PENGEMBANGAN APLIKASI *HEALTH COMMUNICATION BOARD* BERBASIS KENDALI PERGERAKAN LINEAR HEMOCS (*HEAD MOVEMENT CONTROL SYSTEM*)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ivan Putera Pratama
NIM: 135150207111071

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

PENGESAHAN

PENGEMBANGAN APLIKASI *HEALTH COMMUNICATION BOARD* BERBASIS KENDALI PERGERAKAN LINEAR HEMOCS (*HEAD MOVEMENT CONTROL SYSTEM*)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ivan Putera Pratama
NIM: 135150207111071

Skripsi ini telah diuji dan dinyatakan lulus pada
15 Mei 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Herman Tolle, Dr. Eng, S.T, M.T	Hanifah Muslimah Az Zahra, S.Sn, M.Ds
NIP: 19740823 200012 1 001	NIK: 2016078908112001

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 didalam 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, 1 Juni 2017

Ivan Putera Pratama

NIM: 135150207111071

KATA PENGANTAR

Puji Syukur penulis panjatkan kepada Allah SWT karena hanya dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi dengan judul "Pengembangan Aplikasi *Health Communication Board* Berbasis Kendali Pergerakan Linear HEMOCS (*Head Movement Control System*)". Melalui kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang sebesar – besarnya kepada seluruh pihak yang telah memberikan bantuan dan dukungan selama penulisan skripsi ini, diantaranya:

1. Bapak Dr. Eng Herman Tolle, ST., MT. selaku dosen pembimbing I yang telah memberikan masukan, ilmu, serta saran yang bermanfaat dalam proses penyelesaian skripsi ini.
2. Ibu Hanifah Muslimah Az Zahra, S.Sn., M.Ds selaku dosen pembimbing II yang juga telah memberikan masukan, ilmu, serta saran yang bermanfaat dalam proses penyelesaian skripsi ini.
3. Kepada kedua orang tua penulis, yaitu Bapak Agus Effendi dan Ibu Etty Herdinawati beserta keluarga besar yang selalu memberikan segala do'a, nasehat, dan dukungan hingga terselesaikannya skripsi ini.
4. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Drs. Mardji, M.T, dan Bapak Edy Santoso, S.Si, M.Kom selaku Dekan, Wakil Dekan 1, Wakil Dekan 2, dan Wakil Dekan 3 Fakultas Ilmu Komputer, Universitas Brawijaya.
5. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D dan Bapak Agus Wahyu Widowo, S.T, M.Cs selaku Ketua Jurusan Teknik Informatika dan Kepala Program Studi Teknik Informatika Fakultas Ilmu Komputer, Universitas Brawijaya.
6. Seluruh Dosen Fakultas Ilmu Komputer, Universitas Brawijaya atas kesediaannya dalam mengajarkan dan membagikan ilmu yang bermanfaat bagi penulis.
7. Seluruh teman – teman mahasiswa Himpunan Mahasiswa Informatika yang selalu memberi do'a dan semangat dalam proses pengerjaan skripsi ini.
8. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat secara langsung atau tidak langsung dalam proses pengerjaan skripsi ini.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan. Oleh karena itu, kritik dan saran yang bersifat membangun sangat diharapkan untuk menyempurnakan skripsi ini. Semoga skripsi ini membawa manfaat bagi pihak lain yang menggunakannya.

Malang, 1 Juni 2017

Penulis

ivanputeraa@gmail.com

ABSTRAK

Ivan Putera Pratama. 2017. *Pengembangan Aplikasi Health Communication Board* Berbasis Pergerakan Linear HEMOCS (*Head Movement Control System*). Skripsi Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Herman Tolle, Dr. Eng, S.T, M.T dan Hanifah Muslimah Az Zahra, S.Sn, M.Ds.

Head Movement Control System (HEMOCS) merupakan sebuah metode yang memanfaatkan gerakan kepala sebagai kontrol akan objek yang ada di dalam sebuah sistem. Dalam bidang penanganan medis, *Health Communication Board* (HCB) merupakan sebuah solusi yang umum digunakan oleh dokter kepada pasien dalam membantu mereka menyampaikan informasi yang diinginkan. Namun solusi tersebut rupanya tidak dapat diaplikasikan pada pasien yang sedang mengalami keterbatasan fisik di bagian anggota gerak tangan. Oleh sebab itu, peneliti merumuskan adanya pengembangan sistem yang serupa dengan HCB yang diimplementasikan pada sebuah perangkat bergerak platform iOS dan dikendalikan menggunakan pergerakan kepala pengguna. Sistem yang dikembangkan akan memadukan penggunaan perangkat bergerak yang mengimplementasikan teknologi *augmented reality* (AR) dengan sebuah *Head-Mounted Display* (HMD). Pergerakan kepala yang dikenali oleh sistem antara lain gerakan kepala menoleh ke kiri, kanan, melihat ke atas, dan bawah. Pergerakan kepala tersebut akan digunakan sebagai kontrol dan navigasi pengguna dalam berinteraksi dengan sistem. Hasil pengujian usabilitas yang telah dilakukan terhadap lima buah responden menunjukkan bahwa aplikasi yang dikembangkan mendapatkan rata – rata indeks persentase kepuasan pengguna sebesar 74,8% yang menunjukkan bahwa penggunaan aplikasi ini memberikan hasil yang memuaskan bagi pengguna.

Kata kunci: HEMOCS, HMD, augmented reality, *mobile,* iOS, komunikasi, kesehatan, kontrol.

ABSTRACT

Ivan Putera Pratama. 2017. *Development of Health Communication Board Application Using Linear Head Movement Control System* (HEMOCS). Informatics Engineering Department Thesis, Faculty of Computer Science, Brawijaya University, Malang. Supervisors: Herman Tolle, Dr. Eng, S.T, M.T and Hanifah Muslimah Az Zahra, S.Sn, M.Ds.

Head Movement Control System (HEMOCS) is a method which utilizes head movements to control objects that exist in a system. In medical treatment cases, Health Communication Board (HCB) acts as a solution that is commonly used by doctors to patients in helping them deliver their desired information. But the solution is apparently not applicable in patients who are experiencing hand physical limitations. Therefore, we propose the development of a similar system with HCB implemented on an iOS mobile device platform which are controlled using the user's head movement. The system will combine the use of mobile devices that implement augmented reality (AR) technology with a Head-Mounted Display (HMD). Head movement types that are recognized by the system including left, right, up, and down head movements. These movements used as a control for user to interact with the system. The results of usability testing has been conducted on five respondents indicated that this application has an average percentage of user satisfaction index of 74.8%, which shows that the use of application provide a satisfactory result for the user.

Keywords: HEMOCS, head-mounted display, augmented reality, mobile, iOS, communication, health, control.

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR TABEL xi
DAFTAR GAMBAR xiii
DAFTAR KODE PROGRAM xiv
DAFTAR LAMPIRAN xv
DAFTAR ISTILAH 1
BAB 1 PENDAHULUAN 2
1.1 Latar Belakang 2
1.2 Rumusan Masalah 3
1.3 Tujuan 3
1.4 Manfaat Penelitian 4
1.5 Batasan Masalah 4
1.6 Sistematika Pembahasan 4
BAB 2 LANDASAN KEPUSTAKAAN 6
2.1 *Head Movement Control System* (HEMOCS) 6
2.1.1 Kajian Pustaka 6
2.1.2 Algoritme HEMOCS 11
2.2 Kendali Pergerakan Linear 14
2.3 Sensor Gerak 15
2.3.1 Accelerometer 15
2.3.2 Gyroscope 16
2.4 Pemrograman *Objective-C* 16
2.4.1 Struktur *Class* 17
2.4.2 *Framework* UIKit 18
2.4.3 Siklus Hidup *View* 19

2.5 Framework Core Motion 20
2.6 *Head-Mounted Display* (HMD) 21
BAB 3 METODOLOGI 23
3.1 Strategi dan Rancangan Penelitian 23
3.1.1 Studi Literatur 24
3.1.2 Analisis Kebutuhan 24
3.1.3 Perancangan 24
3.1.4 Implementasi 24
3.1.5 Pengujian 25
3.1.6 Pengambilan Kesimpulan dan Saran 25
BAB 4 ANALISIS KEBUTUHAN 26
4.1 Gambaran Umum Aplikasi 26
4.2 Identifikasi Aktor 27
4.3 Kebutuhan Fungsional Sistem 27
BAB 5 PERANCANGAN 29
5.1 Perancangan Arsitektur Sistem 29
5.2 Perancangan *User Experience* 29
5.2.1 Perancangan Ikon Komunikasi 29
5.2.2 Perancangan *Tab* 33
5.2.3 Perancangan Interaksi 35
5.2.4 Perancangan *Screenflow* 37
5.3 *Unified Modeling Language* (UML) 39
5.3.1 Diagram *Use Case* 39
5.3.2 Skenario *Use Case* 39
5.3.3 Diagram Aktivitas 44
BAB 6 IMPLEMENTASI 48
6.1 Batasan Implementasi 48
6.2 Spesifikasi Perangkat Keras 48
6.3 Spesifikasi Perangkat Lunak 49
6.4 Implementasi Kode Program 49
6.4.1 Variabel Sistem 50
6.4.2 Implementasi Fungsional Sistem 55

6.5 Implementasi Antarmuka 69
BAB 7 PENGUJIAN 72
7.1 Pengujian Fungsional 72
7.1.1 Penentuan Kasus Uji 72
7.1.2 Hasil Pengujian Fungsional 78
7.2 Pengujian Non Fungsional 79
7.2.1 Pengujian *Usability* 79
7.2.2 Analisis Hasil Pengujian *Usability* 81
BAB 8 Penutup 85
8.1 Kesimpulan 85
8.2 Saran 86
DAFTAR PUSTAKA 87
LAMPIRAN 89

DAFTAR TABEL

Tabel 4.1 Aktor Sistem 27
Tabel 4.2 Kebutuhan Fungsional Sistem 27
Tabel 5.1 Perancangan Ikon Komunikasi 30
Tabel 5.2 Pemetaan Pergerakan Kepala 35
Tabel 5.3 Skenario *Use Case* Memindahkan Posisi Halaman / *Tab* 40
Tabel 5.4 Skenario *Use Case* Memindahkan Posisi Kursor 40
Tabel 5.5 Skenario *Use Case* Memilih Aksi 41
Tabel 5.6 Skenario *Use Case* Menyusun Kata 42
Tabel 5.7 Skenario *Use Case* Hapus Karakter 42
Tabel 5.8 Skenario *Use Case* Memutar Suara Kata 43
Tabel 6.1 Spesifikasi Perangkat Keras Komputer Macbook Pro 48
Tabel 6.2 Spesifikasi Perangkat Keras *Smartphone* iPhone 49
Tabel 6.3 Spesifikasi Perangkat Lunak Komputer Macbook Pro 49
Tabel 6.4 Spesifikasi Perangkat Lunak *Smartphone* iPhone 49
Tabel 6.5 *Class Controller* Sistem 50
Tabel 6.6 Variabel Sistem 50
Tabel 7.1 Kasus Uji Pindah ke *Tab* Kedua dari *Tab* Pertama 72
Tabel 7.2 Kasus Uji Pindah ke *Tab* Pertama dari *Tab* Kedua 73
Tabel 7.3 Kasus Uji Pindah Kursor ke Kanan 73
Tabel 7.4 Kasus Uji Pindah Kursor ke Kiri 74
Tabel 7.5 Kasus Uji Pindah Kursor ke Atas 74
Tabel 7.6 Kasus Uji Pindah Kursor ke Bawah 75
Tabel 7.7 Kasus Uji Pilih Aksi Pengguna 75
Tabel 7.8 Kasus Uji Memilih Ikon Karakter 76
Tabel 7.9 Kasus Uji Hapus Karakter 76
Tabel 7.10 Kasus Uji Memutar Suara Kata yang Telah Disusun 77
Tabel 7.11 Hasil Pengujian Fungsional 78
Tabel 7.12 Daftar Pernyataan Kuisioner 80
Tabel 7.13 Klasifikasi Tingkat Kepuasan Pengguna 82
Tabel 7.14 Perhitungan Indeks Kepuasan Pengguna 82

Tabel 7.15 Hasil Akhir Pengujian Parameter *Usability*..... 83

DAFTAR GAMBAR

Gambar 2.1 Gerakan Kepala Menoleh ke Kanan 7
Gambar 2.2 Gerakan Kepala Menoleh ke Kiri 7
Gambar 2.3 Gerakan Kepala Melihat ke Atas 8
Gambar 2.4 Gerakan Kepala Melihat ke Bawah 8
Gambar 2.5 Gerakan Kepala Miring ke Kiri 9
Gambar 2.6 Gerakan Kepala Miring ke Kanan 9
Gambar 2.7 *Flowchart* Algoritme HEMOCS 13
Gambar 2.8 Penerapan *Focus Engine* pada tvOS 14
Gambar 2.9 Sumbu Kartesian *Accelerometer* dan *Gyrsocope* 16
Gambar 2.10 Siklus Hidup View dalam Platform iOS 20
Gambar 2.11 Google Cardboard 22
Gambar 3.1 Strategi dan Rancangan Penelitian 23
Gambar 4.1 VIDATAK EZ *Picture Board* 26
Gambar 5.1 Rancangan Antarmuka *Tab* Pertama 33
Gambar 5.2 Rancangan Antarmuka *Tab* Kedua 34
Gambar 5.3 Rancangan Antarmuka *Tab* Ketiga 34
Gambar 5.4 Rancangan Antarmuka *Tab* Keempat 34
Gambar 5.5 Contoh Perpindahan Kursor ke Arah Kanan 36
Gambar 5.6 Aliran Antarmuka Sistem 37
Gambar 5.7 *Use Case* Sistem 39
Gambar 5.8 Diagram Aktivitas Memindahkan Posisi Halaman / *Tab* 44
Gambar 5.9 Diagram Aktivitas Memindahkan Posisi Kursor 45
Gambar 5.10 Diagram Aktivitas Memilih Aksi 45
Gambar 5.11 Diagram Aktivitas Menyusun Kata 46
Gambar 5.12 Diagram Aktivitas Menghapus Karakter 46
Gambar 5.13 Diagram Aktivitas Memutar Suara Kata 47
Gambar 6.1 Implementasi Antarmuka Aplikasi 70

DAFTAR KODE PROGRAM

Kode Program 6.1 Implementasi *File Header* 55
Kode Program 6.2 Pembacaan Pergerakan Kepala..... 58
Kode Program 6.3 Memindahkan Posisi Kursor..... 62
Kode Program 6.4 Memutar Suara Aksi yang Dipilih..... 64
Kode Program 6.5 Memutar Suara Kata 66
Kode Program 6.6 Menghapus Karakter..... 68

DAFTAR LAMPIRAN

Lampiran 1 - Hasil Kuisioner Usabilitas Aplikasi (Responden 1) 89
Lampiran 2 - Hasil Kuisioner Usabilitas Aplikasi (Responden 2) 90
Lampiran 3 - Hasil Kuisioner Usabilitas Aplikasi (Responden 3) 91
Lampiran 4 - Hasil Kuisioner Usabilitas Aplikasi (Responden 4) 92
Lampiran 5 - Hasil Kuisioner Usabilitas Aplikasi (Responden 5) 93

DAFTAR ISTILAH

Cocoa *Touch*

Cocoa *Touch* merupakan sebuah *framework* UI yang digunakan untuk membangun sebuah aplikasi perangkat lunak yang dapat dijalankan pada sistem operasi iOS (untuk iPhone, iPod Touch, dan iPad), watchOS untuk Apple *Watch*, dan tvOS untuk TV Apple generasi keempat, yang dikembangkan oleh Apple. Cocoa *Touch* menyediakan lapisan abstraksi pada sistem operasi iOS yang didasarkan pada API milik Cocoa macOS yang ditulis dalam bahasa *Objective*-C. Cocoa *Touch* memungkinkan penggunaan perangkat keras dan fitur yang tidak ditemukan pada macOS, sehingga penggunaannya hanya ditujukan bagi perangkat yang menjalankan sistem operasi iOS saja. Sama seperti Cocoa, Cocoa *Touch* juga menggunakan arsitektur perangkat lunak *Model-View-Controller* (MVC).

iOS

iOS (dahulu iPhone OS) merupakan sebuah sistem operasi *mobile* yang dibuat dan dikembangkan oleh Apple. Sistem operasi ini dijalankan pada perangkat *mobile* yang diciptakan perusahaan Apple, seperti iPhone, iPad, dan iPod Touch. iOS juga menjadi sistem operasi *mobile* kedua yang paling populer di seluruh dunia setelah Android. Antarmuka pengguna iOS didasarkan pada manipulasi secara langsung dengan menggunakan *multi-touch gestures*. Elemen kontrol antarmuka terdiri dari *slider*, *switch*, dan *button*. Interaksi dengan OS mencakup gerak tubuh seperti *swipe*, *tap*, *pinch*, dan *reverse pinch*, yang seluruhnya memiliki definisi spesifik dalam konteks penggunaan sistem operasi iOS serta antarmuka *multi-touch* yang ada.

tvOS

tvOS merupakan sebuah sistem operasi yang dikembangkan oleh Apple Inc. dan dijalankan pada perangkat media player digital Apple TV generasi ke-4. Sistem operasi tvOS secara garis besar didasarkan pada sistem operasi iOS yang telah dikembangkan sebelumnya. Berbeda dengan iOS, antarmuka pengguna tvOS didasarkan pada manipulasi secara tidak langsung. Interaksi pengguna dengan tvOS dibantu dengan sebuah perangkat yang bernama *siri remote*. Sistem operasi ini juga mendukung perangkat lunak pihak ketiga (3rd *party software*) yang dapat diunduh oleh pengguna melalui App Store.

Xcode

Xcode merupakan sebuah lingkungan pengembangan terpadu (IDE) untuk macOS yang dikembangkan oleh Apple. Xcode berisi sejumlah alat pengembangan perangkat lunak yang digunakan oleh *developer* aplikasi untuk mengembangkan suatu perangkat lunak bagi sistem operasi macOS, iOS, watchOS dan tvOS.

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Wearable computing merupakan salah satu kajian ilmu baru dalam bidang teknologi yang kini sedang banyak dipelajari selama beberapa tahun belakangan ini (K. Arai 2013). Kajian yang merupakan bagian dari kajian ilmu *Human and Computer Interaction* (HCI) ini membahas studi seputar penciptaan, perancangan, pembangunan, serta penggunaan perangkat komputer dan sensor yang dapat dikenakan di tubuh pengguna, seperti di tangan, kepala, dan kaki. Tren akan kajian ilmu tersebut kemudian memunculkan sejumlah perangkat yang lebih kita kenal dengan sebutan *wearable device*, yaitu perangkat yang merupakan penggabungan antara pakaian atau aksesoris dengan komputer dan komponen elektronika canggih yang memiliki sejumlah fungsi dan fitur tertentu. *Wearable device* memberikan sejumlah fitur mulai dari yang spesifik dan terbatas layaknya monitor detak jantung, dan langkah kaki sampai dengan fitur yang canggih seperti mengambil foto atau video, menulis pesan, membaca e-mail, memberikan perintah melalui suara, dan lainnya layaknya yang diberikan oleh ponsel pintar maupun jam tangan pintar. Namun sampai dengan saat ini, pengembangan *wearable device* masih fokus dan terbatas pada bidang kesehatan dan kebugaran manusia saja dikarenakan penggunaan dari *wearable device* yang sifatnya tidak begitu *urgent* dan tidak terlalu dibutuhkan dalam menunjang produktivitas manusia secara umum.

Berkat munculnya era *mobile computing*, kebutuhan manusia akan ponsel pintar saat ini hampir menjadi kebutuhan pokok (Tolle, et al. 2015). Ponsel pintar yang merupakan salah satu contoh perangkat bergerak yang banyak digunakan oleh masyarakat saat ini telah dilengkapi oleh sejumlah sensor internal seperti *accelerometer, gyroscope, barometer,* dan GPS yang digunakan sebagai alat untuk mendeteksi serta merespon sejumlah kondisi lingkungan fisik tertentu. Dengan kombinasi antara ponsel pintar dengan sebuah *Head-Mounted Display* (HMD) layaknya Google Cardboard, maka ponsel pintar tersebut dapat berubah menjadi sebuah *wearable device* yang memberikan sejumlah fitur canggih bagi penggunanya, seperti contoh penggunaan aplikasi edukasi berbasis *Virtual Reality* (VR) mengenai anatomi tubuh manusia, aplikasi permainan berbasis *Augmented Reality* (AR), aplikasi *StreetView,* dan masih banyak lagi yang lainnya. Maka dari itu, studi seputar pemanfaatan ponsel pintar sebagai sebuah *wearable device* yang dapat membantu kehidupan manusia merupakan hal yang sangat mungkin dilakukan di masa yang akan datang.

Dalam bidang penanganan medis, komunikasi antara seorang pasien dan perawat sangatlah penting. Ketika kemampuan untuk saling berkomunikasi tersebut hilang dikarenakan adanya keterbatasan fisik, trauma, prosesi medis tertentu, atau keterbatasan pemahaman bahasa, dibutuhkan suatu teknik alternatif untuk dapat menyampaikan perintah yang ingin disampaikan oleh seorang pasien. *Health Communication Board* (HCB) merupakan salah satu solusi yang lazim diberikan oleh pihak rumah sakit kepada pasien untuk membantu

komunikasi seputar masalah medis, fisik, dan emosional dengan dokter atau perawat mereka. HCB berbentuk sebuah papan yang berisi sejumlah aksi pasien secara umum, seperti makan, minum, meminta bantuan, dan sebagainya yang direpresentasikan ke dalam bentuk teks dan ikon gambar tertentu. Ikon ini kemudian ditunjuk oleh pasien menggunakan jari tangan mereka ketika berhadapan dengan seorang perawat atau dokter untuk memberitahu keadaan atau keinginan mereka saat itu. Namun solusi ini memiliki kelemahan, yaitu solusi ini tidak dapat diterapkan pada pasien yang mengalami keterbatasan fisik di bagian anggota gerak tangan.

