Circuit Cellar / WIZnet
iMCU Design Contest 2010

Submission Deadline: June 30st, 2010

Complete Documentation Set

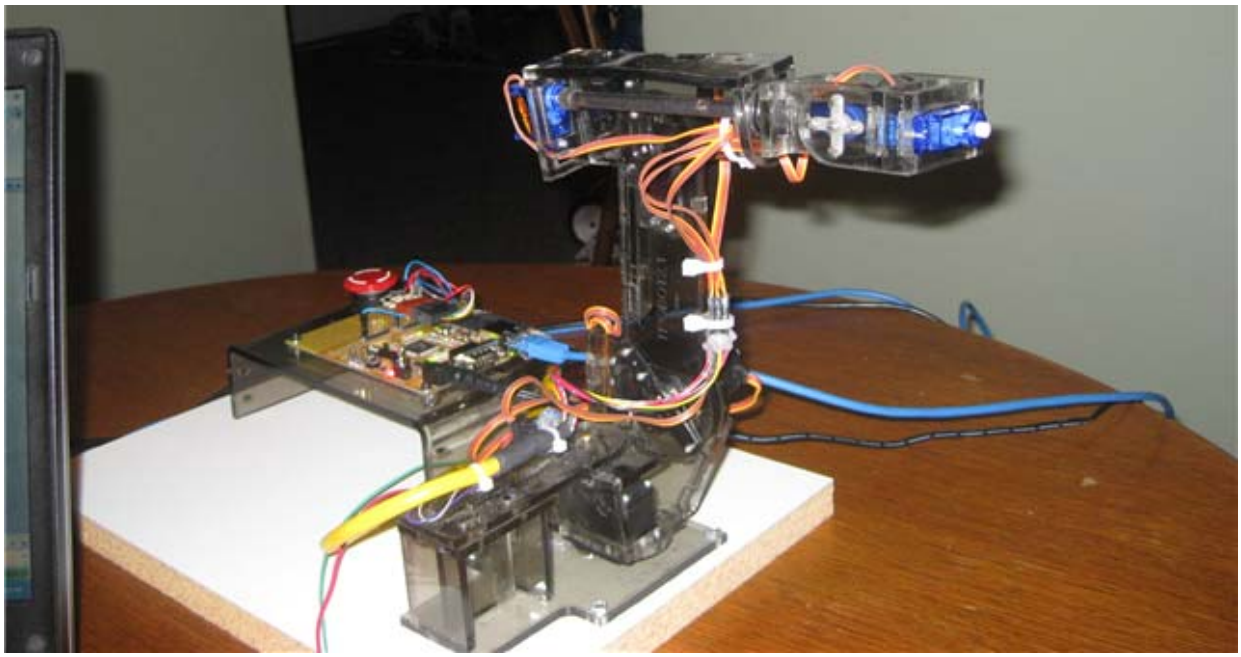
Project Number: **003184**

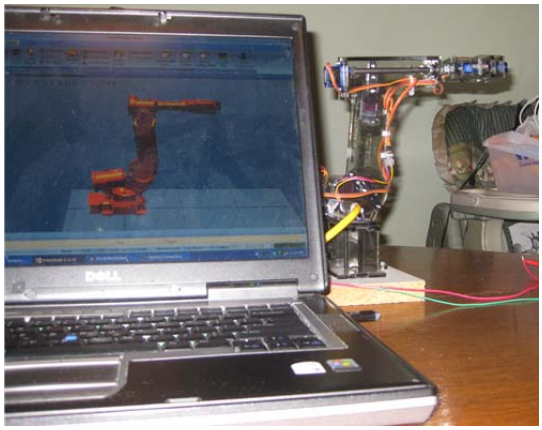
WIZnet Device: **W7100 EVB**

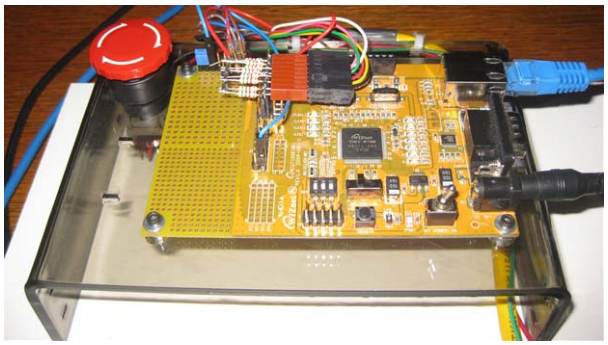
Project Name:
TROBOT 2.0
An ABB Robot Studio interface for a miniature articulated robot.

