Contact-Free Heart and Respiration Rate Measurement System Based on Real-Time Human Face Video

Eugene Dobryakov (Team Leader)
Quan Nguyen
Roman Melnik

Team / Project: 115

Department of Electrical Engineering and Computer Science, Cleveland State University

Submitted to—

Dr. Almabrok Essa
Department of Electrical Engineering and Computer Science, Cleveland State University

Dr. Sanchita Mal-Sarkar
Department of Electrical Engineering and Computer Science, Cleveland State University

Contents

Executive Summary

As the COVID-19 pandemic continues, early symptom detection is of upmost importance. According to a recent study, elevated heart rate (HR) and respiration rate (RR) are physiological signs associated with COVID-19 [1]. The heart is at the center, and the most important organ of the circulatory system. The clinical standard to measure accurate HR is through an electrocardiogram, a process commonly referred to as an ECG test. In an ECG test, multiple pads consisting of electrodes are placed on the patient to measure the change in potential fluctuations from different angles. ECG tests are performed by professionals. The test requires direct contact with skin, and the costs for an ECG test can add up. In a recent survey by the COVID-19 Healthcare Coalition, more that 50% of responders suggested they would delay their care during the COVID-19 pandemic if they did not have access to telehealth services [3]. While telehealth is a good option, it still lacks the ability to read the patient's HR and RR.

According to the CDC, there is a need for a major overhaul to deliver health care during the pandemic [4]. According to WebMD, an individual infected with COVID-19 is likely to transmit the virus to at least two additional people [5]. The virus that causes COVID-19, is highly infectious and can spread from person to person, including through aerosol transmission of particles produced when an infected person exhales, talks, vocalizes, sneezes, or coughs. Even people who show no symptoms can also carry the virus. Particles containing the virus have a radius of spreading of more than 6 feet. Indoors and dry conditions have even more risk of spreading the virus by people with or without knowing that they are infected. One way to limit the spread of the virus is to eliminate unnecessary contact and interactions with other individuals. There is a need to improve the delivery of health care during the pandemic, and an opportunity to introduce a contact-free method to measure the HR and RR in real-time.

The Contact-Free Heart and RR Measurement System Based on Real-Time Human Face Video can measure accurate HR and RR in real-time. With a camera that can capture the person's face and a display that can tell the vital information, anyone can have their HR and RR measured in a convenient and safe way. By identifying the optical difference on the epidermis due to blood volume variations during a cardiac cycle, the system is capable of accurately reading the patient's HR and RR in real-time. The lift and shift system provides a way to keep track of vitals in real-time while contributing to overhauling the delivery of health care during the pandemic and reducing the opportunity to spread COVID-19.

1. Problem Statement and Background

Advancements in the medical field are not proportional. While some sectors see continuous evolution with new products or services, others remain unchanged with little to no innovation. For example, some fields welcome the use of 3D printers to create implants, such as cranial plates, or external prostheses, such as hands [6]. Meanwhile, the leading method for individuals to measure HR is to lightly press their index and middle finger on the opposite wrist and perform a manual count [7].

Electrocardiogram, referred to as an ECG, continues to remain the clinical standard for accurate HR measurement. The ECG test may not be for everyone since it requires multiple electrode pads to make direct contact with skin. Over the years, feedback has shown the electrode pads may cause tissue breakdown or skin irritation should the pads be left on for too long [8]. Certain factors, such as pregnancy or obesity, could interfere with the results [8a].

Over the last five years, wearable technology such as fitness trackers became a popular item to track and measure HR. Most fitness trackers utilize optical sensors to detect blood flow. For example, the Apple Watch emits a green LED light and calculates the HR by determining the amount of green light absorbed into the skin through the process known as photoplethysmography [9]. But this is not without its issues. Fitness trackers measure the blood flow when it is the farthest away from the heart. Naturally, this and various other factors reduce the accuracy of the reading [10].

The COVID-19 pandemic caused a global shift in everyone's lives. Routine tasks such as going to a doctor became anything but routine. According to the CDC, 4 in 10 U.S. adults stated they would avoid medical care due to concerns regarding COVID-19 [11]. The survey also found out those with two or more underlying conditions, such as heart disease, were more likely to avoid urgent care or emergency care altogether [11].

All these challenges present an opportunity to re-evaluate the delivery of health care both during and subsequent of the pandemic. The heart will continue to remain at the center and will still be the most important organ of the circulatory system. It acts as a muscular pump that has four chambers and runs independently on its own electrical system, which is known as the cardiac conduction system. From the top of the heart, electricity starts flowing to both the right and the left ventricle. The right ventricles' job is to carry blood to the lung and the left ventricles carry blood to the rest of the body. The heart muscle cells can also create electricity independently of this cardiac conduction system in diseased states [2]. Studies have shown that elevated HR and RR are physiological signs associated with COVID-19 [1]. This disease can cause the heart cells to generate abnormal electricity, as well as create short circuits that lead to the irregular heart rhythms. Heart rhythm disorders fall primarily into two subgroups: fast rhythms and slow rhythms. Fast rhythms occur when the heart beats rapidly, becomes fatigued, and causes shortness of breath. This is when the RR exceeds 22 breaths per minute. The mortality risk for patients hospitalized with COVID-19 increases at the rate of 1.9 to 3.2-fold compared to patients with normal RR [12]. People having this symptom are at a higher risk of experiencing a stroke. Slow rhythms occur when the electricity flow in the heart becomes slow or blocked. However, those kinds of blockages may resolve on their own. Still, an early prediction of an abnormally fast or slow HR and RR has

a vital role in preventing further dangerous situations. There is no better time than now to overhaul the HR and RR measurement process and introduce a new lift and shift system capable of capturing the HR and RR contact-free in real-time.

2. Design Objectives

There are several design objectives that are necessary to create a contact-free system capable of measuring the HR and RR in real-time through a process known as remote photoplethysmography (rPPG). The following objectives must be performed in real-time. The objectives are as follows:

1. Use an RGB-D camera to capture the subject's frontal view and pass the video stream to the program.
2. For each frame of the video, extract the subject's facial landmarks using a Python library called Dlib.
3. From the landmarks, extract the Region of Interest (ROI) which is the subject's forehead.
4. Design an algorithm to extract the raw signals from the ROI.
5. Design a system to convert the output to the frequency domain to extract the HR and RR.
6. Display the HR and RR as an output visible to the subject.

The first objective requires capturing the ROI on the subject's frontal view by using an RGB-D camera. RGB-D cameras are commonly referred to as range cameras. It is a 3D imaging system that concentrates on capturing dynamic human activities with minimal interference [13]. The RGB sensors on the camera make it possible to split the captured image into channels such as red, green, and blue. Some cameras come equipped with a software development kit (SDK) to allow for added flexibility.

The ROI is the subject's forehead, because the forehead is one of the most common locations to measure arterial oxygen saturation using pulse oximetry [14]. Since the forehead region is less susceptible to movement and "[since it] is well perfused by arteries branching from the internal carotid" [14], it serves as a solid foundation to extract a reliable reading. Using a pre-trained artificial intelligence model in the Dlib library, we can extract the facial landmarks. The model can detect 68 facial landmarks from the face and return the x and y coordinates.

The next objective is to extract the raw signal from the ROI. Typically, RGB-D cameras can capture video at the rate of 30 frames per second (30-fps). Some RGB-D cameras may be capable of capturing at a higher rate. Processing a pre-recorded two second video captured at 30-fps will result in 60 frames. However, the number of frames captured for the same video processed in real-time remains to be determined. The raw signal extracted in this objective is derived from the subset of data collected from the video captured in real-time on the ROI.

The third objective is to convert the captured output to the frequency domain. The Fast Fourier Transform (FFT) is a mathematical method that is used to convert a time domain signal to a frequency domain signal. The frequency domain signal indirectly provides a subset of the QRS complex, which represents ventricular depolarization. In other words, it is the line depicted by the

heart monitor typically found beside a patient at the hospital. This information could then be used to calculate the actual HR and RR.

The fourth and final objective is to display the output. The client should see their HR and RR in a readable and understandable format.

3. Technical Approach

The rPPG measurement method will be utilized in this project to obtain the HR and RR. In rPPG, the optical difference in the human skin due to the blood volume variations during a cardiac cycle is captured. This leads to a difference in the color space of the same area on the body in different frames of the video due to the reflected light.

The ongoing COVID-19 pandemic, limitations with current medical technology, and inaccurate readings with wearable health trackers illustrate the lack of accurate and contact-free systems to obtain the HR and RR of the human body in real-time. This presents an opportunity to develop a new, contact-free system that can accurately read an individual's HR and RR in real-time. It is the aspiration of the team to take on this mission and deliver this product.

Monitoring the HR and RR by this no-contact method relies on the difference in the color space in the human skin due to the blood volume variations in a cardiac cycle. To derive the HR and RR, the data from a real-time video is processed by a Python program from an RGB-D camera through a wired connection. The algorithm identifies a region of interest and focuses its efforts on the green channel because the skin is more likely to absorb the green color than red or blue colors.

The chosen area for data extraction and computation is the forehead due to its accessibility and efficiency for computation. The forehead area is detected using a pre-trained AI model of CMake and Dlib library in Python. The model is able to detect 68 facial landmarks from the face and return the x and y coordinates. The indices of 68 pairs of coordinates can be seen in Figure 1.

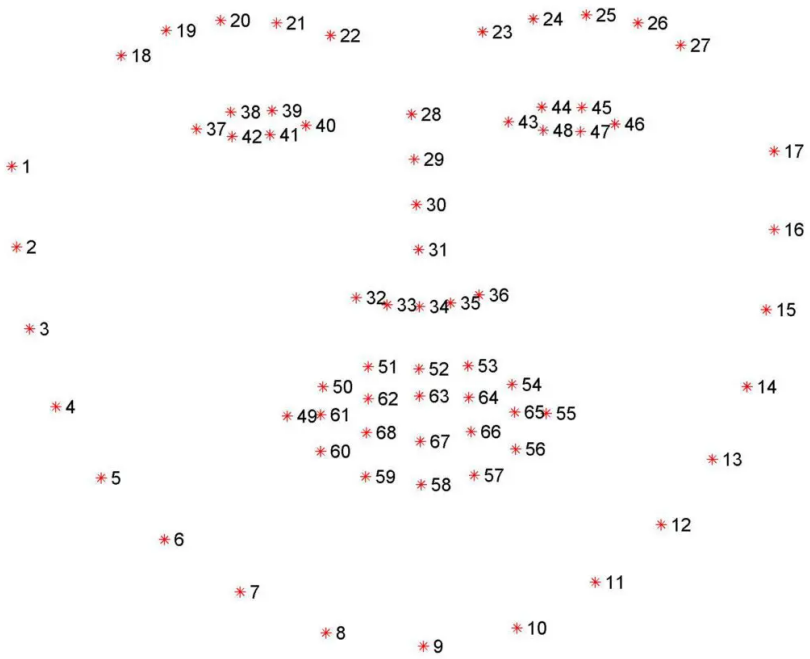


Figure 1: 68 Facial Landmarks

In each frame of the video, pixels in the region of interest are captured and stabilized. The stabilization is needed to ensure the input ROI in each frame does not have significant change even when the subject moves far away or closer to the camera, or when they tilt their head.

The signal is then converted from the time domain to the frequency domain by using the FFT function. The HR and the RR are calculated based on the peak of the signal corresponding to the frequency range of [0.8, 2.2] and [0.18, 0.5] respectively. The results are then displayed to the subject.

3.1. The Scope

For each periodic cardiac cycle, there are skin tone effects that can be extracted to calculate the HR and RR. The light in the environment has many effects on the human skin of the forehead. One of the effects is the change in the blood volume and blood vessels [14-15]. With careful research, it has been proven that natural light is good enough to receive acceptable plethysmography signals [15]. With the help of powerful image processing libraries and algorithms, it is possible to detect the change in the intensity of the pixels and convert that data to the frequency domain to calculate the HR and RR. The implementation to process live video and to extract the features on the subject's forehead will be done in Python since it has powerful image processing libraries. In this project, we will primarily use OpenCV and NumPy for video frames processing, CMake and Dlib for facial recognition and facial landmarks extraction, and SciPy for signal processing.

3.2. Feeding Live Video into the Program

To meet the primary goals of this project, the program must accurately determine the subject's HR and RR in real time. That means the input must be a live video from the camera connected to the computer. By using Python libraries such as OpenCV, we can get the live video from the input source. It is important to consider the connection from the camera to the computer. The data must be clear enough for the algorithms to process it accurately and successfully. The number of frames processed per second from the real-time video must be high enough to achieve an accurate reading. To achieve this goal, wired connection must be considered since the data is transferred with higher speeds, lower loss rate, and lower latency. Moreover, with a wired connection the data is less likely to be intercepted and manipulated by outside sources during the transfer process. Thus, wired connection is also the more ethical approach.

3.3. Selecting the Region of Interest (ROI)

When selecting the region on the body to extract the rPPG information, we care about its accessibility and minimizing its area to ensure the efficiency of computation. The human face is chosen since it is not covered by clothes or accessories and has the minimum area. Previously conducted research has shown that the forehead is the most suitable region on the face from which to collect the rPPG signal for extraction and computation [16].

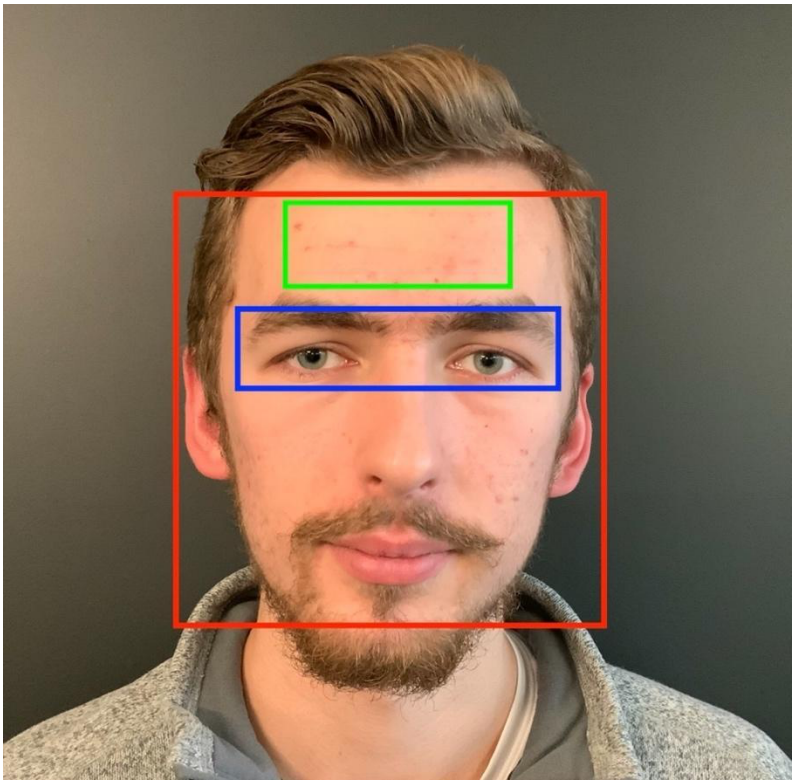


Figure 2: Identifying ROI

3.4. Detecting the Forehead

To find the ROI on the subject from real-time video input, we need to detect the subject's face and eyes using a pre-trained model of CMake and Dlib library functions in Python. This pre-trained model can detect 68 facial landmarks on the human face with specific indices. As can be seen in Figure 1, to get the forehead, we only need 4 important landmarks, 20, 25, 38 and 44. With the need of detecting human faces in real time, the Dlib library has proved to be a robust and effective method [17]-[18]. The pre-trained model is a .dat file which needs to be placed inside the program root folder to be loaded during program execution. The face detection needs to be done on grayscale images. Therefore, the input frames are converted to grayscale using the OpenCV function called cvtColor.

