Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

הפקולטה להנדסה
ע"ש איבי ואלדר פליישמן
אוניברסיטת תל-אביב

Project no.16-1-1-1112

Final Report

An experiment system interfacing equipment for recording electrical activity of single neurons in the human brain

Authors:

Tal Marziano	200299014
Omer Shamia	301497376

Supervisor: Dr. Ariel Tankus

Contents

1 Abstract

Our project is aimed to build experiment system software and to design and implement a more flexible and faster synchronization system between the neural activity recorder and the software system. The new software should perform in “near Real-time” on the one hand and on the other hand should be more flexible, maintainable, dynamical and compatible for both Linux and window operating systems.

The experiment system will serve for investigating speech in patients implanted with electrodes for recording single unit activity from their brain.

On the hardware side, precise log creation and synchronization at the millisecond level are required in order to synchronize the experiment system (“the paradigm”) with the neuronal recording machine in a precise manner.

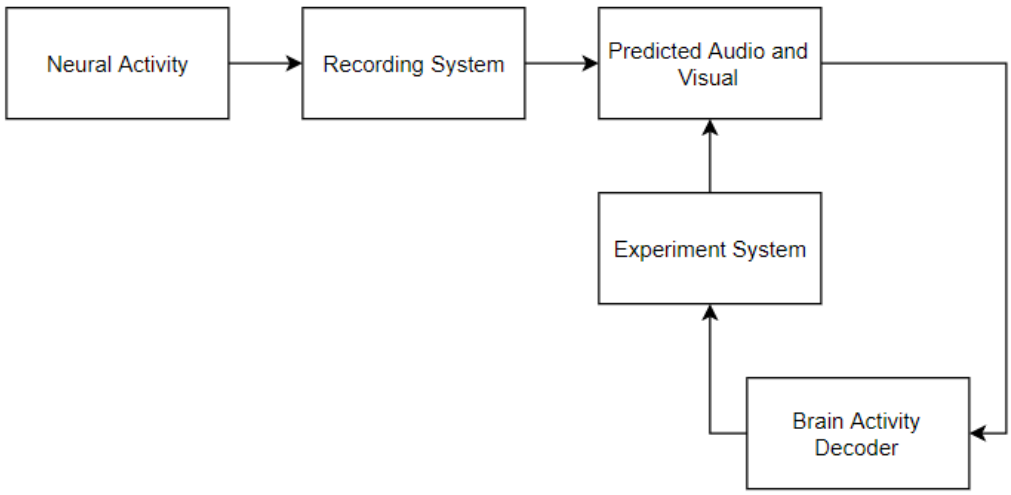

FIGURE 1-SYSTEM’S BLOCK DIAGRAM

2 Introduction

Dr. Ariel Tankus field of research focuses on the subject of speech and enables a unique view to new brain studies with the help of volunteering patients with Parkinson's disease and Epilepsy. For clinical reasons, these patients have been implanted with depth electrodes in their brains for better diagnostics of the disease. These studies are done using neuron recordings, Intra-operative experiments, speech analysis and decoding behavior from electrical neural activity.

The main research for which the project is done for is based on two components – the experimental paradigm software which is used to interact with the patient and the neural recording system (Neuroport, Blackrock) which is used to record the neural activity as detected by the implanted electrodes.

The combinations of both technologies allows to crosscheck the behavior of the patient, that is his or her interaction with the software in comparison with changes in brain activity which appears while performing those tasks.

A number of experiments have already been done with those systems and in this project we would like to improve the experiment software and its interface as well as the synchronization between these two systems for future and more advances uses in this scope.

The project will focus on two layers of the paradigm – upgrading the software, including implementation for existing features as well as adding new advances capabilities in order to build dynamical and maintainable infrastructure for future research and upgrades. In addition, the system which run the experiment software would be connected to a synchronization device which will be used for synchronize between the neural activity recording to the time logs recorded which keep track of the patient's actions in the experiment.

3 Theoretical Background

3.1 Visuomotor Coordination

Visuomotor coordination[3] is a skill employed in a wide range of activities, from the most fundamental, for example, identifying and picking up food, to the highly specialized, such as endoscopic operation or flying an airplane. These processes require the brain to process visual input information and transform it into an actual output motor plan, relying on high-level perception of the world being visualized. Visual perception has long been an interdisciplinary field integrating insights and data from neuroscience, psychology, and computation and so is the field of motor perception and coordination.

