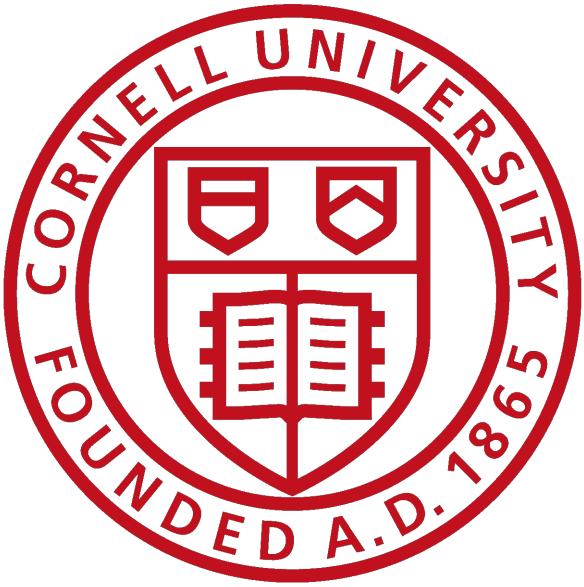
# HEART BRAIN CLOESD-LOOP MACHINE

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering




**Member of Team** : Naiqing Zhou (nz248) Yiting Wang (yw883) Hao Wu(hw622)

Project Advisor: Prof. Bruce Robert Land

MEng outside Advisor: Yu Hao

Date: May 18th, 2018

# Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

**Project Title:** Heart Brain Closed-Loop Machine

**Author:** Naiqing Zhou (nz248) Yiting Wang (yw883) Hao Wu(hw622)

**Abstract:**

The purpose of the project is to develop a Heart Brain Computer Interface system for research work in investigating people's emotion responses to the feedback factors such as color, frequency and sound. We have developed a heart machine interface that can perform two different operating modes, blinking LEDs in desired frequency/color pattern and following subjects' heartbeat. Based on our testing results we found out that LED blinking frequency and background music can influence people's heart rate. While we can't find a strong relationship between two subjects who look at each other's LED blinking in peak frequency, we believed more testing results will help to test our hypothesis that heart rate of two subjects can be synchronized.

Contribution：

Yiting Wang is mainly responsible for Arduino that control LED lighting pattern. Yiting also handles the communication between use interface module and Arduino. Her solution of the direct control on Arduino greatly helps to solve the timing problem of MATLAB. Yiting also solves the problem that users are influencing each other in certain operating modes by using timer objects on two users to simulate a multi-threads applications.

Hao Wu is mainly responsible for developing our Pan Tompkins Peak Detection Algorithm and build our user interface module. Hao came up with an adaptive thresholds technique to increase the peak value to enhance the accuracy of peak detection. The user interface module he built is helpful to the testing procedure and meet the design specification of our client. Hao also took initiative in the testing part and calculate the correlation between heart waveform.

Naiqing Zhou is responsible for system integration and optimization. His solution of modifying data attribution of the line object to new display variable within a loop helps to reduce display processing time and enhance efficiency. Naiqing coordinates the project work, keeps track of the project progress and record details for testing procedure .

**Executive Summary**

Human being's emotional state can be reflected by its heart and brain data under different conditions. The feedback factors that influence users' emotional state can be figured out with the help of a heart-brain computer interface system.

The aim of the project is to develop a heart machine interface to help the research work in investigating people's emotion responses to the feedback factors such as color, frequency and sound. For this purpose, our heart machine interface has a OpenBCI Bio-sensing device to collect heart data, which is fed to computer via Bluetooth. The collected heart data is analyzed for peak detection based on Pan Tompkins Algorithm and is plotted in real-time with MATLAB. We also design a user interface module in MATLAB that can switch between two operating modes, following heart beat and blinking in desired frequency, respectively. To control the LEDs color pattern and blink frequency, experimenter will need to use the GUI to send commands to Arduino. Our system can support two users simultaneously without interventions and data will be recorded into a file.

Our experiment results indicate that stimuli such as changing lighting frequency or background music does effect heart rate. We can't find a strong relationship between two subjects who look at each other's LED blinking in peak frequency. But we believed more testing results are required to test the hypothesis that heart rate of two subjects can be synchronized.

**Table of contents**

## 1. Introduction

Recent studies have shown that electroencephalogram(EEG） and electrocardiogram (ECG) are closely related to human's emotional state. Prior research mostly focuses on how environmental factors influence neural activities and determine emotional state based on the output of alpha, beta, theta and gamma bands**.** (see reference 1)The brain computer interface devices we saw today are widely applied in early R&D stages that can't monitor user's emotional state while applying affective feedback stimuli at the same time. According to US food and Drug administration, they initiate a human factor program that encourages device manufactures in to consult for a review on human factors and usability validation. (see Reference 2) Therefore, the medical sector and the government are showing positive sign that they are anticipating the arrival of such devices. We believe that the combination of brain and heart data will be most accurate, effective practical in detecting changes in emotion and guide users towards desired emotional state.

We build a heart machine interface that gives subjects feedback visually and acoustically so that we can begin the testing and research on how feedback factors influence people's emotion responses.

## 2. Design Alternative

Our system can record heart data for peak detection, give two types of feedback based on experimenters' choice and help the research of human being's emotion responses to the feedback factors such as color, frequency and sound.

Our client expects us to focus on the development of heart machine interface because the brain machine interface will be developed by her. We are also required to use OpenBCI Ganglion bio-sensing device that is provided by our client for recording data. MATLAB will be used perform peak detection, plot heart data and design a user interface module that can support two users. To enable the LEDs to follow heartbeat and blink in selected frequency/color, we decide to use microcontroller to control the LEDs.

*Switch to Arduino*

We originally consider using our on-board microcontroller of OpenBCI Ganglion to control LEDs. To avoid software issues on OpenBCI Ganglion board, we decided that it is more practical to use OpenBCI Ganglion to collect heart data and Arduino to drive the LEDs.

