Automotive UI

Cedric Wambe, Asmaa Alzoubi, Akeem Abrahams

|

Computer Engineering Technology April 9, 2020

Declaration of Joint Authorship

We, Asmaa Alzoubi, Akeem Abrahams, and Cedric Wambe, confirm that this work submitted is the joint work of our group and is expressed our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included. The work breakdown is as follows: Each of us provided functioning, documented hardware for a sensor. Asmaa Alzoubi provided VMA340 Heart/Pulse Rate Sensor. Akeem Abrahams provided MAX30100 Pulse Oximetry Sensor. Cedric Wambe provided AMG8833 Thermal Camera. In the integration effort Akeem Abrahams is the lead for further development of our mobile application, Cedric Wambe is the lead for the Hardware, and Asmaa Alzoubi is the lead for connecting the two via the Database.

Proposal

We have created a mobile application, worked with databases, completed a software engineering course, and prototyped a small embedded system with a custom PCB as well as an enclosure (3D printed/laser cut). Our internet of things (IoT) capstone project uses a distributed computing model of a smart phone application, a database accessible via the internet, an enterprise wireless (capable of storing certificates) connected embedded system prototype with a custom PCB as well as an enclosure (3D printed/ laser cut).

Intended project key component descriptions and part numbers

Development platform:

Sensor/Effector 1: MAX30100 Pulse Oximetry Sensor

Sensor/Effector 2: VMA340 Heart/Pulse Rate Sensor

Sensor/Effector 3: AMG8833 Thermal Camera

The objective for the User Interface of our mobile application was to build a platform that could be useful to paramedics. This user interface should contain features that are capable of assisting a paramedic in completing tasks during an emergency scenario, thus, should reduce the amount of work that a paramedic has to perform when going towards the scene.

To facilitate this, we included features in the mobile application that will allow a paramedic to, communicate with a patient through video chat, retrieve vital readings of a

patient in real-time, and track a patient's whereabouts automatically with the use of GPS tracking.

The design specifications that we decided on, is as follows:

The User Interface must:

- be easy to use and understand
- allow patient->paramedic communication
- able to automatically track the location of patient
- able to provide some vital information about the patient's physical and internal state to the paramedic, before arriving at the scene

Paramedic must be able to:

- View patient history
- Access vital information about the patient
- Retrieve client location in real time

Patient must be able to:

- Call and connect with a paramedic through the UI
- View the progress of paramedic as they head towards the scene
- Rate a paramedic

We will continue to develop skills to configure operating systems, networks, and embedded systems, using these key components to create a system capable of determining results for human vital readings in real-time, which includes: heart rate,

temperature, and blood oxygen rate readings. This data will be sent to a database platform known as firebase, over the network, where it will then be retrieved for display on our mobile application.

Our project description/specifications will be reviewed by, Dennis Kappen, ideally an employer in a position to potentially hire once we graduate. They will also ideally attend the ICT Capstone Expo to see the outcome and be eligible to apply for NSERC funded extension projects. This typically means that they are from a Canadian company that has revenue generating for a minimum of two years, and have a minimum of two full time employees.

The small physical prototypes that we build are to be small and safe enough to be brought to class every week as well as be worked on at home. In alignment with the space below the tray in the Humber North Campus Electronics Parts kit the overall project maximum dimensions are 12 13/16" x 6" x 2 7/8" = 32.5cm x 15.25cm x 7.25cm.

Keeping safety and Z462 in mind, the highest AC voltage that will be used is 16Vrms from a wall adapter from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will not exceed 20 Watts. We are working with prototypes and that prototypes are not to be left powered unattended despite the connectivity that we develop.

Executive Summary

Throughout the development of the project, the main goal was to create a simple, compact, user-friendly system which serves to reduce the work-load of a paramedic while heading towards an emergency scenario. With this in mind, we designed a system which is capable of providing some helpful information about the patient, that will make life just that much easier for a paramedic, and in some cases, the patient as well.

This system consists of three sensors, an embedded system, a database over the network, and an android application, implemented/developed by the team. The role of the three sensors are to provide vital information about the patient, and save this information automatically in the database by taking advantage of a network. These sensors function by outputting data about a patient who comes in contact with these sensors. This data provided by each sensor can be processed to, determine the temperature, heart rate, and the oxygen concentration in the blood, of a patient. In order to get this data, a patient must be in contact with the sensors, that is, all they need to do is place one of their fingers on each sensor, and the rest will be handled by the sensor and embedded system. For the user interface, our android application was designed to have these following capabilities:

- Video chat capability
- Retrieve patient information for display
- GPS tracking capability
- Update/Modify patient information as needed

Our goal for this user interface with regards to the paramedic, is to create a platform where paramedics can, easily and efficiently access the location of the patient by automatically tracking the location of the patient, provide visual communication that allows a paramedic to observe the state/condition of a patient, enable accessibility of a paramedic to vital information about the patient by displaying this information in a comprehensive format. With these functionalities, a paramedic will be better prepared for the emergency situation, since they won't be required to ask a patient for their current location, and they will have some information about the state of the patient before arriving at the scene. This way, they can focus on the patient and be better prepared for their treatment upon arrival at the emergency scene.

Our goal for this user interface with regards to the patient, is to create a platform where patients can, visually communicate with a paramedic, track a paramedics progress when travelling towards the scene, and provide options that allows a patient to type in his/her home address if emergency location is not their home location. These functionalities will serve to lessen the stress on a patient while communicating with the paramedic, since they won't be burdened with a requirement to provide their current location, or to possibly provide a description of their current state. The GPS tracking and the visual capability of the video chat app will remove these unimportant roles that a patient has to perform. By giving a patient the capability to track the progress of the paramedic when heading towards their location, this will also help to reduce stress on the patient since they will be relieved knowing that a paramedic is that much closer to providing them with the treatment they deserve.

We believe that this system will contribute greatly to the medical field, since this will have a positive impact on the responsiveness of paramedics towards an emergency scenario. Thus, investing in this product should have good prospects for success given time and effort.

Contents

List of Figures

1.0 Introduction

We plan to build a platform specifically for the use of paramedics. The main goal of this system is to reduce the number of roles that are required by a paramedic during an emergency event. We will accomplish this by building a User Interface through the implementation of an android application to, enabling video-chat communication in order to help a paramedic have an idea of the current well-being of the patient, enabling GPS tracking so that the paramedic may be able to quickly retrieve the current location of the patient, creating a means of display for a paramedic to conveniently observe a patients vital readings so that they may be able to have an idea about the internal state of the patient, and advising how to proceed. These functions will allow a paramedic to execute their roles more quickly and easily, and they will be able to gather accurate information regarding the condition of the patient, before arriving at the emergency scene. For the hardware side, we will be implementing the use of three sensors, the MAX30100 Pulse Oximetry Sensor, the AMG8833 IR Grid Eye Thermal Camera, and the VMA340 Heart/Pulse Rate Sensor. These sensors will be implemented as follows: MAX30100 Pulse Oximetry will be used to detect blood oxygen levels in a patient, VMA340 Heart/Pulse Rate will be used to detect the heart rate in the body in Beats per Minute, the Thermal camera sensor will be use to measure temperature by scanning the patient's affected body area and visually represent it on the screen. This data will be sent to firebase, a database over the network, and will be retrieved in the android application for the use of the paramedic.

1.1 Scope and Requirements

Throughout the first half of the development of the project, there were both hardware and software guidelines necessary for the successful completion of the project.

The hardware requirements were as follows: configuring a raspberry Pi by installing an operating system, establishing a stable connection to the internet, enabling required functions such as I2C, SPI and VNC to retrieve data from the I2C and SPI interface, and enabling remote access to the RPI via VNC. Additionally, we are required to design and test PCB sensors, test connections of PCB with Raspberry Pi and sensor, design an enclosure for the Raspberry with integrated embedded system, and finally, test the functionality of the sensors through programming.

The software requirements for the android application were as follows: User interface should be designed in a simple format to reduce confusion as much as possible for a paramedic and especially for the patient. Paramedic should be able to track patient automatically. Paramedic should be able to view vital information about the patient on the app. Paramedic should be able to search his/her history of completed emergency scenarios. Patient must be able to connect through video chat with a paramedic. Patient should be able to view the progress of a paramedic as they head towards the scene.

Throughout the planning and development stage, there were milestones set in order to ensure the completion of required tasks before the second half of development. We were tasked to, build a Gantt chart to set out all the milestones and due dates for completion of these milestones, create test plans for the app which is an evaluation

sheet for app functionality, conduct presentations and attend interviews to advertise our app.

For the second half of development, we will be required to, integrate our PCB's into one PCB, test connections of the PCB to the Raspberry Pi, test Sensor connections with the Raspberry Pi, test sensor functionality through programming, write a program to send data to firebase, and finally build a new enclosure for the RPI and PCB with all three sensors integrated.

Report

/1 Hardware present?

/1 Introduction (500 words)

/1 Scope and Requirements

/1 Background (500 words)

/1 References

2.0 Background

We would like to thank Haki Sharifi, professor of Humber College and mentor for android application development, and Dennis Kappen, the collaborator for our project, for their support and feedback throughout the course of software and hardware development. Professor Haki Sharifi was especially helpful in providing us with innovative ideas and practices to improve how we approach aspects of software development. Due to this, we were able to efficiently design a User Interface that meets all the specified requirements.

Throughout the development of our project, we used various platforms/third party software to execute main tasks, working towards the completion of the project. Android Studio, an android application development platform, was used to design, program and simulate the functionality of our android app. Fritzing, a hardware design platform, was used to design the PCB, schematic, and bread board layout for the sensors selected by each member of the team. Inkscape, a software tool useful for creating custom hardware designs and tools, was used to design the enclosure for the microcontroller with embedded system attached.

With regards to the sensors chosen, each sensor was chosen specifically to accomplish a task that would be useful to a paramedic in an emergency setting. After some research about sensors currently used in the medical field, we were able to select three sensors that would meet this criterion.

2.1 MAX30100 Pulse Oximetry Sensor

The MAX30100 Pulse Oximetry Sensor, selected by team member: Akeem Abrahams, is a device used for measurement of heart pulse rate and oxygen concentration in the blood. The sensor consists of two LEDs, modifiable optics, low noise signal processor which detects heart pulse rate signal (Strogonvs, 2017). This sensor communicates with the Raspberry Pi 3 B+ Microcontroller using I2C. For this project, only the blood oxygen detection function of the MAX30100 will be used. This sensor detects the blood oxygen concentration level in the body by measuring the amount of red light and infrared light that is absorbed by the tissue of a finger that has been placed on it (Strogonvs, 2017). Depending on the quantity of oxygen in a human body, the ratio of absorbed red light and infrared light will be different (Strogonvs, 2017). This ratio, multiplied by a 100, represents the blood oxygen level in the blood hemoglobin of a human (Strogonvs, 2017). The blood oxygen rate can be represented as a formula: Blood Oxygen Rate % = red light / infrared light * 100. This sensor can be used for wearable devices, fitness assistant devices, and medical monitoring devices (SPO2 Sensor MAX30100 Pulse Oximeter heart Rate Sensor Module, n.d.).

