RASSBlasterII

Mathew Tang

Charlie Martin

July 13, 2005

Summary

The RASSBlasterII is a Radio Acoustic Sounding System (RASS) tone generation device, used to generate audio signals that can drive a power amplifier and speakers.

A RASS system is based on the principal that a radar signal will be (weakly) reflected by density changes in clear air. An acoustic wave is a density perturbation, and when the wavelength of the radar is matched to the acoustic wavelength[1], a relatively strong radar return can be obtained. The speed of sound in air is related to the air density, which is a function of temperature, and thus the air temperature can be deduced by using the radar to measure the propagation speed of the sound waves.

Because the Bragg match condition depends on the air temperature, the correct acoustic frequency is not known in advance. The solution is to broadcast a varying sequence of tones whose frequencies bracket the expected temperature range. This also addresses the characteristic decrease of temperature with height.

The Bragg condition matches the radar wavelength to acoustic wavelengths. For a 915MHz RASS, the audio frequencies are in the vicinity of 2 KHz. For a 50 MHz profiler, the tones will be much lower[2]. Typically, the tone frequencies are randomly varied within a frequency range that corresponds to the RASS radar frequency and specified temperature range. A relatively short duration, e.g. 50 ms, is given to each tone.

A RASS is usually an add-on component to an existing wind profiler system. In the past, the RASS tone synthesis was often performed by hardware that was integral to the wind profiler. The RASSBlasterII will be a standalone device that can be utilized by any profiler system. The general characteristics are:

- The device is a network “appliance”, with access is via an Ethernet interface.
- A web server runs on the RASSBlasterII, and users can query and control the device via a web browser.
- Control from application programs on the network is accomplished through HTTP protocols. The *Wget* package provides a lightweight multiplatform API and library for incorporating

---

1 This condition is known as the “Bragg” match.

2 Legend has it that the 1812 Overture has been used as the acoustic source for low frequency profilers.

HTTP protocols.

Hardware Overview

The core of the RASSBlasterII is an embedded Linux controller. The controller has a network interface, a small number of GPIO lines, and a console serial port. The GPIO lines are used to interface to peripheral chips and to read the reset switch. Embedded controllers usually have a limited number of GPIO lines available, and so it is desirable to use peripheral chips which support serial interfaces, such as SPI or I2C. Even in this case, most of the devices require individual chip select lines which can quickly exhaust the available GPIO lines. A 3 to 8 line decoder integrated circuit can be utilized to multiplex chip select lines.

The peripheral chips and functions include:

1. A direct digital synthesis (DDS) integrated circuit. This device is programmed to generate an arbitrary output frequency. It has a serial digital control interface.

2. An LCD display. It displays the device status, and has a serial digital control interface.

3. A reset switch. This switch is read during bootup.

4. A programmable gain amplifier, used to set the output level of the audio tone. This is implemented with a combination of digital potentiometers and operational amplifiers.

Interfaces

The RASSBlasterII communicates with the external environment via:

1. An LCD display.

2. An Ethernet port

3. A serial port, used as a system console. This port may need to be enabled/selected via jumpers. The system console may be necessary for operating system maintenance. The serial port will typically not be brought to a connector on the outside of the enclosure; rather it will be accessed by header on the circuit board.

4. A reset switch. If this switch is closed during boot up, the device is initialized to the default state.

Non-volatile Storage

The RASSBlasterII will store a small amount of configuration information, and during a reboot, will restore this configuration:

1. The network configuration (IP address, netmask)

2. The RASS operating configuration

Network Configuration

In the default state, the network IP address is 192.168.1.100. Powering the device on with the reset switch held closed will reset the device to this address. This allows the RASSBlasterII to be reset to a known network address.

Once the RASSBlasterII can be accessed via the web server, a configuration page can be used to change the IP address of the device.

Web Interface

Web pages provide the following functionality:

- Configure the network characteristics.
- Reboot system
- Sound mute state
- Specify the acoustic characteristics. The relevant parameters will be:
    - Radar frequency (e.g. 915 MHz)
    - Minimum and maximum temperatures.
    - Tone length (e.g. 25 ms).
    - Modulation scheme (e.g. random versus sweep)
    - Start and stop fade time.
    - Output volume.
    - Sound Mute state

LCD Display

The LCD screen displays:

- A cool animated power on welcome logo. This can use the whole screen during the startup sequence.
- The current IP address.

