PERANCANGAN DAN IMPLEMENTASI PENGOLAHAN SINYAL RADAR UNTUK PENGUKURAN DOPPLER, RANGE DAN SAR IMAGING MENGGUNAKAN RASPBERRY PI

DESIGN AND IMPLEMENTATION OF RADAR SIGNAL PROCESSING FOR DOPPLER, RANGE AND SAR IMAGING USING RASPBERRY PI

Lazuardi Rea Rizkina[1], Edwar[2], Levy Olivia Nur[3]

[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Telkom University

[1]lazuardi.rea@gmail.com, [2]edwarm@telkomuniversity.ac.id, [3]levyolivia@telkomuniversity.ac.id,

Abstrak

Perkembangan teknologi serta pengolahan sinyal digital mendorong penerapan radar yang lebih fleksibel dengan miniaturisasi perangkat radar. Merujuk radar penelitian sebelumnya yang berbasis *laptop*, penelitian ini merancang pengolahan sinyal radar berbasis *Raspberry Pi* untuk pengolahan *Doppler*, *Range* dan SAR *Imaging*. Pengukuran *range* diperoleh melalui perhitungan *delay* dua arah dengan penanda waktu untuk membedakan sinyal kirim dan terima dari target bergerak. *Delay* ini menimbulkan pergeseran *doppler* yang menunjukan kecepatan radial target. Pemetaan gambar SAR dilakukan manual dengan mengambil *range* dari posisi berbeda. Menggunakan *Raspberry Pi* sebagai *platform* pengolahan sinyal dengan *software* Octave, parameter yang digunakan adalah waktu komputasi serta konsumsi memori sebagai tolak ukur performansi program.

Hasil perancangan dan optimasi didapatkan waktu proses pengolahan DTI selama 34 detik dengan konsumsi memori 36,17%. Sedangkan pengolahan RTI diperoleh waktu proses 3 menit 37 detik dan konsumsi memori 21,71% dengan perubahan visual pada *2-pulse magnitude only canceler clutter rejection*. Pengolahan SAR *Imaging* didapatkan waktu proses selama 15 menit 8 detik dengan konsumsi memori 10,14%. Parameter dimensi IFFT 2D diturunkan menjadi ¼ dan ½ nilai asli dengan waktu proses 11 menit 31 detik yang konsumsi memorinya sebesar 7,82% Dimensi sinyal sebelum FFT diuji dengan nilai 1024 mengalami distorsi dengan waktu komputasi 11 menit 29 detik dan konsumsi memori 7,82%. Fungsi *window* Hamming, Bartlett dan Blackman diuji dengan Blackman menghasilkan gambar yang lebih jernih.

Kata Kunci: Pengolahan Sinyal, Radar, *Octave*, *Raspberry Pi*, *Doppler*, *Range*, SAR *Imaging*

Abstract

Technological and digital signal processing advancement allows more flexible radar implementation with miniaturization of radar devices. Referring to previous research which based on laptop, this research designed and implemented a Raspberry Pi based radar capable of processing Doppler, Range and SAR Imaging. Range is computed by measuring two-way delay with timing marks of transmitted and received signal of the target. This delay causes doppler shifts at received signal containing radial velocity of moving target. SAR Imaging is done manually by measuring range from different positions then combined into SAR Image. Raspberry Pi function as processing platform with Octave as the processing software. Processing time and memory consumption measured as benchmarking parameters.

Implementation and optimization results for DTI are 34 seconds with 36.17% memory consumption. As for RTI, processing time measured at 15 minutes 8 seconds with 10.14% memory consumption with visual distortion for 2-pulse magnitude only canceler clutter rejection. SAR imaging yield processing time at 15 minutes 31 seconds with 7.82% memory consumption. IFFT 2D dimension reduced to ¼ and ½ of original value resulting 11 minutes 31 seconds processing time at 7.82%. Signal dimension for FFT tested at 1024 and 2048 with processing time 11 minutes 29 seconds and 7.82% memory consumption. Hamming, Bartlett and Blackman window functions tested with Blackman generates clearer image.

Keyword: Signal Processing, Radar, Octave, Raspberry Pi, Doppler, Range, SAR Imaging

1. Pendahuluan

Perkembangan teknologi serta pengolahan sinyal digital kini memungkinkan penerapan sistem radar yang lebih fleksibel untuk diaplikasikan pada kebutuhan nonmiliter. Hal ini mendorong miniaturisasi perangkat radar untuk diterapkan pada *platform* yang lebih kecil seperti pada mobil sebagai *sensor* bagi sistem kemudi otomatis.

Sehingga, pada penelitian ini, dirancang program pengolahan sinyal radar yang disesuaikan dengan *resource* perangkat *Raspberry Pi*. Bahan referensi yang digunakan merujuk pada penelitian sebelumnya [1] yang menggunakan *laptop* sebagai *platform* pengolahan sinyal. Hasil pemindaian disimpan dalam *format* .wav untuk diproses pada tiga *mode* percobaan, yaitu *doppler* terhadap waktu, *range* terhadap waktu dan SAR *Imaging*.

Perangkat lunak *Octave* versi 4.2.1 digunakan sebagai perangkat lunak pengolahan sinyal pada *Raspberry PI* 3 berbasis sistem operasi *Raspbian* "*stretch*". Pengolahan *doppler* dan *range* dilakukan dengan mengambil *range profile* pada target bergerak dimana pada pengolahan *range* digunakan sinyal sinkronisasi sebagai *timing mark*. SAR *imaging* dilakukan dengan mengambil sejumlah *range profile* secara manual pada posisi berbeda berjarak 5 cm sepanjang lintasan 3 meter.

2. Dasar Teori

2.1. Radar

Sistem radar menggunakan gelombang termodulasi dan antena terarah untuk memancarkan energi elektromagnetik pada arah tertentu dan menerima sinyal yang terpantul yang diolah untuk mendapatkan informasi mengenai jarak, kecepatan, posisi *angular* dan lainnya. Radar sinyal kontinyu mentransmisikan gelombang secara terus-menerus dan menggunakan antena penerima dan pengirim yang terpisah [2].

Synthetic Aperture Radar (SAR) merupakan radar pencitraan yang dipasang pada sebuah *platform* bergerak. Seperti radar konvensional, SAR mentransmisikan gelombang elektromagnetik secara berurutan dan menerima sinyal yang dipantulkan oleh target yang dipindai. Proses *transmit*/*receive* yang berulang serta pergerakan *platform* ini menghasilkan data dari target pada posisi ataupun perspektif yang berbeda. Hal ini memungkinkan pembentukan data yang diperoleh seakan berasal dari antena dengan *aperture* sangat besar [3].