One of the problems of the 68 facial landmarks model is it does not represent the correct ROI which is the subject's forehead without some configuration in the landmark's indices. From Figure 1, the highest locations that can form the shape of the ROI is from the upper part of the eyes to the eyebrows. To find the coordinates of the point above the eyebrows, we need to take a symmetrical point of the top of the eye through the eyebrow on both sides. For example, from location 38, we need to take its symmetrical location through location 20 in order to get the upper left corner of the forehead. The same can be done for location 44. In addition, the ROI needs to be modified into a smaller region. The reason for that is to reduce the interference of the eyebrow to the signals captured as well as to reduce surrounding noises.

3.5. Extracting Raw Signal

Using the coordinates of the ROI, we can extract the raw signal from each frame of the video. In the normal environment, sometimes the light condition is unstable with sudden light change or objects can block the light source in a brief moment. Therefore, the signal is detrended to avoid the interference of light change. The signal is then passed through interpolation and a hamming window to make the signal become more periodic to avoid spectral leakage. After that, the raw signal is normalized to cancel out the noise. Then by using the FFT function in the NumPy library, the raw signal is then converted to a frequency and stored in the result list. The FFT function is one of the most efficient and popular tools in signal processing [19]. It converts a signal into individual spectral components, so it provides a frequency spectrum of the signal in the time domain.

3.6. Analyzing the Signals

The frequency domain is used to find frequency range that corresponds to the HR and RR. The HR frequency can range from 0.8 Hertz to 2.2 Hertz, and the RR frequency can range from 0.18 Hertz to 0.5 Hertz [20]. Equipped with this information, it is possible to leverage the cutoff frequency of the bandpass filter to extract the HR and RR signals from the original rPPG signal. The peak of the signal in the range of 0.8 to 2.2 returns the value of the HR. The peak of the signal in the range of 0.18 to 0.5 returns the value of the RR. This resembles the QRS complex. The value can be multiplied by 60 to get the number of heart beats per minute and the number of breaths per minute. These results, the HR and RR, are then displayed to the subject.

4. Detailed Design

The goal of our project is to create a contact free HR and RR system that operates in real time. When we began to work on the project, we experienced unexpected and unforeseen difficulties. The COVID-19 pandemic caused massive issues in the global supply chain, and this caused the delivery of the equipment for this project to be delayed. However, we were fortunate to find an identical RGB-D camera for nearly the same price on eBay. We placed the order and received the camera in less than a week. Upon delivery of the camera, we were able to begin our work.

4.1. Equipment

There are three pieces of equipment that are fundamental to the success of this project. To create a functional system, we need an RGB-D camera to capture a video of the user's face, a Python program to process the video, and an output display to project the results. The chart in Figure 3 shows this sequence of events.

Figure 3: Process Diagram

4.1.1. Python Program

The program for this project will be developed in Python. We chose Python for many reasons. It is a high-level, general purpose programming language. It is one of the most popular languages in the world. It has a robust presence of third-party libraries and modules that are easily installed. Python is flexible, open source, user friendly, and follows object-oriented methodology. A Python application can be run on a Windows, Mac (Intel-based architecture), and Linux machine.

This project involves image/video processing, and Python is a powerful language to utilize for this purpose. It has the ability to use advanced image editing and data analytics libraries such as NumPy, OpenCV, PIL, Pandas, Dlib, ImUtils, and many more. Using these libraries together, we are able to receive a video input of a person's face, analyze the video to obtain the HR and RR, and display the data in a user-friendly Graphical User Interface (GUI).

In our project, it is assumed that the user has a computer that is capable of running the Python program. Although the computer can be used as a form of a display, it is also assumed that the computer will only be dedicated to video processing, and there will be a monitor that is dedicated to display the results.

4.1.2. RGB-D Camera

The second piece of critical equipment is the RGB-D camera. After extensive research on various cameras, we settled on the Intel RealSense D455, the latest model in the D400 series. The D455 model has increased range and accuracy by a factor of 2 compared to the D435 model. The camera can be used in an indoor and outdoor environment, with an ideal range of 0.6 to 6 meters. The camera has an RGB frame resolution of 1280 x 800 and a frame rate of 30 FPS. The camera also comes with a SDK and plenty of supporting documentation from the manufacturer. The camera has a wired connection, as illustrated in Figure 4. The internal block diagram of the D400 series is shown in Figure 5. The camera itself can be seen in Figure 6.

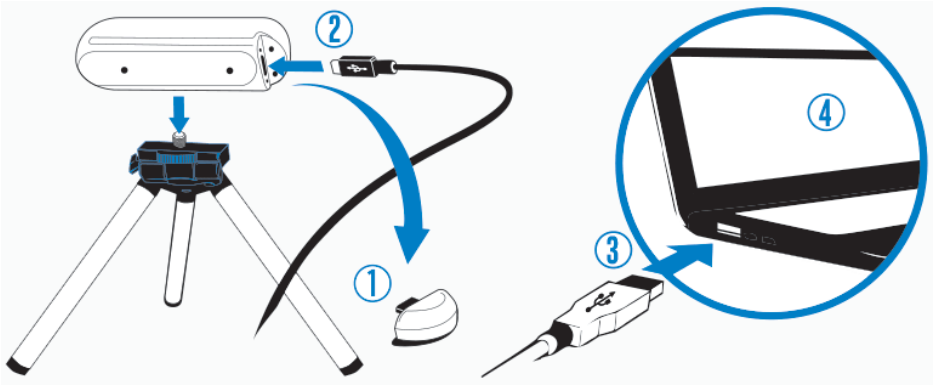


Figure 4: Intel RealSense D400 Series Camera - Connection

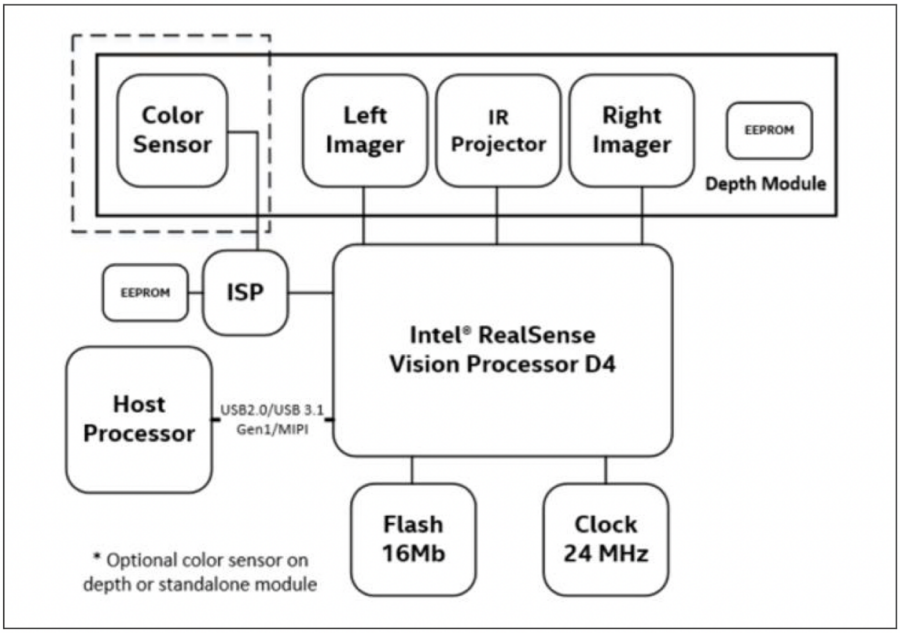


Figure 5: D400 Series Camera Block Diagram (Documentation)

Figure 6: Intel RealSense D455 RGB-D Camera

4.1.3. Display

The third critical piece of equipment is the display monitor. For this project, we have acquired a touch screen monitor that can be connected to a computer via an HDMI cable. We were able to get the monitor connected and test the touchscreen and speaker features which both work successfully. The monitor does require a driver to be configured. Figure 7 shows the monitor.

Figure 7: Touch Screen Monitor

4.2. Software and Libraries

In our project we make extensive use of existing libraries such as NumPy, OpenCV, Dlib, SciPy, PIL, and Matplotlib, to name a few. NumPy is a powerful library built on top of Numarray and Numeric libraries and aims to deliver numeric computing with vectorization, indexing, and array computing. OpenCV is an Open-Source Computer Vision Library. It contains many computer vision algorithms specializing in image processing and video analysis, and many other things. Dlib is a C++ toolkit that contains various machine learning algorithms and image processing tools to detect objects. SciPy is a powerful library that allows to leverage many algorithms, interpolation, and algebraic equation tools. PIL is short for Python Image Library – it allows us to convert frames and display output in the GUI. Matplotlib is a library with extensive plotting functionality, which we use to plot the output.

4.3. Software Architecture

The software architecture behind this project is not overly complex. For the software to work accordingly, we rely on the RGB-D camera for input. The camera is connected to a computer and captures video of the user. The video is returned to the computer in the form of frames. These frames are successions of still images. The program has a total of four classes and each class is responsible for a section of the program. For example, HRRR.py can be considered as the ‘main’ class and file. This is the file the user would run to launch the software. HRRR.py explicitly interacts with three other classes, and implicitly with another class along with a face detection data model. Figure 8 below shows an overview of the configuration.

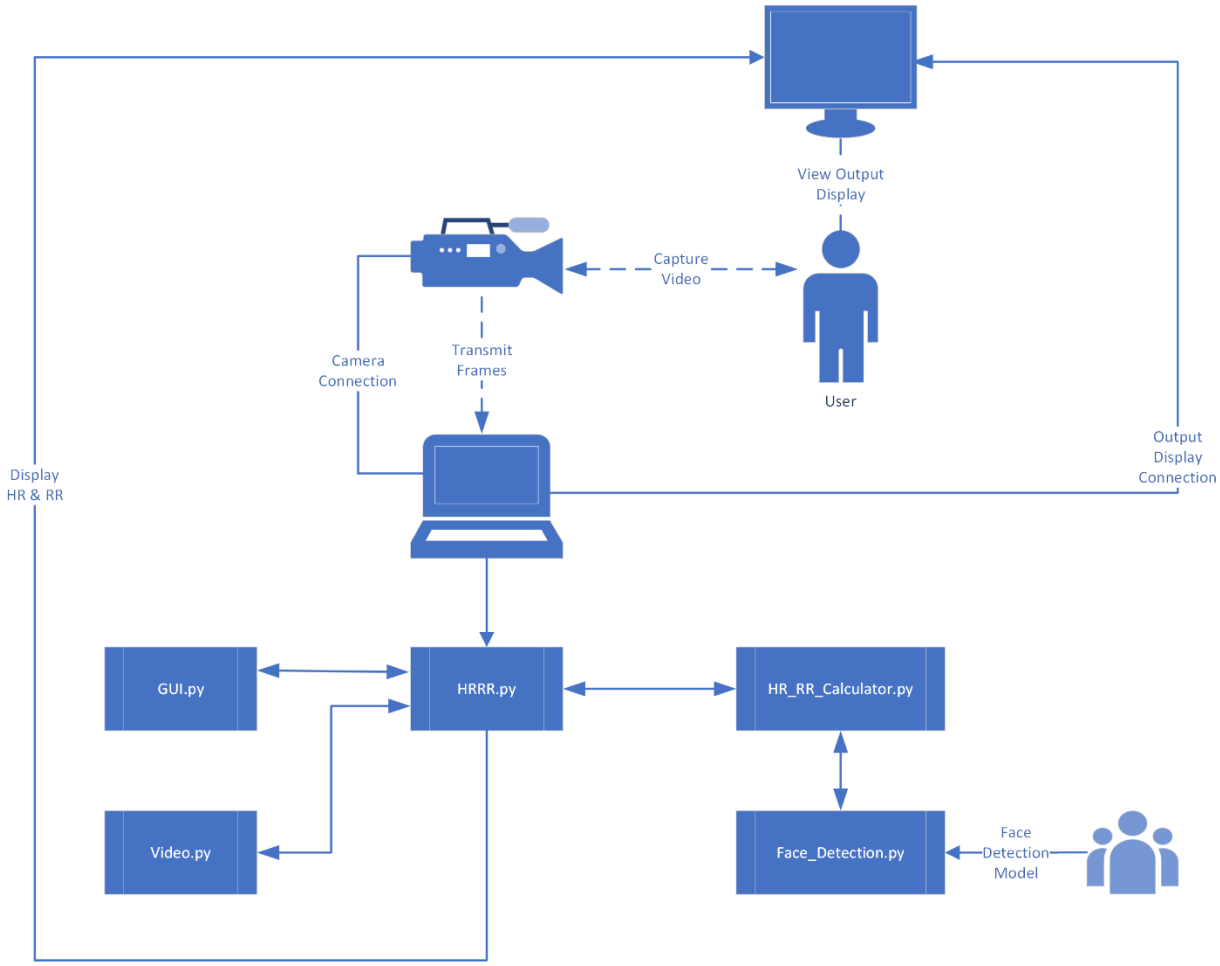


Figure 8: Software Architecture Overview

Both the classes and the logic are further explained individually in the sections below.

4.3.1. Video.py

The video class is relatively simple. The idea behind the class arose by following good programming standards. The goal was to have all related operations under one roof. Since the program uses OpenCV to capture video, there is a linear process of steps that needs to be followed. These steps identity the index of the camera being used (the default index is zero, but with multiple camera connections the index for the desired camera may or may not be zero), initiate the video capture process, read the frame, flip the frame, and stop the capture to release resources when the program ends. All these functions are grouped in the Video.py class so they can be reused as needed. This class also contains the *status* of the Start and Stop buttons from the GUI and this functionality will be further explained in HRRR.py. As shown in Section 4.3, Video.py interacts with HRRR.py. While the captured frames used for processing are passed to HRRR.py, and not back, HRRR.py makes calls to Video.py to set and get the status of the GUI buttons.