*Serial Communication between MATLAB and Arduino*

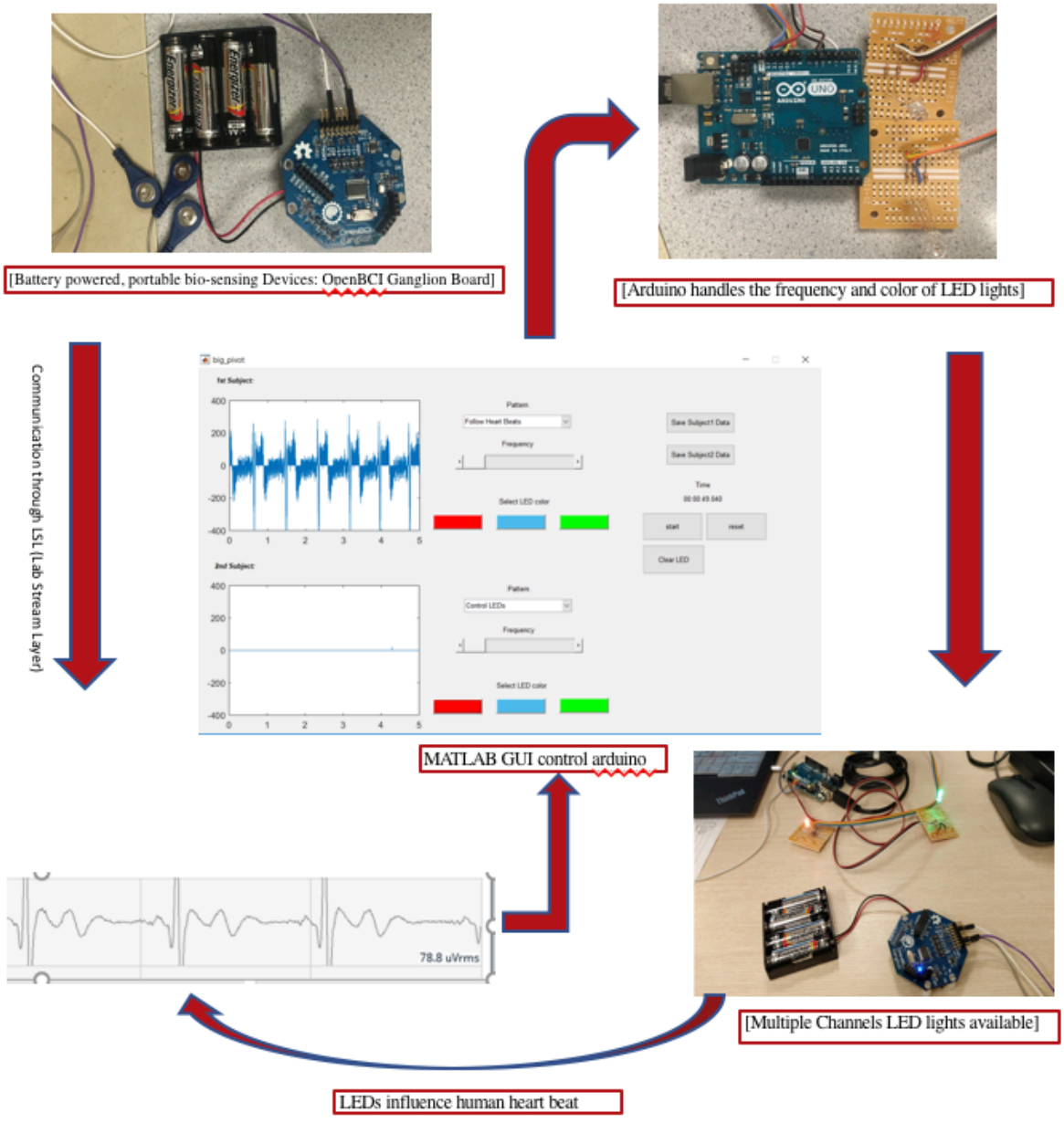
The method we initially used to handle communication between MATLAB and Arduino is serial communication. We use "arduino=serial('COM4','BaudRate',9600)" to set up the Arduino in MATLAB before using "fprintf" to send a string to Arduino. We directly use "fopen" and "fclose" to create serial port object but it didn't work well because of the timing problem on Arduino. Specifically, even if one LED is always on without blinking and the other is blinking following subject's heart rate, the first LED will see changes in brightness. This is because, a string is send to the serial port by MATLAB when a peak is detected. Then the parsing of the string will renew all the flag variables in the Arduino and "serial read" in Arduino needs longer processing time, which increases the delay in setting LED pins. To solve this, we use packages in MATLAB to directly control the IO pins of Arduino.

## 3. Final Design

### Top-Level Design

First of all, heart data will be first collected by our bio-sensor. Then it will be sent to computer to perform peak detection and will be plotted in our user interface designed in MATLAB. The experimenter will be able to select any of the two modes by sending commands to Arduino, which is used to control the LEDs lighting pattern. The figure below (see Figure 1) guides you through our design step by step from data collection of bio-sensor, peak detection, plot and GUI designed with

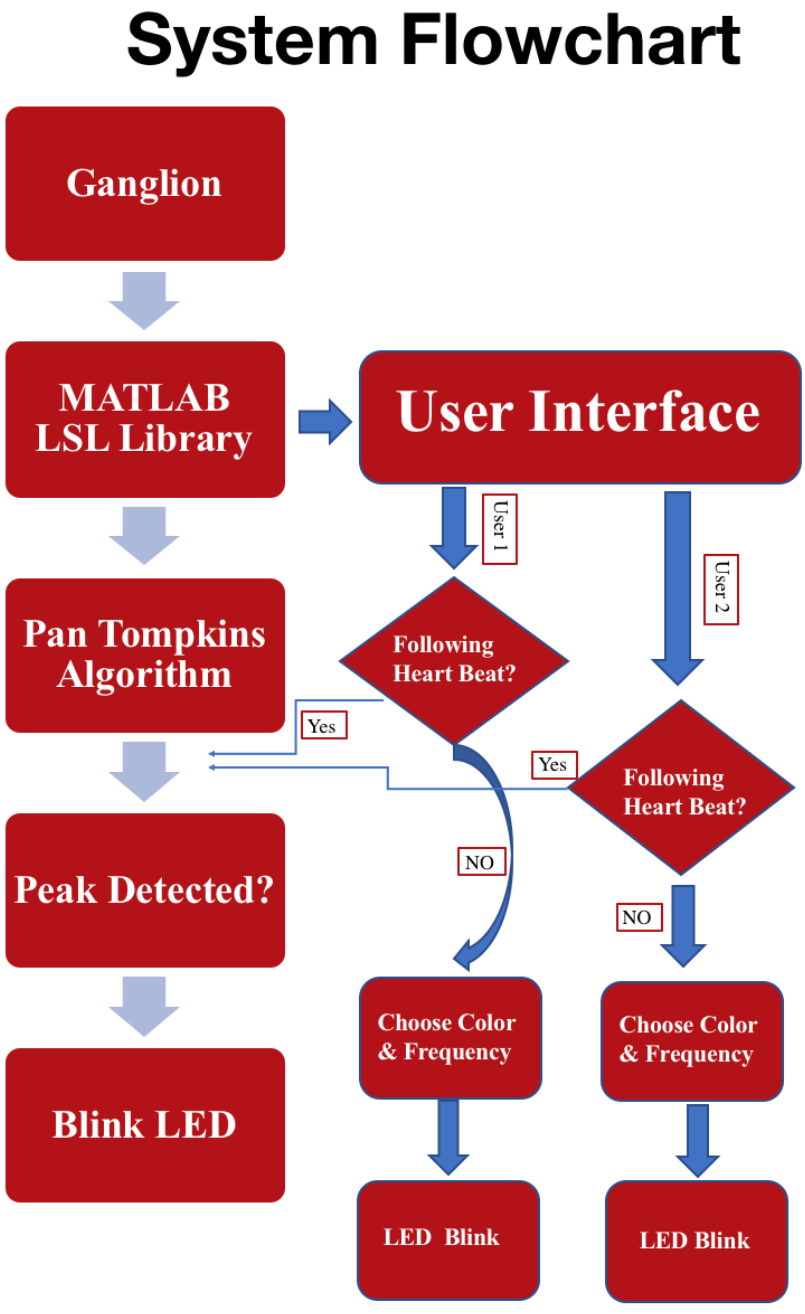
MATLAB, LEDs controlled by Arduino, to our final integration of each components. Each of the components will be explained in detail.




**[Figure 1: Final integration]**

The system flowchart below (see Figure 2)basically build a decision tree. It starts with our Ganglion bio-sensing device. In the second step, Lab streaming layer (LSL) synchronizes streaming data for live analysis or recording. Then the computer will perform Pan Tompkins

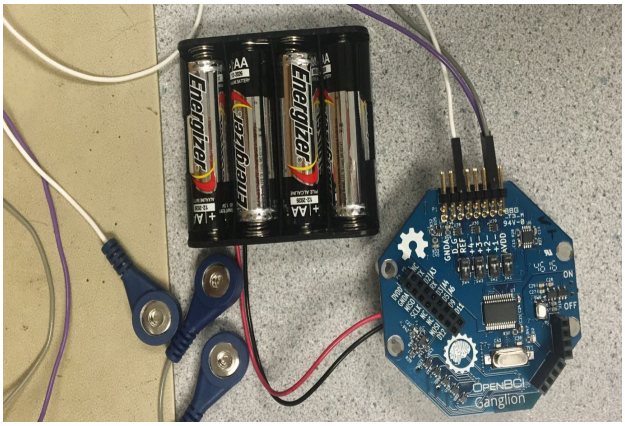
peak detection on the input heart data and display heart waveform on the GUI. The next step is to explain the logic design of the feedback system. If “following heart beat ” mode is selected, the collected heart data will be analyzed for peak detection and enable LEDs to blink with heart beat. Otherwise, the color/frequency of LEDs can be selected to achieve the desired feedback.




