NAIT

Edmonton, Alberta

Making a Reaction Time Testing System

As a submission to

Marc Anderson, Instructor

Kelly Shepherd, English and Communications Department

Submitted by

Andy Tran, Student

CMPE2960

Computer Engineering Technology

December 9, 2016

12345 – 67 Ave NW

Edmonton, AB T5R 1A1

November 19, 2013

Marc Anderson, Kelly Shepherd

Instructor, English and Communications Department

NAIT

11762 – 106 St NW

Edmonton, AB T5G 2R1

Dear Marc Anderson and Kelly Shepherd,

I am submitting the report “Making a Reaction Time Testing System”, as you requested, for your evaluation.

The report will outline the purpose and processes of the testing system and its application for measuring cognitive capabilities. It details the various frameworks and technologies used and their integration into the overall system.

I would like to thank all the instructors who have offered advice and guidance on several aspects of the project. With their help, my partner and I were able to overcome various difficulties in the implementation of the reaction time testing system.

Sincerely,

Andy Tran

CNT Student

Table of Contents

List of Figures and Tables

Figures

Tables

Abstract

Mental chronometry is the study of the brain's processing speed. As an aspect of cognitive psychology, mental chronometry provides a method to evaluate the effectiveness of the brain's ability to focus on an event/task. A way to quantify this is to take a measurement of an individual's response/reaction time to a visual or auditory stimulus. A simple response used is often a button press but can also manifest in other forms such as eye movement or having a vocal response; any observable behavior that can be measured or tracked is sufficient. Some forms of stimuli used in testing include a light turning on, a sound being played or physical contact with an object.

Consider an example where a driver encounters a roadside hazard suddenly appearing in front of them. In response to this visual stimulus, various decisions and actions need to be made as soon as possible. Hands move to stir the wheel, the feet move to press the brakes, and the eyes scan the environment for a safer location. All of the information absorbed as well as any actions taken are part of the cognitive processes that occur in response to stimuli. Observing and measuring these physical behaviors and the timing that these behaviors occur can provide an insight to the thought processes and mental capabilities of an individual.

Reaction Time Tester

1.0 Introduction

The cognitive psychology of the mind has been an area of study for a significant part of human history. As early as 387 BCE, Plato proposed that the psyche was the seat of mental processes. One way to measure the effectiveness of the human mind is through the use of mental chronometry, or simply the measurement of one's response time (RT) to a stimulus. Studies into an individual's RT have been conducted across various branches of psychology. Sir Francis Galton, a major contributor and perhaps founder to differential psychology, used reaction time tests to measure the mental differences between individuals (Arthur, 2006).

The purpose of this report is to walk through the process of designing and building a reaction time testing system. There are many types of reaction tests available but the focus of the report shall be the use of a test which involves a visual stimulus (green LED) which will require a physical response (button press). This relatively simple test will help to minimize the distractions that may be present to achieve a higher accuracy of response times. Alternative reaction tests may include the use of an auditory stimulus as well as the measurement or tracking of other physical responses such as eye movement.

2.0 Project Outline

This section will outline the design of the reaction time test system, detailing the purposes and functions of the project and each of its component and material.

2.1 Overview

One goal of a reaction test is to provide insight to an individual's cognitive abilities. A way to quantify this ability is to measure the physical responses caused by a stimulus. This reaction test system will make use of 1 visual stimulus in the form of a green light and 1 physical response in the form of a button press. At a random time, the green light will turn on, starting a timer. In response to the light turning on, the user will press the button, stopping the timer and turning off the light. The difference in the start and end points of the timer will determine the response/reaction time. Afterwards, the data collected shall be stored into a database and can also be viewed through a website.

2.2 Design and Features

The testing system will need to handle data in 3 ways: Retrieval, storage, and feedback.

Data retrieval occurs during the testing procedure when an individual presses the button in response to the green light turning on. To facilitate this test procedure, a device is needed to capture the response times and manage the light and button states. For this, a 9S12 microboard will be used. Further details of the data retrieval phase will be discussed in section 3.0.

To enable the retrieved data to be stored, it will first need to be processed and packaged so that it can be transferred into a database. For this, a C# application is used to manage communications between the microboard and a database through serial communication. As well, this application will act as a general user interface and handle user logins so that retrieved test data can be associated to a user. Section 4.0 will further cover the specifics of the C# application.

To store the data, a database will be used to hold the test results. The database's design and implementation will be covered in section 5.0.

As a testing system, feedback should also be displayed back to the user. Therefore, the data in the database has been made publicly available through a website. An overview of the website design and features is covered in 6.0.

