2018•2019

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Development and validation of data acquisition methods for rehabilitation purposes

PROMOTOR :

Prof. dr. ir. Ronald THOELEN

PROMOTOR :

prof.dr. Raf MEESEN

COPROMOTOR :

De heer Thijs VANDENRYT

Bart Souverijns

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding UHasselt en KU Leuven

2018•2019

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Development and validation of data acquisition methods for rehabilitation purposes

PROMOTOR :

Prof. dr. ir. Ronald THOELEN

PROMOTOR :

prof.dr. Raf MEESEN

COPROMOTOR :

De heer Thijs VANDENRYT

Bart Souverijns

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

UHASSELT

KU LEUVEN

Preface

This thesis represents the final step in a journey down a long, winding road. A journey riddled with detours and even some off-road escapades, but in hindsight, a beautiful journey that made me into the person I am today.

I want to take this opportunity to thank my supervisors prof. dr. ir. Ronald Thoelen and ing. Thijs Vandenryt, as well as drs. ing. Marijn Lemmens for their guidance and counseling. I further want to extend my gratitude to my external supervisor prof. dr. Raf Meesen and dr. Kim van Dun, for their input and for welcoming me so warmly into their team.

Special thanks goes to my parents, to whom I owe everything, who never stopped believing in me. Without your never-ending patience I couldn't have dreamed of achieving this.

Contents

List of Tables

List of Figures

Glossary

analog-to-digital converter (ADC) A device which converts a analog signal into a digital signal.

bimanual tracking task (BTT) A task used in experiments concerning bimanual coordination. Subjects are presented with a trace drawn on screen, which they are instructed to follow using two input devices in the form of rotatable disks. Each of these input devices is manipulated by the index finger of each hand, and controls the onscreen movement in either the x- or y-axis respectively. By altering the angle of the trace, the relative frequencies of the left and right hand are altered as well. Additionally, a moving target indicator can be displayed, which moves along the length of the trace.

BIOMED-REVAL (REVAL) A research group tied to UHasselt, dedicated to research into most aspects of rehabilitation. These include neurological, musculoskeletal, cardiorespiratory, pediatric as well as mental rehabilitation.

character-separated-values (CSV) A CSV file is a plaint text file that contains tabular data. Each line in the file represents a data record, and fields of a record are separated by character. Also commonly referred to as comma-seperated-values, as commas are often used as delimiters.

command-line-interface (CLI) An interface where the user interacts with a program or device by issuing commands solely in the form of successive lines of text. Some examples include MS-DOS/Apple DOS, Windows PowerShell and the Linux terminal.

diamagnet A material that repels magnetic fields. This in contrast to a magnet, which attracts magnetic fields..

Electroencephalography A method of monitoring brain activity. Electrodes are placed on the scalp and measure voltage fluctuations resulting from neuron activity.

graphical user interface (GUI) An interface that allows the user to interact with a device or computer program through the use of graphical components.

Hall-effect The Hall-effect is a phenomenon where a voltage is produced inside an electrical conductor when a magnetic field is applied perpendicular to the electric current in the conductor.

latency The delay between a stimulus and the desired outcome. E.g. the delay between sending and reception of data over a network, or between pressing a button on a keyboard and that keypress being registered by the operating system.

light dependent resistor (LDR) a variable resistor controlled by light. The resistance value is inversely proportional to the incident light intensity.

liquid crystal display (LCD) a flat-screen technology which employs liquid crystals in combination with a polarizing filter to modulate the light throughput of each pixel, thus modulating its brightness.

microprocessor A computer processor that incorporate functionality of a processor into a single integrated circuit.

Misfittech NEMA17 Smart Stepper (Smart Steppers) A combination of a control board featuring a rotary encoder with a NEMA17 stepper motor, designed by Misfit Technologies.

NEMA17 A stepper motor based on the NEMA standard with a faceplate size of 1.7 by 1.7 inches or 42x42mm. The NEMA standard is named after the National Electrical Manufacturers Association.

proportional-integral-derivative (PID) controller A PID controller is a feedback mechanism used to provide continuous control over a process or device. It achieves this by measuring a certain process variable, calculating the error $e(t)$ between this variable and the desire setpoint. From this error, a correction is calculated using proportional (P), integral (I) and derivative (D) terms:

1) P-term: amplifies the correction based on $e(t)$;
2) I-term: sums the $e(t)$ over time;
3) D-term: considers the rate of change of $e(t)$.

The goal is to eliminate $e(t)$ with minimal delay or overshoot, as well as eliminate offset. To achieve this goal, the P-, I- and D-term should be carefully tuned to the system.

pulse width modulation (PWM) Pulse width modulation is the modulation of an electrical signal by rapidly switching on and off. By adjusting the frequency as well as the on/off ratio, the average power delivered to a load can be regulated first.

real-time operating system (RTOS) A real-time operating system is an operating system that is especially suited for time critical systems. It features advanced scheduling of tasks and deterministic behaviour. This allows tasks to adhere to a much stricter timing than on general purpose operating systems.

rotary encoder A rotary encoder is a device that converts a mechanical rotational position or motion into a digital signal. An absolute encoder indicates the angle of the shaft with respect to a certain fixed angle, while an incremental encoder only reports position changes relative to their previous output.

serial peripheral interface (SPI) A synchronous serial communication interface, allowing for communication between two or more devices. An SPI network can contain only one master device, but multiple slave devices. An individual Chip Select (CS) line is needed for each slave device.

serial reaction time task (SRTT) A task in which participants are presented a series of visual cues, to which they are instructed to respond appropriately by pressing a button corresponding the presented cue. Repeating sequences can be alternated with random sequences, to provide insights into implicit learning, motor skill learning, work memory, etc.

standard deviation (SD) a measure to indicate how measurements are spread with relation to the mean. The lower the SD, the more measurements are concentrated around the mean value. A higher SD indicates measurements are more spread out.

unit circle A circle with radius equal to 1 unit, whose center coincides with the origin of the coordinate system in which it exists.

Abstract

To further their research into bimanual learning and motor skills, the neurological branch of REVAL, a research group dedicated to rehabilitation and tied to the university of Hasselt, is always in need of the right tools to gather valuable data. This thesis aims to provide some of these tools.

It describes the development of a bimanual tracking task application: its initial goals, the implementation and firmware adaptation of controllers to meet specifications, the conception of a functional graphical user interface which allows for straightforward entry as well as saving and loading of trial parameters and addition to the gathering and exporting of trial data.

The second part discusses the validation of reaction time measurements in an existing serial reaction time task, running in a general purpose operating system. The effects of system parameters on measurements performed by E-Prime, an established data acquisition platform and a self-developed Java AWT implementation are analyzed and compared. In conclusion, an embedded system implementation of the serial reaction time task is proposed and a proof of concept developed.

Abstract (Nederlands)

Teneinde hun onderzoek naar bimanuele leervaardigheden en coördinatie te bevorderen, is de neurologische tak van REVAL, een onderzoeksgroep gewijd aan revalidatie en gebonden aan de UHasselt, steeds op zoek naar de juiste hulpmiddelen om kostbare data te verzamelen. Deze masterthesis heeft als doel deze middelen te voorzien.

Deze masterthesis beschrijft de ontwikkeling van een bimanuele taak applicatie: de initiële doelen, de implementatie en nodige firmwareaanpassingen van controllers om te voldoen aan de vereiste specificaties, het ontwerp van een functionele grafische gebruikersinterface die toelaat om eenvoudig onderzoeksparameters in te geven, op te slaan en te laden alsook het verzamelen en exporteren van onderzoeksdata.

Het tweede deel bespreekt de validatie van reactietijdmetingen in een bestaande seriële reactietijd taak, die wordt uitgevoerd in een besturingssysteem voor algemeen gebruik. De effecten van verschillende systeemparameters op metingen uitgevoerd in E-Prime, een gevestigd data-acquisitieplatform, en een zelfontwikkelde implementatie in Java AWT worden geanalyseerd en met elkaar vergeleken. Ter conclusie wordt een implementatie van de seriële reactietijdtest op embedded systemen voorgesteld en een "proof of concept" ontwikkeld.

Chapter 1

Introduction

To further the field of rehabilitation, or any field of research for that matter, it is necessary that researchers dispose of the right tools for them to uncover the underlying mechanics of the problems they are trying to solve. The development of these tools often requires the collaboration of people with different areas of expertise. This applies to this thesis too, as it is born out of the collaboration between BIOMED-REVAL (REVAL), a research group tied to the university of Hasselt, dedicated to research concerning most aspects of rehabilitation, and the Faculty of Engineering Technology.

The goal of this thesis is to aid the neurological branch of REVAL in their research into bimanual motor and learning skills by developing new and validating existing data acquisition techniques. The first part of this thesis will describe the development of a bimanual tracking task, emphasizing key points such as the creation of a user-friendly interface combined with the generation of quality single trial as well as aggregated data.

The second part focuses on the validation of reaction times measurements performed in an existing implementation of the bimanual serial reaction time task. This is done by analyzing the influences of several factors on measurements performed in E-Prime, an established data acquisition suite. Furthermore, an embedded systems implementation of the serial reaction time task is proposed to further improve reaction time measurement accuracy as well as precision, and a proof of concept is developed and tested.

Chapter 2

Development of a Bimanual Tracking Task Application

2.1 Context

2.1.1 Bimanual Tracking Task

The application described in this chapter will enable REVAL researchers to perform experiments involving the bimanual tracking task (BTT), a task used in bimanual coordination research. To perform this task, subjects are instructed to follow a trace drawn on the screen. Input is provided using two controllers to which rotating disks are attached. Each of these controllers is manipulated by the index finger of either hand and controls movement on the x- or y-axis respectively. The angle of the presented trace can be altered, depending on a given frequency ratio between both controllers as well as the given quadrant, as is illustrated in Figure 2.1 below.

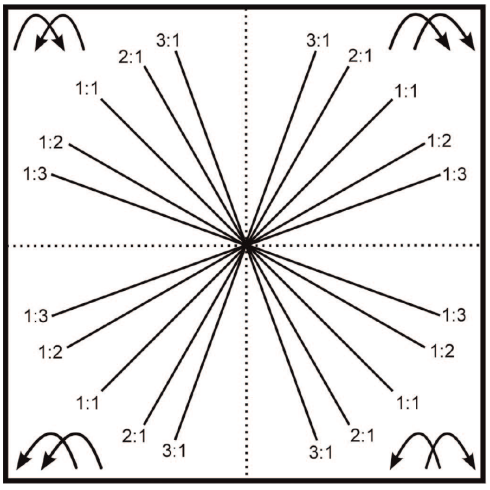


Figure 2.1: Overview of trace angles corresponding to given frequency ratio and quadrant.

The double arrows in the corners indicate the rotational direction of the corresponding hand for each quadrant. The number left of the ':' indicates the relative frequency of the left hand, the number to the right corresponds to the relative frequency of the right hand, i.e. a frequency ratio of 3:1 indicates the subject should turn the left controller three times faster than the right controller. In addition to this trace, a target indicator can be displayed. This target moves along the trace at a given constant speed, from the start of the trace outward. Here the subject should not only follow the trace as closely as possible,

but also match the speed of the target. The trial ends when the time limit is exceeded or either the target or subject reach the edge circle of which the trace forms the radius.

2.1.2 Objectives

While Reval researchers already had a web platform with an implementation of the BTT at their disposal [1], this platform had developed stability issues due to the many additions and thus proven to be unreliable and too maintenance heavy for novices in information technology. Therefore, it was decided to create a new standalone application to provide Reval researchers with a stable and reliable tool for their research. The main objectives for this application were as follows:

1) A polling rate of 100Hz;

2) An intuitive graphical user interface (GUI);

3) The ability for researchers to save task parameters for (sequences of) tests for easy repeatability;

4) The inclusion of a 'torque mode,' in which the controller counteracts the subject's input with a predetermined level of torque.

2.2 Misfit Technologies NEMA17 Smart Stepper

To allow for the rotating disks to be used as input for the BTT, the disks must be mounted to rotary encoders. Furthermore, to facilitate the torque mode mentioned in 2.1.2, each controller should contain an actuator to provide the opposing torque.

