

**PENGEMBANGAN PROGRAM SERTA SISTEM TRANSMISI
UNTUK INTERFACING *PULSE OXIMETER* DENGAN
SMARTPHONE**

LAPORAN KERJA PRAKTEK

di

Rumah Sakit Umum Anwar Medika, Krian, Sidoarjo

Oleh

DAFA FARIS MUHAMMAD

NIM: 13215044



**PROGRAM STUDI TEKNIK ELEKTRO
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2018**

© 2018
Dafa Faris Muhammad
ALL RIGHTS RESERVED

ABSTRAK

PENGEMBANGAN PROGRAM SERTA SISTEM TRANSMISI UNTUK INTERFACING *PULSE OXIMETER* DENGAN SMARTPHONE

Oleh

Dafa Faris Muhammad

NIM: 13215044

PROGRAM STUDI TEKNIK ELEKTRO

Dalam masa pelaksanaan studinya, seorang mahasiswa teknik memperoleh banyak sekali hal mulai dari ilmu tentang bidang keahliannya hingga pengalaman dalam lingkungan laboratorium. Tidak jarang juga pengalaman berorganisasi diperoleh selama berkegiatan di kampus. Sayangnya pengalaman yang lebih dekat dengan kondisi kerja nyata tidak mudah untuk didapatkan; dimana kemampuan teknis, sosial, maupun berorganisasi diuji dalam kondisi yang jauh lebih profesional. Oleh karena itulah, demi mempersiapkan lulusan yang lebih matang, dimunculkan urgensi untuk melaksanakan kerja praktek – penempatan mahasiswa ke suatu perusahaan.

Kerja praktek yang dipilih merupakan Rumah Sakit Umum Anwar Medika, suatu Rumah Sakit Umum yang ditempatkan pada Krian, Sidoarjo. Pemilihan lokasi tersebut tentunya akan membuat fokus terletak pada dunia medis, dunia yang sedang mengalami perkembangan pesat termasuk integrasi teknologi. Sayangnya tidak jarang harga yang tinggi dengan fungsionalitas yang terbatas menjadi kelemahan dari produk-produk tersebut, padahal umumnya dibutuhkan suatu alat yang sama dengan kuantitas yang besar. Sehingga Penciptaan alat yang relatif murah dan mudah untuk digantikan akan sangat revolusioner.

Secara umum kerja praktek ini dibagi menjadi tiga bagian: pelaksanaan proyek mandiri, pembelajaran melalui observasi secara langsung para karyawan berkerja, maupun pembelajaran melalui penjelasan secara langsung dari orang-orang yang lebih berpengalaman. Dua bagian akhir tentunya sudah cukup jelas. Untuk bagian pertama, proyek mandiri, tahapan yang dilalui adalah pendesainan GUI untuk app smartphone, pendesainan sistem transmisi data dari oximeter, implementasi sistem transmisi pada oximeter, implementasi sistem rekonstruksi data pada app, dan diakhiri dengan pengujian dan penyempurnaan.

Dari dilakukannya kerja praktek selama dua bulan di Rumah Sakit Umum, ada banyak sekali yang bisa diperoleh. Untuk bagian proyek mandiri, hal-hal yang diperoleh tentunya akan sangat teknis. Namun pada umumnya hal-hal tersebut bersangkutan dengan aspek praktis dari transmisi modulasi amplituda (metode transmisi yang dipilih untuk penyelesaian proyek), serta pemahaman lebih mendalam tentang pengembangan program Android. Sedangkan untuk bagian lainnya, diperoleh ilmu-ilmu tentang bagaimanakah sistem kelistrikan dasar suatu perusahaan yang besar (terlebih lagi dalam perusahaan dimana ketersediaan listrik adalah faktor yang krusial), listrik dalam keadaan darurat maupun penanganan keadaan darurat pada umumnya, pemompaan air utama, hydrant, proses-proses dalam menambah maupun merawat sarana-sarana yang ada, hal-hal tentang pelayanan maupun peningkatan kepuasan pengunjung, dll.

Kerja praktek yang berjalan cukup singkat ini pada akhirnya kembali ke bentuk edukasi yang sangat fundamental, yaitu pembelajaran melalui lingkungan sekitar. Namun sudah sangat jelas bahwa pembelajaran yang diperoleh akan sangat relatif antara perusahaan yang satu dengan yang lain, bahkan antara mahasiswa yang satu dengan yang lain meski di perusahaan yang sama. Perolehan yang diterima tetaplah bertitik berat pada sebagaimana besar para pelaksana kerja praktek mengejar pengalaman-pengalaman baru itu tadi.

Kata kunci: pengalaman, lingkungan, perusahaan, rumah sakit, proyek, smartphone

ABSTRACT

DEVELOPING PROGRAM AND TRANSMISSION SYSTEM FOR INTERFACING *PULSE OXIMETER* WITH SMARTPHONE

By

Dafa Faris Muhammad

NIM: 13215044

ELECTRICAL ENGINEERING STUDY PROGRAM

During their studying period, a college student obtained a substantial amount of stuffs ranging from knowledge about their field of expertise to experience inside a laboratorial environment. It's also pretty common that organizational experience obtained alongside their campus life. Unfortunately, experience closely related to real working conditions isn't something straightforward to be acquired; where technical, social, and organizational ability are tested inside a professional environment. Therefore, under the preface of preparing seasoned graduates, urgency arised for their students to apply for internship – placing students inside a company.

Company chosen for this internship is Anwar Medika General Hospital, a hospital located in Krian, Sidoarjo. It's obvious that choosing that company will make the whole focus pointed at medical world, a world that's currently undergoing rapid development -- including technological integration. Unfortunately, it's common that high price and very specific functionality become a shortcoming from all of those technological products, even though buying a high quantity of a certain product is often a requirement. Therefore, the development and manufacturing of an instrument with cheaper and easier to replace will be revolutionary.

Generally, this internship divided into three main sections: individual project, learning through directly observing the employees working, and learning through directly explained by the more experienced. The latter two sections is surely obvious. For the first part, individual project, the stages are pretty much designing smartphone app GUI, designing oximeter transmission system, implementing data reconstruction inside the smartphone app, and finalized with testing and polishing.

During these two months internship inside a hospital, there're a lot of things acquired across all three main sections. For individual project, it's pretty much evident that those things are very technical. However, for the most part, those all are related to practical implementation of transmission using amplitude modulation (the transmission method selected for completing the project), and a deeper understanding around developing an Android program. Meanwhile for the other parts, knowledge about basic electrical system inside a big company (moreover a company where electrical availability is a crucial factor), emergency electrical system, big water pump system, hydrant, process for adding and maintaining available infrastructure, some basic things about service and increasing visitors satisfaction, etc. are also acquired.

Internship, while pretty short living, comes as a very fundamental form of education; which is learning from the wholly new environment. Nevertheless, it's very clear that things learned will be very relative between companies, even between students in the same company. The gain is focused on how earnest are the students at chasing those new experiences.

Keywords: experience, environment, company, hospital, project, smartphone

**PENGEMBANGAN PROGRAM SERTA SISTEM TRANSMISI
UNTUK INTERFACING *PULSE OXIMETER* DENGAN
SMARTPHONE**

Oleh

Dafa Faris Muhammad

Laporan kerja praktek ini telah diterima dan disahkan
sebagai persyaratan untuk memperoleh nilai

MATA KULIAH EL4091

di

**PROGRAM STUDI TEKNIK ELEKTRO
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

Bandung, 3 Agustus 2018

Disetujui oleh:

Penanggung Jawab
Mata Kuliah EL4091,

Penanggung Jawab
di Lokasi Kerja Praktek,

.....

Subakri

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, yang atas rahmat dan karunia Nya penulis dapat menyelesaikan penulisan laporan kerja praktek ini. Shalawat dan salam tercurah kepada Rasulullah Muhammad SAW beserta keluarganya.

Selama melaksanakan kerja praktek ini, penulis mendapat bantuan dan dukungan dari berbagai pihak. Untuk itu, penulis ingin mengucapkan terima kasih kepada :

1. RSUD Anwar Medika, yang telah memberikan kesempatan kepada penulis untuk melaksanakan kerja praktek;
2. bapak Subakri, selaku pembimbing di lokasi tempat kerja praktek dilaksanakan, yang telah memberikan bimbingan selama melaksanakan kerja praktek ini;
3. ibu dr. Nungky Taniasari, MARS., selaku Direktur RSUD. Anwar Medika, yang telah memberikan pengarahan selama melaksanakan kerja praktek;
4. ibu dr. Farida Anwari, MPH., MM., selaku ketua senat akademik STIKES RS. Anwar Medika, yang telah memotivasi dan memberikan gambaran luas tentang alat kesehatan;
5. ibu, selaku dosen penanggung jawab mata kuliah EL4091,
6. seluruh staf dan karyawan IPS & Umum, yang telah memberikan bantuannya;
7. kedua orang tua yang telah memberikan jalan, saran, maupun support yang besar selama proses kerja praktek;
8. dan semua pihak yang membantu, yang tidak dapat penulis sebutkan.

Penulis menyadari bahwa kerja praktek ini bukanlah tanpa kelemahan, untuk itu kritik dan saran sangat diharapkan.

Akhir kata, semoga laporan kerja praktek ini dapat bermanfaat bagi para pembacanya.

Bandung, Agustus 2018

Penulis

DAFTAR ISI

ABSTRAK	iii
ABSTRACT	v
KATA PENGANTAR	viii
DAFTAR ISI	ix
DAFTAR SINGKATAN	x
BAB I PENDAHULUAN	1
I.1 Latar Belakang Masalah	1
I.2 Tujuan	2
I.3 Batasan Masalah	2
I.4 Metodologi Percobaan	3
I.5 Sistematika Pembahasan	7
BAB II LANDASAN TEORI	9
II.1 Amplitude Modulation	9
II.2 Mikrokontroler	10
II.3 Oximeter	10
BAB III PENGEMBANGAN ANTARMUKA APLIKASI ANDROID	12
III.1 Perekaman Sinyal	13
III.2 Fungsionalitas Tambahan	18
III.3 Grafik Kustom	23
III.4 Sistem Filtrasi	28
BAB IV PENDESAINAN DAN IMPLEMENTASI SISTEM TRANSMISI	32
IV.1 Pulse Oximeter	33
IV.2 Arduino	36
IV.3 Android	53
BAB V KESIMPULAN DAN SARAN	57
V.1 Kesimpulan	57
V.2 Saran	57
DAFTAR PUSTAKA	59
LAMPIRAN A	A-1
A.1 Program utama: Oximeter.ino	A-1
A.2 Program filter: Filter.cpp	A-5
A.3 Header filter: Filter.h	A-7
LAMPIRAN B	B-1
B.1 Koefisien filter-filter pada program utama: MainActivity.java	B-1
B.2 Mode-mode grafik kustom: SimpleWaveform.java	B-3
LAMPIRAN C	C-1

DAFTAR SINGKATAN

SINGKATAN	Nama	Pemakaian pertama kali pada halaman
GUI	Graphical User Interface	iv
STIKES	Sekolah Tinggi Ilmu Kesehatan	viii
RS	Rumah Sakit	viii
RSU	Rumah Sakit Umum	viii
IPS	Instalasi Pemeliharaan Sarana	viii
BPM	Beats per Minute	2
USB	Universal Serial Bus	5
CPU	Central Processing Unit	10
SDK	Standard Development Kit	12
DAC	Digital to Analog Converter	13
PC	Personal Computer	13
OTG	On-The-Go	32
HPF	High-Pass Filter	32
ADC	Analog to Digital Converter	33
SPICE	Simulation Program with Integrated Circuit Emphasis	33
BJT	Bipolar Junction Transistor	33
DC	Direct Current	41
LPF	Low-Pass Filter	45

BAB I

PENDAHULUAN

I.1 Latar Belakang Masalah

Dengan perkembangan teknologi yang sangat pesat pada era modern ini, tentunya seluruh sektor kehidupan juga ikut terpengaruhi secara drastis. Kesehatan, bisnis, ekonomi, keamanan, entertainment, dll sehingga memilih untuk tidak berkembang saja sudah termasuk kerugian yang besar. Dalam sektor kesehatan sendiri, teknologi memiliki predominasi sebagai hal yang dapat menjadi kekuatan sekaligus kekurangan. Teknologi sangatlah penting dalam hal kesehatan karena memang ada banyak sekali kegiatan medis yang hanya bisa dilakukan oleh suatu alat teknologi. Ketergantungan yang tinggi, standar yang ketat, dan persaingan yang tidak seberapa besar dalam teknologi medis pada akhirnya mengakibatkan peningkatan harga yang sangat tinggi.

Secara general, memanfaatkan peralatan keseharian yang sudah ada adalah teknik terbaik dalam meningkatkan hampir segala hal. Contoh yang paling umum adalah memanfaatkan keberadaan smartphone, alat dengan kemampuan yang sangat luas namun dimiliki oleh hampir semua kalangan masyarakat. Namun hal yang paling mendasar sebagai fungsi smartphone tidak lain adalah sebagai interface, perantara antara manusia dengan suatu alat sederhana. Dalam sektor medis, keperluan layar/interface dan perangkat komputasi bisa dibilang memerlukan waktu pembuatan yang cukup lama ataupun biaya yang tinggi. Dengan memanfaatkan platform yang sudah sangat umum seperti iOS maupun Android, tentunya proses development suatu interface yang jauh lebih indah dengan fungsionalitas tinggi dapat dilakukan dengan mudah, terlebih lagi menggunakan perangkat yang dimiliki oleh siapa pun.

Melalui pemanfaatan smartphone dengan platform Android, suatu instrumentasi yang sangat sederhana dapat dibuat. Kapabilitas komputasi instrumentasi tidak perlu besar dalam kasus ini, karena kebanyakan proses dilakukan oleh smartphone. Sedangkan salah satu peralatan medis yang sangat mendasar pemanfaatannya, namun cukup sederhana untuk diterapkan adalah Oximeter. Alat tersebut dapat mengukur

perubahan volume darah (menggambarkan detak jantung) maupun saturasi oksigen di darah. Smartphone dalam kasus ini tentunya bisa digunakan untuk menampilkan sinyal perubahan volume, merekam dan menyimpan data yang diperoleh, menghitung persentase oksigen, serta menghitung kecepatan detak jantung.

I.2 Tujuan

Dilaksanakannya kerja praktek serta proyek ini memiliki berbagai tujuan utama, yaitu:

- Memenuhi syarat kelulusan jurusan S1 Teknik Elektro.
- Mengembangkan peralatan yang bisa bermanfaat untuk dunia medis.
- Mengembangkan peralatan yang cocok untuk digunakan secara umum.
- Menambah keilmuan teknis.

I.3 Batasan Masalah

Sifat yang umum dari proyek beserta banyaknya perkembangan yang bisa dilakukan membuat diperlukan adanya batasan-batasan masalah yang jelas mengenai apa yang dibuat, diselesaikan, maupun dianalisis dalam proyek ini. Adapun batasan-batasan masalah sebagai berikut:

1. Pengujian intensif hanya dilakukan pada smartphone Samsung Galaxy E5 dan Samsung Galaxy Grand Neo.
2. Oximeter yang digunakan masih hanya bisa mengambil data HbO_2 atau hemoglobin kaya oksigen, sehingga kalkulasi SpO_2 belum bisa dilakukan.
3. Perhitungan BPM hanya didasarkan pada perhitungan jarak puncak antara dua gelombang.

4. Transmisi data dari oximeter ke smartphone memerlukan audio jack, yang mana disupport oleh banyak smartphone namun memerlukan konektor khusus untuk sebagian kecil lainnya.
5. Metode penyimpanan data pada aplikasi Android dilakukan dengan mengkopi nilai mentah grafik, sehingga perlu *dipaste*, atau bisa juga bergantung pada ada tidaknya kapabilitas screenshot smartphone.

I.4 Metodologi Percobaan

Selama pengembangan proyek ini, ada banyak sekali skema percobaan yang dilakukan untuk menguji karakteristik dari produk yang dibuat. Namun secara garis besar, percobaan-percobaan tersebut terbagi menjadi tiga jenis: pengujian aplikasi Android dengan input buatan dari komputer, pengujian output oximeter melalui komputer, serta pengujian total.

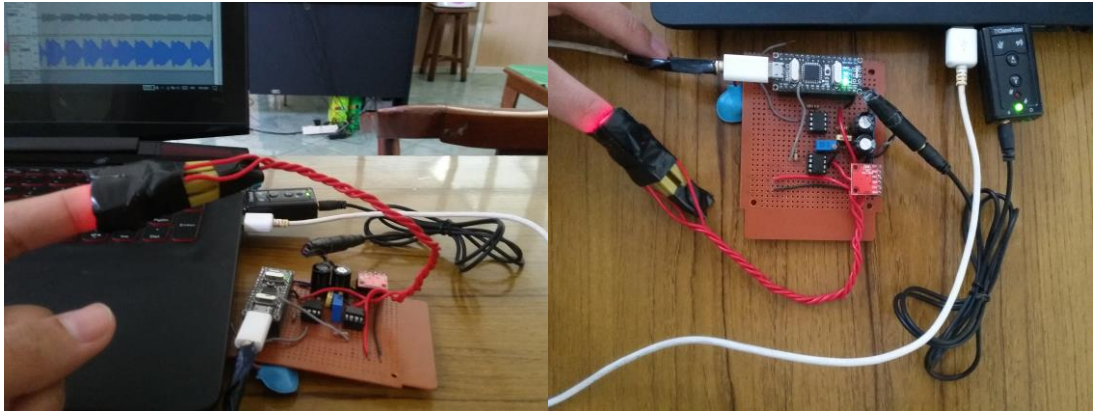
I.4.1 Rancangan

I.4.1.1 Pengujian Aplikasi Android dengan Input Buatan dari Komputer



Gambar I-1 Rancangan dalam pengujian aplikasi Android dengan input buatan dari komputer, dengan program "Audacity" sebagai tempat sintesis sinyal input.

I.4.1.2 Pengujian Output Oximeter melalui Komputer



Gambar I-2 Rancangan dalam pengujian output oximeter melalui komputer, digunakan program “Audacity” sebagai tempat perekaman dan pengujian sinyal output serta program “SerialPlot” untuk melihat data digital oximeter.

I.4.1.3 Pengujian Total



Gambar I-3 Rancangan dalam pengujian total, digunakan program “SerialPlot” untuk melihat data digital oximeter.

I.4.2 Alat dan Bahan

I.4.2.1 Alat

- Oximeter buatan

- Smartphone (Samsung E5 & Samsung Grand Neo)
- Kabel micro USB x 2
- Kabel audio 3.5mm tiga pin
- Kabel splitter audio 3.5mm empat pin ke tiga pin
- Komputer
- USB sound card

I.4.2.2 Bahan

- Selotip
- Baterai

I.4.3 Variabel

- a. Variabel kontrol : Input (jari atau sinyal Audacity)
- b. Variabel manipulasi : Algoritma (Android atau Oximeter)
- c. Variabel respons : Output (Grafik aplikasi Android, rekaman Audacity, atau output serial)

I.4.4 Prosedur

I.4.4.1 Pengujian Aplikasi Android dengan Input Buatan dari Komputer

1. Menyiapkan peralatan sesuai dengan rancangan pada Gambar I-1, koneksi output audio jack komputer atau sound card tersambung dengan koneksi input audio jack smartphone.
2. Menjalankan program Audacity dan mengatur agar pilihan output sudah cocok.
3. Menyiapkan suatu sinyal untuk diujikan.

4. Membuka aplikasi Android buatan.
5. Menjalankan Audacity dan melihat responsnya pada aplikasi Android buatan.
6. Menyimpan data-data yang diperoleh dan menganalisis hasil.
7. Memodifikasi algoritma aplikasi Android dan mengulang dari prosedur 4.

