Keep in Touch

Niranjan Kumar, Amy Liao, Siddarth Sen, Eric Tu EECS 149/249A

Project Vision

In many parts of the world, touch typing has become such a ubiquitous skill that children learn it in their first few years of schooling. Although numerous interactive learning systems for touch typing exist, currently no software exists to provide feedback on whether the correct finger is used to press each key or the force with which the key was pressed. Incorrect finger usage and typing force not only reduces typing efficiency, but could also lead to ergonomic injuries later, such as repetitive motion injury and carpal tunnel syndrome. To address this problem, we have developed a smart glove capable of correlating the finger used to each keystroke, thus enabling the learning software to track finger usage and pressure while typing.

Objectives

- Design and optimize a smart glove with pressure sensors located at the fingertips
- Develop an interactive learning software for touch typing that integrates pressure sensor feedback with visual feedback for the user
- Integrate software and hardware components

Design Criteria

In designing the Smart Glove, we considered several key design criteria that needs to be met. First, our system must be able to detect all keystrokes and the associated finger pressures generated by the user. We set the design goal such that our system should be able to accurately detect key presses at 212 words/minute or 17.8 keystrokes/second, a rate achieved by the world record holder. In reality, most users of a touch type learning program will type under the average speed of 40 words/minute or 3.3 keystrokes/sec [1]. In addition, since different keyboards vary in the force required to register a key press, our force sensors must be able to detect forces in the range of 0.25N to 0.85N for our Smart Glove to be compatible with all keyboards [2]. Finally, the system should be highly accurate and achieve at leave 90% sensitivity and specificity.

Systems Overview

The Smart-Glove system consists of an instrumented glove containing force sensors, a Freescale mbed microcontroller[2] to conduct basic thresholding processes, and a software-based typing framework that compares keystroke data with force data and displays user feedback on a GUI. When the user presses a key, two inputs are generated and passed to the system. First, the key listener on the PC registers a "key pressed" event. Simultaneously, the resistance of the force sensors on the finger used will drop below a threshold value. An mbed microcontroller collects and conducts basic thresholding of the data from the force sensor. The typing framework receives data from the mbed software over serial connection using the RXTX Library[4]. In addition, the framework is notified of "key press" events by the key listener. The Typing Framework integrates this data and uses it to provide feedback via the GUI so that the user can correct the fingering or pressure used.

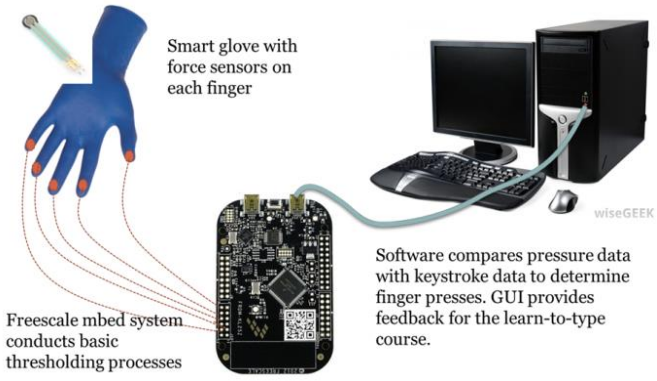

Figure 1. Schematic of hardware setup

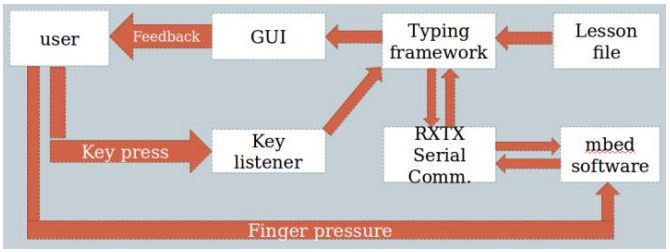

Figure 2. Software architecture

Instrumented Smart-Glove

In order to map key strokes to the fingers being used, a force sensor is attached to each finger of the Smart Glove. The FSR400 small round force sensing resistor from Interlink Electronics was used to measure the finger pressure. The FSR400 is a flexible force sensor with a 7.62 mm diameter force sensitive area that fits easily on the tips of the fingers without hindering finger movements.