**[Figure 2: Heart Machine system Flowchart]]**

### 3.1 OpenBCI Ganglion Bio-Sensor

The OpenBCI Ganglion (Figure 3 ) is used as our bio-sensing device. It has 4 high-impedance differential inputs, a driven ground (DRL), a positive voltage supply (Vdd), and a negative voltage supply (Vss). In our case, the inputs are used as individual differential inputs for measuring ECG Data with sampling rate at 200Hz. The device is capable of sending collected heart data through Bluetooth.



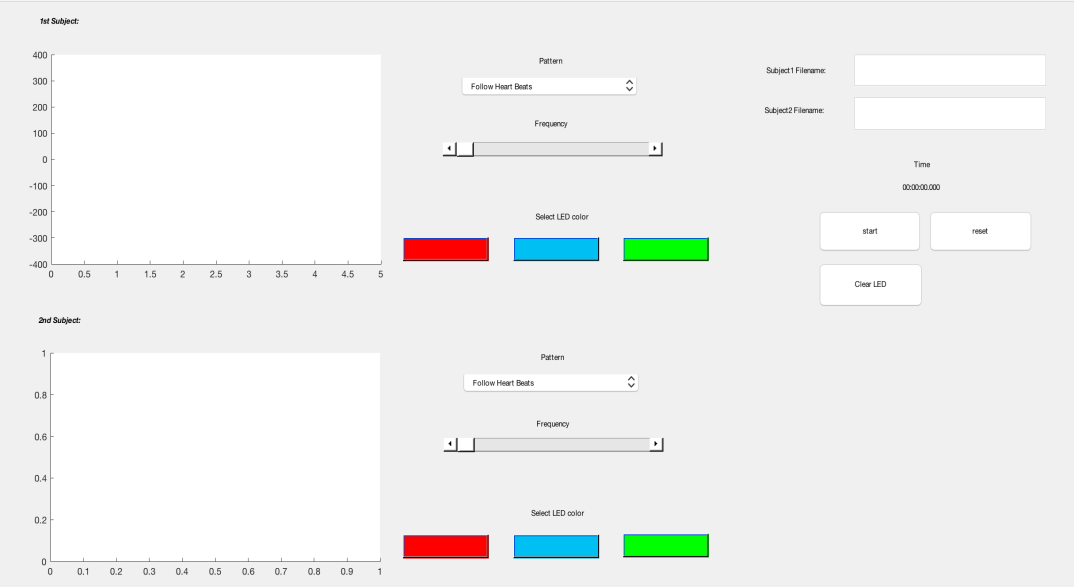

[Figure 3: Bio-sensor]

#### 3.1.1 Lab Streaming Layer

Lab streaming layer (LSL) is a system for synchronizing streaming data for live analysis or recording. (see reference 3 )The way we make LSL work is to choose the correct data type and paired device name.

### 3.2 GUI Design with MATLAB

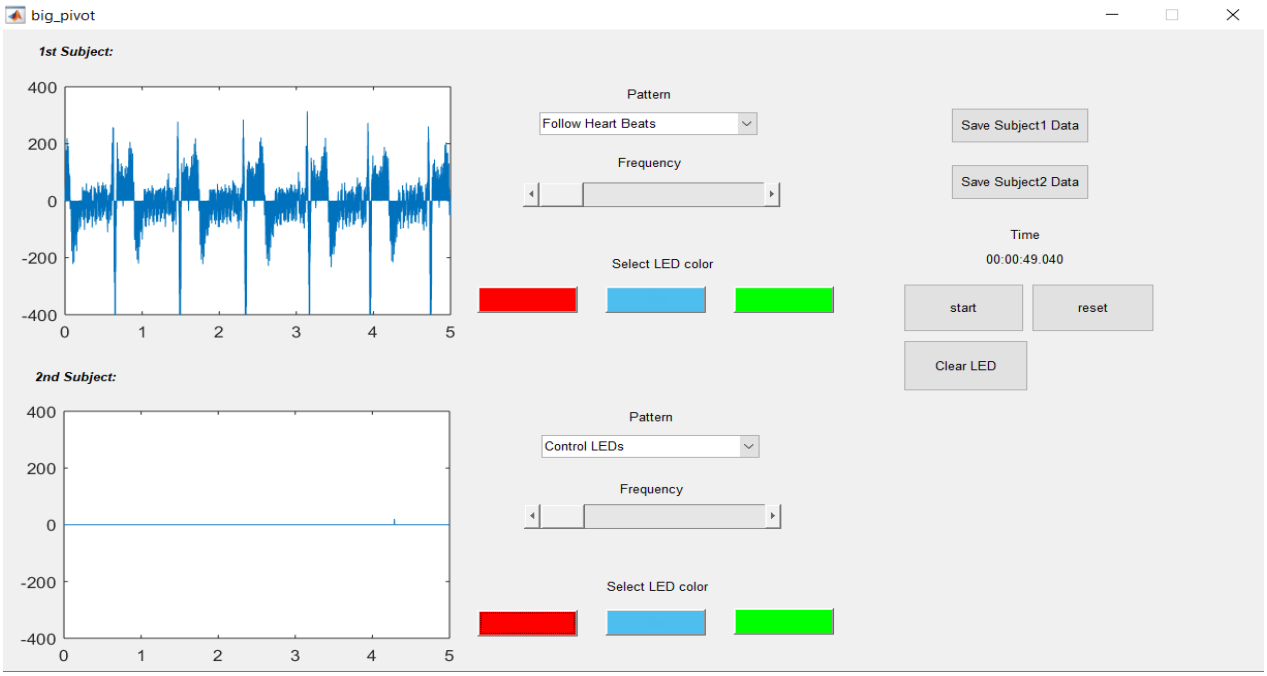
We design a graphical user interface (see Figure 2) that can support two users simultaneously. The user interface module can execute the selected LEDs modes by sending commands to Arduino and achieve desired pattern. We build our MATLAB application under the framework of MATLAB GUIDE, which is a callback type GUI builder. It combines all our application functions into callback functions in MATLAB and can be triggered when a UI event happens.