2.1.1. Range Resolution

Range r*esolution* merupakan kemampuan radar untuk membedakan dua target dengan posisi identik. Bila terdapat dua buah target dengan posisi berdekatan, sinyal pantul yang diterima akan saling berinterferensi bila jarak antara kedua target lebih kecil dari $\frac{\tau\, c}{2}$ dengan τ adalah durasi transmisi sinyal. Frekuensi sinyal termodulasi disebut *chirp* dengan *range resolution* [3]

$$\Delta R = \frac{c}{2 \times BW} \tag{1}$$

2.2. Frequency Modulated Continuous Wave Radar

Gelombang yang dikirim oleh radar sinyal kontinyu dapat dianggap sebagai gelombang sinusoidal. Frekuensi gelombang pantul dari target diam terkonsentrasi pada f_0 sementara frekuensi tengah dari gelombang pantul objek bergerak akan bergeser sejauh frekuensi *doppler* (f_d). Radar sinyal kontinyu yang tidak dimodulasi dapat mengukur kecepatan radial (pergeseran *Doppler*) dan posisi *angular* namun tidak dapat mengukur jarak. Agar dapat mengukur jarak, sinyal yang dikirim harus dimodulasi [2]. Diperlukan *Narrow Band Filter* (NBF) untuk meminimalisasi daya *noise* yang dapat diimplementasikan menggunakan *Fast Fourier Transform* (FFT) namun mengakibatkan pengolahan data pada satu waktu terbatas jumlahnya. Jumlah data ini disebut *dwell time* yang menentukan *frequency resolution* atau *bandwidth* dari NBF.

$$T_{Dwell} = {}^{N_{FFT}}/_{2B} \tag{2}$$

Sinyal terima didapatkan dengan menurunkan persamaan radar High PRF dengan $P_{av} = P_{cw}$ di mana P_{cw} adalah daya kirim rata-rata selama *dwell interval*, $T_i = T_{Dwell}$, G_t *gain* antena pengirim, G_r *gain* antena penerima. L_{win} adalah *loss* terkait dengan jenis *window* yang digunakan dalam perhitungan FFT [2].

$$SNR = \frac{P_{av} T_i G^2 \lambda^2 \sigma}{(4\pi)^3 R^4 k T_e F L} \tag{3}$$

2.2.1. Linear FM (LFM) *Continuous Wave Radar*

LFM tidak dapat terus berubah pada satu arah maka umumnya digunakan sifat periodik dari modulasi. Frekuensi *beat* f_b menunjukkan perbedaan frekuensi antara sinyal yang diterima dan dikirim [2].

Frekuensi modulasi dipilih $f_m = \frac{1}{2T_0}$ sehingga didapat laju perubahan frekuensi

$$\dot{f} = \frac{\Delta f}{t_0} = \frac{\Delta f}{(1/2f_m)} = 2f_m \Delta f \tag{4}$$

Sementara frekuensi *beat* didefinisikan dengan

$$f_b = \frac{4R f_m \Delta f}{c} \tag{5}$$

Ketika target tidak diam, maka sinyal terima mengalami pergeseran *doppler* akibat *delay* Δt. Bagian positif dari frekuensi *doppler* mengurangi frekuensi *beat* (f_{bu}) sementara bagian negatif akan menambah frekuensi *beat* (f_{bd}) [2].

$$f_{bu} = \frac{2R}{c}\dot{f} - \frac{2\dot{R}}{\lambda} \tag{6}$$

$$f_{bd} = \frac{2R}{c}\dot{f} + \frac{2\dot{R}}{\lambda} \tag{7}$$

$\dot{R}$ adalah laju *range* atau kecepatan radial target.

Range dihitung dengan menambahkan persamaan (6) dan (7), sementara laju *range* dihitung dengan mengurangi persamaan (7) dan (6).

$$R = \frac{c}{4\dot{f}}(f_{bu} + f_{bd}) \quad (8)$$

$$\dot{R} = \frac{\lambda}{4\dot{f}}(f_{bd} - f_{bu}) \quad (9)$$

2.3. Fast Fourier Transform

Fast Fourier Transform merupakan algoritma yang merubah sinyal dari *domain* waktu ke *domain* frekuensi. FFT sendiri merupakan penerapan dari *Discrete Fourier Transform* (DFT) yang telah dioptimasi dimana DFT sendiri adalah *Fourier Transform* untuk *domain* waktu diskrit. Secara matematis, DFT dapat dirumuskan sebagai berikut [4]:

$$X[k] = \sum_{n=0}^{N-1} x[n].e^{-\frac{j2\pi nk}{N}}, \qquad k = 0, 1, 2, \dots, N-1 \quad (10)$$

Sementara, *Inverse Discrete Fourier Transform* (IDFT) mengembalikan sinyal dari *domain* frekuensi ke *domain* waktu diskrit.

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[k].e^{\frac{j2\pi nk}{N}}, \qquad n = 0, 1, 2, \dots, N-1 \quad (11)$$

Perhitungan DFT secara langsung membutuhkan operasi artimatika sebanyak $O(N^2)$, sedangkan perhitunggan dengan FFT akan membutuhkan operasi sebanyak $O(N \log N)$. Kompleksitas operasi DFT memerlukan $2N^2$ evaluasi trigonometri, $4N^2$ perkalian real dan 4N(N-1) penjumlahan real atau sering disebut $O(N^2)$.

