by Robert Lang

High Stepping Automatic Coil Winder

An Intro to Precision Positioning

This Month's Projects

Coil Winder 40
AD/DA Converter . . 48
Spoke Signals 52

The Fuzzball Rating System

To find out the level of difficulty for each of these projects, turn to Fuzzball for the answers.

The scale is from 1-4, with four Fuzzballs being the more difficult or advanced projects. Just look for the Fuzzballs in the opening header.

You'll also find information included in each article on any special tools or skills you'll need to complete the project.

Let the soldering begin!

Have you ever thought of a project where you needed to position an object with great precision? Maybe you wanted to build a precision milling machine or maybe a computer plotter. My introduction to precision positioning came about from a need to wind wire coils for solenoids. Coils are used for all kinds of electrical applications – from solenoids for electromechanical actuators to coils for electric motors.

My first coil winder (shown in Figure 1) was just a DC motor turning a coil form. It required a steady hand to lay down the wire on the coil. The winder really had no idea of how many windings were on the coil. What I wanted was an automatic coil winder that I could set up, walk away from, and when I came back, have a finished coil. Even if you do not need to wind coils, this project provides a good introduction to precision positioning using stepper motors under computer control.

The automatic coil winder in this article uses stepper motors to position the wire in increments of 0.000651 inch. The motors are controlled through a personal computer's (PC) parallel interface. The Visual Basic software automatically estimates the wire length and stops the winding process when the desired number of windings is reached. The coils are wound without any human interaction except to mount the coil form and to remove the finished coil. The automatic winder is useful in winding a number of identical coils.

Stepper Motors

Stepper motors are well-equipped for this project where rotation and lateral motion 'have to be controlled with great precision. There are many stepper motor controllers available for doing computer numerical control (CNC) machining; however, for this project, I decided to build a simplified stepper motor controller from scratch. I decided to use a PC as the "brains" of the controller and to use the PC parallel port to carry the control signals. Visual Basic (VB) was the language used to program the PC.

A unipolor stepper motor has two center tapped coils. A unipolar motor has five, six, or eight leads. The motor I chose has six leads. I identified the two center taps by measuring resistance with a suitable ohmmeter. The resistance from a terminal to the center tap is half the resistance of the two terminals of a coil. With the center taps identified as BLUE&WHITE and BLACK&WHITE, the remaining wires must be the coil terminal leads.

For the motor I chose, each energizing of a coil should rotate the shaft 7.5 degrees.

Figure 1. First Attempt at Automatic COILWINDER.

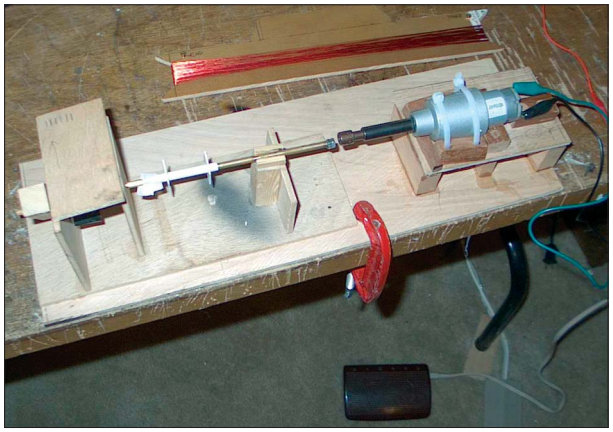
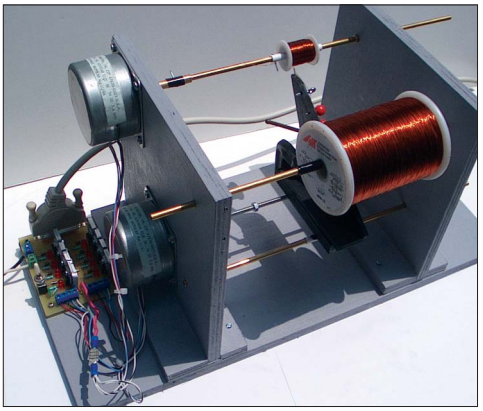


Figure 2. Final Completed Automatic COILWINDER.



