SOFTWARE GUIDE

INTRACRANIAL PRESSURE WAVEFORM GENERATOR

May 6, 2019

Senior Design Team 3
Wright State University
Department of Biomedical, Industrial, and Human Factors Engineering

Contents

List of Figures

Listings

RASPBERRY PI SETUP

1. Hardware

Raspberry Pi 3
Pi Cobbler
Adafruit Digital Analog Converter (DAC)
32 GB or more microSD Card
Keyboard
Computer Mouse
Wireless HDMI

2. Software Installation

NOOBS Install

NOOBS was installed as the operating system. A microSD card with at least 32 GB of storage is required. The set up was as follows:
1. Insert microSD into computer
2. Go to `https://www.raspberrypi.org/downloads/noobs/` and download the NOOBS zip file
3. Extract the zip flie onto the microSD
4. Eject the microSD and insert into the Raspberry Pi 5. Power the Raspberry Pi on. The first screen that will appear select Raspbian and install
6. The Pi will now reboot and after it has power back on the Raspberry Pi is now ready for use

Set up: `https://www.youtube.com/watch?v=wvxCNQ5AYPg`

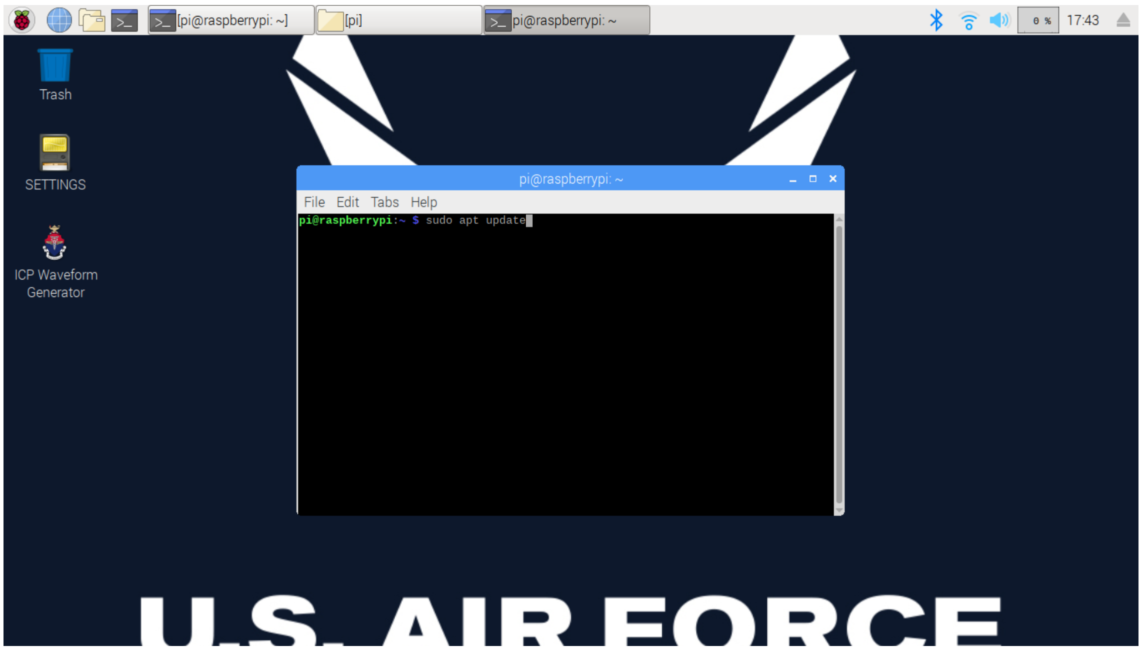
3. Login Credentials

The username is "pi" – all lowercase
The password is "ICP" – all capitalized letters

4. Package Installation

The process for installing any python packages should be done as follows from inside the terminal:

1. Run "sudo apt update" through the terminal before installing new software packages. This will ensure that all packages are the lastest version and avoid any versioning issues with newer installed packages. An example of interfacing with the terminal can be seen below.



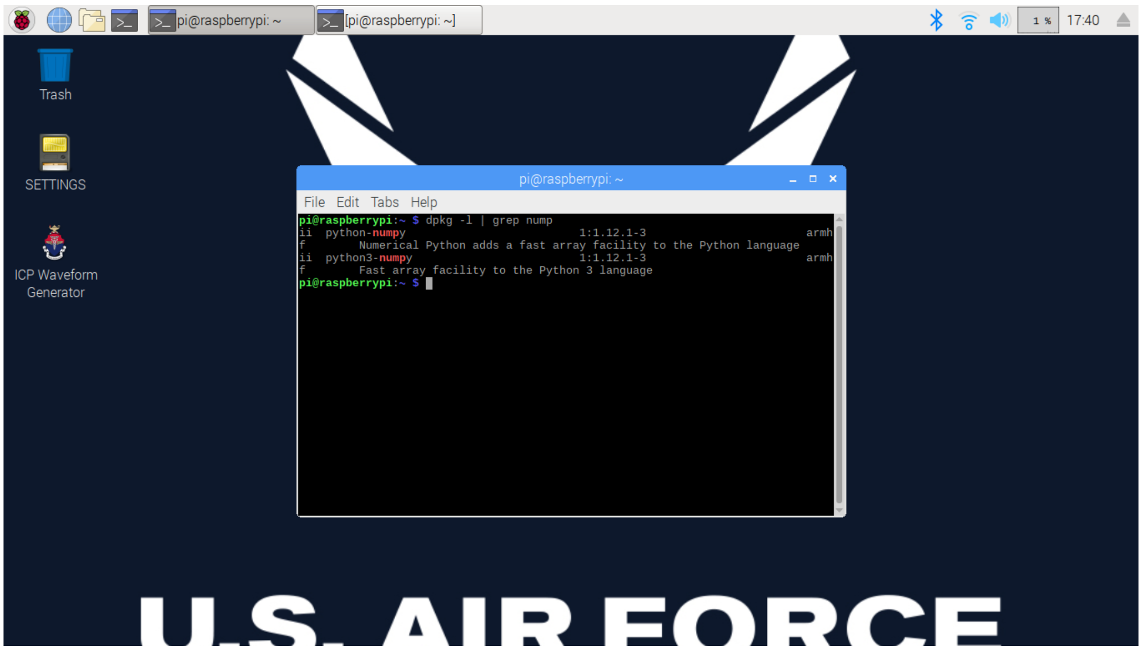
```
pi@raspberrypi:~ $ sudo apt update
```

Figure 1: Interfacing in the Terminal

2. Type "sudo pip3 install xxxxxx" where xxxxxx is the name of the package with which you wish to install.
3. This will not include all necessary packages so if one of them doesn't work using "sudo pip3 install" then it is acceptable to use "sudo apt-get install python3-xxxxxx".