4.3.2. GUI.py

The GUI.py class is a static collection of utilities that can be reused over and over without having to write the same code more than one time. Our program makes use of Tkinter – a built-in Python library which allows us to create simple Graphical User Interfaces which can be customized in many ways to meet many modern and friendly UI design requirements.

The geometry that powers Tkinter is a simple grid starting from (0, 0) and continuing through (n, n). Let's call the grid G = [(0, 0) – (n, n)]. Just like with any grid, its rows and columns have additional properties such as width, height, span, and more. Every object created on the main grid inherits its own [(0, 0) – (n, n)] subgrid. For example, if a text label TL_1 was added to G at coordinates (1, 0) then TL_1 now has its own subgrid [(0, 0) – (n, n)]. We can add another text label, TL_2 to TL_1 at coordinates (1, 0) and this would not interfere with the text label added to G because TL_2 belongs to the grid of TL_1 and TL_1 belongs to the grid of G. Figure 9 below shows the Tkinter grid outline of our program.

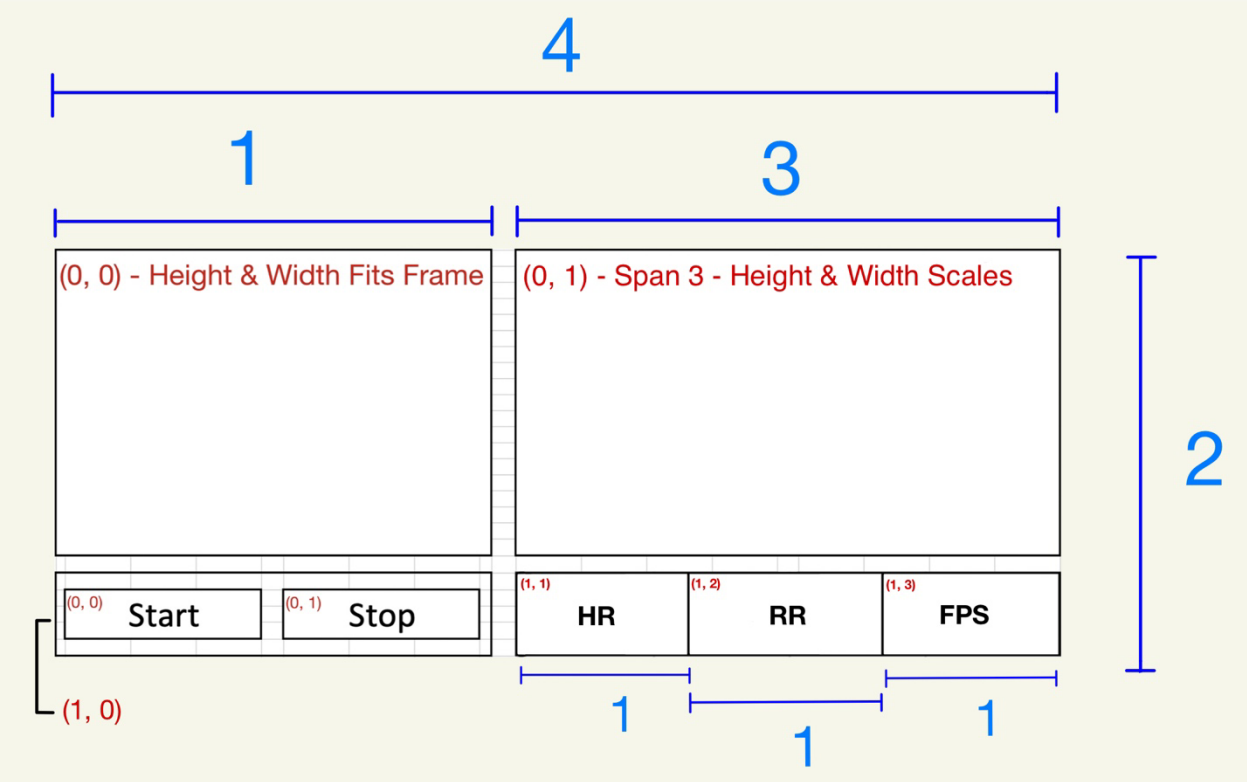


Figure 9: Tkinter Grid Outline

In the geometry shown above, it is evident that the start button and the label for the frame are both at position (0, 0). However, the frame belongs to the main window, while the start button belongs to another parent, thus their indexes and positioning are different because they belong to different elements.

The benefit of our class design is we followed the write once – call anything architecture (not run, but call). For example, the program contains various text wrapper labels around each

section. Since we have four such labels, we would have had to write the same statement four times, we would have had to write the grid position statements for each label separately, and so on. Since we grouped all the necessary GUI functions and calls in one class, we end up writing one parameterized function which can then be reused in the HRRR.py class. As a result, we're able to simply say button_text_label = create_button_text_label(p1, p2, .., pn). Should we have the need to edit the GUI elements in the future, we would only need to make the change in one place, regardless of the number of elements created. While the GUI.py class has no bearing on the heart rate and respiration rate calculations, it is critical to driving the UI and the GUI for the application.

4.3.3. HRRR.py

For lack of a better term, the HRRR.py is the 'main' file of this program. To run the program through an IDE or an interpreter you must run HRRR.py. This function is in the center and interacts with all other segments of the program, either explicitly, or in some cases, implicitly. This function is the ignition and the gas pedal – it starts and continues the process until it is interrupted, either implicitly or explicitly. The architecture of the class and its components are shown in Figure 10 below.

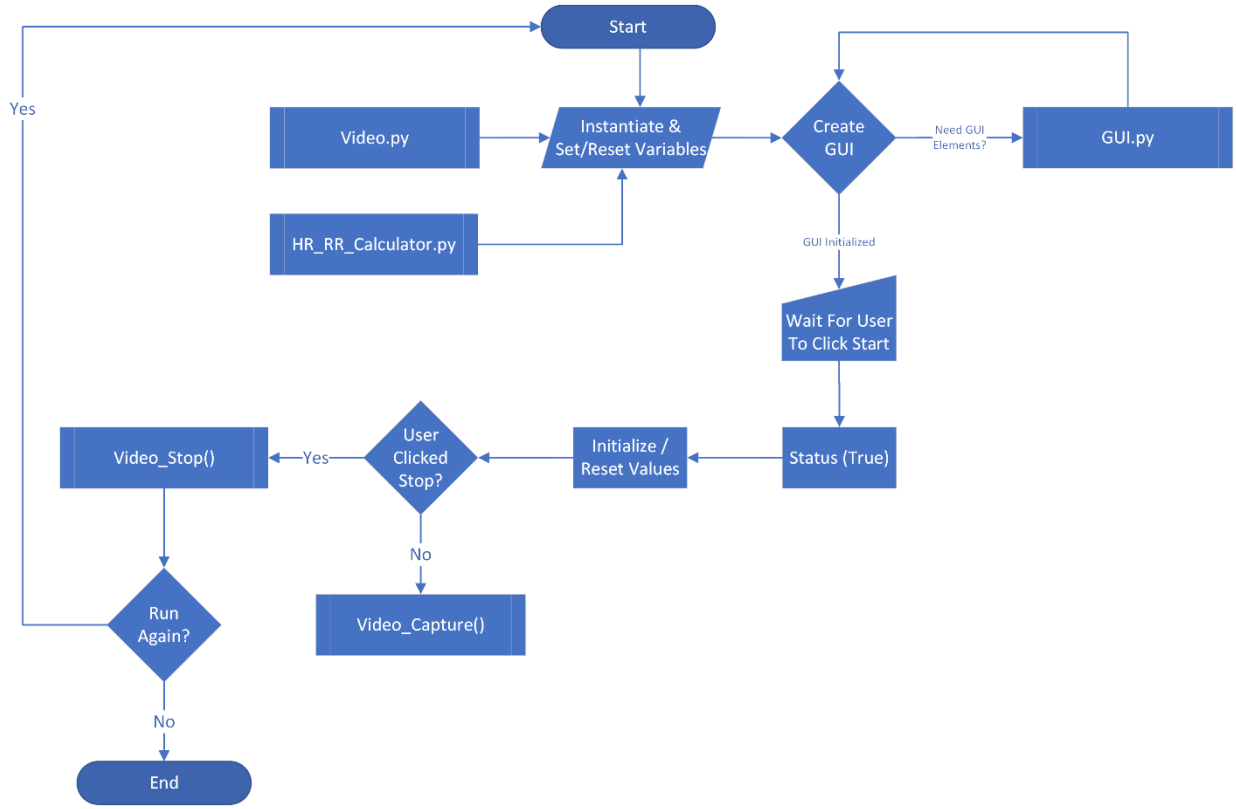


Figure 10: HRRR.py Architecture

While GUI.py contains all the necessary operations to create a GUI, it is HRRR.py that puts the GUI together and launches the program. When the program launches, it instantiates the Video.py and HR_RR_Calculator.py classes. Video.py contains the variable status. This is a special Boolean variable which dictates recursive calls of Video_Capture(). One of the things we

learned throughout this project is that Tkinter uses a Label to display anything from text to an image to a video. Since video is just a collection of still images, Tkinter uses recursion to drive the output. Video_Capture() is a recursive function that is called every 10 milliseconds and it is responsible for updating the still image from the video to the next image. In our case, it is responsible for moving from one still frame to another. Since the function calls itself 10 times each second, the still images displayed in the GUI make it look like the captured video is streaming, but we know it's moving fast through still images.

The status variable is used as an exit condition for the recursion. The Video_Capture() function only runs when its status is set to True. The status is set to True when the user clicks the Start Button. Similarly, when the user clicks the Stop button, the status is set to false. Thus, the Video_Capture() function knows when to and when not to capture and process frames. The architecture of the Video_Capture() function is shown below in Figure 11.

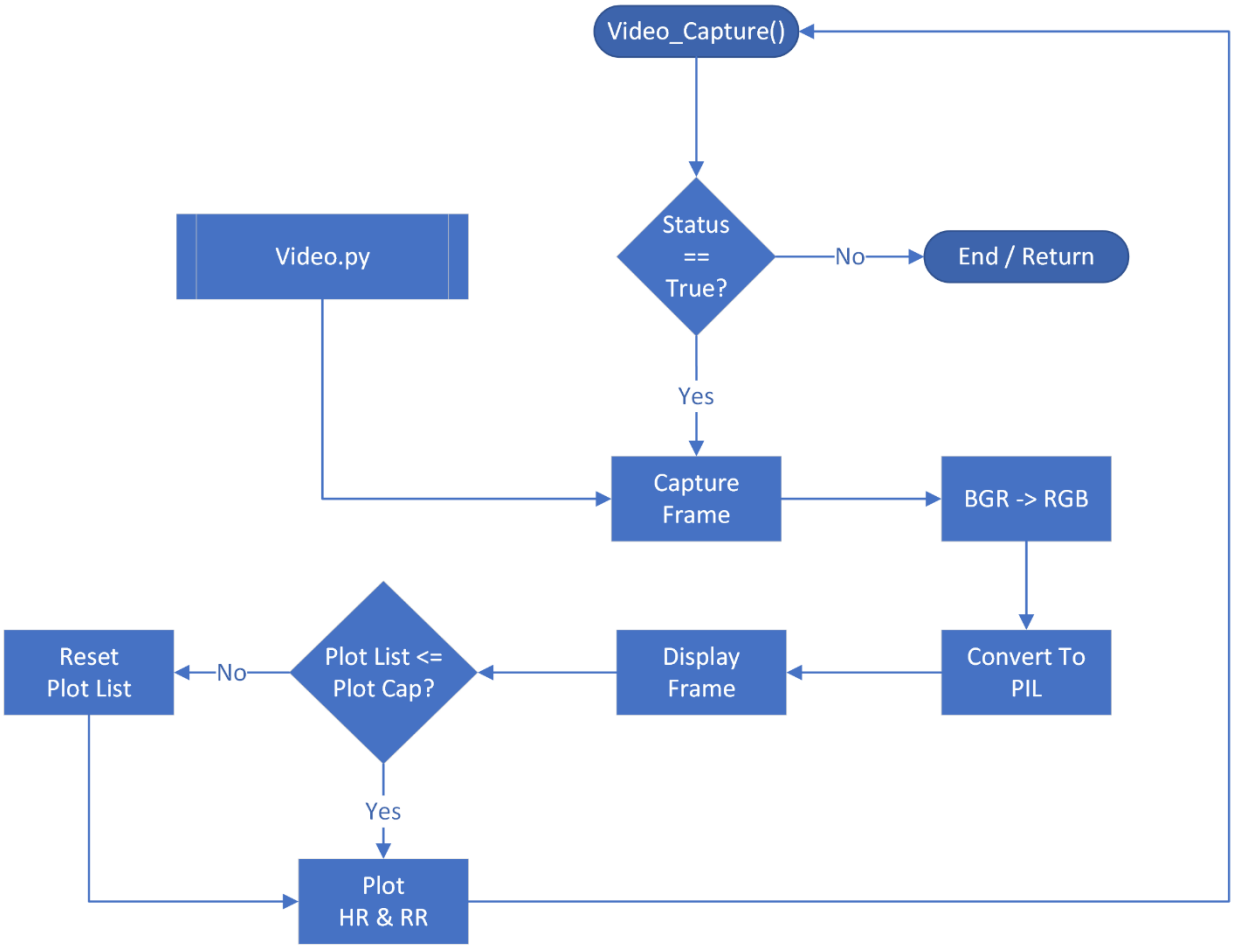


Figure 11: Video_Capture() Function Architecture

One thing to note about this process is what happens after the frame is captured. Since the frame is captured using OpenCV it is returned in BGR format. However, Tkinter does not support BGR and instead supports RGB. As a result, we use a built-in OpenCV method to convert the image to RGB. Then, using PIL, the Python Image Library, we can make the image compatible

with Tkinter. As such, the frame is then displayed in the GUI. The Video_Capture() function is also responsible for driving the plots on the GUI. The Plot List is a Python list of captured heart rates (beats per minute). Throughout our testing, we noticed as the list got larger, the frequency for the FFT function began to decrease, and so did the frame rate. In a matter of minutes, the frame rate was cut by half and was in continuous decline. As a result, we implemented a feature that resets the plot list after it reaches a certain amount of data points. On an average computer plot list may be set to 200, although that number will differ based on specs.

Throughout the development of the HRRR.py class, we noticed the program would often crash if no face was detected. The HRRR.py class contains a try/except block in the Video_Capture() function. If, at any moment during the video, a user leaves the confines of the camera such that they are no longer detected, the program will pause. It isn't until a user walks back into the view of the camera that the program will resume. It was important to add this feature because the program was not persistent and would often crash.