2.2 AMG8833 IR Grid Eye Thermal Camera

This sensor, selected by team member: Cedric Wambe, is an 8x8 array of IR thermal sensors from Panasonic. It measures temperatures with a range from 0oC to 80oC or 32oF to 176oF having an accuracy of +-2.5oC which is 4.5oF (Miller, n.d.). The SciPy python library provided help to process the image. Also, the library helps in getting some nice image result.

When connected to the raspberry pi, the sensor will return an array of 64 individual infrared temperature which will be read through the I2C interface. This sensor is a great add on to project which requires visually detecting heat instead of just getting the values. Like is the case with this current project, the sensor will be used to scan the body of the patient and report which part of the body is more affected and how hot it is. Also, this sensor can be use as a human detector as it has the capacity to detect humans from a distance of up to 7 meters (Miller, n.d.) which is up to 23 feet. Doing this at a rate of up to 10Hz. In addition, the sensor has an interrupt pin than can be invoked when a certain level of pixel is reached. This level could be below or above depending on pre set value. The sensor can be connected to the raspberry pi through 3V or 5V; the 3.3V regulator level that comes with the sensor will be able to adjust the voltage accordingly.

The following pins are used to connect the sensor to the raspberry pi.

Vin: power pin which will take the input voltage and convert it using the voltage regulator to 3.3V used by the sensor.

3Vo: the output voltage from the regulator which is 3.3V.

GND: is the common ground

INT: the interrupt pin which can be use to detect motion or change.

SDA: I2C data pin that connects to the data pin 3 of the raspberry pi.

SCL: I2C clock pin that connects to the clock pin 5 on the raspberry pi.

2.3 VMA340 Heart/Pulse Rate Sensor

The VMA340 Heart/Pulse Rate Sensor, selected by team member: Asmaa Alzoubi, is a device used for measurement of heart pulse rate in the blood. It is a pulse/heart rate sensor which easily measures your own pulse and thus monitor permanently your health condition, using Raspberry Pi (Raspberry Pi Heartbeat / Pulse measuring, n.d.). It measures the number of times the heart beats per minute (BPM). This sensor can also be used in mobile Raspberry Pi applications. The Raspberry Pi pulse sensor can not be read out digitally, thus we need an analog-to-digital converter. Such an ADC makes it possible to read out analog signals on the Raspberry Pi, communicates with microcontroller using SPI interface. It uses an infrared light to measure the pulses Heartbeat can be measured based on optical power variation as light is scattered or absorbed during its path through the blood. It uses 3.3 V from the Raspberry Pi (Raspberry Pi Heartbeat / Pulse measuring, n.d.).

3.0 Methodology

3.1 Required Resources

Report

/1 Parts/components/materials (500 words)

/1 PCB, case (500 words)

/1 Tools, facilities (500 words)

/1 Shipping, duty, taxes (250 words)

/1 Working time versus lead time (250 words)

3.1.1 Parts, Components, Materials

The main parts required for the blood oxygen rate functionality of the system includes: SD card ranging between 16Gigabytes and 32 Gigabytes, Raspberry PI 3 B+ microcontroller, MAX30100 Pulse Oximetry Sensor and Light Emitting Diodes, a 2N4124 transistor, soldering lead-free metal, electrical wires. Throughout the development stage, other parts needed to complete an up-coming task were purchased. These include, wire cutter and headers. Over time, some issues came up with some of the major parts which required purchasing. These parts became defective due to carelessness when handling parts during the configuration and testing stage. The parts that had to be re-purchased were, Raspberry Pi 3 B+, three SD cards with 32 Gigabyte capacity, and another MAX30100 Sensor. The MAX30100 Pulse Oximetry Sensor consists of seven pins. They are placed in order from VIN, SCL, SDA, INT, IRD, RD, GND respectively. VIN accepts source voltage ranging from 1.8V to 3.3V, SCL and SDA

are used to enable I2C communication, IRD and RD not applicable since not needed for functionality, GND for ground. Parts should be handled with utmost care as damage of parts will more often than not result in replacement, which is very expensive and takes time to get shipped. Some tips for success with regards to the testing stage of the hardware are as follows: do not, at any point, bring your microcontroller or sensor in contact with loose metal as this might possibly cause the sensor or the microcontroller to become defective, and maintain an organized environment when testing as this reduces the occurrence of possible accidents that could delay the completion of a dated task.

The connectivity and the functionality of the AMG8833 IR Grid Eye Thermal Camera was done in conjunction with a number of components and materials namely, the raspberry pi 4 which came with a 32 GB memory card with Raspbian pre install and a pawer supply adaptor to power up the pi, two 220 ohms resistors, one 2.2 kilo ohms resistor, two red LED and finally, one NPN-Transistor. Also, for the connections was use female connectors to connect the sensor to the board and the PCB board to the raspberry pi. The raspberry pi 4 was purchased from Amazon. The AMG8833 IR Grid Eye Thermal Camera was purchased from the manufacture’s website Adafruit. The resistors, transistor and LED, were available in the parts kit.

The main parts required for the heart rate functionality of the system includes:

VMA340 Heart/Pulse Rate Sensor that requires an MCP 3008 analog to digital converter, a Raspberry PI 3 B+ microcontroller, a 2N4124 transistor, 220 and 2.2k ohms resistors, LED, soldering lead metal, and electrical wires. Six attempts were made for production and soldering of PCB, five of which were flawed due to incorrect PCB

design. Throughout the initial stages of testing the sensing capabilities of the sensor, it wasn't possible to pick up any readings at all. After some weeks of troubleshooting and research online, we were able to come across a solution. Unfortunately, this problem resulted in a repurchase of a second VMA340 Heart/Pulse Rate sensor as the team couldn't overlook the possibility that there could be some defect with the first sensor, but the good new is, all the other parts were functioning as expected.

3.1.2 Manufacturing

The production of the PCBs and case was done at Humber College Institute of Technology and Advanced Learning, using laser-cut technology from the prototype lab. Soldering of the PCB for each sensor was done by the respective members of the team to which each sensor was assigned.

The manufacturing face of the project started with the designing of the PCB board with all its components using Fritzing. The first stage was the designing of the schematic drawing. The schematic drawing comprises of all the component used in the circuit and how there are interconnected to each other. Next was the designing of the PCB board. In fritzing, once the schematic is drawn the layout of the PCB board is automatically generated and just need the parts to be rearranged. The second stage was the printing of the board which was send for production in the campus prototype lab. There were two version of the board and it was printed four times. The first version had an issue with the soldering; one of the components was not properly soldered and had some lose connections. The second version was revised by adding two new components. A second LED and a new resistor were added to the circuit. The extra printing was printed

for backup in case something went wrong. The enclosure was designed using Inkscape. It was printed using laser cutting. The final case was reached after the third try. On the first try, the case was smaller than the raspberry pi. Tolerance was not taken in to consideration when giving the dimensions. As for the second print, just two part of the case did not fit and also the opening for the outlet puts did not match on the raspberry pi. After modification of all the minor details, the final enclosure was printed.

Causes of the re-modification and reproduction of cases were done due to incorrect measurements when situating the smaller details needed in the enclosure. Such as, designing and situating of slots for the peripherals of the raspberry pi microcontroller. Since these parts were very small, there was more focus on these areas than others, in order to ensure that slot measurements were accurate, thus reducing the amount of material being used up from repetitive production due to the case not meeting the necessary requirements.

For the finishing product, it was possible to attach the parts for the bottom of the PCB, and peripherals. However, it wasn’t possible to attach the top piece due to the unmeasured height of sensors, which were attached to the microcontroller while suspended in the air. Professor, Austin Tian, allowed the team to present the case with the top piece. For the upcoming PCB case, we plan to build a case that will also take into account the height of the sensor above the microcontroller, and adjust the measurements of the case accordingly. Since the size of the PCB with integrated sensors, is expected to be slightly bigger in comparison to the previous versions. We will have to increase the size of the case, and use screws and bolts to maintain the stability of the raspberry pi and PCB case.

3.1.3 Tools and Facilities

Throughout each milestone for the half of the development, various tools and facilities were used to achieve a working model by end of December 2019. The software tools used were, Inkscape, fritzing, Gantt, SD Card Formatter, and Android Studio. Inkscape was used to design the case for the embedded system. Fritzing was used to build a schematic design, bread board design, and PCB design for the circuit. A significant amount of time was spent on Inkscape and fritzing, since there were some minor errors in the design figured out at a later time, which needed to be fixed by revisiting these software tools. Gantt was used to set out the timeline for all the tasks required to be completed over the course of four months. It was encouraged, when building the chart, to follow Agile methods in order to increase productivity. This schedule, designed using Gantt, was then used to follow up on deadlines to ensure we complete tasks on time. SD card formatter is a utility that was used in the first stage of stage of configuration and installation of raspberry Pi operating System on the Microcomputer. SD card must be formatted so as to prevent the possibility of sensitive files getting corrupted during the installation phase. Android studio, is a software tool used to build the User Interface for the system. Android Studio was used to build an android application, and contains all the features needed to accomplish the User Interface requirements. For minor tools needed, such as safety glasses, extra soldering lead metal, electrical wires, extra transistors, and so on, were all borrowed from the parts crib at Humber College. In order to configure Operating System on microcontroller, some peripheral devices were needed. These were, a mouse and a keyboard, directly connected to the microcontroller

via the USB bus. After enabling VNC to allow the system to allow a remote connection over the network, it was then possible to use the VNC viewer software tool to interact with the microcontroller wirelessly.

3.1.4 Shipping, duty, taxes

For shipping of parts for MAX30100 Pulse Oximetry sensor, the maximum shipping time was three weeks, however, parts came one week earlier. The planned budget for shipping cost including taxes, were $94.25 in total. However, due to unexpected defects that occurred over the development period, more parts had to be purchased. Since time was limited during this time, even more money was spent to ensure the quick shipment of the parts so that they could be quickly integrated before the deadline. By the end of December 2019, a total of $262.91 was spent in total.