3.2 Brain Activity

Two different neural populations in the human medial-temporal lobe were discovered[3] - One consists of motor-like neurons representing hand position, speed or acceleration during the motor task but not during the visuomotor or visual tasks. The other is specific to the para-hippocampal gyrus (an area known to process visual motion) and encodes speed, acceleration, or direction of hand.

movements, but only during the visuomotor task: neither during visual-only nor during motor tasks. These findings suggest a functional basis for the anatomical sub pathway between the dorsal stream and the medial-temporal lobe. Similar to the recent expansion of the motor control process into the sensory

cortex, findings render the human medial-temporal lobe an important junction in the process of planning and execution of motor acts whether internally or externally (visually) driven. Thus, the medial-temporal lobe might serve as an integration node between the two processing streams. New findings shed new light on the brain mechanisms underlying visuomotor coordination which is a crucial capacity for everyday survival, whether it is identifying and picking up food, sliding a key into a lock, driving a vehicle, or escaping a predator.

4 Methods

4.1 Subject population

Our subject population are divided for two main groups of patients been implanted with depth electrodes in their brains. The first are patients with epilepsy which the electrodes are being held in their brain for long term of few days (up to a week) in order to get enough seizure info. On the other hand, patients with movement disorder such as Parkinson are being held for only few hours and the experiment are performed during operation.

4.2 Experimental Paradigm

Patients performed hand movements to control a computer cursor (with or without visual feedback of the consequences of their actions) or passively viewed the same visual feedback cues[2]. All tasks were performed in the same session, and the visual objects involved, namely the cursor and target, were kept simple and did not incorporate components that would emphasize ventral stream processing to reveal dorsal-stream-like processing in the ventral stream. In the visuomotor task (i.e., hand movements with visual feedback), the patients moved a joystick to control a computer cursor. They first had to position the cursor inside a ring-shaped target at the center of the screen. Following a short delay period, the target would jump to one of eight randomly lit targets, equally spaced on a virtual circle around the center. The patients would then move the cursor to follow the target (namely “Center-Out”). The target would then return to the center, with the patients following it, and the task will repeat.

In the visual-only task, another individual performed the same Center-Out task, while the patient only watched the computer screen with no hand movements. The hand of this individual was outside the visual field of the patient. Patients were explicitly instructed to track the cursor with their eyes. As no eye-tracking was performed, this was monitored by an experimenter. In the motor task, the patients moved the joystick either to the left or right according to a cue to control for motor responses. Before each

run of this task, patients were played randomly selected cues. The cue was either auditory or visual. Vision of the movement was prevented either by having the hand and joystick completely outside the field of view or by closing the eyes.

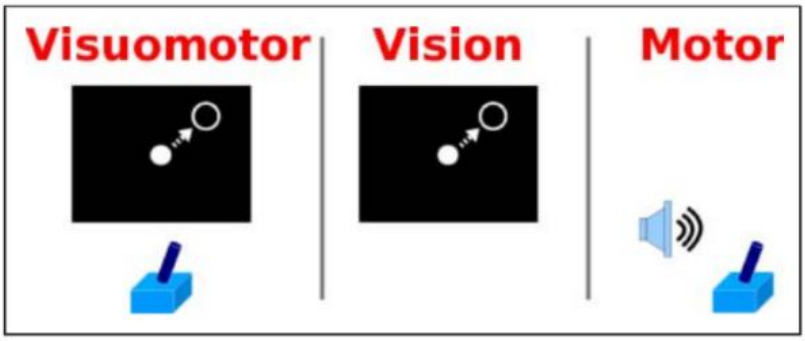

FIGURE 2-CENTER OUT TASK

4.3 Neuroport - Neural recording system

The Neuroport system (Blackrock Microsystems, Inc.) is a highly configurable, easy to use, multichannel data acquisition system used for recording and monitoring brain electrical activity. It can record both high-frequency (action potentials) and low-bandwidth (EEG) signals simultaneously, while also providing clinicians with the tools to analyze them.

The system's Neural Signal Processor provides real-time processing for up to 128 electrodes, 16 auxiliary analog channels and individual TTL or strobed-word experiment events. Multiple systems can be synchronized for more channels. The Front-End Amplifier in turn amplifies, filters and digitizes neural signals before converting them to a single, multiplexed optical output. This technology excels in real-time processing of spikes, field potentials and other physiological signals and also equipped with flexible I/O options for synchronizing with behavior, stimulus and video systems.