Lastly, when the user clicks Stop, the status is updated to be False – which terminates the recursive Video_Capture() call. The video capture resources from OpenCV are then released. As the program runs, the user may have a need to stop and start the program again and again. By clicking the Stop button, the program stops, but the GUI remains operational. When the user clicks Start, the program will begin anew. It will reset all previous variables, including the graphs, and continue with the operation

4.3.4. FaceDetection.py

The FaceDetection.py file is a utility class designed to identify the subject's face and a ROI on the subject's forehead. This class leans on previous work performed by the Python community to advance the field of image processing. Specifically, it leverages existing libraries, such as Dlib, and existing pre-trained data models, such as shape_predictor_68_face_landmark.dat.

To make use of the face landmark data model and the Dlib library, we must follow a set of rules. The process is divided into two steps. The first step revolves around Dlib.get_frontal_face_detector(). As noted by its name, this function belongs to the Dlib library. The frontal face detector returns a function known as *detector*, which is literally a function that takes input and returns output. The input must be a grayscale image, or in this case, a grayscale video frame, that contains the subject's face. The output is a list of objects where each object is a face detected in the frame.

The second step is to initialize the *predictor* function. This introduces a pre-trained facial recognition model that can detect facial features and plot facial landmarks. For this project, we used a 68 landmark data model which detects 68 facial landmarks, all below the forehead. This proved to be a challenge because the ROI is on the subject's forehead, and it forced us to improve our process to fit the constraints of the model. The flowchart below in Figure 12 is an overview of how the class operates.

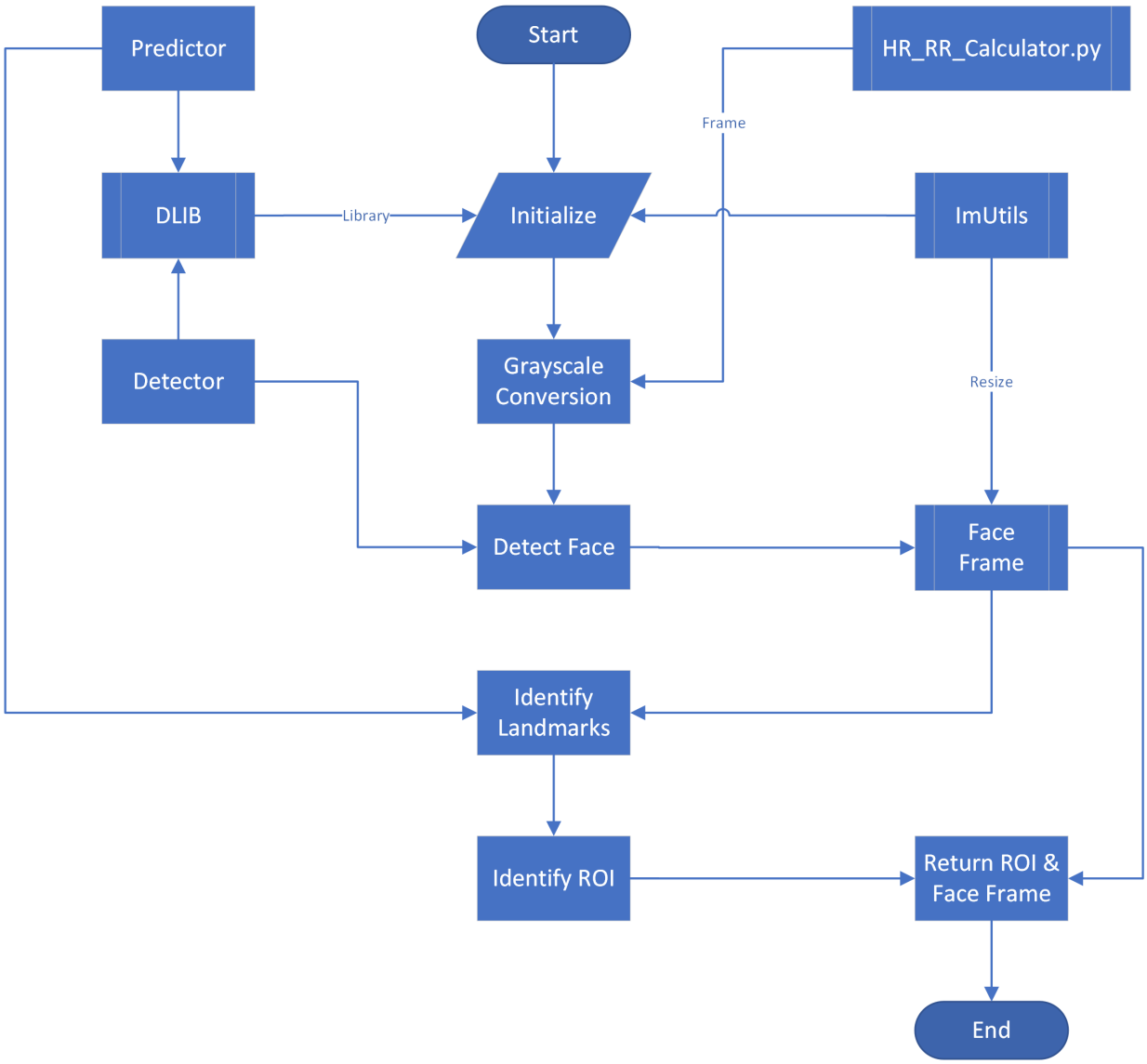


Figure 12: FaceDetection.py Architecture

To summarize the face detection process, the *detector* and *predictor* operations both belong to the Dlib library. We process an incoming frame from the HR_RR_Calculator.py class and convert it to grayscale. This allows us to make use of the *detector* to detect objects (faces) in the frame. And the *predictor* maps facial landmarks based on a pre-trained data model to the objects (faces) returned by the *detector*.

Face detection is just half of the battle in this class. The other half of the battle consists of resizing and keeping track of the subject's face in the frame regardless of the subject's position and location. We refer to this process as normalizing the face frame. Whether the subject is a foot away from the camera or 10 feet away from the camera, the goal is to extract the ROI from a leveled playing field. This means the size of the face must be near identical regardless of the subject's location and position away from the camera. To accomplish this, we make use of the ImUtils library. This allows us to constantly resize each frame to meet our needs.

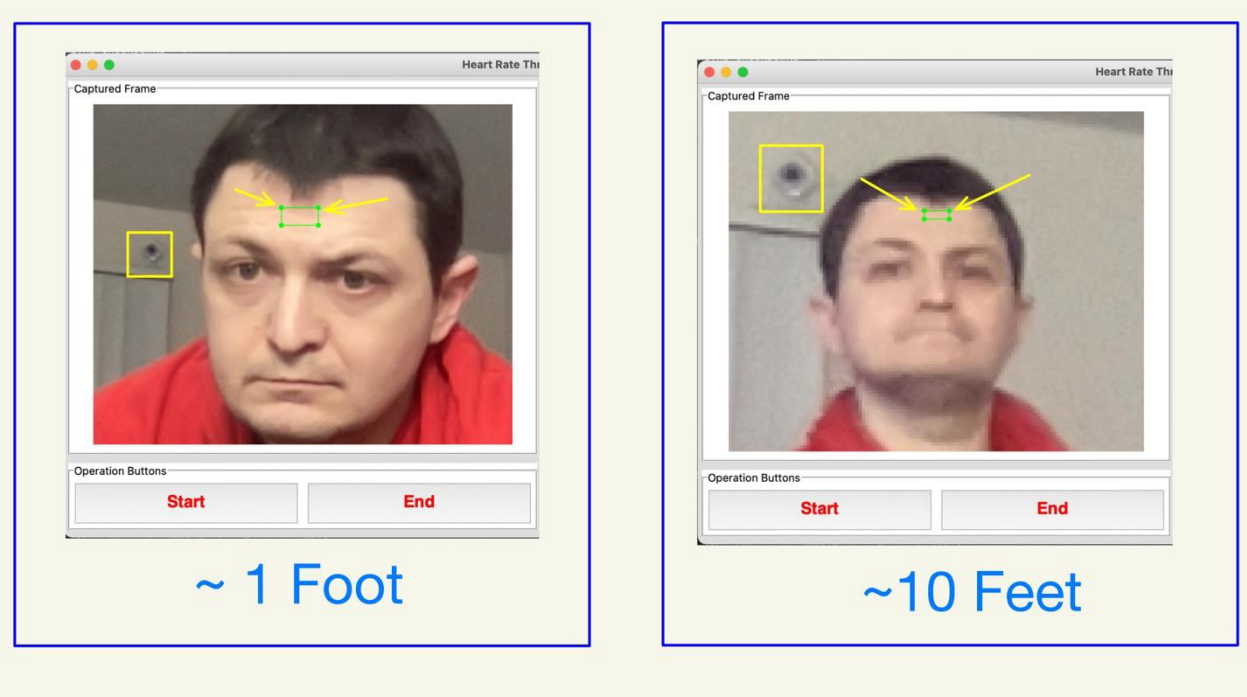


Figure 13: Resizing Frame of Subject's Face

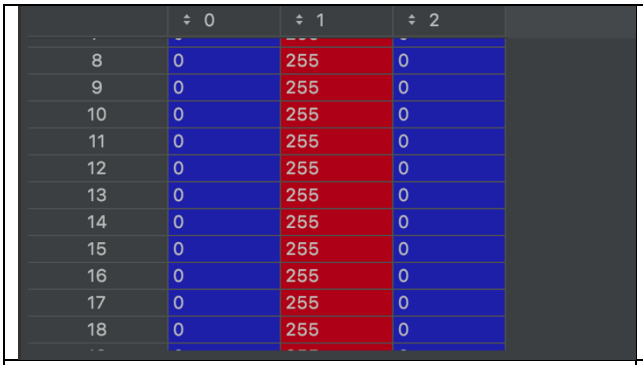
For example, Figure 13 above was captured while the subject was approximately 1 foot and 10 feet away from the camera. This was captured on a native camera on a MacBook. Note the object in the yellow square in both photos. In the photo on the left, the size of the camera is relative to the subject. However, for the photo on the right, as the subject moved further away from the camera, the FaceDetection.py class tried to keep its concentration on the subject all while resizing the frame. Had the subject moved back 10 feet, his face would be a lot smaller and the object in yellow would stay the same size. It's clearly visible that this class tries to resize the subject regardless of their distance away from the camera so that ROI readings can be performed under similar circumstances – such as the size of the subject's face. Also note the yellow arrows in both screenshots pointing to the ROI box. You can see that the ROI remained rather consistent regardless of the distance of the user. Had the actual frame been captured from 10 feet away the ROI would be much smaller.

4.3.5. HR_RR_Calculator.py

The HR_RR_Calculator.py is arguably the most important class in the entire program. This is the heart of the entire operation. It's important to understand how data makes its way through the class and how the data looks at various junctions. This function picks up right where FaceDetect.py left off. Before any calculation could be performed, the program needs to collect several seconds worth of data. We refer to this as the buffer. It's not practical to detect heart rate from a single frame. We need a time domain consisting of data points across several seconds worth of video before making a conversion to the frequency domain

As the function kicks off, there are two very important steps that drive the process. To be a valid time domain, this class marks a time when the ROI was captured and converts the ROI to a data point. For example, ROI_1 was captured at 0:00, ROI_2 was captured at 0:05, and so on. The

ROI data also needs to go through a transformation phase. HR_RR_Calculator.py returns ROI as a NumPy array, and it needs to be converted to a singular value. See Figures 14, 15 and 16.



	0	1	2
8	0	255	0
9	0	255	0
10	0	255	0
11	0	255	0
12	0	255	0
13	0	255	0
14	0	255	0
15	0	255	0
16	0	255	0
17	0	255	0
18	0	255	0

Figure 14: ROI Data Returned BY HR_RR_Caclurator.py

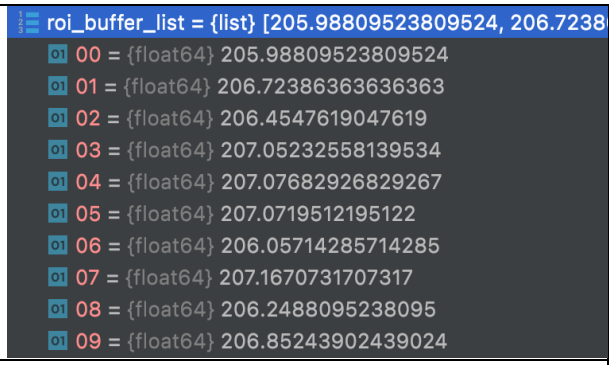


Figure 15: Converted Mean ROI Value For Each Frame

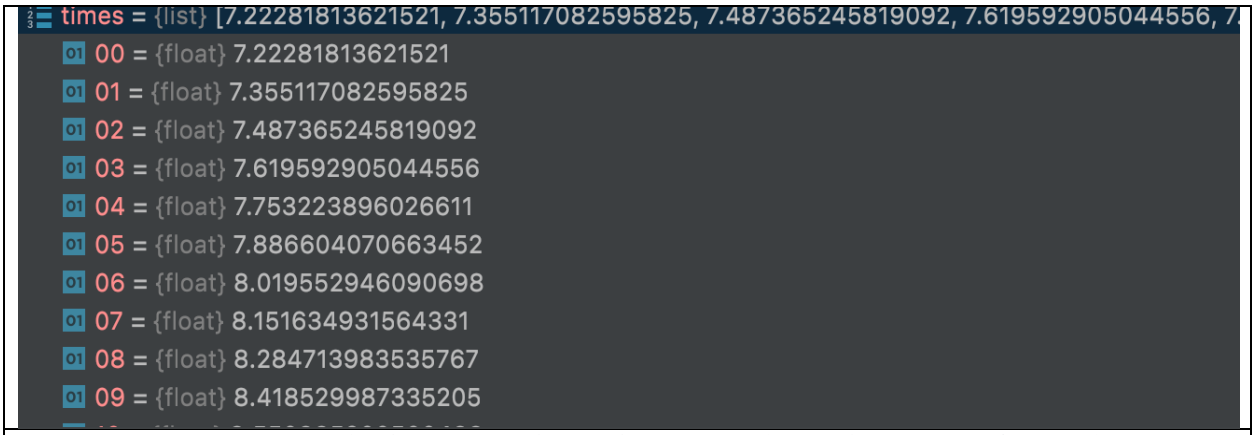
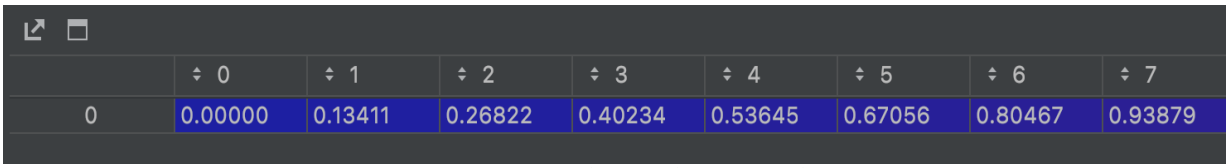


Figure 16: Time Value that Corresponds to Each Mean ROI – (Time Relative to Start of Program)

The program moves on to its next phase of the operation once the buffer requirement has been met. For best FFT function results, it is recommended that the size of time domain be a value that's an element of $\mathbf{2^n}$, thus the buffer for our testing was set mostly around 32 and 64 frames. The next step is to generate a list of evenly spaced time period data points between the first and the last time point of the captured frames. This list becomes important during the interpolation phase.