For this purpose, the Misfittech NEMA17 Smart Stepper (Smart Stepper) was selected. The Smart Stepper is an open-source project that aims to convert standard NEMA17 stepper motors into closed-loop servos. It achieves this by combining the stepper motors with a control board, on which a proportional-integral-derivate (PID) controller is implemented. Firmware can conveniently be uploaded to the board via the Arduino IDE.

The Smart Stepper connects to the pc via the micro-USB port and enumerates as a USB serial device. It provides a command-line-interface (CLI) for the user to issue commands or receive data information from the device. Some of the most important and often used commands include:

1) help: returns the list of supported commands along with a brief description;

2) readpos: requests the current angle of the motor axis. The response is a 16-bit float;

3) move: rotates the axis to the requested angle. Note that this angle is relative to reference zero angle of the device, not to the current angle. E.g. the command "move 1080" will rotate the axis 3 full rotations clockwise if the current angle is 0°, but 1 rotation anti-clockwise if the current angle is 1440°. An optional second paramater sets the speed of the rotation in rotations/minute, e.g. "move 1080 30";

4) setzero: sets the current axis angle as the reference zero angle of the device;

5) maxcurrent: sets the maximum current that can be supplied to the motor in mA. This provides a means for control of the torque deliverd by the motor;

6) calibrate: calibrates the device;

7) spid: with no arguments, prints the the PID parameters. When combined with arguments (spid K_p K_i K_d), this command sets the PID parameters accordingly.

2.2.1 Components

Atmel SAM D21G

The PID-loop runs on the Atmel SAM D21G, a 32-bit microprocessor based on the ARM Cortex-M0+ architecture. It features a native 48Mhz clockspeed, 256KB of programmable flash memory as well as 32KB of SRAM. Of its 48 pins, 32 are general purpose input/output (GPIO) pins. Of these, 16 support external interrupts. Other features include 14 analog-to-digital (ADC) channels and 6 serial communication interface instances, including uART, SPI and I^2C [2].

AS5047D High Speed Position Sensor

To accurately measure the position of the motor axis, the AMS AS5047D rotary encoder is used. A block diagram is shown in Fig. 2.2 below.

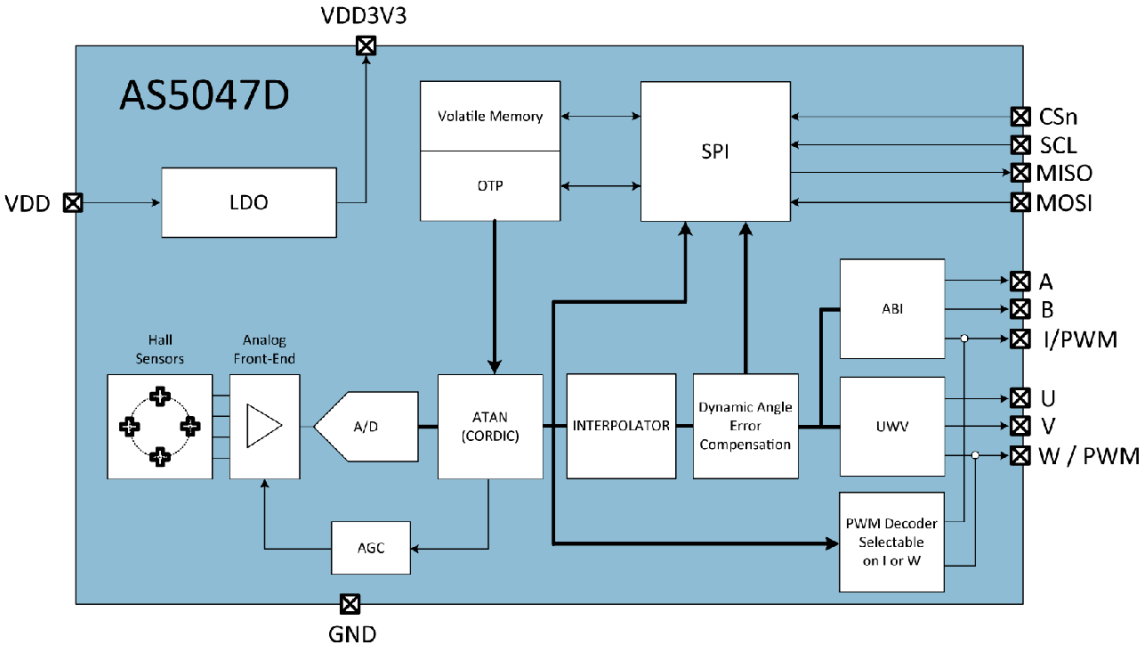


Figure 2.2: Block Diagram of the AS5074D rotary encoder [3]

The AS5047D is an absolute angle rotary position sensor. It is positioned on the back of the stepper motor, perpendicular to the stepper axis to which a strong diamagnet is attached. As the rotation of the axis changes, an array of Hall-effect sensors senses the change in magnetic field. The signal is then amplified, filtered and converted before being processed to compute the angle of the axis. The output is provided via the Serial Peripheral Interface (SPI)-protocol in 14-bit resolution.

A4954 Motor Driver

The Allegro Systems A4954 is a dual full-bridge PWM motor driver. As the name implies, a dual bridge driver is designed to be able to drive two DC motors simultaneously. As NEMA17 stepper motors are driven by two sets of coils, connecting each of these bridged outputs to one of the motor coils enables microstepping. By seperately manipulating the current into each coil, opposing forces are generated in these coils, allowing for the axis to hold intermediate positions between steps. This provides greater control, smoothness and precision from the motors [4].

2.3 Application Framework

2.3.1 JavaFX

JavaFX was selected as the software platform on which to base the application. As it is the most recent Java GUI library, it provides a consistent and pleasing, yet professional aesthetic. Furthermore, as the application is compiled into bytecode rather than executable code, the application will support a wide range of platforms such Windows, Mac, Linux, etc., out of the box once compiled. Thanks to Java Applet and WebStart, it even provides the possibility to add the application to the existing Reval web platform at a later date.

Model-View-Controller

The application is built upon the Model-View-Controller paradigm. In this design pattern, the application is split into three distinct yet interconnected parts. A diagram is shown in Figure 2.3.

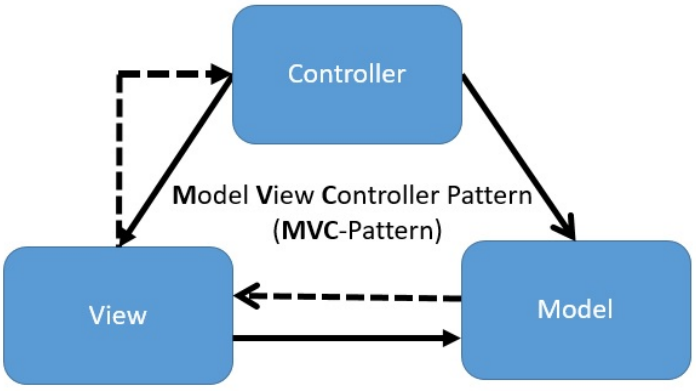


Figure 2.3: Diagram of the Model-View-Controller Design Pattern [5]

The Controller is the top-level class. It handles inputs from the user, such as key presses, button clicks, etc., and updates the Model accordingly. These events are passed through from the View, which in its turn presents the model to the user. The Model is not aware of Controller or View, it only handles application logic. This division of the application in three parts instills a clear structure in the application and ensures that changes in the model do not necessitate changes in the presentation or vice versa.

RXTX

In order for the application to communicate with the Smart Steppers, a serial connection should be established. RXTX is a library which provides serial and parallel connectivity for Java across platforms.

2.3.2 Class Overview

This section will provide an overview of the class structure of the application model. It contains concise descriptions of the most important classes.

BimanualTask

BimanualTask is the top-level class of the model. It connects all underlying classes and provides all functionality of the application.

TaskParams

TaskParams objects contain the parameters for each trial, such as the frequency ratios and quadrant, the torque ratios, whether or not a moving target should be shown in addition to the trace, etc.

Path

The Path class has two instances: userPath and targetPath. These instances contain the collection of coordinates which form the trajectories followed by the subject and target respectively as well as methods to calculate overview data.

SerialController

The SerialController class handles the serial communication with the Smart Steppers and translates method calls from the model to compatible commands and vice versa.

EEGConnection

Similarly to the SerialController class, the EEGConnection class manages the serial connection to the EEG monitoring device.

2.4 Firmware Adaptations

In order for all objectives in section 2.1.2 to be attained, some adaptations had to be made to the Smart Stepper firmware.

CLI Verbosity

While the CLI provides a clear, human-readable format for interaction with the Smart Steppers, this interface introduces latency in application-to-device communication due to its verbosity. The CLI prints a message prompt, ":¿", at the start of each line. Furthermore, it repeats commands sent by the user and replies with long strings. The following example illustrates the flow of a request:

```
:>
:>readpos           % user request
:>readpos           % device repeats request
:>encoder 90.08     % device response
```

In initial testing, these extra lines generated sufficient latency for the 100Hz polling rate to become unattainable. Therefore, the CLI was adapted to respond more concisely and efficiently. To achieve this, the message prompt was removed, the device no longer repeats user requests and responses are abbreviated. The result is as follows:

```
r           % user request
enc 90.09   % device response
```

A new 'r'-command was also created, which replies in a more concise manner. The 'enc' portion of the reply was retained to provide a means for the application to verify response validity.

Torque Feedback

To enable the desired torque mode, an additional function was implemented into the firmware of the Smart Stepper. The purpose of torque mode is for the device to deliver a certain counter torque to the user's efforts to rotate the controller. In a way, this is exactly what the PID loop does: it constantly attempts to correct the motor position to the set point. However, this set point is fixed to the set angle issued by the "move"-command. To overcome this, a new 't'-command, in which t stands for torque, was created. When this command is processed, it returns the position of the angle of the axis similar to the 'r'-command, yet simultaneously adjusts the set angle to the current angle of the axis. As a result, the PID loop attempts to hold the last known position of the axis. As this angle is updated constantly at a frequency of 100Hz, the user perceives this as a smooth counter torque to his or her input.

2.5 EEG Triggers

To monitor brain activity during trials, REVAL makes use of the Biosemi ActiveTwo EEG monitoring system. To allow researchers to synchronize trial data with EEG data, the ability to send triggers to the ActiveTwo USB receiver was implemented. While the aforementioned USB receiver enables monitoring of EEG data on a computer via USB, the receiver only provides the option of sending or receiving trigger signals via parallel port. The Biosemi USB trigger interface cable enables the serial signal to be converted to a parallel signal, with less than 200μS of delay [6]. However, this does limit the interface to 8 outputs rather than the 16 inputs and 16 outputs via the native parallel interface, although this still allows for up to $2^8 - 1 = 255$ trigger combinations. Currently, no more than three trigger signals are utilized, as show in Table 2.1.

Table 2.1: EEG trigger signals and interpretation

Trigger interpretation	Binary signal
Start of countdown	0000 0001
Start of trial	0000 0010
End of trial	0000 0100

2.6 Graphical User Interface

An essential part of any application is its user interface. As this is the main point of interaction for the user with the application, it greatly impacts the user experience. It should provide a pleasing aesthetic, yet still maintain a professional look. Moreover, it should be intuitive and thus enhance the functionality of the application. Buttons, settings, input fields, etc., should be placed and grouped in such a way that the user intuitively navigates them. To become an efficient tool for bimanual coordination research, the application should combine robust functionality with a simple yet efficient and powerful interface.

2.6.1 OperatorView and SubjectView

During the design of the application, it immediately became apparent that this application would be approached from two fundamentally different points of view: that of the researcher, and that of the subject. Because of this, a distinction was made between the presentations for researcher and subject, as illustrated in Figure 2.4.

