SISTEM KENDALI NAVIGASI *QUADCOPTER* MENGGUNAKAN SUARA MELALUI *SMARTPHONE* DAN ARDUINO DENGAN METODE *TEXT PROCESSING*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Faviansyah Arianda Pallas
NIM: 135150307111042

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

SISTEM KENDALI NAVIGASI *QUADCOPTER* MENGGUNAKAN SUARA MELALUI *SMARTPHONE* DAN ARDUINO DENGAN METODE *TEXT PROCESSING*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Faviansyah Arianda Pallas
NIM: 135150307111042

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Gembong Edhi Setyawan, S.T., M.T.	Barlian Henryranu Prasetio, S.T., M.T.
NIK: 201208 761201 1 001	NIP. 201102 821024 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2017

Faviansyah Arianda Pallas

NIM: 135150307111042

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik serta hidayah-Nya, sehingga peneliti dapat menyelesaikan laporan skripsi dengan judul "Sistem kendali navigasi *quadcopter* menggunakan suara melalui *smartphone* dan arduino dengan metode *text processing*" dapat diselesaikan dengan baik.

Dalam penyusunan dan penelitian skripsi ini tidak lepas dari bantuan moral dan materiil yang diberikan dari berbagai pihak, maka peneliti mengucapkan banyak terima kasih kepada:

1. Bapak Suryanto dan Ibu Nurniamah selaku orang tua yang penulis cintai serta seluruh keluarga bbesar yang selalu memberi dukungan dan do'a agar penulis dapat menyelesaikan skripsi ini dengan lancar.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Heru Nurwarsito, Ir., M.Kom. selaku Wakil Ketua I Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
5. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang.
6. Bapak Gembong Edhi Setyawan, S.T., M.T. selaku dosen pembimbing satu yang telah memberikan ilmu, saran, penjelasan dan motivasi serta membantu dalam penyusunan laporan penulis.
7. Bapak Barlian Henryranu Prasetio, S.T., M.T., selaku dosen pembimbing dua yang telah memberikan ilmu, saran, penjelasan dan motivasi kepada penulis.
8. Seluruh civitas akademika Informatika Universitas Brawijaya dan terkhusus untuk teman-teman Teknik Komputer Angakatan 2013 yang telah banyak memberi bantuan dan dukungan selama peneliti menempuh studi di Teknik Komputer Universitas Brawijaya dan selama penyelesaian skripsi ini.
9. Amalia Apsari S. Si., grup Kedai 999, grup *Quadcopter* dan seluruh teman-teman dari Teknik Komputer yang tidak dapat disebutkan namanya satu persatu yang telah memberikan dukungan dan do'a.
10. Seluruh pihak yang tidak dapat diucapkan satu persatu, peneliti mengucapkan banyak terima kasih atas segala bentuk dukungan dan doa sehingga laporan skripsi ini dapat terselesaikan.

Peneliti menyadari bahwa tulisan ini masih jauh dari kata sempurna dan masih memiliki berbagai macam kekurangan. Oleh karena itu penulis mengharapkan kritik dan saran yang membangun, agar ke depannya penulis dapat menjadi lebih baik lagi. Semoga isi Laporan Skripsi ini dapat memberi manfaat bagi perkembangan ilmu pengetahuan di kemudian hari.

Malang, 2 Agustus 2017

Faviansyah Arianda Pallas

faviansyah1@gmail.com

ABSTRAK

Quadcopter merupakan kategori robot terbang yang dilengkapi dengan empat motor dan empat baling-baling di bagian sampingnya. Selama ini, sistem kendali *quadcopter* membutuhkan *remote control*, sehingga perlu keahlian khusus supaya *quadcopter* dapat terbang sesuai dengan keinginan. Berdasarkan masalah tersebut, maka dikembangkan sistem kendali menggunakan suara melalui *smartphone* dan Arduino dengan metode *text processing.* Data suara yang masuk diubah menjadi teks. Setiap teks yang masuk berupa kalimat dipisah menjadi satuan kata. Kemudian tiap kata dicocokkan dengan *database* yang telah dibuat sebelumnya menggunakan algoritma *stopword removal wordlist*. Hasil dari pecocokan *database* dikirim ke Arduino dan dilakukan pengolahan data untuk menghasilkan keluaran berupa gerakan *quadcopter. Quadcopter* yang digunakan dalam penelitian ini berjenis *Parrot Ar Drone 2.0*. Pengujian ditekankan pada pengguna yang berbeda-beda, performa aplikasi dan performa terbang *quadcopter*. Dari hasil pengujian, sistem ini memiliki nilai kebenaran 100% pada pengujian fungsional dan pengujian pengolahan masukan pengguna. Pada pengujian ketepatan gerakan *quadcopter*, akurasi sistem mencapai 89.3%. Selanjutnya Pengujian performa aplikasi, sistem ini memiliki rata-rata waktu yang bertambah ±1 detik dengan banyaknya masukan kata, sedangkan proses untuk pengolahan kalimat bertambah sebesar ±300 *nano* detik seiring dengan banyaknya masukan kata. Untuk pengujian performa terbang, diambil nilai data terbang *quadcopter* berupa sudut *roll, pitch,* dan *yaw* dengan *range* antara 180^{O} hingga -180^{O}. Sedangkan nilai ketinggian minimal 0,062 meter dan maksimal 3 meter.

Kata kunci: Arduino, *quadcopter*, *smartphone,* suara, dan *text processing*.

ABSTRACT

Quadcopter categorized as flying robot that has four motors and four propelers each side. At this time, remote control requires to navigate quadcopter, so special skill and experience needed to navigate it well. Based on that problem, we developed a control system using sound through smartphone and Arduino with text processing method. Incoming sound from user converted into text, and then each text splitted into words. Each word will be matched into database that has been created before using stopword removal wordlist algorithm. The results will be sent to Arduino to do data processing becoming quadcopter movement. Using parrot Parrot Ar Drone 2.0 as quadcopter in this research. Testing method focused on different user, application and fliying quadcopter performance. From testing result, this system has 100% value at functional testing and user masukan *processing. At quadcopter precision movement testing, accuracy of this system reach 89.3%. At application performance testing, this system average time for sound processing increase ±1 second and average time for text preprocessing ±300 nanosecond along with the increase of words. At quadcopter fliying performance, taken quadcopter angel value of roll, pitch, and yaw within -180^{O} to +180^{O}. While minimum value of height is 0.062 meter and maximum height is 3 meter.*

Keywords: Arduino, quadcopter, smartphone, sound, and text processing.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Quadcopter merupakan robot terbang dengan jenis pesawat tanpa awak atau biasa disebut dengan *Unmanned Aerial Vehicle* (UAV). *Quadcopter* termasuk kategori UAV *Micro* yaitu robot kecil dan ringan dengan berat kurang dari 5 kg yang dilengkapi dengan empat motor dan baling-baling di bagian sampingnya, dua motor bergerak searah jarum jam dan dua bagian motor lainnya bergerak berlawanan jarum jam. Dua jenis pergerakan motor tersebut menyebabkan *quadcopter* mampu terbang secara stabil di udara (Robotika, 2015). *Quadcopter* memiliki beberapa kelebihan, bentuknya yang kecil dan mampu terbang secara vertikal tanpa memerlukan landasan pacu membuat *quadcopter* cukup leluasa untuk bergerak di tempat-tempat yang sulit. Selain itu *quadcopter* juga mampu bergerak ke empat arah mata angin tanpa perlu memutar badannya terlebih dahulu (Tristiadi, 2015).

Quadcopter dapat dikendalikan sesuai keinginan manusia dengan menggunakan sistem kendali berupa *remote control* yang memanfaatkan gelombang radio sebagai sistem komunikasinya. Selain itu, *quadcopter* juga dapat diimplementasikan menggunakan berbagai macam sistem kendali seperti *joystick*, *leap motion*, dan lain-lain (Hernandez-Martinez, et al., 2015). Namun dengan alat tersebut membutuhkan petunjuk penggunaan dan keahlian. Pada penelitian sebelumnya, *quadcopter* dapat dikontrol menggunakan pengolahan suara dengan metode *Support Vector Machine* (SVM), di mana suara diolah menggunakan *MatLab* (Supimrosl & Wongthanavasu, 2014). Selain itu, *quadcopter* dapat dikontrol menggunakan *gesture* dan suara dengan memanfaatkan *LabView* sebagai platform pengolahan suaranya (Krishna, et al., 2011). Namun, penelitian tersebut kurang efektif, hal ini dikarenakan tingkat kesalahan deteksi suara dan kata masih tinggi, serta sulit untuk mengeksekusi banyak perintah dalam satu waktu. Dari permasalahan tersebut, dikembangkan sistem kontrol *quadcopter* menggunakan suara melalui *smartphone* dan arduino dengan metode *text processing*. Kelebihan dari sistem ini, pengguna tidak membutuhkan keahlian apapun untuk mengendalikan *quadcopter*, karena hanya diperlukan perintah suara menggunakan bahasa sehari-hari. Metode yang digunakan akan lebih efektif dibandingkan metode sebelumnya, karena sistem yang dibuat berorientasi pada deteksi dan pengolahan kata masukan dari *user* (Dye, 2016).

Speech recognition sendiri merupakan teknologi komputer masa kini yang digunakan untuk mengidentifikasi suara yang diucapkan oleh seseorang tanpa mengetahui siapa yang berbicara. Implementasi *speech recognition* dapat digunakan untuk berbagai macam platform, misalnya perintah suara untuk menjalankan aplikasi komputer, perintah suara untuk mengoperasikan *hardware*, dan lain-lain. Cara kerja yang dilakukan oleh sistem dengan mendeteksi tingkat penekanan suara, lalu dicocokkan dengan *database* yang tersedia (Kisumal, 2010). Penggunaan *speech recognition* dapat digabungkan dengan berbagai macam *mikrokontroler*, seperti *Arduino*, *Raspberry Pi*, dan *AVR.* Pada penelitian ini

digunakan mikrokontroler Arduino, Karena perangkat lunaknya mudah dikembangkan dan mampu mencangkup semua kebutuhan dalam penelitian.

Smartphone atau Android *device* dimanfaatkan untuk masukan suara dari *user* dan tempat pengolahan kalimat, sedangkan Arduino sebagai menerima dan mengolah data dari Android *device*. Hasil proses pada Arduino akan dijadikan gerakan *quadcopter*. *Quadcopter* yang digunakan pada penelitian ini berjenis *Parrot Ar Drone 2.0. Quadcopter* jenis ini dapat dikembangkan dengan bebas karena setiap komponen penyusunnya dapat diakses untuk menghasilkan *feedback* berupa data navigasi atau data terbang dan dilengkapi dengan mekanisme sistem stabilisasi sehingga pengembang dapat berfokus pada algoritma dengan level yang lebih tinggi (Vyskovsky, 2015).

Parameter yang diuji berupa masukan suara dari individu yang berbeda beda, pengujian efektivitas sistem dan uji waktu yang dibutuhkan aplikasi untuk mengolah data, uji masukan suara dan uji pemisahan kalimat dengan *user* yang berbeda, dan yang terakhir uji performa gerakan *quadcopter*. Dengan adanya skripsi ini, penulis berusaha mengembangkan teknologi kontrol *quadcopter* supaya lebih mudah pengoperasiannya dalam berbagai bidang khususnya melalui suara.

1.2 Rumusan Masalah

Seperti uraian latar belakang sebelumnya, maka perumusan masalah yang akan dibahas adalah sebagai berikut:

1. Bagaimana menggunakan *speech recognition* Android berbasis Arduino sebagai sistem kontrol *quadcopter*?
2. Bagaimana akurasi dan efektivitas pengolahan teks terhadap masukan suara dari *user*?
3. Bagaimana efektivitas *speech recognition* dan *text processing* jika digunakan untuk kontrol *quadcopter*?
4. Berapa waktu yang dibutuhkan oleh sistem untuk melakukan pemrosesan teks?
5. Bagaimana performa *quadcopter* ketika dilakukan kontrol?

1.3 Tujuan

Tujuan dari penelitian ini menjawab dari rumusan masalah yang ada seperti:

1. Untuk mengetahui penggunaan sistem *speech recognition* Android *device* berbasis Arduino sebagai kontrol *quadcopter*.
2. Untuk mengetahui akurasi dan efektivitas pengolahan teks terhadap masukan suara dari *user*.
3. Untuk Meneliti efektivitas *speech recognition* dan *text processing* jika digunakan untuk kontrol *quadcopter*.
4. Untuk mengetahui waktu yang dibutuhkan sistem untuk melakukan pemrosesan teks.
5. Untuk meneliti performa *quadcopter* ketika dikontrol.

1.4 Manfaat

Manfaat yang diperoleh dengan adanya penelitian ini adalah:

1. Bisa mengontrol *quadcopter* dengan menggunakan perintah suara.
2. Bagi para peneliti, hasil penelitian dapat dikembangkan menjadi penelitian baru.
3. Mengetahui tingkat efektivitas *speech recognition* sebagai kontrol *quadcopter*.
4. Mengetahui tingkat efektivitas *text processing* yang dipadukan dengan *speech recognition* sebagai kontrol *quadcopter*.
5. Mengetahui performa gerakan *quadcopter* dengan menggunakan gambaran dari nilai data navigasi *quadcopter*.

1.5 Batasan Masalah

Adapun batasan masalah agar tidak menyimpang dari perumusan masalah adalah sebagai berikut:

1. Masukan suara Memanfaatkan *Android Google Speech*.
2. Pengolahan *speech recognition* menggunakan Android *device*.
3. *Quadcopter* yang digunakan adalah *Parrot AR Drone 2.0*.
4. Sistem yang dibuat diuji di dalam ruangan.
5. Kata yang digunakan berupa gabungan gerakan dasar pada *quadcopter* dan kata imbuhan.
6. Digunakan kata penghubung untuk stuktur kata yang berbeda.
7. Akurasi sistem dan performa terbang *quadcopter* sebagai tolak ukur efektivitas sistem.

1.6 Sistematika Pembahasan

Sistematika penulisan bertujuan sebagai penjelasan umum dari bagian-bagian bab yang ada dalam penelitian ini agar memudahkan pembaca dalam mengkuti alur pembahasan penelitian. Adapun sistematika penulisan adalah sebagai berikut:

BAB I: Pendahuluan

Pada bab I memuat latar belakang permasalahan, rumusan masalah, pembatasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II: Dasar Teori

Pada bab II berisi tentang penjelasan teori-teori dasar yang menjadi acuan dalam melaksanakan penerapan penelitian kontrol *quadcopter* menggunakan suara melalui *smartphone* dan arduino dengan menggunakan metode *text processing*, beberapa penjelasannya dikutip dari beberapa studi literatur seperti *paper*, buku, dan lain-lain.

BAB III: Metodologi Penelitian

Pada bab III ini berisi tentang metodologi penelitian beberapa hal yang akan dibahas pada bab ini di antaranya analisis kebutuhan, perancangan sistem, implementasi, pengujian serta analisisnya, dan yang terakhir berupa penarikan kesimpulan dan pemberian saran dari penelitian yang akan dilakukan.

BAB IV: Analisis Kebutuhan

Pada Bab IV berisi penjelasan mengenai kebutuhan-kebutuhan yang terkait dalam penelitian ini seperti kebutuhan pengguna,Kebutuhan sistem,kebutuhan *software*, dan kebutuhan *hardware*.

BAB V: Perancangan sistem dan Implementasi

Pada bab V berisi penjelasan mengenai perancangan dan implementasi sistem, seperti diagram komunikasi sistem dan diagram alur kerja sistem.

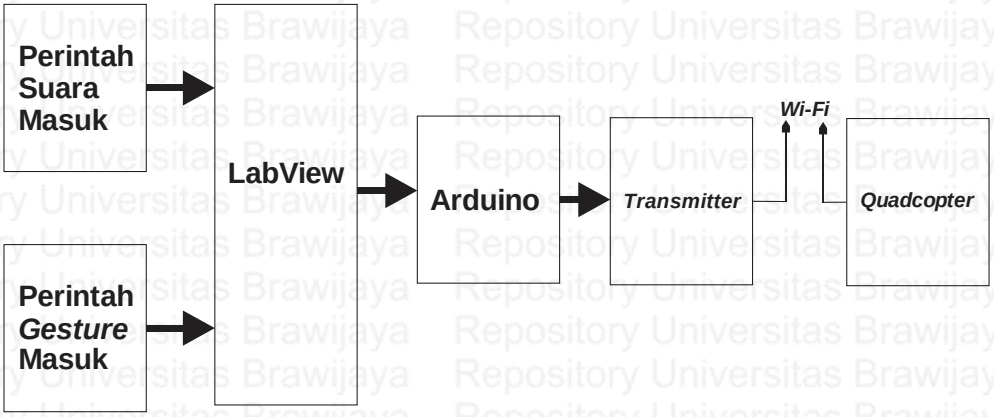
BAB VI: Pengujian dan Analisis

Pada bab VI berisi penjelasan mengenai proses pengujian dimulai dari tujuan pengujian, prosedur pengujian, pelaksanaan pengujian, hasil pengujian dan dilakukan analisis hasil pengujian yang sudah dilakukan.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian sebelumnya yang telah dilakukan yaitu dengan menerapkan pengolahan suara atau *gesture* sebagai kontrol *quadcopter*. *Quadcopter* dikendalikan melalui olahan suara dengan didasarkan perbedaan amplitudo dan frekuensi pada setiap individu. Sistem ini menggunakan *LabView* sebagai platform untuk masukan suara dan pengolahannya. Hasil pengolahan suara tersebut diolah oleh Arduino menjadi gerak pada *quadcopter*. Penelitian yang dikerjakan menggunakan *voice recognition* sebagai masukan untuk pergerakan *quadcopter*, di mana *voice recognition* lebih berorientasi kepada "siapa yang berbicara" daripada "apa yang dibicarakan", sehingga perlu pengolahan data suara untuk nantinya bisa dikenali jenis perintah apa yang di masukkan oleh *user*.

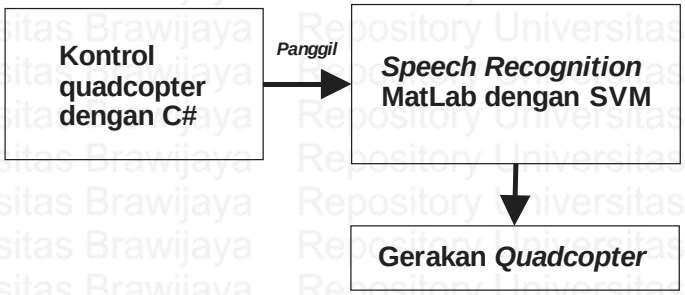


Gambar 2.1 Proses pengolahan *speech recognition*

Sumber: (Krishna, et al., 2011)

Alur program yang dibuat dapat dilihat pada Gambar 2.1 awalnya *user* memberikan masukan berupa suara yang akan diolah oleh *labView* menghasilkan masukan serial yang dikirimkan ke Arduino. Kemudian, *Arduino* memiliki *transmitter* dihubungkan dengan *quadcopter*, sehingga *quadcopter* akan bergerak sesuai dengan suara masukan dari *user* (Krishna, et al., 2011).

Penelitian rujukan yang kedua membahas kontrol *quadcopter* menggunakan *Speech recognition* menggunakan *Support Vector Machine* (SVM) dengan memanfaatkan *MatLab* sebagai platform pengolahannya. *Quadcopter* dikontrol menggunakan masukan suara yang diolah oleh *Matlab*, dengan keluaran yang dihasilkan berupa *speech recognition* untuk menggerakkan *quadcopter*. Mekanisme kerjanya dapat dilihat pada Gambar 2.2.



Gambar 2.2 *Interface Computer* dengan *quadcopter*

Sumber: (Supimrosl & Wongthanavasu, 2014)

Pada Gambar 2.2 Pergerakkan *quadcopter* diatur dengan bahasa C#, di mana perintah arah pergerakan didapatkan melalui pengolahan suara yang telah di masukkan oleh *user*. Hasil perintah suara dalam penelitian Supimrosl & Wongthanavasu (2014) dilatih menggunakan *Support Vector Machine* (SVM). Sistem SVM yang digunakan menggunakan integrasi *Matlab*, sehingga mempermudah dalam pengolahan frekuensi suara. Langkah pertama yang dilakukan untuk pengolahan suara dalam penelitian rujukan ini yaitu *Preprocessing* sinyal, menggunakan filter *Finite Impulse Response*. Lalu dilakukan ekstraksi fitur dengan beberapa parameter yang diambil berupa "frekuensi *fundamenal*", "energi", dan "*Frekuensi Mel*". Dengan langkah terakhir yaitu mengklasifikasikan *Speech recognition* menggunakan SVM (Supimrosl & Wongthanavasu, 2014). Hasil pengujian sistem ini memiliki akurasi sebesar 74,44%, dengan 10 kali percobaan pada setiap gerakannya.

Berdasarkan pada penelitian Krishna (2011) sistem yang dibuat mengandalkan *voice recognition*, yang mana pada sistem tersebut lebih berfokus pada "siapa yang berbicara". Sedangkan penelitian Supimrosl & Wongthanavasu (2014) berfokus pada pengolahan suara pada MatLab dengan hasil pengujian akurasi sebesar 74,44% yang menurut peneliti perlu dikembangkan lebih lanjut. Selain itu, 2 penelitian yang telah dilakukan sebelumnya berfokus pada kontrol menggunakan satuan kata, pengguna tidak bisa memberikan perintah suara dalam bentuk kalimat. Sehingga perlu dibuat kendali *quadcopter* menggunakan suara dengan memanfaatkan *smartphone* yang dipadukan dengan metode *text processing* untuk pemrosesan masukan kalimat dan meningkatkan akurasi kendali.

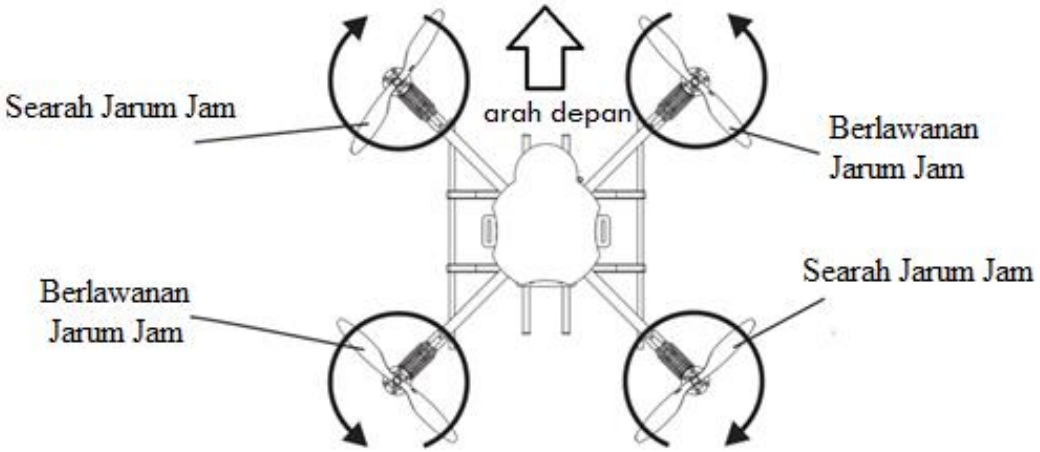
2.2 Dasar Teori

Pada bagian ini akan dibahas teori-teori yang berkaitan dengan penelitian yang akan dilakukan di antaranya *Unamnned Aerial Vehicle* (UAV) *Quadcopter*, Arduino, *Bluetooth Module*, *Speech recognition*, *Application Programming Interface,* dan *Text processing.*

2.2.1 *Unamnned Aerial Vehicle*(UAV) *Quadcopter*

Quadcopter merupakan pesawat helikopter mini dengan jenis pesawat tanpa awak atau *Unamnned Aerial Vehicle*(UAV) yang memiliki 4 motor dan 4 baling-baling pada tiap sisinya (Robotika, 2015). Terdapat 2 jenis baling-baling

yang digunakan pada *quadcopter*, yaitu baling-baling yang menghadap ke atas dan ke bawah. Keempat letak baling-baling serta motornya diberi jarak yang sama pada tiap sisi sampingnya dengan titik tengah *mainboard quadcopter*. Gambar 2.3 merupakan pergerakan baling-baling pada *quadcopter*.



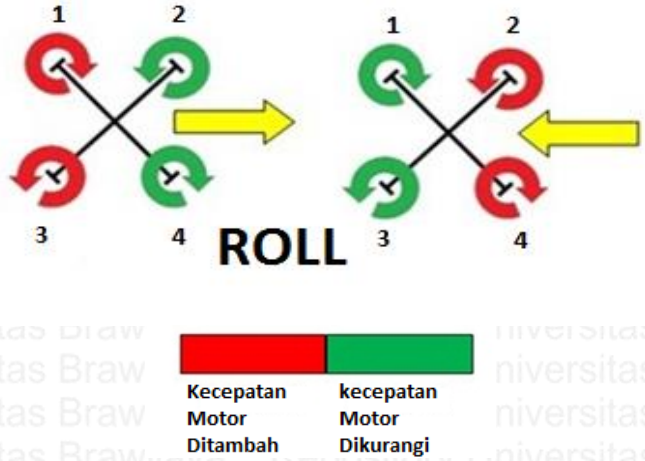
Gambar 2.3 Pergerakan Motor *Quadcopter*

Sumber: (Lawson & F., 2017)

Kecepatan setiap motor pada *quadcopter* dapat dikendalikan sesuai keinginan pengguna, sehingga *quadcopter* mampu melakukan manuver dengan berbagai jenis gerakan. Selain itu, *quadcopter* juga mampu melakukan *takeoff* dan *landing* pada bidang miring. *Quadcopter* digunakan dalam berbagai aplikasi seperti pengawasan, pencarian, dan berbagai macam aplikasi lainnya. Bentuknya yang kecil dan pegerakannya fleksibel membuat banyak instansi yang menggunakan *quadcopter* sebagai bahan percobaan.

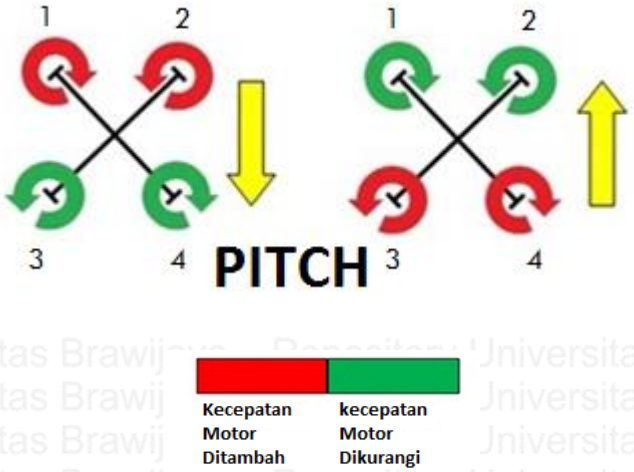
Setiap motor pada *quadcopter* memiliki fungsi pendorong, 2 motor yang sejajar berputar searah jarum jam, 2 motor lainnya berputar berlawanan dengan arah jarum jam. Motor pada *quadcopter* memiliki kecepatan perputaran yang bervariasi pada tiap baling-balingnya untuk dapat melakukan pergerakan tertentu. Untuk terbang melayang di udara, dikenal dengan sebutan *hover*, *quadcopter* perlu menggerakkan keempat baling-balingnya dengan kecepatan yang sama. Setiap pasangan baling-baling memiliki arah gaya dorong yang berbeda,satu pasang sebagai pendorong(*pusher*) dan satu pasang sebagai penarik(*puller*). Macam-macam pergerakan manuver pada *quadcopter* dapat dilihat pada Gambar 2.4.

Gerakan *ROLL* merupakan arah pergerakan *quadcopter* menurut sumbu x (Romero1, et al., 2014). pada Gambar 2.4 dapat dilihat, jika motor nomor 1 dan 3 bertambah kecepatannya sedangkan nomor 2 dan 4 berkurang, maka *quadcopter* akan bergerak ke arah kanan. Jika motor nomor 1 dan 4 kecepatannya perputarannya berkurang sedangkan motor nomor 2 dan 4 bertambah maka *quadcopter* akan bergerak ke arah kiri.



Gambar 2.4 Gerakan *ROLL quadcopter*

Sumber: (Ramadhan, 2015)

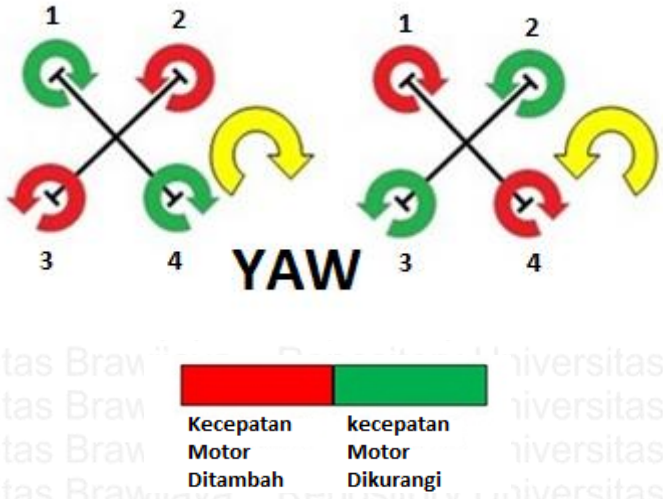


Gambar 2.5 Gerakan *PITCH quadcopter*

Sumber: (Ramadhan, 2015)

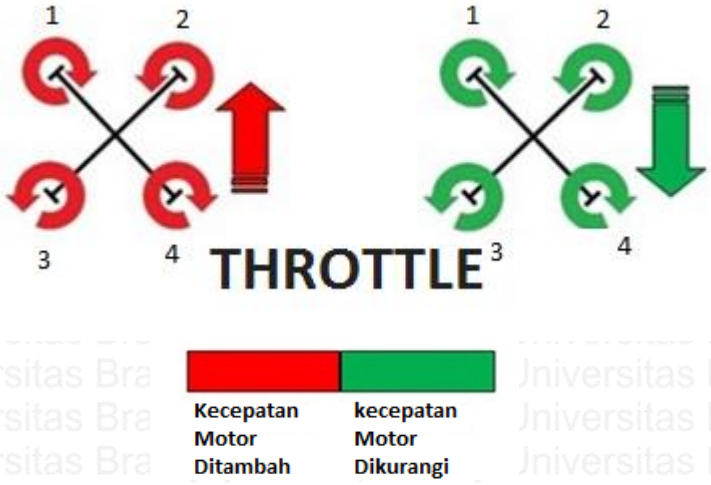
Gerakan *PITCH* merupakan arah pergerakan *quadcopter* menurut sumbu Y (Romero1, et al., 2014). pada Gambar 2.5 dapat dilihat arah pergerakan *quadctoper* akan mengarah ke depan jika kecepatan putaran motor nomor 1 dan 2 ditambah, sedangkan motor nomor 3 dan 4 dikurangi. *Quadcopter* akan bergerak ke belakang jika kecepatan putaran motor nomor 3 dan 4 ditambah, sedangkan motor nomor 1 dan 2 dikurangi.

Gerakan *YAW* merupakan arah gerakan *quadcopter* menurut sumbu Z (Romero1, et al., 2014). pada Gambar 2.6 dapat dilihat jika kecepatan putaran motor nomor 2 dan 3 bertambah sedangkan motor nomor 1 dan 4 berkurang, maka *quadcopter* akan bergerak berputar searah jarum jam. *Quadcopter* akan bergerak melawan arah jarum jam, jika kecepatan putaran motor nomor 1 dan 4 berkurang, sedangkan kecepatan motor nomor 2 dan 3 bertambah.



