ECE 398 Project Proposal

Mobile Music Composition

1. Introduction

The aim of this project is to build a device that can capture motion and tactile data from a musician and generate MIDI (Musical Instrument Digital Interface) signals, music notation, and sounds on a mobile platform.

Unlike many other types of artists, musicians are unable to record their ideas for a new piece of music when they are away from their recording equipment. Unlike a poet or a mathematician, there is no easy way to get an idea with layers musical notation onto a sheet of paper for later consumption. This is a unique problem faced by musicians. To prevent this large creative loss, we propose to build a mobile platform by which musicians can convert the motion of their hands and fingers into samples that can be used to compose music while mobile. Many current solutions exist, such as Air Beats[1], a glove that allows musicians to compose music digitally without interacting with a screen. However, there is yet to be a commercially successful product as there is a high reliance on either expensive pre-existing technology with questionable reliability (digital paper technique), or a lack of throughput capability due to inexpressive language during pattern recognition. Our solution aims to solve both of these problems with current solutions by using a minimal set of viable hardware and an expressive language that can be used to quickly translate motion into synthesizable sound.

i. Benefits
   1. Able to compose music while on the move, or when other recording equipments are not available.
   2. Ideal for a quick sketch to save creative ideas
   3. Able to quickly demo the idea in their mind to other people
ii. Features

1. User Interface
   - An easy-to-learn and expressive gesture language that uses hand and fingers motion to express music.
   - Personalizable shortcuts and gestures can extend the language.
   - Interface APP runs on user's smartphone to communicate with the wearable hardware.

2. Technical Features
   - A wearable hardware that can capture hand and fingers movement and generate MIDI data.
   - Wireless bluetooth headphones connection
   - Adaptable recognition algorithm based on each user’s movement
   - Battery lasts 4 hours after each full charge

3. Music Production Features
   - Instant music playback during the composition
   - Generate music notation in real-time
   - Ability to draw from a sound sample library and attach
   - Two or more devices can be connected and work together.

iii. Development Goals

1. Demonstrate what makes a "language" expressive, easy to learn, and powerful to use. Possible examples: vim, stenotypining, Cornell note taking, Dvorak keyboard layout, American Sign Language, etc.
2. Evaluate the kinesthetics of the hand to make it easy to keep a rapid pace without damage to your hand health (eg carpal tunnel)
3. Propose alternatives to the hand for impaired individuals
4. Develop a prototype language
2. Design

Figure 1: Block Diagram for implementation of system
IMU: Inertial Measurement Unit
Our Motion Pattern Recognition algorithm employs a hidden markov model (HMM) to estimate the motion of the device, based on reasonable assumptions about the probability distribution of the sensor values. This allows us to filter out improbable hand motions that occur due to sensor failure. The output of this model is a hand motion estimation vector, which encodes the zenith and azimuth motion of each fingertip in the device. This data is then combined with the output of the flex sensor in order to determine which gesture the user is making. This combined output from the flex sensor and HMM is passed to a particle filter output as a hand gesture estimation for use in the software.

Gesture Recognition occurs continuously, and the current gesture, including its validity and estimated position parameters, are continuously shared with the mobile application over Bluetooth.
Software State Machine

Figure 3: Interface APP state machine

Our mobile application employs a simple producer consumer design pattern to allow event-driven consumption of the gestures where needed. The validity of the current gesture is encoded in the Bluetooth data packet, and used to trigger the series of changes that must happen when a new event is registered.
Project Subsystems

There are three main subsystems in the system: power, wearable hardware and interface APP.

1. The **power subsystem** is responsible for the power supply of the entire system. The power of our system is supplied by a Lithium-ion battery, and a power board with voltage feedback regulation to maintain a stable 5V+/-.1V with a current supply capability of 0mA to 2500mA. Our power subsystem is responsible for safely charging and discharging the battery, while also maintaining a stable voltage level for the sensitive electronics in our circuit. The power subsystem will be able to detect faulty power levels, and if necessary, turn off the device to protect the onboard electronics.

2. The **wearable hardware** is a hardware subsystem, which contains a vibration sensor unit, memory unit, bluetooth data transmission unit, and a microcontroller.
   - The **sensor unit** includes a flex sensor unit which can be used to detect the motion of fingers. It also includes an inertial measurement unit (IMU), which contains an accelerometer, a gyroscope and a magnetometer. The IMU can be used to detect more complex hand movements and gestures. Angles necessary for gesture detection are measured relative to a gyroscope at the base of the glove. Two IMUs are place on the back of each hand, 2 cm above the wrist. Each finger of the glove is attached with one flex sensor. So, together there are 10 flex sensors for the flex sensor unit and 2 IMUs.
   - The **memory unit** is used to store all the data needed for the hardware subsystem, including program code and data generated during operation. The memory unit also stores the necessary data for executing instructions. It has an on-chip memory for fast data access and also a DRAM for large data storage.
   - The **Bluetooth unit** allows users to connect their wireless speaker and headphone, and it also enables the communication between the wearable hardware and the smart device that runs the interface APP.
   - The **microcontroller** is the CPU of the hardware subsystem. It executes instructions and orchestrates communication among different units within the hardware subsystem. The microcontroller will be used to coordinate communication between the various hardware subsystems and the user. It will accomplish this by collecting sensor data over a serial connection in a raw format, processing that sensor data into an acceptable format for Bluetooth transmission, and parsing any response over Bluetooth into the appropriate hardware commands.

