**TUGAS AKHIR – TE 141599**

# PENERAPAN MODUL PENGENALAN SUARA V3 UNTUK PENGATURAN POSISI PURWARUPA RANJANG LIPAT

Riza Kamelia
NRP 07111645000061

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M.Eng.Sc
Astria Nur Irfansyah, ST., M.Eng .,Ph.D

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2018

**FINAL PROJECT – TE 141599**

# *IMPLEMENTATION OF VOICE RECOGNITION MODULE FOR POSITION CONTROL PROTOTYPE BED FOLDING*

Riza Kamelia
NRP 07111645000061

Advisor
Dr. Ir. Hendra Kusuman, M.Eng.Sc
Astria Nur Irfancyah,ST.,M.Eng.,Ph.D

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

# PERNYATAAN KEASLIAN
# TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Penerapan Modul Pengenalan Suara V3 untuk Pengaturan Posisi Purwarupa Ranjang Lipat**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, **Juli 2018**

Riza Kamelia
NRP 07111645000061

-----Halaman ini sengaja dikosongkan-----

**PENERAPAN MODUL PENGENALAN SUARA V3 UNTUK PENGATURAN POSISI PURWARUPA RANJANG LIPAT**

# TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Dr. Ir. Hendra Kusuma, M. Eng.Sc | Astria Nur Irfansyah, ST., M.Eng., Ph.D |
| NIP. 196409021989031003 | NIP. 198103252010121002 |

**SURABAYA**
**JULI, 2018**

-----Halaman ini sengaja dikosongkan-----

**PENERAPAN MODUL PENGENALAN SUARA V3 UNTUK PENGATURAN POSISI PURWARUPA RANJANG LIPAT**

**Nama : Riza Kamelia**
**Pembimbing I : Dr. Ir. Hendra Kusuma, M. Eng.Sc**
**Pembimbing II : Astria Nur Irfansyah,ST., M. Eng.,Ph.D**

# ABSTRAK

Ranjang pasien yang berupa ranjang lipat merupakan hal yang dibutuhkan oleh pasien terutama yang mengalami kelumpuhan total. Gerakan dari ranjang pasien yang sudah ada saat ini yaitu menggunakan tombol kontrol dan juga ada yang menggunakan pengendali jarak jauh. Bagi pasien lumpuh total masih memiliki kendala dalam menggerakkan ranjangnya karena mereka tidak dapat menggerakkan tubuh dan tangannya. Pada tugas akhir ini, dibuat suatu sistem pengatur posisi ranjang yang dapat dikontrol dengan perintah suara. Sistem ini menggunakan mikrokontroler STM32F407 *Discovery* sebagai unit kontrol aktuator pengatur posisi ranjang berupa motor servo dengan masukan suara yang telah direkognisi oleh modul *Voice Recognition V3* (VR3). Dalam pembelajaran modul VR3 untuk mempelajari satu kata dapat mencoba sebanyak maksimal 10x dalam 1 kali percobaan. Hasil yang dicapai menunjukkan perintah suara untuk mengatur posisi ranjang lipat dapat bekerja dengan baik untuk 3 macam perintah suara yaitu dengan rata-rata tingkat pengenalan suara sebesar 72%. Sedangkan pergerakan posisi ranjang dari $0^0$ dimana posisi ranjang adalah horizontal lurus menuju posisi $30^o$ membutuhkan waktu selama 5 detik.

**Kata Kunci :** Ranjang Lipat Pasien, Pengenalan suara, STM32F407 *Discovery*

-----Halaman ini sengaja dikosongkan-----

**IMPLEMENTATION OF VOICE RECOGNITION MODULE FOR POSITION CONTROL PROTOTYPE BED FOLDING**

**Name : Riza Kamelia**
**Advisor 1st : Dr. Ir. Hendra Kusuma, M. Eng.Sc**
**Advisor 2nd : Astria Nur Irfansyah,ST., M. Eng.,Ph.D**

## *ABSTRACT*

*Bed for patient as in folding bed is a thing that the patient need, especially for completely paralyzed people. the one that available nowadays are using control button or even using remote control to control it from far away. for a completely paralyzed people, it is stilla struggle to move their bed because of limit on moving part of their body. in this final project, created a system for controlling bed position using the patient's voice. this system is using STM32F407 Discovery as a unit control for an actuator that is a servo motor using patient's voice as an input that has been recognized with voice recognition V3 module(VR3). The result shows the voice command to adjust the folding bed position to work well for 3 kinds of voice commands with an average voice recognition rate of 72%. While the movement of the bed position of $0^o$ where the position of the bed is horizontally straight towards the position $30^o$ takes time for 5 seconds.*

***Keywords*** **:** *Folding Bed for Patients, Voice Recognition, STM32F4 Discovery*

-----Halaman ini sengaja dikosongkan-----

# KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang selalu memberikan rahmat dan hidayah-Nya sehingga Tugas Akhir ini dapat terselesaikan dengan baik. Shalawat serta salam semoga selalu dilimpahkan kepada Rasulullah Muhammad SAW, keluarga, sahabat, dan umat muslim yang senantiasa meneladani beliau.

Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Strata-1 pada Bidang Studi Elektronika, Jurusan Teknik Elektro, Fakultas Teknologi Industrim, Institut Teknologi Sepuluh Nopember Surabaya dengan judul:

**PENERAPAN MODUL PENGENALAN SUARA V3 UNTUK PENGATURAN POSISI PURWARUPA RANJANG LIPAT**

Dengan selesainya Tugas Akhir ini penulis menyampaikan terimakasih yang sebesar-besarnya kepada :

1. Kedua orang tua atas limpahan doa, kasih sayang dan teladan hidup bagi penulis
2. Bapak Dr. Ir. Hendra Kusuma, M. Eng.Sc dan bapak Astria Nur Irfansyah,ST., M. Eng.,Ph.D selaku dosen pembimbing yang telah meluangkan waktu dan tenaganya untuk membingbing dalam menyelesaikan Tugas Akhir ini.
3. Seluruh staf pengajar dan administrasi departemen Teknik Elektro FTE – ITS.
4. Semua pihak yang telah banyak membantu untuk menyelesaikan Tugas Akhir ini yang tidak dapat penulis sebutkan satu persatu

Harapan kami sebagai penulis adalah semoga terselesaikannya Tugas Akhir ini dapat bermanfaat bagi kami serta pembaca. Sadar atas keterbatasan yang dimiliki oleh penulis karena hasil dari Tugas Akhir ini jauh dari kesempurnaan. Demikian penulis sudah berusaha semaksimal mungkin dan pintu maaf serta saran kritik yang membangun penulis harapkan.

Surabaya, **Juli 2018**

Riza Kamelia

-----Halaman ini sengaja dikosongkan-----

# DAFTAR ISI

HALAMAN

# DAFTAR GAMBAR

HALAMAN

# DAFTAR TABEL

HALAMAN

-----Halaman ini sengaja dikosongkan-----

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Ranjang pasien merupakan hal yang sangat penting dan harus ada di sebuah rumah sakit. Ranjang pasien yang ada di rumah sakit saat ini sudah ada yang otomatis dengan menggunakan remot kontrol. Jenis ranjang ini biasa disebut dengan ranjang elektronik dimana semua panel dan fungsinya bisa dijalankan dengan menggunakan tombol-tombol yang dioperasikan oleh mesin. Apabila pasien yang menggunakan ranjang ini merupakan pasien dengan kelumpuhan total yang hanya dapat berbicara dan menggerakkan kepala saja, maka pasien tersebut tidak dapat menggerakkan ranjangnya sendiri dengan menggunakan remot kontrol. Mereka harus meminta bantuan orang lain untuk membantu mengatur ranjangnya. Pengaturan tempat tidur dengan menggunakan pengenalan suara dapat memudahkan pasien dengan kondisi lumpuh total dapat mengatur sendiri tempat tidurnya tanpa harus memanggil orang lain.

Penggunaan pengenalan suara atau yang lebih dikenal dengan *voice recognition* telah ada sejak dahulu untuk membantu penyandang disabilitas dalam melakukan kegiatan sehari-hari. Metode pengenalan suara biasanya mengkonversi dan menganalisa sinyal suara dan di kelompokkan kedalam jenis kata dengan diimplementasikan pada program komputer. Sinyal suara memiliki 2 tipe informasi penting yaitu maksud kata yang diucapkan dan pengidentifikasian dari yang mengucapkan. Banyak metode yang digunakan dalam pengenalan suara seperti algoritma genetika, *Dinamic Time Warping* (DTW), *Hidden Markov Model*, *Mel frequency Spectral coefficients* (MFRCs), *Gaussian mixture model* (GMM) dan lain-lain [1].

Beberapa penelitian terdahulu tentang pengontrolan otomatis pada ranjang pasien yaitu oleh A. Alin Linsie menggunakan MEMS akselerometer dengan metode gesture recognition dengan menggerakkan tangan [2]. Sehingga metode ini tidak dapat diimplementasikan untuk penderita lumpuh total. Penelitian lainnya dilakukan oleh Sanur Kumar Das, Vitthal Rathof dan Akhillef Yadaf B. Penelitian ini menggunakan *voice recognition* dengan metode *artificial neural network* dan dikendalikan menggunakan arduino [3].

## 1.2 Permasalahan

Pada Tugas Akhir ini yang menjadi permasalahan utama adalah ranjang pasien umumnya hanya dapat digerakkan dengan remot kontrol, sehingga pasien dengan penyakit lumpuh total yang hanya dapat berbicara dan menggerakkan kepalanya sehingga tidak dapat mengontrol sendiri ranjangnya.

## 1.3 Batasan Masalaha

Batasan masalah pada tugas akhir ini adalah:

1. Pengujian yang dilakukan hanya untuk miniatur ranjang lipat tanpa beban.
2. Pengguna berbicara dengan lancar dan berintonasi normal.
3. Data masukan suara hanya dengan bahasa Indonesia dengan jumlah kata terbatas sebanyak 3 kata yaitu kata oke, kata naik dan kata turun.

## 1.4 Tujuan

Tujuan dari tugas akhir ini adalah untuk mendapatkan sistem pengenalan perintah suara yang digunakan pada pengaturan posisi ranjang lipat.

## 1.5 Metodologi Penelitian

Perancangan miniatur ranjang berbasis pengenalan suara terbagi menjadi empat tahapan, yaitu studi literatur, perancangan sistem, uji coba dan hasil pengujian, serta penyusunan laporan.

Pada tahap studi literatur, dilakukan pencarian literatur buku maupun kumpulan makalah dan jurnal yang mengarah pada topik yang dibahas. Tahapan ini dilakukan untuk mengumpulkan informasi tentang metode yang digunakan dalam pengolahan pengenalan sinyal suara.

Selanjutnya pada perancangan sistem, dibuat suatu sistem pengenalan suara dengan bantuan modul *voice recognition* V3 (VR3) sebagai pengenalan suara dan dibantu dengan mikrokontroler STM32F4 *Discovery* sebagai eksekusi dari perintah suara yang diberikan. Pada tahapan ini membahas tentang koneksi secara perangkat keras maupun secara perangkat lunak.

Setelah itu dilakukan pengujian baik perangkat keras maupun perangkat lunak. Pada tahap ini pengujian perangkat lunak dilakukan untuk menguji seberapa baik algoritma yang diterapkan pada pengenalan

suara, sedangkan pengujian perangkat keras dilakukan agar mengetahui sistem telah berjalan dengan baik atau belum.

## 1.6 Sistematika Laporan

Pembahasan tugas akhir ini dibagi menjadi lima bab dengan sistematika sebagai berikut:

**Bab I Pendahuluan**

Pada bab pendahuluan, menjelaskan mengenai latar belakang pemilihan topik, perumusan masalah dan batasannya. Bab ini juga membahas mengenai tujuan penelitian, metodologi, sistematika laporan, dan relevansi dari penelitian yang dilakukan.

**Bab II Tinjauan Pustaka untuk Perancangan Sistem Pengenalan Suara untuk Pengendalian Ranjang Lipat**

Penjelasan mengenai komponen perangkat keras maupun perangkat lunak pendukung untuk sistem pengenalan suara pada pengendalian ranjang lipat.

**Bab III Perancangan Alat**

Pembahasan yang dilakukan pada bab ini, mengenai perancangan sistem secara keseluruhan dari perancangan perangkat keras berupa perancangan mekanik dan perancangan koneksi rangkaian, serta perancangan perangkat lunak berupa algoritma pemrograman sistem pengenalan suara dan algoritma pemrograman kontrol ranjang lipat tersebut.

**Bab IV Pengujian Perangkat Keras Miniatur Ranjang Lipat**

Pembahasan pada bab ini mengenai hasil dari uji coba rangkaian yang telah dibuat serta pengujian program.

**Bab V Penutup**

Pada bagian bab penutup, dibahas mengenai kesimpulan dan saran dari hasil pengujian.

## 1.7 Relevansi

Hasil yang diperoleh dari tugas akhir ini menjadi langkah awal perencanaan purwarupa (*prototype*) ranjang lipat untuk pasien baik pasien secara umum maupun secara khusus seperti pasien yg menderita kelumpuhan organ tubuh namun masih dapat berbicara secara lancar.

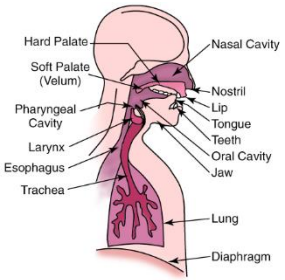
# BAB II
# TEORI DASAR

Pada bab ini akan dibahas mengenai tinjauan pustaka dari proses pengenalan suara. Mulai dari mekanisme produksi sinyal suara manusia, sinyal suara dapat didengar oleh telinga, pengambilan sinyal suara, penyamplingan dan digitalisasi sinyal suara hingga sinyal suara yang dikatakan manusia dapat dikenali oleh mikrokontroler.

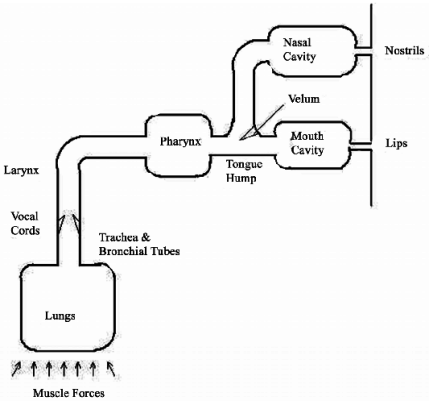
## 2.1 Mekanisme Produksi Sinyal Suara Manusia

Sebagian besar manusia berkomunikasi sesama manusia dengan bahasa yang saling dimengerti melalui ucapan yang dikeluarkan dari mulut. Sekarang ini tidak hanya komunikasi antar manusia, namun manusia dan mesin juga saling berkomunikasi lewat ucapan seperti yang dikembangkan oleh produsen Google yaitu Google Speech-to-text ataupun yang telah dikembangkan oleh Apple yaitu Siri dan lain sebagainya. Sehingga untuk memberikan pendekatan sistem pengenalan suara untuk mesin, maka pemahaman mengenai pembangkitan sinyal suara di tubuh manusia hingga menghasilkan kata yang dapat dipahami oleh manusia lainnya.

Organ manusia yang digunakan untuk menghasilkan suara seperti pada Gambar 2.1 adalah paru-paru (*lungs*), tenggorokan (*Trachea*), laring (*larynx*), faring (*pharyngeal Cavity*), rongga mulut (*Oral Cavity*), dan hidung (*nose*). Apabila dimodelkan dalam blok diagram, hasil pemodelan dari produksi sinyal suara seperti pada Gambar 2.2.




**Gambar 2.1 Sistem Produksi Suara Manusia [4]**




**Gambar 2.2 Blok Diagram Pemodelan Sinyal Suara [4]**

Otot (*muscle force*) menekan udara dari paru-paru menuju laring, dimana tekanan udara menyebabkan *vocal cord* bergetar. Getaran ini mendorong udara dan menghasilkan gelombang tekanan quasi periodik. Frekuensi sinyal tekanan (*pitch impulse*)merupakan frekuensi pitch atau frekuensi dasar. Frekuensi dari sinyal tekan merupakan bagian dari melodi sinyal suara. *Pitch impulse* merangsang udara di saluran oral dan untuk suara tertentu juga saluran hidung. Saat rongga bergema, mereka memancarkan gelombang suara yang merupakan sinyal ucapan. Kedua saluran (Vocal dan nasal) bertindak sebagai resonator dengan frekuensi resonansi karakteristik yang disebut frekuensi formant. Hal ini dimungkinkan untuk mengubah rongga mulut dengan memotong rahang, lidah, velum, bibir dan mulut. Karena itu beberapa suara yang berbeda dapat diucapkan.

## 2.2 Sampling dan Kuantisasi

Sinyal suara yang nantinya diolah oleh mikrokontroler berupa sinyal analog. Sedangkan mikrokontroler hanya dapat mengolah sinyal digital. Karena itu, dibutuhkan konverter sinyal analog ke digital (ADC). Untuk mengkonversi sinyal analog ke digital ada 2 buah metode yang diterapkan, yaitu sampling dalam waktu dan kuantisasi dalam amplitudo.

Pada proses sampling, sinyal kontinyu dalam waktu ($T_s$) dengan sinyal sample (x[n]) dan sinyal kontinyu (x(t)) dapat direpresentasikan pada persamaan 2.1.

$X[n] = x(nT_s)$ (Persamaan 2.1)

Efek dari persamaan 2.1 dalam domain frekuensi yang dilihat dari proses pengambilan sampel adalah membuat versi spektrum sinyal berulang secara periodik pada kelipatan frekuensi sampling $f_s = 1/T_s$. Sehingga hubungan ini di tulis seperti pada persamaan 2.2.

