GNG 1103

Deliverable G: Prototype II & Customer Feedback

OPIOID OVERDOSE DETECTOR

By

GNG1103, SECTION C00, GROUP 5

Yi Ru Loh, 300134586
Mohamad Ahmad, 300108630
Dhvani Patel, 300121365
Sheetal Harakh, 300123274

Introduction

The purpose of this deliverable is to develop the second phase of the prototypes with improvements and more details that is built on the first phase of the prototype. Feedback from the client and our peer were reviewed and taken into consideration when developing prototype II to ensure that our product reflects the client's and user's needs as closely as possible. Based off of the first prototype, a simulation of the watch display(a hollow box structure) and a ring structure was modelled and 3D printed to allow it to fit the actual components that will be used such as a bluno nano arduino, touch screen display, circuit board, speaker, battery, etc. Tests were carried out to determine whether components are able to function when assembled on a circuit board. Besides that, codes for the bluetooth, sensor, and GPS system were written and tested individually before integrating all the codes into one for the next phase of the prototype.

This prototype leads us one step closer to our milestone of Design Day, as it allows our team to fully utilize and get the most out of the feedback that was received from the client as well as from the professor and our peers.

Client Feedback

On Friday, March 6, we presented a brief presentation of our current progress and future steps to the client, Tali Cahil, as well as the rest of the class. *Table 1* displays the feedback gathered from our presentation and our reflection upon it.

Table 1: Result from the In-Class Project Review

Presentation Attendee	Their Question/Feedback	Reflection
Tali Cahil	"How much do you think this will be? It seems like it will be expensive since you have 2 components."	At the moment, our project is cost effective because there is still 40% of our budget left.
Dr. Knox - course instructor	"So what if you don't have a smartphone?"	It may be a constraint but we believe that the majority of our users do own a smartphone due to their working requirement.

Classmate	“Have you tested the wires going from the ring to the watch itself and how mobile you can be with it on?”	In prototype II, we printed an improvised model for both the watch display and the ring that allows us to test out the functionality when a wire is connected between the ring and the watch display.

Discussion Regarding Client Meeting

From the feedback received by the client, the minor concern for cost was the one thing brought to our attention, along with how reliable and flexible the wire that connects the watch and ring would be, and the scenario where the user could possibly not have a cell phone to use. We ensured the client that the device is cost effective because we are left with roughly 40% of our budget after having most/all of our parts for the device. As for the wire, we intend to make it so that it is non-invasive and still does the job that it is intended to do, as well as keeping in mind that it needs to be durable for long use. A majority of the opioid users are known to be in the trades area regarding their working lives, therefore we assumed that a high number of them will have cellphones that can be used with the app.

Budget Analysis

As of today - March 8, 2020 - we have roughly half of our budget remaining. *Table 2* displays the purchases we have made so far (not from makerspace) as well as the amount we have remaining.

Table 2 - A budget analysis of our project as of March 8, 2020.

Date	Purchased Item	Cost of Item (CAD$)	Remaining Budget (CAD$)
Feb 28, 2020	MAX30100 Pulse Oximeter Chip	11.50	88.50
Feb 28, 2020	Bluno Nano Microcontroller	47.03	41.47

With around CAD$ 40 remaining, we are at a comfortable spot. We don't really have much left to buy, except the polymer we choose to 3D print with for our final design. Most of the investments we have to make are now regarding time.

An issue came up last week which could potentially impact our non-makerspace budget. The Bluno Nano device we purchased was quickly realized to be sub-optimal for our project. Most of our experience is in the Arduino IDE or with Arduino compatible devices. We are having some trouble with applying our software and hardware knowledge to the Bluno Nano system.

For this reason, we have decided to use an Arduino Nano (from makerspace) and file a ticket to return the Bluno Nano device. Although we may be reimbursed the $CAD 47.03 (if the vendor, Robotshop, accepts our return) we will have to pay shipping that will not be covered. This cost will likely be just 10-20 $CAD, however it is quite a letdown since this money is genuinely wasted.

If Robotshop accepts our return, our remaining non-makerspace budget is estimated to be CAD $73.50

Software Subsystem

Our current prototype consists of code that creates a connection between each part of our design. This means that we have been meshing different codes related to different systems (ex: oximeter, speaker, touchscreen, arduino, etc) into one homogenous code. This is important because we can't have 3 or 4 different programs running at the same time, but rather we aim to have a single program which fluidly relates the information from the oximeter with the speaker and screen's output - all of which is interpreted by the arduino nano. *Appendix A* displays our current coding progress.