2.3.1. Zero Padding

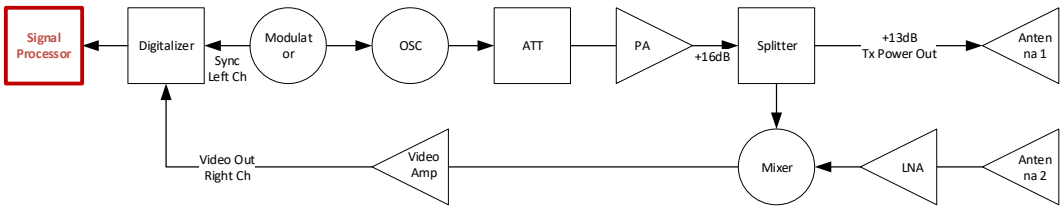
Salah satu cara umum pada analisis frekuensi sinyal diskrit dapat menggunakan *zero padding* untuk meningkatkan resolusi frekuensi dari *discrete* Fourier *transform* (DFT). Dengan menambahkan nilai nol pada suatu sinyal diskrit, akan dihasilkan frekuensi DFT yang lebih rapat. DFT sendiri sering dianggap sebagai perkiraan dari *truncated* DTFT pada suatu set frekuensi diskrit. Pendekatan lainnya terkait masalah nilai diluar *N* dapat didekati dengan mengganggap DFT periodik dengan periode *N*. Berdasarkan hal ini, jika sinyal diskrit dengan panjang *N* ditambah nilai nol hingga mencapai panjang *M*, maka periode sinyal berubah menjadi *M* yang menjadikan hasil DFT makin mendekati nilai *truncated* DTFT yang artinya mengingkatkan resolusi frekuensi. [6] Pertimbangan lain penggunaan *zero padding* adalah karena sifat Radix-2 dalam perhitungan FFT yang lebih efisien untuk sinyal dengan panjang kelipatan dua.

2.4. Windowing

Dalam pengolahan sinyal, *window function* adalah fungsi matematis dimana nilai diluar interval tertentu dibuat nol [5]. Penggunaan *window* sendiri memungkinkan sinyal yang berubah cukup cepat untuk diterapkan DFT atau *autocorrelation*. Sebagian besar pengolahan sinyal ini dilakukan dengan mengambil *window* kecil dari sinyal yang biasa disebut frame. Penerapan *window* pada suatu sinyal dilakukan dengan cara mengalikan fungsi *window* yang panjangnya terbatas pada sinyal asli untuk mendapatkan sinyal dengan panjang terbatas yang magnitudenya terskala dari sinyal asli [6]. Bentuk dari fungsi *window* memiliki pengaruh pada perubahan magnitude sinyal asli. Beberapa fungsi *window* yang sering digunakan antara lain seperti *Hanning*, *Hamming*, *Bartlett* dan *Blackman*.

3. Hasil Perancangan dan Pembahasan

Sistem radar yang digunakan pada penelitian ini berbasis FMCW dengan blok diagram pada Gambar 3.1. Osilator membangkitkan sinyal lalu diturukan oleh *Attenuator* untuk dikuatkan *Power Amplifier* sebesar 16 dBm. *Splitter* memisahkan sinyal untuk dipancarkan melalui antena 1 dan masuk ke *Mixer* untuk digabungkan dengan sinyal terima dari antena 2 yang dikuatkan oleh *Low Noise Aplifier*. Keluaran *Mixer* dikuatkan kembali oleh *Video Amplifier* untuk direkam pada kanal kanan *digitalizer* sementara sinyal sinkronisasi direkam pada kanal kiri. Keluaran *digitalizer* berupa *file* dengan *format* .wav yang diproses oleh *Signal Processor* yang merupakan fokus dari penelitian ini.

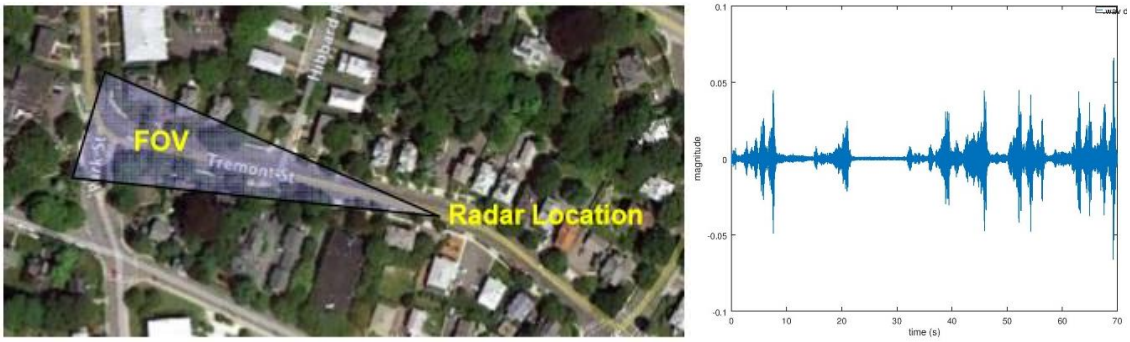


Gambar 3.1. Blok diagram sistem radar FMCW

Parameter radar yang digunakan pada proses pengolahan sinyal tergambar pada Tabel 3.1.

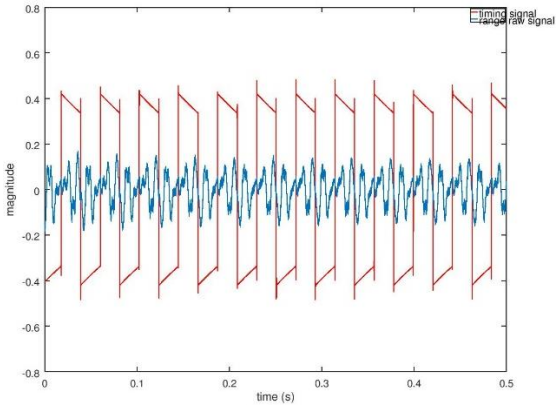
Tabel 3.1. Parameter pengolahan sinyal

Spesifikasi	**Nilai**
Jenis Sinyal	*Coherent* FMCW
Frekuensi f_c	2.4 GHz
***Pulse* Time** T_p **(*Doppler*)**	0.25 s
***Pulse* Time** T_p **(Range & SAR Imaging)**	$2x10^{-3}s$
Bandwidth LFM	330 MHz



Gambar 3.2. Lanskap pemindaian radar dan sinyal hasil yang didapat

Dalam pengambilan data *doppler*, tegangan *input* VCO (OSCI) diset sebagai tegangan DC stabil untuk mendapatkan sinyal kontinyu tanpa modulasi. Radar diarahkan ke kendaraan yang bergerak dengan kecepatan tinggi pada Tremont *street* seperti pada Gambar 3.2. Pengolahan *doppler time intensity* (DTI) dimulai dengan pembacaan data dari *file* berformat .wav yang ditunjukan sinyalnya pada Gambar 3.2, kemudian dibulatkan sesuai jumlah *chirp* dan dibentuk menjadi matriks berdimensi jumlah *chirp* x *sample* per *chirp* yang selanjutnya kurangi dengan nilai rata-rata sinyal. Kemudian dilakukan operasi IFFT lalu dikonversi dalam dB dan hanya setengah panjang *chirp* yang digunakan. Informasi waktu didapat dengan membentuk matriks 1 x jumlah *chirp* bernilai antara 1 hingga *pulse time* * jumlah *chirp* yang jarak antar elemennya *linear*. Pengolahan kecepatan dari nilai delta F *chirp* dibagi setengah lambda. Hasilnya kemudian dinormalisasi dan diplot.

