Faculty of Media Engineering and Technology

Dept. of Computer Science and Engineering

CSEN 702: Microprocessors

Winter 2018

Course Project Specification Document

Deliverable Name: Behavioral Biometric Classifier on 8051 Microcontroller

Grading Weight: The full project accounts for 20% of the course mark (10% for the implementation and 10% for the report).

Deliverable Overview: The goal of this deliverable is to develop a simple classifier on 8051 Microcontroller that can identify users based on their keystroke dynamics profiles.

Development tool: Keil C51, https://www.keil.com/demo/eval/c51.htm

Implementation language: C

Team Size: 1-5. You should send your teams to the TA by email on or before Thursday, October 25th 2018.

Code size: Less than 2KB

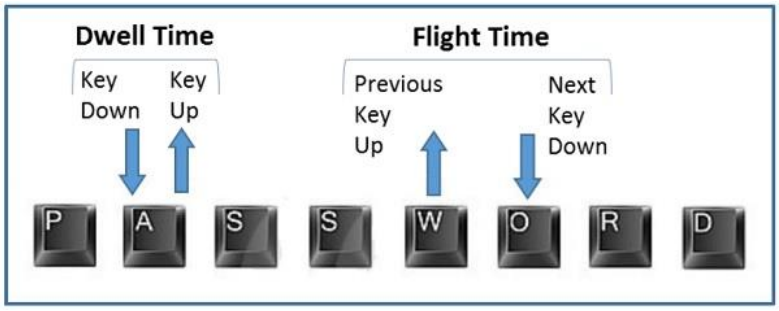
Important Plagiarism notice: You have to write your own code from scratch. Deliverables based on others code will receive a grade of zero in the entire deliverable (even if the code is heavily refactored/modified, etc…).

Deliverable Deadline: Tuesday, November 20th, 2018.

Evaluations will take place in the following week according to a schedule that will be posted on the MET website.

1. Introduction

The behavioral biometric of Keystroke Dynamics uses the manner and rhythm in which an individual types characters 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.



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

2. Project Description:

In this project, you will write a C code on an 8051 Microcontroller to measure the flight time between different keystrokes. Participants will be asked to enter the word ".tie5Ronal" 5 times and then your code will calculate the average flight time of the following keys and creates a profile for each participant. This process is called the training phase.

f	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8	Feature 9
	Flight Time (.-t)	Flight Time (t-i)	Flight Time (i-e)	Flight Time (e-5)	Flight Time (5-R)	Flight Time (R-o)	Flight Time (o-n)	Flight Time (n-a)	Flight Time (a-l)

For simplicity, we will consider just receiving a character as a single event (we don't have to distinguish between up and down events). After saving the profiles of participants during the training phase, your code will start the test phase. In the test phase, one of the participants will be selected randomly and will be asked to enter the same passcode ".tie5Ronal". Your code should then measure the distance between the test vector and the saved profile and identify the user based on the shortest distance.

3. Simulator inputs:

1. Start Training Phase
2. User A gets selected
3. User A inputs ".tie5Ronal" on the serial port 5 times.
4. User B gets selected
5. User B inputs ".tie5Ronal" on the serial port 5 times.
6. The system extracts the features and creates a profile for each user

7. Switch to Test Phase
8. Random User (A or B) inputs ".tie5Ronal"
9. The system measures the distance between the input vector and the saved profiles and classifies the user as A or B based on shortest distance

4. I/O Connections:

1. Switch 1 – (Training vs Testing)
2. Switch 2 – (User A vs User B)
3. LED – User Classification results (User A (Flash once) , User B (Flash twice))
4. The UART interface.

5. System Implementations Guidelines:

- Write a program to configure your Microcontroller's UART to get keystrokes
- Write a program to identify the received characters from the UART
- Write a program that reads a port and makes a branching decision based on the value.
- Write a program to use the timers/counter to count the time between two events.
- Write a program that calculates the average values and saves them in temporary memory locations.
- Write a program that calculates the Euclidean distance between two vectors.
- Write a program that flashes an LED in two different ways.

6. Decision Making

To simplify the calculations, the Microcontroller might measure the distance between the test vector and each user's profile as follows (using Euclidean distance):

$$D_a = \sqrt{\sum_{i=0}^{n} (f_{ti} - f_{ai})^2}$$

$$D_b = \sqrt{\sum_{i=0}^{n} (f_{ti} - f_{bi})^2}$$

n is the number of features -1 (n = 8), f_{ti} is the feature i of the user under test, f_{ai} is the feature i of the user A, f_{bi} is the feature i of the user B.

If ($D_a < D_b$) , then the user is classified as D_a, otherwise D_b

7. Project Report:

In addition to your team member names, the report should include:

1. A brief description of your implementation
2. A summary of how the work was split among your team members (who did what exactly)
3. A user guide including a full simulation example step---by---step with snapshots for the training and the test phase.
4. A brief discussion of the obtained results, including the performance of your system.
5. A copy of your code (separate C file).

8. Marking Rubric (Implementation)

Component	Marks	Description
User Interface	10	- The microcontroller can receive keystrokes from the users through the UART.
Timer Configuration	55	- The microcontroller uses the timers correctly to calculate the time of each feature.
I/O, Memory and Register Access	20	- The microcontroller creates the feature vector of each user through the training phase.
Decision making and LED flashing	15	- The microcontroller classifies the user under test by calculating the distance in the test phase.
TOTAL	**100%**	

9. Marking Rubric (Report)

Component	Marks	Description
Description	20	- A brief description of the implementation.
Team work	20	- A summary of how the work was split among the team members.
User Guide	20	- A user guide including a full simulation example step---by---step with snapshots for the training and the test phase.
Results	20	- A brief discussion of the obtained results, including the performance of your system
Code	20	- A copy of your code (separate C file).
TOTAL	**100%**	