

# A HANDHELD SOFTWARE RADIO BASED ON THE IPAQ PDA: SOFTWARE

Andrew Chiu (Vanu, Inc., Cambridge, MA, USA; [agchiu@vanu.com](mailto:agchiu@vanu.com));  
Jessica Forbess (Vanu, Inc., Cambridge, MA, USA; [jforbess@vanu.com](mailto:jforbess@vanu.com));

## ABSTRACT

Vanu, Inc. has demonstrated analog FM two way radio and digital APCO Project 25 waveforms on an iPAQ-based handheld software radio. The iPAQ contains a low-power 206 MHz StrongARM processor that is used for all signal processing. This paper explains the software approaches used and lessons learned from implementing the waveforms on this platform. Commercial and public safety applications of the handheld are also described.

## 1. INTRODUCTION

The public safety community in the United States and elsewhere faces a serious interoperability problem. Different agencies operate radio systems with incompatible frequencies and/or waveforms. Within a particular agency, newly acquired advanced digital radios may not communicate with legacy analog radios.

Vanu, Inc., in partnership with General Dynamics Decision Systems, has developed a handheld software radio prototype to address this interoperability problem. The handheld currently runs analog FM two way radio and the digital APCO Project 25 waveform and can tune to any frequency between 100 MHz and 500 MHz. By applying the Vanu Software Radio approach, the development time and cost was greatly reduced [1].

This paper describes the background on the handheld prototype, the Vanu Software Radio approach to software design, the experience gained from the prototype implementation, and the performance of the software.

## 2. SYSTEM DESCRIPTION

The handheld system consists of three components: the iPAQ, the FPGA interface card, and the radio transceiver. The block diagram is shown in Figure 1. The iPAQ contains a 206 MHz StrongARM processor running Linux. All of the signal processing software is implemented as an application-level program written in C and C++ running on top of Linux. By architecting the system in this way using a high level language, standard operating system, and a general purpose processor, the effort required to port to the handheld platform was very low.

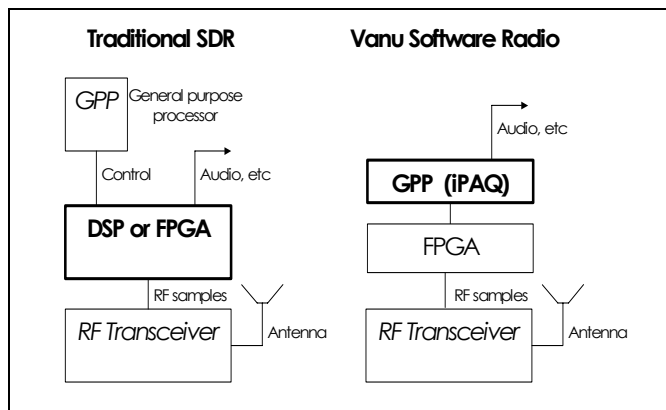


Figure 1: A comparison of the traditional software defined radio architecture and the Vanu Software Radio architecture

The hardware subsystem, including the radio transceiver and FPGA interface card is described in a separate paper [2].

## 3. SOFTWARE ARCHITECTURE

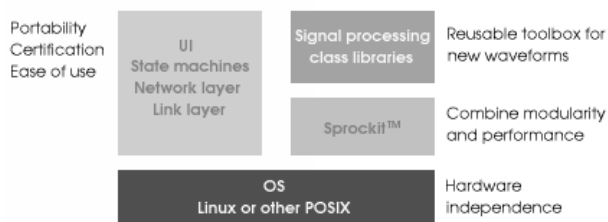


Figure 2: Vanu Software Radio software architecture

The Vanu Software Radio layered architecture, as shown in Figure 2, has four components. The lowest layer is the operating system, which is the interface to the hardware. The operating system is critical to the Vanu Software Radio approach because it allows the waveform software to be hardware-agnostic and thus, portable. The operating system chosen for the handheld system is Linux; however, any POSIX operating system may be used.

Two of the remaining components are signal processing components. The library of modules contains the core signal processing algorithms for operations such as filtering, FM modulation, and Viterbi decoding, as well as the

specialized waveform-specific modules. These modules are plugged together through the Sprockit™ middleware layer, which provides highly efficient data movement.

The control component consists of a runtime system that takes a description of the radio, selects the proper modules from the signal processing library, sets its parameters, and makes the high-speed data connections between the modules and the lower-speed control connections to the application. In addition to the runtime system, the control component also contains the user interface.

#### 4. HANDHELD SOFTWARE IMPLEMENTATION

The handheld software radio currently supports two waveforms: half-duplex analog FM, and digital APCO Project 25. The analog waveform is a 12.5 KHz wide FM modulated waveform with Continuous Tone Coded Squelch System support. This is the legacy waveform that is used by most of the public safety community.

The second supported waveform is APCO Project 25. Project 25 is a digital waveform that uses CQPSK modulation running at 9.6 Kbps and the Improved Multi-Band Excitation vocoder running at 4.4 Kbps [3]. Project 25 is also a half-duplex waveform, and it is the waveform that many public safety agencies are upgrading to.