[Figure 2: User interface Design with MATLAB]

### 3.2.1 Draw Heart Waveform

The way to draw heart waveform (see Figure 3) in MATLAB GUI is realized by the scrolling display of data arrays that are within a certain range. The range is determined by two pointers, head (points to first node) and tail (points to last node). An example that shows the heart waveform is as below:




**[Figure 3: heart waveform plot]**

### 3.2.2 Lists of MATLAB Objects

Our GUI contains two plot axes, eleven pushbuttons, two sliders, two pop-up menus and three fixed text blocks. Each of them is a MATLAB object with a callback function to react with other objects' events. The utility of each object is here:

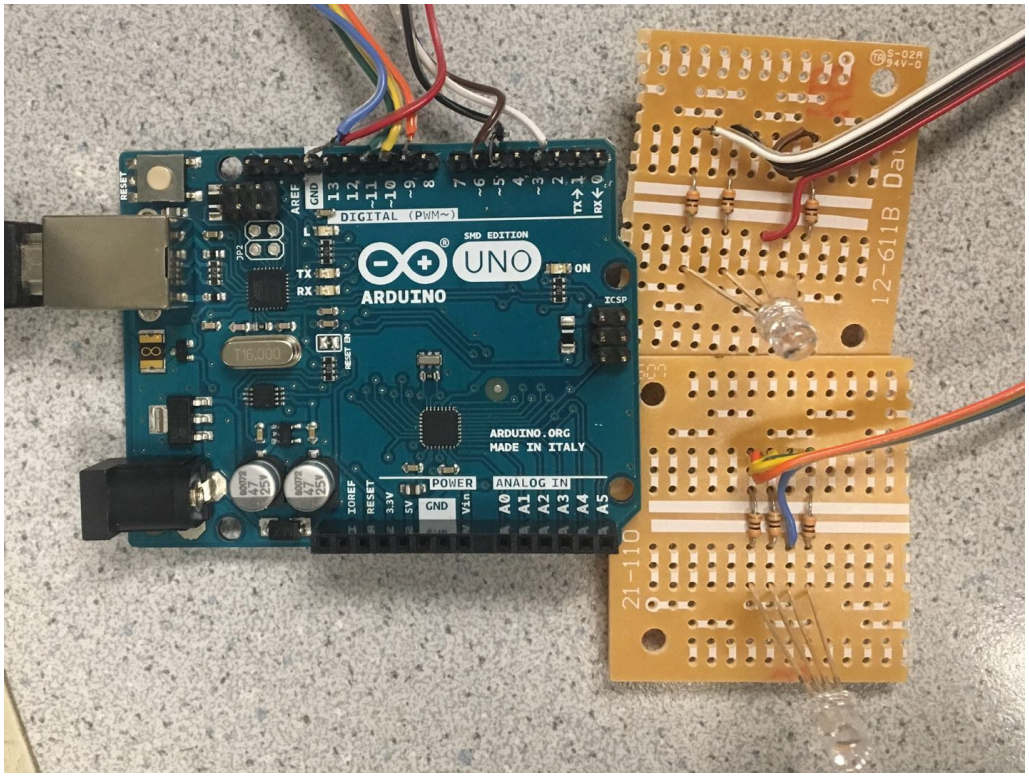
- Axes: displaying the ECG signals getting from Ganglion board, two with two subjects.
- Pushbutton: six of them used for LED color selecting, a start button, a stop button, a clear button of LED status and two buttons for saving subjects data into local files.
- Pop-up menu: used for selecting working mode of our application – the control LED mode and LED following ECG peak mode.
- Fixed text blocks: used for displaying which axes belongs to which subjects and the elapsed experiment time.

## 3.3 Pan Tompkins Peak Detection Algorithm

Our algorithm is implemented on Pan Tompkins Algorithm by using filters to produce high peak value and applying adaptive thresholding method to avoid missing peaks.(see reference 4) We combine Pan-Tompkins algorithm with adaptive thresholding method to make sure we accurately detect each peak. Moving threshold is a necessity to avoid missing peaks. The threshold changes every 5s. The core of PT algorithm is using filters to produce a very high value of each peak.

### 3.4 Arduino that drives LED

The way that Arduino (see Figure 4) was initiated in MATLAB is that, MATLAB will pre-configure Arduino with its supporting packages.(see reference 5) The packages will be downloaded to Arduino from the server to enable the control of Arduino from MATLAB.




**[Figure 4: Arduino drives LED]**

**Low-Level Design**

### 3.5. 1 User Interface Flow Diagram

Since MATLAB has an efficient data structure – handles, for objects sharing variables globally, we can push every variable into the GUI object's handles. That is to say, once the GUI is initiated, the variables can be accessed by any sub-objects within the GUI. We use GUIDE to build our GUI instead of building UI-controls objects and put them together in a while loop. The GUI uses callbacks to

respond to events, which can save more resources for us to do the computing on real-time data analysis, instead of checking on each components' status.

Now let's describe how our GUI works. The main callback function is integrated with the "START" pushbutton, which do the following things when clicked:

- Reset elapsed experiment time to "2:00:000", which is two minutes according to our client requirement.
- Extract axes object from handles, prepare for the displaying on axes.
- Connect to LSL broadcast station, prepare to receive data from LSL inlet.
- Create local variables used for Pan-Tompkins Algorithm.
- Start the while loop to receive LSL broadcasted data, check whether "STOP" button pushed, execute Pan-Tompkins Algorithm if application works in follow ECG peak mode, and plot the data onto axes, otherwise just plot the data onto the axes object.
- Calculate the elapsed time and display the remaining time.
- When the time used out, push data into handles.

To exit the callback function, function should be interruptible by another callback function. Adding a pause is a way for a callback function to be interrupted by another callback function. The pause should not be longer than 0.1 millisecond, otherwise it will cause noticeable delay in the total time of data receiving and processing.

### 3.5.2 Timer-object enable two-way communication

We solved the problem that users are influencing each other in certain operating modes combination. We found that the loss of connection between MATLAB and Arduino could

happen unless we use timer objects on two users to simulate a multi-threads applications in order to fix it. When one subject is in the control mode, another subject is in the following mode, the timing of triggering the writing operation from MATLAB to Arduino is decided not only by the fixed blinking rate, but also by the detection of a signal peak. If the peak is detected the same time as the fixed blinking rate is written to Arduino, the loss in connection of the device will happen. So we use MATLAB timer objects to schedule different LEDs blinking.

### 3.5.3 Optimization and improvement

Figure display speed-up: We identify the main cause of delay to be the constant update of the whole data arrays and re-create the whole plot. By modifying data attribution of the line object to new display variables within the loop and setting “handle y” data, we successfully speed up the figure display process. When figure object is created, its attributes can be accessed by the figure’s handles. So we came up with this way to access the display data attributes without plotting over and over again, which reduces processing time and enhances efficiency.

## Data and Testing

Since the system is built for our client to do experiments on subjects for biomedical data collection and analysis, we need to test the system and do some trial experiments to verify the reliance of our system. To do experiments on human subjects, we finished the Human Subjects Research Training (IRB) under the guidance of our outside advisor and obtained allowance to do human body tests on our subjects.

The experiment is designed to collect ECG data of subjects in the following procedure:

- Subjects in normal status (relaxed)

- Subjects looking red LED blinking in 5 Hz
- Subjects looking red LED blinking in 15 Hz
- Subjects under their favorite music
- Subjects looking red LED blinking following their own ECG peaks
- Subjects looking red LED blinking following another subject' ECG peaks

Every testing step is two minutes long, and the testing room is settled to be quiet and like daily working places. The normal status data will be used as a general basis compared with other testing status. The favorite music status is for our interests about whether music can play an effect on ECG data but can also make our subjects relaxed and have a preparation on the following experiments. We do not decide to apply blue LED and green LED experiments right now, because these colors have a relatively irritative feeling on our subjects, so we move the different color parts in the future real experiments, after LEDs can be shadowed by a ground glass or something with the same effect. During our trial experiments, our system meets our expectation and function without the conflict between two users interfere and correctly record our data into csv files locally. The data is presented in the box plot below:

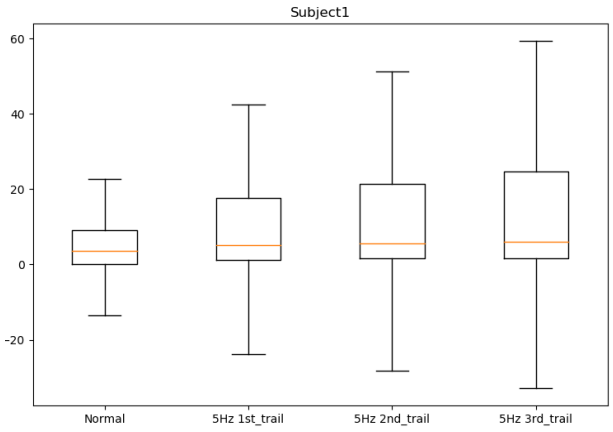



Figure 5.1: Subject1 looking red LED with 5 Hz

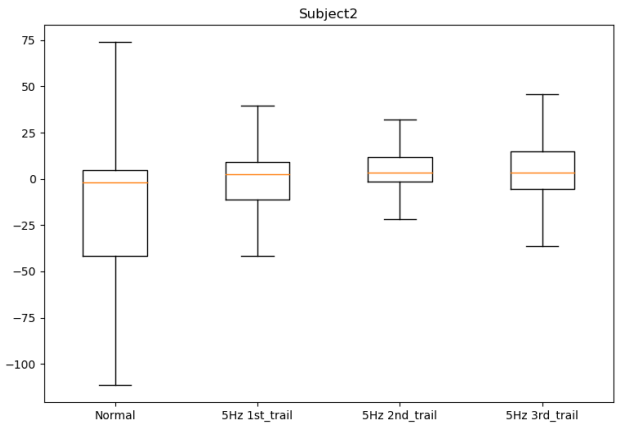



Figure 5.2: Subject2 looking red LED with 5 Hz

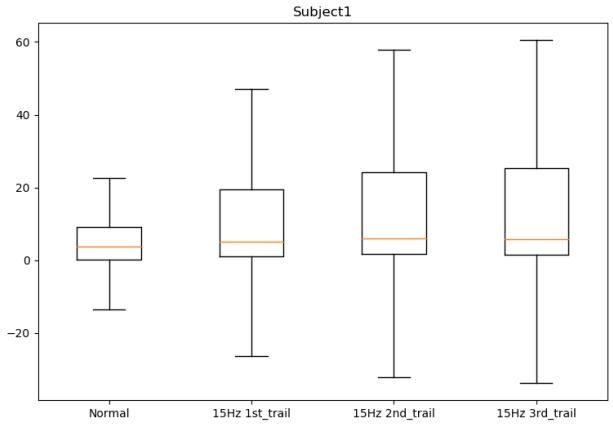



Figure 6.1: Subject1 looking red LED with 15 Hz

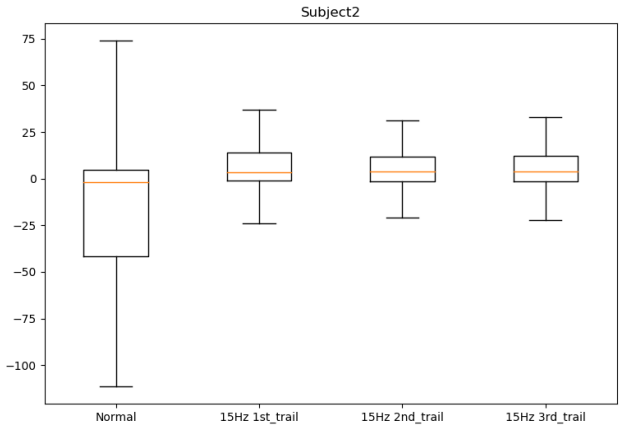



Figure 6.2: Subject2 looking red LED with 15 Hz

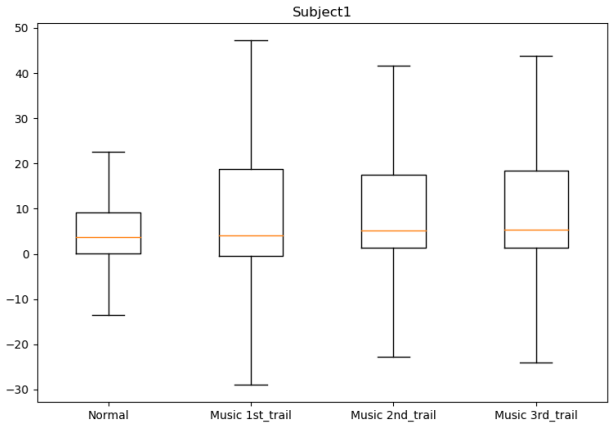



Figure 7.1: Subject1 under music

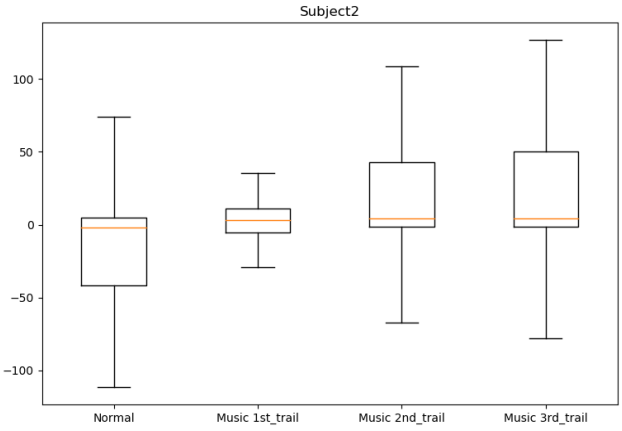



Figure 7.2: Subject2 under music

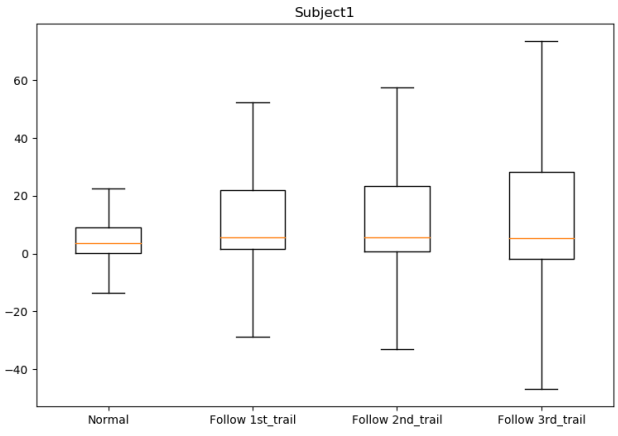



Figure 8.1: Subject1 - red LED following own ECG peak

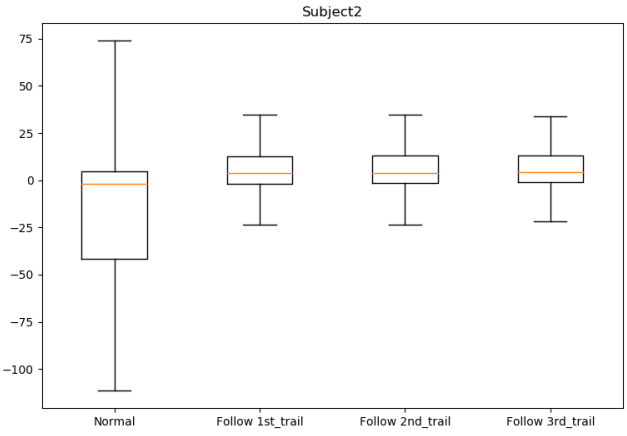



Figure 8.2: Subject2 - red LED following own ECG peak

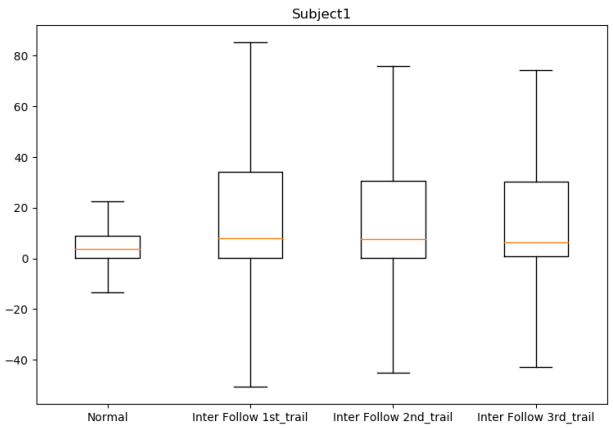



Figure 9.1: Subject1 - red LED following Subject2 ECG peak

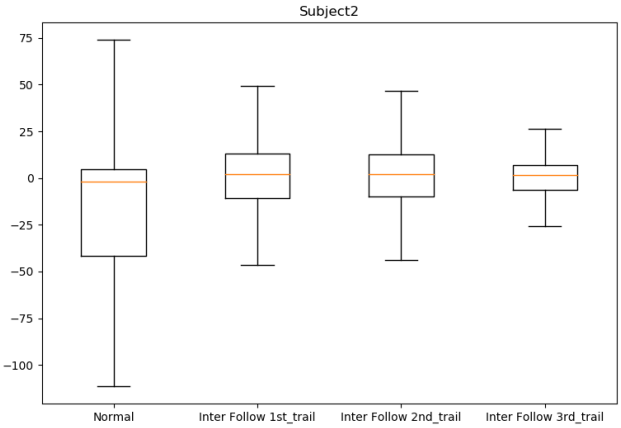



Figure 9.2: Subject2 - red LED following Subject1 ECG peak

The box plot was used as a standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum. From all these plots, the ECG signals of different controlled environment are different from our subjects' normal status. According to existing research conclusion from our client, the bigger range of the box plot presents a relatively relaxed emotion status, and the smaller range shows a relatively nervous emotion status reversely. So, we can see that our subject1 is a nervous person under normal status, and all these changes in the environment can make this subject become relatively relaxed. While on the other side, our subject2 is a relatively relaxed person in normal status, and the LED blinking seems to make this subject more nervous, but the favorite music can provide a release for this subject.

Besides what we can get from above, we are also interested in whether the two subjects looking at each other's peak blinking LED can bring a synchronization on two subjects' heart data. Here we use the correlation of two dataset to present whether these two serials have a relativity, we obtain the correlation coefficient of subject1's ECG data and subject2's of the inter following test is 0.0658, 0.0186 and -0.1574. In this phase, we just calculate the correlation of the waveform itself but we haven't calculated the correlation of heart rate. We believed that waveform is not likely to be correlated based on current testing results and the heart rate are more likely to be correlated. More