Gambar 2.6 Gerakan *YAW quadcopter*

Sumber: (Ramadhan, 2015)



Gambar 2.6 Gerakan *THROTTLE quadcopter*

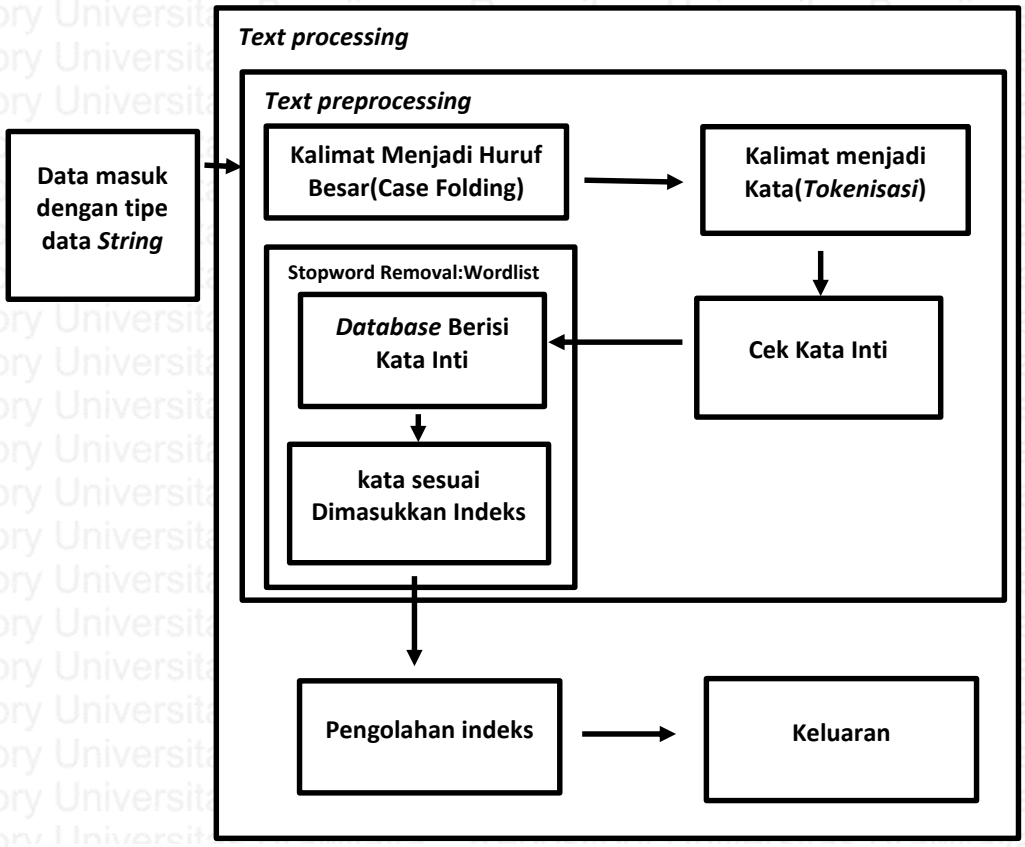
Sumber: (Ramadhan, 2015)

Gerakan *THROTTLE* adalah gerakan *quadcopter* yang bergerak dengan arah ke atas atau ke bawah (Hernandez-Martinez, et al., 2015). Pada Gambar 2.6 dapat dilihat jika pergerakan 4 motor *quadcopter* kecepatan putarannya bertambah, maka *quadcopter* akan naik ke atas, sedangkan jika 4 motor kecepatannya berkurang, maka *quadcopter* akan turun ke bawah.

Empat pergerakan yang telah dijelaskan merupakan gerakan utama pada *quadcopter.* Gerakan *roll* merupakan arah navigasi *quadcopter* menurut sumbu X*,* Gerakan kedua yaitu *pitch* merupakan arah navigasi *quadcopter* menurut sumbu Y*,* gerakan *yaw* merupakan arah navigasi *quadcopter* menurut sumbu Z*,* gerakan *throttle* merupakan gerakan naik dan turun dengan memanfaatkan ketinggian sebagai tolak ukurnya.

2.2.2 *Text Processing*

Text processing merupakan bagian dari *text mining* yang mana metode ini berfokus pada pengolahan kalimat. Kata hasil pengolahan di masukkan ke dalam indeks dan diolah dengan mengelompokkan kata inti dan imbuhan. Alur metode *text processing* yang digunakan dapat dilihat pada Gambar 2.7.

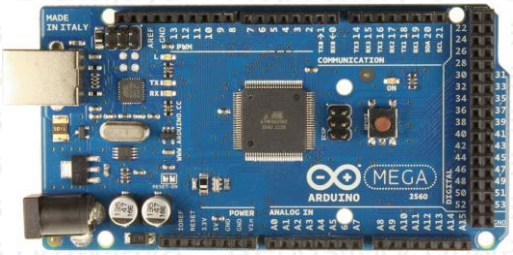


Gambar 2.7 Alur metode *Text processing*

Masukan dari metode ini berupa kalimat dengan tipe data *string*. Langkah awal yang dilakukan yaitu pada metode ini yaitu *text preprocessing*. Tahap pertama, kalimat yang masuk diubah menjadi huruf besar tahap ini dinamakan *case folding*. Selanjutnya kalimat tersebut dipecah menjadi satuan kata tahap ini dinamakan *tokenisasi*. Tahap pemilihan kata menggunakan *stopword removal* dengan algoritma *wordlist*, tahap ini untuk memilih kata-kata penting dan menyaring kata-kata yang tidak diperlukan. Hasil dari tokenisasi dicari di dalam *database* yang berisi kumpulan-kumpulan kata inti, jika kata ditemukan di dalam *database* maka data akan disimpan di dalam indeks (Fauzi, 2016). Kumpulan kata indeks yang di dapat dikelompokan dan diolah menjadi keluaran yang dinginkan.

2.2.3 Arduino

Arduino merupakan jenis mikrokontroler *Open Source* yang dikembangkan untuk mempermudah pembuatan alat-alat elektronik dalam berbagai bidang. Arduino mega dapat dilihat pada Gambar 2.8.



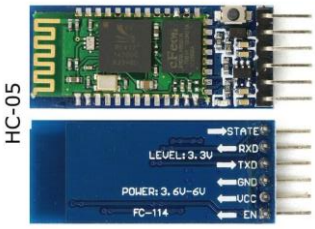
Gambar 2.8 Arduino

Sumber: (Yulias, 2013)

Arduino menggunakan prosessor *Atmel AVR* dan *softwarenya* memiliki bahasa pemrograman sendiri.

2.2.4 *Bluetooth Module* HC-05

HC-05 Adalah sebuah modul Bluetooth SPP (*Serial Port Protocol*) yang banyak digunakan untuk komunikasi tanpa kabel, prinsip kerjanya yaitu mengkonversi *port* serial ke *Bluetooth*. HC-05 memiliki 2 mode konfigurasi, yaitu *AT mode* dan *Communication mode*. *AT mode* berfungsi untuk melakukan pengaturan konfigurasi dari HC-05, Sedangkan *Communication* mode berfungsi untuk melakukan komunikasi. Pada *Bluetooth* dengan piranti lain. Jarak sinyal dari HC-05 adalah 30 meter, dengan kondisi tanpa halangan (TerryKing, 2011).



Gambar 2.9 HC-05

Sumber: (TerryKing, 2011)

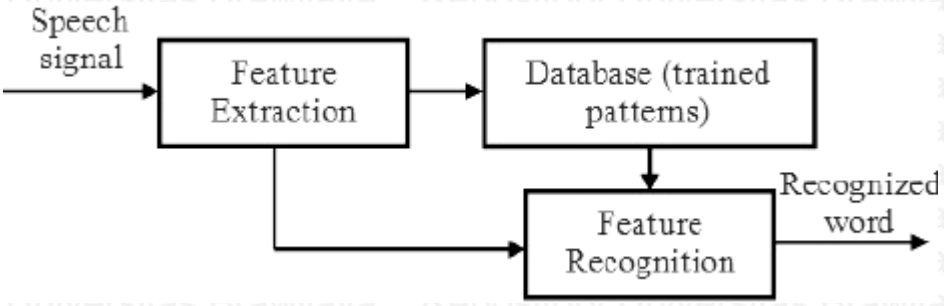
Mode komunikasi yang digunakan berupa mode *Slave,* mode ini memungkinkan Android *device* untuk terhubung dengan modul ini. Digunakan tegangan 5V supaya memiliki performa yang maksimal dalam penerimaan data.

2.2.5 *Speech recognition* Android *Device*

Speech recognition adalah proses identifikasi suara berdasarkan kata yang diucapkan dengan melakukan konversi sebuah sinyal akustik. Berbeda dengan *Sound Recognition*, *Speech recognition* lebih berorientasi kepada apa yang diucapkan daripada siapa yang mengucapkan. Di mana terdapat perbedaan yang signifikan apabila dibandingkan dengan *Sound recognition*, dilihat dari hasilnya *sound recognition* lebih berorientasi kepada manusianya, sedangkan *speech recognition* lebih berorientasi kepada hasil dari ucapan manusianya.

Terdapat 4 langkah utama dalam sistem pengenalan suara:

- Penerimaan data masukan.
- Ekstraksi, yaitu penyimpanan data masukan sekaligus pembuatan *database* untuk *template*.
- Pembandingan / pencocokan, yaitu tahap pencocokan data baru dengan data suara (pencocokan tata bahasa) pada *template*.
- Validasi identitas pengguna.



Gambar 2.10 *Speech recognition* Feautre

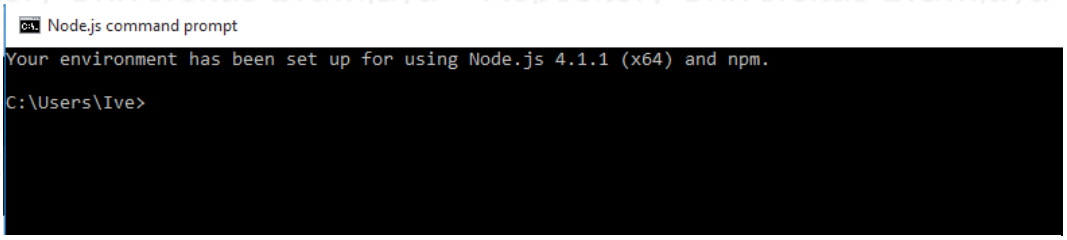
Sumber: (Chapaneri, 2012)

Langkah-langkah pengolahan speech recognition dapat dilihat pada Gambar 2.10, langkah pertama dari pengolahan *speech recognition* yaitu sinyal suara masuk ke dalam suatu platform, lalu dilakukan ekstraksi fitur yaitu pengambilan nilai suara berupa frekuensi agar nantinya bisa diolah lebih lanjut. Langkah selanjutnya yaitu mencocokkan hasil ekstraksi fitur dengan *database* yang ada, dengan hasil akhir didapatkan berupa deteksi kata yang telah diucapkan.

Dalam penggunaanya *speech recognition* dapat diolah menggunakan berbagai macam platform, seperti Matlab, LabView, Android *device*, Arduino dan masih banyak platform lainnya.

2.2.6 Node.JS

Node.JS adalah suatu platform yang dibuat dengan menggunakan *javaScript engine*. Node.JS dibuat dengan tujuan *javaScript* tidak dipakai hanya pada platform web saja, namun bisa digunakan diberbagai platform.



Gambar 2.11 Node.JS *Command* Prompt

Pada Gambar 2.11 dapat dilihat Node.JS *command prompt* yang digunakan untuk menjalankan berbagai perintah *node*. Pada sistem ini Node.JS digunakan sebagai tempat program yang bertujuan untuk mengendalikan *quadcopter*. Pada

Node.JS terdapat kumpulan *node* yang disebut dengan *Node Package Manager* atau yang biasa di singkat dengan nama NPM semua komponen pada Node.JS terintegrasi satu sama lain dan menghasilkan berbagai perintah *quadcopter* seperti *takeoff*, *landing*, kanan, Kiri, Atas, dan lain lain . Setiap perintah tersebut dapat digunakan secara bersama-sama sehingga mempermudah *user* dalam pengoperasiannya.

2.2.7 *Node Package Manager*

Node Package Manager(NPM) merupakan sekumpulan paket yang tersedia pada Node.JS. NPM dibuat untuk mempermudah para pengembang *javaScript* dalam membangun suatu sistem dengan berbagai paket yang disediakan, paket tersebut dapat dikembangkan secara bebas untuk dimanfaatkan pada suatu penelitian (Prameswara, 2016). Penggunaan NPM dalam kontrol *quadcopter* berupa NPM *AR.Drone* 2.0 dan NPM *serialport*.

2.2.7.1 *Node Ar.Drone 2.0*

Node AR Drone 2.0 merupakan sekumpulan perintah yang digunakan untuk mengeksekusi atau memberikan perintah pada *quadcopter*, seperti *takeoff*, *landing*, ataupun mendapatkan data sensor yang terdapat pada *quadcopter* (Felixge, 2014). *Node AR Quadcopter* ditulis menggunakan bahasa *javaScript*, serta dapat membantu *user* untuk mengembangkan penelitian tentang *mainboard* AR *Quadcopter* 2.0. Dalam pengembangannya *Node AR.Drone* mampu menyediakan berbagai macam perintah yang ditujukan untuk pergerakan *quadcopter*, dengan mengintegrasikan pergerakan utama, *Roll*,*Pitch*, dan *Yaw*.

2.2.7.2 *Node serialport*

Node Serialport adalah suatu perantara yang menghubungkan mikrokontroler pada *quadcopter* dengan mikrokontroler lain, seperti Arduino, *raspberi pi*, *AVR*, dan lain lain. 2 mikrokontroler dapat saling berkomunikasi menggunakan komunikasi *serial*. Pada dasarnya *serial port* merupakan komponen terpenting dalam hal kontrol *quadcopter*, karena integrasi antar mikrokontroler dilakukan menggunakan Node ini.

2.2.8 Penghitungan Persentase Akurasi

Nilai Persentase Akurasi suatu sistem dapat dihitung menggunakan rumus 2.1

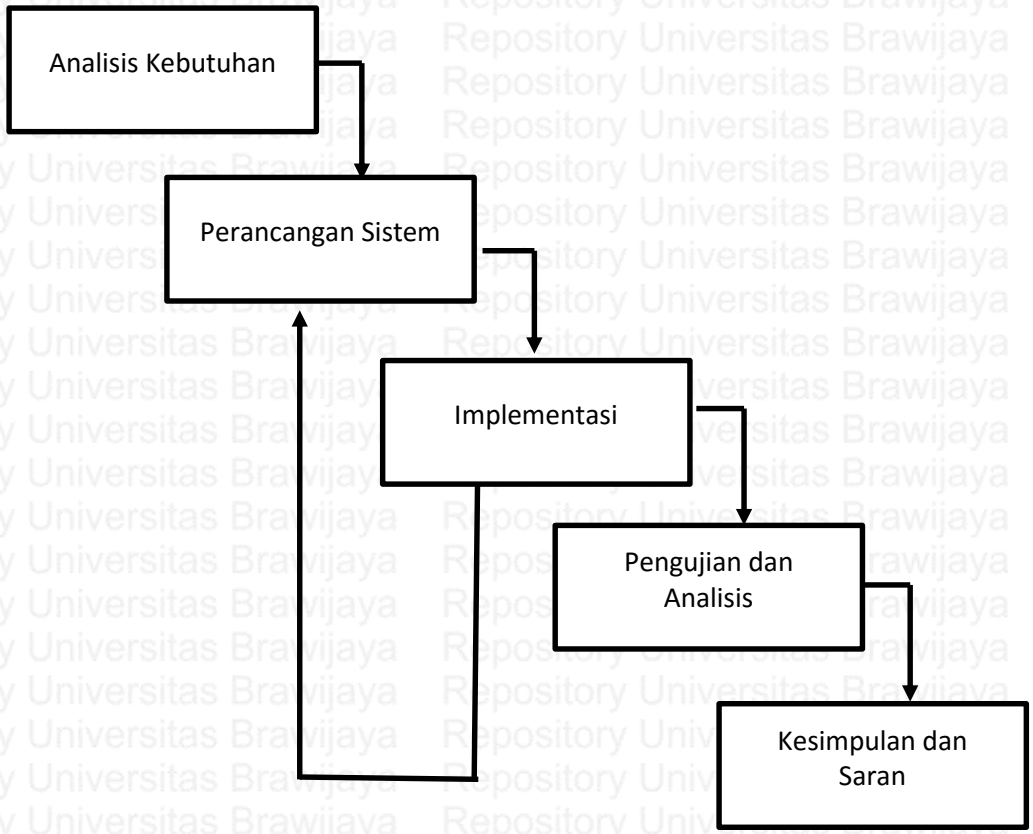
$$\text{Nilai Persentase Akurasi} = \frac{\text{Jumlah nilai Benar}}{\text{Jumlah nilai keseluruhan}} \text{X } 100\% \qquad (2.1)$$

Nilai Persentase Akurasi diperoleh dari Nilai data benar pada suatu sistem dibagi dengan nilai data keseluruhan.

BAB 3 METODOLOGI

3.1 Metode Penelitian

Pada bab ini akan menjelaskan tentang metodologi penelitian yang akan digunakan dalam melakukan penelitian maupun dalam penulisan skripsi. Adapun gambaran diagram alir metodologi yang digunakan dalam penelitian dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur metodologi penelitian

Gambar 3.1 merupakan metodologi penelitian yang digunakan, pada bab analisis kebutuhan akan dibahas kebutuhan yang digunakan untuk penelitian ini. Pada bab perancangan sistem dan implementasi akan dibahas mengenai bagaimana merancang sistem yang dibuat lalu di implementasi ,namun apabila implementasi tidak sesuai dengan yang diharapkan maka akan dirancang kembali. jika sesuai dengan harapan, maka akan di lanjut ke bab pengujian dan analisis. Pada bab pengujian dan analisis hasil yang telah dirancang akan diuji dan dianalisis hasil pengujiannya lalu ditarik kesimpulan dan pengambilan saran.

3.2 Analisis kebutuhan

Pada bagian ini dibahas mengenai dasar-dasar teori yang digunakan untuk mendukung penelitian tentang perancangan kontrol *quadcopter* menggunakan

suara melalui *smartphone* dan Arduino dengan metode *text processing*. Terdapat beberapa sub bab yang dibahas, di antaranya gambaran umum sistem yang menjelaskan tentang bagaimana sistem ini dibuat, analisis kebutuhan sistem yang membahas kebutuhan-kebutuhan yang digunakan untuk penelitian dan dibagi menjadi kebutuhan pengguna, *hardware*, komunikasi, dan *software.* Lalu kebutuhan *fungsional* dan kebutuhan *non fungsional*. Penjelasan lebih rinci akan dibahas pada bab 4 analisis kebutuhan.

3.3 Perancangan Sistem

Perancangan sistem merupakan tahapan bagaimana membangun sebuah sistem dari penelitian yang dilakukan. Tahapan ini dilakukan setelah melakukan tahapan analisis kebutuhan. Dengan adanya tahap ini maka sistem akan dapat digambarkan secara sistematis dan terstruktur. Perancangan sistem akan dibahas dari segi perancangan sistem pada Android, perancangan sistem pada Arduino, dan Perancangan sistem pada Node.Js. Penjelasan lebih rinci akan dibahas pada bab 5 perancangan dan implementasi.

3.4 Implementasi Sistem

Implementasi sistem dilaksanakan sesuai dengan perancangan yang telah ditentukan sebelumnya, mulai dari analisis kebutuhan hingga perancangan sistem. Langkah pertama yang dilakukan dalam implementasi sistem yaitu *User* memberikan masukan berupa perintah suara melalui program yang telah dibuat menggunakan *platform* Android. Program tersebut mampu melakukan *Text preprocessing* yaitu langkah untuk memecah hasil masukan suara dan diolah untuk nantinya dikirim ke Arduino melalui komunikasi *bluetooth*. Langkah selanjutnya pada Arduino merangkai *hardware* untuk komunikasi dengan memanfaatkan HC-05. Setelah komponen tersebut saling terhubung, dirangkai perangkat lunak pada Arduino melakukan proses pengolahan kalimat untuk dikirim ke Node.JS. Selanjutnya Node.JS yang akan memberikan perintah kepada *quadcopter* ke mana arah terbang *quadcopter* tersebut. Tentunya arah terbang *quadcopter* sesuai dengan perintah suara yang dihasilkan oleh *user*. Quadcopter akan memberikan feedback berupa data arah terbang. Penjelasan lebih rinci akan dibahas pada bab 5 perancangan dan implementasi.

3.5 Pengujian dan Analisis Sistem

Tahap pengujian dan analisis sistem untuk menguji apakah sistem yang dibuat sudah sesuai seperti yang diharapkan oleh peneliti. Pengujian dan analisis menggunakan beberapa Parameter, Berikut ini adalah parameter yang digunakan dalam pengujian sistem:

1. Pengujian fungsional dari sistem. Hal ini ditujukan untuk mengetahui apakah sistem sudah berjalan sesuai dengan keinginan peneliti. Tolak ukur dalam hal ini dapat dilihat melalui *console log* pada Android studio, serial monitor pada Arduino, dan *command prompt* pada Node.JS .

2. Pengujian masukan kalimat dan pengolahan teks. Pengujian ini ditujukan untuk melihat akurasi dari pengolahan suara dan pengolahan teks, dengan menggunakan individu yang berbeda.
3. Pengujian gerakan *quadcopter* dengan individu yang berbeda-beda. Individu yang berbeda digunakan untuk menguji efektivitas sistem, apakah sistem mampu mengolah masukan dan menghasilkan keluaran sesuai dengan harapan peneliti.
4. Pengujian waktu yang diperlukan aplikasi untuk mengolah suara dari *user*. Tahap ini untuk menguji *delay* dari penggunaan perintah suara jika digunakan untuk pergerakan *quadcopter*.
5. Pengujian performa gerakan pada *quadcopter*. Pengujian ini untuk membuktikan dan menganalisis apakah *quadcopter* mampu bergerak sesuai dengan keinginan. Data yang diperoleh dari navigasi data *quadcopter* akan dipetakan menjadi grafik.

3.6 Penarikan kesimpulan dan saran

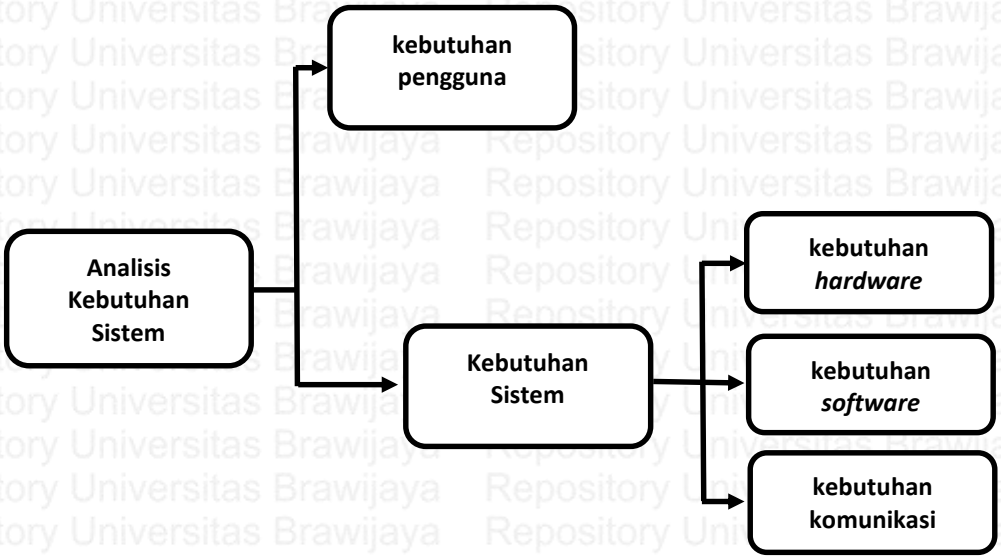
Penarikan kesimpulan merupakan tahap yang dilakukan setelah melakukan seluruh kegiatan pengujian sistem yang telah dirancang sebelumnya. Tujuan penarikan kesimpulan, agar penelitian ini dapat digunakan sebagai tolak ukur dan dapat dilanjutkan menjadi penelitian yang lebih baik serta tidak berhenti sampai kegiatan penulis selesai. Pengambilan saran bertujuan agar penelitian ini dapat dikembangkan menjadi penelitian yang lebih baik ke depannya.

BAB 4 ANALISIS KEBUTUHAN

4.1 Gambaran umum sistem

Gambaran umum pada sistem yang dibuat, *quadcopter* mampu bergerak sesuai dengan perintah suara yang dimasukkan oleh *user* melalui Android *device*. Adapun sistem yang digunakan berupa Android *device* yang berfungsi untuk masukan suara sekaligus tempat untuk *text preprocessing*, tahap ini dimulai dari *case folding*, langkah selanjutnya pemecahan kalimat menjadi kata, lalu setiap kata tersebut akan dicocokkan di dalam *database* dengan langkah *stopword removal* menggunakan algoritma *wordlist*. Jika kata sesuai dengan salah satu isi tabel yang ada, maka akan dikirim ke Arduino melalui komunikasi *bluetooth*. Arduino yang dilengkapi dengan modul *bluetooth* HC-05 sebagai *receiver* berfungsi untuk menerima data dari Android untuk dilakukan *text processing* untuk nantinya dikirim ke Node.JS melalui komunikasi serial. Node.JS sendiri berfungsi untuk memberikan perintah pergerakan *quadcopter* menggunakan *Node Package Manager* AR.*Drone* 2.0 melalui komunukasi *Wi-fi*.

4.2 Analisis kebutuhan sistem



Gambar 4.1 Analisis kebutuhan sistem

Pada bagian ini akan dibahas lebih lanjut mengenai kebutuhan-kebutuhan yang digunakan untuk sistem ini, seperti kebutuhan pengguna, kebutuhan *hardware*, kebutuhan *software*, dan yang terakhir kebutuhan komunikasi. Mengacu pada Gambar 4.1 semua komponen dalam analisis kebutuhan saling berkaitan satu sama lain, hal ini dikarenakan sistem ini membutuhkan semua komponen pada Gambar 4.1 agar berjalan sesuai dengan keinginan.

4.2.1 Kebutuhan pengguna

Kebutuhan pengguna menggambarkan hal-hal yang dilakukan oleh pengguna agar sistem ini berjalan sesuai dengan keinginan, di antaranya:

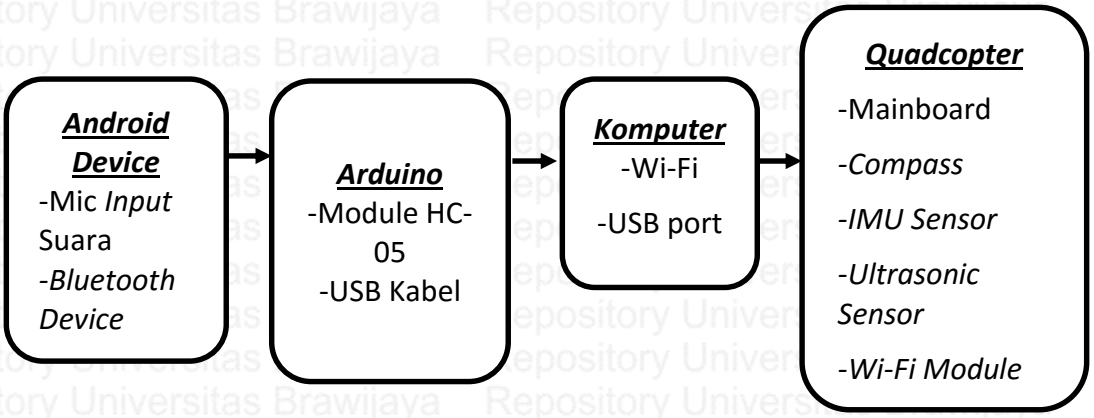
1. Pengguna membutuhkan aplikasi yang dibuat oleh peneliti dengan nama *SpeechCopter* sebagai masukan suara dan koneksi *bluetooth* untuk inisialisasi bahwa sistem dimulai.
2. Pengguna membutuhkan Node.JS sebagai tempat kumpulan Pergerakan *Quadcopter*.
3. Pengguna dapat melihat status komunikasi pada *console log* Node.JS .
4. Melalui *console* Node.JS, *user* mampu melihat arah pergerakan *Quadcopter*.

4.2.2 Kebutuhan sistem

Kebutuhan sistem meliputi apa saja yang dibutuhkan agar sistem dapat berjalan sesuai dengan keinginan peneliti. Pada tahap ini kebutuhan terdiri dari kebutuhan *hardware*, *software* dan komunikasi.

4.2.2.1 Kebutuhan *Hardware*

Berikut adalah kebutuhan *hardware* yang digunakan pada perancangan sistem yang dibuat, pada Gambar 4.2 dapat dilihat skema kebutuhan *hardware.*



Gambar 4.2 kebutuhan *Hardware*

Kebutuhan *hardware* Gambar 4.2 akan dijelaskan lebih lanjut,Yaitu:

1. Android *device*

   Smartphone atau bisa disebut Android *device* merupakan kebutuhan *hardware* yang berfungsi sebagai pengolahan serta masukan suara dari *user*. Android *device* yang digunakan harus memiliki *Bluetooth device*, tidak ada syarat minimal spesifikasi Android *device* yang digunakan. Namun dalam penelitian ini spesifikasi Android *device* dapat dilihat pada tabel Tabel 4.1

   Pada Tabel 4.1 Android *device* yang digunakan berjenis *Samsung Galaxy J5 API-22*, memiliki sistem operasi Android *5.0 Lollipop*, dengan *RAM* 1,5 GB dan *Internal storage*: 8 GB. *Chipset* pada *Samsung galaxy J5* digunakan *CPU Snapdragon 410 + GPU Adreno 306* yang setara dengan *quadcore CPU* dan support 32 atau 64 bit.

Tabel 4.1 Spesifikasi Android *Device* yang Digunakan

Nama	Spesifikasi
Merek	Samsung Galaxy J5
API	Api-22
Sistem Operasi	Android 5.0 Lollipop
Memori	RAM : 1.5 GB , Internal : 8 GB
Chipset	CPU Snapdragon 410 + GPU Adreno 306

Hardware yang dibutuhkan pada android *device* di antaranya:

a. *Mic* untuk masukan suara dari *user*
b. *Bluetooth device* digunakan untuk komunikasi dengan Arduino.

2. Arduino

Arduino merupakan *hardware* yang bersifat *open source* dan *software*nya mudah dikembangkan. Rician spesifikasi Arduino dapat dilihat pada Tabel 4.2.

Tabel 4.2 Spesifikasi Arduino

Komponen	Arduino Mega	Arduino Uno	Arduino Nano
Chip Mikrokontroller	ATmega2560	ATmega328P	ATmega328P
SRAM	8KB	2KB	2KB
EEPROM	4KB	1KB	1KB
Memori Flash	256 KB	32 KB	32 KB

pada penelitian ini menggunakan Arduino Mega karena memiliki spesifikasi lebih besar daripada jenis Arduino lainnya. *Hardware* yang dihubungkan dengan Arduino diantaranya:

a. *Module Bluetooth* HC-05 dihubungkan dengan Arduino berfungsi sebagai penerima data dari Android *device*.
b. *Kabel USB* berfungsi untuk menghubungkan Arduino dengan komputer sekaligus memberikan daya untuk menyalakan Arduino.

3. Komputer

Komputer sebagai tempat untuk pengembangan dan menjalankan program Node.js. spesifikasi komputer yang digunakan untuk penelitian dapat dilihat pada Tabel 4.3.

Tabel 4.3 Spesifikasi Komputer yang Digunkan

Nama	Spesifikasi
Sistem operasi	Windows 10 Pro-64 Bit
Ram	8Gb
Port USB	USB 2.0 atau USB 3.0
Prosesor	Intel Core i5-7200

Spesifikasi komputer yang digunakan menggunakan Sistem operasi windows 10-64 Bit dengan ram 8GB dan prosesor Intel Core i5-7200. *Usb Port* yang digunakan bisa menggunakan USB 2.0 maupun USB 3.0. *Hardware* yang digunakan pada komputer sebagai berikut:

a. *Hardware* yang terdapat pada Komputer berupa *Wi-fi device* yang digunakan untuk menghubungkan Komputer dengan AR. *Drone* 2.0 *Quadcopter*.
b. *USB Port* digunakan untuk menghubungkan Arduino dengan Node.Js menggunakan kabel USB.