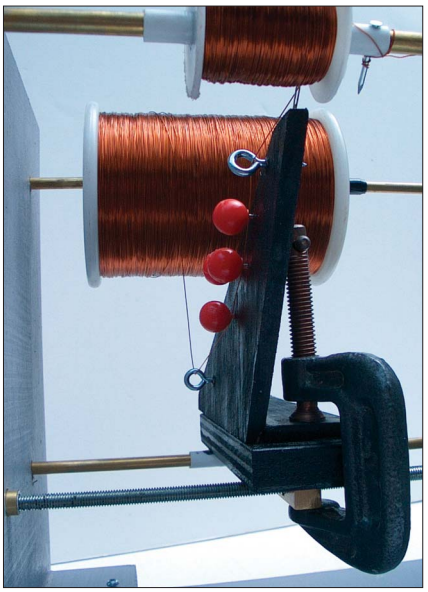


Figure 3. Finished Carriage.

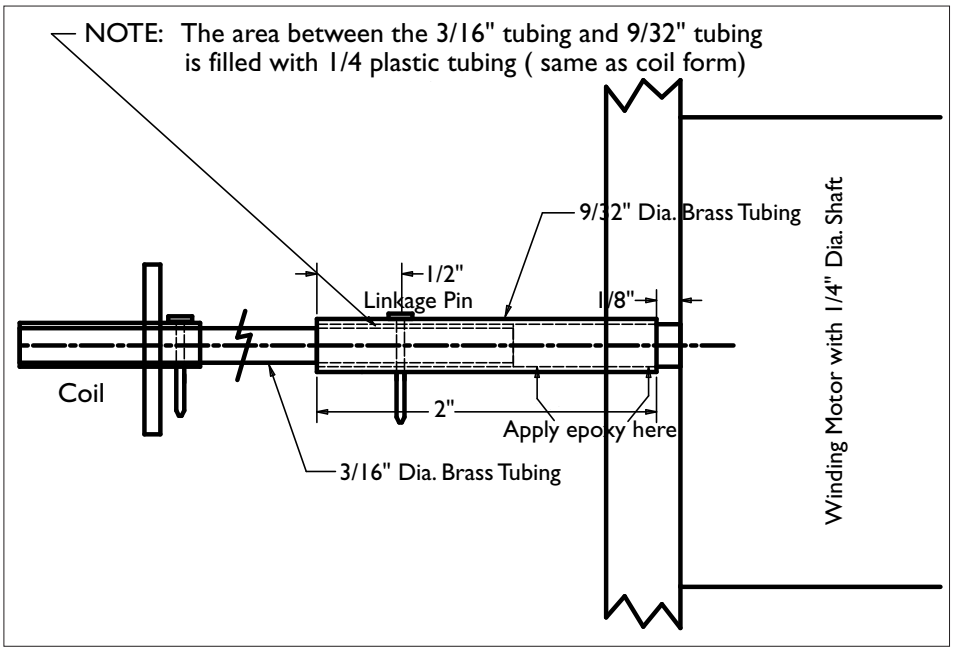


Figure 4. Detail of Shaft Coupling.

Current flowing through a coil produces a magnet field which attracts a permanent magnet rotor which is connected to the shaft of the motor. By trial-and-error grounding of the terminal leads while five volts was applied to the two center taps, the proper sequence for clockwise rotation was found to be RED, BLACK, BLUE, and WHITE. Reversing the energizing order causes counter-clockwise rotation.

Two modes of motor operation are user selectable in the software. The default mode is WAVE, which means that each of the four motor coils is energized one at a time, in sequence. This mode draws minimum current. A second mode is HI-TORQUE. In this mode, two coils are energized at a time. This mode requires twice the current of WAVE mode; however, it does produce more torque in the motors.

I thought the COILWINDER would require two stepper motors. One motor controls the rotation of the coil and is called the winding motor. The winding motor turns the coil directly. The other motor controls the movement of a small carriage feeding the wire back and forth across the rotating coil form. This motor is called the carriage motor. The carriage motor drives a lead screw which translates the rotational motion of the motor to linear motion of the carriage.