Viewing installed packages

To check if a package is installed type into the terminal "dpkg -l | grep <keyword>", where <keyword> is (part of) the name of the package you are interested in finding. For example, searching "dpkg -l | grep nvidia" will list all installed packages with "nvidia" in the name or description. This method will not work for meta-packages or repositories, though it will work for most cases. You can check where a package is installed by typing "sudo dpkg -S packagename".



```
pi@raspberrypi:~ $ dpkg -l | grep nump
ii  python-numpy                         1:1.12.1-3                    armh
f        Numerical Python adds a fast array facility to the Python language
ii  python3-numpy                        1:1.12.1-3                    armh
f        Fast array facility to the Python 3 language
pi@raspberrypi:~ $
```

Figure 2: Checking for Python Packages

The packages that are required for our system are as follows:

1. Adafruit MCP4275
Used for the DAC chip. Install was set up using:
`https://learn.adafruit.com/mcp4725-12-bit-dac-with-raspberry-pi`

2. Guizero
Used for the creation of the Graphical User Interface (GUI). This package was chosen because it is compatible with the Raspberry Pi and easy to use as a first time GUI creation.

3. ScipPy
Contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks. This package is used for the calculations in our codes.

4. Numpy
Used for multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions. This package is used to convert the output from the MATLAB files into a 1-dimensional array that can be read into the DAC code.

5.I2C
A serial computer bus that is used for attaching integrated circuits to the Raspberry Pi via intra-board communication. This allowed our code on the Raspberry Pi to be send through the GPIO pins. Setup was done through:
`https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial/all`

6. GPIO
These are the pins that the connects the digital code output to the DAC hardware. This connection is performed via the Pi Cobbler. This software allows you to pick the pin the digital signal will be exported out of. Setup for this was done through:
`https://www.raspberrypi-spy.co.uk/2012/05/install-rpi-gpio-python-library/`

TROUBLESHOOTING

HDMI Will Not Display

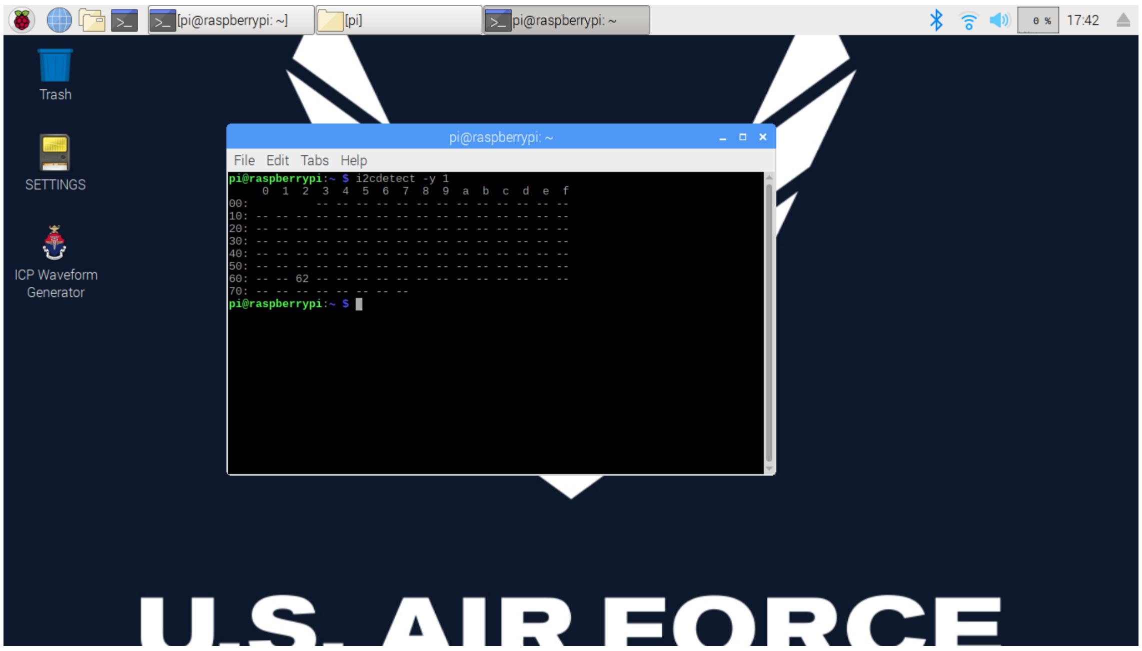
The Raspberry Pi must first be plugged in before the HDMI can be plugged in. Just unplug the Raspberry Pi and plug it back into the outlet. This tends to happen frequently with Raspberry Pis.

Code Will Not Execute Because Pi Cannot Discover I2C Bus Location

Chances are that the DAC chip has fried and needs to be replaced. If this is not the case, check to see which I2C location the Pi is trying to output the code by typing into the terminal "i2cdetect -y 1". There should be a number that appears in one of the locations as shown in the figure. If this does not occur, then the DAC most likely needs to be replaced. The figure below is what should display if your I2C is properly working.