I.4.4.2 Pengujian Output Oximeter melalui Komputer

1. Menyiapkan peralatan sesuai dengan rancangan pada Gambar I-2, koneksi input audio jack komputer atau sound card tersambung dengan koneksi output oximeter buatan serta kabel micro USB menyambungkan Arduino dengan komputer.
2. Menjalankan program Audacity dan mengatur agar pilihan input sudah cocok.
3. Menjalankan program SerialPlot dan mengatur sumber input serta baud rate.
4. Mulai pengambilan data pada program Audacity dan SerialPlot.
5. Memasukkan jari pada sensor oximeter dan mengatur posisinya agar terlihat sinyal yang baik pada Audacity atau SerialPlot.
6. Menghentikan perekaman Audacity.
7. Menyimpan data-data yang diperoleh dan menganalisis hasil.
8. Memodifikasi algoritma Arduino dan mengulang prosedur 4.

I.4.4.3 Pengujian Total

1. Menyiapkan peralatan sesuai dengan rancangan pada Gambar I-3, output oximeter tersambung dengan koneksi input audio jack smartphone serta kabel micro USB menyambungkan Arduino dengan komputer.

2. Menjalankan program SerialPlot dan mengatur sumber input serta baud rate.
3. Membuka aplikasi Android buatan.
4. Memasukkan jari pada sensor oximeter dan mengatur posisinya agar terlihat sinyal yang baik pada aplikasi Android, SerialPlot atau aplikasi Android.
5. Menyimpan data-data yang diperoleh dan menganalisis hasil.
6. Memodifikasi algoritma aplikasi Android atau Arduino dan mengulang dari prosedur 4.

I.5 Sistematikan Pembahasan

Laporan kerja praktek ini dibagi atas 5 bab. Bab pertama merupakan pendahuluan, yang mana menguraikan latar belakang penelitian, permasalahan-permasalahan, tujuan, batasan-batasan yang harus diketahui, metodologi, serta diakhiri dengan sistematika pembahasan. Seluruh aspek-aspek tersebut ditujukan untuk memberikan konteks singkat mengenai keseluruhan isi laporan.

Pada bab selanjutnya, bab kedua, terdapat landasan teori yang berisikan seprangkat definisi, konsep, alur berpikir, serta proposisi tentang berbagai konsep berkaitan erat dengan pembahasan laporan. Konsep-konsep tersebut umumnya merupakan topik yang perlu diketahui agar bisa memahami secara menyeluruh apa yang ada di dalam laporan ini.

Dua bab selanjutnya merupakan pembahasan utama tentang proyek yang dilaksanakan. Bab ketiga membahas keseluruhan aspek antarmuka aplikasi Android. Hal-hal mengenai desain, menu-menu, grafik, dll diuraikan berdasarkan tingkat keutamaannya.

Selanjutnya bab keempat membahas sistem transmisi keseluruhan alat. Di sini dipenuhi dengan pembahasan teknis dalam mengirimkan data dari Oximeter ke suatu

smartphone. Permasalahan-permasalahan terbesar, cara menuntaskan, trade-off, dll dari berbagai alternatif teknis akan dibahas dan dibandingkan.

Laporan diakhiri dengan bab kelima, berisikan kesimpulan dan saran. Elaborasi terhadap berbagai kesimpulan hasil pelaksanaan proyek yang bisa ditarik dari pembahasan dirincikan di sini. Selain itu, bab ini mengandung saran untuk kajian lebih lanjut maupun implikasi praktis yang bisa ditarik setelah melaksanakan kerja praktek ini.

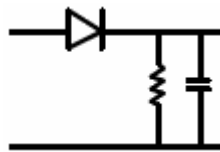
BAB II

LANDASAN TEORI

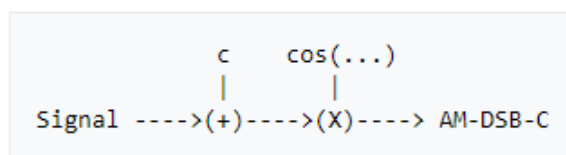
II.1 Amplitude Modulation

Modulasi adalah proses mencampurkan suatu sinyal dengan sinusoid agar diperoleh sinyal baru, umumnya sering digunakan dalam kasus suatu data tidak bisa ditransmisikan secara langsung [1]. Salah satu karakteristik dari modulasi adalah suatu sinyal pesan dapat digeser ke atas frekuensinya, sehingga apabila memiliki rentang frekuensi masing-masing, dapat mengakibatkan lebih dari satu sinyal dikirimkan bersamaan dalam satu saluran [1]. Sebagaimana yang sering ditemui dalam bentuk tulisan “AM” pada konfigurasi radio, amplitude modulation adalah salah satu teknik yang paling awal dalam modulasi radio [2].

Meski ada beragam metode dalam mengimplementasikan AM, metode dengan teknik modulasi dan demodulasi (rekonstruksi) yang paling sederhana adalah AM-DSB-C atau DSBAM; di mana hanya dibutuhkan perkalian sederhana untuk memodulasi dan suatu envelope detector untuk rekonstruksi [2]. Skema untuk kedua hal tersebut berturut-turut ditunjukkan pada Gambar II-1 dan Gambar II-2 [2].



Gambar II-1 Envelope Detector [2].



Gambar II-2 Skema transmitter DSBAM [2].

II.2 Mikrokontroller

Translasi dari artikel Basics of Microcontrollers pada website Circuits Today menjelaskan: “Mikrokontroller adalah mikrokomputer yang dibuat dalam satu buah chip melalui fabrikasi VLSI. Suatu mikrokontroller dapat juga disebut sebagai kontroller *embedded* karena mikrokontroller dan rangkaian penyokongnya umumnya dibuat secara tertanam di peralatan yang dikendalikan” [3].

Selain itu, mikrokontroller umumnya memiliki satu atau lebih komponen di bawah ini [3].

- Central processing unit (CPU)
- Random Access Memory (RAM)
- Read Only Memory (ROM)
- Input/output ports
- Timers and counters
- Interrupt controls
- Analog to digital converters
- Digital to analog converters
- Serial interfacing ports
- Oscillatory circuits

Suatu mikrokontroller pada dasarnya memiliki fitur-fitur dasar untuk melakukan komputasi sederhana layaknya suatu komputer dengan pin-pin yang bisa diprogram karakteristiknya oleh pengguna, proses desain yang mudah, harga yang murah, ukuran yang kecil, dll [3].



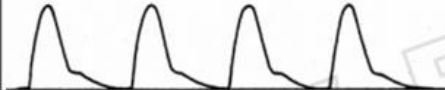



II.3 Oximeter

Secara singkat, pulse oximeter merupakan alat non-*invasive* yang dapat mengukur seberapa banyak hemoglobin di dalam darah yang sedang membawa oksigen (saturasi

oksigen) [4]. Sedangkan parameter-parameter hasil pulse oximeter dapat digunakan untuk mendiagnosis kondisi kesehatan seseorang, sebagai contoh ditunjukkan dalam Tabel II-1 dan Tabel II-2 [5].

Casualty	SpO ₂
Normal – Healthy	≥ 95%
Normal - COPD	88% – 92%
Hypoxic	85 – 94%
Severely Hypoxic	< 85%
Trigger: Any drop of 3% or more indicates cause for concern	

Tabel II-1 Nilai parameter SpO₂ beserta penyebabnya [5].

Waveform	Pulse type	Physiological causes	Possible disease
	Normal	-	-
	Small and weak	Decreased stroke volume Increased peripheral resistance	Heart failure, hypovolaemia
	Large and bounding	Increased stroke volume Decreased peripheral resistance.	Fever, anaemia, aortic regurgitation, traumatic brain injury
	Pulsus alternans	Pulse amplitudes varies	Left ventricular failure
	No dicrotic notch	Increased arterial resistance	Coronary heart diseases
	Chaotic	Arrhythmia Motion artefact	Ventricular Tachycardia, Ventricular Fibrillation, background movement.

Tabel II-2 Berbagai karakteristik gelombang pulse oximeter beserta penyebab fisiologis dan kemungkinan penyakitnya [5].

BAB III

PENGEMBANGAN ANTARMUKA

APLIKASI ANDROID

Aplikasi Android dikembangkan menggunakan aplikasi Android Studio, dengan bahasa pemrograman Java. Melalui SDK yang disediakan, berbagai macam fungsionalitas dasar hingga fungsionalitas lanjut dari suatu smartphone Android dapat dipergunakan tanpa terlalu banyak memikirkan aspek yang mendalam. Meski demikian, hal teknis tetaplah dominan dalam proses development aplikasi. Namun bisa dibayangkan bahwa pendesainan GUI menjadi jauh lebih solid dengan mengikuti standar yang sudah ditetapkan oleh Developer-developer Android lainnya.

Pengembangan aplikasi secara garis besar didasarkan pada beberapa komponen utama: fungsionalitas utama, fungsionalitas & pengaturan tambahan, grafik kustom, diakhiri dengan sistem penerimaan serta filtrasi data termodulasi. Fungsionalitas utama terdiri atas metode pengambilan/perekaman data suara (analog) serta metode mengalirkan data dari suatu sub-sistem digital ke sub-sistem lainnya. Fungsionalitas & pengaturan tambahan secara umum berisikan metode untuk mengopi data mentah grafik, resume atau pause pengambilan data, serta konfigurasi yang sangat lengkap (lebar grafik, downsampling, tampilkan atau hilangkan suatu elemen GUI, dll). Lalu grafik kustom merupakan metode penampilan sinyal yang dapat memenuhi berbagai kriteria desain yang diinginkan. Diakhiri dengan serentetan sistem yang digunakan untuk demodulasi data, filtering, serta kalkulasi-kalkulasi teknis lainnya yang diperuntukkan sebagai penyajian data. Bagian akhir inilah yang memerlukan ruang komputasi yang cukup signifikan.

Dikarenakan source code yang sangat panjang dan terpisah-pisah, source code tidak akan sering disisipkan bersamaan dengan konten laporan. Bagian-bagian utama dari source code untuk Android akan tersedia dalam LAMPIRAN A, sedangkan secara keseluruhan tersedia pada repository berikut [6].

III.1 Perekaman Sinyal

Metode pengiriman data secara analog memanfaatkan DAC yang telah terpasang di dalam setiap audio jack perangkat-perangkat pintar (PC, smarphone, speaker digital, dll). Pertimbangan dipilihnya metode tersebut akan dibahas di bab tentang sistem transmisi. DAC adalah suatu sistem yang dapat mengonversi data analog (tegangan dengan nilai kontinu) ke dalam rana digital (biner atau tegangan dengan dua nilai saja) sehingga bisa dipahami oleh prosesor digital – perangkat yang berperan sebagai pengatur maupun pengolah seluruh data digital [7].

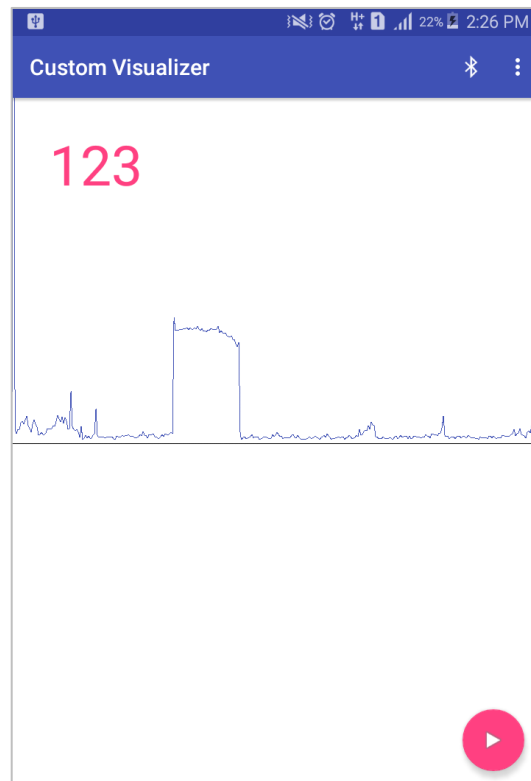
Pengambilan data dari audio jack memanfaatkan karakteristik standar yang dimiliki oleh perangkat-perangkat digital; yaitu apabila ada kabel yang terkoneksi dengan audio jack, maka I/O suara yang digunakan bukan lagi dari perangkat built-in seperti mic ataupun sound speaker internal melainkan melalui audio jack tadi. Koneksi input audio jack yang biasanya tersambung dengan microphone external (seperti pada headset) kali ini akan disambungkan langsung dengan suatu generator sinyal, sehingga DAC bisa membaca nilai tegangan yang telah diberikan. Alhasil penerimaan data oleh handphone bisa dilakukan layaknya merekam suara pada umumnya.

Implementasi perekaman suara tentu merupakan hal yang sangat standar pada suatu handphone. Namun proses yang sangat mendasar tersebut sebenarnya perlu memikirkan cukup banyak faktor teknis ketika diimplementasikan. Beberapa faktor teknis tersebut antara lain:

- Spesifikasi perekaman
- Kompatibilitas perangkat terhadap spek yang diinginkan
- Ukuran buffer
- Mekanisme pengolahan buffer

Dalam melakukan perekaman sinyal, Android menyediakan berbagai library yang bisa digunakan. Pengujian awal dilakukan dengan menggunakan library *MediaRecorder*. Hal tersebut dilakukan demi mengkonfirmasi tentang karakteristik DAC yang sudah dijelaskan sebelumnya. Skema pengujian adalah kabel audio jack male-male disambungkan dari audio jack PC ke dalam audio jack handphone, sehingga

sinyal buatan bisa diinputkan secara langsung kepada handphone. Dalam menyiapkan sinyal, program yang digunakan adalah Audacity. Pengujian memberikan hasil sebagai berikut untuk sinyal sinusoidal beberapa saat.



Gambar III-1 Hasil perekaman sinyal menggunakan library MediaRecorder.

Gambar III-1 menunjukkan bahwa sinyal tidak pernah masuk ke daerah negatif, bahkan ketika diberikan input sinusoidal (daerah yang paling mencolok). Hal ini dikarenakan karakteristik dasar library tersebut adalah mengambil amplitudo maksimum dari deretan sampel, bukan data mentah [8]. Sehingga masuk akal apabila sinyal sinusoidal akan memberikan sinyal kotak, karena amplitudonya pada waktu itu akan konsisten. Dengan demikian, bisa dikatakan bahwa memang benar bahwa pengambilan data analog secara langsung bisa dilakukan.

Karena sifat yang tidak diinginkan, implementasi perekaman suara diganti dari library *MediaRecorder* menjadi library *AudioRecord*. Dalam library tersebut, data dari speaker internal atau audio jack (tergantung tersambung atau tidaknya audio jack) akan diambil secara mentah-mentah. Spesifikasi perekaman utama yang digunakan adalah sebagai berikut.

```
private static final int REC_RATE = 44100;
private static final int REC_CH = AudioFormat.CHANNEL_IN_MONO;
private static final int REC_AUDIO_ENC = AudioFormat.ENCODING_PCM_16BIT;
```

Algoritma III-1 Spesifikasi utama recorder.

Secara singkat data-data tersebut berarti perekaman dapat mengambil 44.100 data sinyal dalam satu detik, dimana dalam satu waktu hanya diambil satu suara, dengan ukuran satu buah datanya 2 bytes (menentukan resolusi data). Sedangkan penanganan pengambilan data buffer perekaman, pengolahan, serta update data-data GUI ditangani oleh potongan program di bawah ini.

```
// Default configuration which guaranteed to work by the docs (though
// actually not for low-ends)
recorder = new AudioRecord(MediaRecorder.AudioSource.MIC,
    REC_RATE, REC_CH,
    REC_AUDIO_ENC, BufferElements2Rec * BytesPerElement);

// Not compatible, decided to seek different configs
if (recorder.getState() == AudioRecord.STATE_UNINITIALIZED) {
    recorder = findAudioRecord();
}

recorder.startRecording();
recordingThread = new Thread(new Runnable() {
    @Override
    public void run() {
        while (doRun) {
            try {
                Thread.sleep(1);
            } catch (Exception e) {
                // Thread waking up earlier due to an interrupt and able
                // to be relocated
            }
        }
    }
});
```

```

        recorder.read(sData, 0, BufferElements2Rec);

        //... Data processing

        // New, separate, UI Thread
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                //... GUI management
            }
        });
    }
}, "AudioRecorder Thread");
recordingThread.start();

```

Algoritma III-2 Potongan program yang menangani pengambilan data rekaman.

Algoritma III-2 menunjukkan beberapa hal penting. Pada bagian awal, program akan mencoba untuk membuat metode perekaman berdasarkan konfigurasi utama yang dispesifikasikan pada Algoritma III-1. Meskipun dokumentasi dari web Android Developers mengatakan apabila sampling rate 44,1kHz merupakan sampling rate yang dijamin berkerja untuk semua perangkat Android, ada beberapa sumber yang mengatakan bahwa handphone low-end (kapabilitasnya cenderung rendah) tidak mengikuti aturan tersebut [9]. Sehingga perlu adanya mekanisme untuk menguji satu-per-satu konfigurasi perekaman apabila kondisi utama tidak bisa dipenuhi. Pengujian tersebut dilakukan oleh metode *findAudioRecord()*, dimana pengujian akan dimulai dari konfigurasi yang paling diinginkan ke arah konfigurasi yang paling tidak diinginkan (semakin jauh dari konfigurasi utama).

Hal penting lainnya adalah diterapkannya pemrosesan paralel melalui *threading*. Dengan demikian, bisa lebih dari satu deretan program berjalan dalam satu waktu. Alhasil akan diperoleh proses yang tidak saling tunggu (polling). Fitur ini memanfaatkan fakta bahwa hampir semua handphone sekarang memiliki CPU dengan core ataupun thread lebih dari satu. Dampak dari digunakannya metode ini adalah pembuatan/pengimplementasian algoritma akan cenderung lebih rumit, sebab algoritma tersebut harus *thread-safe* – algoritma berjalan dengan benar ketika diakses banyak thread. Sebagai contoh dalam pemrosesan paralel, suatu variabel bisa saja terakses ataupun termodifikasi oleh lebih dari satu proses secara bersamaan.

Hasil dari perubahan-perubahan tersebut akan memberikan gambar di bawah ini. Input yang diberikan seperti dengan sebelumnya, sinusoidal dengan amplitudo konstan.



Gambar III-2 Hasil perekaman sinyal menggunakan library AudioRecord.

Bisa dilihat bahwa pergerakan naik turun sinyal sudah sangat terlihat. Perlu diperhatikan bahwa ada sedikit bug yang mengakibatkan data tidak kontinu saat melewati titik nol, sehingga grafik pada bagian atas tidak tersambung dengan baik dengan grafik bagian bawah. Pada proses pembaruan lebih lanjut hal ini sudah terselesaikan bersamaan dengan penambahan fitur lain. Dengan demikian, landasan paling utama program ini telah selesai.

Karena sifat buffer yang berbentuk array, proses pengiriman data juga tidak bisa asal. Ada beberapa bentukan data yang dipertimbangkan untuk saling memberikan antara sistem yang satu ke sistem lainnya, diantaranya: array secara langsung, linked list, dan satu-per-satu data. Opsi penggunaan linked list menunjukkan waktu yang secara drastis lebih lama, meskipun sudah digunakan akses data secara iterasi (sehingga waktu pengaksesan serupa dengan array biasa). Sedangkan kelebihan dari linked list utamanya adalah struktur program yang jauh lebih indah dan mudah dibaca, biaya

yang mudah untuk melakukan penggeseran data (shifting), dll. Dalam kasus array, proses manipulasi secara total jauh lebih berantakan. Terlebih lagi karena setiap filter harus memiliki buffer masing-masing dengan ukuran yang berbeda. Terakhir pengiriman data-satu-per-satu akan sangat rumit, pasalnya data awal – buffer perekaman – berbentuk array. Satu buah sistem/filter harus memproses keseluruhan buffer sekaligus dan menyimpannya ke suatu tempat, yang berarti hasil akhir bukanlah kesatuan data. Dengan demikian, akan digunakan array.