Building the Hardware

The completed unit is shown in Figure 2. If you decide to use another stepper motor, just make the appropriate changes in the software for STEPANGLE and MAX_STEPS. If a threaded rod other than 32 TPI is used, change LEADSCREW_TPI appropriately. To drive the carriage, a "connector nut" for the leadscrew should be purchased or made. I made a connector nut from a one-inch piece of 9/32" square brass stock.

I drilled and tapped a #32 thread through the stock. Figure 3 shows that the carriage is made up of three pieces that are clamped together using a one-inch "C"

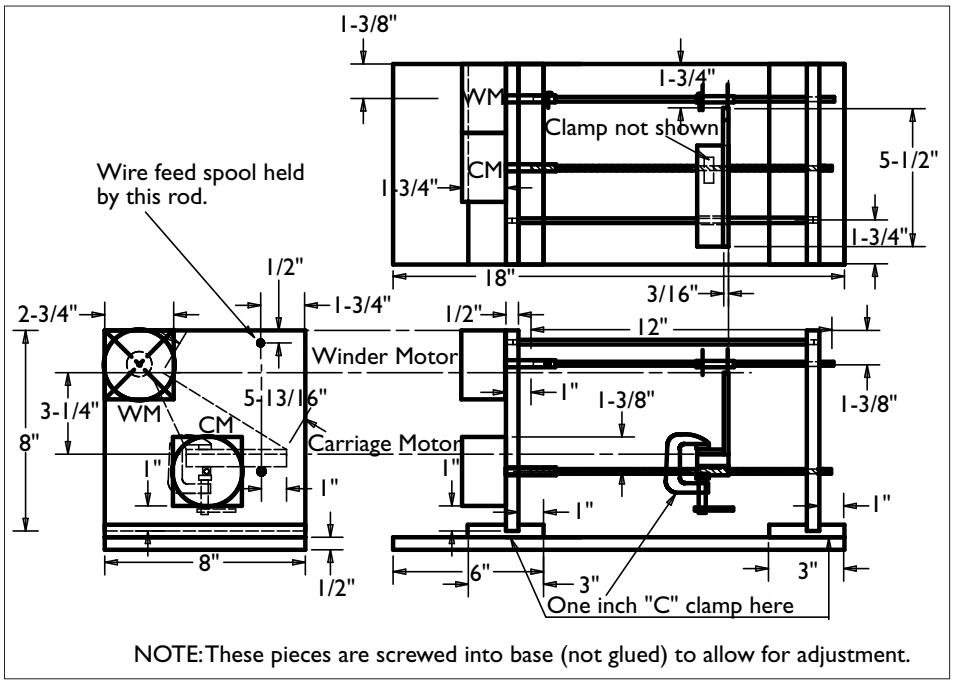


Figure 5. COILWINDER Drawing.

clamp: the connector nut, the carriage unit, and the wire feed unit. The reason for the clamp is to allow removal/adjustment of the carriage without the need to run the connector nut completely off the threaded rod. The joining of the leadscrew and winder rod to the stepper motors was done as shown in Figure 4.

The winder frame was built from 1/2" inch birch plywood. A detailed plan is shown in Figure 5. The assembly is straightforward, but the builder should be careful in drilling the holes for the motor shafts to ensure that they are parallel. The tensioning pins on the carriage pins are used to increase the tension on the winding spool. The number of tensioning pins used is dependent on the wire gauge. The frame end supports are adjustable which should allow for winding up to 11" coils.

The motor driver circuit is relatively simple and was initially built on a breadboard, as shown in Figure 6. While the parallel port does provide 0-5 volt signals, it does not provide enough current to drive the motors. For this reason, a separate power supply is needed for the motors. In the case of five volt motors I chose, it is a five-volt power supply capable of supplying 2.8 amps at five volts. If you are using a stepper motor that uses a different voltage, you must provide an adequate power supply.