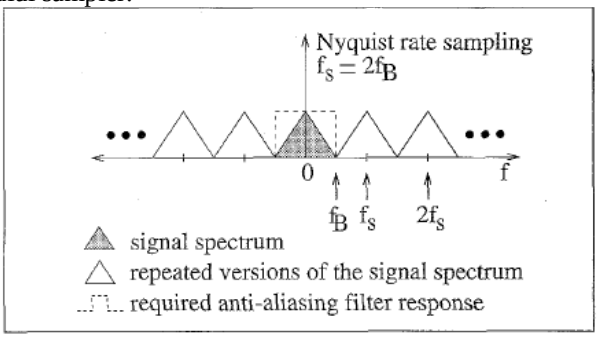
$$X_s(f) = \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X(f - kf_s) \quad \text{(Persamaan 2.2)}$$

Dimana :

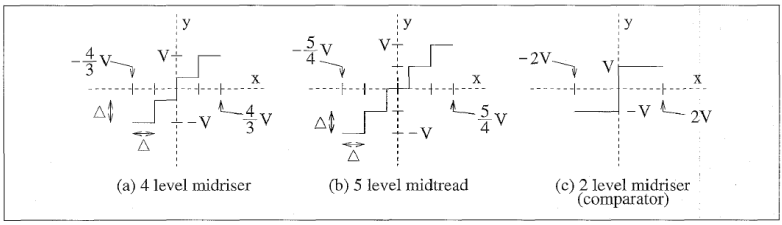
$X_s(f)$ = Spektrum sinyal sample

$X(f)$ = Spektrum sinyal asli

Proses sampling ditunjukkan dalam Gambar 2.3, dimana $f_s = 2f_b$ dan $f_b$ merupakan bandwidth sinyal dan $f_s$ adalah sinyal sampling. Secara umum, sinyal dapat dikembalikan ke domain waktu apabila spektrum sinyal tidak tumpang tindih dan juga penyamplingan harus dua kali lebih besar dari bandwidth sinyal informasi. Terjadinya interferensi sinyal (sinyal tumpang tindih) antara spektrum sinyal yang berulang disebut sebagai aliasing. Aliasing ini mencegah terjadinya rekonstrusi sinyal.
Bandwidth sinyal dibatasi oleh frekuensi sebesar $f_s/2$ dan filter anti-aliasing terkadang juga digunakan untuk memastikan pembatasan sinyal sudah sesuai. Misalnya, sinyal suara memiliki bandwidth sebesar 4 kHz dan prinsipnya di sampling dengan frekuensi 8 kHz. Namun, ada beberapa sinyal yang masuk diatas 4 kHz sehingga menghasilkan sinyal aliasing dengan frekuensi sampling 8 kHz. Maka, filter anti-aliasing diterapkan. Filter anti-aliasing adalah filter analog kontinyu yang mendahului sampler.




**Gambar 2.3 Spektrum Sinyal Sampling Nyquist [5]**




**Gambar 2.4 Karakteristik Kuantisasi ADC [4]**

Setelah mensampling sinyal, sinyal sampling juga harus dikuantisasi dalam amplitud ke sejumlah nilai output yang terbatas. Karakteristik dari kuantisasi ADC dengan sinyal input x[n] dan output y[n] ditunjukkan pada Gambar 2.4. Kuantisasi adalah proses yang tidak dapat dibalik, karena jumlah nilai amplitudo masukan yang tak terbatas dipetakan ke sejumlah nilai amplitudo keluaran yang terbatas. Kuantisasi amplitudo keluaran biasanya direpresentasikan dengan kode digital yang terdiri dari sejumlah bit terbatas. Contohnya, untuk 1 bit ADC seperti pada Gambar 2.4c level keluaran V dan -V di petakan ke kode digital '1' dan '0'. Kode digital ini dapat disebut sebagai format *pulse code modulation* (PCM).

Sebuah ADC atau quantizer dengan tingkat keluaran Q dikatakan memiliki N bit resolusi dimana N = log2 (Q). Untuk nilai ADC dengan tingkat kuantisasi Q dengan nilai masukan sekurang-kurangnya $\Delta$ = 2V/(Q-1) maka dapat dibedakan ke tingkat keluaran yang berbeda-beda.

## 2.3 Transformasi Fourier

Dalam menganalisa sinyal analog dalam domain frekuensi sering menggunakan Transformasi Fourier. Sedangkan dalam menganalisa sinyal digital (sinyal diskrit) biasa menggunakan metode FFT (*Fast Fourier* Transform) dimana algoritma ini merupakan pengembangan dari algoritma DFT (*discrete fourier transform*).

Pada transformasi fourier, variabel waktu (t) dan frekuensi ($\omega$) merupakan nilai yang bervariasi antar $-\infty$ hingga $\infty$. Persamaan dari transformasi fourier sinyal analog seperti pada persamaan 2.3 dalam domain waktu dan persamaan 2.4 dalam domain frekuensi.

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \qquad \text{(Persamaan 2.3)}$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega \qquad \text{(Persamaan 2.4)}$$

Sedangkan transformasi fourier diskrit dengan komponen $X(e^{j\omega})$ didefinisikan pada persamaan 2.5 dengan nilai x(n) harus dapat dijumlahkan secara absolut dengan interval n seperti persamaan 2.6.

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \quad \text{(Persamaan 2.5)}$$

$$\sum_{n=-\infty}^{\infty} |x(n)| = s < \infty \quad \text{(Persamaan 2.6)}$$

Komponen dari transformasi fourier diskrit memiliki pasangan seperti pada Tabel 2.1.

**Tabel 2.1 Sinyal Diskrit dan Transformasi Fouriernya[6]**

| No | Sinyal diskrit | Transformasi Fpurier |
|---|---|---|
| 1 | $\delta(n)$ | $1$ |
| 2 | $\delta(n-d)$ | $e^{-j\omega d}$ |
| 3 | $1\ (-\infty < n < \infty)$ | $\sum_{k=-\infty}^{\infty} 2\pi\delta(\omega + 2\pi k)$ |
| 4 | $a^n u(n)\ (\|a\| < 1)$ | $\frac{1}{1-ae^{-j\omega}}$ |
| 5 | $u(n)$ | $\frac{1}{1-ae^{-j\omega}} + \sum_{k=-\infty}^{\infty} \pi\delta(\omega + 2\pi k)$ |
| 6 | $(n+1)a^n u(n)\ (\|a\| < 1)$ | $\frac{1}{(1-ae^{-j\omega})^2}$ |
| 7 | $\frac{r^n \sin\omega_0(n+1)}{\sin\omega_0} u(n) \quad (\|r\| < 1)$ | $\frac{1}{1-2r\cos\omega_0 e^{-j\omega} + r^2 e^{-j2\omega}}$ |
| 8 | $\frac{\sin\omega_c n}{\pi n}$ | $H(e^{j\omega}) = \begin{cases} 1 & \|\omega\| \le \omega_c \\ 0 & \omega_c < \|\omega\| \le \pi \end{cases}$ |
| 9 | $x(n) = \begin{cases} 1 & 0 \le n \le M \\ 0 & n \text{ lainnya} \end{cases}$ | $\frac{\sin[\omega(M+1)/2]}{\sin(\omega/2)} e^{-j\omega M/2}$ |
| 10 | $e^{j\omega_0 n}$ | $\sum_{k=-\infty}^{\infty} 2\pi\delta(\omega - \omega_0 + 2\pi k)$ |
| 11 | $\cos(\omega_0 n + \emptyset)$ | $\sum_{k=-\infty}^{\infty} \pi[e^{j\emptyset}\delta(\omega - \omega_0 + 2\pi k) + e^{-j\emptyset}\delta(\omega + \omega_0 + 2\pi k)]$ |

Ada 2 jenis transformasi fourier, yaitu *Discrete Fourier Transform* (DFT) dan *Fast Fourier Transform* (FFT). Kedua transformasi ini akan dijelaskan pada sub-bab berikut.

### 2.3.1 *Discrete Fourier Transform* (DFT)

*Discrete Fourier Transform* (DFT) adalah transformasi fourier yang digunakan untuk menganalisa sinyal diskrit yang diketahui sebanyak N dengan sampling waktu sebanyak T dalam jumlah yang terbatas. Semisal dengan sinyal kontinyu f(t) dimana terdapat N *sample* data dan diberikan notasi f[0], f[1], f[2], ... , f[k], ... , f[N-1]. Transformasi fourier dari sinyal f(t) seperti pada persamaan 2.7.

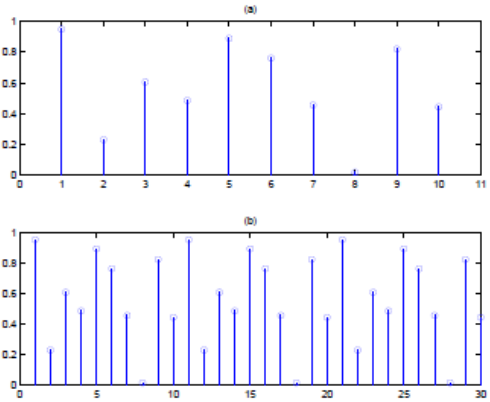
$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \qquad \text{(Persamaan 2.7)}$$

Setiap *sample* (f[k]) dapat dianggap sinyal *impulse* dengan area sebesar f[k], kemudian integral hanya ada di titik sample, maka persamaan menjadi persamaan 2.8.

$$F(j\omega) = \int_{0}^{(N-1)} f(t)e^{-j\omega t}dt$$
$$= f[0]e^{-j0} + f[1]e^{-j\omega T} + \cdots + f[k]e^{-j\omega kT} + \cdots + f(N-1)e^{-j\omega(N-1)T}$$

Sehingga $F(j\omega) = \sum_{k=0}^{N-1} f[k]e^{-j\omega kT}$ (Persamaan 2.8)

Pada prinsipnya, DFT dapat mengevaluasi data sebanyak N secara terus menerus seperti f(N) data hingga f(2N-1) atau f(0) hingga f(N-1). Contoh urutan periode dalam DFT seperti pada gambar dibawah.




**Gambar 2.5 a. Urutan Sample Sebanyak N = 10 dan b. Periode dalam DFT[7]**

Dalam DFT memerlukan data yang dianggap periodik, maka dalam mengevaluasi data dengan DFT lebih mudah dengan menggunakan domain frekuensi dan harmonisanya. Untuk itu dalam menentukan set data dalam frekuensi domain (ω) maka dapat ditulis seperti persamaan 2.9 dan persamaan DFT menjadi seperti pada persamaan 2.10.

$$\omega = 0, \frac{2\pi}{NT}, \frac{2\pi}{NT} x2, \ldots \frac{2\pi}{NT}\ x\ n, \ldots \frac{2\pi}{NT}\ x\ (N-1) \qquad \text{(Persamaan 2.9)}$$

Sehingga

$$F[n] = \sum_{k=0}^{N-1} f[k] e^{-j\frac{2\pi}{N}nk} \quad (n = 0 : N-1) \qquad \text{(Persamaan 2.10)}$$

Dimana f[n] merupakan DFT dari urutan data sebanyak f[k]. Dari persamaan 8, maka DFT dapat dihitung menggunakan matriks seperti dibawah:

$$\begin{pmatrix} F[0] \\ F[1] \\ F[2] \\ \vdots \\ F[N-1] \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & W & W^2 & W^3 & \ldots & W^{N-1} \\ 1 & W^2 & W^4 & W^6 & \ldots & W^{N-2} \\ 1 & W^3 & W^6 & W^9 & \ldots & W^{N-3} \\ \vdots & & & & & \\ 1 & W^{N-1} & W^{N-2} & W^{N-3} & \ldots & W \end{pmatrix} \begin{pmatrix} f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[N-1] \end{pmatrix}$$

Dimana W = $e^{(-j2\frac{\pi}{N})}$ dan W = $W^{2N}$ dan seterusnya = 1.

### 2.3.2 *Fast Fourier Transform* (FFT)

Waktu yang dibutuhkan untuk mengevaluasi DFT pada komputer tergantung pada jumlah perkalian yang terlibat, karena ini adalah operasi paling lambat. Dengan DFT, nomor ini secara langsung terkait dengan n2 (perkalian matriks vektor), di mana N adalah panjang transformasi. Untuk sebagian besar masalah, N dipilih untuk setidaknya 256 untuk mendapatkan perkiraan yang masuk akal untuk spektrum urutan yang dipertimbangkan - maka kecepatan komputasi menjadi pertimbangan utama.

Algoritma komputer yang sangat efisien untuk mengestimasi Discrete Fourier Transforms telah dikembangkan sejak pertengahan tahun 60-an. Ini dikenal sebagai algoritma Fast Fourier Transform (FFT) dan mereka bergantung pada fakta bahwa DFT standar melibatkan banyak perhitungan yang berlebihan seperti pada persamaan 2.11.

$$F[n] = \sum_{k=0}^{N-1} f[k] e^{-j\frac{2\pi}{N}nk} \text{ as } F[n] = \sum_{k=0}^{N-1} f[k] W_N^{nk}$$

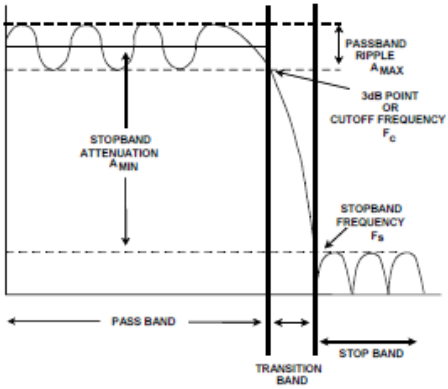
(Persamaan 2.11)

Mudah untuk menyadari bahwa nilai-nilai yang sama dari w dihitung berkali-kali sebagai hasil perhitungan. Pertama, produk integer nk mengulangi untuk kombinasi k dan n yang berbeda. kedua, W adalah fungsi periodik dengan hanya N nilai yang berbeda.

## 2.4 Filter Analog

Dalam pengolahan sinyal suara, filter dibutuhkan untuk menghilangkan frekuensi derau (*noise)*. Filter analog digunakan untuk memfilter sinyal input analog menjadi sinyal output yang diinginkan. Filter yang ideal akan memiliki respon dengan penguatan 1 pada frekuensi yang di tunjuk (*pass band*) dan penguatan 0 pada frekuensi yang lain (*stop band*). Perubahan respon dari *pass band* menuju *stopband* disebut frekuensi *cut-off*. Dalam filter, terdapat 5 parameter yang digambarkan seperti pada Gambar 2.6.

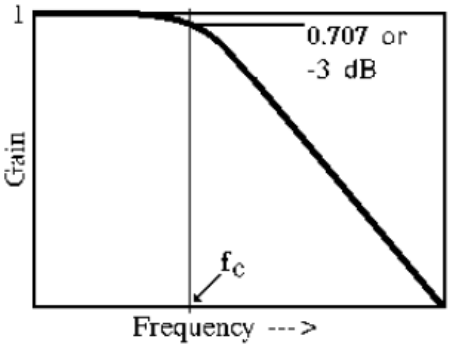
Pada Gambar 2.6, frekuensi *cut-off* ($F_c$) merupakan respon frekuensi yang difilter meninggalkan *error band* (untuk *butterworth* filter berada pada -3dB). Frekuensi *stopband* ($F_s$) merupakan frekuensi dimana atenuasi minimum pada *stopband* tercapai. *Ripple pass band* ($A_{max}$) merupakan variasi (*error band*) pada respon *pass band*. Atenuasi minimum pass band ($A_{min}$) didefinisikan sebagai sinyal atenuasi minimum dalam *stop band*. Kecuraman filter didefinisikan sebagai urutan (M) filter. M juga didefinisikan sebagai nilai *pole* pada fungsi transfer. *Pole* merupakan akar penyebut di *transfer function.* Sebaliknya, *zero* merupakan akar pembilang dari *transfer function*. [8] Jenis-jenis respon dari filter analog adalah sebagai berikut.




**Gambar 2.6 Parameter Filter [8]**

### 2.4.1 *Low Pass Filter* (LPF)

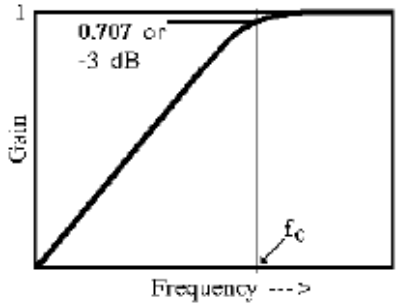
Filte LPF merupakan filter yang meloloskan sinyal dibawah frekuensi *cut-off*. Penggambaran respon dari LPF seperti pada Gambar 2.7. Penguatan LPF adalah 1 yang turun sebanyak 0,707 atau -3dB sampai sudut frekuensi dan terus menerus turun selama frekuensi terus naik.




**Gambar 2.7 Respon Low Pass Filter[9]**

### 2.4.2 *High Pass Filter* (HPF)

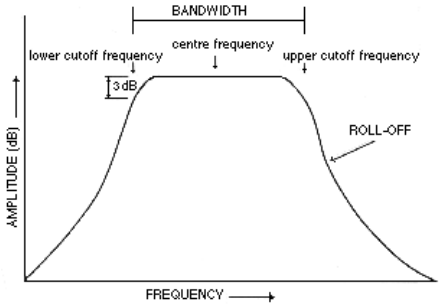
*High pass filter* merupakan kebalikan dari *low pass filter*, dimana akan melewatkan frekuensi diatas frekuensi *cut-off*. Respon dari HPF seperti pada Gambar 2.8. Penguatan HPF adalah 1 yang turun sebanyak 0,707 atau -3dB sampai sudut frekuensi dan terus menerus turun selama frekuensi masukan semakin kecil. Sehingga selama frekuensi masukan dibawah frekuensi *cut-off* maka tidak akan dilewatkan oleh HPF.