Proses pengaliran data sangatlah sederhana, suatu array akan dibaca satu-per-satu dan diproses satu-per-satu pula. Tergantung spesifikasi, apabila ada perintah downsampling maka akan ada beberapa data yang dilompati. Untuk sistem yang mengeluarkan data, selesainya pemrosesan satu data akan dimasukkan ke suatu data buffer. Alhasil pada umumnya diperoleh algoritma sebagai berikut.

```
public void addWaveArray(double[] arr, SimpleWaveform simpleWaveform,
int downSample) {
    int arrSize = arr.length;

    for (int i = 0; i < arrSize; i++) {
        if (i % downSample == 0) {
            //... Process the singular data
        }
    }
}
```

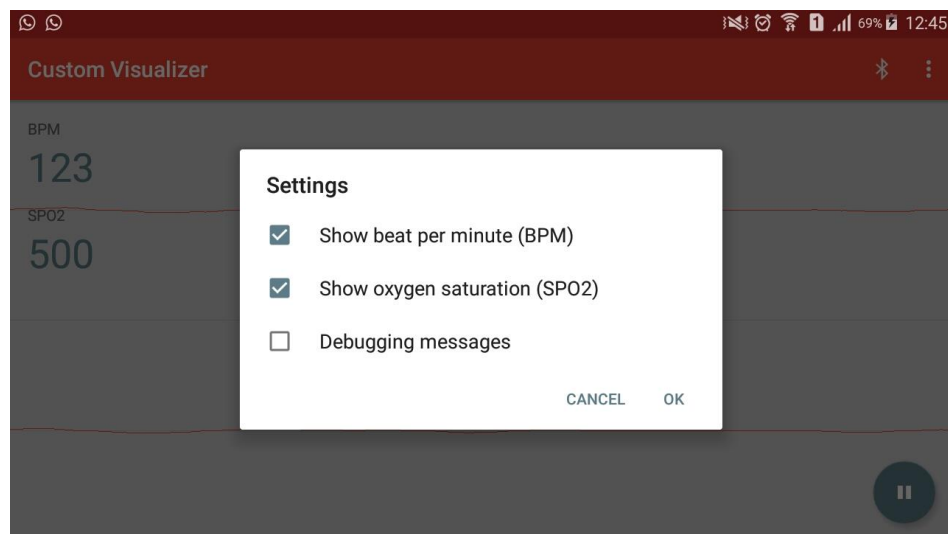
Algoritma III-3 Bentuk umum algoritma pemrosesan data.

III.2 Fungsionalitas Tambahan

Sebagaimana telah dijelaskan sedikit pada bagian awal, fungsionalitas-fungsionalitas ini adalah keseluruhan fungsionalitas yang dapat meningkatkan kemampuan pengguna untuk mengatur lebih lanjut aplikasi ini, pemanfaatan lebih baik fitur-fitur perangkat sebagai suatu smartphone, GUI untuk masing-masing fungsionalitas tersebut, debugging messages, dll. Karena signifikansinya yang tidak seberapa tinggi, penjelasan untuk bagian ini tidak terlalu banyak. Meski demikian, bisa dikatakan

bahwa hal-hal yang termasuk dalam kategori ini memakan porsi pengembangan aplikasi yang cukup signifikan.

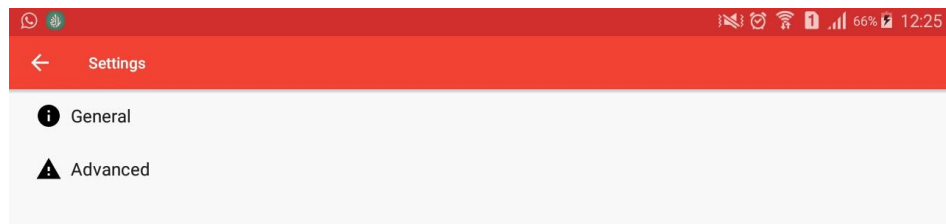
Kebanyakan dari fitur tambahan termotivasi dari keperluan debugging. Hal pertama yang diimplementasikan pada kelompok fungsionalitas ini adalah debugging messages, atau teks sederhana tambahan yang terpasang langsung pada aplikasi. Fungsi tersebut sangat bermanfaat apabila ingin mengecek suatu data yang diperlukan tanpa terkoneksi ke komputer. Untuk fungsi ini, suatu menu *settings* (pengaturan) sederhana dibuat sebagaimana terlihat pada Gambar III-3. Menu tersebut diimplementasi hanya berdasarkan suatu list checkbox yang otomatis tercipta sendiri (tanpa perlu mengatur layout secara manual). Cukup dengan menyiapkan array, respon, dan beberapa teks, suatu opsi tambahan bisa diberikan. Tentunya bisa terlihat bahwa konfigurasi tersebut tidak bisa mengandung pengaturan yang lebih kompleks.



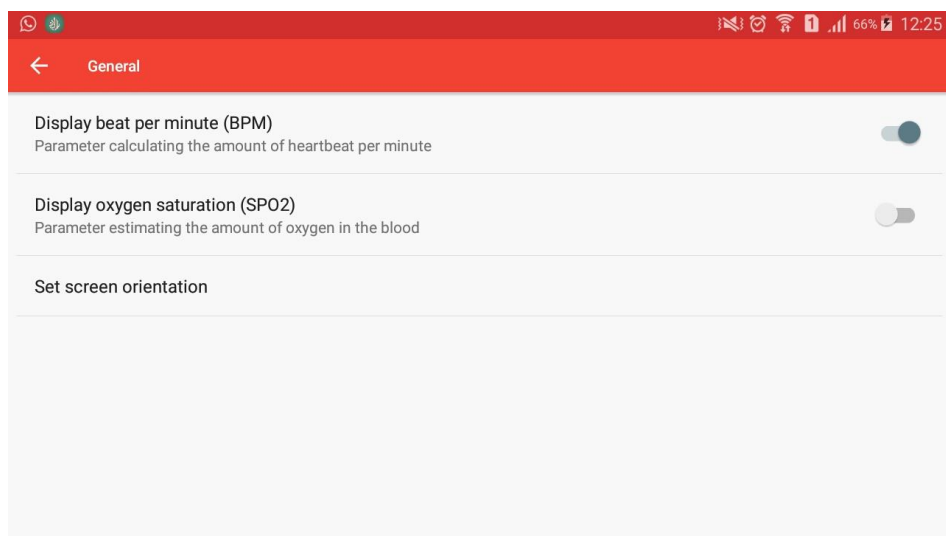
Gambar III-3 Tampilan utama Settings pada tahapan awal

Kebutuhan untuk mengatur suatu variabel secara langsung tanpa perlu kembali ke IDE, melakukan proses building, uploading, etc membuat metode settings diubah secara total. Kali ini digunakan template universal yang sudah disediakan & dicontohkan oleh Android Studio, yaitu *Settings Activity*. Dalam template tersebut,

berbagai macam jenis pengaturan bisa ditambahkan seperti multiple choice, checkbox, slider, text input, dll. Pengimplementasian ini dapat memberikan pengaturan yang berlapis sebagaimana ditunjukkan pada Gambar III-4, Gambar III-5, dan Gambar III-6.

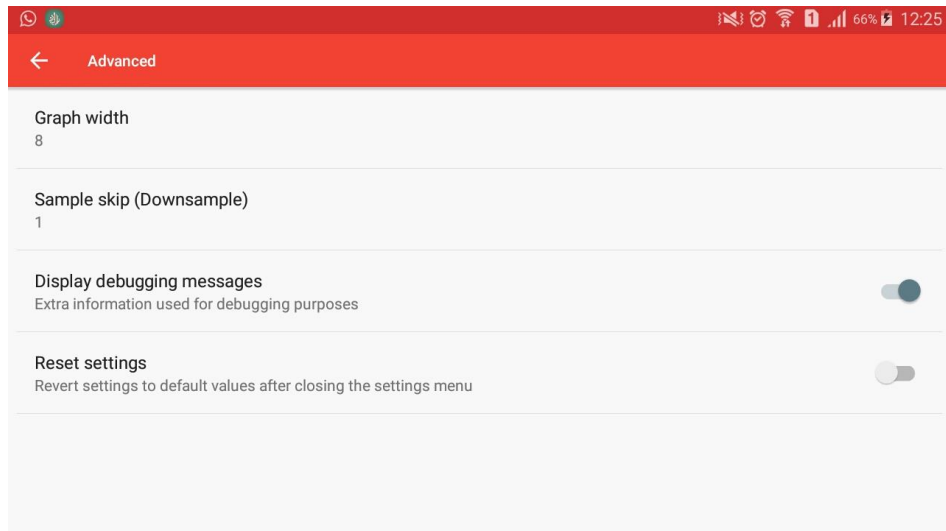


Gambar III-4 Sebagian tampilan utama Settings.



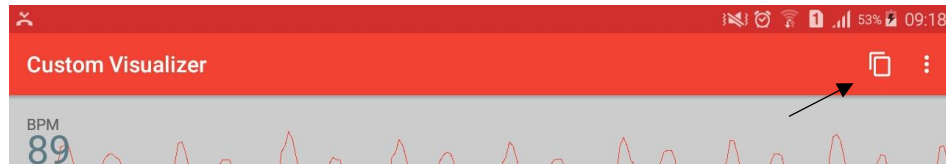
Gambar III-5 Tampilan General Settings, berisikan pengaturan sederhana.

Penjelasan tentang sebagian opsi ada di bawah masing-masing opsi tersebut. Untuk downsampling, sama dengan fungsi umum yang sudah dijelaskan sebelumnya: mengambil hanya sampel data dari x sampel yang diterima, sehingga memperkecil grafik lebih dari batasan terkecil atau menyederhanakan grafik. Sedangkan Graph Width memperlebar jarak antar sampel, sehingga memperlebar grafik secara menyeluruh.



Gambar III-6 Tampilan Advanced Settings, berisikan pengaturan yang lebih kompleks

Fitur lainnya yang sangat bermanfaat untuk pengumpulan data adalah fitur penyalinan data mentah. Seperti yang diketahui secara umum, penggunaan smartphone sebagai antarmuka memiliki manfaat berupa fitur2 mendasar tersedia langsung seperti screenshot. Meski jauh lebih mudah, data berbentuk screenshot suatu grafik tidak bisa secara langsung disimpan secara tepat, ataupun diolah & dianalisis lebih lanjut. Untuk itulah opsi menyalin data grafik secara mentah akan menjadi tambahan yang cukup penting, juga sekaligus menjadi pemanfaatan fitur integral suatu perangkat pintar. Selain data-data tersebut bisa disimpan dalam suatu database – memungkinkan diciptakannya sistem analisis yang lebih cerdas – proses debugging, statistik sederhana, dll juga bisa menjadi salah satu pemanfaatan mendasar. Sedangkan metode penyalinan data serupa dengan menyalin suatu teks. Cukup dengan menekan ikon kotak yang ada di pojok kanan atas seperti pada Gambar III-7, data akan tersalin dan bisa diperoleh kembali ketika pengguna melakukan *paste* layaknya proses penyalinan teks pada umumnya. Dua buah deretan angka yang dipisahkan dengan kurung kotak akan muncul, merepresentasikan data dari grafik atas dan bawah berturut-turut.



Gambar III-7 Ikon untuk menyalin data grafik, ditunjuk dengan panah hitam.

Sistematika dibalik fitur copy/paste pada suatu perangkat pintar selalu diatur dengan penyimpanan khusus yang dinamakan *clipboard*. Sehingga hal yang perlu dilakukan untuk melakukan copy adalah menulis suatu deretan teks ke dalam penyimpanan khusus tersebut. Algoritma yang digunakan untuk menyalin deretan data di dalam array ada pada

```
private void toClipboard(double[] values, boolean append) {
    // Convert to linked list
    LinkedList<Integer> ll = new LinkedList<Integer>();

    int size = values.length;

    for (int i = 0; i < size; i++) {
        ll.add((int) values[i]);
    }

    toClipboard(ll, append);
}
```

Algoritma III-4 Metode penyimpanan array ke dalam clipboard untuk fungsionalitas copy.

Sedangkan komponen antarmuka lainnya adalah tombol pause/resume yang dapat digunakan untuk menghentikan maupun melanjutkan perekaman data. Hal ini bermanfaat apabila grafik yang muncul ingin dilihat lebih seksama tanpa perlu khawatir terganggu lagi karena adanya gerakan atau hal lainnya. Panah hitam pada Gambar III-8 menunjukkan tombol tersebut.



Gambar III-8 Tombol untuk pause/resume, ditunjuk dengan panah hitam.

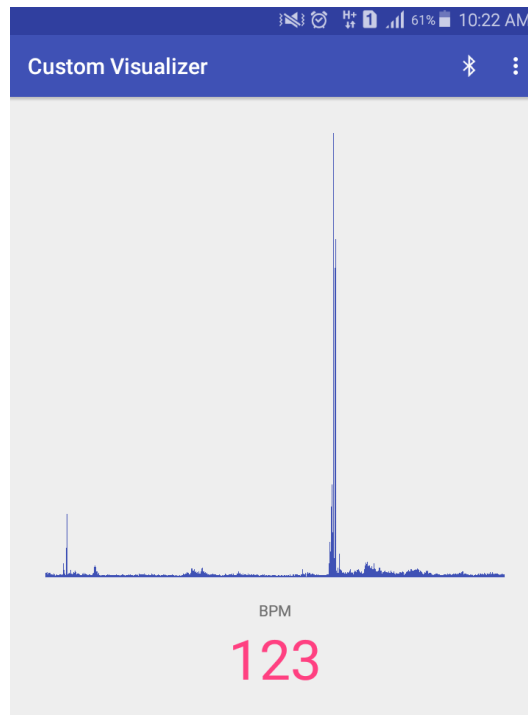
III.3 Grafik Kustom

Dalam subbab III.1, bisa dilihat bahwa ada faktor tambahan yang berperan penting dalam keseluruhan proses. Faktor tersebut adalah grafik – GUI yang berperan dalam proses debugging untuk tahapan awal serta penunjuk gelombang oximeter untuk tahapan final. Gelombang dari oximeter adalah hal yang penting karena dapat memberikan informasi tentang perubahan volume, sehingga ada penamaan khusus berupa *plethysmograph* (atau *pleth*) untuk pulse oximeter yang memiliki kapabilitas tersebut [10]. Bentuk gelombang yang dihasilkan dapat bermanfaat untuk memberikan diagnosa yang lebih mendalam jika dibandingkan dengan data SpO_2 saja. Sayangnya untuk pasaran oximeter portable, penambahan fitur penggambaran pleth dapat meningkatkan harga secara signifikan jika dibandingkan dengan yang tidak memiliki fitur tersebut [11].

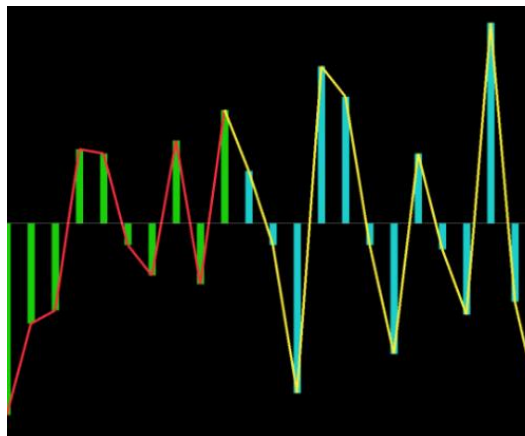
Ada banyak sekali perubahan yang dilakukan ketika memilih template dasar maupun konfigurasinya masing-masing. Pemilihan template dari sumber luar dilakukan karena tidak ada template dasar dengan parameter yang beragam datang dari Android Studio, berbeda dengan komponen GUI lainnya seperti tombol, list, checkbox, dll. Sedangkan grafik pada tahapan awal ditunjukkan oleh Gambar III-9, yang mana menunjukkan hanya daerah positif (karena memang pada saat itu baru diterapkan metode perekaman dengan library MediaRecorder, amplituda max yang diambil).

Pengimplementasian berjalan hingga akhirnya bertemu dengan template dasar yang lebih baik, terletak pada repository *SimpleWaveform* oleh Maxyou [12]. Salah satu dari banyak konfigurasi grafik yang bisa diperoleh dengan menggunakan template tersebut dapat dilihat di Gambar III-10 [12]. Konfigurasi tersebut support terhadap data positif dan negatif, sehingga cocok. Di samping itu juga, ada banyak sekali parameter yang

dapat diubah dengan mudah tanpa banyak mengganti program; satuan metode penggambaran, warna, ada tidaknya bar, jarak antara data, dll.

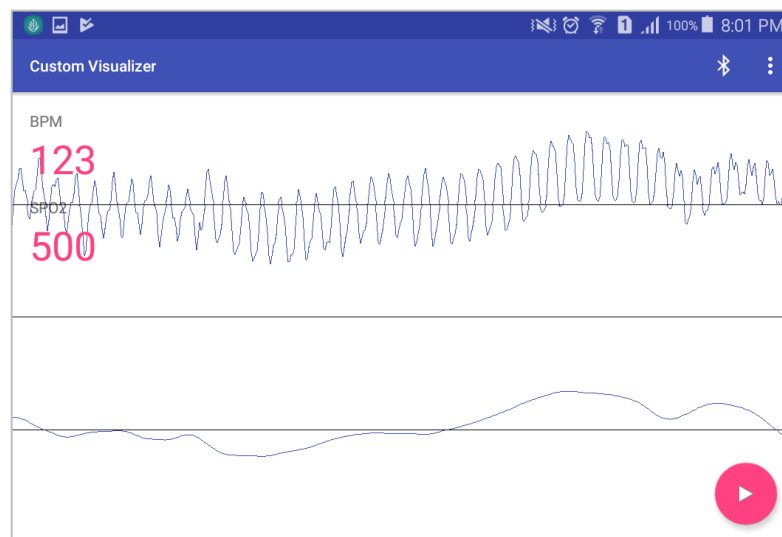


Gambar III-9 Grafik pada fase awal.



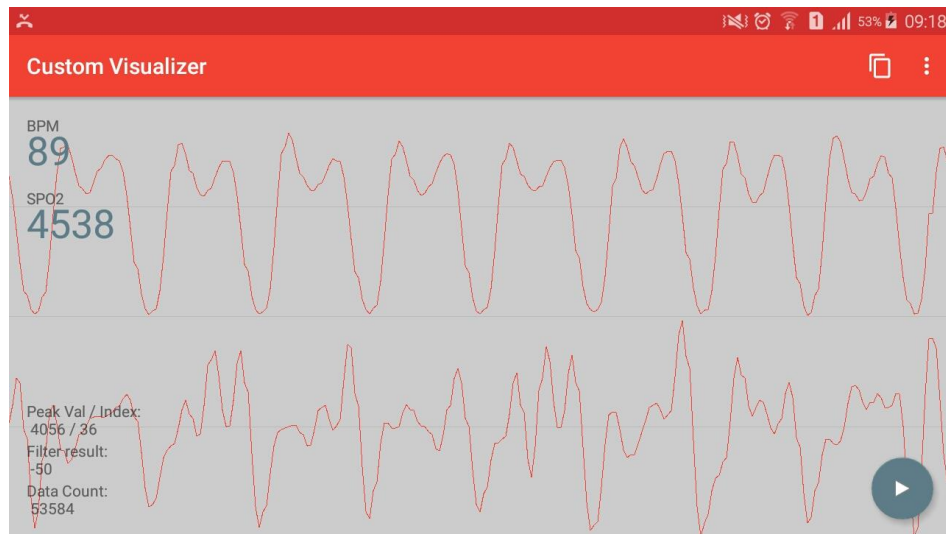
Gambar III-10 Salah satu bentuk grafik yang bisa dibuat menggunakan template SimpleWaveform oleh Maxyou. [12].