For example, if we had 100 data points, where the first frame was recorded at second 10, while the last frame was recorded at second 20, the goal is to generate 100 evenly spaced data points between [10, 20] seconds. We would end up with a list that read [0:00, 0.10, 0.20, .., 9.8, 9.9]

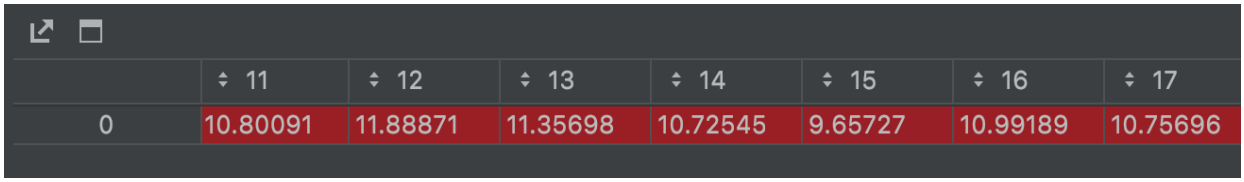
Figure 17 below shows an evenly spaced time period for an actual run where the buffer size was 64, and the time spanned from [0.00, 8.8.4490]



	0	1	2	3	4	5	6	7
0	0.00000	0.13411	0.26822	0.40234	0.53645	0.67056	0.80467	0.93879

Figure 17: Evenly Spaced Time Period

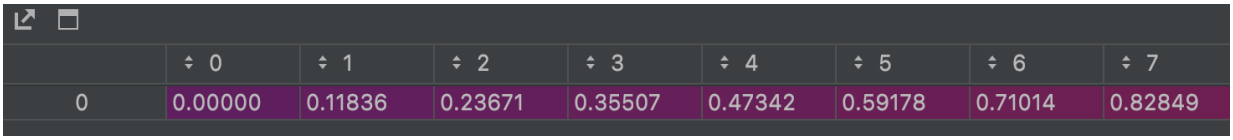
After enough ROI values have been collected in a list to meet the buffer condition, we use the SciPy library to detrend the values. That is, we remove any and all linear trends from the dataset. Figure 18 below shows the results after the detrend operation is complete.



	11	12	13	14	15	16	17
0	10.80091	11.88871	11.35698	10.72545	9.65727	10.99189	10.75696

Figure 18: Detrend Values Using SciPy Library

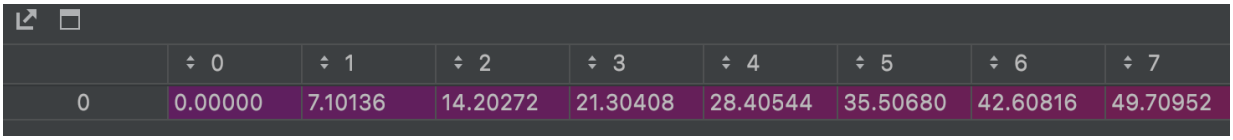
Before we move on to interpolation, we make use of a NumPy function to generate a list of frequencies based on this signal size and the frequency rate (where frequency is the delta in time divided by the buffer value (i.e., 8.44908.. / 64 = 0.13201)). The frequency data is shown in Figure 19 below.



	0	1	2	3	4	5	6	7
0	0.00000	0.11836	0.23671	0.35507	0.47342	0.59178	0.71014	0.82849

Figure 19: NumPy List of Frequencies

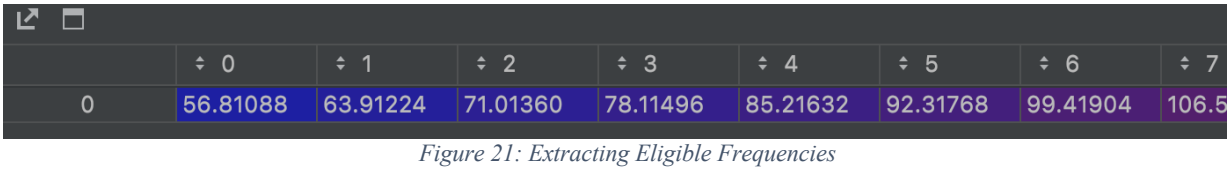
The scope for the frequency values are seconds, thus it needs to be multiplied by 60 to be converted to a value that is easier to recognize. The result is shown in Figure 20 below.



	0	1	2	3	4	5	6	7
0	0.00000	7.10136	14.20272	21.30408	28.40544	35.50680	42.60816	49.70952

Figure 20: NumPy Frequency List After Being Multiplied By 60

This is actually the first sign of the heart rate. The values shown above are stored in buckets. Whichever bucket the logic ends in, that represents the heart rate. We do one last operation to the frequency dataset. We extract frequencies deemed eligible – values between 50 and 180. The final frequency bucket list is shown in Figure 21 below.



	0	1	2	3	4	5	6	7
0	56.81088	63.91224	71.01360	78.11496	85.21632	92.31768	99.41904	106.5

Figure 21: Extracting Eligible Frequencies

More on that after interpolation and the actual Fast Fourier Transformation

Interpolation can be thought of as the process of estimating unknown values that fall between known values. One of the reasons why we cannot use our collected data (such as the time when the ROI was recorded) is because we need an evenly spaced domain. Interpolation allows us to look at collected data and make an educated calculation where the data point would've been had it occurred at the time in question. The graphs below in Figure 22 demonstrate the process of interpolation as it occurs during execution. The values in blue are actual recorded values, while the values in orange are interpolated values.

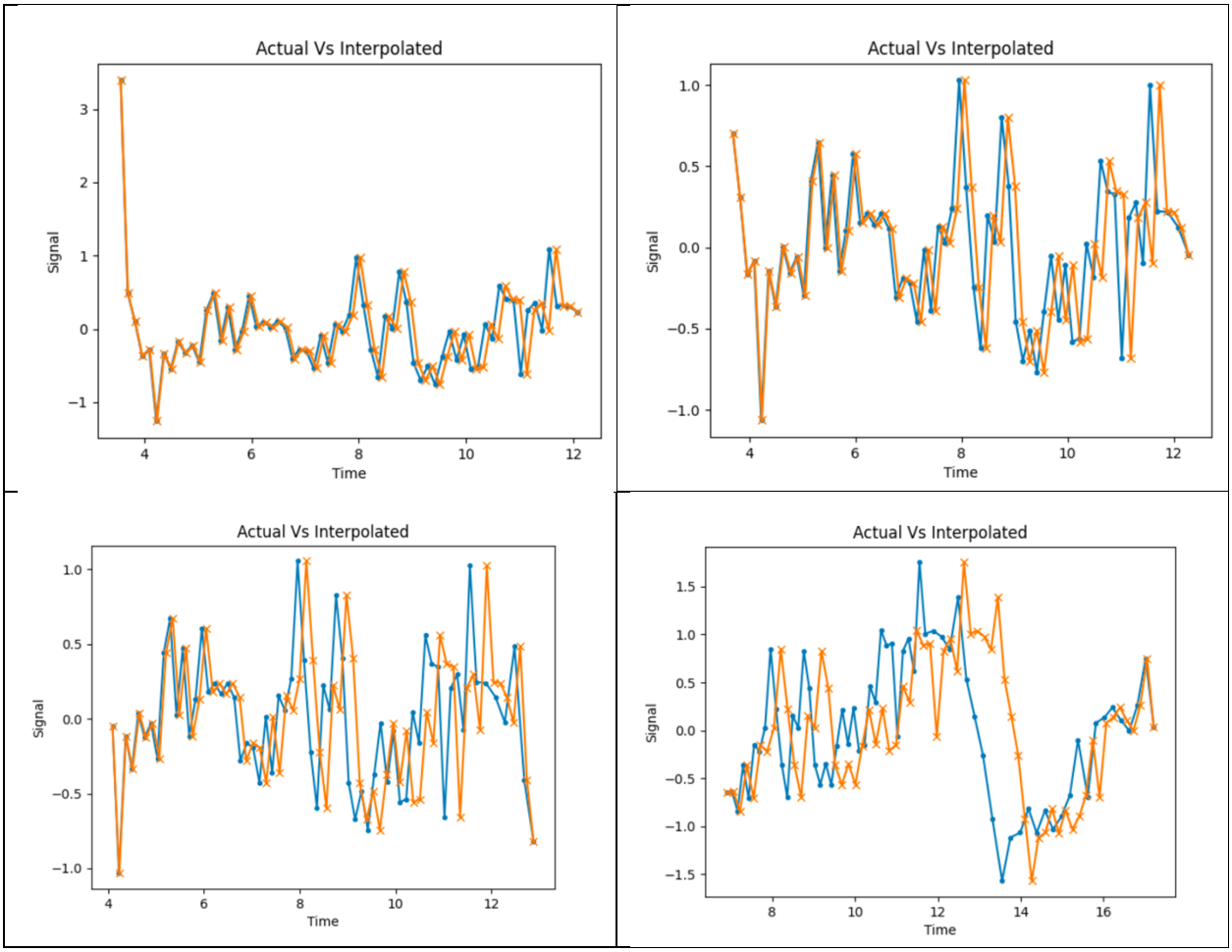
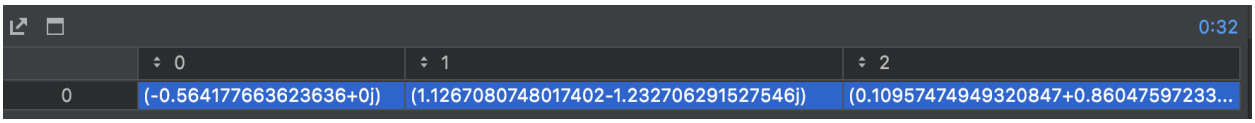


Figure 22: Examples of Actual vs. Interpolated Plots

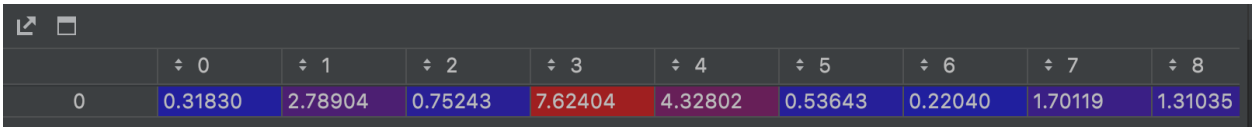
Interpolation is not perfect – and while we begin to deviate away on the X-Axis, the Y-Axis remains in a consistent state, which is good, because that's the signal. The interpolated data becomes normalized and that's when we convert it from the time domain to the frequency domain by performing a Real Fast Fourier Transformation. In this case, reals means the FFT returns a set of complex numbers, and we only want the positive values, hence half of the values will be dropped regardless. Performing a Real Fast Fourier Transformation saves us an extra step. The output of the FFT is shown in Figure 23.



	0	1	2
0	(-0.564177663623636+0j)	(1.1267080748017402-1.232706291527546j)	(0.10957474949320847+0.86047597233...

Figure 23: FFT Function Output

After squaring the data and taking its absolute value we end up with a more familiar dataset shown in Figure 24.

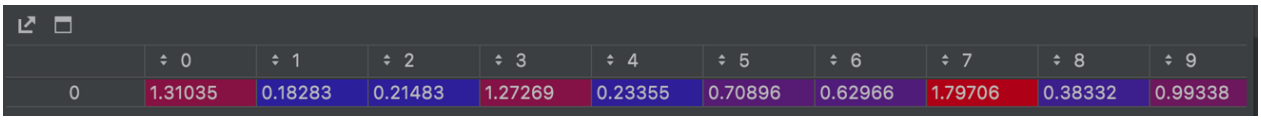


	0	1	2	3	4	5	6	7	8
0	0.31830	2.78904	0.75243	7.62404	4.32802	0.53643	0.22040	1.70119	1.31035

Figure 24: Squaring Data and Taking Absolute Value

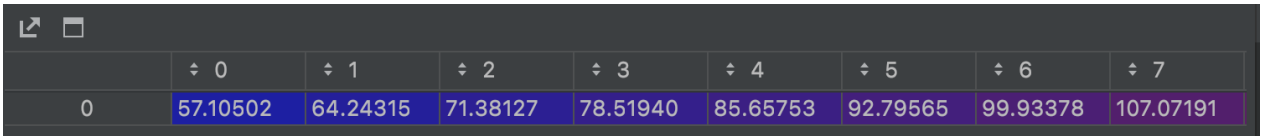
This is the frequency domain from the Fast Fourier Transformation. Recall that earlier we extracted the eligible frequency buckets. The eligible frequency range is defined as 50 – 180. We extract the corresponding index values of the FFT to end up with a one-to-one mapping of FFT-to-Frequency.

For example: Assume buckets of index [9, 19] fall into the eligible range. That means we now extract buckets [9, 19] from FFT to set up the mapping. We end up with the following FFT buckets in Figure 25 and frequency buckets in Figure 26, respectively.



	0	1	2	3	4	5	6	7	8	9
0	1.31035	0.18283	0.21483	1.27269	0.23355	0.70896	0.62966	1.79706	0.38332	0.99338

Figure 25: FFT Function Buckets



	0	1	2	3	4	5	6	7
0	57.10502	64.24315	71.38127	78.51940	85.65753	92.79565	99.93378	107.07191

Figure 26: Frequency Buckets

Even though we extracted index [9, 19], the index is moved down to start with zero. And since the mapping is one-to-one, the index becomes arbitrary.

Based on this information from the frequency domain we select the max value from the FFT bucket. That happens to be bucket 7 with a value of 1.79706. Bucket 7 from FFT corresponds to Bucket 7 from Frequencies, which is a heart rate value of 107. This summarizes the process of how the heart rate is extracted.

Throughout the development process we've experienced various spikes in the output of the heart rate that did not fit the circumstances of testing. As we continued our development effort, we've made various attempts to make a fix. It's likely that the spikes can be attributed to outside noise coming from various sources such as light, or even hair getting into the ROI. This is something that can be improved over time with better cameras and better image filtering processes.

5. Verification

Whenever a computer program is being developed, it is critical that the program be put through rigorous testing to verify it works as required. Throughout the testing cycle, healthy feedback can be provided to the development team to adjust and improve to the program. As a team, we tested our code very often as we would progress from one task to another.