Dengan adanya permasalahan tersebut, diperlukan sebuah penelitian untuk mengembangkan sebuah sistem yang dapat menggantikan atau menjadi substitusi fungsi jari tangan manusia dalam melakukan pemilihan perintah tertentu. Berkat keberadaan sejumlah sensor internal pendeteksi pergerakan perangkat yang dimiliki oleh perangkat bergerak dan *wearable device* seperti Cardboard, memungkinkan adanya kesempatan untuk mengembangkan sebuah aplikasi yang memiliki prinsip dasar seperti halnya solusi HCB yang telah ada, namun dikendalikan menggunakan pergerakan kepala pengguna. Metode kontrol dan pengendalian berbasis pergerakan kepala ini kemudian diberi nama *Head Movement Control System* (HEMOCS). Diperlukan pula pengujian untuk mengetahui kualitas aplikasi yang telah dikembangkan dan kesesuaian dengan kebutuhan pengguna. Hasil akhir dari penelitian ini diharapkan dapat membantu komunikasi antara pengguna yang mengalami keterbatasan fisik di bagian anggota gerak tangan dengan orang lain secara normal.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah ditulis sebelumnya, maka dapat dirumuskan masalah sebagai berikut:

1. Bagaimana rancangan aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS (*Head Movement Control System*)?
2. Bagaimana implementasi algoritme kendali pergerakan kepala yang digunakan pada aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS (*Head Movement Control System*)?
3. Seberapa besar tingkat usabilitas aplikasi aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS (*Head Movement Control System*) pada pengguna?

1.3 Tujuan

Adapun tujuan yang ingin dicapai dari penelitian ini adalah sebagai berikut:

1. Mengembangkan sebuah aplikasi perangkat bergerak yang diaplikasikan pada pengguna yang mengalami keterbatasan fisik di bagian anggota gerak tangan dalam membantu mereka berkomunikasi dengan orang lain secara normal.

2. Mengimplementasikan alogritme kendali pergerakan kepala pada aplikasi perangkat bergerak yang menggunakan kendali kepala sebagai kontrolnya.
3. Mengetahui tingkat usabilitas aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS (*Head Movement Control System*) pada pengguna.

1.4 Manfaat Penelitian

Adapun manfaat yang diperoleh dari penelitian ini adalah sebagai berikut:

1. Bagi peneliti : Memperdalam pengetahuan seputar proses pengembangan sistem yang memanfaatkan sensor gerak perangkat bergerak.
2. Bagi pengguna : Membantu komunikasi yang dilakukan oleh pengguna yang mengalami keterbatasan fisik di bagian anggota gerak tangan kepada orang lain secara normal.
3. Bagi peneliti selanjutnya : Menjadi sumber referensi yang bermanfaat untuk digunakan di penelitian selanjutnya yang membahas seputar pemanfaatan sensor gerak perangkat bergerak.

1.5 Batasan Masalah

Penelitian ini dibatasi oleh sejumlah hal sebagai berikut:

1. Pengembangan sistem dilakukan pada *platform* perangkat bergerak Apple iOS.
2. *User interface* sistem hanya diimplementasikan dalam satu bahasa, yaitu Bahasa Inggris.
3. Pengujian usabilitas sistem masih dilakukan pada responden yang merupakan orang normal.

1.6 Sistematika Pembahasan

Penyusunan dokumen ini menggunakan kerangka pembahasan yang tersusun atas:

1. BAB I – PENDAHULUAN
   Memuat latar belakang, rumusan masalah, tujuan, manfaat, batasan, dan sistematika pembahasan dari penelitian.

2. BAB II – LANDASAN KEPUSTAKAAN
   Memuat kajian pustaka dan teori – teori yang berkaitan serta menunjang proses penelitian.

3. BAB III – METODOLOGI
   Memuat alur kerja yang dilakukan oleh peneliti dalam menyelesaikan permasalahan penelitian.

4. BAB IV – ANALISIS KEBUTUHAN
   Memuat hal – hal yang terkait seputar proses penggalian kebutuhan dari sistem yang akan dikembangkan.

5. BAB V – PERANCANGAN
   Memuat hal – hal yang terkait dengan pemetaan kebutuhan sistem menjadi rancangan dasar atau model dari sistem yang akan dikembangkan.

6. BAB VI – IMPLEMENTASI
   Memuat hal – hal yang terkait dengan proses implementasi sistem yang mengacu kepada model yang telah dirancang sebelumnya.

7. BAB VII – PENGUJIAN
   Menjelaskan proses pengujian sistem yang dilakukan oleh responden beserta analisis hasilnya.

8. BAB VIII – PENUTUP
   Memuat kesimpulan dari penelitian yang telah dilakukan dan saran dalam pengembangan penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 *Head Movement Control System* (HEMOCS)

Head Movement Control System (HEMOCS) merupakan sebuah sistem kontrol yang memanfaatkan data yang berasal dari sensor gerak perangkat sebagai acuan dalam menentukan pergerakan yang diinginkan. Sistem kontrol ini dirancang untuk menjadi subtitusi jari tangan manusia dalam membantu komunikasi antara pengguna yang mengalami keterbatasan fisik di bagian anggota gerak tangan dengan orang lain secara normal. Sistem kontrol ini kemudian diimplementasikan pada sebuah aplikasi perangkat bergerak yang interaksinya dikendalikan menggunakan gerakan kepala pengguna.

2.1.1 Kajian Pustaka

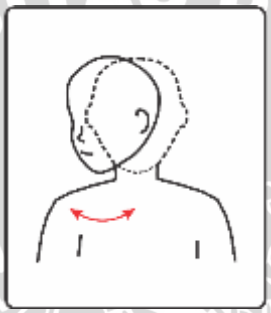
Penelitian terkait pertama yang berjudul "*Mobile Device Based 3D Image Display Depending on User's Action and Movement*" membahas permasalahan mengenai teknik dalam mengontrol navigasi kamera pada sebuah citra 3 dimensi pada sebuah perangkat bergerak sesuai dengan pergerakan badan pengguna (Arai, Tolle and Serita 2013). Navigasi kamera pada sebuah citra tiga dimensi (*3D Image*) dilakukan sesuai dengan memanfaatkan pergerakan perangkat bergerak yang dipasangkan pada anggota tubuh pengguna. Ketika pengguna bergerak berputar ke kiri, kamera pada tampilan tiga dimensi akan mengarah ke arah kiri, lalu ketika pengguna bergerak maju ke depan, maka kamera tersebut juga seolah – olah bergerak ke arah depan, begitu pula dengan pergerakan lainnya seperti melihat ke arah atas, melihat ke arah bawah, berputar ke kiri atau ke kanan, dan kondisi diam. Penentuan kondisi pergerakan tersebut mengacu pada perubahan data yang terjadi pada sensor *accelerometer* dan *gyroscope*. Data pergerakan yang diperoleh dari kedua sensor tersebut kemudian diproses dengan bantuan *framework core motion* untuk menghasilkan data *deviceMotion*, yang merupakan enkapsulasi dari sejumlah data pergerakan perangkat. Dengan adanya data tersebut, peneliti lalu menggunakannya sebagai basis pergerakan navigasi kamera yang ditampilkan melalui citra tiga dimensi.

Penelitian terkait kedua adalah "Pengembangan Metode Pendeteksian Pergerakan Kepala berbasis Sensor Internal pada Perangkat Bergerak Berbasis iOS". Penelitian ini membahas seputar perancangan serta pengembangan metode pendeteksian pergerakan kepala pengguna yang memanfaatkan data yang diperoleh dari sensor gerak perangkat bergerak (Safii 2016). Dalam penelitian ini, *framework core motion* yang tersedia di dalam *platform* iOS juga digunakan sebagai alat bantu dalam memperoleh data pergerakan perangkat. Pengembangan metode dalam penelitian tersebut menggunakan data *deviceMotion* yang terdiri atas data *device attitude*, *rotation rate*, *gravity*, serta *user acceleration*. Data sumbu *yaw* digunakan untuk mendeteksi pergerakan kepala ke kanan dan kiri, data sumbu *roll* untuk mendeteksi pergerakan kepala ke atas dan bawah serta data sumbu pitch untuk mendeteksi pergerakan kepala miring ke kanan dan kiri. Enam buah jenis pergerakan kepala yang terdiri atas

gerakan menoleh ke kiri, menoleh ke kanan, melihat ke atas, melihat ke bawah, miring ke kanan, dan miring ke kiri dijadikan acuan perancangan pendeteksian pergerakan kepala pengguna. Berikut ini penjelasan lebih detil dari masing – masing jenis pergerakan kepala yang ada:

a) Menoleh ke kanan

Gerakan kepala menoleh ke kanan atau dalam Bahasa Inggris disebut dengan istilah *head axial right movement* merupakan sebuah jenis pergerakan kepala yang dilakukan dengan memutar kepala secara horizontal dari titik awal tegak lurus ke depan berputar ke arah kanan, kemudian dilanjutkan dengan berputar kembali ke titik awal. Gambar 2.1 merupakan ilustrasi jenis pergerakan kepala menoleh ke kanan.

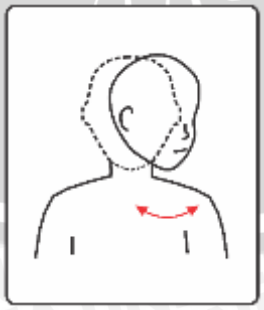


Gambar 2.1 Gerakan Kepala Menoleh ke Kanan

Sumber: (Safii 2016)

b) Menoleh ke kiri

Gerakan kepala menoleh ke kiri atau dalam Bahasa Inggris disebut dengan istilah *head axial left movement* merupakan sebuah jenis pergerakan kepala yang dilakukan dengan memutar kepala secara horizontal dari titik awal tegak lurus ke depan berputar ke arah kiri, kemudian dilanjutkan dengan berputar kembali ke titik awal. Gambar 2.2 merupakan ilustrasi jenis pergerakan kepala menoleh ke kiri.

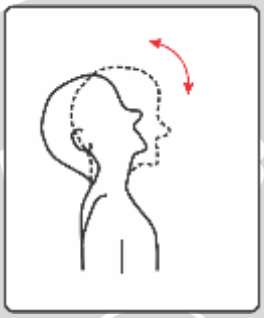


Gambar 2.2 Gerakan Kepala Menoleh ke Kiri

Sumber: (Safii 2016)

c) Melihat ke atas

Gerakan kepala melihat ke atas atau dalam Bahasa Inggris disebut dengan istilah *head extension movement* merupakan sebuah jenis pergerakan kepala yang dilakukan dengan memutar kepala secara vertikal dari titik awal tegak lurus ke depan berputar ke arah atas, kemudian dilanjutkan dengan berputar kembali ke titik awal. Gambar 2.3 merupakan ilustrasi jenis pergerakan kepala melihat ke atas.

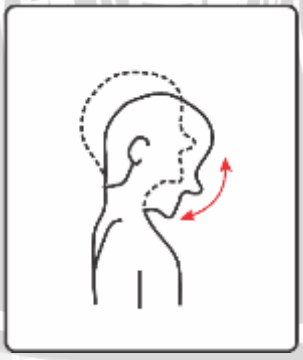


Gambar 2.3 Gerakan Kepala Melihat ke Atas

Sumber: (Safii 2016)

d) Melihat ke bawah

Gerakan kepala melihat ke bawah atau dalam Bahasa Inggris disebut dengan istilah *head flexion movement* merupakan sebuah jenis pergerakan kepala yang dilakukan dengan memutar kepala secara vertikal dari titik awal tegak lurus ke depan berputar ke arah bawah, kemudian dilanjutkan dengan berputar kembali ke titik awal. Gambar 2.4 merupakan ilustrasi jenis pergerakan kepala melihat ke bawah.

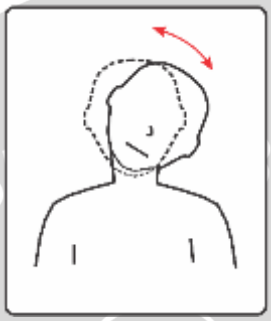


Gambar 2.4 Gerakan Kepala Melihat ke Bawah

Sumber: (Safii 2016)

e) Memiringkan ke arah kiri

Gerakan kepala memiringkan ke arah kiri atau dalam Bahasa Inggris disebut dengan istilah *head lateral bending left movement* merupakan sebuah jenis pergerakan kepala yang dilakukan dengan memiringkan kepala ke arah kiri kemudian dilanjutkan dengan pergerakan menegakkan kembali kepala seperti semula. Gambar 2.5 merupakan ilustrasi jenis pergerakan kepala miring ke arah kiri.

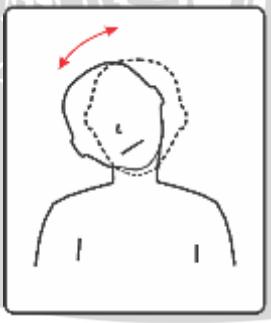


Gambar 2.5 Gerakan Kepala Miring ke Kiri

Sumber: (Safii 2016)

f) Memiringkan ke arah kanan

Gerakan kepala memiringkan ke arah kanan atau dalam Bahasa Inggris disebut dengan istilah *head lateral bending right movement* merupakan sebuah jenis pergerakan kepala yang dilakukan dengan memiringkan kepala ke arah kanan kemudian dilanjutkan dengan pergerakan menegakkan kembali kepala seperti semula. Gambar 2.6 merupakan ilustrasi jenis pergerakan kepala miring ke arah kanan.



Gambar 2.6 Gerakan Kepala Miring ke Kanan

Sumber: (Safii 2016)

Penelitian terkait yang terakhir adalah "Perancangan Kontrol Keyboard Menggunakan Head Movement System Dengan Pergerakkan Linier Pada Perangkat Bergerak Berbasis IOS". Penelitian ini membahas seputar perancangan sistem kontrol *keyboard* menggunakan pergerakan kepala pengguna (Rusyda 2016). Di dalam penelitian tersebut, peneliti merancang sebuah tampilan *keyboard* berbasis konsep *augmented reality* yang dimunculkan pada layar perangkat. Terdapat delapan buah fungsi gerakan kepala yang diimplementasikan di dalam penelitian tersebut, antara lain menoleh ke kiri untuk memindahkan *pointer* pada tombol *keyboard* ke sebelah kiri / *previous*, menoleh ke kanan untuk memindahkan pointer pada tombol *keyboard* ke sebelah kanan / *next*, melihat ke atas untuk memindahkan pointer pada tombol *keyboard* ke sebelah atas /*up* dan menghapus huruf / *backspace,* melihat ke bawah untuk memindahkan pointer pada tombol *keyboard* ke sebelah bawah dan memilih tombol huruf / simbol, gerakan miring ke kiri untuk mengosongkan teks / *clear text*, dan gerakan miring ke kanan untuk memberi karakter spasi / *space*. Selanjutnya, dari hasil pengujian akurasi *threshold* yang telah dilakukan peneliti, didapatkan bahwa sudut optimal yang digunakan dalam mengenali gerakan kepala pengguna adalah dua belas derajat dengan akurasi sebesar 87%.

Dari ketiga buah penelitian tersebut, dapat diambil sejumlah informasi bahwa pengembangan sistem menggunakan kendali HEMOCS dapat dilakukan dengan memanfaatkan data pergerakan perangkat yang berasal dari *accelerometer* dan *gyroscope*. Data *deviceMotion* yang didapatkan kemudian diproses menggunakan suatu algoritme tertentu untuk dapat dipetakan menjadi sejumlah aksi yang terkait dengan navigasi yang dapat dilakukan pengguna terhadap sistem. Adapun jenis pergerakan kepala yang dapat digunakan di dalam sistem yang menggunakan kendali HEMOCS secara umum meliputi gerakan kepala ke kanan, kiri, melihat ke atas, bawah, serta gerakan miring ke kanan, dan kiri. Sistem dengan kendali HEMOCS diimplementasikan ke dalam sebuah aplikasi perangkat bergerak dengan menerapkan konsep *augmented reality*. Dan terakhir adalah penggunaan sudut *threshold* optimal sebesar dua belas derajat yang digunakan dalam mengenali gerakan kepala yang dilakukan pengguna terhadap perangkat. Seluruh informasi yang telah didapatkan dari tiga buah penelitian terkait sebelumnya akan digunakan oleh peneliti dalam menunjang proses pengembangan aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS (*Head Movement Control System*) ke depan.

2.1.2 Algoritme HEMOCS

Algoritme merupakan sebuah susunan yang logis, sistematis, dan terbatas yang digunakan untuk menyelesaikan suatu masalah ataupun tujuan tertentu. Terdapat enam buah tahap yang menyusun algoritme dari sistem yang mengimplementasikan *Head Movement Control System* (HEMOCS). Berikut ini penjelasan dari setiap tahap yang ada secara berurutan:

1. Mendapatkan data pergerakan perangkat

   Sistem mengambil data pergerakan perangkat yang berasal dari sensor gerak. Data tersebut kemudian diproses lalu digunakan pada tahapan selanjutnya untuk menentukan aksi yang sesuai untuk dijalankan oleh sistem.

2. Penentuan inisialiasi pergerakan kepala

   Pada tahap ini, sistem akan menentukan inisialiasi pembacaan pergerakan kepala pengguna. Inisialisasi pembacaan pergerakan kepala mengacu pada seleksi kondisi antara data pergerakan perangkat dengan *threshold* sumbu pergerakan kepala yang telah ditentukan di awal. Apabila data tersebut melebihi atau sama dengan nilai *threshold* sumbu pergerakan kepala, maka proses perhitungan interval pergerakan kepala pengguna akan dimulai, dimana proses ini akan ditandai oleh adanya *increment* terhadap sebuah variabel *counter* selama proses tersebut berjalan. Namun apabila data pergerakan tersebut tidak melebihi *threshold* sumbu pergerakan kepala, maka proses perhitungan interval pergerakan kepala pengguna tidak akan dilakukan atau dengan kata lain, pengguna dinyatakan tidak melakukan pergerakan kepala apapun terhadap sistem.

3. Penentuan arah pergerakan kepala

   Setelah proses inisialiasi pergerakan dimulai, sistem akan menentukan arah pergerakan kepala yang dilakukan oleh pengguna. Penentuan arah yang dimaksud terdiri atas gerakan kepala ke kanan, kiri, atas, atau bawah. Penentuan tersebut mengacu kepada pendeteksian nilai positif atau negatif dari data sumbu pergerakan perangkat yang mengalami perubahan nilai serta orientasi perangkat yang sedang digunakan oleh pengguna saat itu. Sebagai contoh, apabila perangkat berada pada orientasi *landscape right* dan data yang terdeteksi dari sumbu X perangkat bernilai positif, maka arah pergerakan kepala yang dilakukan oleh pengguna adalah ke bawah, sebaliknya apabila data terdeteksi dari sumbu X perangkat bernilai negatif, maka arah pergerakan kepala yang dilakukan oleh pengguna adalah ke atas. Begitupula dengan yang terjadi pada sumbu Y perangkat, apabila data yang terdeteksi dari sumbu Y perangkat bernilai positif, maka arah pergerakan kepala yang dilakukan oleh pengguna adalah ke kiri, sebaliknya apabila data yang

terdeteksi dari sumbu Y perangkat bernilai positif, maka arah pergerakan kepala yang dilakukan oleh pengguna adalah ke kanan.

4. Penentuan durasi pergerakan kepala

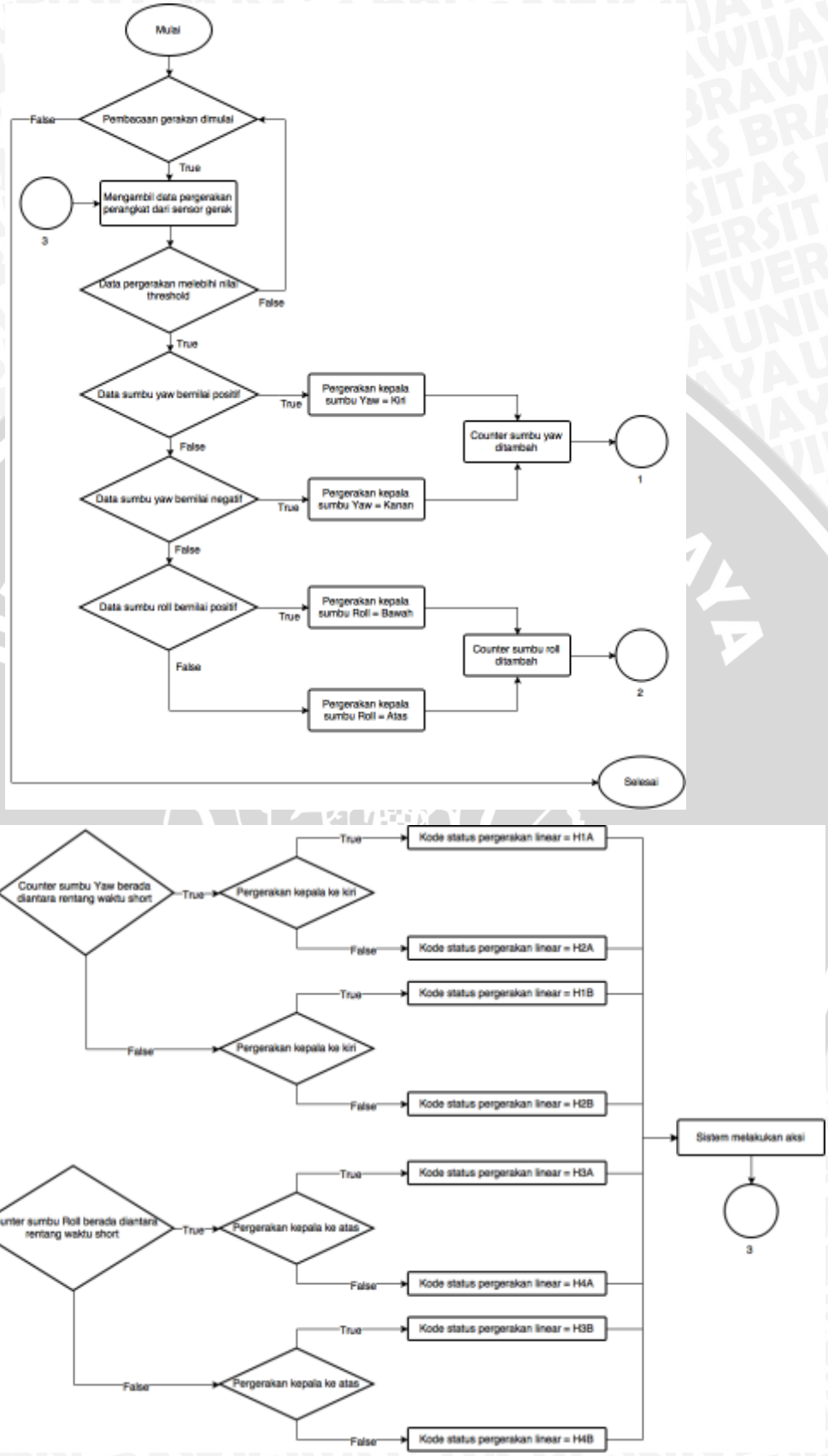
   Setelah arah pergerakan kepala berhasil ditentukan, selanjutnya sistem akan menentukan durasi pergerakan kepala yang dilakukan oleh pengguna. Durasi ini mengacu kepada seleksi kondisi antara nilai variabel *counter* yang didapatkan pada tahapan sebelumnya dengan tiga buah *threshold* durasi pergerakan yang telah ditentukan di awal. Fungsi dari ketiga *threshold* tersebut ialah untuk menentukan jenis durasi pergerakan kepala yang dilakukan pengguna, yang dihitung mulai dari perputaran kepala pengguna dari titik awal hingga berputar kembali ke titik awal. Apabila nilai variabel *counter* tersebut berada diantara *threshold* pertama dan kedua, maka durasi pergerakan kepala yang dilakukan oleh pengguna merupakan durasi yang pendek, namun apabila *counter* tersebut berada diantara *threshold* kedua dan ketiga, maka durasi pergerakan kepala yang dilakukan oleh pengguna merupakan durasi yang panjang.