**Gambar 2.8 Respon frekuensi High Pass Filter[9]**

### 2.4.3 Band Pass Filter

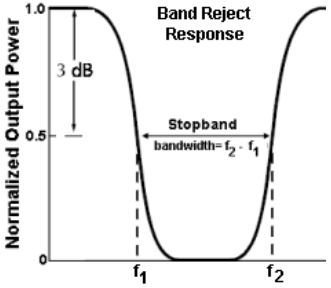
*Band pass filter* bekerja dengan meloloskan sinyal dengan frekuensi yang berada pada *bandwidth* (lebar bidang frekuensi) tertentu. Pada Gambar 2.9 bandwidth didapat dari frekuensi *cutoff* bawah (*lower cutoff frequency*($\omega_{c2}$)) – frekuensi *cutoff* atas (*upper cutoff frequency* ($\omega_{c1}$)). Sedangkan frekuensi $\omega_o$ dimana amplitido tertinggi yang diloloskan disebut sebagai frekuensi tengah (*centre frequency*). Frekuensi yang diloloskan didefinisikan sebagai $\omega_{c1} \leq \omega_o \leq \omega_{c2}$.[10]




**Gambar 2.9 Respon Frekuensi Band Pass Filter [11]**

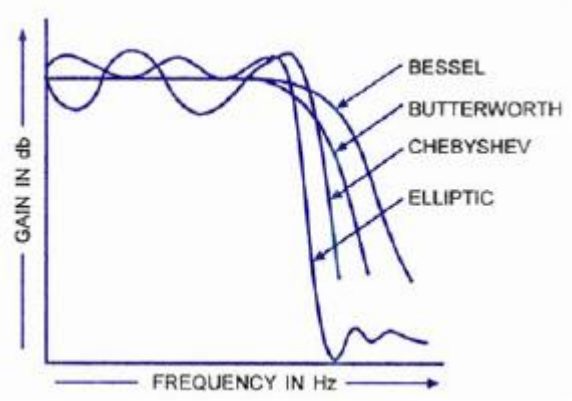
### 2.4.4 *Band Stop* (*Notch*) Filter

*Band stop filte*r merupakan kebalikan dari *band pass filter* dimana filter ini meloloskan sinyal diluar frekuensi *bandwith*. Seperti pada Gambar 2.10, filter ini akan meloloskan sinyal dengan frekuensi di bawah $f_1$ dan diatas $f_2$.




**Gambar 2.10 Respon Frekuensi Band Stop Filter [12]**

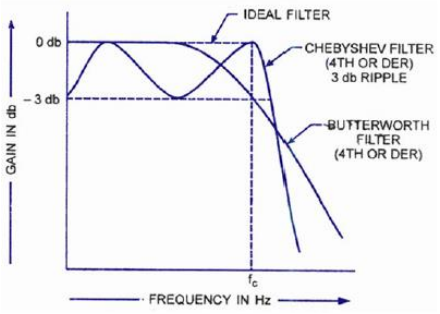
Jenis filter selain berdasarkan responnya, juga dapat dilihat berdasarkan ketajamannya. Jenis filter berdasarkan ketajamannya dibagi menjadi *butterworth* filter, *chebyshev* filter, *bessel* filter dan *elliptic* filter. Frekuensi respon dari ke-4 filter ini seperti pada Gambar 2.11.




**Gambar 2.11 Kurva Frekuensi Respon[13]**

### 2.4.5 Filter *Butterworth*

Filter ini juga disebut sebagai flat filter. Kelas filter ini mendekati filter ideal dengan baik di frekuensi *band pass*. Filter Butterworth memiliki respons frekuensi amplitudo rata-rata datar hingga frekuensi *cut-off*. Ketajaman *cut-off* dapat dilihat pada Gambar 2.12. Perlu dicatat bahwa ketiga filter mencapai kemiringan roll-off -40 db / dekade pada frekuensi yang jauh lebih besar daripada cut-off. Meskipun filter Butterworth mencapai redaman paling tajam, pergeseran fasa mereka sebagai fungsi frekuensi adalah non-linear. Filter ini memiliki penurunan monoton dalam gain dengan frekuensi di wilayah cut-off dan respon datar maksimal di bawah frekuensi cut-off, seperti yang diilustrasikan pada Gambar 2.12. Filter Butterworth memiliki karakteristik di suatu tempat antara filter Chebyshev dan Bessel. Ini memiliki roll-off moderat dari rok dan respon fase yang kurang linear.




**Gambar 2.12 Filter Butterworth dan Filter Chebyshev[13]**

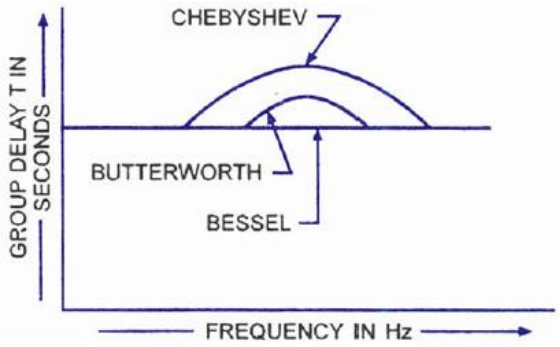
### 2.4.6 Filter *Chebyshev*

Ini juga disebut filter riak yang sama dengan filter Butterworth. Filter ini memberikan cut-off yang lebih tajam daripada filter Butterworth di passband. Filter Butterworth dan Chebyshev menunjukkan pergeseran fase yang besar di dekat frekuensi cut-off. Kelemahan filter Chebyshev adalah munculnya gain maksimum dan minimum di bawah frekuensi cut-off.

Karakteristik dari filter Chebyshev adalah semakin cepat roll-off, semakin besar puncak-ke-puncak riak di passband. Respon fase sangat non-linear di wilayah steady state. Penundaan yang tidak sama dari frekuensi data dalam passband menyebabkan distorsi pulsa yang parah dan dengan demikian meningkatkan kesalahan pada demodulator. Hal ini dapat diatasi dengan meningkatkan Bandwidth filter sehingga daerah fase diperpanjang. Filter Chebyshev digunakan di mana roll-off sangat tajam diperlukan. Namun, ini dicapai dengan mengorbankan riak keuntungan di frekuensi yang lebih rendah.

### 2.4.7 Filter *Bessel*

Filter Bessel memberikan karakteristik fase ideal dengan respons fase linier hingga frekuensi hampir terpotong. Meskipun memiliki respon fase yang sangat linier tetapi memiliki kemiringan respon yang cukup lembut, seperti yang ditunjukkan pada Gambar 2.13. Untuk aplikasi di mana karakteristik fase penting, filter Bessel biasa digunakan. Filter Bessel adalah adalah filter pergeseran fase minimal dengan karakteristik cut-off yang tidak terlalu tajam, sehingga sangat cocok untuk aplikasi pulsa.

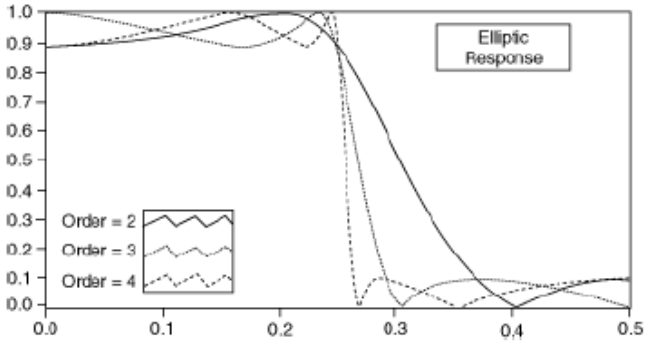



**Gambar 2.13 Respon Frekuensi Filter Bessel[13]**

### 2.4.8 Filter *Elliptic*

Filter ini memiliki roll-off paling tajam dari semua filter di wilayah transisi tetapi memiliki riak di daerah band pass dan stop band, seperti yang diilustrasikan pada Gambar 2.14. Filter elips dapat dirancang untuk memiliki atenuasi yang sangat tinggi untuk frekuensi tertentu di stop band dan mengurangi atenuasi untuk frekuensi lain di stop band.

Namun, jika yang menjadi perhatian utama adalah meneruskan frekuensi yang jatuh dalam pita frekuensi tertentu dan memblokir frekuensi di luar pita frekuensi tersebut dan terlepas dari pergeseran fasa, maka respons eliptik akan melakukan fungsi itu dengan filter urutan yang terendah.




**Gambar 2.14 Respon Frekuensi Filter Elliptic[14]**

## 2.5 Filter Digital

Filter digital digunakan untuk memfilter sinyal digital yang dikarakterisasi dengan persamaan beda koefisien konstan linear orde ke-N dan dikelompokkan menjadi filter *Finite Impulse Response* (FIR) dan filter *Infinite Impulse Response* (IIR). Sinyal digital merupakan sinyal analog yang telah di rubah menjadi sinyal diskrit atau sinyal dengan respon impuls.

Filter digital adalah linear time invarian (LTI) dengan proses mencari respon impuls h(n) yang dapat diperoleh dengan melakukan transformasi z sehingga mendapatkan sistem H(z). Respon frekuensi H(z) diubah menjadi $H(e^{j\omega})$ untuk menentukan jenis filter yang digunakan berupa LPF, HPF, BPF, atau BSF. Pengaplikasian filter digital seperti pada Gambar 2.15 dimana masukan analog x(t) diproses dengan ADC dengan sampling 1/T sehingga menjadi sinyal digital x(n). Selanjutnya sinyal x(n) di rubah dengan transformasi z menjadi X(z) dan di konvolusi dengan filter H(z) menghasilkan keluaran Y(z) yang dirubah menjadi y(n). Kemudian sinyal digital y(n) dirubah menjadi sinyal analog y(t) dengan DAC (*Digital to Analog Converter*). [6]

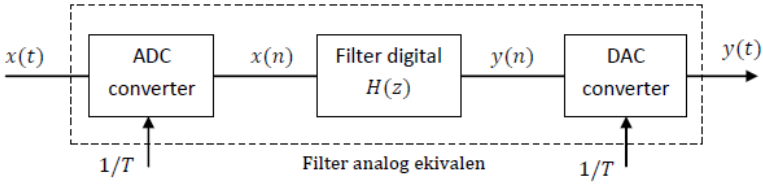
Dalam filter digital, strukturnya merupakan cascade dan kombinasi paralel dari komponen yang mengandung orde 2 dan ditentukan dari persamaan beda, yaitu dalam domain waktu seperti pada persamaan 2.12 dan persamaan 2.13.

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \qquad \text{(Persamaan 2.12)}$$

Persamaan beda :

$$y(n) = \sum_{k=1}^{M} a_k y(n-k) + \sum_{k=-NF}^{Np} b_k x(n-k) \qquad \text{(Persamaan 2.13)}$$

Filter digital dapat dikategorikan dalam 2 kategori, yaitu *infinite impulse response* (IIR) dan *Finite impulse response* (FIR). Filter IIR merupakan filter yang menghasilkan respon sinyal diskrit dengan durasi yang tak terhingga. Sedangkan filter FIR sama seperti filter IIR namun memiliki durasi yang terbatas.[15]




**Gambar 2.15 Blok Diagram Filter Analog Ekivalen [6]**

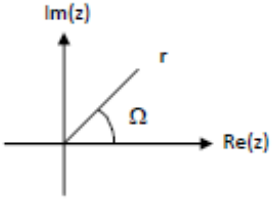
## 2.6 Transformasi Z

Transformasi Z merupakan suatu metode analisa sinyal diskrit sebagai transformasi laplace dalam analisa waktu kontinyu. Transformasi Z dalam sinyal diskrit didefinisikan sebagai deret pangkat seperti pada persamaan 2.14.

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n} \qquad \text{(Persamaan 2.14)}$$

Dimana z adalah suatu variabel bilangan komplek z = $re^{j\Omega}$. Dalam penggambaran grafik, antara bilangan real dan imajiner, transformasi z digambarkan seperti Gambar 2.16.

Seperti disebutkan sebelumnya, bahwa transformasi z merupakan deret pangkat yang nilainya tak berhingga, maka transformasi z hanya berlaku untuk nilai dengan deret konvergen. Daerah konvergensi dari masukan X(z) merupakan himpunan dari seluruh nilai z agar X(z) nantinya mencapai nilai yang tak hingga.[16]




**Gambar 2.16 Grafik Bilangan Real dan Imajiner dari Transformasi z**

## 2.7 Voice Recognition Module V3 [17]

Dalam mengimplementasikan pengenalan suara pada tugas akhir ini, digunakan modul *voice recognition* dari Elechouse berbasis chip serial dan memory flash winbond W25Q16DV. Modul ini dapat mendukung 80 perintah suara, namun hanya dapat melaksanakan 7 perintah suara yang dapat bekerja secara bersamaan dengan lebar tiap kata sebesar 1500ms. Perintah suara dalam modul ini akan disimpan dalam 1 kelompok seperti sebuah library. Tiap 7 perintah suara dapat diimpor ke *recognizer*. Ini berarti ada 7 perintah yang aktif pada waktu yang bersamaan. Bentuk fisik dari *voice recognition module v3* (VR3) ini seperti pada Gambar 2.17.

Modul *voice recognition* ini dapat diberi *supply* tegangan sebesar 4,5 volt hingga 5 volt dengan arus kurang dari 40 mA. Antarmuka digital dari modul ini dapat berupa 5 volt TTL UART (*Universal Asynchronous*

*Receiver-Transmitter*) dan GPIO (*General Purposed Input - Output*). Untuk antarmuka analognya adalah berupa *microphone mono* 3,5mm dan pin *microphone interface*. Ukuran modul ini lumayan kecil yaitu sebesar 31mm x 50 mm dengan akurasi pengenalan kata sebesar 99% dalam keadaan ideal.

Metode dalam menggunakan modul *voice recognition* ini ada 2, yaitu dengan *library* arduino dan dengan VR3 protokol. Untuk mengkoneksi modul ini dengan arduino adalah seperti Tabel 2.2 berikut.

**Tabel 2.2 Koneksi Arduino dan Modul VR3**

| Arduino | VR Modul |
|---|---|
| 5V | 5V |
| Pin 2 | Tx |
| Pin 3 | Rx |
| GND | GND |

Dalam mentrain modul VR3 dengan arduino yaitu dengan membuka *sketch* vr_sample_train di arduino (File – Examples – VoiceRecognitionV3 – vr_sample_train) kemudian setting *board* arduino dan serial port sesuai *board* yang digunakan. Setelah itu klik tombol *upload*, tunggu hingga arduino telah selesai *upload*. Buka serial monitor dan atur baudrate sebesar 115200, atur *set send* dengan ***Newline*** atau ***Both NL & CR***. Untuk men *train* modul VR3 ini dengan mengirim perintah sigtrain – nomor index – nama perintah. Untuk memeriksa *suara* yang telah di *train* kirim perintah *load* – nomor index.

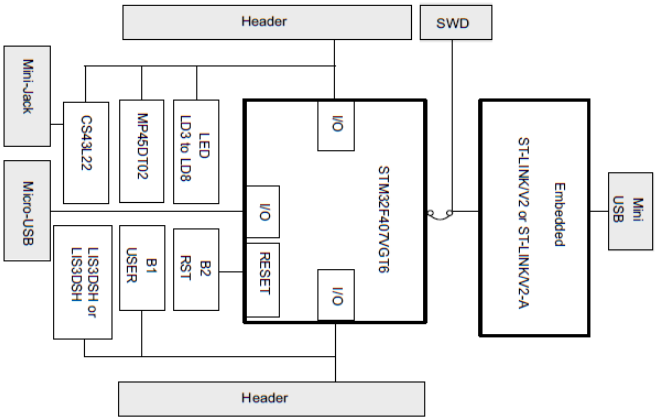
Selain dengan arduino, cara lain untuk memprogram modul VR3 adalah dengan protokol VR3. Protokol VR3 diperuntukkan untuk memprogram modul VR3 selain menggunakan arduino, seperti pada tugas akhir ini yang menggunakan STM32. Semua perintah yang diberikan dengan protokol VR3 ini menggunakan format hexadesimal.

Gambar 2.17 Voice Recognition Module V3

## 2.8 Mikrokontroler STM32

Mikrokontroller adalah kontroller yang digunakan untuk mengatur jalannya sistem. Mikrokontroller sendiri terdiri dari CPU (*Central Processing Unit*), memori, I/O tertentu dan unit pendukung seperti *Analog-to-Digital Converter* (ADC) yang sudah terintegrasi di dalamnya. Pada tugas akhir ini menggunakan mikrokontroller STM32 yang dimana spesifikasi ADC, I2C (*Inter-Integrated Circuit)*, PWM *(Pulse Widht Modulation)* yang jumlahnya cukup untuk pengambilan data maupun menggerakkan aktuator pada sistem. Pada tugas akhir ini akan digunakan mikrokontroller STM32F4 *Discovery* karena dapat digunakan untuk digital filter dan juga memiliki frekuensi *clock* yang lebih besar dibanding Arduino. Blok diagram perangkat keras dari STM32F4 seperti pada Gambar 2.18.

