Wearable Interfaces for Cyberphysical Musical Expression

Andrew B. Godbehere
Cornell University
Ithaca, NY, USA
abg34@cornell.edu

Nathan J. Ward
Cornell University
Ithaca, NY, USA
njw23@cornell.edu

ABSTRACT
We present examples of a wireless sensor network as applied to wearable digital music controllers. Recent advances in wireless Personal Area Networks (PANs) have precipitated the IEEE 802.15.4 standard for low-power, low-cost wireless sensor networks. We have applied this new technology to create a fully wireless, wearable network of accelerometers which are small enough to be hidden under clothing. Various motion analysis and machine learning techniques are applied to the raw accelerometer data in real-time to generate and control music on the fly.

Keywords
Wearable computing, personal area networks, accelerometers, 802.15.4, motion analysis, human-computer interaction, live performance, digital musical controllers, gestural control

1. INTRODUCTION
Music and dance are rarely separated, as they complement each other so fully. The rhythms of music echo in the movements of the bodies of performers and audience alike. We describe a digital interface which seeks to fully integrate music and dance by transforming the human body itself into a musical instrument.

The system described in this paper allows the user to create and manipulate music with motion and dance. To offer the maximum flexibility for the musician, dancer, performing artist, or DJ, the system is fully programmable and configurable for a wide variety of musical scenarios. Machine learning techniques offer robust customizable gesture support to create motion-based control commands. When coupled with choreography, performance of electroacoustic compositions is possible with organic input introduced by the motions of a live performer.

2. HARDWARE
The hardware components of the system, in essence, comprise a basic Motion Capture (MC) system. Accelerometers placed at different points on the arms, legs, and head, track the motion of the user. This data, collected at different points around the body, must be transmitted to a computer for analysis and translation into music. To minimize hindrance to the user, our MC system completely eliminates wires. Data is transmitted wirelessly and independently from each accelerometer to a base station, which is attached to a computer.

This constitutes a wearable wireless sensor network, made possible by the emergence of the IEEE 802.15.4 standard [4]. Though each node in the network is independently battery powered, each uses such little power that a small, light battery is used for each, which can last for tens of hours of continuous use. The robustness of the IEEE protocol enables reliable communication within 100 feet of the base station, suitable for a typical performance environment.

2.1 Sensor Node Design Background
While sensor networks are relatively new, several have previously been implemented [3] [7]. In one instance, sensor networks comprised of Eco motes have been applied to dance [8]. Sensor networks used in live performance situations have strict design requirements. Our system, focusing on real-time music creation, is subject to these constraints and requires a high level of perceived interactivity with minimal latency.

The system in [8] utilizes a mix of low-data-rate wireless nodes in the 2.4 GHz band (with similar characteristics to 802.15.4 networks) co-located with 802.11 transceivers. The 802.11 transceivers were responsible for communication across the performance environment. However, 802.11 transceivers are bulky and consume a lot of power. Additionally, it has been indicated that 802.11 networks co-located with 802.15.4 networks significantly interfere with the communication of the 802.15.4 networks [9]. Because of these concerns, our design relies solely on 802.15.4 nodes. These nodes are still capable of communicating across a performance environment. A basic interference prediction technique, similar to a more sophisticated version [6], is applied to minimize incidental 802.11 interference and allow for fast and reliable data throughput.

2.2 Sensor Node Design
Each sensor node, or mote, consists of three main components: the accelerometer, the radio, and the microcontroller. The

microcontroller and the radio (see Figure 1) are available together from Atmel's Z-Link series, designed for Zigbee and 802.15.4 networks. The Atmel radio, the AT86RF230, offers a digital radio solution that requires a bare minimum of external components, allowing for low cost and a physically small footprint. A Linx chip antenna is used to minimize the form factor of the devices. The three-axis accelerometers from Kionix offer 6-g sensitivity and 12 bits of resolution, allowing the sensors to detect fluctuations in acceleration as small as 0.003 g's in any direction. The radio and the accelerometer both interface with the microcontroller through an SPI (Serial Peripheral Interface) link, with speeds up to 2 Mbps, as the ATMega644 microcontroller is operated at 4 MHz.