(a) OperatorView

(b) SubjectView

Figure 2.4: The OperatorView and SubjectView side by side

When the program is started, two windows are instantiated: one contains the OperatorView, the other the SubjectView. The SubjectView is kept intentionally simple. The goal here is to provide a functional presentation of the task and to prevent any possible distractions. This window contains the trace to follow (the black line originating from the center), the subject indicator (the black circle) and its traveled trajectory (the blue path), and optionally the target indicator (a red circle).

The OperatorView, intended for researchers, is slightly more elaborate. Like the SubjectView, it too displays the playfield of the task. This allows the researcher to observe the progress and performance of the subject from a separate screen, without having to look over his or her shoulder. In addition to this, the researcher window encompasses an array of controls. The controls in the bottom left directly concern the task itself and provide following functionality:

1) Start: self-explanatory;

2) Parameters: this opens a new window where trial parameters can be set. This is explained in greater depth in section 2.6.3;

3) +/-: The plus and minus buttons respectively increase and decrease the size of the view presented to the subject, and thus allows for the task to be presented uniformly across different screen sizes and resolutions.

It should be noted that the option of adjusting the presentation size of the subject view automatically to the actual screen size and resolution of the used monitor was considered. In theory, if the screen resolution and pixel density are known, the screen size and resolution for the subject view could be easily calculated. However, some operating systems and monitors fail to correctly report monitor information such as PPI (pixel-per-inch, the unit for pixel density). This approach was thus not deemed sufficiently reliable to implement.

The frame in the upper left, however, contains options regarding the Smart Steppers. As these settings could potentially cause issues if changed inadvertently, a toggle switch was provided to prevent accidental presses. These will be further discussed in section 2.6.2.

2.6.2 Controller Settings

As previously mentioned, the controller settings are located in a separate frame in the upper left-hand corner. As previously indicated, these settings are greyed out unless a toggle switch labeled 'Advanced' is switched on. This is to prevent accidental changes. This section of the GUI contains three functions, which are explained below.

Realign

The Smart Steppers do not store a fixed reference angle in non-volatile memory. This means that each time the Smart Steppers are booted, the axis angle at that time is set as the new reference angle. However, for continuity, it is important that all trials start from the same position. The 'realign'-button was implemented to overcome this issue. When this button is pressed, the user is presented with the prompt show in Figure 2.5.

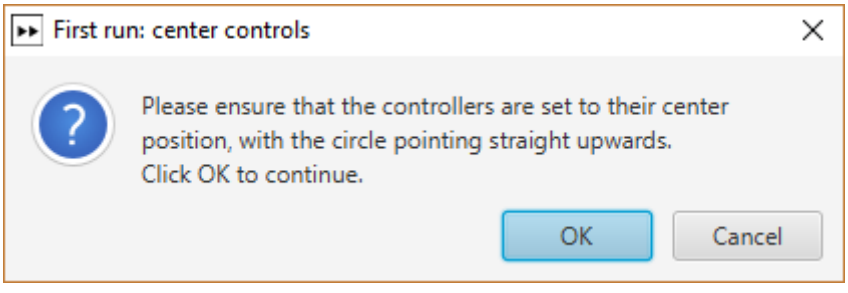


Figure 2.5: The controller realignment prompt

Any current supplied to the motors is temporarily subdued to allow them to rotate freely. The prompt instructs the user to rotate the dials of the controllers so that the recess in the dials points straight upward. If the user responds 'OK', the current axis angle is set as reference and the preceding hold current to the controllers is restored. Otherwise, the previous reference angle is maintained.

Calibrate

As calibration may drift slightly over time, an option to recalibrate the controllers was implemented.

COM settings

In most operating systems, COM ports are assigned dynamically. This means that when the Smart Stepper is unplugged from the computer, it might be assigned a different COM port when plugged into another USB port. To ensure that the controllers are correctly recognized, and to distinguish between the left and right controller, the COM settings window was introduced (see Figure 2.6).

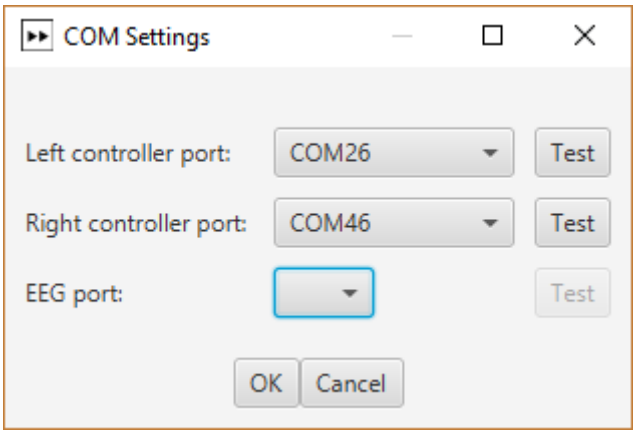


Figure 2.6: The COM settings window

To correctly distinguish Smart Steppers from other USB peripherals, the application briefly connects to each detected COM device, submits a command and checks for the appropriate response. This guarantees that the drop down menus for the left and right controller only contain compatible devices. To differentiate between the left and right controller, a test-button is provided, which will cause the selected controller to rotate 180°.

Optionally, an EEG port can be selected in order to send triggers to the EEG monitor. As the EEG monitor acts as a mute device, i.e. it can only receive triggers and not reply, the list of devices in the drop down menu can not be limited to compatible devices in the same way as the Smart Stepper drop down menus. Instead, the test button sends three triggers in quick succession, to allow researchers to identify the correct port by monitoring the triggers in the acquisition software.

2.6.3 Parameter Entry

The parameters window enables the researcher to input the desired trial parameters. It was designed to provide a clear overview of the set parameter at a single glance, and to facilitate saving and loading sequences of trials. A screen shot is presented in Figure 2.7.

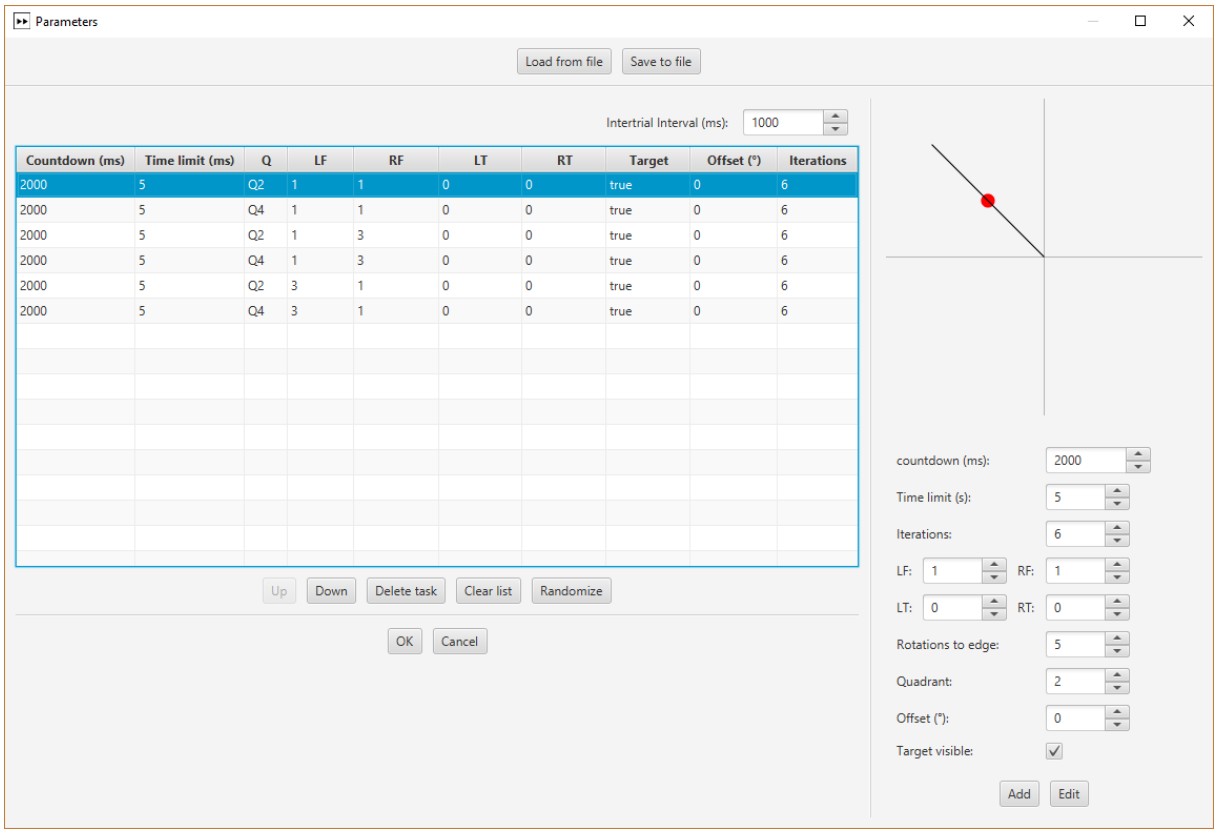


Figure 2.7: Screen shot of the parameters window, with set of parameters loaded

The main portion of this window is occupied by a table, representing the list of currently entered trials and their parameters. The order in which this sequence of trials will be run is in the order in which they are displayed, from top to bottom. To manipulate this list, the researcher can select a line by simply clicking on it, and use the buttons below to move the entry up or down in the list or delete it altogether. Additionally, the list can be cleared entirely or the order randomized. Above the table is a single input field for the inter trial interval, i.e. the interval between the end of one trial and the start of the next. This field is isolated from the other controls, as it pertains to the test sequence as a whole. This in contrast to the other controls, which affect only the parameters of a single entry in the table.

The right hand portion of the window allows the researcher to add or edit entries in the table. The input fields consist mostly of spinner controls, which allow the value to be in- or decremented by pressing the up or down arrows. The following control options are provided:

1) Countdown (ms): the amount of milliseconds during which the trace should be displayed, before the actual trial is started;

2) Time limit (ms): the maximum duration of the trial, in milliseconds;

3) Iterations: the amount of times the trial in question should be repeated;

4) LF and RF: the left and right components of the trace frequency ratio respectively;

5) LT and RT: The counter torque to be generated by the left and right controller respectively. This setting acts as a multiplier for the current supplied to the stepper motors;

6) Rotations to edge: this setting defines how many rotations of the controller translate to an on-screen distance traveled to the edge of the circle of which the radius equals the trace length;

7) Quadrant: the quadrant in which the trace should be drawn;

8) Offset (°): the offset angle with which to visually rotate the canvas anticlockwise, compounding the difficulty for the subject. Adjustable in 15° increments. Effect illustrated in Figure 2.8 below;

9) Target visible: this toggles whether or not a moving target for the subject to follow should be displayed.

To facilitate visualization of the effects of parameters on the trial, a trial preview was implemented in the top right hand corner (see Figure 2.8).

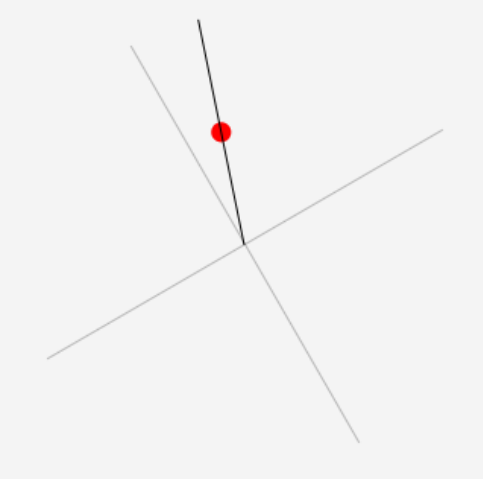


Figure 2.8: Trial preview with a 120° visual offset applied (1:3 ratio, Q4).

This preview displays a scaled-down image of the trace as it would be presented during the actual trial, given the currently set parameters. Note that while a cross is visible in the trial preview, this is not displayed to the subject during the task. It is merely there to assist the researcher in ascertaining the rotation angle of the coordinate system.

Finally, the parameter window allows researchers to conveniently save new or load predefined parameter files. The trial parameters are condensed into files with a '.rvl' extension. The contents of such a file are shown in Figure 2.9.