3. The **interface APP** is a software subsystem that serves as the interface between user and hardware. The APP runs on user’s smart devices. It communicates with the hardware via
Bluetooth. It allows the user to replay the music just composed and generates the sheet music for the user. It also runs the pattern recognition algorithm that can recognize user’s motions and generate the music accordingly.
Language Gestures
The language is currently not something we have fully fleshed out, but we are taking cues from American Sign Language to create the details. A gesture is defined by the unique combination of the position and motion of the 10 fingers, and the open or closed-ness of the palm. Each finger can take on three positional states: flat, curled, and contacted (e.g., touching the palm or another finger). Contact supersedes the previous two. The palm being open or closed allows us to encode more gestures.

Figure 4: Change current instrument

Figure 5: Rewind current track. Reverse direction of right finger for fast forward.

Figure 6: Change volume of current track

Figure 7: Define base note (combination of thumb/finger uniquely defines note)

Figure 8: Change active track

Figure 9: Define relative note transition (thumb position is sharpness/flatness)
3. Requirements and Verification

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery lasts at least 4 hours (5V +/- .1V, 2500mA)</td>
<td>Fully charge a battery once with 110V charging voltage; Record the lifetime of the battery.</td>
</tr>
<tr>
<td>Regulator can output 3.8V output given a 5V input voltage</td>
<td>Measure the input voltage of the regulator. Make sure it is 5V; then measure the output voltage of the regulator, and check if the output is 3.8V</td>
</tr>
<tr>
<td>Gyroscope within each fingertip needs to be able to distinguish zenith angle to within +/- .1 radians</td>
<td>Take three measurements from the gyroscope under z-axis, y-axis, and x-axis bias using known angle. More details in the tolerance analysis section.</td>
</tr>
<tr>
<td>Gyroscope within each fingertip needs to be able to distinguish azimuth angle to within +/- .1 radians</td>
<td>Take three measurements from the gyroscope under z-axis, y-axis, and x-axis bias using known angle. More details in the tolerance analysis.</td>
</tr>
<tr>
<td>Sensor unit can capture the motion with greater than 95% accuracy. See detail requirements in tolerance analysis.</td>
<td>Run the motion capture algorithm with all the possible movements in the language 20 times by 5 different people, and then confirm the accuracy and make sure the algorithm is well-generalized among different users.</td>
</tr>
<tr>
<td>Motion recognition algorithm has 95% accuracy</td>
<td>Use the motion data from the above step as testing dataset to test the recognition algorithm and check if the accuracy is above 95%. The detailed testing procedure is described in the tolerance analysis.</td>
</tr>
<tr>
<td>Motion recognition allows user defined new gestures/movements with greater than 90% accuracy.</td>
<td>Recognize and save novel observation vectors when instructed. Test repeat execution of said instruction.</td>
</tr>
</tbody>
</table>
| Bluetooth connection allows communication between the hardware and the interface APP within 0 to 5.5m | - Test the data transmission between the APP and hardware by transmitting testing audio and motion data in the 5.5m range;  
- Confirm the transmissions have an packet bit error rate smaller than 1% successful by examining the packet bit error rate. |
4. Tolerance Analysis

Critical components:

- Sensor unit is the most critical component of our system. It is essential for capturing the user’s gestures and it is the bottleneck for the performance of recognition algorithm. The accuracy of the entire system depends on the accuracy of the sensor unit.
- There are two subsystems within the sensor unit: inertial measurement unit (IMU) and flex sensor unit. The accuracy of the sensor unit is a function of the accuracies of the IMU and flex sensor unit.

Acceptable Tolerance:

The acceptable tolerance for the sensor unit should be within 10%, since this is the maximum error rate for the recognition algorithm to be able to function. If the accuracy for the sensor unit is less than 95%, then the recognition algorithm will not be able to distinguish different gestures, and thus greatly impair overall performance of the system.

The accuracy of the sensor unit can be calculated as a function of the accuracies of IMU and flex sensor:

\[
Accuracy(Sensor\ Unit) = \frac{3}{4} \cdot Accuracy(IMU) + \frac{1}{4} \cdot Accuracy(Flex\ Sensor)
\]

The accuracy of the IMU is weighted more heavily than the flex sensor as the measurements it takes are more critical to normal function.

The IMU has more impact on the accuracy of the sensor unit because the gestures are mainly determined by the hand movements, whereas the finger gestures only provide a supplementary information about the user’s gesture. Also, flex sensors in general are not known for their accuracy.