STM32F4 *Discovery* merupakan mikrokontroler berbasis ARM Cortex M4 dan dapat digunakan untuk memproses transformasi *fourier* seperti DFT dan FFT serta dapat digunakan untuk memfilter sinyal berupa filter *Finite Impulse Response* (FIR) yang berkaitan dengan *digital signal processing* dengan lebih baik [18]. STM32F4 memiliki frekuensi *clock* mencapai 168 MHz dengan *Flash memory* mencapai 1 Mbytes dan SRAM mencapai 192+4 Kbytes termasuk 64 Kbytes CCM (*core coupled memory*) RAM data. Tegangan *supply* pada STM32 ini adalah sebesar 3,3 volt dengan toleransi tegangan hingga 5 volt. Untuk port *input* dan *output* mencapai 140 pin.



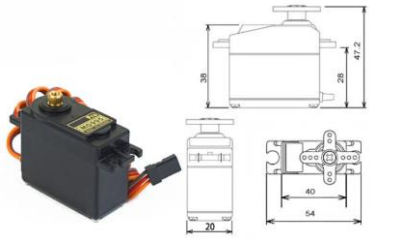

**Gambar 2.18 Blok Diagram Mikrokontroller STM32[19]**

## 2.9 Motor Servo

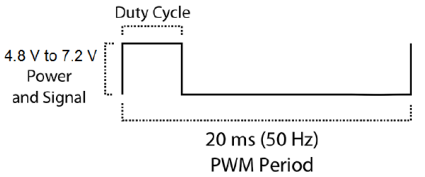
Motor servo adalah motor DC yang dapat digunakan untuk mengatur posisi derajat secara tepat. Bagian dari motor servo terdiri dari motor DC, serangkaian gear, potensiometer dan rangkaian kontrol. Potensiometer digunakan untuk menentukan batas sudut putaran servo. Sedangkan sudut dari sumbu servo diatur berdasarkan lebar pulsa yang masuk ke servo melalui kaki sinyal dari kabel motor. Kecepatan putaran dari motor servo juga diatur oleh rangkaian roda gigi.[20]

Jenis motor servo berdasarkan putarannya ada 2, yaitu standard dan *continous*. Motor servo standard adalah motor servo yang berputar dari 0$^o$ hingga 180$^o$ dan biasa digunakan pada sistem robotika, seperti untuk membuat lengan robot. Sedangkan servo *continous* merupakan servo yang dapat berputar dari 0$^o$ hingga 360$^o$ dan biasa digunakan pada mobile robot. Bentuk Fisik dari motor servo dapat dilihat pada Gambar 2.19.

Sistem kerja dari servo adalah dikendalikan oleh PWM (*Pulse Width Modulation*) dengan periode ±20ms (frekuensi 50 Hz) dengan lebar pulsa antara 1ms hingga 2ms untuk servo standard. Apabila motor servo diberikan pulsa dengan lebar sebesar 1 ms maka posisi yang dihasilkan sebesar 0$^o$. Apabila motor servo diberikan pulsa dengan lebar sebesar 1,5ms maka servo akan berputar 90$^o$, sedangkan jika diberi pulsa dengan lebar 2ms maka posisi servo akan berubah ke 180$^o$. Bentuk pulsa dan perputaran servo dapat dilihat pada Gambar 2.20.




**Gambar 2.19 Bentuk Fisik Motor Servo [21]**



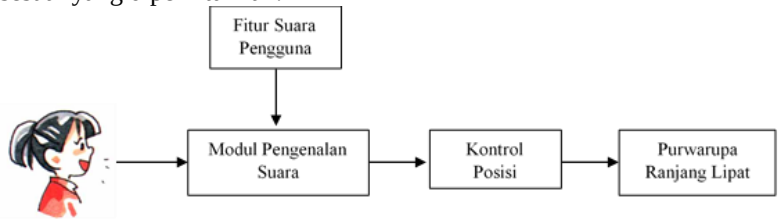

**Gambar 2.20 Lebar Pulsa PWM Motor Servo [21]**

# BAB III
# PERANCANGAN PURWARUPA RANJANG LIPAT

Pada bab ini akan dibahas mengenai perancangan dan pembuatan sistem perancangan pengenalan suara untuk posisi ranjang lipat yang meliputi blok fungsional sistem yang akan menjelaskan proses kerja alat dalam bentuk alur diagram, perancangan mekanik yang membahas tentang desain dan pembuatan mekanik yang mendukung cara kerja alat, perancangan perangkat elektrik yang membahas perancangan rangkaian pendukung alat dan perancangan program.

## 3.1 Blok Fungsional Sistem Purwarupa Ranjang Lipat

Pada tugas akhir ini, diterapkan sistem pengenalan suara untuk mengatur posisi naik dan turun dari suatu purwarupa ranjang lipat agar mempermudah pasien yang tidak dapat menggerakkan tubuhnya dapat dengan mudah menggerakkan ranjangnya dengan perintah suara. Seperti yang dapat dilihat pada Gambar 3.1, sistem ini dimulai dari masukan suara dari pengguna yang ditangkap oleh mikrofon kemudian diproses oleh modul pengenalan suara. Apabila suara yang ditangkap oleh modul pengenalan suara sesuai dengan salah satu perintah suara pada fitur suara pengguna maka modul akan meneruskan ke mikrokontroler STM32 sebagai kontrol posisi untuk diproses sesuai perintah suara yang diberikan. Kemudian dari mikrokontroler diproses untuk menggerakkan posisi servo motor sebagai eksekusi pada purwarupa ranjang lipat agar sesuai yang diperintahkan.

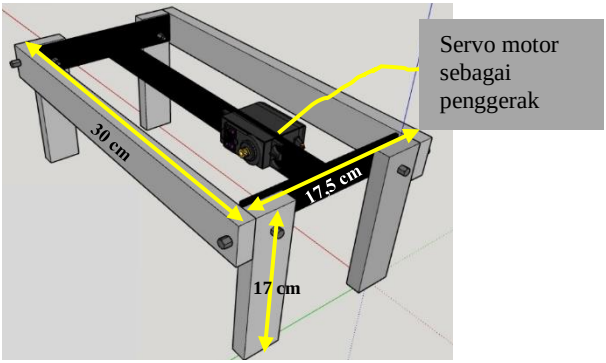



**Gambar 3.1 Blok Fungsional Sistem Pengaturan Posisi Ranjang Lipat**

## 3.2 Perancangan Mekanik Purwarupa Ranjang

Perancangan mekanik dari ranjang lipat ini meliputi perancangan purwarupa ranjang yang terdiri dari motor servo serta ranjang itu sendiri dan peletakan rangkaian kontroler. Dalam perancangan mekanik ini

bertujuan agar purwarupa ini dapat digambarkan seperti ranjang lipat pada aslinya serta agar komponen dapat tertata dengan rapi dan efisien. Gambar perancangan mekanik sistem ini seperti ditunjukkan pada Gambar 3.2. Hasil dari implementasi alat seperti pada Gambar 3.3. Pada Gambar 3.3 (a) ditunjukkan tampilan ranjang lipat tampak dari atas. Sedangkan pada Gambar 3.3 (b) ditunjukkan posis dari purwarupa ranjang lipat dilihat pada posisi samping. Spesifikasi dari perangkat yang digunakan adalah seperti ditunjukan pada Tabel 3.1.




**Gambar 3.2 Perancangan Mekanik Ranjang Lipat**

(a) (b)

**Gambar 3.3 Hasil Implementasi Alat (a) Tampak Atas dan (b) Tampak Samping**

**Tabel 3.1 Spesifikasi Perangkat Sistem Purwarupa Ranjang Lipat**

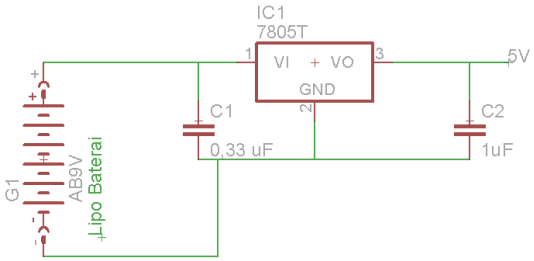
| **Tipe Penggerak Ranjang Lipat** | **Motor Servo MG995** |
|---|---|
| *Variety Power* | DC *Power* |
| Tegangan (*Voltage*) | 6 Volt |
| *Operation Speed* | 0.16 s/$60^0$ |
| *Temperature Range* | $0^0$C – $55^0$C |
| Berat | 55g |
| Dimensi | 40,7 x 19,7 x 42,9 mm |
| *Stall Torque* | 10 kgf.cm |
| **Tipe Kontrol Posisi Ranjang Lipat** | **STM32F4VGT** |
| Jenis Prosesor | 32-bit ARM Cortex®-M4 *with FPU core* |
| Tegangan (*Voltage*) | 3Volt atau 5 Volt DC |
| Dimensi | 66 mm x 97 mm |
| **Tipe Modul Pengenalan Suara** | **Modul *voice recognition* V3** |
| Tegangan (*Voltage*) | 4,5 – 5,5 volt |
| Arus | < 40 mA |
| Dimensi | 31 mm x 50 mm |
| Akurasi Pengenalan | 99% (dibawah kondisi ideal) |

## 3.3 Perancangan Perangkat Elektrik Purwarupa Ranjang Lipat

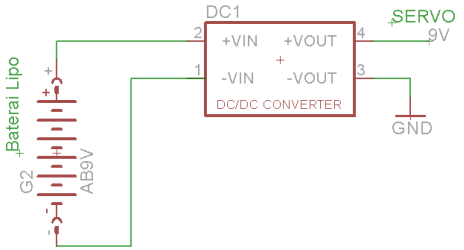
Perancangan perangkat elektrik meliputi *wiring input* dan *output* dari kontroler serta perancangan *supply* tegangan.

### 3.3.1 Perancangan Sumber Tegangan

Pada perancangan ini didesain *supply* untuk masukan tegangan pada STM32, Modul *voice recognition* dan motor servo. Untuk masukan pada STM32 dan modul *voice recognition* dengan tegangan masukan sebesar 5 volt dan berasal dari baterai lipo 11,1 volt kemudian di masukkan ke regulator tegangan 7805. Gambar rangkaian *supply* 5 volt pada Gambar 3.4.




**Gambar 3.4 Koneksi *Supply* 5 volt**




**Gambar 3.5 Koneksi *Supply* 9 volt**

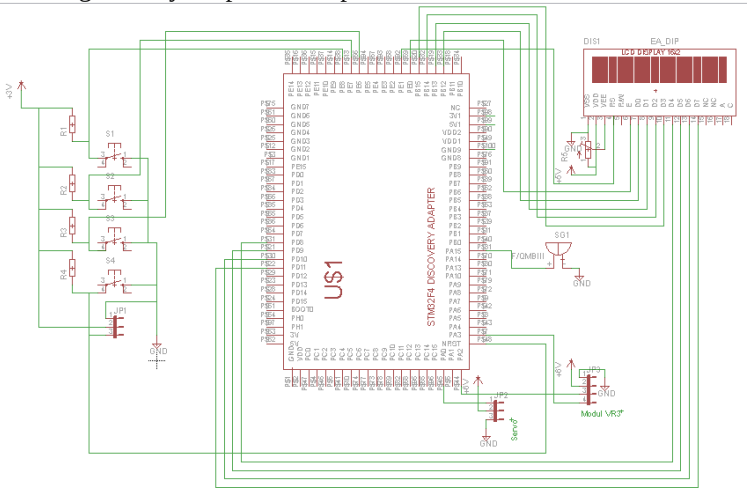
Untuk masukan pada motor servo diberi tegangan sebesar 9 volt dari baterai lipo 11,1 volt yang dihubungkan dengan *buck converter* yang tegangannya diturunkan menjadi 9 volt. Gambar rangkaian *supply* 9 volt dapat dilihat pada Gambar 3.5.

### 3.3.2 Perancangan Rangkaian Kontrol Ranjang Lipat

Pada perancangan ini dirancang masukan STM32 dari modul *voice recognition* dan keluaran STM32 ke servo motor dan *buzzer*. Untuk *wiring* diagramnya dapat dilihat pada Gambar 3.6. Dimana Pin PA2, PA3, Gnd dan 5V terhubung dengan pin modul VR3. Sedangkan pin PA0 terhubung ke pin data dari servo yang berfungsi untuk PWM pada servo motor. Sedangkan pin PA15 terhubung dengan buzzer.

Untuk melakukan pembelajaran pada pasien baru, maka digunakan 3 buah tombol yaitu tombol untuk pembelajaran kata “oke” yang diletakkan pada pin PE6. Sedangkan tombol untuk kata “naik” diletakkan pada pin PE5 dan untuk kata “turun” diletakkan pada pin PE4. Selain itu terdapat 2 buah tombol emergency dimana sebuah tombol untuk operator

yang membantu pasien untuk melakukan pembelajaran pada pasien baru dan tombol lainnya untuk pasien tersebut. Untuk keterangan peletakan pin dan kegunaannya dapat dilihat pada Tabel 3.2.




**Gambar 3.6 Wiring Diagram STM32F4 dengan Masukan dan Keluaran**

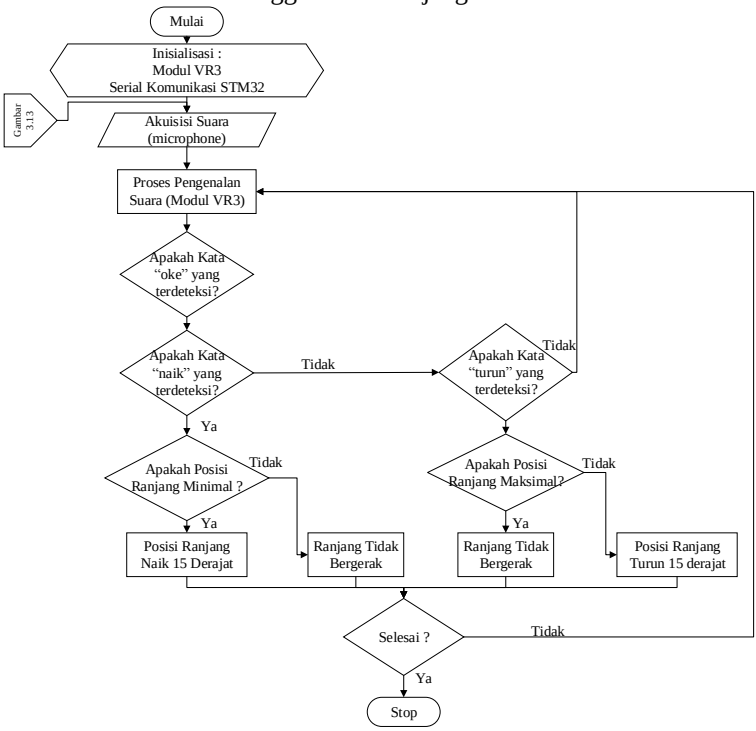
**Tabel 3.2 Koneksi Pin STM32F4 dengan Masukan dan Keluaran**

| Pin STM32F4 | Pin Masukan / Keluaran | Pin STM32F4 | Pin Masukan / Keluaran |
|---|---|---|---|
| NRST | *Emergency / Reset* | PD8 | Pin D4 LCD 16x2 |
| PA0 | Servo | PD9 | Pin D5 LCD 16x2 |
| PA2 | Rx Modul VR3 | PD10 | Pin D6 LCD 16x2 |
| PA3 | Tx Modul VR3 | PD11 | Pin D7 LCD 16x2 |
| PA15 | Buzzer | PE0 | Pin RS LCD 16x2 |
| PB12 | Pin D0 LCD 16x2 | PE1 | Pin E LCD 16x2 |
| PB13 | Pin D1 LCD 16x2 | PE4 | Train Kata “Turun” |
| PB14 | Pin D2 LCD 16x2 | PE5 | Train kata “Naik” |
| PB15 | Pin D3 LCD 16x2 | PE6 | Traik Kata “oke” |

## 3.4 Perancangan Perangkat lunak Purwarupa Ranjang Lipat

Perancangan perangkat lunak berupa perancangan algoritma program dari sistem pengenalan suara pada ranjang lipat. Diagram Alir dari algoritma pengatur posisi ranjang lipat seperti pada Gambar 3.7.

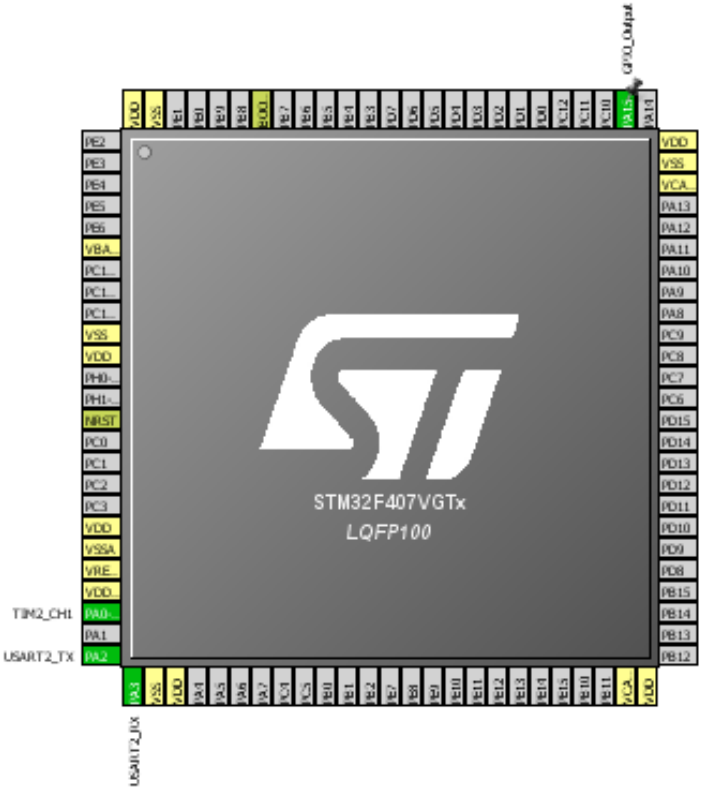
Dari diagram alir pada Gambar 3.7, pengguna dapat mengatur posisi ranjangnya dengan mengucapkan kata “naik” atau kata “turun”. Namun, terdapat inisialisasi kata yang harus diucapkan sebelum mengucapkan kata “naik” atau “turun” yaitu kata “oke”. Hal ini bertujuan agar tidak terjadi kesalahan saat pengguna menggunakan kata “naik” ataupun “turun” selain untuk menggerakkan ranjang.