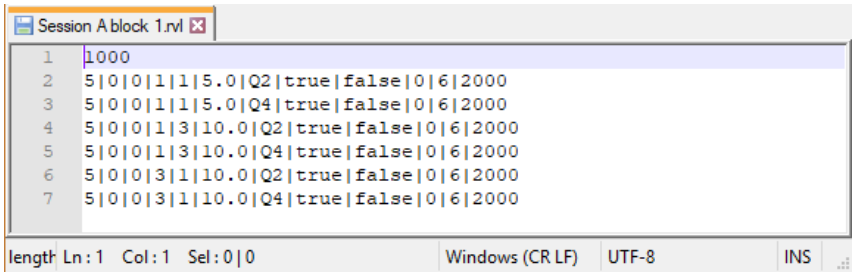


Figure 2.9: Contents of a .rvl-file, displayed in NotePad++

The first line contains the inter trial interval, while each subsequent line represents the defined trials in order of execution. While with some effort these files are human-readable and can be opened and edited in most text editors, it is not advised to do so. This because great effort has been made to validate user input and manually adjusting these values might lead to corrupt files. Section 2.6.4 discusses input validation in greater depth. Files which contain unsupported values will be refused.

2.6.4 User Input Validation

To ensure a smooth user experience and security, great effort was made in performing user input validation. This not only entails analyzing parameters entered by the user for validity, but - perhaps more importantly - also to actively prevent the user from entering invalid parameters or commands. This is achieved in several ways, such as:

1) greying out buttons, input fields and other controls these are not applicable to the current state of the application. E.g. disabling 'move up' when the top row in the parameter table is selected, 'clear list' and 'randomize' when the table is empty, 'start' when no parameters are entered or no controllers are connected, etc;

2) utilizing spinner controls rather than text fields for parameter input, as this ensures the user can only select preapproved values;

3) where text fields are used, to only allow safe characters to be typed: aA-zZ, 0-9, '-' and '_'. If the application were to be ported to the web platform in the future, this limitation on special characters will also help impede SQL injection.

4) warn the user in case an illegal frequency ratio is selected, no COM ports assigned, the controllers were not realigned, etc., and inform him or her how to resolve the issue.

2.7 Data Acquisition

Data is collected at a frequency of 100Hz. Every 10ms, the axis angle of the Smart Steppers is read, as well as the current coordinates of the target, if applicable. The angle is then converted into coordinates. As at the onset of each trial the reference angle is reset, the coordinates can directly be calculated from the current axis angle as follows:

$$x = \frac{\theta_x}{360^\circ \times n_{rot}}$$

With x the x-coordinate, θ_x the measured angle and n_{rot} the value for "rotations per unit" entered in the parameters window. In other words, n_{rot} defines how many rotations of the controller equate to a distance of 1 unit traveled on the corresponding axis. The y-coordinate is calculated analogously. This also implies that measurements are mapped to a unit circle, normalizing all coordinates in the process, irrespective of on-screen resolution. Each trial ends when the preset time limit is reached, or the subject reaches the edge of the unit circle.

2.8 Output

Data is exported to character-separated-values (CSV) files with the pipe symbol ('|') as field separator. This symbol was chosen over the comma as, depending on the locale of the user, the comma might be set as decimal delimiter. This would cause the structure of the tables to be compromised when parsed by spreadsheet editors such as Microsoft Excel. For each test sequence, the application outputs generates three types of files: one trial data file for each trial in the sequence, a single parameters file and a single overview file.

2.8.1 Parameters File

The parameters file contains the parameter data as entered by the researcher in the parameters window. These are described in section 2.6.3.

2.8.2 Trial Data

At the conclusion of each trial, a trial data file is created. This file contains all data points acquired during the trials. After feedback from Reval researchers, each row also contains some global information about the trials as well as some parameters, to simplify data processing. These include the subject and trial number, time and date at which the trial was run, the trace quadrant and frequencies, the torque settings as well as the visual offset angle and whether or not a target was displayed. The actual measurements present the following data:

Time (ms)

A timestamp displaying the amount of milliseconds passed since the start of the trial.

SubjectX and SubjectY

The x and y coordinates of the subject indicator at the corresponding time stamp.

Quadrant

This returns in which quadrant the subject indicator is currently located. Q1 signifies the top right quadrant, Q2 the top left, Q3 bottom left and Q4 bottom right. As technically the origin and axes are not part of the quadrants, some additional identifiers were added: origin, xpos, xneg, ypos and yneg. The '-pos' and '-neg' suffixes indicate whether the subject is situated in either the positive or negative part of the respective axis.

Trace-SubjectX and Trace-SubjectY

These fields represent the x and y components of the vector pointing from the subject coordinate to the nearest point on the trace. The point on the trace nearest the subject coordinate is at all times defined as the intersection of the line perpendicular to the trace which passes through the subject coordinate. This point can be defined as point $P(x_P, y_P)$, the subject coordinate as point $S(x_S, y_S)$. As the trace passes through the origin, the trace can be defined as $y = ax$ with slope $a = tan(\frac{f_{left}}{f_{right}})$. The slope of the line perpendicular to the trace is then defined as $b = \frac{-1}{a}$. Assuming d is the intersection of this perpendicular line with the x-axis, $d = y_S - bx_S$. In conclusion, point P is defined as:

$$ax = bx + d$$
$$\Rightarrow x_P = \frac{d}{a-b},\ y_P = a \times \frac{d}{a-b}$$

In conclusion, the outputs generated in these fields are the x and y components of the vector pointing from the subject coordinate to the nearest point on the trace: Trace-SubjectX = $x_P - x_S$ and Trace-SubjectY = $y_P - y_S$.

TargetX and TargetY

These are optional fields which are only generated if the target was displayed. They contain the x and y coordinates respectively of the target indicator.

Target-SubjectX and Target-SubjectY

These are optional fields which are only generated if the target was displayed. Analogously to Trace-SubjectX and Trace-SubjectY these fields report the x and y components of the vector pointing from the subject coordinates to the target coordinates. Assuming the target coordinate as $T(x_T, y_T)$: Target-SubjectX = $x_T - x_S$ and Target-SubjectY = $y_T - y_S$.

2.8.3 Overview File

The overview file generates one row of data for each trial run in the sequence. In addition to the same global trial information as the trial data files, it contains additional statistical values calculated from the full data set of each trial.

Total time (ms)

The total run-time of the trial in milliseconds

Length of line

This field returns the total distance traveled by the subject, or the cumulative sum of the absolute distance between each subject coordinate and its previous value, with n the total amount of measurements:

$$Length\ of\ line = \sqrt{x_1^2 + y_1^2} + \sum_{i=2}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

Total trace deviation

This field returns the total deviation from the trace by the subject, or the cumulative sum of the absolute distance between the subject and the nearest point on the trace:

$$Total\ trace\ deviation = \sum_{i=1}^{n} \sqrt{(x_{S,i} - x_{P,i})^2 + (y_{S,i} - y_{P,i})^2}$$

With $S(x_S, y_S)$ the subject coordinate and $P(x_P, y_P)$ the nearest point on the trace. The equation for point P is described in section 2.8.2.

Average trace deviation

This field returns the average deviation of the subject from the trace, or the cumulative sum of the absolute distance between the subject and the nearest point on the trace divided by the total amount of measurements (n):

$$Average\ trace\ deviation = \frac{\sum_{i=1}^{n} \sqrt{(x_{S,i} - x_{P,i})^2 + (y_{S,i} - y_{P,i})^2}}{n}$$

Final trace deviation

The final trace deviation is the absolute distance from the subject to the nearest point on the trace, in the last record of the trial, or:

$$Final\ trace\ deviation = \sqrt{(x_{S,n} - x_{P,n})^2 + (y_{S,n} - y_{P,n})^2}$$

Total target deviation

This optional field returns the total deviation from the target by the subject, or the cumulative sum of the absolute distance between the subject and the target:

$$Total\ target\ deviation = \sum_{i=1}^{n} \sqrt{(x_{S,i} - x_{T,i})^2 + (y_{S,i} - y_{T,i})^2}$$

With $S(x_S, y_S)$ the subject coordinate and $T(x_T, y_T)$ the target coordinate.

Average target deviation

This optional field returns the average deviation of the subject from the target, or the cumulative sum of the absolute distance between the subject and the target divided by the total amount of measurements (n):

$$Average\ trace\ deviation = \frac{\sum_{i=1}^{n} \sqrt{(x_{S,i} - x_{T,i})^2 + (y_{S,i} - y_{T,i})^2}}{n}$$

Final trace deviation

This optional field returns the absolute distance from the subject to the nearest point on the trace, in the last record of the trial, or:

$$Final\ target\ deviation = \sqrt{(x_{S,n} - x_{T,n})^2 + (y_{S,n} - y_{T,n})^2}$$

2.9 Smart Stepper Issues

During the pilot program, during which the application was employed by actual experiment participants, some issues with the Smart Steppers became apparent:

A first issue was a difference in smoothness of the encoders when rotated left versus right: When the axis of the Smart Stepper is rotated clockwise, the counter torque is applied very smoothly. However, when rotated anti-clockwise, this counter torque feels much less controlled as the torque rapidly increases and decreases, producing a twitching motion. In addition to this, a difference in speed when rotating anti-clockwise versus clockwise is discernible.

A second issue occurred frequently while the controllers returned to their center position after trials. The task is programmed to return the controllers to their starting position before the start of each trial, in a slow and controlled manner. However, during some experiments, the Smart Steppers would exhibit anomalous behaviour. At certain times, the controller would jerk rapidly towards the starting position, overshoot slightly and spring back, to then suddenly idle for 0.5s to 1s, before resuming and returning to the center position. At other times, the controller would overshoot and spring back, to then start rotating uncontrollably at full speed. Once the motor was stopped manually, it would then spin in the opposite direction at full speed until it reached its set position.

After some time, it became apparent that the issues discussed in the previous paragraph, occurred more often with older participants than with younger participants. By observing their behaviour, a trigger for these anomalies was discovered. When the stepper motors are powered, they will always try to reach their set position, counteracting any opposition with torque corresponding to the maximum allowed current. Additionally, when the move command is issued with a second parameter indicating the rotational velocity, this retarded movement is achieved by a timer which continuously updates the set position in the background, according to these movements. The perceived slow movement is actually a rotation at full speed, but in small increments with delays in between. Older participants were triggering these issues because they had a tendency to counteract the motors as they were trying to return to their starting position. This caused the axis position to lag behind the slowly incrementing set position; when the dial was released, the Smart Stepper would attempt to catch up, overshoot its target and then, sometimes, exhibit the issues explained in the above paragraph. This behaviour could also be reliably reproduced by simply rotating the disk 200-220° anti-clockwise and releasing it.

While initially, this seemed like an unstable PID loop, tuning the PID parameters did not remedy these issues. To confirm that these problems were not caused by the BTT application or the adapted firmware, the original firmware (v.0.39) was reinstalled and the stepper motors were tested outside of the application, using the CLI. The issues persisted and could be reproduced reliably on all four specimens, which leads to believe their cause to most likely be firmware-related. While the developer of the Smart Stepper firmware was informed of these anomalies, he has unfortunately not yet been able to relay a possible cause or solution at time of submission.

While these issues generally do not interfere with the execution of the task by the partipant nor with the accuracy of measured data, in some rare cases the erratic behavour of the Smart Steppers can lead slight anxiety or annoyance with participants. As such, the BTT can continue to be used in its current form, pending a solution to be found.

Chapter 3

Validation of Serial Reaction Time Measurements

3.1 Context

3.1.1 Serial Reaction Time Task

This chapter discusses the validation of reaction time measurements registered in a serial reaction time task (SRTT). During an SRTT, the subject is presented a series of stimuli in the form of visual cues on a computer screen. Each trial starts when such a cue appears on screen. The subject then responds appropriately by pressing the keyboard button corresponding to the stimulus shown, ending the trial. After a short delay, the next trial starts.

During each trial, the response time of the subject is measured. Unbeknownst to the subject, repeating sequences of trials are alternated with random trials. This distinction allows researches to compare subjects' reaction times to sequential versus random sequences. Because of this, the SRTT is a valuable tool in rehabilitation and neurological research as it can provide insights into sequential learning, implicit [7] versus explicit learning and motor learning [8], work memory [9], etc. A schematic of the SRTT as currently used by REVAL is shown in Figure 3.1 below.