Pi Will Not Turn On

Check to see what color is being displayed by the LED. If only red is showing, the SD card is either inserted incorrectly or corrupted. It should have a red LED with a green LED that intermittently blinks. If the SD card is corrupted, it should just be thrown away and a new SD card used.



```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- 62 -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

Figure 3: I2C

Board Goes Off Intermittently

This issue is when the Raspberry Pi will constantly be rebooting without any commands from the user. This is due to the power. Make sure that the Raspberry Pi is plugged into a power supply that outputs 5V and 2.5A.

Digital Analog Converting Software

The MCP4725 is a 12-bit DAC chip that converts the digital signal into an analog output to be properly displayed on the ProPAQ patient monitor. Determining what output voltage is given is. The voltage that is output is determined by the following equation:

$$V_{out} = \frac{(V_{cc} * Bit \quad Value)}{4096}$$

The V_{cc} is the output voltage from the Raspberry Pi. The system has a V_{cc} of 5V. There are two sources from the Raspberry Pi, 3.3V and 5V. The 5V source output higher resolution waveform and that is why that source was chosen.

FREQUENCY CALCULATION

In order for the ICP waveform to have the same time synchronization as the ECG waveform, they must have the same frequency. This equation calculates the frequency of the waveform based off of the beats per minute that is input by the user through the GUI. This is calculated using the following equation developed by last year's Senior Design group:

$$Period \quad of \quad Waveform = e^{\frac{-(BPM+193.74)}{54.91}}$$

The calculations supporting the values in the equation were unfortunately not provided by last year's group. This equation can be seen in each of the waveform model files (ex. modelone.py).

GUI

The figure below displays the GUI that is used for our system. The buttons in the waveform shape output the specified code. The heart rate is manually inputted by the user and used to calculate the frequency of the waveform (refer to frequency calculation for more information). No code is executed until the user clicks the update button. Once the user has completed their desired simulation, quit must be clicked to kill any process that is running and close out of the window.

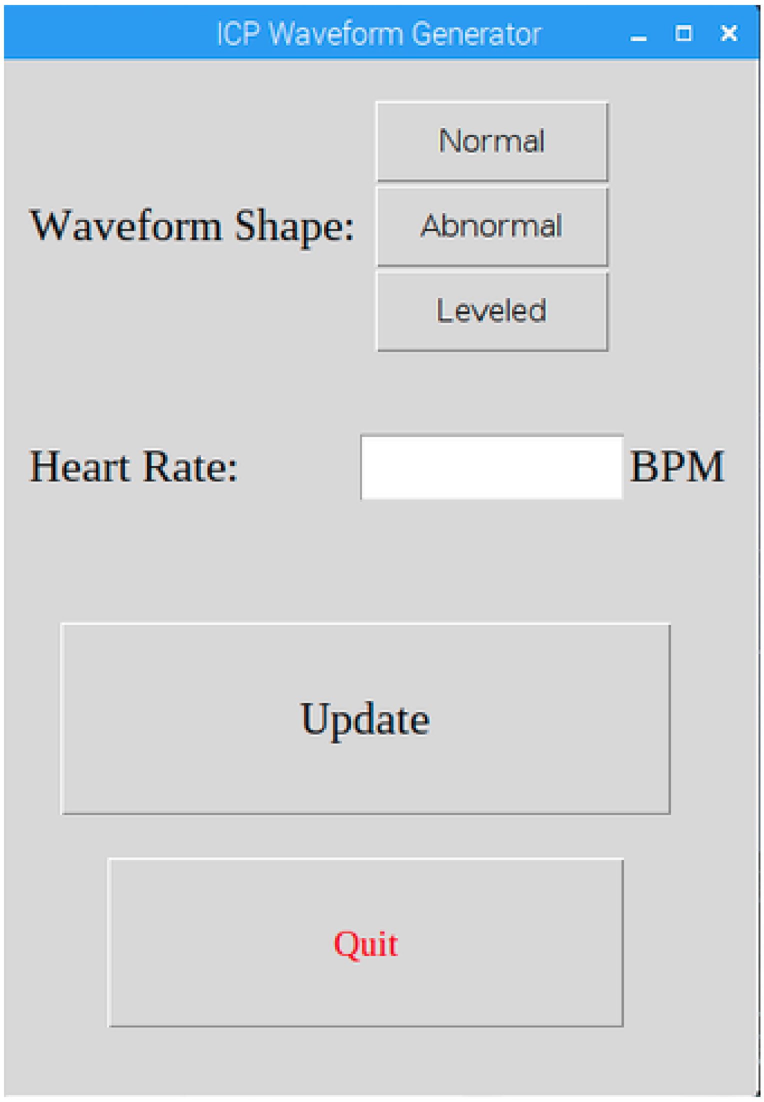


Figure 4: GUI

Listing 1: GUI Code

```
from guizero import Combo, App, Box, Text, Slider, TextBox, ButtonGroup, PushButton, CheckBox, MenuBar
import subprocess
import signal
import os
import sys

#Pulled in the necessary tools to create the GUI

class MyGUI:
    def __init__(self, master):
        self.master = master
       #Creates the application window

        self.space2 = Text(master, text=" ", grid=[0,0], width="fill")

        #Creates text for waveform shape
        self.Shapemessage1 = Text(master, size=18, text="  Waveform Shape:", grid=[0,2], align="left", font="Times")


        #Creates the waveform buttons
        self.Waveform1 = PushButton(master, text="Normal", grid=[6,1,2,1], width = "10", command = self.normal)
        self.Waveform1.text_size = 12
        self.Waveform2 = PushButton(master, text="Abnormal", grid=[6,2,2,1], width="10", command = self.abnormal)
        self.Waveform2.text_size = 12
        self.Waveform3 = PushButton(master, text="Leveled", grid=[6,3,2,1], width="10",  command = self.leveled)
        self.Waveform3.text_size = 12

        self.space2 = Text(master, text=" ", grid=[0,6])
        self.space2 = Text(master, text=" ", grid=[0,7])

        #Creates space between Waveform Shape and Heart Rate

        self.HRmessage = Text(master, size=18, text="  Heart Rate:", grid=[0,10], align="left", font="Times")
        #Creates the HR subtitle

```