Implementasi selanjutnya lebih mengarah pada kemudahan untuk debugging, yaitu menambah satu buah grafik lagi. Hal utama yang dilakukan agar tercapai goal ini adalah mengubah hal-hal bersangkutan dengan grafik agar menjadi lebih modular, umumnya sesederhana mengubah metode agar menerima argumen berupa objek grafik itu tadi. Hasil dari implementasi ini adalah sebagai berikut. Data atas merupakan data mentah sedangkan data di bawahnya merupakan data hasil filter.



Gambar III-11 Hasil perekaman sinyal menggunakan library AudioRecord.

Penambahan berbagai fitur lainnya, perubahan desain, dll memberikan bentukan akhir sebagai berikut. Dalam aspek pengaturan tema, Android Studio memiliki tool bernama Theme Editor yang dapat mengubah keseluruhan hal desain secara konsisten. Di lain aspek, pemilihan warna background yang sedikit lebih gelap terinspirasi dari paradigma pendesainan umum di era modern ini; yaitu untuk memberikan kesan yang lebih lembut dan lebih nyaman untuk dilihat karena kontras (perbedaan) dengan konten tidak tinggi. Tentunya berbeda dengan era lama, yang mana warna yang bisa dimunculkan suatu alat seperti TV, PC, dll sangat terbatas. Sedangkan konten berwarna merah digunakan untuk memberikan kesan medis.



Gambar III-12 Kondisi final desain.

Keseluruhan metode yang mengatur style dasar dari grafik ada di bawah ini. Keterangan dari tiap-tiap parameter telah dijelaskan oleh komentar source code.

```
// Specify the theme of every waveform
private void amplitudeWave(SimpleWaveform simpleWaveform,
LinkedList<Integer> ampList) {
    // Receive which waveform and data list

    simpleWaveform.init();

    simpleWaveform.setDataList(ampList);

    //define background
    background.setColor(Color.LTGRAY);
    simpleWaveform.background = background;

    //define bar gap
    simpleWaveform.barGap = setGap;

    //define the full height range normalization
    // Set full height as 2*amplitude
    simpleWaveform.modeNormal = SimpleWaveform.MODE_NORMAL_VALUE_MAX;

    //define x-axis direction
    simpleWaveform.modeDirection =
        SimpleWaveform.MODE_DIRECTION_RIGHT_LEFT;

    //define if draw opposite pole when show bars.
    // Doing so will make negatives as absolutes.
    simpleWaveform.modeAmp = SimpleWaveform.MODE_AMP_ORIGIN;
    //define if the unit is px or percent of the view's height
    simpleWaveform.modeHeight = SimpleWaveform.MODE_HEIGHT_PX;
```

```

//define where is the x-axis in y-axis
simpleWaveform.modeZero = SimpleWaveform.MODE_ZERO_CENTER;
//if show bars?
simpleWaveform.showBar = false;

//define how to show peaks outline
simpleWaveform.modePeak = SimpleWaveform.MODE_PEAK_ORIGIN;
//if show peaks outline?
simpleWaveform.showPeak = true;

//show x-axis
simpleWaveform.showXAxis = true;
xAxisPencil.setStrokeWidth(1);
xAxisPencil.setColor(getResources().getColor(R.color.divider));
simpleWaveform.xAxisPencil = xAxisPencil;
//show x-axis on top of outline or under
simpleWaveform.modePriority = SimpleWaveform.MODE_AXIS_UNDER_AMP;

//define pencil to draw bar
barPencilFirst.setStrokeWidth(1);
barPencilFirst.setColor(0xff1dcf0f);
simpleWaveform.barPencilFirst = barPencilFirst;
barPencilSecond.setStrokeWidth(1);
barPencilSecond.setColor(0xff1dcfcf);
simpleWaveform.barPencilSecond = barPencilSecond;

//define pencil to draw peaks outline
peakPencilFirst.setStrokeWidth(1);
peakPencilFirst.setColor(getResources().
    .getColor(R.color.colorPrimary));
simpleWaveform.peakPencilFirst = peakPencilFirst;
peakPencilSecond.setStrokeWidth(1);

peakPencilSecond.setColor(getResources().
    .getColor(R.color.colorPrimary));
simpleWaveform.peakPencilSecond = peakPencilSecond;

//the first part will be draw by PencilFirst
simpleWaveform.firstPartNum = 20;

//define how to clear screen
simpleWaveform.clearScreenListener =
new SimpleWaveform.ClearScreenListener() {
    @Override
    public void clearScreen(Canvas canvas) {
        canvas.drawColor(Color.TRANSPARENT, PorterDuff.Mode.CLEAR);
    }
};
}

```

Algoritma III-5 Metode yang mengatur tema grafik.

III.4 Sistem Filtrasi

Bagian matematis utama dari keseluruhan program Android adalah proses filtering. Algoritma filtering didasarkan pada rumus umum implementasi filter digital, yang bisa digunakan untuk implementasi filter FIR maupun filter IIR. Berikut perumusan tersebut [13].

$$y(n) = \sum_{i=0}^M b_i x[n-i] - \sum_{j=1}^N a_j y[n-j] \quad (\text{III. 1})$$

Dari perumusan tersebut, tentunya implementasi secara langsung dalam bahasa pemrograman akan mudah. Namun tentunya dalam implementasi nyata akan ada hal yang harus diuji, terutama atas ada tidaknya bentukan matematis yang bisa lebih efisien untuk dijalankan dalam suatu smartphone Android.

Sebelum menguji bentukan yang efisien, algoritma dasar penanganan filter harus dibuat terlebih dahulu. Karakteristik utama yang diinginkan adalah filter-filter tersebut memiliki sifat modular yang tinggi. Dengan menggunakan bahasa Java, tentunya hal ini sangat cocok karena Java adalah bahasa pemrograman OOP. Nanti tiap-tiap objek filter akan memiliki nilai input (x), output (y), parameter-parameter (a & b) serta buffer sendiri-sendiri, alhasil diperoleh objek-objek yang saling terpisah – sifat modular terpenuhi. Dari tiap-tiap objek itu, filter akan saling kirim-terima data dengan tahapan-tahapan (stage) filter sebelum & selanjutnya menggunakan algoritma seperti yang digunakan oleh proses pembaruan data grafik; yaitu Algoritma III-3.

Mula-mula dalam deklarasi awal, suatu objek filter akan meminta koefisien-koefisien – b saja untuk FIR ditambah a untuk IIR. Perpotongan source code class filter ini beserta cara umum untuk mendeklarasikannya ada di Algoritma III-6 dan Algoritma III-7 berturut-turut. Sebagaimana dijelaskan pada subbab III.1 bagian metode pengaliran data, bentukan tipe data array adalah pilihan terbaik. Namun array sendiri masih memiliki dua paradigma umum yang cukup berbeda jauh: penggunaan secara normal atau linear, serta penggunaan secara berputar atau *circular*.

```

public class Filter {
    private int order = 1;
    private boolean iir = false;
    private double[] inputs;
    private double[] outputs;
    private double[] a = {0};
    private double[] b = {0};
    private boolean rectified = false;

    private int buffSize = 4;    // Length of the buffer array
    private double[] buffer;
    private int ib = 0;          // Buffer pointer

    Filter(int order, double[] b, int buffSize, boolean rectified) {
        this(order, buffSize);
        this.rectified = rectified;
    }

    //... Other constructors

    // FIR
    Filter(int order, double[] b) {
        this.order = order;
        this.b = b;
        this.iir = false;
        if (b.length != order) throw
            (new IllegalArgumentException("Invalid b-array size."));
        init();
    }

    private void init() {
        // From language spec, def val of an array of double is pos zero
        if (iir) outputs = new double[order];
        else outputs = new double[1];
        inputs = new double[order];
    }

    //...
}

```

Algoritma III-6 Potongan source code untuk class Filter berisikan salah dua konstruktor umum, inisiasi, serta variabel lokal.

```

public void addVal(double value) {
    double temp;

    // Shifting data
    int i;
    for(i = order - 1; i > 0; i--) {
        inputs[i] = inputs[i - 1];
        if (iir) outputs[i] = outputs[i - 1];
    }

    // Get the new first data, so current input
    if (rectified) inputs[0] = abs(value);
    else inputs[0] = value;

    // Calculate the convolution from koefs
    temp = 0;
    for (i = 0; i < order; i++)

```

```

temp += inputs[i] * b[i];

if (iir) for (i = 1; i < order; i++)
    temp -= outputs[i] * a[i];

// Result
outputs[0] = temp;
addBuffer(temp);    // Add data to buffer
}

```

Algoritma III-7 Perpotongan source code class Filter yang berisikan penanganan satu data baru, yaitu proses matematis filtering.

Secara singkat, bentukan linear berarti urutan data dimasukkan setara dengan index array, sehingga proses pergeseran (shifting) memerlukan pergeseran secara nyata terhadap tiap-tiap data di dalam array. Dalam bentukan circular, data-data terurut secara berputar di dalam array (urutan data runtut awal-akhir kembali ke awal secara kontinu) dengan titik awal mengikuti suatu variabel penunjuk kepala data (head). Dalam kasus tersebut data terlama akan tertimpa oleh data terbaru ketika penuh, dan pergeseran data hanya perlu dilakukan dengan menggeser variabel penunjuk kepala data (head). Efek sampingnya adalah algoritma yang sedikit lebih kompleks & pembebanan tinggi pada operasi modulo. Metode Circular terlihat seperti tawaran yang baik, namun menggeser data (shifting) satu-per-satu ternyata jauh lebih baik daripada menggunakan beberapa operasi modulo dalam smartphone Android.

Sedangkan tiap-tiap filter maupun sistem lainnya akan saling mengasih & menerima melalui buffernya masing-masing. Bentukan yang sudah dirancang modular membuat perancangan tahapan filter yang satu dengan yang lainnya sangat mudah. Berikut potongan source code dari sebagian aliran data input ke final.

```

recorder.read(sData, 0, BufferElements2Rec);

// Full scheme written on Readme.md
// Anti-aliasing, dual stage
filaal.addArray(sData, n, DS0 / fsdev);
filaa2.addArray(filaal.getBuffer(), n, DS1);

filChA.addArray(filaa2.getBuffer(), n, DS2);    // HbO2
filChB.addArray(filaa2.getBuffer(), n, DS2);    // Hb

```



```
// Rectify then clear carrier  
filDemodA.addArray(filChA.getBuffer(), n, DS3);  
filDemodB.addArray(filChB.getBuffer(), n, DS3);  
  
//...
```

Algoritma III-8 Perpotongan source code dari sebagian aliran data input sebelum ditampilkan ke grafik.

BAB IV

PENDESAINAN DAN IMPLEMENTASI

SISTEM TRANSMISI

Dalam pemilihan metode transmisi, ada berbagai faktor yang dipertimbangkan seperti kompatibilitas, kemudahan, performa, kualitas data, dll. Namun dalam kasus ini, untuk memenuhi goal sebagai peralatan yang sangat umum penggunaannya, faktor kompatibilitas menjadi faktor yang sangat utama. Secara singkat, kompatibilitas berarti peralatan oximeter dapat cocok dan berkerja baik untuk smartphone sebanyak mungkin. Sedangkan opsi pengiriman data kontinyu umumnya ada dua hal, melalui audio jack sebagai data analog ataupun melalui USB OTG sebagai data serial (digital).

Hasil pencarian singkat menunjukkan dalam sautu katalog web jual beli online, hanya ada sekitar 795 handphone yang memiliki fitur OTG dari total 6.887 yang ada di katalog tersebut – 11,54% jika dikonversikan [14], [15]. Hal ini tentunya sangat sedikit jika dibandingkan handphone yang tidak memiliki audio jack, karena pergerakan desain ke arah tersebut memang baru dimulai akhir-akhir ini dan itupun banyak perlawanan. Dan tidak jarang pembelian handphone tanpa audio jack tidak dilengkapi secara langsung dengan adaptor khusus agar fungsionalitas audio jack bisa diperoleh kembali. Sehingga dengan faktor kompatibilitas, metode transmisi melalui audio jack dipilih.

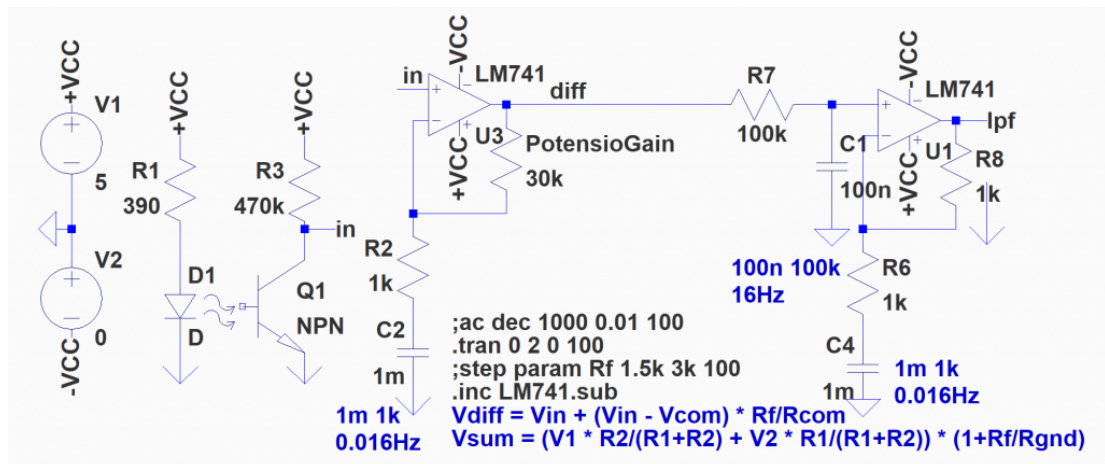
Dari pemilihan metode ini, akan bermunculan berbagai pemasalahan teknis. Permasalahan-permasalahan teknis itu didasarkan pada dua permasalahan yang paling mendasar: Keperluan untuk mengirim dua sinyal dalam satu kabel & HPF internal audio jack. Pengiriman dua buah sinyal diperlukan untuk perhitungan SpO_2 , yaitu sinyal HbO_2 (hemoglobin kaya oksigen) dan sinyal Hb (hemoglobin tidak kaya oksigen) [4]. Sedangkan efek HPF internal membuat sinyal data harus diangkat ke frekuensi tinggi agar sinyal dari oximeter tidak hilang ataupun terdistorsi. Kedua hal tadi umumnya diselesaikan dengan modulasi, dalam fungsinya sebagai salah satu

metode multiplexing pembagian frekuensi (FDM) untuk kasus pertama dan sebagai penggeser frekuensi keseluruhan untuk kasus kedua.

Sebelum melihat grafik-grafik yang ada pada laporan ini, perlu diperhatikan bahwa konfigurasi penggambaran grafik sering diubah-ubah seiring melakukan pengujian. Lebar, range vertikal, downsampling, dll. Selain itu data-data yang ditunjukkan pada GUI BPM dan SpO₂ hanyalah placeholder dan berisikan data lain, kecuali dispesifikasikan sebaliknya pada caption gambar.

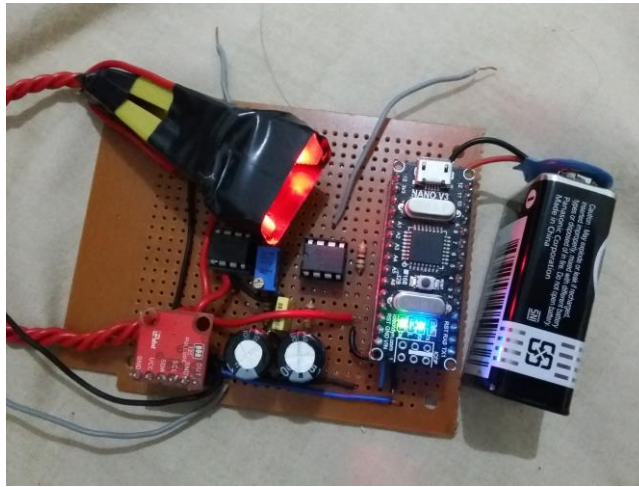
IV.1 Pulse Oximeter

Sebelum membahas dalam pada aspek teknis sistem transmisi, penjelasan akan dimulai dengan alat oximeter buatan yang dikembangkan sebelum berjalannya kerja praktek. Alat oximeter yang digunakan memiliki skematis serupa gambar di bawah ini (BJT NPN merupakan Phototransistor). Tampilan ini merupakan tampilan dalam workspace SPICE, sehingga berisikan informasi-informasi tambahan yang dapat diabaikan. Perlu diperhatikan bahwa rangkaian di Gambar IV-1 hanyalah rangkaian pengondisian analog. Sehingga perlu ditambahkan bahwa sinyal akhir dimasukkan ke dalam input ADC Arudino Nano yang berfungsi sebagai DSP oximeter. Kemudian diakhiri dengan modul DAC agar dapat dihasilkan sinyal analog dari Arduino. Dalam implementasi ini, digunakan Arduino Nano (ATmega328p, 5V) serta DAC MCP4725.



Gambar IV-1 Rangkaian pengondisian analog oximeter buatan (BJT NPN merupakan phototransistor).

Hal lain yang cukup penting, terutama dalam menyikapi data-data yang ada pada laporan, adalah penjepit sensor yang bisa dibilang masih terlalu kasar. Perbedaan kemiringan ketika meletakkan jari pada penjepit dapat mengakibatkan perubahan bentuk gelombang, di samping perubahan offset. Namun dalam kasus perubahan offset, suatu HPF dapat mengatasi hal tersebut. Gambar IV-2 menunjukkan alat oximeter yang digunakan dalam pemenuhan proyek ini. Secara general, proyek memang menjadi satu keutuhan dengan oximeter tersebut. Namun karena memang perancangan alat tidak dilaksanakan pada masa kerja praktek, fokus hanya akan tertuju pada pengembangan alat agar bisa diberikan antar muka melalui handphone. Sedangkan informasi segala sesuatu tentang Pulse Oximeter ini bisa dilihat pada website robotika.unit.itb.ac.id dengan artikel berjudul *Implementasi Pulse Oximeter (Sensor Detak Jantung) melalui Rangkaian Penguat Sederhana beserta Digital Filtering* [16].

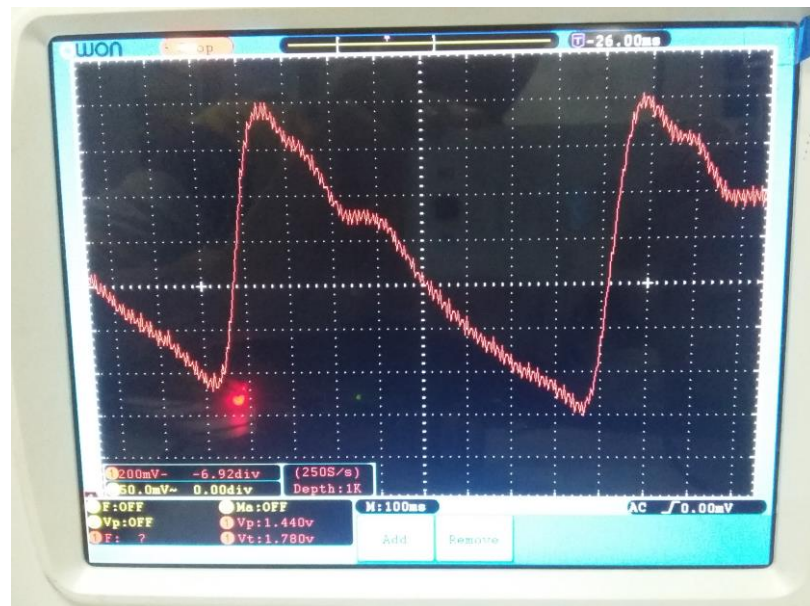


Gambar IV-2 Alat oximeter buatan yang digunakan.

