TUGAS AKHIR

RANCANG BANGUN PULSE OXIMETRY DIGITAL BERBASIS RASPBERRY PI 3

Diajukan Untuk Melengkapi Tugas-Tugas Dan sebagai Persyaratan Memperoleh gelar Sarjana Teknik (S.T) Pada Program Studi Teknik Elektro Fakultas Teknik Universitas Muhammadiyah Sumatera Utara

Oleh :

PANDE GUNA KUSWARA

1207220029

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
MEDAN
2019

HALAMAN PENGESAHAN

Tugas akhir ini diajukan oleh :

Nama	: Pande Guna Kuswara
NPM	: 1207220029
Program Studi	: Teknik Elektro
Judul Skripsi	: Rancang Bangun Pulse Oximetry Digital Berbasis Raspberry Pi

Telah berhasil dipertahankan di hadapan Tim Penguji dan diterima sebagai salah satu syarat yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Muhammadiyah Sumatera Utara.

Medan, 07 Oktober 2019

Mengetahui dan menyetujui :

Dosen Pembimbing I

Dr. Ir. Suwarno, M.T

Dosen Pembimbing II

Elvy Shahnur, S.T, M.Pd

Dosen Pembanding I

Faisal Irsan P, S.T, M.T

Dosen Pembanding II

Partaonan Harahap, S.T, M,T

Program Studi Teknik Elektro

Ketua

Faisal Irsan P, S.T, M.T

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Saya yang bertanda tangan di bawah ini:

Nama Lengkap	: Pande Guna Kuswara
Tempat/Tanggal Lahir	: Bayur Sumbar / 01 September 1994
NPM	: 1207220029
Fakultas	: Teknik
Program Studi	: Teknik Elektro

Menyatakan dengan sesungguhnya dan sejujurnya, bahwa laporan Tugas Akhir saya yang berjudul :

"Rancang Bangun Pulse Oximetry Digital Berbasis Raspberry Pi"

Dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Skripsi ini adalah asli dari pemikiran saya, tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di salah satu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Medan, 07 Oktober 2019

Sava yang menyatakaan,

PANDE GUNA KUSWARA

KATA PENGANTAR

Pertama-tama perkenankanlah saya memanjatkan puji syukur kehadapan ALLAH S.W.T. Tuhan Yang Maha Esa, karena hanya atas anugrah-Nya laporan tugas akhir yang berjudul "*RANCANG BANGUN PULSE OXIMETRY DIGITAL BERBASIS RASPBERRY PI* " ini dapat diselesaikan.

Dalam penyusunan laporan tugas akhir ini, penulis banyak memperoleh petunjuk dan bimbingan dari berbagai pihak. Sehingga pada kesempatan ini perkenankanlah saya mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Bapak Faisal Irsan Pasaribu, S.T., M.T. selaku kepala Program Studi Teknik Elektro Fakultas Teknik Universitas Pamulang.
2. Bapak Partaonan Harahap, S.T., M.T. selaku sekretaris Program Studi Teknik Elektro Fakultas Teknik Universitas Pamulang.
3. Bapak Dr. Ir. Suwarno, M.T. selaku pembimbing I.
4. Bu Elvy Shahnur S.T.,M.T. selaku pembimbing II.
5. Kedua orang tua tersayang. Terima kasih atas doa, kasih sayang, semangat serta dukungan yang telah diberikan hingga sekarang.
6. Rekan-rekan mahasiswa yang membantu dan mendukung peulisan proposal tugas akhir ini.

Penulis menyadari bahwa laporan tugas akhir ini masih jauh dari sempurna. Oleh karena itu segala kritik dan saran yang bersifat membangun sangat diharapkan demi kesempurnaan penyusunan laporan tugas akhir ini. Semoga ALLAH S.W.T. Tuhan Yang Maha Esa selalu melimpahkan rahmat-Nya kepada semuah pihak yang telah membantuh pelaksanaan dan penyelesaian laporan tugas akhir ini.

Medan, September 2019

Penulis

ABSTRAK

Pulse Oximetry adalah sebuah metode *non-invasive* untuk memonitor oksigen saturasi dalam darah. Saturasi oksigen dalam darah dinyatakan dalam presentase kejenuhan dari total hemoglobin (SpO_2). Metode ini memanfaatkan perbedaan panjang gelombang dari cahaya merah (660 nm) dan cahaya inframerah (940 nm) yang ditangkap oleh sensor cahaya setelah melewati pembuluh balik dan pembuluh kapiler pada ujung jari telunjuk. Dalam penelitian ini digunakan sebuah LED merah dan inframerah yang diarahkan ke ujung jari dan cahaya yang melewati pembuluh darah ditangkap oleh fotodioda. Sinyal dari fotodioda dikirimkan ke Raspberry pi dan diolah untuk mendapatkan persentase dari oksigen saturasi dalam darah.

Kata kunci : *Pulse Oximetry, non-invasive,* SPO_2 dan Raspberry Pi.

ABSTRACT

Pulse Oximetry is a non-invasive method for monitoring blood oxygen saturation. The oxygen saturation in blood is expressed in percentage of saturation of total hemoglobin (SpO2). This method utilizes the different wavelengths of red light (660 nm) and infrared light (940 nm) captured by the light sensor after passing through vessels and capillaries at the tip of the index finger. In this study used a red and infrared LED that is directed to the fingertips and light passing through the blood vessels captured by photodiode. The signal from the photodiode is sent to Raspberry pi and processed to get a percentage of oxygen saturation in the blood.

Kata kunci : *Pulse Oximetry, non-invasive,* SPO_2 and Raspberry Pi.

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang

Oksigen merupakan salah satu unsur yang sangat penting bagi tubuh manusia. Dikutip dari laman web medicalogy.com (2017), oksigen melalui serangkaian proses untuk mencapai sel-sel yang ada dalam tubuh. Pertama, oksigen dihirup dan masuk ke saluran pernapasan untuk kemudian masuk ke dalam kantong-kantong kecil dalam paru-paru bernama Alveoli. Alveoli memiliki dinding tipis yang dikelilingi oleh pembuluh darah, sehingga oksigen mampu menembus dinding tipis ini kemudian masuk kedalam peredaran darah. Bersamaan dengan masuknya oksigen ke dalam peredaran darah, karbon dioksida juga dilepaskan dari peredaran darah dan menembus dinding Alveoli, untuk kemudian dikeluarkan dalam tubuh [1].

Karena sangat penting fungsi oksigen dalam tubuh, maka informasi mengenai kadar oksigen yang terikat dalam darah sangatlah penting untuk diketahui. Terlebih untuk pasien perawatan intensif dan monitoring pasien anesthesia. Menurut Umi Salamah (2015:60) jika tubuh manusia kekurangan oksigen, maka akan menimbulkan penyakit dan gangguan fungsi kerja tubuh yang lain. Beberapa penyakit yang ditimbulkan diantaranya adalah hipoksemia, anemia dan lain sebagainya. Pada tingkat tertentu, penyakit tersebut dapat menimbulkan resiko kematian. Oleh karena itu informasi mengenai kadar oksigen yang terikat dalam darah menjadi sangat penting, karena akan menjadi salah satu pertimbangan untuk menentukan kebijakan klinis pada pasien [2].

Oksigen yang terdapat pada peredaran darah diikat oleh salah satu komponen penyusun darah yaitu Hemoglobin (Hb). Transportasi oksigen dalam darah ada dua bentuk yaitu yang terlarut dalam plasma dan terikat dengan hemoglobin. Normalnya, sekitar 97% oksigen yang ditransport dari paru-paru ke jaringan terikat dengan hemoglobin dan sisanyanya 3 % terlarut dalam plasma. Oleh karena itu, maka warna darah yang banyak mengikat oksigen dengan yang sedikit mengikat oksigen akan terlihat

berbeda. Berdasarkan hal tersebut maka jika kadar warna merah darah tersebut dapat diketahui maka dapat diketahui pula kadar oksigen yang terlarut dalam darah. Jika terdapat sebuah sumber cahaya menembus bagian tubuh manusia yang tipis dan menggambarkan bagaimana spektrum warna darah dalam kulit tersebut, maka akan dapat diketahui kadar oksigen dalam darah orang tersebut [2].

1.2. Rumusan Masalah

Rumusan masalah yang menjadi pokok bahasan pada penelitian ini adalah bagaimana cahaya yang ditangkap oleh fotodioda dapat diproses oleh Raspberry Pi, sehingga menunjukkan nilai saturasi oksigen dalam darah.

1.3.Batasan Masalah

Untuk memfokuskan pembahasan pada laporan tugas akhir ini, maka masalah yang dibahas hanya berfokus pada pemrosesan sinyal yang dihasilkan oleh fotodioda di Raspberry Pi.

1.4. Tujuan Penelitian

Adapun tujuan dilakukannya penelitian ini adalah sebagai berikut :

1. Mempraktekan ilmu yang didapat dari kegiatan perkuliahan untuk bidang instrumentasi medis.
2. Membuat alat *Pulse Oximetry* yang relatif lebih murah dibanding yang dijual pada umumnya.

1.5. Metode Penelitian

Agar penelitian ini berjalan dengan lancar dan mendapatkan hasil yang baik maka diperlukan metode-metode yang terstruktur dalam melakukan penelitian ini. Metode-metode yang digunakan dalam penelitian ini adalah sebagai berikut :

1. Studi Literatur

   Metode ini dilakukan dengan cara mengumpulkan referensi-referensi yang terkait dengan penelitian ini.

2. Perancangan alat

   Pada tahap ini penulis merancang sebuah rancang bangun *Pulse Oximetry* digital berbasis Raspberry Pi yang kemudian alat tersebut diuji coba agar mendapat hasil yang cukup akurat.

3. Pengujian

Setelah alat dibuat, maka dilakukan pengujian untuk mengetahui apakah alat berfungsi dengan baik atau tidak dan apakah hasilnya sesuai dengan yang diharapkan atau tidak yaitu dapat membaca nilai saturasi oksigen dalam darah.

1.6. Sistematika Penulisan

Penulisan skripsi ini terbagi menjadi 5 bab yang terdiri dari :

BAB I : PENDAHULUAN

Bab ini akan menguraikan tentang latar belakang, rumusan masalah, batasan masalah, metode penelitian dan sistematika penulisan.

BAB II: DASAR TEORI

Bab ini akan menguraikan tentang teori-teori yang dijadikan landasan dalam penelitian ini.

BAB III : METODE PENELITIAN

Bab ini berisi proses akan dijelaskan mengenai detail langkah-langkah yang harus dilalui untuk mencapai tujuan dan simpulan akhir dari penelitian ini.

BAB IV : DATA DAN ANALISA

Bab ini berisi data hasil pengukuran tegangan pada alat serta analisanya dan presentase SPO_2 sebagai hasilnya.

BAB V: KESIMPULAN DAN SARAN

Bab ini berisi simpulan dari data yang telah dianalisa saran-saran yang dapat digunakan pada penelitian lebih lanjut.

BAB 2
DASAR TEORI

2.1. Pengertian Pulse Oximetry

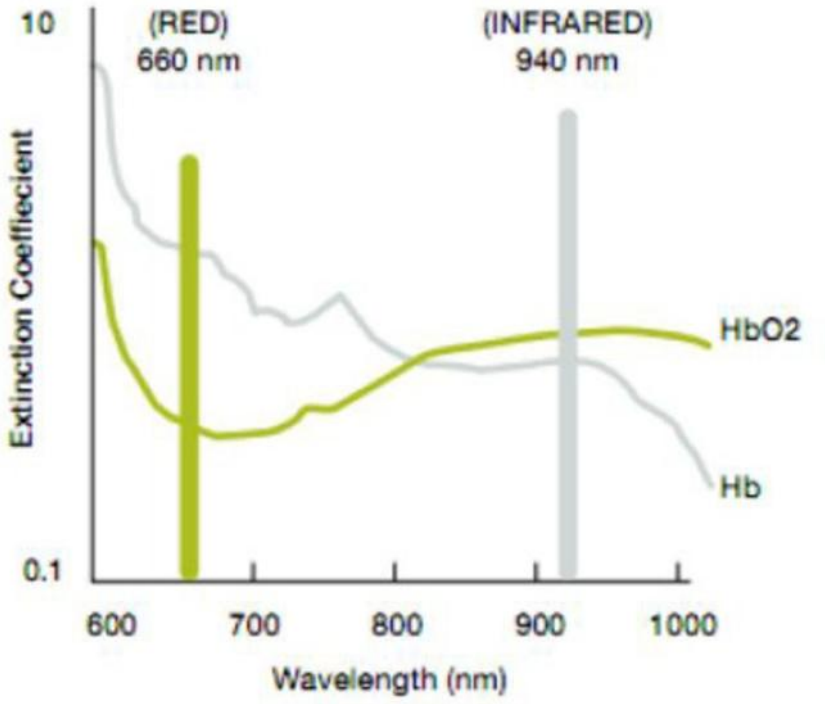
Pulse Oximetry adalah sebuah metode *non-invasive* untuk memonitor oksigen saturasi dalam darah. Menurut Andrey (2011:1-2) saturasi adalah persentase dari jumlah hemoglobin yang mengikat oksigen dibanding dengan total hemoglobin keseluruhan. Saturasi oksigen dalam darah dinyatakan dalam persentase kejenuhan dari total hemoglobin atau disebut dengan SPO_2 (*Saturated Peripheral Oxygen*). Metode ini memanfaatkan perbedaan panjang gelombang dari cahaya merah (660 nm) dan cahaya inframerah (940 nm) yang ditangkap oleh sensor cahaya setelah melewati pembuluh balik dan pembuluh kapiler pada ujung jari telunjuk [3].

Cara kerja alat ini adalah dengan cara membandingkan intensitas cahaya yang diserap oleh fotosensor setelah melewati ujung jari dan berinteraksi dengan sel darah merah yang mengalir pada ujung jari. Cahaya merah dan inframerah yang melewati ujung jari mengalami pengurangan intensitas cahaya yang berbeda. Absopsi dan pengurangan intensitas cahaya dari cahaya merah dan inframerah ini dibandingkan dan diolah agar didapat nilai persentasi saturasi oksigen dalam darah (SpO2) [3].

2.2. Metode Pulse Oxymetry

Sensor pulse oximetry menggunakan cahaya untuk pengukuran saturasi oksigen, yaitu menentukan kuantitas salah satu komponen darah (hemoglobin). Menurut Guruh (2013:2) saturasi oksigen adalah persentase hemoglobin yang mengandung oksigen. Sensor ditempatkan pada jaringan tubuh yang tipis seperti ujung jari atau daun telinga. Transmisi cahaya melalui arteri adalah denyutan yang diakibatkan pemompaan darah oleh jantung [4].

Pulse Oximetry bekerja berdasarkan kondisi dimana Hemoglobin yang mengikat Oksigen (HbO2) menyerap cahaya dengan panjang gelombang yang berbeda dengan Hemoglobin yang tidak mengikat oksigen (Hb), seperti terlihat pada gambar 2.1 berikut ini.



Gambar 2.1 Grafik absorbsi cahaya oleh Hemoglobin [10].

Menurut Andrey (2011:3) oksigen saturasi pada arteri ditentukan sebagai perbandingan HbO2 dengan total Hb yang tersedia pada arteri. Perbandingannya dapat dilihat pada persamaan berikut ini :

$$SpO_2 = \frac{[Hb0_2]}{[Hb0_2]+[Hb]} \quad \text{.....(2.1)}$$

Keterangan :

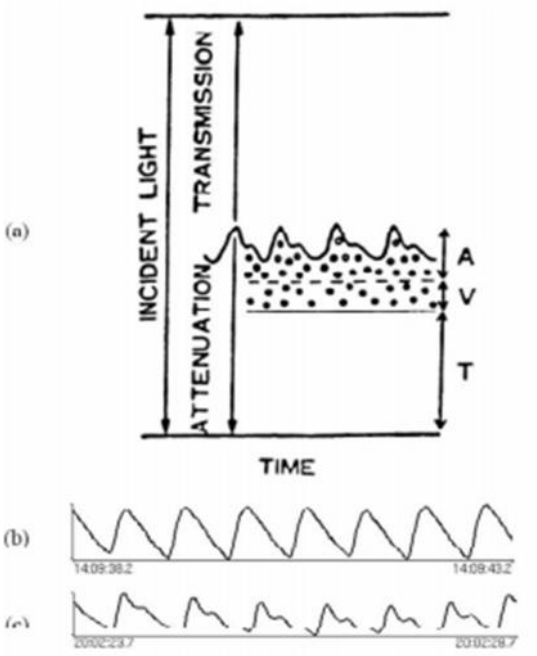
SPO_2 = Presentase saturasi oksigen.

HbO_2 = Hemoglobin yang mengandung oksigen.

Hb = Hemoglobin yang tidak mengandung oksigen.

Alat Pulse Oxymetri menggunakan LED (Light Emitting Diode) merah dan inframerah secara bersama-sama dengan fotosensor

untuk mengukur intensitas cahaya yang telah melewati ujung jari. Intensitas cahaya yang telah melewati ujung jari mengalami pengurangan intensitas. Pengurangan intensitas ini disebabkan oleh aliran darah pada vena, aliran darah pada arteri dan jaringan. Pengurangan intensitas cahaya oleh aliran darah vena dan jaringan menghasilkan sinyal yang relatif stabil, yaitu berupa sinyal DC. Pengurangan intensitas cahaya oleh aliran darah arteri menghasilkan sinyal yang relatif tidak stabil, yaitu berupa sinyal AC. Menurut Guruh (2013:4) penyerapan cahaya merah lebih dari cahaya inframerah adalah indikasi dari saturasi oksigen yang rendah, dan sebaliknya penyerapan cahaya inframerah lebih dari cahaya merah merupakan indikasi dari saturasi oksigen yang tinggi [4]. Pengurangan intensitas cahaya tersebut dapat dilihat pada gambar 2.2 berikut ini.



Gambar 2.2 (a) Transmisi cahaya melalui jari ketika pengurangan cahaya disebabkan oleh darah arteri (A), darah vena (V) dan jaringan (T). (b) dan (c) menunjukan sinyal denyutan dari pengurangan cahaya setelah komponen pengurangan cahaya oleh vena dan jaringan dihilangkan [3].

Menurut Andrey (2011:3) untuk menghitung nilai dari oximetry dibutuhkan untuk mencari nilai R. R adalah perbandingan dari penyerapan cahaya merah dan inframerah yang menghasilkan komponen AC dan DC.

$$R = \frac{AC_{red}/DC_{red}}{AC_{ired}/DC_{ired}} \quad \text{.....(2.2)}$$

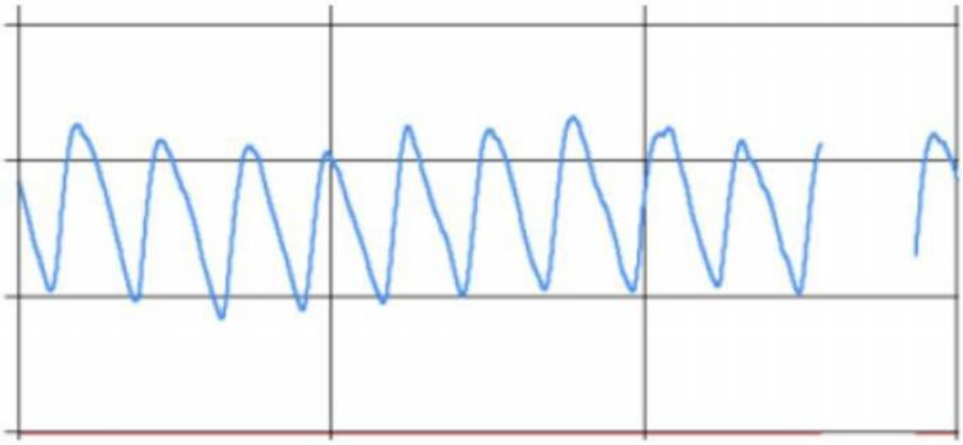
Setelah nilai R diketahui, maka dapat ditentukan nilai dari SpO2 dengan memasukan nilai R ke persamaan linear berikut ini.

$$SpO2 = 110 - 25R \text{.....(2.3)}$$

Keterangan :

R = Rasio tegangan hasil penyerapan cahaya merah dan infra merah.
AC_{red} = Nilai tegangan AC dari penyerapan cahaya merah.
DC_{red} = Nilai tegangan DC dari penyerapan cahaya merah.
AC_{ired} = Nilai tegangan AC dari penyerapan cahaya infra merah.
DC_{ired} = Nilai tegangan DC dari penyerapan cahaya infra merah.
SPO_2 = Presentase saturasi oksigen.