3.0 Hardware Implementation

A MC9S12XDP512 microboard is used to run the testing procedures. The microboard will be responsible for capturing the timings of the button press, connecting the external circuitry required, managing test states, and sending the data over a serial connection. Specific functions and specifications of the microboard can be found in the MC9S12XDP512 Data Sheet from Freescale's website.

3.1 Circuitry and Wiring

For the test, an external button will need to be connected to the microboard. To connect it, a logic switch circuit connected to PortJ will be used. PortJ was chosen as it can be configured to trigger an interrupt when the button is pressed. When the button is pressed, a logic high of 5V will be present along a wire connecting the circuit to PortJ0/ pin 22 of the microboard. Otherwise, a pull-down resistor will bring the voltage to 0V. Additionally, the center switch found on the microboard has also been wired to PortJ, serving as a backup to the external button. Shown below is a circuit diagram for the external button connection.

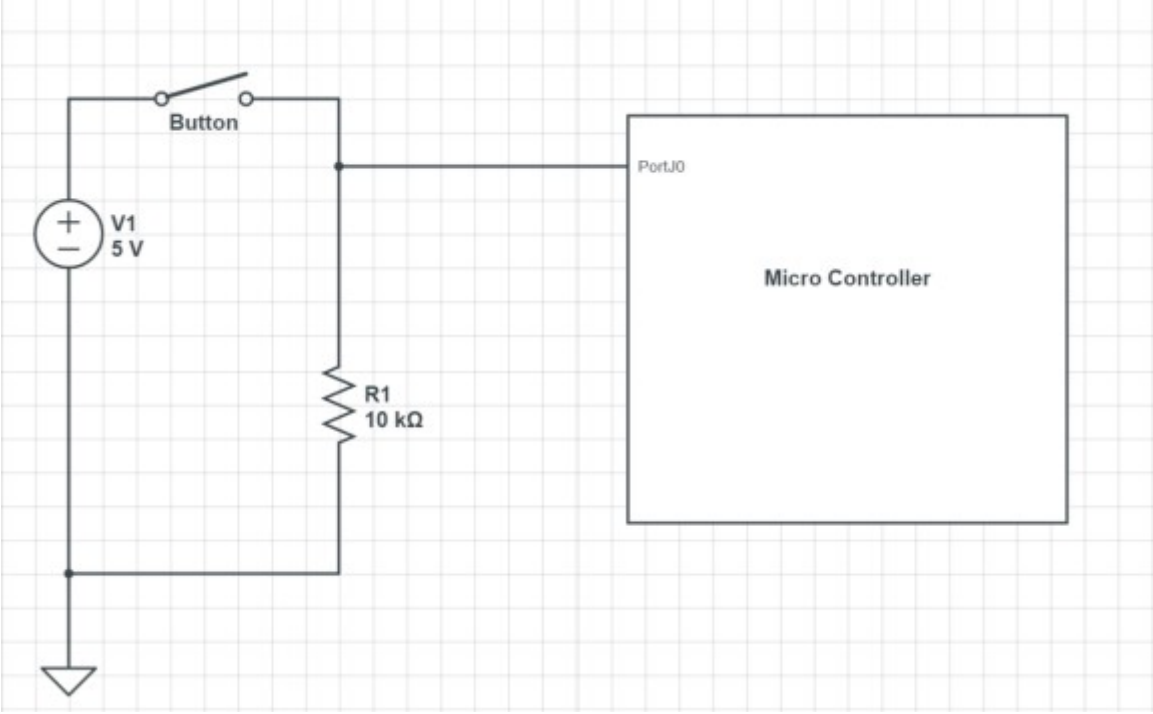


Figure 1: External Button Circuit Diagram

3.2 Microboard Code

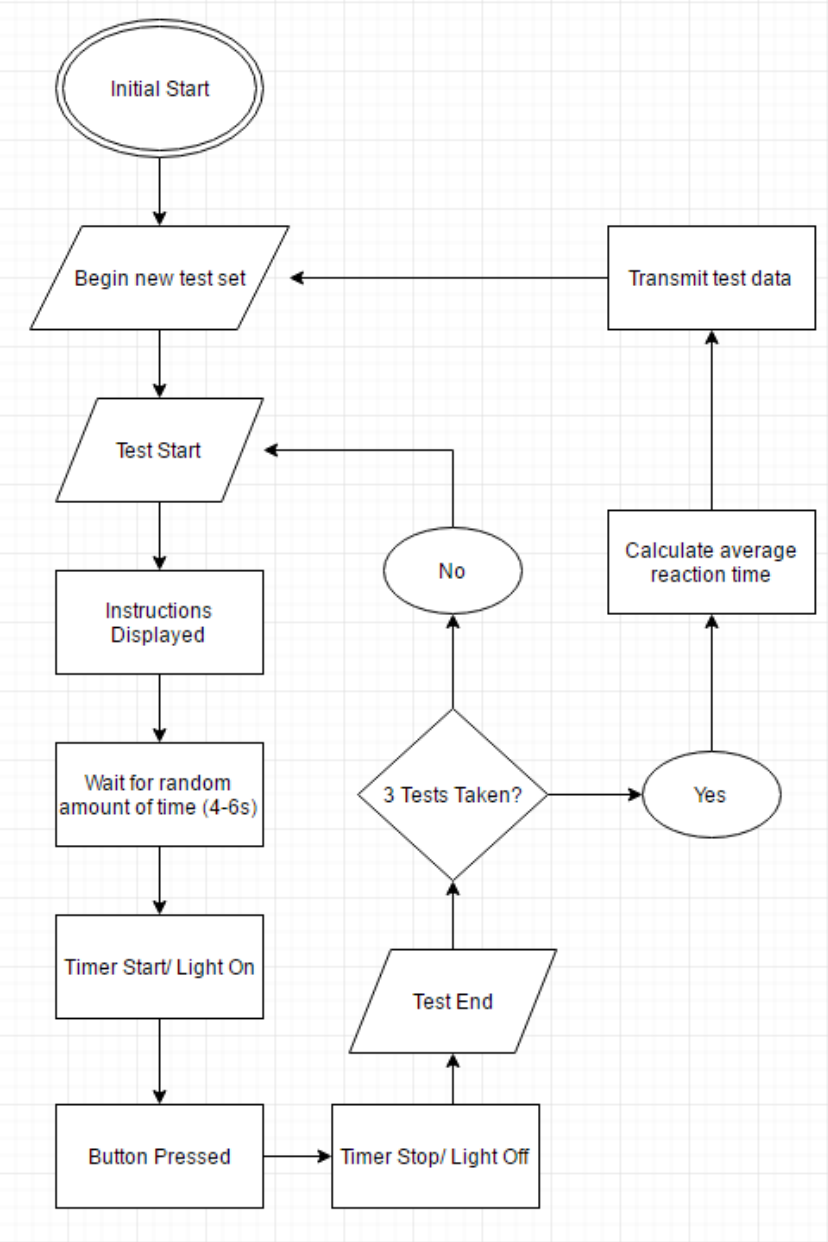


Figure 2 - Microboard program flowchart

The flowchart above outlines the program flow of the microboard. The program used to run the testing procedure has been made using C. After running initialization procedures, the program will enter a standby phase where it will wait for a user to press the button to indicate that the test should start. A random time between 4-6 seconds will be chosen