- The current acoustic configuration.
- Indication of whether the acoustic generation is on or off.
- (Optional) The audio gain setting.

Audio Tone Control

Once the device has been configured for a specified audio behavior, it can be commanded to start and stop the audio tone generation. On a start command, the tone generation will begin, and the audio volume will be ramped up to the specified output level, over the specified fade time. The reverse sequence is followed on a stop command.

Software Design

Overview

The software should be designed to meet the RASSBlasterII requirements, allowing for extensibility. The Armadillo J is limited in it's non-volatile storage and ram so the software design takes this into consideration. The system is a basic client/server application where the client(s) can communicate via the http protocol. For example a web browser or wget running on a PC connected to the same network as the RASSBlaster.

When powered up the RASSBlasterII uses a simple database to restore the network and acoustic state from when it was last operational. One server program(rassblaster) is responsible for updating acoustic generation, the display, the volume and listening for changes in the acoustic configuration. A CGI program acousticCGI can be invoked via http, which will transmit the current acoustic configuration and receive updated data values. This program communicates with the rassblaster when the acoustic configuration has changed and the configuration database. A CGI program networkCGI allows users to view and change the current network settings of the system. Attributes are a statically assigned IP address, or the ability to obtain a IP address via DHCP. A reboot CGI script allows users to remotely reboot the system.

The rassblaster program communicates with the custom hardware, i.e. DDS, volume control, and LCD display using a common SPI interface. This interface uses four of the Armadillo J's General Purpose Input Output (GPIO) lines to communicate with the hardware.

Use Case: RASSBlasterII Power up configuration

1. User turns on the RASSBlasterII
2. At an early point during bootup, the RASSBlasterII's state is set to the last state it was in before it was powered down unless the reset button is being pressed or closed. If the reset switch is closed then the configuration database is deleted and reinitialized using a default read-only configuration file.

Use Case: Specify the acoustic characteristics via a web browser

1. The user brings up the RASSBlasterII acoustic settings web page.
2. The web page displays the current acoustic settings
3. The user changes all or one of the settings and presses an update button
4. The RASSBlasterII's acoustic state is updated, including the LCD display and configuration database

Use Case: Configure network settings via web browser

1 . User brings up the RASSBlasterII network settings web page.

2. The web page displays the current network IP address and if it was statically assigned or dynamically assigned.

3. User changes network settings, being static or dynamic IP assignment.

4. The update button is pressed.

5. The server receives the new values, restarts the network with the new values, updates the display and configuration database. If the static IP option was chosen, the IP address is displayed on the browser and the LCD screen, otherwise a message to check the LCD screen for the IP address is displayed.