The circuit diagram is shown in Figure 7. Figure 8 shows the pinout for the PC parallel port. After testing on the breadboard, the driver circuit was built on a 3 x 4 inch printed circuit board shown in Figure 9. Heatsinks were later added to the MOSFETs to keep them cool during extra long coil winds. The layout of the circuit board is available at **www.nutsvolts.com**

Parallel Interface

Writing programs to talk with the PC parallel port was pretty easy in the old DOS days and in Win95/98, too. With the new era of NT-clone operating systems like WIN NT4, WIN2000, and WINXP, all this simplicity goes away. Trying to run a computer program accessing the parallel port on a WINXP computer will give a "PRIVILEGED INSTRUCTION EXCEPTION" error message. Being relatively secure operating systems, Windows NT/2000/XP assign some privileges and restrictions to different types of programs. It classifies all the programs into two categories: User

Figure 7. Circuit Diagram for Motor Driver.

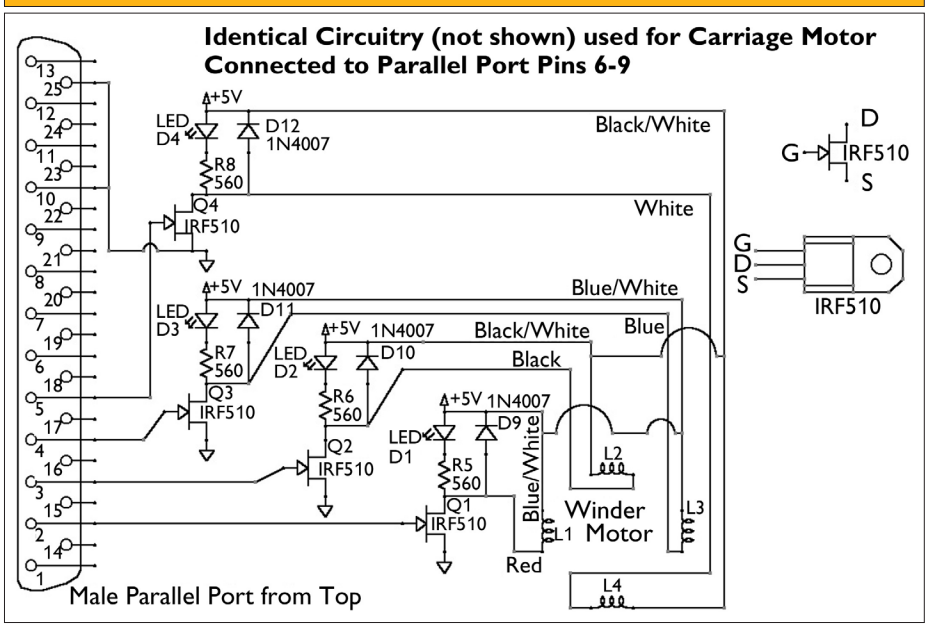
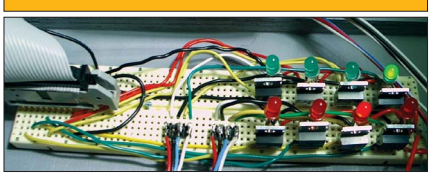


Figure 6. Completed Motor Driver Circuit.



mode and Kernel mode.

The programs that users generally write fall into the user mode category which does not allow direct access to the parallel port. The workaround for the above problem is to write a kernel mode driver capable of reading and writing data to the parallel port and let the user mode program communicate with the driver. Fortunately, someone else has already incorporated the driver into a dynamic link library called INPOUT32.DLL, and it is available for free at **www.logix4u.net/inpout32.htm**

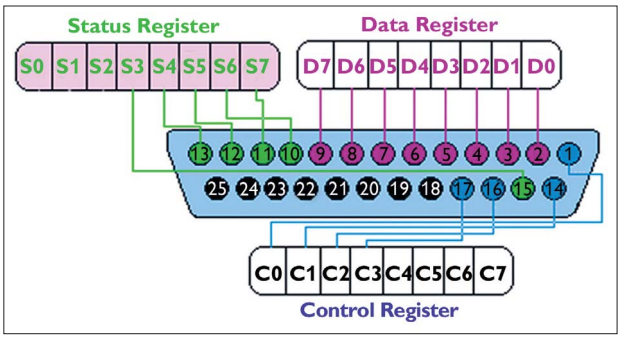


Figure 8. Parallel Port Pinout.