Sinyal oxymetri secara tipikal yang ditampilkan dapat dilihat pada gambar 2.3. Sinyal yang ditampilkan pada gambar 2.3 merupakan representasi dari denyutan absorbsi pada arteri.

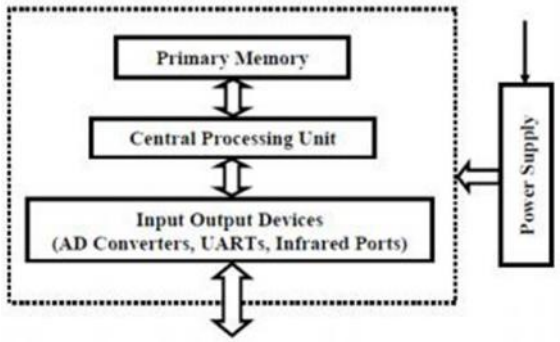


Gambar 2.3 Sinyal Oxymetry [10].

2.3. Sistem Tertanam (*Embedded System*)

Sistem tertanam (*embedded system)* adalah suatu sistem komputer yang mempunyai fungsi spesifik tertentu dalam skala yang bervariasi

dalam perangkat keras maupun perangkat lunaknya. Sistem *embedded* berfungsi melaksanakan tugas-tugas spesifik yang telah didefinisikan terlebih dahulu sehingga berbeda dengan PC (*personal computer*) yang bersifat *multi purpose*. *Embedded system* merupakan bagian dari sistem yang lebih besar. Salah satu contoh aplikasinya adalah dalam suatu sistem instrumentasi medik (*medical instrumentation*). Berikut ini merupakan diagram struktur dari sistem tertanam.



Gambar 2.4 Struktur sistem tertanam [14].

2.3.1. Kategori Sistem Tertanam

1. **Mandiri (*Standalone*)**

   Sebuah sistem tertanam yang ada pada kategorri ini dapat bekerja secara independen tanpa perangkat lainnya. Dapat juga merujuk kepada perangkat lunak yang dapat bekerja tanpa perangkat lunak lainnya.

2. ***Real Time***

   Suatu sistem tertanam yang bekerja dengan spesifikasi waktu tertentu disebut dengan *real time*. Sistem *real time* terdiri dari sistem *hard real time* dan *soft real time*. Pada sistem *hard real time*, sistem tersebut harus melaksanakan tugas dengan *deadline* waktu tertentu, sedangkan pada sistem *soft real time* sistem tersebut tidak mengharuskan adanya *deadline* dalam melaksanakan tugas, sehingga menyebabkan adanya penundaan atau *delay*.

3. **Networked**

Sistem tertanam yang memiliki antarmuka jaringan dan dapat diakses oleh jaringan lokal ataupun internet disebut dengan *Networked Information Applience*. Contoh dari sistem ini adalah sebuah kamera yang dapat mengirimkan gambar secara langsung melalui jaringan lokal ataupun internet menuju perangkat tertentu yang telah ditentukan sebelumnya.

4. **Mobile Devices**

Perangkat *mobile* seperti PDA, *smartphone*, *smartwatch*, dan lain-lain adalah sistem tertanam yang termasuk ke kategori *mobile devices. Mobile devices* bisa dikatakan sebagai sistem tertanam karena masih memiliki kendala dalam penyimpanan dan kurangnya performa, walaupun dapat menjalankan banyak aplikasi atau bersifat *general purpose*.

2.3.2. Perangkat Sistem Tertanam

1. Single Board Computer

Single Board Computer (SBC) adalah sebuah perangkat komputer yang disusun dalam sebuah papan tunggal *printed circuit board* (PCB). Komputer tersebut sudah dilengkapi *processor*, *random acces memory* (RAM), I/O, ADC/DAC dan fitur-fitur lainnya yang diperlukan untuk menjalankan fungsinya sebagai sebuah komputer dalam satu papan.

Sebuah SBC dapat diimplementasikan dalam berbagai sistem tertanam, seperti dalam sistem kendali waktu nyata, pemantauan ataupun antarmuka I/O.

Raspberry Pi adalah sebuah komputer papan tunggal (*Single Board Computer*) yang berukuran kecil hampir menyamai ukuran kartu kredit. Raspberry pi dapat digunakan untuk menjalankan program perkantoran, permainan komputer, memutar audio dan video serta dapat berinteraksi dengan peralatan masukan dan keluaran melalui port-port GPIO

(*General Purpose Input Output*). Rapberry Pi dibuat dan dikembangkan oleh sebuah yayasan Nirlaba bernama *Raspberry Pi Fondation* dari Universitas Cambridge, Inggris.

Pada tugas akhir ini *Single Board Computer* yang digunakan adalah Raspberry Pi 3 Model B. Alasan dipilihnya Raspberry Pi tipe ini adalah keandalannya untuk menerima dan mengolah data dari sensor untuk kemudian langsung menampilkan data hasil pengolahan berupa persentasi saturasi oksigen (SpO2). Gambar dari Raspberry Pi yang digunakan dalam tugas akhir ini adalah sebagai berikut :

Gambar 2.5. Raspberry Pi 3 Model B

2. Mikrokontroler

Mikrokontroler adalah sebuah mikroprosesor yang didalamnya terdapat CPU, RAM (*Random Access Memory*), ROM (*Read Only Memory*), I/O dan *clock* yang terintegrasi dalam satu *chip*. Mikrokontroler merupakan perangkat dari sistem tertanam karena memiliki fungsi khusus tertentu sesuai dengan program yang diisikan.

Pada tugas akhir ini, mikrokontroler yang digunakan adalah Arduino Uno R3. Alasan digunakannya Arduino ini adalah karena keandalannya untuk memproses sinyal-sinyal melalui jalur komunikasi I2C (*Inter Integrated Circuit*), karena memiliki pin SDA (*Serial Data*) dan SCL (*Serial Clock*). Gambar dari mikrokontroler Arduino Uno R3 yang digunakan dalam tugas akhir ini adalah sebagai berikut :

Gambar 2.6 Arduino Uno R3.

2.1. Komunikasi I2C (*Inter Integrated Circuit*)

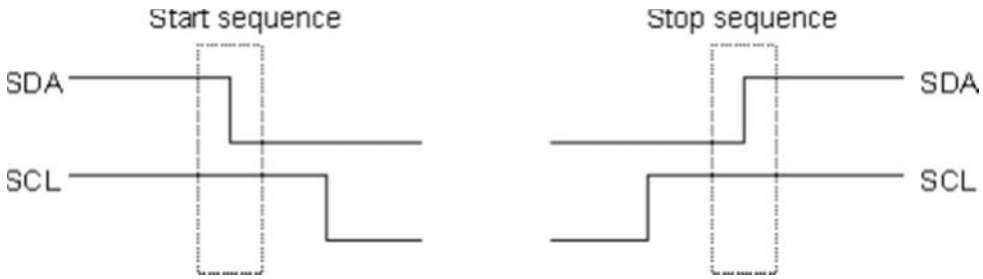
I2C (*Inter Integrated Circuit*) merupakan protokol komunikasi serial yang diciptakan oleh *Philips* untuk komunikasi antar perangkat-perangkat digital seperti *Single Board Computer*, *Embedded System,* dan lain-lain. Protokol komunikasi ini hanya memiliki dua jalur, yaitu SDA (*Serial Data*) sebagai jalur untuk data dan SCL (*Serial Clock*) sebagai jalur untuk *clock.* Peralatan-peralatan yang menggunakan protokol komunikasi I2C mempunyai sifat *serial synchronuous half duplex bidirectional* yang berarti data dalam protokol komunikasi tersebut dikirim secara serial melalui jalur SDA, menggunakan jalur bergantian dalam transmisi data, dan dikirim dari sebuah perangkat ke perangkat lainnya.

2.4. Komunikasi I2C (*Inter Integrated Circuit*)

I2C (*Inter Integrated Circuit*) merupakan protokol komunikasi serial yang diciptakan oleh *Philips* untuk komunikasi antar perangkat-perangkat digital seperti *Single Board Computer*, *Embedded System,* dan lain-lain. Protokol komunikasi ini hanya memiliki dua jalur, yaitu SDA (*Serial Data*) sebagai jalur untuk data dan SCL (*Serial Clock*) sebagai jalur untuk *clock.* Peralatan-peralatan yang menggunakan protokol komunikasi I2C mempunyai sifat *serial synchronuous half duplex bidirectional* yang berarti data dalam protokol komunikasi tersebut dikirim secara serial melalui jalur SDA, menggunakan jalur bergantian dalam transmisi data, dan dikirim dari sebuah perangkat ke perangkat lainnya.

Protokol komunikasi I2C memiliki perangkat yang bertindak sebagai *master* dan perangkat yang bertindak sebagai *Slave*. Perangkat *master* bertindak sebagai sumber *clock* melalui jarul SCL.

Untuk melakukan komunikasi pada protokol I2C, dimulai dengan mengirimkan *start sequence* dan diakhiri dengan *stop sequence*. *start sequence* merupakan pertanda dimulainya proses transmisi data, sedangkan *stop sequence* merupakan pertanda akhir dari proses transmisi data. Berikut ini merupakan bentuk sinyal *start* dan *stop sequence*.



Gambar 2.7 Sinyal *start* dan *stop sequence* [17].

2.5. Presisi (Ketelitian)

Presisi atau ketelitian merupakan nilai yang menunjukkan derajat keterulangan dalam suatu analisis atau pengukuran. Nilai ketelitian dari suatu hasil pengukuran dapat dicari dengan menghitung nilai simpangan baku (standar deviasi). Menurut Guruh (2013:36) untuk mencari nilai simpangan baku dapat ditentukan dengan persamaan berikut ini :

$$SD\sqrt{\frac{\sum(xi-\bar{x})^2}{n}} \quad \text{.....(2.4)}$$

Keterangan :

SD = Standar deviasi/simpangan baku.

x_i = Nilai SPO_2 tiap pengukuran.

$\bar{x}$ = Nilai SPO_2 rata-rata.

n = jumlah pengujian.

2.6. Akurasi

Akurasi menunjukan nilai kedekatan hasil pengukuran terhadap nilai yang menjadi acuan/nilai standar. Akurasi dapat diketahui apabila kesalahan (*error*) sudah diketahui. Akurasi dalam alat yang dirancang dalam tugas akhir ini mengacu pada alat standar yang telah terkalibrasi. Menurut Guruh (2013:36-37) Cara mencari nilai kesalahan dan akurasi adalah sebagai berikut:

$$Error(\%) = \frac{selisih\ pembacaan\ alat\ dengan\ pembanding}{pembacaan\ alat\ pembanding} \times 100\% \quad \text{.....(2.5)}$$

$$Akurasi(\%) = 100\% - Error(\%) \quad \text{.....(2.6)}$$

Keterangan :

Selisih pembacaan alat dengan pembanding = |Nilai yang dikeluarkan alat-Nilai yang dikeluarkan pembanding|.

BAB 3
METODE PENELITIAN

3.1. Lokasi Penelitian

lokasi penelitian dan perancangan alat dilakukan di Lab. Teknik Elektro Universitas Muhammadiyah Sumatera Utara, Jl. Kapten Muchtar Basri, Medan, Sumatera Utara.

3.2. Alat dan Bahan

Alat dan bahan yang digunakan dalam perancaangan alat ini adalah sebagai berikut :

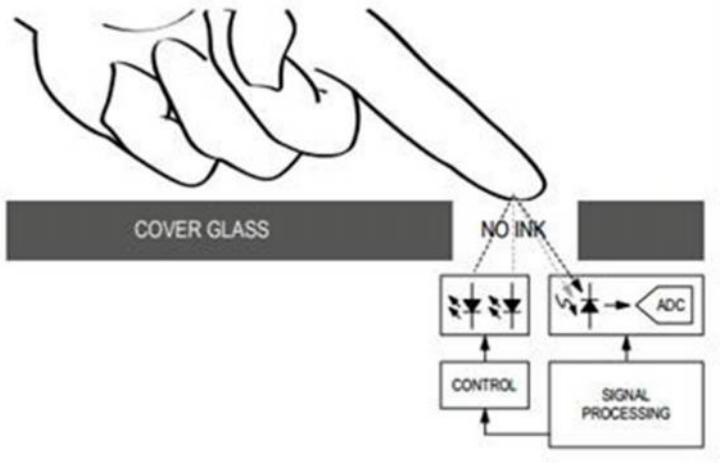
Tabel 3.1. Alat dan bahan yang digunakan.

No.	Alat dan bahan	Jumlah
1	Solder	1 Buah
2	Avo Meter	1 Buah
4	Komputer/Laptop	1 Buah
5	Raspberry Pi 3 Model B	1 Buah
6	Kabel LAN	± 1 Meter
7	Kabel Jumper	Secukupnya
8	Oximetry Sensor IC MAX30100	1 Buah
9	Arduino Uno R3	1 Buah
10	Kabel USB	1 Buah
11	Kabel Coaxial	2 Buah
8	LCD Touchscreen 3,5”	1 Buah

3.1. Perancangan Alat

3.1.1. Oximetry Sensor IC MAX30100

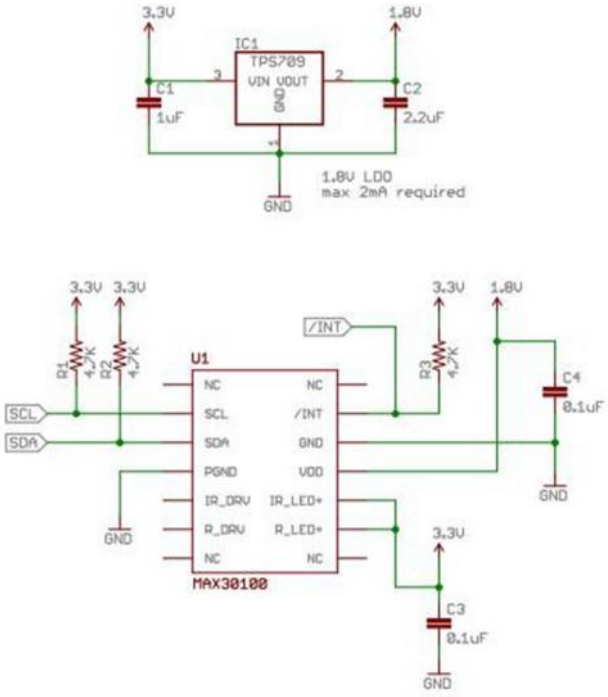
Merupakan satu paket modul sensor yang komponen utamanya adalah IC MAX30100. IC MAX30100 adalah IC yang tersusun dari LED merah dan infra merah serta pengkondisi sinyal yang terintegrasi dalam satu paket IC. Fungsi dari modul sensor ini adalah untuk mengambil data tegangan untuk mencari nilai SPO_2 dan mendeteksi denyutan nadi. Diagram blok dan penempatan modul sensor dapat dilihat pada gambar berikut ini :



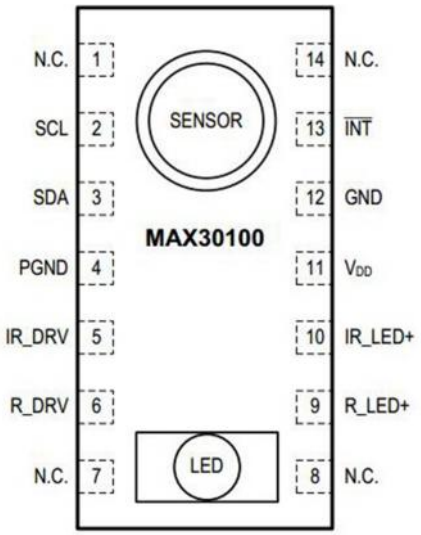
Gambar 3.1 Diagram blok dan penempatan IC MAX30100.

Modul sensor MAX30100 ini ditempatkan pada jaringan tubuh yang tipis seperti ujung jari. Cahaya dari kedua LED yaitu merah dan infra merah diaktifkan secara bergantian dengan frekuensi 100Hz. Cahaya dari kedua LED tersebut dipancarkan ke ujung jari sehingga sebagian terserap oleh ujung jari dan dipantulkan kembali. Cahaya yang terpantul ditangkap oleh fotodioda pada IC MAX30100 sehingga menimbulkan tegangan pada fotodioda dan tegangan tersebut diteruskan ke pengkondisi sinyal yang sudah terintegrasi dalam IC MAX30100. Sinyal yang berasal dari modul sensor MAX30100 diteruskan ke Arduino Uno menggunakan protokol Komunikasi I2C. Berikut ini adalah bentuk fisik dari modul sensor MAX30100 :

Gambar 3.2 Modul sensor MAX30100.



Gambar 3.3 Skema modul sensor oximetry MAX30100.



Gambar 3.4 Konfigurasi pin IC MAX3010

3.1.2. Arduino Uno

Arduino Uno adalah papan mikrokontroler berbasis Atmega328P. Arduino Uno memiliki 14 pin *input/output* (I/O) digital dengan keenam I/O-nya dapat berfungsi sebagai *Pulse Width Modulation* (PWM), 6 analog *input*, koneksi USB, ICSP *header* dan tombol reset. Dapat digunakan langsung dengan menghubungkannya dengan komputer ataupun dengan sumber tegangan 5 volt [18]. Modul *hardware* Arduino diciptakan oleh Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David A. Mellis dan Nicholas Zambetti di Ivrea, Italia pada tahun 2005 [8].

Untuk menjalankan fungsinya, Arduino memerlukan IDE (*Integrated Development Environment*). IDE Arduino adalah perangkat lunak (*software*) yang digunakan untuk menulis program, mengkompilasi dan memasukan program ke Arduino. Bahasa Arduino merupakan turunan dari bahasa *Wiring Platform* dan bahasa *Processing*. Agar Arduino dapat berkomunikasi dengan IDE, diperlukan *bootloader* yang sudah terinstal pada blok memori *flash*.

Arduino memakai standar lisensi terbuka (*open source*) untuk perangkat keras dan IDE-nya, sehingga dapat dimanfaatkan dan dikembangkan tanpa izin tertulis dari

penciptanya. Berikut ini merupakan bentuk fisik dari Arduino Uno :

Gambar 3.5 Arduino Uno R3.

3.1.3. Raspberry Pi

Raspberry Pi adalah sebuah komputer papan tunggal (*Single Board Computer*) yang berukuran kecil hampir menyamai ukuran kartu kredit. Raspberry pi dapat digunakan untuk menjalankan program perkantoran, permainan komputer, memutar audio dan video serta dapat berinteraksi dengan peralatan masukan dan keluaran melalui port-port GPIO (*General Purpose Input Output*). Rapberry Pi dibuat dan dikembangkan oleh sebuah yayasan Nirlaba bernama *Raspberry Pi Fondation* dari Universitas Cambridge, Inggris [5].