The current code shown in *Appendix A*, is responsible for controlling the oximeter aspect of our design. The code utilizes a previously created library called *Wire* and uses its components and variables to make a functioning oximeter. The programming regarding the actual sensor is also from a library called *MAX30100,* which specifically works with using the sensor component we had purchased. A large part of the code is refreshing and constantly updating the values to ensure the pulse ox reading are as accurate as possible. For the future, more coding will be required to ensure the values from the sensor are sent to the arduino and finally the display. The values will also need to be used in order to determine whether the user is overdosing and follow through with the emergency contact alert.

Hardware Subsystem

Our hardware as of now is all present. However there are some issues with some components.

Table 3: displays our hardware components and the issues they have (if applicable).

Component	Issue(s)	Solution
Bluno Nano Microcontroller	-May not be compatible with all of our other components -Not optimal for this course since most our our training was regarding the Arduino IDE	-We have decided to use an Arduino Nano from makerspace. This will be much more comfortable for us and give us a higher chance of making a working model.
OLED Touchscreen	-Way too large	-Use a non-touchscreen system which is much smaller. Sheetal already has two of these which is great in case one of them fails.
MAX30100 Pulse Oximter	-Not working very consistently. It might turn on for a few seconds but then that's all.	-Some troubleshooting information we found online was to de-solder 3 micro-resistors. We did this and the same issues persisted. We will look into our code to see if there are issues there.
Micro-Speaker	-The system is no longer	-We will go into makerspace

	working for some reason.	again this week and see if we can get another one (hopefully a different model which doesn’t spontaneously fail).

Device Housing Subsystem

Some more 3D models were printed for this prototype (physical and CAD models displayed in *Appendix* C). The following issues were found with the models:

- The ring (responsible for housing the oximeter) has too small of a cutout to actually fit the sensor in it.
- The watch-housing model (responsible for containing all components except the oximeter) was found to be too small to contain the oversized screen aforementioned.

In order to resolve these problems, we plan on taking more accurate measurements and re-modelling and printing the refined systems in aims of having a better fit.

Remaining Work

As mentioned in the Software Subsystem section, some code needs to be looked over and added in order to make sure that everything is pristine.

With regards to hardware, *Table 3* really sums up our issues and our plans to resolve them (if they’re not already taken care of).

For the device housing systems, we just have to keep taking measurements and experimenting with different CAD models and 3D models - seeing if everything fits and if everything is comfortable to wear.

A big thing that we have to start working on is the android app which will be responsible for displaying all information gathered by the oximeter - and then sending it to an emergency contact if SpO2 readings are below 90%.

We have yet to start actually coding and testing the app, however we have some great fundamentals already established. *Appendix D* displays some code which is used to display oximeter data from a MAX300100 sensor, as well as code used to automatically send an SMS (as well as where we got the information from, of course). We plan on merging the two codes into one app.

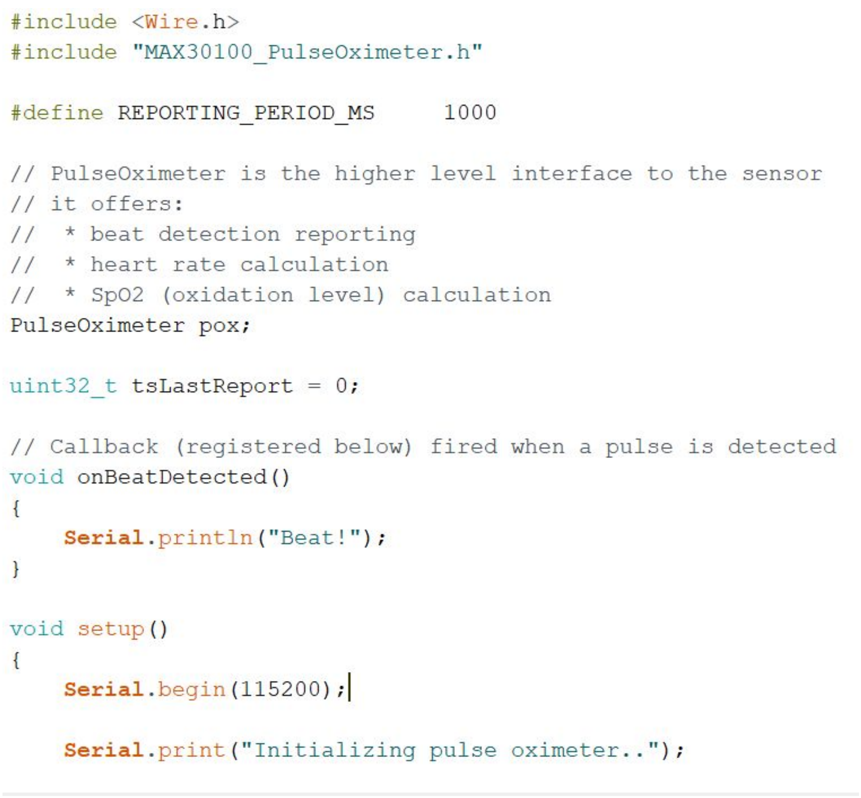
This will require once again, a lot more coding and a lot of testing (especially when it comes to bluetooth connection, accuracy of information, and time it takes to send an SMS to the emergency contact).

