

Project Final Report

Network Controlled Camera Servo

April 27th 2007

Jack Wang, jack_h_wang@hotmail.com

Project Abstract

This project shows how to use an embedded java microcontroller to control a pan and tilt servo kit through a web browser. A USB camera is position on top of the pan and tilt kit, so the servos can position the camera's view. The user can have the freedom to "peep" around the through the camera's by controlling the servos over the internet from the other side of the world! Because the servo kit helps the application user to peep around, we call this tilt and pan servos "Peeping Mary".

Status

The final architecture implemented for this project has been slightly different from the original proposal. In the original proposal, the TStik has been designated as the servo driver which supposedly to send the pulse width modulated signal to the Pan and Tilt Servos.




However, because TStik cannot generate the pulse width resolution require by the Hitec HS422 servo, the R232 serial port ServoCenter board is added to the TStik to drive the servos. The TStik eventually hosts the servo control server "Peeping Mary Control Center" that the user can access through a web browser, so the server parses the use's command from the browser, and it sends the corresponding command packets from TStik to the servo driver board. In turn, the servo driver board drives the pan and tilt servos to position the attached camera.

Specification

Hardware

The major hardware components are as follows:



Component Name	Description	Vendor	Picture
TSTik	TStik is an embedded Java microcontroller	Systronix.com	
Tilt 400	Tilt 400 is the socket board for the TStik.	Systronix.com	
Pan and Tilt kit with HS422 Servos	Servo kit	Robotstore.com	
Servo Center 3.1 RS232	Servo control board with RS232 serial port	yostengineering.com	

Additional hardware parts:

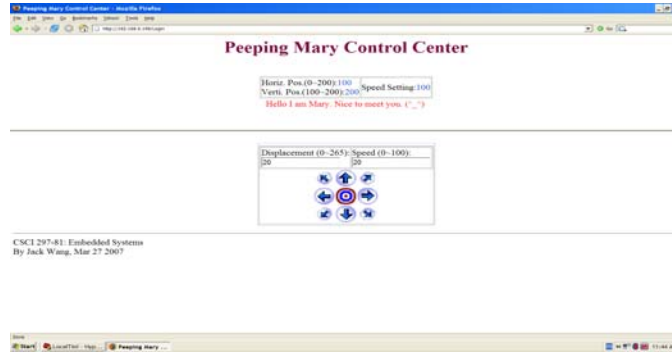
- A DHCP capable router
- 2 Ethernet cords
- 2 RS232 serial cables
- 2 12VDC 1000mA power adapters
- 1 USB webcam

Software Development Tools

The software development kits can be downloaded from <http://www.tstik.com/downloads.html>. Here you will find TStik2API and examples, Tilt400 API, and JavaKit400.bat, as well as configuration instructions. Also more of the java tools installation details can be found at <http://www.practicalembeddedjava.com/tools/javatools.html>. The information helps to setup the development environment. In particular, the Eclipse IDE is a very handy tool to use for the software development. I strongly recommend the use of this tool except the TINI Ant which has been working for this project. Instead using TINI Ant to compile and convert the Java source code, just use the ordinary command prompt and batch files. I always find it to be working well. Although Tstik.com recommends the use of TINI 1.16, I find it not compatible with the latest TstikAPI and Tilt400API. TINI1.17 has been working well with the latest TstikAPI and Tilt400API.

Application Software Requirement

Front end – the front end provides application user interface that the user can send in the servo's displacement increment and speed to the TINI server, and the result of the new position is sent back to the front end user. The design of the front end may look like in the following picture:



The front-end is divided into the top and the bottom frames, so the pictures of the arrow buttons in the bottom frame will not be re-loaded when the servo result is sent back to the user. Thus the user can have a faster response time.

Server

User posts three values to the PostServer: displacement, direction, and speed. The PostServer uses the `com.dalsemi.tininet.http.HTTPServer` class included in the TINI firmware download from <http://files.dalsemi.com/tini/index.html>, and the download file includes the HTTPServer example. The displacement parameter is to be broken down into horizontal dX and vertical dY displacement by the Pythagoras theorem, and each of the displacement value drives its corresponding server. The direction D is used to compute the final position of the servo. Positive and negative direction corresponds to up and down for the vertical servo, right and left for the horizontal servo respectively. Thus the final positions P for the servos are computed as follows:

$$P_{\text{horizontal}} = P_{\text{horizontal initial}} + D_{\text{horizontal}} * dX$$
$$P_{\text{vertical}} = P_{\text{vertical initial}} + D_{\text{vertical}} * dY$$

These two values will be sent to the servo driver and the user's browser.