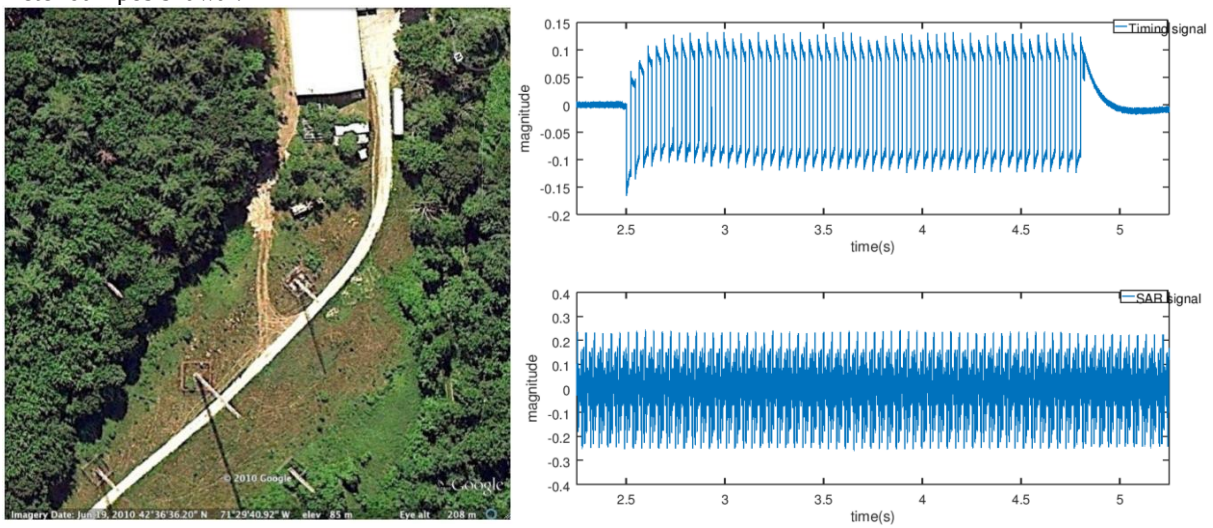


Gambar 3.3. Cuplikan 0.5 detik pertama dari sinyal timing dan range hasil pemindaian dua orang berlari di hutan

Pada pengolahan *range time intensity* (RTI) digunakan sinyal sinkronisasi sebagai *timing mark* pada proses deteksi *chirp*. Radar diarahkan pada dua orang yang berjalan di hutan dengan kedua kanal audio digunakan selama merekam *file* .wav dimana cuplikan 0,5 detik pertama dari sinyal *timing* dan range ditunjukan Gambar 3.3. Deteksi *chirp* dilakukan dengan menerapkan nilai *threshold* sebesar 0.08 pada sinyal sinkronisasi kemudian dibentuk matriks indeks dari nilai diatas *threshold* sebagai acuan iterasi. Kondisi deteksi *chirp* adalah 10 *sample* terakhir berada dibawah *threshold*. Ketika kondisi terpenuhi, diambil *chirp* sepanjang 0.25 detik. Iterasi dilanjutkan ke *sample* sinyal sinkronisasi diatas *threshold* minimal berjarak 0.25 detik berikutnya. Setelah proses deteksi, dilakukan pengolahan RTI dalam 3 *mode*, yaitu tanpa *clutter rejection*, dengan *2-pulse canceler clutter rejection* dan *2-pulse magnitude only canceler clutter rejection*. RTI sendiri dilakukan dengan menerapkan operasi IFFT yang hasilnya dikonversi dalam dB untuk mendapatkan nilai kecepatan. Nilai kecepatan ini hanya diambil setengah durasi *chirp* pertama yang dinormalisasi terhadap nilai kecepatan maksimum. *2-pulse canceler clutter rejection* mengurangi setiap *chirp* dengan *chirp* sebelumnya sebelum operasi IFFT sementara *2-pulse magnitude only canceler clutter rejection* mengurangi setiap *chrip* dengan *chirp* sebelumnya setelah operasi IFFT.

Pengambilan data SAR dilakukan secara manual dengan cara melakukan pemindaian pada sebuah Gudang seperti ditunjukan Gambar 3.4 dan merekam file .wav selama satu hingga dua detik, kemudian menonaktifkan sinyal sinkronisasi untuk kemudian radar dipindahkan sejauh 5 cm dan mengaktifkan kembali

sinyal sinkronisasi. Proses ini dilakukan terus-menerus hingga perpindahan radar mencapai aperture sepanjang 3 meter dari posisi awal.



Gambar 3.4. Lanskap yang dipindai [1] dan cuplikan sinyal timing dan SAR yang diolah

Perancangan algoritma pengolahan SAR *imaging* dilakukan dengan deteksi posisi berdasarkan *mute* pada sinyal sinkronisasi sebagai penanda perpindahan posisi. Dilanjutkan dengan deteksi dan penggabungan *chirp* tiap posisi. Normaliasi hasil penggabungan *chirp* dengan nilai rata-ratanya serta dilakukan proses *windowing* sebelum operasi FFT. Hasilnya kemudian diterapkan *matched filter* dan dilakukan *stolt interpolation* untuk mengkoreksi fasa terhadap kelengkungan jarak. Terakhir dilakukan IFFT 2D yang sebelumnya dilakukan *windowing*. Hasilnya dikonversi dalam dB dan diplot sebagai SAR i*mage*.

Perangkat *Raspberry Pi* yang digunakan sebagai *platform* pengolahan sinyal merupakan *Raspberry Pi model* B dengan spesifikasi pada Tabel 3.2. Konfigurasi perangkat *Raspberry Pi* menyangkut instalasi sistem operasi *Raspbian* dengan melakukan *flashing* menggunakan *tools Etcher* ke perangkat *micro* SD, koneksi jaringan dan mengaktifkan fitur SSH. Konfigurasi awal ini dilakukan dengan perangkat *monitor*, *mouse* dan *keyboard* namun selanjutnya pengoperasian dan instalasi perangkat lunak *Octave* dilakukan secara *headless* dengan koneksi SSH. Instalasi perangkat lunak *Octave* sendiri hanya dengan menggunakan perintah *sudo apt-get install octave* karena *repository octave* secara *default* tersedia. Setelah konfigurasi perangkat selesai, program diunggah melalui protokol SFTP dari perangkat pengembangan.