4.4 Arduino Uno – synchronization device

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller.

4.5 Matlab PsychtoolBox – software implementation

Matlab is a high-level interpreted language with extensive support for numerical calculations. The Psychophysics Toolbox interfaces between Matlab and the computer hardware. This unique package is a collection of free functions for MATLAB intended for use by neuroscience and vision researchers. It synthesizes and shows accurately controlled visual and auditory stimuli and interacts with the observer. The routines at the core of Psychtoolbox provide access to the display frame buffer and color lookup table, allow synchronization with the vertical retrace, support sub-millisecond timing, support video playback and capture as well as low-latency audio or hardware triggers, and facilitate the collection of observer responses through regular and specialized input devices.

5 Results

This section will be divided into two. We will first describe our work around the synchronization between the two systems, the laptop which runs the experiment and the Neuroport recording system.

The second part will describe the various features implemented in the new software as developed in Matlab.

5.1 Synchronization system

As mentioned before our goal is to send sync pulses to neural activity recorder as described in Figure 3.

FIGURE 3-SYNCHRONIZATION SYSTEM BLOCK DIAGRAM

5.1.1 Synchronization software

In this part we were learning how to send a byte through a parallel port from a laptop to an Arduino, in a method which works both for Linux and Windows operating systems.

How it works:

1. Throughout the experiment we would like to register time logs of different events and other data in order to extract important information such as cursor coordinates, user's actions and decisions, which are later processed during the analysis of the experiment.

   The purpose of generating times logs at different time intervals is to be able to match between the registered logs in the laptop and those created in the neural recording machine. Since there are two different clocks in both systems and

since time logs could get lost in the way, the use of arbitrary time intervals allows us to match between the two systems (explained in the following Figure 4).

2. Program Arduino to be able to receive incoming data from parallel port and when it does, it turns on a bit signal which is then sent to the neural recorder system.