5. Pemetaan kode status pergerakan linear

   Setelah arah dan durasi pergerakan kepala pengguna berhasil diketahui, maka sistem akan memetakan pergerakan kepala tersebut ke dalam bentuk kode status pergerakan linear. Kode status tersebut akan menjadi dasar bagi sistem dalam menjalankan aksi yang sesuai.

6. Sistem menjalankan aksi

   Pada tahap yang terakhir, sistem akan menjalankan aksi sesuai dengan kode status pergerakan linear yang telah dipetakan sebelumnya, sebagai contoh apabila sebuah kode status berkorelasi dengan aksi memindahkan kursor, maka sistem akan melakukan aksi perpindahan kursor, begitupula juga dengan aksi – aksi sistem yang lainnya. Perlu diketahui bahwa seluruh tahapan yang telah dijelaskan mulai tahap 1 sampai dengan 6 akan diulangi secara terus menerus selama aplikasi dibuka dan tidak diberhentikan secara paksa oleh pengguna. Adapun *flowchart* dari algoritme HEMOCS ditujukan pada Gambar 2.7.



Gambar 2.7 *Flowchart* Algoritme HEMOCS

2.2 Kendali Pergerakan Linear

Kendali pergerakan linear merupakan sebuah sistem kendali yang menggunakan arah pergerakan garis lurus sebagai acuan dalam menentukan arah pergerakan yang diinginkan. Di dalam dunia komputer, kendali pergerakan linear biasa dijadikan sebagai metode untuk berpindah antar sejumlah objek di dalam sebuah sistem. Navigasi yang umum digunakan seperti navigasi ke arah kanan, kiri, atas, dan bawah. Salah satu perangkat yang memanfaatkan kendali pergerakan linear tersebut adalah Apple TV. Implementasi dari kendali pergerakan linear pada Apple TV dapat kita lihat pada *focus engine* yang ada di dalam sistem operasi tvOS.

Gambar 2.8 Penerapan *Focus Engine* pada tvOS

Sumber : (Bachand and Michela 2015)

Apple TV merupakan sebuah perangkat pemutar media digital serta perangkat hiburan yang mampu melakukan *stream* konten media melalui sebuah perangkat televisi yang mendukung koneksi kabel *High Definition Media Input* (HDMI). Pada perangkat yang mendukung sistem operasi iOS (iPhone, iPad, iPod touch), pengguna dapat berinteraksi secara langsung dengan sistem dengan bantuan layar sentuh yang tersedia pada perangkat. Tipe interaksi tersebut dinamakan dengan istilah *direct interaction*. Berbeda halnya pada perangkat Apple TV yang mendukung sistem operasi tvOS, pengguna tidak dapat berinteraksi secara langsung dengan sistem, sehingga Apple menyertakan sebuah perangkat tambahan yang bernama *siri remote* yang digunakan sebagai subtitusi layar sentuh bagi pengguna dalam melakukan interaksi dengan sistem secara tidak langsung. Tipe interaksi tersebut dinamakan dengan *mediated interaction*.

Untuk mengakomodir *mediated interaction* pada sistem operasi tvOS, pengguna melakukan navigasi terhadap sebuah objek yang diinginkan menggunakan bantuan *siri remote*. *Siri remote* memiliki sejumlah tombol serta sensor yang digunakan untuk menerima masukan pengguna, antara lain sebuah sensor sentuh, enam buah tombol multi fungsi, serta sebuah mikrofon yang digunakan untuk melakukan komando sistem melalui suara. Dalam *focus-based interaction* yang diterapkan pada tvOS, objek yang menjadi saat itu menjadi *Point of Interest* (*PoI*) bagi pengguna akan dinyatakan dalam keadaan fokus (*focused state*) yang umumnya memiliki karakteristik perbesaran ukuran objek serta perubahan warna *background* objek, sedangkan objek yang tidak menjadi PoI bagi pengguna dinyatakan dalam keadaan tidak fokus (*unfocused state*) yang memiliki karakteristik berlawanan dengan objek yang menjadi PoI. Ketika sebuah

objek sedang berada dalam keadaan fokus, pengguna dapat memindahkan fokus tersebut kepada objek yang lain dengan melakukan navigasi menggunakan bantuan sensor sentuh pada *siri remote*. Ketika pengguna melakukan navigasi antar objek yang ada di dalam sistem, tvOS akan memicu perubahan fokus pada objek yang dilewati oleh pengguna, sehingga pengguna akan tahu bahwa ia sedang melakukan navigasi antar objek yang ada di dalam sistem. Objek yang berada pada kondisi fokus juga digunakan sebagai target dari setiap aksi yang dilakukan oleh pengguna, sebagai contoh jika objek yang menjadi fokus pengguna saat itu adalah sebuah tombol, maka aksi pengguna akan dipicu ketika ia memilih tombol tersebut melalui *siri remote*.

Seluruh proses tersebut dilakukan dengan adanya *focus engine*, yaitu sebuah *engine* pada tvOS yang berfungsi dalam menangani seluruh proses terkait navigasi dan perubahan fokus akan objek yang terlihat pada layar televisi pengguna. *Engine* tersebut mampu menerjemahkan gerak sentuh tangan pengguna berupa gerak sentuh ke kiri, kanan, atas, atau bawah menjadi sebuah masukan bagi sistem dalam memindahkan fokus terhadap objek yang diinginkan. Gambar 2.8 merupakan sebuah ilustrasi penerapan *focus engine* pada tvOS yang sedang menunjukkan fokus pada sebuah tombol 'C'. Jika pengguna melakukan gerak sentuh ke kiri pada *siri remote*, maka akan memicu perubahan fokus pada objek asal 'C 'yang menyebabkan objek tujuan 'B' berubah keadaannya menjadi *focused state* dan objek 'C' menjadi *unfocused state*, begitu pula seterusnya.

2.3 Sensor Gerak

Sensor gerak yang ada pada sebuah perangkat bergerak secara umum terdiri atas *accelerometer* dan *gyroscope*, dimana kedua sensor ini bertugas dalam memberikan data seputar pergerakan yang terjadi pada suatu perangkat. Adapun Gambar 2.9 merupakan ilustrasi sumbu kartesian yang terdapat pada *accelerometer* dan *gyroscope.*

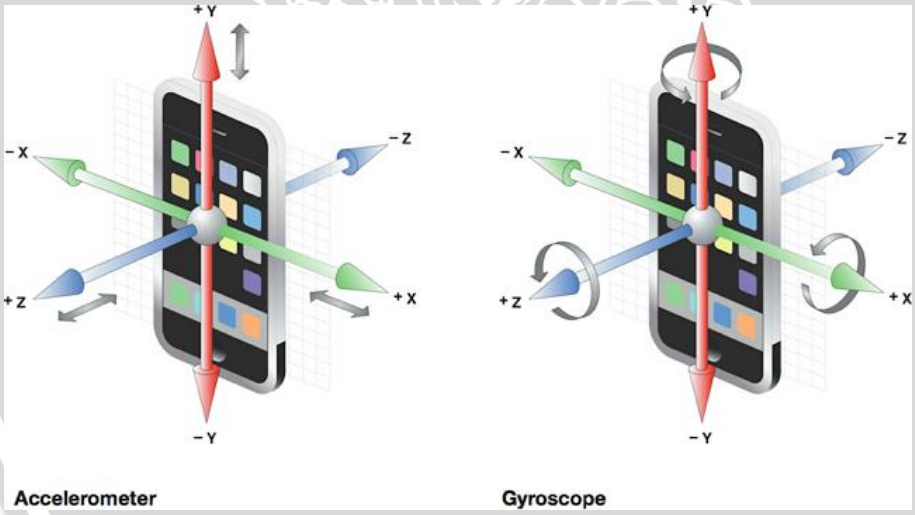
2.3.1 Accelerometer

Accelerometer merupakan sebuah sensor yang di desain untuk mengukur nilai percepatan non gravitasional dari suatu perangkat (Livescience 2016). *Accelerometer* menggunakan respon getaran sebagai metode dalam mendeteksi suatu gerakan, seperti gerakan diam hingga bergerak dengan kecepatan tertentu. Jenis *accelerometer* yang ada di dalam perangkat bergerak iOS ialah jenis *three-axis accelerometer,* yaitu *accelerometer* yang dapat menggambarkan rotasi perangkat dalam istilah *roll*, *pitch*, dan *yaw*. Jika kita memegang perangkat dalam orientasi *potrait*, maka *roll* akan menggambarkan sudut rotasi terhadap sumbu yang membentang di bagian atas dan bawah telepon, lalu *pitch* menggambarkan sudut rotasi tentang sumbu yang membentang di sisi perangkat (sisi dimana tombol volume perangkat berada), dan *yaw* menggambarkan sudut rotasi tentang sumbu yang membentang melalui bagian depan dan belakang perangkat (BAKER 2016). Dengan ketiga nilai ini, kita lalu dapat menentukan bagaimana pengguna memegang perangkat mereka dengan mengacu pada level

dasar, yang mana nilai tersebut diberikan oleh sensor masih dalam bentuk skala radian.

2.3.2 Gyroscope

Gyroscope merupakan sebuah sensor yang menggunakan gaya gravitasi bumi dalam membantu menentukan orientasi dari suatu perangkat (Livescience 2016). Berbeda dengan *accelerometer*, *gyroscope* dapat mengukur nilai percepatan gravitasional dari suatu perangkat, sehingga sensor ini mampu menentukan gerakan sesuai dengan gaya gravitasi yang dialami oleh perangkat. Sumbu kartesian yang digunakan *gyroscope* dalam menentukan pergerakan perangkat juga identik dengan sumbu kartesian yang terdapat pada *accelerometer*. Pada kenyataannya, kebanyakan perangkat yang difungsikan sebagai alat pendeteksi pergerakan memanfaatkan data yang diperoleh dari *gyroscope* dan *accelerometer* untuk memberikan keluaran data pergerakan yang relatif lebih akurat. Data pergerakan yang didapatkan kemudian di proses menggunakan algoritme tertentu untuk mengetahui serta mendeteksi sejumlah hal terkait dengan pergerakan yang terjadi pada perangkat, seperti orientasi, kecepatan linier, akselerasi, dan lainnya.



Gambar 2.9 Sumbu Kartesian *Accelerometer* dan *Gyrsocope*

Sumber : (Apple 2016)

2.4 Pemrograman *Objective-C*

Objective-C merupakan bahasa pemrograman berorientasi objek yang menjadi bahasa utama yang digunakan oleh perusahaan Apple dalam membangun sistem operasi macOS dan iOS, serta *Application Programming Interface* (API) mereka yaitu *Cocoa* dan *Cocoa Touch* sebelum adanya introduksi bahasa pemrograman baru mereka, Swift (Apple 2016). Bahasa ini dibangun pada awal tahun 1980 oleh Brad Cox dan Tom Love di perusahaan mereka yang

bernama Stepstone yang selanjutnya dipopulerkan oleh NeXT, sebuah perusahaan startup yang dibangun oleh Steve Jobs semasa merinits karirnya, yang juga menjadi sejarah dari perusahaan miliknya sendiri, yaitu Apple Inc. Bahasa ini juga merupakan *superset* dari bahasa pemrograman C, yang artinya bahwa bahasa ini memberikan kemampuan bagi *programmer* untuk melakukan *Object-Oriented Programming* (OOP) serta *runtime* secara dinamis.

2.4.1 Struktur *Class*

Sebuah program *Objective-C* secara logis dibagi menjadi tiga buah bagian, yaitu bagian @interface, @implementation, dan bagian program. Bagian @interface mendeskripsikan *class* dan fungsinya, lalu bagian @implementation mendeskripsikan data dan kode aktual yang merupakan implementasi dari fungsi yang dideklarasikan pada bagian @interface, dan terakhir yaitu bagian program yang berisi kode program untuk melaksanakan tujuan program yang dimaksud. Deklarasi sebuah *class* di dalam bahasa pemrograman *Objective-C* dipisahkan ke dalam dua buah bagian *file* yang terdiri atas bagian *header* dan *implementation*. Deklarasi serta implementasi antarmuka akan dilakukan di dalam sebuah *file* bernama Main.storyboard. Berikut ini merupakan penjelasan singkat dari masing – masing bagian tersebut:

1. *Header*

   Header memiliki ekstensi *file* .h, dimana *file* ini berisi deklarasi sebuah *class* yang akan didefinisikan di dalam bagian *implementation.* Deklarasi sebuah *class* meliputi deklarasi variabel, struktur data, objek, dan fungsi. Bagian *header* memiliki sifat *public*, yang berarti bahwa seluruh deklarasi *class* yang ada di dalamnya dapat diakses atau dimodifikasi oleh *class* yang lain secara bebas. Untuk melakukan hal tersebut, seorang *programmer* harus menyertakan (*import*) *header class* yang diinginkan di dalam *header class* yang lain terlebih dahulu. Setelah itu barulah mereka dapat mengakses atau memodifikasi variabel, struktur data, objek, atau fungsi yang dideklarasikan di dalam *header* yang diinginkan pada bagian *implementation class* yang lain.

2. *Implementation*

   Implementation memiliki ekstensi *file* .m, dimana *file* ini berisi definisi sebuah *class* yang mengacu pada *file header* tertentu. Definisi *class* disini memiliki artian implementasi kode program secara utuh dari sebuah *class*. Bagian *implementation* memiliki sifat *private*, yang berarti bahwa seluruh variabel, struktur data, objek, dan fungsi yang dideklarasikan di dalamnya hanya dapat diakses dan dimodifikasi di dalam bagian *implementation* itu sendiri.

3. Main.storyboard

Main.storyboard merupakan sebuah *file* yang berisikan seluruh implementasi antarmuka sistem beserta alirannya. Sebuah antarmuka / *view* yang ada di dalam Main.storyboard akan berasosiasi dengan sebuah *class* tertentu, dimana *class* tersebut juga merupakan *class* turunan dari *class* UIViewController.

2.4.2 *Framework* UIKit

Terdapat sejumlah *class* utama yang sering digunakan dalam proses pengembangan sistem menggunakan bahasa pemrograman *Objective-C*. Mayoritas *class* tersebut berasal dari *framework* UIKit, yaitu sebuah *framework* yang menyediakan infrastuktur penting dalam hal manajemen antarmuka aplikasi, *event-handling* yang digunakan dalam merespon *input* pengguna, dan model aplikasi yang digunakan dalam berinteraksi dengan sistem. Berikut ini sejumlah *class* yang ada di dalam *framework* UIKit beserta penjelasan singkatnya:

1. UIButton

   Merupakan sebuah objek *view* yang menjalankan suatu kode program tertentu sebagai respon terhadap interaksi yang dilakukan oleh pengguna.

2. UIColor

   Objek yang digunakan untuk menyimpan data warna beserta sejumlah atribut warna yang terkait (*alpha*, dll).

3. UIImage

   Objek yang digunakan untuk mengelola data gambar di dalam aplikasi.

4. UILabel

   Merupakan sebuah *view* yang menampilkan satu baris atau lebih teks *read-only*.

5. UINavigationController

   Class yang digunakan dalam mengelola navigasi antar *view* yang ada di dalam aplikasi secara hierarkis.

6. UIView

   Merupakan sebuah *view* yang mendefinisikan area *rectangular* pada layar perangkat dan *interface* untuk mengelola konten yang ada di dalam area tersebut.

7. UIViewController

   Menyediakan infrastruktur untuk mengelola *view* yang ada di dalam aplikasi.

2.4.3 Siklus Hidup *View*

Terdapat enam buah siklus hidup (*life cycle*) yang dialami oleh sebuah *view* di dalam lingkungan pengembangan aplikasi *platform* iOS. Siklus hidup ini direpresentasikan ke dalam bentuk fungsi yang akan dipanggil secara otomatis oleh sistem setiap kali sebuah *view* di *load*, dimunculkan, maupun disembunyikan dari layar perangkat. Fungsi tersebut terdiri atas viewDidLoad, viewWillAppear, viewDidAppear, viewWillDisappear, viewDidDissapear, dan didReceiveMemoryWarning. Keenam fungsi tersebut melekat pada *class* UIViewController yang terdapat di dalam *framework* UIKit. Adapun siklus hidup *view* ditujukan pada Gambar 2.10.

a. ViewDidLoad

Fungsi viewDidLoad dipanggil ketika sebuah *view* di load ke memori perangkat. Fungsi ini cocok digunakan sebagai inisiasliasi awal sebuah *view* dan hanya dipanggil satu kali selama *view* tersebut masih berada di dalam memori.

b. ViewWillAppear

Fungsi viewWillAppear dipanggil sesaat sebelum *view* muncul di layar, sehingga cocok digunakan untuk menyembunyikan, menampilkan *field,* atau sejumlah operasi ringan yang akan dilakukan sesaat sebelum *view* tersebut muncul di layar. Karena pengguna sangat mungkin berpindah antar *view* satu dengan *view* yang lain, maka fungsi ini akan dipanggil setiap kali *view* dimunculkan di layar.

c. ViewDidAppear

Fungsi viewDidAppear akan dipanggil setelah *view* berhasil dimunculkan di layar, cocok digunakan untuk memulai sebuah animasi atau melakukan *load* terhadap data eksternal yang berasal dari API tertentu.

d. ViewWillDissapear

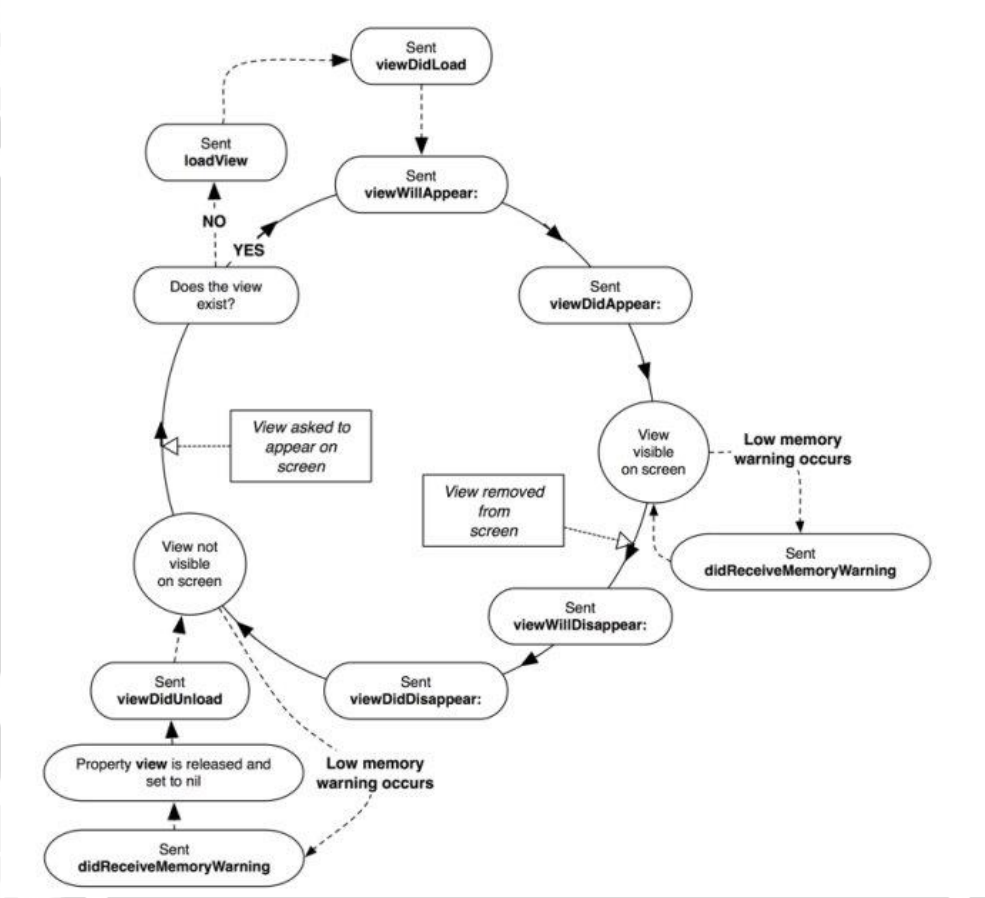
Fungsi viewWillDisappear dipanggil sesaat setelah *view* disembunyikan dari layar, sehingga cocok digunakan untuk melakukan sejumlah operasi ringan yang dilakukan sesaat setelah *view* tersebut menghilang dari layar pengguna. Karena pengguna sangat mungkin berpindah antar *view* satu dengan *view* yang lain, maka fungsi ini akan dipanggil setiap kali *view* tersebut menghilang dari layar.

e. ViewDidDissapear

Fungsi viewDidDisappear akan dipanggil setelah *view* berhasil disembunyikan dan tidak lagi terlihat di layar.

f. DidReceiveMemoryWarning

Fungsi didReceiveMemoryWarning akan dipanggil ketika aplikasi menerima sebuah peringatan memori perangkat yang rendah, sehingga fungsi ini biasanya digunakan untuk melepaskan sejumlah memori yang umumnya berasal dari penggunaan stuktur data, *cache,* ataupun data yang berasal dari sebuah aktivitas *networking* tetentu.