4. *Quadcopter*

Quadcopter yang digunakan pada penelitian ini adalah parrot AR *Drone* 2.0. Spesifikasi *quadcopter* yang digunakan dapat dilihat pada Tabel 4.4

Tabel 4.4 Spesifikasi *Quadcopter*

Nama	Spesifikasi
Prosesor	1GHz 32 bit ARM Cortex A8 prosessor dengan 800 MHz
Sistem Operasi	Linux 2.6.32
RAM	1GB DDR2 RAM Berkecepatan 200 MHz
Konektivitas	Wi-Fi b, g, n
Gyroscope	3 axis *gyroscope* 2000^{o} / seconds precision
Accelerometer	3 axis *Accelerometer* $\pm$ 50mg precision
Magnetometer	3 axis *magnetometer* 6^{o} precision
Ultrasonik Sensor	Ultrasonik sensor untuk penghitungan darat
Kamera	60 FPS vertikal QVGA kamera untuk pengukuran kecepatan darat

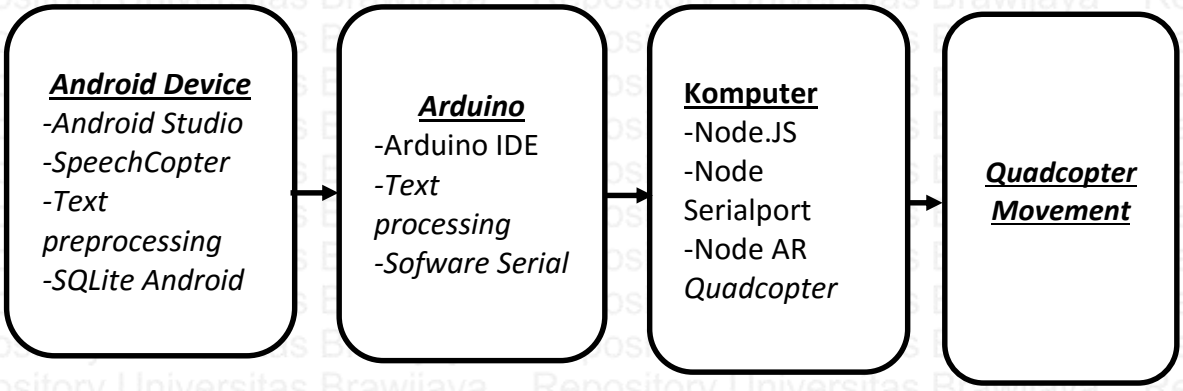
Parrot AR Drone pada *motherboardnya* memiliki prosessor 1GHz 32 bit ARM *Cortex* A8 prosessor dengan 800 MHz video DSP sebesar 1GB dengan sistem operasi *linux* sebagai sistem operasinya, serta memiliki *RAM*

sebesar 1GB. Parrot AR Drone ini dilengkapi dengan modul *Wi-Fi* dan modul kamera. Kebutuhan *hardware* yang digunakan pada *quadcopter* sebagai berikut:

a. *Mainboard* tempat pusat kendali semua komponen yang ada pada *quadcopter*, dilengkapi dengan sistem operasi *linux busyBox* v.1.14.0 yang dapat diakses secara *ad-hoc* melalui komputer.
b. Compass merupakan penunjuk arah atau navigasi sistem pada *quadcopter*.
c. IMU sensor pada *quadcopter* sebagai penghitung nilai *roll*, *pitch*, dan *yaw*. Fungsinya sebagai penyeimbang kestabilan saat terbang.
d. sensor *Ultrasonic* berfungsi untuk mengukur ketinggian dari permukaan saat terbang.
e. *Wi-fi Module* berfungsi sebagai media komunikasi antara *quadcopter* dengan platform lain.

4.2.2.2 Kebutuhan *software*

Pada bagian ini akan dijelaskan *software* apa saja yang dibutuhkan untuk melakukan penelitian, dapat dilihat pada Gambar 4.3 skema kebutuhan *software*.



Gambar 4.3 Kebutuhan *Software*

Kebutuhan *Software* Gambar 4.3 akan dijelaskan lebih lanjut, yaitu:

1. Android *device*

   Android memiliki peran penting dalam penelitian ini, pada membutuhkan beberapa *software* untuk pengembangannya di antaranya:

   a. Android Studio Merupakan platform untuk mengembangkan aplikasi berbasis mobile dengan sistem operasi Android. Android studio merupakan *software* keluaran Google yang diperkenalkan pada tahun 2013. Pada penelitian ini Android studio berfungsi sebagai tempat pembuatan aplikasi SpeechCopter.

b. SpeechCopter adalah aplikasi karya peneliti yang digunakan untuk masukan perintah suara dari *user*, nantinya hasil dari masukan suara akan diolah lebih lanjut di aplikasi ini.

c. *Text preprocessing* Merupakan tahap di mana hasil dari masukan suara yang berupa kalimat dengan tipe data *String* diubah menjadi huruf besar lalu dicacah menjadi potongan kata. Lalu kata tersebut akan dicocokkan dengan kumpulan kata yang ada pada *database* dengan algoritma *stopword removal wordlist*. Jika kata sesuai dengan apa yang ada pada *database*, maka tabel kolom kedua yaitu tabel *command* sebagai perintah kontrol akan dikirim ke Arduino.

d. SQLite merupakan *database* yang terdapat pada Android, dapat diolah dengan mudah melalui Android studio dengan *query* dasar *database* seperti *Insert*, *Update*, *Delete*, *Select*.

2. Arduino

Dalam *Arduino* ada beberapa kebutuhan software dan penggunaan *library* di antaranya:

a. Arduino IDE merupakan platform untuk mengembangkan kode untuk Arduino selain itu, meng-upload code algoritma yang kita inginkan ke dalam Arduino.

b. *Text processing* Merupakan tahap dimana data yang diterima dari Android diolah oleh *Arduino*, tiap-tiap kata masukan yang berbeda akan menghasilkan keluaran berupa pergerakan *quadcopter* yang berbeda beda pula.

c. *Software* Serial merupakan *library* yang tersedia pada Arduino. *Library* ini digunakan untuk menghubungkan *Bluetooth* Modul HC-05 dengan Arduino.

3. Komputer

Ada beberapa *software* yang digunakan menggunakan komputer di antaranya:

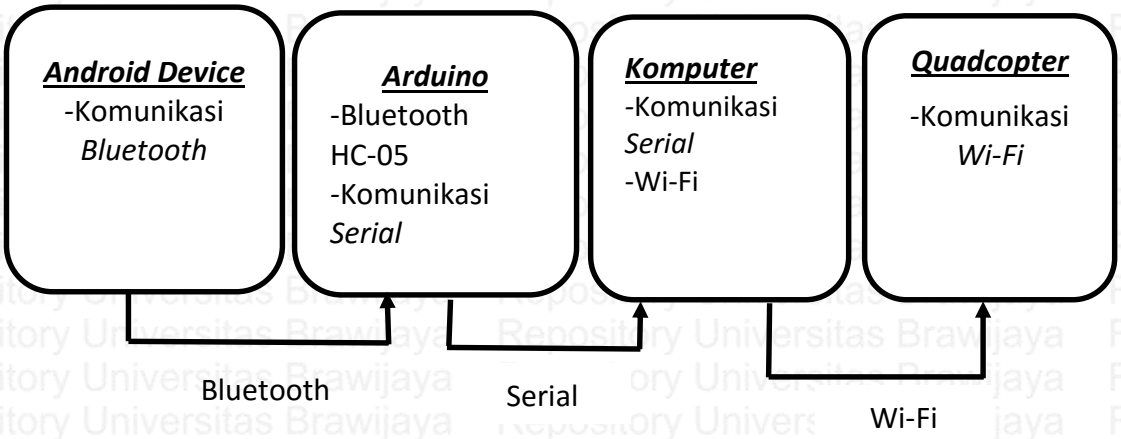
a. Node.JS merupakan platform untuk memberikan perintah pada *quadcopter*, platform ini membantu para pengembang untuk membangun suatu sistem dengan menggunakan bahasa pemrograman *JavaScript*. *User* memberikan inisialisasi bahwa program dimulai melalui Node.JS *Command prompt*

b. Node SerialPort berguna untuk menjembatani Arduino dengan Node.JS. Memanfaatkan komunikasi serial antar keduanya sehingga bisa saling mengirimkan data.

c. Node AR-Quadcopter berguna untuk menghubungkan perintah suara yang telah diolah oleh Arduino dan Android menjadi gerakan *quadcopter*.

4.2.2.3 Kebutuhan komunikasi

Pada bagian ini akan dijelaskan Komunikasi apa saja yang dibutuhkan untuk melakukan penelitian ini, skema kebutuhan *software* dapat dilihat pada Gambar 4.4.



Gambar 4.4 Kebutuhan Komunikasi

Kebutuhan Komunikasi Gambar 4.4 akan dijelaskan lebih lanjut, yaitu:

1. Untuk menghubungkan Android *device* dengan Arduino dibutuhkan komunikasi:
    a. Komunikasi yang digunakan pada Android dan Arduino melalui *bluetooth*. Data hasil pengolahan *Text preprocessing* dikirim ke Arduino. *Bluetooth* status pada Android sebagai Master, sedangkan pada Arduino sebagai *slave*, sehingga kedua *device* dapat saling terhubung satu sama lain.
2. Di dalam *Arduino* membutuhkan 2 kebutuhan komunikasi yaitu*:*
    a. Pada Arduino digunakan HC-05 sebagai modul untuk menjalin komunikasi *Bluetooth* dengan Android *device*, status *Bluetooth* pada Modul ini berupa *Slave*. Arduino sendiri berfungsi untuk menerima hasil *Text preprocessing* untuk nantinya dilakukan *text processing* melalui komunikasi *Bluetooth*.
    b. Penggunaan Serial komunikasi digunakan untuk menghubungkan Arduino dengan Node.JS dan *Bluetooth module* HC-05 dengan memanfaatkan *Library Software*Serial. Arduino dihubungkan menggunakan kabel USB ke komputer, setelah melakukan *Text processing* maka hasilnya akan dikirim ke Node.JS melalui komunikasi Serial.
3. Komputer membutuhkan beberapa komunikasi agar dapat menerima data dari Arduino dan mengirim data ke *quadcopter* di antaranya:

d. Komunikasi yang terjalin antara Arduino dengan Komputer melalui kabel USB, akan dihubungkan dengan Node.JS dengan memanfaatkan *Node SerialPort*. komunikasi serial antar Arduino dan Node.js dapat terjalin melalui *Port COM* yang ditentukan sebelumnya , sehingga antar keduanya saling mengirimkan data.
e. Komunikasi *Wi-fi* digunakan untuk menghubungkan komputer dengan *Quadcopter*. Melalui Node.JS, *quadcopter* akan diperintah berdasarkan hasil pengolahan *text processing*. Pertukaran data antara Node.JS dan *quadcopter* Melalui Komunikasi *Wi-fi* Ad-Hoc. Data yang dikirim dari quadcopter menuju node.js berupa data arah terbang.

4.3 Kebutuhan fungsional

Kebutuhan Fungsional digunakan untuk memberikan rincian kebutuhan-kebutuhan pada penelitian ini agar sesuai dengan harapan peneliti di antaranya:

a. *Quadcopter* dapat bergerak sesuai perintah *user* yang berbeda-beda. Pada fungsi ini sistem diharuskan memiliki keluaran yang sama walaupun memiliki masukan suara dari *user* yang berbeda-beda.

b. *Database* yang digunakan dapat mencari kata secara akurat dan terstruktur. Sistem *database* yang digunakan dalam penelitian ini terdapat pada Android Studio yaitu SQLite, yang mana sistem *database* ini dapat digunakan menggunakan query dasar *database*. Sistem *database* diharuskan mencari kata masukan dari *user* secara akurat, jika masukan sesuai dengan salah satu tabel maka akan dikirim tabel kedua yaitu tabel *command*, jika tidak maka akan dikirim angka 0. Selain akurat, sistem ini harus terstruktur dan sesuai dengan masukan dari *user* sehingga kalimat perintah mampu dieksekusi secara sekuensial.

c. Sistem berjalan dengan lancar melalui koneksi *Bluetooth* dan *Wi-fi*. Pada fungsi ini, penggunaan komunikasi menggunakan *Bluetooth* dan *Wi-fi*, yang mana *Bluetooth* untuk menghubungkan Android *device* dengan Arduino, sedangkan *Wi-fi* untuk menghubungkan komputer dengan *quadcopter*. Jarak maksimal *Bluetooth* 8 meter jika digunakan di dalam ruangan tanpa hambatan, sedangkan *Wi-fi* mampu berjarak sampai 45 meter.

d. *Quadcopter* memiliki data navigasi sesuai dengan hasil yang diharapkan. Sistem ini mampu menghasilkan sistem navigasi *quadcopter* melalui Node.JS untuk nantinya digunakan untuk analisis pergerakan *quadcopter*. Data yang diambil berupa perubahan nilai *roll*, *pitch*, dan *yaw* dengan memanfaatkan *Plot.ly* untuk membuat grafik data secara *real-time*.

4.4 Kebutuhan non fungsional

Kebutuhan *non* fungsional merupakan kebutuhan yang menjelaskan mengenai apa saja yang menjadi batasan terhadap kebutuhan perancangan sistem. Adapun kebutuhan *non* fungsional dari sistem ini adalah sebagai berikut.

4.4.1 Karakteristik yang berbeda tiap individu

Sistem ini dibuat untuk *user* secara umum dengan harapan akan mempermudah penggunaan *quadcopter* dalam berbagai bidang khususnya kontrol dan bidang pengolahan suara. Dalam sistem ini *user* yang berbeda mampu mengkontrol *quadcopter* tanpa membutuhkan keahlian khusus apapun.

4.4.2 Lingkungan pengoperasian *quadcopter*

Pada dasarnya lingkungan pengoperasian sistem ini dapat dilakukan pada lingkungan manapun baik di dalam ruangan maupun luar ruangan, namun untuk pengujian digunakan ruangan tertutup agar menghasilkan nilai navigasi data yang baik.

4.4.3 Asumsi ketergantungan tiap komponen

Tiap komponen yang digunakan memiliki ketergantungan satu sama lain agar memiliki hasil yang sesuai dengan harapan.

a. Aplikasi Android membutuhkan koneksi *Bluetooth* yang dihubungkan dengan Arduino sebagai inisialisasi bahwa program dimulai. Di dalamnya terdapat button ”ngomong” untuk memberikan masukan suara.
b. Menggunakan *module Bluetooth* HC-05 yang mana Pin TX dihubungkan dengan pin 10 pada Arduino, Pin RX dihubungkan dengan pin 11 pada Arduino, Pin *VCC* dan *Ground* dihubungkan ke Pin 5V dan *Ground* pada Arduino.
c. Arduino membutuhkan kabel USB yang terhubung dengan komputer untuk berkomunikasi secara serial dengan Node.js. Data yang dikirim ke Node.JS Adalah data hasil olahan kata.
d. Di dalam Node.JS terdapat NPM *serial port* yang digunakan untuk berkomunikasi secara Serial dengan Arduino, dan juga terdapat NPM Ar.*Drone* 2.0 yang berisi kumpulan perintah untuk kontrol *quacopter.*

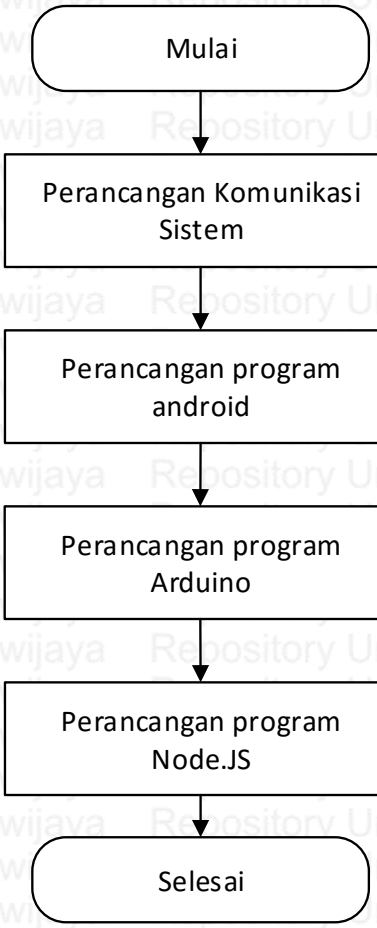
4.4.4 Pengambilan data tiap gerakan *quadcopter*

Pengumpulan data dilakukan dengan menggunakan *user* yang berbeda, apakah *database* memiliki akurasi yang baik atau tidak. Pengumpulan data yang kedua berupa data pergerakan *quadcopter* dengan mengambil data *ROLL*, *PITCH*, dan *YAW* yang ada pada data navigasi. Tiap pergerakan yang berbeda akan menghasilkan nilai *ROLL*, *PITCH*, dan *YAW* yang berbeda pula. Nilai tersebut dijadikan grafik baik secara *real time* ataupun *non real-time*, menggunakan *platform* Plot.Ly untuk mengambil data secara *real-time*, dan menggunakan Microsoft excel untuk data secara *non real-time*.

BAB 5 PERANCANGAN DAN IMPLEMENTASI SISTEM

5.1 Perancangan sistem

Perancangan sistem dibagi menjadi empat bagan, yaitu perancangan komunikasi sistem, perancangan program Android *device*, perancangan program Arduino, dan perancangan program kontrol *quadcopter* pada Node.JS. Alur perancangan dapat dilihat pada Gambar 5.1.

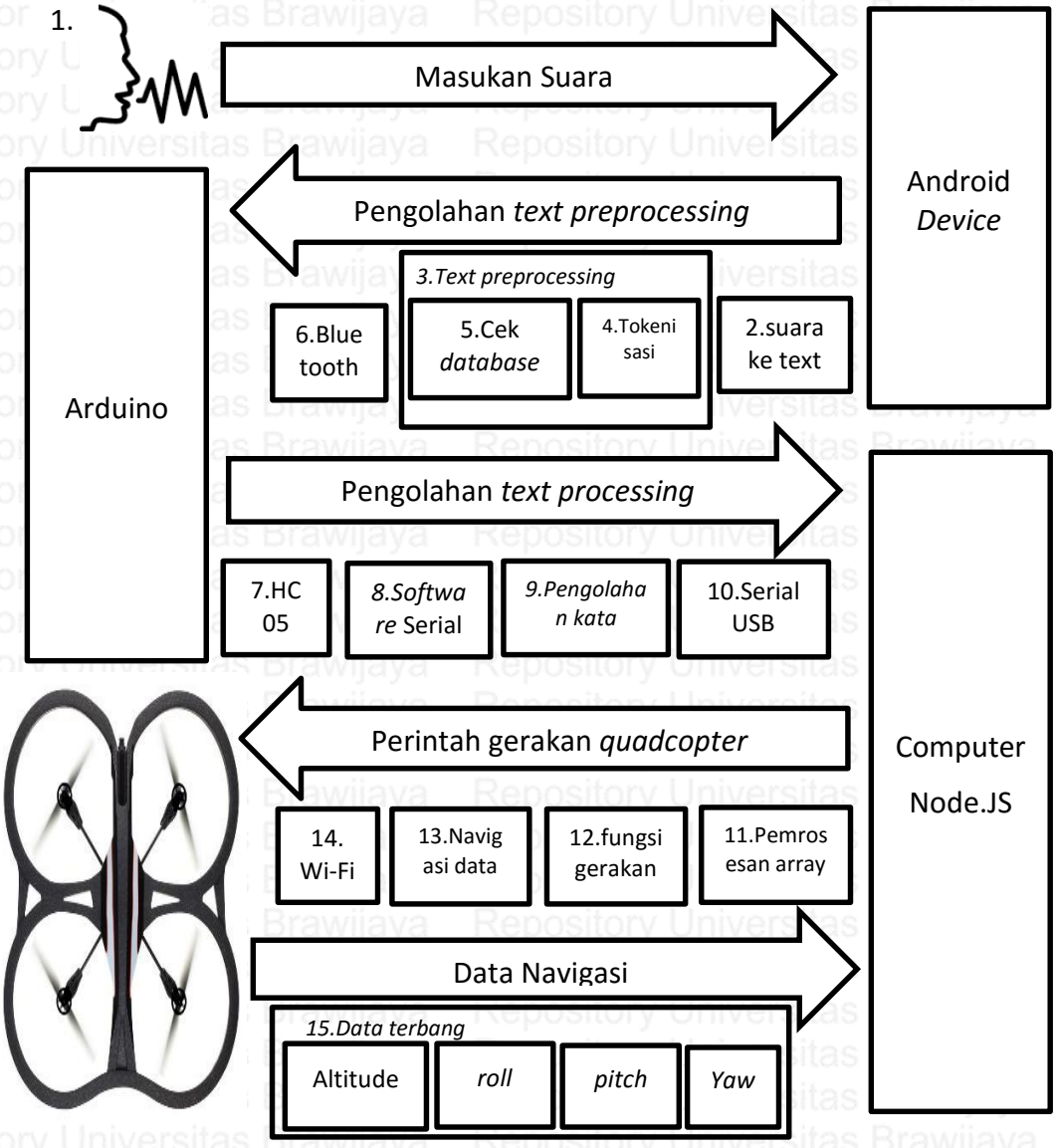


Gambar 5.1 Alur Perancangan Sistem

Pada Gambar 5.1 menjelaskan alur perancangan sistem, penelitian diawali dengan perancangan alur komunikasi keseluruhan sistem, cara kerja setiap komponen yang digunakan. Selanjutnya merancang program Android yang digunakan, nantinya program tersebut sebagai inisialisasi awal dimulainya sistem. Setelah itu merancang program pada Arduino, pada tahap ini Arduino berfungsi sebagai tempat pengolahan data untuk diklasifikasikan menjadi gerakan *quadcopter*. Yang terakhir, perancangan program pada komputer menggunakan Node.JS menggunakan Bahasa pemrograman *javaScript*, pada tahap ini program dibuat untuk kontrol *quadcopter.*

5.1.1 Perancangan komunikasi sistem

Perancangan komunikasi sistem merupakan penjelasan bagaimana setiap komponen berkomunikasi. Alur komunikasi program yang digunakan pada penelitian dapat dilihat pada Gambar 5.2.

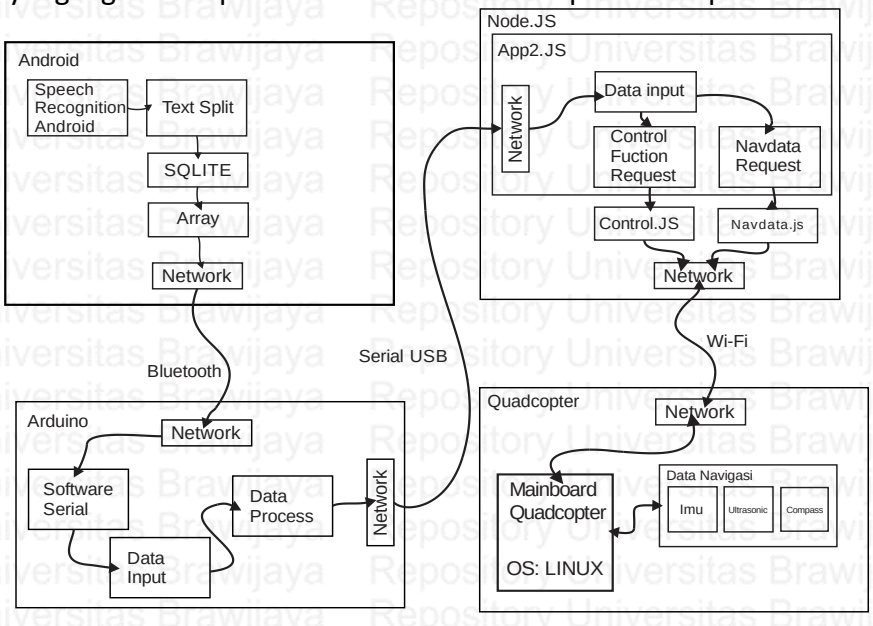


Gambar 5.2 Perancangan Komunikasi Program

1. Sistem dimulai ketika *user* memberikan masukan suara.
2. Suara diubah ke teks memanfaatkan *speech recognition* pada Android. Hasil dari pengolahan suara akan menghasilkan data dengan tipe *string*.
3. Lalu dilakukan proses *Text preprocessing*, yang terdiri dari tokenisasi dan pengecekan *database*.
4. Pada proses tokenisasi terdapat proses *case folding*, lalu pemecahan kalimat menjadi kata.

5. selanjutnya tiap kata dicek di dalam *database* dengan algoritma *stopword removal wordlist*.
6. Data hasil pengolahan *text preprocessing* dikirim ke Arduino melalui komunikasi *bluetooth*.
7. Arduino membutuhkan HC-05 untuk komunikasi *bluetooth*.
8. Arduino membutuhkan *software serial* untuk inisialisasi *bluetooth module* HC-05
9. Data yang diterima dari Android akan di lakukan pengolahan kata untuk menjadi keluaran yang diinginkan
10. Hasil pengolahan kata dari arduino dikirim ke Node.JS melalui serial komunikasi USB.
11. Node.js menerima data dari Arduino dan dimasukkan ke dalam *array*, untuk diolah menjadi gerakan *quadcopter*.
12. Node.js memberikan perintah untuk gerakan *quadcopter* dengan memanggil fungsi gerakan *quadcopter*.
13. Selain memberikan perintah kontrol, Node.JS akan meminta untuk menampilkan data navigasi dari *quadcopter* dari pengolahan sensor yang dimilikinya.
14. Komunikasi yang digunakan Antara Node.JS dan *quadcopter* menggunakan *Wi-fi*.
15. *Quadcopter* akan mengirim *feedback* data terbang atau data *navigasi* nilai *Roll*, *Pitch*, *Yaw*, dan *altitude* untuk nantinya ditampilkan di *console log*.

Alur yang digunakan pada komunikasi sistem dapat dilihat pada Gambar 5.3.

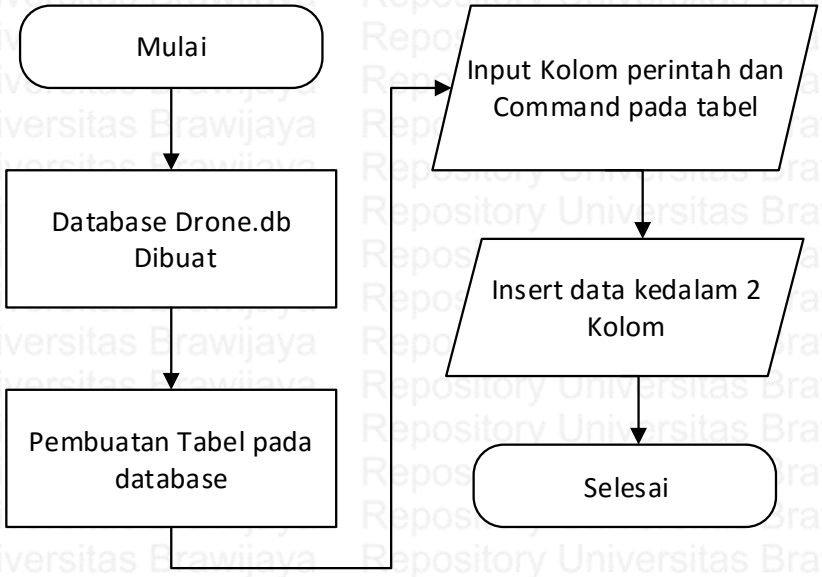


Gambar 5.3 Alur Komunikasi Sistem

Alur komunikasi sistem pada Gambar 5.3 dimulai dengan inisialisasi awal sistem pada Android, memanfaatkan *speech recognition* pada Android *device* sebagai masukan dari *user*, lalu keluaran dari *speech recognition* yang berupa kalimat akan dipecah menjadi satuan kata. Tiap kata akan dicocokkan pada *database* yang telah dibuat sebelumnya dengan memanfaatkan SQLite. Hasil dari pencocokan akan dimasukkan ke dalam *array* lalu dikirim ke Arduino. Pada Arduino memanfaatkan *software* serial untuk menghubungkan *Bluetooth module*, sebagai komunikasi dengan Android. Data yang masuk dari Android akan di proses menjadi keluaran yang diinginkan lalu dikirim ke Node.JS melalui komunikasi serial. Aplikasi yang dibuat pada Node.js diberi nama App2.Js, aplikasi ini terintegrasi dengan Navdata.JS dan Control.JS. data yang masuk dari Arduino akan diolah menjadi *request*, yang mana *request* tersebut terintegrasi dengan *parrot AR drone 2.0 quadcopter*. Pada *quadcopter* terdapat *mainboard* sebagai tempat kontrol semua kompenen yang terhubung, Sensor yang dibutuhkan diataranya *Compass* sensor, yang berfungsi untuk Navigasi Arah terbang *Quadcopter*. *Imu sensor*, yang terdiri dari *Akselerometer* dan *Gyroscope* berfungsi untuk mengambil nilai sudut *ROLL*, *PITCH*, *YAW* dari *quadcopter*. Sensor *ultrasonic* yang berfungsi untuk melihat ketinggian saat *quadcopter* terbang.

5.1.2 Perancangan sistem pada Android

Android *device* berfungsi untuk masukan suara dan melakukan *Text preprocessing*, menggunakan Android studio IDE sebagai tempat untuk pembuatan dan pengembangan *software* pengolahan suara. *Database* dibuat dengan menggunakan algoritma *stopword removal wordlist* yaitu *database* yang berisi kumpulan kata penting, dengan memanfaatkan SQLite yang tersedia di dalam Android Studio. Alur perancangan *database* dapat dilihat pada Gambar 5.4.



Gambar 5.4 Perancangan *database*

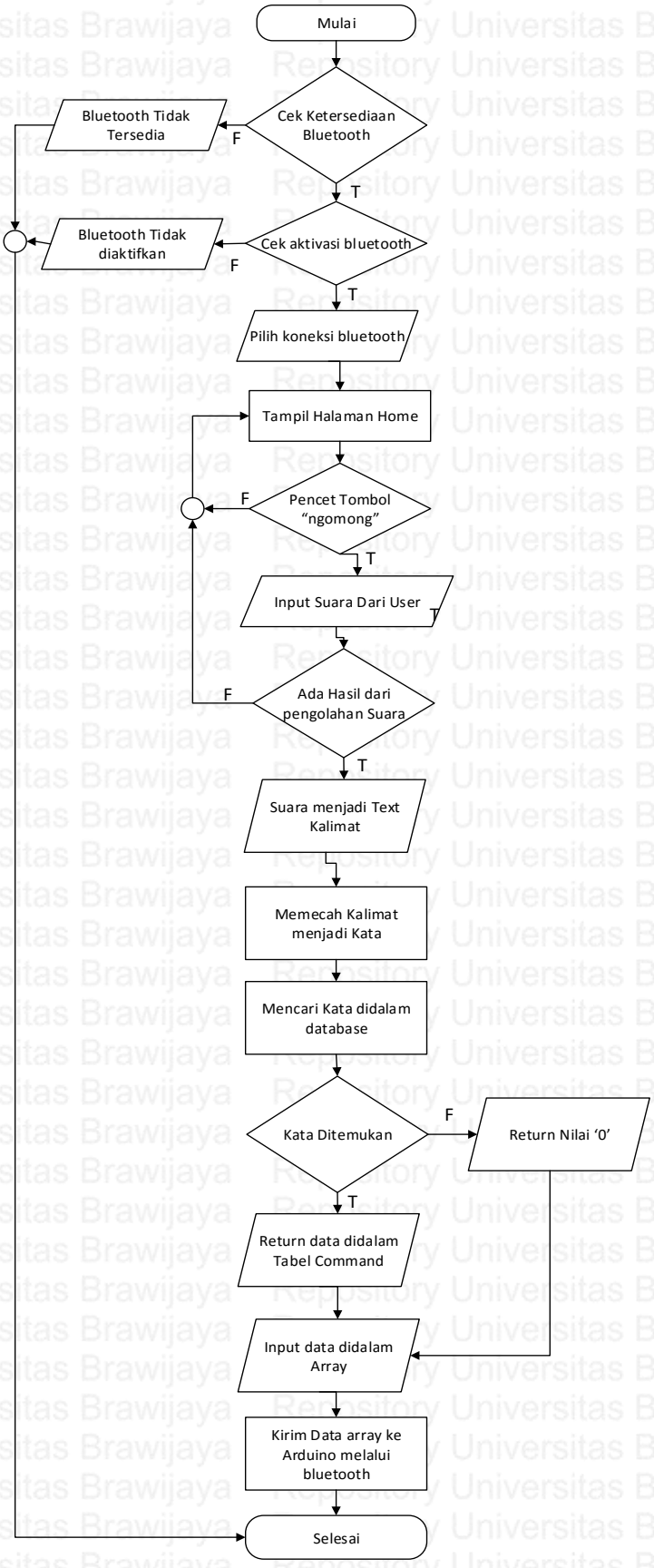
Pada Gambar 5.4 merupakan alur perancangan *database*. Awal mula instalasi, program dimulai dengan pembuatan *database* dengan nama *Drone*.db.

tahap kedua yaitu pembuatan tabel dengan nama drone_*command*. Tabel ini memiliki 2 kolom yaitu kolom Name dan kolom Command, nantinya 2 kolom ini akan diisi dengan data kumpulan kata dan perintah. Kolom pada tabel diisi dengan perintah pergerakan *quadcopter* ,kata hubung, kata imbuhan, dan lain-lain. Tabel 5.1 merupakan kata penting di dalam *database* yang digunakan sebagai perintah untuk kontrol dasar *quadcopter*.