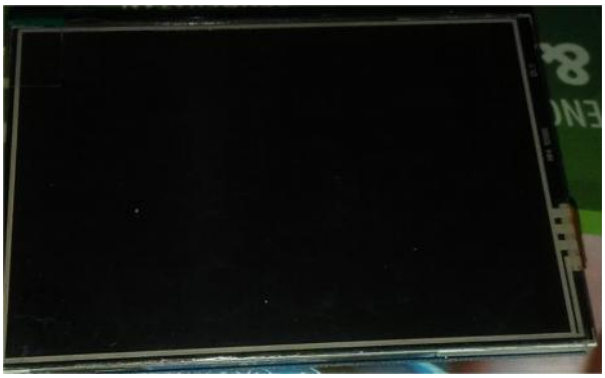
Pada penelitian ini, Raspberry Pi digunakan untuk mengolah data yang diterima dari Arduino dan menampilkannya dalm bentuk grafik dan presentase SPO_2. Berikut ini merupakan bentuk fisik dari Raspberry Pi 3 :

Gambar 3.6 Raspberry Pi 3.

3.1.4. LCD Touchscreen 3,5”

LCD (*Liquid Crystal Display*) adalah jenis media penampil yang menggunakan kristal cari sebagai medianya. LCD sudah banyak digunakan pada peralatan elektronik seperti monitor, tv, *smartphone* dan lain sebagainya. Pada LCD terdapat titik-titik cahaya (piksel) yang terdiri dari satu buah kristal cair sebagai sebuah titik cahaya. Piksel inilah yang merupakan komponen pembentuk grafik yang ditampilkan di LCD. Sumber cahaya dari LCD adalah *backlight* yang diletakkan dibelakang LCD.

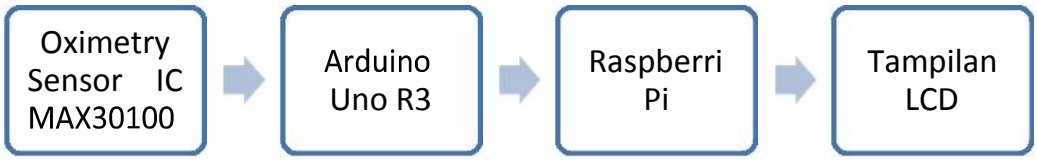
Pada penelitian ini, LCD digunakan untuk menampilkan grafik representase denyutan pada arteri dan presentase SPO_2 yang terbaca oleh alat. Berikut ini merupakan bentuk fisik dari LCD yang digunakan dalam penelitian ini :



Gambar 3.7 LCD *Touchscreen* 3,5”.

3.2. Diagram Blok Perancangan Alat *Pulse Oximetry*

Alat *Pulse Oximetry* yang dibuat dalam tugas akhir ini dirancang dengan diagram blok sebagai berikut :

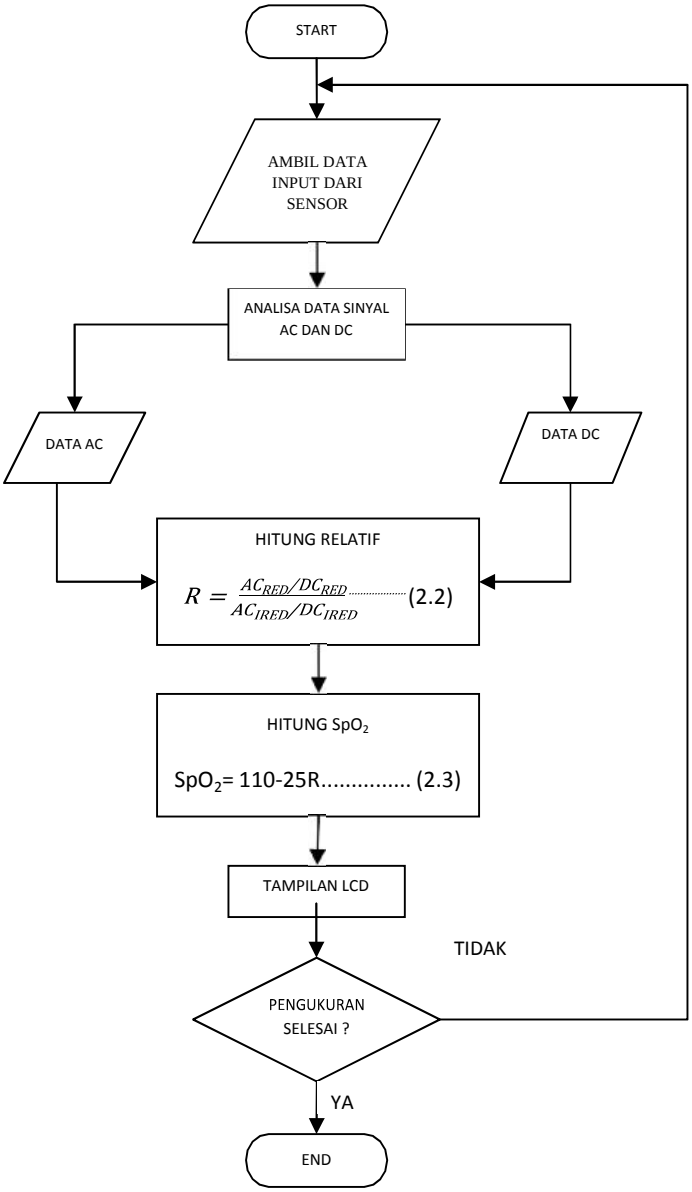


Gambar 3.8. Diagram blok *pulse oximetry.*

Sensor Oximetry IC MAX30100 memancarkan cahaya merah dan inframerah secara bergantian, lalu cahaya yang mengenai ujung jari dipantulkan dan ditangkap oleh fotodetektor yang ada pada IC tersebut. Data tegangan yang dihasilkan oleh IC tersebut dikirimkan ke Arduino menggunakan komunikasi serial I2C (*Inter integrated circuit*). Data yang diterima Arduino, dikirimkan ke Raspberry Pi melalui jalur komunikasi USB (*Universal Serial Bus*). Data yang dikirimkan oleh Arduino berupa nilai tegangan dari hasil penerimaan cahaya merah dan infra merah yang diterima oleh IC MAX30100. Data tegangan pada Raspberry dimasukkan ke rumus oximetry sehingga menghasilkan presentasi oksigen dalam darah. Nilai presentasi tersebut ditampilkan pada LCD beserta grafik penerimaan sinyal dari komponen cahaya merah dan infra merah.

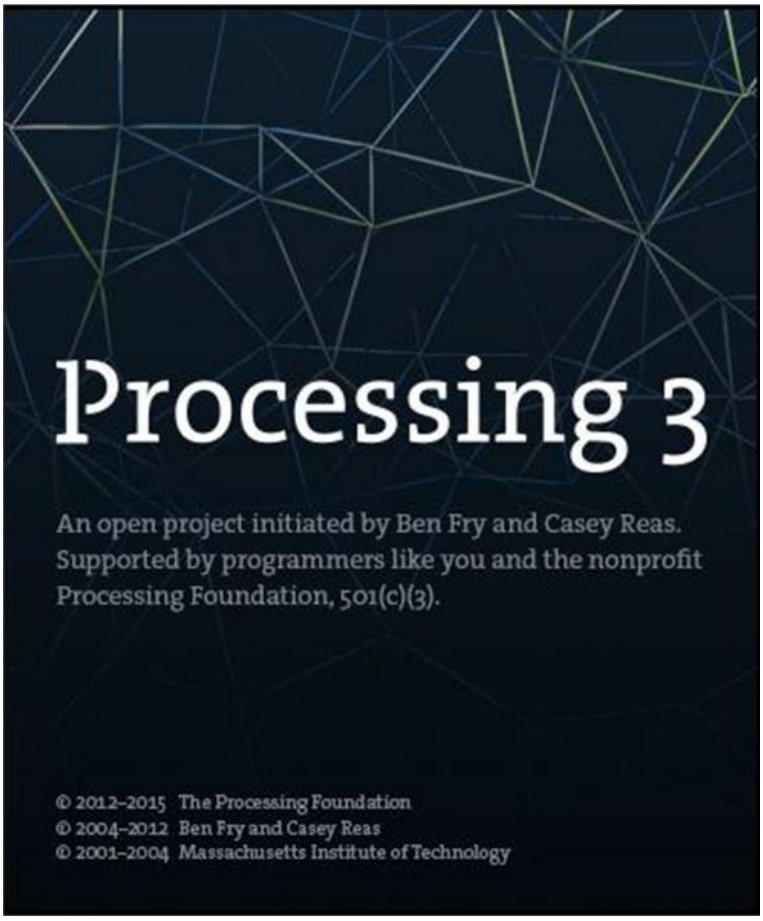
3.3. Perancangan Program

Program yang digunakan dalam perancangan alat ini menggunakan bahasa Arduino dan bahasa Processing. Data dari sensor oximetry MAX30100 diteruskan ke Arduino untuk diolah dan diteruskan ke Raspberry melalui USB (*Universal Serial Bus*). Pada Raspberry, data dari Arduino diolah dengan memasukkannya ke persamaan (2.2) dan (2.3) yang telah dibahas di bab II untuk mendapatkan nilai presentase dari SPO_2. Berikut ini merupakan diagram alir dari program yang digunakan pada alat ini :



Gambar 3.9 Diagram alir program.

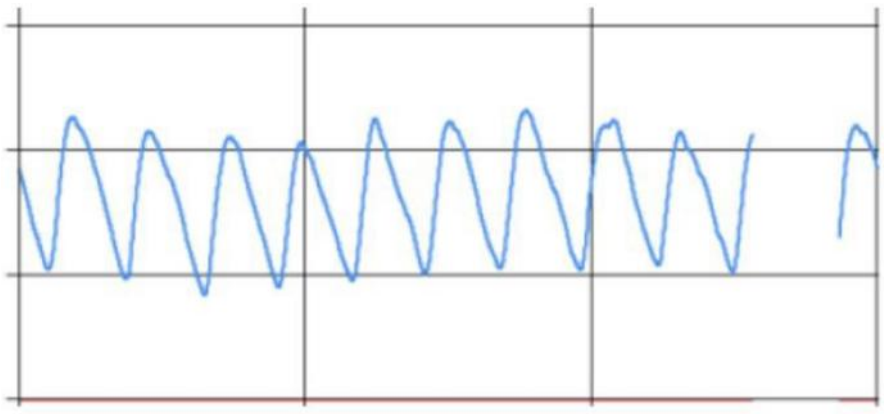
Pada program Arduino, menggunakan kecepatan pengiriman data atau disebut juga *baudrate* sebesar 57600 bps (*bit per second*). Pengaturan *baudrate* ini dapat dilihat pada baris program ”Serial.begin(57600)”, yang berarti pada setiap detik data dikirim sebesar 57600 bit. Kecepatan maksimal pengiriman data melalui USB pada Arduino adalah 115200 bps. Pada tugas akhir ini menggunakan *baudrate* sebesar 57600 bps agar dapat memenuhi kecepatan pengolahan data menggunakan program Processing pada Raspberry Pi.



Gambar 3.10 Logo Processing.

3.4. Hasil Tampilan Pengolahan Sinyal

Alat *Pulse Oxymetry* ini Menghasilkan tampilan berupa grafik dari absorbsi cahaya merah dan inframerah terhadap arteri seperti yang terlihat pada gambar dan menampilkan nilai presentase dari saturasi oksigen dalam darah.



Gambar 3.11 Sinyal denyutan absorbsi pada arteri [6].

3.5. Standar dan Pengujian *Pulse Oximetry*

Pulse Oximetry merupakan alat kesehatan yang mudah digunakan. Namun, akurasi dari *Pulse Oximetry* harus tetap diperhatikan. Kesalahan pengukuran pada *Pulse Oximetry* dapat menyebabkan kesalahan diagnosa pada pasien. Oleh karena itu, keakurasian dari *Pulse Oximetry* harus tetap diperhatikan.

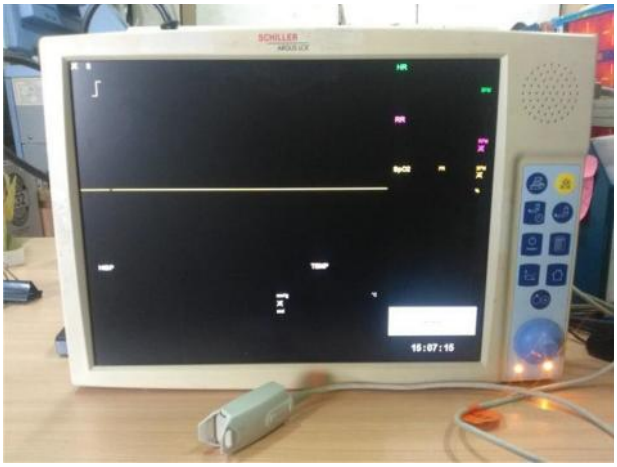
Untuk menjaga akurasi dari Pulse Oximeter, pengujian berdasarkan suatu standar yang spesifik perlu dilakukan. Pada standar umum untuk peralatan kesehatan IEC 60601-1-1 tidak disebutkan akurasi yang harus dicapai oleh suatu Pulse Oximeter. Akan tetapi, ISO 80601-2-61 tentang persyaratan khusus untuk keselamatan dasar dan kinerja utama peralatan Pulse Oximeter klausul 201.12.1.101 menyebutkan bahwa akurasi SpO2 dari Pulse Oximeter yang ditunjukkan dengan perbedaan akar dari nilai kuadrat rata-rata harus ≤ 4% pada saat range SaO2 adalah 70 % sampai 100% [7].

Alat *Pulse Oximetry* yang dibuat pada skripsi ini diuji dengan cara dibandingkan dengan alat *Pulse Oximetry* yang telah terstandar dan terkalibrasi sehingga meminimalisasi kesalahan pengukuran yang terjadi.

3.6. Tahap Pengambilan Data

Pengambilan data dilakukan untuk mengetahui kinerja alat dan seberapa besar kesalahan pengukurannya. Data yang diambil adalah presentase SPO_2 yang ditampilkan langsung oleh alat dan tegangan dari masing-masing hasil penyerapan cahaya merah dan infra merah. Data tegangan didapatkan dari grafik hasil tampilan representasi denyutan nadi. Data tegangan tersebut di hitung dengan dimasukan ke persamaan (2.2) dan (2.3) sehingga didapat nilai presentase SPO_2 secara matematis. Presentase SPO_2 yang ditampilkan langsung oleh LCD dibandingkan dengan alat ukur standar yang telah terkalibrasi sebagai pembanding. Alat yang digunakan sebagai pembanding adalah *Bed Side Monitor* merek Schiller tipe Argus

LCX. Berikut ini merupakan gambar alat ukur standar yang digunakan sebagai pembanding dalam penelitian ini :

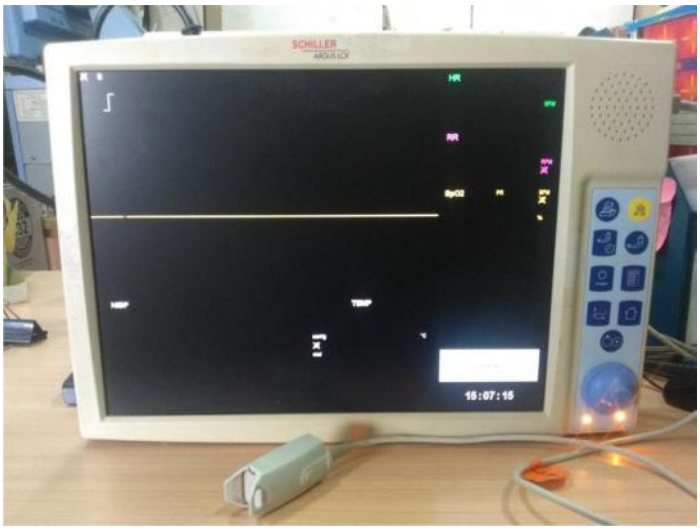


Gambar 3.12 *Bed Side Monitor* merek Schiller tipe Argus LCX.

BAB 4
DATA DAN ANALISA

4.1. Metode Pengambilan Data

Metode pengambilan data yang dilakukan dalam penelitian ini adalah dengan cara mengambil nilai presentase SPO_2, Nilai tegangan dari hasil tangkapan fotosensor terhadap LED merah dan infra merah, dan grafik hasil tegangan yang mengalami kenaikan dan penurunan yang disebabkan oleh denyutan nadi. Data-data tersebut dibandingkan dengan alat ukur SPO_2 yang telah terkalibrasi. Data-data dari alat hasil perancangan dan alat ukur standar dibandingkan dan dicari selisihnya agar didapat presentase kesalahannya (error). Alat ukur SPO_2 yang digunakan sebagai pembanding dalam penelitian ini adalah *Bed Side Monitor* merek Schiller tipe Argus LCX.



Gambar 4.1 *Bed Side Monitor* merek Schiller.

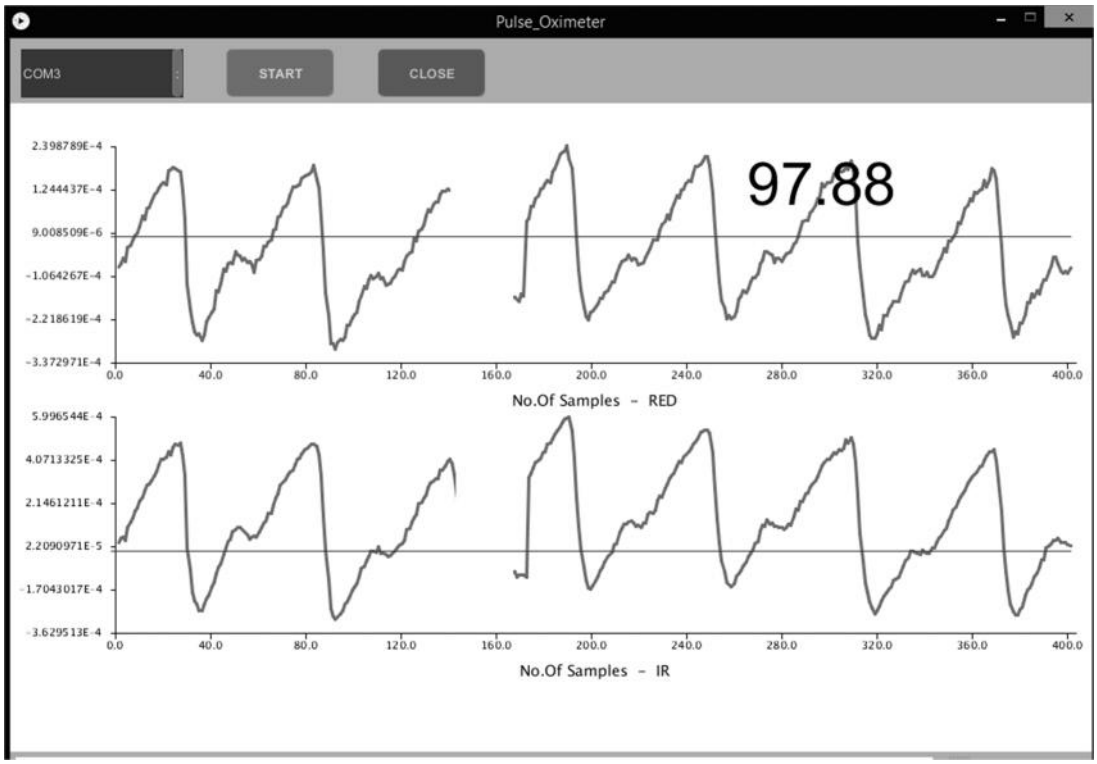
4.2. Hasil Pengambilan Data

Pengambilan data dilakukan dengan mengambil presentase SPO_2 pada alat, lalu dibandingkan dengan alat standar sebagai pembanding untuk dicari presentase kesalahannya serta pengambilan data tegangan dari hasil penyerapan cahaya merah dan infra merah. Berikut ini merupakan tabel hasil pengambilan data.

Tabel 4.1. Tabel hasil pengambilan data presentase SPO_2.