For shipping of parts for VMA340 Heart/Pulse Rate Sensor was two weeks and it took one week to arrive as it did not qualify for free shipping eBay charged me $25.00 shipping fee and %13 of the cost of the sensor for tax, there was no duty, I ordered the Pi and the MCP 3008 from Amazon it was free shipping and got delivered in one week, %13 of the cost of them were charged as well. However, as I had to reorder the sensor last minute, it was available on sparkfun only, had to choose express shipping so I receive it in two days, express shipping costed me $37.98 and after the order arrived I received a mail asking for $15.54 duties that I was not aware of plus the %13 tax of the sensor amount. All of the parts cost me $235.

The original shipping time for the raspberry pi 4 bought on Amazon was less than a week, however, it took longer than that due to some miscommunication from the shipping company. During the time of purchased, the shipping address was mentioned

and the process was completed without any issues. However, after waiting for the parcel for a couple of days and after further investigations, I came to notice that the shipping company do not actually ship at the address that was mention earlier and had to change the address for a new destination. The AMG8833 IR Grid Eye Thermal Camera on the other hand was shipped within the period of two weeks as mentioned during purchase time. In both cases, taxes were payed and did not have to pay extra charges nor duty fee.

3.1.5 Time expenditure

With regards to the lead time. All milestones were planned at and set in order so that every week there was something to do. For the first week, we were tasked to select our team members, choose the project we plan to take part in, and schedule a meeting with the collaborator for the project. For the second week, we did some research on the project, selected our sensor, submitted milestone for the proposal. In the third week, we were expected to have already ordered the parts, and should have already started planning the design of the User interface for the android application. For the fourth week, should have received our shipment, and begin configuration of equipment, and breadboard, schematic and PCB design. After reading week and mid-term, working on beta release for android application, should have all functionalities up and running before the due date. At this point, should have a working design for PCB, and send off to the prototype lab for production. In the 8th week, the testing phase of PCB, soldering, and testing connection with microcontroller. In the ninth week, beta release of android application should be completed and submitted before the deadline. For weeks of 10 to

13, designing case for Raspberry Pi and embedded system, and sending it to the prototype lab for laser or 3D print. Perform final touches to User Interface of android application, present working hardware, and working android application.

For the working time, most of the tasks set out for each week of the lead time was followed. However, there were periods where some tasks were submitted late due to unexpected situations occurring during development. Work load was also a major factor in development, since there were various other tasks given during this period so much so that it was difficult to keep on track.

3.2 Development Platform

3.2.1 Mobile Application

Our mobile application was built with the primary purpose to reduce the amount of work that a paramedic must undergo throughout an emergency scenario. In order to accomplish that, the mobile application was developed with, Android Studio as the software development platform, and was tested using android emulator and android devices owned by the team. Throughout the development process, various libraries, third party software, and software tools were used to ensure that the user interface of the app, meets the guideline set out for user requirements. We met these requirements by:

- Setting up GPS tracking and map modules for paramedic to locate a patient's whereabouts', and also for a patient to view the progress of a paramedic from hospital to his/her home location.

- Researching, downloading, and using jar files and SDK tools available from third party software Companies to help in the development of a video-chat module and a database for data storage. Firebase was used to create a database over the network for storing data, and resources provided from tokbox, an API platform Company that develops video conferencing interfaces from various platforms, was used to build a Video-Chat interface for android.
- Building, a simple, easy to understand user interface that would be easily navigated by both parties with ease. This was done by reducing the number of actions needed by a user in order to navigate the app, adding labels for clarification at some areas that might possibly cause some confusion to the user.

With regards to functionality, upon initialization, that mobile application functions as follows:

- Upon start-up, a splash screen (Figure 1) is displayed and remains for three seconds before moving on to the next activity. This serves to welcome the user to the app, and advertise the name of the mobile application

Figure 1. By Android Studio. Splash screen view of mobile application: Paramed

- After the splash screen activity has ended, a dialog box will then be displayed. This will ask the user if he/she is a paramedic. If the user selects ‘yes’, then user will be taken to the paramedic login activity. If user selects ‘no’, then user will be taken to the video chat activity.

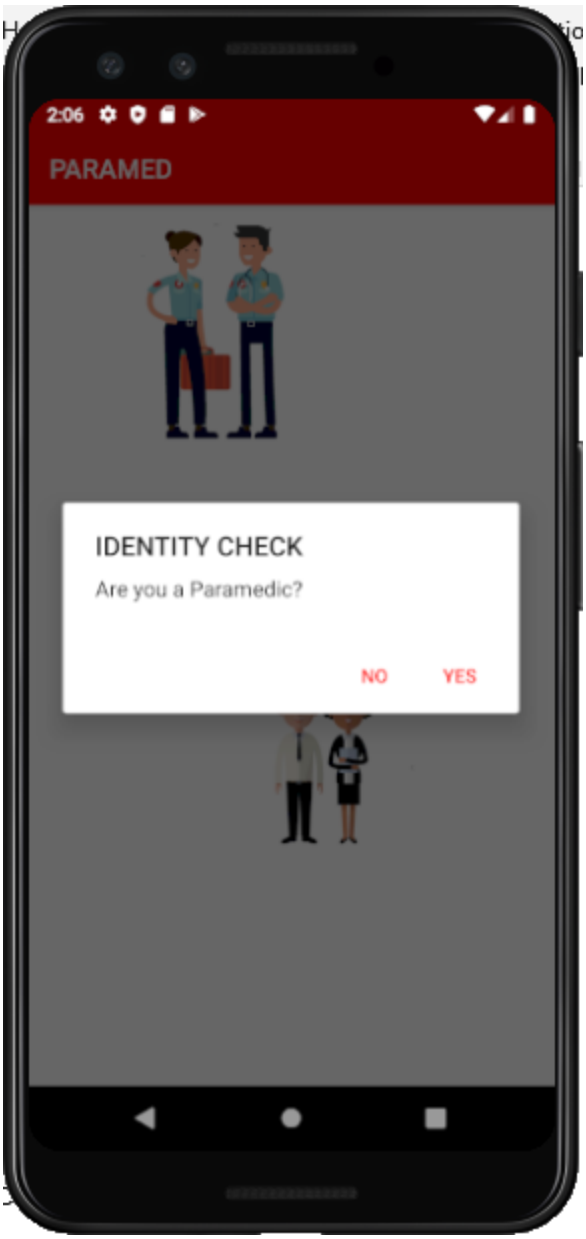


Figure 2. By Android Studio. Home screen, including dialog box, prompting user to choose his identity

- If the user is a paramedic, then user will be required to type in his/her username and password, which will be validated by comparing their inputs to the credentials stored on firebase database. When user types in his/her credentials, he/she must select the button labelled as ‘sign in’, this will initiate the validation phase. By fetching the username and password saved on the firebase and

comparing these credentials to the input typed in by the user, and if the username and password match, then user/paramedic will be logged into his profile. If a match was not found on the database for the input typed in by the user, then a toast will be displayed to inform the user that the credentials are incorrect, and will remain in this activity until user provides the correct credentials. A paramedic's profile has three sections. The 'home', 'maps', and 'recent' sections. In the home section, paramedic is able to view the numbers of emergency scenarios completed thus far, and view the details of the emergency simply by selecting the desired emergency scenario to view. Upon selection, details such as: a patient's location and contact information, and their vital readings. These vital readings, which includes, temperature, blood oxygen concentration, and heart rate, of the patient, and a status report to specify whether the emergency was completed or still in progress. All this information is fetched from the database for view by the paramedic. At the bottom of the screen, there is a bottom navigation bar that allows a paramedic to traverse between each section with ease. If paramedic selects the maps section, he/she is able to view and change his/her current hospital location. This section will be used by the paramedic when travelling towards the location of the patient. The third section: 'recent', shows the most recent emergency scenario that has been completed by the paramedic or still in progress. At the top right corner of a paramedic's profile, menu options are given to allow a paramedic to sign out of his/her account, or check their current status information and hospital information.

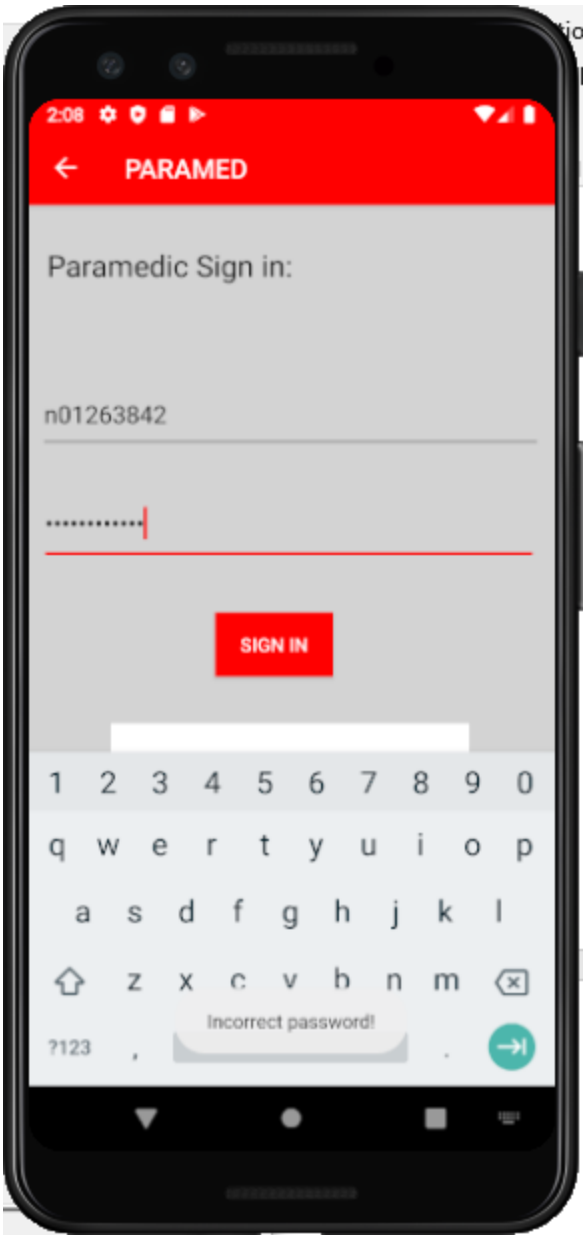
- 

Figure 3. By Android Studio. Paramedic login screen. Toast displayed at bottom of the screen to inform a user that their credentials are incorrect