The radio operates in the 2.4 GHz band, although the IEEE specification defines two other bands, around 800 and 900 MHz, which may be used when there is too much noise in the 2.4 GHz band. The radio, when operating at 2.4 GHz is capable of a raw throughput of 250 kbps. As each sample from the accelerometer contains approximately 50 bits (12 bits * 3 axes plus protocol overhead bits), each node is itself theoretically capable of transmitting around 5000 samples per second. With a 5-sensor node system, the theoretical limit of the rate at which samples may be collected from the entire system is around 1000 samples every second. This time resolution is more than sufficient for a responsive system without noticeable latency. Our experiments have used as few as 60 samples per second with excellent results and no noticeable latency. This wide range allows for successful operation of the system even in electrically noisy environments where the communications rate is forced to drop.

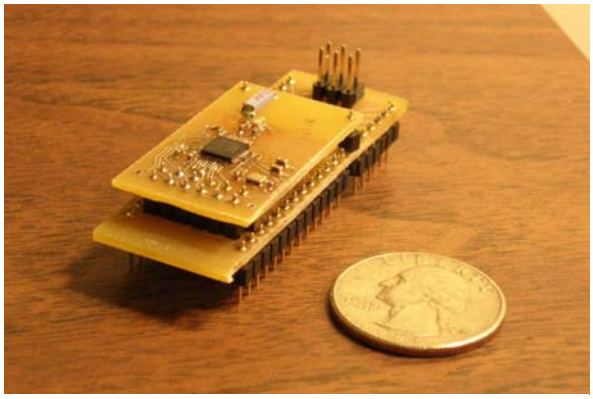

Figure 1. A wireless node

2.3 Network Layer Design

The software that runs on each node in the network is built on top of a custom library, designed according to an AT86RF230 software programming document [1], which encapsulates the physical layer of the network. The network layer is kept very simple to allow for fast implementation of new techniques, which are not incorporated into a typical 802.15.4 Medium Access Control (MAC) layer. In addition, we are interested in a single-hop network and do not need many of the features the full 802.15.4 specification provides. The networking layer we have implemented is not 802.15.4 compatible, although the physical layer is.

Our system requires several independent sensor nodes to communicate with a single base-station. Communication latencies must be kept to a minimum, samples should be collected from each node at regular intervals, and power consumption should be minimized. The 802.15.4 standard describes the Guaranteed Time Slot (GTS) feature that allows rigid, reliable data transmission rates between network slaves and a network master. However, the GTS feature requires the slaves to be either persistently listening, which wastes power, or time-synchronized, which requires extra communication.

To solve this problem, our system utilizes a collaborative virtual time slot allocation technique, which takes advantage of the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) feature. In essence, when each node wants to transmit, it listens to see if the channel is busy. If it is not busy, it will wait a random interval before transmitting. After a successful transmission, the node starts a deterministic timer, corresponding with the desired sampling rate, which indicates when the node should transmit its next sample. In the steady state, the node will transmit the next message after this pre-determined interval and will settle into a regular transmission schedule. If the node listens and finds the channel busy, it will wait a random interval before attempting to transmit again. It will continue to wait and check the channel until it finds the channel is not busy. At this point, the node will transmit its message.

With every node following this behavior, and using the same sampling rate, they will eventually settle into a schedule that fits for every node, where no message overlaps, assuming the message lengths are short enough given the sampling rate that is used. In addition, between each sample, the node can enter a standby mode to reduce power consumption and extend battery life. This scheme works well in a system such as this sensor network where each data frame to be transmitted will be of exactly the same length, and each node is taking samples at exactly the same rate. Since the clocks are not synchronized, however, and may actually run at slightly different rates, the "set" schedule for each node is not actually fixed. This scheme is flexible: as each sample timer is started only after a successful transmission, the schedule is readjusted such that no messages overlap. To minimize the latency jitter this may introduce, a reasonably low sampling rate is required, to allow some room in the transmission schedule for readjustments.

In short, this transmission scheme allows for high throughput without the communication overhead that would be required with other schemes. Samples are transmitted on reasonably tight schedules that allow for little random jitter in the time intervals between them, and is done without the use of timestamps and the overhead of clock coordination.

3. SOFTWARE

The base station is connected to the computer via a USB connection. FTDI's D2XX drivers[1] allow direct access to the USB device through a DLL so our software can access it through a series of DLL function calls. We wrote this software using flext[2], a C++ layer for cross-platform development of

[1] http://www.ftdichip.com/Drivers/D2XX.htm

[2] http://grrrr.org/ext/flext/

Max/MSP[3] and Pure Data (Pd).[4] This gives us an object, or external, to use in either of these graphical programming languages that interfaces directly with the base station through the USB connection and streams the accelerometer data into our Max/MSP or Pd programs, or patches.

