* will not have any problems using it. The programming guides and libraries are available for free download from the Arduino web site.

The software was originally written for another project and had the ability to add the Dallas Semiconductor "One Wire" temperature sensor. Since this was not used in this project, the code for this is commented out.

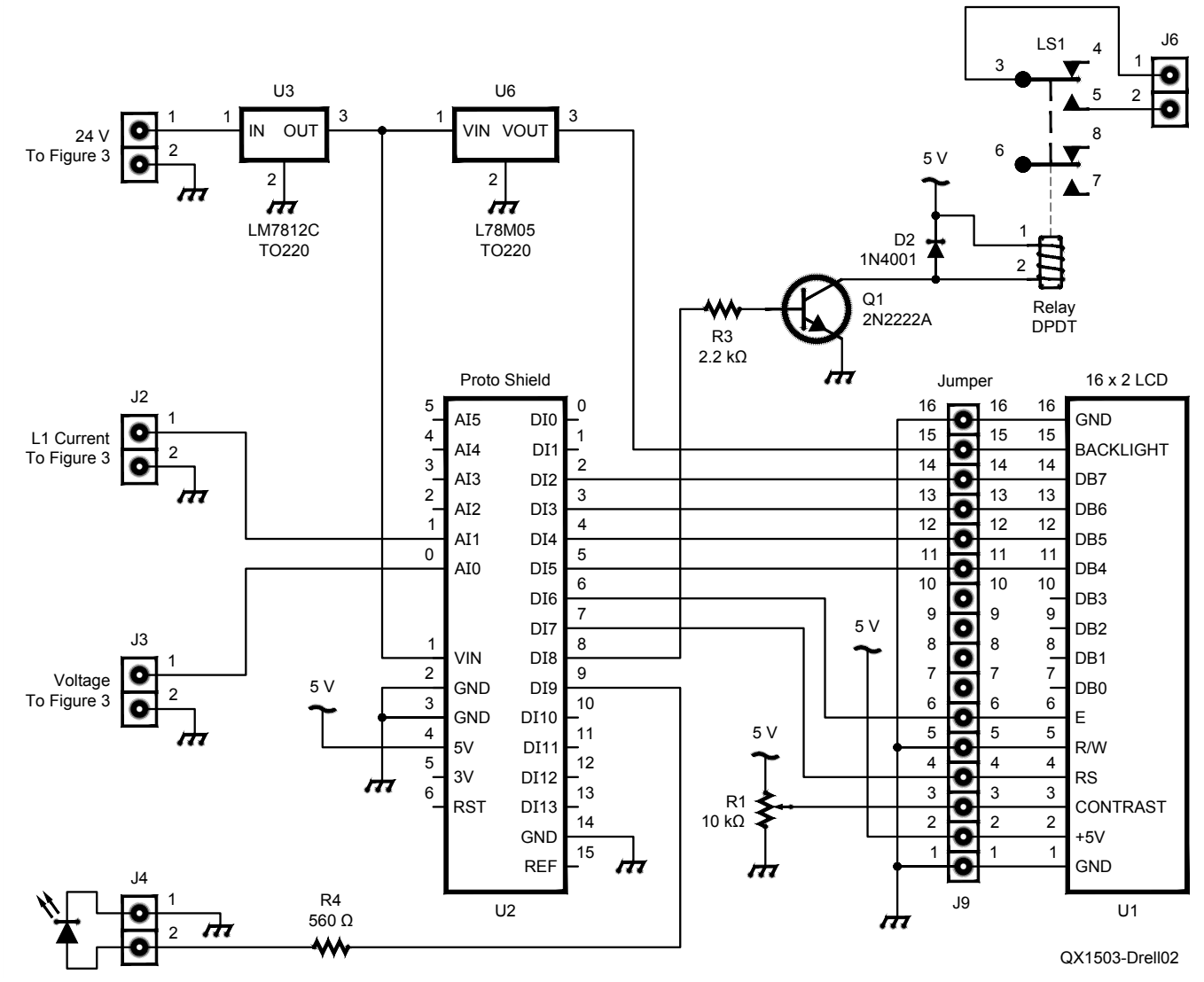

Figure 2 — The schematic diagram of the Arduino protoshield.