Tabel 3.2. Spesifikasi *Raspberry Pi*

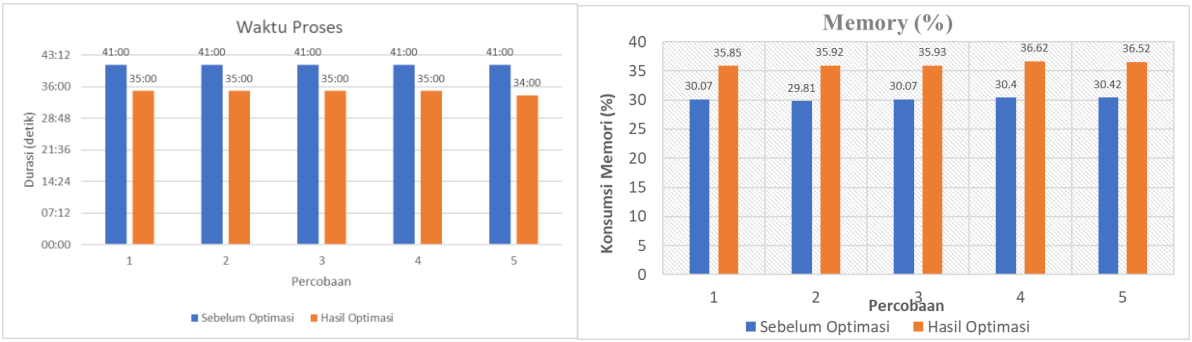
Spesifikasi	**Nilai**
System on Chip (SoC)	BCM2837
CPU	Quad Cortex A53 @ 1.2 GHz
RAM	1 GB SDRAM
Operating System	Raspbian “Stretch”
Storage	16 GB micro-SD
Wireless	802.11n / Bluetooth 4.0

Pengujian performasi komputasi pengolahan sinyal dilakukan dengan bantuan *script python* untuk menjalankan skenario pengujian dengan nilai variabel tertentu dan pencatatan kondisi sistem seperti utilisasi CPU, konsumsi memori dan *elapsed time* (waktu proses) serta temperatur.

4. Pengukuran dan Analisis

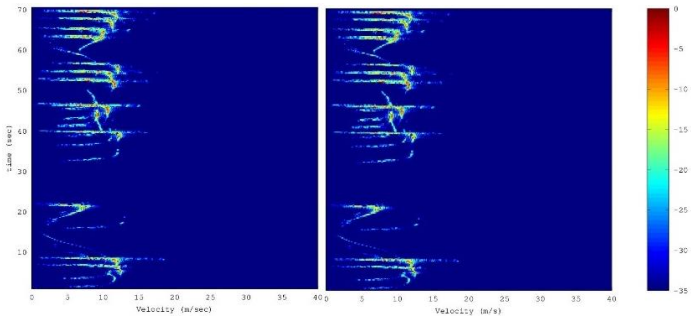
Pengukuran dilakukan dengan menggunakan data *dummy* dari penelitian sebelumnya sebagai perbandingan tingkat akurasi pengolahan sinyal. Optimasi dilakukan dengan pengurangan penggunaan *loop* baik menggunakan operasi vektorisasi ataupun fungsi bawaan perangkat lunak *Octave*.

Gambar 4.1 menunjukan perbandingan waktu proses dan konsumsi memori sebelum dan setelah optimasi pengolahan DTI dengan lima kali percobaan terlihat bahwa hasilnya konsisten mengalami perbaikan dari rata-rata 41 detik menjadi 35 detik. Sementara perbandingan konsumsi memori mengindikasikan peningkatan konsumsi memori dari 31,16% ke 36,17%. Hal ini terjadi karena teknik optimasi vektorisasi yang dilakukan memproses data secara keseluruhan dengan waktu proses yang lebih cepat.



Gambar 4.1. Perbandingan waktu proses dan konsumsi memori DTI sebelum dan setelah optimasi

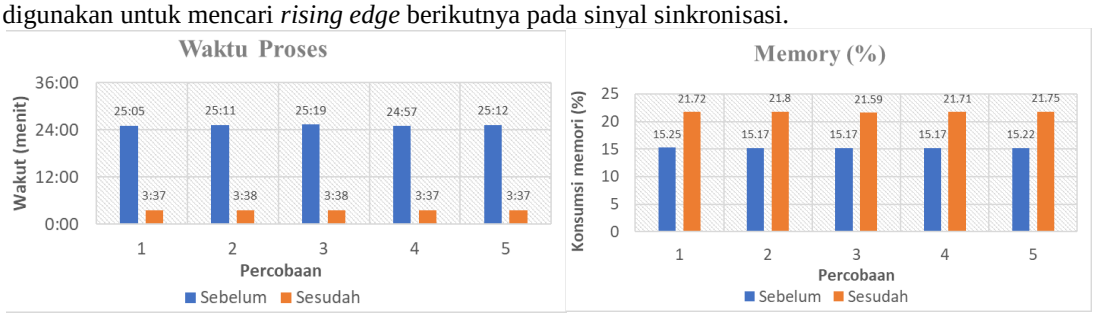
Hasil pengolahan DTI secara visual tidak terdapat perbedaan antara sebelum dan setelah dilakukan optimasi. Gambar 4.2 sendiri menunjukan nilai perubahan kecepatan terhadap waktu serta intensitas atau nilai sinyal yang didapat.