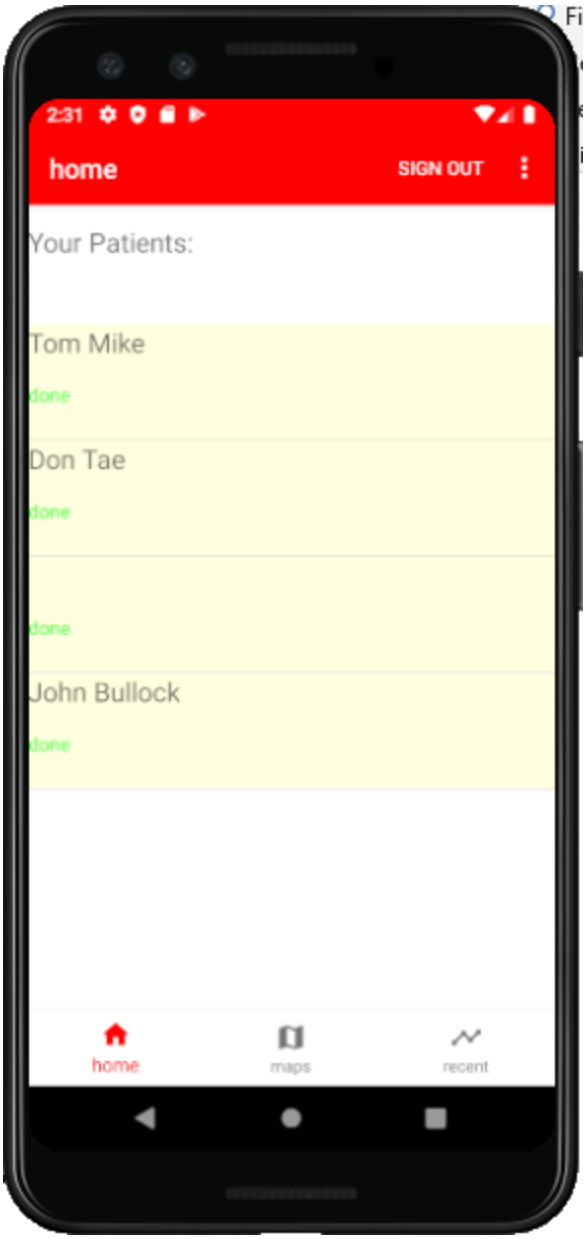


Figure 4. By Android Studio. Paramedic Home screen showing, bottom navigation menu, sign out option, and a list of emergency scenarios

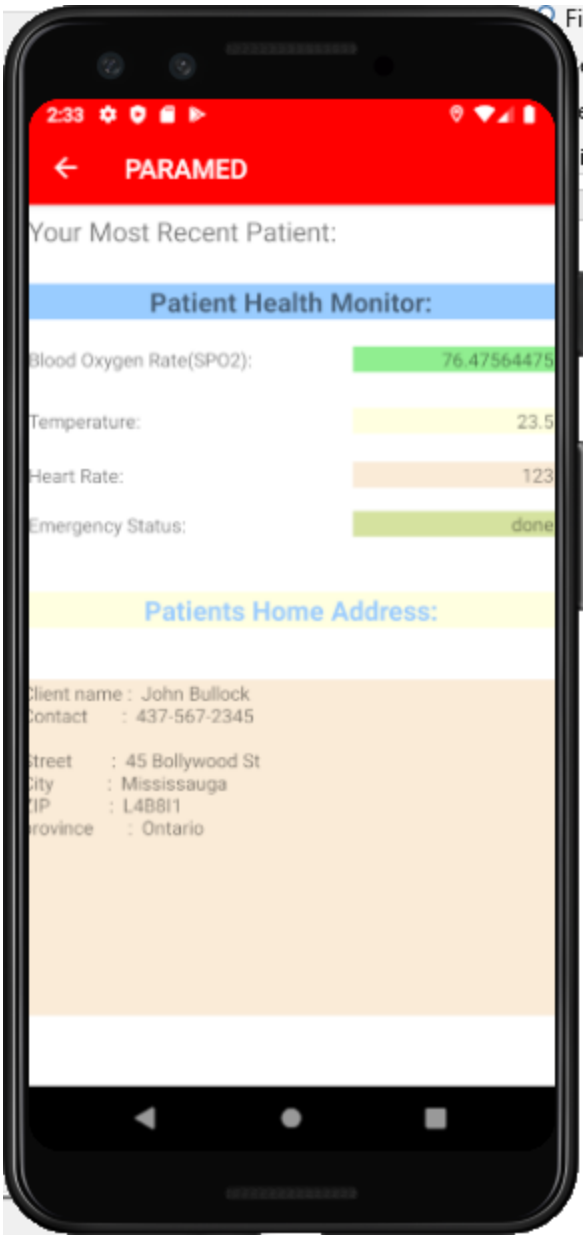


Figure 5. By Android Studio. Simulates how patient information is displayed when an emergency scenario is selected

- If the user is a patient (that is, user is assumed to be a patient when the 'no' option is selected at the dialog screen), then he/she is then forwarded to the video chat activity where they will then wait for a connection with an available paramedic. Patient also has a menu which provides them with the functions

needed to type in their home information if needed. This menu, also provides a map option which allows a patient to navigate manually to the map module.