The handheld software radio is able to switch between the two half-duplex waveforms, just by changing the software. While this is similar in functionality to existing Project 25 products, the Vanu, Inc. implementation has major advantages. First, the handheld software radio has a wider frequency range, enabling it to interoperate with a greater number of existing radio systems. Second, and more importantly, the software approach allows the handheld to be upgraded with new waveforms. By adding other low bandwidth waveforms such as IS-136 (North American TDMA cellular) and CDPD, the handheld software radio can now serve as a cell phone and a mobile data device in addition to its original function as a two way radio.

#### 5. VANU SOFTWARE RADIO APPROACH

The Vanu Software Radio approach depends on modularity, hardware independence, and software portability. When developing to these three key principles, waveforms and systems can be designed and built much more quickly, for less cost, and with fewer bugs.

With modular software, the waveform processing software can be easily configured to perform partial processing on a signal. For example, to investigate an audio problem on a received FM signal, the signal processing chain can be altered so that output of the FM demodulator is saved to disk instead of passed down the chain to the de-

emphasis filter. This technique is extremely useful, as it allows the engineers to access the internal values in the exact location in the system that they are debugging. Also, each signal processing function is implemented as a separate module with standard interfaces, so preexisting modules developed for different waveforms may be reused for a new waveform. Since these modules have already been written and tested, development and debugging time is reduced. These standard interfaces also make it easy to change the algorithm used in signal processing chains. Multiple algorithms for a particular function, such as FM demodulation, may exist as separate modules, and it is very easy to change algorithms and test them against live signals just by changing the module that is active.

With portability and hardware independence, waveform software can be first written for an architecture that has a better development and testing environment. Once the waveform is tested, it can then be ported to the target platform. This was the approach taken to develop the handheld software radio prototype.

#### 6. IMPLEMENTATION EXPERIENCE

All of the waveform software was first developed on the x86 platform. In fact, all of the waveform software on the x86 was complete and tested before the handheld project began. Both the FM and Project 25 waveforms were developed on x86-based Linux workstations, using the extensive set of development and debugging tools available on that platform. The workstations are fully functional radios, utilizing PCI A/D and D/A cards connected to external analog receivers and transmitters. This system also serves as a debugging tool that can capture signals off the air or send known signals to the device under test.

With the existing waveform software developed for the x86 as the starting point, there was a two step process to port to the iPAQ. The software was first ported to a StrongARM-based desktop computer. While the CPUs in the iPAQ and this desktop computer were identical, the desktop computer had two distinct advantages: it had an ethernet connection, and it had a large hard drive for permanent storage. These two factors allowed the engineers to use the same development and debugging tools that were used to develop for the x86. Familiarity with the tools led to a much easier and quicker porting process to the StrongARM. These tools were not as convenient to use on the iPAQ, and thus, attempting to port directly to the iPAQ would have resulted in a longer development process.

The major porting challenge was the lack of a floating point unit on the StrongARM processor. While the StrongARM is capable of floating point emulation, this is extremely slow. In order to get efficient performance on the StrongARM, some of the computationally intensive

processing modules were converted to use fixed point math instead of floating point. While the original floating point code was written for simplicity and ease of development, the new fixed point modules required a much more careful analysis of the signal flow and the magnitudes of those signals. Issues that did not matter in floating point math, such as overflow, underflow, and precision were now major factors.

For the most part, the conversion to fixed point math affected just the low-level arithmetic operations, such as addition and multiplication. However, a major algorithmic change to the FM demodulator was required. The original demodulator design makes heavy use of a particular operation, calculating the argument of a complex number. With a good floating point unit, the cost of this operation is small. However, when performed via floating point emulation on the StrongARM, this became the slowest part of the system. Various attempts were made to write a better fixed point approximation. However, none of these were acceptable. The solution was to replace the original FM demodulator with one that uses an FIR filter-based differentiator. This algorithm, since it is based solely on multiplications and additions, was much simpler to adapt for fixed point arithmetic. This conversion to fixed point, while necessary to port to the StrongARM, still preserves the key principle of portability, since the new code written for the StrongARM also runs on the x86; any processor with 32-bit types can run this code.

Using a performance-accurate desktop reference platform had major benefits for the software development process. At one point during the port, we noticed that the CPU utilization on the iPAQ was higher than predicted by the desktop platform. An investigation quickly revealed an erroneous compiler setting that had left expensive debugging code operating in the system. Without the desktop platform for reference, this might not have been detected until much later. The error would have affected decisions about where to invest optimization effort in subsequent phases of the port.

Once the new fixed point code was tested and verified on the StrongARM-based desktop, the code was moved to the iPAQ. Here, the main task was integration since the hardware, waveform software, and user interface were all developed separately. On the iPAQ, most of the integration issues centered around the interaction between the hardware components, and very few waveform software changes were necessary.