Gambar 2.10 Siklus Hidup View dalam Platform iOS

2.5 Framework Core Motion

Framework core motion merupakan sebuah *framework* yang dikembangkan oleh tim *developer* Apple untuk membantu sebuah aplikasi dalam menerima data dan juga memproses data terkait pergerakan yang berasal dari sensor gerak perangkat. Pengembang aplikasi dapat menggunakan data yang berasal dari sensor *accelerometer*, *gyroscope*, *magnetometer*, atau ketiganya di

dalam aplikasi atau permainan yang menggunakan gerakan sebagai masukan atau sebagai penambah *user experience* dari aplikasi yang dikembangkan. Selain itu, pengembang aplikasi juga dapat memanfaatkan *framework* ini dalam mengamati respon dari perangkat iOS pada saat terjadi pergerakan atau perubahan orientasi (Allan 2011).

Berikut ini merupakan sejumlah *class* yang ada di dalam *framework core motion* yang umum digunakan dalam mendapatkan data terkait dengan pergerakan perangkat beserta penjelasan singkatnya:

1. CMAttitude

   Objek yang diinstansiasi dari *class* CMAttitude merepresentasikan data tingkah laku pergerakan perangkat (*attitude*) yang terjadi dalam sebuah waktu tertentu. *Attitude* ini akan mengacu pada orientasi tubuh yang relatif terhadap kerangka acuan yang diberikan.

2. CMDeviceMotion

   Objek yang diinstansiasi dari *class* CMDeviceMotion mengenkapsulasi data yang terkait dengan tingkah laku pergerakan, kecepatan rotasi, dan percepatan dari perangkat.

3. CMMotionManager

   Objek yang diinstansiasi dari *class* CMMotionManager bertindak sebagai pintu gerbang dalam mengakses layanan terkait pergerakan perangkat yang disediakan oleh iOS. Layanan ini menyediakan aplikasi dengan data *attitude* (tingkah laku pergerakan perangkat), *rotation rate* (tingkat rotasi), *magnetic field* (medan magnet yang telah terkalibrasi), *gravity* (arah gravitasi), dan *acceleration rate* (percepatan yang dilakukan pengguna terhadap perangkat), yang mana data tersebut berasal dari sensor *accelerometer* dan (pada beberapa model) juga berasal dari *magnetometer* dan *gyroscope* perangkat.

2.6 *Head-Mounted Display* (HMD)

Sebuah *Head-Mounted Display* (HMD) merupakan sebuah perangkat yang digunakan sebagai tampilan layar yang dikenakan di kepala seseorang. Sejumlah HMD yang ada saat ini bahkan dirancang khusus untuk mendukung pemanfaatan teknologi *augmented reality* (AR) di masa yang akan datang. Salah satu contoh perangkat HMD yang terkenal di kalangan masyarakat saat ini adalah Google Cardboard. Ilustrasi Google Cardboard ditujukan pada Gambar 2.11. Google Cardboard merupakan sebuah perangkat yang diciptakan oleh Google untuk memenuhi kebutuhan akan perkembangan teknologi *virtual* dan *augmented reality*. Dengan adanya ponsel yang memiliki kemampuan untuk menjalankan teknologi VR atau AR serta perangkat HMD seperti Cardboard, pengguna dapat menikmati aplikasi yang memanfaatkan keberadaan teknologi tersebut melalui

layar ponsel yang diletakkan di dalam Cardboard. Beberapa contoh aplikasi VR atau AR yang memanfaatkan perangkat Cardboard dalam penggunaannya antara lain aplikasi Google *Maps StreetView,* YouTube via VR, *Camera* 360, dan masih banyak lagi lainnya.

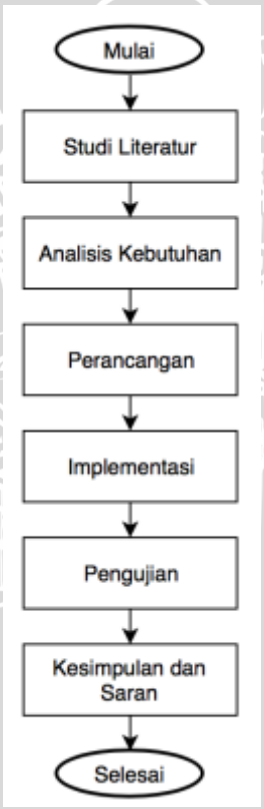
Gambar 2.11 Google Cardboard

Sumber: (Burak Kanber, Engineer 2016)

BAB 3 METODOLOGI

3.1 Strategi dan Rancangan Penelitian

Tahapan yang akan ditempuh dalam penelitian ini meliputi tahap studi literatur, analisis kebutuhan, perancangan, implementasi, pengujian, dan terakhir pengambilan kesimpulan dan saran yang seluruhnya ditujukan pada Gambar 3.1. *Software Development Life Cycle* (SDLC) yang diterapkan pada pengembangan sistem ke depan adalah model SDLC *waterfall.* SDLC ini memiliki konsep bahwa pengerjaan tahapan selanjutnya akan dimulai setelah tahapan sebelumya telah benar – benar selesai dilakukan, dengan kata lain setiap tahapan yang ada di dalam SDLC tersebut akan dikerjakan secara sekuensial. Alasan peneliti menggunakan pendekatan *waterfall* antara lain ialah kebutuhan yang digunakan pada penelitian ini bersifat jelas dan tetap serta waktu pengembangan sistem yang relatif singkat.



Gambar 3.1 Strategi dan Rancangan Penelitian

3.1.1 Studi Literatur

Tahap ini menjelaskan pencarian serta pembelajaran literatur yang bersumber dari buku, jurnal, dan penelitian sebelumnya yang terkait dengan pengembangan sistem yang memanfaatkan kendali pergerakan kepala. Studi ini dilakukan untuk memperkaya pengetahuan akan teori serta menjadi acuan dalam pengerjaan penelitian kedepan. Berikut ini adalah beberapa bidang ilmu yang akan dipelajari dalam penelitian ini, antara lain:

1. *Head Movement Control System* (HEMOCS)
2. Kendali Pergerakan Linear
3. Sensor Gerak
4. Pemrograman *Objective*-C
5. *Framework Core Motion*
6. *Head-Mounted Display* (HMD)

3.1.2 Analisis Kebutuhan

Tahap ini akan menjelaskan seputar analisis kebutuhan dari sistem yang akan dikembangkan. Tujuan dari proses analisis kebutuhan ialah untuk mendapatkan kebutuhan sistem yang tepat dalam menyelesaikan permasalahan ada. Proses ini dilakukan dengan cara menganalisis sejumlah fitur dasar yang ada pada solusi *Health Communication Board* (HCB) yang diterapkan secara manual. Setelah proses analisis tersebut dilakukan, peneliti akan menggali sejumlah kebutuhan yang tepat untuk diterapkan pada sebuah aplikasi perangkat bergerak yang interaksinya dikendalikan menggunakan gerakan kepala. Kebutuhan yang didapatkan meliputi identifikasi pelaku sistem (aktor), kebutuhan fungsional, dan kebutuhan non fungsional dari sistem.

3.1.3 Perancangan

Tahap ini akan menjelaskan seputar proses perancangan dari sistem. Tujuan dari perancangan ialah untuk menerjemahkan kebutuhan yang telah didapatkan dari proses analisis kebutuhan menjadi bentuk spesifikasi sistem yang lebih detil. Perancangan sistem ini terdiri atas perancangan arsitektur, antarmuka, penentuan jenis pergerakan kepala, pemetaan kendali pergerakan linear, perancangan metode deteksi pergerakan kepala, dan pemodelan UML yang terdiri atas *use case*, skenario *use case*, dan diagram aktivitas.

3.1.4 Implementasi

Tahap ini menjelaskan proses implementasi sistem yang mengacu pada tahap perancangan yang telah dilakukan sebelumnya. Tahap ini akan memaparkan implementasi kode program dari sejumlah fungsi utama yang mendukung proses berjalannya sistem. Aplikasi *Health Communication Board* akan diimplementasikan di dalam *platform* iOS dengan menggunakan bahasa pemrograman *Objective*-C serta bantuan *framework core motion* yang telah tersedia di IDE Xcode milik Apple.

3.1.5 Pengujian

Tahap ini menjelaskan proses pengujian sistem untuk mengukur kesesuaian antara kebutuhan yang telah didapatkan diawal dengan implementasi yang telah dibuat serta mengetahui tingkat usabilitas dari sistem. Terdapat dua buah pengujian, yaitu pengujian fungsional yang terkait dengan kebutuhan fungsional sistem serta non fungsional yang terkait dengan aspek akurasi dan usabilitas. Pengujian fungsional akan dilakukan dengan metode pengujian *black box*, sedangkan pengujian non fungsional dilakukan dengan menguji usabilitas sistem secara langsung pada responden yang dilakukan dengan menggunakan sejumlah kuisioner. Jawaban kuisioner responden akan dinilai dengan menggunakan metode skala *likert* yang akan dianalisis untuk mendapatkan informasi serta kesimpulan atas pengujian yang dilakukan.

3.1.6 Pengambilan Kesimpulan dan Saran

Tahap ini berisi kesimpulan terkait dengan hasil akhir penelitian yang menjawab rumusan masalah yang telah didefinisikan diawal. Setelah kesimpulan telah didapatkan, dilanjutkan pula dengan saran yang bertujuan untuk memperbaiki kekurangan yang ada di dalam penelitian ini dan memberikan pertimbangan atas pengembangan metode penelitian yang akan digunakan kembali oleh peneliti selanjutnya di masa yang akan datang.

BAB 4 ANALISIS KEBUTUHAN

4.1 Gambaran Umum Aplikasi

Aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS merupakan aplikasi perangkat bergerak yang dikembangkan untuk membantu komunikasi antara pengguna yang mengalami keterbatasan fisik di bagian anggota gerak tangan dengan orang lain secara normal. Aplikasi ini secara umum akan mengacu pada *Health Communication Board* yang dibuat oleh perusahaan VIDATAK bernama EZ *Picture Board*. Adapun Gambar 4.1 merupakan ilustrasi dari EZ *Picture Board*.

Gambar 4.1 VIDATAK EZ *Picture Board*

Sumber: (VIDATAK Innovation in Patient Communication 2016)

Ketika aplikasi dijalankan oleh pengguna, ia akan mendapatkan sebuah tampilan dunia nyata yang ditangkap oleh kamera perangkat, dimana pada tampilan itu juga ditambahkan sejumlah elemen grafis yang diproses oleh perangkat bergerak. Di dalam tampilan tersebut, pengguna dapat melihat sejumlah aksi layaknya yang ada pada EZ *Picture Board*. Pengguna dapat berpindah antar aksi yang ada dengan bantuan sebuah kursor. Ketika pengguna memilih sebuah aksi yang ia inginkan, sistem akan memutar suara terhadap aksi yang dipilih melalui *speaker* yang ada pada perangkat. Selain memilih aksi, pengguna juga dapat menyusun suatu kata yang terdiri atas sejumlah karakter tertentu. Apabila ia melakukan kesalahan dalam memasukkan sebuah karakter, ia dapat menghapus karakter tersebut. Pengguna juga dapat memutar suara terhadap kata yang telah ia susun melalui *speaker* yang ada pada perangkat. Untuk memudahkan pencarian terhadap aksi yang diinginkan, seluruh aksi yang ada di dalam sistem akan dibagi ke dalam empat buah halaman, dimana tiga

halaman akan berisi sejumlah aksi yang spesifik terhadap jenis aksi tertentu dan sebuah halaman yang berisi sebuah *layout keyboard*. Karena aplikasi ini ditujukan kepada pengguna yang mengalami keterbatasan fisik di bagian anggota gerak tangan, maka seluruh bentuk interaksi yang dilakukan oleh pengguna terhadap aplikasi, seperti memindahkan posisi kursor dan sebagainya akan dilakukan dengan menggunakan gerakan kepala pengguna.

4.2 Identifikasi Aktor

Aktor merupakan seseorang ataupun sistem eksternal yang dapat berinteraksi dengan sistem. Adapun aktor sistem yang dapat diidentifikasi di ditujukan pada Tabel 4.1.

Tabel 4.1 Aktor Sistem

Nama Aktor	Deskripsi
Pengguna	Merupakan orang yang memiliki keterbatasan fisik di bagian anggota gerak tangan. Orang yang memiliki kondisi fisik normal juga termasuk dalam kategori pengguna sistem.

4.3 Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem merupakan kebutuhan yang merepresentasikan hal – hal apa saja yang dapat dilakukan oleh sistem (Kurniawan 2012). Kebutuhan tersebut harus mencakup fungsi / kapabilitas yang harus dijalankan oleh sistem untuk dapat menyeselaikan permasalahan yang diberikan diawal. Pemberian kode kebutuhan pada dokumen dilakukan untuk menjaga konsistensi terkait dengan kebutuhan yang ada, mulai dari proses perancangan hingga proses pengujian selesai dilakukan Adapun deskripsi kebutuhan sistem serta spesifikasinya ditujukan pada Tabel 4.2.

Tabel 4.2 Kebutuhan Fungsional Sistem

NO	Kode Kebutuhan	Kebutuhan Fungsional
1.	SRS-HCB-1.0	Sistem menyediakan mekanisme untuk memindahkan posisi halaman sesuai dengan keinginan aktor.
2.	SRS-HCB-2.0	Sistem menyediakan mekanisme untuk memindahkan posisi kursor yang ada di dalam sebuah halaman sesuai dengan keinginan aktor.
3.	SRS-HCB-3.0	Apabila aktor memilih sebuah aksi tertentu, sistem harus mampu memutar suara aksi tersebut melalui

		speaker perangkat.
4.	SRS-HCB-4.0	Apabila aktor sedang menyusun suatu kata tertentu, sistem menyediakan mekanisme untuk menyimpan kata yang disusun oleh aktor tersebut.
5.	SRS-HCB-5.0	Apabila aktor melakukan kesalahan dalam menyusun suatu kata, sistem menyediakan mekanisme untuk menghapus karakter terakhir yang dimasukkan oleh aktor.
6.	SRS-HCB-6.0	Apabila aktor memutuskan untuk memutar suara kata yang telah ia susun, maka sistem harus mampu memutar suara kata tersebut melalui *speaker* perangkat.

BAB 5 PERANCANGAN

5.1 Perancangan Arsitektur Sistem

Arsitektur sistem yang diterapkan pada pengembangan aplikasi *Health Communcation Board* ini adalah arsitektur MVC yang diterapkan secara *native* pada sistem operasi iOS. Aplikasi *native* merupakan aplikasi yang dikembangkan untuk digunakan pada *platform* tertentu saja, sehingga aplikasi ini mendapatkan keuntungan berupa pemanfaatan fitur, sensor, serta perangkat lunak lain yang juga terpasang pada *platform* tersebut. Kebutuhan akan data yang berasal dari sensor gerak perangkat membuat peneliti memutuskan untuk mengembangkan sistem secara *native.* Implementasi arsitektur *native* pada *platform* iOS dilakukan dengan menggunakan bahasa pemrograman *Objective-C* buatan Apple.

Arsitektur MVC (*Model*, *View*, *Controller*) merupakan arsitektur yang secara *default* diterapkan di dalam lingkungan pengembangan aplikasi perangkat bergerak *platform* iOS. Arsitektur ini terdiri atas tiga buah bagian, yaitu *model* yang merepresentasikan data dan kegiatan inti yang dilakukan oleh sistem, *view* yaitu bagian yang merepresentasikan tampilan antarmuka sistem, dan *controller* yaitu bagian yang berisi logika antarmuka yang berfungsi untuk menghubungkan komunikasi yang terjadi antara *view* dengan *model* yang ada di dalam sistem. Proses utama yang dilakukan oleh sistem adalah melakukan pembacaan data pergerakan perangkat kemudian menentukan aksi sistem yang tepat berdasarkan data tersebut. Proses pembacaan serta penentuan aksi berdasarkan data pergerakan perangkat dilakukan di dalam *controller*, sedangkan interaksi yang terjadi antara pengguna dengan sistem dilakukan melalui perantara *view. View* akan menampilkan tampilan kamera serta sejumlah ikon yang dapat dipilih oleh pengguna menggunakan pergerakan kepala, dimana pergerakan tersebut akan menjadi *input* bagi sistem. *Output* yang diberikan sistem kepada pengguna berupa pembaharuan *view* terkait dengan posisi ikon yang dipilih serta *output* suara ketika pengguna memilih sebuah ikon tertentu di dalam *view*.

5.2 Perancangan *User Experience*

Perancangan *user experience* sistem terdiri atas sejumlah sub-perancangan yang lain, yaitu perancangan ikon komunikasi, perancangan *tab*, perancangan interaksi, dan perancangan *user interface*. Berikut ini akan dijelaskan secara lebih detil dari setiap sub-perancangan yang ada:

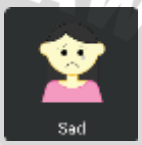
5.2.1 Perancangan Ikon Komunikasi

Ikon komunikasi merupakan bentuk representasi aksi pengguna yang terdapat pada EZ *Picture Board*. Ikon komnunikasi ini juga bertindak sebagai objek kursor di dalam sistem, dimana arah perpindahan kursor nantinya dikendalikan menggunakan gerakan kepala pengguna. Sebuah ikon komunikasi tersusun atas gambar serta nama aksi yang bersangkutan. Gambar yang ada di dalam sebuah ikon komunikasi mewakili sebuah aksi pengguna tertentu, dimana

pemberian gambar ini bertujuan untuk memudahkan pengguna dalam mencari aksi yang ia inginkan. Di bawah gambar tersebut, terdapat nama aksi yang mewakili gambar yang berada diatasnya. Untuk mengetahui dan membedakan kursor yang sedang dipilih pengguna dengan yang tidak, sebuah mekanisme pemberian warna *background* berbeda pada ikon komunikasi akan diberikan, dimana ikon komunikasi yang sedang dipilih oleh pengguna (*selected state*) akan diberi warna *background* hitam dengan warna teks putih, sedangkan ikon komunikasi yang sedang tidak dipilih pengguna (un*selected state*) akan diberi warna *background* putih dengan warna teks hitam. Adapun perancangan ikon komunikasi yang akan digunakan di dalam sistem ditujukan pada Tabel 5.1.