3. When received on the recorder's end, a time log is automatically created using a built-in feature.

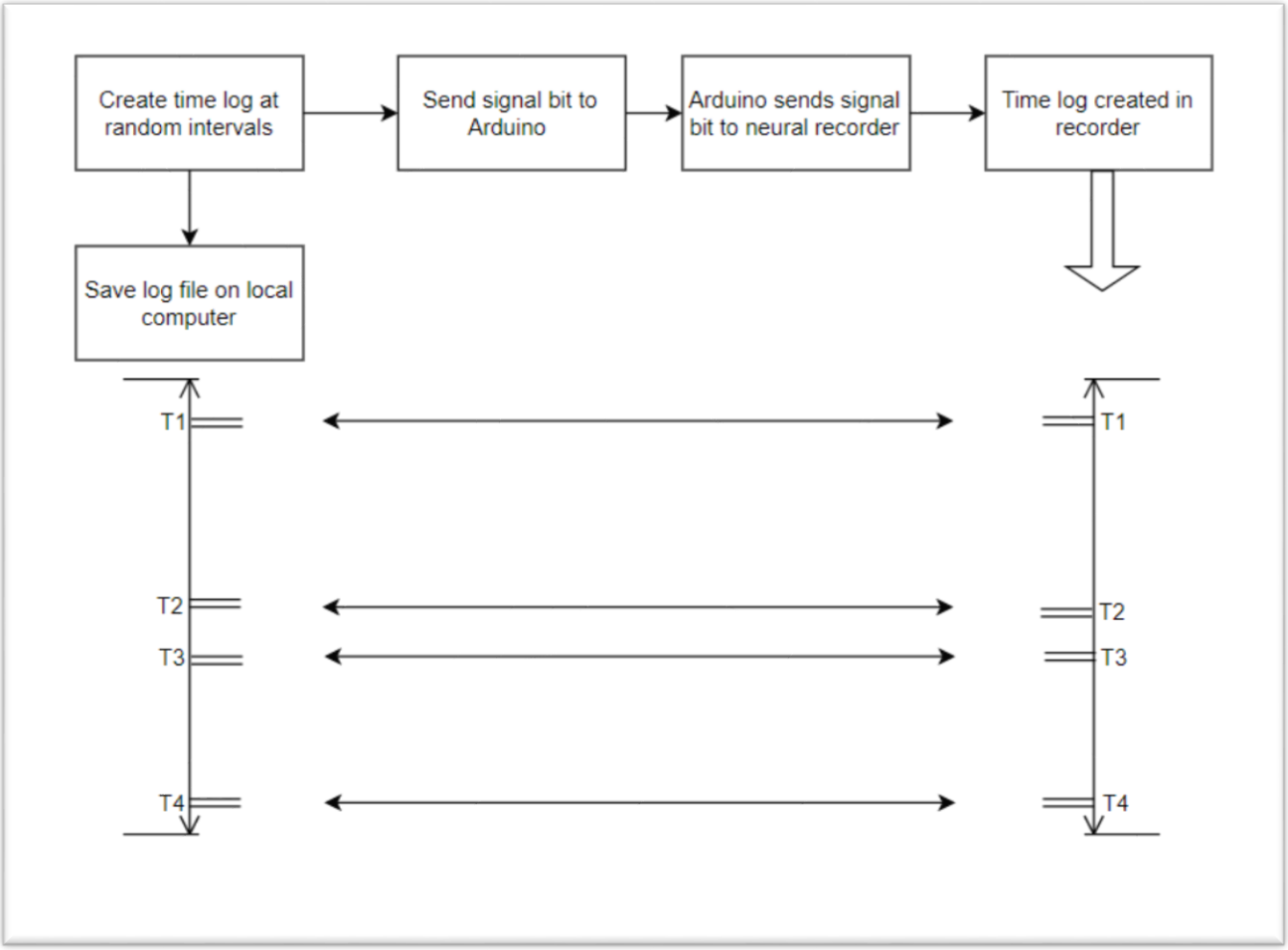

FIGURE 4-SYNCING PROCESS DIAGRAM

5.1.2 Synchronization hardware

Components used (as shown in Figure 5):

1. laptop
2. Arduino Uno board
3. BNC cable

We used an Arduino card to bridge between the two systems such that the computer sends syncing pulses and provides electricity to the board.

During the experiment we will send signals threw the USB port to the Arduino, sending of '1' will increase the voltage to 5V, while sending '2' will drop it back to 0V.

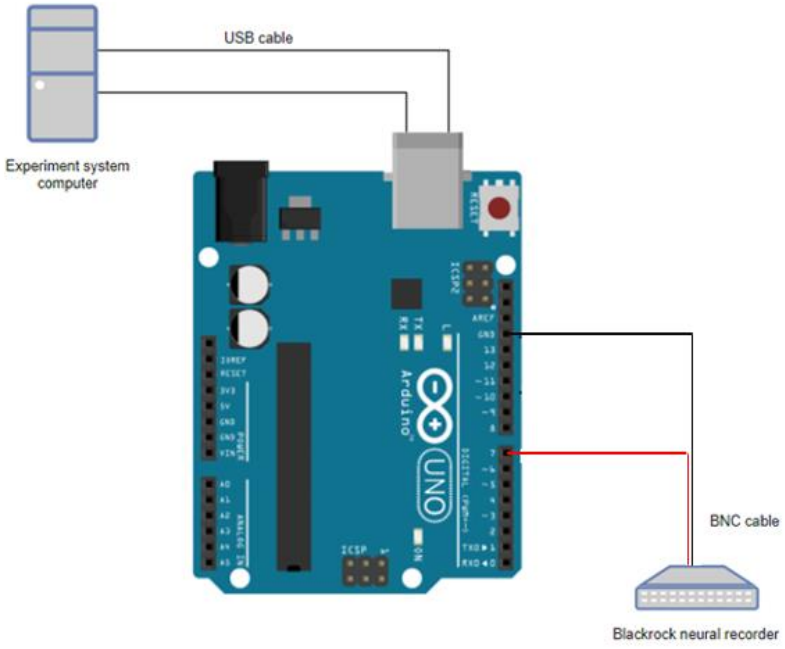

FIGURE 5-HARDWARE WIRING DIAGRAM

5.2 Experiment program

In this section we will present the new program written in Matlab using Psychtoolbox and based on specifications supplied by Dr. Ariel Tankus. This part was done in order to add new features to the program which is used for the experiment as well and making it more modular and accessible for changes and modification for future use.