The first step was to receive live video feed into the program through a camera using the OpenCV library. After getting successful live video, we began to develop the ROI algorithm. During this step, we needed to get the facial recognition landmarks plotted on the desired region of the user's face. At first, we had a large rectangle on the forehead, however, this gave a large variation in our data. Therefore, we went back and reduced the ROI down to a smaller square in the center of the forehead to gain more precise data. The continuous improvement of the ROI algorithm is shown below in Figure 27.

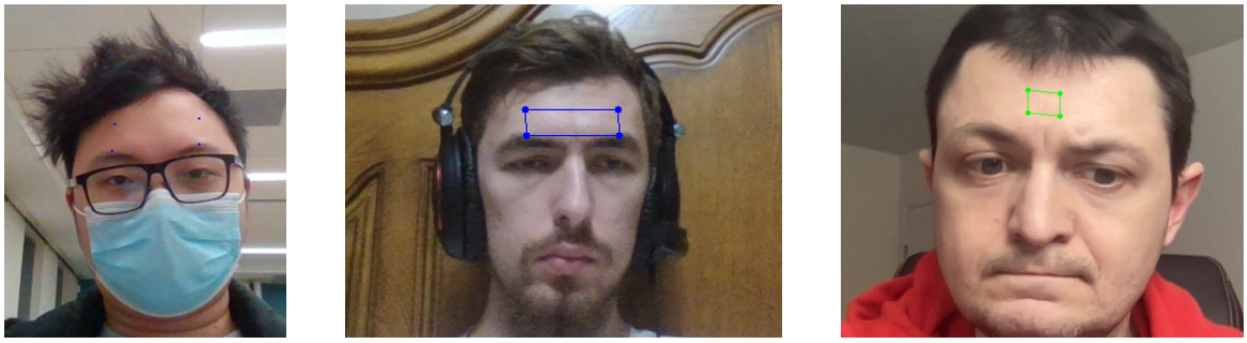


Figure 27: Continuous Improvement of ROI Algorithm

Once the ROI algorithm was completed, we needed to begin extracting values from the region and process them to display a HR and RR. As the algorithm was being developed, the team was actively testing on their respective computers. Eugene was working on a MacOS device, while Roman and Quan were working with Windows 10 devices. This allowed us to see that the program can work with two of the world's most popular operating systems. It is important to note that Quan was running Windows 11 and the Dlib library in Python did not work on the operating system. The RGB-D camera would also not work with Windows 11. Quan had to switch to Windows 10 because of this. Each computer varied in performance, which helped with broadening our testing results. Eugene's device was processing 9 frames per second, Quan was processing 14 frames per second, and Roman was processing 16 frames per second. Figure 28 below shows testing results of the raw signal extractions. The data was having several spikes and wasn't being consistent. It showed that our HR was 160 beats per minute in the first two graphs. In the third graph, the beats per minute jumped from 80 up to 160. In order to validate the program results, the data was compared live with the Apple Watch HR monitor during testing.

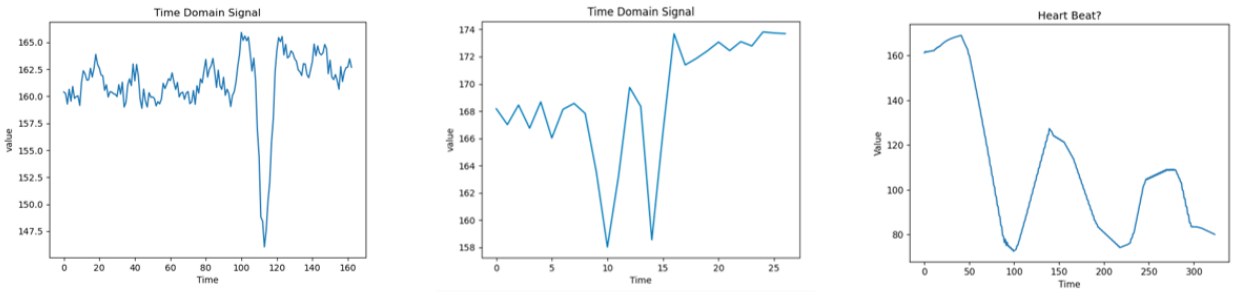


Figure 28: Inaccurate Testing Results

After further improvement of the logic to convert the raw ROI data into an accurate HR, we were able to get the following results shown in Figure 29. The data was again compared with the Apple Watch HR monitor during live testing. The testing results came out much better. There were still a few spikes that rarely occurred, but the data was now much more accurate. The first graph shows a HR of 84 beats per minute. The second graph was a very smooth and accurate test run that displayed a HR of 57 beats per minute. The third graph shows a HR of 70 beats per minute. We were able to extract a graph for the Apple Watch HR monitor during the test run to show a side-by-side comparison, as seen in in Figure 30 and Figure 31. The Apple Watch detected an HR of 56 beats per minute, and our program also detected a HR right at 56-57 beats per minute.

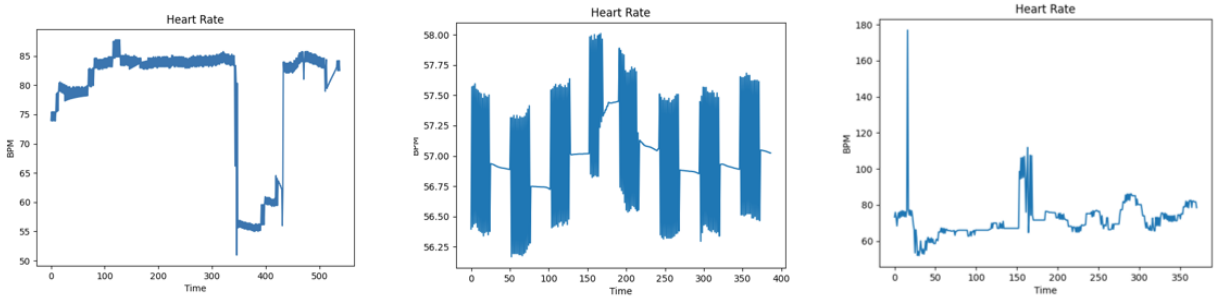


Figure 29: Accurate Testing Results

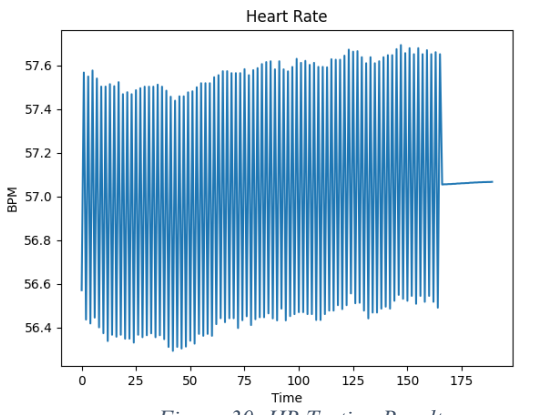


Figure 30: HR Testing Results

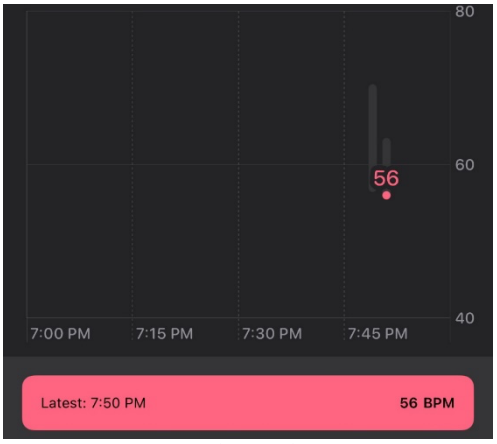


Figure 31: Apple Watch HR Monitor

After refining the raw signal processing algorithm, the next step to test was the GUI. The GUI development and improvement was much simpler than the previous parts of digging into the logic of the HR processing algorithm. There were issues with the ROI data not being cleared if the program were restarted by clicking the 'End' and then 'Start' buttons again. The second issue was the program crashing if the user's face moved out of view from the camera. Both of these issues were resolved, and the GUI is now working smoothly as expected. We did notice a decrease in performance of the image processing by about four frames per second due to this interface. Figure 32 shows the GUI.

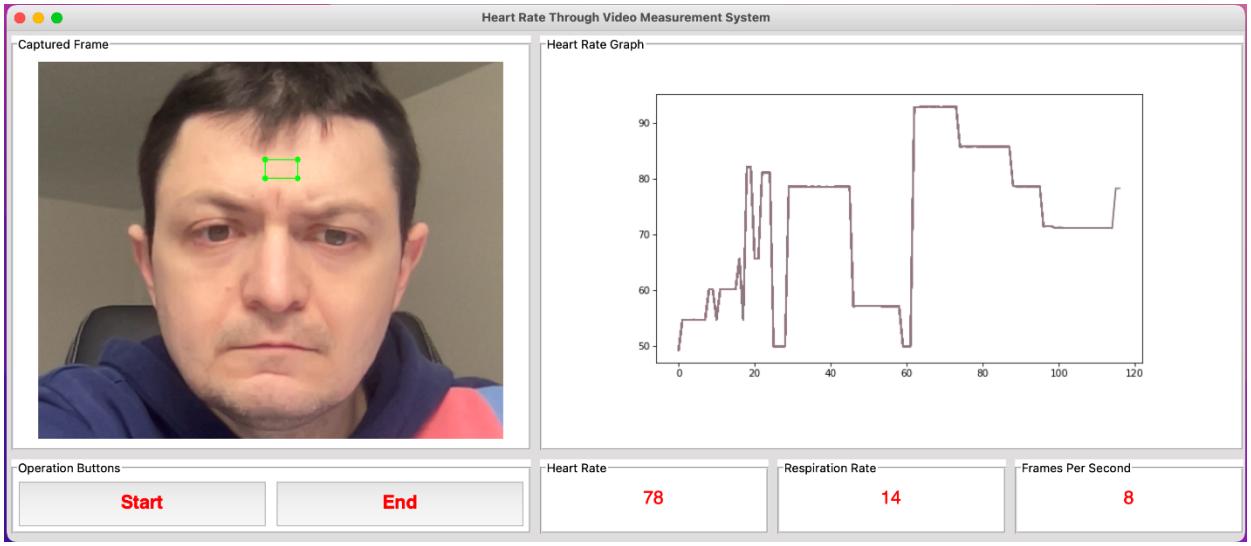


Figure 32: GUI Screen

As for testing the data for the RR, this was easier to validate than a heartbeat. To do this, we counted the number of breaths taken in a 20 second time period and then multiplied the result by three to obtain a breathes per minute number. According to Hopkins Medicine, the average RR for an adult person is in the range of 12 to 16 breaths per minute when they are at a resting state [21]. In Figure 32 above, the program detected the user's RR to be 14 breaths per minute. During this test run, the user recorded five breaths in the 20 second window, which comes out to be 15 breathes per minute.

6. Project Management

As with any project, it is essential to understand the scope, create a plan, delegate the workload, and have communication with team members. It is critical to manage the project in an orderly manner. Our team consists of three Cleveland State University senior students who have come together to unite their skillsets and tackle this project. Over the past two semesters, we have had opportunities to consult with subject matter experts, perform research on the subject matter itself, review current and previous efforts for similar projects, develop a plan and proposal for our idea, obtain vital pieces of equipment, and develop and test a Python program to produce the desired results.

6.1. Team Qualifications

Each member brings a unique perspective to the team. The members of this group are Eugene Dobryakov, Quan Nguyen, and Roman Melnik. The team has broad experience with popular programming languages such as Python and Java. On an individual level, each member brings a unique area of concentration, whether it be ETL, Python libraries, or Info-Sec. The diversity in the areas of concentration and skills of each team member will offer a broad perspective on design elements that will contribute to the success of the project itself.

Eugene Dobryakov

Eugene Dobryakov is a senior at Cleveland State University where he is pursuing a Bachelor of Science in Computer Science degree. Eugene also has an Associate of Applied Business in Computer Science/Software degree from Lakeland Community College. While attending school part-time, Eugene worked as a Junior Developer at The Sherwin-Williams Company for six years before accepting an offer from The Federal Reserve Bank of Cleveland. For the last year, he's been working as a System Specialist – Developer with the Research Department. His primary responsibilities include supporting the economists, research analysts, and community development specialists with their efforts to drive the economic policy for the Fourth District of the United States of America. He specializes in Python, JavaScript, as well as complex relational database programming with SQL, PL/SQL, T-SQL, and ETL tools. Eugene has a 3.69 GPA and is the group leader on this project.

Eugene has a wide range of experience that follows him, as he has been working as a developer for 8 years now. Eugene started working at The Sherwin-Williams Company in the Professional Apprenticeship program as an Oracle Applications Analyst. During this intern position, he developed and deployed internal applications for the business by using the Oracle Applications Express framework. During his time in the professional apprenticeship, he was one of three interns who received the Sherwin-Williams Cooperative education scholarship, alongside the 2014/2015 Sherwin-Williams excellence award. Eugene later moved up in the company to become a Junior IT Developer and continuing to grow and improve his skillset. In this position, Eugene provided support and continuous improvement for IBM ICM Cognos, which is an application used for budgeting, territory management, and incentive compensation. On top of this, Eugene also developed applications from scratch that helped the business run in a more efficient manner. He created an onboarding and offboarding process that was used for about 7,000 employees. This process reduced a large amount of manual work thanks to automation.

Eugene currently works at The Federal Reserve Bank of Cleveland. He is developing customized GUI Python applications to various teams internally. Eugene also continues to apply his knowledge of Oracle applications by developing custom database applications using the Oracle Application Express framework with features such as authentication, authorization, and dynamic JavaScript actions. Eugene also became a certified Amazon Web Services (AWS) Cloud Practitioner and is continuously exploring further cloud training.

Quan Nguyen

Quan Nguyen is a senior pursuing a Bachelor of Science in Computer Science degree at Cleveland State University. He is an undergraduate research assistant who specializes in optimization algorithms and works with Dr. Jingru Zhang of CSU Washkewicz College of Engineering. His research poster with Dr. Zhang was presented at Cleveland State University's Undergraduate Summer Research in 2021. He also works as a digital media assistant at CSU Cleveland-Marshall College of Law. Quan has good knowledge and understanding of C, Java, and Python, with concentration on Keras, Scikit-learn, and OpenCV in Python. He has a GPA of 3.91 and has been on the Dean's List every semester since Summer 2018.

Quan also has a Bachelor's degree in Design for Digital Media from the RMIT University in Vietnam. He is also well-versed with several tools from the Adobe Creative Cloud suite. This suite contains various image editing and processing tools such as Adobe Photoshop, Premiere, After Effects, and Audition. Quan also is skilled in 2D graphic design utilizing Adobe Photoshop and Illustrator from the suite.

In addition to these technical skills, Quan also has a unique belt of work experience that he brings to the team. During his time at the RMIT University of Vietnam, he worked as a designer in the student council to produce promotional content such as videos, photos, and 2D designs. He also worked as a design intern and continued to produce various forms of media for multiple purposes. Quan continues to work in the design area as he is currently a media assistant at CSU, where he is working closely with his supervisor to continue producing various promotional content material.