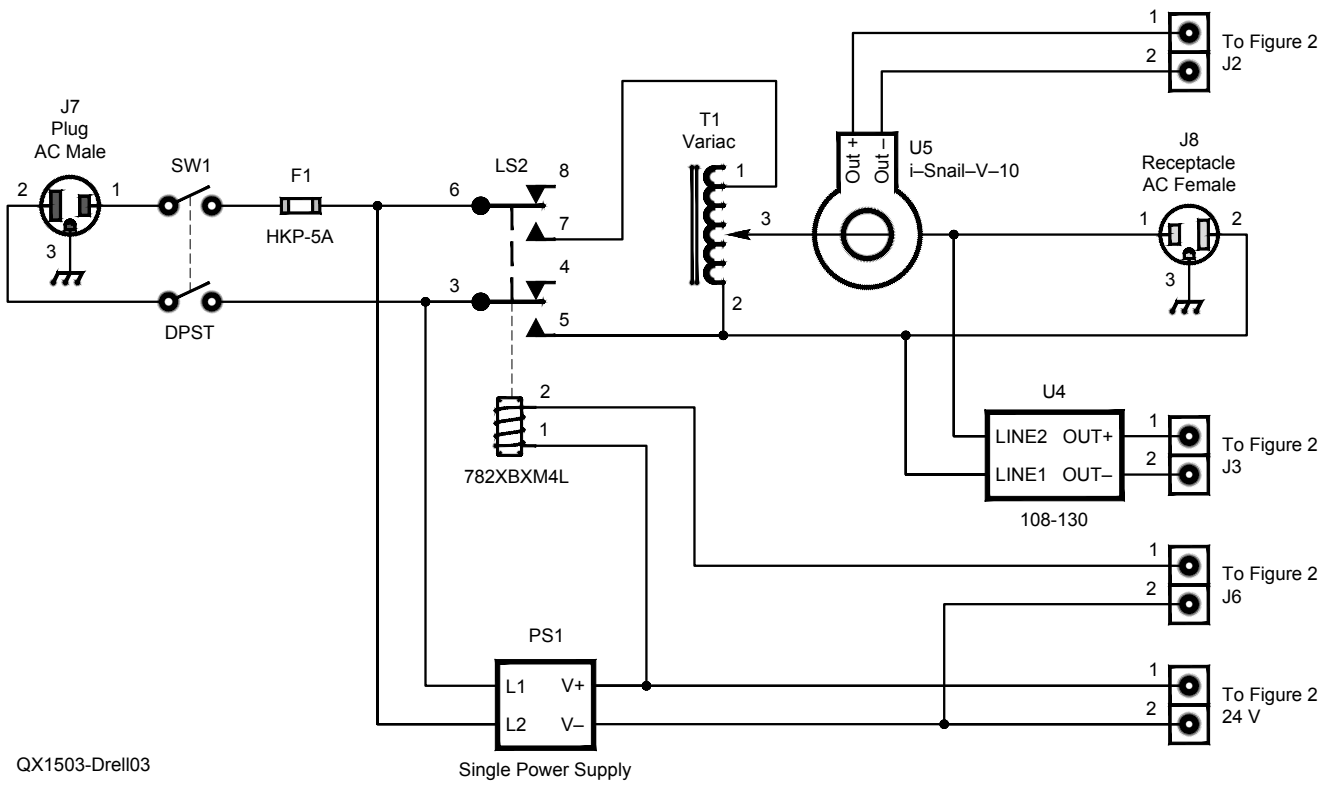

Figure 3 — Here is the chassis wiring diagram.

Table 2
Autotransformer Parts List

Reference	*Quantity*	*Description*	*Source*
U2	1	"Screw" Protoshield	ADAFruit
U3	1	LM7812C/TO220	Newark
U6	1	L78M05/TO220	Newark
LS2	1	DPDT Relay; 24 V DC Coil; Magnacraft with socket 782XBXBM4L	
U1	1	24 × 2 LCD display; Luminix	
D2	1	1N4001 Diode	
Q1	1	2N2222A Transistor	
R1	1	10 kΩ single turn trim pot	
R3	1	2.2 kΩ ¼ W	
R4	1	Not Used In This Version	
D1	1	LED Not Used In This Version	
F1	1	HKP Fuse Holder with 3AG 5 A Fuse	
LS1	1	DPDT Relay, 5 V DC Coil, Miniature (Omiron)	
PS1	1	LPS55 power supply	Newark
U5	1	i-snail –V-10 Current Transducer	ELKOR
U4	1	108 to 130 V Voltage Transducer (AAC)	American Aerospace Controls, Inc
T1	1	Variac 291	Staco
SW1	1	DPDT miniature toggle switch	
J7	1	Input Power Connector 120 V AC @ 10 A	
J8	1	Output Power Connector; 120 V AC @ 10 A	

Construction

The unit is assembled in a standard Bud cabinet with components placed as shown in Figure 4. The only critical wiring was the routing of the ribbon cable from the protoshield to the LCD readout. It needs to be dressed well away from the AC components as AC transients caused the readout to reset and display junk.

The LCD cover glass was made from clear acrylic material available at most home building supply houses. I cut it to size and used my router table to bevel the edges.

Parts layout in the enclosure is not critical and was guided by the fact that all of the controls were added on to the original "simple" autotransformer. The Arduino Uno and protoshield were mounted on the rear panel with the power supply, voltage sensor and power relay mounted on the bottom plate. The current transducer is mounted on the front panel next to the display.

Figure 5 is a close up of the protoshield showing parts layout. The 12 V converter is on the right side while the 5 V converter for the backlight is on the left side.