**Gambar 3.7 Diagram Alir Sistem Pengatur Posisi Ranjang Lipat**

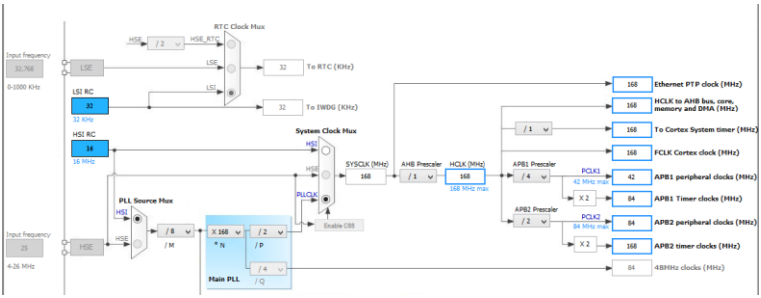
Pada program untuk menggerakkan ranjang dengan menggunakan STM32F4 dengan menuliskan kode program dengan aplikasi KEIL uVision 5 dengan pengaturan *input / output* (I/O) pada CubeMX. Pengaturan yang dilakukan adalah inisialisasi pin yang digunakan seperti pada Gambar 3.8. Pada pengaturan *Input* pin PA2 dan PA3 terkoneksi sebagai USART2 STM32F407 dengan pin PA2 sebagai Tx dan PA3

sebagai Rx. Sedangkan pada *output* pin PA0 sebagai *output* PWM dihubungkan dengan pin data motor servo, sedangkan pin PA15 dihubungkan dengan buzzer.




**Gambar 3.8 Pengaturan *Input* dan *Output* Pada CubeMX**

Setelah itu melakukan pengaturan *clock* STM32F407 seperti pada Gambar 3.9. *Clock* STM32F4 yang digunakan untuk PWM dan koneksi USART2 seperti pada *datasheet* adalah terhubung dengan APB1. Pada APB1 diatur nilai clock sebesar 84 MHz dari PLLCLK dan HCLK diatur pada nilai maksimum sebesar 168MHz.




**Gambar 3.9 Pengaturan *Clock* STM32F4 pada CubeMx**

Kemudian pada konfiigurasi, nilai *clock* pada PWM diturunkan lagi dengan mengatur nilai prescaler dan period. Karena Motor servo bekerja pada periode 20 ms atau 50 Hz maka dilakukan perhitungan seperti pada persamaan 3.2.

$$F = \frac{TIM\_CLK}{(prescaler + 1)x(period + 1)}$$

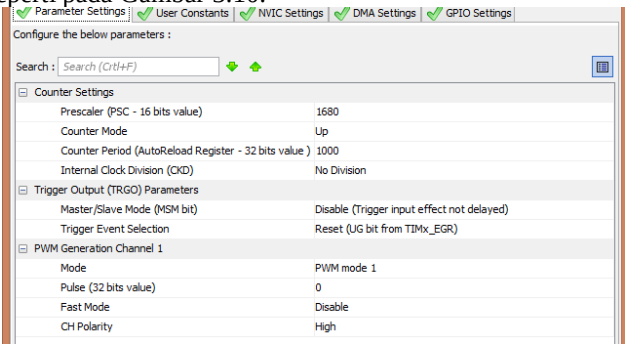
(Persamaan 3.1)

Nilai prescaler dapat diisi sebanyak 16 bit dari 0 hingga 65535. Untuk mendapatkan nilai 50 Hz, maka dihitung seperti pada persamaan 3.3.

$$\frac{84MHz}{50Hz} = 1680\ KHz$$

(Persamaan 3.2)

Sehingga nilai prescaler diisi dengan 1680 dan pada periode diisi dengan 1000 seperti pada Gambar 3.10.




**Gambar 3.10 Konfigurasi Timer 2 STM32F407**

Pada program KEIL uVision 5, nilai PWM di atur setelah melakukan pengujian nilai minimum PWM agar posisi ranjang lipat bergerak sesuai dengan yang diinginkan. Pengaturan nilai PWM dihitung berdasarkan waktu *pulse width* yang diubah dengan nilai yang dimasukkan ke CCR1 sebagai PWM channel 1 STM32F4 yang dihitung seperti pada persamaan 3.4.

$$CCR1 = \frac{Pulse\ Width\ x\ 1000}{20}$$

(Persamaan 3.3)

Hasil Nilai yang diberikan pada CCR1 seperti pada bab 4. Untuk program dalam mengatur posisi ranjang lipat dapat dilihat pada Gambar 3.11.

```
  HAL_UART_Receive(&huart2, rBuf, sizeof(rBuf), 1000);
  for (int i = 0;i<12;i++){
    if(rBuf[i]==oke[i] || rBuf[i]==oke2[i]){
      xx=xx+1;
    }
    else if(rBuf[i]==naik[i] || rBuf[i]==naik2[i] || rBuf[i]==naik3[i]){
      zz=zz+1;
    }
    else if(rBuf[i]==turun[i] || rBuf[i]==turun2[i]){
    yy=yy+1;
    }
  }
if(xx>7){
  status = 1;
  buzz();
}

if(xawal >= 50 && xawal <= 80){
  if (zz>4 && status == 1){
    for(x=xawal;x>=xawal-15;x--){
      buzz();
      TIM2->CCR1=x; //0
      status = RESET;
      HAL_Delay(100);
    }
    xawal = x;
    HAL_Delay(100);
  }
  else if (yy>4 && status == 1){
    for(x=xawal;x<=xawal+15;x++){
      buzz();
      TIM2->CCR1=x; //0
      status = RESET;
      HAL_Delay(100);
    }
  xawal = x;
  HAL_Delay(100);
  }
}
```

**Gambar 3.11 Program Sistem Pengaturan Posisi Ranjang Lipat**

Pada program Gambar 3.11 STM32F4 megirimkan perintah pada modul VR3 untuk me *load* data rekognisi. Apabila pengguna mengatakan kata-kata dan kata tersebut dapat dikenali oleh modul VR3 maka modul VR3 mengirimkan data kata yang dikenali tersebut dalam bentuk data hexa. Kemudian pada STM32 data yang diterima tersebut dibandingkan dan apabila cocok maka akan disimpan pada salah satu variabel xx, yy atau zz.

Pada program ini, nilai awal pada PWM servo adalah 80 dengan posisi servo lurus (ranjang sepenuhnya dalam posisi 0º). Kemudian untuk posisi berdiri maksimal pada ranjang berada pada posisi 55º atau nilai servo PWM sebesar 50.

Cara kerja dari ranjang ini adalah membandingkan nilai PWM servo. Kemudian saat pengguna memberi perintah oke maka pengguna dapat memberikan perintah selanjutnya naik atau turun. Apabila pengguna hanya memberikan perintah naik atau turun tanpa perintah inisialisasi berupa kata oke, maka motor servo tidak akan bergerak.

Pada perancangan ini, ditambahkan proses untuk memudahkan pengguna dalam melakukan pembelajaran modul VR3 dimana modul VR3 ini merupakan modul pengenalan suara yang hanya dapat mengenali perintah suara dari 1 orang. Perancangan dari proses pembelajaran sebagai prosedur dari adanya pasien baru dijelaskan pada subbab berikut.

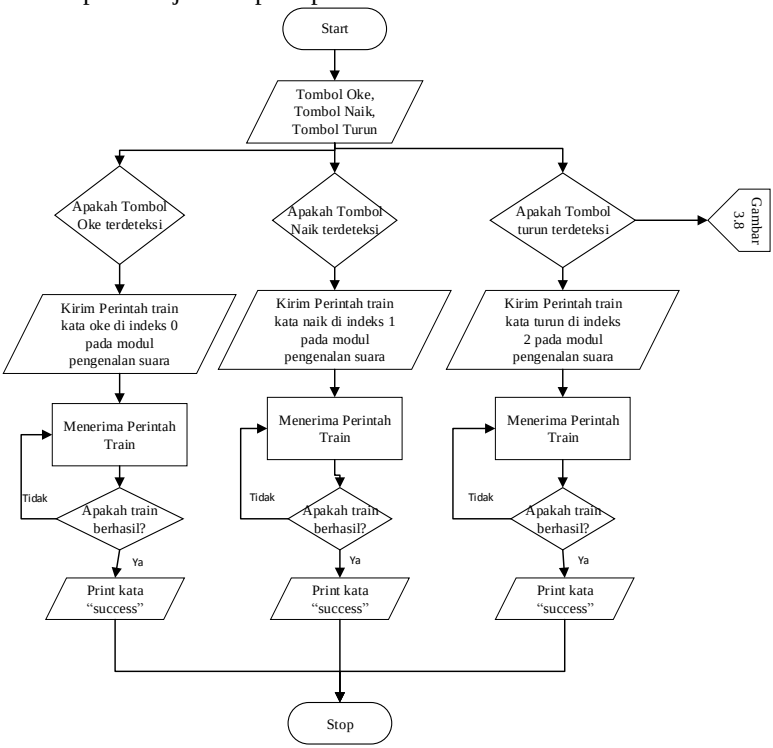
### 3.4.1 Prosedur Adanya Pasien Baru

Karena modul *voice recognition* V3 hanya dapat mengenali perintah suara dari 1 orang. Maka agar dapat digunakan oleh pasien yang lain dibutuhkan pembelajaran untuk pasien baru. Untuk perancangan proses pembelajaran modul VR3 pada pasien baru ditambahkan beberapa tombol dan *display* LCD16x2 agar mudah dioperasikan oleh operator untuk membantu pasien tersebut dalam melakukan suatu pembelajaran seperti pada **Error! Reference source not found.**.

Seperti petunjuk pada datasheet bahwa untuk melakukan pembelajaran modul VR3 selain dengan arduino adalah dengan mengirimkan kode hexa dengan format :

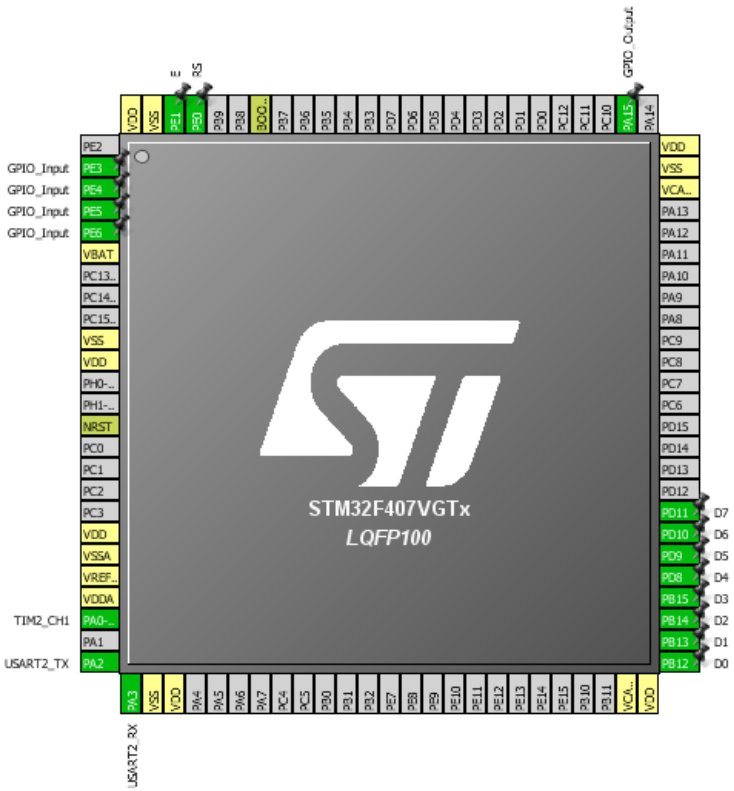
|AA|03+SIGLEN|21|Record|SIG|0A|
Dimana :
Record : nomer indeks suara disimpan
SIG : nama indeks
SIGLEN : panjang nama indeks

Sehingga pada program STM32 ditambahkan kode program untuk melakukan pembelajaran seperti pada Gambar 3.12.




**Gambar 3.12 *Flowchart* Proses Pembelajaran Pasien Baru**

Pada program Keil uVision 5, inisialisasi penggunaan tombol dan *display* LCD16x2 dilihat pada Gambar 3.13. Pada Gambar 3.13, tombol untuk pembelajaran kata naik yang dihubungkan dengan pin pada STM32F4 adalah pada pin PE6, sedangkan tombol untuk pembelajaran untuk kata naik dan turun dihubungkan dengan pin PE5 dan PE4. Untuk *display* LCD 16x2, pin RS dan E dihubungkan dengan pin PE0 dan PE1, untuk pin D0 hingga D3 dihubungkan dengan pin PB12 hingga PB15, sedangkan pin untuk D4 hingga D7 dihubungkan dengan pin PD8 hingga PD11.




**Gambar 3.13 Inisialisasi Pin Tombol dan *Display* Untuk Pembelajaran Pada STM32F4**

Pada program Keil uVision 5, dibuat fungsi untuk pembelajaran tiap kata yang diberikan pada indeks mulai 0 hingga 2 dimana kata oke diberikan ke indeks 0, kata naik diberikan ke indeks 1, dan kata turun diberikan ke indeks 2. Fungsi program untuk pembelajaran kata naik seperti pada Gambar 3.14. Untuk program lengkapnya seperti pada lampiran.

```
void trainoke()
]{
  pData[0]=0xAA;
  HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
  HAL_Delay(10);
  pData[0]=0x06;
  HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
  HAL_Delay(10);
  pData[0]=0x21;
  HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
  HAL_Delay(10);
  pData[0]=0x00;
  HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
  HAL_Delay(10);
  pData[0]=0x6F;
  HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
  HAL_Delay(10);
  pData[0]=0x6B;
  HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
  HAL_Delay(10);
  pData[0]=0x65;
  HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
  HAL_Delay(10);
  pData[0]=0x0A;
  HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
  HAL_Delay(1000);
}
```

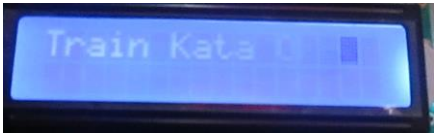
**Gambar 3.14 Program Fungsi untuk Pembelajaran Kata Oke**

Terdapat 4 buah tombol pada perancangan prosedur pembelajaran modul VR3 pada pasien, yaitu tombol untuk pembelajaran kata oke, naik dan turun serta tombol reset seperti pada Gambar 3.15.


**Gambar 3.15 Tombol yang Digunakan untuk Proses Pembelajaran Modul VR3**

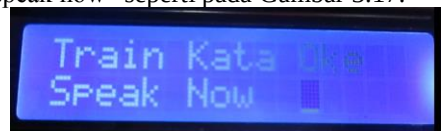
Ketika terdapat pasien baru, prosedur untuk pembelajaran perangkat keras ranjang lipat adalah sebagai berikut.

1. Tekan tombol berwarna putih untuk melakukan pembelajaran kata "oke" dan tunggu hingga sekitar 3 detik.
2. Apabila telah siap akan muncul indikasi pada *display* "train kata oke" seperti Gambar 3.16



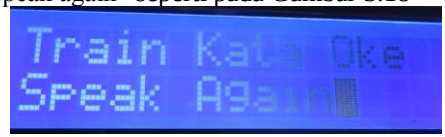

**Gambar 3.16 *Display* yang Menunjukkan Kata "Oke" Siap Dilatih oleh perangkat**

3. Pengguna dapat mengatakan kata oke saat display menunjukkan kalimat "speak now" seperti pada Gambar 3.17.




**Gambar 3.17 *Display* yang Menunjukkan Pasien untuk Mengatakan kata "Oke"**

4. Kemudian ulangi mengatakan kata "oke" saat display menunjukkan kalimat "speak again" seperti pada Gambar 3.18




**Gambar 3.18 *Display* yang Menunjukkan Pasien untuk Mengulangi Mengatakan kata "Oke"**

5. Ulangi langkah 3 dan 4 hingga display menunjukkan kalimat "success!!!" seperti pada Gambar 3.19


**Gambar 3.19 *Display* yang Menunjukkan Pelatihan Selesai Dilakukan**

Selanjutnya adalah melakukan pembelajaran kata “naik” dengan cara menekan tombol kuning seperti langkah nomer 1 hingga, begitu pula untuk pembelajaran kata “turun”. Setelah pembelajaran ketiga kata tersebut selesai dilakukan, maka perangkat ranjang lipat siap digunakan oleh pasien baru.

-----Halaman ini sengaja dikosongkan-----

# BAB IV
# PENGUJIAN PERANGKAT KERAS MINIATUR RANJANG LIPAT

Pada bab ini akan dibahas mengenai pengujian sistem dari purwarupa ranjang lipat berupa pengujian perangkat keras per bloknya yaitu dari *supply* tegangan, pengambilan data sinyal suara yang telah di *pembelajaran*, pengujian motor servo serta pengujian keseluruhan sistem.

## 4.1 Pengujian Modul *Voice Recognition V3*

Pada pengujian modul VR3 dilakukan kepada 5 orang koresponden wanita berumur sekitar 20 tahunan. Koresponden ini akan melakukan rekognisi kata "siap", "naik" dan "turun" pada modul VR3. Pengujian dilakukan dengan saat pengguna menggunakan ranjang lipat dimana saat pengguna menggunakan ranjang tersebut dapat bergerak atau tidak. Hasil yang didapat ditunjukkan pada Tabel 4.1.

