Variations on The Flock: Wireless Embedded Systems in an Undergraduate Curriculum

Bruce Hemingway, Waylon Brunette, and Gaetano Borriello
Department of Computer Science and Engineering
University of Washington
Seattle, WA USA

Abstract— **At the University of Washington, we have built our embedded systems curriculum around an innovative project that uses small wireless nodes to emulate the vocalizations of a flock of birds. In this paper, we describe our experience in building up this project, how it is central to our computer engineering program, and its evolution over the past several years into several variations that add diversity to the students' experience while at the same time keeping us up to date with technology trends. We conclude with a preview of our future plans.**

I. Introduction

In 2002, we introduced wireless embedded devices into our undergraduate computer engineering curriculum. At the Department of Computer Science & Engineering at the University of Washington, the core of this program consists of three classes (that, of course, rely on a large foundational core curriculum shared with computer science majors): advanced digital design, embedded software, and a capstone design experience. Initially, wireless sensor nodes were limited to use in capstone design classes where students work on large group projects. In 2003, wireless sensor nodes became part of the required curriculum in our embedded software course. The emerging technology of wireless sensors networks was a good fit to the embedded software syllabus with many overlapping topics such as interactions between multiple devices and interfacing techniques for connecting microcontrollers to a variety of sensors and actuators (both digital and analog). Moving the topic of wireless sensor networks to a point earlier in the curriculum gave students the skills to apply wireless sensors in their capstone design projects and independent research projects.

Originally, the embedded systems class used 8-bit microcontrollers that interfaced with stepper motors and other actuators, and communicated with other devices using RS-232 and IrDA. To update the course, we first focused on making sensors an integral part. We changed the early exercises to take students through the steps of designing their own USB device using a 2-axis accelerometer to emulate a mouse that was used to control a color selector on a pc. This condensed some material covered in previous edition of the course. We also introduced the ATmega architecture so that it would form a solid foundation for the sensor nodes we used in the larger project in the second half of the 10-week course.

This project was built around the UC Berkeley Mote sensing and communication platform. UCB Motes include a basic run time environment, TinyOS, which was a logical next step to the ad hoc sensor and actuator drivers the students had written in the first half of the course. To make the platform more interesting we decided to add sound generation capabilities to the mote platform. We gravitated to the idea of focusing the motes on sound generation as a way to motivate the students with more than data packet routing.

In the following sections we first describe this new theme, a Flock of Birds, and how we crafted a project where each student designed their own *bird*, using a common rule set. These birds worked together to show emergent behavior reminiscent of a *flock*. We then continue with a series of evolutionary variations on this basic theme that added diversity and better focused our educational efforts. We conclude with plans for our next steps in this evolution that will move us along the trajectory of important technology trends.

II. The Theme

The original Flock of Birds project [1], in autumn 2003, was designed to use motes within the context of our embedded software laboratory course. By incorporating wireless sensor networks as a core topic, students were able to master basic concepts of an emerging technology for use in later classes.

We chose the Crossbow (www.xbow.com) Mica2Dot platform which uses the Atmel ATmega AVR-series microprocessors (www.atmel.com). The Mica2Dot platform runs TinyOS developed at the University of California, Berkeley (www.tinyos.net), developed on top of the AVR-GCC (www.openavr.org) C compiler, which students use in the early part of the embedded software course.

The design of the Flock allowed us to incorporate the use of ad hoc networking and sound generation with the concept of emergent behavior based upon a common set of simple rules. The original Flock hardware required was a Mica2Dot mote and a piezoelectric sound transducer, shown in Figure 1. Earlier in the course the students developed ATmega pulse-width-modulation (PWM) software to play sixteen common birdsongs on the sound transducer, which became the repertoire of the Flock.

The general behavior of birds in the Flock is to sing a chosen song with some repetitions separated by some silence. Over time, different songs emerge as dominant for

some period, with songs starting, then spreading, and then dying out. The specific behavior of each bird in the Flock is controlled by a common rule set programmed by each pair of students into their birds, not from a master controller.