No.	Nama	Pengukuran Ke-	SPO_2 Alat (%)	SPO_2 Pembanding (%)
1	Rivaldy	1	101,27	100
		2	98,4	100
		3	95,2	100
		4	101,6	99
		5	97,83	100
2	Bambang	1	93,86	98
		2	94,5	97
		3	95,8	96
		4	95,52	95
		5	96,34	96
3	Adit	1	96,79	100
		2	99,05	99
		3	95,59	98
		4	96,49	97
		5	94,98	97
4	Jaya	1	96,87	100
		2	98,09	100
		3	98,68	100
		4	98,92	100
		5	98,47	100
5	Denis	1	95,09	100
		2	96,37	98
		3	96,96	97
		4	96,07	96
		5	98,45	96

Data yang diambil dari alat yang dirakit berupa presentase SPO_2 pada saat pengukuran dan grafik representasi denyutan nadi dari masing-masing komponen cahaya yang berbeda, yaitu merah dan infra merah. Berikut ini merupakan tampilan data yang diambil :

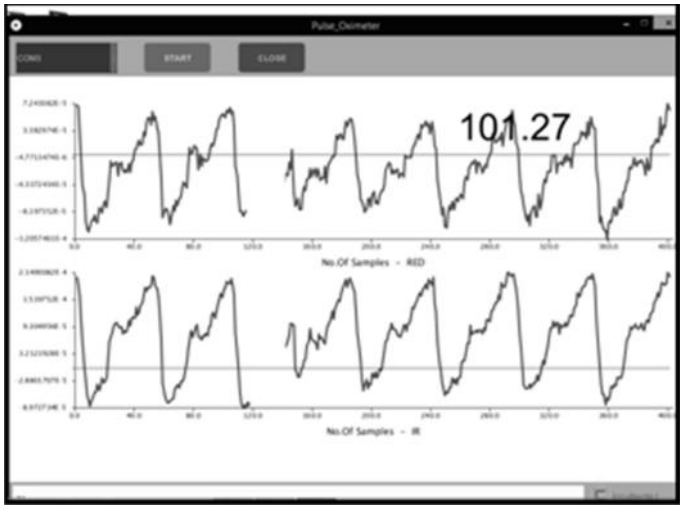


Gambar 4.2 Hasil pengambilan data.

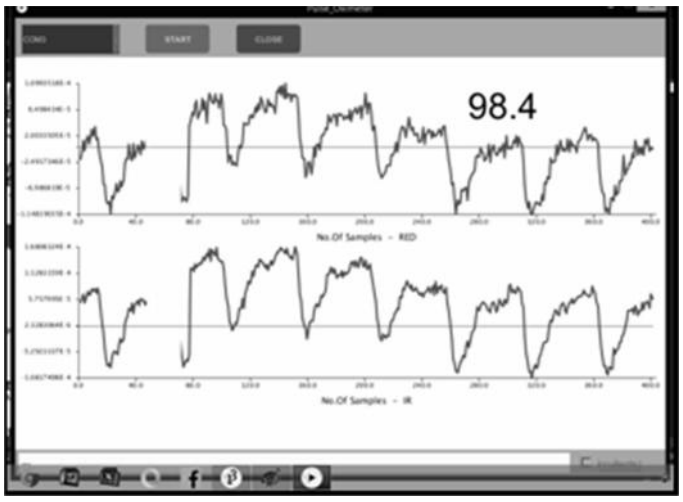
4.3. Analisis dan Perbandingan Data

Data yang telah diambil kemudian dianalisa dengan cara dibandingkan dengan alat pembanding. Berikut ini merupakan grafik dan perhitungan data alat dan pembanding :

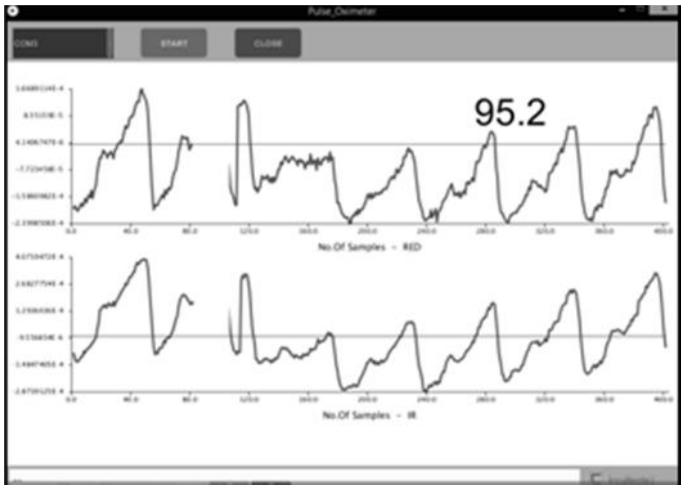
1. Rivaldy



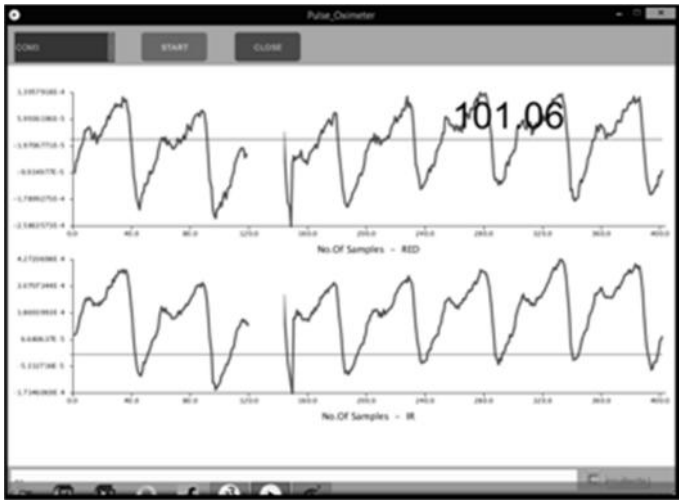
Gambar 4.3 Hasil pembacaan pertama percobaan pertama.



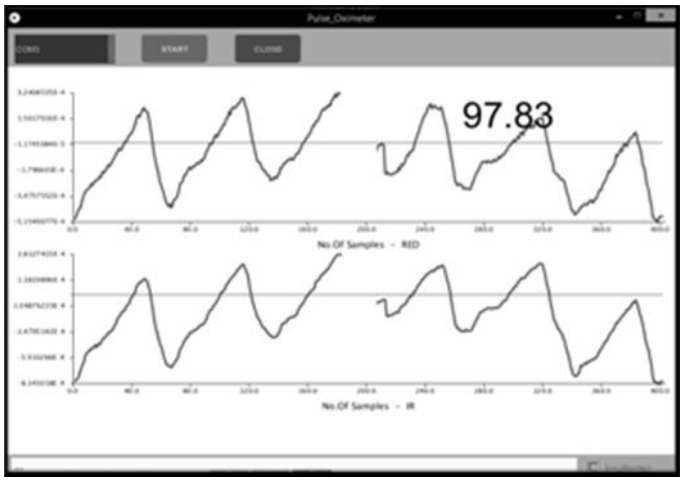
Gambar 4.4 Hasil pembacaan kedua percobaan pertama.



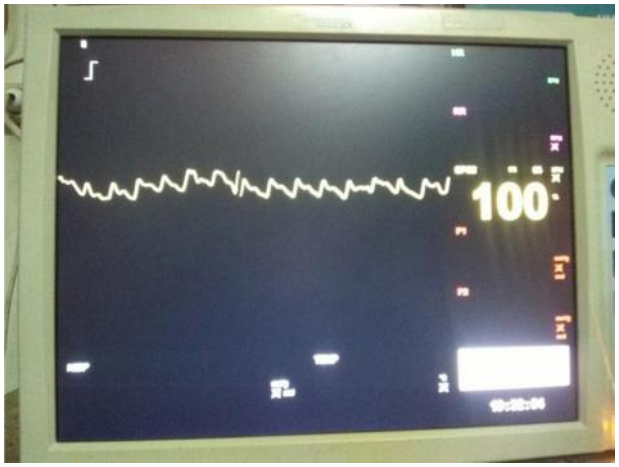
Gambar 4.5 Hasil pembacaan ketiga percobaan pertama.



Gambar 4.6 Hasil pembacaan keempat percobaan pertama.



Gambar 4.7 Hasil pembacaan kelima percobaan pertama.



Gambar 4.8 Hasil pembacaan alat pembanding percobaan pertama.

Dari hasil pengambilan data pertama, bisa tuliskan kembali dalam bentuk tabel berikut ini :

Tabel 4.2 Tabel data percobaan pertama.

Percobaan Ke-	SPO_2 Alat (%)	SPO_2 Pembanding (%)
1	101,27	100
2	98,4	100
3	95,2	100
4	101,6	99
5	97,83	100
Rata-rata	98,86	99,8

Nilai kepresisian dari alat, bisa dicari dengan cara mencari standar deviasi dari data pengukuran. Standar deviasi dari pengukuran pada percobaan pertama, bisa ditentukan dengan cara :

$$SD = \frac{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}}{n}$$

$$SD = \frac{\sqrt{5,81 + 0,21 + 13,4 + 7,51 + 1,06}}{5}$$

$$SD = \frac{\sqrt{27,99}}{5} = \sqrt{5,6} = 2,37$$

Sehingga hasil dari pengukuran bisa dituliskan dengan mencantumkan nilai kepresisiannya menjadi 98,86 ± 2,37 %.

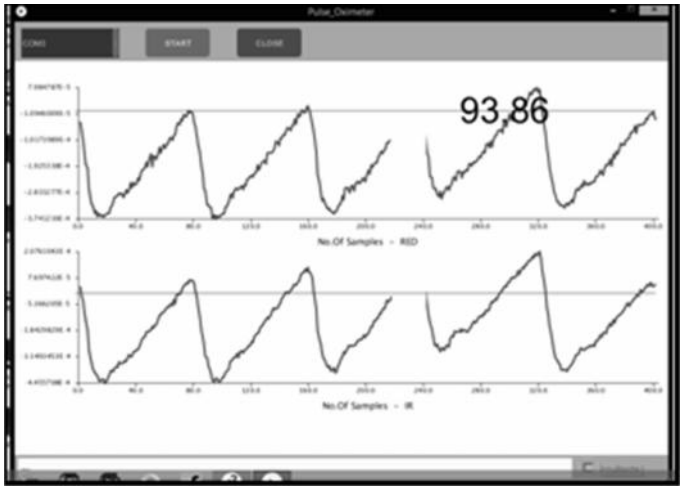
Dari hasil pembandingan antar nilai yang dihasilkan alat dengan nilai yang dihasilkan alat pembanding didapatkan presentase kesalahan sebesar :

$$Error = \frac{Selisih\ pembacaan\ alat\ dengan\ pembanding}{Pembacaan\ alat\ pembanding} x100\%$$

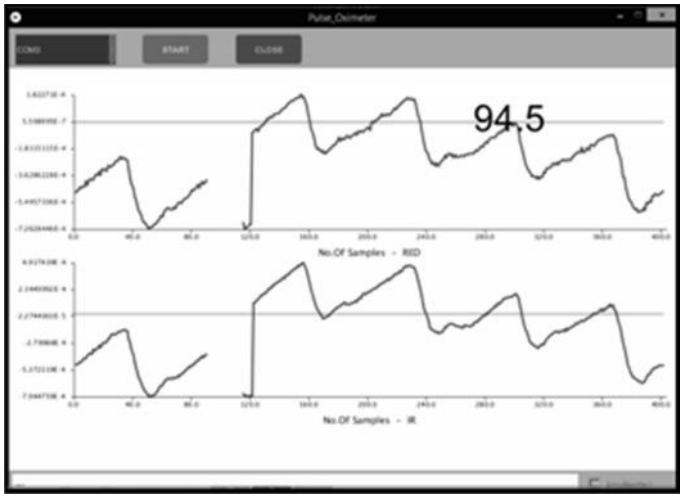
$$Error = \frac{0,94\%}{99,8\%} x100\% = 0,94\%$$

Dari data yang telah diambil, dapat dilihat bahwa hasil pembacaan rata-rata SPO_2 alat sebesar 98,86 ± 2,37 % dengan presentase kesalahan terhadap hasil pembacaan alat pembanding sebesar 0,94%.

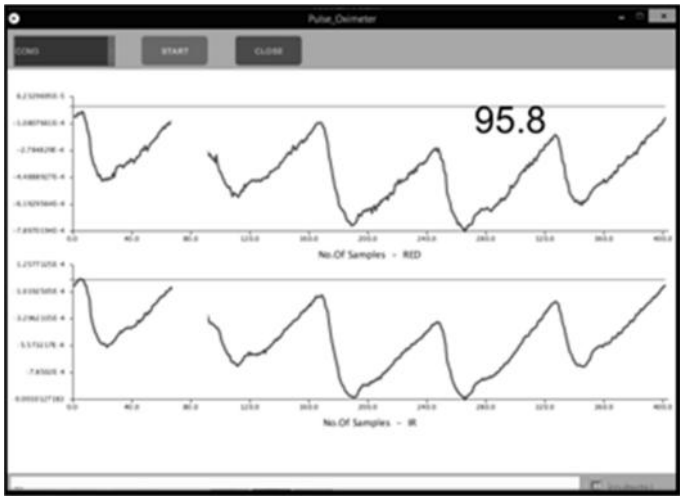
2. Bambang



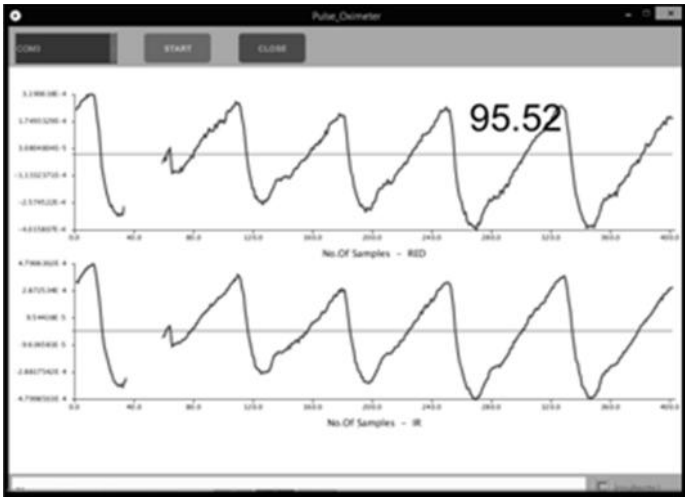
Gambar 4.9 Hasil pembacaan pertama percobaan kedua.



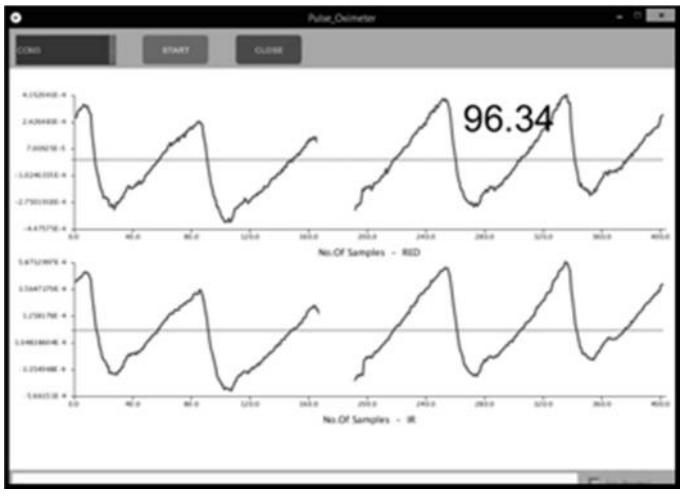
Gambar 4.10 Hasil pembacaan kedua percobaan kedua.



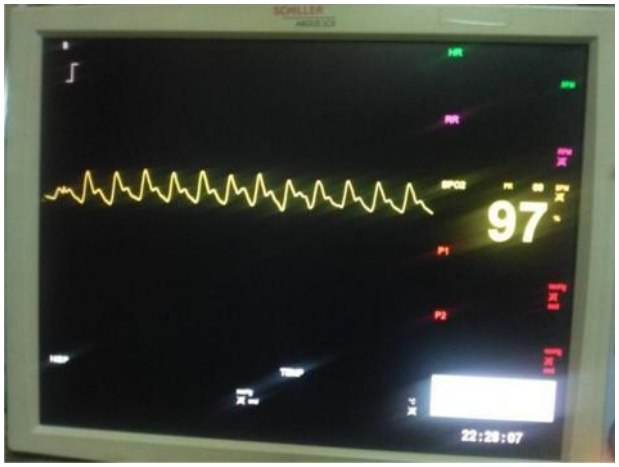
Gambar 4.11 Hasil pembacaan ketiga percobaan kedua.



Gambar 4.12 Hasil pembacaan keempat percobaan kedua.



Gambar 4.13 Hasil pembacaan kelima percobaan kedua.



Gambar 4.14 Hasil pembacaan alat pembanding percobaan kedua.

Dari hasil pengambilan data kedua, bisa tuliskan kembali dalam bentuk tabel berikut ini :

Tabel 4.3 Tabel data percobaan kedua.

Percobaan Ke-	SPO_2 Alat (%)	SPO_2 Pembanding (%)
1	93,86	98
2	94,5	97
3	95,8	96
4	95,52	95
5	96,34	96
Rata-rata	95,2	96,4

Nilai kepresisian dari alat, bisa dicari dengan cara mencari standar deviasi dari data pengukuran. Standar deviasi dari pengukuran pada percobaan pertama, bisa ditentukan dengan cara :

$$SD = \frac{\overline{\sum_{i=1}^{n} (x_i - \bar{x})^2}}{n}$$

$$SD = \frac{\overline{1{,}8 + 0{,}49 + 0{,}36 + 0{,}1 + 1{,}3}}{5}$$

$$SD = \sqrt{\frac{4,05}{5}} = \sqrt{0,81} = 0,9$$

Sehingga hasil dari pengukuran bisa dituliskan dengan mencantumkan nilai kepresisiannya menjadi 95,2 ± 0,9 %.

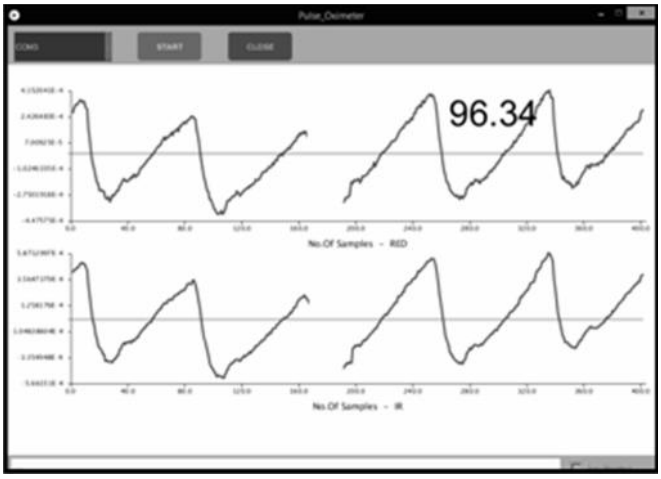
Dari hasil pembandingan antar nilai yang dihasilkan alat dengan nilai yang dihasilkan alat pembanding didapatkan presentase kesalahan sebesar :

$$Error = \frac{Selisih\ pembacaan\ alat\ dengan\ pembanding}{Pembacaan\ alat\ pembanding} x100\%$$

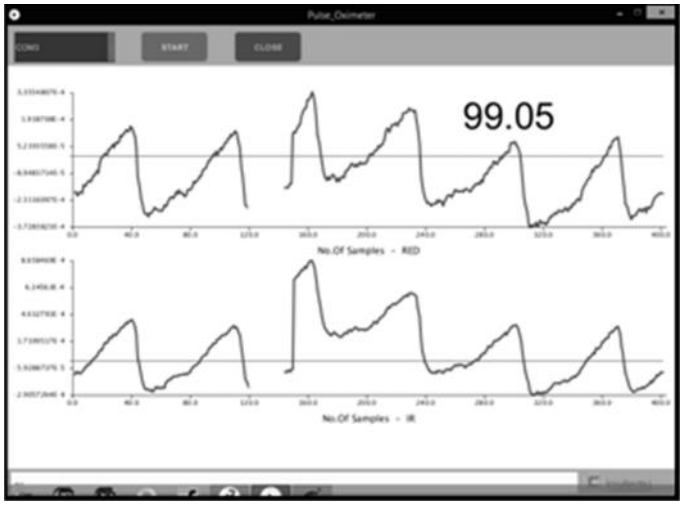
$$Error = \frac{1,2\%}{96,4\%} x100\% = 1,24\%$$

Dari data yang telah diambil, dapat dilihat bahwa hasil pembacaan rata-rata SPO_2 alat sebesar 95,2 ± 0,9 % dengan presentase kesalahan terhadap hasil pembacaan alat pembanding sebesar 1,24%.