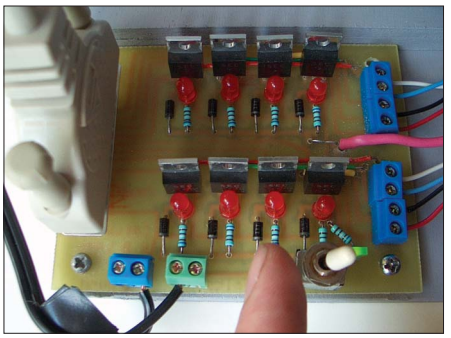


Figure 9. Printed Circuit Driver Board.

The outstanding feature of INPOUT32.DLL is that it works with all the windows versions without any modification in the user code or the DLL itself. The DLL will check the operating system version when functions are called, and if the operating system is WIN9X, the DLL will use INP and OUT functions for reading/writing to the parallel port. On the other hand, if the operating system is WIN NT, 2000, or XP, it will install a kernel mode driver, HWINTERFACE.SYS, and talk to the parallel port through that driver. The user code will not be aware of the OS version on which it is running.

The statements that make the parallel port input (INP) and output (OUT) functions available to the VB program are:

```
Public Declare Function INP Lib "inpout32.dll" _
Alias "Inp32" (ByVal PORTADDRESS As Integer) As Integer
Public Declare Sub OUT Lib "inpout32.dll" _
Alias "Out32" (ByVal PORTADDRESS As Integer, ByVal VALUE As
Integer)
```

The DECLARE statements simply tell VB that the programmer wants to use a function called OUT in his program that is known as Out32 in the INPOUT32.DLL located in the WINDOWS/SYSTEM directory. The OUT function takes an argument called PORTADDRESS which is the address of the parallel port and an argument called VALUE which is the data to be written to the parallel port. Program timing is controlled by a call to the WINDOWS application programming interface SLEEP function using another DECLARE statement.

Designing the Software

The software was written in Visual Basic for an IBM compatible PC and runs on all Windows operating systems. The actual winding of the coil begins when the BUILD COIL command button is pressed. For the actual winding of the coil software, I decided to use programming consisting of "state machines." The use of state machines for each function enables the software to be small yet respond quickly to real-world events giving great precision to the motor control. Each function consists of a number of states. Each individual state requires only a few instructions to complete before passing program control to the next function. This process of breaking functions up into small states is a superior formalism for modeling dynamic real-time behavior. The program consists of four functions. Each function has multiple states, as shown in Table 1.

The functions are called sequentially and the states repeat the same actions until the state is changed. For example, once the winder motor is put into FORWARD state, it will continue winding on each execution until some new condition changes the state. Each state is a separate

Function	States				
PROCESS_ COMMANDS	GETSTART Allows user to manually position carriage to home position using L, R, P, and H keys	INITHOME Reads S or C key	STARTWIND Starts the winding process. Reads the P and T keys.		
CONTROL_ CARRIAGE	FORWARD Moves carriage motor one step forward	BACKWARD Moves carriage motor one step backward	PAUSE Output 0x00 to motors		
CONTROL_ WINDER	INITHOME Turns off winder and carriage motors	FORWARD Moves coil winder motor one revolution forward	PAUSE Calls CONTROL_ CARRIAGE if in JOGMODE otherwise output 0x00 to motors		
DISPLAY_ RESULTS	INIT Requests user to position carriage to home using L, R, and H keys.	HOMESTATUS Displays state of move to home	START Requests user to enter S to start, C to continue, P to pause, or T to terminate	WINDSTATUS Displays wind count and carriage steps from home	NOUPDATE Leaves current display unchanged

TABLE 1. Program Functions and States.

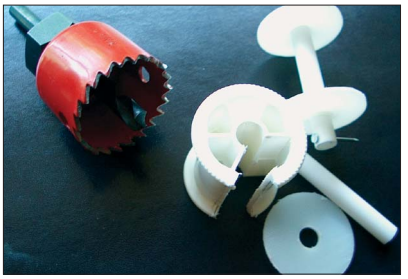


Figure 10. Coil Form Pieces and Tools.