Tabel 5.1 Kata Inti pada *Database Drone.db* SQLite

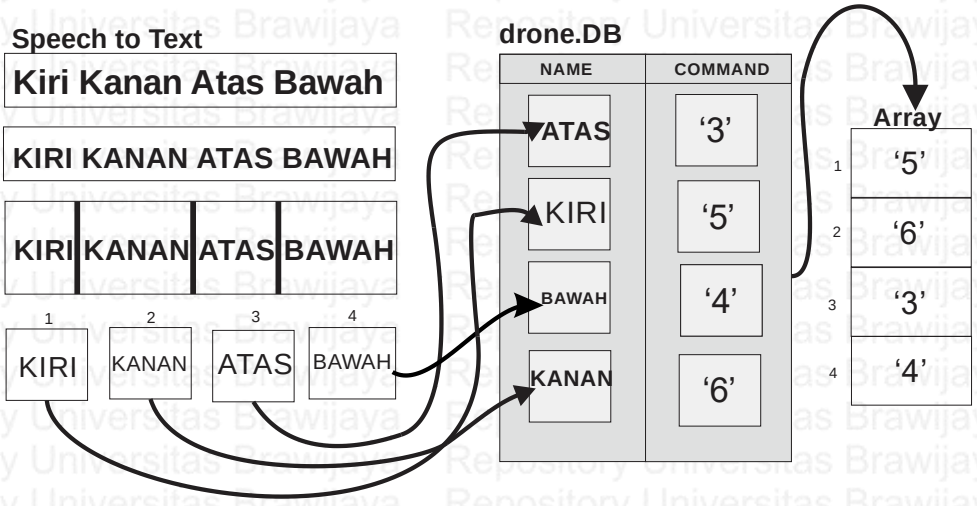
Name	*Command*
"TERBANG"	'1'
"TAKE"	'1'
"*LANDING*"	'2'
"MENDARAT"	'2'
"NAIK"	'3'
"ATAS"	'3'
"TURUN"	'4'
"BAWAH"	'4'
"KIRI"	'5'
"KANAN"	'6'
"MAJU"	'7'
"DEPAN"	'7'
"MUNDUR"	'8'
"BELAKANG"	'8'
"BERPUTAR"	'9'
"PUTAR"	'9'
"LALU"	'Z'
"LANJUT"	'Z'
"SETELAH"	'Z'
"DAN"	'Z'
"BOLEH"	'Z'
"KEMUDIAN"	'Z'
"SELANJUTNYA"	'Z'
"SEKALIAN"	'Z'

Tabel 5.1 merupakan kumpulan kata utama yang digunakan untuk kontrol *quadcopter*, kolom Name berisi kumpulan kata inti untuk kontrol *quadcopter* kolom ini berguna sebagai tempat mengecek apakah data pengolahan suara terdapat pada kolom ini. Kolom Command merupakan data yang nantinya dikirim ke Arduino. Hubungan Antara 2 kolom pada Tabel 5.1 berdasarkan pada 11 gerakan dasar *quadcopter*. Jika kata berupa gerakan dasar maka pada kolom Command diberikan nilai berupa angka agar mudah memprosesnya. Namun jika kata berupa kata imbuhan, maka diberikan nilai berupa huruf untuk membedakan kata gerakan dan kata imbuhan. Perancangan program pada Android dapat dilihat pada Gambar 5.5.



Gambar 5.5 *Flowchart* alur program pada Android

Pada Gambar 5.5 inisialisasi awal program dengan mengecek apakah *smartphone* atau android *device* yang digunakan memiliki *Bluetooth device* atau tidak, jika tidak maka aplikasi akan menutup dengan sendirinya. Langkah berikutnya aplikasi ini akan meminta untuk mengaktifkan *Bluetooth* untuk nantinya ditampilkan *device* yang tersedia. Agar dapat terhubung dengan Arduino, *user* harus memilih *bluetooth* yang terhubung dengan Arduino umumnya dengan nama *Bluetooth* HC-05 yang secara *default* sudah di setting dengan mode *slave*. Setelah komunikasi *Bluetooth* terhubung satu sama lain maka akan tampil halaman *home*, pada halaman ini terdapat tombol untuk memberikan perintah suara. Jika tombol *speech* ditekan maka *user* bisa memberikan masukan suara untuk nantinya dijadikan perintah kontrol *quadcopter*. Jika *user* tidak memberikan perintah suara apapun, maka program akan kembali ke halaman *home*. Setelah masukan suara dari *user* didapat, akan dilakukan *Text preprocessing*. langkah pada tahap ini diawali dengan mengubah dengan kalimat masukan dari *user* menjadi huruf besar, tahap ini dinamakan *case folding*, langkah selanjutnya memecah kalimat menjadi kata, lalu tiap kata dicari dan di cocokan dengan kolom Name yang ada pada *Drone.db*, jika kata yang dicari sesuai dan cocok dengan isi kolom Name, maka isi dari kolom kedua yaitu kolom *Command* akan dikembalikan dan dimasukkan ke dalam *array*. Namun Jika kata tidak ditemukan di dalam *database* akan dikembalikan nilai '0', dan dimasukkan ke dalam *array*. Kumpulan *array* tersebut dikirim ke Arduino melalui komunikasi *Bluetooth*. Diagram alur pengolahan teks dapat dilihat pada Gambar 5.6



Gambar 5.6 Alur Pengolahan Teks

Pada Gambar 5.6 kalimat "Kiri Kanan Atas Bawah" akan diubah menjadi huruf besar terlebih dahulu lalu dipisah menjadi satuan kata "Kiri", "Kanan", "Atas", "Bawah",tiap kata akan dicocokkan dengan *database*, kalimat "Kiri" terdapat pada kolom Kiri dan *command* '5' maka isi dari tabel *COMMAND* akan dimasukkan ke dalam *array*, kalimat "Kanan" terdapat pada kolom Kanan dan *command* '6' maka isi dari tabel *command* akan dimasukkan ke dalam *array* ke dua, kalimat "Atas" terdapat pada kolom Atas dan *command* '6' maka isi dari tabel *command* akan dimasukkan ke dalam *array* ke tiga, kalimat "bawah" terdapat

pada kolom Bawah dan *command* '4' maka isi dari tabel *command* akan dimasukkan ke dalam *array* ke empat. Isi dari data *array* akan dikirim ke Arduino secara sekuensial.

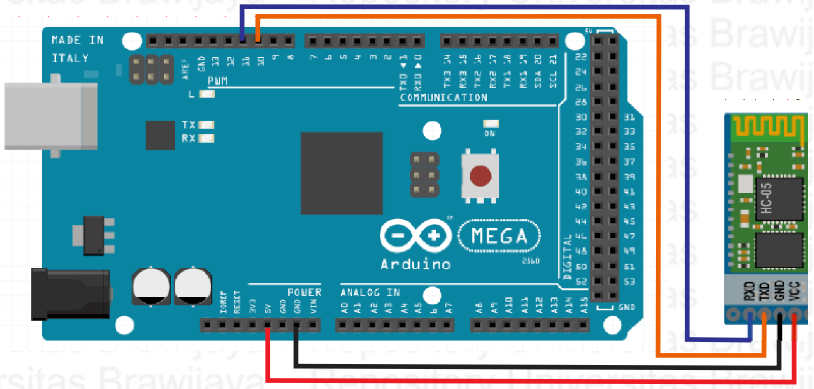
5.1.3 Perancangan sistem pada Arduino

Perancangan sistem pada Arduino meliputi dua hal, yaitu perancangan *hardware* dan perancangan program. Pada perancangan *hardware* sistem ini membutuhkan *module Bluetooth* HC-05, pin perancangan dapat dilihat pada Tabel 5.2

Tabel 5.2 Pin pada *Bluetooth Module* HC-05 ke Arduino

Bluetooth Module HC-05	ARDUINO Uno
VCC	5 V
GND	GND
Tx	Pin Digital 10
Rx	Pin Digital 11

Tabel 5.2 merupakan konfigurasi *Module Bluetooth* HC-05 dengan Arduino, terdapat 4 pin yang digunakan pada HC-05 , yaitu pin VCC dihubungkan dengan 5V pada Arduino, pin GND yang dihubungkan dengan *Ground* pada Arduino, Pin *Tx* dihubungkan dengan pin 10 pada Arduino, dan pin *Rx* dihubungkan dengan pin 11 pada Arduino. Skematik perancangan dapat dilihat pada Gambar 5.7.

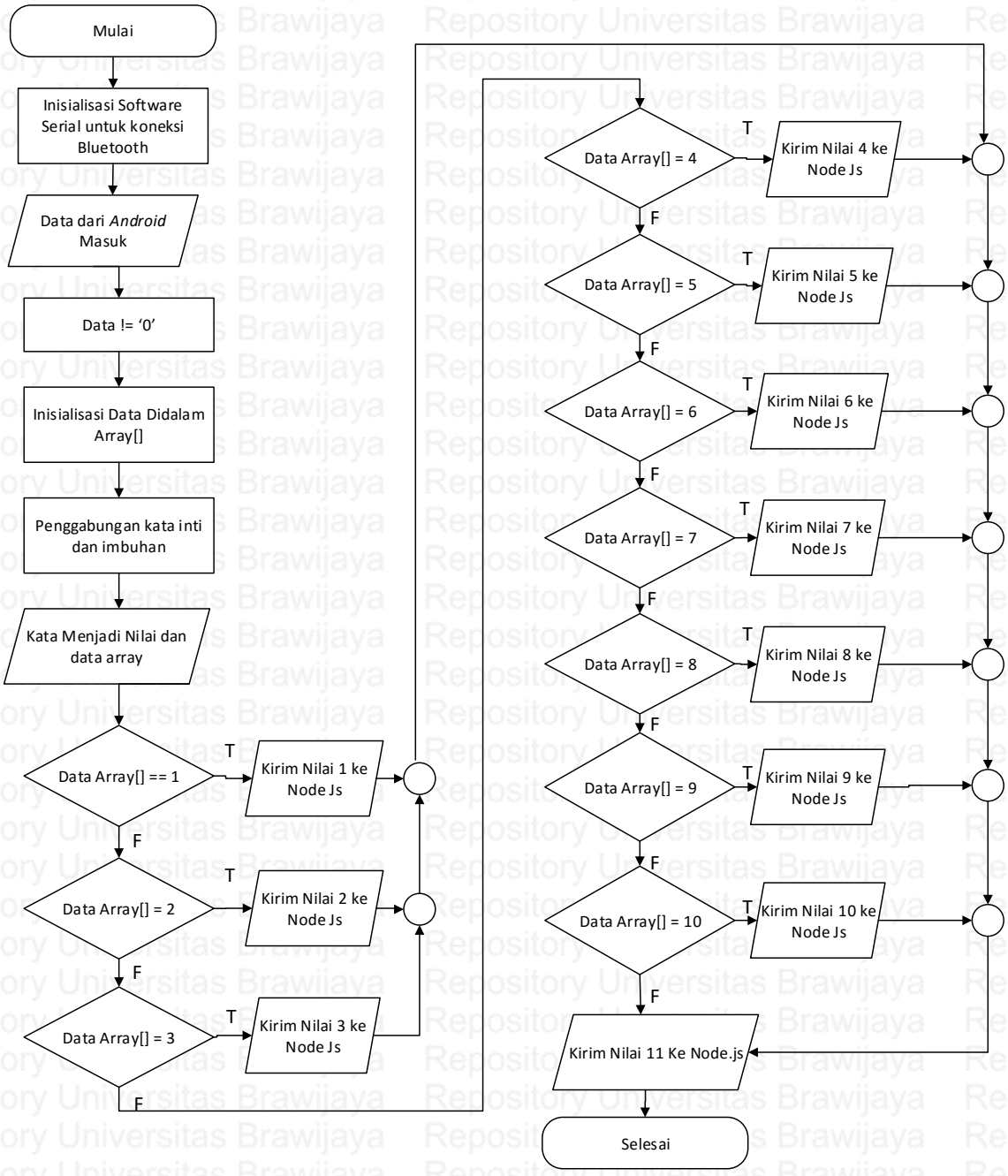


Gambar 5.7 Skematik perancangan *module Bluetooth* HC-05

Untuk menghubungkan Arduino dengan *module Bluetooth* HC-05 menggunakan komunikasi serial, dengan memanfaatkan *library software* serial sebagai inisialisasi komunikasinya. Pada Arduino mega, pin nomor 10 diinisialisasi sebagai pin *transmitter* (*Tx*), sedangkan pada pin 11 diinisialisasi sebagai pin *receiver* (*Rx*). *Module Bluetooth* HC-05 terdapat *transmitter* dan *receiver*, yang mana *transmitter* pada HC-05 disambungkan ke *receiver* pada Arduino Mega, sedangkan *transmitter* pada HC-05 disambungkan ke Arduino Mega.

Selain perancangan *hardware*, pada Arduino membutuhkan Perancangan program. Data yang masuk dari Android akan diolah menjadi kumpulan kumpulan kata. Pada tahap ini kata yang tidak ada di dalam *database* akan dibuang, sedangkan kata yang ada di dalam *database* akan disimpan untuk nantinya diolah

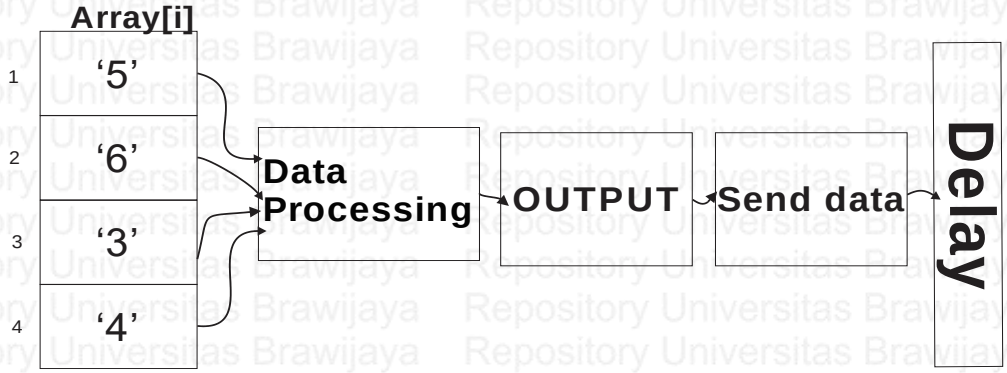
menjadi nilai. Alur perancangan program pada Arduino dapat dilihat pada Gambar 5.8.



Gambar 5.8 *Flowchart* alur program pada Arduino

Pada Gambar 5.8 program diawali dengan memanggil *library software serial*, *library* ini berfungsi sebagai inisialisasi bahwa Arduino terhubung dengan *module Bluetooth* HC-05 melalui komunikasi serial. Data hasil *Text preprocessing* dari Android masuk berupa karakter, selama data yang masuk dari Android tidak sama dengan karakter '0' maka data tersebut akan masuk ke dalam *array*. Dari sekumpulan data di dalam *array* akan dilakukan *text processing*, yaitu

penggabungan data di dalam *array*, data yang digabung berupa kalimat imbuhan dan kalimat inti. Jika ada kata di dalam *array* berbentuk kata imbuhan maka kata tersebut akan digabungkan dengan indeks *array* sebelumnya atau setelahnya. Setelah melakukan penggabungan maka akan dihasilkan nilai berupa karakter angka dengan nilai 1 sampai 11 yang nantinya diubah menjadi *integer*, nilai *integer* tersebut langsung dikirim ke Node.JS melalui komunikasi serial. Alur perancangan program dapat dilihat pada Gambar 5.9.



Gambar 5.9 Alur perancangan program Arduino

Data yang masuk dari Android akan dimasukkan ke dalam *array*, *aray* tersebut akan diolah dengan digabungkan atau berdiri sendiri, proses ini dinamakan *text processing*. Keluaran dari *text processing* berupa nilai *integer*, yang nantinya akan dikirim ke Node.JS dan diberikan *delay* pada tiap tahapnya.

5.1.4 Perancangan Sistem pada komputer Node.JS

Perancangan sistem pada Node.JS berfungsi sebagai tempat untuk kontrol pergerakan *quadcopter*. Arduino mengirim data berupa *integer* ke Node.JS untuk diklasifikasikan menjadi data *array*, Data *array* tersebut akan di inisialisasi menjadi gerakan *quadcopter*. Pengelompokan data yang dapat dilihat pada Tabel 5.3

Tabel 5.3 Tabel perintah gerakan *quadcopter* pada Node.JS

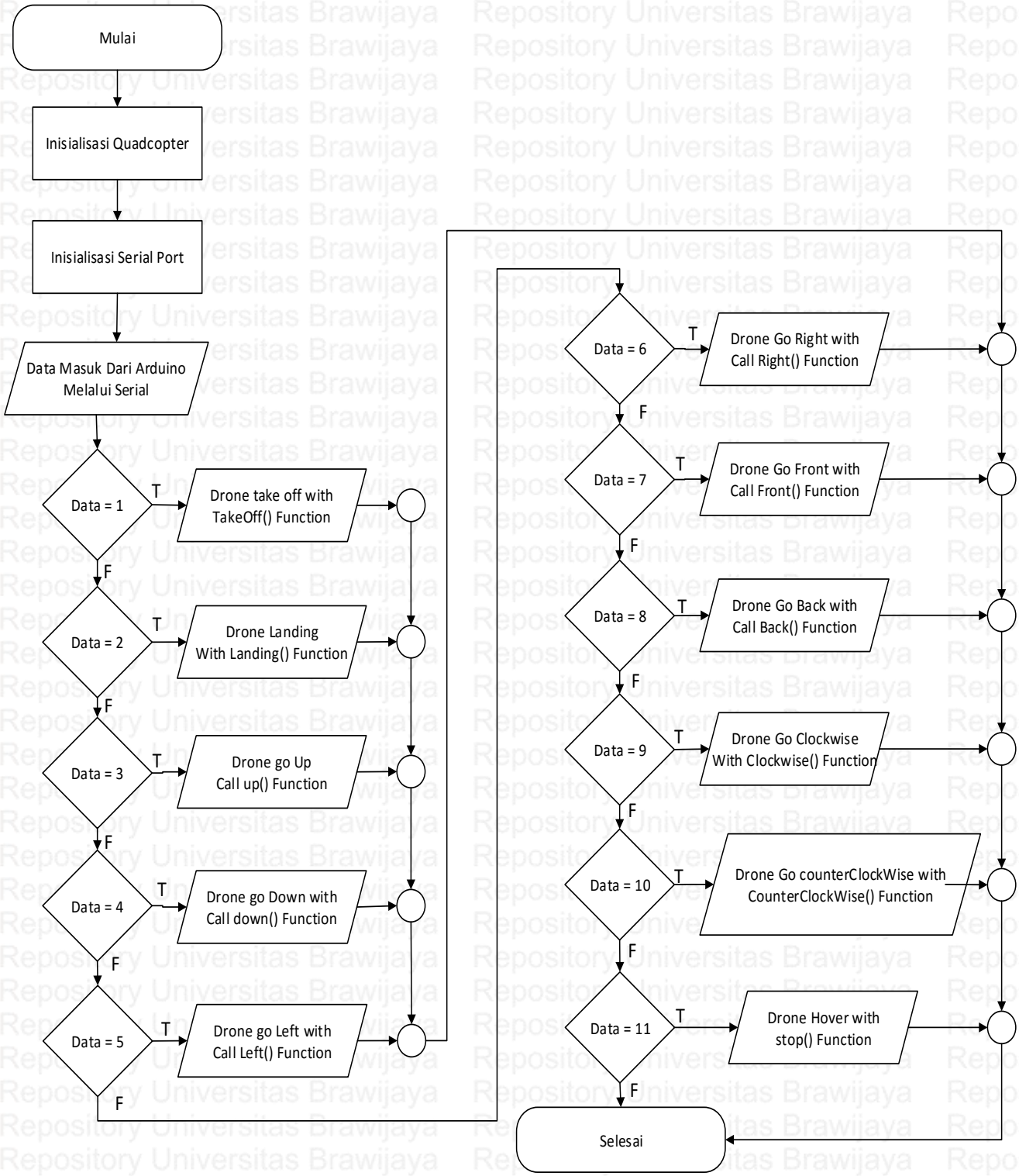
Data	*Command*
1	*TAKEOFF*
2	*LANDING*
3	NAIK
4	TURUN
5	KIRI
6	KANAN
7	DEPAN
8	BELAKANG
9	PUTAR KANAN
10	PUTAR KIRI
11	HOVER

Nilai Data Tabel 5.3 merupakan hasil pengolahan *text processing* yang dikirim melalui serial USB ke Node.JS. jika Data yang masuk bernilai 1, maka Node.JS Akan memberikan perintah kepada *quadcopter* untuk melakukan *takeoff*. Jika data yang masuk bernilai 2, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk melakukan *landing*. Jika data yang masuk bernilai 3, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk Naik Ke atas. Jika data yang masuk bernilai 4, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk turun ke bawah. Jika data yang masuk bernilai 5, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk terbang ke kiri. Jika data yang masuk bernilai 6, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk terbang ke kanan. Jika data yang masuk bernilai 7, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk terbang ke depan. Jika data yang masuk bernilai 8, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk terbang ke belakang. Jika data yang masuk bernilai 9, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk berputar searah jarum jam. Jika data yang masuk bernilai 10, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk berputar berlawanan arah dengan jarum jam. Jika data yang masuk bernilai 11, maka Node.JS akan memberikan perintah kepada *quadcopter* untuk *hover*. Alur perancangan program pada *Node.JS* dapat dilihat pada Gambar 5.10.

Gambar 5.10 merupakan alur program pada Node.JS untuk kontrol *quadcopter*, dengan memanfaatkan *Node AR.Drone* sebagai inisialisasi awal. *Serial port* berfungsi untuk menghubungkan Node.js dengan Arduino melalui komunikasi serial USB. Ada 11 jenis pergerakan dasar *quadcopter* yang digunakan, terbang, mendarat, kanan, kiri, maju, mundur, naik, turun, putar kanan, putar kiri, dan *hover*. Ketika data dari Arduino masuk, Node.JS akan mengirimkan perintah kontrol untuk *quadcopter* melalui komunikasi *Wi-fi* dengan memanggil fungsi yang terdapat pada *Node Package manager AR.Drone 2.0*.

5.2 Implementasi sistem

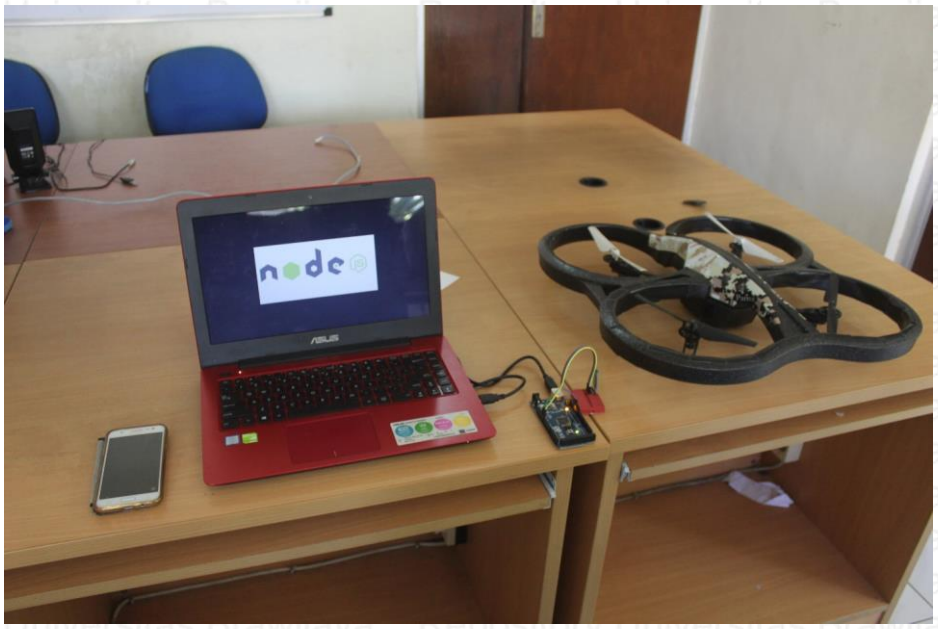
Pada bagian ini akan dijelaskan implementasi dari perancangan sistem yang telah dibuat sebelumnya, Sistem ini dimulai dengan implementasi komunikasi yang berisi bagaimana tiap-tiap komponen di dalam Android, Arduino dan Node.JS saling berhubungan satu sama lain menghasilkan sistem yang diinginkan. Bagian kedua akan dijelaskan bagaimana implementasi sistem pada Android. Pada bagian ketiga membahas tentang implementasi sistem pada Arduino, dan yang terakhir akan dijelaskan bagaimana implementasi pada Node.js. implementasi sistem akan dibahas berdasarkan pada bab perancangan yang telah dibuat sebelumnya.



Gambar 5.10 *Flowchart* Alur program pada Node.JS

5.2.1 Implementasi komunikasi sistem

Pada bagian ini akan dijelaskan bagaimana implementasi komunikasi sistem yang digunakan pada penelitian ini. Implementasi komunikasi sistem dapat dilihat pada Gambar 5.11.



Gambar 5.11 Implementasi Komunikasi Sistem

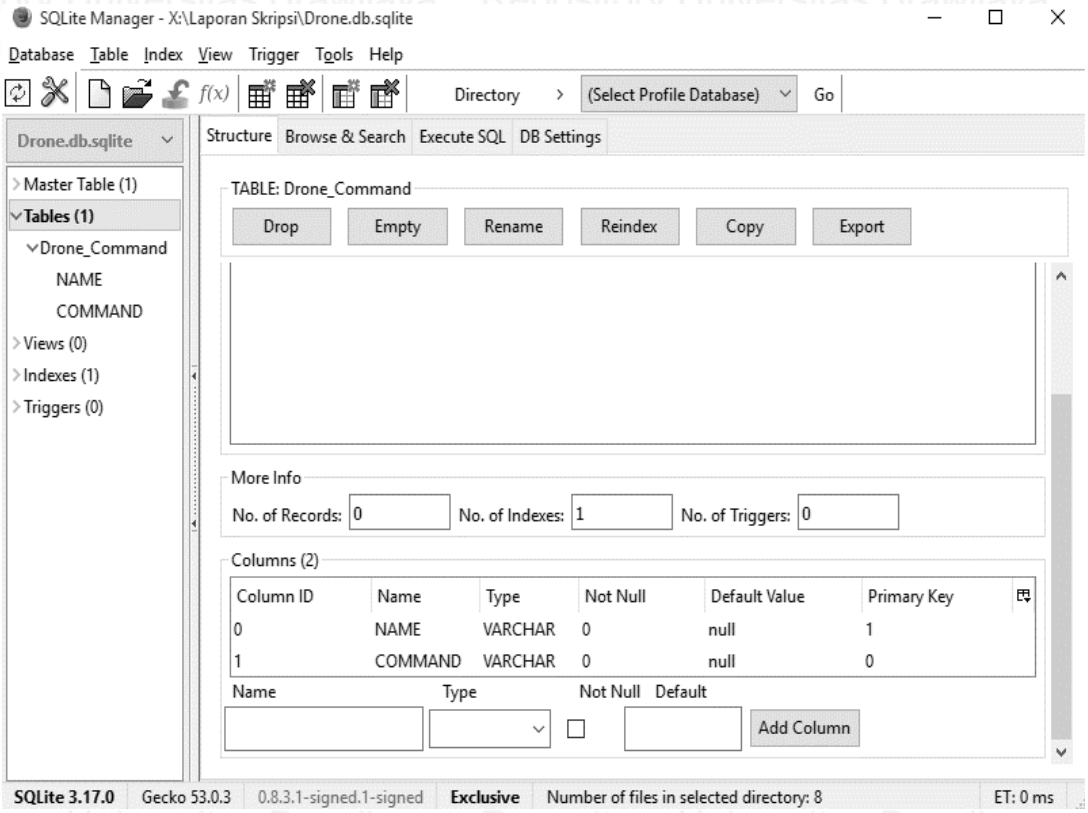
Gambar 5.11 merupakan implementasi perancangan komunikasi *sistem*. Android *device* tidak membutuhkan kabel untuk berkomunikasi dengan Arduino karena jalur komunikasinya menggunakan *Bluetooth*. Arduino dilengkapi dengan modul *Bluetooth* HC-05 yang di setting dengan mode *slave*, untuk berkomunikasi dengan Android melalui jalur *Bluetooth*. Selain itu Arduino membutuhkan USB kabel agar terhubung dengan Node.JS melalui komunikasi serial. Node.JS merupakan tempat yang berisi kumpulan perintah untuk *request navdata* dan kontrol *quadcopter* melalui jalur komunikasi Wi-*Fi*.

5.2.2 Implementasi sistem pada Android

Implementasi awal program dimulai dari pembuatan *database* dengan memanfaatkan SQLite pada Android studio, *source code* pembuatan *database* dapat dilihat pada *source code* sebagai berikut.

```
1   public static final String DATABASE_NAME = "Drone.db";
2   public static final String TABLE_NAME = "Drone_Command";
3   public static final String COL_1 = "NAME";
4   public static final String COL_2 = "COMMAND";
5
6   public DatabaseHelper(Context context) {
7       super(context, DATABASE_NAME, null, 1);
8   }
9
10  public void onCreate(SQLiteDatabase db) {
11      db.execSQL("create table " + TABLE_NAME +" (NAME VARCHAR(20)
12  PRIMARY KEY, COMMAND VARCHAR(20))");
13  }
```

Pada baris pertama sampai ke empat inisialisasi pembuatan variabel dengan tipe data *String* untuk inisialisasi *database*. Pada baris ke enam memanggil fungsi dengan nama *DatabaseHelper*, fungsi ini sebagai *constructor* untuk membuat *database* SQLite. Pada baris ke sebelas code untuk memanggil fungsi untuk membuat *database* saat aplikasi di *install*. Hasil dari *source code* dapat dilihat pada Gambar 5.12.