after which a green LED till turn on. During this process, a timer is started to keep track of the user's response time. Once the green light is on, a button press will trigger an interrupt service routine(ISR) for PortJ. The ISR will signal the end of the test and get the stop value of the timer. Afterwards the program will then run the test 2 more times to get an average reaction time from a total of 3 tests. Once an average is collected, the results are packaged into a series of bytes and transmitted via the serial communications interface. The program will then reset to its initial state, awaiting the start of a new set of tests.

3.2.1 Timer Configuration

The timer used is actually a counter that increments at a rate of 8MHz before pre-scaling. Each count in the timer represents a set value of time. For this project, a pre-scaler of 64 was chosen. This results in each count representing a time value of 8μs. For the tests, an accuracy to 1ms was required, therefore every 125 counts in the timer will signal that 1ms has passed. However, since the timer only counts from 0x0000 to 0xFFFF, a counter is used to keep track of the number of overflows that occurs so that no data is lost. When the green light is turned on, the current value of the timer is captured as a start point. After the user presses the button, the green light is then turned off and the current timer value is then read again for an end point. The difference between the end and start counts will represent the reaction time of the user, adjusting for any overflows that may have occurred. The formula to convert the counts into time using a pre-scaler of 64 s as follows:

$T = (T_2 + TOFCount * 65636 - T_1) * 8\mu s/count$

where: T = Reaction/response time (μs)

T_2 = End time(counter value)

T_1 = Start time(counter value)