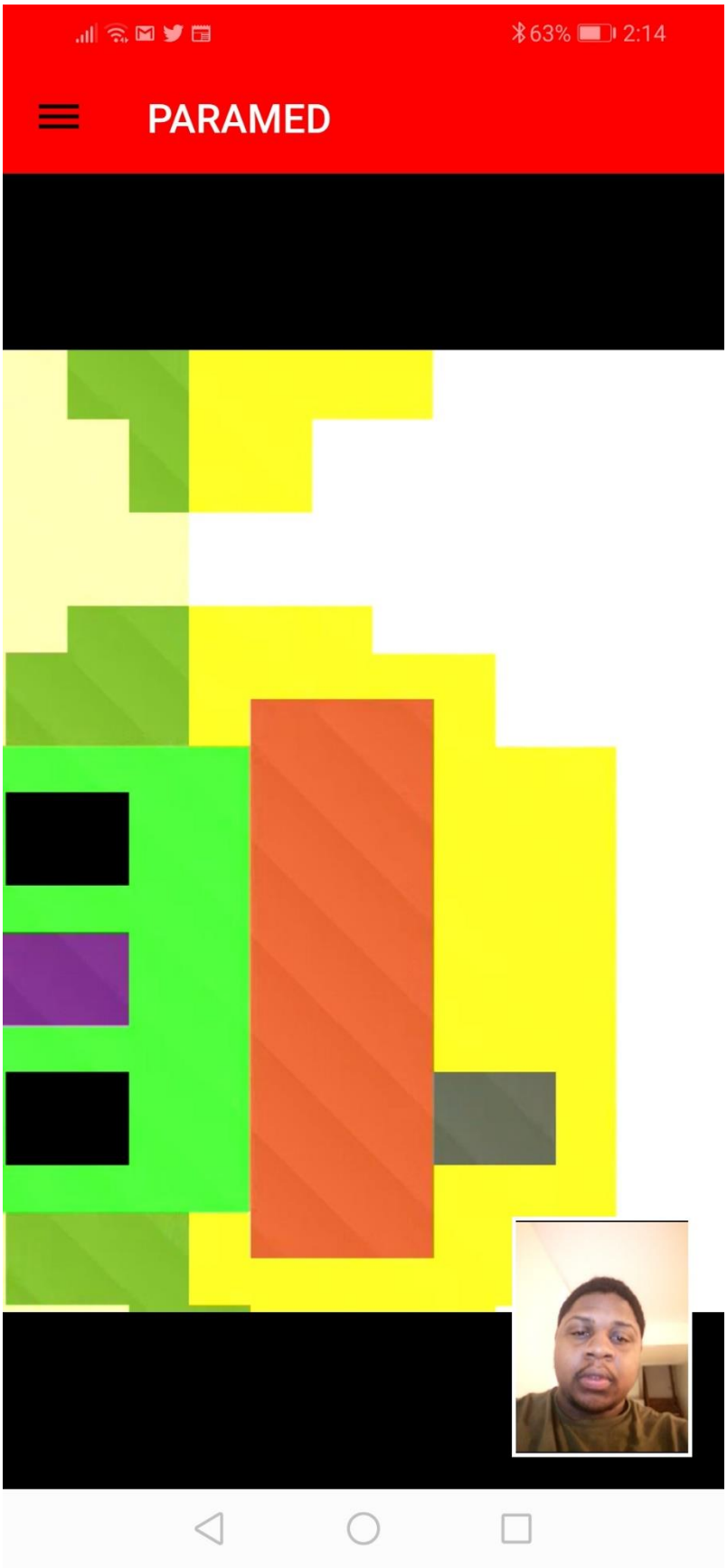

63% 2:14
PARAMED

Figure 6. By personal android device. Implementing the video chat functionality

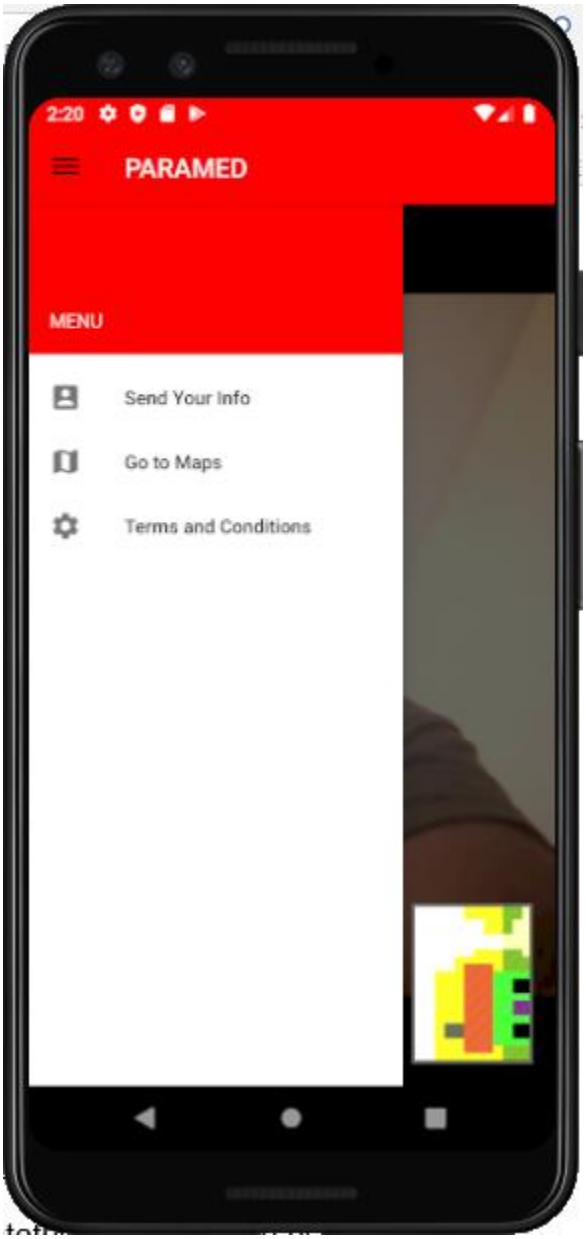


Figure 7. By android Studio. Implementing menu using Navigation Drawer Layout

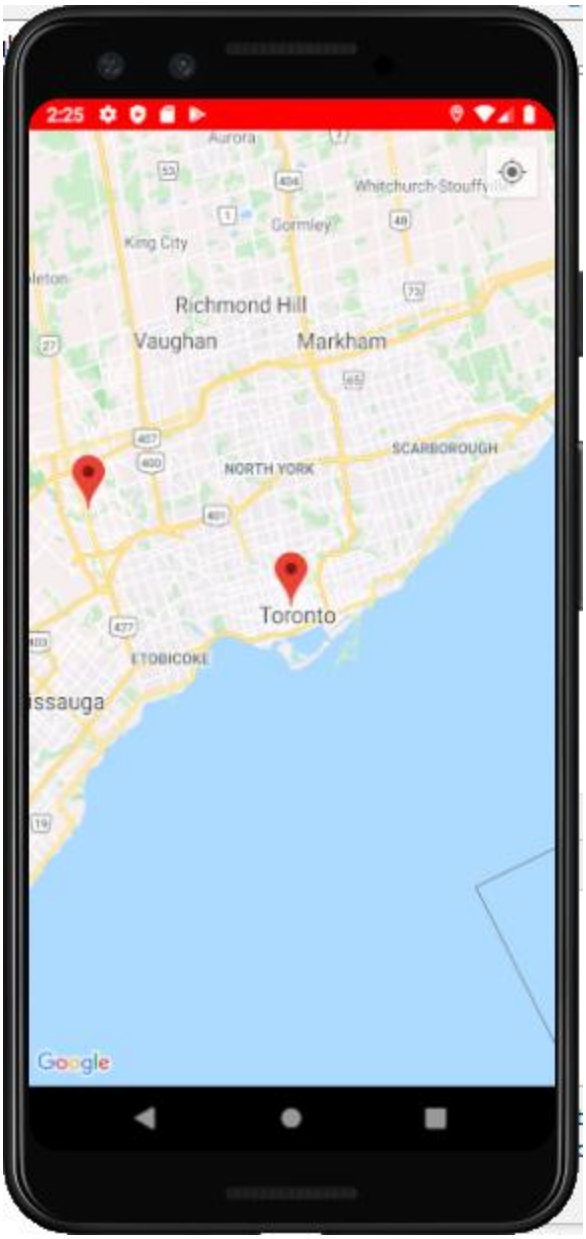


Figure 8. By Android Studio. Clicking the 'Go to Maps' options generates a map module

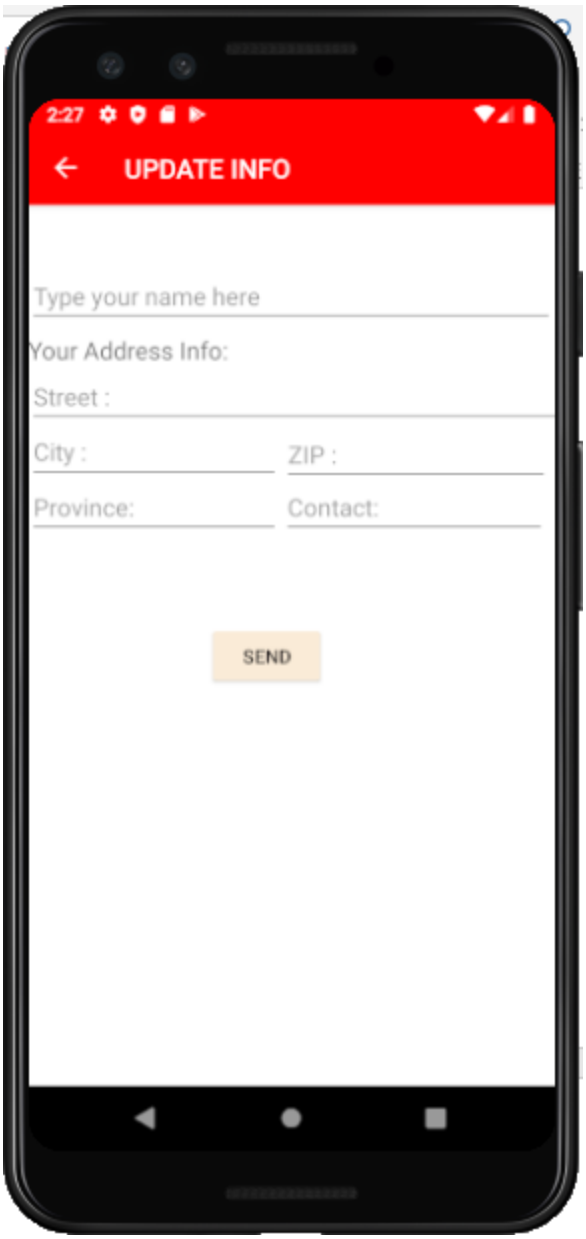


Figure 9. By Android Studio. Selecting menu option 'Send your Info' initiates an activity where patient can types his/her information manually

For the remainder of the development process, there are some functionality that still need to be added. Such as, establishing a secure connection with a paramedic, GPS tracking capability not yet implemented, more work needs be done on UI design. In order to set up a secure connection with a paramedic, mechanisms have to be set to

determine which paramedic is currently accepting calls. When an available paramedic is found, send a connection request to the paramedic. This will notify the paramedic of the connection request and prompts him/her to accept or decline the connection request. All of this will be done by automatically updating the database when a paramedic logs in to his/her account. When logged in, a paramedic status on the database will be ‘active’, and when not logged in, ‘inactive’. If a paramedic is currently undergoing an emergency scenario, then their status will be ‘busy’. In this way, these statuses will then be used to determine which paramedic is available and send a connection request to the available paramedic. With regards to GPS tracking, this will be highly dependent on firebase database. The longitude and latitude values of the patient and paramedic will be constantly updated to firebase and fetched to determine the most up to date location of both paramedic and patient. Additionally, the changes that need to be made to the UI design were decided on from feedback given by Professor Haki Sharifi, who taught the software project course for fall 2019. He helped us by asking questions about the practicality of the User Interface, and what it has to offer to a paramedic and patient. For example, the map section on a paramedic profile. This section won’t be used until a paramedic is heading towards an emergency scenario, and as such, the UI could be much simpler and less confusing by removing this section, only showing the home section and recent section. The map section will be displayed when needed by the paramedic, that is, after the conversation with the patient has ended. As for the patient side, reduce the number of actions that’s required by a patient as much as possible. What matters most is that a patient is allowed quick and easy service without any delays or confusion during their time of need.

Status

/1 Hardware present?

/1 Memo by student A + How did you make your Mobile Application? (500 words)

/1 Login activity

/1 Data visualization activity

/1 Action control activity

Include screenshots such as **Error! Reference source not found.**. Testing. Progress.

3.2.2 Image/firmware

The image/firmware for each sensor was developed using the raspberry Pi 3 B+ microcontroller, and raspberry Pi 4 microcontroller as the platform for sensor operation. For code compilation, sensor testing, python/python3 was the software development tool used. With regards to interactivity/connectivity with the raspberry pi, VNC was enabled during the configuration process of the raspberry in the initial stage, so that the operating system on Raspberry Pi could be accessible remotely over the network. The Raspberry Pi for each team was connected wirelessly through VNC using the Humber network. The functionality of the image/firmware for each sensor, currently, is very basic. We still need to, implement code to save processed readings outputted from sensors to firebase, integrate source code for each sensor as one executable file,

develop a service process in **systemd** so that it automatically runs the integrated program, by taking advantage of the executable file obtained from compilation, upon start up and successful boot of operating system on raspberry pi. The installation process for Raspbian on the raspberry Pi microcontroller is as follows:

- On a web browser, search and download the NOOB file for raspberry Pi. This contains all the boot files needed for the installation of Raspbian on the Raspberry Pi.
- Save the contents of the NOOB files on an empty SD card. Recommended capacity for SD card is 32GB.
- Remove the SD card from computer and place into the required SD card slot on the Raspberry Pi microcontroller.
- Connect the Raspberry Pi to power.
- Using an HDMI-VGA or HDMI-HDMI cable, connect the Raspberry Pi to a PC monitor. When this is connected, a GUI will be displayed providing partitioning information and available Raspbian packages to install.
- Connect two peripherals: keyboard and mouse, for navigation. When this is done, use the mouse to click the Raspbian option to begin installation process.
- Wait for installation process and add information where required, such as username, password, date, network, etc.

In order to use this firmware, the raspberry must be configured to enable I2C, SPI, and VNC using the command **sudo raspi-config**. To check the connection of I2C with sensors using this interface, use the **i2cdetect -y 1**.

With regards to functionality of the program for each sensor, a brief description of how this is done is as follows:

The MAX30100 Pulse Oximetry was implemented by taking advantage of libraries and source code already provided on Github. The link to the github repository:

https://github.com/mfitzp/max30100.git

```
#Name: Akeem Abrahams
#Project: Pulse Oxymetry (MAX30100)
#Student#: N01263842

import max30100
import time

mx30 = max30100.MAX30100()

mx30.set_mode(max30100.MODE_SPO2)

while True:
    mx30.read_sensor()
    if(mx30.red > 0 and mx30.ir > 0):
        print("Raw IR Readings: "+str(mx30.ir))
        print("Raw Red Readings: "+str(mx30.red))
        print("")
        print("Calculating SPO2....")
        time.sleep(1)
        value = (float)(mx30.red)/(float)(mx30.ir)
        print("Calculated Value: "+str(value*100)+"%")
        print("")
    else:
        print("Finger not detected")
    time.sleep(1)
```

Figure 10 Python via RPI, source code for monitoring readings produced from MAX30100 Pulse Oximetry Sensor

In summary, the source code shown in figure 10, functions by outputting a human-readable description of what the sensor is detecting. In order for the sensor to detect the blood oxygen level in an individual, it requires that the individual get in contact with the

sensor by placing his/her finger directly on the surface of the sensor. If sensor doesn't detect human contact, it will output a 0 and send this value via I2C. If sensor detects human contact, it will output nonzero values, namely, the Infrared reading (IR readings) and red readings, sending these values via I2C. By finding the ratio between red readings and IR readings and multiply the result by 100, the current blood oxygen level in the human body can be obtained.