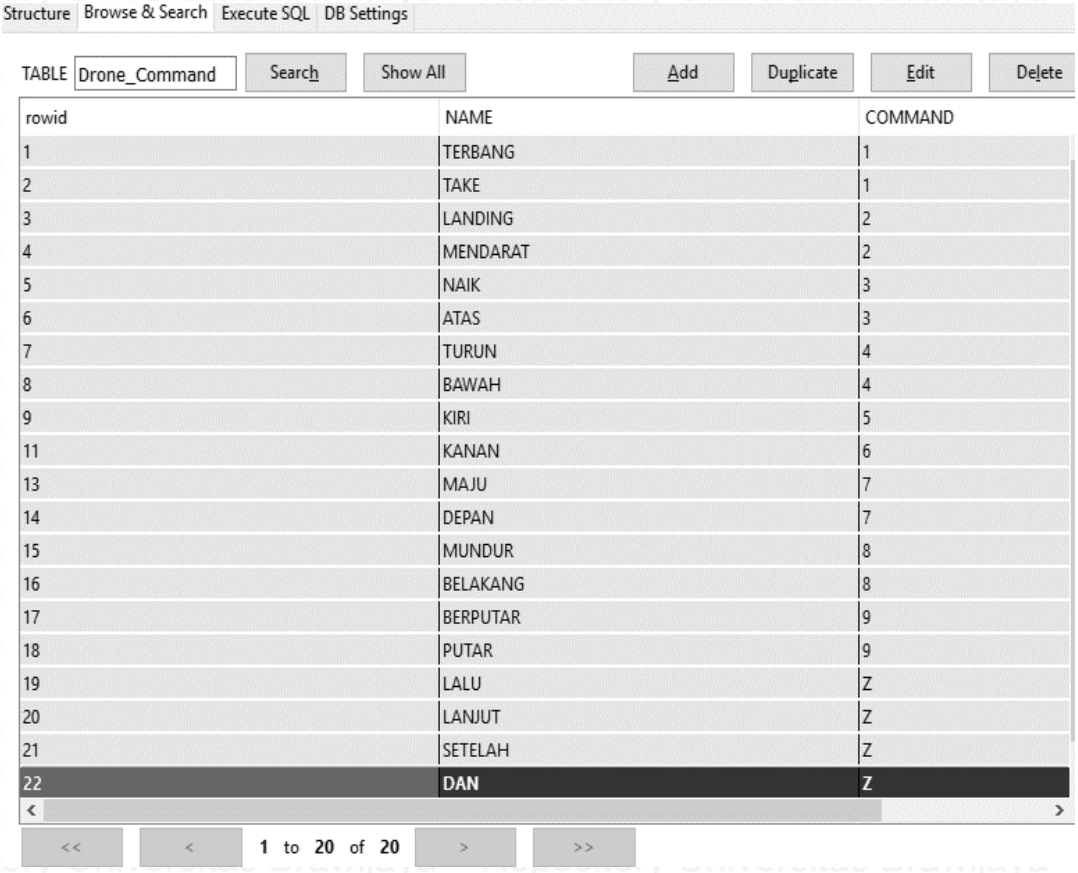


Gambar 5.12 Pembuatan *database* drone.db

Hasil pembuatan *database* dapat dilihat pada Gambar 5.12, terdapat 1 tabel yaitu tabel *Drone_Command*, berisi 2 kolom yaitu kolom "NAME" dan kolom "COMMAND" dengan tipe data VARCHAR. 2. Perintah yang digunakan untuk mengisi kolom pada *database* ini dapat dilihat pada *source code* sebagai berikut.

```
1  public boolean insertData(String name, String command) {
2      SQLiteDatabase db = this.getWritableDatabase();
3      ContentValues contentValues = new ContentValues();
4      contentValues.put(COL_1, name);
5      contentValues.put(COL_2, command);
6
7  }
```

Fungsi *source code* tersebut digunakan jika kita ingin memasukkan data ke dalam kolom NAME dan kolom COMMAND, pada baris ke 4 dan 5 data dimasukkan ke dalam kolom pertama dan kolom ke dua dengan memanggil fungsi *put()*. Implementasi Isi kata inti pada *database* dapat dilihat pada Gambar 5.13.



Structure | Browse & Search | Execute SQL | DB Settings

TABLE Drone_Command | Search | Show All | Add | Duplicate | Edit | Delete

rowid	NAME	COMMAND
1	TERBANG	1
2	TAKE	1
3	LANDING	2
4	MENDARAT	2
5	NAIK	3
6	ATAS	3
7	TURUN	4
8	BAWAH	4
9	KIRI	5
11	KANAN	6
13	MAJU	7
14	DEPAN	7
15	MUNDUR	8
16	BELAKANG	8
17	BERPUTAR	9
18	PUTAR	9
19	LALU	Z
20	LANJUT	Z
21	SETELAH	Z
22	DAN	Z

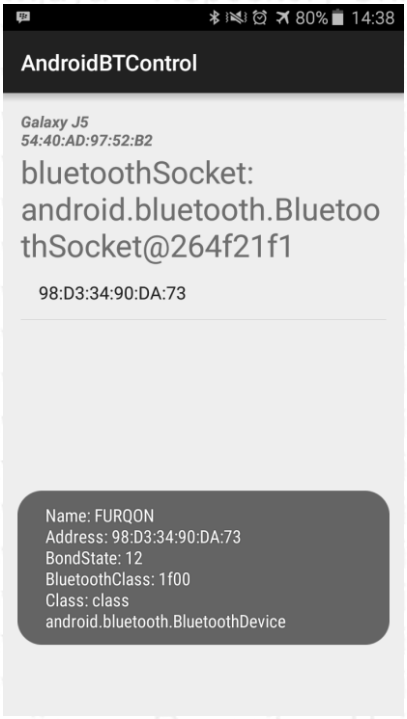
<< | < | 1 to 20 of 20 | > | >>

Gambar 5.13 Isi dari tabel *Drone_Command* kolom NAME dan *COMMAND*

Tampilan aplikasi Android yang dibuat terdapat halaman awal dan halaman utama. Halaman awal untuk memilih koneksi *bluetooth* agar terhubung dengan Arduino. Kode *XML* pada Android studio untuk halaman awal dapat dilihat pada *source code* sebagai berikut.

```
1   <TextView
2       Android:id="@+id/info"
3       Android:textStyle="bold|italic"
4       Android:layout_width="wrap_content"
5       Android:layout_height="wrap_content"/>
6
7   <TextView
8       Android:id="@+id/status"
9       Android:textSize="28sp"
10      Android:layout_width="wrap_content"
11      Android:layout_height="wrap_content"/>
12
13  <ListView
15      Android:id="@+id/pairedlist"
16      Android:layout_width="match_parent"
27      Android:layout_height="match_parent"/>
```

Fungsi *source code* tersebut akan memanggil *text view* yang berfungsi untuk menampilkan alamat dan status *device bluetooth* yang kita gunakan saat ini, selain itu akan menampilkan *list bluetooth* yang terhubung dengan *device* kita saat ini. Hasil dari *XML* dapat dilihat pada Gambar 5.14.



Gambar 5.14 Halaman Awal Program

Pada halaman awal program terdapat *list bluetooth* yang terhubung dengan *device* kita. Jika salah satu *list bluetooth* dipilih maka akan menampilkan nama, alamat, *state*, *class* dari *bluetooth* yang dipilih tadi. Kode program untuk menampilkan *list bluetooth* dapat dilihat pada *source code* sebagai berikut.

```
1   private void setup() {
2       Set<BluetoothDevice> pairedDevices =
3   bluetoothAdapter.getBondedDevices();
4       if (pairedDevices.size() > 0) {
5           pairedDeviceArrayList = new ArrayList<BluetoothDevice>();
6
7           for (BluetoothDevice device : pairedDevices) {
8               pairedDeviceArrayList.add(device);
9           }
10
11          pairedDeviceAdapter = new
12  ArrayAdapter<BluetoothDevice>(this,
13                  Android.R.layout.simple_list_item_1,
14  pairedDeviceArrayList);
15          listViewPairedDevice.setAdapter(pairedDeviceAdapter);
16
17          listViewPairedDevice.setOnItemClickListener(new
18  AdapterView.OnItemClickListener() {
19
20              @Override
21              public void onItemClick(AdapterView<?> parent, View
22  view,
23                                      int position, long id) {
24                  BluetoothDevice device =
25                          (BluetoothDevice)
26  parent.getItemAtPosition(position);
27                  Toast.makeText(MainActivity.this,
28                          "Name: " + device.getName() + "\n"
29                                  + "Address: " +
30  device.getAddress() + "\n"
```

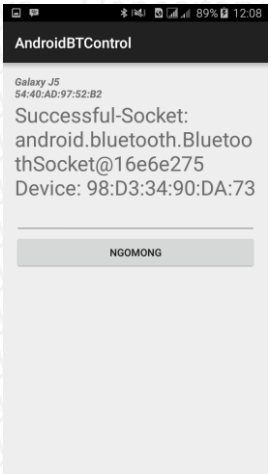
```
31                                      + "BondState: " +
32  device.getBondState() + "\n"
33                                      + "BluetoothClass: " +
34  device.getBluetoothClass() + "\n"
35                                      + "Class: " + device.getClass(),
36                              Toast.LENGTH_LONG).show();
37
38                      textStatus.setText("start ThreadConnectBTdevice");
39                      myThreadConnectBTdevice = new
40  ThreadConnectBTdevice(device);
41                      myThreadConnectBTdevice.start();
42                  }
43              });
44      }
45  }
```

Pada *source code* tersebut, program diawali dengan membuat fungsi *setup()*. Pada baris ke 2 sampai ke 18, program diawali dengan menampilkan *list bluetooth* yang terkoneksi dengan *device* kita. Sedangkan pada baris ke 21 sampai 41, merupakan kode untuk menampilkan nama, *address, bond state, class* ketika salah satu *list bluetooth* yang terhubung dipilih.

Tampilan utama pada aplikasi Android terdapat *button* yang digunakan sebagai tombol *speech recognition*, selain itu terdapat tempat untuk menampilkan hasil dari *speech recognition*. Desain *XML* pada Android studio dapat dilihat pada *source code* sebagai berikut.

```
1   <EditText
2       Android:id="@+id/masukan"
3       Android:layout_width="match_parent"
4       Android:layout_height="wrap_content"/>
5
6   <Button
7       Android:id="@+id/send"
8       Android:layout_width="match_parent"
9       Android:layout_height="wrap_content"
10      Android:text="Ngomong" />
11
```

Fungsi *source code* tersebut digunakan untuk menampilkan edit teks, dan tombol yang nantinya diberikan kode program. Hasil dari *source code* tersebut dapat dilihat pada Gambar 5.15.

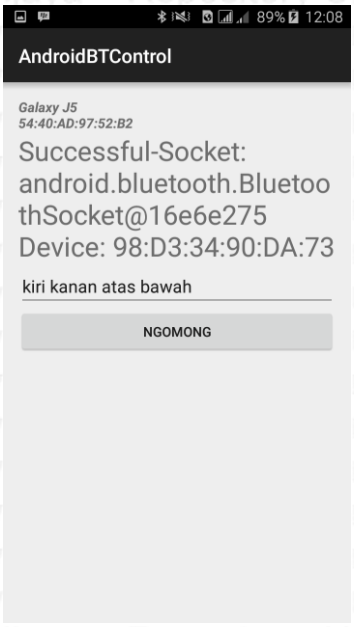


Gambar 5.15 Halaman *Home*

Pada Gambar 5.15 terdapat tombol dengan nama “NGOMONG”, tombol ini untuk memanggil *speech recognition* pada Android. Sedangkan garis kosong pada program merupakan tempat untuk menampilkan hasil *speech recognition*. Kode program yang digunakan untuk tombol “NGOMONG” dapat dilihat pada *source code* sebagai berikut.

```
1   btnSend = (Button)findViewById(R.id.send);
2
3   btnSend.setOnClickListener(new View.OnClickListener(){
4
5       @Override
6       public void onClick(View v) {
7           if(myThreadConnected!=null){
8               StartTime = SystemClock.uptimeMillis();
9               handler.postDelayed(runnable, 0);
10              btnToOpenmic();
11          }
12      }});
13
```

Pada kode program tersebut, tombol “ngomong” diberikan *id* dengan nama “*send*”. pada baris ke 3 sampai 12 kode program ini digunakan untuk inisialisasi jika tombol di klik maka akan memanggil *speech recognition* pada Android *device*. Hasil dari pengolahan suara dapat dilihat pada Gambar 5.16.



Gambar 5.16 implementasi dari pengolahan suara

Jika tombol “NGOMONG” di klik , lalu *user* memberikan masukan berupa suara, maka akan ditampilkan hasil dari pengolahan suara seperti Gambar 5.16 yang menghasilkan kalimat “Kiri Kanan Atas Bawah”. Hasil dari pengolahan suara yang berbentuk kalimat akan dipecah menjadi kata menggunakan perintah *split*, lalu tiap kata akan di cek di dalam *database*, kode untuk mencari data didalam *database* dapat dilihat pada kode sebagai berikut.

```
1   Cursor res = db.rawQuery("select COMMAND from " + TABLE_NAME + "
2   where NAME=\"" + name + "\"", null);
3
4   if(res.getCount() == 0) {
5           return "0";
6       }
7
8       while (res.moveToNext()) {
9           return res.getString(0);
10      }
11      return "0";
12  }
```

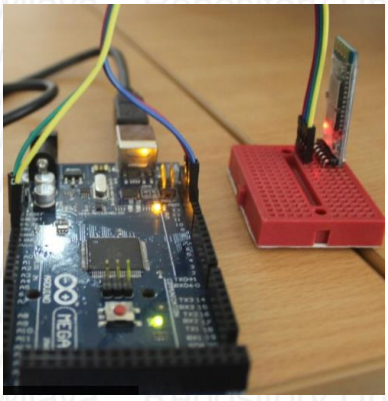
Pada *source code* tersebut, semua kata hasil pemotongan kalimat dicari didalam *database* menggunakan perintah *select.* Jika kata terdapat pada tabel Name, maka akan dikembalikan kolom Command, namun jika kata tidak terdapat pada kolom Name, maka akan dikembalikan nilai "0".

5.2.3 Implementasi sistem pada Arduino

Pada Arduino program dimulai dengan memanggil *library softwareserial* untuk inisialisasi bahwa *Bluetooth* HC-05 terhubung melalui komunikasi serial. Implementasi Kode program dapat dilihat pada *source code* sebagai berikut.

```
1   #include <SoftwareSerial.h>
2   SoftwareSerial BT(10, 11);
3
4   void setup() {
5     BT.begin(9600);
6   }
```

Arduino terhubung dengan HC-05 melalui komunikasi serial dengan *baud rate* 9600, pin 10 pada Arduino sebagai *receiver*, dan pin 11 sebagai *transmitter*. Pemasangan *Bluetooth module* HC-05 dapat dilihat pada Gambar 5.17.



Gambar 5.17 Implementasi HC-05 denga Arduino

Bluetooth menggunakan 5 volt tegangan agar bekerja secara maksimal, pin Tx pada HC-05 dihubungkan ke pin 10 Arduino, pin Rx pada dihubungkan ke pin 11 Arduino. Selain menggunakan HC-05 Arduino berfungsi untuk melakukan pengolahan data, implementasi alur penerimaan data dari Android dapat dilihat pada *source code* sebagai berikut.

```
1   while (BT.available()) {
2     char c = BT.read();
3     delay(10);
4     if (c != '0') {
5       Command[totalCom] = c ;
6       totalCom++;
7     }
8   }
```

Data yang masuk dari Android dimasukkan ke dalam *array*, dengan inisialisasi *BT.Available()* data yang masuk dari Android dipisah dan dikumpulkan menjadi kumpulan *array*. *Array* tersebut akan diolah dengan digabungkan atau berdiri sendiri, proses ini dinamakan *array processing*. Proses pengolahan data utama pada Arduino untuk dikirim ke Node.Js dapat dilihat pada *source code* sebagai berikut.

```
1   if (Command[i] == '1' || Command[i] == '2' || Command[i] == '3' ||
2   Command[i] == '4' || Command[i] == '5' || Command[i] == '6' ||
3   Command[i] == '7' || Command[i] == '8') {
4             int class1 = (int)Command[i] - 48;
5             delay(12);
6             Serial.write((int)class1);
7             delay(2500);
8             if (Command[i] == '1') {
9               delay(3000);
10            }
11            Serial.write(30);
12            delay(1500);
13            if (Command[i] == '2') {
15              delay(3010);
16              hoverstat = 1;
```

Keluaran dari *text processing* berupa nilai *integer*, yang langsung dikirim ke Node.JS langsung. Diberikan *delay* pada tiap tahapnya, serta program ini akan berulang sampai *user* memberikan perintah *landing* untuk *quadcopter*.

5.2.4 Implementasi sistem pada Node.Js

Implementasi sistem pada Node.JS berguna untuk kontrol *quadcopter*, program diawali dengan memanggil fungsi *ar.drone client node* dan *serial port*. Kode pemanggilan dapat dilihat pada *source code* sebagai berikut.

```
1   var arDrone = require('ar-drone');
2   var SerialPort = require("serialport");
3
4   var client = arDrone.createClient();
5   var serialport = new SerialPort('/dev/ttyO3);
```

Baris pertama dan kedua merupakan kode untuk memanggil *node ar-drone* dan *node serialport*, yang terdapat pada *Node Package Manager*(NPM), lalu inisialisasi variabel baru dengan nama *client* untuk memanggil kelas *Ar-drone*, dan variabel *serialport* untuk memanggil kelas *serial port*.

Data yang masuk dari Arduino dimasukkan ke dalam *array*, dan langsung diolah menjadi gerakan *quadcopter* dengan memanggil fungsi yang ada pada *node Ar.Drone*. *Source code* untuk perintah kontrol dapat dilihat pada *source code* sebagai berikut.

```
1  serialport.on('open', function(){
2          console.log('Serial Port Opened');
3          serialport.on('data', function(data){
4          if(data[0] == 1){
5          console.log("Drone Terbang\n");
6          client.takeoff();
7          client.disableEmergency();
8          }
```

Kode program Node.JS tersebut merupakan kumpulan perintah untuk kontrol *quadcopter*, jika data yang masuk dari Arduino bernilai 0, maka akan program akan menjalankan baris ke 6 yang merupakan *source code* untuk memanggil fungsi *client.takeoff()* , dengan fungsi ini *quadcopter* akan melakukan *takeoff*. Pada kode program ini ini sebagai inisialisasi bahwa program dimulai. Kode program untuk pemanggilan fungsi kontrol *quadcopter* dapat dilihat pada *source code* sebagai berikut.

```
1          else if(data[0]== 2){
2          console.log("Drone Stop\n");
3      client.land();
4          }
```

Jika data yang masuk dari Arduino bernilai 2, maka *quadcopter* akan *landing* dengan memanggil fungsi *client.land(),* ketika fungsi ini dipanggil *quadcopter* tidak akan bergerak ke manapun sebelum perintah *takeoff* diberikan kembali.

```
1          else if(data[0] == 3){
2          client.up(0.15);
3          console.log("Drone Keatas\n");
4          }
```

Jika data yang masuk dari Arduino bernilai 3, maka *quadcopter* akan terbang keatas dengan memanggil fungsi *client.up().*

```
1          else if(data[0] == 4){
2          client.down(0.15);
3          console.log("Drone Kebawah\n");
4          }
```

Jika data yang masuk dari Arduino bernilai 4, maka *quadcopter* akan terbang kebawah dengan memanggil fungsi *client.down().*

```
1          else if(data[0] == 5){
2          client.left(0.07);
3          console.log("Drone Kekiri\n");
4          }
```

Jika data yang masuk dari Arduino bernilai 5, maka *quadcopter* akan belok kekiri dengan memanggil fungsi *client.left().*

```
1          else if(data[0]== 6){
2          client.right(0.07);
3          console.log("Drone Kekanan\n");
4          }
```

Jika data yang masuk dari Arduino bernilai 6, maka *quadcopter* akan belok ke kanan dengan memanggil fungsi *client.right().*

```
1           else if(data[0]== 7){
2           client.front(0.07);
3           console.log("Drone Kedepan\n");
4           }
```

Jika data yang masuk dari Arduino bernilai 7, maka *quadcopter* akan maju ke depan dengan memanggil fungsi *client.front().*

```
1           else if(data[0]== 8){
2           client.back(0.07);
3           console.log("Drone Kebelakang\n");
4           }
```

Jika data yang masuk dari Arduino bernilai 8, maka *quadcopter* akan mundur ke belakang dengan memanggil fungsi *client.back().*

```
1           else if(data[0]== 9){
2           client.clockwise(0.7);
3           console.log("Putar Kekanan\n");
4           }
```

Jika data yang masuk dari Arduino bernilai 9, maka *quadcopter* akan berputar ke kanan searah jarum jam dengan memanggil fungsi *client.clockwise().*

```
1           else if(data[0]== 10){
2           client.counterClockwise(0.7);
3           console.log("Putar Kekiri\n");
4           }
```

Jika data yang masuk dari Arduino bernilai 10, maka *quadcopter* akan berputar ke kiri berlawanan arah jarum jam dengan memanggil fungsi *client.counterClockwise().*

```
1           else if(data[0]== 11){
2           console.log("Drone Hover\n");
3           client.stop();
4           }
```

Jika data yang masuk dari Arduino bernilai 11, maka *quadcopter* akan *hover* atau diam di tempat dengan memanggil fungsi *client.stop().* State ini terjadi setelah perintah *user* sudah habis.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini menjelaskan mengenai pengujian yang dilakukan pada penelitian, dimulai dari prosedur serta proses pengujian yang akan menghasilkan data untuk nantinya di analisis. Proses pengujian dilakukan untuk mengetahui apakah sistem yang dibuat sesuai dengan analisis kebutuhan yang diinginkan. Pengujian dilakukan dengan tahapan-tahapan yang telah ditentukan, hasil yang diperoleh dari pengujian dianalisis agar dapat ditarik kesimpulan dari penelitian yang telah dilakukan. Pengujian pada bab ini meliputi pengujian fungsional sistem, proses pengolahan teks, kendali *quadcopter*, performa aplikasi, dan performa *quadcopter*.

6.1 Pengujian Fungsional Sistem

6.1.1 Tujuan pengujian

Pengujian ini untuk mengetahui apakah sistem sudah berjalan sesuai dengan keinginan peneliti. Semua sistem yang digunakan akan diuji, dari segi Android, Arduino dan Node.JS .

6.1.2 Prosedur pengujian

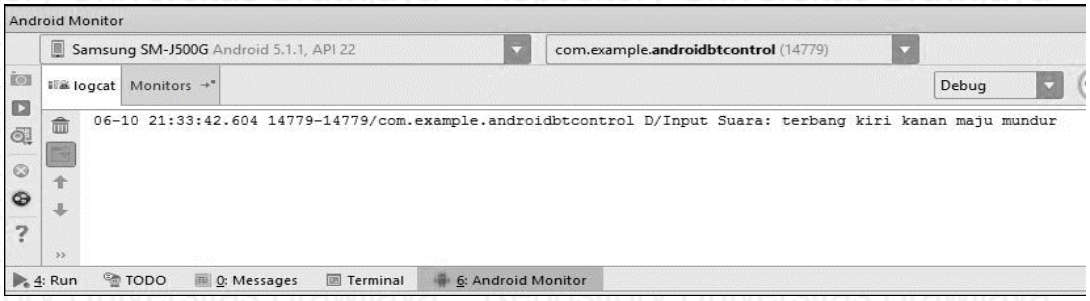
Pengujian masukan suara dari Android, lalu pengolahan hasil pencocokan *database*, hasil keluaran dari pengujian akan ditampilkan melalui *console log*. Hasil dari pengolahan akan dikirim ke Arduino, pengujianya dengan melihat dari serial monitor yang ada pada Arduino. *Command prompt* pada Node.JS akan menampilkan arah *quadcopter* berjalan. Tahap pengujian fungsional akan dilakukan sebanyak 10 kali pengujian dengan banyak masukan kata yang berbeda.

6.1.3 Pelaksanaan pengujian

Untuk melakukan pengujian ini, *user* memberikan masukan suara dari Android *device*, hasilnya keluarannya dilihat pada *console log*, lalu kalimat masukan dilakukan *Text preprocessing*, hasilnya akan dilihat pada *console log*, lalu dikirim ke Arduino. Arduino menerima data dari Android yang bisa dilihat pada melalui serial monitor. Data dari Android akan diolah di dalam Arduino untuk dikirim ke Node.JS . data pada node js akan diolah lagi menjadi gerakan *quadcopter*, tiap gerakan *quadcopter* dapat dilihat melalui Node.JS *command prompt*. Pengujian fungsional berhasil jika masukan suara dari Android sesuai dengan pergerakan *quadcopter*.

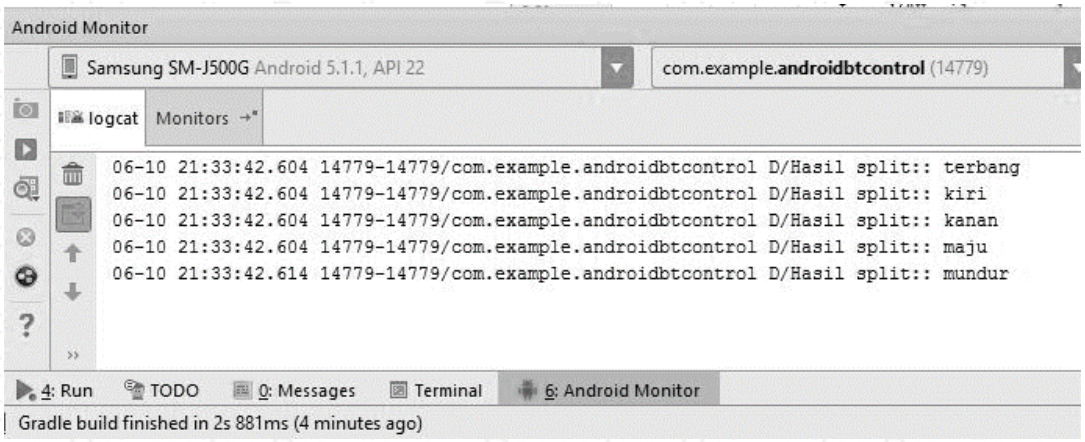
6.1.4 Hasil Pengujian

Pengujian pada Android *device* dimulai dari tampilan pengolahan *speech recognition*. Jika keluaran dari Android *device* sesuai dengan yang diinginkan maka sistem akan melakukan *text splitting* dan pencocokan pada *database*. Salah satu hasil pengujian dapat dilihat pada Gambar 6.1.



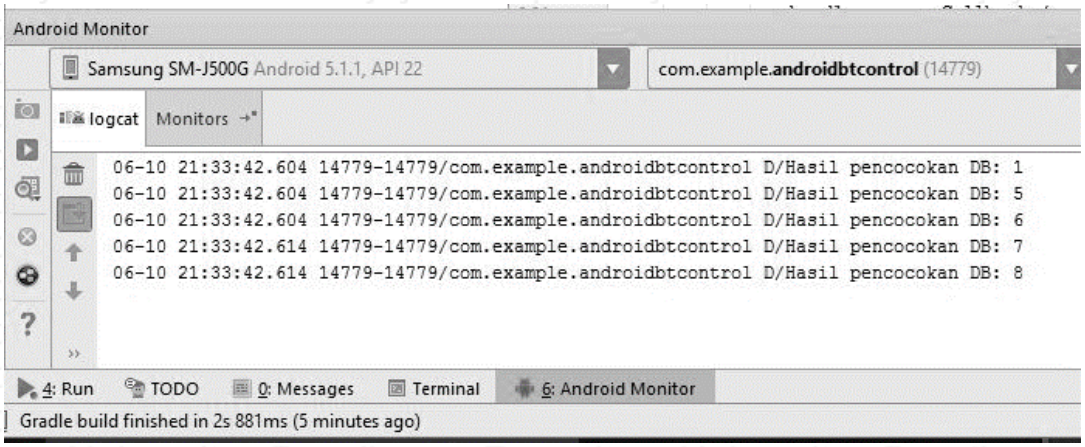
Gambar 6.1 Hasil pengujian masukan suara

User memberikan masukan suara berupa kalimat "terbang kiri kanan maju mundur", melalui *speech recognition* pada Android lalu dilakukan *splitting*, hasil pengujian pemisahan kalimat dapat dilihat pada Gambar 6.2.



Gambar 6.2 Hasil pengujian pemotongan kalimat menjadi kata

Pada Gambar 6.2 Kalimat masukan dari *user* dipisah menjadi satuan kata, "terbang" ,kiri", "kanan", "maju", dan "mundur".

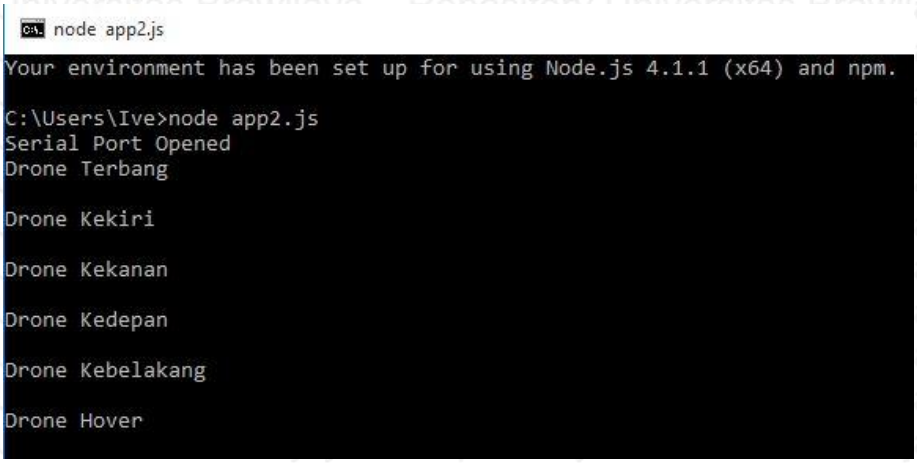


Gambar 6.3 Hasil pencocokan tiap kata pada *database*

Tiap satuan kata yang masuk akan dicek di dalam *database* Gambar 6.3 merupakan pengujian pengecekan tiap kata di dalam *database*, lalu dikirim ke Arduino.

Gambar 6.4 Pengujian fungsional pada Arduino

Gambar 6.4 merupakan pengujian pengiriman data dari Android yang dapat dilihat pada serial monitor pada Arduino. data di atas akan diolah dan dikirim ke Node.js untuk menjadi kontrol *quadcopter*.



Gambar 6.5 Pengujian fungsional pada Node.JS

Hasil pengujian pengiriman data dari Arduino dapat dilihat pada Gambar 6.5 melalui *command prompt*, dapat dilihat bahwa Node.JS berhasil menerima dan mengolah perintah sesuai dengan data pengolahan dari Arduino. Hasil 10 kali pengujian secara keseluruhan dapat dilihat pada Tabel 6.1

Tabel 6.1 Hasil Pengujian Fungsional Sistem

Banyak Masukan*/kata*	Kata	Hasil Benar/kata	Hasil Salah/kata
1	kanan	1	0
	kiri	1	0
	maju	1	0
	mundur	1	0
	atas	1	0
	bawah	1	0
	kiri	1	0
	kanan	1	0
	terbang	1	0

Tabel 6.1 Hasil Pengujian Fungsional Sistem Lanjutan

Banyak Masukan*/kata*	Kata	Hasil Benar/kata	Hasil Salah/kata
	landing	1	0
2	kanan kiri	2	0
	maju kanan	2	0
	terbang mundur	2	0
	kanan turun	2	0
	terbang maju	2	0
	turun landing	2	0
	putar kanan	2	0
	putar kiri	2	0
	terbang mundur	2	0
	kiri bawah	2	0
3	terbang kanan maju	3	0
	maju mundur turun	3	0
	kanan turun kiri	3	0
	kiri turun kanan	3	0
	maju mundur kanan	3	0
	kanan maju mundur	3	0
	kanan maju naik	3	0
	terbang kiri naik	3	0
	turun maju maju	3	0
	putar kanan kiri	3	0
4	terbang jangan ke kiri	4	0
	terbang kanan kiri maju	4	0
	mundur kiri kanan landing	4	0
	ke kiri lalu landing	4	0
	ke kanan lanjut depan	4	0
	ke depan kanan kiri	4	0
	terbang kiri kanan landing	4	0
	saya ingin kamu terbang	4	0
	ke kiri dan maju	4	0
	kanan lalu ke depan	4	0

Tabel 6.1 Hasil Pengujian Fungsional Sistem Lanjutan

Banyak Masukan*/kata*	Kata	Hasil Benar/kata	Hasil Salah/kata
5	terbang kiri kanan maju mundur	5	0
	kiri kanan maju naik landing	5	0
	kanan kiri maju depan kanan	5	0
	maju kiri kanan depan landing	5	0
	terbang ke kanan lalu landing	5	0
	terbang lalu depan lalu landing	5	0
	ke depan kanan maju bawah	5	0
	kiri kiri kanan kanan bawah	5	0
	kanan bawah maju maju landing	5	0
	jangan ke kiri lalu maju	5	0
Jumlah		150	0

Tabel 6.1 merupakan nilai hasil pengujian fungsional sistem dengan 10 kali percobaan pada setiap masukan katanya. Hasil nilai kata benar pada pengujian ini sebanyak 150 benar, tanpa ada nilai salah.