These small force sensors contain a piezoresistive component. In the non-actuated state, the force sensors have a resistance of 10 MΩ. When actuated, the resistance of the force sensors (2.5 kΩ to 1 MΩ) varies linearly with respect to the amount of force in the range between 0.1-10N, where a lower resistance value corresponds to a higher force. Equation 1 shows the relationship between force and resistance, fitted to an affine model. The active range of these force sensors is able to detect the range of forces necessary to actuate all types of keyboards as specified in the design criteria.

$$Resistance = -100758Force + 1010076$$

Equation 1. Affine relationship between force and resistance

Figure 3. Force sensor

A voltage divider is used to measure the resistance of the force sensor (figure 3). The voltage divider is powered using the MBed's 3.3V output. A 1 kΩ resistor (R_1) is used on each branch of the voltage divider to ensure that the current never exceeds 3.3 mA and that the 3.3V power is never shorted to ground.

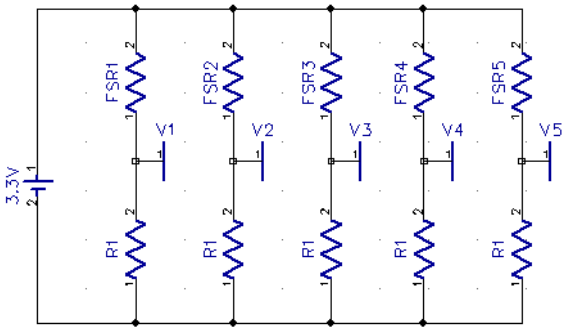

Figure 4. Voltage divider circuit

The resistance of the force sensors is calculated using equation 2.

$$R_i = 1000\left(\frac{1 - v_i}{v_i}\right)$$

Equation 2. Resistance of force sensor where v_i is the voltage measured by the voltage divider.

Freescale mbed Board

A Freescale FRDM-KL25Z, an mbed Kinetis board [2] with an ARM Cortex M0+ processor [3] is collects and conducts basic thresholding of the data from the force sensor.

Since the mbed board only has 6 analog channels, two mbed boards must be used to acquire data from both hands. For each Mbed board, channels PTB0, PTB1, PTB2, PTB3, and PTC2 were used corresponding to the pinky, ring finger, third finger, pointer finger, and thumb respectively.

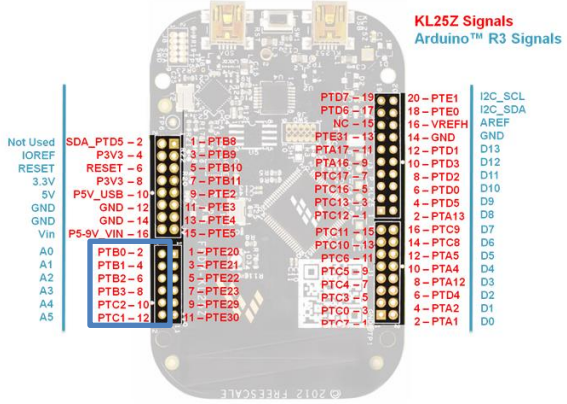

Figure 5. Mbed LK25Z Freescale Mbed signal

Based on calibration data from an experienced typist, we have set the threshold at 100 kΩ for a finger press detected and 10 kΩ for finger pressed too hard. The thresholds were determined by having the user type a calibration sentence "The quick brown fox jumps over the lazy dog." while recording the forces necessary actuate each key press 10 times. The threshold for finger presses was chosen to maximize detection of true presses, while minimizing false positives due to the overlap between the observed resistance ranges for finger presses and resting finger positions.