Hal yang diubah dalam aspek hardware adalah penambahan jack female pada output DAC. Sebelumnya output secara langsung dikeluarkan melalui dua buah kabel jumper. Alhasil akan diperoleh koneksi yang lebih konsisten dan bersih. Kondisi inilah mengapa opsi modulasi amplitudo (AM) dipilih, pengaruh dari noise aditif – kelemahan utama dari AM – akan sangat rendah karena digunakan koneksi secara langsung. Sebagai tambahan, Gambar IV-3 menunjukkan sinyal analog yang dikeluarkan DAC Oximeter.

Hal terpenting lainnya mengenai rangkaian analog tersebut adalah mengenai relasi antara penguatan terhadap clipping atau terpotongnya data. Data yang terlalu kecil tentunya akan membuat data rentan terhadap noise, dan juga membuat rata-rata resolusi satu buah data valid semakin kecil ketika dibaca oleh ADC. Namun data yang terlalu besar dapat mengakibatkan data penting mengalami clipping; terlebih lagi adanya fakta bahwa offset masing-masing pengguna yang berbeda-beda, atau bahkan pengaturan posisi yang berbeda. Alhasil semakin besar penguatan sinyal sensor, rentang offset yang bisa diatasi akan semakin kecil – mengurangi kompatibilitas alat terhadap sekelompok orang ataupun meningkatkan ketergantungan alat terhadap pemosisian alat yg akurat. Inilah alasan mengapa penguatan total rangkaian analog diterapkan menggunakan suatu potensiometer, alhasil penguatan bisa diatur-aturl.

Skema teknis keseluruhan sistem-sistem yang digunakan ada pada repository yang sudah disebutkan sebelumnya di BAB III, tepat pada deskripsi repo ataupun pada file README.md [6]. Potongan README.md juga terlampir pada LAMPIRAN C. Sedangkan isi dari skema tersebut tidak lain adalah skema transmisi untuk sisi Arduino dan skema rekonstruksi untuk sisi Android, keduanya akan dijelaskan pada dua subbab selanjutnya – permasalahan, mengapa digunakan, alternatif, dll.



Gambar IV-3 Pengujian output analog Oximeter menggunakan osiloskop.

IV.2 Arduino

Bagian program Arduino adalah bagian yang mengalami banyak sekali perkembangan, akibat dari keperluan transmisi khusus yang sudah dijelaskan sebelumnya. Karena banyaknya permasalahan besar yang didapat selama mengimplementasi sistem modulasi, akan disebutkan poin permasalahan-permasalahan tersebut terlebih dahulu.

- Efek pergeseran fasa oleh filter IIR berdampak sangat signifikan sehingga memerlukan implementasi filter FIR. Filter FIR akan memerlukan orde yang jauh lebih tinggi (bisa 10x lebih) dibanding IIR dengan spesifikasi yang sama.
- Kecepatan mikrokontroller yang tidak seberapa cepat membuat implementasi carrier sinusoidal memperlambat proses utama, sehingga frekuensi carrier hanya akan bisa rendah dalam kasus tersebut.
- Sinyal sinusoidal akan sangat terkotak-kotak akibat faktor kecepatan mikrokontroller, sehingga dapat munculnya distorsi ketika dilewatkan audio jack (sinyal kotak memunculkan komponen frekuensi rendah yang cukup dominan).

Melakukan modulasi dengan kecepatan Mikrokontroller yang cukup lamban mengakibatkan efek yang sangat signifikan. Beberapa pengujian menunjukkan hasil yang memang benar “berbentuk” tapi jauh dari konsisten. Sehingga perubahan selalu terjadi dengan tradeoff yang sangat signifikan.

Keseluruhan proses pada tahap awal dimulai dengan secara matematis menghitung waktu yang dibutuhkan untuk melakukan setiap loop atau melakukan suatu proses. Definisi waktu untuk “melakukan suatu proses” perlu ditekankan dibanding “setiap loop” karena setiap sistem dirancang dengan sampling yang berbeda-beda (melalui downsampling & upsampling relatif dari proses sblmnya). Hal ini diwujudkan melalui suatu loop counter, yang kemudian dilewatkan suatu persamaan modulo sehingga masing-masing sistem akan dieksekusi setiap jumlah loop yang berbeda-beda, tergantung dengan berapakah frekuensi loop dan berapakah frekuensi ketika sistem tersebut didesain. Tentunya suatu sistem nantinya tidak bisa memiliki spesifikasi frekuensi yang lebih tinggi daripada $\frac{1}{2}$ frekuensi loop berdasarkan teorema Nyquist, agar tidak terjadi aliasing [17]. Keperluan downsampling/upsampling ini sangat penting untuk selain menyederhanakan spesifikasi filter yang diperlukan, mengurangi pemanggilan suatu sistem dalam satu loop, serta menyederhanakan bentukan data.

Cara yang cukup “kasar” tadi justru sangat konsisten, karena keseluruhan sistem akan berjalan dengan frekuensi relatif terhadap frekuensi global (frekuensi loop), berbeda jika dibandingkan metode penggunaan interrupt. Namun penggunaan interrupt

memungkinkan suatu proses sederhana untuk dijalankan dalam frekuensi yang lebih cepat dari frekuensi loop utama. Oleh karena itu, pada tahapan akhir, kombinasi kedua metode ini akan digunakan (akan dijelaskan pada bagian kemudian-kemudian).

Pada pengujian awal, algoritma yang sudah cukup lengkap dibuat berdasarkan kecepatan loop yang diasumsikan. Kecepatan yang lebih tepat kemudian diperoleh dengan menggunakan fungsi `micros()` dari Arduino, fungsi yang menggunakan timer internal Arduino (namun terpisah dari CPU) untuk menghitung waktu dengan resolusi 4us. Perlu diperhatikan bahwa meski Oximeter yang dikembangkan pada saat ini hanya menerima satu buah input, sistem agar Oximeter dapat dengan mudah menerima dua input sensor secara bergantian sudah diimplementasikan dan diperhitungkan dalam pengukuran waktu. Alhasil diperoleh hasil-hasil sebagai berikut.

- Laju satu buah loop tercepat (75%) = ~1,3kHz – 1,4kHz
- Laju satu buah loop terlambat (25%) = ~227Hz - 250Hz
- Laju loop rata-rata = ~1kHz – 1,1kHz
- Laju pengambilan data baru = ~37Hz - 41Hz

Dari data-data tersebut, dapat ditarik batasan-batasan teknis yang sangat menarik. Laju pengambilan data baru memberikan batasan frekuensi sinyal yang menjadi input. Untungnya suatu sinyal oximeter yang penting ada pada rentang 0,66-15Hz, yang mana cukup di bawah $\frac{1}{2}$ sampling rate [18]. Sedangkan parameter laju loop akan menentukan seberapa besar upsampling (memperluas satu sampel layaknya x buah sampel) yang bisa dilakukan, misal untuk keperluan modulasi. Laju loop rata-rata menentukan frekuensi maksimum carrier yang bisa digunakan untuk memodulasi sinyal. Sedangkan parameter-parameter frekuensi yang berfluktuasi memang berpengaruh dalam karakteristik filter, dalam artian titik cutoff bisa tergeser-geser (frekuensi nyata lebih besar dari frekuensi desain membuat titik cutoff lebih besar, dan sebaliknya) karena spesifikasi desain suatu filter digital pada dasarnya bukanlah “ x Hz” namun “ x kali lipat frekuensi sampling”. Di lain sisi, frekuensi carrier juga bisa tergeser, namun hal tersebut tidak bermasalah selama tidak melewati bandwidth yg sudah ditentukan.

Keseluruhan spesifikasi lengkap namun lengkap untuk sistem Arduino juga ada pada repo [6] maupun LAMPIRAN C. Mula-mula Arduino dimulai dengan pembacaan data oximeter, yaitu output rangkaian analog. Sebagaimana telah dijelaskan sebelumnya, mengenai keperluan untuk pembacaan dua sinyal demi pengembangan lebih lanjut, diperlukan metode pembacaan input yang bisa saling bergantian dengan sedikit waktu jeda di antara keduanya. Skenario yang digunakan adalah sebagai berikut: data 1, kosong, data 2, kosong – satu data 1 dan data 2 masing-masing diambil tiap 4 loop. Selanjutnya hanya cukup memberikan mekanisme switching sumber dua LED – infrared atau merah – agar data HbO₂ maupun Hb bisa diperoleh secara bergantian. Alhasil diperoleh algoritma sebagai berikut. Dalam potongan tersebut juga terdapat salah satu contoh penggunaan mekanisme downsampling yang sudah sering disebutkan sebelumnya.

```
input = analogRead(A2);  
  
// Data mapping & downsampling, add data only every x indexes  
// TODO electrical settings on switching  
if (modulo(n, DS0) == 0) {  
    if (modulo(n, DS0 * DS1) == 0) {  
        // HbO2  
        inA = input;  
    } else if (modulo(n, DS0 * DS1) == 1) {  
        // Off  
        inOff = input;  
    } else if (modulo(n, DS0 * DS1) == 2) {  
        // Hb  
        inB = input;  
    } else if (modulo(n, DS0 * DS1) == 3) {  
        // Off  
        inOff = input;  
    }  
}
```

Algoritma IV-1 Potongan source code yang bersangkutan dengan pengambilan sumber

Setelah data diambil, masing-masing data akan dimasukkan filternya masing-masing. Filter dalam kasus ini dipisah karena memang kedua data tersebut memiliki jalurnya masing-masing, bukan berarti filter2 tersebut memiliki karakteristik yang berbeda

(namun boleh saja). Dalam kasus ini filter yang digunakan merupakan HPF dengan karakteristik yang sama, untuk menghilangkan offset. Jenis-jenis filter lainnya bisa digunakan, cukup dengan mengganti koefisien2 objek filter yang bersangkutan – manfaat terbesar penggunaan OOP dalam implementasi filter. Sekali lagi bisa dilihat

```
// Offset suppression
if (modulo(n, (DS0 * DS1 * DS2)) == 0) {
    dsA = inA;
    dsB = inB;

    filNormA.addVal(dsA);
    filNormB.addVal(dsB);

    // Set interpolation start as the prev result
    interA = filA;
    interB = filB;

    filA = filNormA.getVal();
    filB = filNormB.getVal();

    distA = filA - interA;
    distB = filB - interB;
}
```

Algoritma IV-2 Potongan source code yang bersangkutan dengan filtering dalam Arduino.

Tahapan selanjutnya umumnya memang melakukan modulasi, namun perlu diingat dari spek2 frekuensi yang sudah disebutkan sebelumnya bahwa data mentah saja memiliki frekuensi sangat rendah (< 50Hz). Sedangkan diinginkan penggunaan carrier dengan frekuensi yang lebih tinggi (> 50Hz), agar bisa mengangkat sinyal oximeter melewati HPF internal smartphone. Belum lagi adanya efek downsampling yang sudah dilewati, yang mana akan menurunkan frekuensi sampling data pada tahapan ini equivalen dengan ($DS_1 = 4$ karena data diperbarui tiap 4 loop):

$$\text{Setelah filtering: } F_{S_3} = F_{S_0} : DS_0 : 4 : DS_2 \quad (\text{IV.1})$$

Cara untuk meningkatkan sampling rate tidak lain adalah dengan upsampling. Dalam melakukan upsampling, ada dua buah skenario dasar yang bisa digunakan; antara data ditahan (hold) ataupun interpolasi (rekonstruksi). Implementasi yang sangat mudah

serta efek yang sangat signifikan membuat interpolasi dipilih, lebih spesifiknya interpolasi linear. Singkatnya, interpolasi linear cukup dilakukan dengan menarik garis lurus antara dua buah titik sampel yang saling berdekatan. Seberapa banyak data yang dimunculkan di antara dua buah sampel bergantung pada seberapa besar upsampling yang dilakukan. Algoritma upsampling akan sangat sederhana, namun memerlukan variabel yang menampung kondisi data sebelumnya (sehingga akan telat 1 sampel). Algoritma IV-3 menjelaskan proses upsampling itu sendiri. Namun kebutuhan untuk upsampling sendiri sudah dipersiapkan oleh variabel *interX* sebagai titik awal (atau sederhananya titik sampel sebelumnya) maupun *distX* sebagai jarak antara dua sampel, sebagaimana terlihat pada Algoritma IV-2, atau tahapan tepat sebelumnya.

```
// Linear interpolation
if (n % (DS0 * DS1 * DS2 / US3) == 0) {
    interA += distA / US3;
    interB += distB / US3;
}
```

Algoritma IV-3 Potongan source code yang bersangkutan dengan upsampling dalam Arduino.

Dengan demikian, proses modulasi bisa dilakukan. Jenis AM yang digunakan adalah DSBAM demi proses demodulasi yang paling sederhana. Hal ini dikarenakan sinyal yang dipunyai memiliki daerah negatif, sehingga perlu adanya pemberian sinyal DC atau offset. Sebagaimana dijelaskan pada subbab sebelumnya bahwa konfigurasi rangkaian analog telah dibuat agar rentang pesan lebih kecil (secara substansial) dibanding rentang keseluruhan demi menghindari clipping. Sehingga batasan-batasan absolut dari sinyal memang dapat dicari/diketahui, namun tidak pasti dengan rentang data penting. Namun menggunakan batasan tersebut tentunya akan membuat rasio komponen data penting terhadap data keseluruhan semakin kecil, mengurangi kemungkinan sinyal bisa didemodulasi/direkonstruksi dengan baik. Dengan demikian nilai offset DC, sebagai komponen yang bersangkutan langsung dengan rentang data penting, merupakan salah satu komponen yang perlu pengujian serta kalibrasi di sisi

digital – di samping komponen penguatan. Implementasi tersebut ditunjukkan pada algoritma di bawah ini.

```
// Based on manual testing
#define FS 200

// Carriers
#define FCA 70
#define FCB 100

//...

carrA = cos(2 * PI * FCA / FS * m);
carrB = cos(2 * PI * FCB / FS * m);

// Currently MESSAGE is [-100, 100], converted to [0, 200]
// Thus modulated at [-200, 200]
chA = ((interA > -NEG) ? (NEG + interA) : (0)) * carrA;
chB = ((interB > -NEG) ? (NEG + interB) : (0)) * carrB;

output = (chA + chB) / 2;
```

Algoritma IV-4 Potongan source code yang bersangkutan dengan modulasi

Yang tersisa adalah pengondisian sinyal akhir. Skema multiplikasi dan sejenisnya merupakan hasil pengujian terakhir. Baik algoritma serta skema pengambilan keputusan kalibrasi ada pada

```
// Final output handling
// Amplify & offset to maximize range & avoid negatives
/* Data about the range:
 * - For a full range wave [-1, 1] in Audacity, phone data is
 *   good at ~50% vol
 * - For a half-range wave [-.5,.5] in Audacity, phone data is
 *   good at ~80-90% vol
 * - DAC sine with amp of 512 (4096 / 8, or 0,625 Vp) will result
 *   in a very little clipping
 * - Currently combined output is [0, 400] for dual ch
 * - Audacity full-range wave at 10% vol playback equals to a
 *   full-range wave at recording
 * - Optimal conclusion: 50% Audacity playback => 5% Audacity
 *   record => 25.6 Arduino DAC (3mVp)
 */
```

```

int temp = (output + 2*NEG) / 8 + BOTTOM;      // [-200, 200] =>
[0, 400]

if (temp > BOTTOM) {
    dac.setVoltage(temp, false);
} else {
    dac.setVoltage(BOTTOM, false);
}

```

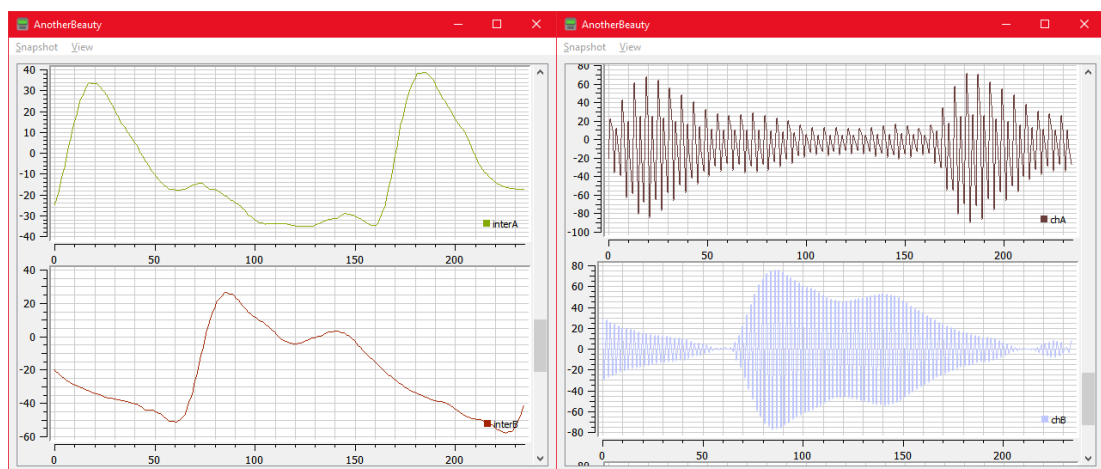
Algoritma IV-5 Potongan source code pengondisian akhir.

Mengujikan data di tiap-tiap tahapan akan memberikan bermacam-macam grafik pada Gambar IV-4 dan . Perlu diperhatikan bahwa pada implementasi lanjut, input di sini akan berarti data yang berubah-ubah karena adanya switching antara lampu infrared dengan lampu merah. Namun dalam kasus ini, input merupakan data ketika lampu merah menyala (volume HbO₂). Alhasil keseluruhan ch A dan ch B akan bersumber pada data yang sama namun dengan offset pembacaan yang berbeda – menghasilkan perbedaan yang cukup signifikan pada kasus di Gambar IV-4.

Hal yang menarik dari melihat hasil interpolasi (interA & interB) adalah bentukan yang jauh lebih halus dibanding hasil tanpa interpolasi (filA & filB). Dengan demikian, bisa dilihat bahwa dengan solusi yang cukup sederhana, interpolasi linear ternyata baik dalam memperbaiki rendahnya resolusi suatu sinyal. Sedangkan jenis-jenis interpolasi sendiri ada beragam seperti interpolasi polynomial, interpolasi datar, dll [19].

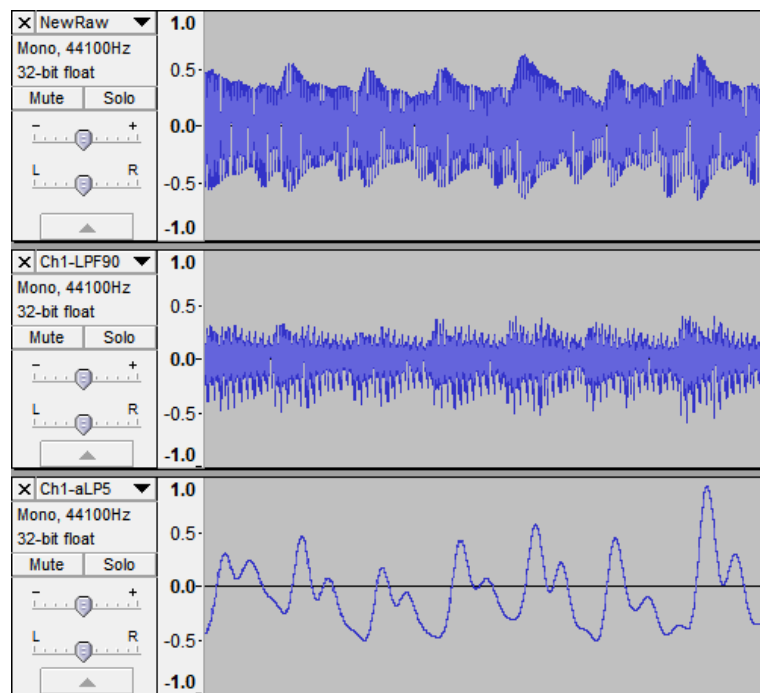


Gambar IV-4 Grafik per tahapan ketika alat dipasangkan ke jari (data valid). Berturut-turut pada sisi kiri: penunjuk looping global index, input, input ch A & B. Pada sisi kanan: downsampled ch A & B, filtered ch A & B.