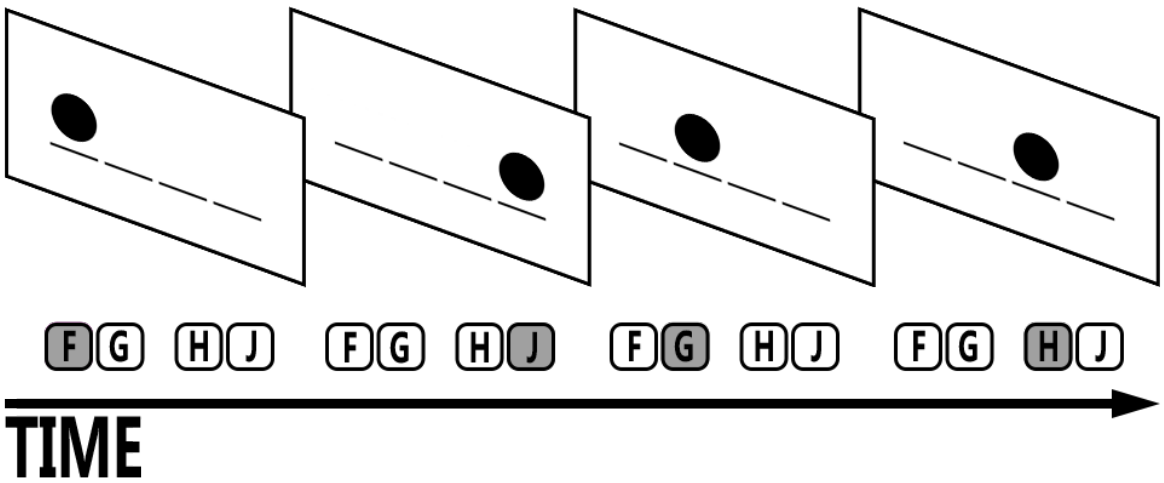


Figure 3.1: Schematic overview of the SRTT

In this interpretation of the SRTT, four black horizontal lines are presented on a white screen. These lines mark the possible positions of the visual cues. When such a cue is

presented, a black circle appears above one of these lines. The subject then responds by pressing the corresponding button on the keyboard as quickly and as accurately as possible. The keyboard buttons corresponding to the stimuli are the "F", "G", "H" and "J" keys. The "F" and "G" keys are manipulated by the middle and index finger of the left hand, the "H" and "J" by the index and middle finger of the right hand respectively. As both hands are needed to perform the the task and complete sequences, a bimanual aspect is also introduced.

3.1.2 Problem Definition

When measuring reaction times, much like when collecting any other data for research, the primary goal is to obtain data that reflects reality as closely as possible. After all, one can only draw sound conclusions from sound data. Ideally, millisecond accuracy would be achieved.

However, due to the architecture of modern operating systems, absolute millisecond simply cannot be guaranteed on modern personal computers. In the words of Schneider et al.:

> If you interpret millisecond precision to mean that no individual time measurement is ever off by more than a millisecond, then you cannot use modern personal computers running common desktop operating systems to conduct the research. At times, operating systems take away processing time from any program. Even if the program is reading a microsecond precision clock, the precision is only as good as the accuracy with which the processor is actually free to read the clock [10, p. 72].

Indeed, general purpose operating systems (such as Linux, Mac OS and most Microsoft Windows) are constantly performing critical background tasks, effectively pausing other processes. This can cause applications to miss certain clock ticks. In theory, real-time operating system (RTOS)s such as FreeRTOS, Windows CE, etc. should provide the ability to adhere to these strict timings. However, such operating systems are often more suited to lightweight embedded applications due to their restrictions and limited functionality.

In addition to this, there are many other factors to consider when discussing reaction time measurements. While the task of displaying a visual cue on screen, waiting for a response and recording the time difference between these events may seem straightforward, this simple task requires a series of subtasks to be performed. Each of these may induce its own source of latency. For example, when an application processes an instruction to display a stimulus on the screen, there is a certain delay between the moment this instruction is processed and the stimulus is actually displayed. I.e. when this instruction is processed it is converted into a system call for the operating system to process. However, the operating system might first grant priority to other processes. Furthermore, most monitors have a fixed refresh rate: for example, if a monitor has a refresh rate of 60Hz, it means that its display will only update every $1000ms/60Hz = 16.66ms$. This means that if the draw call is not synchronized with the monitor refresh rate, an additional 16ms of latency might occur if the draw call was processed immediately after the monitor had refreshed. Finally, pixels of liquid crystal display (LCD) monitors themselves can have response times of up to twenty milliseconds [11]. Similar to stimulus presentation, the processing of stimulus responses introduces delays as well, depending on the sampling rates of keyboards and latency of the operating system relaying the key press to the application.

3.1.3 Objectives

The primary goal of this chapter is to identify and measure the aforementioned delays as accurately as possible, as well as the influence of hardware, software and other parameters in order to determine whether these are significant to the acquired test data. The secondary goal is to research and propose, if possible, a solution which offers more accurate measurements.

3.2 E-Prime

To perform SRTT measurements, REVAL currently makes use of E-Prime 2. E-Prime is a software suite developed by Psychology Software Tools, Inc. (PST) which is primarily intended for use in behavioural and neurological research. It comprises, among others, E-Studio, E-Run and E-Basic. E-Studio is the experiment design environment, E-Run the runtime environment in which trials are executed after compilation and E-Basic its proprietary scripting language. Creating experiments is relatively straightforward, as objects can simply be dropped on a procedural time line (see Figure 3.2).

Figure 3.2: Procedural time line of an E-Prime SRTT trial

Each object on the time line is executed in order, from left to right. The 'preparation' object sets the stimulus to present in the current trial. The 'basicConfiguration' object displays a blank screen (i.e. a white screen containing only the horizontal lines, but no stimulus) after which the 'strimTrigger' clears accumulated key press events and re-enables the keys. The 'stim' object presents the desired stimulus, followed by the 'evaluate' object finally processing the registered key press.

On their website, PST [12] claims "E-Prime provides millisecond precision timing to ensure the accuracy of your data." In the E-Prime manual, this claim is further elaborated upon as follows:

> Operationally, E-Prime defines the interpretation of millisecond precision as the ability to report any reaction time or timing duration such that:
>
> 1) a measured and reported timing precision standard deviation is less than half a millisecond.
> 2) recordings are withing a millisecond from the time they are available to the computer hardware. When errors occur, they should be detectable, should not increase measurement variance by more than $1ms^2$. [...]
> 3) screen refreshes (displays of the full screen) can be tracked without misses 99.9% of the time, and misses can be detected and reported.

3.3 Java

To provide a point of reference to compare the performance of E-Prime to, a simple SRTT application was developed in Java. As this application is only intended for comparison purposes, it encompasses only the first of eight sequence blocks included in the full SRTT.

3.3.1 JavaFX

Initially, a JavaFX version was considered. However, early testing revealed that the drawing method alone required execution times upwards of 1 millisecond. This was measured by calling the 'System.nanoTime()' method before and after calling the drawing method.

3.3.2 AWT

Because of the high JavaFX latency mentioned earlier in Section 3.3.1, the decision was made to migrate to AWT. To ensure optimal performance, the different stimuli screens are pre-rendered to 'BufferedImages'. These BufferedImages are then stored in an array. When the screen is updated, the pre-rendered frame is simply copied to the Graphics object. The application contains no further optimizations, such as synchronization with monitor refresh rate, as the Java Development Kit does not provide this functionality.

3.4 Measurements

3.4.1 System specifications

All measurements were performed on an Asus Zenbook UX31A notebook computer, unless otherwise specified. The system specifications are presented in table 3.1

Table 3.1: Asus Zenbook UX31A specifications

Processor	Intel Core i7-3517U
Memory	4GB DDR3 1600Mhz
Graphics	Intel HD Graphics 4000 (integrated
Storage	SSD
Monitor	13.3" 1920x1080 IPS panel (integrated)
Refresh rate	60Hz
Operating System	Windows 10

3.4.2 Java AWT Processing time

To verify the performance and inherent latency of the Java AWT application, the processing time was measured under differing circumstances (see Figure 3.3).

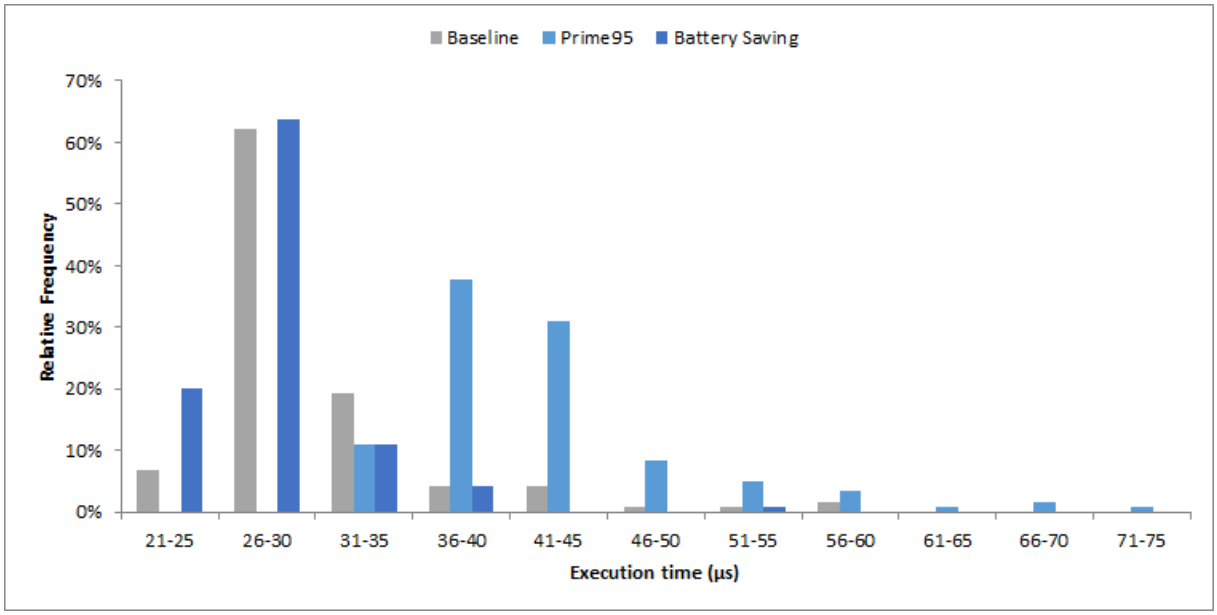


Figure 3.3: Execution times of the AWT version of the SRTT

These measurements represent the time between the registration of the key pressed event by the application and completion of the repaint method. The graph shows histograms of the measured execution times under three circumstances:

1) baseline: The system is plugged into the power outlet with power plan settings set to best performance;

2) Prime95: The system is plugged into the power outlet with power plan settings set to best performance and running a Prime95 stress test on all cores;

3) battery saving: the system is running on battery power, with power plan settings set to power saver.

Measurements indicate that 100% of execution times (t_{ex}), even those during the stress test, lie at or below 75μs with a significantly right-skewed distribution.

3.4.3 Measuring Apparatus

To perform the delay measurements, it is first necessary to determine an adequate method of measurement. The delays of interest are delay in stimulus presentation (t_{stim}) and the delay in registration of stimulus response (t_{resp}). It is, however, not possible to measure these separately, as there is no manner for the computer to externalize a signal at the exact moment the draw call is called. Additionally, determining the exact actuation point of a keyboard button is nigh impossible without specialized equipment. Because of this, the combined delay (t_{delay}) will be measured: this is achieved by measuring the delay between the stimulus response and the presentation of the next stimulus. It should be noted that this method of measurement introduces a small error, as t_{ex} is included in these measurements. However, as the delay is expected to be tens of milliseconds and $t_{ex} \leq 75\mu s$, this influence is considered negligible.