6.1.5 Analisis pengujian

Hasil pengujian 10 kali yang telah dilakukan dapat dihasilkan persentase nilai benar dengan menggunakan rumus 2.1.

Hasil perhitungan sebagai berikut.

$$\text{Nilai Persentase Akurasi} = \frac{150}{150} X\ 100\% = 100\%$$

Pengujian fungsional pada sistem ini dapat disimpulkan berhasil karena memiliki persentase kebenaran sebesar 100%.

6.2 Pengujian Proses Pengolahan Teks

6.2.1 Tujuan Pengujian

Pengujian ini berfungsi untuk mengetahui apakah sistem mampu mengolah kalimat yang dijadikan masukan dengan *user* yang berbeda-beda secara benar dan akurat.

6.2.2 Prosedur Pengujian

Prosedur pengujian yang dilakukan menggunakan 10 *user* berbeda lalu mencoba sistem pengolahan kalimat, hasilnya keluarannya dilihat melalui console log yang ada pada Android studio.

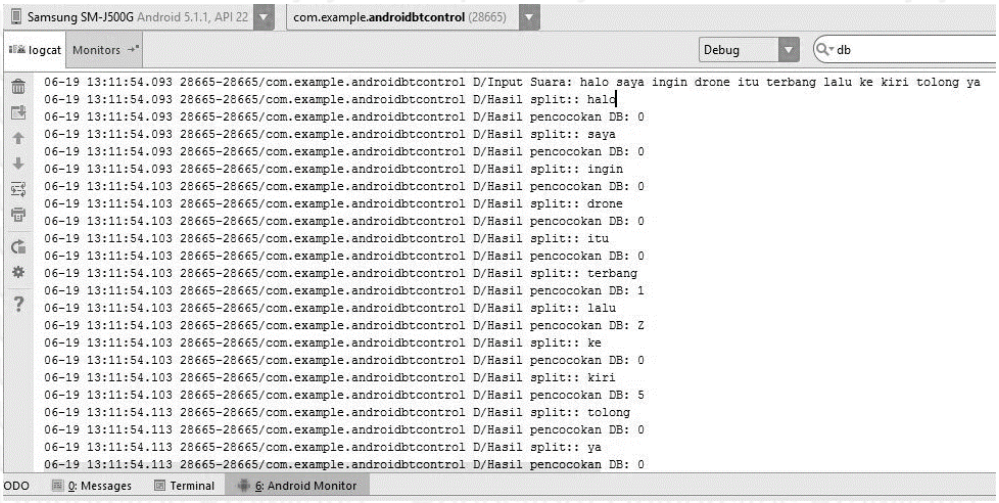
6.2.3 Pelaksanaan Pengujian

Sepuluh *User* yang berbeda memberikan perintah suara tanpa ada batasan kata apa saja yang harus diucapkan, hasil pengolahan teks akan dicatat dan dijadikan analisis. Sistem akan menampilkan hasil pengolahan teks, jika kata yang diucapkan terdapat pada *database*, maka akan dihasilkan tabel kedua pada *database*, jika kata tidak terdapat pada *database* akan menghasilkan nilai '0'.

6.2.4 Hasil Pengujian

a. *User* 1

User 1 memberikan perintah "halo saya ingin drone itu terbang lalu ke kiri tolong ya". Hasil pengolahan kalimat dapat dilihat pada Gambar 6.6.



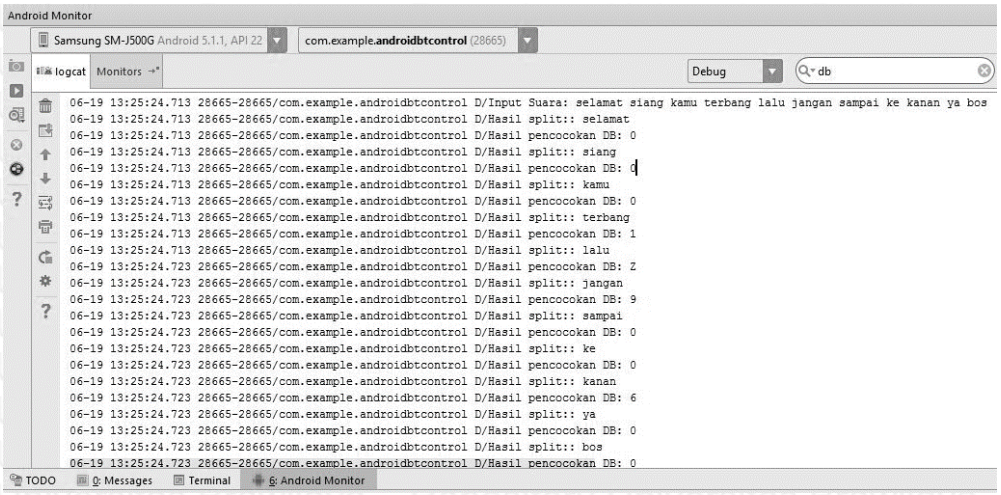
Gambar 6.6 Hasil pengujian *user* pertama

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata "halo", "saya", "ingin", "drone', "itu", "terbang", "lalu" "ke", "kiri", "tolong, "ya", lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai "0".

b. *User* 2

User 2 memberikan perintah "selamat siang kamu terbang lalu jangan sampai ke kanan ya bos". Hasil pengolahan kalimat dapat dilihat pada Gambar 6.7.

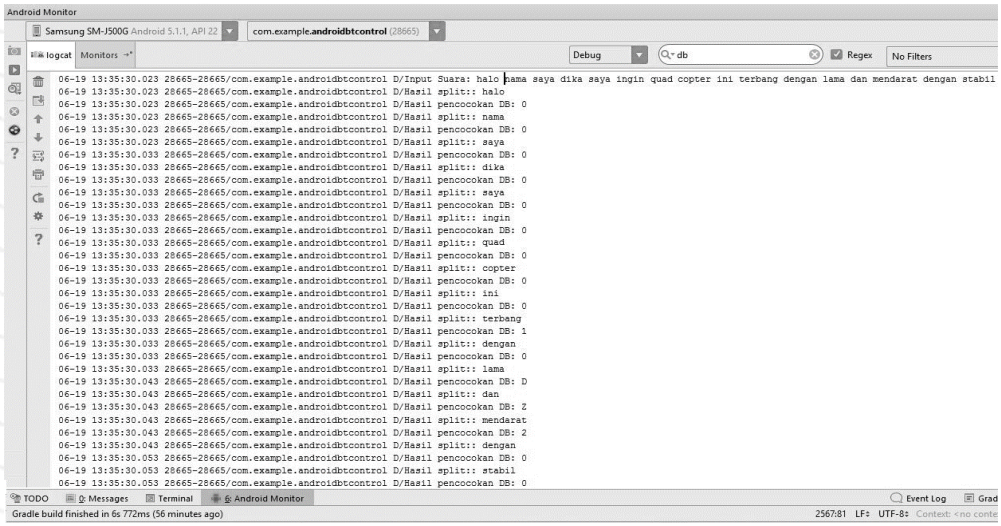
Pada Gambar 6.7 Hasil pengujian *user* kedua sistem memisah kalimat yang masuk dari *user* menjadi satuan kata "halo", "selamat", "siang", "kamu, "terbang", "lalu", "jangan" "sampai", "ke", "kanan, "ya", "bos". lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai "0".



Gambar 6.7 Hasil pengujian *user* kedua

c. *User* 3

User 3 memberikan perintah “halo nama saya dika saya ingin quad copter ini terbang dengan lama dan mendarat dengan stabil”. Hasil pengolahan kalimat dapat dilihat pada Gambar 6.8.

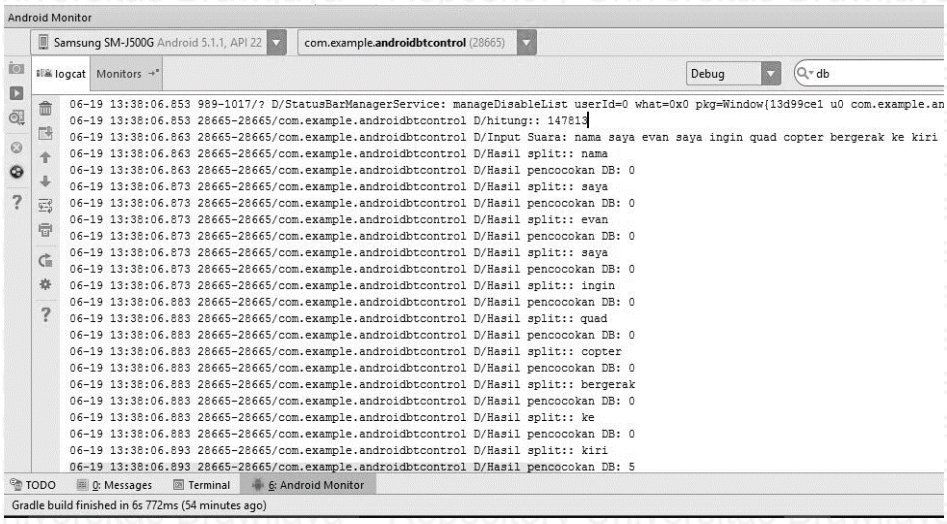


Gambar 6.8 Hasil pengujian *user* ketiga

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata “halo”, “nama”, “saya”, “dika”, “saya”, “ingin”, “quad”, “copter”, “ini”, “terbang”, “dengan”, “lama”, “dan”, “mendarat”, “dengan”, “stabil”. lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai “0”.

d. *User* 4

User 4 memberikan perintah “nama saya evan saya ingin quad copter bergerak ke kiri”. Hasil pengolahan kalimat dapat dilihat pada Gambar 6.9.

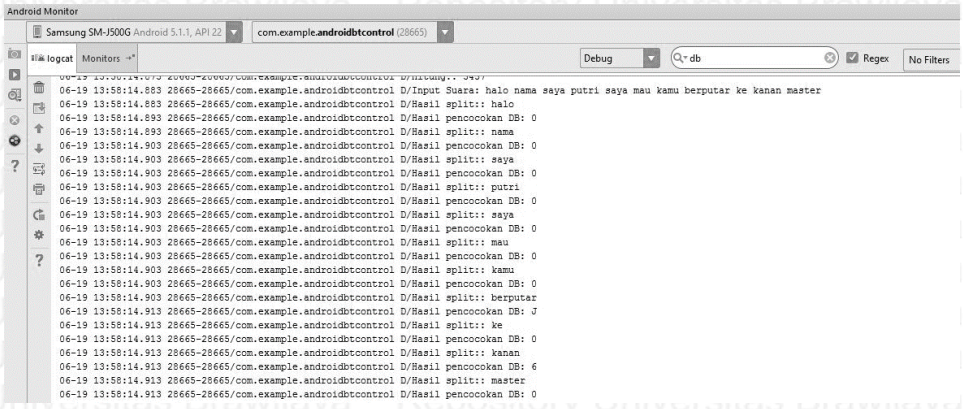


Gambar 6.9 Hasil pengujian *user* keempat

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata “nama”, “saya”, “evan”, “saya”, “ingin”, “quad”, “copter”, “bergerak”, “ke”, “kiri”. lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai “0”.

e. *User* 5

User 5 memberikan perintah “halo nama saya putri saya ingin kamu berputar ke kanan master”. Hasil pengolahan kalimat dapat dilihat pada Gambar 6.10.

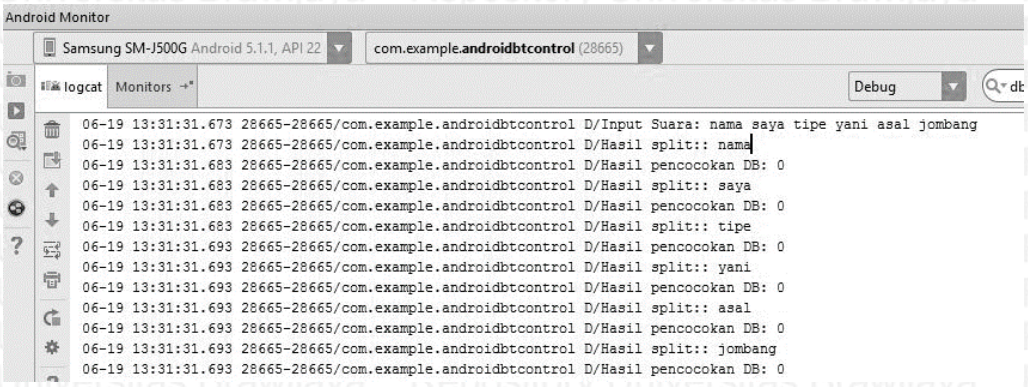


Gambar 6.10 Hasil pengujian *user* kelima

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata “halo”, “nama”, “saya”, “putri”, “saya”, “ingin”, “kamu”, “berputar”, “ke”, “kanan”, “master”. lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai “0”.

f. *User* 6

User 6 memberikan perintah “nama saya tipe yani asal jombang”. Hasil pengolahan kalimat dapat dilihat pada Gambar 6.11.

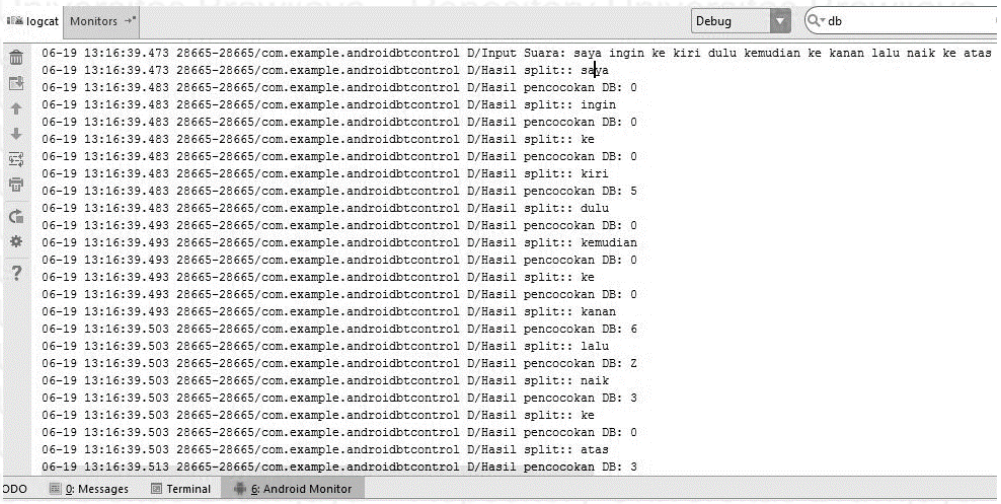


Gambar 6.11 Hasil pengujian *user* keenam

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata “nama”, “saya”, “tipe”, “yani”, “asal”, “jombang. lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai “0”.

g. *User* 7

User 7 memberikan perintah “saya ingin ke kiri dulu kemudian ke kanan lalu naik ke atas”. Hasil pengolahan kalimat dapat dilihat pada Gambar 6.12.

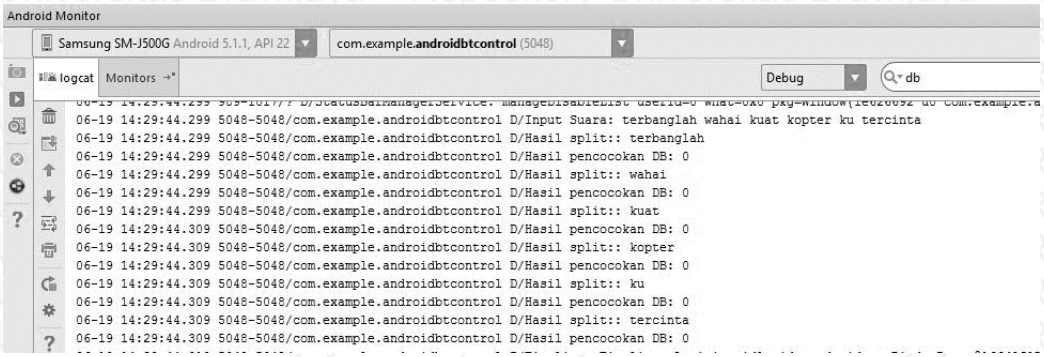


Gambar 6.12 Hasil pengujian *user* ketujuh

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata “saya”, “ingin”, “ke”, “kiri”, “dulu”, “kemudian”, “ke”, “kanan”, “lalu”, “naik”, “ke”, “atas”. lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai “0”.

h. *User* 8

User 8 memberikan perintah “terbanglah wahai quad copter ku tercinta”. Hasil pengolahan kalimat dapat dilihat pada Gambar 6.13.

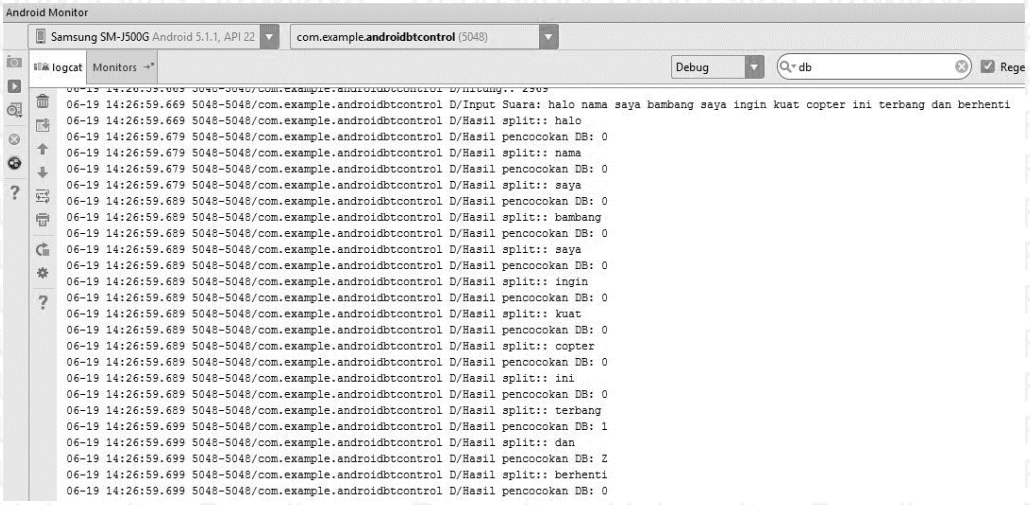


Gambar 6.13 Hasil pengujian *user* kedelapan

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata "terbanglah", "wahai", "quad", "copter", "ku", "tercinta". lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai "0".

i. *User* 9

User 9 memberikan perintah "halo nama saya bambang saya ingin *quadcopter* ini terbang dan berhenti". Hasil pengolahan kalimat dapat dilihat pada Gambar 6.14.

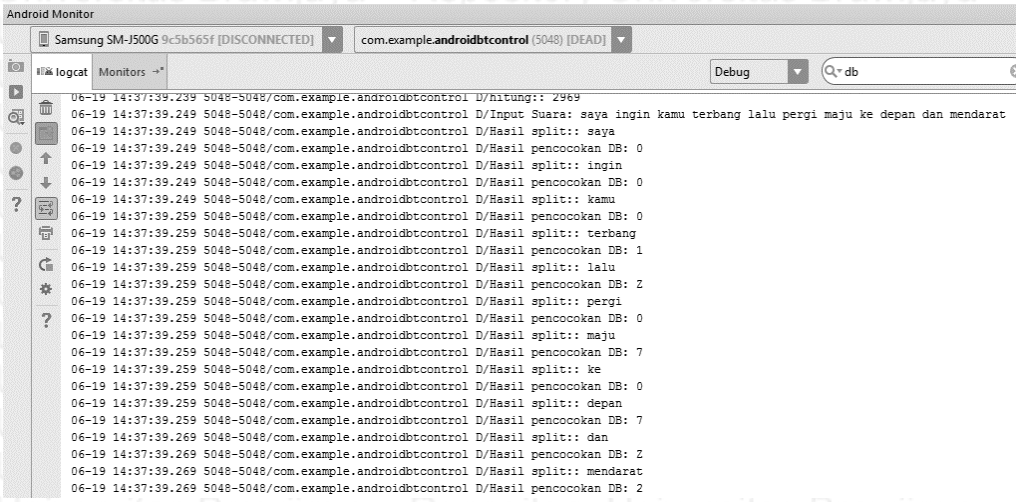


Gambar 6.14 Hasil pengujian *user* kesembilan

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata "halo", "nama", "saya", "bambang", "saya", "ingin", "quad", "copter", "ini", "terbang", "dan", "berhenti". lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai "0".

j. *User* 10

User 10 memberikan perintah "saya ingin kamu terbang lalu pergi maju ke depan dan mendarat". Hasil pengolahan kalimat dapat dilihat pada Gambar 6.15.



Gambar 6.15 Hasil pengujian *user* kesepuluh

sistem memisah kalimat yang masuk dari *user* menjadi satuan kata “saya”, “ingin”, “kamu”, “terbang”, “lalu”, “pergi”, “maju”, “ke”, “depan”, “dan”, “mendarat”. Lalu tiap kata akan di cek di dalam *database*, jika kata terdapat pada *database* akan dikirim nilai tabel command, namun jika data tidak terdapat pada *database* akan dikirim nilai “0”.

6.2.5 Analisis Pengujian

Dari hasil pengujian 10 *user* yang mencoba sistem ini, pemotongan kalimat untuk menghasilkan kumpulan kata memiliki hasil yang sesuai dengan harapan peneliti. Hasil penghitungan pada pengujian ini dapat dilihat pada Tabel 6.2

Tabel 6.2 Hasil Pengujian Pengolahan Teks dan Suara

No	*User* ke	Masukan Kata	Banyak Kata	Jumlah Benar
1	*User* 1	“halo saya ingin drone itu terbang lalu ke kiri tolong ya”	11	11
2	*User* 2	“selamat siang kamu terbang lalu jangan sampai ke kanan ya bos”	11	11
3	*User* 3	“halo nama saya dika saya ingin quad copter ini terbang dengan lama dan mendarat dengan stabil”	16	16
4	*User* 4	“nama saya evan saya ingin quad copter bergerak ke kiri”	10	10
5	*User* 5	“halo nama saya putri saya ingin kamu berputar ke kanan master”	11	11
6	*User* 6	“nama saya tipe yani asal jombang”	6	6

Tabel 6.2 Hasil Pengujian Pengolahan Teks dan Suara Lanjutan

No	*User* ke	Masukan Kata	Banyak Kata	Jumlah Benar
7	*User* 7	"saya ingin ke kiri dulu kemudian ke kanan lalu naik ke atas"	12	12
8	*User* 8	"terbanglah wahai quad copter ku tercinta"	6	6
9	*User* 9	"halo nama saya bambang saya ingin *quadcopter* ini terbang dan berhenti"	11	11
10	*User* 10	"saya ingin kamu terbang lalu pergi maju ke depan dan mendarat"	11	11
Jumlah			105	105

Dari data yang diperoleh dapat dihitung persentase nilai akurasi *database* dan pengolahan teks menggunakan rumus 2.1.

dengan hasil perhitungan.

$$\text{Nilai Persentase Akurasi} = \frac{105}{105} X\ 100\% = 100\%$$

Dari hasil penghitungan, pengujian sistem memiliki akurasi 100% Hasil pengolahan *database* juga memiliki akurasi yang baik, maka dapat disimpulkan tingkat kebenaran pengolahan teks dapat dipastikan akurat dan benar.

6.3 Pengujian Kendali *Quadcopter* Dengan Individu Yang Berbeda

6.3.1 Tujuan pengujian

Tujuan dari pengujian ini untuk menguji efektivitas sistem, dan akurasi sistem. Selain itu memastikan bahwa sistem ini berjalan lancar dengan masukan suara dari *user* yang berbeda-beda.

6.3.2 Prosedur pengujian

Prosedur pengujian pada tahap ini sebanyak 10 *user* yang berbeda diberikan sistem ini. Setiap *user* akan mencoba berbagai kombinasi perintah yang berbeda. Lalu diambil nilai data *quadcopter* dan dijadikan grafik untuk pembuktian gerakan. Penjelasan mengenai data yang dihasilkan oleh *quadcopter* yang lebih rinci dapat dilihat pada sub bab pengujian performa *quadcopter*.

6.3.3 Pelaksanaan pengujian

10 *User* memberikan masukan suara, lalu dilihat hasil akhirnya apakah kalimat tersebut sesuai dengan gerakan *quadcopter*. Lalu dihitung tingkat akurasi gerakan yang dihasilkan.

6.3.4 Hasil pengujian

Hasil pengujian pada sistem ini berupa 10 *user* yang berbeda, lalu tingkat benar dan kesalahan dihitung, hasil pengujian dapat dilihat pada Tabel 6.3

Tabel 6.3 Hasil pengujian 10 *user* yang berbeda

User- **Ke**	***Banyak Kata***	**Kata Masukan**	**Kata Keluaran**	**Jumlah gerakan benar**	**Jumlah gerakan salah**
1	*1*	maju	maju	1	0
	2	mundur maju	mundur maju	2	0
	3	kiri kanan landing	kiri kanan landing	3	0
	4	terbang kiri kanan landing	terbang kiri canon landing	3	1
	5	terbang maju mundur kiri kanan	terbang maju mundur kiri kanan	5	0
2	*1*	kiri	kiri	1	0
	2	maju landing	maju landing	2	0
	3	terbang kiri landing	terbang kiri landing	3	0
	4	terbang kiri kanan maju	terbang kiri kanan maju	4	0
	5	terbang naik turun kanan landing	terbang naik turun tanan wedding	3	2
3	*1*	kanan	kanan	1	0
	2	terbang maju	terbang maju	2	0
	3	terbang kiri kanan	terbang kiri kanan	3	0
	4	terbang kanan kanan kiri	terbang kanan kanan kiri	4	0
	5	kanan kiri maju naik landing	kanan kiri maju naik banding	4	1
4	*1*	kanan	kanan	1	0
	2	mundur kanan	mundur cannon	1	1

Tabel 6.3 Hasil pengujian 10 *user* yang berbeda Lanjutan

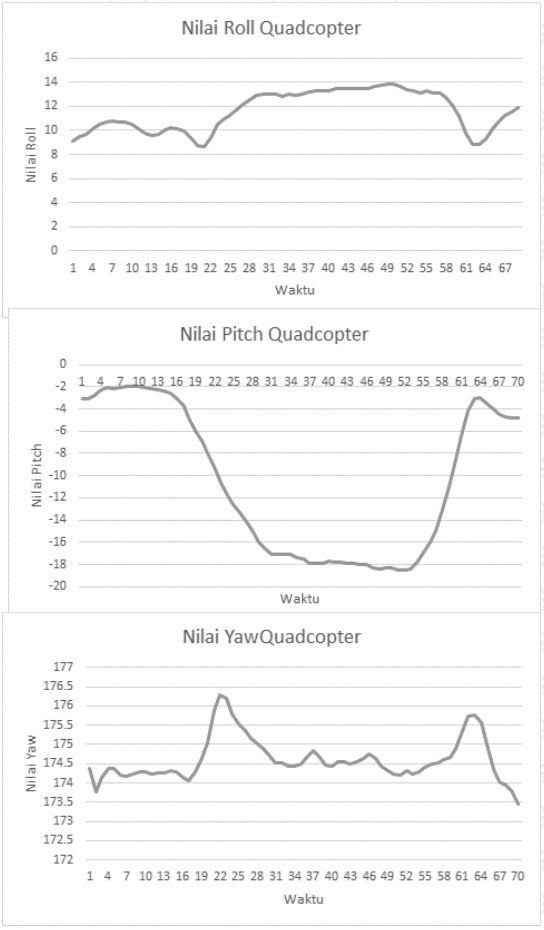
User-Ke	*Banyak Kata*	Kata Masukan	Kata keluaran	Jumlah gerakan benar	Jumlah gerakan salah
	3	kanan kiri landing	kanan kiri landing	3	0
	4	terbang kiri kanan naik	terbang kiri kanan naik	4	0
	5	terbang kanan kiri atas landing	terbang kanan kiri atas landing	5	0
5	*1*	kanan	kanan	1	0
	2	sedikit kanan	sedikit kanan	2	0
	3	terbang kanan landing	terbang kanan bandung	2	1
	4	terbang kanan maju landing	terbang kanan maju wedding	3	1
	5	terbang kanan sedikit kiri landing	terbang kanan sedikit kiri landing	5	0
6	*1*	maju	maju	1	0
	2	maju atas	maju patas	1	1
	3	terbang maju mundur	terbang maju mundur	3	0
	4	terbang maju kiri kanan	terbang maju kiri kanan	4	0
	5	terbang maju kiri kanan landing	terbang maju kiri kanan landing	5	0
7	*1*	kanan	kanan	1	0
	2	terbang maju	terbang maju	2	0
	3	terbang naik kiri	terbang naik ciri	2	1
	4	kanan dua detik landing	kanan dua detik landing	4	0
	5	terbang naik kiri kanan landing	terbang naik kiri kanan landing	5	0
8	*1*	maju	maju	1	0
	2	kanan atas	kanan atas	2	0
	3	terbang kiri maju	terbang kiri maju	3	0

Tabel 6.3 Hasil pengujian 10 *user* yang berbeda Lanjutan

User-Ke	*Banyak Kata*	Kata Masukan	Kata Keluaran	Jumlah gerakan benar	Jumlah gerakan salah
	4	terbang kanan ke atas	terbang kanan ke atas	4	0
	5	terbang atas mundur kiri landing	terbang atas mundur ciri wedding	3	2
9	1	landing	landing	1	0
	2	maju atas	maju atas	2	0
	3	terbang kiri landing	terbang kiri banding	2	1
	4	terbang kanan kiri landing	terbang kanan ciri landing	3	1
	5	maju mundur kanan kiri landing	maju budur cannon ciri landing	2	3
10	1	maju	maju	1	0
	2	kiri landing	kiri landing	2	0
	3	terbang maju mundur	terbang maju mundur	3	0
	4	terbang kiri kanan maju	terbang kiri kanan maju	4	0
	5	maju dua detik mundur landing	maju 2 detik mundur landing	5	0
Jumlah				134	16

Pada Tabel 6.3 dapat dilihat hasil 10 responden pada pengujian ini. Nilai pergerakan benar sebanyak 134 gerakan, sedangkan pergerakan salah sebanyak 16 gerakan salah. Nilai gerakan salah dari hasil pengujian terletak pada kata keluaran yang dihasilkan *speech recognition* pada Android sehingga hasil pencocokan *database* tidak terdeteksi.

Dari hasil percobaan yang dilakukan diambil nilai data terbang *quadcopter*. Data yang diambil berupa nilai sudut yang dibentuk *quadcopter*, yaitu sudut *roll, pitch, dan yaw* salah satu data hasil pengujian yang dilakukan dapat dilihat pada Gambar 6.16.



Gambar 6.16 Grafik Pengujian *User* 1 dengan masukan 1 kata

Pada Gambar 6.16 dapat dilihat data yang dihasilkan oleh *quadcopter* dengan masukan suara dari *user* 1 dan masukan 1 kata. Grafik gerakan quadcopter lebih dapat dilihat pada lampiran B.1.

6.3.5 Analisis pengujian

Dari hasil pengujian dengan 10 responden, dapat dihitung nilai akurasi persentase tingkat kesalahan dari sistem yang dibuat. Rumus menghitung nilai persentase akurasi dapat dilihat pada rumus 2.1.