Test Procedure:

To test accuracy of the overall system, we need to first test the accuracy of the IMU and the accuracy of the flex sensor unit separately.

Testing flex sensor unit:

The flex sensor only has to distinguish whether the bend is between 0° to 45° (closed fist) or between 45° to 90° (open palm). The following is the test procedure for testing the flex sensors:

1. Mount all flex sensors on the hardware.
2. Bend each sensor in the range of 0° to 45° 100 times and record the output angles from the flex sensor in a binary representation. We will test to 5° of granularity, giving 10 total tests for each range. If the output angle falls within the range, then record as 1; otherwise, record as 0.
3. Repeat the same experiment for the range of 45˚ to 90˚.
4. Compute the accuracy by summing up the total all the results and divided it by the total number of results.

**Testing IMU:**
The main components from the IMU that are useful for our system is the accelerometer and gyroscope. Both units should have an accuracy no lower than 90%. The overall accuracy of IMU is the average accuracy of the accelerometer and the gyroscope.

1. **Testing Accelerometer:**
   1. Test the accuracy of the gravitational acceleration
   2. Use measurements of another accelerometer with known superior performance (with accuracy within 1%) as the ground truth and compare it with the measurements with our accelerometer readings to obtain the accuracy for our accelerometer.

2. **Testing Gyroscope**
   1. Make sure that in the duration of 4 four hours, the drifting effect of the gyroscope will not make the accuracy of the gyroscope unacceptable:
   2. Run the gyroscope for four hours.
   3. Set the gyroscope on a constant rotation rate of 60˚/s around all three axises in the phone coordinate system, and record the readings from the gyroscope for all three axises at the same time.
   4. Compute the accuracy of the gyroscope by comparing the measured value with the ground truth.

**Derive the accuracy of the sensor unit:**
After testing IMU and gyroscope individually, we can derive the overall accuracy of the sensor unit by using the accuracy function.

5. **Cost**
   Our estimate is that in total each group member will contribute 100 hours to the project, averaging 12.5 hours per week. This estimate comes from a conservative aggregation of the amount of time it will take to implement the simpler features and the extended testing that will take place in the last three weeks of the project. Four prototypes will be built.

**Labor:**
● $(40/\text{hr}) \times (2.5) \times 100 \text{ hours} = \$10,000/\text{partner}$
● $(\$10,000) \times (3 \text{ partners}) = \$30,000$

Parts:

<table>
<thead>
<tr>
<th>Part</th>
<th>Cost (per unit)</th>
<th>Quantity (units)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSP430 (Microcontroller)</td>
<td>$1.54</td>
<td>10</td>
<td>$15.40</td>
</tr>
<tr>
<td>785060 2500MAh (Battery)</td>
<td>$14.95</td>
<td>10</td>
<td>$149.50</td>
</tr>
<tr>
<td>LSM303DLHC (IMU)</td>
<td>$13.46</td>
<td>10</td>
<td>$134.60</td>
</tr>
<tr>
<td>239-BT800 (Bluetooth unit)</td>
<td>$10.60</td>
<td>10</td>
<td>$106</td>
</tr>
<tr>
<td>Elastic Bands</td>
<td>$1</td>
<td>20</td>
<td>$20</td>
</tr>
<tr>
<td>Other (resistors, capacitors, etc.)</td>
<td>$50</td>
<td>1</td>
<td>$50</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>$475.50</td>
</tr>
</tbody>
</table>

Grand Total: $30,475.50
6. Schedule

Week 1:
Research and Preparation
- Adam: Sensor Research and Purchase
- Jeremy: Microcontroller and Memory Subsystem Research and Purchase
- Ziheng: Powerboard Design and Printing (or Purchase)

Week 2:
Sensor and Power System Validation
- Adam: Complete IMU Validation
- Jeremy: Complete Bluetooth Validation
- Ziheng: Complete Powerboard Validation

Week 3:
Subsystem and Architecture Design
- Adam: Microcontroller State Machine Framework
- Jeremy: Control Word Specification and Serial Interfaces
- Ziheng: State Machine / Interface input

Week 4:
Architecture Implementation (Units)
- Adam: Flex Sensor Module
- Jeremy: Bluetooth Transmitter Module
- Ziheng: Accelerometer Module

Week 5:
Architecture Implementation (Integration)
- Adam: Flex Sensor to Microcontroller
- Jeremy: Microcontroller to Bluetooth
- Ziheng: Accelerometer to Microcontroller

Week 6:
Mobile Application Implementation (Units)
- Adam: Pattern Recognition Algorithm (acquisition and processing)
- Jeremy: Sheet Music Generation
- Ziheng: Audio Processing

Week 7:
Mobile Application Implementation (Integration)

- Adam: State Machine Framework
- Jeremy: State Machine Framework/User Interface
- Ziheng: User Interface

Week 8:
Full System Integration Testing

- Adam: Serialization and Deserialization Testing
- Jeremy: Bluetooth Communication Testing
- Ziheng: User Experience Testing

References