```python
        self.rate = TextBox(master, grid=[6,10,2,1])
        self.rate.text_size = 20
        self.rate.width = 8
        self.HRunit = Text(master, size=18, text="BPM", grid=[8,10], font="Times")
        #Creates the HR textbox to allow for typed input

        self.space3 = Text(master, text=" ", grid=[0,11], font="Times")
        self.space3 = Text(master, text=" ", grid=[0,12], font="Times")
        self.space3 = Text(master, text=" ", grid=[0,13], font="Times")
        #Creates space between Heart Rate and the Update Button

        self.button = PushButton(master, command=self.update_value, width="18", height="3", text="Update", grid=[0,16,9,1], align="bottom")
        self.button.text_size = 18
        self.button.font = "Times New Roman"
        #Creates an Update Button that updates the inputed information and exports the user-inputs to the next part of the program


        self.when_clicked = self.update_value


        self.space3 = Text(master,text = " ", grid = [0,17])
        self.button = PushButton(master, command=self.quit, width="15", height="3", text="Quit", grid=[0,18,9,1], align="bottom")
        self.button.text_size = 15
        self.button.font = "Times New Roman"
        self.button.text_color = "red"
        #Creates the quit button which will exit out of app and stop the process from running

        self.pid = -1 #Initialize process id


    #Colors the Normal button black while other buttons go back to grey
    def normal(self):
        #self.done()
        self.waveform = "normal"
        self.Waveform1.bg = (0,0,0)
        self.Waveform1.text_color = (255,255,255)
        self.Waveform2.text_color = (0,0,0)
        self.Waveform2.bg = (214,214,214)
        self.Waveform3.text_color = (0,0,0)
        self.Waveform3.bg = (214,214,214)


    #Colors the abnormal button black while other buttons go back to grey
    def abnormal(self):
        #self.done()
        self.waveform = "abnormal"
        self.Waveform2.bg = (0,0,0)
        self.Waveform2.text_color = (255,255,255)
        self.Waveform1.text_color = (0,0,0)
        self.Waveform1.bg = (214,214,214)
        self.Waveform3.text_color = (0,0,0)
        self.Waveform3.bg = (214,214,214)


    #Colors the leveled button black while other buttons go back to grey
    def leveled(self):
        #self.done()
        self.waveform = "leveled"
        self.Waveform3.bg = (0,0,0)
        self.Waveform3.text_color = (255,255,255)
        self.Waveform1.text_color = (0,0,0)
        self.Waveform1.bg = (214,214,214)
        self.Waveform2.text_color = (0,0,0)
        self.Waveform2.bg = (214,214,214)


    #Command for updating the heart rate value
    #Also calls specific waveoform value based off of which button is pressed
    def update_value(self):
        self.done()
        if self.rate.value == "":
            self.rate.value=65
        if self.waveform == "normal":
            self.process = subprocess.Popen('python normal.py {}'.format(self.rate.
    value), shell=True, preexec_fn=os.setsid)
            self.pid = self.process.pid


        if self.waveform == "abnormal":
            self.process = subprocess.Popen('python abnormal.py {}'.format(self.rate.
    value), shell=True, preexec_fn=os.setsid)
            self.pid = self.process.pid


        if self.waveform == "leveled":
            self.process = subprocess.Popen('python leveled.py {}'.format(self.rate.
    value), shell=True, preexec_fn=os.setsid)
            self.pid = self.process.pid


    #Kills the process that is currently running
    def done(self):
        if self.pid > 0:
            try:
                os.killpg(os.getpgid(self.pid), signal.SIGTERM)
                self.pid = -1
            except ProcessLookupError:
                pass
    #Kills all processes and closes the window
    def quit(self):
        self.done()
        self.master.destroy()

#Specifies the size of the window for the GUI and the Title
root = App(title="Intracranial Pressure Waveform", width=400, height=550, layout="
    grid")
#Pulls up the GUI
my_gui = MyGUI(root)
root.display()
#Creates then application window
```

Waveform Codes

The waveform codes each rely on a different MATLAB file which contains a figure of the specified waveform. The data points forming this figure are pulled into the Python DAC code to translate each y-value into bit values. These values are read by the DAC via i2c through the Raspberry Pi's GPIO pins. These values essentially tell the DAC how much voltage to output via the DAC's Vout pin. This time-varying voltage is plotted onto the Propaq, forming the ICP waveform. These bit values were specified as hard-coded values due to the time constraint imposed during the project. The specific values were initially taken from last year's group's code. They were repeatedly adjusted throughout the testing phase of the project via a trial and error process. After each adjustment, the smoothness and peak definition of each waveform was visually evaluated and maximized.

Normal Waveform

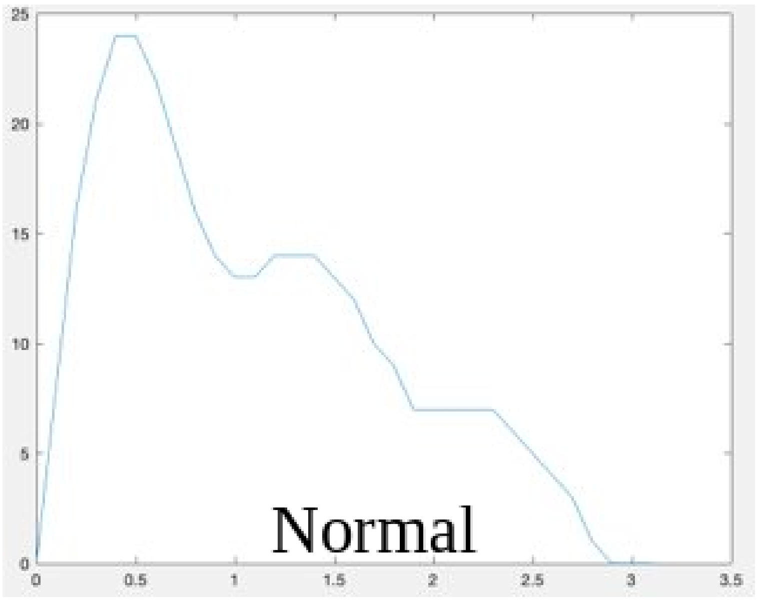


Figure 5: Normal Waveform in MATLAB

Listing 2: Normal Waveform MATLAB Code