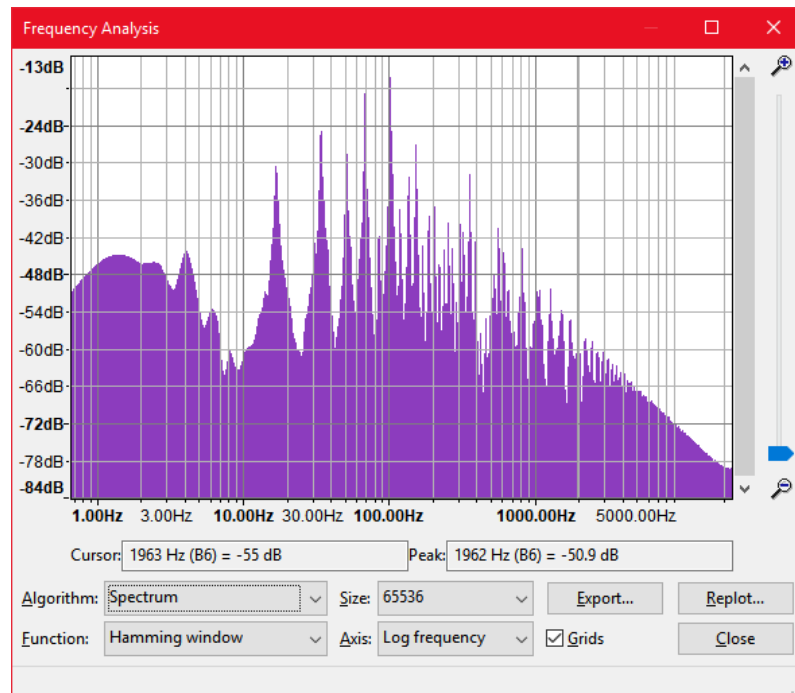


Gambar IV-5 Grafik per tahapan ketika alat dipasangkan ke jari (data valid). Berturut-turut pada sisi kiri: interpolated ch A & B. Pada sisi kanan: Modulated ch A & B.

Untuk menguji karakteristik nyata yang telah dibuat, output DAC Oximeter akan direkam menggunakan audio jack komputer & program Audacity. Untuk pengujian dua channel serta mekanisme transmisi, sinyal oximeter yang sama akan dibaca dua kali dengan sinyal pertama dimodulasi dengan carrier 70Hz dan sinyal kedua dengan carrier 100Hz. Alhasil kedua data ketika direkonstruksi seharusnya serupa hanya bergeser fasanya. Hasil perekaman data, serta runtutan apabila direkonstruksi dengan envelope detector ditunjukkan pada Gambar IV-6 (beturut-turut: data mentah, pemisahan channel dengan LPF 90Hz, lalu rectifier + LPF). Sedangkan spektrum dari data mentah yang diperoleh ada pada Gambar IV-7.



Gambar IV-6 Perekaman output DAC Oximeter menggunakan audio jack komputer & program Audacity. Grafik pertama dari atas adalah data mentah, grafik kedua hasil LPF 90Hz, grafik ketiga hasil rectifier & LPF 5Hz.



Gambar IV-7 Analisis spektrum data mentah dari data yang ada pada Gambar IV-6.

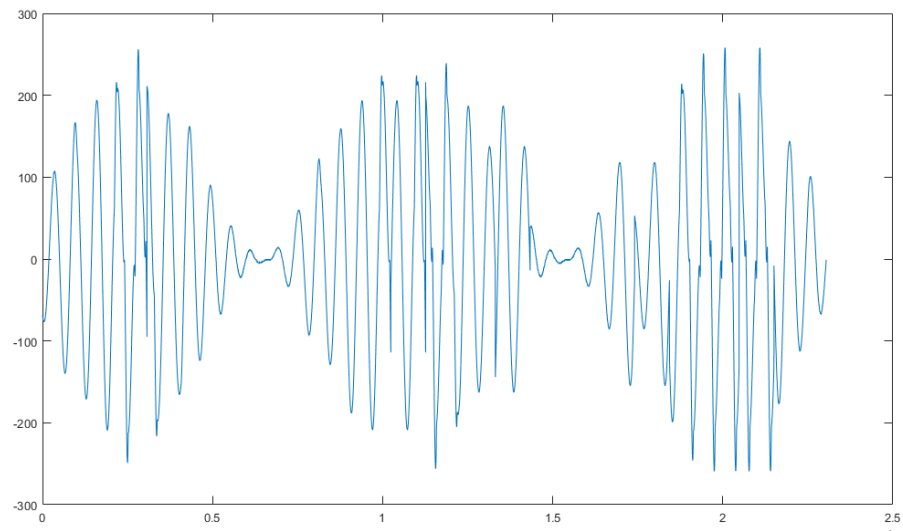
Hal terpenting yang perlu dikonfirmasi adalah bagaimanakah komponen dari sinyal output oximeter. Dari grafik spektrum di Gambar IV-7, diperoleh puncak-puncak di titik frekuensi 101Hz dan 68Hz. Tentunya kedua parameter ini menunjukkan bahwa skema “kasar” yang telah disebutkan sebelumnya dapat berkerja dengan baik. Sebab timing di sini hanya bergantung pada loop saja.

Pada grafik-grafik per tahap untuk rekonstruksi, bisa dilihat bahwa sinyal oximeter cukup bagus dengan fitur detail seperti *notch* (cekungan kecil) terjaga. Sayangnya melakukan pengujian langsung dengan smartphone pada Gambar IV-8 menunjukkan data yang sangat tidak konsisten (abaikan grafik bawah). Bahkan tidak jarang gelombang sangat hancur dengan fitur sekedar satu buah bukit tanpa adanya notch. Padahal notch tadi cukup penting dalam dunia medis, karena merupakan salah satu aspek yang digunakan sebagai alat mendiagnosis kondisi seseorang [5]. Sering juga bentukan bukit yang terbalik bermunculan (bukit kedua lebih tinggi dari bukit pertama dalam satu gelombang), dan kasus-kasus lainnya. Salah satu hasil dari banyak pengujian dengan smartphone ditunjukkan pada Gambar IV-8.

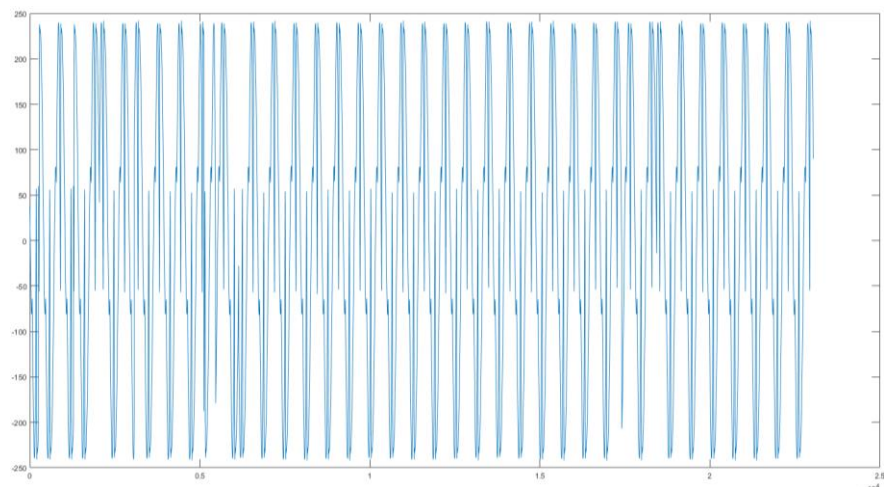


Gambar IV-8 Salah satu pengujian modulasi-demodulasi menggunakan carrier sinusoidal 70Hz (abaikan grafik bawah).

Pengujian lebih lanjut dengan mengubah-ubah konfigurasi, spesifikasi, maupun mencari metode lain menunjukkan usaha yang sia-sia. Hal yang membuat aneh lagi adalah pengujian menggunakan sinyal dari PC (dengan menyiapkan sinyal termodulasi secara manual) ke smartphone menunjukkan hasil yang lebih rapi dan stabil. Alhasil pengujian dengan fokus ke data perekaman mentah smartphone dilakukan (pengambilan data melalui fitur copy dari app). Mula-mula pengujian terhadap suatu sinyal sinusoidal 5Hz yang termodulasi dengan komponen 7Hz menghasilkan grafik pada Gambar IV-9 (diplot di Matlab karena data sangat panjang). Kemudian dibandingkan dengan sinyal kotak 70Hz yang menunjukkan grafik pada Gambar IV-10.



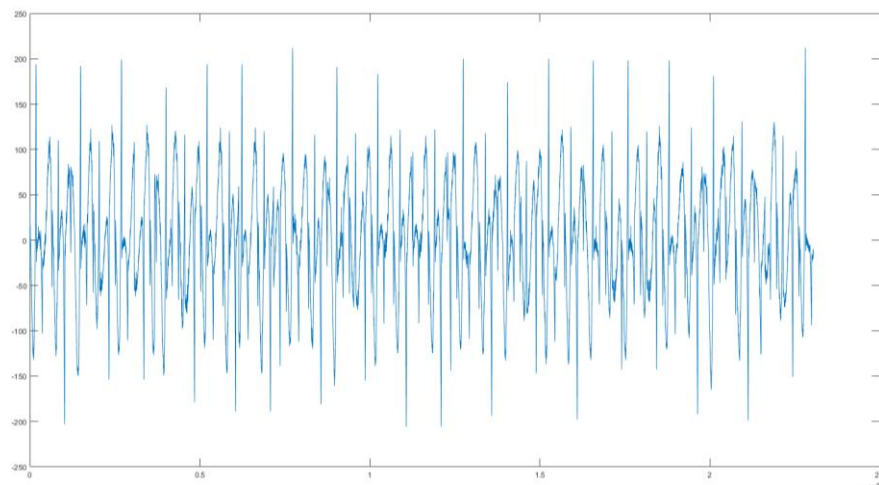
Gambar IV-9 Data mentah aplikasi smartphone ketika diberikan input sinusoidal 5Hz termulasi 70Hz dari PC.



Gambar IV-10 Data mentah aplikasi smartphone ketika diberikan input sinyal kotak 70Hz dari PC.

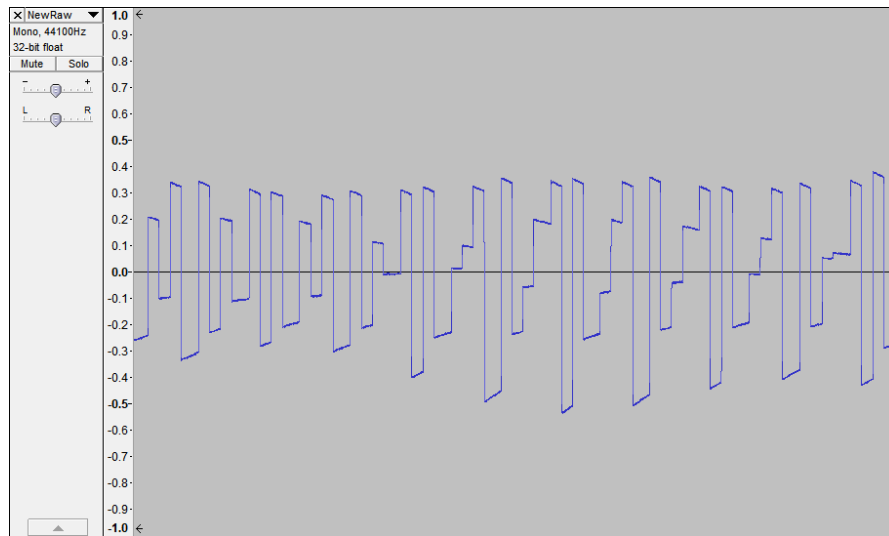
Dari kedua grafik tersebut, bisa dilihat bahwa ada komponen sinyal frekuensi tinggi yang tidak sesuai dengan sesungguhnya untuk input kotak. Pada saat input sinusoidal, terbentuk pula sinyal sinusoidal. Namun hal ini berbeda dengan sinyal kotak, yang mana terbentuk atas “jarum-jarum” atau *spike*. Kasus ini tentunya serupa dengan kasus ketika suatu sinyal kotak dilewatkan HPF. Memang terlihat tidak ada korelasinya dengan skema awal tadi, dimana input melalui PC lebih bagus dibanding input melalui

oximeter langsung. Namun ketika Arduino memberikan sinyal pesan flat (sensor tidak dipasang) termodulasi, dilihat karakteristik “spike” yang sama pada Gambar IV-11. Padahal, sebagaimana yang diketahui, output Arduino seharusnya sinusoidal dan memang tidak ada masalah pada PC.

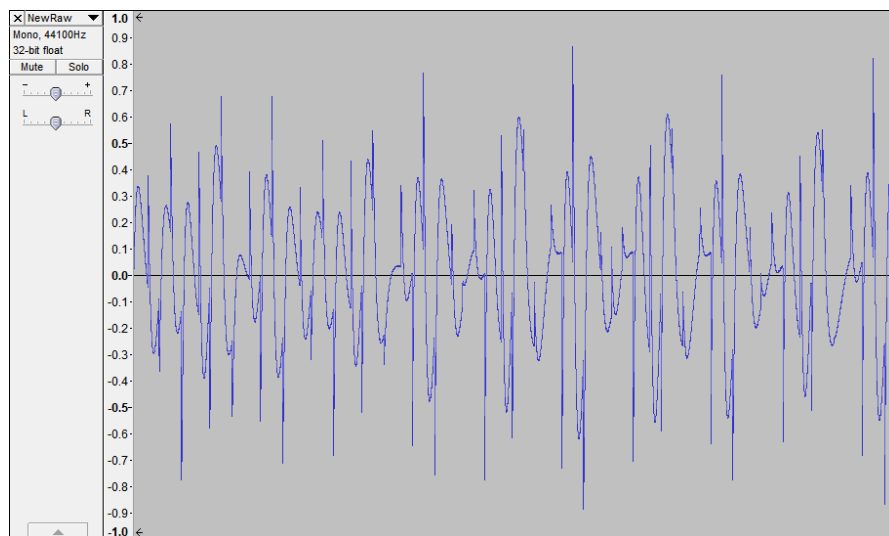


Gambar IV-11 Data mentah aplikasi smartphone ketika diberikan input sinyal sinusoidal 70Hz dari Arduino.

Permasalahan kemudian terlihat ketika data Arduino yang direkam di Audacity sebelumnya diperbesar, memperlihatkan sinyal pada Gambar IV-12. Dapat jelas terlihat bahwa output yang sangat lamban mengakibatkan sinyal kotak. Hal ini menunjukkan tidak bermasalah untuk PC yang digunakan, meski memang tetap terpengaruhi oleh HPF internal (dengan bukti ujung atas sinyal kotak menurun, bukan datar). Namun HPF smartphone yang digunakan terbukti memiliki titik cutoff yang lebih tinggi dari perkiraan, mengakibatkan sinyal-sinyal kotak tersebut mengalami *decay* dan membentuk *spike*. Simulasi penerapan internal HPF ini bisa dilihat pada Gambar IV-13, yang mana sangat serupa dengan kasus smartphone pada gambar sebelumnya.



Gambar IV-12 Perekaman output DAC Oximeter menggunakan audio jack komputer & program Audacity kemudian diperbesar.



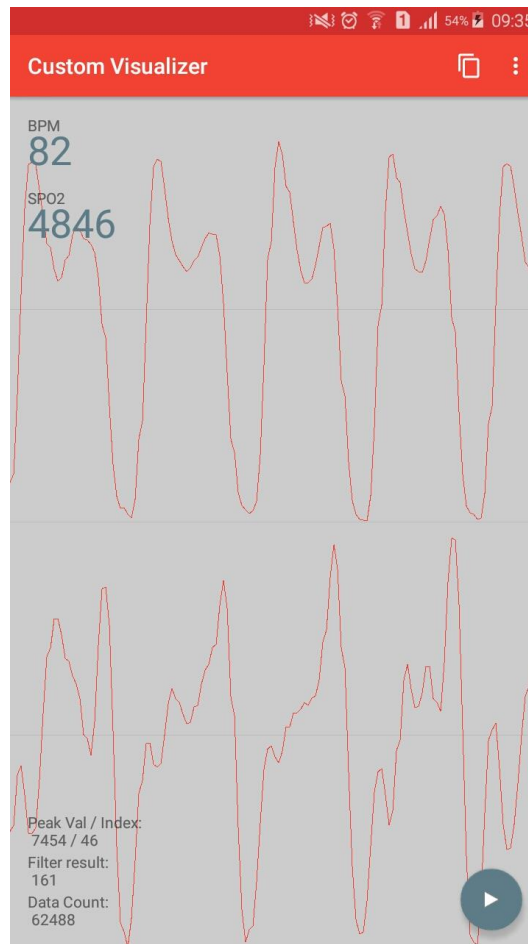
Gambar IV-13 Perekaman output DAC Oximeter menggunakan audio jack komputer & program Audacity dan diberikan filter high-pass dengan frekuensi cutoff relatif tinggi.

Tentunya karakteristik sudah dipahami, bahwa DAC akan mengeluarkan sinyal yang terkotak-kotak apabila jarang diupdate atau bahkan sinyal yang dikeluarkan melebihi kecepatan DAC. Namun efek yang sangat dominan seperti ini baru terlihat sekarang. Hal yang perlu difaktorkan adalah DAC yang digunakan memiliki kecepatan yang

sangat tinggi, sehingga menambah fungsi untuk mengupdate output seharusnya tidak berpengaruh besar. Sedangkan upaya-upaya yang dilakukan serta hasilnya antara lain sebagai berikut.

- Interpolasi linear untuk output sinusoidal metode polling: Tidak bisa dilakukan karena kecepatan loop sudah dekat dengan $\frac{1}{2}$ kecepatan carrier, sehingga segala cara untuk memberikan output lebih tinggi lagi secara polling tidak bisa dilakukan tanpa menggunakan fungsi delay.
- Interpolasi linear untuk sinyal output sinusoidal metode interrupt: Bisa diimplementasi namun output yang diperoleh sangat kacau karena sinkronisasi yang buruk. Malah menghasilkan komponen sinyal segitiga berfrekuensi rendah.
- Melakukan proses perhitungan modulasi (trigonometri) dengan interrupt, yang mana disetting lebih cepat dari sebelumnya agar frekuensi carrier bisa lebih tinggi: Malah mengakibatkan proses yang terhenti tanpa ada respon dari Arduino.
- Menggunakan carrier kotak, interrupt, dan meningkatkan frekuensi carrier: Bisa dilakukan dan mempercepat keseluruhan proses secara signifikan. Namun serial tidak bisa selancar kondisi tanpa interrupt, yang mana tidak masalah.

Dari banyak pengujian, opsi carrier kotak memberikan sinyal yang paling konsisten meski tetap ada distorsi yang muncul. Distorsi kali ini dimungkinkan bisa diatasi dengan pengaturan filter lebih lanjut, karena sifatnya merata dan tidak random. Sebagian contoh kekonsistenan ini bisa dilihat pada Gambar IV-14 (abaikan grafik bawah). Secara teknis, limitasi dari sinyal kotak adalah adanya alias utama pada titik $3x$, $5x$, dst frekuensi dasar. Hal ini lebih baik dibanding sinyal sawtooth yang memiliki alias utama pada $2x$, $3x$, $4x$, dst frekuensi dasar. Alhasil bisa digunakan jarak carrier yang lebih dekat apabila menggunakan carrier kotak.



Gambar IV-14 Salah satu pengujian modulasi-demodulasi menggunakan carrier kotak 100Hz (abaikan grafik bawah).

Sedangkan efek penggunaan carrier kotak pada laju keseluruhan adalah sebagai berikut.