TOFCount = # of timer overflows that occurred

65636 is the full range of the timer

8μs/count is the pulse width of the timer, determined by the pre-scaler value

3.2.2 Interrupt Routines

Two interrupt service routines were configured to help run the testing procedure. The first is the timer channel 0 ISR which is set to occur every millisecond. The timer ISR will

check if a timer overflow has occurred by reading the 8th bit on the TFLG2 register. If the flag is set, a counter for the number of overflows will increment and the flag will be cleared. Bellow is a code snippet for the timer ISR:

```
//Interrupt for timer channel 0
//Description:  periodically checks the timer to see if
//              and overflow occured. A running counter
//              will track the overflows counts
//              for response time calculations
interrupt VectorNumber_Vtimch0 void TimerInterval(void)
{
    TFLG1 = 0b00000001;  //acknowledge interrupt

    //check if timer overflow occured
    if((TFLG2 & 0b10000000) == 0b10000000)   //Checks the TOF flag in the TFLG2 register
    {
        TOFCount++;           //add to overflow counter
        TFLG2 |= 0b10000000;  //clear TOF/overflow flag
    }

}
```

The second ISR is responsible for any PortJ interrupts. This is caused by a rising edge trigger on any of the PortJ pins which will be caused by the button being pressed. If the green light is on and the button is pressed, the ISR will turn the light off, grab the end value of the timer, and raise a flag signaling the end of the test. The response time will then be calculated as shown in section 3.2.1. To set up PortJ for interrupt on a rising edge, the below code was used:

```
//Setting up PortJ for button press interrupts
DDRJ &= 0b11111110;   //set PortJ0 for input
PPSJ |= 0b00000001;   //trigger on rising edge
PIEJ |= 0b00000001;   //enable PortJ0 for interrupt
PIFJ |= 0b00000001;   //clear interrupt flag
```

3.2.3 Serial Communication

Once data from three runs of the test have been collected, the microboard will then attempt to transmit the data using a serial connection via the RS-232 at 19200baud. The microboard will act as the sender in the serial communications, packaging all the test values that need to be transferred. The reaction time data is saved as various floating point values which will be converted into their ASCII character equivalents using the sprintf function found in the stdio.h library. To ensure data integrity, special marker bits have also been placed such that all data sent is packaged into a frame. ASCII character of '~' signals the start of a frame of data and the '!' marker indicates the end of the frame. The table shown below outlines the expected data frame to be sent:

Table 1 – Data Frame Format

Byte Offset	Value/Range (ASCII Encoded)	Purpose
0	'~'	Fixed Start Marker
1	0-9	Time 1 Ones digit
2	'.'	Decimal point
3	0-9	Time 1 Tenths digit
4	0-9	Time 1 Hundreds digit
5	0-9	Time 1 Thousandths digit
6	0-9	Time 2 Ones digit
7	'.'	Decimal point
8	0-9	Time 2 Tenths digit
9	0-9	Time 2 Hundreds digit
10	0-9	Time 2 Thousandths digit
11	0-9	Time 3 Ones digit
12	'.'	Decimal point
13	0-9	Time 3 Tenths digit
14	0-9	Time 3 Hundreds digit
15	0-9	Time 3 Thousandths digit
16	'!'	Fixed End Marker

The receiver will then parse the data received based on the format above to get the reaction time values. To save some computation time, it was decided that the average value of the reaction times did not need to packaged into the frame. Instead, the average could be recalculated on the receiving side of the serial communication. If additional error checking is required, the decimal point characters also serve as fixed markers. As

well, a checksum could be added to ensure that the data being sent is accurate as noise can affect the byte values.

4.0 User Interface Application

The user interface application is a C# program that runs on a Windows PC consisting of two main components. The first component of the program is to handle incoming data transmitted over a serial connection. The second component of the program involves establishing a connection to a database to store the reaction times as well as authenticate user login information.

4.1 Serial Communication

This application will need a means to retrieve data before sending it to the database. Therefore, the application will act as a receiver during serial communications. Microsoft has simplified the process of setting up a serial port for serial communications by providing the *SerialPort* class and user control in the .NET framework.

4.1.1 Serial Port Setup

Certain properties of the *SerialPort* will need to be set in order for data to be received correctly over the serial wire. These properties will need to match those found on the other device that the application will connect to which will be the 9S12 microboard for this project. The properties that need to be set are:

1. Baud Rate – Rate that data is transmitted, must be set to a specific standard value. Such as 19200 baud.
2. Data Bits – Format of the data, can be 7-bit ASCII characters, 8-bit extended ASCII, or binary data.
3. Parity Bit – For error checking.
4. Number of Stop Bits – Represents rest time between data characters.
5. Handshaking Type – For setting up and maintain sessions.