During the debugging and integration phase on the iPAQ, use of the Vanu, Inc. x86-based software radio system was indispensable. This system contains capture and analysis tools that Vanu, Inc. engineers have been using for several years. As a software radio system, engineers are able to rapidly modify the processing performed on it to

match the needs for each different debugging scenario. When running the system as a spectrum analyzer, the spectral characteristics of the transmitted signal from the iPAQ can be observed. When running as a modulation analyzer, the accuracy of the modulator and the symbol clock can be measured. As an arbitrary waveform generator, full FM or Project 25 signals using known audio inputs can be transmitted to the iPAQ. Test tones and other diagnostic signal patterns can also be sent. The system can also act as a receiver for each waveform and decode the signal transmitted from the iPAQ. In addition to emulating a full FM or Project 25 radio and fully processing the signal down to audio, this system is able to probe the receiver chain at intermediate points and report those outputs. This ability allows the engineers to see internal values such as the samples entering the demodulator or the bits entering the convolutional decoder. These values can be then processed other signal processing chains for debugging or manipulated in octave (a signal-processing program similar to Matlab).

Similar signal traces could also be captured on the iPAQ itself and then transferred to the x86 system for analysis. This is an extremely powerful debugging technique that allows the engineers to get access to the right information deep within the radio, which significantly reduces development and debugging time. More details of the integration process are given in [2].

## 7. SOFTWARE RADIO PERFORMANCE

In addition to enabling the low-cost, rapid development of waveforms and systems, the Vanu Software Radio development process also results in highly efficient code. Even though the StrongARM processor in the iPAQ is a relatively weak processor in the SDR world, it is more than powerful enough to run the Vanu, Inc. implementations of most narrowband waveforms. The Project 25 software, at its peak, consumes 24% of the 206 MHz StrongARM processor. This includes all functions to turn voice into symbols and vice versa, including modulation, demodulation, synchronization, decoding, and the vocoder. This implementation processes samples in 20 msec blocks. This is the minimum possible payload size, and hence the minimum possible latency, since it is the block size of the Project 25 vocoder.

The software on the handheld prototype also excels in other areas. The number of lines of code required to implement the Project 25 and FM waveforms are approximately 27,000 and 3,900, respectively. These figures include all the signal processing and control code as well as comments and unit test code. The figures do not include some general infrastructure code, such as the Sprockit™ middleware or device drivers. StrongARM binary size for the Project 25 waveform is about 480 KB,

which includes approximately 220 KB of user interface code and libraries. Code with such a small footprint is also quick to execute. Not counting the time to set up the RF front end, it takes approximately two-thirds to one second to launch either radio application.

## 8. CONCLUSION

Vanu, Inc., in partnership with General Dynamics Decision Systems, has created a very successful handheld demonstration system that illustrates the power of software radio. In its current state, its capabilities are competitive with current products available in the public safety market. With additional modifications to the hardware and software, it can be upgraded with capabilities that surpass most fielded public safety radios. The planned hardware upgrades are to extend the upper limit of the frequency range to 900 MHz and to expand the bandwidth to 200 KHz. These upgrades, coupled with the port of our existing TDMA, GSM, and CDPD code, will give the handheld new capabilities in the commercial cellular and mobile data areas. This new handheld would be able to function as a legacy analog two way public safety radio, a digital Project 25 public safety radio, a TDMA and GSM cell phone, and a mobile data terminal. Such a combination would be unique in the marketplace.

The key to rapid software development on a new platform is modularity, code portability, and hardware independence. While the holy grail of software radio, 100% code reuse, is nearly impossible due to inherent platform differences such as drivers and other hardware interface code, the porting effort should be minimized to the extent possible. This project was largely successful in that respect. Floating point to fixed point conversion was an additional required task, but much of that code was absorbed back into the main x86 code base, so that future porting projects of other waveforms to the StrongARM or other processors will be able to take advantage of this work. The end result of the Vanu Software Radio approach is that the port to the iPAQ resulted in an efficient, functional prototype and was completed in much less time, with far fewer people, and at much less cost, than would be expected with other software radio design approaches.

## 9. BACKGROUND ON VANU, INC.

Vanu, Inc. evolved out of the SpectrumWare software radio research project at MIT in the mid 1990s. Since its inception in 1998, the company has focused on waveform software development for software radios. The extent to which Vanu Software Radio systems implement signal processing in software distinguishes them from other software radio design techniques. Vanu systems perform as much signal processing as possible in application-level software running on top of standard processors and operating systems. The advantage of moving signal processing into software is increased flexibility: a Vanu, Inc. waveform (air interface) implementation can run on a range of devices, from handhelds to scalable infrastructure.

Vanu, Inc. licenses software radio technologies and waveforms and provides design-consulting services to wireless device manufacturers, system integrators, and service providers. The company's core competencies include strong software engineering, use of innovative signal processing algorithms, development of portable code, and the creation of commercial-off-the-shelf-based system design.

## 10. REFERENCES

- [1] J. Chapin, V. Bose, "The Vanu Software Radio System", 2002 Software Defined Radio Technical Conference, San Diego, November 2002.
- [2] J. Forbess, A.G. Chiu, "A Handheld Software Radio Based on the iPAQ PDA: Hardware," SDR Forum Technical Conference 2003.
- [3] TIA/EIA Standard, ANSI/TIA/EIA – 102.BAAA-1998, Project 25 FDMA Common Air Interface, May, 1998.