Servo Driver

This is the software interface between the server on the TStik and the ServoCenter connected via RS232 serial4 port on the Tilt400. This software package provides the driver, command, and ServoCenter proxy components, so the server can send the commands to the driver, and the driver in turn drives the ServoCenter proxy using Uart class from then Systronix. The ServoCenter typically has the command packet in the following format:

240(0xF0)+BoardID
Command ID
Command Data
Command Data
Command Data
Checksum Value

Checksum value is ignored if it is zero. Suppose we want the servo on the S2 servo connector of the ServoCenter board 0 to move raw to position 135 at speed 34, we will have the following command packet:

240
16
2
135
34
0

For the complete list of commands, see ServoCenter 3.1 User's Manual from the Yost Engineering web site, or in the attachment. Thus the java code to send this packet can be as follows:

```
// open serial port on the Tilt400
Uart comPort = new Uart("serial4", 9600);
OutputStream output = comPort.getOutputStream();

// configure command packet
int commandPacket[6];
commandPacket[0]=240;
commandPacket[1]=16;
commandPacket[2]=2;
commandPacket[3]=135;
commandPacket[4]=34;
commandPacket[5]=0; //checksum is ignored

// sent the command to ServoCenter
for( int packetField=0;
    packetField<commandPacket.length;
    packetField++)
    output.write(commandPacket[packetField]);
output.flush();
output.close();
comPort.close();
```

Servo start position configuration

There is no need to reconfigure the start setting of the horizontal servo because the factory setting already centers the servo. However, for the vertical servo, its starting position needs to be reconfigured to position 200, so when the servo is powered up, it will initialize it self to position the camera looking up straight ahead. Use (0x06) set start command of the ServoCenter to achieve the configuration as in the following:

```
...
// configure command packet
int commandPacket[4];
commandPacket[0]=240; //server board ID +0xF0
commandPacket[1]=6; //corresponding command ID to set start position
commandPacket[2]=200; //start position
commandPacket[3]=0; //checksum is ignored

// sent the command to ServoCenter
for( int packetField=0;
    packetField<commandPacket.length;
    packetField++)
    output.write(commandPacket[packetField]);
output.flush();
...

```

Source Code Compilation and Conversion

Although Systronix recommends TINI Ant for compiling and converting the java source into a single TINI executable, I have found it not working. However the DOS command works well with the batch file. There are two steps in building the TINI file. The step one is to compile the java source into java class files. For example, the following command line compiles the entire java source from src and src/servoDriver subdirectory into the “build” subdirectory:

```
javac -source 1.2 -target 1.1 -bootclasspath %TINI_HOME%\bin\tiniclasses.jar -classpath
%JAVA_HOME%\lib\comm.jar;%TINI_HOME%\bin\modules.jar;%CLASSPATH% -d build src/servoDriver/*.java
src/*.java
```

The step two is to convert the compiled class files into one TINI file as follows:

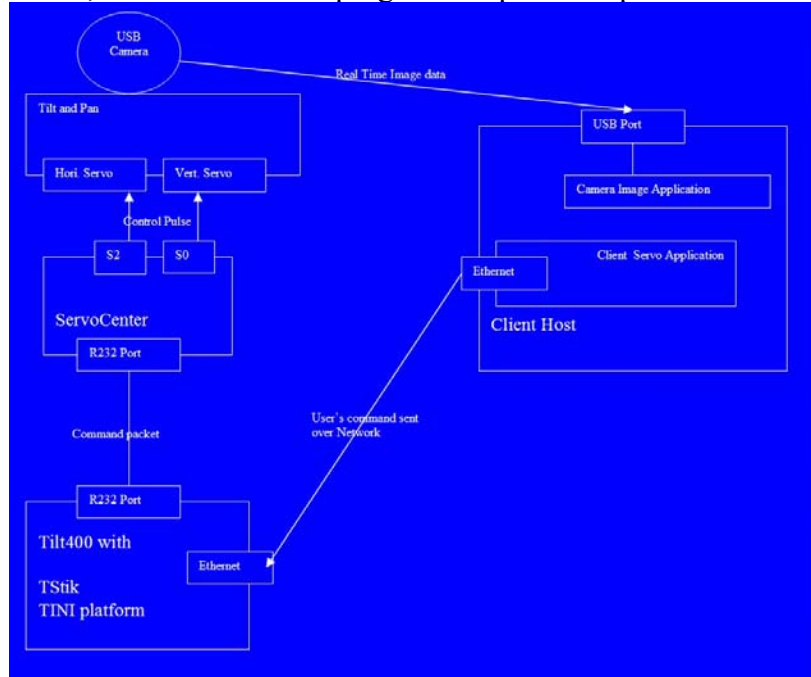
```
java -classpath %TINI_HOME%\bin\tini.jar;%CLASSPATH% BuildDependency -add HTTPSERVER -x
%TINI_HOME%\bin\owapi_dep.txt -p %TINI_HOME%\bin\modules.jar -m PostServer -f build -o
bin\PeepingMaryServer.tini -d %TINI_HOME%\bin\tini.db
```