testing results are required to verify the assumptions. According to regular correlation analysis criterion, none of these coefficients can state a strong relationship between the subjects inter following data. Seeing this result, we may say that the synchronization of two subjects heart beats does not happen, but this may just because subject1 and subject2 has inborn differences in heart rate, and the truth behind biomedical data is something related to biology, which we may not analyze in a correct way. So, we still hold our position in this synchronization problem and wait for more testing data in the future.

[Figure 11: human body test by group member]

**Future Work**

Our future work includes the implementation with brain data by making simple modification on the second channel of our heart interface. We are proposing a portable commercial 4- sensor electroencephalograph (EEG) headband with sampling rate at 220Hz. It can be added to our existing electrocardiograph (ECG) sensing system. We can use a LED strip, vibration (can be

implemented on a bracelet), sound or graphical interface to represent the feedback stimuli. The combination of ECG and EEG devices will be used as an interface tool for measuring emotional status and generating stimuli to influence users in office setting. At that point, we will be able to further develop our system as feedforward interface to an artificial intelligent machine that will provide optimal stimuli to help drive individuals towards a desired emotional state. We expect to see the incorporation of machine learning algorithm. In other words, our system will be trained with different heartbeat patterns and learn to identify users' emotional status. Specifically, our logged data file records cardiac data and neural data that can be used for analysis. These data can be used to determine users' emotional status. In a later phase, we can expect the device to early detect negative emotions, intervene simultaneously and apply stimuli for desired heartbeat rate. Meanwhile, we expect to see the removal of Arduino to make our device more compact and portable for office setting.

**Reference:**


1. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3680185/

2. https://www.fda.gov/downloads/medicaldevices/deviceregulationandguidance/humanfactors/ucm320905.pdf

3. http://docs.openbci.com/3rd%20Party%20Software/04-LSL

4. http://www.robots.ox.ac.uk/~gari/teaching/cdt/A3/readings/ECG/Pan+Tompkins.pdf

5. https://www.mathworks.com/help/supportpkg/arduino/examples/communicating-with-arduino-hardware.html

6. https://www.mathworks.com/help/supportpkg/arduinoio/ug/connection-and-communication-issues.html