Use Case: Web Reboot via web browser

1. User brings up the Reboot web page.

2. A rebooting message is displayed on the browser

3. The Armadillo J reboots.

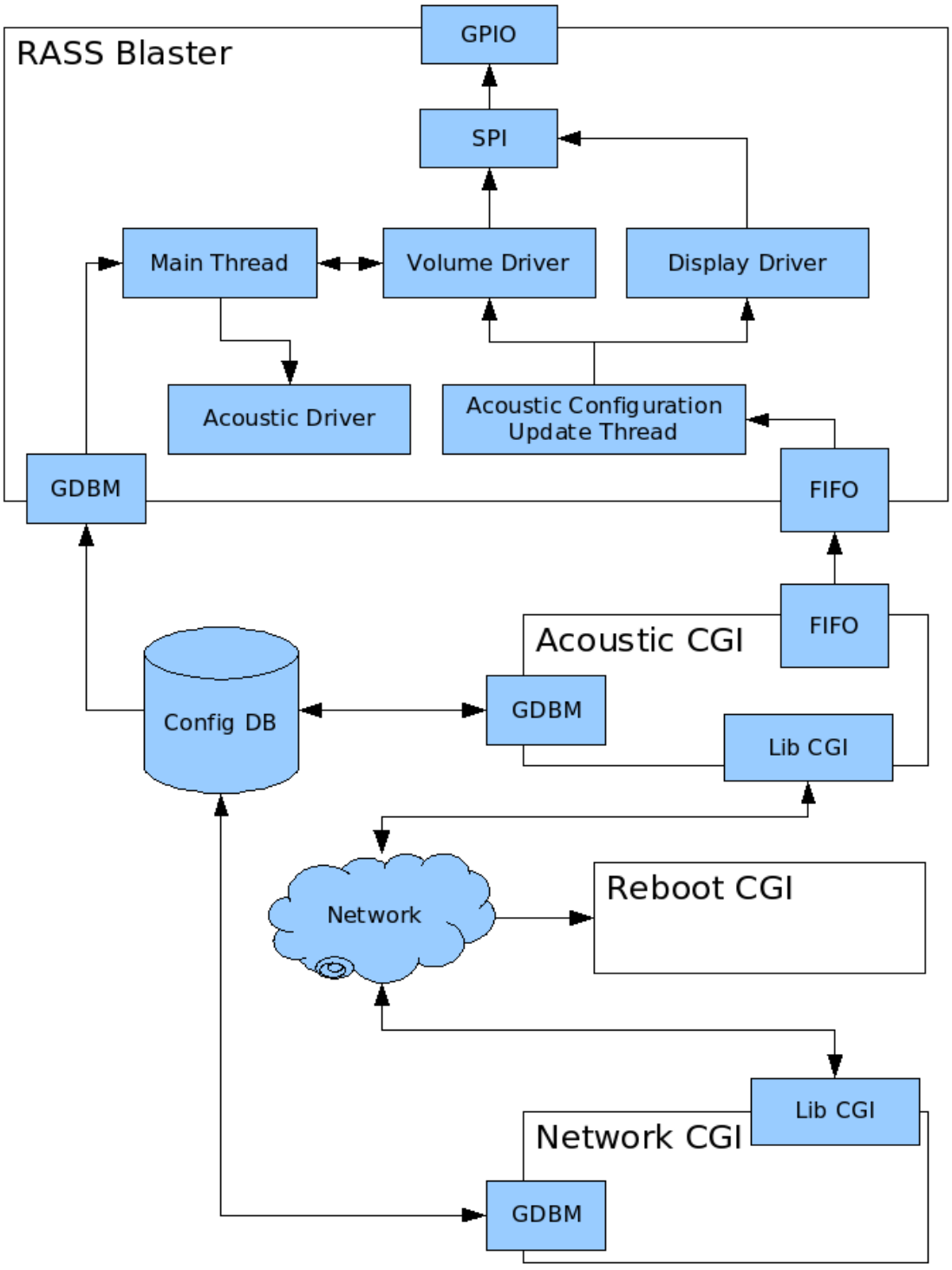

Fig 1. Component communication.

Components:

RASSBlasterII (Main Thread):

Type: Process

Data Flow: In and Out

The main driver. When this process is first started it loads the last acoustic configuration from the database. Does basic initialization of components and enters the main for loop. This for loop updates the acoustic driver and the volume driver.

RASSBlasterII (Acoustic Configuration Update Thread)

Type: Process

Data Flow: In and Out

This is a child process or thread of the RASSBlaster, its job is to listen on a named pipe or FIFO for new acoustic configurations. When it gets one it updates the display, and informs the volume driver the configuration has changed. An acoustic configuration data structure is shared with two processes, one is the acoustic configuration update thread and the other is the main thread of the rassblaster program. Therefore a mutex is needed to insure the rassblaster main thread is not reading the acoustic configuration data structure while the acoustic configuration update thread is writing to it.

Configuration Database:

Type: Database

Data Flow: In and Out

Used to store the current state of the RASSBlasterII system. State information is stored as key and data pairs. The database is synchronized to ensure it is thread safe.

GPIO:

Type: API

Data Flow: In and Out

Provides an API for common GPIO operations such as changing the output to an on or off state for a GPIO pin, and reading the current input state of a GPIO pin. The GPIO operations are to use the alternate GPIO function. The possibility of another process or the kernel using the Armadillo J's GPIO registers for something other then the alternate GPIO functions is taken care of by simply setting the

GPIO function select to the alternate function each time a GPIO operation is used from this API.

This part is almost the only hardware dependent part of the system. There is a GPIO API implementation for at the NS7520, and the PXA255. Users are able to specify what GPIO pins are used for the SPI buses clock, data, and the demultiplexer address lines.