Typing Framework

The Typing Framework is in charge of the PC side of the Keep In Touch system, and is implemented in TypingFramework.java. As mentioned in the Systems Overview section, the KeyListener send "key press" data to the Typing Framework every time the user presses a key on the keyboard and each of the two mbed boards, one for each hand, send finger data to the PC 40 times per second via the serial port, which is read by the Typing Framework using the Java RXTX serial communication library [4]. The Typing Framework is

responsible for processing these three data streams and turning them into meaningful feedback for the user. To do this, it first efficiently interleaves the two high-throughput finger data streams from the two mbeds to produce one stream with the data points in chronological order. It then compares the resulting finger data stream with the key press stream gathered from the KeyListener and performs inference to determine which finger was most likely used to press each key in the key press stream. It also records if a finger was pressed too hard. In addition, the Typing Framework also reads from the lesson file and uses it to assess correctness of the keys typed. Thus, for each character in the lesson, it will figure out if the user typed it correctly, typed the wrong key, typed with the wrong finger, typed too hard, or made some combination of these three errors. Finally, it computes some basic statistics such as accuracy and words per minute. It sends the typing feedback and statistics to the GUI which displays it to the user.

Timing and Synchronization

Accurate timing and synchronization is crucial to correlating finger press data with key press events. Our typing program must be able to detect at least 1060 keystrokes/min or 17.7 keystrokes/sec, which is the typing speed of the current world record holder. This is far faster than a beginning typist learning to type using our program. However, to be safe, we have decided to use the world record holder's typing speed as the design criteria. To detect 18 keystrokes/sec, based on the Nyquist theorem, the force sensors must be sampled at twice the frequency, or 36 keystrokes/sec. Since our Mbed code is capable of sampling all finger pressure data at a rate of 25 samples/sec, no data will be lost due to undersampling.

To establish synchronization, we used the Ticker object [5] from the mbed.h library. The Ticker object generates timed interrupts at a user-specified rate. It can be tied to interrupt service routines via its attach() method. The Typing Framework on the PC will send a signal to each of the two mbed boards to start the Ticker, while recording the PC timestamp at that instant. Once the Ticker has been started, each mbed will continuously measure the resistances of the force sensors and compare it with the thresholds for finger pressed and force overload at a sampling rate of 25 samples/sec. Based on whether the thresholds for any of the fingers were achieved, a finger ID is generated and stored in the mbed serial port buffer The data is stored in a 4 byte packet, with the first byte representing the finger ID and the last 3 bytes encoding the timestamp.

On the PC side, the Typing Framework collects data from both mbeds and interleaves them to create a stream of data points corresponding to finger presses from both hands that is in chronological order. The Typing Framework must transfer data from the RXTX serial port input stream [4] buffer to PC memory every two seconds. This is because the input stream is just an abstraction hiding the serial port buffer handled by the serial port hardware. At the rate data arrives from the two mbed boards, this buffer will overflow shortly after two seconds of neglect.

Inference Problem:

To eliminate nonsensical force sensor data (eg. from random finger taps) or to break ties, we can calculate the probability that a finger was used given a keypress to determine which finger was most likely used. To do so, we make a few key assumptions:

- The timestamp of the finger press and the timestamp of the keypress are very close
- For any given finger, it is unlikely that a key far from that finger was pressed

Based on these assumptions, we can solve for the Maximum a Posteriori estimate of X given Y to find the X which maximizes P(Y|X) using the equations listed in equation 3. Let X = finger used + timestamp recorded on MBED and Y = key pressed + timestamp recorded on PC.

$$MAP[X|Y=y] = argmax_x \frac{P(Y=y, X=x)}{P(Y=y)}$$
$$MAP[X|Y=y] = argmax_x P(Y=y, X=x)$$
$$MAP[X|Y=y] = argmax_x P(Y=y \mid X=x) P(X=x)$$
$$MAP[X|Y=y] = argmax_x P(Y=y \mid X=x)$$

Equation 3. Inference problem

This set of equations further assumes an uniform distribution of X (that the user is equally likely to have selected any key). To further improve the accuracy of this model, we can add a non-uniform prior to better predict finger usage.