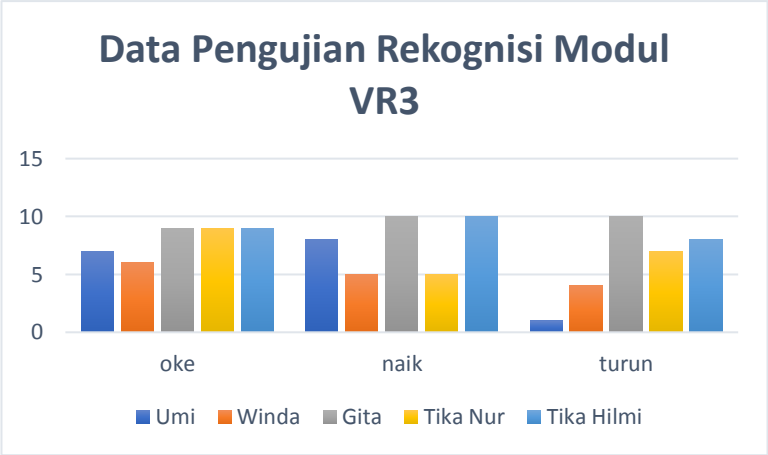
**Tabel 4.1 Hasil Pengujian Rekognisi Modul VR3**

| Nama | Kata Oke | | Kata Naik | | Kata Turun | |
|---|---|---|---|---|---|---|
| | Banyak Percobaan | Dikenali Sebagai | Banyak Percobaan | Dikenali Sebagai | Banyak Percobaan | Dikenali Sebagai |
| umi | 10x | - | 10x | - | 10x | Turun |
| | | Oke | | - | | - |
| | | - | | Naik | | - |
| | | Oke | | Naik | | - |
| | | Oke | | Naik | | - |
| | | - | | Naik | | - |
| | | Oke | | Naik | | - |
| | | Oke | | Naik | | - |
| | | Oke | | Naik | | - |
| | | Oke | | Naik | | - |
| winda | 10x | Oke | 10x | - | 10x | Turun |
| | | - | | - | | - |
| | | - | | - | | - |

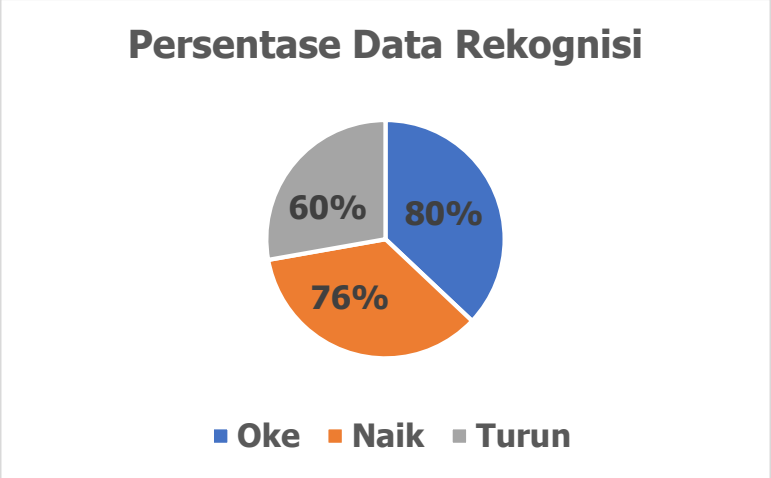
| Nama | Kata Oke | | Kata Naik | | Kata Turun | |
|---|---|---|---|---|---|---|
| | Banyak Percobaan | Dikenali Sebagai | Banyak Percobaan | Dikenali Sebagai | Banyak Percobaan | Dikenali Sebagai |
| | | - | | Naik | | - |
| | | Oke | | Naik | | - |
| | | Oke | | - | | Turun |
| | | - | | - | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | - |
| | | Oke | | Naik | | - |
| gita | 10x | - | 10x | Naik | 10x | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| tika | 10x | Oke | 10x | - | 10x | - |
| | | Oke | | - | | - |
| | | Oke | | - | | - |
| | | Oke | | Naik | | Turun |
| | | - | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | - | | Turun |
| | | Oke | | - | | Turun |

| Nama | Kata Oke | | Kata Naik | | Kata Turun | |
|---|---|---|---|---|---|---|
| | Banyak Percobaan | Dikenali Sebagai | Banyak Percobaan | Dikenali Sebagai | Banyak Percobaan | Dikenali Sebagai |
| tika h | 10x | - | 10x | Naik | 10x | - |
| | | Oke | | Naik | | - |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| | | Oke | | Naik | | Turun |
| Rata-rata per kata | 80 % | | 76 % | | 60% | |
| Rata – rata rekognisi keseluruhan | 72% | | | | | |

Data hasil pengujian rekognisi disajikan dalam diagram batang dan diagram lingkaran. Diagram batang pada Gambar 4.1 menunjukkan data korespondensi yang suaranya berhasil direkognisi. Sedangkan pada diagram lingkaran Gambar 4.2 ditunjukkan persentase keberhasilan tiap kata yang diucapkan oleh korespondensi. Data Selengkapnya dari korespondensi yang melakukan rekognisi dapat dilihat pada lampiran.



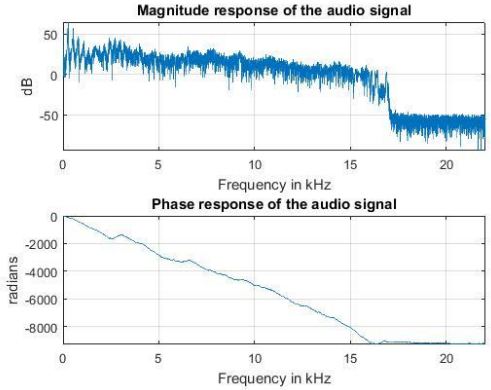

**Gambar 4.1 Diagram Batang Hasil Rekognisi Modul VR3**



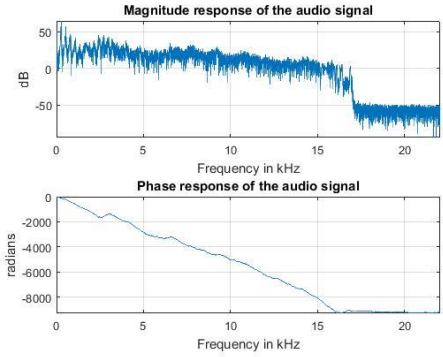

**Gambar 4.2 Diagram Lingkaran Tingkat Keberhasilan Rekognisi Modul VR3**

### 4.1.1 Data Respon Suara

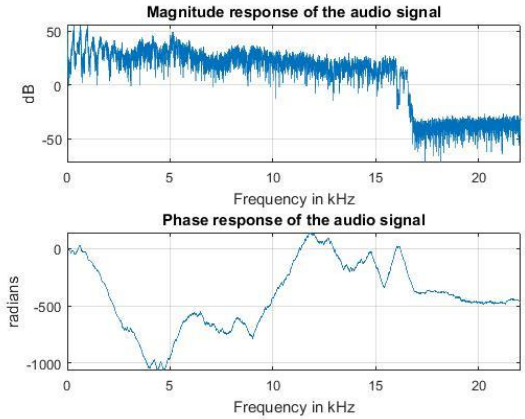
Pada pengujian tingkat keberhasilan rekognisi didapatkan rata-rata tingkat keberhasilan sebesar 72%. Dari data tersebut, dilakukan uji coba dengan merekam suara dari pengguna dan suara koresponden yang berhasil di rekognisi oleh modul VR3 dan suara koresponden yang tidak berhasil dikenali. Kemudian di tampilkan dalam grafik dengan aplikasi MATLAB dari respon magnitudo dan fasa dari sinyal suara yang telah di FFT. Hasil FFT sinyal suara dari kata naik ditampilkan pada grafik Gambar 4.3 hingga Gambar 4.5 dibawah.




**Gambar 4.3 Respon Suara yang Dijadikan Pembelajaran**




**Gambar 4.4 Suara Penguji yang Berhasil Terekognisi**




**Gambar 4.5 Respon Suara Penguji yang Tidak Berhasil Terekognisi**

Dari hasil FFT ini, didapatkan bahwa modul VR3 dapat mengenali kata dengan nada suara yang sama. Dimana frekuensi dan magnitudo dari FFT yang ditampilkan memiliki grafik yang sama.

## 4.2 Hasil Pengujian Pembelajaran Pasien Baru

Pengujian pembelajaran pasien baru dilakukan untuk menguji berapa kali maksimal pasien dapat mengulang kata yang sama dalam satu kali tombol ditekan. Hasil dari pengujian ditampilkan pada tabel. Dari hasil pengujian didapatkan bahwa pembelajaran modul VR dengan menggunakan STM32F4 untuk pembelajaran pasien baru tidak dapat hanya sekali dalam mengulang perkataan.

**Tabel 4.2 Pengujian Pembelajaran Pasien Baru**

| Nama | Banyaknya Pengulangan | | |
|---|---|---|---|
| | Oke | Naik | Turun |
| Umi | 2 | 3 | 2 |
| Winda | 3 | 2 | 5 |
| Gita | 4 | 5 | 5 |
| Tika | 6 | 6 | 6 |
| Tika H. | 5 | 5 | 3 |

## 4.3 Pengujian Motor Servo

Pengujian motor servo dilakukan untuk mendapatkan nilai sudut untuk penggerak ranjang lipat. Pengujian dilakukan dengan memberikan masukan berupa pulsa dengan periode 20 milidetik dan lebar pulsa yang diubah-ubah untuk mendapatkan sudut yang tepat seperti pada Gambar 2.20. Cara pengujian motor servo adalah dengan memberikan masukan PWM (*Pulse Width Modulation*) dari STM32 ke pin masukan pulsa motor servo.

Sudut perubahan derajat pada motor servo yang diuji hanyalah dari sudut $0^o$ dan $45^o$ diukur dari penempatan motor servo pada purwarupa ranjang lipat. Untuk ukuran sudut yang sebenarnya pada motor servo adalah dari 0,3 milidetik yang merupakan sudut $0^o$ dan 3 milidetik adalah pada sudut $180^o$. Hasil pengujian sudut motor servo seperti pada Tabel 4.3.

**Tabel 4.3 Hasil Pengujian Motor Servo**

| Sudut (derajat) | Lebar Pulsa (milidetik) |
|---|---|
| 0 | 1,6 |
| 45 | 1,1 |

Dari tabel diatas diketahui pada sudut $0^o$ lebar pulsa PWM adalah 1,6 milidetik, sedangkan saat $45^o$ lebar pulsa adalah sebesar 1,1 milidetik. Sehingga dapat disimpulkan bahwa peletakan motor servo dengan sudut yang sebenarnya adalah dari $180^o$ ke $0^o$. Karena semakin besar sudut motor servo, lebar pulsa semakin kecil.

Untuk dapat mengetahui waktu yang dibutuhkan motor servo bergerak dari posisi $0^o$ menuju ke $30^0$ dan dari posisi $30^o$ ke sekitar $60^0$ dihitung mulai kata naik terdeteksi didapatkan hasil seperti Tabel 4.4.

**Tabel 4.4 Pengukuran Waktu untuk Pergerakan Motor Servo**

| Sudut (derajat) | Waktu yang Dibutuhkan (detik) |
|---|---|
| $0^0$ ke $30^0$ | 5 |
| $30^0$ ke $60^0$ | 4 |

-----Halaman ini sengaja dikosongkan-----

# BAB V
# PENUTUP

## 5.1 Kesimpulan

Dari perancangan purwarupa ranjang lipat sistem pengenalan suara sebagai pengatur posisi ranjang lipat menggunakan mikrokontroler STM32F407VG dan modul *voice recognition* v3 (VR3) didapatkan hasil bahwa modul VR3 memiliki kemampuan dalam merekognisi perintah suara sebesar 72%. Dalam proses pembelajaran modul VR3 didapatkan percobaan dalam satu kata sekali pembelajaran adalah sebanyak maksimal 10x percobaan.

Untuk pergerakan dari aktuator berupa motor servo didapatkan bahwa besar beban mempengaruhi sudut motor servo. Sehingga pengujian harus dilakukan ketika diberikan beban agar posisi motor servo sesuai yang diinginkan. Dari perancangan pergerakan posisi ranjang dari $0^0$ dimana posisi ranjang adalah horizontal lurus menuju posisi $30^o$ membutuhkan waktu selama 5 detik.

## 5.2 Saran

Saran untuk penelitian selanjutnya, sebaiknya diberi filter analog yang sesuai agar sistem pengenalan suara lebih akurat. Sehingga diperlukan penelitian lebih lanjut untuk pengenalan suara dengan modul VR3.

-----Halaman ini sengaja dikosongkan-----

# DAFTAR PUSTAKA


[1] S. K. Gaikwad, B. W. Gawali, dan P. Yannawar, "A Review on Speech Recognition Technique," *Int. J. Comput. Appl.*, vol. 10 No. 3, hlm. Novemmber 2010.

[2] A. A. Linsie, "An Automated Bed Positioning and Windows System Using MEMS Accelerometer," *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.*, vol. 3, no. 11, Nov 2014.

[3] S. K. Das, V. Rathod, dan A. Yadav, "Voice Recognition Based Automation System for Medical Applications and For Physically Challenged Patients," *Int. Res. J. Eng. Technol. IRJET*, vol. 4, no. 03, Mar 2017.

[4] "Speech production, Lab 9a - speech processing (part 1), By OpenStax (Page 1/5)," *QuizOver.com*. [Daring]. Tersedia pada: https://www.quizover.com/course/section/speech-production-lab-9a-speech-processing-part-1-by-openstax. [Diakses: 01-Mar-2018].

[5] "Sampling Frequency and Bit Resolution for Speech Signal Processing (Theory) : Speech Signal Processing Laboratory : Electronics & Communications : IIT GUWAHATI Virtual Lab." [Daring]. Tersedia pada: http://iitg.vlab.co.in/?sub=59&brch=164&sim=474&cnt=1. [Diakses: 14-Feb-2018].

[6] Suwadi, *Pengolahan Sinyal Digital*. Institut Teknologi Sepuluh Nopember.

[7] "Lecture 7 - The Discrete Fourier Transform." [Daring]. Tersedia pada: http://webcache.googleusercontent.com/search?q=cache:-gDufBXwYn0J:www.robots.ox.ac.uk/~sjrob/Teaching/SP/l7.pdf+&cd=1&hl=en&ct=clnk&gl=id. [Diakses: 02-Apr-2018].

[8] H. Zumbahlen, *Basic linear design*. Norwood, Mass.: Analog Devices, 2007.

[9] "elchap20.pdf." .

[10] P. Sannuti, "332:224 Principles of Electrical Engineering II Laboratory." The State University of new jersey RUTGERS School of Engineering Department of Electrical and Computer Engineering, 16-Des-2005.

[11] "Filter." [Daring]. Tersedia pada: https://www.sfu.ca/sonic-studio/handbook/Filter.html. [Diakses: 21-Mar-2018].

[12] "Dictionary of Electronic and Engineering Terms 'B.'" [Daring]. Tersedia pada: http://www.interfacebus.com/Glossary-of-Terms_B.html. [Diakses: 21-Mar-2018].
[13] "TYPES OF ACTIVE FILTERS." [Daring]. Tersedia pada: http://webcache.googleusercontent.com/search?q=cache:ZVU9qJEkwZYJ:www.idc-online.com/technical_references/pdfs/electronic_engineering/Types_of_Active_Filters.pdf+&cd=1&hl=id&ct=clnk&gl=id. [Diakses: 02-Apr-2018].
[14] "Classic Filters There are 4 classic analogue filter types: Butterworth, Chebyshev, Elliptic and Bessel. There is no ideal filter." [Daring]. Tersedia pada: http://webcache.googleusercontent.com/search?q=cache:ueYvMeM_LlMJ:electronics-2.weebly.com/uploads/1/3/0/5/13056901/classicfilters.pdf+&cd=1&hl=id&ct=clnk&gl=id. [Diakses: 02-Apr-2018].
[15] "STRUKTUR FILTER DIGITAL I. Pendahuluan Struktur filter digital ditentukan dari persamaan beda atau fungsi sistem yang disebut Di." [Daring]. Tersedia pada: http://webcache.googleusercontent.com/search?q=cache:mP7MVmskU18J:staffnew.uny.ac.id/upload/197912142010122002/pendidikan/8.%2520STRUKTUR%2520FILTER%2520DIGITAL.pdf+&cd=1&hl=id&ct=clnk&gl=id. [Diakses: 02-Apr-2018].
[16] "III. TRANSFORMASI Z 3.1. Pengertian Transformasi Z memainkan peran yang sama dalam analisis sinyal waktu diskret dan sistem LTI." [Daring]. Tersedia pada: http://webcache.googleusercontent.com/search?q=cache:iMnGCGW9qF8J:staffnew.uny.ac.id/upload/197912142010122002/pendidikan/5.%2520.Transformasi%2520Z.pdf+&cd=14&hl=id&ct=clnk&gl=id. [Diakses: 02-Apr-2018].
[17] Elechouse, "Voice Recognition Module V3 'Speak to Control (Arduino compatible).'" 09-Mei-2014.
[18] AN4841, "Digital signal processing for STM32 microcontrollers using CMSIS." STMicroelectronics.
[19] "stm32f4 discovery pinout," *Virtual Pinball Universe*. [Daring]. Tersedia pada: http://vpuniverse.com/forums/gallery/image/572-stm32f4-discovery-pinout/. [Diakses: 25-Mei-2018].

