Behavioral Biometric Classifier on 8051 Microcontroller*

Abdullah Elkady, Mohamed Ibrahim, Amr Kayid, Ahmed Shawky, and Ahmed Sabek

German University in Cairo, Egypt
{abdullah.elkady, mohamed.meeza, amr.kayid, ahmed.ahmedhussein, ahmed.sabek}@guc.edu.eg

Abstract. A simple behavioral biometric classifier on 8051 Microcontroller that can identify users based on their keystroke dynamics profiles. Implemented using C language on Keil C51 development tool.

Keywords: Behavioral Biometric · Classifier · Microcontroller · Dwell time · Flight time.

1 Introduction

1.1 behavioral biometric

In this project we used the behavioral biometric of Keystroke Dynamics to identify an individual's character typing profile on a keyboard or keypad. The keystroke rhythms of a user are measured to develop a unique biometric profile of the user's typing pattern for future authentication, the project idea is similar to the one that was used by Coursera [1] for verifying learners before submitting graded quizzes and assignments.

Keystroke dynamics can therefore be described as a software-based algorithm that measures both dwell and flight time to authenticate identity. where dwell time is the duration that a key is pressed, while flight time is the duration between keystrokes.

* Supported by CSEN 702: Microprocessors Course

1.2 Project Idea:

This project is divided into two phases, traning and testing phase.

In the training phase, the participants are asked to enter a certain word **".tie5Ronal"** 5 times, then during the training, the average flight time of each character in the word is calculated, taking the average of the 5 trails, a unique profile is created for each participant which will be used to verify the user in the testing phase.

In the testing phase, one of the participant will be selected randomly and will be asked to enter the same passcode ".tie5Ronal", then our implementation measures the distance between the test vector and the saved profiles and identify the user's identity based on the shortest distance between the two vectors, using the Euclidean metric.

2 Implementation

2.1 Brief Description

Our implementation starts with initializing and setting the serial port for 2400 baud at 12MHz, and enabling mode 1 (8-bit UART) to enable receiver, along with enabling the interrupts for T0 and the global interrupts.

After the initialization phase, we continuously check for training and testing phases by checking on **P1.1**. If we are in the training phase, we check for which user profile (*A or B*) will be trained, then the training phase starts for the selected user using the following functions

```
void trainUser(unsigned int timings[])
```

This function calculate the total training time for the user in the 5 attempts, the function calculate the time of each keystroke usign the following function

```
unsigned char *calculateTimings()
```

after 5 successful attempts of input runs, we average the resulting time and save it in the user's profile.

In the testing phase, a user is selected randomly and asked to enter the same passcode, then we calculate the timing of the user, and using the Euclidean Distance Algorithm in the following function

```
unsigned int calculateEculidean(unsigned int timings[])
```

we determine the users identity by the closest vector in the saved profile.

2.2 Team Work

The code was divided into several steps and functions as follows:

- ***Serial port Setup and Initialization***

```
void init()
```

- ***Time Setup and Calculations***

```
void resetTimer()
void stopTimer()
void handleOverflow(void) interrupt 1
```

- ***Training Phase***

```
void trainUser(unsigned int timings[])
```

- ***Similarity Calculations***

```
unsigned int calculateEculidean(unsigned int timings[])
```

- ***Testing Phase***

```
unsigned char *calculateTimings()
void flashLED(void)
```

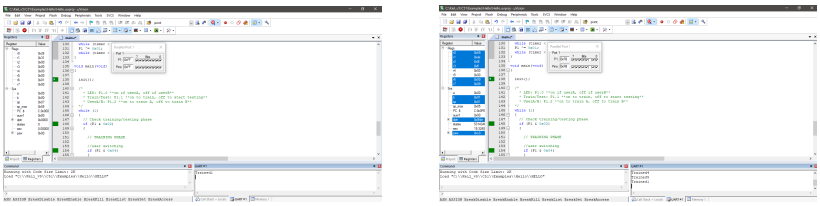
The work was distributed among the team members equally and they worked together to combine the final functions and testing the final version of the system.

Our Implementation and source code can be found here[1]

[1] https://github.com/AbdullahKady/behavioral-biometric-classifier

3 User Guide

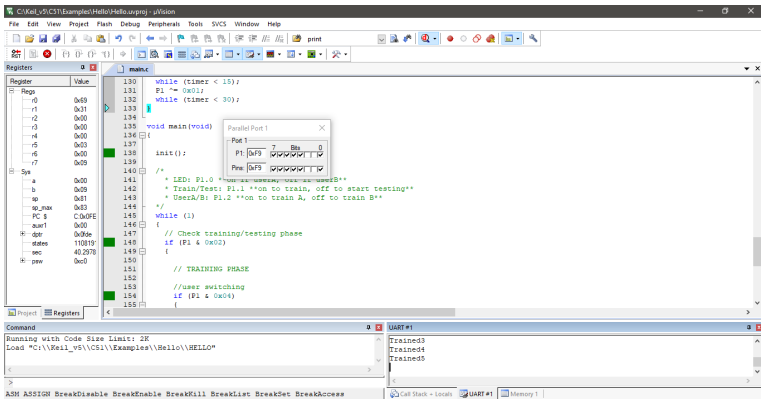
Our System starts with checking on **P1**
then we start by training two user ***(A & B)***, as indicated in the following two figures.



(a) User A (b) User B

Fig. 1: Training User A & B

After the training is done, we start the testing phase and the system is able to indicated which user is using the system now as indicated in the following figure.



(a) Testing

Fig. 2: Testing Phase

4 Results and Future Work

Testing our implementation, we have tried it against multiple participants ($\leq$ 10), and it worked perfectly for this small set of tests.

However our implementation faces 2 aspects which can be enhanced:

- The accuracy of measuring the average profile of a user, this has two subcategories: Increasing the training phase measurement count, for instance; do 10 runs instead of 5 runs, this would enhance the values of a user to be a more accurate representation of his expected typing speed.

- Theres a problem with our average being biased in case one of the entries had a relatively big weight, adding the standard deviation of the samples as a factor of calculating the average, would lead to better and more accurate results.

References

1. A. Maas, C. Heather, C. Do, R. Brandman, D. Koller, A. Ng, Ubiquity symposium: MOOCs and technology to advance learning and learning research: offering verified credentials in massive open online courses, Volume 2014, Number May (2014), Pages 1-11, ACM Digital Library