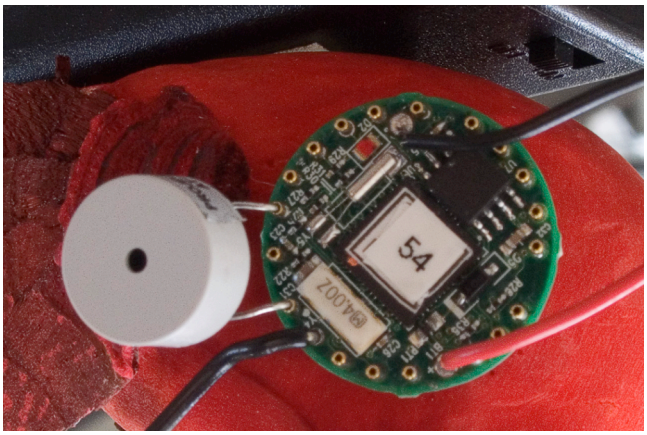

Figure 1. Mica2Dot mote with sound transducer

We presented a simplified set of rules that was purposely incomplete to elicit students' imagination. This was analogous to real-world experience, where a high-level specification is presented to the engineer who must then implement an appropriate design. This initial presentation of the Flock concept was done in a manner designed to interest and challenge the students. After explaining the concept of emergent behavior and the basics of cellular automata as exemplified by Conway's *Game of Life*, we presented the following simplified, template algorithm to the students:

Flock Process Flow:

a) Initialization tasks; select x = random(0-15)
b) Radio off; Sing birdsong[x]; Radio on
c) Listen for Random(min1, max1) sec., log data
d) SendMessage "I sang song x"
e) Listen for Random(min2, max2) sec., log data
f) Decide which song to sing next:
    1. Determine nearest songs from data log
    2. If my song is the same as any of the nearest songs, then repeat the same song
    3. If all nearby songs are the same, then switch to a different song
    4. If all nearby songs are distinct, then switch to a different song
g) Go to step (b) and repeat.

Students were assigned the task of inventing a methodology for predicting the success of this algorithm, and to suggest three improvements to it. To a person, the students decided the algorithm wouldn't work for numerous interesting reasons.

Some of the suggestions were:

Unless the song is growing, limit the number of times you repeat it.

If the number of other birds singing a given song is above a threshold, sing that song.

Birds should refuse to sing a song they have sung in the last 3 or 4 songs (if they decide not to re-sing the one they've just finished).

Weighting based on RSSI: sum of signal strengths for each song we've heard. Favors closer neighbor songs, and clustering.

Silence is golden. Occasionally a birdie should decide NOT to sing.

Most of these suggestions were incorporated into the final algorithm, giving the students a sense of ownership of the project design.

The students had to *qualify* their bird to participate in the final demonstration by passing a set of tests that helped the course staff identify problem birds. Birds that did not pass the test were re-programmed with code from birds that successfully passed the tests. The conclusion of the project was a *concert* of fifty motes in the Microsoft Atrium of the Paul G. Allen Center for Computer Science & Engineering, University of Washington (Figure 2).

Figure 2. Microsoft Atrium, Paul G. Allen Center for Computer Science & Engineering, University of Washington

The first Flock project was a success from an instructional viewpoint. It successfully integrated motes with a novel application. Students were exposed to communication protocols, constrained resources, hardware interfacing and a lightweight embedded operating system. From an experiential perspective, the opinion of students and other observers was that the process worked and the emergent behavior was aurally apparent and actually pleasing. The Flock provided a rich basis for subsequent quarters' iterations.

III. VARIATION 1

The second iteration of the Flock in autumn 2004 expanded the requirements to incorporate additional sensing and actuation. A photo resistor was added to the mote to sense the surrounding light level. Students were required to perform data processing on the incoming sensor data to detect a sudden change created by a shadow. Once the shadow was detected the birds would model a fear response of a possible predator by playing a *startle* song. Additionally, a startled bird would transmit a distress packet to the rest of the birds causing them to also play their startle song.

IV. VARIATION 2

One serious shortcoming of the original hardware (figure 1) was the poor quality of the sound generated by the piezoelectric transducer. For the third iteration of the Flock project in spring 2005, we introduced a new sound module, shown in Figure 3. The new printed circuit board included a small speaker, a rechargeable lithium-ion battery, a tri-color RGB LED, a light sensor, and a Yamaha FM-synthesis sound generator. The board also included pins to connect to the Mica2Dot mote and additional pin headers for debugging. These interfaces allowed the board to be controlled by either the Mica2Dot or by a breadboard. With the new board the students developed a control interface for the Yamaha FM device, which replaced the PWM software stack for the piezeoelectric transducer.