Tabel 5.1 Perancangan Ikon Komunikasi

NO	Nama Ikon Komunikasi	Ilustrasi Ikon Komunikasi (*selected*)	Ilustrasi Ikon Komunikasi (*unselected*)
1.	*Short of breath* (sesak nafas)		
2.	*In pain* (mengalami kesakitan)		
3.	*Choking* (tersedak)		
5.	*Hungry* (lapar)		
6.	*Thirsty* (haus)		
7.	*Tired* (lelah)		

8.	*Dizzy* (pusing)		
9.	*Afraid* (takut)		
10.	*Frustrated* (frustasi)		
11.	*Sad* (sedih)		
12.	*To be suctioned* (dibantu minum)		
13.	*Lip moistened* (bibir dilembabkan)		
14.	*To be comforted* (dibuat nyaman)		
15.	*To sleep* (tidur)		
16.	*To watch TV* (menonton TV)		
17.	*To call light remote* (mengambil remot cahaya)		

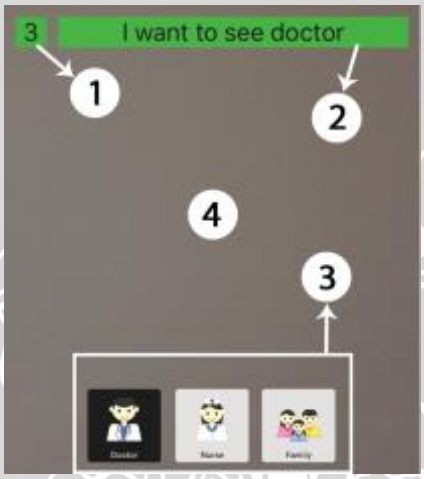
18.	*To be quiet* (keadaan sunyi atau diam)		
19.	*To turn lights on* (lampu dihidupkan)		
20.	*To turn lights off* (lampu dimatikan)		
21.	*To get out of bed* (bangun dari tempat tidur)		
22.	*Doctor* (dokter)		
23.	*Nurse* (suster)		
24.	*Family* (keluarga)		

5.2.2 Perancangan *Tab*

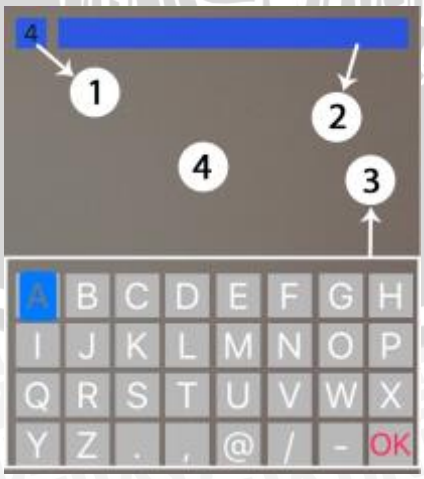
Halaman atau *tab* yang ada di dalam sistem digunakan untuk mengelompokkan ikon komunikasi sesuai dengan jenis aksinya. Terdapat empat buah *tab* di dalam sistem yang mengacu kepada empat buah jenis aksi yang terdapat pada EZ *Picture Board*. *Tab* pertama berisi sepuluh buah ikon komunikasi yang merepresentasikan kondisi pengguna, lalu *tab* kedua yang berisi sepuluh buah ikon komunikasi yang merepresentasikan permintaan pengguna, kemudian *tab* ketiga yang berisi tiga buah ikon komunikasi yang merepresentasikan permintaan pengguna untuk bertemu dengan orang lain, dan terakhir *tab* keempat yang berisi sebuah papan ketik atau *keyboard* dengan *layout* alfabetik. Selain ikon komunikasi, setiap *tab* juga berisi nomor *tab*, label aksi pengguna, dan tampilan kamera. Gambar 5.1 sampai dengan Gambar 5.4 secara berurutan merupakan rancangan antarmuka dari *tab* pertama, kedua, ketiga, dan keempat yang ada di dalam sistem. Terlihat pada keempat gambar tersebut, bagian yang ditunjuk oleh angka 1 merupakan nomor *tab*. Nomor tersebut digunakan oleh pengguna untuk mengetahui posisi *tab* ia berada saat itu. Bagian yang ditunjuk oleh angka 2 merupakan label aksi pengguna. Label tersebut berisi nama aksi yang sedang dipilih oleh pengguna. Bagian yang ditunjuk oleh angka 3 merupakan lokasi dari ikon komunikasi atau *keyboard*. Bagian yang ditunjuk oleh angka 4 yang merupakan tampilan kamera. Tampilan tersebut berisi tampilan dunia nyata yang ditangkap oleh kamera perangkat yang digunakan pengguna dalam melihat dunia sekitarnya. Warna latar belakang dari nomor *tab* dan label aksi pengguna juga diberi warna tema yang berbeda untuk memudahkan pengguna dalam mengetahui posisi *tab* ia berada saat itu. Adapun warna tema yang digunakan adalah kombinasi empat warna utama secara berurutan, yaitu warna merah untuk *tab* pertama, warna kuning untuk *tab* kedua, warna hijau untuk *tab* ketiga, dan warna biru untuk *tab* keempat.

Gambar 5.1 Rancangan Antarmuka *Tab* Pertama

Gambar 5.2 Rancangan Antarmuka *Tab* Kedua



Gambar 5.3 Rancangan Antarmuka *Tab* Ketiga



Gambar 5.4 Rancangan Antarmuka *Tab* Keempat

5.2.3 Perancangan Interaksi

Jenis interaksi yang dilakukan oleh pengguna terhadap sistem adalah jenis interaksi secara tidak langsung (*mediated interaction*) yang dilakukan dengan menggunakan gerakan kepala. Aksi yang dapat dilakukan oleh pengguna terhadap sistem antara lain memindahkan posisi *tab*, memindahkan posisi kursor, memilih sebuah aksi tertentu, memilih karakter untuk menyusun sebuah kata, dan menghapus karakter. Ketika pengguna berada di dalam sebuah *tab* tertentu, ia dapat berpindah ke posisi *tab* sebelum dan sesudahnya dengan melakukan gerakan kepala ke kiri atau ke kanan. Selain itu, pengguna juga dapat memindahkan posisi kursor di dalam sebuah *tab* dengan melakukan gerakan kepala ke kiri, kanan, atas, atau bawah.

Untuk membedakan gerakan kepala pengguna yang digunakan untuk memindahkan posisi *tab* dengan memindahkan kursor, maka dibuatlah sebuah mekanisme pembacaan gerakan kepala secara pendek (*short*) dan panjang (*long*). Pergerakan kepala secara pendek akan digunakan untuk memindahkan posisi kursor yang ada di dalam sebuah *tab*, sedangkan pergerakan kepala secara panjang akan digunakan untuk berpindah antar *tab*, memilih aksi, memilih karakter, dan menghapus karakter. Arah serta durasi pergerakan kepala yang dilakukan oleh pengguna dalam berinteraksi dengan sistem kemudian dipetakan dalam bentuk kode status pergerakan linear, dimana kode status tersebut akan merepresentasikan jenis pergerakan kepala dan aksi yang akan dijalankan oleh sistem. Adapun pemetaan pergerakan kepala pengguna ditujukan pada Tabel 5.2 berikut:

Tabel 5.2 Pemetaan Pergerakan Kepala

NO	Kode Status Pergerakan	Jenis Pergerakan Kepala	Aksi yang Dilakukan
1	H1A	*Axial Left – Short*	Pindah kursor ke kiri
2	H1B	*Axial Left – Long*	Pindah ke *tab* sebelumnya
3	H2A	*Axial Right – Short*	Pindah kursor ke kanan
4	H2B	*Axial Right – Long*	Pindah ke *tab* selanjutnya
5	H3A	*Extension – Short*	Pindah kursor ke atas
6	H3B	*Extension – Long*	Menghapus karakter pada *tab* keempat
7	H4A	*Flexion – Short*	Pindah kursor ke bawah
8	H4B	*Flexion – Long*	Memilih aksi atau memilih karakter pada *tab* keempat

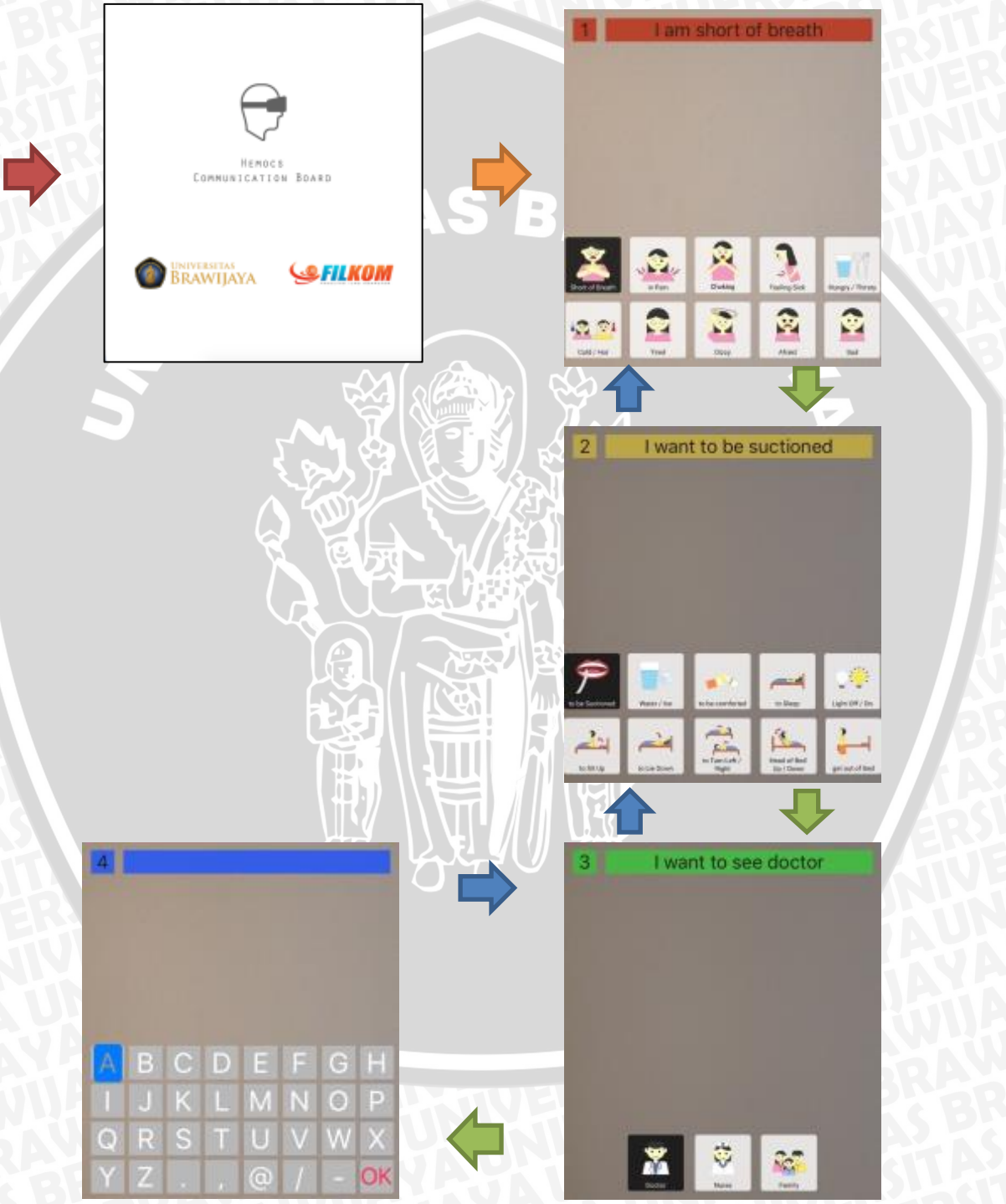
Interaksi pengguna yang digunakan untuk memindahkan posisi *tab* adalah gerakan kepala menoleh (*axial*) yang dilakukan dengan durasi *long*. Untuk berpindah ke posisi tab selanjutnya, pengguna diharuskan untuk melakukan gerakan kepala *axial right – long*, sedangkan untuk berpindah ke posisi *tab* sebelumnya, pengguna diharuskan melakukan gerakan kepala *axial left – long*. Untuk memilih sebuah aksi atau memilih karakter dalam menyusun suatu kata di dalam *tab* keempat, pengguna melakukan gerakan kepala dengan jenis *flexion – long*. Di dalam *tab* keempat, pengguna juga dapat melakukan gerakan kepala *extension – long* untuk menghapus karakter terakhir apabila ia melakukan kesalahan dalam menyusun kata.

Gambar 5.5 Contoh Perpindahan Kursor ke Arah Kanan

Adapun interaksi pengguna yang digunakan untuk memindahkan posisi kursor di dalam sebuah *tab* terdiri atas gerakan kepala menoleh (*axial*), melihat ke atas (*extension*), dan melihat ke bawah (*flexion*) yang dilakukan dengan durasi *short*. Gerakan kepala *axial right – short* dilakukan untuk memindahkan kursor ke arah kanan, kemudian gerakan kepala *axial left – short* dilakukan untuk memindahkan kursor ke arah kiri, lalu gerakan kepala *extension – short* atas dilakukan untuk memindahkan kursor ke arah atas, dan gerakan kepala *flexion – short* dilakukan untuk memindahkan kursor ke arah bawah. Perpindahan kursor juga memicu perubahan isi label yang ada di dalam sebuah *tab*, dimana isi label akan berasosiasi dengan aksi ikon komunikasi pada kursor yang sedang dipilih oleh pengguna. Gambar 5.5 merupakan contoh aksi perpindahan kursor ke kanan yang ada di dalam sebuah *tab*.

5.2.4 Perancangan *Screenflow*

Screenflow merupakan sebuah istilah yang digunakan untuk menggambarkan aliran antarmuka yang terjadi di dalam sebuah sistem. Adapun Gambar 5.6 merupakan ilustrasi aliran antarmuka yang ada di dalam sistem.



Gambar 5.6 Aliran Antarmuka Sistem

Pada Gambar 5.6, terlihat lima buah tampilan layar yang terdiri atas *splash screen*, *tab* pertama, *tab* kedua, *tab* ketiga, dan *tab* keempat. Kelima tampilan tersebut akan menjadi penyusun antarmuka dari sistem yang akan dikembangkan. Selain tampilan layar, terlihat pula delapan buah anak panah yang menunjukkan arah transisi antar tampilan layar yang ada. Delapan anak panah tersebut tersusun atas sebuah anak panah berwarna merah, sebuah anak panah berwarna jingga, tiga buah anak berwarna hijau, dan tiga buah anak panah berwarna biru. Warna yang terlihat dari setiap anak panah akan menunjukkan kondisi transisi yang berbeda pula. Anak panah berwarna merah menunjukkan kondisi ketika pengguna membuka aplikasi untuk pertama kalinya, anak panah berwarna jingga menunjukkan transisi dari *splash screen* ke posisi *tab* pertama, anak panah berwarna hijau menunjukkan transisi ke posisi *tab* yang selanjutnya, dan anak panah berwarna biru menunjukkan transisi ke posisi *tab* yang sebelumnya.

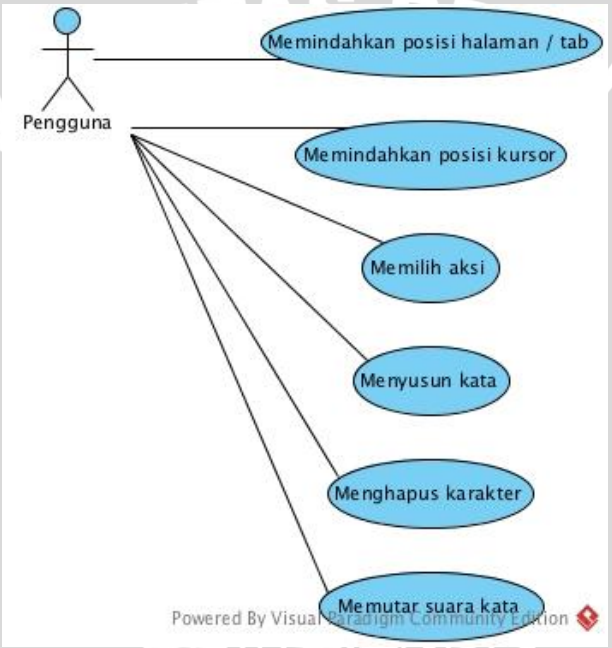
Ketika aplikasi pertama kali dibuka oleh pengguna, aplikasi akan menampilkan sebuah *splash screen* yang dilanjutkan dengan tampilan *tab* pertama. *Splash screen* merupakan sebuah antarmuka yang ditampilkan sesaat setelah aplikasi dibuka oleh pengguna. Dari posisi *tab* pertama, pengguna hanya dapat berpindah ke *tab* kedua. Apabila pengguna berada di posisi *tab* kedua, pengguna dapat memilih untuk berpindah ke posisi *tab* sebelumnya yaitu *tab* pertama atau ke *tab* selanjutnya yaitu *tab* ketiga. Apabila pengguna berada di posisi *tab* ketiga, pengguna dapat memilih untuk berpindah ke posisi *tab* sebelumnya yaitu *tab* kedua atau ke *tab* selanjutnya yaitu *tab* keempat. Dan terakhir, apabila pengguna berada di posisi *tab* keempat, pengguna hanya memilki pilihan untuk berpindah ke posisi *tab* sebelumnya yaitu *tab* ketiga. Dapat diketahui bahwa aplikasi ini belum menyediakan mekanisme perpindahan dari *tab* keempat ke *tab* pertama secara langsung atau dari *tab* pertama ke *tab* keempat secara langsung, sehingga apabila pengguna ingin kembali ke *tab* pertama dari *tab* keempat, maka pengguna harus melewati posisi *tab* ketiga dan kedua hingga *tab* pertama secara berurutan, begitupula juga sebaliknya untuk perpindahan dari *tab* pertama ke *tab* keempat.

5.3 *Unified Modeling Language* (UML)

Untuk memudahkan pemahaman akan perancangan sistem yang akan dikembangkan, sistem akan dimodelkan ke dalam bentuk diagram UML yang terdiri atas diagram *use case,* skenario *use case*, dan diagram aktivitas.

5.3.1 Diagram *Use Case*

Diagram *use case* berisi sejumlah aksi yang mendefinisikan interaksi yang terjadi antar seorang aktor terhadap sistem dalam mencapai suatu tujuan tertentu. Aktor disini dapat berupa manusia ataupun sistem eksternal. Diagram *use case* dari sistem ditunjukkan pada Gambar 5.7.



Gambar 5.7 *Use Case* Sistem

Di dalam sistem ini, terdapat seorang aktor yang bernama pengguna. Berdasarkan deskripsi kebutuhan fungsional sistem yang ada pada Tabel 4.2, aksi yang dapat dilakukan aktor terhadap sistem antara lain memindahkan posisi halaman / *tab*, memindahkan posisi kursor, memilih aksi, menyusun kata, menghapus karakter, dan memutar suara dari kata yang telah disusun oleh aktor.

5.3.2 Skenario *Use Case*

Skenario *use case* merupakan penjabaran lebih detil pada tiap *use case* yang ada di dalam sistem. Secara umum, skenario *use case* terdiri atas beberapa bagian utama, yaitu nama *use case*, kode kebutuhan, aktor yang berinteraksi, tujuan, *pre-condition*, *main flow*, *alternative flow,* dan *post-condition*. *Pre-condition* merupakan kondisi awal yang harus dipenuhi sebelum aktor melakukan *main flow use case*, *main flow* merupakan alur pengerjaan *use case*, *alternative*

flow merupakan alur alternatif dari pengerjaan *use case* ketika terdapat suatu kondisi yang terjadi diluar skenario yang telah dituliskan pada *main flow*, dan terakhir adalah *post-condition* yang merupakan kondisi akhir setelah aktor melakukan *main flow*. Skenario *use case* sistem ditujukan pada Tabel 5.3 sampai Tabel 5.7.

Tabel 5.3 Skenario *Use Case* Memindahkan Posisi Halaman / *Tab*

Nama Use Case	Memindahkan posisi halaman / *tab*
Kode Kebutuhan	SRS-HCB-1.0
Aktor	Pengguna
Tujuan	Memindahkan posisi *tab* sesuai yang diinginkan oleh aktor.
Pre-condition	Aktor berada pada salah satu *tab* yang ada di dalam sistem.
Main flow	Aktor melakukan pergerakan kepala *axial left – long* untuk berpindah ke *tab* sebelumnya atau *axial right – long* untuk untuk berpindah ke *tab* selanjutnya.
Post-condition	Sistem akan memindahkan posisi *tab* sesuai dengan arah pergerakan kepala yang dilakukan oleh aktor.
Alternative flow	1. Apabila aktor melakukan gerakan kepala *axial left – long* ketika berada pada *tab* pertama, maka sistem tidak akan melakukan aksi apapun. 2. Apabila aktor melakukan gerakan kepala *axial right – long* ketika berada pada *tab* keempat, maka sistem tidak akan melakukan aksi apapun.

Tabel 5.4 Skenario *Use Case* Memindahkan Posisi Kursor

Nama Use Case	Memindahkan posisi kursor
Kode Kebutuhan	SRS-HCB-2.0
Aktor	Pengguna
Tujuan	Memindahkan posisi kursor sesuai dengan arah gerakan kepala yang dilakukan oleh aktor.
Pre-condition	Aktor berada pada *tab* pertama, kedua, atau ketiga.

Main flow	Aktor melakukan pergerakan kepala *axial left – short* untuk memindahkan kursor ke arah kiri atau *axial right – short* untuk memindahkan kursor ke arah kanan atau *flexion – short* untuk memindahkan kursor ke arah bawah atau *extension – short* untuk memindahkan kursor ke arah atas.
Post-condition	Sistem akan memindahkan posisi kursor sesuai dengan arah gerakan kepala yang dilakukan oleh aktor.
Alternative flow	1. Apabila aktor melakukan gerakan kepala *axial left – short* ketika kursor berada pada posisi ikon komunikasi yang pertama, maka sistem akan memindahkan posisi kursor ke posisi ikon komunikasi yang terakhir di dalam *tab* tersebut. 2. Apabila aktor melakukan gerakan kepala *axial right – short* ketika kursor berada pada posisi ikon komunikasi yang terakhir, maka sistem akan memindahkan posisi kursor ke posisi ikon komunikasi yang pertama di dalam *tab* tersebut.