For the case of #32 magnet wire, we get:

Round(0.007951 inch /0.000651 inches/thread) = Round(12.21) = Steps of carriage motor = 12. = STEPSPERTURN

CASE in the Visual Basic SELECT CASE statement. The CONTROL_CARRIAGE function is a sub-function of the CONTROL_WINDER function.

One of the key parameters in the program is INCH_PER_STEP. This parameter defines how far the carriage moves laterally on each step of the carriage motor. It is based on the steps per revolution of the carriage motor and the threads per inch of the carriage lead screw. For the motor and lead screw I chose, the STEPSPERREV=48 and the LEADSCREW_TPI=32. INCH_PER_STEP is calculated as follows:

INCH_PER_STEP = 1.000 / (LEADSCREW_TPI * STEPSPERREV) = 0.000651 inch per step

The main problem in the coil winder is determining how many revolutions the carriage motor should make per revolution of the winder motor. The value is calculated as follows:

One revolution of winder motor = WIRE_DIA = Steps of carriage motor * INCH_PER_STEP

Forty-eight steps are required for one revolution of the winder motor, therefore an output step to the carriage motor should occur every four winder steps=STEP_RATIO. If this is a non-integer value, it is rounded. In no case are more carriage steps output than STEPSPERTURN per winder revolution.

Since the coil winder is designed to wind coils with multiple layers of windings, a second problem was to determine when the carriage motor should reverse direction to lay down the next layer of wire. This was accomplished by using the SLIDESTEP counter and comparing it to the maximum slidesteps per layer, MAX_SLIDESTEPS. For example, for a one-inch coil using #32 magnet wire:

MAX_SLIDESTEPS = Int(WINDS_PER_LAYER *STEPSPERTURN) = 1512

Where WINDS_PER_LAYER = Round(COIL_LENGTH/ WIRE_DIA) = Round(125.77)=126

In theory, the calculations above should result in a per-

Figure 11. Program User Interface.

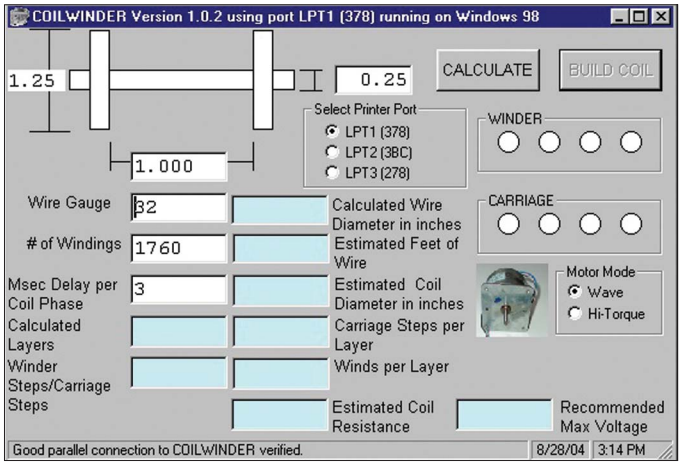
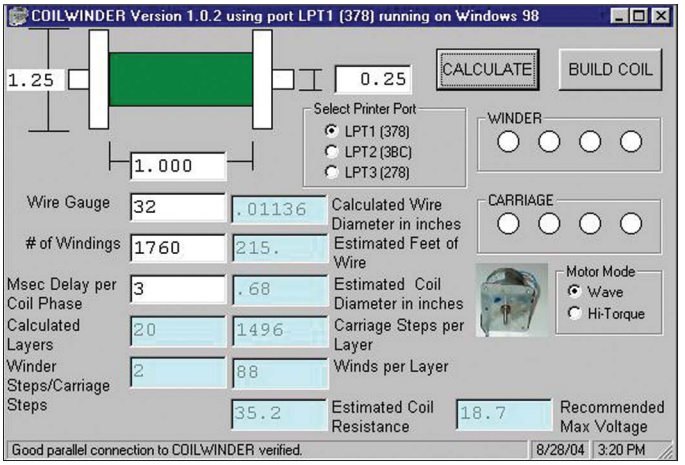