[20] R. Am, Kemalasari, B. Sumantri, dan A. Wijayanto, "PENGATURAN POSISI MOTOR SERVO DC DENGAN METODE FUZZY LOGIC," Mei 2018.
[21] "MG995 MG995 High Speed Metal Gear Dual Ball Bearing Servo." .
[22] "Voice Recognition Module V3." [Daring]. Tersedia pada: https://webcache.googleusercontent.com/search?q=cache:qeR57hNb2gAJ:https://www.elechouse.com/elechouse/images/product/VR3/VR3_manual.pdf+&cd=1&hl=id&ct=clnk&gl=id. [Diakses: 01-Jun-2018].

-----Halaman ini sengaja dikosongkan-----

# LAMPIRAN A

## A.1. Program Keseluruhan

```
/**
******************************************************************
  * @file           : main.c
  * @brief          : Main program body
******************************************************************
  ** This notice applies to any and all portions of this file
  * that are not between comment pairs USER CODE BEGIN and
  * USER CODE END. Other portions of this file, whether
  * inserted by the user or by software development tools
  * are owned by their respective copyright owners.
  *
  * COPYRIGHT(c) 2018 STMicroelectronics
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *    1. Redistributions of source code must retain the above copyright notice,
  *       this list of conditions and the following disclaimer.
  *    2. Redistributions in binary form must reproduce the above copyright notice,
  *          this list of conditions and the following disclaimer in the documentation
  *       and/or other materials provided with the distribution.
  *    3. Neither the name of STMicroelectronics nor the names of its contributors
  *          may be used to endorse or promote products derived from this software
  *       without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
```

```
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************
 */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f4xx_hal.h"

/* USER CODE BEGIN Includes */
#include "STM_MY_LCD16X2.h"
#include <string.h>
/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim2;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
```

```
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM2_Init(void);
void HAL_TIM_MspPostInit(TIM_HandleTypeDef *htim);


/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/
void trainoke();
void trainnaik();
void trainturun();
void load();
void buzz();

#ifdef __GNUC__
	#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
	#define PUTCHAR_PROTOTYPE int fputc( int ch, FILE *f)
#endif

PUTCHAR_PROTOTYPE
{
 HAL_UART_Transmit(&huart2, (uint8_t *)&ch,1,0xFFFF);
	return ch;
}
/* USER CODE END PFP */

/* USER CODE BEGIN 0 */
uint8_t pData[1];
uint8_t rBuf[20];
uint8_t oke[20];
uint8_t naik[20];
uint8_t naik3[20];
uint8_t turun[20];
uint8_t oke2[20];
uint8_t naik2[20];
uint8_t turun2[20];
uint8_t speaknow[20];
```

```
uint8_t speakagain[20];
uint8_t success[20];

int zz;
int yy;
int xx;
int ss1;
int ss2;
int ss3;
int status;
int push1;
int push2;
int push3;
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  *
  * @retval None
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

            GPIO_PinState train;
            GPIO_PinState train2;
            GPIO_PinState train3;
            oke[0]=0xAA;
            oke[1]=0x0A;
            oke[2]=0x0D;
            oke[3]=0x00;
            oke[4]=0xFF;
            oke[5]=0x00;
            oke[6]=0x00;
            oke[7]=0x03;
            oke[8]=0x6F;
            oke[9]=0x6B;
            oke[10]=0x65;
            oke[11]=0x0A;
```

```
        //data naik
naik[0]=0xAA;
naik[1]=0x0B;
naik[2]=0x0D;
naik[3]=0x00;
naik[4]=0xFF;
naik[5]=0x01;
naik[6]=0x01;
naik[7]=0x04;
naik[8]=0x6E;
naik[9]=0x61;
naik[10]=0x69;
naik[11]=0x6B;
naik[12]=0x0A;

//data turun
turun[0] = 0xAA;
turun[1] = 0x0C;
turun[2] = 0x0D;
turun[3] = 0x00;
turun[4] = 0xFF;
turun[5] = 0x02;
turun[6] = 0x02;
turun[7] = 0x05;
turun[8] = 0x74;
turun[9] = 0x75;
turun[10] = 0x72;
turun[11] = 0x75;
turun[12] = 0x6E;
turun[13] = 0x0A;

//data oke2
oke2[0]=0xAA;
oke2[1]=0x0C;
oke2[2]=0x0D;
oke2[3]=0x00;
oke2[4]=0xFF;
oke2[5]=0x03;
```

```
oke2[6]=0x03;
oke2[7]=0x05;
oke2[8]=0x6f;
oke2[9]=0x6b;
oke2[10]=0x65;
oke2[11]=0x32;
oke2[12]=0x0A;

//data naik2
naik2[0] = 0xAA;
naik2[1] = 0x0C;
naik2[2] = 0x0D;
naik2[3] = 0x00;
naik2[4] = 0xFF;
naik2[5] = 0x04;
naik2[6] = 0x04;
naik2[7] = 0x05;
naik2[8] = 0x6E;
naik2[9] = 0x61;
naik2[10] = 0x69;
naik2[11] = 0x6B;
naik2[12] = 0x32;
naik2[13] = 0x0A;

naik3[0] = 0xAA;
naik3[1] = 0x0C;
naik3[2] = 0x0D;
naik3[3] = 0x00;
naik3[4] = 0xFF;
naik3[5] = 0x06;
naik3[6] = 0x06;
naik3[7] = 0x05;
naik3[8] = 0x6E;
naik3[9] = 0x61;
naik3[10] = 0x69;
naik3[11] = 0x6B;
naik3[12] = 0x33;
naik3[13] = 0x0A;
//data turun2
```

```
	turun2[0] = 0xAA;
	turun2[1] = 0x0D;
	turun2[2] = 0x0D;
	turun2[3] = 0x00;
	turun2[4] = 0xFF;
	turun2[5] = 0x05;
	turun2[6] = 0x05;
	turun2[7] = 0x06;
	turun2[8] = 0x74;
	turun2[9] = 0x75;
	turun2[10] = 0x72;
	turun2[11] = 0x75;
	turun2[12] = 0x6E;
	turun2[13] = 0x32;
	turun2[14] = 0x0A;

	speaknow[1] = 0x0D;

	speakagain[1] = 0x0F;

	success[1] = 0x65;

		rBuf[0]=0;
		rBuf[1]=0;
		rBuf[2]=0;
		rBuf[3]=0;
		rBuf[4]=0;
		rBuf[5]=0;
		rBuf[6]=0;
		rBuf[7]=0;
		rBuf[8]=0;
		rBuf[9]=0;
		rBuf[10]=0;
		rBuf[11]=0;
		rBuf[12]=0;
		rBuf[13]=0;
		rBuf[14]=0;
/* USER CODE END 1 */
```

```
  /* MCU Configuration--------------------------------------------------------
*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick.
*/
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
 //trainoke();

LCD1602_Begin8BIT(RS_GPIO_Port,RS_Pin,E_Pin,D0_GPIO_Port,D
0_Pin,D1_Pin,D2_Pin,D3_Pin,D4_GPIO_Port,D4_Pin,D5_Pin,D6_Pin,
D7_Pin);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  { LCD1602_clear();
                        rBuf[0]=0;
                        rBuf[1]=0;
                        rBuf[2]=0;
                        rBuf[3]=0;
```

```
                rBuf[4]=0;
                rBuf[5]=0;
                rBuf[6]=0;
                rBuf[7]=0;
                rBuf[8]=0;
                rBuf[9]=0;
                rBuf[10]=0;
                rBuf[11]=0;
                rBuf[12]=0;
                rBuf[13]=0;
                rBuf[14]=0;
                zz=0;
                yy=0;
                ss1=0;
                ss2=0;
                ss3 = 0;
                //HAL_UART_Receive(&huart2,  rBuf,  sizeof(rBuf),
1000);


                train = HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_6);
                train2 = HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_5);
                train3 = HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_4);
                push1=train;
                push2=train2;
                push3=train3;
                if (push1 == 0){
                        LCD1602_print("Train Kata Oke");
                trainoke();
        HAL_Delay(100);
                while(1){
                HAL_UART_Receive(&huart2,  rBuf,  sizeof(rBuf),
1000);
                for (int i = 1;i<3;i++) {
                if (rBuf[i] == speaknow[i]){
                ss1=ss1+1;
                }
                else if (rBuf[i] == speakagain[i]){
                ss2=ss2+1;
```

```
		}
		else if (rBuf[i] == success[i]){
		ss3=ss3+1;
		}
		else if (rBuf[i] == success[i]){
			ss3 = ss3+1;}
		}

		if (ss1==1){
			LCD1602_2ndLine();
			LCD1602_print("Speak Now  ");
			ss1=0;
		}
		else if (ss2==1){
			LCD1602_2ndLine();
			LCD1602_print("Speak Again");
			ss2=0;
		}
		else if (ss3==1){
			LCD1602_2ndLine();
			LCD1602_print("Success!!!!!");
			ss3=0;
			HAL_Delay(1000);
			LCD1602_clear();
			break;
		}
		}
		}

		else if (push2 == 0){
			LCD1602_print("Train Kata Naik");
		trainnaik();
	HAL_Delay(100);
		while(1){
		HAL_UART_Receive(&huart2,  rBuf,  sizeof(rBuf),
1000);
		for (int i = 0;i<3;i++) {
		if (rBuf[i] == speaknow[i]){
		ss1=ss1+1;
```

```
                }
                else if (rBuf[i] == speakagain[i]){
                ss2=ss2+1;
                }}
                if (ss1==1){
                        LCD1602_2ndLine();
                        LCD1602_print("Speak Now  ");
                        ss1=0;
                }
                else if (ss2==1){
                        LCD1602_2ndLine();
                        LCD1602_print("Speak Again");
                        ss2=0;
                }
                else if (ss3==1){
                        LCD1602_2ndLine();
                        LCD1602_print("Success!!!!!");
                        ss3=0;
                        HAL_Delay(1000);
                        LCD1602_clear();
                        break;
                }}
                }

                else if (push3 == 0){
                LCD1602_print("Train Kata Turun");
                trainturun();
        HAL_Delay(100);
                while(1){
                HAL_UART_Receive(&huart2,  rBuf,  sizeof(rBuf),
1000);
                for (int i = 1;i<2;i++) {
                if (rBuf[i] == speaknow[i]){
                ss1=ss1+1;
                }
                else if (rBuf[i] == speakagain[i]){
                ss2=ss2+1;
                }}
                if (ss1==1){
```

```
                LCD1602_2ndLine();
                LCD1602_print("Speak Now  ");
                ss1=0;
        }
        else if (ss2==1){
                LCD1602_2ndLine();
                LCD1602_print("Speak Again");
                ss2=0;
        }
        else if (ss3==1){
                LCD1602_2ndLine();
                LCD1602_print("Success!!!!!");
                ss3=0;
                HAL_Delay(1000);
                LCD1602_clear();
                break;
        }}
        }
                load();
        buzz();
          LCD1602_clear();
          LCD1602_print("Siap");
                                        LCD1602_2ndLine();
                LCD1602_print("Dipakai");

HAL_TIM_PWM_Start(&htim2,TIM_CHANNEL_1);
                TIM2->CCR1 = 80;
                int xawal = 80;
                int x;
                status = 0;

                while(1){
        train = HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_6);
        train2 = HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_5);
        train3 = HAL_GPIO_ReadPin(GPIOE,GPIO_PIN_4);
        push1=train;
        push2=train2;
        push3=train3;
```

```
                                                    if (push1 == 0 ||
push2 == 0 || push3 == 0){
                                                    break;}
                    zz=0;
                    yy=0;
                    xx = 0;
          HAL_UART_Receive(&huart2, rBuf, sizeof(rBuf), 1000);
          for (int i = 0;i<12;i++){
                    if(rBuf[i]==oke[i] || rBuf[i]==oke2[i]){
                              xx=xx+1;
                    }
                    else     if(rBuf[i]==naik[i]     ||     rBuf[i]==naik2[i]     ||
rBuf[i]==naik3[i]){
                              zz=zz+1;
                                        buzz();
                    }
                    else if(rBuf[i]==turun[i] || rBuf[i]==turun2[i]){
                    yy=yy+1;
                                  buzz();
                    }
          }

if(xx>7){
          status = 1;
                              buzz();
}

if(xawal >= 40 && xawal <= 80){
          if (zz>4 && status == 1){
                    for(x=xawal;x>=xawal-20;x--){

                              TIM2->CCR1=x; //0
                              status = RESET;
                              HAL_Delay(100);
                    }
                    xawal = x;
                    HAL_Delay(100);
          }}
          if(xawal < 80){
```

```
                if (yy>4 && status == 1){
                for(x=xawal;x<=xawal+20;x++){

                        TIM2->CCR1=x; //0
                        status = RESET;
                        HAL_Delay(100);
                }
        xawal = x;
        HAL_Delay(100);
        }}
                        }

        //HAL_UART_Receive(&huart2, rBuf, sizeof(rBuf), 1000);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

  }
  /* USER CODE END 3 */

}
void buzz()
{
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_15,GPIO_PIN_SET
);
        HAL_Delay(100);
        HAL_GPIO_WritePin(GPIOA,GPIO_PIN_15,GPIO_PIN_RE
SET);
}

void trainoke()
{
        pData[0]=0xAA;
        HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
        HAL_Delay(10);
        pData[0]=0x06;
        HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
        HAL_Delay(10);
        pData[0]=0x21;
```

```
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x00;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x6F;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x6B;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x65;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x0A;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(1000);
}

void trainnaik()
{
		pData[0]=0xAA;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x07;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x21;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x01;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x6E;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x61;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
```

```
	pData[0]=0x69;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x6B;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x0A;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(1000);
}

void trainturun()
{
	pData[0]=0xAA;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x08;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x21;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x02;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x74;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x75;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x72;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x75;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x6E;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
```

```
	HAL_Delay(10);
	pData[0]=0x0A;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(1000);
}

void load()
{
	uint8_t pData[1];
	pData[0]=0xAA;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x02;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x31;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x0A;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(1000);
	pData[0]=0xAA;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x03;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x30;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x00;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(10);
	pData[0]=0x0A;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
	HAL_Delay(1000);

	pData[0]=0xAA;
	HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
```

```
HAL_Delay(10);
pData[0]=0x03;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x30;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x01;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x0A;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(1000);

        //load index 2
pData[0]=0xAA;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x03;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x30;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x02; //yg diganti
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x0A;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(1000);
//=========================================
//load index 3
pData[0]=0xAA;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x03;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x30;
```

```
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x03; //yg diganti
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x0A;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(1000);
//=========================================
//load index 4
pData[0]=0xAA;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x03;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x30;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x04; //yg diganti
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x0A;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(1000);
//=========================================
//load index 5
pData[0]=0xAA;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x03;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x30;
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
pData[0]=0x05; //yg diganti
HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
HAL_Delay(10);
```

```
		pData[0]=0x0A;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(1000);
			//load index 6
		pData[0]=0xAA;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x03;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x30;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x06; //yg diganti
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(10);
		pData[0]=0x0A;
		HAL_UART_Transmit(&huart2,pData, sizeof(pData),10);
		HAL_Delay(1000);
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
  __HAL_RCC_PWR_CLK_ENABLE();


__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_
VOLTAGE_SCALE1);

    /**Initializes the CPU, AHB and APB busses clocks
```

```
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 168;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  /**Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType                                    =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource                                 =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if                    (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_5) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  /**Configure the Systick interrupt time
  */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

  /**Configure the Systick
  */
```

```
HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK
);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/* TIM2 init function */
static void MX_TIM2_Init(void)
{

  TIM_MasterConfigTypeDef sMasterConfig;
  TIM_OC_InitTypeDef sConfigOC;

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 1680;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 1000;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode                                    =
TIM_MASTERSLAVEMODE_DISABLE;
  if                  (HAL_TIMEx_MasterConfigSynchronization(&htim2,
&sMasterConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
```

```
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC,
TIM_CHANNEL_1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  HAL_TIM_MspPostInit(&htim2);

}

/* USART2 init function */
static void MX_USART2_UART_Init(void)
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
static void MX_GPIO_Init(void)
{
```

```
  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB,          D0_Pin|D1_Pin|D2_Pin|D3_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD,          D4_Pin|D5_Pin|D6_Pin|D7_Pin,
GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_15, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOE, RS_Pin|E_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pins : PE3 PE4 PE5 PE6 */
  GPIO_InitStruct.Pin                                          =
GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

  /*Configure GPIO pins : D0_Pin D1_Pin D2_Pin D3_Pin */
  GPIO_InitStruct.Pin = D0_Pin|D1_Pin|D2_Pin|D3_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pins : D4_Pin D5_Pin D6_Pin D7_Pin */
  GPIO_InitStruct.Pin = D4_Pin|D5_Pin|D6_Pin|D7_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
```

```
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pin : PA15 */
  GPIO_InitStruct.Pin = GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : RS_Pin E_Pin */
  GPIO_InitStruct.Pin = RS_Pin|E_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  file: The file name as string.
  * @param  line: The line in file as a number.
  * @retval None
  */
void _Error_Handler(char *file, int line)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  while(1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
```

```
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
*/
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/**
  * @}
  */

/**
  * @}
  */

/************************ (C) COPYRIGHT STMicroelectronics
*****END OF FILE****/
```

# LAMPIRAN B

## B.1 Dokumentasi Percobaan Rekognisi Modul VR3

-----Halaman ini sengaja dikosongkan-----

# LAMPIRAN C

## C.1 Prosedur Pemakaian Ranjang Lipat

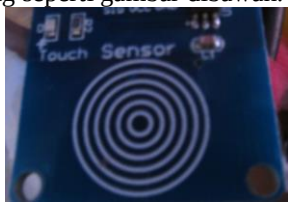
Setelah power dinyalakan, tunggu sekitar 9 detik untuk dapat digunaka yang ditandai dengan bunyi beep dan *display* menampilkan kata-kata siap dipakai seperti pada gambar dibawah.