Conclusion

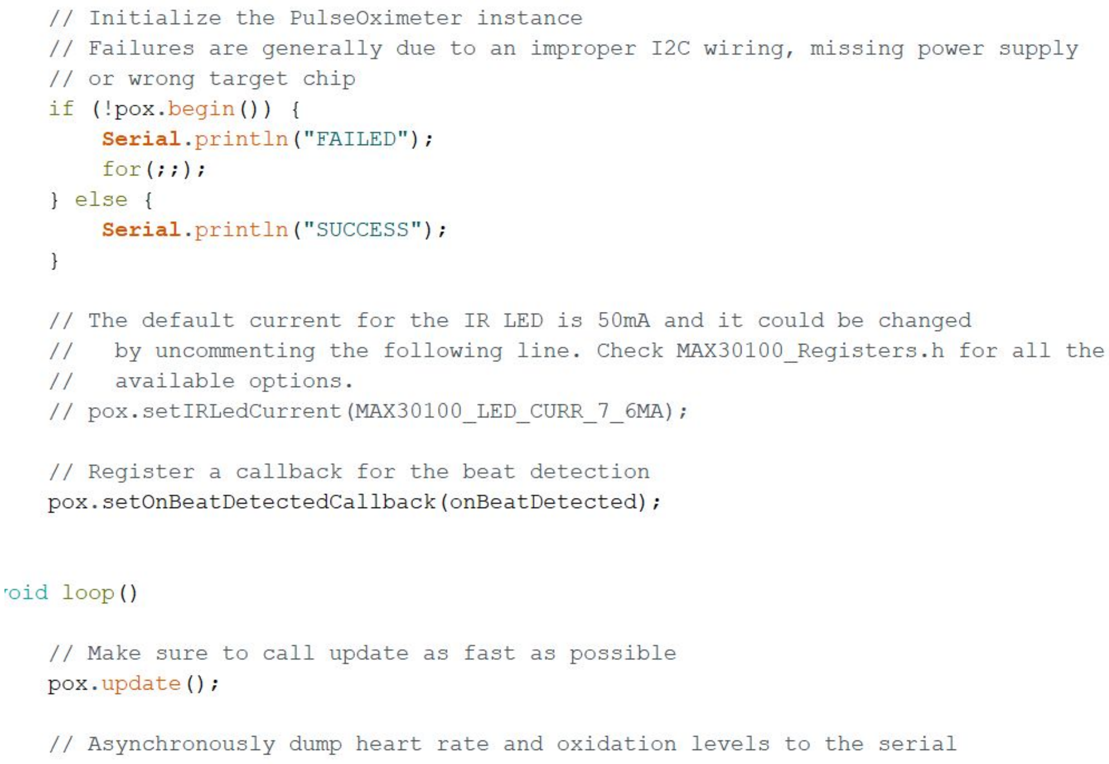
Unfortunately, during our second round of prototyping we ran into a problem with the bluno nano that was intended to be used for our device in order to replace an arduino. This set back will most likely put us behind schedule and require us to put more time than planned originally on a regular basis. However, we plan to get back on schedule as we have decided to use an arduino nano instead of the bluno nano that is incompatible with recognizing code that the arduino nano identifies and runs easily. Now that we have figured out what does and doesn’t work for our prototype, we are easily able to collaborate and divide the responsibilities. This would mean that the assigned task for each individual can be done on their own time and might require them to put more of their time outside of the lab session in order to stay on schedule and not fall behind.