- Laju satu buah loop tercepat (15/16) = $\sim 2,86\text{kHz} - 6,6\text{kHz}$
- Laju satu buah loop terlambat (1/16) = $\sim 300\text{Hz} - 370\text{Hz}$
- Laju loop rata-rata = $\sim 2,7\text{kHz} - 6,2\text{kHz}$
- Laju pengambilan data baru = $\sim 29\text{Hz} - 50\text{Hz}$

Peningkatan laju yang besar pada laju loop keseluruhan memungkinkan lebih banyak proses untuk dilaksanakan tanpa perlu takut frekuensi sampling bergeser di bawah desain.

Perlu ditekankan bahwa umumnya pengujian dilakukan untuk kedua kanal. Karakteristik beragam mulai dari yang bagus untuk kanal pertama terisolasi atau kanal kedua terisolasi, namun belum ada yang benar-benar sempurna untuk dua kanal sekaligus. Pembahasan mendalam akan ada pada subbab selanjutnya mengenai sistem² dari sisi Android. Namun untuk kasus sinyal carrier kotak, kebersihan data cukup tinggi jika dibandingkan metode lainnya yang sudah disebutkan, untuk ketiga kasus. Namun dalam kasus dua kanal sekaligus, salah satu sinyal mengalami distorsi yang sangat unik untuk kanal pertama. Kanal pertama layaknya terbalik secara vertikal, karena bukit kedua gelombang lebih tinggi dari bukit pertama secara dominan.

Pada kondisi ini, sistem dapat dikonklusikan memiliki diagram sistem digital yang ditunjukkan pada GAMBAR. Tentunya penggambaran di bawah ini kurang akurat karena adanya campur tangan interrupt untuk tahapan modulasi, membuat alur berjalan program tidak satu arah lagi. Sedangkan untuk tahapan² sebelumnya, dapat dipastikan bahwa memang sistem bisa cukup akurat digambarkan dalam diagram tersebut. Dari penggambaran ini, tentunya

IV.3 Android

Senasib dengan bagian Arduino, bagian transmisi untuk program Android juga banyak mengalami perubahan. Namun secara general perubahan-perubahan yang terjadi pada sisi ini ada pada filter yang digunakan, sehingga koefisien saja. Sedangkan perubahan secara substansial pada struktur program sering dilakukan namun dengan efek yang malah memburuk dari bentukan paling dasar (hal ini sudah di bahas pada bab sebelumnya, mengenai metode pengaliran data) – tidak banyak perubahan yang

benar-benar digunakan pada program akhir. Sedangkan permasalahan-permasalahan terbesar yang menjadi penghalang pada sisi ini antara lain:

- Efek pergeseran fasa oleh filter IIR berdampak sangat signifikan sehingga memerlukan implementasi filter FIR. Filter FIR akan memerlukan orde yang jauh lebih tinggi dibanding IIR dengan spesifikasi yang sama ($\sim 10x - 20x$).
- Frekuensi sampling yang disupport oleh kebanyakan Android adalah 44,1kHz – sangat tinggi jika dibandingkan band frekuensi kerja.[3] Frekuensi sampling yang tinggi akan mengakibatkan spesifikasi filter secara dominan membesar untuk suatu filter yang curam.
- Penurunan frekuensi sampling dapat dilakukan dengan cara downsampling, namun sangat mudah terkena distorsi dari efek aliasing apabila ada komponen sinyal dengan frekuensi lebih besar dari $\frac{1}{2}$ frekuensi sampling baru.

Alhasil secara general solusi yang diperoleh adalah kolaborasi antara LPF dengan downsampling. Dari sini diperoleh strategi yang serupa dengan strategi pada Arduino, hanya saja dalam skala yang jauh lebih besar karena besarnya rasio frekuensi data yang dibutuhkan terhadap rentang frekuensi keseluruhan (15Hz terhadap 22.050Hz). Belum lagi keperluan untuk menghilangkan suatu kanal, yang mana apabila dilihat oleh suatu kanal lainnya ekuivalen dengan suatu noise namun dengan daya yang sama dan jarak frekuensi yang tidak jauh (dalam kasus ini).

Alur filtering untuk skema yang sudah tersedia dalam LAMPIRAN C (bagian Android) sempat ditunjukkan pada bab sebelumnya, di Algoritma III-8. Potongan source code dengan informasi lebih lengkap mengenai karakteristik masing-masing tahapan akan dijabarkan pada Algoritma IV-6. Koefisien untuk masing-masing filter ada pada LAMPIRAN B.

```
//... Relevant variables  
  
// Filters, specified based on the note on Readme.md  
// LPF, 111  
private double[] b_aa1 = { /*... Koef ...*/};
```



```

// LPF, 61
private double[] b_aa2 = { /*... Koef ...*/ };
// LPF, 26
private double[] b_chA = { /*... Koef ...*/ };
// HPF, 28
private double[] b_chB = { /*... Koef ...*/ };
// LPF, 83
private double[] b_demod = { /*... Koef ...*/ };

// Downsampling rates with the data before/after
private int fsdev = 1; // Multiplier to compensate the difference in
sampling rate
// Raw
private final int DS0 = 1; // 44100 Hz (1/1 from before)
// First anti-aliasing result
private final int DS1 = 30; // 1470 Hz (1/30 from before, 1/30 total)
// Second anti-aliasing result
// Anti-aliased data
private final int DS2 = 3; // 490 Hz (1/3 from before, 1/90 total)
// Channel separator result
private final int DS3 = 2; // 245 Hz (1/2 from before, 1/180 total)
// Edge detector result
private final int DS4 = 5; // 49 Hz (1/5 from before, 1/900 total)
// Offset suppressor result
// Final data
private final int DS5 = 7; // 7 Hz (1/7 from before, 1/6300 total)
// Simplified for BPM calculation

//... Inside recording thread

recorder.read(sData, 0, BufferElements2Rec);

// Full scheme written on Readme.md
// Anti-aliasing, dual stage
filaa1.addArray(sData, n, DS0 / fsdev);
filaa2.addArray(filaa1.getBuffer(), n, DS1);

filChA.addArray(filaa2.getBuffer(), n, DS2); // HbO2
filChB.addArray(filaa2.getBuffer(), n, DS2); // Hb

// Rectify then clear carrier
filDemodA.addArray(filChA.getBuffer(), n, DS3);
filDemodB.addArray(filChB.getBuffer(), n, DS3);

// Iterate global index
if (n < (BufferElements2Rec * MAXINDEX)) {
// Since it'll be assured that for every loop, sData'll be fully processed
n += sData.length;
} else {
n = 1;
}
}

//...

```

Algoritma IV-6 Perpotongan source code dari sebagian aliran data input sebelum ditampilkan ke grafik.

Dengan pilihan modulasi adalah DSBAM, tentunya bisa dirasakan manfaat yang sangat besar dalam kompleksitas maupun cost keseluruhan sistem. Cara yang paling

dasar untuk mendemodulasi atau merekonstruksi ulang suatu sinyal DSBAM adalah dengan memberikan menerapkan suatu envelope detector. Suatu envelope detector hanyalah terdiri atas suatu rectifier serta suatu LPF, sehingga diciptakan suatu sinyal yang hanya merupakan “sampul” dari sinyal semula [2]. Sebagaimana disebutkan dalam sisi Arduino, inilah mengapa sinyal pesan harus diangkat dari titik negatif ketika dimodulasi. Sinyal yang menyentuh titik negatif ketika dimodulasi akan memiliki karakteristik yang sama layaknya sinyal positif, hanya saja ada beda fasa 180 derajat. Untuk mendeteksi karakteristik perpindahan fasa tersebut, dibutuhkan sistem yang lebih kompleks dalam demodulasi.

BAB V

KESIMPULAN DAN SARAN

V.1 Kesimpulan

Projek Pengembangan Program serta Sistem Transmisi untuk Pulse Oximeter merupakan kesatuan proses yang terdiri atas dua bagian utama, antarmuka aplikasi Android serta sistem transmisi data melalui audio jack. Keduanya tentu hal teknis yang cukup berjauhan, sehingga ada banyak sekali topik pembahasan, permasalahan, alternatif, optimasi, serta berbagai aspek-aspek teknis lainnya yang saling tegak lurus antara keduanya. Meski demikian ada banyak sekali hal yang diperoleh bahkan hanya sebatas didaparkannya proyek untuk mengembangkan suatu alat jadi dan jauh dari kondisi prototipe, sehingga dari sisi cara penggunaan, ketepatan, keamanan, daya tahan, dll ada tingkatan yang cukup berbeda dibanding peralatan yang hanya digunakan dalam lingkungan percobaan.

Dari banyaknya aspek yang perlu diperhatikan, tetap saja ada beberapa permasalahan yang menjadi fokus utama untuk sebagian besar pengujian. Dimulai dari efek HPF internal serta memenuhi keperluan pengiriman dua buah sinyal dalam satu kabel, yang mana merupakan sisi transmisi dari proyek. Solusi dipilih modulasi amplitudo dengan jenis DSBAM. Sedangkan pada sisi GUI, tidak ada permasalahan yang benar-benar besar. Sebab hampir keseluruhan aspek membutuhkan fokus yang merata, demi diperolehnya user experience yang baik. Aspek GUI menunjukkan sebagian kecil betapa besarnya transisi yang diperlukan dari suatu barang prototype menjadi suatu barang jadi.

V.2 Saran

Dalam proses pengujian alat, ada dua pengelompokan utama jika dilihat dari metode pengujiannya. Metode pertama adalah pengujian dengan cara nyata, sama dengan sebagaimana suatu alat akan digunakan. Metode kedua adalah pengujian dengan

lingkungan khusus, di mana data dan kondisinya dibuat sebatas menyerupai cara nyata. Selama proses pengerjaan proyek ini, menguji menggunakan Audacity secara mentah-mentah menunjukkan karakteristik yang cukup jauh dibandingkan dengan cara langsung menggunakan Arduino. Perbedaan tersebut tidak lain adalah efek rendahnya output rate sinyal sinusoidal pada sisi Arduino. Tentunya karakteristik serupa bisa dipenuhi juga melalui Audacity, namun permasalahan tersebut hanya bisa diketahui setelah melakukan pengujian nyata; yang mana terjadi lama setelah pengujian dengan Audacity. Dengan demikian, meski pengujian secara khusus (dalam kasus ini dengan Audacity) memberikan kemudahan secara substansial (yaitu menyintesis beragam sinyal untuk Audacity) dalam pengujian, pengujian nyata baiknya dilakukan jauh lebih awal demi mengetahui seberapa besar permasalahan praktis yang akan ditemui.

DAFTAR PUSTAKA

- [1] "What is Modulation?," WikiBooks, [Online]. Available: https://en.wikibooks.org/wiki/Communication_Systems/What_is_Modulation%3F. [Diakses 30 07 2018].
- [2] "Amplitude Modulation," WikiBooks, [Online]. Available: https://en.wikibooks.org/wiki/Communication_Systems/Amplitude_Modulation. [Diakses 30 07 2018].
- [3] "Basics of Microcontrollers," Circuits Today, [Online]. Available: <http://www.circuitstoday.com/basics-of-microcontrollers>. [Diakses 30 07 2018].
- [4] "How pulse oximeters work explained simply.," How Equipment Works, [Online]. Available: https://www.howequipmentworks.com/pulse_oximeter/. [Diakses 30 07 2018].
- [5] "Pulse & SpO2," Real First Aid, [Online]. Available: <http://www.realfirstaid.co.uk/pulse/>. [Diakses 30 07 2018].
- [6] "Custom-Grapher: Graph data from audio source.," GitHub, [Online]. Available: <https://github.com/ArsenicBismuth/Custom-Grapher>. [Diakses 16 07 2018].
- [7] "A Beginner's Guide to Digital Signal Processing (DSP)," Analog Devices, [Online]. Available: <http://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html>. [Diakses 30 07 2018].
- [8] "MediaRecorder," Android Developers, [Online]. Available: <https://developer.android.com/reference/android/media/MediaRecorder>. [Diakses 20 07 2018].
- [9] "AudioRecord," Android Developers, [Online]. Available: <https://developer.android.com/reference/android/media/AudioRecord>. [Diakses 20 07 2018].
- [10] "Pulse Oximeter Pleth (Plethysmograph)," Amperor Direct, [Online]. Available: <https://www.amperordirect.com/pc/help-pulse-oximeter/z-what-is-oximeter-plethysmograph.html>. [Diakses 21 07 2018].
- [11] "Buy pulse oximeter and get free shipping," AliExpress, [Online]. Available: <https://www.aliexpress.com/wholesale?SearchText=pulse+oximeter>. [Diakses 21 07 2018].
- [12] "SimpleWaveform: Highly customized to draw waveform or bar chart.," GitHub, [Online]. Available: <https://github.com/maxyou/SimpleWaveform>. [Diakses 21 07 2018].
- [13] [Online].
- [14] "Mobile Phones with USB OTG Support," PriceBaba, [Online]. Available: <https://pricebaba.com/mobile/pricelist/phones-with-otg-support-in-india>. [Diakses 18 07 2018].

- [15] "Price Lists of Mobile Phones in India," PriceBaba, [Online]. Available: <https://pricebaba.com/search?category=MOBILE&from=Mobile%20Phones>. [Diakses 18 07 2018].
- [16] "Implementasi Pulse Oximeter melalui Rangkaian Penguat Sederhana beserta Digital Filtering," Unit Robotika ITB, [Online]. Available: <http://robotika.unit.itb.ac.id/main/1321-implementasi-pulse-oximeter-seperti-sensor-detak-jantung-melalui-rangkaian-penguat-sederhana-beserta-digital-filtering.html>. [Diakses 23 07 2018].
- [17] "Nyquist Sampling Theorem," UC San Diego, [Online]. Available: http://musicweb.ucsd.edu/~trsmyth/digitalAudio170/Nyquist_Sampling_Theorem.html. [Diakses 24 07 2018].
- [18] "Optimal Filter Bandwidth for Pulse Oximetry," Scitation, [Online]. Available: <https://aip.scitation.org/doi/full/10.1063/1.4759491>. [Diakses 24 07 2018].
- [19] "Interpolation," Wikipedia, [Online]. Available: <https://en.wikipedia.org/wiki/Interpolation>. [Diakses 28 07 2018].

LAMPIRAN

LAMPIRAN A

Source code Arduino

A.1 Program utama: Oximeter.ino

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
#include "Filter.h"
#include <TimerOne.h>

Adafruit_MCP4725 dac;
#define SERIAL_BUFFER_SIZE 32

/* Due to SRAM limitation, combined order all filters of more than
~2x60 will induce some array to be filled with zeroes.
 * Meanwhile combined order of more than ~2x70 will make the MCU
not even working.
 * Alternative would be using PROGMEM, which is the flash memory.
 * SRAM: 2kB, thus for every order in a filter, space needed for
even
 * the array = 4 byte * 2 (ch) * 2 (coefs & inputs) * 2 (if iir).
 */

// Filter Coefficients (from MATLAB
// HPF
float bNorm[51] = {0.00412053114641298, -0.0312793191356909, -
0.0115236109767417, -0.0100213992087269, -0.0108385904761499, -
0.0119734717946732, -0.0131685337169099, -0.0143842189071186, -
0.0156078845365078, -0.0168299462147875, -0.0180409785596737, -
0.0192314427835679, -0.0203917450600907, -0.0215123386063274, -
0.0225838323470475, -0.0235970999596655, -0.0245433871199924, -
0.0254144153854427, -0.0262024812734020, -0.0269005491468212, -
0.0275023365897874, -0.0280023910540817, -0.0283961566788859, -
0.0286800303246559, -0.0288514060155109, 0.971091292850008, -
0.0288514060155109, -0.0286800303246559, -0.0283961566788859, -
0.0280023910540817, -0.0275023365897874, -0.0269005491468212, -
0.0262024812734020, -0.0254144153854427, -0.0245433871199924, -
0.0235970999596655, -0.0225838323470475, -0.0215123386063274, -
0.0203917450600907, -0.0192314427835679, -0.0180409785596737, -
0.0168299462147875, -0.0156078845365078, -0.0143842189071186, -
0.0131685337169099, -0.0119734717946732, -0.0108385904761499, -
0.0100213992087269, -0.0115236109767417, -0.0312793191356909,
0.00412053114641298};
// Difference
//float bNorm[] = {0.5, -0.5};

// Down/upsampling rate, specified relative to previous process
#define DS0 1 // Downsampling rate FS0 = FS / DS1, for input
#define DS1 4 // Downsampling due to switching of 4 states

// Remember b's passed by reference
//Filter filNoiseA(sizeof(bNoise)/sizeof(float), bNoise);
//Filter filNoiseB(sizeof(bNoise)/sizeof(float), bNoise);
```



```

#define DS2 1 // Downsampling for precise offset-suppression
Filter filNormA(sizeof(bNorm)/sizeof(float), bNorm);
Filter filNormB(sizeof(bNorm)/sizeof(float), bNorm);

#define US3 4 // Upsampling for interpolation

#define MAXINDEX (DS0 * DS1 * DS2) // Used to reset the
global iterator, avoiding overflow

/* Both FS and USO can be focused to obtain:
 * - FS: More detailed data, including the carrier, but less
room for interpolation
 * - USO: Less blocky output, but will reduce loop speed FS and
thus output triangular data rather than sinusoidal
 */
#define FS 990 // Loop rate (Hz)
#define FSI 1000 // Interrupt rate (Hz), somehow multiple of
carrier freq produce a big low-freq noise
#define FSC (FS / DS0 / DS1 / DS2 * US3) // Data rate at the time
for modulation

// Carriers
#define FCA 100
#define FCB 200

// Value of square carrier iterator until they're switched
#define FLIPA (FSI / FCA)
#define FLIPB (FSI / FCB)

// Negative limit for modulation
#define NEG 100

// Bottom limit at output
#define BOTTOM 10

// Var
int input;
int inA;
int inB;
int inOff;
float dsA = 0;
float dsB = 0;
float filA = 0;
float filB = 0;
float interA = 0;
float interB = 0;
float distA = 0;
float distB = 0;
float chA = 0;
float chB = 0;
short carrA = 1;
short carrB = 1;
int output = 0;
int n = 1; // Global index
int mA = 1; // Index used for cosine
int mB = 1; // Index used for cosine
unsigned long t0, t1, t2, t3, t4, t5, t6;

```

```

void result() {
    sei(); // To enable I2C inside ISR

    // Modulation
    carrA = (mA <= FLIPA / 2) ? 1 : -1;
    carrB = (mB <= FLIPB / 2) ? 1 : -1;

    // Currently MESSAGE is [-100, 100], converted to [0, 200]
    // Thus modulated at [-200, 200]
    chA = ((interA > -NEG) ? (NEG + interA) : (0)) * carrA;
    chB = ((interB > -NEG) ? (NEG + interB) : (0)) * carrB;

    // Combine, creating wave with range of [-200, 200]
    // output = (chA + chB) / 2;
    output = chA;

    // Final output handling
    // Amplify & offset to maximize range & avoid negatives
    /* Data about the range:
    * - For a full range wave [-1, 1] in Audacity, phone data
    is good at ~50% vol
    * - For a half-range wave [-.5,.5] in Audacity, phone data
    is good at ~80-90% vol
    * - DAC sine with amp of 512 (4096 / 8, or 0,625 Vp) will
    result in a very little clipping
    * - Currently combined output is [0, 400] for dual ch
    * - Audiacity full-range wave at 10% vol playback equals to
    a full-range wave at recording
    * - Optimal conclusion: 50% Audacity playback => 5% Audacity
    record => 25.6 Arduino DAC (3mVp)
    */
    int temp = (output + 2*NEG) / 8 + BOTTOM; // [-200, 200]
    => [0, 400]

    if (temp > BOTTOM) {
        dac.setVoltage(temp, false);
    } else {
        dac.setVoltage(BOTTOM, false);
    }