```
t = 0:0.1:(2*pi-0.2);
windowWidth = 13;
y = (-sawtooth(t)+1);
y = conv(y,ones(1,windowWidth)/windowWidth,'same');
X = ((0.5*sinc(t-4)+(0.25*sinc(t-2.5) + sinc(t+0.2))) + (y));
%Normalize the data between 1 and 21
normX = X - min(X(:));
normX = normX./max(normX(:));
normX_two = 21*normX+1;
%Repeat the Data Points to get multiple waveforms connected togethor
X1 = repmat(normX_two,1,2);
t = 0:length(X1)-1;
plot(t,X1)
%Create the Model By isolate the three peaks and Grabbing the data between them
[peaks, location] = findpeaks(X1);
t_data = t(location);
peaks = peaks;
t_data_reshape = vec2mat(t_data,3);
peaks_reshape = vec2mat(peaks, 3);
[one_period, index_period] = min(t_data_reshape,[],2);
jdex_period = [1:1:length(index_period)]';
[subscript_period] = sub2ind(size(t_data_reshape), jdex_period, index_period);
finaltdata = t_data_reshape(subscript_period);
modeloney = X1(finaltdata(1):finaltdata(2));
%Final Model Data
modelonex = t(finaltdata(1):finaltdata(2));
%Interpolate the Data across 100 points to create a model waveform matrix of 100
    points
initialdata = linspace(0, 1, 62);
V = linspace(0,1,100);
interpolateddata = interp1(initialdata, modeloney, V);
%Round all of the datapoints to integers,
%ICP used to be old interpoalteddatay variable that is used in
%modelonescale.mat and in modelone
ICP = floor(interpolateddata)
plot(V, ICP)
%save('ICP01.mat', 'ICP')
```

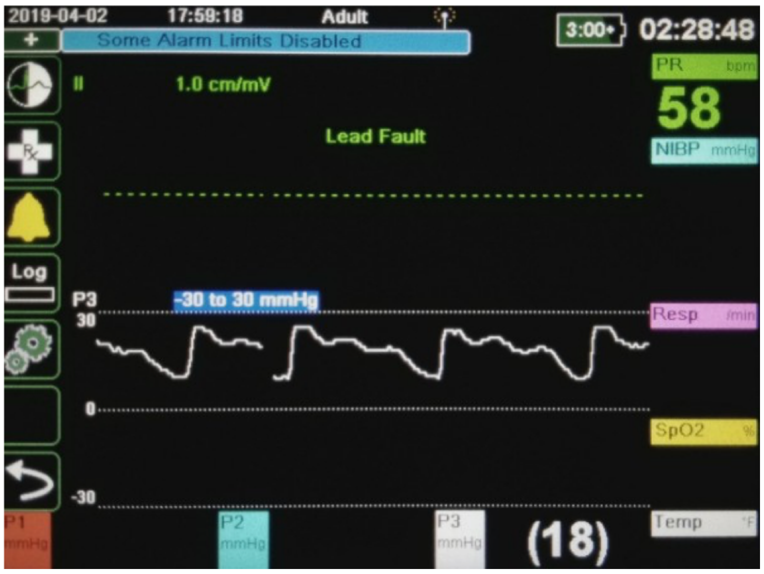


Figure 6: Normal Waveform on ProPAQ

Listing 3: Normal Waveform DAC Code

```
from scipy import io
import numpy as np
import time
import sys
import math
import Adafruit_MCP4725

dac = Adafruit_MCP4725.MCP4725()
first_arg = sys.argv[1]

def modelone(intvalue = first_arg): #first_arg
    intvalue = int(intvalue)
    mat = io.loadmat('modelonescale.mat');
    y_value = mat['interpoalteddatay'];
    y_value = np.transpose(y_value);
    new_value = np.zeros(y_value.shape)

    for jj in range(len(y_value)):
        if y_value[jj] == 28:
            new_value[jj] = 208
        elif y_value[jj] == 27:
            new_value[jj] = 206
        elif y_value[jj] == 26:
            new_value[jj] = 204
        elif y_value[jj] == 25:
            new_value[jj] = 202
        elif y_value[jj] == 24:
            new_value[jj] = 200
        elif y_value[jj] == 23:
            new_value[jj] = 198
        elif y_value[jj] == 22:
            new_value[jj] = 196
        elif y_value[jj] == 21:
            new_value[jj] = 194
        elif y_value[jj] == 20:
            new_value[jj] = 192
        elif y_value[jj] == 19:
            new_value[jj] = 189
        elif y_value[jj] == 18:
            new_value[jj] = 187
        elif y_value[jj] == 17:
            new_value[jj] = 184
        elif y_value[jj] == 16:
            new_value[jj] = 182
        elif y_value[jj] == 15:
            new_value[jj] = 179
        elif y_value[jj] == 14:
            new_value[jj] = 177
        elif y_value[jj] == 13:
            new_value[jj] = 174
        elif y_value[jj] == 12:
            new_value[jj] = 172
        elif y_value[jj] == 11:
            new_value[jj] = 169
        elif y_value[jj] == 10:
            new_value[jj] = 167
        elif y_value[jj] == 9:
            new_value[jj] = 164
        elif y_value[jj] == 8:
            new_value[jj] = 162
        elif y_value[jj] == 7:
            new_value[jj] = 158
        elif y_value[jj] == 6:
            new_value[jj] = 156
        elif y_value[jj] == 5:
            new_value[jj] = 154
            elif y_value[jj] == 4:
                new_value[jj] = 152
            elif y_value[jj] == 3:
                new_value[jj] = 149
            elif y_value[jj] == 2:
                new_value[jj] = 147
            elif y_value[jj] == 1:
                new_value[jj] = 144

    var = 1;