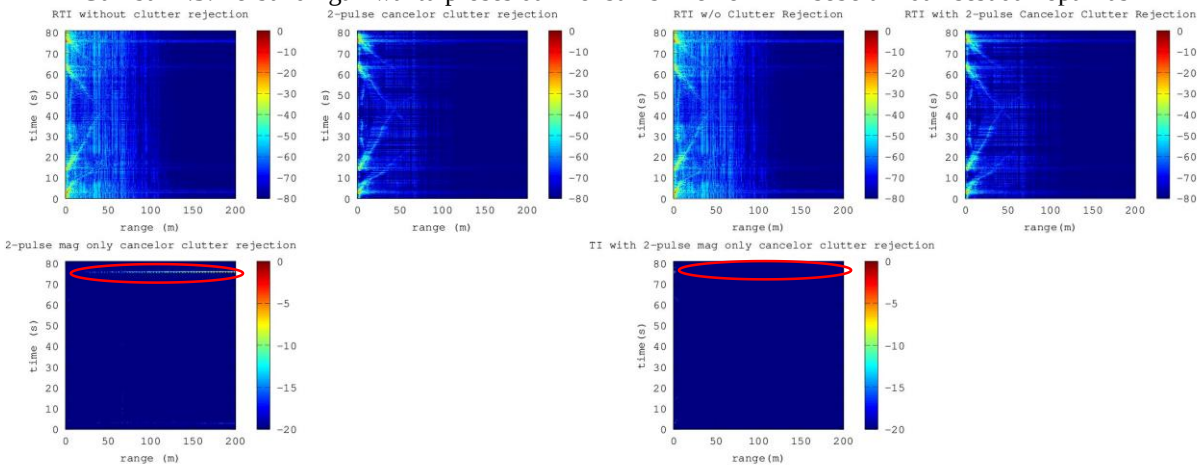
Gambar 4.2. Hasil DTI setelah optimasi

Pada pengolahan RTI, optimasi dilakukan pada deteksi *chirp* yang semula melakukan pengecekan setiap *sample* sinyal sinkronisasi yang dinilai tidak efisien. Proses ini dioptimasi dengan melakukan iterasi hanya pada *sample* sinyal sinkronisasi diatas *threshold* dan mengabaikan sepanjang durasi satu *chirp* bila didapatkan deteksi.

Hasil pengujian RTI pada *Raspberry PI* dilakukan sebanyak lima kali. Gambar 4.3 menunjukan perbandingan waktu proses dan konsumsi memori sebelum dan sesudah optimasi dimana terjadi peningkatan performa secara signifikan dari rata-rata waktu proses 25 menit 9 detik ke 3 menit 27 detik. Sementara konsumsi memori mengalami peningkatan dari 15,18% ke 21,71%. Peningkatan memori ini terjadi karena operasi vektorisasi yang digunakan untuk mencari *rising edge* berikutnya pada sinyal sinkronisasi.



Gambar 4.3. Perbandingan waktu proses dan konsumsi memori RTI sebelum dan sesudah optimasi



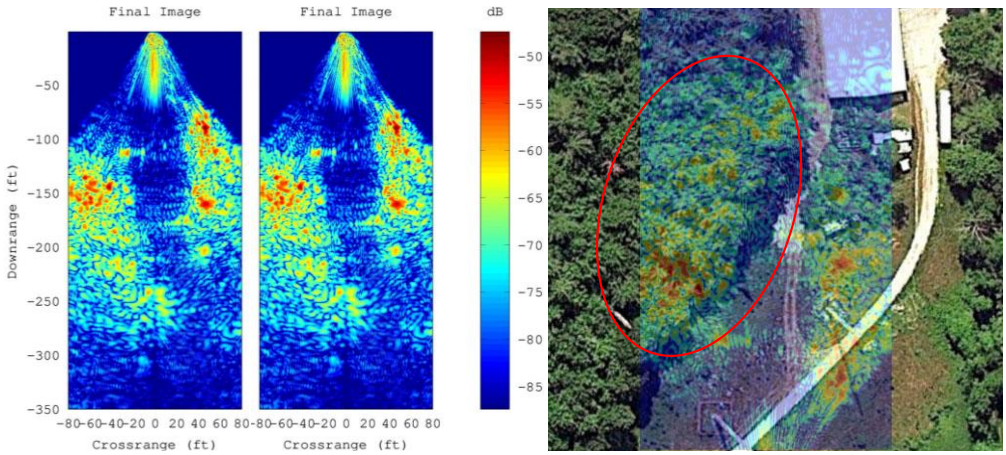
Gambar 4.4. Perbandingan hasil akhir RTI sebelum dan sesudah optimasi

Hasil pengolahan RTI secara visual hanya terdapat perbedaan pada RTI dengan *2-pulse magnitude only cancelor clutter rejection* seperti lingkaran merah pada Gambar 4.4 yang menunjukan pergerakan dua target terhadap waktu dengan intensitasnya yang dapat diartikan sebagai ukuran target ataupun jarak dari radar.

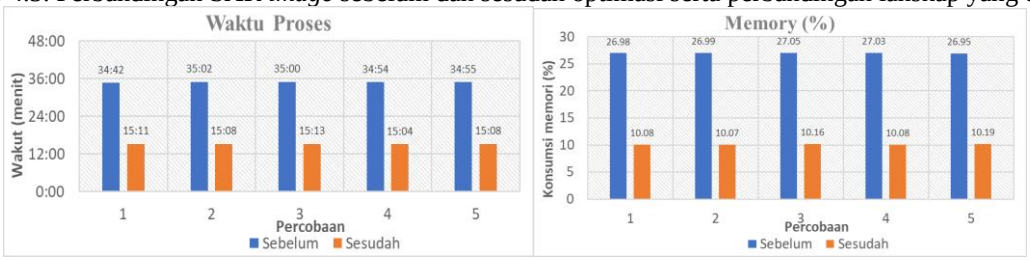
Optimasi yang dilakukan pada pengolahan SAR *imaging* antara lain pengurangan loop terutama pada proses pembentukan *window*, operasi matriks tiap posisi dan penambahan *zero padding*. Perbandingan hasil akhir SAR *imaging* ditunjukan pada Gambar 4.5 dapat disimpulkan bahwa hasil optimasi memiliki tingkat akurasi yang sama dengan sebelum optimasi dan mampu memetakan objek dari lanskap yang dipindai.

Perbandingan waktu proses dan konsumsi memori yang didapat dari lima kali percobaan menunjukan peningkatan yang konsisten dimana sebelum optimasi waktu proses rata-rata 34 menit 54 detik dengan konsumsi memori 27% sementara setelah optimasi didapatkan rata-rata waktu proses 15 menit 8 detik dan konsumsi memori rata-rata 10,14% yang ditunjukan Gambar 4.6.

Zero padding yang digunakan sebesar 2048 menentukan ukuran pelebaran sinyal sebelum FFT, perhitungan *stolt interpolation* dan *cross range*. Selain itu, *zero padding* 1024 (kiri) juga diuji dengan perbandingan yang ditunjukan Gambar 4.7.a. dimana *zero padding* 1024 mengalami distorsi terlihat pada lingkaran merah. Faktor pengali dimensi IFFT 2D hasil *stolt interpolation* diturunkan menjadi ½ (kanan) dan ¼ (kedua dari kanan) guna menghindari *out of memory error* yang perbandingannya ditunjukan Gambar 4.7.b.