Tabel 5.5 Skenario *Use Case* Memilih Aksi

Nama Use Case	Memilih aksi
Kode Kebutuhan	SRS-HCB-3.0
Aktor	Pengguna
Tujuan	Memutar suara terhadap aksi yang dipilih oleh aktor melalui *speaker* perangkat.
Pre-condition	Pengguna berada pada *tab* pertama, kedua, atau ketiga.
Main flow	Aktor melakukan pergerakan kepala *flexion – long* pada ikon komunikasi yang ingin dipilih.
Post-condition	Sistem memutar suara terhadap aksi yang dipilih oleh melalui *speaker* perangkat.
Alternative flow	-

Tabel 5.6 Skenario *Use Case* Menyusun Kata

Nama Use Case	Menyusun kata
Kode Kebutuhan	SRS-HCB-4.0
Aktor	Pengguna
Tujuan	Menyusun suatu kata yang terdiri atas satu / lebih karakter.
Pre-condition	Aktor berada pada *tab keempat.*
Main flow	Aktor melakukan pergerakan kepala *flexion – long* pada karakter yang diinginkan.
Post-condition	Karakter yang dipilih akan disimpan dan ditampikan pada label yang berada di dalam *tab* keempat.
Alternative flow	-

Tabel 5.7 Skenario *Use Case* Hapus Karakter

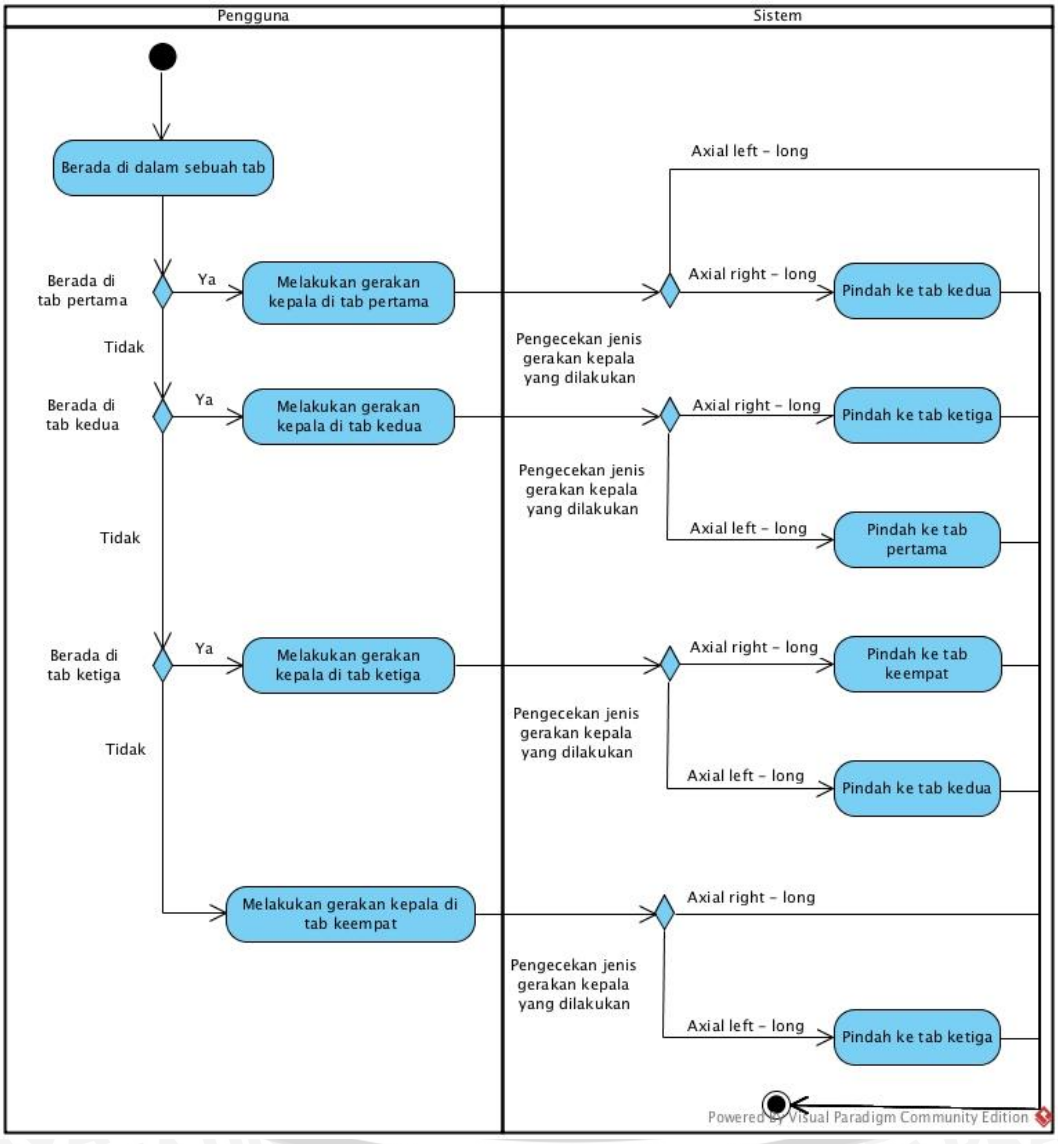
Nama Use Case	Menghapus karakter
Kode Kebutuhan	SRS-HCB-5.0
Aktor	Pengguna
Tujuan	Menghapus karakter terakhir yang dimasukkan oleh aktor.
Pre-condition	Aktor berada pada *tab* keempat dan telah memasukkan sebuah karakter ke dalam sistem.
Main flow	Aktor melakukan pergerakan kepala *extension – long* pada sembarang karakter.
Post-condition	Karakter terpilih terakhir akan dihapus dari label yang berada di dalam *tab* keempat.
Alternative flow	Apabila aktor melakukan pergerakan kepala *extension – long* saat label masih kosong, maka sistem tidak akan melakukan aksi apapun.

Tabel 5.8 Skenario *Use Case* Memutar Suara Kata

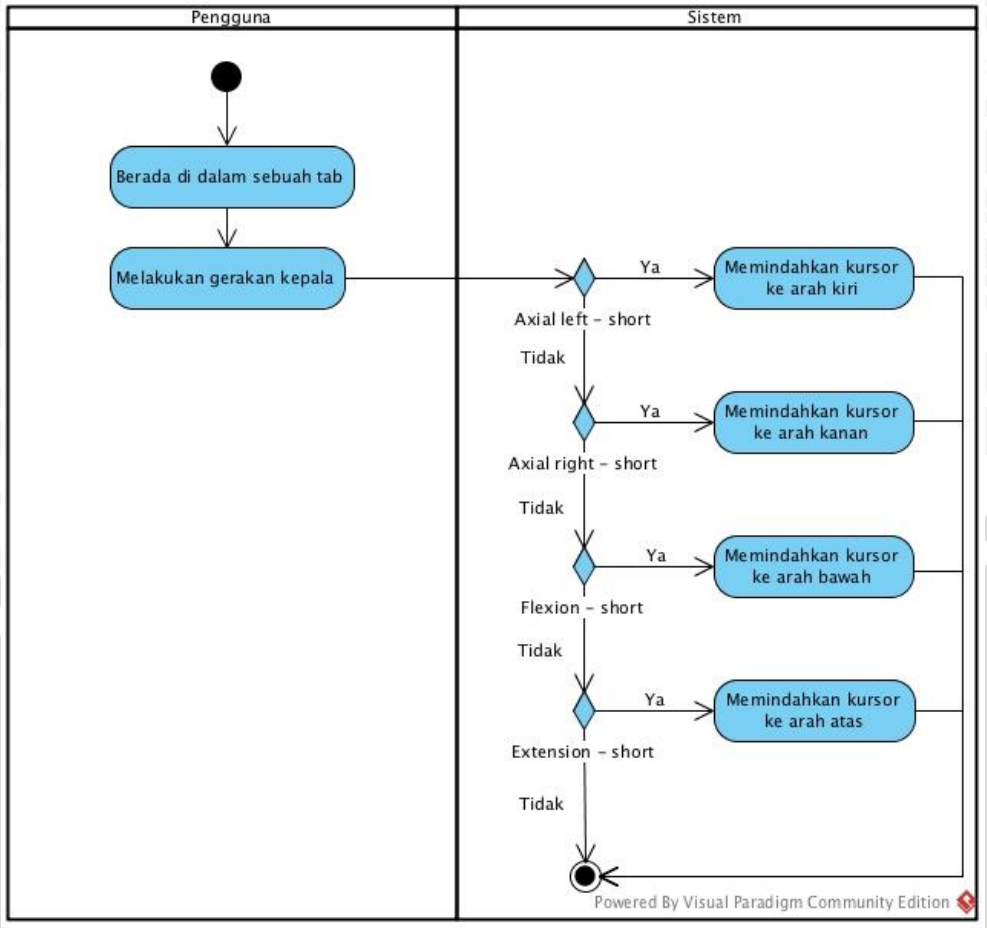
Nama Use Case	Memutar suara kata
Kode Kebutuhan	SRS-HCB-6.0
Aktor	Pengguna
Tujuan	Memutar suara terhadap kata yang telah disusun oleh aktor melalui *speaker* perangkat.
Pre-condition	Aktor berada pada *tab keempat* dan telah memasukkan sebuah karakter ke dalam sistem.
Main flow	Aktor melakukan pergerakan kepala *flexion – long* pada ikon bertuliskan OK.
Post-condition	Sistem akan memutar suara dari kata yang telah disusun oleh aktor melalui *speaker* perangkat.
Alternative flow	Apabila aktor melakukan pergerakan kepala *flexion – long* pada ikon betuliskan OK saat label masih kosong, maka sistem tidak akan memutar suara apapun.

5.3.3 Diagram Aktivitas

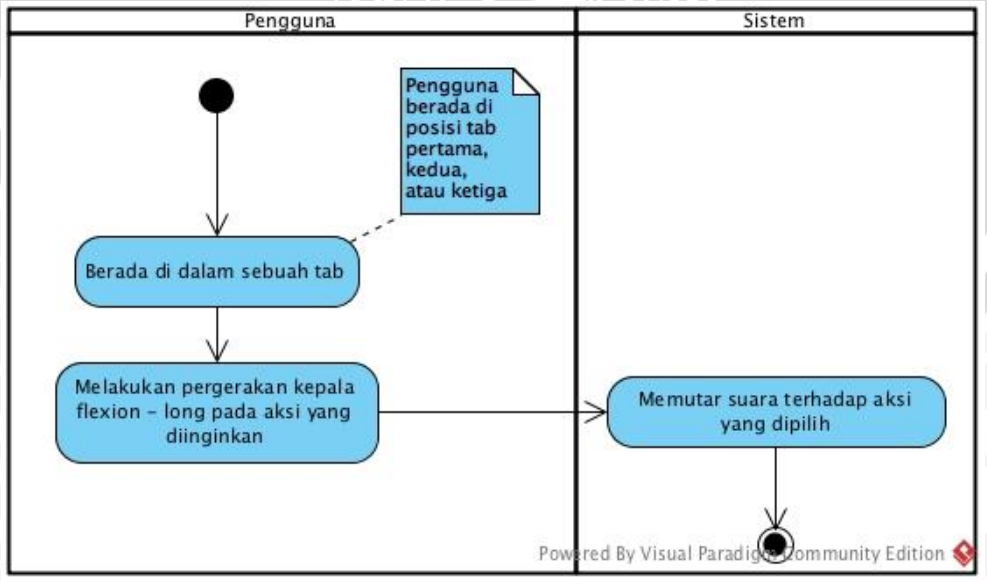
Diagram aktivitas merupakan diagram yang menggambarkan aspek dinamis dari sebuah sistem, dimana diagram ini merepresentasikan aliran aktivitas yang terjadi antar antara aktor sistem dengan sistem yang dikembangkan. Masing – masing diagram aktivitas akan mewakili sebuah *use case* sistem yang telah dijelaskan sebelumnya. Adapun diagram aktivtas sistem ditujukan pada Gambar 5.8 sampai dengan Gambar 5.12.



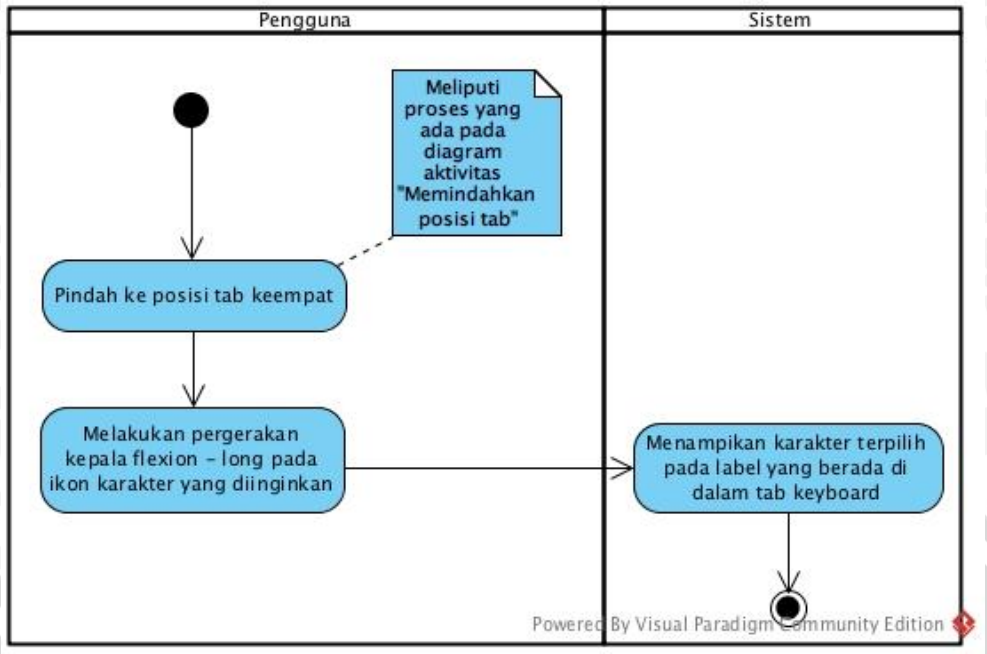
Gambar 5.8 Diagram Aktivitas Memindahkan Posisi Halaman / *Tab*



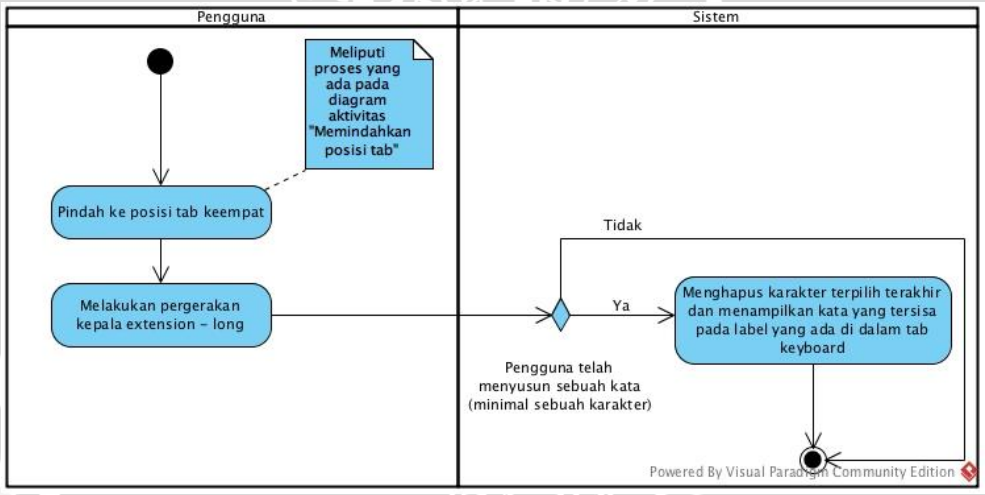
Gambar 5.9 Diagram Aktivitas Memindahkan Posisi Kursor



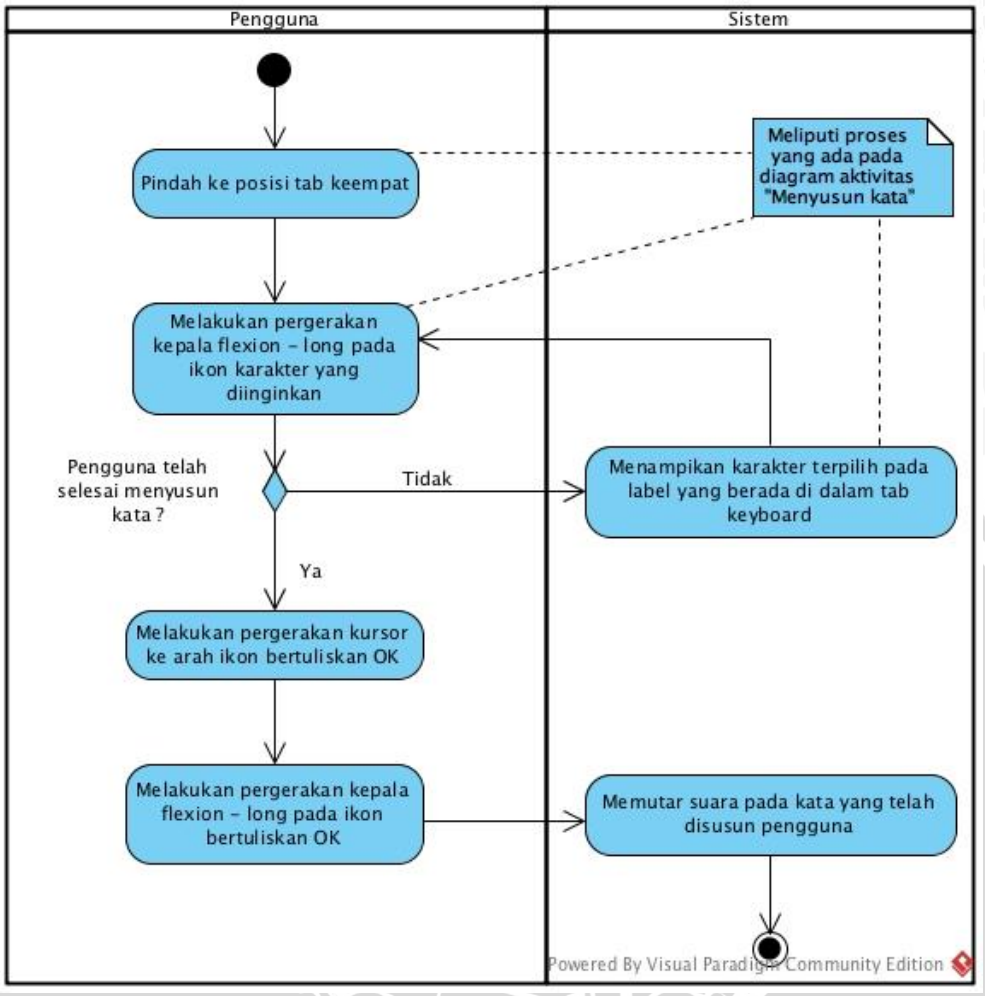
Gambar 5.10 Diagram Aktivitas Memilih Aksi



Gambar 5.11 Diagram Aktivitas Menyusun Kata



Gambar 5.12 Diagram Aktivitas Menghapus Karakter



Gambar 5.13 Diagram Aktivitas Memutar Suara Kata

BAB 6 IMPLEMENTASI

6.1 Batasan Implementasi

Pada penelitian ini, terdapat sejumlah batasan terkait dengan proses implementasi sistem yang akan dilakukan. Batasan tersebut antara lain sebagai berikut:

1. Aplikasi akan dikembangkan pada *smartphone* iPhone yang berada pada versi sistem operasi iOS 9.2.1.
2. Aplikasi akan dikembangkan dalam satu jenis Bahasa, yaitu Bahasa Inggris.
3. Aplikasi akan dikembangkan pada orientasi perangkat berjenis *landscape right*.
4. Seluruh proses pengembangan aplikasi dilakukan di dalam *Integtated Development Environment* (IDE) Xcode dengan menggunakan bahasa pemrograman *Objective*-C serta *framework core motion* buatan Apple.

6.2 Spesifikasi Perangkat Keras

Implementasi aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS dilakukan pada perangkat buatan Apple. Perangkat yang digunakan dalam proses implementasi serta pengujian aplikasi adalah komputer Macbook Pro, sedangkan perangkat tujuan implementasi adalah *smartphone* iPhone. Adapun spesifikasi perangkat keras dari komputer Macbook Pro serta *smartphone* iPhone yang digunakan dalam penelitian ini ditujukan pada Tabel 6.1 dan Tabel 6.2.

Tabel 6.1 Spesifikasi Perangkat Keras Komputer Macbook Pro

Nama Komponen	Spesifikasi Perangkat Keras
Model	Macbook Pro (15-inch, Mid 2012)
Prosesor	2,6 GHz Intel Core i7
RAM	16 GB 1600 MHz DDR3
GPU	Intel HD Graphics 4000 + NVIDIA GeForce GT 650M

Tabel 6.2 Spesifikasi Perangkat Keras *Smartphone* iPhone

Nama Komponen	Spesifikasi Perangkat Keras
Model	iPhone 5C
Prosesor	Dual Core Apple A6 1.0 GHz
RAM	1 GB
Kapasitas Penyimpanan	16 GB (tersedia 12 GB)

6.3 Spesifikasi Perangkat Lunak

Implementasi aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS dilakukan pada *platform* perangkat lunak macOS dan iOS milik Apple. Adapun spesifikasi perangkat lunak yang terpasang pada komputer Macbook Pro serta *smartphone* iPhone yang digunakan dalam penelitian ini ditujukan pada Tabel 6.3 dan Tabel 6.4.