The main operation modes and features:

5.2.1 Center Out

The screen board is based on 9 locations – one is located at the center of the screen and 8 other locations spread out in a circle in eight equally distanced spots on a black background in the main directions.

1. The task begins with a circular white target appearing at the center of the screen and a white dot cursor centered inside the target

2. After a certain time during which the cursor is within the center , the target “jumps” to one of the 8 locations at random.

3. Patient then needs to move the cursor to the target with either a joystick or a mouse.

4. When target arrives to its destination or if time-out has been reached, the target moves back to the center.

5. Patient needs to move the cursor back to the center and begin another round.

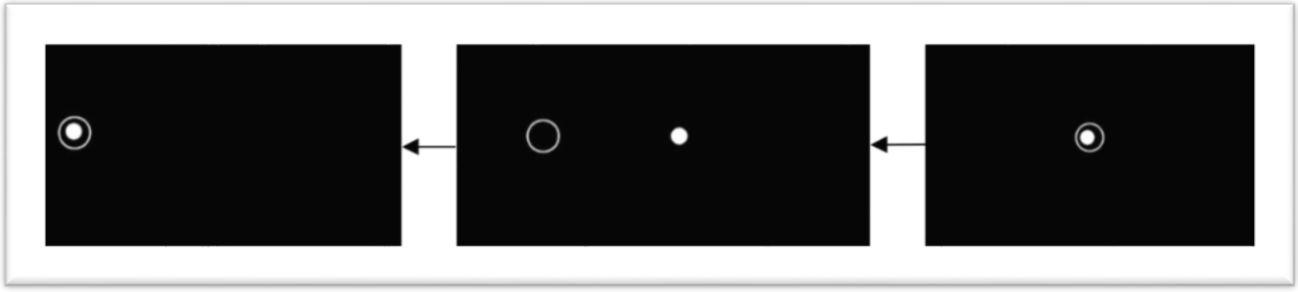

FIGURE 6-CENTER OUT TASK

5.2.2 Go-No-Go

An extension to Center-Out which includes a “Go” or “No Go” signal before the target moves. The round begins the same as Center-Out with the white circle in the middle. In this mode, the target then changes its color to either green or red randomly, then after a certain period of time (random between two limits configurable in the software) it “jumps” to one of the locations.

Only when green color (“Go” signal) was shown, the patient should follow the target as usual. When “No-Go” (red) is signaled, one should refrain from movement and stay in the center.

5.2.3 Paradigm cue manager for speech experiment

As part of the experiment, there are more advanced tasks that the patient has to perform, which are instructed by the experimenter. This mode is an aid to the experimenter and allows an organized order of tasks and creation of event logs for optimized experimental environment. When running this program, one screen is shown to the instructor with the current task, and a task screen to the patient.

The cue can be either auditory signal – a beep or a series of beeps, which are configurable, or a visual one – a rectangle in the center of the screen which changes its color to green (for hard of hearing patients).

This feature is mainly used for speech experiment (as shown in Figure 7) and also for some other movement tasks (see 5.2.4 Image Mode).

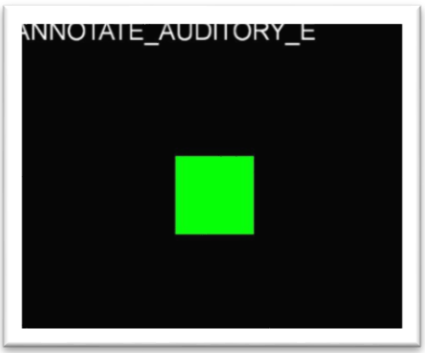

FIGURE 7-SPEECH EXPERIMENT WITH VISUAL CUE

5.2.4 Image Mode

We added the option of changing the background in order to add more options for movement task. i.e as shown in Figure 8, we implemented a maze by changing the background into a picture.