The code in figure10 works as follows:

- Library for max30100 are imported in order to use functions in the library. Time is included so that delays could be used in the code
- **mx30 = max30100.MAX30100()** initializes the sensor
- **mx30.set_mode(max30100.MODE_SPO2)** enables blood oxygen detection on the sensor
- Continue to read the values outputted from the sensor, by putting **mx30.read_sensor()** in a while loop.
- If the value of IR Readings (**mx30.ir**) and red readings (**mx30.red**) is zero, then print out a description saying, "**finger not detected"**.
- If the value of **mx30.ir** and **mx30.red** is nonzero, then print out raw IR and red readings, print out a description to inform the user that SPO2 is being calculated, wait for 1 second, then print out the calculate value by implementing this formula: (**mx30.red/mx30.ir)x100**.
- Wait 1 second before delaying the next reading.

The AMG8833 IR Grid Eye Thermal Camera sensor is able to read real time value with the help of a software which is written to control the behavior of the sensor and help

display the value in various forms. The values from the sensor can be read and display in pixel value; a series of number stored in an array. Also, the representation of the temperature could be displayed on a visual screen display. The image resents the temperature with different colors with each color describing different state of temperature. Red and yellow colors represent the warmer temperature while green and purple colors represent the cooler temperatures. The code for running the sensor was provided by the manufacturing company of the sensor Adafruit with the instruction on how to download and run the code. The code was written in python and was slightly modify to meet the need of the project; like adding code to turn on the red LED to indicate the pi is on and flash another LED during run time. The code was ran and downloaded from the Adafruit git repository through command line in addition to other libraries like the color, python, pygame and scipy.

The github link for the implementation of the AMG8833 IR Grid Eye Thermal Camera sensor is: https://github.com/adafruit/Adafruit_AMG88xx_python.git

Implementation of MCP3008 Heart Rate Sensor, was done by using libraries and code retrieved from: https://github.com/tutRPi/Raspberry-Pi-Heartbeat-Pulse-Sensor

MCP3008.py, in this file is the code for the ADC chip using the SPI interface.

README.MD

Example.py, is the file that was used to run the sensor and it has the two conditions with print statements that if the sensor detected a finger on it to print a value, otherwise, print “No Heartbeat found”. the Pulsesensor.py and MCP3008.py was imported in the source code, to use the function needed.

Pulsesensor.py : has the code for the pulse sensor to measure the heart rate per minute, it measure the time between beats in milliseconds, then takes the average and keep track of time for next pulse.

Status

/1 Hardware present?

/1 Memo by student B + How did you make your Image/firmware? (500 words)

/1 Code can be run via serial or remote desktop

/1 Wireless connectivity

/1 Sensor/effector code on repository

3.2.3 Breadboard/Independent PCBs

The major components of the hardware were designed using the Fritzing software; a software that helps in drawing and designing the schematic, breadboard layout and the PCB board. Based on the project's requirements and different components datasheet together with the required connection of the AMG8833 IR Thermal Camera proposed by Adafruit, the finalized schematic shown in Figure 11 was produced. The completion of the schematic drawing lead to the breadboard layout show in Figure 12 which was later tested by physically connecting all the components on a physical breadboard. Lastly, was the PCB design. The PCB was designed in two different phases and printed twice with both times having a copy in case of any error during the soldering phase. The issue encountered with the first version Figure 13 was the soldering of the capacitor; the capacitor was not properly soldered on the PCB board connection thereby causing the

LED to flicker anytime it was touched or moved. Later on, the first version was modified resulting to version two Figure 14. The modification included the addition of a resistor and a second LED which will turn on and off while the sensor reads in values. Also, the connecting wires of the resistors, LEDs and capacitor were sent to the back for easy soldering. The PCB board was printed in the prototype lab at no cost. On the other hand, the case Figure 15 was design in Inkscape and was laser cut in the prototype lab. The case like the PCB was printed at no cost. However, some components like the sensor and the raspberry pi were purchased. A brief description of the bill of materials is shown in Figure 16.

The MAX30100 Pulse Oximetry Sensor

The MAX30100 Pulse Oximetry Sensor is fully functional. When connected to the raspberry Pi, the register for communication over the I2C is recognized, and is determined with the use of **i2cdetect -y 1** command. The sensor is able to output non-zero readings when in contact with a human. When operational, the sensor emits a red light.

VMA340 Heart/Pulse Rate Sensor

My sensor was not functioning properly in the beginning. It was giving correct readings on the first trial of testing, but, when tested for the second time and for the rest of the test trials to come, readings were random and inaccurate. After some help from Professor, Kristian Medri, the problem with the sensor was fixed. The problem wasn't in the code itself or the functionality of the sensor. It was the gesture needed to operate the sensor properly that was done wrong. In order to obtain accurate readings, the sensor must, first, be taped on the user's hand.

AMG8833 IR Thermal Camera

The AMG8833 IR Thermal Camera sensor is fully functional and ready for integration as the sensor is able to read values and display them on the screen through the help of the supporting software program. The code that displays the values on the screen and that controls the behavior of the sensor is written in python.

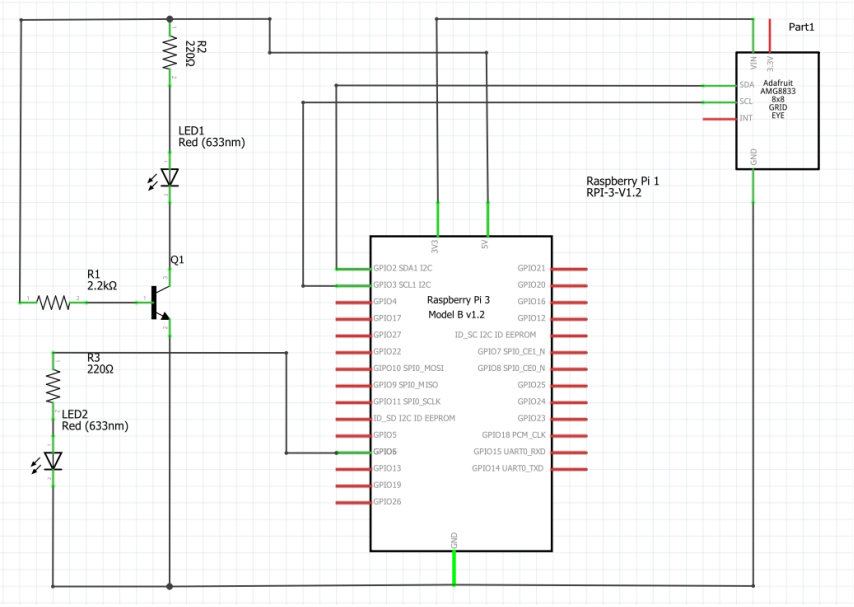


Figure 11. Initial schematic. by Cedric Wambe

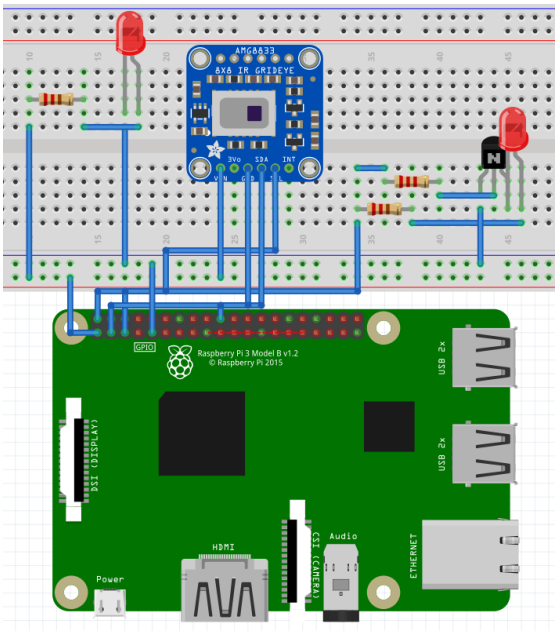


Figure 12 Breadboard image by Cedric Wambe

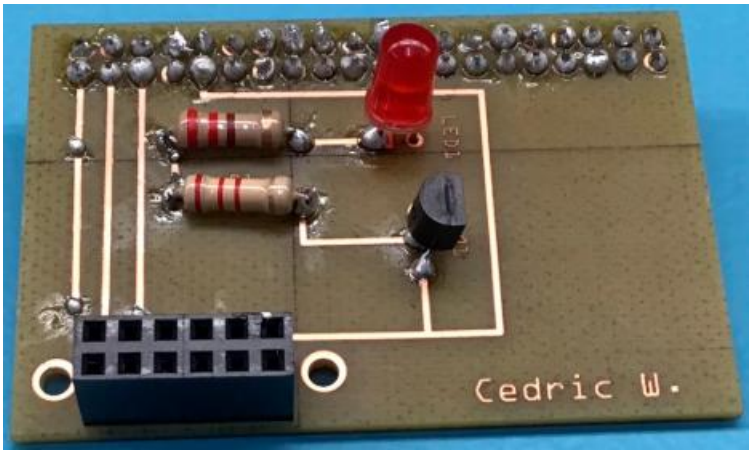


Figure 13 PCB Version 1.