Appendices

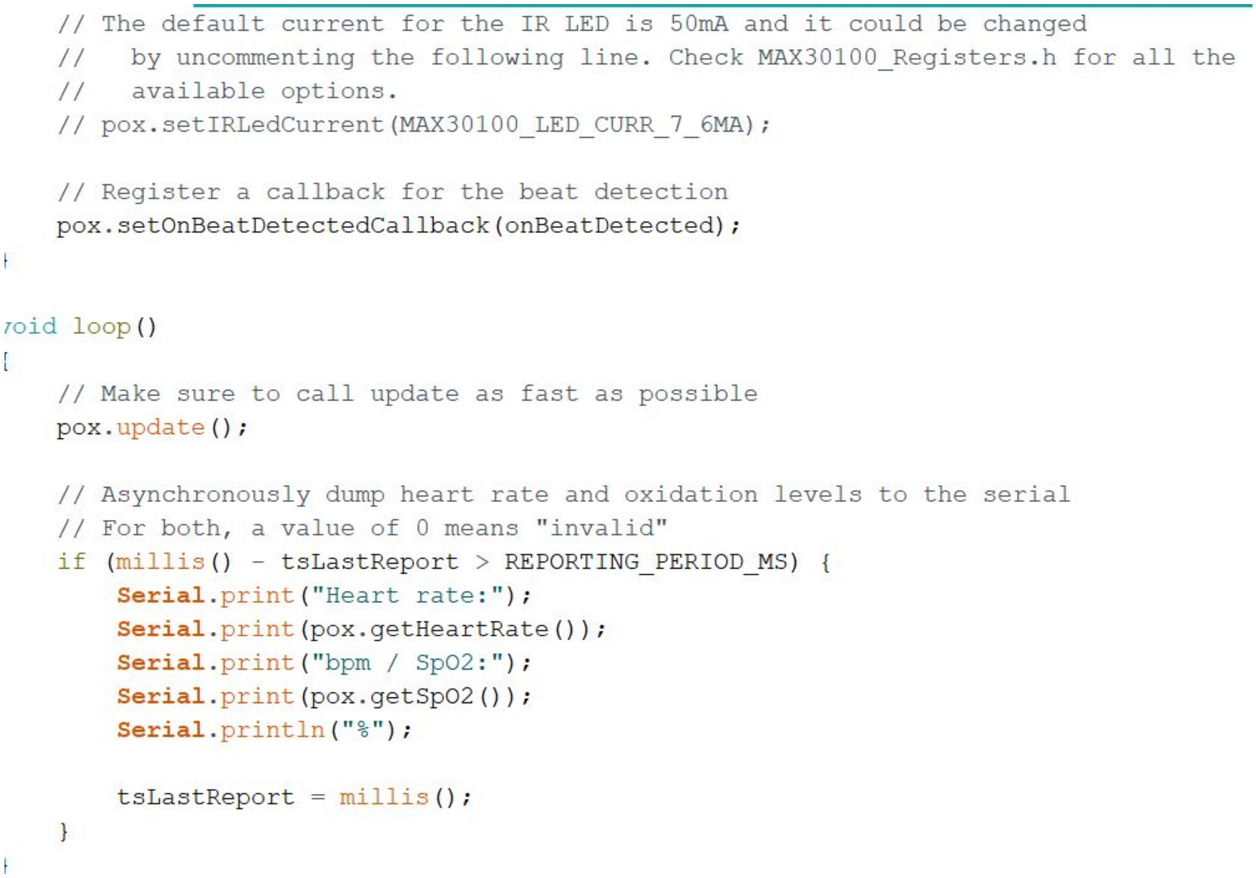
Appendix A: Our current code is done for purposes of linking all hardware components together into one program.



```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS     1000

// PulseOximeter is the higher level interface to the sensor
// it offers:
//  * beat detection reporting
//  * heart rate calculation
//  * SpO2 (oxidation level) calculation
PulseOximeter pox;

uint32_t tsLastReport = 0;

// Callback (registered below) fired when a pulse is detected
void onBeatDetected()
{
    Serial.println("Beat!");
}

void setup()
{
    Serial.begin(115200);

    Serial.print("Initializing pulse oximeter..");
```