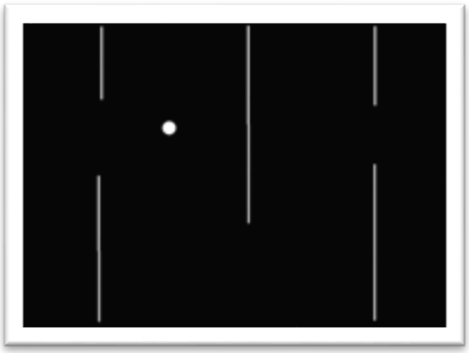

FIGURE 8-MAZE TASK USING IMAGE MODE

5.2.5 Brain Control

Current technology implemented in the experiment is the use of recorded neural activity and decoding of that information to instructions. Existing Matlab code is able to deconstruct neuronal recordings to movement – identifying the brain activity pattern which occurs when a patient is thinking about moving a cursor to a certain side or about saying a vowel. That program communicates with "BRAIN-CONTROL" section in our code, using TCP/IP protocol, and translate the data into actions such as, moving cursor and saying a vowel.

In order to demonstrate the brain control feature we built a simulator which send signals threw the socket by pressing on specific buttons on the keyboard and by that act like the brain decoder (Run as separate process).

Using these features, a recorded neural activity will be able to generate a movement (in small steps) on the screen online, to complete closed-loop tasks which involve cursor movement, as shown above.

5.2.6 Events logs

One of the main specification of our software was to produce a detailed and accurate output logs.

We can document all kinds of events in a very accurate time stamps. First, we are documenting everything that the patient sees or hear such as objects shown on the screen (i.e. Target) and exact time where beep played). Second, we are also document in a very precise way the time that the pulses are sent to the neural recorder.

Along with that we are also able to record cursor position in a very high frequency of more than 200Hz (200 mouse samples in a second).

5.2.7 Audio Record Software

In order to synchronize audio record during the experiment with the events log and with the neural record we had to build an audio record software which will work with the same timestamps of the events log.

We want to make sure we are not losing data in case of system failure so we implemented a sound recorder which saves audio data into temporary files every period which we define.

See explanation at Appendix 8.5

5.2.1. Software Configurability

In order to maintain the software's relevant, we've created a configurable software which is easy to use and configure and therefore easy for access for future developments by others which are not familiar with the software.

6 Summery and Conclusions

This project is of significant contribution to the research field of speech and brain studies. We've implemented a synchronization system which would help providing more accurate results related to the brain recording and the identification of a more precise neural activity.

Moreover, we've created a software which enhances the experimenter's ability to conduct more elaborate tests with the patients and conduct improved experiments including multiple modular modes of work with.

One of key targets was to create a dynamic and configurable software with low maintenance for long-term use and easy to use, as well as creating a clean and easy to understand code for future changes and modifications.

In conclusion, we hope our work would help in creating a better brain machine infrastructure with the ability to improve in the future and helping people with various disabilities.

7 Bibliography

1. Blackrock Microsystems website
   http://blackrockmicro.com/
2. Structured neuronal encoding and decoding of human speech features -- Ariel Tankus, Itzhak Fried and Shy Shoham - Nature Communications, 2012
   https://www.nature.com/articles/ncomms1995
3. “Visuomotor Coordination and Motor Representation by Human Temporal Lobe Neurons” - Ariel Tankus and Itzhak Fried - Journal of Cognitive Neuroscience, 2012
4. "Arduino UNO datasheet", Arduino Website.
   https://www.farnell.com/datasheets/1682209.pdf
5. PsychtoolBox Matlab package documents

   http://docs.psychtoolbox.org

8 Appendix

8.1 Important notes

1. Keyboard buttons accepted in center out and gonogo mode: ‘q’ for exiting the program
2. Keyboard buttons accepted for other opmodes:
   ‘q’ for exit
   ‘1’-‘9’ for playing beeps (as shown in 8.2.1.4)
   ‘1’-‘2’ for visual cues (as shown in 8.2.1.5)
3. Brain simulator (XXXX) and audio recorder (XXXX) should be executed seperatly in other Matlab process.
4. For better debugging process one can comment the line HideCursor() in order to navigate outside the experiment screen (set_screen.m line 48).