Tabel 6.3 Spesifikasi Perangkat Lunak Komputer Macbook Pro

Nama Komponen	Spesifikasi Perangkat Lunak
Sistem Operasi	macOS Sierra 10.12.4
IDE	Xcode versi 8.3
Bahasa Pemrograman	*Objective*-C

Tabel 6.4 Spesifikasi Perangkat Lunak *Smartphone* iPhone

Nama Komponen	Spesifikasi Perangkat Lunak
Sistem Operasi	iOS 9.2.1

6.4 Implementasi Kode Program

Implementasi kode program terkait dengan kebutuhan fungsional sistem akan dilakukan di dalam empat buah *class controller*. Tiap *class controller* yang ada di dalam sistem merupakan turunan dari *class* UIViewController. Tiap *class controller* juga akan berasosiasi dengan sebuah *view* yang ada di dalam Main.storyboard. Setiap *class* akan tersusun atas dua buah *file,* yaitu *file header*

yang akan berasosiasi dengan *file implementation*. Adapun detil dari empat buah *class controller* tersebut ditujukan pada Tabel 6.5 berikut:

Tabel 6.5 *Class Controller* Sistem

Posisi *Tab*	Nama *File Header* (.h)	Nama *File Implementation* (.m)	Nama *View* yang Terkait
1	FirstViewController.h	FirstViewController.m	FirstViewController
2	SecondViewController.h	SecondViewController.m	SecondViewController
3	ThirdViewController.h	ThirdViewController.m	ThirdViewController
4	KeyboardViewController.h	KeyboardViewController.m	KeyboardViewController

6.4.1 Variabel Sistem

Konvensi variabel yang ada di dalam bahasa pemrograman *Objective-C* sistem terdiri atas *declared property* dan variabel primitif. *Declared property* sejatinya merupakan sebuah *property* atau atribut yang dimiliki oleh sebuah *class* yang juga menyediakan penyederhanaan sintaks dalam mendeklarasikan sebuah *accessor method*, sehingga ketika seorang *programmer* mendeklarasikan sebuah *declared property,* maka sistem secara otomatis akan menghasilkan sebuah fungsi *set* dan *get* terhadap *property* tersebut. @IBOutlet merupakan salah satu contoh dari *declared property* yang digunakan untuk menghubungkan sebuah elemen yang ada di dalam sebuah *view* dengan *controller* yang berkaitan, sedangkan variabel primitif merupakan variabel yang menyimpan sebuah nilai dengan tipe data primitif tertentu. Adapun variabel yang ada di dalam sistem beserta deskripsi singkatnya ditujukan pada Tabel 6.6 sebagai berikut:

Tabel 6.6 Variabel Sistem

NO	Nama Variabel	Tipe Data	Deskripsi
1.	cameraBackground	UIView (@IBOutlet)	*Declared property* yang menghubungkan tampilan kamera yang ada di dalam *view* dengan *controller.*

2.	actionLabel	UILabel (@IBOutlet)	*Declared property* yang menghubungkan label yang menampilkan aksi pengguna yang ada di dalam *view* dengan *controller.*
3.	deviceMotion ReadingTimer	NSTimer (@property)	*Declared property* yang digunakan untuk melakukan perulangan pembacaan pergerakan kepala pengguna.
4.	selectedState CursorBackground ImageNameArray	NSArray (@property)	*Declared property* yang berisi nama gambar *background* kursor dengan kondisi terpilih.
5.	unselectedState CursorBackground ImageNameArray	NSArray (@property)	*Declared property* yang berisi nama gambar *background* kursor dengan kondisi tidak terpilih.
6.	deviceMotionManager	CMMotionManager (@property)	*Declared property* yang digunakan untuk mengakses data pergerakan perangkat.
7.	cameraCaptureSession	AVCaptureSession (@property)	*Declared property* yang digunakan untuk menginisialisasi tampilan kamera.
8.	cameraPreviewLayer	AVCaptureVideo PreviewLayer (@property)	*Declared property* yang digunakan sebagai *layer* kamera yang akan ditambahkan ke dalam *view*.
9.	synthesizer	AVSpeech Synthesizer (@property)	*Declared property* yang digunakan untuk memutar suara yang merupakan *output* sistem.

10.	rawYawData	double	Variabel primitif yang menyimpan data pergerakan perangkat yang terjadi pada sumbu *yaw.*
11.	rawRollData	double	Variabel primitif yang menyimpan data pergerakan perangkat yang terjadi pada sumbu *roll*.
12.	yawNormal	double	Variabel primitif yang menyimpan nilai *default* dari sumbu *yaw.*
13.	rollNormal	double	Variabel primitif yang menyimpan nilai *default* dari sumbu *roll.*
14.	processedYawData	double	Variabel primitif yang menyimpan data sumbu *yaw* yang telah diproses.
15.	processedRollData	double	Variabel primitif yang menyimpan data sumbu *roll* yang telah diproses.
16.	yawCounter	double	Variabel primitif yang digunakan sebagai *counter* yang akan bertambah nilainya ketika pengguna melakukan pergerakan kepala terhadap sumbu *yaw.*
17.	rollCounter	double	Variabel primitif yang digunakan sebagai *counter* yang akan bertambah nilainya ketika pengguna melakukan pergerakan kepala terhadap sumbu *roll.*
18.	currentCursorTag	double	Variabel primitif yang digunakan untuk

			menyimpan nilai yang merepresentasikan posisi kursor yang sedang dipilih oleh pengguna saat itu.
19.	oldCursorTag	double	Variabel primitif yang digunakan untuk menyimpan nilai yang merepresentasikan posisi kursor yang lama.
20.	newCursorTag	double	Variabel primitif yang digunakan untuk menyimpan nilai yang merepresentasikan posisi kursor yang baru.
21.	actionCount	double	Variabel primitif yang digunakan untuk menyimpan nominal angka ikon komunikasi yang ada di dalam sebuah *view.*
22.	actionPerRowCount	double	Variabel primitif yang digunakan untuk menyimpan nominal angka ikon komunikasi per baris yang ada di dalam sebuah *view.*
23.	p0Yaw	double	Variabel primitif yang digunakan sebagai nilai batas bawah dari pergerakan perangkat yang terjadi pada sumbu *yaw.*
24.	p1Yaw	double	Variabel primitif yang digunakan sebagai nilai batas tengah dari pergerakan perangkat yang terjadi pada sumbu *yaw.*
25.	p2Yaw	double	Variabel primitif yang

			digunakan sebagai nilai batas atas dari pergerakan perangkat yang terjadi pada sumbu *yaw.*
26.	yawThreshold	double	Variabel primitif yang digunakan sebagai nilai batas inisialisasi pembacaan pergerakan perangkat pada sumbu *yaw.*
27.	p0Roll	double	Variabel primitif yang digunakan sebagai nilai batas bawah dari pergerakan perangkat yang terjadi pada sumbu *roll.*
28.	p1Roll	double	Variabel primitif yang digunakan sebagai nilai batas tengah dari pergerakan perangkat yang terjadi pada sumbu *roll.*
29.	p2Roll	double	Variabel primitif yang digunakan sebagai nilai batas atas dari pergerakan perangkat yang terjadi pada sumbu *roll.*
30.	rollThreshold	double	Variabel primitif yang digunakan sebagai nilai batas inisialisasi pembacaan pergerakan perangkat pada sumbu *roll.*
31.	movementCounter	double	Variabel primitif yang digunakan sebagai *counter* yang nilainya akan bertambah selama pengguna melakukan pergerakan kepala tertentu terhadap

			perangkat.
32.	movementCounter Threshold	double	Variabel primitif yang digunakan sebagai batas dari variabel movementCounter.

6.4.2 Implementasi Fungsional Sistem

Fungsionalitas sistem yang terjadi di dalam sebuah *view* akan diimplementasikan ke dalam sebuah *class* yang terbagi menjadi dua buah *file*, yaitu *file header* dan *implementation*. Secara umum, *file header* akan berisi struktur data enum serta *statement import* terhadap *framework* yang akan digunakan di dalam bagian *implementation*, sedangkan *file implementation* berisi kode aktual implementasi sistem yang terdiri atas sejumlah variabel dan *declared property* yang dideklarasikan secara *private*, fungsi terkait dengan *life cycle* dari *view,* dan sejumlah fungsi yang digunakan untuk menjalankan fungsionalitas dari sistem. *Class* yang akan dijadikan sebagai contoh pembahasan di dalam sub bab ini adalah *class* FirstViewController, dimana *class* tersebut bertanggung jawab dalam memproses seluruh *input* dan *output* sistem yang berada di dalam *tab pertama*.

a. File Header

File header berisi dua buah struktur data enum yang akan digunakan di dalam *file implementation*. Enum digunakan untuk membantu utilisasi pengecekan kode program saat *compile-time*, meminimalisir adanya *error* ketika terjadi *passing* konstanta tertentu, dan mendefinisikan sejumlah konstanta yang dapat digunakan oleh *programmer* di dalam bagian *implementation*. Enum yang dimaksud terdiri atas enum Direction dan StatusCode. Implementasi kode program *file header* ditujukan pada Kode Program 6.1.

Kode Program 6.1 Implementasi *File Header*

NO	Kode Program
1	`#import <UIKit/UIKit.h>`
2	
3	`@interface FirstViewController : UIViewController <UIAccelerometerDelegate> {`
4	
5	`}`
6	
7	`typedef NS_ENUM(NSInteger, Direction)`
8	`{`
9	`    UP,`
10	`    DOWN,`
11	`    LEFT,`
12	`    RIGHT,`
13	`};`
14	

```
15  typedef NS_ENUM(NSInteger, StatusCode)
16  {
17      NONE = 1,
18      H1A,
19      H2A,
20      H3A,
21      H4A,
22      H1B,
23      H2B,
24      H3B,
25      H4B
26  };
27
28  @end
```

Penjelasan dari isi Kode Program 6.1 adalah sebagai berikut:

1. Baris 1 : *Statement import* yang digunakan untuk menyertakan seluruh *class* yang terdapat pada *framework* UIKit.
2. Baris 3 – 5 : Deklarasi *class* FirstViewController yang diturunkan dari *class* UIViewController dan mengadopsi sebuah protocol / *interface* yang bernama UIAccelerometerDelegate.
3. Baris 7 – 13 : Deklarasi struktur data enum dengan tipe Direction yang berisi enam buah arah pergerakan kepala yang dikenali oleh sistem yang terdiri atas *up*, *down*, *left*, dan *right*.
4. Baris 15 – 26 : Deklarasi struktur data enum dengan tipe StatusCode yang berisi sembilan buah kode status pergerakan linear yang akan digunakan di dalam sistem yang terdiri atas kode status NONE, H1A, H2A, H3A, H4A, H1B, H2B, H3B, dan H4B.

b. File Implementation

File implementation berisi lima buat fungsi terkait *life cycle* dari *view* dan lima buah fungsi yang digunakan dalam menjalankan fungsionalitas sistem ke depan. Fungsi *life cycle* dari *view* terdiri atas fungsi viewDidLoad, viewWillAppear, viewDidAppear, viewWillDisappear, dan viewDidDisappear, sedangkan fungsi yang terkait dengan fungsionalitas sistem terdiri atas fungsi readUserHeadMovement, moveCursor, performUserAction, resetLabelFontColor, dan resetAllCounter. Khusus pada *tab* keempat, fungsi performUserAction yang digunakan pada *tab* pertama, kedua, dan ketiga akan disubtitusikan oleh sebuah fungsi yang bernama performKeyboardAction. Fungsi tersebut bertugas untuk memutar suara terhadap kata yang telah disusun oleh pengguna. Selain itu, terdapat juga sebuah fungsi bernama deleteCharacter yang digunakan untuk menghapus karakter terakhir yang telah dimasukkan pengguna di dalam *tab* keempat. Berikut ini akan dijelaskan lebih detil mengenai tiap fungsi yang ada di dalam *file implementation*:

i. **Fungsi viewDidLoad**

Fungsi viewDidLoad digunakan untuk menjalankan sejumlah proses terkait dengan inisialiasi awal *tab* pertama, antara lain:

1. Melakukan inisialisasi array selectedStateCursorBackgroundImageNameArray dan unselectedStateCursorBackgroundImageNameArray.
2. Menyiapkan tampilan kamera pada layar perangkat.
3. Melakukan inisialiasi objek synthesizer yang digunakan untuk memutar suara terhadap aksi yang dipilih oleh pengguna yang dilakukan di dalam fungsi performUserAction.
4. Memulai proses pembacaan pergerakan perangkat dari sensor gerak.

ii. **Fungsi viewWillAppear**

Fungsi viewWillAppear digunakan untuk menjalankan sejumlah proses sesaat sebelum *tab* pertama dimunculkan di dalam layar, antara lain:

1. Menyembunyikan *status bar* yang terlihat pada layar perangkat.
2. Menentukan posisi kursor pengguna.
3. Menetapkan isi label actionLabel (label aksi pengguna) sesuai dengan aksi yang ditunjuk oleh kursor yang dipilih pengguna.
4. Menjalankan fungsi kamera pada tampilan yang telah disiapkan pada fungsi viewDidLoad sebelumnya.

iii. **Fungsi viewDidAppear**

Fungsi viewDidAppear digunakan untuk memanggil fungsi readUserHeadMovement setelah *tab* pertama dimunculkan di dalam layar. Pemanggilan fungsi tersebut akan diulangi setiap 0,1 detik, yang berarti bahwa sistem akan melakukan perulangan dalam mengambil serta memproses data pergerakan perangkat sebanyak sebanyak satu kali tiap 0,1 detik. Durasi perulangan tersebut berasal dari penelitian sebelumnya yang berjudul "Perancangan Kontrol Keyboard Menggunakan Head Movement System Dengan Pergerakkan Linier Pada Perangkat Bergerak Berbasis IOS", dimana kode program yang digunakan untuk merancang kontrol *keyboard* juga menggunakan durasi pemanggilan fungsi yang diulangi setiap 0,1 detik.

iv. **Fungsi viewWillDisappear**

Fungsi viewWillDisappear digunakan untuk menghentikan fungsi kamera sementara hingga pemanggilan fungsi viewWillAppear dilakukan kembali.

v. Fungsi viewDidDisappear

Fungsi viewDidDisappear digunakan untuk menghentikan fungsi readUserHeadMovement sementara hingga pemanggilan fungsi viewDidAppear dilakukan kembali.

vi. Fungsi readUserHeadMovement

Fungsi readUserHeadMovement digunakan untuk memproses pergerakan kepala yang dilakukan oleh pengguna terhadap perangkat menjadi sebuah aksi yang akan dijalankan oleh sistem. Isi kode program fungsi readUserHeadMovement ditujukan pada Kode Program 6.2.

Kode Program 6.2 Pembacaan Pergerakan Kepala

NO	Kode Program
1	`CMDeviceMotion *motionData = _deviceMotionManager.deviceMotion;`
2	`CMAttitude *attitude = motionData.attitude;`
3	
4	`rawYawData = attitude.yaw * 180 / M_PI;`
5	`rawRollData = (attitude.roll * 180 / M_PI) - 90;`
6	
7	`processedYawData = fabs(rawYawData - yawNormal);`
8	`processedRollData = fabs(rawRollData - rollNormal);`
9	
10	`static Direction headMovementDirection;`
11	`static StatusCode headMovementStatusCode;`
12	
13	`if (processedYawData >= yawThreshold) {`
14	`    yawCounter++;`
15	
16	`    if (rawYawData - yawNormal > 0) {`
17	`        headMovementDirection = LEFT;`
18	`    } else {`
19	`        headMovementDirection = RIGHT;`
20	`    }`
21	`} else if (processedRollData >= rollThreshold) {`
22	`    rollCounter++;`
23	
24	`    if (rawRollData - rollNormal > 0) {`
25	`        headMovementDirection = DOWN;`
26	`    } else {`
27	`        headMovementDirection = UP;`
28	`    }`
29	`}`
30	`else {`
31	`    if (yawCounter >= p0Yaw && yawCounter <= p2Yaw) {`
32	`        if (yawCounter >= p0Yaw && yawCounter <= p1Yaw) {`
33	`            if (headMovementDirection == LEFT) {`
34	`                headMovementStatusCode = H1A;`
35	`            } else if (headMovementDirection == RIGHT) {`
36	`                headMovementStatusCode = H2A;`
37	`            }`
38	`        } else if (yawCounter > p1Yaw && yawCounter <= p2Yaw) {`
39	`            if (headMovementDirection == RIGHT) {`
40	`                headMovementStatusCode = H2B;`
41	`            }`
42	`        }`
43	`    } else if (rollCounter >= p0Roll && rollCounter <= p2Roll) {`
44	`        if (rollCounter >= p0Roll && rollCounter <= p1Roll) {`
45	`            if (headMovementDirection == DOWN) {`
46	`                headMovementStatusCode = H3A;`
47	`            } else if (headMovementDirection == UP) {`

```
48                      headMovementStatusCode = H4A;
49                  }
50              } else if (rollCounter > p1Roll && rollCounter <= p2Roll) {
51                  if (headMovementDirection == DOWN) {
52                      headMovementStatusCode = H3B;
53                  }
54              }
55          } else {
56              headMovementStatusCode = NONE;
57          }
58
59          UIViewController *secondViewController = [[UIViewController alloc]
60  init];
61
62          switch (headMovementStatusCode) {
63              case H1A:
64                  currentCursorTag--;
65                  if (currentCursorTag < 1) currentCursorTag = actionCount;
66                  break;
67              case H2A:
68                  currentCursorTag++;
69                  if (currentCursorTag > actionCount) currentCursorTag = 1;
70                  break;
71              case H3A:
72                  currentCursorTag += actionPerRowCount;
73                  if (currentCursorTag > actionCount) currentCursorTag -=
74  actionCount;
75                  break;
76              case H4A:
77                  currentCursorTag -= actionPerRowCount;
78                  if (currentCursorTag < 1) currentCursorTag += actionCount;
79                  break;
80              case H2B:
81                  secondViewController = [self.storyboard
82  instantiateViewControllerWithIdentifier:@"SecondViewController"];
83                  [self.navigationController
84  pushViewController:secondViewController animated:true];
85                  break;
86              case H3B:
87                  [self performUserAction: currentCursorTag];
88                  break;
89              case NONE:
90                  movementCounter++;
91                  break;
92              default:
93                  break;
94          }
95
96          [self resetAllCounter];
97          newCursorTag = currentCursorTag;
98
99          if (newCursorTag != oldCursorTag) {
100             [self moveCursor: currentCursorTag];
101             [self resetAllCounter];
102         }
103     }
104
105     if (movementCounter >= movementCounterThreshold || yawCounter >=
106 movementCounterThreshold || rollCounter >= movementCounterThreshold) {
107         yawNormal = rawYawData;
108         rollNormal = rawRollData;
109         [self resetAllCounter];
110     }
```

Penjelasan dari isi Kode Program 6.2 adalah sebagai berikut:

1. Baris 1 : Instansiasi objek yang berasal dari *class* CMDeviceMotion bernama motionData yang digunakan untuk menyimpan data pergerakan

perangkat yang didapatkan dari pengaksesan atribut deviceMotion yang terdapat di dalam objek deviceMotionManager sensor gerak.

2. Baris 2 : Inisialisasi sebuah objek yang berasal dari *class* CMAttitude bernama attitude yang digunakan untuk menyimpan data tingkah laku pergerakan perangkat dengan mengakses atribut attitude yang terdapat di dalam objek motionData yang telah diinisialisasi sebelumnya.
3. Baris 4 : Konversi data pergerakan perangkat yang terjadi pada sumbu *yaw* dari bentuk skala *radian* menjadi bentuk skala *degree* lalu menyimpannya ke dalam variabel rawYawData.
4. Baris 5 : Konversi data pergerakan perangkat yang terjadi pada sumbu *roll* dari bentuk skala *radian* menjadi bentuk skala *degree* lalu menyimpannya ke dalam variabel rawRollData.
5. Baris 7 : Pemrosesan data pergerakan perangkat pada sumbu *yaw* yang dilakukan dengan mengurangi variabel rawYawData yang telah didapatkan sebelumnya dengan variabel yawNormal. Nilai yang didapatkan kemudian diubah menjadi nilai absolut dengan menggunakan fungsi fabs yang ada di dalam *framework* foundation. Nilai tersebut kemudian disimpan di dalam variabel processedYawData.
6. Baris 8 : Pemrosesan data pergerakan perangkat pada sumbu *roll* yang dilakukan dengan mengurangi variabel rawRollData yang telah didapatkan sebelumnya dengan variabel rollNormal. Nilai yang didapatkan kemudian diubah menjadi nilai absolut dengan menggunakan fungsi fabs yang ada di dalam *framework* foundation. Nilai tersebut kemudian disimpan di dalam variabel processedRollData.
7. Baris 10 : Instansiasi sebuah objek enum Direction bernama headMovementDirection yang akan digunakan untuk menampung arah pergerakan kepala yang dilakukan pengguna terhadap perangkat.
8. Baris 11 : Instansiasi sebuah objek enum StatusCode bernama headMovementStatusCode yang akan digunakan untuk menampung kode status pergerakan kepala yang dilakukan pengguna.
9. Baris 13 – 29 : Pengecekan kondisi, apabila variabel pergerakan perangkat yang telah diproses (processedYawData dan processedRollData) telah melebihi atau sama dengan salah satu dari variabel batas (yawThresold dan rollThresold), maka proses penentuan arah pergerakan

kepala akan dilakukan oleh sistem. Apabila kondisi yang memenuhi terjadi pada kondisi sumbu *yaw*, maka penentuan arah pergerakan kepala ke kanan atau kiri akan dilakukan, sedangkan apabila kondisi yang memenuhi terjadi pada sumbu *roll*, maka maka penentuan arah pergerakan kepala ke atas atau bawah dilakukan. Selama pengguna melakukan pergerakan kepala ke arah yang ia kehendaki, sebuah variabel *counter* (yawCounter atau rollCounter) akan bertambah nilainya sesuai dengan sumbu pergerakan yang memenuhi kondisi diatas.

10. Baris 30 – 103 : Pengecekan kondisi, apabila salah satu variabel *counter* yang didapatkan (yawCounter atau rollCounter) berada diantara dua dari tiga buah nilai *threshold* (p0, p1, atau p2), maka proses pemetaan kode status pergerakan linear oleh sistem akan dilakukan. Kode status tersebut disimpan ke dalam objek enum headMovementStatusCode yang akan digunakan sebagai dasar dalam menentukan aksi sistem yang akan dilakukan. Pemberian kode status pergerakan linear dipetakan sesuai dengan arah dan durasi pergerakan kepala yang dilakukan oleh pengguna. Durasi pergerakan kepala pendek akan dipetakan apabila variabel *counter* berada diantara nilai variabel p0 dan p1, sedangkan durasi pergerakan kepala panjang akan dipetakan apabila variabel *counter* berada diantara nilai variabel p1 dan p2. Apabila sistem tidak mendeteksi pergerakan kepala apapun, maka sistem akan menetapkan isi objek enum headMovementStatusCode dengan "NONE" yang berarti tidak ada pergerakan apapun.
11. Baris 58 – 60 : Instansiasi sebuah objek *view* yang berasal dari *class* UIViewController bernama secondViewController. *View* ini nantinya merepresentasikan *view* dari *tab* kedua yang akan digunakan oleh sistem dalam memindahkan *tab* pengguna dari *tab* pertama ke *tab* kedua.
12. Baris 62 – 94 : Sistem akan mengecek kode status pergerakan linear yang telah dipetakan sebelumnya di dalam objek enum headMovementStatusCode kemudian menjalankan aksi sistem sesuai dengan kode tersebut. Adapun kode status pergerakan linear beserta aksi yang dijalankan oleh sistem ditujukan pada Tabel 5.2.
13. Baris 96 : Setelah sistem menjalankan sebuah aksi, sistem akan memanggil sebuah fungsi bernama resetAllCounter.
14. Baris 97 : Menetapkan isi variabel newCursorTag dengan isi variabel currentCursorTag untuk mewakili posisi kursor yang sedang dipilih oleh pengguna saat itu.