For purposes of this project, data will be transmitted at a baud rate of 19200 baud and formatted in 8-bit extended ASCII. As the distance between the microboard and the PC's serial port is not expected to be large and the electrical noise is assumed to be significant, no parity bit will be required. Figure 3 shown below represents the default settings of the serial port connection using the programs user interface. The COM port of the PC where the data streams into can be viewed by accessing Window's Device Manager.

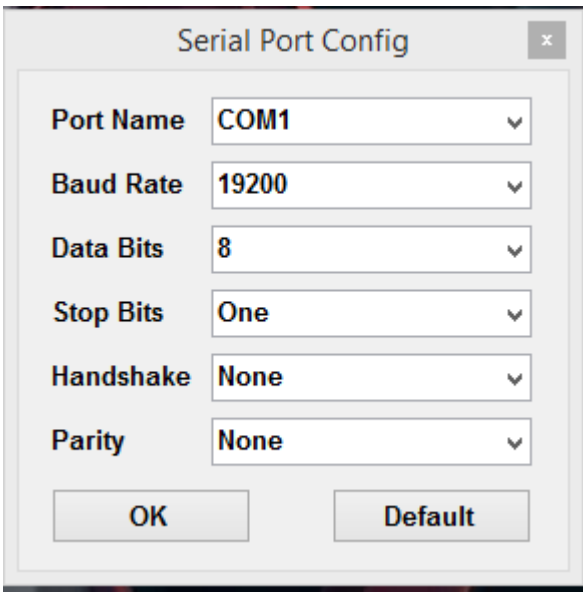


Figure 3 – Serial port configuration user interface

4.1.2 Data Retrieval and Transfer

All bytes received from the serial port are ASCII encoded as bytes. The bytes received can be read and kept into a byte buffer. Referencing the data frame layout in Table 1, the bytes in the buffer can be parsed until a frame is valid. Once the frame has been validated, the data bits containing the test values located on position 1 – 15 can be parsed. Afterwards, the average of the reaction time is then recalculated. Figure 4 shows the algorithm for parsing the data out of the buffer.

