

# RSS Reader v 0.1

Ravjot Singh Pasricha

## Overview

The RSS Reader is a simple embedded system utilizing a TINI microcontroller, an LCD screen, and a dual axis accelerometer as an input device. The TINI downloads RSS feeds and displays them on the LCD. The accelerometer is used to scroll text left and right by tilting it along the x axis. To cycle between feeds, the user can tilt along the y axis, up for the previous feed and down for the next feed.

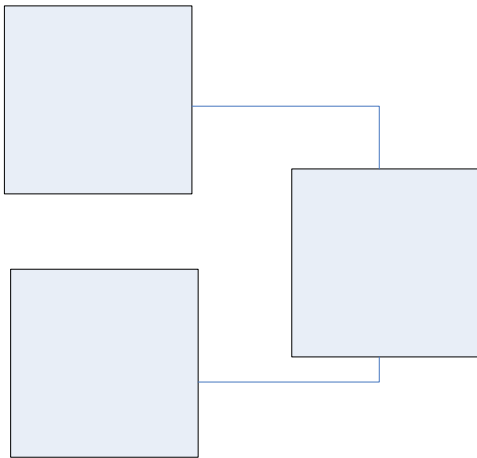


Figure 1 – Overview Block Diagram

## Hardware Detail

RSS Reader uses a Maxim TINI DS80C390 microcontroller as the heart of the system. The TINI was chosen for several reasons. First, TINI stands for Tiny INternet Interfaces because it is made specifically for projects which require network interfacing. Not only does the board feature a built in Ethernet jack, it also supports several TCP/IP protocols, most importantly (for this application) HTTP. The TINI is also suitable because it supports Java 1.1 through its internal JVM. This made coding the program quite easy because of Java's built in HTTP classes.

For the LCD, I chose Sparkfun.com's SerLCD v1.1. The SerLCD is a 16 character wide, 4 line LCD screen with a built in back light. Communication to the LCD is done over serial using UART. The LCD came assembled with serial support, and can just be sent ASCII characters over a serial UART connection and will display them. Figure 2 shows the commands supported by the LCD.

Display

CPU

Scroll  
Control

HD44780 Commands	
Clear Display	0x01
Move cursor right one	0x14
Move cursor left one	0x10
Scroll right	0x1C
Scroll left	0x18
Turn visual display on	0x0C
Turn visual display off	0x08
Underline cursor on	0x0E
Underline cursor off	0x0C
Blinking box cursor on	0x0D
Blinking box cursor off	0x0C
Set cursor position	0x80 +

Figure 2 - HD44780 Commands

The accelerometer used in this project is the Analog Devices ADXL202E. This is a low cost dual axis accelerometer with duty cycle output.

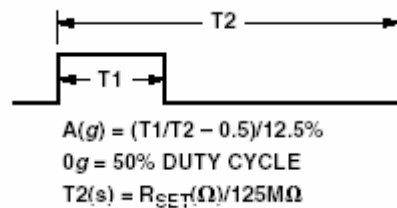
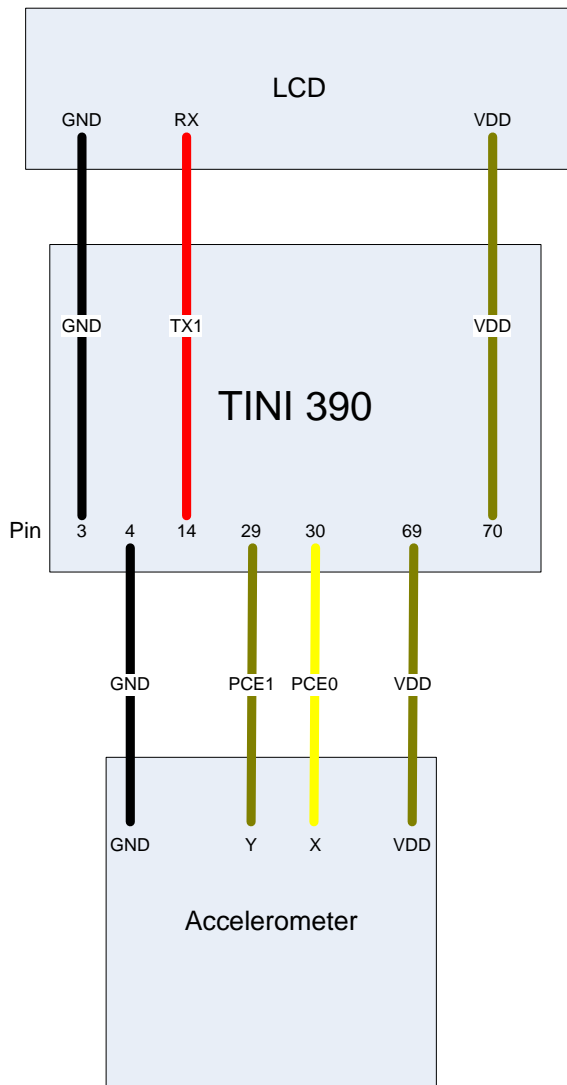