Roman Melnik

Roman Melnik is a senior at Cleveland State University. He is pursuing a Bachelor of Science in Computer Science degree. Roman works as an IT Security Analyst Intern and is part of the Sherwin-Williams Professional Apprentice program. He is on track to be a software developer at DebtNext Software upon graduation from CSU. He also has experience with Java and Python, as well as exposure to media processing stemming from his schoolwork. Roman has a GPA of 3.31 and brings a unique perspective to the team with his background.

Roman began working for the Digital Workplace team when he started his time with The Sherwin-Williams Company. During this role, he was involved with asset management. He made several contributions to the receiving and shipping processes in order to make them flow in a more efficient manner. In this role, Roman also utilized several back-end administrative management tools such as Active Directory, PowerShell scripts, and Bomgar. These tools were used to help end-users with resolving computer issues, deploying software, and assisting with licensing renewals.

Roman transitioned to the Information Security department at Sherwin-Williams, where he currently works on the Identity Management team as an IT Security Analyst within the Professional Apprenticeship program. He is working with his team to maintain and support the SailPoint IdentityNow Access Management platform within Sherwin-Williams to make sure that access requests are submitted properly. Whenever requests fail, he is responsible for investigating

the cause of the failure and getting the request resubmitted. Roman has had several opportunities to write Python scripts to help automate processes for his day-to-day tasks.

6.2. Task Assignment

As the Spring 2022 semester began, the team was ready to begin working on developing the Python program. Our research was completed from the Fall 2021 semester, our plan was created, we were well rested from winter break and ready to tackle the second stretch of senior design.

At the end of the Fall semester, we wrote a breakdown of each primary task, with subtasks that fell underneath each primary. This gave us a clear direction of where to start, and how to step forward. It is important that each team member has a fair chance to learn and contribute to the success of the project. We set up a GitHub repository with our code, where each of us could create our own fork, work on the code individually, and compare progress as we went along. This allowed each of us to be familiar with the code. The timeline shown in Figure 32 below illustrates the primary tasks and subtasks for this spring semester. The tasks and subtask assignments are as follows:

The first task was to obtain the RGB-D camera and extract the ROI.

RGB-D Camera to Extract ROI			
	Eugene	Quan	Roman
Order and Receive RGB-D Camera	N/A		
Explore and Configure RGB-D Camera	33%	33%	33%
Implement ROI Extraction Algorithm	20%	40%	40%
Pivot To Real-time	N/A		

The first subtask was to order and obtain the RGB-D camera. Unfortunately, this is where we experienced a delay in our plans. The estimated delivery date from the manufacturer was getting pushed back further and further due to the global supply chain disruptions. This forced us to cancel the order and find an alternative source which can deliver the camera in a timely fashion. We were able to place an order on January 27, 2022 and receive the camera only 4 days later on January 31, 2022 from a vendor on eBay who had the exact camera model that we were looking for in new condition for nearly the same price as the manufacturer. We focused on this primary task for five weeks.

The second task was to develop an algorithm to extract the raw signal.

Algorithm to Extract Raw Signal			
	Eugene	Quan	Roman
Develop Initial Algorithm	20%	40%	40%
Test And Improve Cycle	33%	33%	33%
Pivot To Real-time	N/A		

During this task, the focus was to extract data from the ROI and place it into a tine domain. This effort involved continuous testing and refinement in order to make extraction as quick, accurate, and efficient as we could. We focused on this task for four weeks.

The third task was to convert the time domain to the frequency domain.

Convert Time Domain to Frequency Domain			
	Eugene	Quan	Roman
Implement Fast Fourier Transform	80%	10%	10%
Testing and Improving	10%	45%	45%

This task involved a lot more math than the previous task. At this point in the project, we made a decision with the team for Eugene to take over the implementation of the FFT function and Roman and Quan to begin shifting their focus onto the final project report. Roman and Quan helped Eugene testing different approaches he was implementing and giving Eugene feedback on how the program ran on each of our computers. We worked on this task for four weeks.

The fourth task is to develop a graphical user interface (GUI).

GUI Development			
	Eugene	Quan	Roman
Design and Implement GUI	80%	10%	10%
Testing and Improving	10%	45%	45%

The third and fourth tasks are closely intertwined, therefore Eugene continued to have a greater focus on the development aspect, given his strong and vast knowledge of Python. Roman and Quan continued to help with testing and giving feedback, while also having their focus set on the report and presentation. We worked on this task for two weeks.

The fifth task is for supporting documentation. Our plan is to work on this task for a total of eight weeks. Two weeks were spent during the middle of the semester when we had the progress report due. The remaining six weeks are dedicated to the final month and a half of the semester as we will be working on the final report, poster, and presentation.

Supporting Documentation			
	Eugene	Quan	Roman
Lift and Shift System	50%	25%	25%
Report	10%	45%	45%
Poster	10%	45%	45%
PowerPoint Presentation	100%		

6.3. Timeline

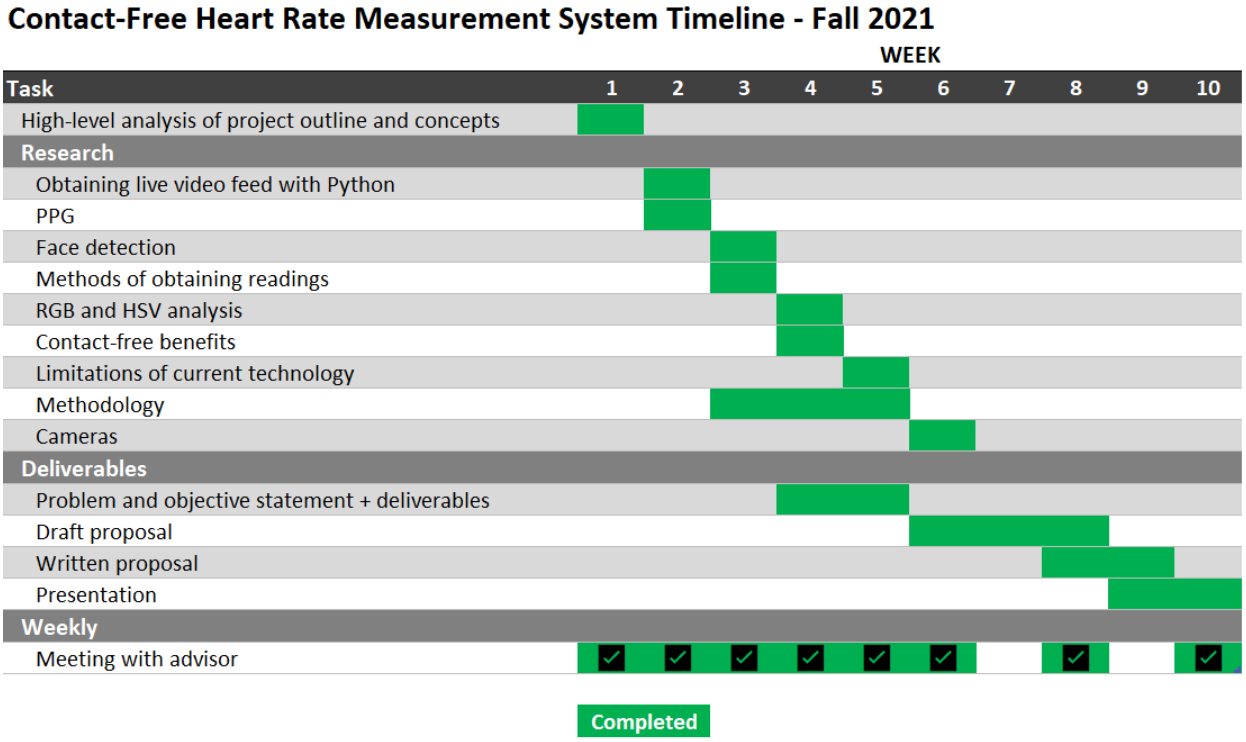


Figure 33: Gantt Chart - Fall 2021

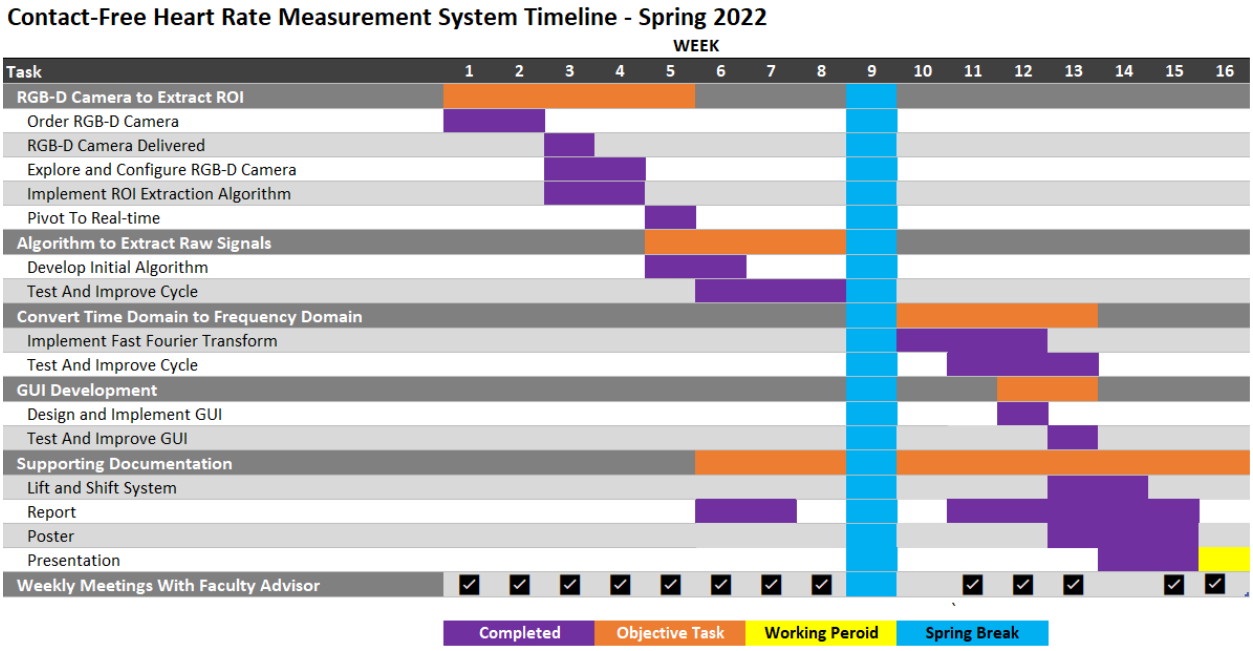


Figure 34: Gantt Chart - Spring 2022

Figure 33 and Figure 34 shown above illustrate the Gantt charts for the Fall 2021 semester and Spring 2022 semester respectively. When we submitted our final proposal at the end of the Fall

2021 semester, the Spring 2022 Gantt chart was only an estimation of our development plan. The Gantt chart for the Spring 2022 semester has since been updated to reflect how our development has gone throughout the semester. We have five primary tasks that are broken down to smaller subtasks. The last row represents all the meetings we had with our faculty advisor.

6.4. Deliverables

The final product is a contact-free lift and shift system capable of measuring accurate HR and RR in real-time. The system consists of an Intel RealSense D455 RGB-D camera to capture real-time video, a Python program to process real-time video, and an interactive touchscreen output monitor to display the HR and RR to the subject.

The system will be able to extract the ROI on the subject with high accuracy and precision. The system will function and work on popular operation systems such as Windows and MacOS. The system will be capable of running on a magnitude of machine specifications, not just high-end computing machines. The system could be used independently, or as part of a telehealth assessment. The system behaves like a function, that is it receives real-time video as input and displaying real-time HR and RR as output.

The lift and shift system means the system is not fixed to a single location. The system could be moved from point A to point B. So long as the camera and the display device are plugged into a power outlet, the system will be able to read the subject's HR and RR and display the output in a matter of seconds. Figure 35 below shows the final product.

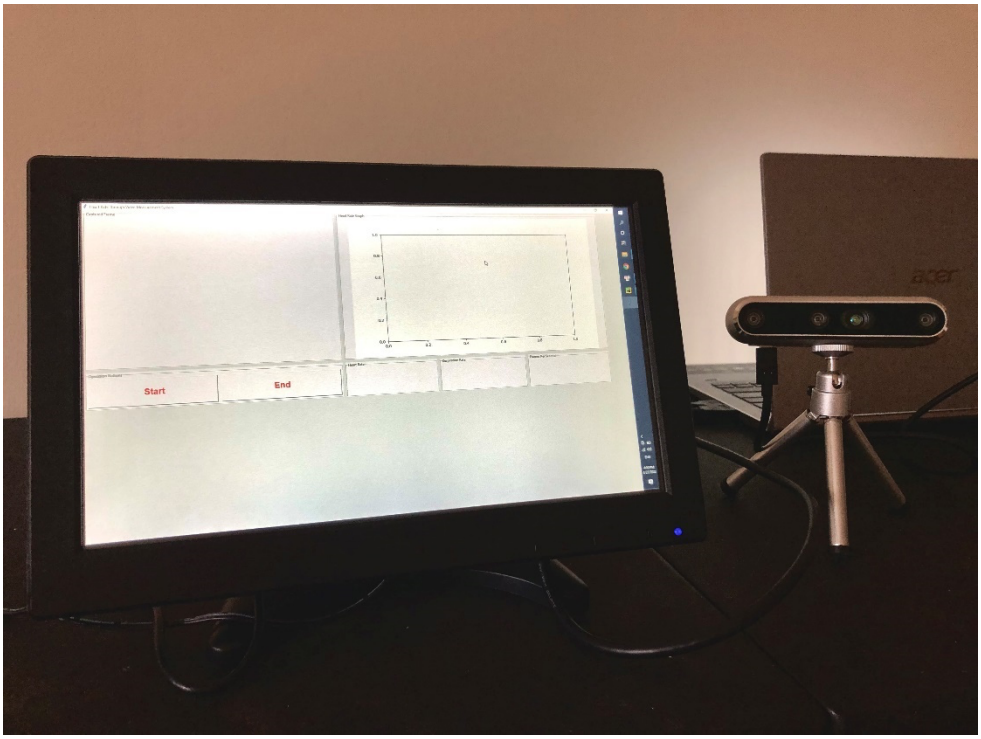


Figure 35: Touchscreen Monitor and Camera

6.5. Budget

There are only two items that we needed to purchase for this project: an RGB-D camera and an output display.