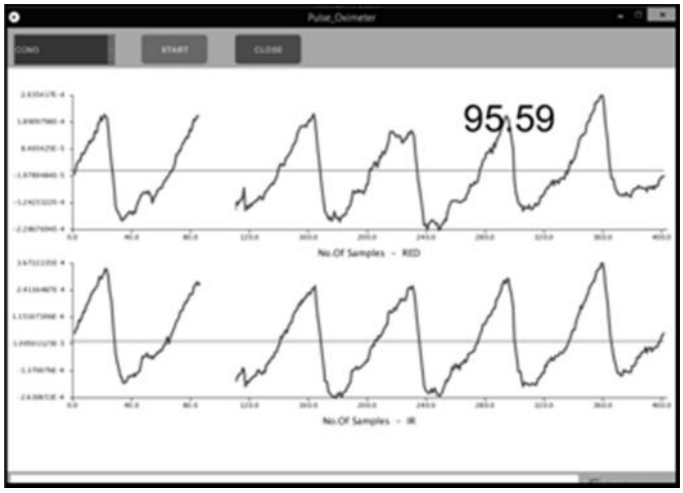
3. Adit



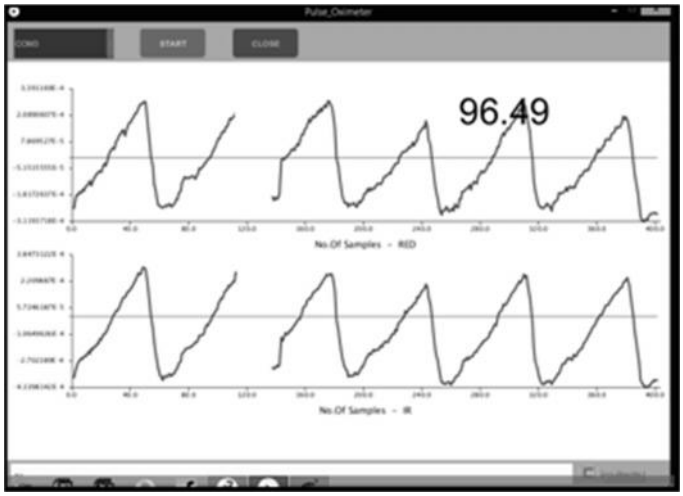
Gambar 4.15 Hasil pembacaan pertama percobaan ketiga.



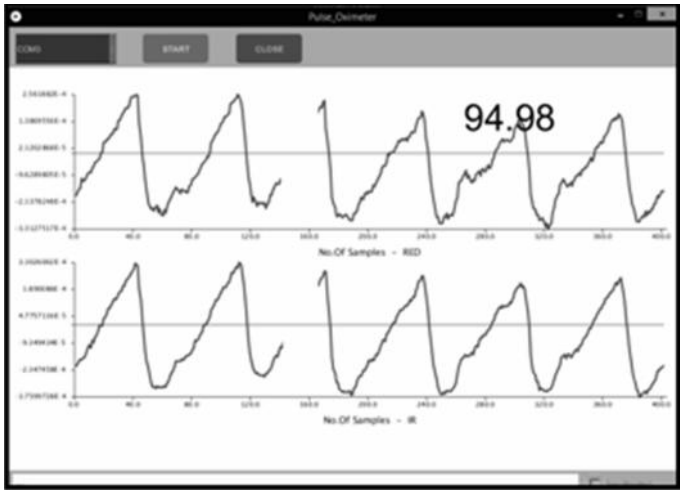
Gambar 4.16 Hasil pembacaan kedua percobaan ketiga.



Gambar 4.17 Hasil pembacaan ketiga percobaan ketiga.



Gambar 4.18 Hasil pembacaan keempat percobaan ketiga.



Gambar 4. 19 Hasil pembacaan kelima percobaan ketiga.

Gambar 4.20 Hasil pembacaan alat pembanding percobaan ketiga.

Dari hasil pengambilan data ketiga, bisa tuliskan kembali dalam bentuk tabel berikut ini :

Tabel 4.4 Tabel data percobaan ketiga.

Percobaan Ke-	SPO_2 Alat (%)	SPO_2 Pembanding (%)
1	96,79	100
2	99,05	99
3	95,59	98
4	96,49	97
5	94,98	97

Rata-rata	96,58	98,2

Nilai kepresisian dari alat, bisa dicari dengan cara mencari standar deviasi dari data pengukuran. Standar deviasi dari pengukuran pada percobaan pertama, bisa ditentukan dengan cara :

$$SD = \frac{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}}{n}$$

$$SD = \frac{\sqrt{0,04 + 6,1 + 0,98 + 0,01 + 2,56}}{5}$$

$$SD = \frac{\sqrt{9,69}}{5} = \sqrt{1,94} = 1,39$$

Sehingga hasil dari pengukuran bisa dituliskan dengan mencantumkan nilai kepresisiannya menjadi 96,58 ± 1,39 %.

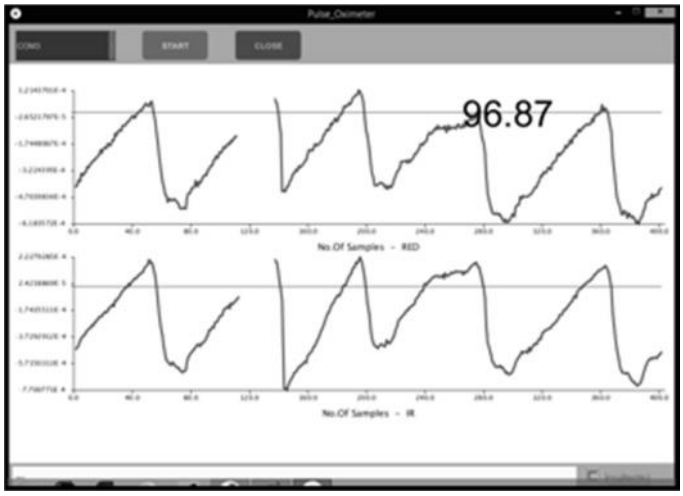
Dari hasil pembandingan antar nilai yang dihasilkan alat dengan nilai yang dihasilkan alat pembanding didapatkan presentase kesalahan sebesar :

$$Error = \frac{Selisih\ pembacaan\ alat\ dengan\ pembanding}{Pembacaan\ alat\ pembanding} x100\%$$

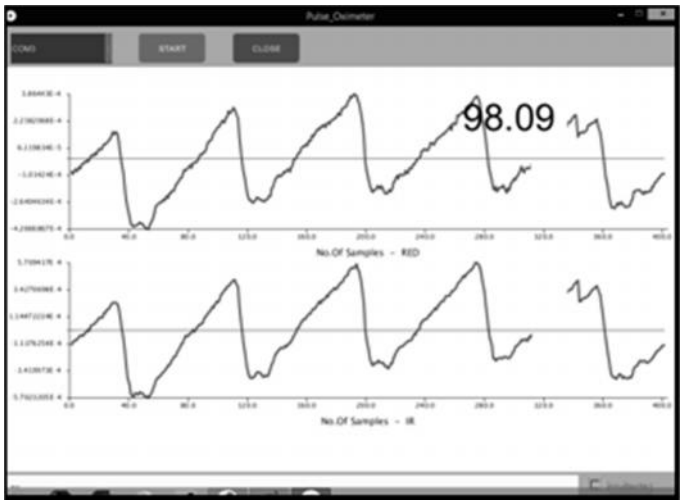
$$Error = \frac{1,62\%}{98,2\%} x100\% = 1,65\%$$

Dari data yang telah diambil, dapat dilihat bahwa hasil pembacaan rata-rata SPO_2 alat sebesar 96,58 ± 1,39 % dengan presentase kesalahan terhadap hasil pembacaan alat pembanding sebesar 1,65%.

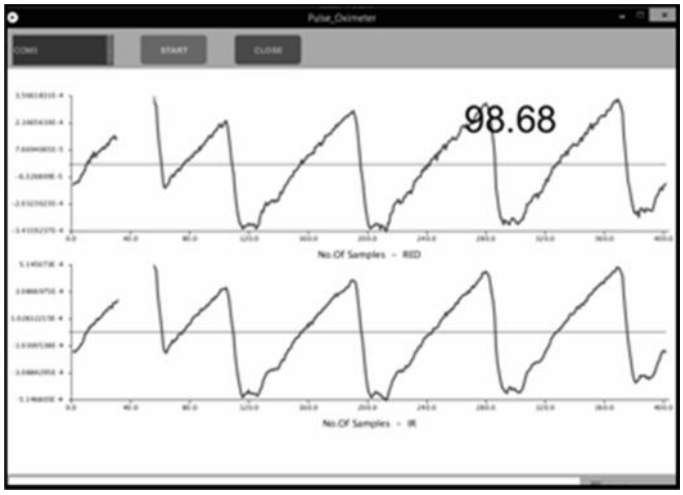
4. Jaya



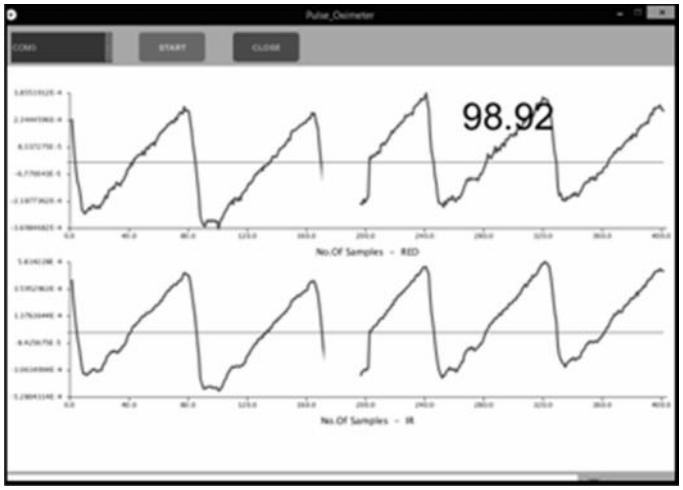
Gambar 4.21 Hasil pembacaan pertama percobaan keempat.



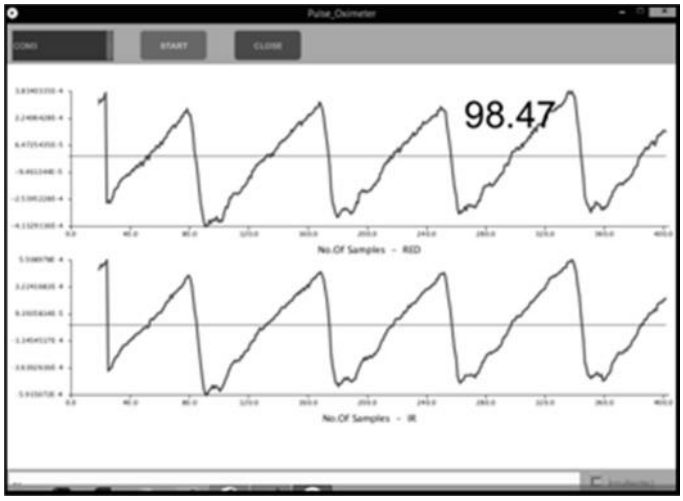
Gambar 4.22 Hasil pembacaan kedua percobaan keempat.



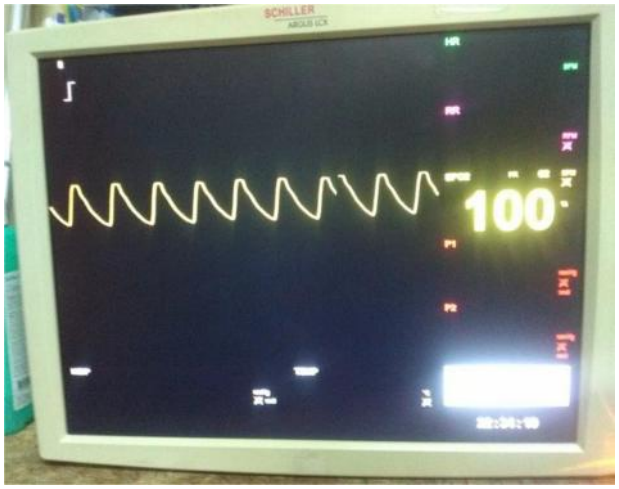
Gambar 4.23 Hasil pembacaan ketiga percobaan keempat.



Gambar 4.24 Hasil pembacaan keempat percobaan keempat.



Gambar 4.25 Hasil pembacaan kelima percobaan keempat.



Gambar 4.26 Hasil pembacaan alat percobaan keempat.

Dari hasil pengambilan data keempat, bisa tuliskan kembali dalam bentuk tabel berikut ini :

Tabel 4.5 Tabel data percobaan keempat.

Percobaan Ke-	SPO_2 Alat (%)	SPO_2 Pembanding (%)
1	96,87	100
2	98,09	100
3	98,68	100
4	98,92	100
5	98,47	100
Rata-rata	98,21	100

Nilai kepresisian dari alat, bisa dicari dengan cara mencari standar deviasi dari data pengukuran. Standar deviasi dari pengukuran pada percobaan pertama, bisa ditentukan dengan cara :

$$SD = \frac{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}}{n}$$

$$SD = \frac{\sqrt{1,8 + 0,01 + 0,22 + 0,5 + 0,07}}{5}$$

$$SD = \frac{\sqrt{2,6}}{5} = \sqrt{0,52} = 0,72$$

Sehingga hasil dari pengukuran bisa dituliskan dengan mencantumkan nilai kepresisiannya menjadi 98,21 ± 0,72 %.

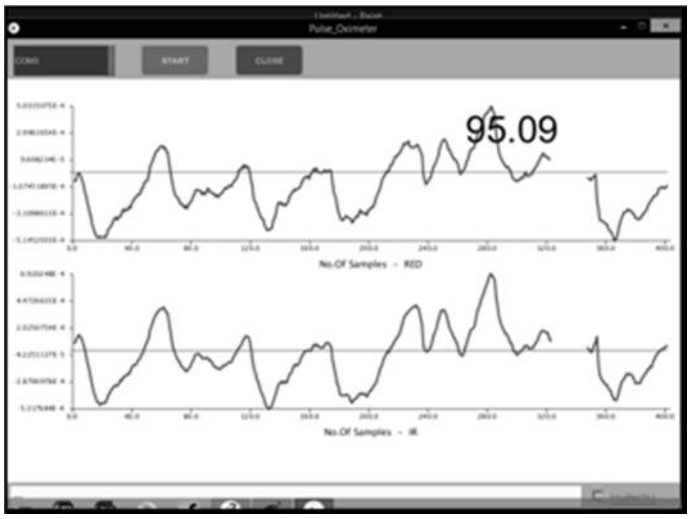
Dari hasil pembandingan antar nilai yang dihasilkan alat dengan nilai yang dihasilkan alat pembanding didapatkan presentase kesalahan sebesar :

$$Error = \frac{Selisih\ pembacaan\ alat\ dengan\ pembanding}{Pembacaan\ alat\ pembanding} x100\%$$

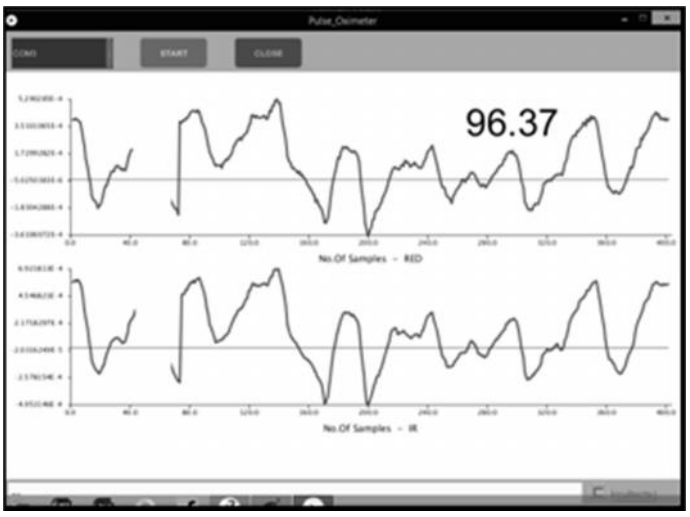
$$Error = \frac{1,79\%}{100\%} x100\% = 1,79\%$$

Dari data yang telah diambil, dapat dilihat bahwa hasil pembacaan rata-rata SPO_2 alat sebesar 98,21 ± 0,72 % dengan presentase kesalahan terhadap hasil pembacaan alat pembanding sebesar 1,79%.

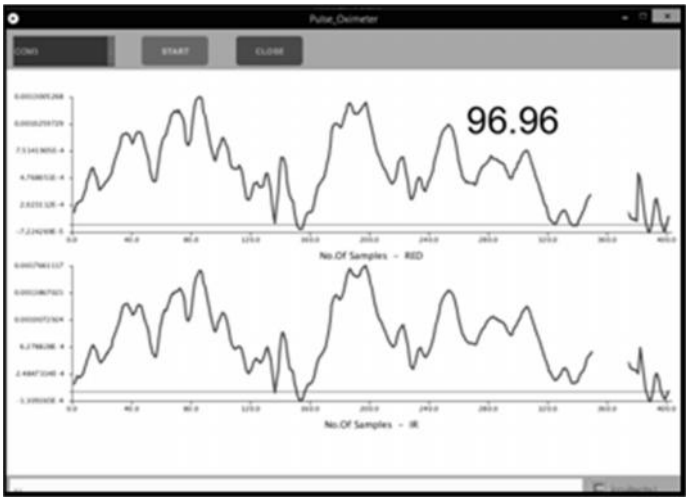
5. Denis



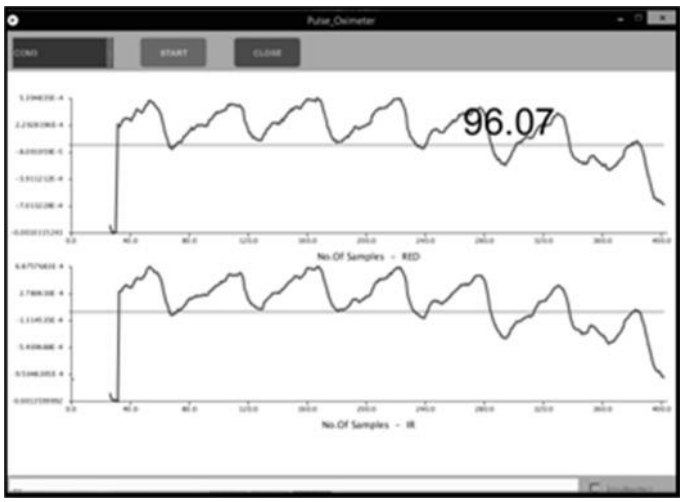
Gambar 4.27 Hasil pembacaan pertama percobaan kelima.



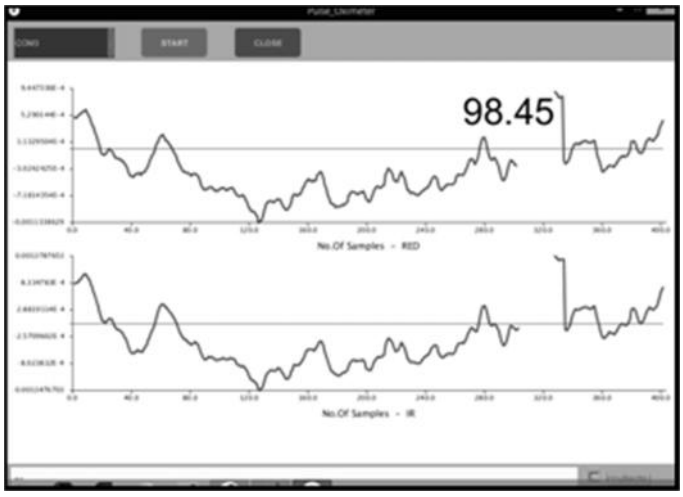
Gambar 4.28 Hasil pembacaan kedua percobaan kelima.