The Project:

The TROBOT 2.0 is a miniature 6-axis robot powered by small RC style servo motors. TROBOT utilizes a WizNet W7100 controller (Evaluation kit), which acts as servo controller interface between the robot and a PC running ABB's Robot Studio.







ABB's Robot Studio (www.robotstudio.com) is a powerful software tool used for simulation and offline programming of the full line of ABB Industrial robots. The 3D virtual environment makes it possible to setup, simulate, and program complex robot cells in a virtual world. ABB's Virtual Robot Controller technology can simulate almost all of the features of their robot programming language known as RAPID on an IRC5 robot controller.

This project adapts the Robot Studio application to control a custom made robot called simply the TROBOT. Watch the included video 'What Is TROBOT .wmv' for more information about the TROBOT 2.0 project.

Program Code:

The TROBOT application was written by starting out with the example TCP application "Loopback "provided in the WizNet evaluation kit software. This sample code allowed me to get the basic communication portion talking to the Robot Studio very quickly (less than one hour). The TCP app provided all of the resources needed to establish basic socket communication.

A similar application was written in Robot Studio to setup and establish socket communication on the robot side. Once basic communication was established, the WizNet board was able to echo back whatever data I sent to it from the Robot Studio application. An example of the RAPID code for Robot Studio is shown below.

The following routine is used to initialize the Socket Communication in the virtual robot:

```
PROC InitSocketComm(string IPAddress, num Port)
  SocketCreate socketTrobot;
  SocketConnect socketTrobot, IPAddress, Port \time:=15;
  TPWrite "Now connected to TROBOT....";
ENDPROC
```

Once Communication is established following code is used read the jont angles from the virtual robot, format them into a string and send them as a socket message.

```
PROC SendRobByteData()
  VAR string string1:=" ";
  jointpos1 :=  CJointT();
  string1:="all "+ValToStr(jointpos1.robax.rax_1)+" "
  +ValToStr(jointpos1.robax.rax_2)+" "
  +ValToStr(jointpos1.robax.rax_3)+" "
  +ValToStr(jointpos1.robax.rax_4)+" "
  +ValToStr(jointpos1.robax.rax_5)+" "
  +ValToStr(jointpos1.robax.rax_6);
  SocketSend socketTrobot,\Str:=string1;
ENDPROC
```

From that point it was just a matter of sending the correct data to the WizNet board and then turning the board into a PWM driver.

The WizNet application is setup to receive a socket message simply containing a string of data. Formatted as follows:

125:250:500:7530:12520:1220;

The message contains six colon (:) delimited 16-bit numbers followed by a semicolon (;). These numbers represent the values for the PWM outputs for each servo.

The following example code was used to extract the numbers from TCP string data and assign them top the variables used to set the PWM outputs for each servo motor.

```
/* extract first string from string sequence */
str1 = strtok(str, ":");
printf("%i: %s\n", x, str1);
x++;
str2 = strtok(NULL, ":");
printf("%i: %s\n", x, str2);
x++;
str3 = strtok(NULL, ":");
printf("%i: %s\n", x, str3);
x++;
str4 = strtok(NULL, ":");
printf("%i: %s\n", x, str4);
x++;
str5 = strtok(NULL, ":");
printf("%i: %s\n", x, str5);
x++;
str6 = strtok(NULL, ";");
printf("%i: %s\n", x, str6);
/*  Set variables equal to the token values  */
     axis1 = strtol (str1,NULL,10);
     axis2 = strtol (str2,NULL,10);
     axis3 = strtol (str3,NULL,10);
     axis4 = strtol (str4,NULL,10);
     axis5 = strtol (str5,NULL,10);
     axis6 = strtol (str6,NULL,10);
```

Once the data is received and assigned to variables the following simple loop is used to update the servo values. Because the time required to update all of the servos is less than 2mS, the following loop is only thing the WizNet processor is doing during this is time. After the update is complete the processor goes back to waiting for new socket data or for the next update servo time.