We then designed a suite of patches to enable use of the sensor network with direct and indirect mappings and to allow the user to create or manipulate music in real-time. The accelerometer data can be processed in various manners to extract inclination and orientation when accelerometers are not moving (i.e. when overall acceleration is about 1g) and detect movements and gestures when in motion. By creating a library of low level data processing patches that analyze the raw accelerometer data and extract meaningful parameters about the sensor nodes, we were able to provide functional components for use in higher-level designs.

3.1 Data processing

The low-level library includes patches for calibration and converting ADC values to real measures of acceleration in g's, calculating total acceleration, jerk, frequency, and overall activity, and determining orientation and inclination. The total acceleration patch can be used for detecting overall acceleration of a sensor, but is also important in inclination error control. If the total acceleration of a sensor goes above 1g, there are forces other than gravity acting on it and inclination calculations are no longer valid.

One simple orientation patch takes the raw acceleration of three axes as input and essentially outputs which axis is facing upward, with a check that the accelerometer isn't in motion and a small bias toward the current orientation. For a more accurate indication of the accelerometer's position in three-dimensional space, we created an inclination patch to use on a per-sensor basis. It includes trigonometric calculations that use gravity to determine angles referred to as pitch, roll, and yaw for rotation about the accelerometer's x, y, and z axes. The method maintains constant sensitivity and allows tilt angles greater that 45° to be sensed accurately and precisely by using the acceleration of all three axes in each calculation of pitch, roll, and yaw [5]. For example, the pitch (X-tilt) calculation is given by ϕ in Equation 1.

$$\phi = \arctan\left(\frac{a_x}{\sqrt{a_y^{\,2} + a_z^{\,2}}}\right) \qquad (1)$$

After performing the three inclination calculations, making further corrections with sign recognition, and testing whether the sensor is moving and its data is valid, the patch outputs accurate measures of pitch, roll, and yaw in degrees.

Note that while designed for our sensor system, these patches also work with popular accelerometer-based input devices such as the Nintendo Wii Remote and Apple iPhone.

[3] http://www.cycling74.com/products/maxmsp

[4] http://puredata.info

3.2 Motion analysis

Patches were also written for movement and gesture recognition. Patches were created to determine the magnitude and direction of movements. Directionality is determined by using the last known orientation of the sensor at rest as the initial state and comparing this to the detected vector of movement. While we often used a simple measure of acceleration for the magnitude of a movement, we also found it helpful to track the duration of a movement as an important basic parameter.

We considered two forms of gesture recognition, essentially separating them into programmed and learned gestures. The programmed gesture schemes used a simple patch that detects when one specified action follows another within a specified time frame. This enabled us to combine multiple movements such that the overall gesture occurs when one movement is followed by another movement within a certain time period. A useful instance of these manually programmed gestures was that of recognizing a specified orientation followed by motion in certain direction. We designed this example with an accelerometer attached to the wrist to detect 6 orientations (palm up, palm down, thumb up, thumb down, fingers up, fingers down) and 6 directions of movement (up, down, left, right, forward, backward), which provide 36 different orientation/movement combinations. When combined with a second sensor for the other hand, the number of orientation/movement combinations is in the thousands. This example illustrates the ability to use the system to make commands with an "alphabet" of gestures, much like flag semaphore signaling uses two flags held in specific positions to signify letters.

The second form of gesture recognition uses machine learning techniques to teach the computer a set of gestures. Then, an arbitrary motion can be recognized from that set in real-time. We explored gesture recognition with hidden Markov models (HMM) by utilizing the FTM and MnM libraries [1]. The system has the capability to learn gestures, e.g. drawing shapes or numbers in the air, perform gesture following, and detect gestures with accompanying degrees of certainty.

Figure 2. Dancer performing with sensor system

4. APPLICATIONS

Our hardware and software infrastructure was applied to a number of scenarios with success. One of the most valuable was using the system on trained dancers (see Figure 2) with

the intention of not requiring the learning of any specific motions or gestures. In this situation, we wanted the design of the piece to allow for creativity and freedom of expression of the dancer. We attached four sensors to the performer's hands and feet, mapping continuous parameters of the dancer's motion onto algorithmic compositions. In a typical example, movement of each sensor would influence particular instruments. For each sensor, subtle movements could generate quieter sounds while quicker or longer motions triggered louder sounds that could be from different sets of instruments. Although the performer doesn't have control over the particular notes being played in this scenario, the type of movement influences the harmonic direction of the piece. We were able to effectively communicate these types of mappings to the choreographer, who was free to focus on dance without a need for the dancer to correctly perform specific gestures. This scheme worked well because the responsibility of musical content is shifted to the programmer.