    X = math.exp(-(intvalue+193.74)/54.91)
    new_value = new_value.flatten()
    while var == 1:
        for val in new_value:
            val = int(val)
            dac.set_voltage(val)
            time.sleep(X)
if __name__ == "__main__":
    modelone()
```

Abnormal Waveform

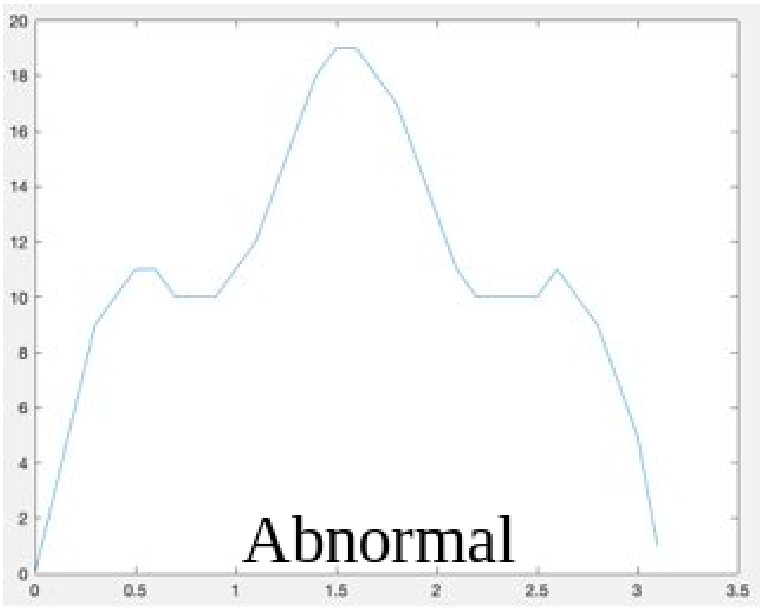


Figure 7: Abnormal Waveform in MATLAB

Listing 4: Abnormal Waveform MATLAB Code

```
to = linspace(0,20,200);
y = square(to);
x = 1.1*sin(5*to);
y2 = x + y;
ts = linspace(6,10, 200);
y3 = 2.5*sinc(2.3*(ts-7.6));
y3(110:end) = 0;
y4 = y3 + y2;
%Parabola for the left side to smooth the edges
yp = -6*(to-6.5).^2 + 2.5;
yp(1:55) = 0;
yp(64:end) = 0;
%Parabola for the right side to smooth the edges
yr = -3*(to-9).^2 + 2.5;
yr(1:94) = 0;
yr(103:end) = 0;
ybr = 3*(to-10.6).^2 + 0.0018;
ybr(1:102) = 0;
ybr(107: end) = 0;
threey = (y4 + .5*yp + .5*yr + 2*ybr)
newthree = threey(58:100);
xdata = linspace(0,1,size(newthree,2))
plot(xdata, newthree)
endpoint = newthree(end);
d = [10:0.1:12];
y = (d-11).^2 + (endpoint-1);
yf = y(1:15);
newthreeprime = newthree;
newthreeprime(43:57) = yf
%Matrix that holds the model
xf = [1:1:57];
%Normalize the data points between 1 and 28
normthree = newthreeprime - min(newthreeprime(:));
normthree = normthree./max(normthree(:));
normthree_five = 28*normthree +1;
%Repeat the Data Points to get multiple waveforms connected together
X1 = repmat(normthree_five,1,2);
t = 0:length(X1)-1;
plot(t,X1);
%Create the Model By isolating the three peaks and Grabbing the data between them
[peaks, location] = findpeaks(X1);
```

```
t_data = t(location);
peaks = -peaks;
t_data_reshape = vec2mat(t_data,3);
peaks_reshape = vec2mat(peaks, 3);
[one_period, index_period] = max(t_data_reshape,[],2);
jdex_period = [1:1:length(index_period)]';
[subscript_period] = sub2ind(size(t_data_reshape), jdex_period, index_period);
finaltdata = t_data_reshape(subscript_period);
modelthreeyvalue = X1(finaltdata(1):finaltdata(2))
modelthreexvalues = t(finaltdata(1):finaltdata(2))
%Interpolate the Data across 100 points to create a model waveform matrix
%of 100 points
initialdata = linspace(0, 1, size(modelthreeyvalue,2));
V = linspace(0,1,100);
ICP = interp1(initialdata, modelthreeyvalue, V)
%Round all of the datapoints to integers
ICPA = floor(ICP)
plot(V, ICPA)
%ICP used to be old interpoalteddatay variable that is used in
%modelonescale.mat and in modelone
%save('ICPA30.mat', 'ICPA')
```

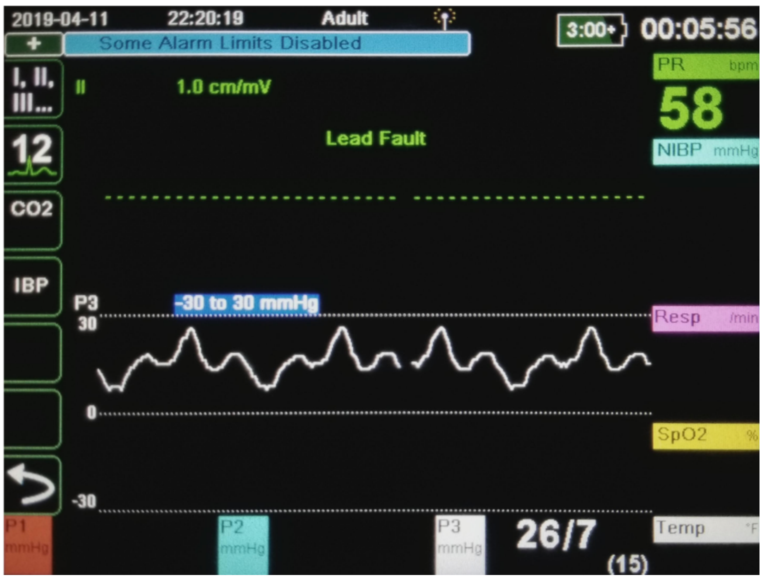


Figure 8: Abnormal Waveform on ProPAQ

Listing 5: Abnormal Waveform DAC Code