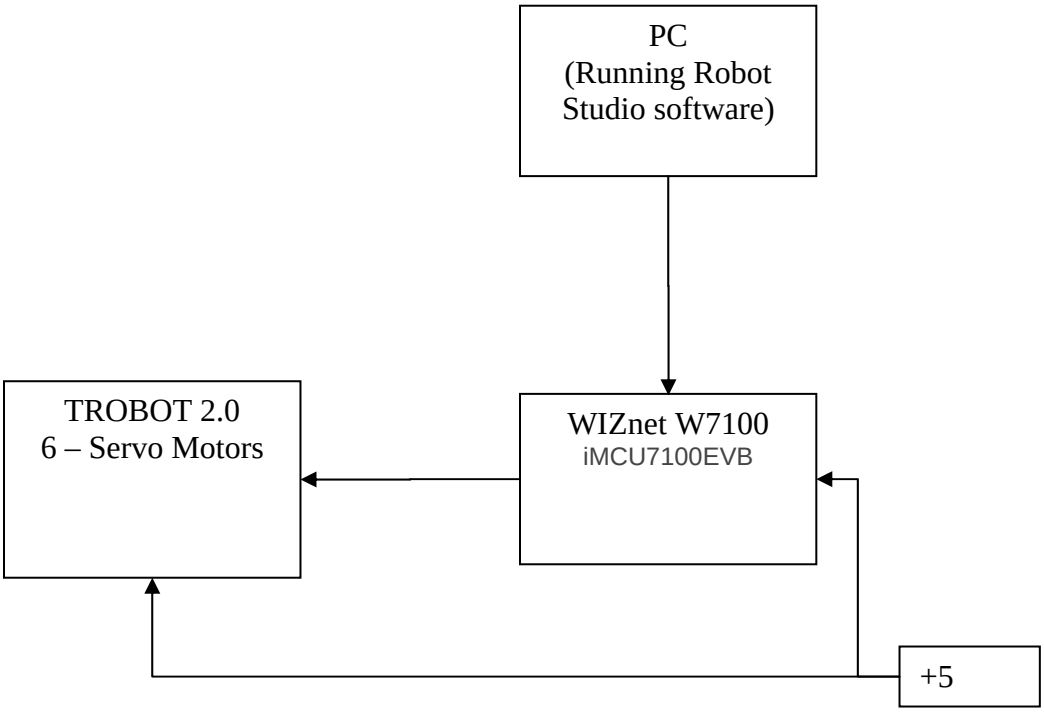
This PWM interrupt occurs every 20mS, it first sets all of the servo pwm's to high then loops WHILE all of the values are greater than the counter (at 1uS intervals) until the time expires for and clears the appropriate output when the counter exceeds the value for it's pwm.

```
SERVOS_B_ON();
TBdelay(12);
for(j = 100; j<(600); j++) {
     if (j>=(g_cur_PWM1)) SERVO_B0_OFF();
     if (j>=(g_cur_PWM2)) SERVO_B1_OFF();
     if (j>=(g_cur_PWM3)) SERVO_B2_OFF();
     if (j>=(g_cur_PWM4)) SERVO_B3_OFF();
     if (j>=(g_cur_PWM5)) SERVO_B4_OFF();
     if (j>=(g_cur_PWM6)) SERVO_B5_OFF();
}
```