This is achieved by providing the keyboard input using not a traditional keyboard, but a Sparkfun Pro Micro micro controller. The Pro Micro is centered around the ATmega32U4 microprocessor, which features a full-speed USB transceiver, enabling it to natively emulate a USB keyboard [13]. To complete the measuring apparatus, the Pro Micro will be combined with an Arduino Uno: the Pro Micro acts solely as input device, while the Arduino performs the delay measurement. This distinction is made to ameliorate accuracy: as each micro controller performs a single task, both can dedicate as many clock cycles as possible while performing their respective task. A diagram of the circuit is shown in figure 3.4 below.

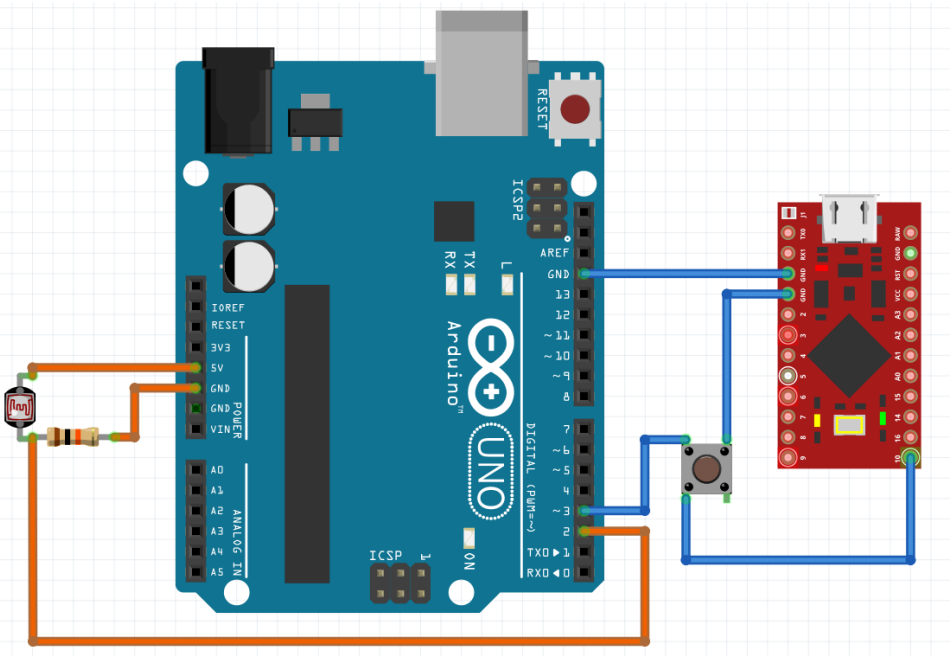


Figure 3.4: Diagram of the measuring circuit

Both microcontrollers share a common ground, as well as a push button. By connecting the push button to both micro controllers, both receive the button signal simultaneously.

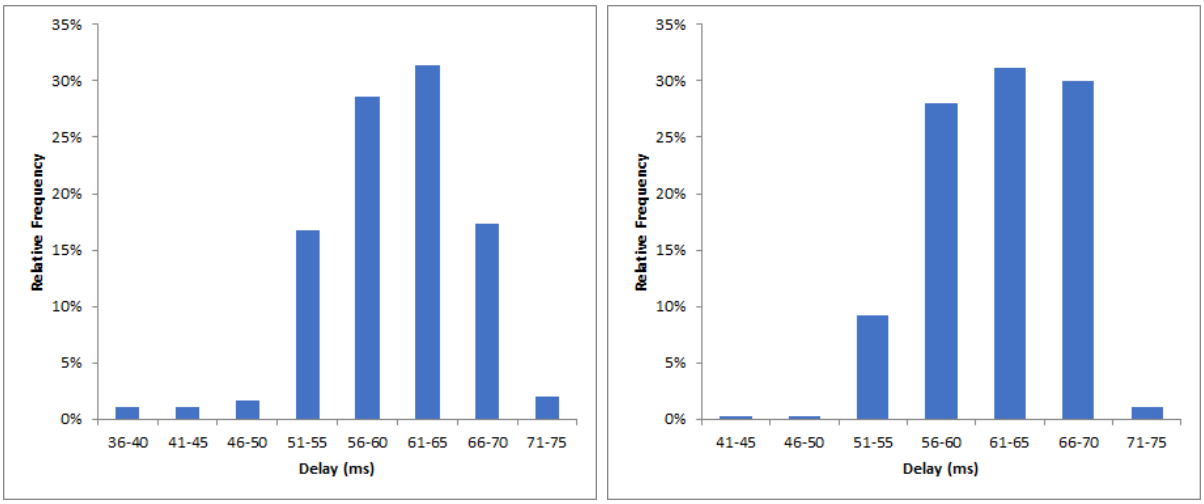
While the Pro Micro sends a key prompt to the SRTT computer, the Uno records the time stamp using the micros()-function. On 16Mhz microprocessors, this function has a resolution of $4\mu s$ [14]. Detection of stimulus presentation is achieved with a light dependent resistor (LDR): the LDR is held against the screen on the location of the currently presented stimulus. As the next stimulus is presented, this black circle will move to a new location, causing the pixels in front of the LDR to change from black to white. This sudden change in light intensity triggers an interrupt on the Uno and a second time stamp is saved, after which the delay is then calculated by subtracting the first time stamp from the second. To prohibit light other than that of the screen to reach the LDR, it is shielded by two pads of black foam.

3.4.4 Delay and Influence of System Parameters

This section discusses the measurements of the delays ($t_{delay} = t_{resp} + t_{ex} + t_s tim$, with $t_{ex} \approx 0$), as described in section 3.4.3. First, a baseline measurement is performed. This measurement serves a dual purpose: to report the performance of the system described in section 3.4.1 as well as provide a reference point to which the influences system parameters can be compared.

Baseline

The baseline measurements were performed under optimal conditions: i.e. the system was plugged into the power outlet and power plan settings set to 'best performance'. Results are shown in Figure 3.5 below.



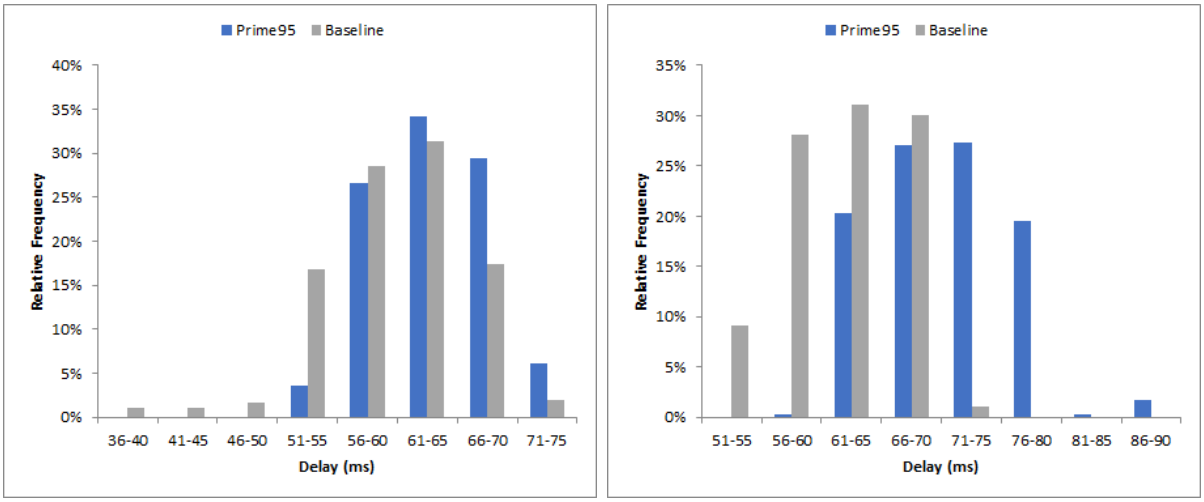
(a) Java AWT - mean 60.4ms - SD 6.1ms (b) E-Prime - mean 62.3ms - SD 4.9ms

Figure 3.5: The baseline delay measurements

Results are quite similar for both applications: the Java data exhibits a slightly lower mean delay with some sub forty millisecond values, while E-Prime demonstrates a slightly lower SD.

System Load

The measurements shown in Figure 3.6 below were performed while running a Prime95 stress test on all cores.



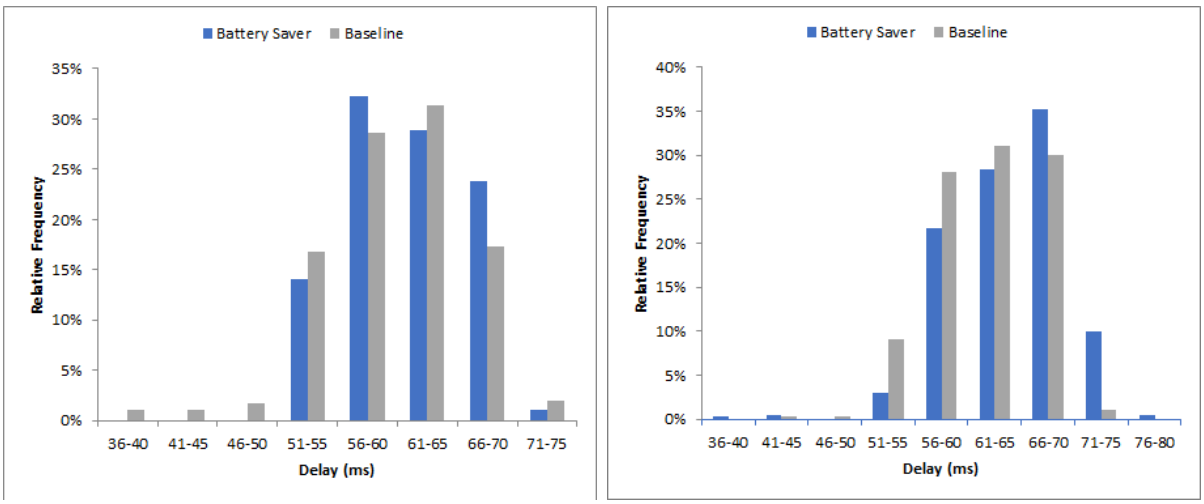
(a) Java AWT - mean 63.4ms - SD 4.8ms (b) E-Prime - mean 72.3ms - SD 14.7ms

Figure 3.6: Influence of heavy system load

Under heavy system load, the Java latency is less skewed to the left, resulting in a slightly higher mean but lower SD. E-Prime, however, does show a distinct difference under heightened system load: the histogram shows a slight skew to the right as well as an overall shift towards higher latency, resulting in a significant increase of mean as well as SD.

Battery Saver

Figure 3.7 below depicts the measurements performed with the system operating on battery power, with the power plan settings set to 'power saver'.



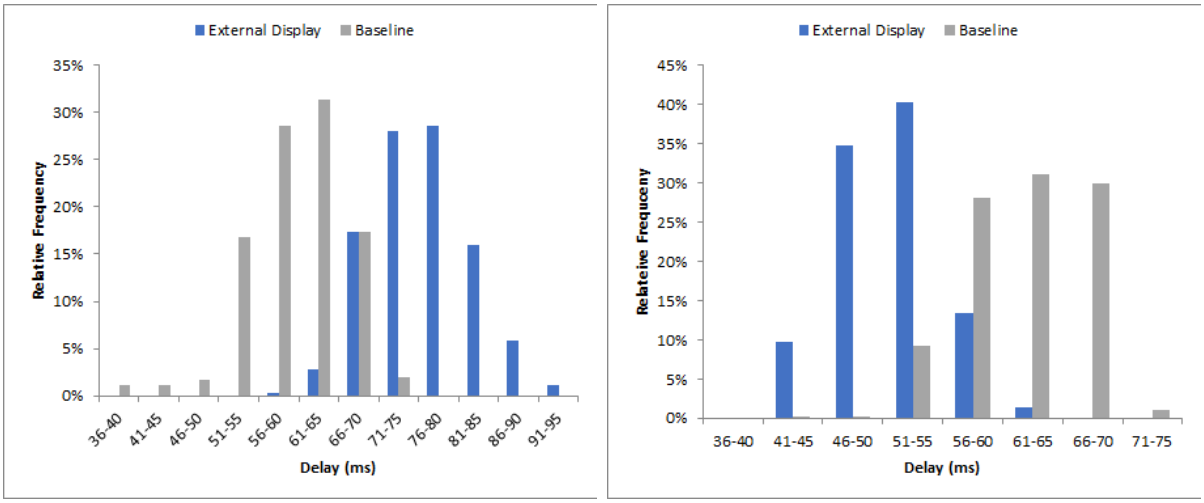
(a) Java AWT - mean 61.3ms - SD 4.9ms (b) E-Prime - mean 64.8ms - SD 12.34ms

Figure 3.7: Influence of Battery Saver Mode

Interestingly, the mean value only very slightly increses while the standard deviation drops, for the Java AWT version. This is probably due to the fact that the low power mode of the system prevents very low latency instances, resulting in a decrease in skew to the left. In contrast, while he mean value for E-Prime has decreased from the Prime95 measurement, the standard deviation remains high as, in this case, low latency measurements do still occur.