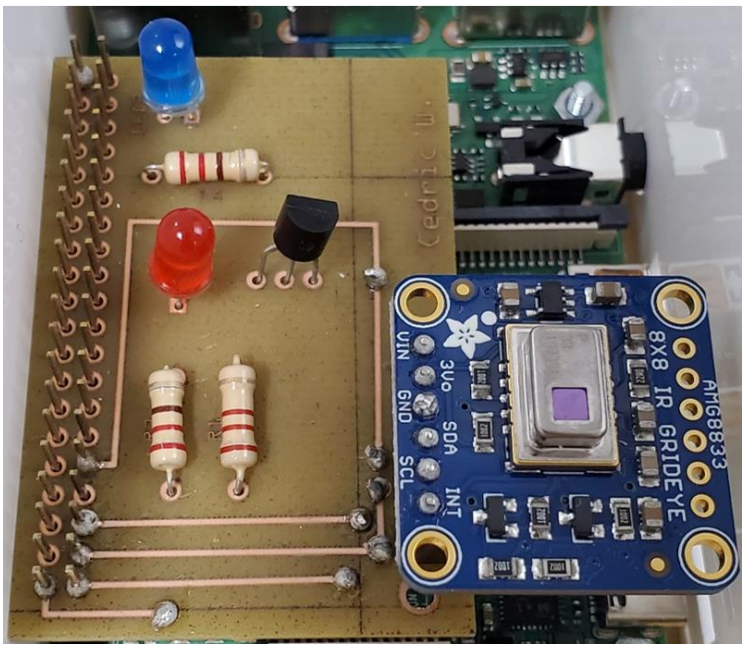


Figure 14 PCB version 2 with AMG8833 IR Grid Eye sensor connected

Figure 15 Case

EXPENDITURES	Quantity	Unit Price	Sub Total
Raspberry Pi 4 4GB	1	$74.95	$74.95
CanaKit Raspberry Pi 4 Power Supply (USB-C)	1	$12.95	$12.95
Raspberry Pi - 32 GB Memory Card w/NOOBS	1	$16.95	$16.95
Raspberry Pi 4 3 B+ 3.5" Touch Screen Display LCD TFT Game Monitor	1	$36.97	$36.97
Adafruit AMG8833 IR Thermal Camera Breakout	1	$39.95	$39.95
		Sub Total	**$181.77**
		Shipping	$26.10
		Tax - HST	$27.02
		Total	**$234.89**

Figure 16 Budget

Status

/1 Hardware present?

/1 Memo by student C + How did you make your hardware? (500 words)

/1 Sensor/effector 1 functional

/1 Sensor/effector 2 functional

/1 Sensor/effector 3 functional

3.2.4 Printed Circuit Board

The integrated PCB was done by merging the circuit diagrams for the AMG8833 IR Thermal Camera, the MAX30100 Pulse Oximetry Sensor, and the MCP3008 Heart Pulse Rate Sensor. This was facilitated using third party software, Fritzing. PCB was tested using the DLC check function, provided by the fritzing software, before exporting the design for production using Gerber format. The zip file created, containing these Gerber files, were then sent to the prototype lab for production via email. About 3 days after sending the design, the PCB was built, and was soldered by the team. We later realized that the copper wires connecting the pin-outs for the Raspberry Pi, was

designed to wrong, that is, when designing the PCB wires must meet pin-outs at the top of the PCB instead of the bottom. Revisions were made, and another PCB was sent for production at the prototype lab shortly after.

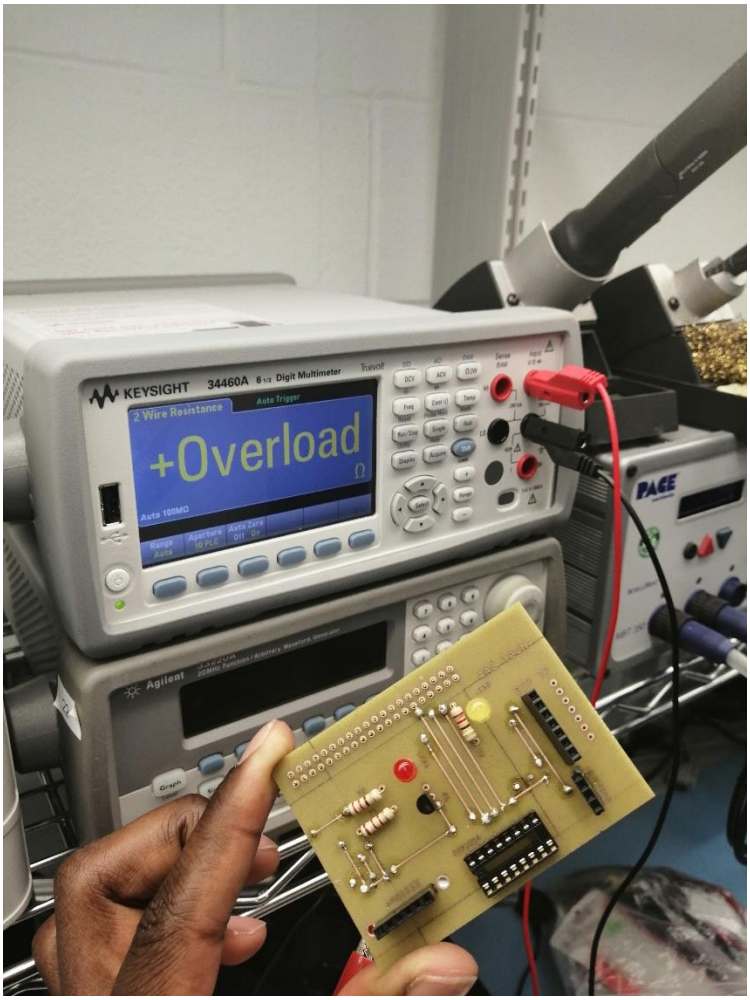


Figure 17- Testing Integrated PCB using Multimeter: demonstrating results in ohms when testing pinouts where there is no electrical connection

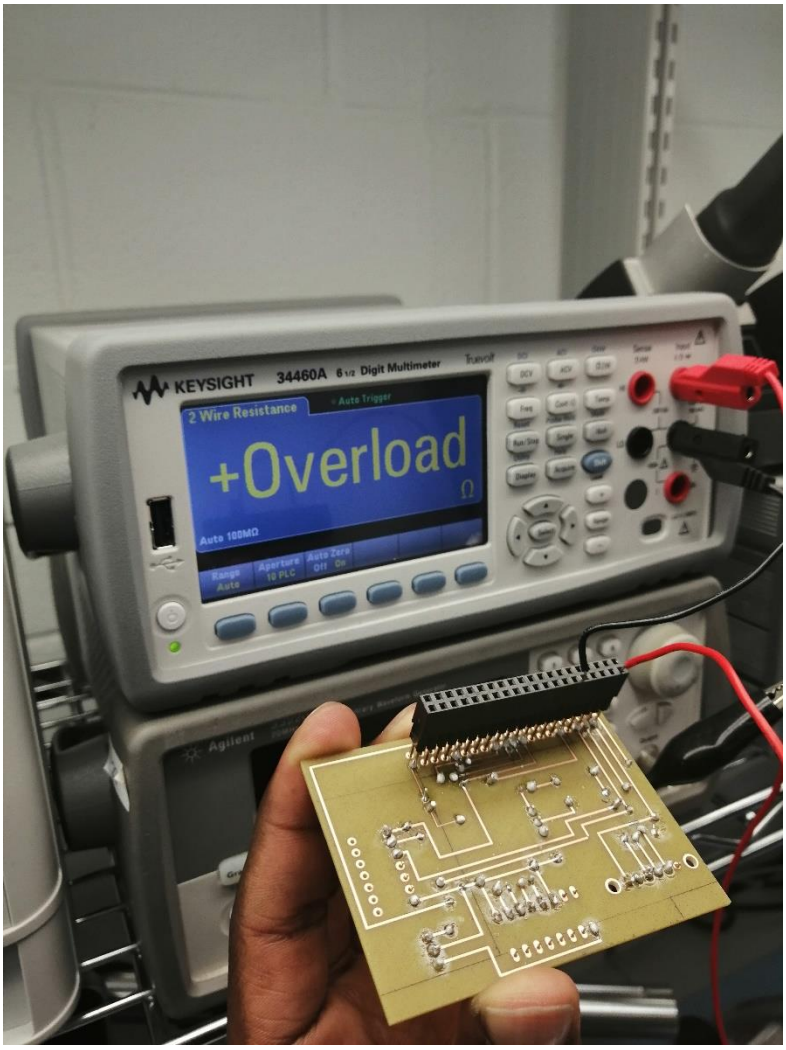


Figure 18- Testing of Integrated PCB using multimeter: Displaying the rear side of the PCB and another example of multimeter results when there is no electrical connection

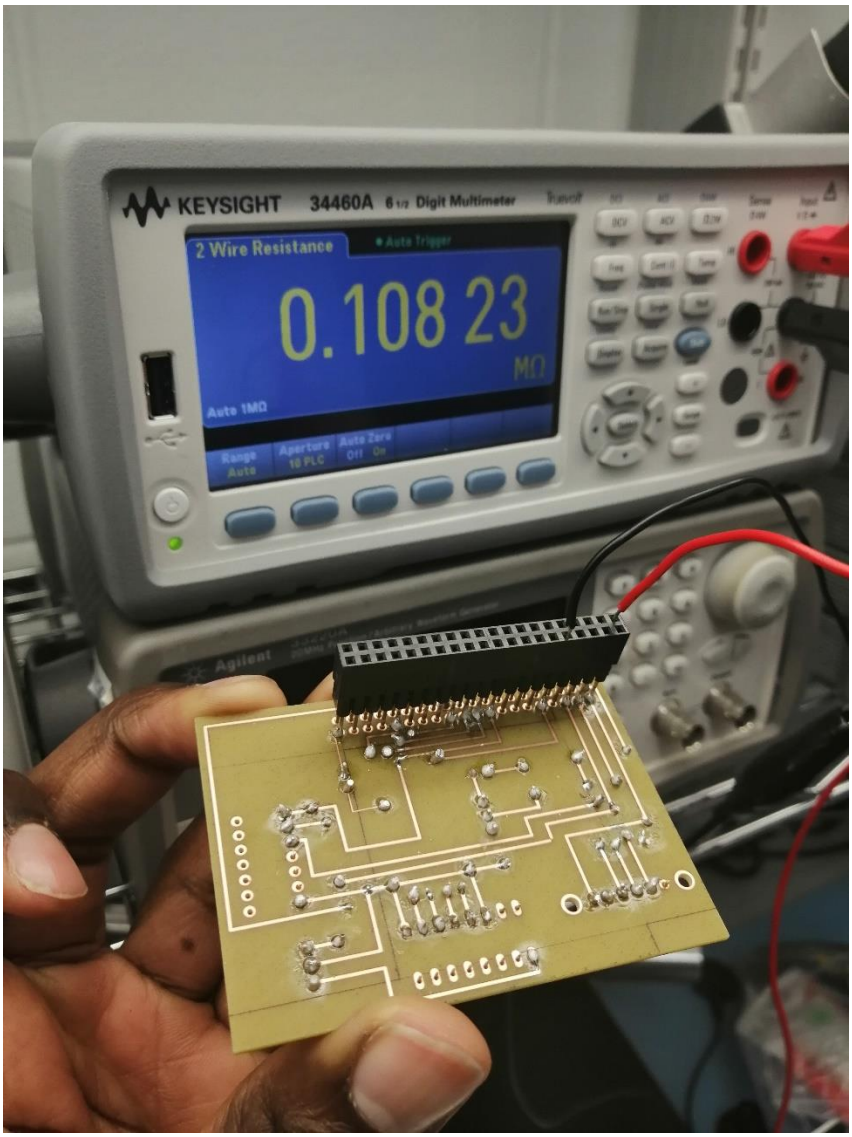


Figure 19- Testing integrated PCB using multimeter: Showing an example of multimeter results in ohms when there is an electrical connection on the PCB

Demo

/1 Hardware present?