7. https://www.mathworks.com/help/supportpkg/arduinoio/index.html

## Appendix

```
function varargout = try1(varargin)
% TRY1 MATLAB code for try1.fig
%      TRY1, by itself, creates a new TRY1 or raises the existing
%      singleton*.
%
%      H = TRY1 returns the handle to a new TRY1 or the handle to
%      the existing singleton*.
%
%      TRY1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TRY1.M with the given input arguments.
%
%      TRY1('Property','Value',...) creates a new TRY1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before try1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to try1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help try1

% Last Modified by GUIDE v2.5 07-Apr-2018 20:53:35

% Begin initialization code - DO NOT EDIT gui_Singleton =
1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @try1_OpeningFcn, ...
                   'gui_OutputFcn',  @try1_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []); if
nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
end

% End initialization code - DO NOT EDIT

% --- Executes just before try1 is made visible.
function try1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to try1 (see VARARGIN)
```

```
% Choose default command line output for try1 handles.output =
hObject;

% The slider for changing frequency
set(handles.slider3, 'Min', 0);
set(handles.slider3, 'Max', 20);
set(handles.slider3, 'Value', 0); set(handles.slider3,
'SliderStep',[5/20,5/20]);

set(handles.slider4, 'Min', 0);
set(handles.slider4, 'Max', 20);
set(handles.slider4, 'Value', 0); set(handles.slider4,
'SliderStep',[5/20,5/20]);

% Global variables, may be optimized by pushed into handles. global TIME;
global lib; global
result; global inlet;
lib = lsl_loadlib(); result = {};

handles.follow1 = false;
handles.control1 = false;
handles.follow2 = false;
handles.control2 = false;
% UIWAIT makes try1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% Update handles structure
guidata(hObject, handles);

handles.data = '0 a 0 a';
guidata(hObject, handles);
%arduino
clear arduino;
if ~isempty(instrfind)
        fclose(instrfind);
                delete(instrfind);
end
if ~isempty(seriallist) handles.arduino=serial('COM3','BaudRate',9600);
      fopen(handles.arduino); guidata(hObject,handles);
else
      disp('Please check Arduino connection');
end

end


% --- Outputs from this function are returned to the command line. function varargout =
try1_OutputFcn(hObject, eventdata, handles)
% varargout       cell array for returning output args (see VARARGOUT);
% hObject         handle to figure
% eventdata       reserved - to be defined in a future version of MATLAB
```

```
% handles      structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure varargout{1} =
handles.output;
end



% --- Executes on selection change in popupmenu2. function
popupmenu2_Callback(hObject, eventdata,handles)
% hObject      handle to popupmenu2 (see GCBO)
% eventdata    reserved - to be defined in a future version ofMATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returnspopupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item frompopupmenu2 str = get(hObject, 'String');
val = get(hObject, 'Value'); switch
str{val}
    case 'Follow Heart Beats' handles.follow1
        = true; handles.control1 = false;
        disp('Follow subject1 heart beats...'); guidata(hObject,
        handles);
    case 'Control LEDs' handles.control1 =
        true; handles.follow1 = false;
        disp('Control subject1 LEDs...'); guidata(hObject,
        handles);
end
end



% --- Executes during object creation, after setting all properties. function
popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu2 (see GCBO)
% eventdata    reserved - to be defined in a future version ofMATLAB
% handles      empty - handles not created until after all CreateFcnscalled

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end



% --- Executes on slider movement. first person's frequency function
slider3_Callback(hObject, eventdata, handles)
% hObject      handle to slider3 (see GCBO)
% eventdata    reserved - to be defined in a future version ofMATLAB
% handles      structure with handles and user data (see GUIDATA)\
% Hints: get(hObject,'Value') returns position of slider
```

```
%               get(hObject,'Min') and get(hObject,'Max') to determine rangeof slider
sliderValue=get(handles.slider3, 'Value'); herz =
num2str(sliderValue) ; set(handles.text6,'String',herz);
if (handles.control1 == true) if(sliderValue >= 0 &&
        sliderValue <5)
                fre = handles.data; fre(3) = 'a';
                handles.data = fre;
                guidata(hObject,handles);
                fprintf(handles.arduino,'%s',handles.data); % send answer variable content to arduino
        end

        if(sliderValue >= 5 && sliderValue < 10) fre =
                handles.data;
                fre(3) = 'b'; handles.data = fre;
                guidata(hObject,handles);
                fprintf(handles.arduino,'%s',handles.data);
        end

        if(sliderValue >= 10 && sliderValue < 15) fre =
                handles.data;
                fre(3) = 'c'; handles.data = fre;
                guidata(hObject,handles);
                fprintf(handles.arduino,'%s',handles.data);
        end

        if(sliderValue >= 15 && sliderValue < 20) fre =
                handles.data;
                fre(3) = 'd'; handles.data = fre;
                guidata(hObject,handles);
                fprintf(handles.arduino,'%s',handles.data);

        end

        if(sliderValue == 20 ) fre =
                handles.data; fre(3) = 'e';
                handles.data = fre;
                guidata(hObject,handles);
                fprintf(handles.arduino,'%s',handles.data);

        end
end
end

% --- Executes during object creation, after setting all properties. function slider3_CreateFcn(hObject,
eventdata, handles)
% hObject           handle to slider3 (see GCBO)
% eventdata         reserved - to be defined in a future version ofMATLAB
% handles           empty - handles not created until after all CreateFcnscalled
```

```
% Hint: slider controls usually have a light gray background. if
isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
end
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton1 (see GCBO)
% eventdata         reserved - to be defined in a future version ofMATLAB
% handles           structure with handles and user data (seeGUIDATA) end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton2 (see GCBO)
% eventdata         reserved - to be defined in a future version ofMATLAB
% handles           structure with handles and user data (seeGUIDATA) end

% --- Executes on button press in pushbutton4. person 1 red function
pushbutton4_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton4 (see GCBO)
% eventdata         reserved - to be defined in a future version ofMATLAB
% handles           structure with handles and user data (seeGUIDATA) color = handles.data;
color(1) = '1'; handles.data = color;
guidata(hObject,handles);
fprintf(handles.arduino,'%s',handles.data); end

% --- Executes on button press in pushbutton5.person 1 blue function
pushbutton5_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton5 (see GCBO)
% eventdata         reserved - to be defined in a future version ofMATLAB
% handles           structure with handles and user data (seeGUIDATA) color = handles.data;
color(1) = '2'; handles.data = color;
guidata(hObject,handles);
fprintf(handles.arduino,'%s',handles.data); end

% --- Executes on button press in pushbutton6.person 1 green function
pushbutton6_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton6 (see GCBO)
% eventdata         reserved - to be defined in a future version ofMATLAB
% handles           structure with handles and user data (seeGUIDATA) color = handles.data;
color(1) = '3'; handles.data = color;
guidata(hObject,handles);
fprintf(handles.arduino,'%s',handles.data); end
```

```
% --- Executes during object creation, after setting all properties. function text6_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called end

% --- Executes during object creation, after setting all properties. function text7_CreateFcn(hObject,
eventdata, handles)
% hObject    handle to text7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
%set(handles.text7,'String','00:00:000'); end


% --- Executes on button press in pushbutton7.
% --- Contains the real-time drawing of heart beat and peak detection. function
pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Global variables, can be optimazed by pushed into handles in the future. global lib;
global result; global
inlet; global TIME;
TIME = 60.0;
str = formatTimeFcn(TIME);
set(handles.text7,'String',str);
% Axes handles, used for drawing in axes of GUI. axesHandle1 =
handles.axes1;
axesHandle3 = handles.axes3;
handles.cameraCleared = 0;
guidata(hObject, handles);

% LSL connection, more details of LSL please google LSL-matlab. while isempty(result)
    result = lsl_resolve_byprop(lib, 'type', 'EEG');
end
inlet = lsl_inlet(result{1});

% Variables used for computing the peak, may be optimazed in the future. time_elap = 0:1/400:5;
time_total = 0:1/400:60;
low = zeros(2, length(time_total)); high = zeros(2,
length(time_total)); diff = zeros(2, length(time_total));
square = zeros(2, length(time_total));
threshold = zeros(2, length(time_total)); hb_disp = zeros(2,
length(time_elap)); hb_total = zeros(2, length(time_total));
lh1 = plot(axesHandle1, time_elap, hb_disp(1, :)); lh2 = plot(axesHandle3,
time_elap, hb_disp(2, :)); t = 1;
count = 1;
```

```
while TIME > 0
        % Tiny pause needed for responding to stop callback pause(0.0001);
        handles = guidata(hObject); set(axesHandle1, 'YLim',
        [-400 400]);
        set(axesHandle3, 'YLim', [-400 400]);

        % Check whether stopped
        if handles.cameraCleared == 1 break;
        end

        % The sample data at each iteration from Ganglion v =
        inlet.pull_sample();

        % Wipe out the noise if
        (abs(v(1)) > 20)
                hb_total(1, count) = v(1);
        else
                hb_total(1, count) = 0;