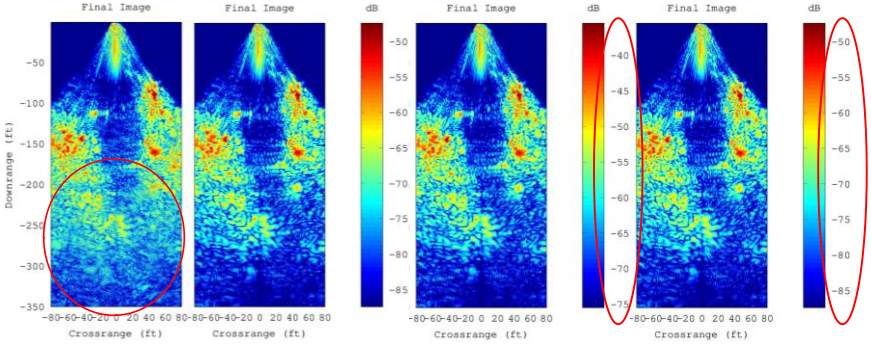


Gambar 4.5. Perbandingan SAR *image* sebelum dan sesudah optimasi serta perbandingan lanskap yang dipindai



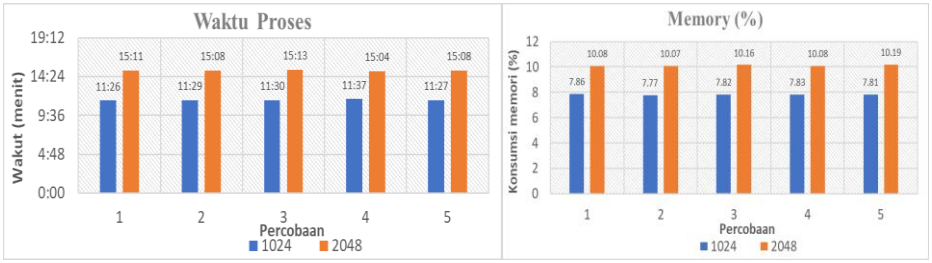
Gambar 4.6. Perbandingan waktu proses dan konsumsi memori SAR *imaging* sebelum dan sesudah optimasi

Faktor pengali dimensi ini memiliki pengaruh nilai intensitas sinyal yang didapat dengan selisih 10 dB. Gambar 4.8 menunjukan perbandingan waktu proses dan konsumsi memori antara *zero padding* 1024 dengan 2048 dari lima kali percobaan dimana 1024 memerlukan waktu proses 11 menit 29 detik dibanding 2048 yang memerlukan waktu proses 15 menit 8 detik. *Zero padding* 1024 memerlukan konsumsi memori yang lebih kecil sebesar 7,82% dibanding dengan zpad 2048 sebesar 10,14%.



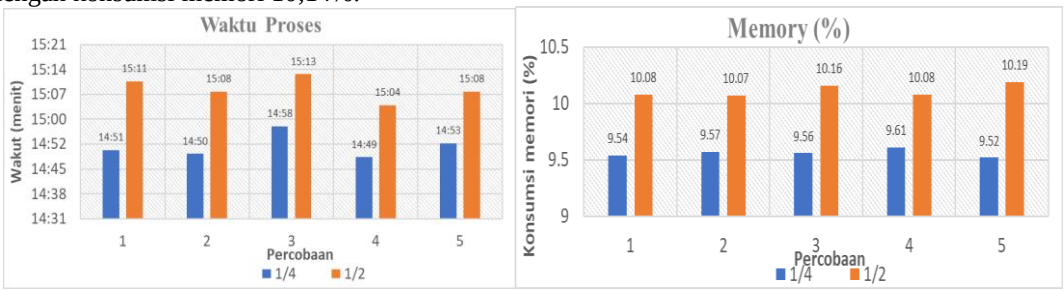
(a) (b)

Gambar 4.7. Perbandingan *zero padding* 1024 & 2048 (a) serta faktor pengali dimensi IFFT 2D ¼ & ½ (b)

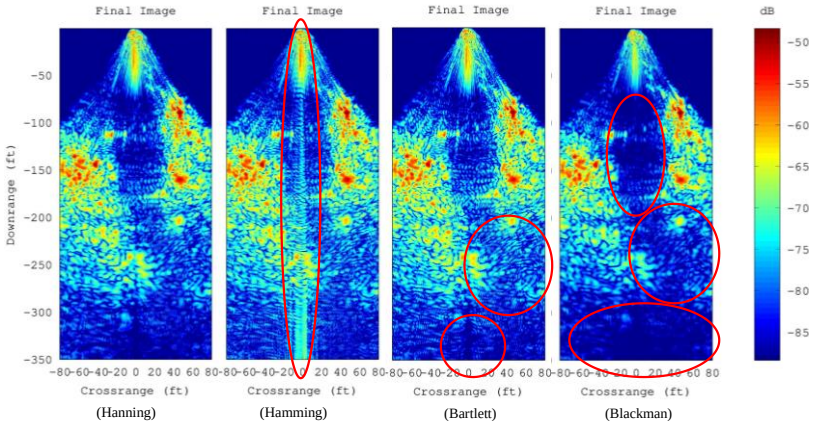


Gambar 4.8. Perbandingan waktu proses dan konsumsi memori *zero padding* 1024 dan 2048

Sementar perbandingan waktu proses dan konsumsi memori faktor pengali dimensi IFFT 2D ditunjukan oleh Gambar 4.9 dari lima kali percobaan dengan peningkatan performa untuk faktor pengali ¼ sebesar rata-rata 11 menit 31 detik pada konsumsi memori 9.56% dibandingkan dengan faktor pengali ½ yaitu selama 15 menit 8 detik dengan konsumsi memori 10,14%.