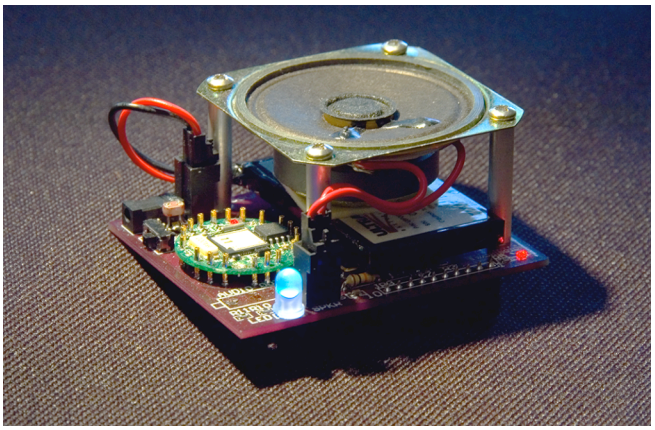

Figure 3. FM sound module

Since the Yamaha FM device was controlled through a FIFO interface, concurrent processing of radio packets and other interrupt-driven tasks could take place while sound was playing. Unlike the original hardware that required all of the interrupts to be turned off during sound production, the new board allowed more coverage in the collection of radio packet data. For this version of the Flock we required the students to perform more data processing with the extra cycles.

The new Flock songs were based on 49 song phrases of one individual Western Meadowlark (*Sturnella neglecta*) documented near Mitchell, South Dakota [2]. This meadowlark would sing songs consisting of several phrases and their repetitions. Our birds composed similar songs from these 49 phrases.

For a new common rule set, we introduced basic concepts from evolutionary computation [3], [4], so that birds would now broadcast their song phrase sequence (their songDNA) and would listen for similar songDNA from other birds. When a close match was found, the bird would breed by modifying its own songDNA by performing a random crossover splice to insert material from the other bird into their own song. In addition, the bird would then calculate the probability of a genetic mutation in one gene and possibly change one phrase number in its song. Each bird would then sing the new song, and the cycle would begin again. The birds indicated the closeness of the genetic match by the color of their RGB LED.

Life cycles were also incorporated in the new rule set. Each bird had five stages of life that affected how frequently the bird performed breeding (changing its songDNA). After a bird *died* it would restart with a random songDNA sequence.

Each bird was additionally affected by the ambient light level. Based on the light readings the bird would alter its behavior by increasing its song tempo and decreasing its silence between songs repetitions.

The overall aural effect in concert was a rich tapestry of evolving detail in birdsongs based on the changing songDNA. We had been concerned that the songs would evolve into a single repetitive song, but that was not the case. Indeed, the songs seemed to become more complex.

V. VARIATION 3

Variation 2 of the flock was successful, but was becoming too complex for students to finish in a short two week time span. Therefore, in the fourth iteration of the flock we used the original sixteen birdsongs and rule set with the newer sound board. Additionally, some of the coding requirements associated with song selection and emergent behavior were simplified by supplying students with a prepackaged TinyOS module that performed the song decision making. This allowed the flock project focus to shift to more central embedded systems topics. This variation of the flock also implemented a loosely synchronized network (at the resolution of 250ms). The network time enabled new flock characteristics such as being able to schedule a specific song to play at a specific time and to play in unison (at least within the 250ms). To achieve this loose synchronization, each node/bird took their internal time and averaged it with their neighbors reported time that was added to the packet they exchanged.

In addition, the flock protocol was changed allowing each node to be identified both from a central network controller and at each node individually. The goal of the identification was to be able to walk into a room with the Flock running and be able to identify any node without the nodes being labeled. Node identification could occur by the central control software sending a control packet to a limited

set of nodes to cause them to generate a visual or audio signal (e.g., setting their LEDs to a specific color or playing a song). The ability to address a node or a group of nodes allowed parts of the flock to operate with different global parameters enabling different behavior in separate parts of the flock. Moreover, a node was able to self-identify by transmitting an identification packet to the central controller when a person generated a coded sequence by shielding the light sensor. The sequence consisted of 6 extreme light edge detections at a speed of about one edge detection per second.