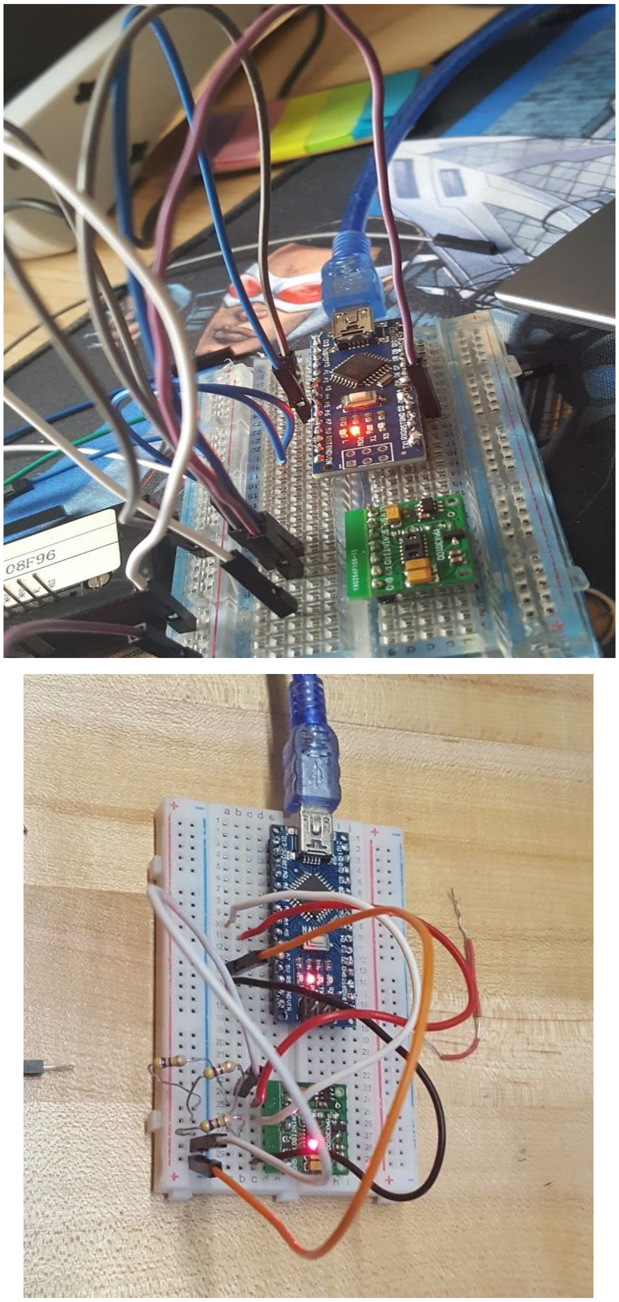


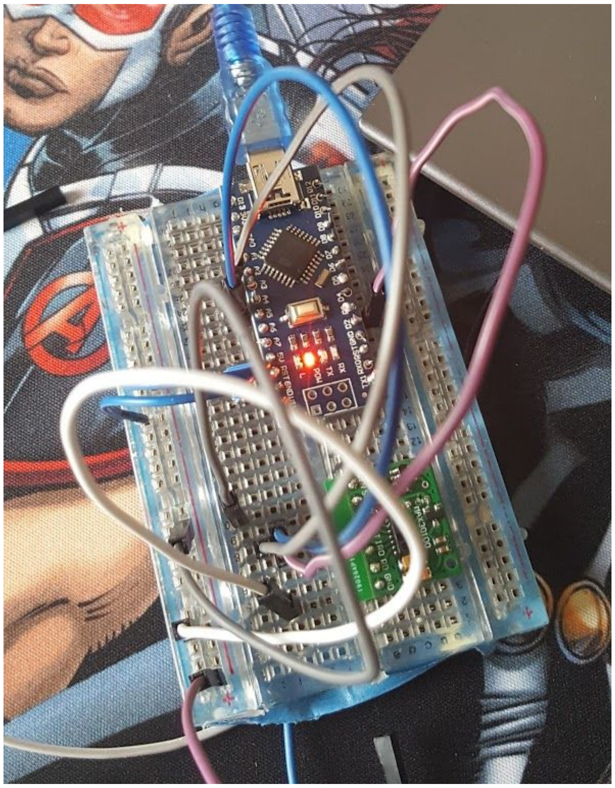
```
    // Initialize the PulseOximeter instance
    // Failures are generally due to an improper I2C wiring, missing power supply
    // or wrong target chip
    if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }

    // The default current for the IR LED is 50mA and it could be changed
    //   by uncommenting the following line. Check MAX30100_Registers.h for all the
    //   available options.
    // pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

    // Register a callback for the beat detection
    pox.setOnBeatDetectedCallback(onBeatDetected);


void loop()

    // Make sure to call update as fast as possible
    pox.update();

    // Asynchronously dump heart rate and oxidation levels to the serial
```