SPI:

Type: Class

Data Flow: In

The SPI bus is shared by more then one device, so there needs to be a way to synchronize access to the bus, allowing all devices to receive valid data. By connecting a demultiplexer to two of the GPIO lines, three different SPI devices share a single SPI bus using only four GPIO lines. A mutex or semaphore is used to ensure only one process has access to the bus at a time allowing it to completely send its serial data in full to the destined device.

The SPI API provides basic SPI functions. For example reading and writing bits, bytes, and bit streams to the bus. SPI clients can select a device by a device select function. This function blocks until the bus is no longer being used by other clients. SPI clients must free the bus by selecting the NULL device when they are done using it.

Display Driver:

Type: Class

Data Flow: In

A class to display a cool animated power on welcome logo. (This is suppose to be done during startup, this might take some research. The Crystalfontz display has the “Control the Boot Screen” command, this might be all that is needed)

Provides methods to display the current IP address, radar frequency, tone length, start and stop fade time, modulation scheme, indication of whether the acoustic generation is on or off, and the volume.

Acoustic Driver:

Type: Class

Data Flow: In

This class is responsible for updating the output frequency every time the tone length expires. This class takes into consideration the settings of the acoustic configuration, being the radar frequency, minimum and maximum temperatures, tone length, modulation scheme (e.g. random versus sweep), sweep step frequency and if the DDS should be shut down, i.e. Muted.

Volume Driver:

Type: Class

Data Flow: In and Out

This class provides a function to set the output volume using a AD8402 chip and a linear gain OP amp circuit. It also takes into consideration the start and stop fade times, the output volume, and the mute state of the acoustic configuration. The class contains a state variable which can be: lowering the volume, raising the volume, or the volume is in a non-changing state. If the volume is lowered by the acoustic driver, then the volume is set right away. If the volume is being raised, then the volume ramps up to the desired value. The speed at which the volume ramps up depends on the start time. For example if the volume is at %50 and the start time is one second, then it will take half a second to retch the desired volume. Stop time or ramp down period is used when the system is muted. When the state is changing to a muted state the volume is lowered from it's current setting to zero in the time period defined by the stop time. Vise versa for disabling mute.

Bootup Init:

Type: Script

Data Flow: In and Out

This script is one of the first scripts executed by the Armadillo J when it boots up. It calls a program to check the state of the reset switch using a GPIO line. If the reset switch is being pressed or is closed, the configuration database will be initialized, the entire RASSBlasterII state will be reset to the default values.

GDBM:

Type: Class

Data flow: In and Out

This class provides easy synchronized read and write access to the configuration database.

Acoustic CGI:

Type: Process

Data Flow: In and Out

This process queries the configuration database and builds a web page containing key and data pairs, with the labels as keys and the text fields as the data values from the acoustic configuration database. The class displays the radar frequency, minimum and maximum temperature, tone length, modulation scheme, sweep step frequency, volume, and mute state. The user can change the text fields and post the

data via the wget tool or a web browser. Changed attributes are saved to the configuration database via the GDBM class. The new acoustic configuration is sent on a named pipe or FIFO to the acoustic configuration update thread in the rassblaster program. HTTPD get and post requests are handled using the libcgi library

Network CGI:

Type: Process

Data Flow: In and Out

This process queries the configuration database and builds a web page displaying the current network settings. The IP address is obtained from the Linux operating system and if it was statically assigned or assigned via DHCP is obtained from the configuration database. If changes are sent to the webserver, then the network CGI process updates the database, kills the rassblaster process and restarts it. This is a crude method to update the IP address being displayed on the RASSBlaster, but the network configuration should not be changed that much.

Reboot CGI:

Type: Script

Data Flow: In

A simple bash shell script to reboot the system. This script is run by the web server. First it saves the current state of the database to the flash memory by sending the USR1 signal to flatfsd, the process responsible for the file system to ensure the last configuration is saved. Then it calls the reboot program.