VI. VARIATION 4

For the latest iteration of the Flock in the spring of 2006, we kept the original sixteen birdsongs and rule set, and added a second state machine that controlled whether members of the Flock could succumb to the bird flu. A bird could be either healthy (Green), infected (Red), immune (Blue), or dead (Off), and would indicate its health by displaying the corresponding color on its LED.

We introduced the basic concepts of viral propagation [5] and we implemented a set of infection rules based on probabilities [6]. The students added the concept of viral mutation, so that immunity would be ineffective against a virus that had mutated far enough away from the virus that caused the immunity.

By setting the correct global variables, the Flock could be made to survive or die, activity that was shown by our monitoring software (Figure 4).

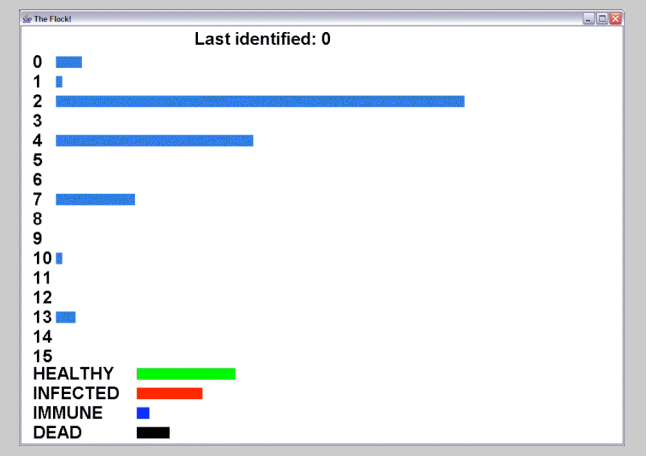

Figure 4. Monitoring software

VII. FUTURE VARIATIONS

Future variations of the Flock will be able expand into new areas of sensing and output. This winter we are moving to a new platform that includes high-quality sound and additional sensor-based behavior.

We are currently integrating a new advanced sensor network node platform into our embedded systems curriculum. The Intel iMote2 platform (Figure 6) is built around a low power XScale® processor, PXA271 running in the range of 13 to 416 MHz. It integrates an 802.15.4 radio with 256kB SRAM, 32MB FLASH, and 32MB SDRAM. This platform supports multiple interfaces: 3xUART, I2C, 2xSPI, SDIO, I2S, AC97, USB host, camera I/F, GPIO, Mini-USB port for direct PC connection, all in a compact size of 36x48 mm. It runs both TinyOS and Linux.

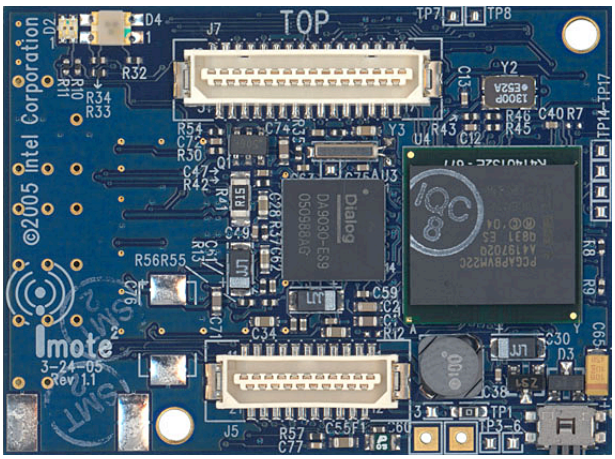

Figure 5. Intel Imote2 platform

Several sensor boards are available to us, including the Multi-Sensor Board [7] co-designed by Intel Research Seattle and the University of Washington, and the Basic Sensor Board from Intel Research Santa Clara. These incorporate sensors for: visible and IR light, 3-axis accelerometer, sound, temperature, humidity, and barometric pressure. In addition, we have designed and are manufacturing a new UW/CSE sensor board, which includes a cell-phone camera, small color LCD display, heart-rate monitor amplifier, USB host, and CD-quality stereo audio codec with speaker and microphone, along with battery charging power circuitry, in a cell-phone-like form factor.

We will be using Linux as our operating system, in order to capitalize on our students' familiarity with the Linux environment, and on the immense libraries of pre-existing software and device drivers. Under Linux, we can also support high-quality music synthesis environments such as Supercollider (http://sourceforge.net/projects/supercollider).