    // Carrier iterator
    if (mA < FLIPA) {
        mA++;
    } else {
        mA = 1;
    }

    if (mB < FLIPB) {
        mB++;
    } else {
        mB = 1;
    }

    // Only max 1 or 2 serial prints may be activated
    Serial.println(fila);
}

```

```

void setup() {
    Serial.begin(250000);
    pinMode(5, OUTPUT);
    dac.begin(0x62);

    Timer1.initialize(1000000 / FSI);    // Time in us
    Timer1.attachInterrupt(result);      // Do modulation & final
output at 1kHz
}

void loop() {
    t0 = micros();
    input = analogRead(A2);

    t1 = micros();

    // Data mapping & downsampling, add data only every x indexes
    // TODO electrical settings on switching
    if (modulo(n, DS0) == 0) {
        if (modulo(n, DS0 * DS1) == 0) {
            // HbO2
            inA = input;
        } else if (modulo(n, DS0 * DS1) == 1) {
            // Off
            inOff = input;
        } else if (modulo(n, DS0 * DS1) == 2) {
            // Hb
            inB = input;
        } else if (modulo(n, DS0 * DS1) == 3) {
            // Off
            inOff = input;
        }
    }

    t2 = micros();

    // Offset suppression
    if (modulo(n, (DS0 * DS1 * DS2)) == 0) {
        dsA = inA;
        dsB = inB;

        filNormA.addVal(dsA);
        filNormB.addVal(dsB);

        // Set interpolation start as the prev result
        interA = filA;
        interB = filB;

        filA = filNormA.getVal();
        filB = filNormB.getVal();

        distA = filA - interA;
        distB = filB - interB;
    }

    t3 = micros();
}

```

```

    // Linear interpolation
    if (n % (DS0 * DS1 * DS2 / US3) == 0) {
        interA += distA / US3;
        interB += distB / US3;
    }

    t4 = micros();

    // Global iterator
    if (n < MAXINDEX) {
        n++;
    } else {
        n = 1;
    }

    t6 = micros();
}

int modulo(int dividend, int divisor) {
    if (dividend == 1) {
        return divisor > 1;
    } else if (divisor == 1) {
        return 0;
    } else if (divisor <= 4) {
        return dividend % divisor;
    } else {
        int result = dividend;
        for (result; result >= divisor; result -= divisor);
        return result;
    }
}

```

A.2 Program filter: Filter.cpp

```

#include "Arduino.h"
#include "Filter.h"
using namespace std;

// Usage must addVal then getVal immediately since there's no
buffer

// IIR if a is given
Filter::Filter(int _order, float *_b, float *_a) {
    // Using "_" to distinguish member and not and avoiding using
    "_".
    order = _order;
    a = _a;
    b = _b;

    int aSize = sizeof(a) / sizeof(float);
    int bSize = sizeof(b) / sizeof(float);

    if (aSize != 0) iir = true;
    init();
}

```

```

}

// FIR
Filter::Filter(int _order, float *_b) {
    order = _order;
    b = _b;
    iir = false;

    int bSize = sizeof(b) / sizeof(float);

    init();
}

void Filter::init() {
    // The last two brackets isn't necessary, just used to s them
    if (iir) outputs = new float[1]();
    else outputs = new float[order]();
    inputs = new float[order]();
}

int st = 0; // The array index for x[n] in circular buffer

// Add value and simultanteously compute the result
void Filter::addVal(float value) {

    // Shifting the start of circular buffer FORWARD, since
    // the prev x(n) is x(n-1) now.
    st = modulo((st + 1), order);

    // Get input and store to the new start, x[n]
    inputs[st] = value;

    // Calculate the convolution from koefs
    outputs[st] = 0;

    for (int i = 0; i < order; i++) {
        // Decrement, x[n-i] * b[i]
        int j = modulo((order + st - i), order);

        outputs[st] += inputs[j] * b[i];

        if ((iir) && (i != 0)) {
            outputs[st] -= outputs[j] * a[i];
        }
    }
}

// Get individual output
float Filter::getVal() {
    return outputs[st];
}

// Alternate modulo function
// Good enough if dividend is sure to be not far bigger than
// divisor
int Filter::modulo(int dividend, int divisor) {

```

```

    if (dividend == 1) {
        return divisor > 1;
    } else if (divisor == 1) {
        return 0;
    } else if (divisor <= 4) {
        return dividend % divisor;
    } else {
        int result = dividend;
        for (result; result >= divisor; result -= divisor);
        return result;
    }
}

```

A.3 Header filter: Filter.h

```

#ifndef Filter_h
#define Filter_h
#include "Arduino.h"

class Filter
{
private:
    int order = 32;
    bool iir = false;

    // Pointers are used because we can't declare anything
    yet.
    // It's only here to be recognized as a class member.
    float* inputs;
    float* outputs;
    float* a;
    float* b;

    void init();
    int modulo(int dividend, int divisor);

public:
    Filter(int order, float *b, float *a);
    Filter(int order, float *b);
    void addVal(float value); // Add value and compute
    float getVal();
};

#endif

```

LAMPIRAN B

Potongan source code aplikasi Android

B.1 Koefisien filter-filter pada program utama: MainActivity.java

```
// Filters, specified based on the note on Readme.md
// LPF, 111
private double[] b_aa1 = {-0.000331550026303021, -0.000372265324462671, -
0.000418866922856345, -0.000472627729976633, -0.000534420609562197, -
0.000604643759698580, -0.000683155014016682, -0.000769217007982752, -
0.000861454956952813, -0.000957828553012607, -0.00105561920917520, -
0.00115143356771268, -0.00124122385051423, -0.00132032527032998, -
0.00138351035005506, -0.00142505962069058, -0.00143884779535902, -
0.00141844415486750, -0.00135722553778236, -0.00124850001247768, -
0.00108563902734299, -0.000862215594841029, -0.000572145871167061, -
0.000209831350731869, 0.000229701192607173, 0.000750664915009880,
0.00135638613018885, 0.00204918833262761, 0.00283028964561548,
0.00369971611828263, 0.00465623303784079, 0.00569729611682538,
0.00681902406595216, 0.00801619367785358, 0.00928225813287758,
0.0106093888034130, 0.0119885403864577, 0.0134095387442971,
0.0148611903892978, 0.0163314121199594, 0.0178073789103221,
0.0192756877819606, 0.0207225350548884, 0.0221339040877727,
0.0234957603850154, 0.0247942507735458, 0.0260159032395069,
0.0271478239670524, 0.0281778881395663, 0.0290949211477954,
0.0298888669983320, 0.0305509409269555, 0.0310737634906417,
0.0314514737344825, 0.0316798193991326, 0.0317562225435901,
0.0316798193991326, 0.0314514737344825, 0.0310737634906417,
0.0305509409269555, 0.0298888669983320, 0.0290949211477954,
0.0281778881395663, 0.0271478239670524, 0.0260159032395069,
0.0247942507735458, 0.0234957603850154, 0.0221339040877727,
0.0207225350548884, 0.0192756877819606, 0.0178073789103221,
0.0163314121199594, 0.0148611903892978, 0.0134095387442971,
0.0119885403864577, 0.0106093888034130, 0.00928225813287758,
0.00801619367785358, 0.00681902406595216, 0.00569729611682538,
0.00465623303784079, 0.00369971611828263, 0.00283028964561548,
0.00204918833262761, 0.00135638613018885, 0.000750664915009880,
0.000229701192607173, -0.000209831350731869, -0.000572145871167061, -
0.000862215594841029, -0.00108563902734299, -0.00124850001247768, -
0.00135722553778236, -0.00141844415486750, -0.00143884779535902, -
0.00142505962069058, -0.00138351035005506, -0.00132032527032998, -
0.00124122385051423, -0.00115143356771268, -0.00105561920917520, -
0.000957828553012607, -0.000861454956952813, -0.000769217007982752, -
0.000683155014016682, -0.000604643759698580, -0.000534420609562197, -
0.000472627729976633, -0.000418866922856345, -0.000372265324462671, -
0.000331550026303021};
// LPF, 61
private double[] b_aa2 = {-0.000508508009580982, -0.000903019197774497, -
0.000444894280259791, 0.000660373417926351, 0.00146789782188540,
0.000886317422313322, -0.00109419558995878, -0.00274535224793622, -
0.00183573312291301, 0.00177603631600603, 0.00491889824185567,
0.00354183667911377, -0.00264446143357028, -0.00822680861941409, -
0.00635400327967241, 0.00361667802495594, 0.0130566204516626,
0.0108449154075161, -0.00459713841200943, -0.0202030322638191, -
0.0181732743674832, 0.00548767495926264, 0.0317081384961672,
0.0314018898476090, -0.00619804290227997, -0.0547135280792258, -
0.0627723490822906, 0.00665569055734947, 0.139912766990054,
0.271950880625500, 0.327055451258022, 0.271950880625500,
0.139912766990054, 0.00665569055734947, -0.0627723490822906, -
0.0547135280792258, -0.00619804290227997, 0.0314018898476090,
```

```

0.0317081384961672, 0.00548767495926264, -0.0181732743674832, -
0.0202030322638191, -0.00459713841200943, 0.0108449154075161,
0.0130566204516626, 0.00361667802495594, -0.00635400327967241, -
0.00822680861941409, -0.00264446143357028, 0.00354183667911377,
0.00491889824185567, 0.00177603631600603, -0.00183573312291301, -
0.00274535224793622, -0.00109419558995878, 0.000886317422313322,
0.00146789782188540, 0.000660373417926351, -0.000444894280259791, -
0.000903019197774497, -0.000508508009580982};
// LPF, 26
private double[] b_chA = {-0.00572225374043716, -0.0180335270633584, -
0.0145649419292811, 0.0116973249207995, 0.0183777359746803, -
0.0181837880487599, -0.0285567261790708, 0.0300597733591670,
0.0470324226228558, -0.0551731175089687, -0.0904225921812429,
0.143330292058622, 0.453920753391290, 0.453920753391290,
0.143330292058622, -0.0904225921812429, -0.0551731175089687,
0.0470324226228558, 0.0300597733591670, -0.0285567261790708, -
0.0181837880487599, 0.0183777359746803, 0.0116973249207995, -
0.0145649419292811, -0.0180335270633584, -0.00572225374043716};
// HPF, 28
private double[] b_chB = {-0.00164282664975775, 0.00207019168002476,
0.00111890391620409, -0.00977838741500030, 0.0204291864474586, -
0.0238175943647322, 0.0107128541634235, 0.0181835143120806, -
0.0461294646359265, 0.0458759643436551, 0.00326089656488181, -
0.0973168723245041, 0.202989178180561, -0.272918435001176,
0.272918435001176, -0.202989178180561, 0.0973168723245041, -
0.00326089656488181, -0.0458759643436551, 0.0461294646359265, -
0.0181835143120806, -0.0107128541634235, 0.0238175943647322, -
0.0204291864474586, 0.00977838741500030, -0.00111890391620409, -
0.00207019168002476, 0.00164282664975775};
// LPF, 83
private double[] b_demod = {0.00521307083722403, 0.000698382709468956,
0.000594567165180754, 0.000375656318506526, 3.18131431167444e-05, -
0.000438977696559757, -0.00103735715375136, -0.00175457525810610, -
0.00258014129427819, -0.00349079605439242, -0.00446055075397255, -
0.00545546289730373, -0.00643439860207394, -0.00735233193193724, -
0.00816002725051333, -0.00880412495686340, -0.00923301280954544, -
0.00939124400075369, -0.00923139737922087, -0.00870375195193797, -
0.00777445615697063, -0.00640778349255515, -0.00458841946948496, -
0.00230972281863533, 0.000425654267619883, 0.00359756997617825,
0.00717240545217001, 0.0111075306342898, 0.0153409901769187,
0.0198025682849520, 0.0243961979307804, 0.0290575741777475,
0.0336733021199335, 0.0381445832112929, 0.0423816387715489,
0.0462793701104400, 0.0497533865671056, 0.0527158338575922,
0.0551003969180422, 0.0568449034965347, 0.0579099127475191,
0.0582673562111828, 0.0579099127475191, 0.0568449034965347,
0.0551003969180422, 0.0527158338575922, 0.0497533865671056,
0.0462793701104400, 0.0423816387715489, 0.0381445832112929,
0.0336733021199335, 0.0290575741777475, 0.0243961979307804,
0.0198025682849520, 0.0153409901769187, 0.0111075306342898,
0.00717240545217001, 0.00359756997617825, 0.000425654267619883, -
0.00230972281863533, -0.00458841946948496, -0.00640778349255515, -
0.00777445615697063, -0.00870375195193797, -0.00923139737922087, -
0.00939124400075369, -0.00923301280954544, -0.00880412495686340, -
0.00816002725051333, -0.00735233193193724, -0.00643439860207394, -
0.00545546289730373, -0.00446055075397255, -0.00349079605439242, -
0.00258014129427819, -0.00175457525810610, -0.00103735715375136, -
0.000438977696559757, 3.18131431167444e-05, 0.000375656318506526,
0.000594567165180754, 0.000698382709468956, 0.00521307083722403};
// HPF, 51
private double[] b_last1 = {0.00412053114641298, -0.0312793191356909, -
0.0115236109767417, -0.0100213992087269, -0.0108385904761499, -
0.0119734717946732, -0.0131685337169099, -0.0143842189071186, -
0.0156078845365078, -0.0168299462147875, -0.0180409785596737, -
0.0192314427835679, -0.0203917450600907, -0.0215123386063274, -

```



```

0.0225838323470475, -0.0235970999596655, -0.0245433871199924, -
0.0254144153854427, -0.0262024812734020, -0.0269005491468212, -
0.0275023365897874, -0.0280023910540817, -0.0283961566788859, -
0.0286800303246559, -0.0288514060155109, 0.971091292850008, -
0.0288514060155109, -0.0286800303246559, -0.0283961566788859, -
0.0280023910540817, -0.0275023365897874, -0.0269005491468212, -
0.0262024812734020, -0.0254144153854427, -0.0245433871199924, -
0.0235970999596655, -0.0225838323470475, -0.0215123386063274, -
0.0203917450600907, -0.0192314427835679, -0.0180409785596737, -
0.0168299462147875, -0.0156078845365078, -0.0143842189071186, -
0.0131685337169099, -0.0119734717946732, -0.0108385904761499, -
0.0100213992087269, -0.0115236109767417, -0.0312793191356909,
0.00412053114641298};

```

B.2 Mode-mode grafik kustom: SimpleWaveform.java

```

public final static int MODE_AMP_ORIGIN = 1;
public final static int MODE_AMP_ABSOLUTE = 2;
public int modeAmp;
public final static int MODE_HEIGHT_PX = 1;
public final static int MODE_HEIGHT_PERCENT = 2;
public int modeHeight;
public final static int MODE_ZERO_TOP = 1;
public final static int MODE_ZERO_CENTER = 2;
public final static int MODE_ZERO_BOTTOM = 3;
public int modeZero;
public final static int MODE_PEAK_ORIGIN = 1;
public final static int MODE_PEAK_PARALLEL = 2;
public final static int MODE_PEAK_CROSS_TOP_BOTTOM = 3;
public final static int MODE_PEAK_CROSS_BOTTOM_TOP = 4;
public final static int MODE_PEAK_CROSS_TURN_TOP_BOTTOM = 5;
public int modePeak;
public final static int MODE_DIRECTION_LEFT_RIGHT = 1;
public final static int MODE_DIRECTION_RIGHT_LEFT = 2;
public int modeDirection;
public final static int MODE_AXIS_OVER_AMP = 1; // Specify whether
axis drawn priority over amplitude
public final static int MODE_AXIS_UNDER_AMP = 2;
public int modePriority;
public final static int MODE_NORMAL_NONE = 1;
public final static int MODE_NORMAL_VALUE = 2; // Normalize full
height to a value (variable normalMax)
public final static int MODE_NORMAL_MAX = 3; // Normalize full
height based on the data list
public final static int MODE_NORMAL_VALUE_MAX = 4; // Use
MODE_NORMAL_VALUE if range is too small, otherwise NORMAL_MAX
public int modeNormal; // Setting how to
scale the full height range

```

LAMPIRAN C

Skema teknis keseluruhan sistem

Skema berikut merupakan info yang terpasang pada file keterangan README.md repository GitHub, sehingga akan ada beberapa simbol untuk formatting yang memang kurang cocok apabila dibaca secara langsung.

```
# CustomGrapher
### Full Scheme
**MCU** [Fs = ~200Hz]
Obtain HbO2 and Hb signal from sensor and transmit them both through
a single signal.
Sampling may be done through interrupt for more accurate rate.
Another way would be
doing it directly, and design filters which with cutoff deviation in
mind.
```

Everything with "Test" tag shows parameter very likely to change in current implementation.

Current implementation specified as:

- Single signal HbO2
- No diode switching, but data input switching implemented

Downsample specs are written respective to previous condition.

[TODO] Test 2 hi-freq sinusoids, one is half amplitude the other

1. Read analog data for Hb and HbO2 alternately
 - 4 states for each cycle: R, off, IR, off
 - The off state can be used to inspect offset
 - Due to low order of the analog system, effect from previous states would be short but still exist
2. Downsample by 4x => 50Hz (Simulating switching)
 - Since there're 4 input states, input downsampled 1x ==> 200Hz
3. [Removed] Noise filtering (Too many memory)
 - FIR, 12 \ ?, -30dB, 11th order => 12 \ 18
4. Downsample by 1x => 50Hz
4. Offset filtering
 - FIR, ? / 0.6, -30dB, 51st order => 0.11 / 0.6 @25Hz (Offset to 0.22 / 1.2 @50Hz)
5. Amplify 1x for better SNR
 - Signal Vpp must be less than 1/2 audio jack range for modulation purpose
6. Upsample to 100Hz & interpolate => Test: 4x prev or 1x total
 - Interpolation receives at min 2 data, and values as small as the interpolation rate to be created
6. Modulate each using DSBAM square carrier
 - HbO2, ch1 (70Hz)
 - Hb, ch2 (100Hz)
 - Modulation index 100% (signal zero at 1/2 modulated signal peak)
7. Decrease 8x so that each has amplitude < half the full range

8. Mix ch1 & ch2 and transmit

Testing rate specs for the scheme above:

- Common single loop speed (15/16) = ~2.86kHz - 6.6kHz
- Slowest single loop speed (1/16) = ~300Hz - 370Hz
- Average loop speed = ~2.7kHz - 6.2kHz
- New data sampling rate = ~29Hz - 50Hz

****Phone**** (Fs = 44100Hz)

Receive a single signal containing two channels through the audio jack.

1. Anti-aliasing filter, multi stage
 - FIR, -6dB @ 700Hz, 111th order => -0.3dB @ 210Hz
 - Downsample by 30 => 1470Hz
 - FIR, -6dB @ 240Hz, 61st order => -0.3dB @ 210Hz
1. Downsample by 3x => 490Hz
2. Separate channels
 - Separate HbO2, ch1 (100Hz): FIR, 110 \ 150, -60dB => 26th order
 - Separate Hb, ch2 (200Hz): FIR, 150 / 190, -60dB => 28th order
3. Downsample by 2x => 245Hz
4. Demodulate each
 - Rectify (actually requires the sampling rate to be more than 2x[2x] of all the carriers)
 - FIR, 5 \ 10, -40dB => 83th order
5. Downsample by 5x => 49Hz
6. Noise & offset filtering
 - Alternatives:
 - FIR, 0.2 / 0.4 8 \ 12, -60dB => 698th order => (X)
 - FIR, 0.1 / 0.6 8 \ 12, -30dB => 165th order
 - FIR, 0.1 / 0.6 8 \ 12, -10dB => 85th order => (X) Very big offset
 - Newer:
 - Offset: Copy Arduino Filter
7. Downsample by 7x => 7Hz
8. [Cancelled] BPM Calculation
 - FIR, ? \ 2, -30dB, 30th order => 1 \ 2
 - Check for rising zero-crossing, averaged at min every 4 wave