Ted Drell was first licensed in 1954 as WN5HEU, and obtained his General class license 1 year later, when his call sign changed to W5HEU. His license expired in 1968 while he was overseas, and he was out of ham radio until 2006, when he earned his Amateur Extra class license.

Ted studied Electrical Engineering at Tulane University and The University Southwestern Louisiana. He also studied computer science at Houston Community College, where his primary focus was on "C" and machine language programming.

Ted worked for AT&T Long Lines, supporting carrier and microwave systems, and Bendix Field Engineering, supporting Stadan and Man Space Programs during the Apollo Space Program (Missions 7-11). He has also been self-employed in Metrology and as a design engineer and engineering manager in multiple areas of energy production.

Ted retired in 2007, although he continues with design and fabrication of one-off products for oil field production control systems. He enjoys antique radios (especially Collins, Drake, National, and other brands from that era) and using microprocessors.

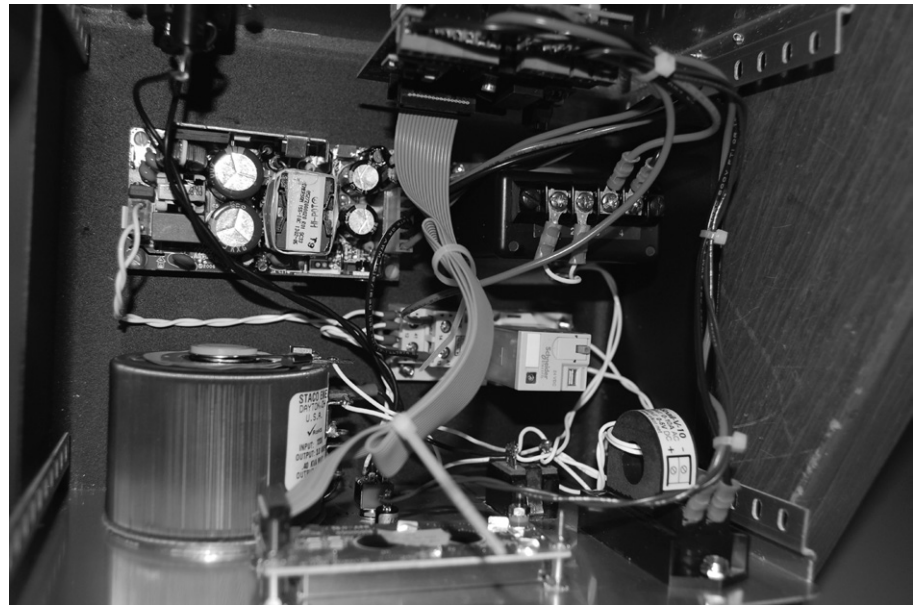

Figure 4 — Top View showing the parts layout.

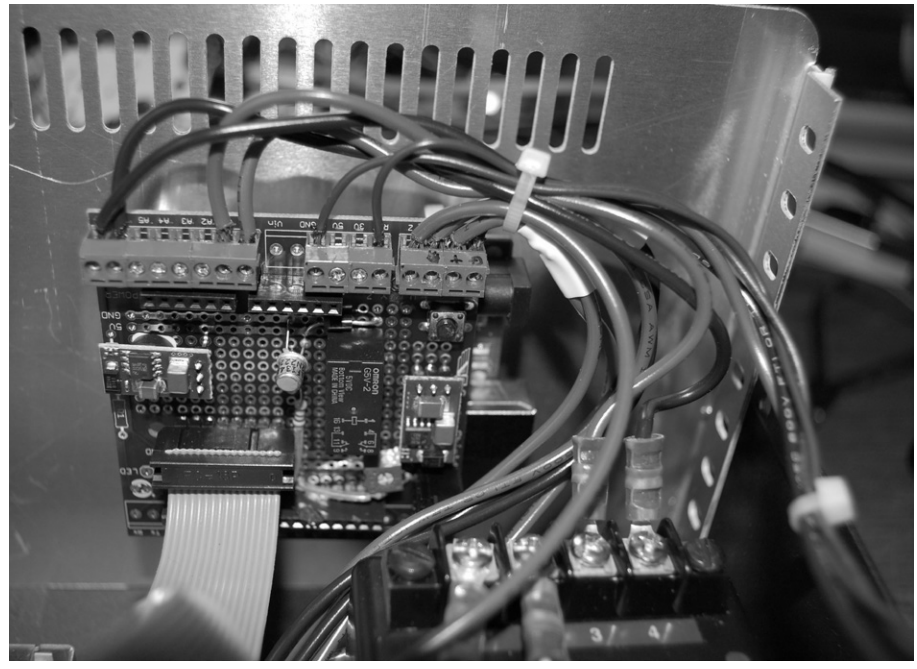

Figure 5 — This is a close-up view of the Arduino protoshield.