/1 PCB Complete and correct

/1 PCB Soldered wire visible but trim, no holes or vacancies

/1 PCB Tested with multimeter

/1 PCB Powered up

How did you build your Prototype: PCB?

3.2.5 Enclosure

Demo

/1 Hardware present?

/1 Case encloses development platform and custom PCB.

/1 Appropriate parts securely attached.

/1 Appropriate parts accessible.

/1 Design file in repository, photo in report.

How did you build your Prototype: Case?

3.3 Integration

Demo

/1 Hardware present?

/1 Data sent by hardware

/1 Data retrieved by mobile application

/1 Action initiated by mobile application

/1 Action recieved by hardware

Report

/1 Enterprise wireless connectivity (250)

/1 Database configuration (250 words)

/1 Security considerations (500 words)

/1 Unit testing (900 words)

/1 Production testing (100 words)

3.3.1 Enterprise Wireless Connectivity

How did you make a Database accessible by both your Prototype and Mobile Application?

3.3.2 Database Configuration

3.3.3 Security

3.3.4 Testing

Unit testing and Production testing.

4.0 Results and Discussions

Throughout the period of Fall 2019 and Winter 2020 of the final two semesters of the Computer Engineering Technology program at Humber college, the team planned and developed a software and hardware-based medium geared towards interaction between a paramedic and a patient during an emergency. The objective behind the development of this project was focused more on significantly improving the receptiveness and efficiency of a paramedic by removing the necessity for a paramedic to retrieve personal information about a patient, which includes a patient's, current whereabouts, current health situation, and current home of residence. By creating a platform capable of retrieving personal information of a patient, a paramedic's role has become reduced to simply heading towards the scene of the emergency as soon as possible.

To facilitate the design, the team went about this by:

- Developing a mobile application with video-chat and GPS tracking capability.
- Setting up a database on the network using Firebase database to store patient and paramedic information
- Setting up three sensors, which includes: the MAX30100 Pulse Oximetry Sensor, MCP3008 Heart Pulse Rate sensor, and AMG8833 IR Thermal Camera, embedded on a Raspberry Pi microcomputer.

By the end of development, the team was able to ensure these following functionalities for a paramedic->patient interaction:

- Paramedic is able to retrieve a patient's vital information from the database for display on the mobile application

- Paramedic is able to track a patient's current location by implementing google map modules on the mobile application. This tracking is done by updating the patient's and the paramedic's current longitude and latitude coordinates on the database, which is then retrieved and processed in the mobile application to display the current location.
- Patient is able to initiate a video chat with a paramedic
- Patient is able to view the progress of a paramedic as they are heading to his/her location
- Embedded system has been programmed to activate the sensors to sense and process a patient's vital information, which includes, blood oxygen levels, heart pulse rate levels, and temperature readings

With regards to the completeness of the project, unfortunately, the team wasn't able to complete all the functionalities of this projects due o the following reasons:

- Sudden defects or accidents in handling of equipment during the course of development. There were situations where sensors wee malfunctioning due to accidental damage, which resulted in the team having to purchase a replacement.
- Sudden change of environment due to the pandemic. The completion of this project, significantly relied upon exposure to the facilities provided by Humber. Since the team must remain at home during this crisis, it became difficult to complete group tasks. Such as, integrating the system
- Bug fixes. With regards to the mobile application, there were bugs in the program that took months to successfully fix. If these fixes were done much quicker then

more time could have been allocated to other tasks throughout the development of the project

The following tasks that have not yet been completed:

- Setting up code for sensor readings to be sent for storage on the database over the network
- Setting up processes on the RPI to run the sensor program automatically upon successful boot
- Some functionalities have not yet been added to the mobile application
- Couldn’t complete the PCB and enclosure due to lack of facilities due to the pandemic

With regard to prospects for mass production, the team believes that, if everything goes well, and sensors could be improved to output readings much quicker, and are much more reliable with regards to accuracy, then this would play a very significant role to paramedics and patients worldwide. Seeing as this product is geared towards removing some of the work that a paramedic has to complete during the course of an emergency, you would expect that a paramedic can now focus more on the well-being of the patient and should be better prepared to deal with any situation and make better decisions on what to prescribe for the patient even before arriving at the scene. The biggest hurdle for this prototype is the cost of equipment and additional funds for backup equipment provided that there are any problems during development, but if this prototype should gain sponsorship, the team believes that it will have a positive impact on the world going forward.

5.0 Conclusions

Throughout the period of Fall 2019, and Winter 2020, an emergency-oriented platform was successfully developed to sufficiently improve the receptiveness of paramedics during an emergency scenario.

Going forward, the team plans to submit the complete mobile application on Google Play store to allow worldwide access for paramedics. Arrangements will be made to set up a meeting with the collaborator to discuss prospects for the future with regard to the feasibility and usefulness of the prototype, with a goal to decide whether its worthy of sponsorship and mass production.

Report

/1 Hardware present?

/1 Checklist truthful

/1 Valid Comments

/1 Results and Discussion (500 words)

/1 Conclusion

Bosch Sensortec. (2019, July). *BME680 - Datasheet.* Retrieved from Robert Bosch GmbH: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME680-DS001.pdf

Kinsella, J. (2019). Five trends predicted for the cloud industry in 2019. *Software World*, 50(1), 11.

Media, O. (2019). *O'Reilly artificial intelligence conference 2019 - San Jose, California.* California: O'Reilly Media, Inc.

Miller, D. (n.d.). *Adafruit AMG8833 8x8 Thermal Camera Sensor*. Retrieved from Adafruit: https://learn.adafruit.com/adafruit-amg8833-8x8-thermal-camera-sensor/overview

OACETT. (2017, March). *I need to Complete a Technology Report*. Retrieved from The Ontario Association of Certified Engineering Technicians and Technologists: https://www.oacett.org/Membership/Technology-Report-and-Seminar

Raspberry Pi Heartbeat / Pulse measuring. (n.d.). Retrieved from Raspberry Pi Tutorials: https://tutorials-raspberrypi.com/raspberry-pi-heartbeat-pulse-measuring/

Robuck, M. (2018, 11). AWS goes deep and wide with machine learning services and capabilities. *Fierceinstaller*.

SPO2 Sensor MAX30100 Pulse Oximeter heart Rate Sensor Module. (n.d.). Retrieved from Hallroad.org: https://hallroad.org/max30100-pulse-oximeter-heart-rate-sensor-module-in-pakistan.html

Strogonvs, R. (2017, 3 8). *Implementing pulse oximeter using MAX30100*. Retrieved from Morf Coding & Engineering: https://morf.lv/implementing-pulse-oximeter-using-max30100

Bosch Sensortec. (2019, July). *BME680 - Datasheet.* Retrieved from Robert Bosch GmbH: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME680-DS001.pdf

Kinsella, J. (2019). Five trends predicted for the cloud industry in 2019. *Software World*, 50(1), 11.

Media, O. (2019). *O'Reilly artificial intelligence conference 2019 - San Jose, California.* California: O'Reilly Media, Inc.

Miller, D. (n.d.). *Adafruit AMG8833 8x8 Thermal Camera Sensor*. Retrieved from Adafruit: https://learn.adafruit.com/adafruit-amg8833-8x8-thermal-camera-sensor/overview

OACETT. (2017, March). *I need to Complete a Technology Report*. Retrieved from The Ontario Association of Certified Engineering Technicians and Technologists: https://www.oacett.org/Membership/Technology-Report-and-Seminar

Raspberry Pi Heartbeat / Pulse measuring. (n.d.). Retrieved from Raspberry Pi Tutorials: https://tutorials-raspberrypi.com/raspberry-pi-heartbeat-pulse-measuring/

Robuck, M. (2018, 11). AWS goes deep and wide with machine learning services and capabilities. *Fierceinstaller*.

SPO2 Sensor MAX30100 Pulse Oximeter heart Rate Sensor Module. (n.d.). Retrieved from Hallroad.org: https://hallroad.org/max30100-pulse-oximeter-heart-rate-sensor-module-in-pakistan.html

Strogonvs, R. (2017, 3 8). *Implementing pulse oximeter using MAX30100*. Retrieved from Morf Coding & Engineering: https://morf.lv/implementing-pulse-oximeter-using-max30100

7.0 Appendix

7.1 Firmware code

```
# Authors  :   Akeem Abrahams, Asmaa Alzoubi, Cedric Wambe
# Team Name: AAC-Tech

from pulsesensor import Pulsesensor
import max30100
import time

# Initializing MCP3008 Heart Rate Sensor
p = Pulsesensor()
p.startAsyncBPM()

# Initializing Max30100
mx30 = max30100.MAX30100()
mx30.set_mode(max30100.MODE_SPO2) # Enabling SPO2 mode on MAX30100
mx30.set_spo_config()
mx30.set_led_current()

print("-----------------------------------------------------------------
---------")
print("--------               PARAMED SENSOR READINGS
--------")
print("-----------------------------------------------------------------
---------")



try:
    while True:
        bpm = p.BPM
```

```
        mx30.read_sensor()
        if bpm > 0: # If bpm greater than 0 then print the current
heart rate pulse to the screen
            print("BPM: %d" % bpm)
        else:
            print("No Heartbeat found")

        if(mx30.red > 0 and mx30.ir > 0):
        #print("Raw IR Readings: "+str(mx30.ir))
        #print("Raw Red Readings: "+str(mx30.red))
        #print("")
        #print("Calculating SPO2....")
        #time.sleep(1)
            value = (float)(mx30.red)/(float)(mx30.ir)
            print("SPO2: "+str(value*100)+"%")
            print("")
        else:
            print("Finger not detected")

        time.sleep(2)
except:
    p.stopAsyncBPM()
```

7.2 Application code

Link to Application code:

https://github.com/N01263842/AACTech/tree/master/AACTech/aac_tech/src/main/java/aac_tech/automotiveui

Demo

/1 Hardware present?

/1 Memo by student A

/1 Login activity

/1 Data visualization activity

/1 Action control activity

Report

/1 Login activity

/1 Data visualization activity

/1 Action control activity

/1 Modified Code Files in Appendix

/1 Link to Complete Code in Repository