The new platform will allow future versions of the flock to generate higher-quality sound and have the ability to simulate many unique sounds that can be used to create soundscapes. Future project ideas include simulating a tropical rain forest in three dimensions using barometric pressure measurements, and exploring the interactions with humans based on visual or sound recognition [8]. Using the heart-rate monitor, we plan to explore uses of aggregate data for biofeedback sound experiments.

In the future, we plan to use more pre-packaged modules that help students complete projects with a richer set of interactive features. Example modules may include complex behavior rules and some device drivers. The goal will be to provide modules that are not the specific focus of the project so that students can concentrate on topics such as sensing, actuation, time synchronization, networking and other topics that are the true focus of the course.

VIII. Conclusion

The Flock project has been a success within our curriculum. We have been able to effectively teach the topics through a hands-on project experience that allows students to express their own imagination. The project has evolved well over the years to keep students interested by presenting a new challenge or twist each quarter the course has been offered.

There are several characteristics that make the Flock a success and may generalize to help others craft other projects.

First the project should incorporate the major embedded systems and wireless communications concepts and techniques. The Flock includes exposure to an embedded operating system, priorities and constrained resources, sensing, actuation, interfacing, and aspects of wireless networks, among others.

Additional characteristics contributed to the Flock's success as a large group based project. The Flock builds on concepts from earlier in the quarter allowing students to begin writing reusable code optimizing student effort in what is necessarily limited lab time. The instructor may also provide students with the parts of the project that are not of pedagogical focus. The Flock relies on emergent behavior with no easily guaranteed result. This leaves some anticipation for the students until the final concert. The Flock can also be easily changed to accommodate new ideas from the students.

The Flock requires that each pair of students complete a full implementation that must play well with others in the group. This contrasts with senior Capstone projects where each student completes a portion of a larger project. In addition, the use of a test fixture for qualification teaches students a valuable lesson in completeness of specifications and testing.

Student interest is enhanced by several characteristics. The Flock is an open-ended project. While we are quite strict in enforcing adherence to the rule-set, we expected variety and some randomness in actual performance by the group. The students appreciate a project that, unlike many others they have seen in lower-division courses, has no single solution. The Flock also allows the introduction of outside ideas and processes. Emergent behavior, autonomous activity, viral propagation, and evolutionary computation have all expanded students' thinking and added to the diversity of their experience. Finally, the Flock has general interest for a non-technical audience. It allows for a group presentation, a performance that can be attended and appreciated by friends.

Acknowledgments

We wish to thank Tom Anderl and Karl Koscher for their contributions to the design of the Flock, and all of our CSE466 students over the last few years for their hard work in making the Flock sing.

References

[1] Hemingway, B.; Brunette, W.; Anderl, T.; Borriello, G., "The flock: mote sensors sing in undergraduate curriculum," *Computer*, vol.37, no.8pp. 72- 78, Aug. 2004

[2] Arlton, Alexander V, *Songs and Other Sounds of Birds*. Parkland, Wash., Lithographed for A.V. Arlton by Eklund Print Co., Hoquiam, Wash. [c1949], pp. 138-139. Special Collections, University of Washington Libraries, 979.744a Ar53s

[3] Bentley et al, in Bentley, P. and D. Corne, *Creative Evolutionary Systems*. 2002, San Francisco, CA, Morgan Kaufmann Academic Press.

[4] Todd, P.; Werner, G.M., "Frankensteinian methods for evolutionary music composition," in Griffith, N. and P.M. Todd, *Musical Networks: Parallel Distributed Perception and Performance*. 1999, Cambridge, MA, MIT Press. xv, 385.

[5] Webster R. "Influenza: An emerging disease", *Emerging Infectious Diseases*, 4:436-441, 1998.

[6] Wilensky, U. (1998). NetLogo Virus model. http://ccl.northwestern.edu/netlogo/models/Virus. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[7] Brunette, W., Lester, J., Rea, A., Borriello, G. "Some sensor network elements for ubiquitous computing", *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, 2005.

[8] J. Lester, T. Choudhury, N. Kern, G. Borriello, B. Hannaford, "A Hybrid Discriminative/Generative Approach for Modeling Human Activities," *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pp. 766 - 722, Edinburgh, Scotland, 2005.