        end

        if (abs(v(2)) > 20) hb_total(2, count) = v(2);
        else
                hb_total(2, count) = 0;
        end


        % Pan-Tompkins Algorithm for peak extraction
        if ((handles.follow1 == true || handles.follow2) == true && count >= 4) low(:, count) = 0.0279*hb_total(:,
                count)+0.0557*hb_total(:,count-
1)...
                        +0.0279*hb_total(:, count-2)+1.4755*low(:, count-1)-0.5869*low(:,
count-2);

                high(:, count)=0.9846*low(:, count)-1.9691*low(:, count- 1)+0.9846*low(:, count-
2)...
                        +1.9689*high(:, count-1)-0.9694*high(:, count-2); diff(:, count)=0.25*high(:,
                count)+0.125*high(:, count-1)-
0.125*high(:, count-2)...
                        -0.25*high(:, count-3);
                square(:, count) = diff(:, count).^2;
                threshold(:, count) = max(square(:, count-3),square(:, count))/3; real_thresh =
                [max(threshold(1, :)), max(threshold(2, :))];
                if (handles.follow1 == true)
                        if (square(1, count)<=real_thresh(1))&&(square(1, count- 1)>=real_thresh(1))
                                fprintf('subject1: peak1\n');
                        else
                                fprintf('subject1: 0\n');
                        end
                 end


                if (handles.follow2 == true)
                        if (square(2, count)<=real_thresh(2))&&(square(2, count- 1)>=real_thresh(2))
                                fprintf('subject2: peak2\n');
                        else
                                fprintf('subject2: 0\n');
                        end
```

```
            end
        end

        % Display on the axes
        hb = hb_total(:, t: t + 2000); set(lh1, 'ydata',
        hb(1, :));
        set(lh2, 'ydata', hb(2, :)); count = count +
        1;
        if count > 2001 t = t +
            1;
        end

        % Display time
        TIME = TIME - 0.002;
        str = formatTimeFcn(TIME);
        set(handles.text7,'String',str);
end

filename1 = handles.filename1;
csvwrite(filename1, hb_total);

end

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton8 (see GCBO)
% eventdata         reserved - to be defined in a future version ofMATLAB
% handles           structure with handles and user data (seeGUIDATA) global TIME;
handles.cameraCleared       =       1;
guidata(hObject, handles); disp('stop');

TIME = 60.0;
set(handles.text7,'String','00:00:00.000'); end

function str = formatTimeFcn(float_time) float_time =
            abs(float_time); hrs = floor(float_time/3600);
            mins = floor(float_time/60 - 60*hrs); secs = float_time -
            60*(mins + 60*hrs); h = sprintf('%1.0f:',hrs);
            m = sprintf('%1.0f:',mins); s =
            sprintf('%1.3f',secs); if hrs < 10
                h = sprintf('0%1.0f:',hrs);
            end
            if mins < 10
                m = sprintf('0%1.0f:',mins);
            end
            if secs < 9.9995
                s = sprintf('0%1.3f',secs);
            end
            str = [h m s];
end


function figure1_CloseRequestFcn(hObject, eventdata, handles)
```

```
if ~isempty(seriallist)
        fclose(handles.arduino);
end delete(hObject);
end

function edit3_Callback(hObject, eventdata, handles)
% hObject           handle to edit3 (see GCBO)
% eventdata         reserved - to be defined in a future version of MATLAB
% handles           structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%               str2double(get(hObject,'String')) returns contents of edit3 asa double
filename1 = get(hObject, 'String') + ".csv"; handles.filename1 =
filename1; guidata(hObject, handles);
end

% --- Executes during object creation, after setting all properties. function edit3_CreateFcn(hObject,
eventdata, handles)
% hObject           handle to edit3 (see GCBO)
% eventdata         reserved - to be defined in a future version of MATLAB
% handles           empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%             See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end
end



function edit4_Callback(hObject, eventdata, handles)
% hObject           handle to edit4 (see GCBO)
% eventdata         reserved - to be defined in a future version of MATLAB
% handles           structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%               str2double(get(hObject,'String')) returns contents of edit4 asa double
end

% --- Executes during object creation, after setting all properties. function edit4_CreateFcn(hObject,
eventdata, handles)
% hObject           handle to edit4 (see GCBO)
% eventdata         reserved - to be defined in a future version of MATLAB
% handles           empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%             See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
        set(hObject,'BackgroundColor','white');
end
end


% --- Executes on button press in pushbutton9.person 2 red function
pushbutton9_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton9 (see GCBO)
% eventdata         reserved - to be defined in a future version of MATLAB
% handles           structure with handles and user data (see GUIDATA) color = handles.data;
color(5) = '1'; handles.data = color;
guidata(hObject,handles);
fprintf(handles.arduino,'%s',handles.data); end


% --- Executes on button press in pushbutton10.person 2 blue function
pushbutton10_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton10 (see GCBO)
% eventdata         reserved - to be defined in a future version of MATLAB
% handles           structure with handles and user data (see GUIDATA) color = handles.data;
color(5) = '2'; handles.data = color;
guidata(hObject,handles);
fprintf(handles.arduino,'%s',handles.data); end

% --- Executes on button press in pushbutton11.person 2 green function
pushbutton11_Callback(hObject, eventdata, handles)
% hObject           handle to pushbutton11 (see GCBO)
% eventdata         reserved - to be defined in a future version of MATLAB
% handles           structure with handles and user data (see GUIDATA) color = handles.data;
color(5) = '3'; handles.data = color;
guidata(hObject,handles);
fprintf(handles.arduino,'%s',handles.data); end

% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject           handle to slider4 (see GCBO)
% eventdata         reserved - to be defined in a future version of MATLAB
% handles           structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%               get(hObject,'Min') and get(hObject,'Max') to determine range of slider
sliderValue=get(handles.slider4, 'Value'); herz =
num2str(sliderValue) ; set(handles.text14,'String',herz);
if (handles.control2 == true) if(sliderValue >= 0 &&
        sliderValue < 5)
                fre = handles.data;
```

```
            fre(7) = 'a'; handles.data = fre;
            guidata(hObject,handles);
            fprintf(handles.arduino,'%s',handles.data); % send answer variable content to arduino
        end

        if(sliderValue >= 5 && sliderValue < 10) fre =
            handles.data;
            fre(7) = 'b'; handles.data = fre;
            guidata(hObject,handles);
            fprintf(handles.arduino,'%s',handles.data);
        end

        if(sliderValue >= 10 && sliderValue < 15) fre =
            handles.data;
            fre(7) = 'c'; handles.data = fre;
            guidata(hObject,handles);
            fprintf(handles.arduino,'%s',handles.data);
        end

        if(sliderValue >= 15 && sliderValue < 20) fre =
            handles.data;
            fre(7) = 'd'; handles.data = fre;
            guidata(hObject,handles);
            fprintf(handles.arduino,'%s',handles.data);

        end

        if(sliderValue == 20 ) fre =
            handles.data; fre(7) = 'e';
            handles.data = fre;
            guidata(hObject,handles);
            fprintf(handles.arduino,'%s',handles.data);

        end
    end
    end

    % --- Executes during object creation, after setting all properties. function slider4_CreateFcn(hObject,
    eventdata, handles)
    % hObject           handle to slider4 (see GCBO)
    % eventdata         reserved - to be defined in a future version of MATLAB
    % handles           empty - handles not created until after all CreateFcns called

    % Hint: slider controls usually have a light gray background. if
    isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
    end
    end
```

```
% --- Executes on selection change in popupmenu3. function
popupmenu3_Callback(hObject, eventdata,handles)
% hObject          handle to popupmenu3 (see GCBO)
% eventdata        reserved - to be defined in a future version ofMATLAB
% handles          structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returnspopupmenu3 contents as cell array
%              contents{get(hObject,'Value')} returns selected item frompopupmenu3 str = get(hObject, 'String');
val = get(hObject, 'Value'); switch
str{val}
      case 'Follow Heart Beats' handles.follow2
            = true; handles.control2 = false;
            disp('Follow subject2 heart beats...'); guidata(hObject,
            handles);
      case 'Control LEDs' handles.control2 =
            true; handles.follow2 = false;
            disp('Control subject2 LEDs...'); guidata(hObject,
            handles);
end
end

% --- Executes during object creation, after setting all properties. function
popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject          handle to popupmenu3 (see GCBO)
% eventdata        reserved - to be defined in a future version ofMATLAB
% handles          empty - handles not created until after all CreateFcnscalled

% Hint: popupmenu controls usually have a white background on Windows.
%            See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
end
end



% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject          handle to pushbutton12 (see GCBO)
% eventdata        reserved - to be defined in a future version ofMATLAB
% handles          structure with handles and user data (see GUIDATA)

ledStatus = '0 a 0 a'; handles.data =
ledStatus; guidata(hObject, handles);
fprintf(handles.arduino,'%s',handles.data);


end
```