Figure 3 - Typical ADXL202E Output Duty Cycle

The acceleration in terms of gravity of the accelerometer is calculated by using the formula  $A(g) = (T1/T2 - 0.5)/0.125$ , where T1 is the amount of time that the “ON” signal is outputted in a given period, T2.

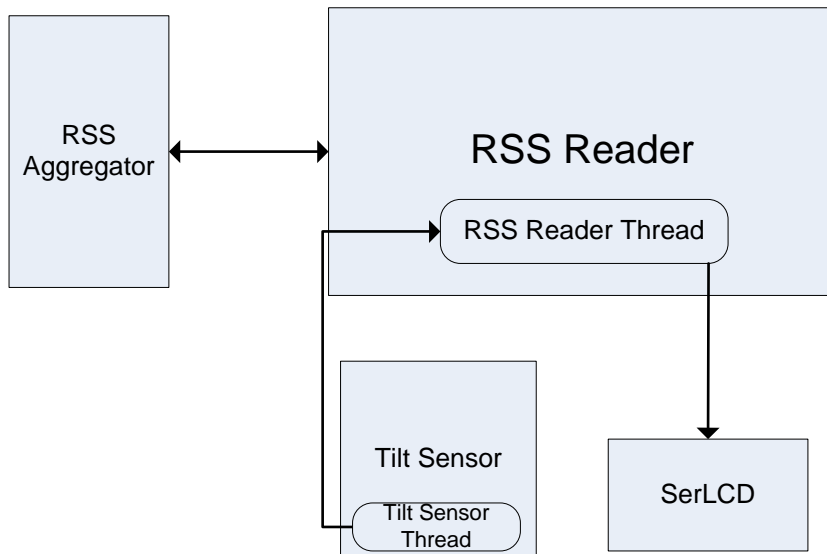


**Figure 4 - Hardware Diagram**

Figure 4 shows the wiring of the device. Both the LCD and Accelerometer are wired to ground and power on the TINI on pins 3 and 70 for the LCD and on pins 4 and 69 for the accelerometer. The LCD has one more wire connected from its receive pin to the UART1 transmit pin on the TINI. The accelerometer has a wire going from its x pin to pin 30 on the TINI which corresponds to bitport PCE0. Another wire goes from the y pin to pin 29 on the TINI, which corresponds to bitport PCE1. These bitports were chosen because they are easily accessed from the TINI API.

## Software Detail

The RSS Reader software uses TINI SDK 1.15, Java COMM API, Java 1.1 and contains 4 java classes: the main class, RSSReader; a class to send commands to the LCD, SerLCD; a class to download and parse RSS feeds, RSSAggregator; and a class to monitor the accelerometer input, TiltSensor.



**Figure 5 - Software Block Diagram**

Figure 5 shows the software block diagram for RSS Reader.

The RSS Reader thread creates a new instance of SerLCD, which initializes the LCD screen through serial communication. It then starts the TiltSensor thread which collects the pulse width modulation from the x and y pins on the accelerometer and stores them in a vector. Then the RSS Reader thread calls a function in the RSS Aggregator class to download and parse the RSS feeds from the given URL. The RSS Reader thread then calls a function in the TiltSensor class that calculates g based on the data collected. Based on this reading, the RSSReader thread either sends a scroll left or scroll right command based on the x axis reading, or sends a clear screen command and a new line of text based on the y axis reading.

Compiling the project is slightly complicated because java class files must first be converted to a form that the TINI can understand, the .tini format. This is done using a java program. To save space, you have to specify what dependencies are to be used for the program. Figure 6 shows the contents of the dependency file used by RSS Reader:

```

RSSRead=bin\RSSAggregator;bin\TiltSensor;bin\SerLCD

PROTOCOL=com.dalsemi.protocol

PROTOCOL_BASE
=%PROTOCOL%.BasicClient;%PROTOCOL%.DefaultFileNameMap;
%PROTOCOL%.HeaderManager;

HTTP=%PROTOCOL%.http.Connection;%PROTOCOL%.http.Handle
r;%PROTOCOL_BASE%;%PROTOCOL%.http.HTTPClient;%PROTOCOL
%.http.HTTPOutputStream

```

**Figure 6 - rssDep.txt**

The first line contains all of the classes written by me, the subsequent lines are from the default TINI dependency definition file which are needed for HTTP support.

To compile the code, the following batch file was used:

```

@echo off
REM Compile the classfile
javac -target 1.1 -bootclasspath
..\..\bin\tiniclasses.jar -d bin src\*.java

REM Run it through TINIConvertor to make it ready to
run on TINI
java -classpath ..\..\bin\tini.jar;%classpath%
BuildDependency -f bin\RSSReader.class -o
bin\RSSReader.tini -d ..\..\bin\tini.db -add
HTTP;RSSRead -p ../../bin/modules.jar;bin -x rssDep.txt

pause

```

**Figure 7 - buildRSS.bat**

The `-add` flag specifies which dependencies are required and are defined in the `rssDep.txt`, which is specified by using the `-x` flag. The `-p` flag tells TINIConverter to look for all class files in the TINI default `modules.jar` as well as the `bin` directory where my class files are. The directory structure is as follows:

<tini sdk directory>\examples\<rss reader directory>.

For example I had mine in `C:\tinisdk1.5\tini1.15\examples\Blinky` because I used the

Blinky example as a template to start my project off. Also note that I have the tini sdk directory a level further away from the c:\. It doesn't matter, as long as the directory containing the batch and dependency files is 2 levels after the root sdk directory. Within the project directory, the two subdirectories are src and bin. The src directory contains the source files and the bin directory is where the compiled class files are kept. The .tini file is generated in the bin directory as well.

### **Changes from the proposal**

The only change from the proposal was that instead of a digital to analog converter, I used an accelerometer with duty cycle output.

### **Problems encountered**

I encountered problems with both the LCD and the accelerometer. Assembly was no problem thanks to Dan, but the LCD had display problems when scrolling, which I do not know the cause of and therefore could not attempt to solve.

I could not get any consistent readings off the accelerometer. I am not sure if it is just too sensitive, or the responses are calculated after too much of a delay. I tried different sample sizes, which made the values less varied, but I could not get the accelerometer to give any sort of consistent output, no matter how many times I tried the same test of tilting it left and then right after a few seconds. I tried just measuring g as well as measuring the difference between measurements and never got anything close to consistent.

I had some problems getting my TINI board up and running because slush did not come loaded on it but the bootloader did, but for a version unsupported by Maxim (1.13). However, I was able to find the older software by posting a message to the Maxim message board and upgraded it to the currently supported version (1.15).

### **Conclusion**

Although the project did not work completely correctly, I feel I learned a lot from the experience. I learned hands on about pulse width modulation and duty cycles, and I also learned how to solder. I also proved that an accelerometer can be used as an input device, which was my goal for this project.