This converts all of the class files from the “build” subdirectory to one PeepingMaryServer.tini file under “bin” subdirectory. In this example, the tini file will use PostServer as the main start-up class.

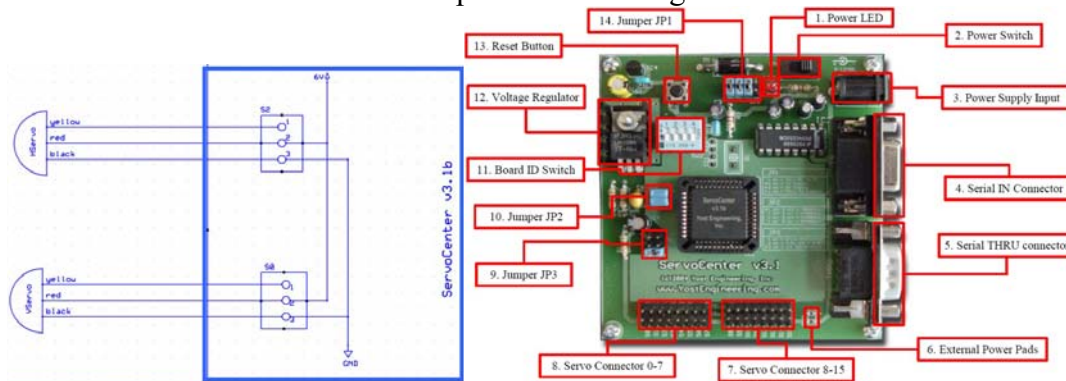
Implementation & Construction

System Design

The following is the finalized system design diagram for the “Peeping Mary” application. The servo control pulses sent from S0 and S2 port are completely abstracted by the ServoCenter, so we need not to program the pulses to position the servos.



The schematic diagram below shows how the servos are connected to the ServoCenter. The horizontal and vertical servos are connected to the servo connector S2 and S0 on the ServoCenter respectively (Label 8 on the lower right picture). Thus servo ID 2 and 0 are used in the ServoCenter command packets in driving the servos.