(External) Display

The following measurements were performed on an external monitor rather than the system's native display. The screen used is a Hewlett-Packard L1950: it supports resolutions of up to 1280x1024 and refresh rates of up to 75Hz. For the purpose of this measurement, the refresh rate was set to 60Hz, identical to that of the native display. Results are shown in Figure 3.8.



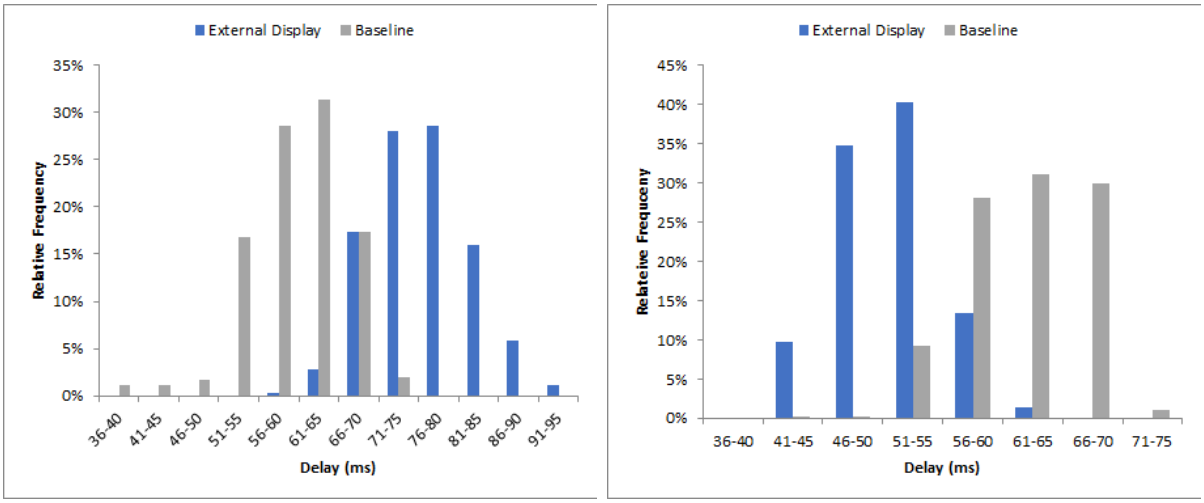
(a) Java AWT - mean 75.9ms - SD 6.1ms (b) E-Prime - mean 51.4ms - SD 6.5ms

Figure 3.8: Influence of External monitor

In this case, Java performance is worse as values are shifted significantly to the right. In contrast, E-Prime measurements are shifted markedly to the left, even outperforming the baseline with both a lower mean delay as well as a smaller SD.

Refresh Rate

The following measurements were performed on an external monitor rather than the system's native display. The screen used is a Hewlett-Packard L1950: it supports resolutions of up to 1280x1024 and refresh rates of up to 75Hz. For the purpose of this measurement, the refresh rate was set to 75Hz, rather than the native 60Hz refresh rate of the system's native display. Results are shown in Figure 3.9.



(a) Java AWT - mean 64.2ms - SD 4.7ms (b) E-Prime - mean 54.6ms - SD 4.8ms

Figure 3.9: Influence of Refresh Rate

Interestingly, while the Java application benefits from an increase in refresh rate from 60 to 75Hz on the external display, E-Prime did not perform better. While at first rather puzzling, this was later discovered to be due to E-Prime simply forcing the highest possible refresh rate. This meant that in the previous results the actual refresh rate was 75Hz, even though a refresh rate of 60Hz had been set in both Windows and E-Prime settings.

3.4.5 Measured versus Real Reaction Time

As the goal of this thesis is ascertain whether the imposed delays significantly influences the acquired data, the measuring apparatus described in 3.4.3 was adapted to measure the Δt between the onset of the stimulus and the response. To achieve this, the Arduino Uno was reprogrammed to collect a first time stamp when the LDR detects a sudden decrease in light intensity, and a second time stamp on detection of a button press. As this decrease in light intensity corresponds to the actual onset of the stimulus (i.e. the moment at which the stimulus becomes visible to the subject) rather than the moment the drawing call was executed, the stimulus presentation delay is eliminated. The same is true for the stimulus response delay, as the Arduino interrupt response time should be no more than 6μs [15].

Java AWT

The following data was acquired by performing the Java AWT SRTT using the aforementioned measuring apparatus as stimulus response input. The data logged with the Java application was then compared to the data obtained using the measuring apparatus. For each measurement, the 'real' response time (t_r) was subtracted from the response time measured using Java (t_j), so that $\Delta t = t_j - t_r$. The results are shown in Figure 3.10.

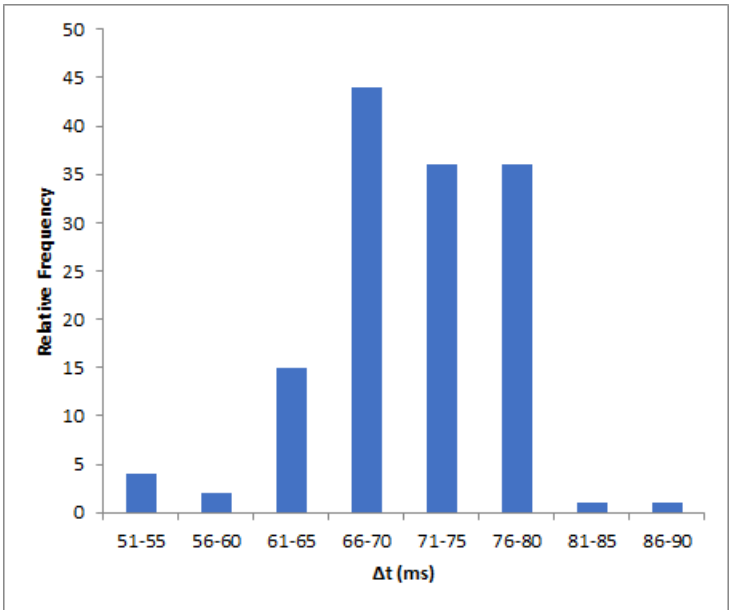
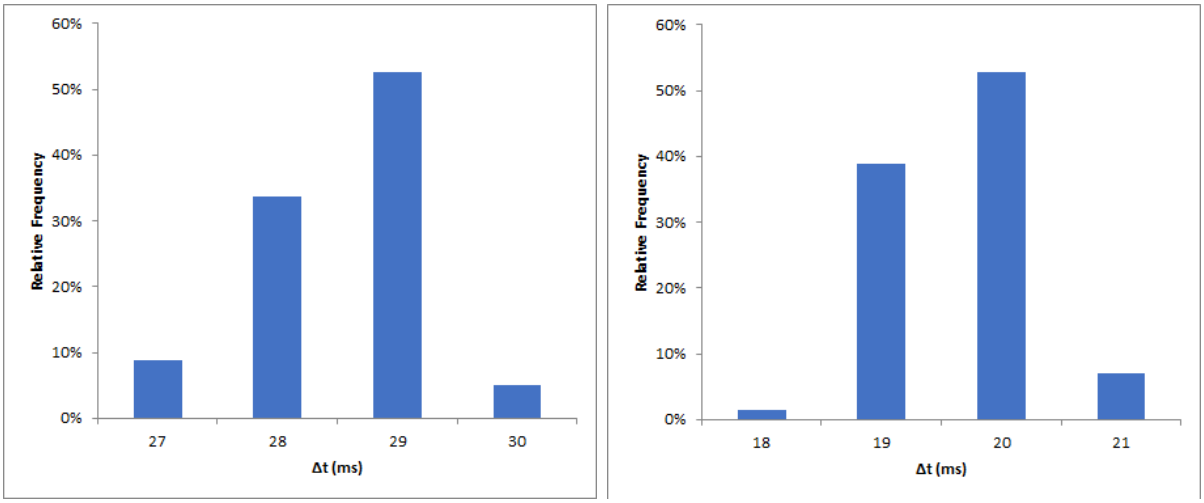


Figure 3.10: Δt (Java)

With mean = 70.9ms and standard deviation (SD) = 6.2ms. A 10ms increase in the mean is obtained in comparison to the baseline delay measurement, with a very similar SD. This is most likely due to the up to 16.66ms discrepancy between calling of the draw method (and thus initiating the reaction time measurement in the SRTT) and the next refresh cycle of the monitor.

E-Prime

The data depicted in Figures 3.11a and 3.11b was generated analogously. Data logged with both E-Prime and the measuring apparatus was compared and subtracted to obtain Δt.



(a) E-Prime 60Hz - mean 28.5ms - SD 0.7ms (b) E-Prime 75Hz - mean 19.7ms - SD 0.6ms

Figure 3.11: Delay in battery saver mode

The data above confirms the claim in section 3.2 that E-Prime does indeed guarantee a standard deviation of less than 1ms. However, an offset is added to the reaction times dependant on the refresh rate: at 60Hz, an offset of 28.5ms is added. At 75Hz, this offset decreases even further to 19.7ms.

3.4.6 Interpretation

The data gathered in sections 3.4.4 and 3.4.5 shows that for the Java application the latency between the physical button press and the presentation of the subsequent stimulus is a good indicator for the reaction time measured by the application itself. This, however, is not the case for E-Prime. While in some circumstances E-prime showed greater mean delay and sometimes much greater standard deviation in presenting the subsequent stimulus, it nevertheless manages to produce markedly consistent timing results. This indicates that in E-Prime, display of the blank screen or subsequent stimulus is not considered time critical while at the same time great effort is made to maintain synchronization with the monitor refresh cycle when the stimulus is eventually presented.

Additionally, thanks to the repeatability and consistency of the offset and marginal standard deviation, this opens up the possibility of absolute rather than reaction time measurement. After all, if the offset and its SD are known, absolute reaction time can simply be obtained by subtracting the offset from the measured reaction time and adding both variances together [16]. It should be noted, that as indicated in section 3.4.4, E-Prime has been shown to occasionally force a different refresh rate than indicated by the OS and even its own settings. A researcher unaware of this phenomenon, might simply join or compare multiple data sets gathered at different refresh rates (despite identical settings), possibly compromising accuracy slightly. For most reaction time research, however, relative reaction time is adequate [17].

Finally, literature suggests that due to the nature of human reaction times and the relatively high accompanying variances, even the relatively large standard deviation obtained

from the Java implementation (section 3.4.5) might not be significant when comparing relative reaction times. For example, observing the OBSERVE group discussed by Heyes et al.: population size $n = 40$, mean reaction time = 555.01 and standard error = 20.1 in [8]:

$$\textit{Standard error or } \sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

$$\Leftrightarrow \sigma_{\bar{x}} \times \sqrt{n} = \sigma$$

$$\Leftrightarrow 20.1 \times \sqrt{40} = 127.12 = \sigma$$

$$\Leftrightarrow \sigma^2 = 16159.5$$

$$\Leftrightarrow \sigma^2_{new} = 16159.5 + \sigma^2_{\Delta t} = 16159.5 + 6.2^2 = 16.165,7$$

$$\Rightarrow \frac{\sigma_{new}}{\sqrt{n}} = \frac{\sqrt{16.165,7}}{\sqrt{40}} = 20,123$$

Which means that after adding both variances together, the standard error is only increased by 0.0011%. This indicates that for relative reaction time measurements, the added variance due to the 'inaccurate' Java implementation is negligible. Only when considering absolute reaction time measurements, would this implementation introduce significant inaccuracy, as the mean offset of 70.9 ms would translate to a 12,8% increase of the mean reaction time.