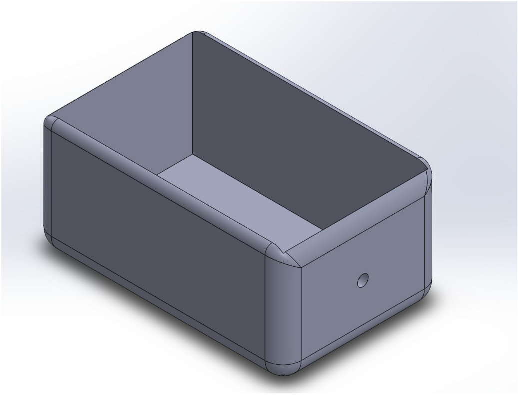
```
    // The default current for the IR LED is 50mA and it could be changed
    //   by uncommenting the following line. Check MAX30100_Registers.h for all the
    //   available options.
    // pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

    // Register a callback for the beat detection
    pox.setOnBeatDetectedCallback(onBeatDetected);
}

void loop()
{
    // Make sure to call update as fast as possible
    pox.update();

    // Asynchronously dump heart rate and oxidation levels to the serial
    // For both, a value of 0 means "invalid"
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        Serial.print("Heart rate:");
        Serial.print(pox.getHeartRate());
        Serial.print("bpm / SpO2:");
        Serial.print(pox.getSpO2());
        Serial.println("%");

        tsLastReport = millis();
    }
}
```

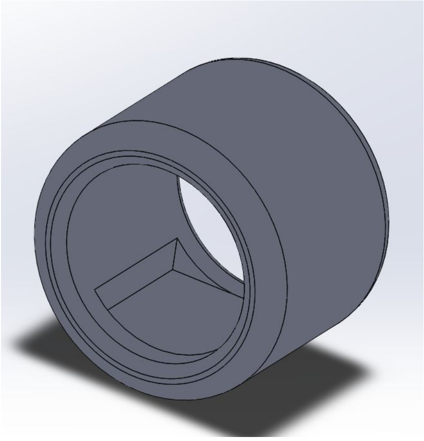
Appendix B: Our current wiring configurations linking all hardware components together.





Appendix C: Housing systems for the components. These are the devices which the user will wear (ring and watch).





Appendix D: Code and programs to be used for the Android app, as well as the source we retrieved the resources from.

I - Code to send SMS. Source: https://programmerworld.co/android/how-to-send-sms-automatically-from-your-phone-by-programming-in-android-studio-java-code/

```
package com.example.mysendsmsapp;

import android.Manifest;
import android.content.pm.PackageManager;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

private EditText editTextNumber;
private EditText editTextMessage;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.SEND_SMS, Manifest.permission.READ_SMS},
PackageManager.PERMISSION_GRANTED);

editTextMessage = findViewById(R.id.editText);
editTextNumber = findViewById(R.id.editTextNumber);
}

public void sendSMS(View view){

String message = editTextMessage.getText().toString();
String number = editTextNumber.getText().toString();

SmsManager mySmsManager = SmsManager.getDefault();
mySmsManager.sendTextMessage(number,null, message, null, null);
}
}
```

```
<?xml version=”1.0″ encoding=”utf-8″?>
<manifest xmlns:android=”http://schemas.android.com/apk/res/android&#8221;
package=”com.example.mysendsmsapp”>

<uses-permission android:name=”android.permission.SEND_SMS”/>
<uses-permission android:name=”android.permission.RECEIVE_SMS”/>
<uses-permission android:name=”android.permission.READ_SMS”/>

<application
android:allowBackup=”true”
android:icon=”@mipmap/ic_launcher”
android:label=”@string/app_name”
android:roundIcon=”@mipmap/ic_launcher_round”
android:supportsRtl=”true”
android:theme=”@style/AppTheme”>
<activity android:name=”.MainActivity”>
<intent-filter>
<action android:name=”android.intent.action.MAIN” />

<category android:name=”android.intent.category.LAUNCHER” />
</intent-filter>
</activity>
</application>

</manifest>
```

II - Android App to display oximeter values on smartphone. Source: https://how2electronics.com/blood-oxygen-heart-rate-monitor-max30100-arduino#The_Android_App

Link to google drive folder with apk files:
https://drive.google.com/file/d/1_s5UopncZnD7AluwccJbL8d6qNV9YIoN/view

Link to google drive folder with aia files:
https://drive.google.com/file/d/1OlbVzFfikyjr9XoBSF2fhOZeH1JRhgTn/view