GUI:

The GUI was programmed using the Java swing and AWT code libraries, which contain classes for making graphical components, such as buttons, labels, and frames. The user interface layout prompts the user to type certain phrases and displays relevant statistics, such as words per minute, percentage correct, and feedback on finger usage. The user will attempt to type the prompted sentence using the correct finger. 10 colored boxes located below the prompt sentence inform the user about which fingers to use to press the next key. As the user types, a cursor located below the prompt sentence indicates where in the sentence progression the user is at and the box corresponding to the finger to use turns

yellow. After typing the full sentence, the GUI will show what the user has typed with each letter color-coded to provide feedback on whether the correct finger and key were pressed.

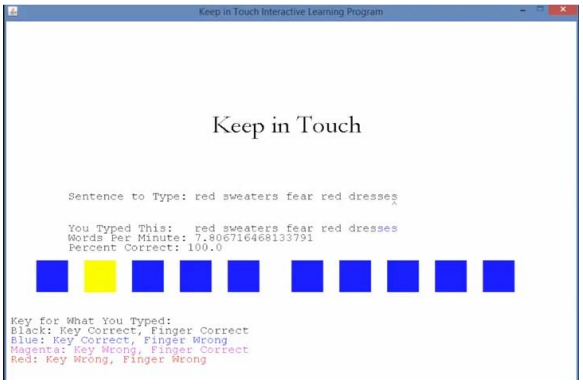

Figure 5. GUI layout

Future Directions:

In order to improve the Smart glove design, we would look into other force sensors. The FSR 400 force sensors were too small to accurately detect finger presses in all hand orientations. Finger pads versus fingertips are used to actuate keys in the extended and curled positions respectively. In order to be useful, the Smart glove must be able to detect both of these cases. However, due to the small size of the sensors, only forces at the finger tips were detected reliably, causing a few extended finger presses to be missed. In addition, flexing the fingers without pressing key occasionally places a small degree of pressure on the force sensors, which can trigger a key press reading. A larger, more flexible sensor is needed to capture all finger press readings. In addition, an improved mounting procedure would help reduce the number of false positives.

To further improve the sensitivity and accuracy of the system, we can incorporate a non-uniform prior assumption to the inference problem to improve our finger usage predictions. We can also add in a calibration mode in the GUI that the user must complete at the beginning of each session to calibrate for the user's natural typing and resting force of each finger as well as variation in keyboard actuation forces. This data will be used to further optimize the thresholding and smoothing functions to improve the accuracy of the device and reduce the number of false positives.

In addition, to improve user experience, we can also display additional statistics in the GUI to give the user more feedback on how much pressure is used on average for each letter or finger. Finally, we can also implement an additional mode to track user history throughout the learning course.

Conclusion:

In conclusion, our team has built and demonstrated a Smart Glove prototype designed to provide touch type learners with more information about finger usage and finger pressure. Small FSR400 force sensors placed at the tips of each finger sense when the finger has been used to press a key. Two Mbed boards collect and process the data before sending it via serial connection to a Java-based Typing Framework on the PC, which then compares finger pressure data with Keystroke data to determine finger usage. Finally, feedback is displayed to the learner on an intuitive graphical user interface. In addition to the speed and accuracy emphasized by existing typing learning programs, our device will help new typists develop good ergonomic habits, which will improve their health and well-being in the long run.

References

1. Average Typing Speed Infographic. http://www.ratatype.com/learn/average-typing-speed/
2. Gerard, M.J., Armstrong, T.J., Franzblau, A., Martin, B.J., Rempel, D.M., 1999. The effects of keyboard stiffness on typing force, finger electromyography, and subjective discomfort. Am. Ind. Hyg. Assoc. J. 60, 762–769.
2. Freescale FRDM-KL25Z mbed http://developer.mbed.org/platforms/KL25Z/
3. ARM Cortex M0+ http://www.arm.com/products/processors/cortex-m/cortex-m0plus.php
4. Jarvi, Trent, et al. "RXTX: A Java Cross-Platform Wrapper Library for the Serial Port" 1997-2014 The GNU Project. https://github.com/rxtx Available under the GNU Lesser General Public License (LGPL) http://www.gnu.org/licenses/lgpl.txt
5. "Ticker Interface." ARM mbed Handbook. https://developer.mbed.org/handbook/Ticker