Gambar 4.29 Hasil pembacaan ketiga percobaan kelima.



Gambar 4.30 Hasil pembacaan keempat percobaan kelima.



Gambar 4.31 Hasil pembacaan kelima percobaan kelima.

Gambar 4.32 Hasil pembacaan alat pembanding percobaan kelima.

Dari hasil pengambilan data kelima, bisa tuliskan kembali dalam bentuk tabel berikut ini :

Tabel 4.6 Tabel data percobaan kelima.

Percobaan Ke-	SPO_2 Alat (%)	SPO_2 Pembanding (%)
1	95,09	98
2	96,37	97
3	96,96	96
4	96,07	96
5	98,45	100
Rata-rata	96,59	97,4

Nilai kepresisian dari alat, bisa dicari dengan cara mencari standar deviasi dari data pengukuran. Standar deviasi dari pengukuran pada percobaan pertama, bisa ditentukan dengan cara :

$$SD = \frac{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}}{n}$$

$$SD = \sqrt{\frac{2{,}25 + 0{,}05 + 0{,}14 + 0{,}27 + 3{,}46}{5}}$$

$$SD = \sqrt{\frac{6{,}17}{5}} = \sqrt{1{,}23} = 1{,}11$$

Sehingga hasil dari pengukuran bisa dituliskan dengan mencantumkan nilai kepresisiannya menjadi 96,59 ± 1,11 %.

Dari hasil pembandingan antar nilai yang dihasilkan alat dengan nilai yang dihasilkan alat pembanding didapatkan presentase kesalahan sebesar :

$$Error = \frac{Selisih\ pembacaan\ alat\ dengan\ pembanding}{Pembacaan\ alat\ pembanding} x100\%$$

$$Error = \frac{0{,}81\%}{97{,}4\%} x100\% = 0{,}83\%$$

Dari data yang telah diambil, dapat dilihat bahwa hasil pembacaan rata-rata SPO_2 alat sebesar 96,59 ± 1,11 % dengan presentase kesalahan terhadap hasil pembacaan alat pembanding sebesar 0,83%.

Dari data-data hasil pengukuran dan pembandingan diatas, bisa dituliskan kembali dalam sebuah tabel berikut ini :

Tabel 4.7 Tabel data hasil pembandingan presentase SPO_2.

No.	Nama	Rata-rata SPO_2 Alat (%)	Rata-rata SPO_2 Pembanding (%)	Error (%)
1	Rivaldy	98,86	99,8	0,94
2	Bambang	95,2	96,4	1,24
3	Adit	96,58	98,2	1,65

4	Jaya	98,21	100	1,79
5	Denis	96,59	97,4	0,83
Error Rata-rata (%)				1,29

Pada tabel diatas bisa diketahui bahwa hasil pengukuran presentase SPO_2 memiliki rata-rat kesalahan (error) terhadap alat pembanding sebesar 1,29%, sehingga dapat ditentukan presentase akurasinya dengan cara berikut ini :

$$Akurasi = 100\% - error$$
$$Akurasi = 100\% - 1{,}29\% = 98{,}71\%$$

Dari perhitungan diatas dapat kita tentukan bahwa akurasi dari alat tersebut sebesar 98,71%.

a. Perbandingan Terhadap Alat Sejenis

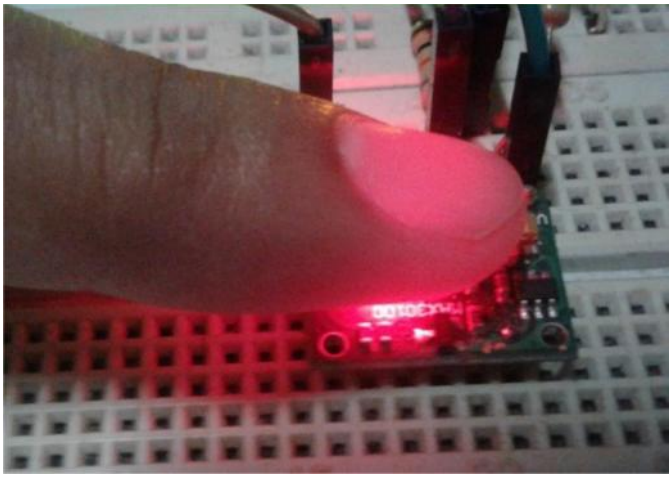
Alat *Pulse Oximetry* yang dibuat dalam penelitian ini memiliki beberapa perbedaan dengan alat sejenis yang digunakan di fasilitas-fasilitas kesehatan dan dengan yang dijual di pasaran, berupa kelebihan dan kekurangan. Berikut ini merupakan kelebihan alat yang dirancang dalam penelitian ini :

i. Lebih mudah dirancang, karena menggunakan komponen-komponen yang biasa digunakan dalam proses pembelajaran, seperti : Arduino dan Raspberry Pi.
ii. Lebih murah. Perbandingan harga alat dengan alat sejenis dipasaran akan dibahas pada subbab berikutnya.
iii. Lebih mudah digunakan untuk pengambilan data dan pengukuran sesaat (tidak berkelanjutan).

Selain memiliki kelebihan, alat yang dirancang dalam penelitian ini juga memiliki beberapa kekurangan, yaitu sebagai berikut :

1. Tidak *mobile*, karena belum menggunakan *batteray*.

2. Sulit untuk melakukan pengukuran yang berkelanjutan, karena sensor tidak terikat di ujung jari, tetapi hanya disentuhkan ke ujung jari. Sedangkan pada alat pembanding, sensor menjepit ujung jari, sehingga terikat kuat di ujung jari. Penempatan sensor pada alat yang dirakit dan alat pembanding dapat dilihat pada gambar-gambar berikut ini :



Gambar 4.33 Penempatan sensor pada alat yang dirakit.

Gambar 4.34 Penempatan sensor pada alat pembanding.

b. Perbandingan Harga

Tujuan dari penelitian ini adalah membuat alat *Pulse Oximetry* yang lebih murah dari yang digunakan di fasilitas-fasilitas kesehatan dan yang dijual di pasaran. Berikut ini merupakan rincian biaya dari perancangan alat dalam penelitian ini :

Tabel 4.8 Tabel rincian biaya perancangan alat

No.	Jenis Rincian Biaya	Jumlah	Harga Satuan (Rp)	Harga Total (Rp)
1	Raspberry Pi 3 Model B	1 pcs	500.000	500.000
2	Arduino Uno	1 pcs	150.000	150.000
3	Oximetry Sensor MAX30100	1 pcs	135.000	135.000
4	Kabel Coaxial	1 unit	35.000	35.000
5	LCD Touchscreen 3,5”	1 pcs	210.000	210.000
6	Kabel	± 0,5 m	5.000	5.000
7	Biaya modifikasi program	1 x	65.000	65.000
Total (Rp)				1.100.000

Dari tabel 4.8 diatas, total biaya yang dikeluarkan dalam perancangan alat yang digunakan dalam tugas akhir ini adalah sebesar Rp. 1.100.000,00. Perbandingan harga alat *Pulse oximetry* yang digunakan dan dijual di pasaran dapat dilihat pada tabel 4.9 berikut ini :

Tabel 4.9 Tabel perbandingan harga.

No.	Merek	Tipe	Harga (Rp)	Harga Alat yang dirancang

				(Rp)
1	GE Healthcare	Tuffsat	13.736.250	1.163.000
2	GE Healthcare	Trusat	40.445.625	
3	Nonin Medical	GO2	1.917.000	
4	Bionet	Oxy9wave	5.995.000	

Dari tabel 4.9 diatas, dapat diketahui bahwa harga alat yang dirancang dalam tugas akhir ini lebih murah daripada alat serupa yang dijual di pasaran, sehingga tujuan dari penelitian ini dapat terlaksana, yaitu merancang alat *Pulse Oximetry* yang lebih murah daripada yang dijual di pasaran.

BAB 5

PENUTUP

5.1. Kesimpulan

Dari hasil pengujian alat, dapat disimpulkan sebagai berikut :

1. Berdasarkan tabel 4.7, didapatkan kesalahan (*error*) rata-rata dari alat Pulse Oximetry Digital Berbasis Raspberry Pi 3 terhadap alat pembanding diperoleh sebesar *1,29*%, sehingga dapat diketahui akurasi alat sebesar *98,71*%.
2. Berdasarkan tabel 4.9, biaya perancangan alat *Pulse Oximetry* berbasis Raspberry Pi 3 pada tugas akhir ini adalah sebesar Rp. 1.163.000. Berdasarkan tabel 4.9, harga *pulse oximetry* termurah yang dijadikan sebagai pembanding harga dalam penelitian ini sebesar Rp. 1.917.000, yaitu *pulse oximetry* merek Nonin Medical Tipe GO2. Sehingga dapat disimpulkan bahwa alat yang dirancang pada tugas akhir ini lebih murah dengan selisih harga sebesar Rp. 754.000 dari alat *pulse oximetry* termurah yang digunakan sebagai pembanding harga.

5.2. Saran

Saran-saran untuk penelitian selanjutnya adalah :

1. Alat ini dapat ditingkatkan dengan menambah parameter lainnya seperti ECG, suhu tubuh, tekanan darah non invasif, sehingga menjadi rancang bangun patient monitor.

2. Untuk penelitian selanjutnya disarankan agar membuat alat yang lebih *portable* sehingga mudah digunakan dan dibawa.

    i. Dapat ditambahkan *Digital Filter* agar mendapatkan hasil pembacaan yang lebih baik.

    ii. Dapat ditambahkan sistem alarm sehingga dapat memperingatkan saat nilai saturasi (SPO_2) diluar batas normal.

    iii. Ketelitian alat dapat ditambah dengan mengurangi derau yang terdapat pada alat.

DAFTAR PUSTAKA

[1]. Dina, Annisa. 2016. Mengukur Saturasi Darah dengan Pulse Oximeter. https://www.medicalogy.com/blog/mengukur-saturasi-darah-dengan-pulse-oximeter/. Diakses pada tanggal 17 Januari 2017.

[2]. Salamah, Umi, dan Margi Sasono. 2015. Rancang Bangun Pulse Oximetry Berbasis Personal Computer Sebagai Deteksi Kejenuhan Oksigen Dalam Darah. Prosiding Pertemuan dan Presentasi Ilmiah Teknologi Akselerator dan Aplikasinya, Vol : 17, November 2015, Hal : 60 – 64. digilib.batan.go.id/e-prosiding/File%20Prosiding/Iptek%20Nuklir/PSTA/.../Umi.pdf. Diakses pada tanggal 17 Januari 2017.

[3]. Putra, A.A., Kemalasari, Susetyo, P. 2011. Rancang Bangun Pulse Oximetry Digital Berbasis Mikrokontroller. http://repo.pens.ac.id/1327/1/paper.pdf. Diakses pada tanggal 17 Januari 2017.

[4]. Hariyanto, Guruh, dkk. 2013. Rancang Bangun Oksimeter Digital Berbasis Mikrokontroller ATMega16. http://journal.unair.ac.id/download-fullpapers-Guruh%20Hriyanto.pdf. Diakses pada tanggal 17 Januari 2017.

[5]. Raspberry Pi Fondation. FAQS.

https://www.raspberrypi.org/help/faqs/#introWhatIs. Diakses pada tanggal 1 Februari 2017.

[6]. Lopez, Santiago. 2012. Pulse Oximeter Fundamentals and Design. Freescale Semiconductor Application Note.

[7]. Hard. 2016. Jaminan Akurasi Pulse Oximeter Melalui Standar dan Pengujian. http://smtp.lipi.go.id/berita441-Jaminan-Akurasi-Pulse-Oximiter-Melalui-Standar-dan-Pengujian.html. Diakses pada tanggal 6 Februari 2017.

[8]. Istiyanto, Jazi Eko. 2013. Pengantar Elektronika & Instrumentasi. Yogyakarta : Penerbit ANDI.

[9]. Pertiwi, Atit, dkk. 2010. Buku Ajar Sistem Tertanam. Depok : Universitas Gunadarma.

[10]. F, Dwi Lintang, dkk. Pengembangan Sistem Kendali Waktu Nyata dengan *Embedded System* Berbasis *Embedded Linux.* Bandung : Lembaga Ilmu Pengetahuan Indonesia.

[11]. As Sadad, Rif'an Tsaqif, dkk. 2011. Implementasi Mikrokontroler Sebagai Pengendali Lift Empat Lantai. Vol. 14, No. 2, 160-165 : Semesta Teknika.

[12]. Surya, Frans. 2007. I^2C Protokol. Jakarta : Binus.

[13]. Admin. 2017. Arduino Uno Rev3. https://store.arduino.cc/usa/arduino-uno-rev3. Diakses pada tanggal 10 Agustus 2017.

[14]. Sumber Program : https://create.arduino.cc/projecthub/protocentral/using-the-max30100-wearable-pulse-sensor-with-arduino-9b6984.

UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
Jalan Kapten Muchtar Basri No. 3 Medan Sumatera Utara 20238 Indonesia

Berita Acara Bimbingan Tugas Akhir (Skripsi)

Nama : PANDE GUNA KUSWARA

NPM : 1207220029

Judul Tugas Akhir : RANCANG BANGUN PULSE OXIMETRY DIGITAL BERBASIS RASPBERRY PI

No	Tanggal	Catatan	Paraf
	05-09-2019	Perbaiki line space	
	05-09-2019	Penulisan Pada Bab 3 diperbaiki	
	05-09-2019	Analisa Data diperbaiki	
	06-09-2019	Tabel & Grafik diperbaiki	
	06-09-2019	Tambah Jurnal tentang Pulse oximetry	
	06-09-2019	Nomor halaman diperbaiki	
	06-09-2019	Rumusan masalah diperbaiki	
	07-09-2019	Lampiran diperjelas	
	07-09-2019	Kesimpulan diperbaiki	

Pembimbing I

Dr. Ir. Suwarno, MT

UNIVERSITAS MUHAMMADIYAH SUMATERA UTARA
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
Jalan Kapten Muchtar Basri No. 3 Medan Sumatera Utara 20238 Indonesia

Berita Acara Bimbingan Tugas Akhir (Skripsi)

Nama : PANDE GUNA KUSWARA

NPM : 1207220029

Judul Tugas Akhir : RANCANG BANGUN PULSE OXIMETRY DIGITAL BERBASIS RASPBERRY PI

No	Tanggal	Catatan	Paraf
1	2/09/2019	latar belakang Diperbaiki	
2	2/09/2019	Rumusan Masalah Diperbaiki	
3	2/09/2019	Buat Rancangan Gambar	
4	3/09/2019	Cari Jurnal Tentang Pulse Oximetri	
5	4/09/2019	Referensi buat dlm daftar pustaka	
6	4/09/2019	Analisa Data perbaiki	
7	4/09/2019	Buat tabel & Grafik	
8	4/09/2019	Kesimpulan perbaiki	

Acc Seminar

Pembimbing II

Elvy Sahnur Nasution, ST., MT

LAMPIRAN

Program atau perangkat lunak yang digunakan dalam tugas akhir ini didapat dari *website* Arduino. Tautan lengkapnya dapat dilihat di daftar pustaka nomer [14], dengan modifikasi pada ukuran tampilan GUI dan beberapa variabel, yaitu pada variabel headerButton, helpWidget, g, g1, portList, close, start dan oxygenSaturation, agar dapat ditampilkan oleh Raspberry Pi dengan LCD *touchscreen* 3,5".