Dengan hasil penghitungan sebagai berikut.

$$\text{Nilai Persentase Akurasi} = \frac{134}{150} \text{X } 100\% = 89.3\%$$

Sistem ini memiliki akurasi sebesar 89.3%, hasil pergerakan salah merupakan kesalah pada deteksi suara menjadi teks. Namun dapat disimpulkan sistem ini memiliki akurasi yang sangat baik.

Pada Gambar 6.16 grafik pengujian *user* 1 dengan perintah "maju" menghasilkan nilai sudut *roll* dan *yaw* yang tidak berubah cukup signifikan, sedangkan sudut *pitch* berubah cukup besar. Hal ini dikarenakan *quadcopter* bergerak maju ke depan, sehingga sudut yang dibentuk menurut sumbu Y atau

sumbu *pitch* akan berubah cukup signifikan. Gambar grafik untuk perintah lebih lengkap dapat dilihat pada lampiran B.1.

6.4 Pengujian Performa Aplikasi

6.4.1 Tujuan pengujian

Tujuan dari pengujian ini untuk mengetahui *delay* dari sistem yang dibuat pada Android, berapa waktu yang dibutuhkan sejak masukan suara dari *user* sampai melakukan *Text preprocessing*.

6.4.2 Prosedur pengujian

Aplikasi yang sudah dibuat sebelumnya akan ditambah penghitungan waktu dengan memanfaatkan *millis* dan sistem *nano time* yang ada pada Android studio. Penghitungan waktu untuk pengolahan suara menggunakan *millis*, sedangkan penghitungan waktu untuk *text preprocessing* menggunakan *nano second*. Gambar aplikasi yang telah dibuat dengan menambahkan waktu untuk penghitungan *delay* aplikasi dapat dilihat pada lampiran C.1.

6.4.3 Pelaksanaan pengujian

User menekan tombol "ngomong" lalu *timer* secara otomatis berjalan dan berhenti. Waktu yang dihasilkan dari aplikasi akan dicatat dan dianalisis.

6.4.4 Hasil Pengujian

Hasil pengujian *delay* dari sistem ini berupa *timer* dengan hitungan *milisecond* dapat dilihat pada Tabel 6.4 sampai Tabel 6.8

Tabel 6.4 Pengujian *delay* dengan masukan 1 kata

user Ke-	Kata	Waktu Pengolahan Suara/Mili Detik	Waktu *Text preprocessing*/Nano detik
1	maju	3125	2448
2	kiri	3303	3115
3	kanan	2975	3230
4	kanan	2977	2865
5	kanan	3169	3437
6	maju	3245	3333
7	kanan	3115	2709
8	maju	3229	2596
9	landing	3248	3146
10	laju	3219	3148
Jumlah		31605	30027
Rata-Rata		3160,5	3002,7

Pada Tabel 6.4 merupakan pengujian *delay* dengan memberikan masukan perintah sebanyak 1 kata, dengan melakukan 10 percobaan. Waktu yang diperoleh dari pengolahan suara memiliki rata-rata selama 3,16 detik dan rata-rata *Text preprocessing* selama 3002,7 *nano* detik.

Tabel 6.5 Pengujian *delay* dengan masukan 2 kata

user ke-	Kata	Waktu Pengolahan Suara/Mili Detik	Waktu *Text preprocessing*/Nano detik
1	maju mundur	3775	2656
2	maju landing	4009	5312
3	terbang maju	4093	3073
4	mundur cannon	3825	2969
5	sedikit kanan	3244	3906
6	maju atas	4842	2396
7	terbang maju	3896	3907
8	kanan atas	4047	2656
9	maju atas	4735	2812
10	kiri landing	4452	3385
Jumlah		40918	33072
Rata-Rata		4091,8	3307,2

Pada Tabel 6.5 merupakan pengujian *delay* dengan memberikan masukan perintah sebanyak 2 kata dengan 10 kali percobaan. Waktu yang diperoleh dari pengolahan suara memiliki rata-rata selama 4,09 detik dan rata-rata *text preprocessing* selama 3307,2 *nano* detik.

Tabel 6.6 Pengujian *delay* dengan masukan 3 kata

Data Ke	Kata	Waktu Pengolahan Suara/Mili Detik	Waktu *Text preprocessing*/Nano detik
1	kiri kanan landing	4618	3477
2	terbang kiri landing	4534	3648
3	terbang kiri kanan	4657	3360
4	kanan kiri landing	4879	3489
5	terbang kanan landing	4976	3765
6	terbang maju mundur	4943	3838
7	terbang naik kiri	4936	3509
8	terbang kiri maju	4704	3429
9	terbang kiri landing	4975	3646
10	terbang maju mundur	5249	3542
Jumlah		48471	35703
Rata-Rata		4847,1	3570,3

Pada Tabel 6.6 merupakan pengujian *delay* dengan memberikan masukan perintah sebanyak 3 kata dengan 10 kali percobaan. Waktu yang diperoleh dari

pengolahan suara memiliki rata-rata selama 4,84 detik dan rata-rata *text preprocessing* selama 3570,3 *nano* detik.

Tabel 6.7 Pengujian *delay* dengan masukan 4 kata

User Ke-	Kata	Waktu Pengolahan Suara/Mili Detik	Waktu *Text preprocessing*/Nano detik
1	terbang kiri kanan landing	6166	3777
2	terbang kiri kanan maju	6156	3864
3	terbang kanan kanan kiri	5977	3812
4	terbang kiri kanan naik	5832	3968
5	terbang kanan maju landing	6059	4844
6	terbang maju kiri kanan	5331	3656
7	kanan dua detik landing	6026	3869
8	terbang kanan ke atas	5609	3665
9	terbang kanan kiri landing	5989	3802
10	terbang kiri kanan maju	6200	3760
Jumlah		59345	39017
Rata-Rata		5934,5	3901,7

Pada Tabel 6.7 merupakan pengujian *delay* dengan memberikan masukan perintah sebanyak 4 kata dengan 10 kali percobaan. Waktu yang diperoleh dari pengolahan suara memiliki rata-rata selama 5,93 detik dan rata-rata *text preprocessing* selama 3901,7 *nano* detik.

Tabel 6.8 Pengujian *delay* dengan masukan 5 kata

User Ke-	Kata	Waktu Pengolahan Suara/Mili Detik	Waktu *Text preprocessing*/ Nano detik
1	terbang maju mundur kiri kanan	7177	4895
2	terbang naik turun kanan landing	7082	4145
3	kanan kiri maju naik landing	7102	3760
4	terbang kanan kiri atas landing	7052	4241
5	terbang kanan sedikit kiri landing	6413	4792
6	terbang maju kiri kanan landing	6538	4594
7	terbang naik kiri kanan landing	7584	4285
8	terbang atas mundur kiri landing	7533	3969
9	maju mundur kanan kiri landing	7410	4098
10	maju dua detik mundur landing	7017	3917
Jumlah		70908	42696
Rata-Rata		7090.8	4269,6

Pada Tabel 6.8 merupakan pengujian *delay* dengan memberikan masukan perintah sebanyak 4 kata dengan 10 kali percobaan. Waktu yang diperoleh dari

pengolahan suara memiliki rata-rata selama 7.09 detik dan rata-rata *Text preprocessing* selama 4269,6 *nano* detik.

6.4.5 Analisis Pengujian

Dari data pengujian yang telah dilakukan didapat Rata-rata waktu yang dibutuhkan di antaranya, pengolahan masukan suara satu kata masukan selama 3,16 detik, 2 kata masukan selama 4,09, 3 kata masukan selama 4,84 , 4 kata masukan selama 5,93 detik, dan 5 kata masukan selama 7.09. Sehingga dapat dianalisis bahwa waktu yang dibutuhkan untuk pengolahan suara akan bertambah sebesar ±1 detik sesuai dengan banyaknya kata yang diberikan.

Waktu yang dibutuhkan untuk pengolahan *text preprocessing* dengan masukan 1 kata membutuhkan 3002,7 nano detik, 2 kata masukan selama 3307,2 nano detik, waktu yang 3 kata masukan selama 3570,3 nano detik, waktu yang dibutuhkan untuk pengolahan 4 kata masukan selama 3901,7 nano detik, 5 kata masukan selama 4269,6 nano detik. Sehingga dapat dianalisis Sedangkan waktu yang dibutuhkan untuk pengolahan *text preprocessing* akan bertambah sebesar ±300 *nano second* seiring dengan banyaknya kata masukan dari *user*.

Berdasarkan 5 jenis pengujian di atas membuktikan bahwa waktu yang dibutuhkan untuk melakukan *text preprocessing* sangat cepat dan dan akan bertambah seiring dengan banyaknya kata. Sedangkan waktu yang dibutuhkan untuk pengolahan suara juga akan bertambah terus menerus seiring dengan banyaknya kata yang diberikan.

6.5 Pengujian Performa *Quadcopter*

6.5.1 Tujuan pengujian

Tujuan dari pengujian ini untuk mengetahui performa terbang dari *quadcopter*.

6.5.2 Prosedur pengujian

Performa terbang diuji dengan memanfaatkan data navigasi yang ada pada *quadcopter*. Data navigasi yang diuji berupa perubahan nilai sudut *roll*,*pitch*, *yaw*, dan nilai *altitude* meter nilai yang didapat akan disajikan dalam bentuk grafik. Pembuatan grafik secara *real time* menggunakan *cloud plot.ly*, dan pembuatan grafik secara *offline* menggunakan *Microsoft excel*.

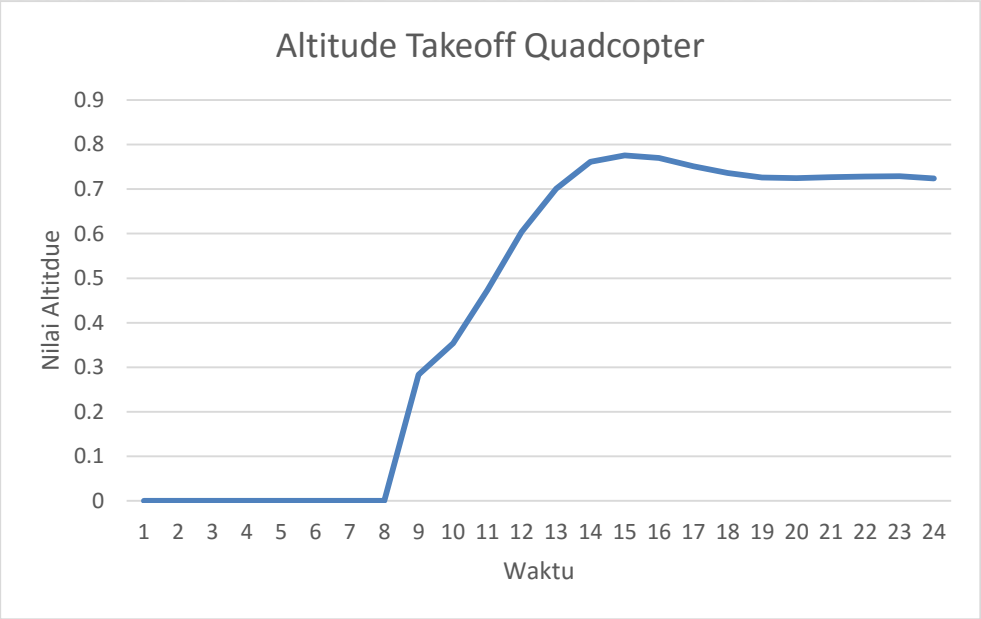
6.5.3 Pelaksanaan pengujian

Nilai data navigasi diambil sejak *quadcopter* bergerak, lalu diberikan perintah untuk satu kali *state*. Nilai grafik Y merupakan perubahan nilai terbang quadcopter yang diuji, sedangkan nilai x merupakan waktu yang dibutuhkan untuk mengambil data dengan satuan 1/10 detik.

6.5.4 Hasil pengujian

a. Gerakan *takeoff quadcopter*

Quadcopter diuji Dengan perintah "terbang" lalu akan diambil nilai perubahan *altitude* meter, yaitu nilai ketinggian dari *quadcopter*. Hasil pengujian dapat dilihat pada Gambar 6.17.



Gambar 6.17 Grafik nilai ketinggian saat *quadcopter takeoff*

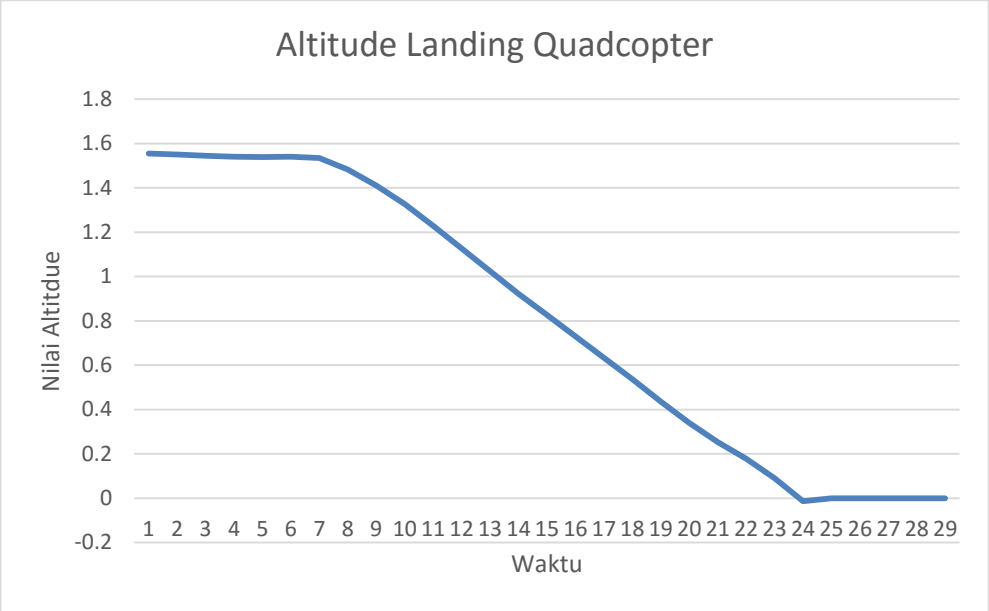
Pada Gambar 6.17 merupakan hasil pengujian *altitude* yang diambil ketika *quadcopter takeoff*. Nilai ketinggian *quadcopter* saat *takeoff* sebesar 0.7 meter sampai 0.8 meter, penerapan *quadcopter* saat *takeoff* dapat dilihat pada Gambar 6.18.

Gambar 6.18 *Quadcopter* saat *takeoff*

Pada Gambar 6.18 dapat dilihat *quadcopter* saat melakukan *takeoff*. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

b. Gerakan *landing quadcopter*

Quadcopter diuji Dengan perintah "*landing*" lalu akan diambil nilai perubahan *altitude* meter, yaitu nilai ketinggian dari *quadcopter*. Hasil pengujian dapat dilihat pada Gambar 6.19.



Gambar 6.19 Grafik nilai ketinggian saat *quadcopter landing*

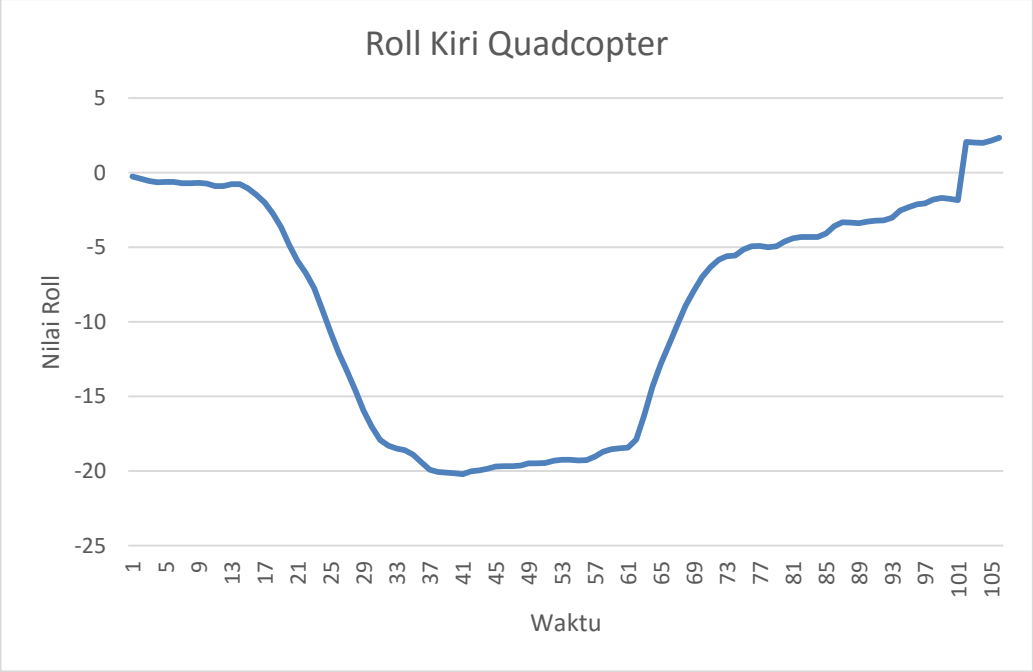
Pada Gambar 6.19 merupakan hasil pengujian *altitude* yang diambil ketika *quadcopter landing.* Nilai ketinggian *quadcopter* saat *landing* akan terus berkurang sampai *quadcopter* menyentuh tanah dengan nilai ketinggian sebesar 0, penerapan *quadcopter* saat *landing* dapat dilihat pada Gambar 6.20.

Gambar 6.20 *Quadcopter* saat *landing*

Pada Gambar 6.20 dapat dilihat *quadcopter* saat *landing*. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

c. Gerakan kiri *quadcopter*

Quadcopter diuji Dengan perintah "kiri" lalu akan diambil nilai perubahan sudut x, yaitu nilai sudut *roll*. Hasil pengujian dapat dilihat pada Gambar 6.21.



Gambar 6.21 Grafik nilai sudut *roll* saat *quadcopter bergerak* ke kiri

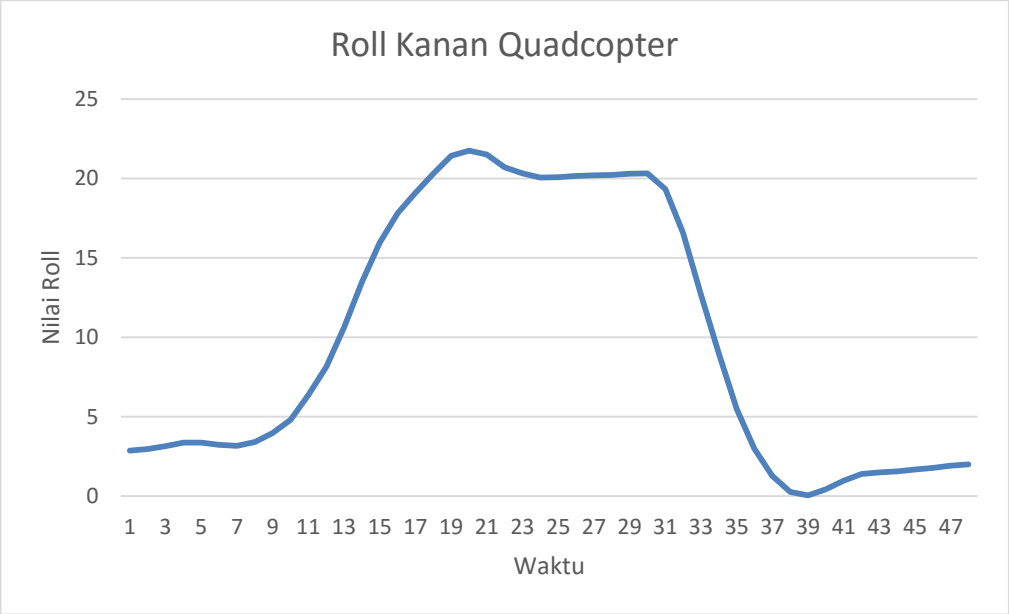
Pada Gambar 6.21 merupakan hasil pengujian nilai *roll* yang diambil ketika *quadcopter* bergerak ke arah kiri. Nilai sudut menurut sumbu x pada *quadcopter* saat ke kiri akan bernilai negatif. Perubahan derajat sudut tergantung dari seberapa cepat *quadcopter* bergerak, penerapan *quadcopter* saat ke kiri dapat dilihat pada Gambar 6.22.

Gambar 6.22 *Quadcopter* saat ke kiri

Pada Gambar 6.22 dapat dilihat *quadcopter* saat bergerak ke arah kiri. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

d. Gerakan kanan *quadcopter*

Quadcopter diuji Dengan perintah "kanan" lalu akan diambil nilai perubahan sudut x, yaitu nilai sudut *roll*. Hasil pengujian dapat dilihat pada Gambar 6.23.



Gambar 6.23 Grafik nilai sudut *roll* saat *quadcopter bergerak* ke kanan

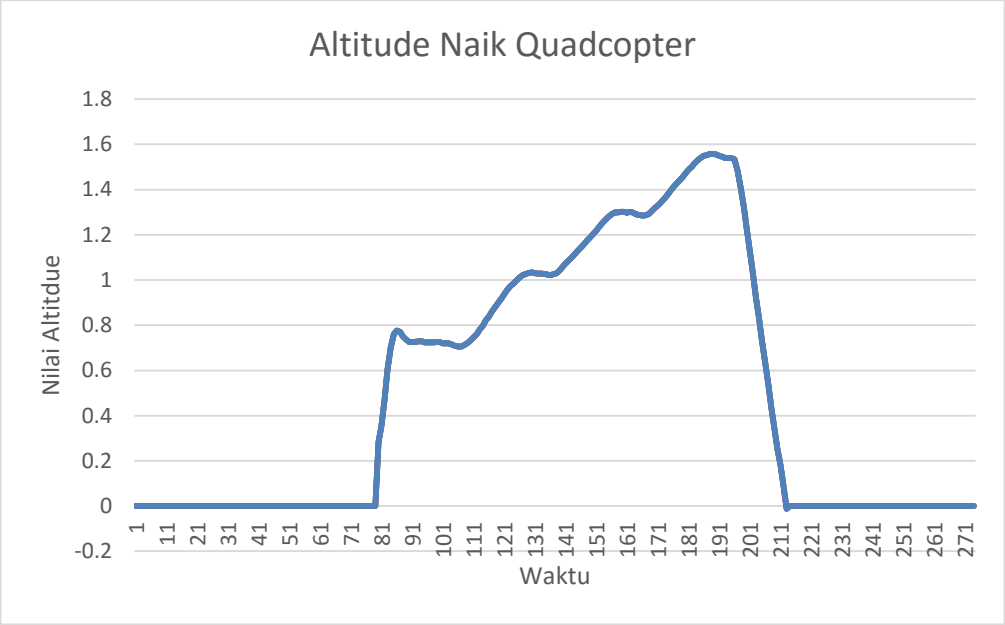
Pada Gambar 6.23 merupakan hasil pengujian nilai *roll* yang diambil ketika *quadcopter* bergerak ke kanan. Nilai sudut menurut sumbu x pada *quadcopter* saat ke kanan akan bernilai positif. Perubahan derajat sudut tergantung dari seberapa cepat *quadcopter* bergerak, penerapan *quadcopter* saat ke kanan dapat dilihat pada Gambar 6.24.

Gambar 6.24 quadcoter saat ke kanan

Pada Gambar 6.24 dapat dilihat *quadcopter* saat bergerak ke arah kanan. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

e. Gerakan naik *quadcopter*

Quadcopter diuji Dengan perintah "naik" lalu akan diambil nilai *altitude meter*, yaitu nilai ketinggian dari *quadcopter*. hasil pengujian dapat dilihat pada Gambar 6.25.



Gambar 6.25 Grafik nilai altitude saat *quadcopter bergerak* naik ke atas

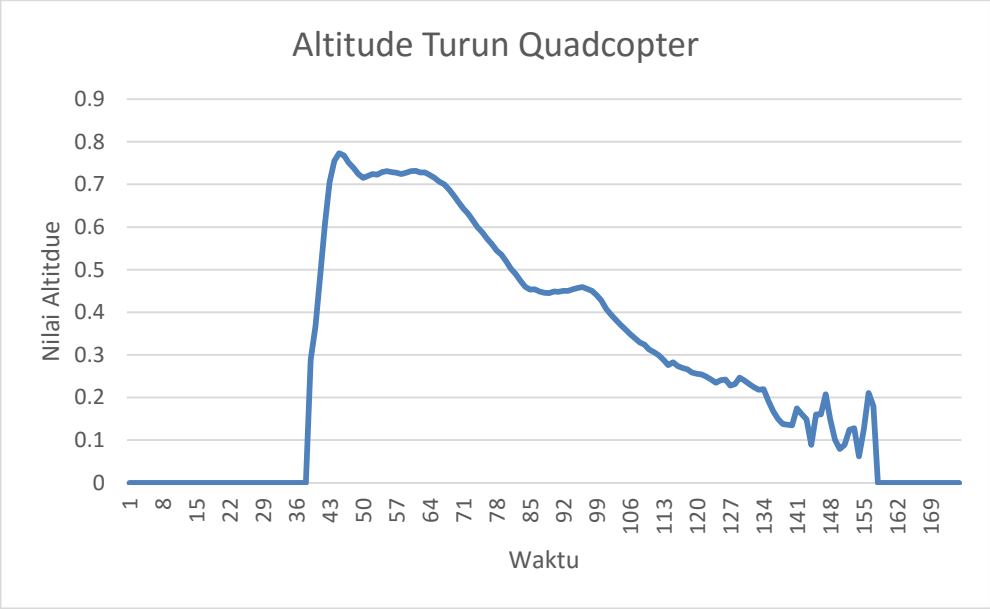
Pada Gambar 6.25 merupakan hasil pengujian nilai *altitude* yang diambil ketika *quadcopter* naik ke atas. Nilai ketinggian *quadcopter* saat naik akan terus bertambah seiring dengan banyaknya perintah naik. penerapan *quadcopter* saat naik ke atas dapat dilihat pada Gambar 6.26.

Gambar 6.26 *quadcopter* saat naik ke atas

Pada Gambar 6.26 dapat dilihat *quadcopter* saat bergerak ke arah atas. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

f. Gerakan turun *quadcopter*

Quadcopter diuji Dengan perintah "turun" lalu akan diambil nilai *altitude* meter, yaitu nilai ketinggian dari *quadcopter*. hasil pengujian dapat dilihat pada Gambar 6.27.



Gambar 6.27 Grafik nilai ketinggian saat *quadcopter* turun ke bawah

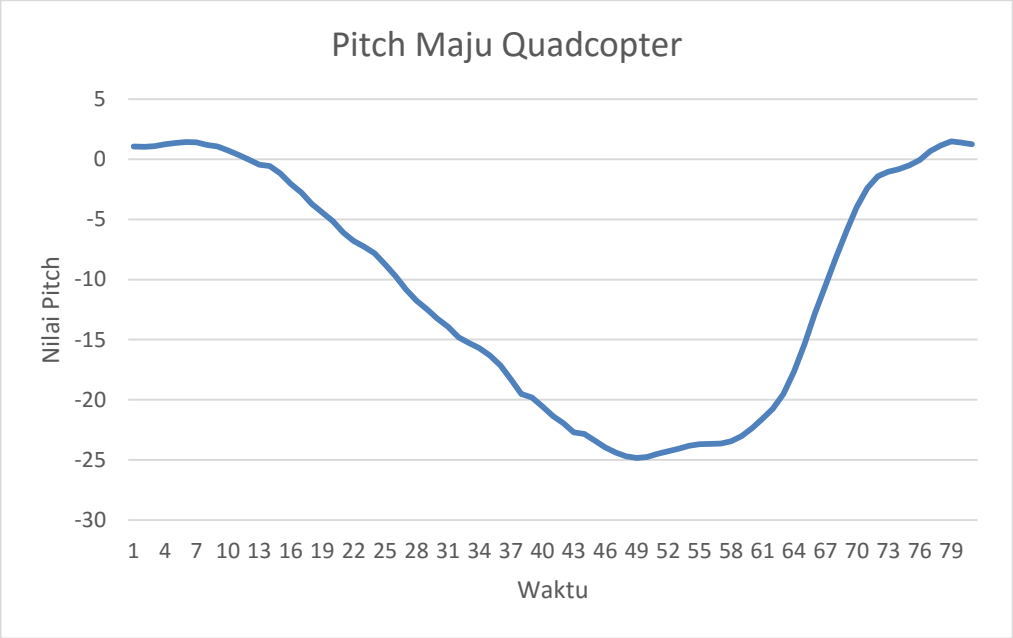
Pada Gambar 6.27 merupakan hasil pengujian *altitude* yang diambil ketika *quadcopter* turun. Nilai ketinggian *quadcopter* saat turun akan terus berkurang sampai *quadcopter* hampir menyentuh tanah dengan nilai minimum 0.062 meter, penerapan *quadcopter* saat turun ke bawah dapat dilihat pada Gambar 6.28.

Gambar 6.28 *Quadcopter* saat turun ke bawah

Pada Gambar 6.228 dapat dilihat *quadcopter* saat bergerak ke arah bawah. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

g. Gerakan maju *quadcopter*

Quadcopter diuji Dengan perintah "maju" lalu akan diambil nilai perubahan sudut y, yaitu nilai sudut *pitch*. Hasil pengujian dapat dilihat pada Gambar 6.29.



Gambar 6.29 Grafik nilai sudut *pitch* saat *quadcopter bergerak* maju

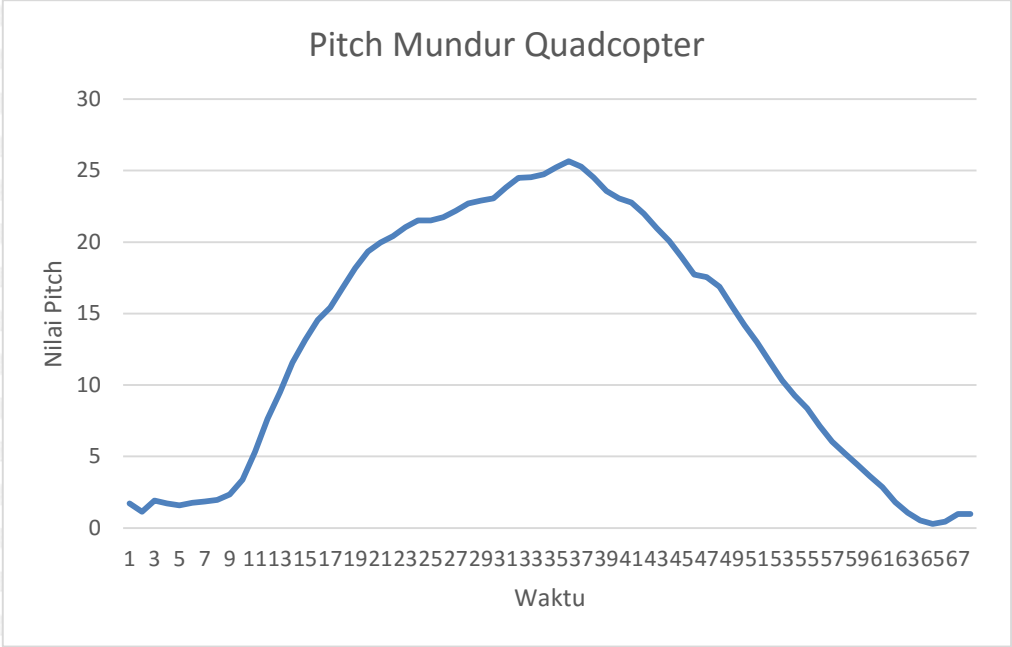
Pada Gambar 6.29 merupakan hasil pengujian nilai *pitch* yang diambil ketika *quadcopter* bergerak maju. Nilai sudut menurut sumbu y pada *quadcopter* saat maju akan bernilai negatif. Perubahan derajat sudut tergantung dari seberapa cepat *quadcopter* bergerak, penerapan *quadcopter* saat bergerak maju ke depan dapat dilihat pada Gambar 6.30.