On the other hand is a contrasting scenario in which the performer has a more direct influence over the music. A DJ or other musician needs functionality for precise control, so we depended more heavily on direct mappings and gesture recognition in these instances. For example, in one case we put a sensor on one hand that allows the performer to make commands and "push buttons" via gesture recognition, and a second sensor on the other hand to control continuous parameters via multi-dimensional inclination and "twist knobs." This case was successful because the first hand was relegated to performing discreet actions with recognized gestures while the second could be used for continuous parameters. For instance, the gestures of the first hand could trigger the next part of a song, control loops, switch instruments, etc. while the second hand could do things such as control the levels of multiple effects or act as a theremin-like instrument.

The system has also been applied in other interactive media settings, including use as an alternative gaming controller and as a human-computer interface for navigating operating systems and controlling computer applications with gestures.

5. DISCUSSION

This sensor system has been a powerful tool for musical expression in translating human movement to music. Although the sensors and auditory output are external processes, they are based on internal human motivations, and the system was able to capture one's natural motion and materialize the intangible processes of the performer.

Further work will include increasing the reliability of the hardware system as well as decreasing its size and power consumption. We also plan to increase the robustness and flexibility of the software patches, hope to improve the usability of the gesture recognition system, and test scenarios using a greater number of sensor nodes.

6. ACKNOWLEDGMENTS

The authors would like to thank Bruce Land and Kevin Ernste for providing an environment conducive to this work. They would also like to thank Cisco Systems, Inc., Kionix, Inc., and Atmel Corporation for their support in developing this system. This work was performed while Andrew Godbehere and Nathan Ward were students at Cornell University where Godbehere was studying Electrical and Computer Engineering and Ward was studying Computer Engineering and Music. Godbehere focused on the hardware for sensor data acquisition while Ward focused on the software for data interpretation. Both authors contributed equally to this paper.

7. REFERENCES

[1] Atmel Corporation. *AVR2001: AT86RF230 Software Programmer's Guide*, 2007. http://www.atmel.com/dyn/resources/prod_documents/doc8087.pdf.

[2] Bevilacqua, F., Muller, M., and Schnell, N. MnM: a Max/MSP mapping toolbox. In *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05)*, Vancouver, Canada, 2005.

[3] Gao, T., Massey, T., Selavo, L., Crawford, D., Chen, B., Lorincz, K., Shnayder, V., Hauenstein, L., Dabiri, F., Jeng, J., Chanmugam, A., White, D., Sarrafzadeh, M., and Welsh, M.: "The Advanced Health and Disaster Aid Network: A Light-weight Wireless Medical System for Triage" in *IEEE Transactions on Biomedical Circuits and Systems*, in press, 2007.

[4] Gutierrez, J., Callaway, E., and Barrett, R. *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4, Second Edition*. IEEE Press, New York, NY, 2007.

[5] Kionix, Inc. *Tilt-Sensing with Kionix MEMS Accelerometers*, (Nov. 30, 2007). http://kionix.com/App-Notes/AN005%20Tilt%20Sensing.pdf

[6] Musăloiu-E., R. and Terzis, A., Minimizing the effect of WiFi interference in 802.15.4 wireless sensor networks. *Int. J. Sensor Networks, Vol. 3, No. 1, 2008*

[7] Park, C., and Chou, P., Eco: ultra-wearable and expandable wireless sensor platform. *International Workshop on Wearable and Implantable Body Sensor Networks*, 2006.

[8] Park, C., Chou, P., and Sun, Y., A Wearable Wireless Sensor Platform for Interactive Dance Performances. *Proceedings of the Fourth Annual IEEE National Conference on Pervasive Computing and Communications*, Pisa, Italy, 2006.

[9] Petrova, M., Wu, L., Mahonen, P., and Riihijarvi, J. Interference Measurements on Performance Degradation between Colocated IEEE 802.11g/n and IEEE 802.15.4 Networks. *Sixth International Conference on Networking*, 2007.