Untuk menggerakkan ranjang naik atau turun dengan cara mengatakan kata “oke” terlebih dahulu. Apabila kata “oke” terdeteksi maka terdengar bunyi beep.

Setelah itu, baru dapat mengatakan kata “naik” untuk menaikkan ranjang, atau kata “turun” untuk menurunkan ranjang. Setiap kata yang diucapkan terdeteksi maka indikator beep akan berbunyi.

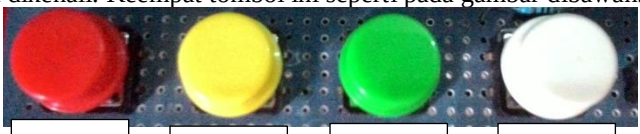
Ket : Setiap perintah untuk menggerakkan ranjang harus didahului inisialisasi dengan kata “oke”.

Apabila terdapat kesalahan terdapat tombol *emergency* untuk pasien yang terdapat pada ranjang seperti gambar dibawah.




### C.1.1 Prosedur Pembelajaran untuk Pasien Baru

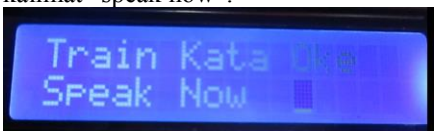
Terdapat 4 tombol yang dapat digunakan untuk melakukan pembelajaran pada ranjang lipat agar perintah yang diberikan oleh pasien dapat dikenali. Keempat tombol ini seperti pada gambar dibawah.



Reset | Turun | Naik | Oke

1. Tekan tombol berwarna putih untuk melakukan pembelajaran kata “oke” dan tunggu hingga sekitar 3 detik.
2. Apabila telah siap akan muncul indikasi pada *display* “train kata oke”


3. Pengguna dapat mengatakan kata oke saat display menunjukkan kalimat “speak now”.




4. Kemudian ulangi mengatakan kata “oke” saat display menunjukkan kalimat “speak again”


5. Ulangi langkah 3 dan 4 hingga display menunjukkan kalimat “success!!!”

# LAMPIRAN D

## D.1 Datasheet Modul VR 3

# Voice Recognition Module V3

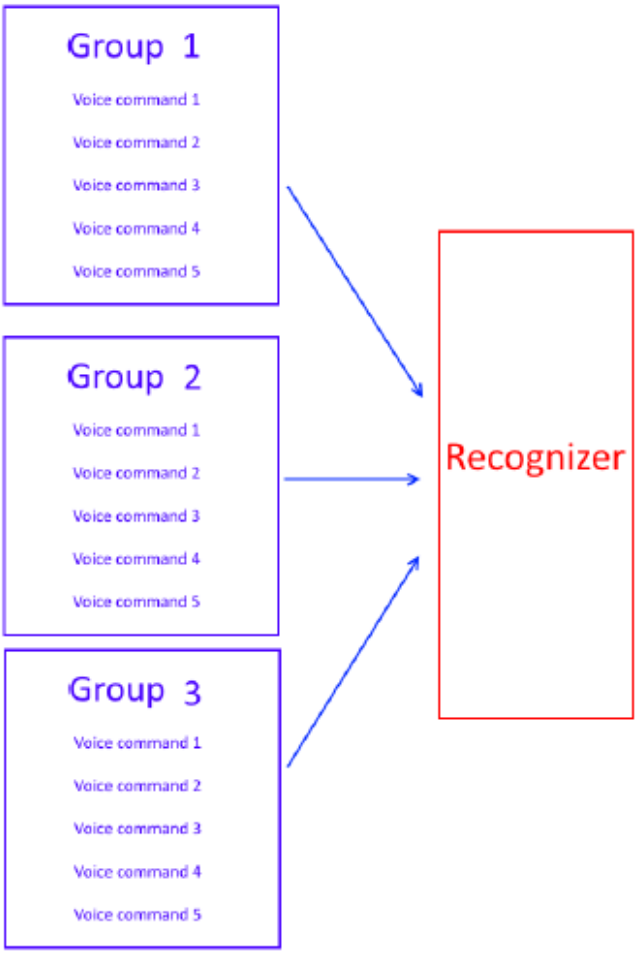
*Speak to Control (Arduino compatible)*


## Overview

ELECHOUSE Voice Recognition Module is a compact and easy-control speaking recognition board.

This product is a speaker-dependent voice recognition module. It supports up to 80 voice commands in all. Max 7 voice commands could work at the same time. Any sound could be trained as command. Users need to train the module first before let it recognizing any voice command.

This board has 2 controlling ways: Serial Port (full function), General Input Pins (part of function). General Output Pins on the board could generate several kinds of waves while corresponding voice command was recognized.




On V3, voice commands are stored in one large group like a library. Any 7 voice commands in the library could be imported into recognizer. It means 7 commands are effective at the same time.

# Parameter

- Voltage: 4.5-5.5V
- Current: <40mA
- Digital Interface: 5V TTL level for UART interface and GPIO
- Analog Interface: 3.5mm mono-channel microphone connector + microphone pin interface
- Size: 31mm x 50mm
- Recognition accuracy: 99% (under ideal environment)

# Feature

- Support maximum 80 voice commands, with each voice 1500ms (one or two words speaking)
- Maximum 7 voice commands effective at same time
- Arduino library is supplied
- Easy Control: UART/GPIO
- User-control General Pin Output

# Terminology

- VR3 -- Voice Recognition Module V3
- Recognizer -- a container where acting voice commands (max 7) were loaded. It is core part of voice recognition module. For example, it works like "playing balls". You have 80 players in your team. But you could not let them all play on the court together. The rule only allows 7 players playing on the court. Here the Recognizer is the list which contains names of players working on the court.
- Recognizer index -- max 7 voice commands could be supported in the recognizer. The recognizer has 7 regions for each voice command. One index corresponds to one region: 0~6
- Train -- the process of recording your voice commands
- Load -- copy trained voice to recognizer

# Instruction

Here we will introduce the Arduino Library and VR3 Protocol

## For Arduino

### Prepare

- Voice Recognition V3 module with microphone
- Arduino board (UNO recommended)
- Arduino Sensor Shield V07 (optional)
- Arduino IDE
- Voice Recognition V3 library (Download zip file)

### Hardware and Software Preparation

1. Connect your Voice Recognition V3 Module with Arduino, By Default:

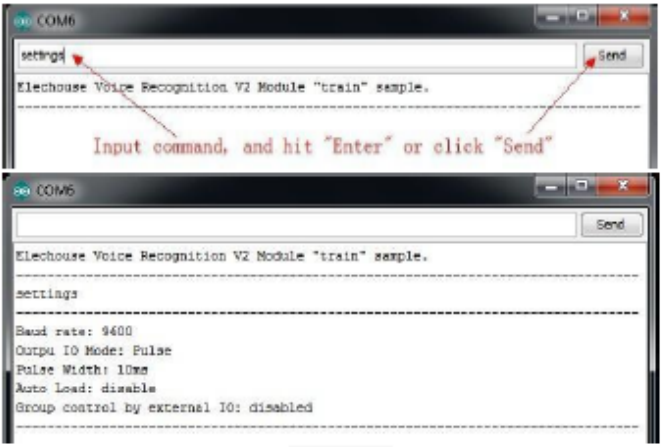
| Arduino | | VR Module |
|---|---|---|
| 5V | ----> | 5V |
| 2 | ----> | TX |
| 3 | ----> | RX |
| GND | ----> | GND |

2. Download VoiceRecognitionV3 library. (download zip file or use `git clone https://github.com/elechouse/VoiceRecognitionV3.git` command)
3. If using zip file, extrsct **VoiceRecognitionV3.zip** to `Arduino Sketch\libraries` folder, or if you use `git clone` command copy **VoiceRecognitionV3** to `Arduino Sketch\libraries`.

### Train

1. Open **vr_sample_train** (File -> Examples -> VoiceRecognitionV3 -> vr_sample_train)
2. Choose right Arduino board(Tool -> Board, UNO recommended), Choose right serial port.
3. Click **Upload** button, wait until Arduino is uploaded.
4. Open **Serial Monitor**. Set baud rate 115200, set send with **Newline** or **Both NL & CR**.


5. Send command `settings`(case insensitive) to check Voice Recognition Module settings. Input `settings`, and hit `Enter` to send.

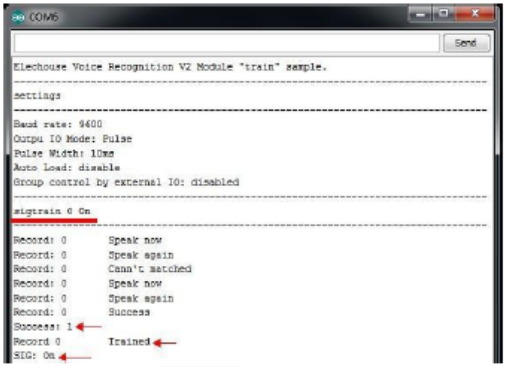



6. Train Voice Recognition Module. Send `sigtrain 0 On` command to train record 0 with signature "On". When Serial Monitor prints "Speak now", you need speak your voice(can be any word, meaningful word recommended, may be 'On' here), and when Serial Monitor prints "Speak again", you need repeat your voice again. If these two voice are matched, Serial Monitor prints "Success", and "record 0" is trained, or if are not matched, repeat speaking until success. What is a signature? Signature is a piece of text description for the voice command. For example, if your 7 voice command are "1, 2, 3, 4, 5, 6, 7", you could train in the following way:

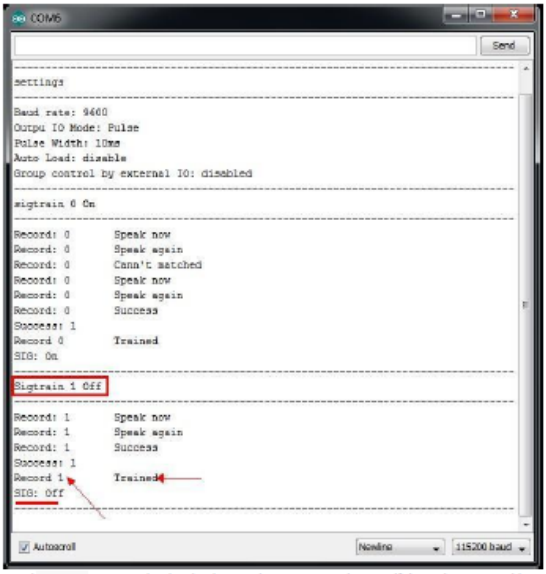
```
sigtrain 0 one
sigtrain 1 two
sigtrain 2 three
sigtrain 3 four
sigtrain 4 five
sigtrain 5 six
sigtrain 6 seven
```

The signature could be displayed if its command was called.

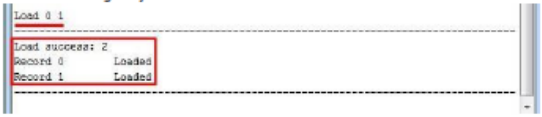
When training, the two led on the Voice Recognition Module can indicate your training process. After sending the training command, the SYS_LED (yellow) is blinking fast which remind you to get ready. Speak your voice command as soon as the STATUS_LED (red) light lights on. The recording process ends once when the STATUS_LED (red) lights off. Then the SYS_LED is blinking again, get ready for next recording process. When the training process ends successful, SYS_LED and STATUS_LED blink together. If the training fails, SYS_LED and STATUS_LED blink together, but quickly.



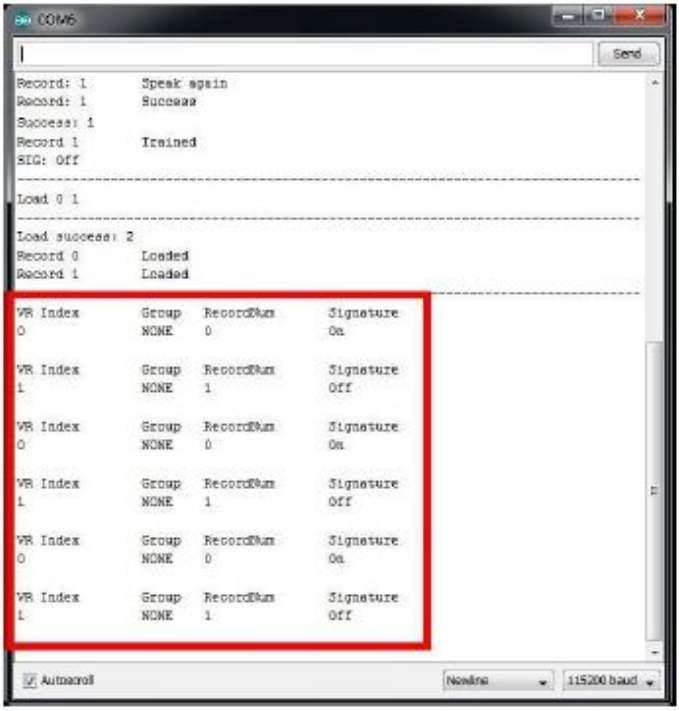

7. Train another record. Send `sigtrain 1 Off` command to train record 1 with signature "Off". Choose your favorite words to train (it can be any word, meaningful word recommended, may be 'Off' here).




8. Send `load 0 1` command to load voice. And say your word to see if the Voice Recognition Module can recognize your words.




If the voice is recognized, you can see.




9. Train finish. Train sample also support several other commands.

| COMMAND | FORMAT | EXAMPLE | COMMENT |
|---|---|---|---|
| train | train (r0) (r1)... | train 0 3 45 | Train records |
| load | load (r0) (r1) ... | load 0 51 2 3 | Load records |
| clear | clear | clear | remove all records in Recognizer |
| record | record / record (r0) (r1)... | record / record 0 75 | Check record train status |
| vr | vr | vr | Check recognizer status |
| getsig | getsig (r) | getsig 0 | Get signature of record (r) |
| sigtrain | sigtrain (r) (sig) | sigtrain 0 ZERO | Train one record(r) with signature(sig) |
| settings | settings | settings | Check current system settings |

### Load a Voice Record or Records to Recognizer (30)

Load records(1~7) to recognizer of VR3, after execution the VR3 starts to recognize immediately.
**Format:**
| AA| 3+n | 30 | R0 | ... | Rn | 0A |
**Return:**
| AA| 3+2n | 30 | N | R0 | STA0 | ... | Rn | STAn | 0A |

| | Description |
|---|---|
| **R0~Rn** | Voice Record index |
| **STA0~STAn** | Load result<br>• **00** -- Success<br>• **FF** -- Record value out of range<br>• **FE** -- Record untrained<br>• **FD** -- Recognizer full<br>• **FC** -- Record already in recognizer |

## Train One Record and Set Signature (21)

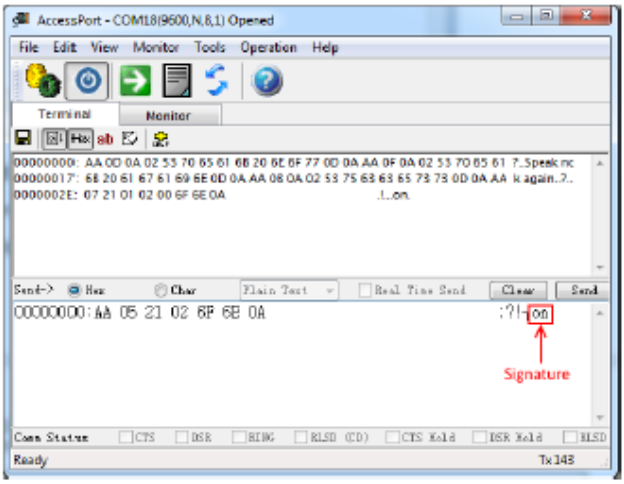
Train one record and set a signature for it, one record one time.
**Format:**
| AA| 03+SIGLEN | 21 | RECORD | SIG | 0A | (Set signature)
**Return:**

| AA| LEN | 0A | RECORD | PROMPT | 0A | (train prompt)
| AA| 05+SIGLEN | 21 | N | RECORD | STA | SIG | 0A |

| | Description |
|---|---|
| **RECORD** | Voice command record index |
| **SIG** | Signature string |
| **PROMPT** | Prompt string:<br>• Speak now<br>• Speak again<br>• Success |
| **N** | Number of successful training voice commands |

Example:

Train command 02 with signature "on"



-----Halaman ini sengaja dikosongkan-----

# RIWAYAT HIDUP

Riza Kamelia lahir di Kediri pada tanggal 31 Juli 1995. Menempuh pendidikannya di SDN Purwoasri 1 pada (2001-2004), kemudian pindah ke SDN Kutorejo 1 Kertosono, Nganjuk dan menempuh pendidikan dari (2004 – 2007). Lalu meneruskan di SMP Negeri 1 Kertosono (2007-2010), SMK Negeri 1 Kediri (2010-2013) dengan mengambil jurusan Teknik Audio Video, dan D3 Teknik Elektro ITS Surabaya (2013-2016). Pada tahun 2016, melanjutkan pendidikan Lintas Jalur di Institut Teknologi Sepuluh Nopember (ITS), Departemen Teknik Elektro. Di Departemen Teknik Elektro ini, kemudian memilih Elektronika sebagai bidang studi yang ditekuni. Pada bulan Juli 2018, mengikuti seminar dan ujian Tugas Akhir sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Teknik.

-----Halaman ini sengaja dikosongkan-----