3.5 An Embedded Approach

As described in section 3.4.6, the improved accuracy of reaction time measurements with E-Prime was not deemed significant. However, when considering absolute reaction time measurements, the opposite is true. Indeed, there is even some room for improvement in completely removing the mean offset. A possible way to achieve even greater accuracy is by transitioning from traditional personal computers to an embedded system. To test this hypothesis, this chapter will first discuss the development a proof of concept, followed by some suggested improvements and finally the advantages and disadvantages of an embedded system versus a PC-based platform.

3.5.1 Proof of Concept

As the Arduino Uno and Sparkfun Pro Micro had proven their worth while performing the measurements earlier in this chapter, they were reused to create and test a prototype for the embedded SRTT (see Figure 3.12.

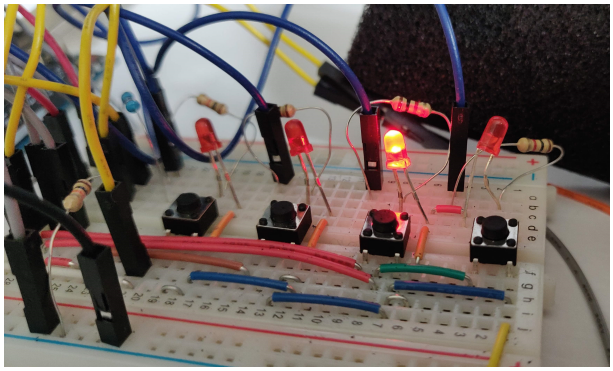


Figure 3.12: Prototype of the embedded SRTT

The construction of the SRTT device is quite straightforward and consists of the following parts: An Arduino Uno, four red LEDs, four push buttons and an assortment of jumper cables. Similar to the traditional SRTT on PC, each LED corresponds to the button directly in front of it. Initially the full first block of the SRTT was implemented on the device. However, as the Uno features no more than 2 interrupt pins, not all push buttons could be connected to an interrupt. This meant that the read outs of the buttons had to be performed during the control loop, which would slightly impede performance as the status of the buttons could only be read after each loop was completed. In order to test optimal performance, the set-up was slightly altered. Two of the buttons were assigned to the interrupt pins 2 and 3 of the Arduino Uno. When one of these buttons is actuated an interrupt is triggered, causing the loop to halt while an interrupt service routine (ISR) is called. In this case, the ISR saves a time stamp to a volatile long using the micros()-function and sets a boolean related to the pressed button to true. The ISR is then exited and during the next execution of the loop, the time stamp and button identifier are added to an array, the boolean is reset to zero and the next stimulus is presented.

The Pro Micro is then used in combination with the LDR to measure the latency between the actuation of the button and fading of the stimulus. The measured latency contains the registration and processing of the stimulus response as well as the presentation of the subsequent stimulus. The results are shown in Figure 3.13 below.

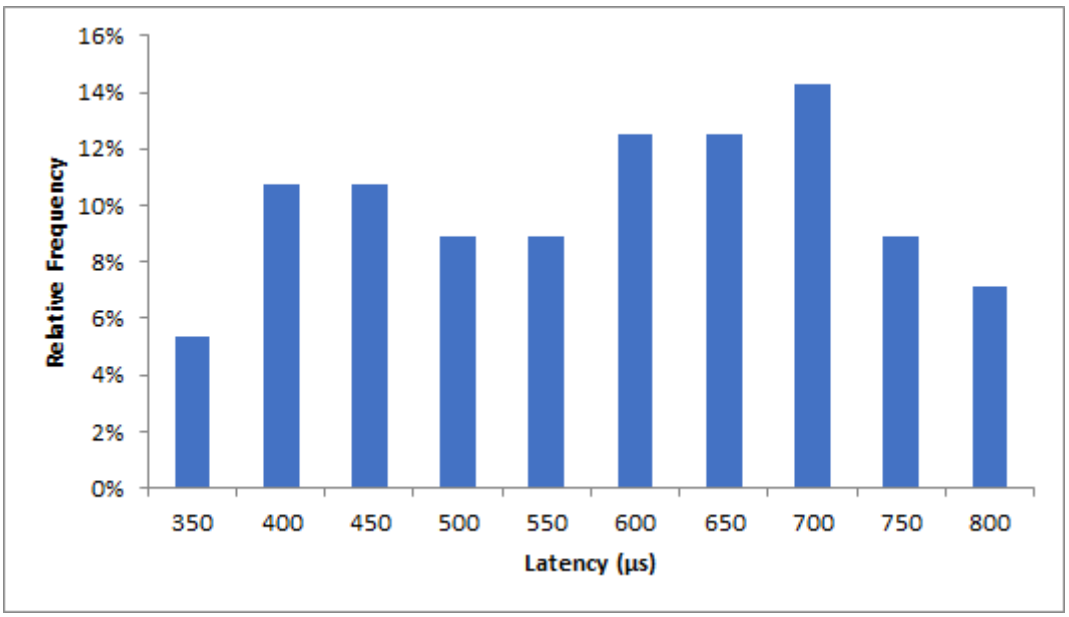


Figure 3.13: Latency of the embedded SRTT

With mean = 0.56ms and standard deviance = 0.13. Not only has the SD decreased further from 0.6ms in E-Prime to now 0.13ms, but the mean offset has also decreased to less than 1ms, demonstrating the possibility of achieving sub millisecond accuracy in measuring absolute reaction time.

3.5.2 Improvements

While the aforementioned proof of concept is promising, it is far from a finished product. Some suggestions to elaborate upon this idea are the following:

1) There are far more capable devices on the market than the Arduino Uno with its limited 16Mhz clock speed and mere two interrupt pins. A suggestion is a STM32 based board, as the STM32 microprocessor is capable of clock speeds of up to 72Mhz and can attach interrupts to any of its input pins.

2) Professionalizing the appearance of the device by designing a custom PCB and enclosure.

3) The development of a lightweight companion PC application, to facilitate transferring of test data from the embedded system to a PC. This application could simply download the data from the device and convert it into a .csv file, or perhaps upload the data to a database. Additionally, a 'moderator' version of this app could be created, allowing researchers to modify trial sequences, inter trial delays, etc. without recompiling the firmware.

3.5.3 Advantages and Disadvantages

Migrating from a PC-based testing environment to an embedded system has both advantages and disadvantages, which are discussed below.

Advantages

1) Measurement accuracy greatly increased;

2) Much less costly than an E-Prime License (995USD for a single user license);

3) Identical performance across devices, in contrast to the influences of refresh rate etc. on PCs;

Disadvantages

1) A PC is still needed to download measurement data from the device;

2) While E-Prime allows for a wide array of tasks to be programmed, embedded hardware is tailored for the application, which means repurposing options for devices are limited.

Chapter 4

Conclusion

The first part of this thesis discussed the development a bimanual tracking task application. This project comprised the implementation of the Misfit Technologies NEMA17 Smart Steppers as input, as well as the necessary firmware adaptations made for them to provide feedback the participant's input in the form of counter torque. Furthermore, the ability to communicate triggers to electroencephalography monitors is provided to allow for synchronization of EEG scans with the execution of the task. A user-friendly yet powerful graphical user interface is provided, with an array of customization options of trials as well as the ability to save and load trial sequences for effortless repeatability, combined with user input validation to prevent erroneous input. Finally, raw data as well as aggregated statistical data is exported in .csv files.

The latter part of this thesis focuses on the validation of serial reaction time measurements and investigates whether latency associated with the use of general purpose operating systems on personal computers can invalidate measurements performed on these aforementioned platforms. It achieves this by comparing measurements obtained from E-Prime, an established data acquisition platform, with a self developed Java AWT implementation across a range of differing system parameters. In the end, E-Prime proves to indeed offer a formidable precision concerning reaction time measurement. The thesis is concluded with a proposal of an embedded implementation of the serial reaction time task, accompanied by the development and initial testing of a proof of concept for this platform.

4.1 Future Work

The main concern towards the future for the bimanual tracking task is for the issues with the Smart Steppers to be remedied as soon as possible, depending on a firmware update or other solution conveyed by Misfit Technologies, thus ensuring normal operation for the master students who will use the platform for their master's thesis. Further, but optional, additions could be implementation of the application into the the REVAL web platform, or the implementation of a database link, allowing centralized storage of data as well as parameter files.

Future work concerning the serial reaction time task mainly encompasses the development of more polished version of the embedded system implementation of the SRTT. Towards this purpose, the improvements suggested in section 3.5.2 might serve as guidelines.

Bibliography

[1] R. Debien and J. Vanstraelen, "Ontwerp en implementatie van een revalidatie platform gebruik makende van de leap motion sensor," Master's thesis, FIIW, UHasselt, Diepenbeek, Belgium, 2018.

[2] *Smart arm-based microcontroller*, SAM D21G, Rev. 09/2015, Atmel, Sep. 2015.

[3] *14-bit on-axis magnetic rotary position sensor with 11-bit decimal and binary incremental pulse count*, AS5047D, Rev. 1-07, AMS, Apr. 2016.

[4] *Dual full-bridge dmos pwm motor driver*, A4954, Rev. 1.4, Allegro MicroSystems, Aug. 2012.

[5] R. Hafijur. (Feb. 2017). Model view controller (mvc) pattern, [Online]. Available: `https://tecorb.com/2017/02/18/model-view-controller-mvc-pattern/`.

[6] Biosemi.com. (n.d.). Biosemi usb trigger interface (for presentation or e-prime), [Online]. Available: `https://www.biosemi.com/faq/USB%20Trigger%20interface%20cable.htm` (visited on 05/13/2019).

[7] E. M. Robertson, "The serial reaction time task: Implicit motor skill learning," *The Journal of Neuroscience*, vol. 27, no. 38, pp. 10 073–10 075, Sep. 2007.

[8] C. Heyes and C. Foster, "Motor learning by observation: Evidence from a serial reaction time task," *The Quarterly Journal of Experimental Psychology*, vol. 55 A, no. 2, pp. 593–607, 2002.

[9] N. Unsworth and R. W. Engle, "Individual differences in working memory capacity and learning: Evidence from the serial reaction time task," *Memory & Cognition*, vol. 33, no. 2, pp. 213–220, 2005.

[10] W. Scheider, A. Eschman, and A. Zuccolotto, *E-prime user's guide*, Psychology Software Tools, Inc., 2002.

[11] T. Elze and T. Tanner, "Liquid crystal display response times estimation for medical applications," *Medical Physics*, vol. 36, pp. 4984–4990, 2009.

[12] pstnet.com. (n.d.). E-prime, [Online]. Available: `https://pstnet.com/products/e-prime/` (visited on 05/28/2019).

[13] *8-bit microcontroller with 16/32k bytes of isp flash and usb controller*, ATmega32U4, Microchip, Apr. 2016.

[14] arduino.cc. (n.d.). Arduino reference: Micros(), [Online]. Available: `https://www.arduino.cc/reference/en/language/functions/time/micros/` (visited on 05/28/2019).

[15] N. Gammon. (2012). Interrrupts, [Online]. Available: `http://www.gammon.com.au/interrupts` (visited on 05/29/2019).

[16] D. Bock. (n.d.). Why variances add - and why it matters, [Online]. Available: `https://apcentral.collegeboard.org/courses/ap-statistics/classroom-resources/why-variances-add-and-why-it-matters` (visited on 05/29/2019).

[17] S. Reimers and N. Stewart, "Presentation and response timing accuracy in adobe flash and html5/javascript web experiments," *Behavioural Research*, vol. 47, pp. 309–327, 2015.