Gambar 6.30 *Quadcopter* saat bergerak maju

Pada Gambar 6.30 dapat dilihat *quadcopter* saat bergerak ke arah depan. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

h. Gerakan mundur *quadcopter*

Quadcopter diuji Dengan perintah "mundur" lalu akan diambil nilai perubahan sudut y, yaitu nilai sudut *pitch*. Hasil pengujian dapat dilihat pada Gambar 6.31.



Gambar 6.31 Grafik nilai sudut *Pitch* saat *quadcopter* bergerak mundur

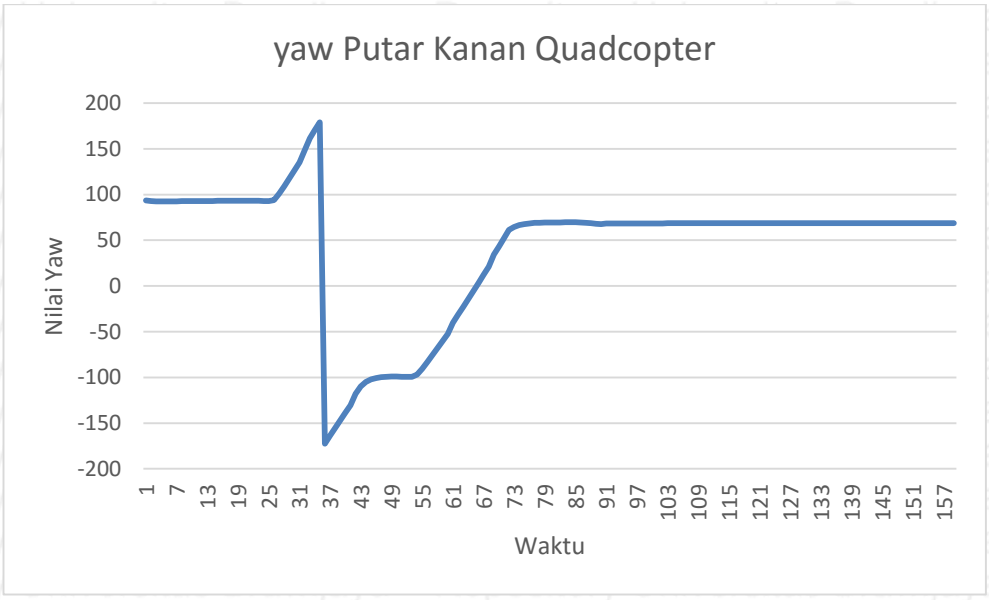
Pada Gambar 6.31 merupakan hasil pengujian nilai *pitch* yang diambil ketika *quadcopter bergerak* mundur. Nilai sudut menurut sumbu y pada *quadcopter* saat mundur akan bernilai positif. Perubahan derajat sudut tergantung dari seberapa cepat *quadcopter* bergerak, penerapan *quadcopter* saat mundur ke belakang dapat dilihat pada Gambar 6.32.

Gambar 6.32 *Quadcopter* saat bergerak mundur

Pada Gambar 6.32 dapat dilihat *quadcopter* saat bergerak ke arah belakang. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

i. Gerakan putar kanan *quadcopter*

Quadcopter diuji Dengan perintah "putar kanan" lalu akan diambil nilai perubahan sudut z, yaitu nilai sudut *yaw*. Hasil pengujian dapat dilihat pada Gambar 6.33.



Gambar 6.33 Grafik nilai sudut *yaw* saat *quadcopter* berputar ke kanan

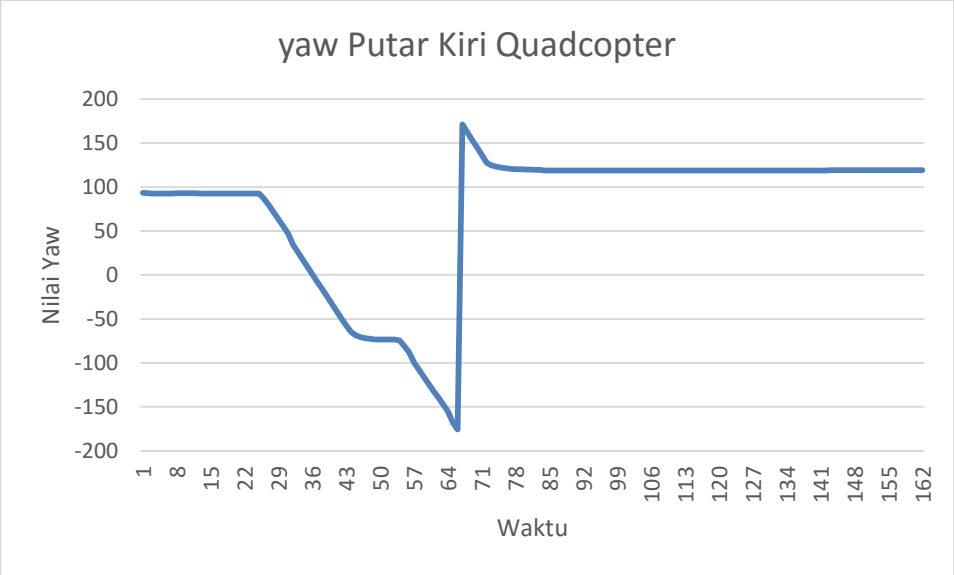
Pada Gambar 6.33 merupakan hasil pengujian nilai *yaw* yang diambil ketika *quadcopter* berputar ke kanan. Nilai derajat sudut menurut sumbu z pada *quadcopter* saat berputar ke kanan akan bertambah tergantung dari seberapa banyak *quadcopter* berputar, penerapan *quadcopter* saat berputar ke kanan dapat dilihat pada Gambar 6.34.

Gambar 6.34 *Quadcopter* saat berputar ke kanan

Pada Gambar 6.34 dapat dilihat *quadcopter* saat bergerak berputar ke arah kanan. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

j. Gerakan putar kiri *quadcopter*

Quadcopter diuji Dengan perintah "putar kiri" lalu akan diambil nilai perubahan sudut z, yaitu nilai sudut *yaw*. Hasil pengujian dapat dilihat pada Gambar 6.35.



Gambar 6.35 Grafik nilai sudut *yaw* saat *quadcopter* berputar ke kiri

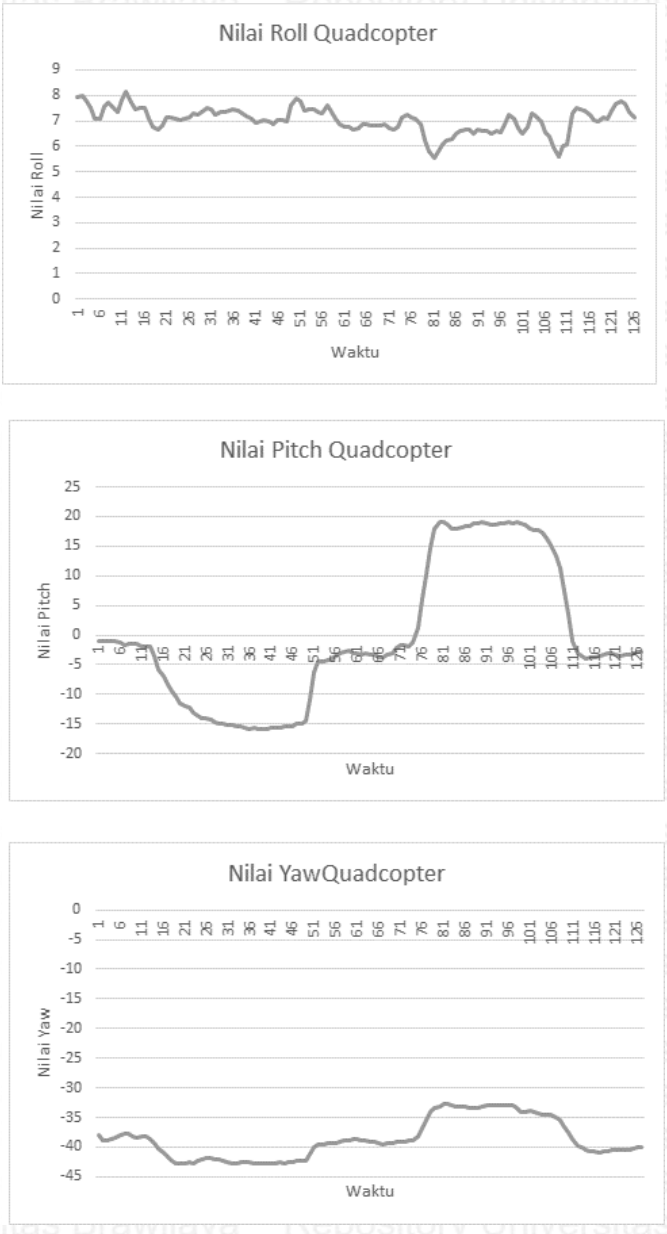
Pada Gambar 6.35 merupakan hasil pengujian nilai *yaw* yang diambil ketika *quadcopter* berputar ke kiri. Nilai derajat sudut menurut sumbu z pada *quadcopter* saat berputar ke kiri akan berkurang tergantung dari seberapa banyak *quadcopter* berputar, penerapan *quadcopter* saat berputar ke kiri dapat dilihat pada Gambar 6.36.

Gambar 6.36 *Quadcopter* saat berputar ke kiri

Pada Gambar 6.36 dapat dilihat *quadcopter* saat bergerak berputar ke arah kiri. *Quadcopter* menghasilkan perubahan nilai data navigasi yang beragam sesuai dengan *state* gerakan yang sedang dilakukan.

k. Gerakan 2 arah quadcopter yang berbeda

Quadcopter diuji dengan perintah "maju mundur" lalu diambil nilai *rol*, *pitch,* dan *yaw* dalam bentuk grafik yang dapat dilihat pada Gambar 6.37.

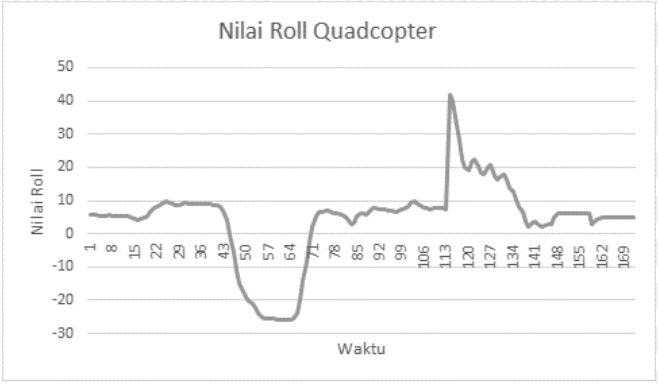


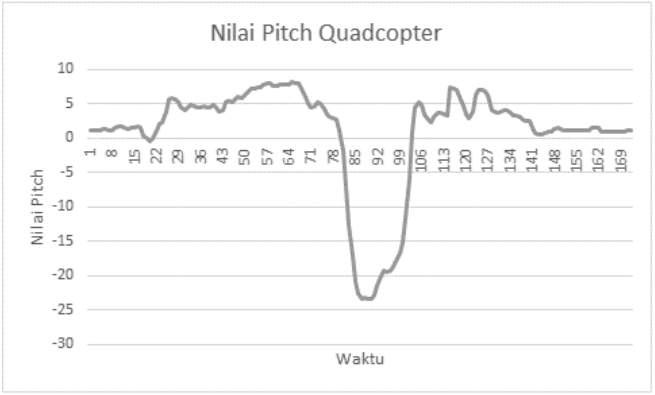
Gambar 6.37 Hasil Pengujian 2 Perintah

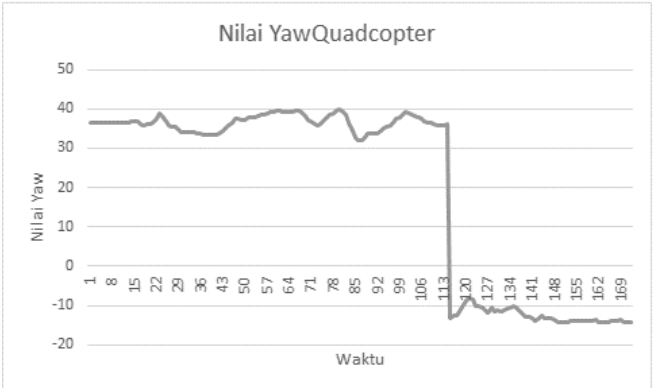
Pada Gambar 6.37 hasil pengujian pergerakan dengan 2 perintah menghasilkan perubahan nilai sudut *rol*, *pitch,* dan *yaw* yang berbeda tergantung state pergerakan quadcopter yang sedang dilakukan. Ketika bergerak mundur dan maju nilai sudut *Roll* dan sudut *yaw* berubah dengan range nilai yang tidak terlalu signifikan, sedangkan nilai sudut *pitch* berubah dengan signifikan.

l. Gerakan 3 arah quadcopter yang berbeda

Quadcopter diberikan perintah “kiri maju landing” lalu diambil nilai *roll, pitch,* dan *yaw* dalam bentuk grafik yang dapat dilihat pada Gambar 6.38.





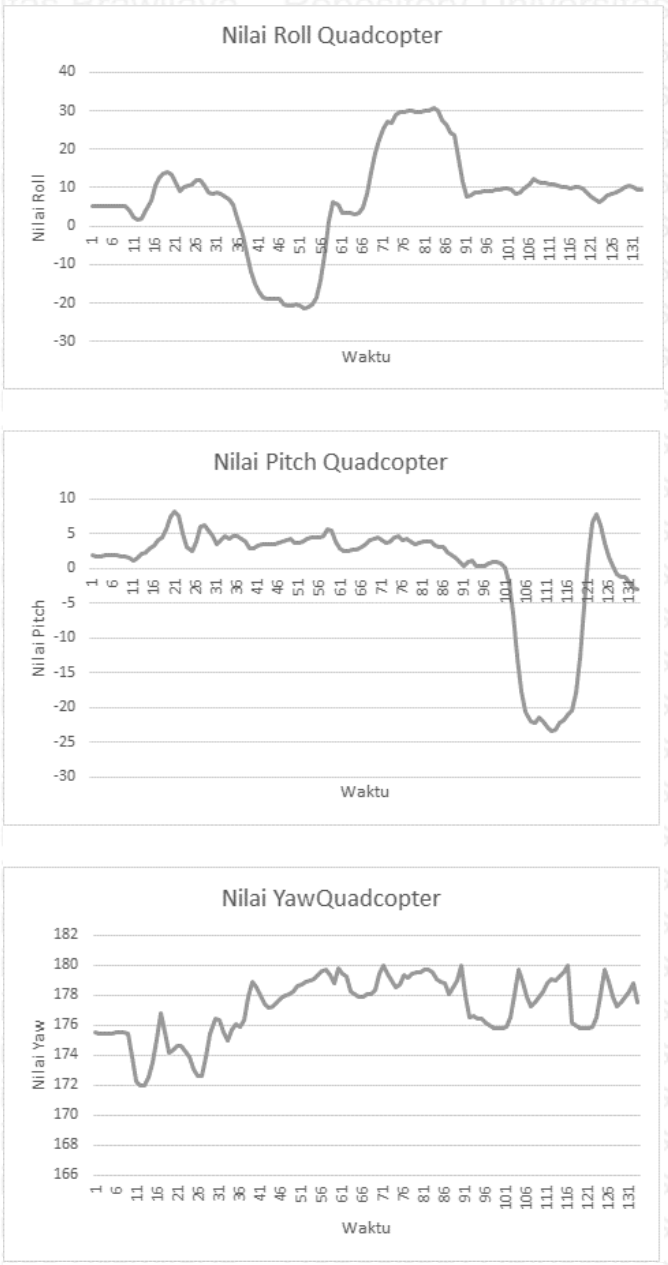


Gambar 6.38 Hasil Pengujian 3 Perintah

Pada Gambar 6.38 hasil pengujian pergerakan dengan 3 perintah menghasilkan perubahan nilai sudut *rol*, *pitch,* dan *yaw* yang berbeda tergantung *state* pergerakan quadcopter yang sedang dilakukan. Ketika bergerak ke kiri nilai sudut *Roll* berubah dengan range nilai yang signifikan, sedangkan nilai sudut *pitch* berubah dengan signifikan ketika quadcopter bergerak ke arah depan. Ketika *landing* semua nilai sudut akan berubah.

m. Gerakan 4 arah quadcopter yang berbeda

Quadcopter diberikan perintah "terbang kiri kanan maju" lalu diambil nilai *rol*, *pitch,* dan *yaw* dalam bentuk grafik yang dapat dilihat pada Gambar 6.39.

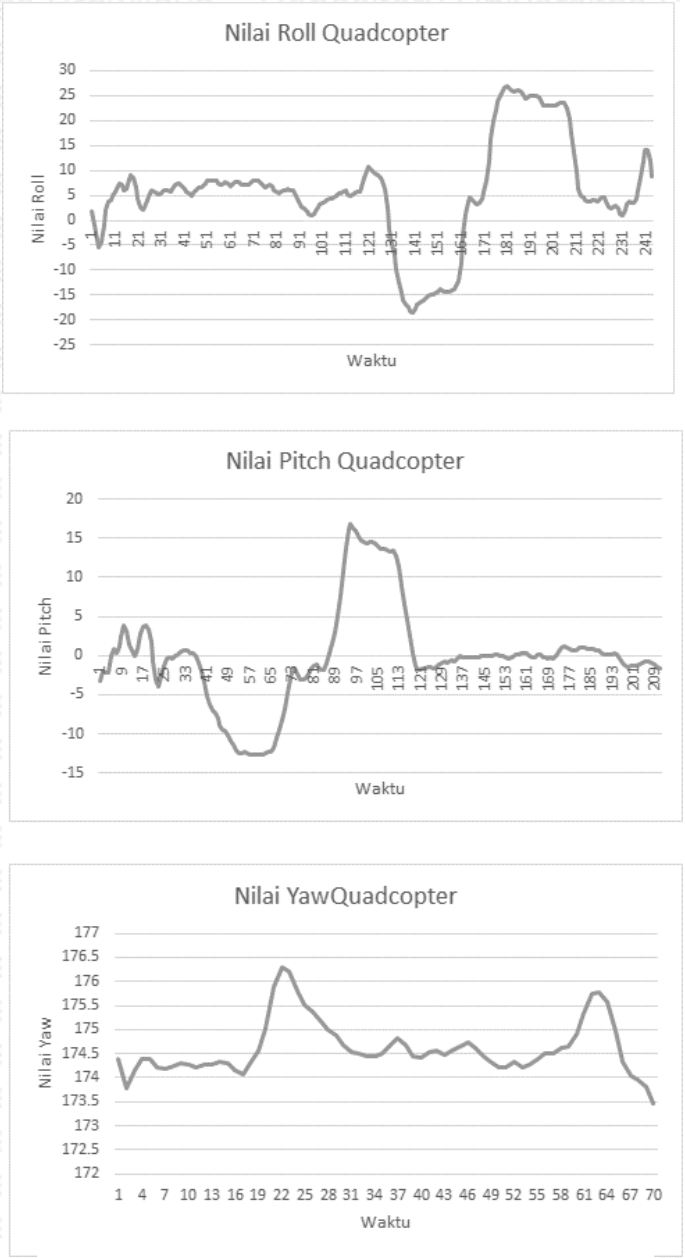


Gambar 6.39 Hasil Pengujian 4 Perintah

Pada Gambar 6.39 hasil pengujian pergerakan dengan 4 perintah menghasilkan perubahan nilai sudut *rol*, *pitch,* dan *yaw* yang berbeda tergantung *state* pergerakan quadcopter yang sedang dilakukan. Ketika bergerak ke kiri dan ke kanan nilai sudut *Roll* berubah dengan range nilai yang signifikan, sedangkan nilai sudut *pitch* berubah dengan signifikan ketika quadcopter bergerak ke arah depan. Ketika terbang semua nilai sudut akan berubah.

n. Gerakan 5 arah quadcopter yang berbeda

quadcopter diberikan perintah “terbang maju mundur kiri kanan” lalu diambil nilai *rol*, *pitch,* dan *yaw* dalam bentuk grafik yang dapat dilihat pada Gambar 6.40.



Gambar 6.40 Hasil Pengujian 5 Perintah

Pada Gambar 6.40 hasil pengujian pergerakan dengan 5 perintah menghasilkan perubahan nilai sudut *rol*, *pitch,* dan *yaw* yang berbeda tergantung *state* pergerakan quadcopter yang sedang dilakukan. Ketika bergerak ke kiri dan ke kanan nilai sudut *roll* berubah dengan range nilai yang signifikan, sedangkan nilai sudut *pitch* berubah dengan signifikan ketika quadcopter bergerak ke arah depan dan belakang. Ketika terbang semua nilai sudut akan berubah.

6.5.5 Analisis pengujian

Nilai data navigasi yang diambil dari *quadcopter* dengan berbagai gerakan akan menghasilkan nilai yang berbeda-beda. Gerakan *quadcopter* saat bergerak ke kanan atau ke kiri akan menghasilkan nilai sudut *Roll*, saat bergerak ke arah depan atau belakang akan menghasilkan nilai sudut *Pitch*, ketika bergerak berputar ke kanan maupun ke kiri akan menghasilkan nilai sudut *Yaw*, dan ketika bergerak naik atau turun, *takeoff atau landing* akan menghasilkan nilai ketinggian atau *altitude* meter.

Gerakan *quadcopter* saat ke kanan akan menghasilkan sudut *Roll* yang bernilai negatif, sedangkan gerakan *quadcopter* saat bergerak ke arah kiri akan menghasilkan sudut *Roll* yang bernilai positif. Gerakan *quadcopter* saat bergerak maju akan menghasilkan sudut *Pitch* yang bernilai positif, sedangkan gerakan *quadcopter* saat bergerak mundur akan menghasilkan sudut *Pitch* yang bernilai negatif. Gerakan *quadcopter* saat berputar ke kanan akan menghasilkan gerakan nilai sudut *yaw* yang akan terus bertambah , sedangkan ketika *quadcopter* bergerak berputar ke kiri akan menghasilkan sudut *yaw* yang akan terus berkurang. Nilai ketinggian diambil sejak *quadcopter takeoff* ,nilai ini bertambah jika diberikan perintah "naik", dan akan berkurang jika diberikan perintah "turun" atau "*landing*".

Dari analisis sudut yang yang diperoleh dari pengujian nilai *Yaw*, sudut yang dibentuk oleh *quadcopter* mencapai $+180^{O}$ sampai -180^{O} tergantung *quadcopter* sedang bergerak pada sumbu apa. Sedangkan nilai negatif dan nilai positif yang dihasilkan merupakan representasi arah terbang *quadcopter*. Nilai klasifikasi *quadcopter* dapat dilihat pada Tabel 6.9 sampai Tabel 6.13.

Tabel 6.9 Klasifikasi Nilai Sudut *Roll Quadcopter*

Gerakan *quadcopter*	Nilai Sudut *Roll*
Bergerak ke Kanan	0 sampai $+180^{O}$
Bergerak Ke Kiri	0 sampai -180^{O}

Pada Tabel 6.9 Sudut *Roll* atau sudut menurut sumbu X yang dibentuk oleh *quadcopter* dengan kemiringan ke arah kanan dapat mencapai $+180^{O}$, sedangkan sudut yang dibentuk saat kemiringan ke arah kiri mencapai -180^{O}.

Tabel 6.10 Klasifikasi Nilai Sudut *Pitch Quadcopter*

Gerakan *quadcopter*	Nilai Sudut *pitch*
Bergerak ke Belakang	0 sampai $+180^{O}$
Bergerak Ke Depan	0 sampai -180^{O}

Pada Tabel 6.10 Sudut *Pitch* atau sudut menurut sumbu Y yang dibentuk oleh *quadcopter* dengan kemiringan ke arah belakang dapat mencapai $+180^{O}$,sedangkan sudut *Pitch* yang dibentuk saat kemiringan ke arah depan dapat mencapai -180^{O}.

Tabel 6.11 Klasifikasi Nilai sudut *Yaw Quadcopter*

Gerakan *quadcopter*	Nilai Sudut *Yaw*
Berputar Ke kiri	0 sampai -180^{O}
Berputar Ke kanan	0 sampai +180^{O}

Pada Tabel 6.11 Sudut *Yaw* atau sudut menurut sumbu Z yang dibentuk oleh *quadcopter* Berputar ke arah kiri dapat mencapai -180^{O}, sedangkan sudut yang dibentuk saat Berputar ke arah kanan mencapai +180^{O}.

Tabel 6.12 Klasifikasi Nilai Ketinggian naik turun *Quadcopter*

Gerakan *quadcopter*	Nilai Ketinggian
Bergerak ke atas	0.062 meter sampai 3 meter
Bergerak Ke bawah	3 meter sampai 0.062 meter

Pada Tabel 6.12 Nilai *altitude* yang dapat dicapai oleh *quadcopter* saat bergerak naik sebesar 3 meter, nilai 3 meter ini didapat dari *datasheet quadcopter*. *Quadcopter* dapat bergerak turun ke bawah dengan nilai minimal setinggi 0.062 meter, *quadcopter* tidak mampu bergerak ke bawah lagi dikarenakan nilai minimum terbang *quadcopter* sebesar 0.062 meter.

Tabel 6.13 Klasifikasi Nilai Ketinggian *takeoff landing Quadcopter*

Gerakan *quadcopter*	Nilai Ketinggian
Takeoff	0 meter sampai ±1 meter
Landing	n meter sampai 0 meter

Pada Tabel 6.13 Nilai ketinggian *quadcopter* ketika diberikan perintah *takeoff* akan naik setinggi ±1 meter. *Quadcopter* dapat melakukan *landing* dari ketinggian berapapun tanpa ada batas, dan nilai ketinggian akan terus berkurang sampai mencapai angka 0.

Untuk pergerakan dengan banyak perintah, pada pergerakan ke arah kanan dan kiri, sudut *roll* yang dibentuk oleh quadcopter berubah dengan signifikan. Sedangkan sudut *pitch* dan *yaw* tidak berubah dengan signifikan. sedangkan pergerakan ke arah depan dan belakang, sudut *pitch* yang dibentuk oleh quadcopter berubah dengan signifikan. Sedangkan sudut *roll* dan *yaw* tidak berubah dengan signifikan. Sudut *yaw* akan berubah jika quadcopter bergerak berputar ke arah kiri atau kanan. Pergerakan terbang dan *landing*, semua sudut yang dibentuk akan berubah, hal ini terjadi untuk menyeimbangkan quadcopter.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, pengujian dan analisis yang telah dilakukan pada penelitian ini, maka dapat diambil kesimpulan sebagai berikut:

1. Android *device* dapat digunakan untuk pengolahan suara menggunakan *speech recognition Google* dan melakukan *text preprocessing* dengan *database SQLite* untuk nantinya diproses menjadi gerakan *quadcopter.* Hasil pengujian fungsional dinyatakan berhasil dengan tingkat keberhasilan 100%.
2. Dari hasil percobaan 10 *user* berbeda yang telah dilakukan, pengolahan teks dan pengolahan suara memiliki tingkat kebenaran mencapai 100%, sehingga dapat disimpulkan sistem ini mampu berjalan seperti yang diharapkan.
3. Dari percobaan 10 *user*, sistem memiliki nilai benar sebesar 134 dan memiliki nilai salah sebesar 16. Dengan nilai yang diperoleh dari pengujian pada bab sebelumnya dapat disimpulkan bahwa pengolahan suara dan pemrosesan teks yang dilakukan untuk menjadi gerakan *quadcopter* memiliki akurasi yang tinggi, yaitu sebesar 89.3%.
4. Rata-rata pertambahan waktu untuk pengolahan suara akan bertambah sebesar ±1 detik seiring dengan pertambahan masukan kata. Sedangkan waktu yang dibutuhkan untuk pengolahan teks akan bertambah sebesar ±300 *nano* detik pada setiap pertambahan kata. Dengan adanya data yang diperoleh sebelumnya dapat disimpulkan bahwa sistem memiliki proses pengolahan suara dan pengolahan teks yang bertambah seiring dengan banyaknya kata.
5. Dari pengujian yang telah dilakukan dapat ditarik kesimpulan bahwa performa *quadcopter* saat terbang dapat dilihat dari grafik hasil percobaan. Nilai sudut *roll* berubah ketika bergerak ke arah kanan atau kiri. Nilai sudut *pitch* berubah ketika bergerak ke arah depan atau belakang. Nilai sudut *yaw* berubah ketika bergerak berputar ke arah kanan atau kiri. Nilai ketinggian akan berubah sesuai dengan *state* pergerakan *quadcopter.*

7.2 Saran

Terdapat beberapa saran agar sistem ini dapat dikembangkan lebih lanjut diantaranya:

1. Untuk ke depannya sistem dapat dikembangkan tanpa menggunakan Arduino.
2. Saat ini sistem masih menggunakan sistem *cloud* untuk penggunaan grafik secara *real time*,saran ke depannya pembuatan grafik menggunakan sistem *cloud* yang dibuat sendiri.
3. Sistem dapat dikembangkan menjadi metode *text mining*.

DAFTAR PUSTAKA

Chapaneri, S. V., 2012. Spoken Digits Recognition using Weighted MFCC. *International Journal of Computer Applications,* 40(3), p. 7.

Dye, J., 2016. *Google says Voice Search faster and more accurate.* [Online] Available at: http://www.androidauthority.com/google-voice-search-improvements-644532/ [Accessed 3 8 2017].

Fauzi, M. A., 2016. *Text Mining.* [Online] Available at: http://malifauzi.lecture.ub.ac.id/ [Accessed 28 5 2017].

Felixge, 2014. *A node.js client for controlling Parrot AR Drone 2.0 quad-copters..* [Online] Available at: https://github.com/felixge/node-ar-drone [Accessed 4 Mei 2017].

Hernandez-Martinez, E. et al., 2015. *Trajectory Tracking of a Quadcopter UAV.* Ciudad, IFAC.

Kisumal, H., 2010. Aplikasi Perintah Suara Pada Windows. In: Zainul & Kodhijah, eds. *Aplikasi Perintah Suara Pada Windows.* Jakarta: Universitas Islam negeri Jakarta, p. 21.

Krishna, R. et al., 2011. *Modelling and control of quadcopter.* Coimbatore, IEEE.

Lawson, D. & F., L., 2017. *Robotics Exploration.* [Online] Available at: http://roboticsfree.blogspot.co.id/2017/02/how-quadcopters-work.html [Accessed 3 Mei 2017].

Prameswara, D., 2016. *Apakah Node.js ?.* [Online] Available at: http://sinau-webhtml.blogspot.co.id/2016/01/apakah-nodejs-dan-cara-install-di-windows-linux-mac.html [Accessed 2 8 2017].

Ramadhan, B., 2015. realisasi-quadcopter-yang-dapat-dikendalikan-menggunakan-pc. In: E. Mozef & Sutrisno, eds. *realisasi-quadcopter-yang-dapat-dikendalikan-menggunakan-pc.* bandung: Polban, pp. 6-7.

Robotika, E., 2015. *Pesawat Tanpa Awak, Unmanned Aerial Vehicle (UAV).* [Online] Available at: http://zonaelektro.net/unmanned-aerial-vehicle-uav/ [Accessed 4 Mei 2017].

Romero1, L. E., Pozo, D. F. & Rosales, J. A., 2014. *Quadcopter stabilization by using PID controllers.* Quito, IEEE.

Supimrosl, S. & Wongthanavasu, S., 2014. *Speech Recognition - based Control System.* Thailand, IEEE.

TerryKing, 2011. *BlueTooth-HC05-HC06-Modules-How-To.* [Online] Available at: https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To [Accessed 21 Mei 2017].

Tristiadi, A., 2015. PENSTABIL KAMERA UNTUK QUADCOPTER MENGGUNAKAN SENSOR MPU6050. *journal.student.uny.ac.id,* 1(14), p. 7.

Vyskovsky, A., 2015. *Any Object Tracking and Following.* Prague, IEEE.

Yulias, Z., 2013. *Arduino Mega 2560.* [Online] Available at: http://blog.famosastudio.com/2013/09/produk/arduino-mega-2560/531 [Accessed 6 Mei 2017].