8.2 Arduino code

```
int byteRead;
void setup() {
// Turn the Serial Protocol ON
  Serial.begin(9600);
}
void loop() {
   /*  check if data has been sent from the computer: */
  if (Serial.available()) {
    /* read the most recent byte */
    byteRead = Serial.read();
    /*ECHO the value that was read, back to the serial port. */
    Serial.println(byteRead);
    Serial.print(" The Arduino received: ");
    Serial.println(byteRead, BIN);
    /*If received 1 (49 = ascii value of one) from serial port then set the output pin 7 to 5v */
    if (byteRead == 49)
      digitalWrite(7,HIGH);
    /*If received 0 (48 = ascii value of zero) from serial port then set the output pin 7 to 0v */
    if (byteRead == 48)
      digitalWrite(7,LOW);

  }
}
```

8.3 Experiment software

First we need to run the file RUNME.m:

```
function [] = RUNME()

   clear all;
    close all;
    global cfg;

    cfg = specs();
```

```
    if cfg.centeroutMode || cfg.gonogoMode
        RunCenterOut();
    else
        RunBasic();
    end

end
```

The function create config file according to the configurations defined in the specs.m file:

8.2.1 Specs.m

In this file we define all kind of preferences of the experiment. We'll show important sections of this file:

8.2.1.1. main operation mode

```
function cfg = specs()



    cfg.centeroutMode = true;
    cfg.gonogoMode = true;
    cfg.BRAIN_CONTROL = false;

    % for non-center out operation mode:
    cfg.TASKS_FILE_PATH = 'input/tasks.txt';
    cfg.CURSOR_ON = true;
    cfg.VISUAL_MODE = false;

    cfg.TTL_PORT =0;              %% 0- no ttl    1-arduino
    cfg.LOG_PREFIX = 'mouse_recording';
    cfg.BACKGROUND = 'black';           %% 'black' or image's path (for example: 'input/bg.jpg')
```

this section defines the main operation mode of the program.

If centeroutMode and gonogoMode are both false the program will need a path for taks list and we can decide if we want mouse recording (CURSOR_ON) .

When VISUAL_MODE is false the cue will be beeps and visual cue otherwise.

8.2.1.2. sync pulse interval

```
%% synce pulses intervals[s]:
    cfg.SYNC_INTERVAL_MIN = 4;
    cfg.SYNC_INTERVAL_MAX = 7;
```

Here we define how much time will go between sync intervals. The program get random time between those two parameters (arduinoConrol.m).

8.2.1.3.Next we have preferences for center out and gonogo modes includes: target and cursor size and color, target's position, rest times for different kind of stages in the experiment etc.

8.2.1.4 Beep configuration (specs.m)

```
%% beeps config:
%general settings:
cfg.BEEP_SAMPLE_RATE = 48000;
cfg.BEEP_NUM_CHANNELS = 2;

% time between two beeps
cfg.BEEP_PAUSE_MIN_S = 5;
cfg.BEEP_PAUSE_MAX_S = 5;

% length og beep
cfg.BEEP_LENGTH_MIN_S = 0.2;
cfg.BEEP_LENGHTH_MAX_S = 0.2;

% specific settings:
cfg.BEEP(1).FREQ = 500;
cfg.BEEP(1).REPETITIONS = 1;

cfg.BEEP(2).FREQ = 1500;
cfg.BEEP(2).REPETITIONS = 2;

cfg.BEEP(3).FREQ = [500 800 1200 1500];
cfg.BEEP(3).REPETITIONS = 3;

cfg.BEEP(4).FREQ = 900;
cfg.BEEP(4).REPETITIONS = 4;

cfg.BEEP(5).FREQ = 1900;
cfg.BEEP(5).REPETITIONS = 5;

cfg.BEEP(6).FREQ = 500;
cfg.BEEP(6).REPETITIONS = 1;

cfg.BEEP(7).FREQ = 800;
cfg.BEEP(7).REPETITIONS = 1;

cfg.BEEP(8).FREQ = 1100;
cfg.BEEP(8).REPETITIONS = 1;

cfg.BEEP(9).FREQ = 1400;
cfg.BEEP(9).REPETITIONS = 1;
```

here we define some general beep configuration: beep length, time between beeps. And specific configuration for pressing each number in keyboard. For example, pressing on '1' will produce single beep in 500Hz while pressing on '4' will produce series of 4 beeps with 5 seconds between beeps. In case of more than one frequency the program will produce the beeps according to its order in array and when end goes to the beginning. In a series of beeps (repetitons>1) we will play in the end of the sequence 3 short and close beep to inform the patient that the sequence is done.