1. Program Processing

Openview_oxymeter

```
// Need G4P library
import processing.serial.*;
import g4p_controls.*;
import java.awt.*;
import javax.swing.JFileChooser;

import java.math.*;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.util.Date;
import static javax.swing.JOptionPane.*;

import java.io.FileReader;
import java.io.BufferedReader;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

//import signal.library.*;

/************** Packet Validation **********************/
private static final int CESState_Init = 0;
```

```
private static final int CESState_SOF1_Found = 1;
private static final int CESState_SOF2_Found = 2;
private static final int CESState_PktLen_Found = 3;

private static final int CESState_EOF_Wait = 98;
private static final int CESState_EOF_Found = 99;

/*CES CMD IF Packet Format*/
private static final int CES_CMDIF_PKT_START_1 = 0x0A;
private static final int CES_CMDIF_PKT_START_2 = 0xFA;
private static final int CES_CMDIF_PKT_STOP = 0x0B;

/*CES CMD IF Packet Indices*/
private static final int CES_CMDIF_IND_START_1 = 0;
private static final int CES_CMDIF_IND_START_2 = 1;
private static final int CES_CMDIF_IND_LEN = 2;
private static final int CES_CMDIF_IND_LEN_MSB = 3;
private static final int CES_CMDIF_IND_PKTTYPE = 4;
private static final int CES_CMDIF_IND_DATA0 = 5;
private static final int CES_CMDIF_IND_DATA1 = 6;
private static final int CES_CMDIF_IND_DATA2 = 7;

/* CES OTA Data Packet Positions */
private static int CES_OTA_DATA_PKT_POS_LENGTH = 0;
private static int CES_OTA_DATA_PKT_POS_CMD_CAT = 1;
private static int CES_OTA_DATA_PKT_POS_DATA_TYPE = 2;
private static int CES_OTA_DATA_PKT_POS_SENS_TYPE = 3;
private static int CES_OTA_DATA_PKT_POS_RSVD = 4;
private static int CES_OTA_DATA_PKT_POS_SENS_ID = 5;
private static int CES_OTA_DATA_PKT_POS_DATA = 6;

private static int CES_OTA_DATA_PKT_OVERHEAD = 6;
```

```
private static int CES_CMDIF_PKT_OVERHEAD = 5;

/********** Packet Related Variables **********/

int ecs_rx_state = 0;
int CES_Pkt_Len;
int CES_Pkt_Pos_Counter, CES_Pkt_Data_Counter1,
CES_Pkt_Data_Counter2;
int CES_Pkt_PktType;
char DataRcvPacket1[] = new char[1000];
char DataRcvPacket2[] = new char[1000];

/********** ControlP5 Related Variables **********/

int colorValue;
HelpWidget helpWidget;
HeaderButton headerButton;
MessageBox msgBox;
SPO2_cal s;
boolean visibility=false;

/********** Graph Related Variables **********/

double maxAxis_ir, minAxis_ir, maxAxis_red, minAxis_red;
double receivedVoltage_RED, receivedVoltage_IR;
BigDecimal min, max;

/********** File Related Variables **********/

boolean logging = false;
FileWriter output;
JFileChooser jFileChooser;
```

```
Date date;
FileReader readOutput;
String line;
BufferedWriter bufferedWriter;
DateFormat dateFormat;

/*************** Port Related Variables **********************/

Serial port = null;
int Ss = -1;
String[] comList;
boolean serialSet;
boolean portSelected = false;
String portName;
char inString = '\0';
String selectedPort;

/*************** Logo Related Variables **********************/

PImage logo;

/*************** General Variables **********************/

boolean startPlot, Serialevent = false;
String msgs;
int startTime = 0;

int pSize = 400;
float[] xdata = new float[pSize];
float[] ydata = new float[pSize];
float[] AvgYdata = new float[pSize];
float[] zdata = new float[pSize];
```

```
float[] AvgZdata = new float[pSize];
int arrayIndex = 1;
Graph g, g1;
float time =0;
BigDecimal avg, rms, a;
double additionFactor_red, additionFactor_ir;
float value1, value2;
float RedAC = 0, RedDC = 0, IrAC = 0, IrDC = 0;
//SignalFilter myFilter;

public void setup() {
 size(450, 300, JAVA2D);
 //fullScreen();
 createGUI();
 customGUI();
 // Place your setup code here

 date = new Date();
 logo = loadImage("unpam.jpeg");

 headerButton = new HeaderButton(0, 0, width, 30);
 helpWidget = new HelpWidget(0, height - 30, width, 20);
 msgBox = new MessageBox();
 s = new SPO2_cal();
 g = new Graph(75, 45, width-100, 100);
 g1 = new Graph(75, 45, width-100, 200);
 setChartSettings();
 for (int i=0; i<pSize; i++)
 {
  time = time + 2;
  xdata[i]=time;
  ydata[i] = 0;
```

```
    zdata[i] = 0;
  }
  time = 0;
  g.GraphColor = color(0, 255, 0);
  g.Title = "RED";
  g1.GraphColor = color( 0, 255, 0);
  g1.Title = "IR";
 // myFilter = new SignalFilter(this);
}

/********************************************** Draw Function
**********************************************************/

public void draw() {
  background(0);
  while (portSelected == true && serialSet == false)
  {
    startSerial(comList);
  }
  background(0);
  if (startPlot)
  {
    g.LineGraph(xdata, ydata);
    g1.LineGraph(xdata, zdata);
  }

  g.DrawAxis();
  g1.DrawAxis();

 // msgBox.MessageBoxAxis(0, height - 100, width, 70);
 // msgBox.draw();
  headerButton.draw();
```

```
 helpWidget.draw();
}

/********************************************** Opening Port Function ********************************************** **************/

void startSerial(String[] theport)
{
 try
 {
  port = new Serial(this, selectedPort, 57600);
  port.clear();
  serialSet = true;
  msgs = "Port "+selectedPort+" is opened Click Start button";
  portName = "\\"+selectedPort+".txt";
 }
 catch(Exception e)
 {
  msgs = "Port "+selectedPort+" is busy";
  showMessageDialog(null, "Port is busy", "Alert", ERROR_MESSAGE);
  System.exit (0);
 }
}

/********************************************** Serial Port Event Function ************************************************************/

void serialEvent (Serial blePort)
{
 Serialevent = true;
```

```
 inString = blePort.readChar();
 ecsProcessData(inString);
}

/********************************************** Getting Packet
Data                                                    Function
**********************************************************/

void ecsProcessData(char rxch)
{
 switch(ecs_rx_state)
 {
 case CESState_Init:
  if (rxch==CES_CMDIF_PKT_START_1)
  ecs_rx_state=CESState_SOF1_Found;
  break;

 case CESState_SOF1_Found:
  if (rxch==CES_CMDIF_PKT_START_2)
  ecs_rx_state=CESState_SOF2_Found;
  else
   ecs_rx_state=CESState_Init;
  break;

 case CESState_SOF2_Found:
  ecs_rx_state = CESState_PktLen_Found;
  CES_Pkt_Len = (int) rxch;
  CES_Pkt_Pos_Counter = CES_CMDIF_IND_LEN;
  CES_Pkt_Data_Counter1 = 0;
  CES_Pkt_Data_Counter2 = 0;
  break;
```

```
 case CESState_PktLen_Found:
  CES_Pkt_Pos_Counter++;
  if (CES_Pkt_Pos_Counter < CES_CMDIF_PKT_OVERHEAD) //Read
Header
  {
   if (CES_Pkt_Pos_Counter==CES_CMDIF_IND_LEN_MSB)
    CES_Pkt_Len = (int) ((rxch<<8)|CES_Pkt_Len);
   else if (CES_Pkt_Pos_Counter==CES_CMDIF_IND_PKTTYPE)
    CES_Pkt_PktType = (int) rxch;
  } else if ( (CES_Pkt_Pos_Counter >= CES_CMDIF_PKT_OVERHEAD)
&&                    (CES_Pkt_Pos_Counter                    <
CES_CMDIF_PKT_OVERHEAD+CES_Pkt_Len+1) ) //Read Data
  {
   if (CES_Pkt_PktType == 2)
   {
    if (CES_Pkt_Data_Counter1 < 4)
    {
     DataRcvPacket1[CES_Pkt_Data_Counter1]= (char) (rxch);
     CES_Pkt_Data_Counter1++;
    } else
    {
     DataRcvPacket2[CES_Pkt_Data_Counter2]= (char) (rxch);
     CES_Pkt_Data_Counter2++;
    }
   }
  } else //All header and data received
  {
   if (rxch==CES_CMDIF_PKT_STOP)
   {
    int data1 = ecsParsePacket(DataRcvPacket1, DataRcvPacket1.length-
1);
```

```
        int data2 = ecsParsePacket(DataRcvPacket2, DataRcvPacket2.length-
1);
        receivedVoltage_RED = data1 * (0.00000057220458984375) ;
        receivedVoltage_IR = data2 * (0.00000057220458984375) ;

        time = time+0.1;
        xdata[arrayIndex] = time;

        //receivedVoltage_RED                                          =
myFilter.filterUnitFloat((float)receivedVoltage_RED);
        //receivedVoltage_IR                                           =
myFilter.filterUnitFloat((float)receivedVoltage_IR);


        AvgYdata[arrayIndex] = (float)receivedVoltage_RED;
        AvgZdata[arrayIndex] = (float)receivedVoltage_IR;
        value1 = (float)( AvgYdata[arrayIndex] - averageValue(AvgYdata));
        value2 = (float)( AvgZdata[arrayIndex] - averageValue(AvgZdata));
        ydata[arrayIndex] = value1;
        zdata[arrayIndex] = value2;

        float RedDC = (float) averageValue(AvgYdata);
        float IrDC = (float) averageValue(AvgZdata);

        arrayIndex++;
        if (arrayIndex == pSize)
        {
          arrayIndex = 0;
          time = 0;
          RedAC = s.SPO2_Value(ydata);
          IrAC = s.SPO2_Value(zdata);
          float value = (RedAC/abs(RedDC))/(IrAC/abs(IrDC));
```

```
        /******** Emprical Formalae  *********/
        //float SpO2 = 10.0002*(value)-52.887*(value) + 26.817*(value) +
98.293;
        // float SpO2 =((0.81-0.18*(value))/(0.73+0.11*(value)));
        float SpO2=110-25*(value);

        SpO2 = (int)(SpO2 * 100);
        SpO2 = SpO2/100;
        oxygenSaturation.setText(SpO2+"");
      }
      if (startPlot) {
      }
      a = new BigDecimal(averageValue(ydata));
      avg = a.setScale(5, BigDecimal.ROUND_HALF_EVEN);
      a = new BigDecimal(RMSValue(ydata));
      rms = a.setScale(5, BigDecimal.ROUND_HALF_EVEN);
      a = new BigDecimal(max(ydata));
      max = a.setScale(5, BigDecimal.ROUND_HALF_EVEN);
      a = new BigDecimal(min(ydata));
      min = a.setScale(5, BigDecimal.ROUND_HALF_EVEN);
      msgBox.msg(min, max, avg, rms);

      if (logging == true)
      {
       try {
         date = new Date();
         dateFormat = new SimpleDateFormat("HH:mm:ss");
         output = new FileWriter(jFileChooser.getSelectedFile(), true);
         bufferedWriter = new BufferedWriter(output);
         //bufferedWriter.write(dateFormat.format(date)+"            :            "
+receivedVoltage_RED+" , "+receivedVoltage_IR);
```

```
    bufferedWriter.write(arrayIndex-1+" , "+value1 +" , "+value2);
    bufferedWriter.newLine();
    bufferedWriter.flush();
    bufferedWriter.close();
   }
  catch(IOException e) {
    println("It  broke!!!");
    e.printStackTrace();
   }
}

maxAxis_red = max(ydata);
minAxis_red = min(ydata);
// println(maxAxis_red,minAxis_red);
maxAxis_ir = max(zdata);
minAxis_ir = min(zdata);

if (g.yMax != maxAxis_red)
{
  g.yMax = (float)(maxAxis_red);
}
if (g.yMin != minAxis_red)
{
  g.yMin = (float)(minAxis_red);
}

if (g1.yMax != maxAxis_ir)
{
  g1.yMax = (float)(maxAxis_ir);
}
if (g1.yMin != minAxis_ir)
{
```

```
        g1.yMin = (float)(minAxis_ir);
       }

       ecs_rx_state=CESState_Init;
      } else
      {
       ecs_rx_state=CESState_Init;
      }
    }
    break;

  default:
   break;
  }
}

/*********************************************** Recursion Function To Reverse The data ************************************************************/

public int ecsParsePacket(char DataRcvPacket[], int n)
{
  if (n == 0)
    return (int) DataRcvPacket[n]<<(n*8);
  else
    return (DataRcvPacket[n]<<(n*8))| ecsParsePacket(DataRcvPacket, n-1);
}

// Use this method to add additional statements
// to customise the GUI controls
public void customGUI() {
comList = port.list();
```

```
  String comList1[] = new String[comList.length+1];
  comList1[0] = "SELECT THE PORT";
  for (int i = 1; i <= comList.length; i++)
  {
    comList1[i] = comList[i-1];
  }
  start.setEnabled(false);
  oxygenSaturation.setVisible(false);
  comList = comList1;
  portList.setItems(comList1, 0);

  oxygenSaturation.setFont(new Font("Arial", Font.PLAIN, 55));
  oxygenSaturation.setLocalColor(2, color(255, 255, 255));

  start.setLocalColorScheme(GCScheme.CYAN_SCHEME);
}

void setChartSettings() {
  g.xDiv=10;
  g.xMax=pSize;
  g.xMin=0;
  g.yMax=0.001;
  g.yMin=0.005;

  g1.xDiv=10;
  g1.xMax=pSize;
  g1.xMin=0;
  g1.yMax=0.001;
  g1.yMin=0.005;
}

double averageValue(float dataArray[])
```

```
{

 float total = 0;
 for (int i=0; i<dataArray.length; i++)
 {
  total = total + dataArray[i];
 }

 return total/dataArray.length;
}

double RMSValue(float dataArray[])
{
 float total = 0;
 for (int i=0; i<dataArray.length; i++)
 {
  total = (float)(total + Math.pow(dataArray[i], 2));
 }
 total /= dataArray.length;
 return Math.sqrt(total);
}

Graph
class Graph
{

 boolean Dot=true;           // Draw dots at each data point if true
 boolean RightAxis;              // Draw the next graph using the right axis if
true
 boolean ErrorFlag=false;        // If the time array isn't in ascending order,
make true
```

```
 boolean  ShowMouseLines=true;    // Draw lines and give values of the
mouse position

 int     xDiv=5, yDiv=5;           // Number of sub divisions
 int     xPos, yPos;           // location of the top left corner of the graph
 int     Width, Height;        // Width and height of the graph


 color  GraphColor;
 color BackgroundColor=color(255);
 color  StrokeColor=color(180);

 String  Title="Title";          // Default titles
 String xLabel="x - Label";
 String  yLabel="y - Label";

 float   yMax, yMin;      // Default axis dimensions
 float xMax=10, xMin=0;
 float yMaxRight=1024, yMinRight=0;

 PFont   Font;                 // Selected font used for text

 Graph(int x, int y, int w, int h) { // The main declaration function
   xPos = x;
   yPos = y;
   Width = w;
   Height = h;
 }

 void DrawAxis() {
```

```
    fill(0);
    color(0);
    stroke(0);
    strokeWeight(1);
    int t=60;

    // rect(xPos-t*1.6,yPos-t,Width+t*2.5,Height+t*2);          // outline
    textAlign(CENTER);
    textSize(25);

    fill(255);
    // text(Title,xPos+Width/5,yPos+20);                        // Heading Title
    textAlign(CENTER);
    textSize(10);
    text("No.Of Samples - "+Title, xPos+Width/2, yPos+Height+t/2);
// x-axis Label

    rotate(-PI/2);                                  // rotate -90 degrees
    text("", -yPos-Height/2, xPos-t*1.6+40);                // y-axis Label
    rotate(PI/2);                                   // rotate back
    fill(255);
    textSize(10);
    noFill();
    stroke(0);
    smooth();
    strokeWeight(1);
    //Edges
    stroke(255);
    line(xPos-3, yPos+Height, xPos-3, yPos);                    // y-axis line
    line(xPos-3, yPos+Height, xPos+Width+5, yPos+Height);          // x-axis
line
```

```
  stroke(200);

  if (yMin<0) {
   line(xPos-7, // zero line
    yPos+Height-(abs(yMin)/(yMax-yMin))*Height, //
    xPos+Width,
    yPos+Height-(abs(yMin)/(yMax-yMin))*Height
    );
  }

  stroke(255);

  for (int x=0; x<=xDiv; x++) {

   line(float(x)/xDiv*Width+xPos-3, yPos+Height, // x-axis Sub devisions
    float(x)/xDiv*Width+xPos-3, yPos+Height+5);

   textSize(7.5);                          // x-axis Labels

   String xAxis=str((xMin+float(x)/xDiv*(xMax-xMin))); // the only way
to get a specific number of decimals
   String[] xAxisMS=split(xAxis, '.');          // is to split the float into
strings
   text(xAxisMS[0]+"."+xAxisMS[1].charAt(0), // ...
    float(x)/xDiv*Width+xPos-3, yPos+Height+15); // x-axis Labels
  }

  for (int y=0; y<=yDiv; y++) {
   line(xPos-3, float(y)/yDiv*Height+yPos, // ...
    xPos-7, float(y)/yDiv*Height+yPos);          // y-axis lines
```

```
    textAlign(RIGHT);
    fill(255);

    String yAxis=str(yMin+float(y)/yDiv*(yMax-yMin));     // Make y Label
a string
    String[] yAxisMS=split(yAxis, '.');                   // Split string
    //println(yAxisMS);
    text(yAxisMS[0]+"."+yAxisMS[1], // ...
      xPos-15, float(yDiv-y)/yDiv*Height+yPos+3);       // y-axis Labels
   }
   stroke(0);
  }
  void LineGraph(float[] x, float[] y) {

   for (int i=0; i<(x.length-1); i++)
   {
    smooth();
    strokeWeight(3);
    stroke(GraphColor);
    line(xPos+(x[i]-x[0])/(x[x.length-1]-x[0])*Width,
      yPos+(Height)-(y[i]/(yMax-yMin)*Height)+(yMin)/(yMax-
yMin)*Height,
      xPos+(x[i+1]-x[0])/(x[x.length-1]-x[0])*Width,
      yPos+(Height)-(y[i+1]/(yMax-yMin)*Height)+(yMin)/(yMax-
yMin)*Height);

   }

   stroke(0);
   fill(0);
   rect(xPos-22+((time-2)-x[0])/(x[x.length-1]-x[0])*Width,0,50,height);
  }
```

```
}

HeaderButton
class HeaderButton {

  public float x, y, w, h;
  int padding = 5;

  HeaderButton(float _xPos, float _yPos, float _width, float _height) {
    x = _xPos;
    y = _yPos;
    w = _width;
    h = _height;
  }

  public void update() {
  }

  public void draw() {

    pushStyle();
    noStroke();

    // draw background of header
    fill(0,102,204);
    rect(x, y, width, h);

    //draw background for the buttons to be placed
    strokeWeight(1);
    stroke(color(0, 5, 11));
    fill(color(0, 5, 11));
```

```
    popStyle();
  }

};
```

HelpWidget

```
class HelpWidget {

  public float x, y, w, h;
  String currentOutput = "..."; //current text shown in help widget, based on most recent command

  int padding = 5;

  HelpWidget(float _xPos, float _yPos, float _width, float _height) {
    x = _xPos;
    y = _yPos;
    w = _width;
    h = _height;
  }

  public void update() {
  }

  public void draw() {

    pushStyle();
    noStroke();

    // draw background of widget
    fill(0,102,204);
    rect(x, height-h, width, h);
```

```
    //draw bg of text field of widget
    strokeWeight(1);
    stroke(color(0, 5, 11));
    fill(color(0, 5, 11));
    rect(x + padding, height-h + padding, width - padding*5 - 128, h -
padding *2);

    textSize(14);
    fill(255);
    textAlign(LEFT, TOP);
    text(currentOutput, padding*2, height - h + padding + 4);

    //draw LOGO
    image(logo, width - (128+padding*2), height - 17,25,15);

    popStyle();
  }
  public void output(String _output) {
    currentOutput = _output;
  }
};

public void output(String _output) {
  helpWidget.output(_output);
}

MessageBox
class MessageBox {

  public float x, y, w, h;
  boolean colorValue = true;
```

```
String Mini = "0.0";
String Max = "0.0";
String Avg = "0.0";
String RMS = "0.0";
String MP = "0.0";
String MPDATE = ""+new Date();
int padding = 5;

void MessageBoxAxis(float _xPos, float _yPos, float _width,  float
_height) {
  x = _xPos;
  y = _yPos;
  w = _width;
  h = _height;
}

public void update() {
}

public void draw() {

  pushStyle();
  noStroke();
  strokeWeight(1);
  stroke(color(0, 5, 11));
  fill(color(0));
  rect(x + padding, y + padding, w - padding*2, h - padding *2);

  fill(255);
  textAlign(LEFT, TOP);
  textFont(createFont("Arial Bold", 18));
```

```
    text("Min", width/3.5, y + padding + 4);
    text(": "+Mini+" Pounds", width/3, y + padding + 4);
    text("Peak", width/1.7, y + padding + 4);
    text(": "+Max+" Pounds", width/1.55, y + padding + 4);


    text("Avg", width/3.5, y + padding + 34);
    text(": "+Avg+" Pounds", width/3, y + padding + 34);
    text("RMS", width/1.7, y + padding + 34);
    text(": "+RMS+" Pounds", width/1.55, y + padding + 34);

    popStyle();
  }
  public void msg(BigDecimal mini, BigDecimal max, BigDecimal avg,
BigDecimal rms) {

    Mini = mini+"";
    Max = max+"";
    Avg = avg+"";
    RMS = rms+"";
    if (Mini.length() >= 10)
    {
      Mini = Mini.substring(0, 10);
    }
    if (Max.length() >= 10)
    {
      Max = Max.substring(0, 10);
    }
    if (Avg.length() >= 10)
    {
```

```
    Avg = Avg.substring(0, 10);
   }
   if (RMS.length() >= 10)
   {
    RMS = RMS.substring(0, 10);
   }
   // prevOutputs.add(_output);
  }
};

/*public void msg(BigDecimal mini, BigDecimal max, BigDecimal avg,
BigDecimal rms) {
 msgBox.msg(mini, max, avg, rms);
 }*/

SPO2_cal
public class SPO2_cal
{
  float Vdc = 0;
  float Vac = 0;
  float spo2_cal_array[] = new float[pSize];

  float SPO2 = 0;


  public float SPO2_Value(float spo2_array[])
  {
   SPO2 = 0;
   int k = 0;
   for(int i = 50; i < spo2_array.length; i++)
   {
    // float roundoff = Math.round((spo2_array[i] * spo2_array[i])*100)/100;
```

```
    spo2_cal_array[k++] = spo2_array[i] * spo2_array[i];
   }
   SPO2 = sum(spo2_cal_array, k);

   return (SPO2);
  }


  public float sum(float array[], int len)
  {
   float spo2 = 0;
   for (int p = 0; p < len; p++)
   {
    spo2 = spo2 + array[p];
   }
   Vac = (float)Math.sqrt(spo2/len);

   //println(Vac);
   return Vac;
  }
}
```