Gambar 4.9. Perbandingan waktu proses dan konsumsi memori faktor pengali IFFT 2D

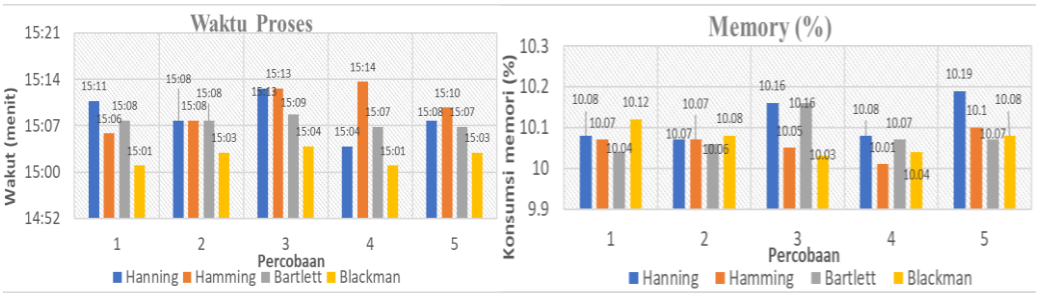


(Hanning) (Hamming) (Bartlett) (Blackman)

Gambar 4.10. Perbandingan SAR Image penggunaan window berbeda.

Perbandingan perbedaan fungsi *window* ditunjukan Gambar 4.10. Pada fungsi *window Hamming* distorsi visual sangat terlihat dibagian tengah sepanjang sumbu vertikal yang dilingkari merah, sementara *Bartlett* hanya terjadi sedikit distorsi. Sedangkan fungsi *window Blackman* menghasilkan gambar lebih jernih dibanding fungsi *window Hanning* seperti pada lingkaran merah. Dari sisi waktu proses serta konsumsi memori tidak didapatkan perbedaan signifikan. Seperti ditujukan oleh Gambar 4.11, selisih waktu proses maksimal sebesar 13 detik dengan rata-rata 8.8 detik, sedangkan selisih konsumsi memori maksimal sebesar 0.13% yang memiliki rata-rata selisih 0.084%.

Berdasarkan hasil percobaan didapatkan rekomendasi konfigurasi pengolahan sinyal SAR *Imaging* yaitu dengan *zero padding* 2048 dan fungsi *window Blackman* untuk mendapatkan hasil yang baik dan faktor pengali IFFT 2D ¼ untuk mempersingkat waktu proses bila selisih intensitas sinyal sebesar 10dB dapat diterima.



Gambar 4.11. Perbandingan waktu proses dan konsumsi memori SAR *Imaging* untuk *window* berbeda.

5. Kesimpulan dan Saran

Pengolahan sinyal radar untuk pengukuran Doppler, Range dan SAR *Imaging* menggunakan Octave pada Raspberry Pi dirancang berdasarkan optimasi dari penelitian sebelumnya yang menggunakan Matlab dan Laptop kemudian direalisasikan pada Raspberry Pi yang dijalankan secara *headless* dan diakses melalui protokol SSH/VNC/FTP sementara hasil pemindaian tersimpan sebagai file berformat .wav untuk masing-masing skenario. Optimasi dilakukan melalui penggantian *loop* dengan vektorisasi matriks ataupun fungsi bawaan Octave. Secara khusus optimasi pada algoritma deteksi *chirp* dan posisi dengan perbaikan proses *loop* yang semula tiap *sample* sinyal timing menjadi hanya sample diatas *threshold* dan mengabaikan satu durasi *chirp* atau posisi ketika didapatkan deteksi. Performa pengolahan sinyal membaik setelah optimasi ditunjukan oleh waktu proses yang lebih cepat untuk ketiga skenario dengan hasil yang akurat. Sementara peningkatan konsumsi memori akibat penggunaan vektorisasi matriks. Pengolahan doppler membaik ke 35 detik dengan konsumsi memori lebih tinggi ke 36.17%. Pengolahan *range* meningkat signifikan ke 3 menit 37 detik dengan peningkatan konsumsi memori ke 21,71%. Perbedaan visual terjadi pada RTI dengan *2-pulse magnitude only canceler clutter rejection.* Pengolahan SAR *Image* meningkat ke 15 menit 8 detik setelah optimasi sedangan konumsi memori menurun ke 10,12% tanpa perubahan visual. Parameter *zero padding* yang menentukan pemekaran dimensi data berpengaruh secara visual terhadap hasil akhir SAR *Image* meski *zero padding* 1024 didapatkan waktu proses lebih cepat selama 11 menit 29 detik dan konsumsi memori lebih baik sebesar 7,82%. Faktor pengali IFFT 2D tidak menimbulkan distorsi hasil namun berpengaruh pada intensitas sinyal dengan selisih 10 dB. Semakin kecil faktor pengalinya didapatkan waktu proses yang lebih cepat ke rata-rata 14 menit 52 detik dengan konsumsi memori rata-rata 9,56% untuk faktor pengali ¼. Penggunaan *window* berbeda tidak memiliki pengaruh waktu proses dan konsumsi memori yang signifikan dengan selisih waktu proses maksimum sebesar 13 detik dan selisih konsumsi memori maksimum sebesar 0.13%, namun memiliki dampak visual yang berbeda-beda dimana Blackman menghasilkan gambar SAR yang paling jernih.

Daftar Pustaka

[1] G. L. Charvat, A. J. Fenn and B. T. Perry, "The MIT IAP Radar Course: Build a Small Radar System Capable of Sensing Range, Doppler, and Synthetic Apertur (SAR) Imaging," in *Radar Conference (RADAR)*, Atlanta, GA, USA, 2012.

[2] B. R. Mahafza, Radar Systems Analysis and Design Using MATLAB, Huntsville, Alabama: Chapman & Hall/CRC, 2000.

[3] M. Elhefnawy and W. Ismail, "Fundamentals of Synthetic Aperture Radar System," *International Journal of Emerging Technology and Advanced Engineering*, vol. 5, no. 9, pp. 172-178, 2015.

[4] S. G. Radiana, "Discrete Fourier Transform Menjadi Fast Fourier Transform," *Jurusan Teknik Elektro, FT Universitas Gajah Mada*, 2008.

[5] E. W. Weisstein, CRC Concise Encyclopedia of Mathematics, Second Edition, CRC Press, 2002.

[6] T. F. Quatieri, Discrete-time Speech Signal Processing, Prentice Hall PTR, 2002.

[7] J. W. Eaton, "GNU Octave: Top," 2017. [Online]. Available: https://www.gnu.org/software/octave/doc/interpreter/. [Accessed 3 December 2017].

[8] Raspberry PI Foundation, "Raspberry Pi FAQs - Frequently Asked Question," Raspberry PI Foundation, 2017. [Online]. Available: https://www.raspberrypi.org/help/faqs/. [Accessed 4 December 2017].