8.2.1.5 Visual cues

```
%% visual mode        press 1 for single cue and 2 for sequence

cfg.VISUAL_CUE_SHAPE = 'FillRect';   %options: 'FillRect' 'FrameRect' 'FillOval'
'FrameOval'
cfg.VISUAL_CUE_SIZE = 150;
cfg.VISUAL_CUE_REST_COLOR = [1 1 1];
cfg.VISUAL_CUE_GO_COLOR = [0 1 0];
cfg.VISUAL_CUE_END_COLOR = [1 0 0];
cfg.VISUAL_CUE_END_S = 2;
cfg.VISUAL_CUE_LENGTH_MIN_S = 0.4;
cfg.VISUAL_CUE_LENGTH_MAX_S = 0.8;
cfg.VISUAL_CUE_REST_MIN_S = 2;
cfg.VISUAL_CUE_REST_MAX_S = 3;

cfg.VISUAL_CUE_REPETITIONS = 10;
```

here we define the visual cue size, shape and color and as we did with the beeps we defined that pressing on '1' will produce single visual cue while pressing on '2' will produce series of cues defined by the variable VISUAL_CUE_REPETITONS followed by cue in other color (VISUAL_CUE_END_COLOR) so the patient know that the sequence is over.

next the RUNME file decide which function to operate: RunCenterOut() or RunBasic() according to what we defined in 8.2.1.2.

both generally works the same way by running an infinite loop while every iteration we document mouse position (if necessary) and check if its inside the target (for center out and gonogo mode). Every 5 iterations we update the screen (we need to do it often since we need to draw cursor as it moves all the time).

Before running the loop those functions intilize all setting by calling to the functions:

Set_screen.m

Create_log_file.m

8.4 Brain control

As explained before there is a small tool that helps us demonstrate how brain control is worked – BRAINSIMULATOR.m.

It should be execute in separate process. It listens to keyboard queue and send the keyboard press to the socket defined.

You must run BRAINSIMULATOR.m in case you enable the option in specs.m: cfg.BRAIN_CONTROL=true.

8.5 Audio Record Software

Sound recorder must be executed in separate process as well.

the audio recorder as 3 main moduls (as shown in the gui presented in Figure 9):

1. recording audio (by pressing on ‘start’) – start record audio and save the data to a temporary file every X seconds as defined in the beginning of the file ‘AudioRecordMain.m’ using the function ‘SaveAudioToDisk’. While the program is recording the REC sign should blink. If it stopped blinking the meaning is something went wrong and the record is no longer running.
2. stop record (by pressin on ‘stop and save’)- stop the record, read all temp file, save the audio file and delete temporary file.
3. Recover (by pressing on ‘restore’)- in case of system failure we can’t press on stop and save. When restore the program will ask the user to choose the audio folder that we want to recover and if the temporary files are still there we will be able to convert it to sound file.

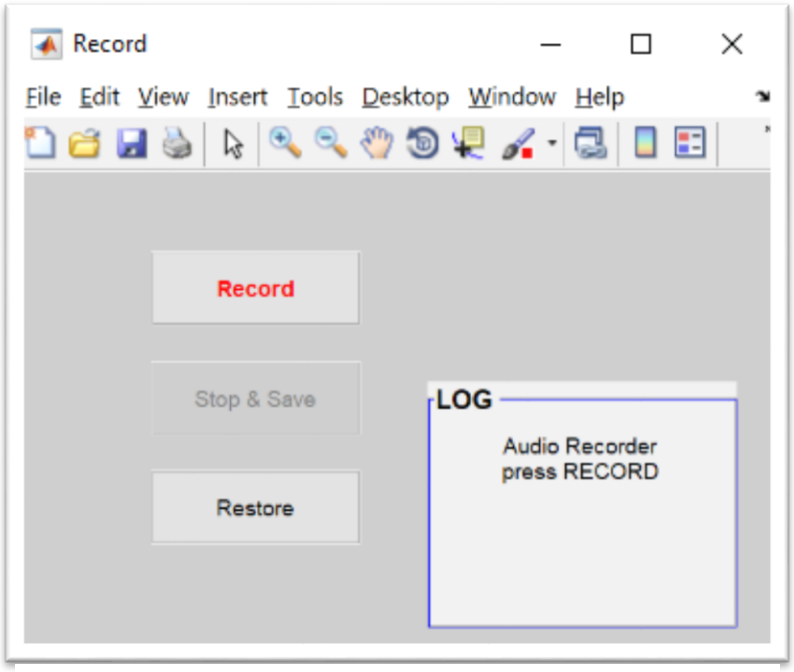

FIGURE 9-AUDIO RECORD GUI