Gui

```
/*
=========================================================
===
 * ====                WARNING                    ===
 *
=========================================================
===
 * The code in this tab has been generated from the GUI form
 * designer and care should be taken when editing this file.
```

```
 * Only add/edit code inside the event handlers i.e. only
 * use lines between the matching comment tags. e.g.

 void myBtnEvents(GButton button) { //_CODE_:button1:12356:
     // It is safe to enter your event code here
 } //_CODE_:button1:12356:

 * Do not rename this tab!
 *
=============================================================
===
 */

public void port_click(GDropList source, GEvent event) {
  selectedPort = portList.getSelectedText();
  portSelected = true;
  portList.setEnabled(false);
  start.setEnabled(true);
  start.setLocalColorScheme(GCScheme.GREEN_SCHEME);
  portList.setLocalColorScheme(GCScheme.CYAN_SCHEME);
}

public void close_click(GButton source, GEvent event) {
  exit();
}

public void start_click(GButton source, GEvent event) {
  start.setEnabled(false);
  startPlot = true;
  oxygenSaturation.setVisible(true);
  start.setLocalColorScheme(GCScheme.CYAN_SCHEME);
}
```

```
// Create all the GUI controls.
// autogenerated do not edit
public void createGUI(){
G4P.messagesEnabled(false);
  G4P.setGlobalColorScheme(GCScheme.BLUE_SCHEME);
  G4P.setCursor(ARROW);
  surface.setTitle("Pulse_Oximeter");
  portList = new GDropList(this, 10, 5, 150, 200, 10);
  portList.setItems(loadStrings("list_626048"), 0);
  portList.setLocalColorScheme(GCScheme.GREEN_SCHEME);
  portList.addEventHandler(this, "port_click");
  close = new GButton(this, 270, 5, 50, 20);
  close.setText("CLOSE");
  close.setTextBold();
  close.setLocalColorScheme(GCScheme.GREEN_SCHEME);
  close.addEventHandler(this, "close_click");
  start = new GButton(this, 200, 5, 50, 20);
  start.setText("START");
  start.setTextBold();
  start.setLocalColorScheme(GCScheme.GREEN_SCHEME);
  start.addEventHandler(this, "start_click");
  oxygenSaturation = new GLabel(this, 250, 30, 200, 70);
  oxygenSaturation.setText("0");
  oxygenSaturation.setTextBold();
  oxygenSaturation.setOpaque(false);
}

// Variable declarations
// autogenerated do not edit
GDropList portList;
```

```
GButton close;
GButton start;
GLabel oxygenSaturation;
```

2. Program Arduino

```
#include "Protocentral_MAX30100.h"
#include <Wire.h>

MAX30100 sensor;
uint8_t data_len=8;
uint8_t DataPacketHeader[15];
volatile long IRR,REDD;

void setup() {
  Wire.begin();
  Serial.begin(57600);
  while(!Serial);
  sensor.begin(pw1600, i50, sr100 );
  sensor.printRegisters();
}

void loop() {
   sensor.readSensor();
   IRR=sensor.IR;
   REDD=sensor.RED;
   DataPacketHeader[0] = 0x0A;
   DataPacketHeader[1] = 0xFA;
    DataPacketHeader[2] = (uint8_t) (data_len);
    DataPacketHeader[3] = (uint8_t) (data_len>>8);
    DataPacketHeader[4] = 0x02;
```

```
    DataPacketHeader[5] = REDD;
    DataPacketHeader[6] = REDD>>8;
    DataPacketHeader[7] = REDD>>16;
    DataPacketHeader[8] = REDD>>24;


    DataPacketHeader[9] = IRR;
    DataPacketHeader[10] = IRR>>8;
    DataPacketHeader[11] = IRR>>16;
    DataPacketHeader[12] = IRR>>24;

    DataPacketHeader[13] = 0x00;
    DataPacketHeader[14] = 0x0b;

    for(int i=0; i<15; i++) // transmit the data
    {
      Serial.write(DataPacketHeader[i]);
     }

//Serial.println(sensor.getPartID());
/*
  Serial.println(REDD,HEX);
  Serial.println(IRR,HEX);*/
  delay(10);
}
```

3. Pustaka Protocentral_MAX30100.h

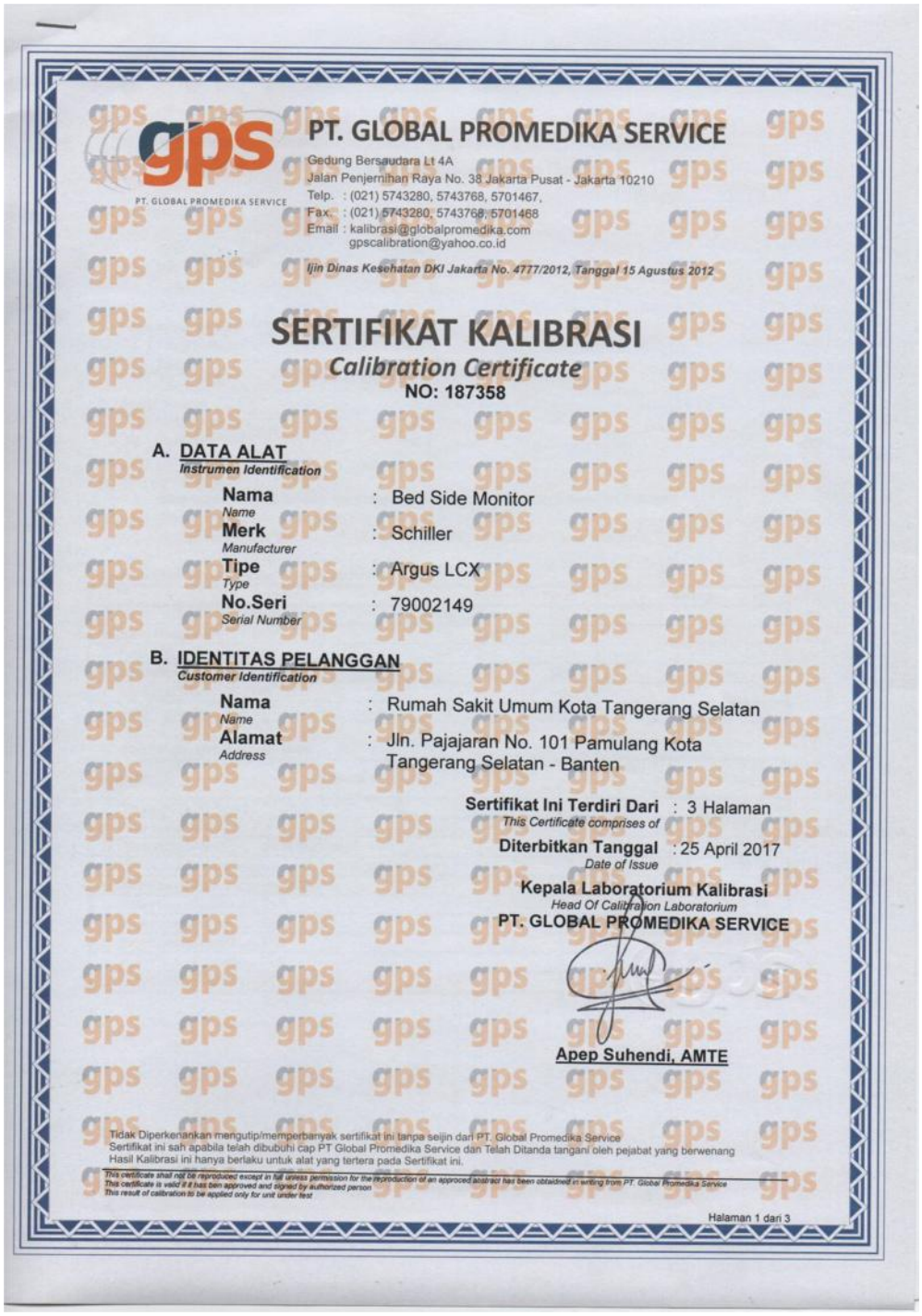
```
#define MAX30100_INT_STATUS     0x00 // Which interrupts are tripped
#define MAX30100_INT_ENABLE     0x01 // Which interrupts are active
#define MAX30100_FIFO_WR_PTR    0x02 // Where data is being written
#define MAX30100_OVRFLOW_CTR 0x03 // Number of lost samples
#define MAX30100_FIFO_RD_PTR    0x04 // Where to read from
```

```
#define MAX30100_FIFO_DATA      0x05 // Ouput data buffer
#define MAX30100_MODE_CONFIG    0x06 // Control register
#define MAX30100_SPO2_CONFIG    0x07 // Oximetry settings
#define MAX30100_LED_CONFIG     0x09 // Pulse width and power of
LEDs
#define MAX30100_TEMP_INTG      0x16 // Temperature value, whole
number
#define MAX30100_TEMP_FRAC      0x17 // Temperature value, fraction
#define MAX30100_REV_ID         0xFE // Part revision
#define MAX30100_PART_ID        0xFF // Part ID, normally 0x11

#define MAX30100_ADDRESS        0x57 // 8bit address converted to 7bit

typedef unsigned char uint8_t;
typedef unsigned int uint16_t;
typedef enum{ // This is the same for both LEDs
  pw200,    // 200us pulse
  pw400,    // 400us pulse
  pw800,    // 800us pulse
  pw1600    // 1600us pulse
}pulseWidth;

typedef enum{
  sr50,    // 50 samples per second
  sr100, // 100 samples per second
  sr167, // 167 samples per second
  sr200, // 200 samples per second
  sr400, // 400 samples per second
  sr600, // 600 samples per second
  sr800, // 800 samples per second
  sr1000 // 1000 samples per second
}sampleRate;
```

```
typedef enum{
  i0,    // No current
  i4,    // 4.4mA
  i8,    // 7.6mA
  i11, // 11.0mA
  i14, // 14.2mA
  i17, // 17.4mA
  i21, // 20.8mA
  i27, // 27.1mA
  i31, // 30.6mA
  i34, // 33.8mA
  i37, // 37.0mA
  i40, // 40.2mA
  i44, // 43.6mA
  i47, // 46.8mA
  i50    // 50.0mA
}ledCurrent;

class MAX30100 {
public:
  uint16_t IR = 0;      // Last IR reflectance datapoint
  uint16_t RED = 0;     // Last Red reflectance datapoint

  MAX30100();
  void setLEDs(pulseWidth pw, ledCurrent red, ledCurrent ir); // Sets the
LED state
  void    setSPO2(sampleRate sr); // Setup the SPO2 sensor, disabled by
default
  int   getNumSamp(void);       // Get number of samples
  void  readSensor(void);       // Updates the values
  void shutdown(void); // Instructs device to power-save
```

```
  void  reset(void);      // Resets the device
  void  startup(void);    // Leaves power-save
  int  getRevID(void);  // Gets revision ID
  int getPartID(void); // Gets part ID
  void begin(pulseWidth pw = pw1600, // Longest pulseWidth
             ledCurrent ir = i50,    // Highest current
             sampleRate sr = sr100); // 2nd lowest sampleRate
  void printRegisters(void); // Dumps contents of registers for debug

private:
  void    I2CwriteByte(uint8_t address, uint8_t subAddress, uint8_t data);
  uint8_t I2CreadByte(uint8_t address, uint8_t subAddress);
  void      I2CreadBytes(uint8_t address, uint8_t subAddress, uint8_t * dest,
uint8_t count);
};
```

4. Sertifikat Kalibrasi Alat Pembanding (Bed Side Monitor Merek Argus LCX)



PT. GLOBAL PROMEDIKA SERVICE

Gedung Bersaudara Lt 4A
Jalan Penjernihan Raya No. 38 Jakarta Pusat - Jakarta 10210
Telp. : (021) 5743280, 5743768, 5701467,
Fax. : (021) 5743280, 5743768, 5701468
Email : kalibrasi@globalpromedika.com
gpscalibration@yahoo.co.id

Ijin Dinas Kesehatan DKI Jakarta No. 4777/2012, Tanggal 15 Agustus 2012

SERTIFIKAT KALIBRASI

Calibration Certificate

NO: 187358

A. DATA ALAT
Instrumen Identification

Nama *Name*	:	Bed Side Monitor
Merk *Manufacturer*	:	Schiller
Tipe *Type*	:	Argus LCX
No.Seri *Serial Number*	:	79002149

B. IDENTITAS PELANGGAN
Customer Identification

Nama *Name*	:	Rumah Sakit Umum Kota Tangerang Selatan
Alamat *Address*	:	Jln. Pajajaran No. 101 Pamulang Kota Tangerang Selatan - Banten

Sertifikat Ini Terdiri Dari : 3 Halaman
This Certificate comprises of

Diterbitkan Tanggal : 25 April 2017
Date of Issue

Kepala Laboratorium Kalibrasi
Head Of Calibration Laboratorium
PT. GLOBAL PROMEDIKA SERVICE

Apep Suhendi, AMTE

Tidak Diperkenankan mengutip/memperbanyak sertifikat ini tanpa seijin dari PT. Global Promedika Service
Sertifikat ini sah apabila telah dibubuhi cap PT Global Promedika Service dan Telah Ditanda tangani oleh pejabat yang berwenang
Hasil Kalibrasi ini hanya berlaku untuk alat yang tertera pada Sertifikat ini.

This certificate shall not be reproduced except in full unless permission for the reproduction of an approved abstract has been obtained in writing from PT. Global Promedika Service
This certificate is valid if it has been approved and signed by authorized person
This result of calibration to be applied only for unit under test

Halaman 1 dari 3

5. Penawaran Harga

PT. IDS Medical Systems Indonesia
Wisma 76, Lt. 17 & 22 Jl. Letjen. S. Parman Kav. 76
Slipi, Jakarta 11410, Indonesia
T : (6221) 2567 8989 F : (6221) 5366 1038

Kepada Yth.
DIREKTUR
RSUD Tangerang Selatan
Di Tangerang

Hal : Informasi Harga
No : 1087/JKT/SLS/AM/III/15 **Tanggal : 31 Maret 2015**

Dengan hormat,

Sebelumnya kami ucapkan terimakasih atas kepercayaan yang diberikan kepada PT. IDS Medical Systems Indonesia. Berikut kami sampaikan informasi harga untuk :

1 (satu) unit	**Pulse Oxymetri** **Type : Tuffsat** **Merk : GE Healthcare**	**Rp. 13.736.250,-**
1 (satu) Unit	**Pulse Oxymetri** **Type : Trusat** **Merk : GE Healthcare**	**Rp. 40.445.625,-**

Syarat dan kondisi Informasi harga kami adalah:

Harga	***: Franco Jakarta, sudah termasuk PPn 10%***
Pembayaran	***: Cash atau sesuai perjanjian***
Pengiriman Barang	***: Ready stock atau Indent 3 (tiga) bulan***
Masa Berlaku	***: 1 (satu) bulan***
Masa Garansi	***: 1 (satu) Tahun***

Demikian informasi harga kami sampaikan. Atas perhatiannya kami ucapkan terima kasih

Hormat Kami,
PT. IDS Medical Systems Indonesia

IDS Medical Systems Indonesia JAKARTA

Rizqi Widhiyanto
Manager

PIC : Ahmad Mukhlis (081347501776)

Noni n Medical G02 Achieve Personal Fingertip Pulse Oximeter, Blue, Made in the USA with 2-y ear Warranty - inti

lulls ulasan urnukprod'uk n (https:/M'MV lazada co.id/nonin-medical-go2-.achkMl-peu.onal-llngttrtlp-pub-oxrne1er-bluemade-1rHh e--usa-'Mth-2-year-warranty-intl-34773563.hl ml?spm=a2o4j.brand-86413.0.0.mt1Dct<&tf=1&$C=-MY1RA0=::& , b-8641 3#)

Bland . Nonin Medical (httpsJNM'W.lazada.co.ld/r.onm-medical_1/)

I Selanjutnya .AksesorlS Otahra,ga & OUtdoofdari Non1n Medica1 (https:/lWww lalada.eo.1d/akSttQltS-<>lah raga-dan-outdoOrlnorun-medM;al_11)

[I]

Tambahkan ke wiShllsl Bagikan 0 (https://www.lazada.co.id/nonin-medical-go2-achieve-p-

#1 Choice of Respiratory Therapists: More RT\'s choose r-.bnin puse oximeters than any other brand

more

Variasi

RP 1.917.000

Sebell.m RP--2.098.000:

Diskon 9%

Clc ll3n hlng!J3 J b ulan, h3nya RP 539.000 per bulan.

Cicllan hingga 6 b ula n. h3nya R.P 3 1 9.500 per bula n.

Clcl!an hlngga 12 bulan , hanya RP 159.750 per bLll3 n.

I-bl$ Terjwt ◦ Stock Seger::i di Update

Kaban saya jtka produk n sudah !ersedia

.,ou,ema@emad com KABARISAYAI

Dattar News!elterl

Piihtn pe(lliriman 0

Dikiim ke

Cengkareng di Kota Jakart.a rat. OKI Jakart..1

ProdlA-:: Ini dikirim dari luar negeri

Baya r di Tempat tidak berlaku .n:.Jt produfi: iri

7 Hari Pergembalian

Bert.bah piki"a n tidak diperkerankan 0

TIL'bk ada garansi

Diiua1ole h

Koo Pang (https://www.lazada co.id/koo-pang)

Kelompok Seller (;) PerlOde di Lazae'.Sa (;)

3 """"

)ME | E-Catalog LKPP » | E-Catalog OneMed » | Pameran

Pulse Oxymeter Oxy9Wave

Pulse Oxymeter Oxy9Wave

Harga : Rp. 5.995.000

Merek : BIONET

Brosur Pulse Oxymeter Oxy9Wave

Feature Spesifikasi

High Performance
- 3.2" Color TFT LCD for Clear View
- Large memory: Up to 30 Days in 10 Secs
- Nellcor compatibility

User Convenieence
- Vertical & horizontal display of measurement value
- Intuitive operation

ADDRESS

AYAMAS MEDICA INDUSTRI
By Pass krian KM 28
Ds. Sidomojo, Krian, Sidoarjo
lawa Timur - Indonesia

Email
ales@onemed.co.id

CALL CENTER

T: +6231 8982349
F: +6231 8985268

SOCIAL PROFILES

Mau dapat info email menarik seputar produk kami? Untuk berlangganan isikan data berikut:
Name

Email *

Subscribe

E-Catalog

Belanja Online Alat Kesehatan Rumah Sakit , Puskesmas , Klinik , Dokter, Bidan praktek Yang transparant , Murah , dan Berkualia

6. Prosedur Penggunaan Alat

Prosedur penggunaan alat *Pule Oximetry* yang dirancang dalam tugas akhir ini diuraikan dalam langkah-langkah berikut ini :

1) Hubungkan *stecker* ke *power outlet*.
2) Aktifkan saklar pada alat.
3) Tunggu hinggga Raspberry menampilkan layar Dekstop.
4) Buka program Processing.
5) Pada Menu Bar program Processing, pilih File → Open Recent → pi → openview_oximeter.
6) Klik simbol Run  .
7) Pilih 

untuk menampilkan port yang tersedia pada raspberry, lalu pilih /dev/ttyACM0.
8) Klik simbol start 

.
9) Tempelkan salah satu ujung jari pada sensor. Tunggu hingga muncul nilai SPO_2 dan grafik yang muncul stabil.
10) Setelah selesai pengukuran, klik simbol close 

.

Setelah selesai melakukan pengukuran, matikan alat dengan prosedur berikut ini :

1) Pada *Application menu*, klik *shutdown*.
2) Tunggu hingga LED kuning yang berkedip pada Raspberry tidak aktif.
3) Nonaktifkan saklar pada alat.
4) Bila perlu, cabut *stecker* dari *power outlet.*

DAFTAR RIWAYAT HIDUP

Nama	: Pande Guna Kuswara
Tempat & Tanggal Lahir	: Bayur Sumbar, 01 September 1994
Jenis Kelamin	: Laki-laki
Agama	: Islam
Status	: Belum Menikah
Alamat	: Permumahan Mekarsari Blok B NO. 14, Jalan Satria, Delitua
No. Telp/Hp	: 089692650090
Email	: pandeguna@gmail.com

PENDIDIKAN FORMAL

Tahun 2012 - 2019	: Mahasiswa S1 Teknik Elektro Universitas Muhammadiyah Sumatera Utara
Tahun 2009 - 2012	: SMA Harapan Mandiri
Tahun 2006 - 2009	: SMP Harapan Mandiri
Tahun 2000 - 2006	: SD Kemala Bhayangkari 1 Medan