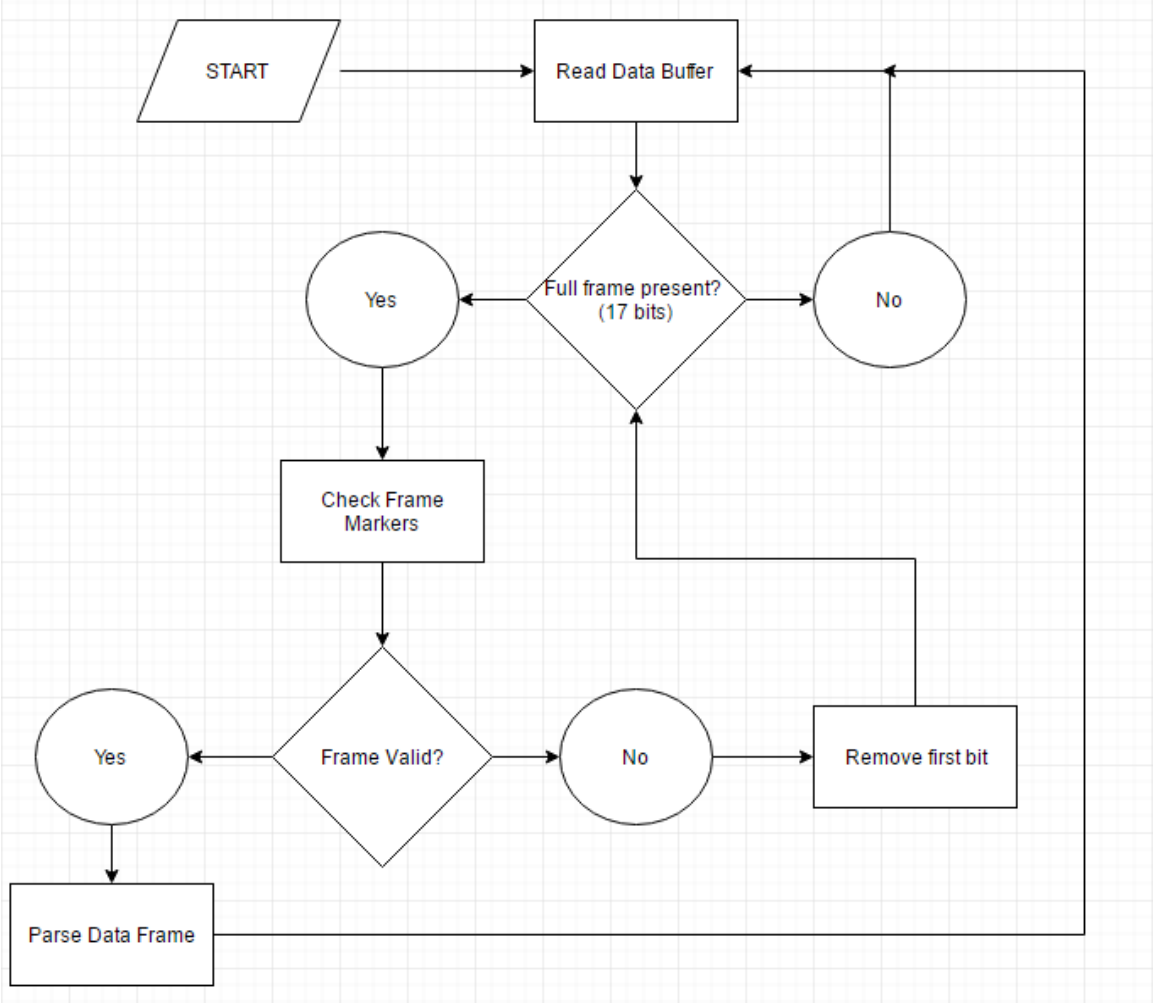


Figure 4 – Flowchart of data frame parsing

As shown in figure 4 above, a minimum amount of bits need to arrive before data can be parsed. Afterwards, the frame is checked to see if all the start and end markers are in the correct position. If the bits are out of order, the first bit is removed from the buffer until a valid frame falls into place in the first 17 bits.

4.2 User Login/Creation

The test data will need to be associated to the user who took the test. Therefore, the user interface app also supports user login/logout, and new user creation. The test data retrieved will be packaged together with the user credentials to be sent to a database. The database server chosen was Microsoft's SQL Server for easy integration with the C# application as SQL integration functions are provided in the .NET framework.

5.0 Database Implementation

As mentioned in section 4.2, the database server selected was Microsoft's SQL Server for easy integration with the C# application. The database will need to store all test data as well as user credentials and provide a way to associate the users to the test data.

5.1 Normalization

To layout the design of the database, the process of normalization is used. Normalization is a process that organizes the tables and columns of a database in a way that reduces data redundancy. According to Microsoft: "Redundant data wastes disk space and creates maintenance problems."

Normalization includes a few rules/procedures that need to be followed called "Forms". The rules for each normalization form listed will be described below using Microsoft's definitions of database normalization.

The first normal form, also known as 1NF, has 3 rules:

1. Eliminate repeating groups in individual tables
2. Create a separate table for each set of related data.
3. Identify each set of related data with a primary key.

The second form/2NF is as follows:

1. Create separate tables for a set of values that apply to multiple records.
2. Relate these tables with a foreign key.

The third form/3NF is as follows:

1. Eliminate fields that do not depend on the key.

Applying this normalization process to the data retrieved from the reaction tiem testing system is outlined below:

Raw Data

No table: User, Test#, T1Time,T2Time,T3Time, AvgTime, DateTaken

1NF

User Table: Username, Password

UserTests Table: Username, Test#, T1Time,T2Time,T3Time, AvgTime, DateTaken

2NF

User Table: Username, Password

UserTests Table: Username, Test#

Tests Table: Test#, T1Time,T2Time,T3Time, AvgTime, DateTaken

3NF

Skipped, as no changes were necessary.

After gathering the types of data that needed to be stored into the database for the reaction time testing system and applying the normalization procedure, the following database layout was created:

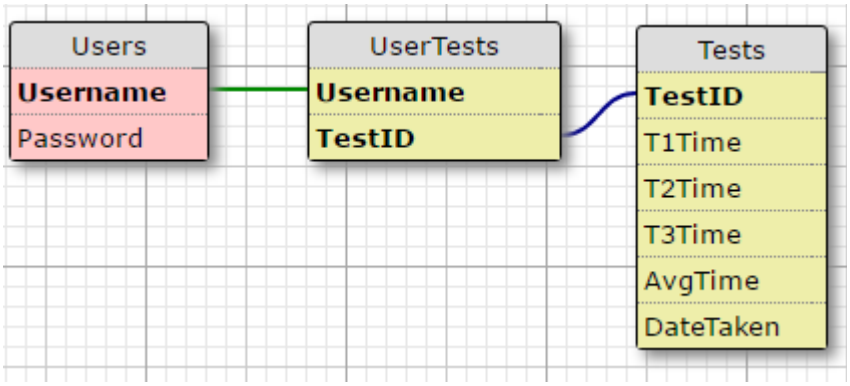


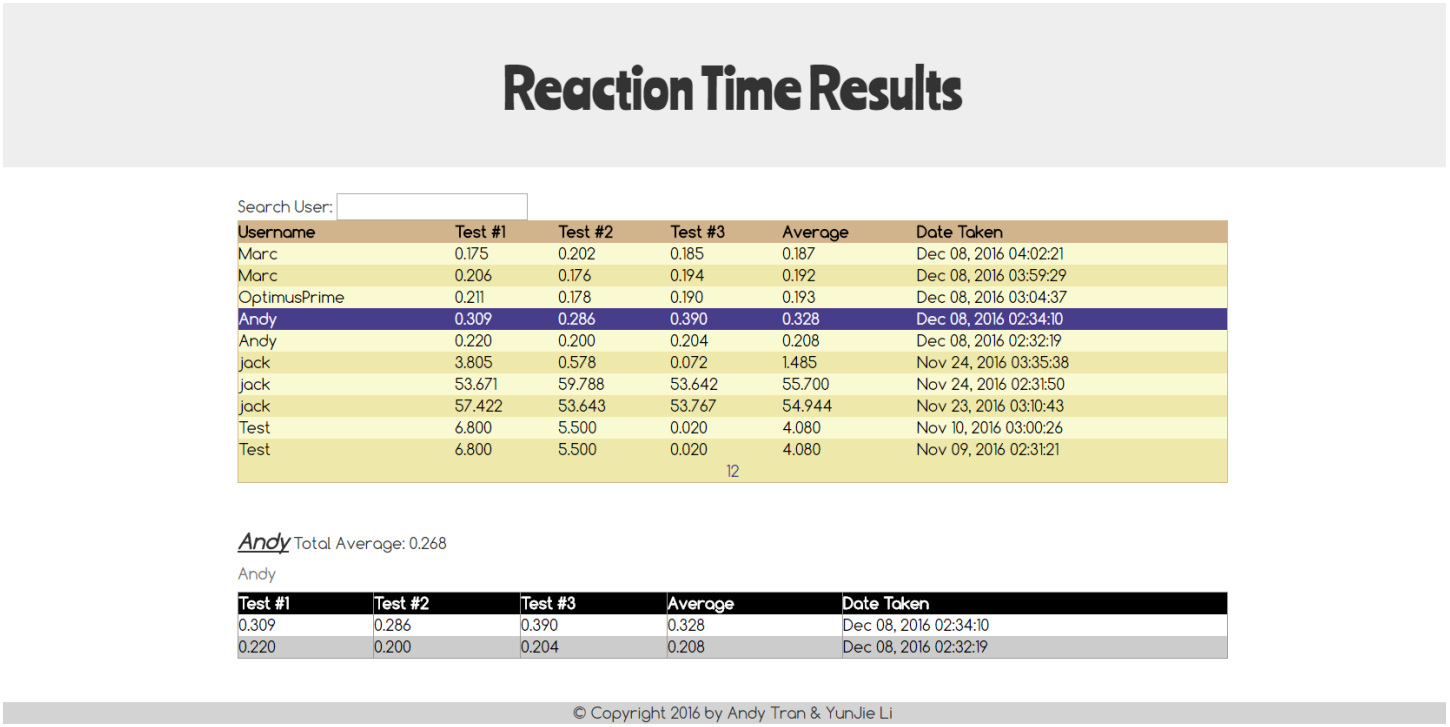
Figure 5 – Reaction Test Database Diagram

The "User" table will contain the information needed to validate user logins while the "Tests" table will store the tests data for each test instance. The third table, "UserTests" serves to link users to their respective test data. This database design allows the association of a single user to multiple tests, also known as a one-to-many relationship through the "UserTests" table.

6.0 Website Implementation

To start, an Empty C# ASP Website was used as the basis of the website makeup. The purpose of the website is to display test results to be viewed by anyone. In order to get the test results, the website will need a way to connect to the database. Since the website was built using ASP, the SqlDataSource control is used to form a connection to the

Microsoft SQL Server. A gridview to display the data has its data source binded to the SqlDataSource which was configured to display test results. A textbox was also added to enable username search/filtering capabilities. When an username is clicked, all test results associated to that user is displayed in a separate table. The figure below shows the layout of the website:



Reaction Time Results

Search User:

Username	Test #1	Test #2	Test #3	Average	Date Taken
Marc	0.175	0.202	0.185	0.187	Dec 08, 2016 04:02:21
Marc	0.206	0.176	0.194	0.192	Dec 08, 2016 03:59:29
OptimusPrime	0.211	0.178	0.190	0.193	Dec 08, 2016 03:04:37
Andy	0.309	0.286	0.390	0.328	Dec 08, 2016 02:34:10
Andy	0.220	0.200	0.204	0.208	Dec 08, 2016 02:32:19
jack	3.805	0.578	0.072	1.485	Nov 24, 2016 03:35:38
jack	53.671	59.788	53.642	55.700	Nov 24, 2016 02:31:50
jack	57.422	53.643	53.767	54.944	Nov 23, 2016 03:10:43
Test	6.800	5.500	0.020	4.080	Nov 10, 2016 03:00:26
Test	6.800	5.500	0.020	4.080	Nov 09, 2016 02:31:21

12

Andy Total Average: 0.268

Andy

Test #1	Test #2	Test #3	Average	Date Taken
0.309	0.286	0.390	0.328	Dec 08, 2016 02:34:10
0.220	0.200	0.204	0.208	Dec 08, 2016 02:32:19

Figure 6 – Website Page

7.0 Project Results

All of the major components of the project were able to be implemented. The project successfully measures a user's reaction time through the hardware implementation using the microboard. As well, the results were able to be transferred for processing through the use of a serial connection between the microboard and a PC running the user interface application. The application's role as the middle-ware between the microboard and the database provided a pathway for the data transfer. As well, the tests results that were stored on the database was able to be displayed to a website to be viewed anytime.

Currently the website is hosted on a Microsoft Azure server under the following URL: http://cntcapstone2016.azurewebsites.net/

Unfortunately, some minor features were unabled to be implemented in time. There were plans to incorporate an audio stimulus as part of the testing procedure by connecting an audio device to the microboard. As well, the serial connection could have been replaced by wireless model, perhaps using Bluetooth. Another feature would be to integrate some form of encryption for storing user credentials. Currently, user information is stored on the database in plain text which poses a security issue. All of these features were omitted in order to finish the project on time.

8.0 Conclusion

Reaction tests come in a variety of forms. They mainly serve to measure an individuals physical behavior by having them respond to some type of stimulus. For this project, a person's reaction time was tested, stored, and displayed. The testing procedure designed involved a green LED as the visual stimulus and a button press for the measurable physical response.

This report gave an overview of the reaction time test design. First it described the code & circuitry required to run the testing procedure. Also, it showed how to set up serial communications for transferring the test results and walked through the design and implementation of the database and website.

Additional features that could be explored to expand the project are mention in the project results.

Overall, the project fulfills its role as a testing system by gathering the test results and providing feedback. The results gathered are publicly available and can be analyzed for research purposes perhaps in the various fields of psychology as an individual's response time and other physical responses are tied to processing capability of the brain.

References

Arthur, R.J. (2006). Clocking the mind: Mental chronometry and individual differnces (1st ed.) [eBook] Retrieved from https://lesacreduprintemps19.files.wordpress.com/2012/11/clocking-the-mind-mental-chronometry-and-individual-differences.pdf

Backyard Brains. (n.d.) Experiment: How Fast Your Brain Reacts To Stimuli [Website] Retrieved from https://backyardbrains.com/experiments/reactiontime

Eli, B. (2009, August 12). Framing in serial communications [Blog post] Retrieved from http://eli.thegreenplace.net/2009/08/12/framing-in-serial-communications

Freescale. (2009, October). MC9S12XDP512 Data Sheet [Data Sheet] Retrieved from http://cache.freescale.com/files/microcontroller/doc/data_sheet/MC9S12XDP512RMV2.pdf

Glenn, P. (2016, November 12). Serial Comms in C# for Beginners [Online article] Retrieved from https://www.codeproject.com/articles/678025/serial-comms-in-csharp-for-beginners

Martyn, S. (n.d.) Aristotle's Psychology[Website] Retrieved from https://backyardbrains.com/experiments/reactiontime

Michael, I.P. (2005, February 15). Timing the Brain: Mental Chronometry as a Tool in Neuroscience[Online Article]. Retrieved from http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.0030051

Microsoft. (2013, July 12). Description of the database normalization basics [Online article]. Retrieved from https://support.microsoft.com/en-us/kb/283878

Microsoft. (n.d.). GridView Class [Website] Retrieved from https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview(v=vs.110).aspx

Microsoft. (n.d.). SqlDataSource Class [Website] Retrieved from https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource(v=vs.110).aspx

Sparkfun. (n.d.). Serial Terminal Basics [Website] Retrieved from https://learn.sparkfun.com/tutorials/terminal-basics/all