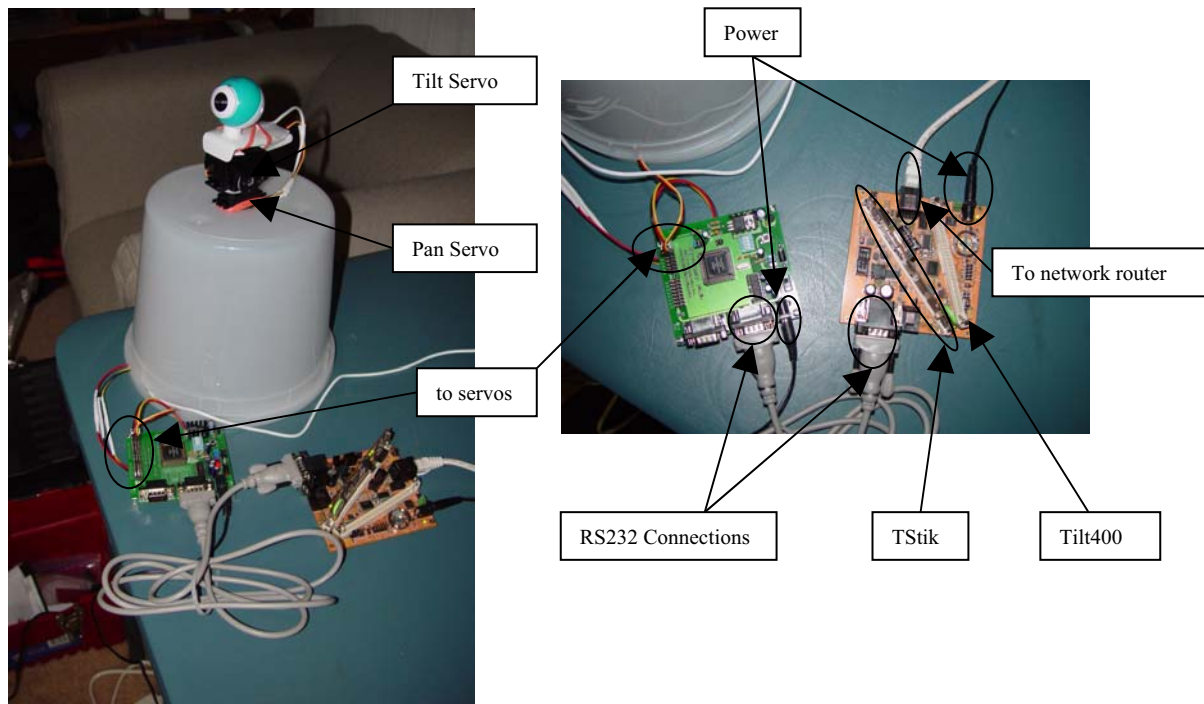


ServoCenter Configuration

The boardID switch is configured to 0 where all of the dip switches are set to off (Label 11 on the top right picture). Since the RS232 port on the Tilt400 can communicate at 38400bps, both positions on the JP2 are installed to extract the maximum performance (Label 10). Everything else is left with the factory setting for this project. Of course, you need to turn the power switch on (Label 2) to drive the servos.

Put Them Together

The following shows the actual hardware put together base on the system design. Both RS232 ports on the Tilt400 and ServoCenter are connected by the serial port cable.



The TStik is installed on the SIMM 72 socket of the Tilt400. Both boards are powered by the power jack from the power adapters. The tilt and pan servos are connected to the connectors on the ServoCenter board. The camera is attached on the tilt and pan kit. The Tilt400 also connects to the network router with the Ethernet cord.

Configure the network

The IP address is configured with the Javakit through the Tilt400's serial1 port connected to the developer's computer, while the Tilt400 is also connected to the DHCP enabled router. After logging into the slush, use `ipconfig -d` to lease the IP address from the router. In the slush, type `ipconfig` again to see the address assigned by the router. The router can further be configured to assign the same IP address to the MAC address of the Tilt400, so it will have a static private network address.

Deploying and Invoke TINI application

After the TINI file is built, the TINI file can be sent to the TStik through FTP. Once the TINI file is on the target TStik, the application can be invoked by the java command for example: `java PeepingMaryServer.tini`

Retrospective

Q. What were the important design decisions you need to make?

A. The most important design decision was to add the ServoCenter driver board that was not in the original design. When the Systronix tech support told me that the TStik would not generate the pulse width for less than 1ms, I immediately sought for the solution by ordering the servo driver board suggested by the tech support. Otherwise this project would not have been completed.

Q. How did they influence your project?

A. The original plan was to use the TStik and its SIMM 32 output pin to send the pulse width signal to the servos. According to the datasheet of the servo, a pulse width of 1.5ms would put the servo to the center position. I have tried the following code, and it would not work:

```
//This will not put the servo at the center position.  
BitPort bp = new BitPort(BitPort.Port5Bit4);  
bp.set();  
Thread.sleep(1,500); //sleep 1.5ms  
bp.clear();
```

Q. What did you learn from this project?

A. I have learned a great deal on the embedded java, and its applications are endless by the way. In particular, I have learned to control the servo through the internet. I have learned to set-up and configure my home network for the embedded java microcontroller, so it can be access it through the internet. Besides merely being a network servo driver, devices can be added to the application on the 1-wire and I2C serial interface of the Tilt400 board. So far, I have not used the TStik and Tilt400 to their fullest exciting potential. I appreciate what I have learned anyway.

Q. How would you do things differently if you knew what that before you started?

A. If I knew how to drive the network servo by TStik, I would not choose it for the project because it seems like a toy problem to me now. Instead, I would do the embedded camera using the TStik, or using the TStik to remotely check the stoves at home. Turn the stove off remotely, so it would not cause fire, or I may use it to set up a wireless home security system. The possibilities are endless.

Attachments

Included with this final report, all of the java program sources are under the “src” directory. The buildPeepingMaryServer.bat files has been named to buildPeepingMaryServer.bat_ to avoid being filtered by the internet mail server. Please rename them back at the mail recipient side. When all of necessary java development tools are properly installed, as described in the Systronix site, this batch file will build the java source code into PeepingMaryServer.tini file under the “bin” directory. In order to save file space, both the bin and build directories are empty. The ServoDataSheet subdirectory contains the datasheets for the HiTec servo, datasheet and user’s manual for the RS232 ServoCenter board. The project brochure, presentation, and Servo-ServoCenterInterface schematic diagrams are under the same directory with this final report.