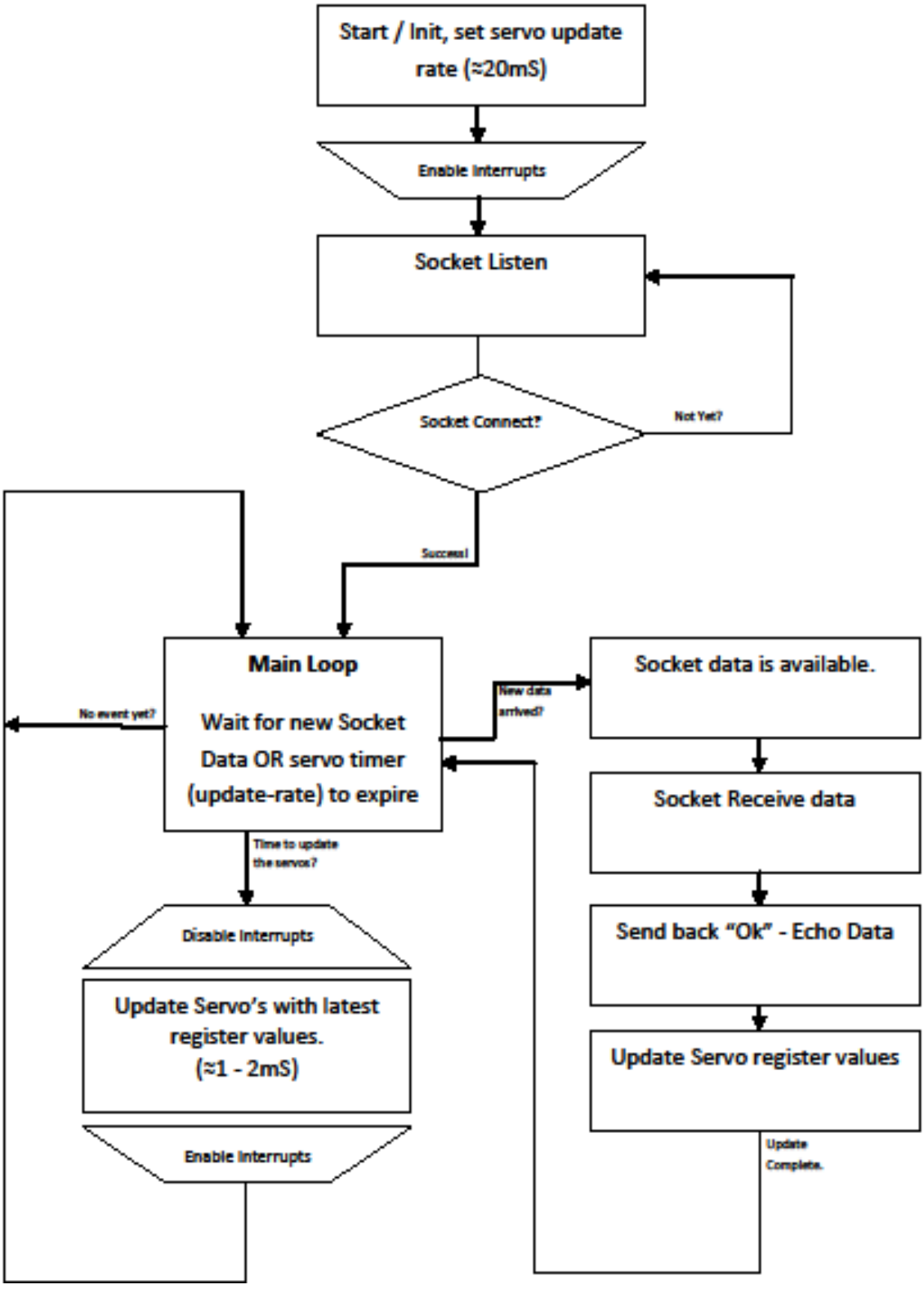
Block Diagram:

The Block diagram below demonstrates how communication is established and the function of the W7100 program with the Robot Studio application.

WIZnet W7100 Block Diagram



Wiznet 7100 Program Flow Chart



Schematic Diagram: