Introduction

After reading about AmForth on hack-a-day I looked around for a project to use it with. I decided to build an Arduino Controlled Digital FM radio with LCD display to gain some experience with AmForth and to bring back my Forth chops that I hadn't used in a very long time. The result is an FM radio that is fully remote controllable that I listen to with headphones while I am working. This project uses relatively simple hardware that can be built by anyone with basic electronic assembly experience. The software however is somewhat complex and took me some time to get working. More on the software a little later.

I place this hardware and software in the public domain so anyone can do with it as they please.

If you add a cool feature to the radio, please send me a note at: calhjh@gmail.com and let me know so I can incorporate it into my radio as well.

Hardware

The hardware is built around an Arduino Uno board running at 16 MHz and 5 volts. Other Arduino's could be used but some of the I/O pin assignment would need to change. Using an Arduino running at a different speed will also impact the design especially in the IR (Infra Red) detection area. I use an IR remote control from adafruit.com (see Figure Three) to control the radio. There are no other controls at all besides a display contrast trimmer adjustment for the LCD which should be set once and left alone.

The parts list for the build in shown in Figure One and a schematic of the hardware is shown in Figure Two. I built my radio using point to point wiring but a prototyping shield could be used for a cleaner build.

I packaged my radio using two 4”x6” pieces of clear 1/8” acrylic in a sandwich arrangement held together with wooden 1 1/2” dowel spacers. I like the look of the naked electronics. The LCD display is mounted to the front acrylic piece and the Arduino Uno and receiver board are mounted to the rear piece. See the photos for details.

The LCD display provides a 4 bit parallel data interface to the Arduino while the Si-4703 FM receiver is connected to the Arduino via an i2c serial interface. Since the Arduino Uno is a 5 volt part and the FM receiver is a 3.3 volt part, a bi-directional level converter must be used between them. The LCD display runs on 5 volts. The backlight for the LCD display is controlled by an output pin from the Arduino.

The stereo audio output cable is used as the antenna for the FM receiver so you must pay attention to the length of the interconnect. The cord on the headphones I use works fine as an antenna as I can pick up all of the FM stations in my area. When strong stations are tuned in, the audio quality is top notch. When the receiver is receiving a stereo broadcast, the stereo indicator on my build lights up. Weaker

stations are received in mono and the stereo indicator remains dark. A red power on LED lights when the radio is on.

An IR receiver (which is what Radio Shack calls it) is used to detect IR codes from the remote control. It filters out the 38 KHz IR carrier frequency thereby making detection of the key codes easier (though not easy).

The radio is powered via a USB cable and USB power supply. Alternatively the radio can be powered by connection to a USB port on your computer.

Software

Development Environment

I developed all of the AmForth software on my MacBook Pro using amforth-shell.py described in the AmForth documentation. Using the shell makes software development fast and convenient. All of the AmForth source files I use have a marker placed at the beginning of the file of the form underscore markerName underscore. That way I can easily unload/forget faulty code before I load a newer version. I installed amforth-shell.py as an alias called *4thterm* via the following entry in my .profile file.

```
#AMFORTH=~/Documents/dev/AmForth/amforth-5.1

#AMFORTH_LIB=$AMFORTH/lib:$AMFORTH/examples:$AMFORTH/../lcd_driver

#export AMFORTH_LIB

AMFORTH=~/Documents/dev/AmForth

AMFORTH_LIB=$AMFORTH

export AMFORTH_LIB

alias 4thterm="$AMFORTH/amforth-5.1/tools/amforth-shell.py -p /dev/tty.usbmodem1411 --no-error-on-output"
```

Then I can bring up amforth-shell by typing *4thterm* in any shell window. Quite convenient.

Software Components

Although the AmForth software spans many files, there are really three blocks of functionality that need to be discussed.

IR Detection

IR detection was by far the trickiest part of the software to get running. The code is contained in the file *irDetect.frt*. The software is tricky because of its real time nature of IR transmissions. Here, the Forth code polls the output of the IR receiver looking for transitions called MARKs and SPACEs. I won't go into the details here but the codes broadcast by a IR remote control form a serial protocol with the lengths of the MARKs determining where the codes start and which bits are to be considered ones and zeros. This code doesn't try to actually decode the codes being sent, instead it creates an identifier via hashing that uniquely identifies the keys on the remote. Consult the code for the details. If you want to

use a different remote with your radio, changes to this code will be necessary.

LCD Display

As mentioned a 4 bit parallel interface is used between the 16 character, 2 line LCD display and the Arduino. The software must first put the LCD display controller into 4 bit mode by executing a series of instructions which it does in its *initLCD* method. Numerous Forth words exists for controlling the display, positioning the cursor and writing text to the display. See the file *lcd_16x2.frt* for the details. The backlight of the display is controlled via an output pin on the Arduino and can be switch on and off without affecting the display content.

Si-4703 Receiver Control

The Si-4703 FM receiver is controlled using a series of 16 16 bit registers. In a typical operation the registers are read from the chip, values are changed and the updated register values are sent back causing the receiver chip to react. The Si-4703 chip has some idiosyncrasies that made code development interesting. For example all registers must be read each time and the Si-4703 provides the register content in the following order: reg 10 … reg 15 followed by reg 0 … reg 9. The code for controlling the Si-4703 is contained in the file *si4703.frt*.

There are Forth words available for setting the volume, setting the channel, reading the channel, seeking upwards or downwards, etc. The code I developed just scratches the surface of what the Si-4703 can do. I'm still working on code to read the RDS data provided by the receiver which identifies the radio station, the song being played and even the current date and time. In the future I hope to be able to display this information on the LCD display.

The file *ui.frt* brings all of the software modules together and forms the user interface for the radio.

Installation

Starting with the stock Arduino Uno installation I use amforth-shell.py to load the required Forth files. I call the file who's content is shown below *loadall.frt*. If you #include this file after a new AmForth install it will load up all the Forth files in the correct order.

```
\ Load up all pre-requisite forth files

#include postpone.frt
#include marker.frt

marker _EMPTY_

#include case.frt
#include buffer.frt
```

```
#include bitnames.frt
#include twi.frt
#include si4703.frt
#include irDetect.frt
#include lcd_16x2.frt
#include ui.frt
```

If all is well and everything loaded/compiled successfully, executing the top level word *runFMRadio* should start the radio. NOTE: the radio will appear off until you click the play/pause button on the remote. At this time the display will light up and the radio becomes functional. Clicking play/pause again makes the radio appear off though it is still listening for the IR code to turn it back on. Figure Three details the remote control key assignments. NOTE: *runFMRadio* is configured to run as a turnkey application which means it will run automatically when every time the receiver is powered up.

Resources

Information about the Si-4703 digital FM receiver can be found at the manufactures website at: www.silabs.com. AN230, AN231, AN243 are application notes concerning the Si-470x series parts. AN332 provides example code useful for programmers.

Figure One
Parts List

Item	Part Number	Source
Arduino Uno	16Mhz 5 volt part	SparkFun, AdaFruit, Radio Shack, eBay
Bi-directional Level Converter	BOB-12009	SparkFun
Evaluation Board for Si4703 FM Tuner	WRL-10663	SparkFun
16x2 line LCD display	many	SparkFun, AdaFruit, Radio Shack, eBay
IR Receiver	#2760640	Radio Shack
Red LED	many	anywhere
Yellow LED	many	anywhere
2 x 300 ohm ¼ w resistor	many	anywhere
10K ohm 10 turn trimmer	many	anywhere
Mini Remote Control	ID: 389	AdaFruit
USB cable for Arduino to USB power supply connection	many	SparkFun, AdaFruit, Radio Shack, eBay
USB power supply 500 mA or greater	many	anywhere
.1” male break away header pins for Arduino connectors	many	SparkFun, AdaFruit, eBay
Wire, solder, packaging, etc		

Figure Two
Schematic Diagram

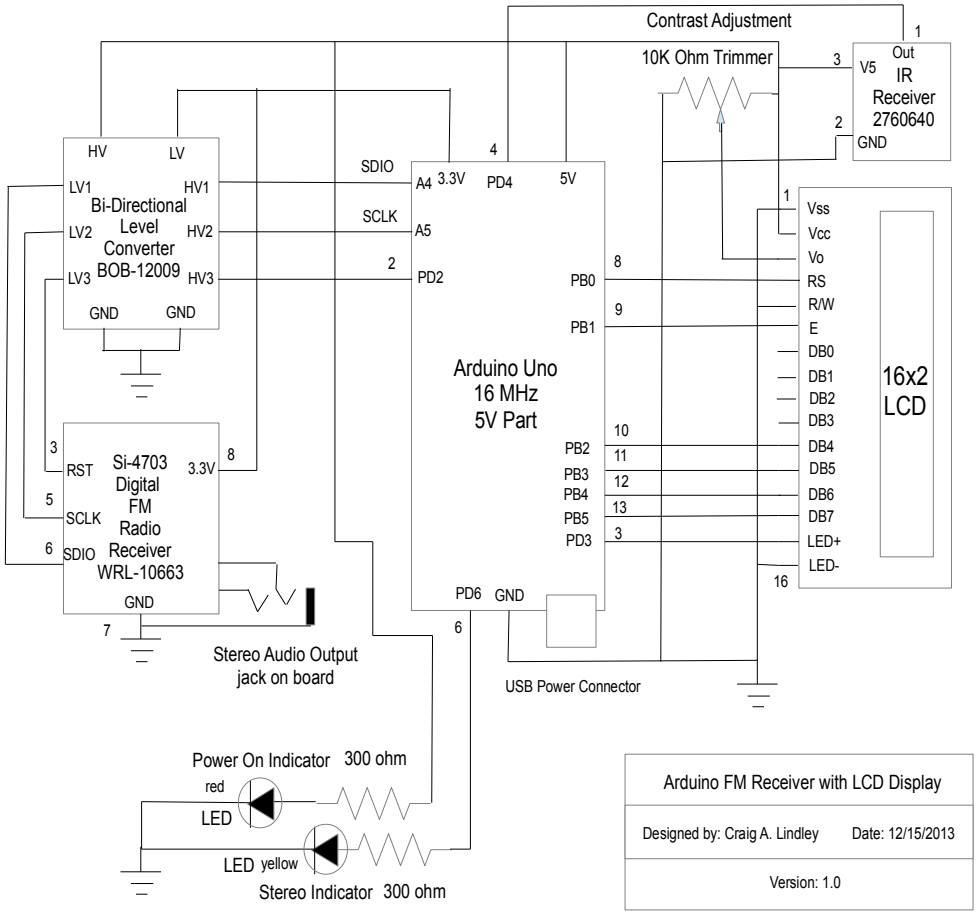

Figure Three
IR Mini Remote Control

Remote Control Functions Coded into the Firmware

Key	**Function**
Vol-	Mutes the audio
Play Pause	Turns the radio off and on
Vol+	Un-mutes the audio
Up Arrow	Volume Up
Down Arrow	Volume Down
Left Arrow	Scan Down for a station/channel
Right Arrow	Scan Up for a station/channel
Enter Save	Sets up for saving a preset. Tune a station, press Enter Save and then a key 1 .. 9 to save a preset.
Keys 1 .. 9	Tunes station if preset set, otherwise does nothing

Photo One
Three Components – LCD, Arduino Uno and Receiver Board

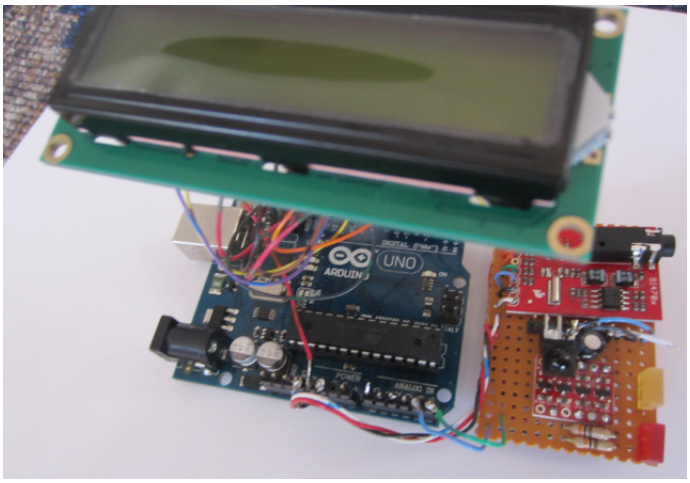

Photo Two
Receiver Board Close Up

At the top is the evaluation board with the black 1/8” stereo output jack, in the middle is the IR receiver facing upward, towards the bottom is the four channel level converter, the yellow rectangular LED is the stereo indicator and the red LED is the power on indicator

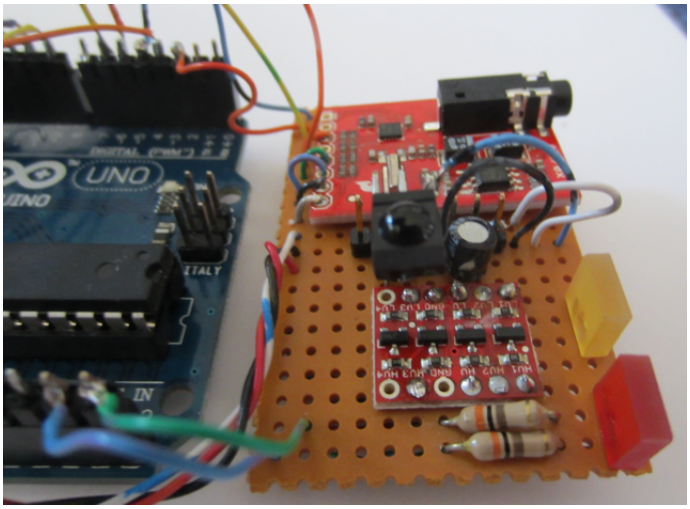

Photo Three
The Working Receiver in its prototype packaging

On the left is the USB cable powering the receiver. On the right I have my headphones plugged onto the receiver. The contrast adjusting trimmer can be seen on the left top of the LCD display.