Test Cases:

Test Case Title	Change acoustic settings via remote computer's web browser.
Customer Requirement	Web Interface Specify to the acoustic characteristics

Steps	1. Boot up the RASSBlasterII with the reboot switch on. 2. Load acoustic settings webpage on a browser 3. When it is booted up change all settings different then the default settings via the web interface. Try different permutations and boundary conditions. For example 100 seconds for the start and stop fade time. Changing the volume when it is muted, etc 4. Verify the new settings are being displayed on the LCD screen and the RASSBlasterII acoustic output matches.
Expected Results	After the web page is submitted, the new values should be displayed on the LCD screen and the acoustic output characteristics should match.
Status	*Passed. Very large values (100sec, 10sec) were tested for the start and stop times, this worked. Small values were also tested, i.e.100ms, and one sec.* *Changing more then one parameter at a time and updating was tested, i.e. Start and stop time, and mute state.* *Different test values were used for the update interval, first 25ms was used, the scope reported 40ms being generated. 10Ms gave 20ms, 1000ms gave 1.01sec, 500ms gave 510ms. This accuracy is good enough.* *Minimum and maximum temperature, and radar frequency values were tested, using practical values like min -10C and max 40C as well as extremes like -200C and 200C. Different radar frequencies were tested also, i.e. 50Mhz,*

Test Case Title	Webserver / system stress test
Customer Requirement	The RASSBlasterII must be able to maintain a stable state after each update.

Steps	1. Boot up the RASSBlasterII with the reboot switch on. 2. Execute the stressTest bash script in the testScripts subdirectory of the rassblaster source tree. 3. Wait a long time, for example leave it on over night. 4. Stop the stressTest script. 5. Load the acoustic settings webpage on a browser 6. Change settings close to the default settings 7. Load the network settings webpage on a browser 8. Change settings 9. Check RASSBlasterII acoustic output, LCD screen, and configuration database state. 10. Verify the results are complaint to the settings submitted via the acoustic settings webpage and network settings webpage.
Expected Results	The RASSBlasterII state, acoustic output, LCD screen, and configuration database should match the values submitted by the acoustic and network settings webpage.
Status	*Passed. If the stress test script is set to post new configuration data one after another then the system crashes. This is probably do to the limited amount of memory the Armadillo J has. But if the configuration update interval is set to one minuet then the test passes.*

Test Case Title	Test web based network configuration tool.
Customer Requirement	The user must be able to change the IP address or have the system obtain an dynamically assigned IP address via a web interface.

Steps	1. Boot up the RASSBlasterII with the reboot switch on. 2. Load the network settings webpage on a browser 3. Change the IP address to 192.168.1.2 4. Verify the IP address has changed on the LCD screen and using ifconfig on the Armadillo J 5. Click the link to the new network configuration page on the browser. 6. Change the IP address assignment to DHCP 7. Verify the IP address has been dynamically assigned by looking at the LCD screen and using ifconfig on the Armadillo J 8. Load up the new RASSBlasterII homepage using the new IP address.
Expected Results	For each update, the IP address should be correct according to the desired setting.
Status	*Passed*

Test Case Title	Test web based reboot command
Customer Requirement	The user must be able to reboot the system via a web interface
Steps	1. Boot up the RASSBlaster 2. Load the reboot webpage 3. Verify the rebooting message on the browser 4. Verify the system is rebooting 5. Verify the system restarted.
Expected Results	The system should reboot.

Status	*Passed*

Test Case Title	Test reset switch
Customer Requirement	A method to reset the system to factory settings.
Steps	1. Boot up the RASSBlasterII. 2. When it is booted up, change all settings different then the default settings via the web interface. (If time permits try different permutations) 3. Reboot the RASSBlasterII with the reset switch held down.
Expected Results	Assert the RASSBlasterII is in the default state. This is done be looking at the LCD screen, and checking the state of the database. (microdb /etc/config/state -p)
Status	*Passed*

Test Case Title	Restore state on reboot
Customer Requirement	The RASSBlasterII will store a small amount of configuration information in non-volatile storage: 1. The network configuration (IP address, netmask) 2. The RASS operating configuration
Steps	1. Boot up the RASSBlaster 2. When it is booted up change all settings to something different then the default settings via the web interface. (If time permits try different permutations) 3. Reboot the RASSBlasterII

Expected Results	`Assert` the settings are the same as they were before the RASSBlasterII was rebooted. This is done be looking at the LCD screen, and checking the state of the data base. (microdb /etc/config/state -p)
Status	*Failed, netmask is currently not being saved. Passed otherwise*