Item	Supplier	Quantity	Unit Price	Total
Intel RealSense Depth Camera D455	Intel	1	$433.94	$468.88
12 Inch Touchscreen Monitor HDMI 1920x1080 FHD IPS Screen	TOGUARDGO	1	$129.99	$140.39
			Total:	**$609.27**

7. Professional Awareness

As young professionals, we have a moral obligation to remain ethical in all aspects of this project. This system will be capturing the ROI from the subject's face. It will also be determining the subject's HR and RR. It's imperative that the ROI data remains private, and that the results are accurate. After each examination of each subject, the data is cleared out and ensures no data is saved anywhere in the program. This prevents any malicious actions of any adversary to the system ranging from stealing, corrupting, or modifying the subject's information.

Once the project is complete, it will have a foundation capable of determining the HR and RR in real-time. Processing data in real-time has an impact on the number of frames the system is capable of processing each second. As technology continues to evolve, the system could be scaled to reflect the new advancements. This presents an opportunity for life-long learning. Capturing more frames may improve the accuracy of the readings.

This project is an exciting opportunity to create a system capable of contributing to the well-being of the society. The COVID-19 pandemic presented a set of unprecedented challenges. As we adjust to the new normal, this project will make it possible to deliver remote care and assess vitals on the spot. Ultimately, it is systems like this that help trigger outside-the-box thinking and that can have a global reach.

8. Conclusion

There are several great benefits that come with this project. The fact that this system is contact-free and performs in real-time is a huge benefit in and of itself. It is also a portable system which can be taken to any location. Telehealth assessments will be greatly impacted in a positive direction as they will have an accurate and reliable system to obtain a patient's HR and RR. As a byproduct, this could result in a decrease in the 50% of people who would choose to delay their medical appointments, as described in the executive summary. This system would serve as an advancement in the medical industry, where the current leading methods for obtaining a HR are to press your fingers on the opposite wrist and perform a manual count or conduct an ECG test, which requires multiple electrode pads to make direct contact with skin. In addition, others such as athletes can use this system to accurately measure their HR during physical activity since wearable technology has inconsistency in measuring a HR. All in all, this is a versatile system with multiple benefits that can impact many people in many industries.

Acknowledgement

We would like to thank Dr. Almabrok Essa for his assistance and guidance on this project. We would also like to thank the Department of Electrical Engineering and Computer Science at Cleveland State University.

References

[1] *COVID-19 can affect the way your heart beats — here's what to look out for*. News. (n.d.). Retrieved April 5, 2022, from https://news.llu.edu/health-wellness/covid-19-can-affect-way-your-heart-beats-here-s-what-look-out-for

[2] Assessment of physiological signs associated with COVID-19 measured using wearable devices Aravind Natarajan, Hao-Wei Su, Conor Heneghan medRxiv 2020.08.14.20175265; doi: https://doi.org/10.1101/2020.08.14.20175265

[3] *COVID-19 Healthcare Coalition*, 11 Apr. 2021, https://c19hcc.org/telehealth/patient-survey-analysis/

[4] "Using Telehealth to Expand Access to Essential Health Services during the COVID-19 Pandemic." *Centers for Disease Control and Prevention*, https://www.cdc.gov/coronavirus/2019-ncov/hcp/telehealth.html

[5] Nazario, Brunilda. "How Coronavirus Is Transmitted: Here Are All the Ways It Can Spread." *WebMD*, WebMD, 18 Aug. 2021, https://www.webmd.com/lung/coronavirus-transmission-overview#1

[6] Center for Devices and Radiological Health. "Medical Applications of 3D Printing." *U.S.Food and Drug Administration*, FDA, 4 Dec. 2017, https://www.fda.gov/medical-devices/3d-printing-medical-devices/medical-applications-3d-printing

[7] "Want to Check Your Heart Rate? Here's How." *Harvard Health*, 17 Aug. 2021, https://www.health.harvard.edu/heart-health/want-to-check-your-heart-rate-heres-how

[8] "Electrocardiogram." *Johns Hopkins Medicine*, https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/electrocardiogram

[9] "Monitor Your Heart Rate with Apple Watch." *Apple Support*, 14 Oct. 2021, https://support.apple.com/en-us/HT204666

[10] Team, Heart and Vascular. "Your Fitness Tracker Isn't the Best Way to Measure Heart Rate." *Cleveland Clinic*, Cleveland Clinic, 18 May 2020, https://health.clevelandclinic.org/your-fitness-tracker-isnt-the-best-way-to-measure-heart-rate/

[11] Czeisler MÉ, Marynak K, Clarke KE, et al. Delay or Avoidance of Medical Care Because of COVID-19–Related Concerns — United States, June 2020. MMWR Morb Mortal Wkly Rep 2020;69:1250–1257. DOI: http://dx.doi.org/10.15585/mmwr.mm6936a4

[12] Chatterjee, Neal A., et al. "Admission Respiratory Status Predicts Mortality in COVID 19." *Wiley Online Library*, John Wiley & Sons, Ltd, 24 May 2021, https://onlinelibrary.wiley.com/doi/10.1111/irv.12869

[13] Drouin, Marc-Antoine, and Lama Seoud. “Consumer-Grade RGB-D Cameras.” *SpringerLink*, 12 Sept. 2020 https://link.springer.com/chapter/10.1007/978-3-030-44070-1_5

[14] Abay, Tomas Y et al. “Perfusion Changes at the Forehead Measured by Photoplethysmography during a Head-Down Tilt Protocol.” *Biosensors* vol. 9,2 71.27 May. 2019, doi:10.3390/bios9020071

[15] Verkruysse W, Svaasan d LO , Nelson J S . Remote plethysmographic imaging using ambient light. Optics Express, 2008, 16(26): 21434–21445.

[16] S. Kwon, J. Kim, D. Lee and K. Park, "ROI analysis for remote photoplethysmography on facial video," 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2015, pp. 4938-4941, doi: 10.1109/EMBC.2015.7319499.

[17] D. Zhang, J. Li and Z. Shan, "Implementation of Dlib Deep Learning Face Recognition Technology," *2020 International Conference on Robots & Intelligent System (ICRIS)*, 2020, pp. 88-91, doi: 10.1109/ICRIS52159.2020.00030.

[18] N. Boyko, O. Basystiuk and N. Shakhovska, "Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library," *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, 2018, pp. 478-482, doi: 10.1109/DSMP.2018.8478556.

[19] C. Zhu, H. Huang, H. Liu, C. Song, F. Ma, Z. Liu, On-line vibration monitoring and diagnosing of a multi-megawatt wind turbine gearbox, Editor(s): Philippe Velex, International Gear Conference 2014: 26th–28th August 2014, Lyon, Chandos Publishing, 2014, Pages 1089-1098, ISBN 9781782421948, https://doi.org/10.1533/9781782421955.1089.

[20] Shourjya Sanyal and Koushik Kumar Nundy, “Algorithms for monitoring heart rate and respiratory rate from the video of a user’s face,” IEEE Journal of translational engineering in health and medicine, vol. 6, pp. 1–11, 2018. https://www.rouast.com/pdf/rouast2016remote_a.pdf

[21] “Vital Signs (Body Temperature, Pulse Rate, Respiration Rate, Blood Pressure).” *Johns Hopkins Medicine*, 13 Dec. 2021, https://www.hopkinsmedicine.org/health/conditions-and-diseases/vital-signs-body-temperature-pulse-rate-respiration-rate-blood-pressure

Appendix A: Résumés of Team Members

QUAN NGUYEN

q.d.nguyen17@vikes.csuohio.edu / C: 216-543-3304
Cleveland, OH 44120

OBJECTIVE

Graduated from RMIT University in Vietnam with a bachelor degree in Design for Digital Media.
Currently studying Computer Science in Cleveland State University, USA.
Seeking Co-op/Internship in Computer Science where I could use my creativity and problem solving skills to pursue my future career.

EDUCATION AND TRAINING

Royal Melbourne Institute of Technology (RMIT) – Ho Chi Minh, Vietnam *2014 - 2017*
Bachelor of Design (Digital Media)

Cleveland State University, Washkewicz College of Engineering, Cleveland, OH *Expedted Graduation: May 2022*
Computer Science with Math Minor

PROGRAMMING SKILLS

- C programming in Linux OS
- Java
- Python (Keras, OpenCV, Scikit-learn)
- HTML, JS, CSS, PHP with MySQL

Languages: English and Vietnamese

OTHER SKILLS

- Microsoft Office (Word and Excel and PowerPoint)
- Capturing and editing photos and videos (using Adobe Photoshop, Premiere, After Effects, Audition)
- 2D Graphic design (using Adobe Photoshop, Illustrator)

WORK EXPERIENCE

Research Assistant / CSU Washkewicz College of Engineering, USA *5/2021 - now*
Working together with Dr. Jingru Zhang to research about optimization algorithms

Media Assistant / CSU Cleveland-Marshall College of Law - Ohio, USA *10/2018 - now*
Working closely with the supervisor to produce digital and print productions as well as website updates.

Design Intern / Dau An Viet - Ho Chi Minh, Vietnam *06/2017 - 10/2017*
Designed digital products that can be used on medias such as photos, videos, posters, banners, flyers and brochures

Desinger / RMIT University› Student Council - Ho Chi Minh, Vietnam *10/2015 - 10/2016*
Worked in Student Council to produce videos, photos, 2D designs which were used for promoting special events such as club days, orientations, music festivals, etc.

ACTIVITIES AND HONORS

Having cumulative GPA 3.91, Dean's List from Summer 2018 - Fall 2020, President's List in Spring 2019

Working closely with Cleveland-Marshall College of Law Marketing Department in big projects, including creating creative content for the Commencement Ceremony

Participated in 48h Film Project in Vietnam in 2017 and achieved second prize for good film trailer

Designer of BITCom 2017 - a competition about Business and Information Technology founded by BIS Club in RMIT University Vietnam

Participated in a Co-op program between RMIT University and Habitat for Humanity Vietnam to create a documentary video about their campaign in Tien Giang province

ROMAN MELNIK

6583 Harrow Dr.
Brook Park, OH 44142
216-280-0626 - roman247m@gmail.com

EDUCATION

Cleveland State University - Cleveland, OH **Expected May 2022**
Bachelor of Computer Science, Washkewicz College of Engineering
GPA: 3.31

COURSEWORK

- Data Structures & Algorithms
- Intro to Algorithms
- Database Concepts
- Operating Systems
- Internet Programming
- Mobile App Dev

SKILLS

- Java
- Python
- Microsoft SQL
- HTML
- CSS
- JavaScript

Hyland Hy-Tech Club – Westlake, Ohio **Fall 2017**

- Learned basic HTML, CSS, and JavaScript.

WORK EXPERIENCE

Sherwin-Williams - Cleveland, OH

IT Security Analyst Professional Apprentice - Identity Management **October 2021– Present**

- Investigate and fix daily failed access requests through the IdentityNow platform.
- Utilize tools such as Apache Directory, SQL Developer, and Postman to fulfill daily tasks.
- Contribute to a shared PA task of completing phishing email incidents using ProofPoint.
- Developed Python program for processing an Excel file with phishing email data.

Digital Workplace Professional Apprentice **August 2020– October 2021**

- Configured, deployed, shipped, processed, and managed devices for users.
- Utilized back-end administrator management systems to deploy software and permissions to users.
- Fulfilled support tickets for users experiencing software issues.

UA Transport LLC - Cleveland, OH **May 2018 – August 2020**

Billing Specialist

- Billed customers for all loads moved for the week.
- Calculated payroll for each driver based on loads moved for the week.
- Organized and filed invoices to keep accurate records.

Online Liquidation Auction LLC - North Royalton, OH **February 2016 – May 2018**

Warehouse Associate

- Unloaded trucks of merchandise with pallet jacks and forklifts.
- Inventoried items to be posted on auction by finding items online and taking photographs of the item's condition.
- Assisted customers with picking up their items and finding their way around the warehouse.

VOLUNTEER

ECC Church **January 2015 - August 2019**

Media

- Amplify sound, record sermons, present PowerPoints, lyrics, and video clips.

Eugene Dobryakov
(216) 704-9899 e.dobryakov@vikes.csuohio.edu

SUMMARY

- Experience supporting enterprise business critical applications using Python, SQL, PL/SQL, T-SQL, ETL, SSIS, and JavaScript. And robust experience with Oracle Application Express (Apex) Framework.
- Senior at Cleveland State University pursuing a BSCS and a graduate of Lakeland Community College with AAB in CS.

TECHNICAL KNOWLEDGE

- ∀ Software Applications: Oracle 11g, 12c, 18c, Oracle Application Express (Apex) Framework (4.2, 5.0.1, and introduction to 18), Toad, SSIS, SSMS, SQL Developer, BMC Control-M.
- Experience With: SQL, PL/SQL, ETL, Python 3.*, C, HTML, JavaScript, Web Design, and Software Development

EDUCATION

Lakeland Community College Kirtland, OH 2013 - 2016
Associate of Applied Business in Computer Science / Software Engineering

- Studied Computer Science, Dean's List, 3.65 GPA
- Member of Phi Theta Kappa Honor Society

Cleveland State University Cleveland, OH 2016 - Present
Pursuing a Bachelor of Science in Computer Science

- ∀ Senior standing, 3.69 GPA. Attend school part time while working full time.

EXPERIENCE

Federal Reserve Bank of Cleveland Cleveland, OH 2020 – Present
Systems Specialist - Developer

- ∀ Develop custom GUI Python applications from scratch to assist the Research department with their effort in supporting the Fourth District of the United States of America.
- ∀ Python projects include a custom program to download repos from GitHub, execute the code in the repo, and push the results back to GitHub. Other projects include using Selenium to pull JSON from web applications, parsing the JSON into SQLite database, and automating various SQL data loads.
- ∀ Develop custom database applications with Oracle Application Express. Features include full authentication and authorization, dynamic JavaScript actions, CSS, jQuery, Ajax callbacks to create custom ICS and EML files for Outlook, and more.
- ∀ Became a Certified AWS Cloud Practitioner (CLF-001). Continuously exploring further cloud training.

Sherwin-Williams Cleveland, OH 2016 - 2020
IT Developer

- ∀ Provide production support and create new functionality for IBM ICM Cognos, an application used for territory management, budgeting for sales representatives, and incentive compensation.
- ∀ Implemented new incentive compensation payout from scratch for the employees who primarily work at factories and upper management at Sherwin-Williams
- ∀ Designed and built onboarding and offboarding process from scratch for Management Compensation Plans, a population of more than 7,000 employees. Redesigned the existing lift-and-shift process in place since 2009 with a modern robust solution to drastically reduce amount of manual work in favor of automation.
- ∀ Certified in Transactional Lean.

Sherwin-Williams Cleveland, OH 2014 - 2016
Oracle Applications Analyst Professional Apprentice

- Developed and deployed internal applications using Oracle Application Express (Apex) Framework
- Became one of three interns awarded the Sherwin-Williams Cooperative Education Scholarship along with 2014/2015 Sherwin-Williams Excellence Award