```
from scipy import io
import numpy as np
import time
import sys
import math
import Adafruit_MCP4725

dac = Adafruit_MCP4725.MCP4725()
first_arg = sys.argv[1]

def modelthree(intvalue = first_arg):
    intvalue = int(intvalue)
    mat = io.loadmat('ICPL.mat');
    y_value = mat['ICPL'];
    y_value = np.transpose(y_value);
    new_value = np.zeros(y_value.shape)

    for jj in range(len(y_value)):
        if y_value[jj] == 31:
            new_value[jj] = 219
        elif y_value[jj] == 30:
            new_value[jj] = 217
        elif y_value[jj] == 29:
            new_value[jj] = 214
        elif y_value[jj] == 28:
            new_value[jj] = 212
        elif y_value[jj] == 27:
            new_value[jj] = 202
        elif y_value[jj] == 26:
            new_value[jj] = 199
        elif y_value[jj] == 25:
            new_value[jj] = 196
        elif y_value[jj] == 24:
            new_value[jj] = 184
        elif y_value[jj] == 23:
            new_value[jj] = 182
        elif y_value[jj] == 22:
            new_value[jj] = 179
        elif y_value[jj] == 21:
            new_value[jj] = 177
        elif y_value[jj] == 20:
            new_value[jj] = 167
        elif y_value[jj] == 19:
            new_value[jj] = 164
        elif y_value[jj] == 18:
            new_value[jj] = 162
        elif y_value[jj] == 17:
            new_value[jj] = 160
        elif y_value[jj] == 16:
            new_value[jj] = 150
        elif y_value[jj] == 15:
            new_value[jj] = 148
        elif y_value[jj] == 14:
            new_value[jj] = 146
        elif y_value[jj] == 13:
            new_value[jj] = 144
        elif y_value[jj] == 12:
            new_value[jj] = 142
        elif y_value[jj] == 11:
            new_value[jj] = 139
        elif y_value[jj] == 10:
            new_value[jj] = 138
        elif y_value[jj] == 9:
            new_value[jj] = 136
        elif y_value[jj] == 8:
            new_value[jj] = 134
        elif y_value[jj] == 7:
            new_value[jj] = 132
        elif y_value[jj] == 6:
            new_value[jj] = 130
        elif y_value[jj] == 5:
            new_value[jj] = 128
        elif y_value[jj] == 4:
            new_value[jj] = 126
        elif y_value[jj] == 3:
            new_value[jj] = 120
        elif y_value[jj] == 2:
            new_value[jj] = 116
        elif y_value[jj] == 1:
            new_value[jj] = 114

    new_value = new_value.flatten()
    var = 3;
    X = math.exp(-(intvalue+193.74)/54.91)
    while var == 3:
        for val in new_value:
            val = int(val)
            dac.set_voltage(val)
            time.sleep(X)

if __name__ == "__main__":
    modelthree()
```

Leveled Waveform

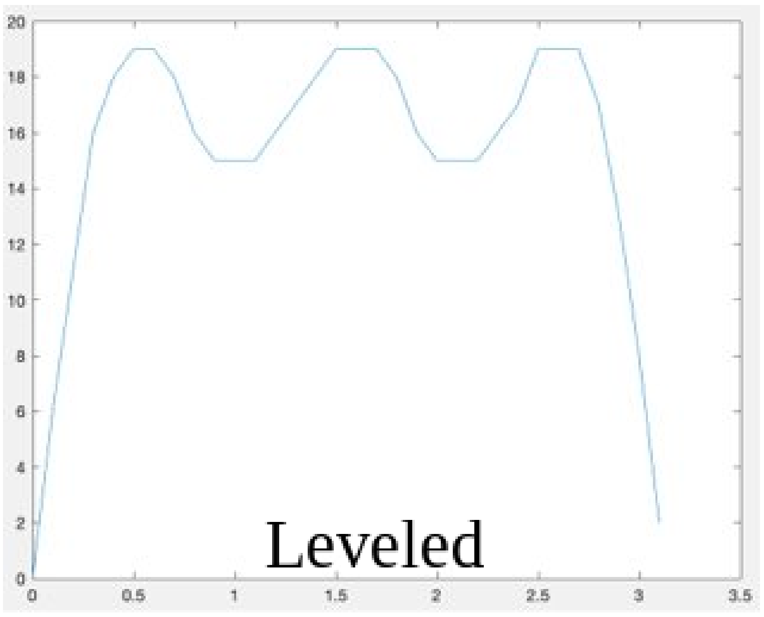


Figure 9: Leveled Waveform in MATLAB

Listing 6: Leveled Waveform MATLAB Code

```
t = [0:0.1:5];
y1 = (-sawtooth(t)+1);
y2 = sinc(t);
y3 = y1 + y2;
y4 = 0.7*sinc(t-0.5);
y5 = y4 + y1;
y6 = 1.2*sinc(t-2);
y7 = y5 + y6;
plot(t,y7,'g');
y8 = 1.8*sinc(t-3.5);
y8(1:25) = 0;
plot(t, y8, 'p');
%Smooth out one of the peaks through convolution
windowWidth = 6;
y = conv(y8, ones(1,windowWidth)/windowWidth,'same');
y9 = y + y7;
%Make the Edges more curved
ynew = y9(1:45);
yendpoint = ynew(end)
d = linspace(4.5, 6.5, 21);
ypar = (d-5.5).^2 + (yendpoint-1);
plot(d, ypar)
yf = ypar(1:15)
newtwoprime = ynew;
newtwoprime(45:59) = yf
xf = [1:1:59];
%Make the Second peak a little higher to match up
ytop = ynew(1);
dt = linspace(-1,1,21);
yptop = -(dt).^2 + (ytop)
yfn = yptop(1:10)
newertwoprime = yfn;
newertwoprime(11:69) = newtwoprime;
%Normalize the data points between 1 and 21
normtwo = newertwoprime - min(newertwoprime(:));
normtwotwo = normtwo./max(normtwo(:));
normtwofive = 21*normtwotwo+1;
%Repeat the Data Points to get multiple waveforms connected togethor
X1 = repmat(normtwofive,1,3);
t = 0:length(X1)-1;
plot(t,X1)
```