Figure 12. User Interface After CALCULATE Button.



fect coil. Unfortunately, the wire diameters in the code are based on wire gauges that do not include the insulation thickness. When I measured my #32 wire with insulation, I got a diameter of 0.01136 instead of the specified 0.0080. I modified the program to use the correct .01136 diameter for my #32 wire, but for other wire gauges, the user will have to modify the wire gauge/wire diameter lookup table after measuring their wire diameter with insulation. The VISUAL BASIC source code and installation package for the COILWINDER program are available from **www.nutsvolts.com**

Using the System

To wind a coil, first a coil form is needed. I built my forms from 1/4" styrene plastic tube and 0.06" thick sheet styrene, both available from your local hobby shop. I used a 1-1/4" hole cutter to cut out the round pieces of the coil spool. It also makes a nice 1/4" hole in the center for the coil tube. The spool end pieces are glued onto the plastic tubing with styrene plastic glue using an alignment tool fabricated from a thread spool. Small holes are drilled in the end piece for the wire feed and wire exit, and in the tube for the coil pin. The tools and pieces are shown in Figure 10.

The drive pin from the winder shaft is removed, freeing the winder shaft. The form is placed on the winder shaft and secured from free rotation with the coil pin. The winder shaft is reattached to the motor with the driver pin. Wire is fed from the supply spool, through the tensioning pins, through the needle's eye, onto the coil form, and out the coil feed hole. We are now ready to wind!

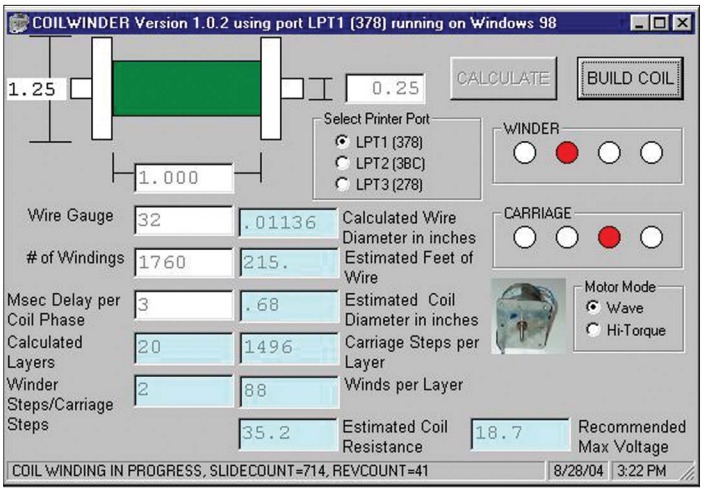


Figure 13. User Interface During Winding Process.

When the program is started, it first checks to see if the COILWINDER is connected to the parallel port. The user

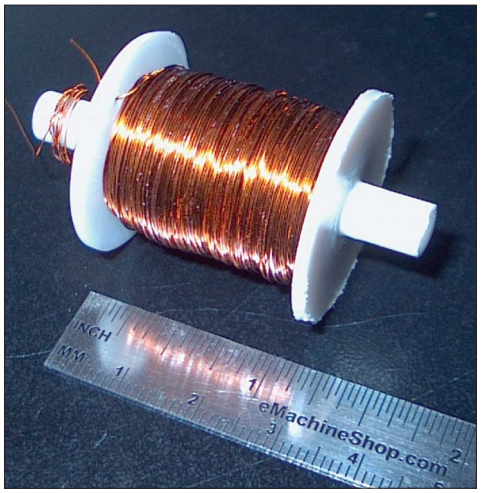


Figure 14. Completed Coil.

Sources and References

Source 1
Magnetic Gun Club
http://mgc314.home.comcast.net/index.htm

Source 2
Eddy Current Testing Information
www.ndted.org/EducationResources/CommunityCollege/EddyCurrents/cc_ec_index.htm

Source 3
Tesla coil design
www.noonco.com/tesla/

Source 4 — www.allelectronics.com

Source 5 — www.jameco.com

interface at program startup is shown in Figure 11. The user can enter wire gauge, coil diameter, coil length, the number of coil windings, and the winding delay. The user next presses the CALCULATE command button and the program calculates the estimated feet of wire required, the number of winding layers, the wire diameter, the outer diameter of the coil, and the number of carriage steps per layer.