```
%Create the Model By isolating the three peaks and Grabbing the data between them
[peaks, location] = findpeaks(X1);
t_data = t(location);
peaks = -peaks;
t_data_reshape = vec2mat(t_data,3);
peaks_reshape = vec2mat(peaks, 3);
[one_period, index_period] = max(t_data_reshape,[],2);
jdex_period = [1:1:length(index_period)]';
[subscript_period] = sub2ind(size(t_data_reshape), jdex_period, index_period);
finaltdata = t_data_reshape(subscript_period);
modeltwoyvalue = X1(finaltdata(1):finaltdata(2))
%Final Model Data
modeltwoxvalues = t(finaltdata(1):finaltdata(2))
%Interpolate the Data across 100 points to create a model waveform matrix
%of 100 points
initialdata = linspace(0, 1, 70);
V = linspace(0,1,100);
ICP = interp1(initialdata, modeltwoyvalue, V);
%Round all of the datapoints to integers
ICPL = floor(ICP)
plot(V, ICPL)
%ICP used to be old interpoalteddatay variable that is used in
%modelonescale.mat and in modelone
%save('ICPL30.mat', 'ICPL')
```

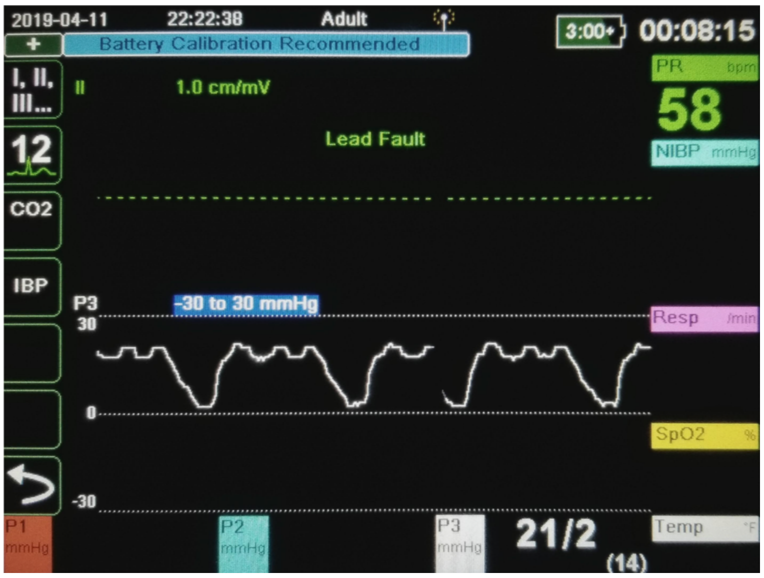


Figure 10: Leveled Waveform on ProPAQ

Listing 7: Leveled Waveform DAC Code

```
from scipy import io
import numpy as np
import time
import sys
import math
import Adafruit_MCP4725

dac = Adafruit_MCP4725.MCP4725()
first_arg = sys.argv[1]

def modeltwo(intvalue = first_arg):
    intvalue = int(intvalue)
    mat = io.loadmat('modeltwoscale.mat');
    y_value = mat['modeltwo'];
    y_value = np.transpose(y_value);
    new_value = np.zeros(y_value.shape)

    for jj in range(len(y_value)):
        if y_value[jj] == 31:
            new_value[jj] = 219
        elif y_value[jj] == 30:
            new_value[jj] = 217
        elif y_value[jj] == 29:
            new_value[jj] = 214
        elif y_value[jj] == 28:
            new_value[jj] = 212
        elif y_value[jj] == 27:
            new_value[jj] = 202
        elif y_value[jj] == 26:
            new_value[jj] = 199
        elif y_value[jj] == 25:
            new_value[jj] = 196
        elif y_value[jj] == 24:
            new_value[jj] = 184
        elif y_value[jj] == 23:
            new_value[jj] = 182
        elif y_value[jj] == 22:
            new_value[jj] = 179
        elif y_value[jj] == 21:
            new_value[jj] = 177
        elif y_value[jj] == 20:
            new_value[jj] = 167
        elif y_value[jj] == 19:
            new_value[jj] = 164
        elif y_value[jj] == 18:
            new_value[jj] = 162
        elif y_value[jj] == 17:
            new_value[jj] = 160
        elif y_value[jj] == 16:
            new_value[jj] = 150
        elif y_value[jj] == 15:
            new_value[jj] = 148
        elif y_value[jj] == 14:
            new_value[jj] = 146
        elif y_value[jj] == 13:
            new_value[jj] = 144
        elif y_value[jj] == 12:
            new_value[jj] = 142
        elif y_value[jj] == 11:
            new_value[jj] = 139
        elif y_value[jj] == 10:
            new_value[jj] = 138
        elif y_value[jj] == 9:
            new_value[jj] = 136
        elif y_value[jj] == 8:
            new_value[jj] = 134
        elif y_value[jj] == 7:
            new_value[jj] = 132
        elif y_value[jj] == 6:
            new_value[jj] = 130
        elif y_value[jj] == 5:
            new_value[jj] = 128
        elif y_value[jj] == 4:
            new_value[jj] = 126
        elif y_value[jj] == 3:
            new_value[jj] = 120
        elif y_value[jj] == 2:
            new_value[jj] = 116
        elif y_value[jj] == 1:
            new_value[jj] = 114

    new_value = new_value.flatten()
    var = 2;
    X = math.exp(-(intvalue+193.74)/54.91)
    while var == 2:
        for val in new_value:
            val = int(val)
            dac.set_voltage(val)
            time.sleep(X)
if __name__ == "__main__":
    modeltwo()
```