Next, the program calculates the estimated coil resistance and a recommended maximum voltage based on the resistance and the current limit of the wire. The program also calculates the ratio of winder steps to carriage steps using the equations above, as shown in Figure 12. The number of feet calculated is accurate to within about 10%. The program recalculates the number of windings based on complete wire layers. If you want the lead and exit wire to be on the same end of the coil, make sure the number of layers is even. For opposite ends, use an odd number of layers.

When the BUILD_COIL command button is pressed, the STATUSBAR first instructs the user to position the carriage to the "start winding" point by using the "L" and "R" keys. The "H" key is used to indicate that the carriage has reached home position. Next, the user is instructed to hit "S" to start winding the coil. When the "S" key is hit, the winding starts and the STATUSBAR continuously updates the carriage steps per layer and the number of windings completed, as shown in Figure 13. The "T" key can be used to terminate the winding at any time. When the winding is complete, the motors stop and the "L" and "R" keys are activated for initializing the next coil. During the winding process, the "P" key can be used to pause the winding and the "C" key can be used to continue.

In order to maximize the speed of the winding, whenever a motor is started, there is a 28-step motor startup sequence where the first step takes ~221 times the input millisecond delay. The delay is reduced exponentially until the 28th step takes the input millisecond delay. This was done to allow the motors to start up gradually to prevent motor "chatter" that occurs when stepper motors are cycled at their maximum rate from a dead stop. If chatter still occurs as the motors get up to speed, then the input millisecond delay should be increased.

Parts List and Suppliers

Quantity	Part ID	Description	Source/Part #	Cost	Total Cost
2	M1, M2	5 volt, 1.38 amp, 48 step/rev, 0.25" diameter x 1" shaft, Stepper motor	Source 6 SMT-62	$3.75	$7.50
1	T1	#10 x 32 threads per inch x 12" rod and connector nut	Source 8	$.96	$.96
3	BS1-3	3/16" x 12" diameter brass tubing	Source 9	$.75	$2.25
1	BL1	9/32" x 12" diameter brass tubing	Source 9	$1.19	$1.19
1	PT1	1/4" x 12" styrene plastic tubing	Source 9	$.75	$.75
1	PS1	.06" x 2" x 6" styrene plastic sheet	Source 9	$.75	$.75
8	Q1-Q8	N channel, 100V, 5.6 amp IRF510 MOSFET (#209234CL)	Source 7 209234CL	$.46	$3.68
1	DB1	DB-25 connector (male)	Source 7 15122CL	$.65	$.65
8	R1-R9	620 ohm, 1/4 watt resistor	Source 7 31376CL	$.01	$.09
8	D9-D16	1N4007 diode	Source 7 36011CL	$.04	$.32
8	D1-D8, D17	LEDs	Source 7 34606CL	$.15	$1.35
1	P1	5 volt/4 amp DC power supply	Source 6 15347-PS	$10.95	$10.95
1	SW1	SPST sub-miniature power switch	Source 7 72160CL	$1.29	$1.29
1	W1	1/2" x 18" x 8" birch plywood	Source 8		
2	W2-3	1/2" x 8" x 8" birch plywood	Source 8		
2	W3-4	1/2" x 3" x 8" birch plywood	Source 8		
1	W6	1/2" x 4" x 1.25" birch plywood (carriage base)	Source 8		
1	W7	3/16" x 4" x 8" birch plywood (wire guide)	Source 8		
1	BB1	Printed Circuit Board Materials or breadboard	Source 7		

Summing it Up

An interesting project of an automatic coil winder has been constructed for less than $35.00. The unit can automatically wind high quality coils as shown in Figure 14. It took approximately 24 minutes to wind the 224 foot coil of # 32 magnet wire consisting of 1,800 windings, using a delay of three milliseconds. This project provides an introduction to Visual Basic, the PC parallel port, and stepper motors. NV