15. Baris 99 – 102 : Sistem akan memanggil fungsi moveCursor dengan parameter currentCursorTag untuk memindahkan posisi kursor pengguna dari posisi yang lama ke posisi yang baru.
16. Baris 105 – 110: Pengecekan kondisi diakhir, apabila variabel totalCounter atau yawCounter atau rollCounter telah melebihi nilai dari variabel counterThreshold, maka sistem akan menetapkan nilai variabel yawNormal dengan variabel rawYawData dan rollNormal dengan variabel rawRollData untuk mengalibrasi ulang posisi kepala pengguna sebagai dasar bagi pergerakan kepala selanjutnya.

vii. Fungsi moveCursor

Fungsi moveCursor digunakan untuk memindahkan posisi kursor pengguna, dimana fungsi ini menerima sebuah parameter bertipe *integer* bernama currentCursorTag. Parameter tersebut digunakan oleh sistem untuk menemukan posisi kursor yang sedang dipilih oleh pengguna saat itu. Logika utama dari fungsi moveCursor ialah melakukan *swap* gambar ikon komunikasi yang sedang dipilih dengan gambar ikon komunikasi yang sebelumnya dipilih, serta menetapkan isi dari actionLabel dengan aksi pengguna yang terdapat pada ikon komunikasi yang sedang dipilih pengguna saat itu. Isi kode program fungsi moveCursor ditujukan pada Kode Program 6.3.

Kode Program 6.3 Memindahkan Posisi Kursor

NO	Kode Program
1	`    UIButton *currentCursor = (UIButton *)[self.view viewWithTag:`
2	`currentCursorTag];`
3	`    currentCursor.selected = !currentCursor.selected;`
4	
5	`    UIButton *oldCursor = (UIButton *)[self.view viewWithTag: oldCursorTag];`
6	`    oldCursor.selected = !oldCursor.selected;`
7	
8	`    NSString *cursorTitle = [currentCursor currentTitle];`
9	
10	`    _actionLabel.text = cursorTitle;`
11	
12	`    UIImage *currentCursorImage = [UIImage`
13	`imageNamed:_selectedStateCursorBackgroundImageNameArray[currentCursorTag -`
14	`1]];`
15	`    [currentCursor setImage:currentCursorImage`
16	`forState:UIControlStateNormal];`
17	
18	`    UIImage *oldCursorImage = [UIImage`
19	`imageNamed:_unselectedStateCursorBackgroundImageNameArray[oldCursorTag - 1]];`
20	`    [oldCursor setImage:oldCursorImage forState:UIControlStateNormal];`
21	
22	`    oldCursorTag = currentCursorTag;`

Penjelasan dari isi Kode Program 6.3 adalah sebagai berikut:

1. Baris 1 – 3 : Instansiasi sebuah objek tombol yang berasal dari *class* UIButton bernama currentCursor. Objek tersebut diinstansiasi berdasarkan variabel currentCursorTag yang mewakili posisi kursor yang sedang dipilih pengguna.
2. Baris 5 – 6 : Instansiasi sebuah objek tombol yang berasal dari *class* UIButton bernama oldButton. Objek tersebut diinstansiasi berdasarkan variabel oldCursorTag yang mewakili posisi kursor terakhir yang dipilih oleh pengguna.
3. Baris 8 : Instansiasi sebuah objek string bernama cursorTitle yang menampung aksi pengguna yang berasal dari objek currentCursor.
4. Baris 10 : Menetapkan isi label actionLabel yang dengan variabel cursorTitle.
5. Baris 12 – 14 : Instansiasi sebuah objek gambar yang berasal dari *class* UIImage bernama currentCursorImage. Objek tersebut akan di *assign* dengan sebuah gambar yang namanya terdapat pada array selectedStateCursorBackgroundImageNameArray. Indeks yang akan digunakan untuk mengakses nama gambar yang diinginkan berasal dari variabel currentCursorTag – 1 (dikarenakan indeks pada array dimulai dari nol). Gambar currentCursorImage akan ditetapkan pada tombol currentCursor sebagai representasi kursor yang sedang dipilih oleh pengguna.
6. Baris 15 – 16 : Menetapkan gambar pada tombol currentCursor dengan gambar currentCursorImage.
7. Baris 18 – 20 : Instansiasi sebuah objek gambar yang berasal dari *class* UIImage bernama oldCursorImage. Objek tersebut akan di *assign* dengan sebuah gambar yang namanya terdapat pada array yang bernama unselectedStateCursorBackgroundImageNameArray. Indeks yang akan digunakan untuk mengakses nama gambar yang diinginkan berasal dari variabel oldCursorTag – 1 (dikarenakan indeks pada array dimulai dari nol). Gambar oldButtonImage akan ditetapkan pada tombol oldCursor sebagai representasi kursor sebelumnya dipilih oleh pengguna.
8. Baris 20 : Menetapkan gambar pada tombol oldCursor dengan gambar oldCursorImage.
9. Baris 22 : Sistem menetapkan isi variabel oldCursorTag dengan variabel currentCursorTag agar pada pemanggilan fungsi yang selanjutnya,

variabel oldCursorTag dapat dibandingkan dengan variabel currentCursorTag yang baru.

viii. Fungsi performUserAction

Fungsi performUserAction digunakan untuk memutar suara aksi yang dipilih oleh pengguna, dimana fungsi ini menerima sebuah parameter bertipe *integer* bernama currentCursorTag. Parameter tersebut digunakan untuk menemukan posisi kursor yang dipilih oleh pengguna saat itu. Logika utama dari fungsi performUserAction ialah memutar suara terhadap aksi pengguna yang dipilih serta mengubah warna teks di dalam actionLabel menjadi hijau selama dua detik sebagai umpan balik kepada pengguna bahwa ia telah memilih sebuah aksi tertentu. Isi kode program fungsi performUserAction ditujukan pada Kode Program 6.4.

Kode Program 6.4 Memutar Suara Aksi yang Dipilih

NO	Kode Program
1	`_actionLabel.textColor = [UIColor greenColor];`
2	
3	`UIButton *currentCursor = (UIButton *)[self.view`
4	`viewWithTag:currentCursorTag];`
5	
6	`NSString *cursorTitle = [currentCursor currentTitle];`
7	
8	`AVSpeechUtterance *utterance = [AVSpeechUtterance`
9	`speechUtteranceWithString:cursorTitle];`
10	
11	`[_synthesizer speakUtterance:utterance];`
12	
13	`[self performSelector:@selector(resetLabelFontColor)`
14	`withObject:nil`
15	`afterDelay:2.0];`

Penjelasan dari isi Kode Program 6.4 adalah sebagai berikut:

1. Baris 1 : Mengubah warna teks yang ada di dalam label actionLabel dengan warna hijau. Label tersebut berisi aksi pengguna yang sedang dipilih.
2. Baris 3 – 4 : Instansiasi sebuah objek tombol yang berasal dari *class* UIButton bernama currentCursor. Objek tersebut diinstansiasi berdasarkan variabel currentCursorTag yang mewakili posisi kursor yang sedang dipilih pengguna.
3. Baris 6 : Instansiasi sebuah objek string bernama cursorTitle yang menampung aksi pengguna yang berasal dari objek currentCursor.

4. Baris 8 – 9 : Instansiasi sebuah objek pemutar suara yang berasal dari *class* AVSpeechUtterance bernama utterance yang bertugas untuk menyiapkan kata yang akan diputar suaranya oleh sistem, yang dalam kasus ini adalah cursorTitle.
5. Baris 11 : Melalui objek synthesizer yang telah diinstansiasi sebelumnya di dalam fungsi viewDidLoad, sistem kemudian memutar suara dengan memanggil fungsi speakUtterance dengan menyertakan objek utterance sebagai parameternya. Suara tersebut akan diputar melalui *speaker* yang ada pada perangkat.
6. Baris 13 – 15 : Setelah dua detik berlalu, fungsi performUserAction akan memanggil fungsi resetLabelFontColor.

ix. Fungsi resetLabelFontColor

Fungsi resetLabelFontColor bertugas untuk mengembalikan warna teks pada label actionLabel yang sebelumnya berwarna hijau dengan warna hitam kembali.

x. Fungsi resetAllCounter

Fungsi resetLabelFontColor berfungsi untuk menetapkan ulang isi variabel yawCounter, rollCounter, dan movementCounter menjadi nol agar sistem dapat kembali mendeteksi pergerakan perangkat yang selanjutnya.

xi. Fungsi performKeyboardAction

Fungsi performKeyboardAction digunakan untuk memutar suara terhadap kata yang telah disusun oleh pengguna di dalam *tab* keempat, dimana fungsi ini menerima sebuah parameter bertipe *integer* bernama currentCursorTag yang digunakan untuk menemukan posisi kursor yang sedang dipilih pengguna saat itu. Logika utama dari fungsi performKeyboardAction adalah menentukan aksi yang akan dilakukan sistem berdasarkan nilai currentCursorTag yang ada saat itu. Apabila nilai dari currentCursorTag bernilai 32 (nilai ini merupakan jumlah ikon karakter yang ada di dalam *tab* keempat), dapat dipastikan bahwa kursor pengguna sedang berada pada ikon bertuliskan OK. Apabila pengguna memilih ikon bertuliskan OK setelah pengguna menyusun sebuah kata, maka sistem akan memutar suara terhadap kata yang telah disusun oleh pengguna. Namun apabila nilai dari currentCursorTag bernilai lebih dari nol dan kurang dari 32, dapat dipastikan bahwa kursor pengguna sedang berada pada ikon karakter selain ikon

bertuliskan OK. Apabila pengguna memilih ikon tersebut maka sistem akan menambahkan karakter yang dipilih pada kata yang ada di dalam textLabel. Label tersebut berfungsi untuk menampilkan kata yang telah disusun oleh pengguna di dalam *tab* keempat. Isi kode program fungsi performKeyboardAction ditujukan pada Kode Program 6.5.

Kode Program 6.5 Memutar Suara Kata

NO	Kode Program
1	`if (currentCursorTag > 0 && currentCursorTag < characterButtonCount) {`
2	
3	`        NSString *currentText = _textLabel.text;`
4	
5	`        UIButton *currentCursor = (UIButton *)[self.view`
6	`viewWithTag:currentCursorTag];`
7	
8	`        NSString *character = [currentCursor currentTitle];`
9	
10	`        NSString *appendedText = [currentText`
11	`stringByAppendingString:character];`
12	
13	`        _textLabel.text = appendedText;`
14	`}`
15	`else if (currentCursorTag == characterButtonCount) {`
16	
17	`        NSString *currentText = _userTextLabel.text;`
18	
19	`        _textLabel.textColor = [UIColor greenColor];`
20	
21	`        AVSpeechUtterance *utterance = [AVSpeechUtterance`
22	`speechUtteranceWithString:currentText];`
23	
24	`        [synthesizer speakUtterance:utterance];`
25	
26	`        [self performSelector:@selector(resetLabelFontColor)`
27	`                      withObject:nil`
28	`                      afterDelay:2.0];`
29	`}`
30	
31	`oldCursorTag = currentCursorTag;`

Penjelasan dari isi Kode Program 6.5 adalah sebagai berikut:

1. Baris 1 – 15 : Jika nilai dari variabel currentCursorTag berada diantara 0 dan variabel characterButtonCount yang bernilai 32, maka posisi kursor pengguna sedang tidak berada pada ikon yang bertuliskan OK, yang berarti bahwa ketika pengguna memilih sebuah ikon karakter, maka sistem akan menambahkan karakter yang dipilih pada kata yang ada di dalam textLabel.
2. Baris 3 : Sistem mengambil teks yang ada di dalam textLabel dan menyimpannya ke dalam variabel dengan tipe data string bernama currentText.

3. Baris 5 – 6 : Inisialisasi sebuah objek tombol bernama currentCursor yang mewakili posisi kursor yang sedang dipilih pengguna saat itu.
4. Baris 8 : Sistem mengambil karakter yang ada di dalam objek currentCursor dan menyimpan karakter tersebut pada sebuah variabel dengan tipe data string bernama character.
5. Baris 10 – 11 : Sistem menambahkan variabel character pada variabel currentText lalu menyimpannya pada variabel bernama appendedText.
6. Baris 13 : Sistem menetapkan isi dari label textLabel dengan variabel appendedText.
7. Baris 15 – 29 : Jika nilai variabel currentCursorTag bernilai sama dengan variabel characterButtonCount yang bernilai 32, maka posisi kursor pengguna sedang berada pada ikon yang bertuliskan OK, yang berarti ketika pengguna memilih ikon tersebut, maka sistem akan memutar suara terhadap kata yang telah disusun sebelumnya oleh pengguna.
8. Baris 17 : Sistem telebih dahulu mengambil teks yang ada di dalam label textLabel dan menyimpannya ke dalam variabel dengan tipe data string bernama currentText.
9. Baris 19 : Mengubah warna teks pada label textLabel dengan warna hijau. Perubahan tersebut menandakan bahwa pengguna sedang melakukan aksi pemutaran suara terhadap kata yang telah ia susun.
10. Baris 21 – 22 : Instansiasi sebuah objek pemutar suara yang berasal dari *class* AVSpeechUtterance bernama utterance yang bertugas untuk menyiapkan kata yang akan diputar suaranya oleh sistem, yang dalam kasus ini adalah currentText.
11. Baris 24 : Melalui objek synthesizer yang telah diinstansiasi sebelumnya di dalam fungsi viewDidLoad, sistem kemudian memutar suara dengan memanggil fungsi speakUtterance dengan menyertakan objek utterance sebagai parameternya. Suara tersebut akan diputar melalui *speaker* yang ada pada perangkat.
12. Baris 26 – 28 : Setelah dua detik berlalu, fungsi performKeyboardAction akan memanggil fungsi resetLabelFontColor yang bertugas untuk mengembalikan warna teks pada objek userTextLabel yang sebelumnya berwarna hijau dengan warna hitam kembali.

13. Baris 31 : Sistem menetapkan isi variabel oldCursorTag dengan variabel currentCursorTag agar pada pemanggilan fungsi yang selanjutnya, variabel oldCursorTag dapat dibandingkan dengan variabel currentCursorTag yang baru.

xii. Fungsi deleteCharacter

Fungsi deleteCharacter digunakan untuk menghapus karakter terakhir yang telah dimasukkan oleh pengguna di dalam *tab* keempat. Logika utama dari fungsi deleteCharacter adalah mengambil teks yang berada pada label textLabel yang ada di dalam *tab* keempat lalu memotong panjang teks tersebut sebanyak satu karakter dari panjang total teks secara keseluruhan. Setelah teks tersebut berhasil dipotong, teks baru tersebut akan di *set* kembali pada textLabel. Isi kode program fungsi deleteCharacter ditujukan pada Kode Program 6.6.

Kode Program 6.6 Menghapus Karakter

NO	Kode Program
1	`if ([_textLabel.text length] > 0) {`
2	
3	`NSString *currentText = _textLabel.text;`
4	
5	`NSString *deletedString = [currentText substringToIndex:[currentText`
6	`length] - 1];`
7	
8	`_textLabel.text = deletedString;`
9	`}`
10	
11	`oldCursorTag = currentCursorTag;`

Penjelasan dari isi Kode Program 6.6 adalah sebagai berikut:

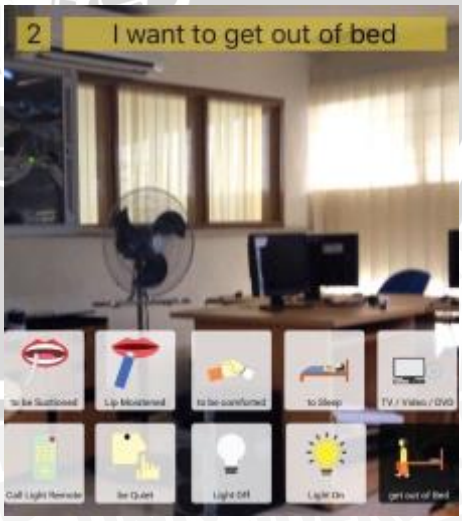
1. Baris 1 – 9 : Jika teks yang ada di dalam label textLabel tidak kosong, maka sistem akan melakukan aksi untuk menghapus karakter terakhir yang telah dipilih pengguna.
2. Baris 3 : Sistem telebih dahulu mengambil teks yang ada di dalam label textLabel dan menyimpannya ke dalam variabel dengan tipe data string bernama currentText.
3. Baris 5 – 6 : Instansiasi sebuah objek string bernama deletedString. Objek tersebut akan menyimpan variabel currentText yang telah dihapus sebanyak satu karakter. Hal ini dapat dilakukan dengan menggunakan

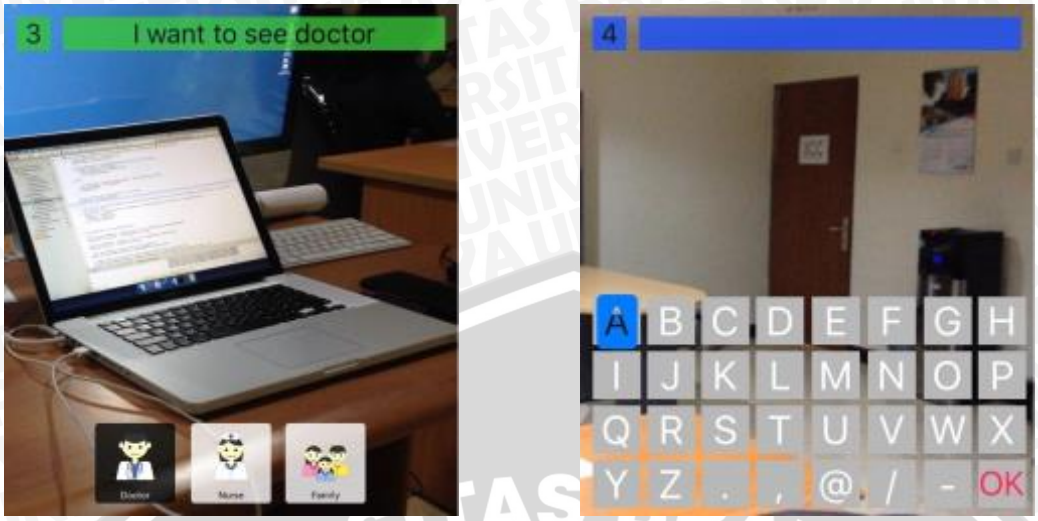
konstruktor *default class* NSString bernama substringToIndex dengan menyertakan parameter currentText.length - 1.

4. Baris 8 : Sistem menetapkan isi dari label textLabel dengan variabel deletedString.
5. Baris 11 : Sistem menetapkan isi variabel oldCursorTag dengan variabel currentCursorTag agar pada pemanggilan fungsi yang selanjutnya, variabel oldCursorTag dapat dibandingkan dengan variabel currentCursorTag yang baru.

6.5 Implementasi Antarmuka

Implementasi antarmuka aplikasi dilakukan di dalam *file* Main.storyboard, dimana implementasi tersebut mengacu pada perancangan *user experience* yang telah dijelaskan sebelumnya pada sub bab 5.2. Interaksi yang dilakukan pengguna untuk memindahkan posisi kursor di dalam sebuah *tab* dilakukan melalui gerakan kepala dengan durasi yang pendek (0,2 sampai dengan 0,6 detik), sedangkan interaksi yang dilakukan pengguna untuk memindahkan posisi *tab* atau memilih sebuah aksi tertentu di dalam sistem dilakukan melalui gerakan kepala dengan durasi yang panjang (0,7 sampai dengan 1,2 detik). Durasi tersebut mengacu kepada hasil pengujian interval pergerakan kepala yang telah dilakukan pada penelitian sebelumnya yang berjudul "Perancangan Kontrol Keyboard Menggunakan Head Movement System Dengan Pergerakkan Linier Pada Perangkat Bergerak Berbasis IOS". Adapun Gambar 6.1 merupakan hasil implementasi antarmuka dari aplikasi yang dikembangkan.





Gambar 6.1 Implementasi Antarmuka Aplikasi

Proses transisi antar *view* yang ada di dalam sistem dilakukan dengan adanya sebuah mekanisme *push* atau *pop* terhadap *view* yang ada saat itu. Ketika sebuah *view* baru di *push* terhadap *view* yang sekarang, pengguna akan dipindahkan ke *view* selanjutnya, begitupula sebaliknya apabila sebuah *view* di *pop*, maka pengguna akan dipindahkan ke *view* sebelumnya. Kedua mekanisme tersebut merupakan salah satu fitur yang diberikan oleh objek yang berasal dari *class* UINavigationController. UINavigationController merupakan sebuah *class* yang berperan penting dalam hal pengelolaan navigasi antar *view* yang ada di dalam sistem secara hierarkis. Mekanisme *push* dan *pull* ini bekerja layaknya struktur data *stack,* dimana *stack* memiliki prinsip kerja LIFO (*Last In, First Out*). Seluruh *view* yang ada di dalam sistem berperan sebagai penyusun dari *stack* UINavigationController, dimana implementasi *view* yang digunakan di dalam penelitian ini berupa tampilan *tab.* Mekanisme *push* dan *pop* dapat dilakukan dengan melakukan *embed* terhadap *root view* yang ada di dalam Main.storyboard terlebih dahulu. Ketika sebuah *root view* berhasil di *embed* ke dalam UINavigationController, maka *view* yang selanjutnya (*child view*) akan secara otomatis tertanam di dalam UINavigationController yang sama dengan *root view*. Dalam penelitian ini, *view* yang menjadi *root view* adalah *view* FirstViewController, sedangkan *view* yang menjadi *child view* adalah *view* SecondViewController, ThirdViewController, dan KeyboardViewController.

Proses transisi antar *view* yang ada di dalam sistem disebut dengan istilah *segue*. Ketika sebuah *segue* terjadi, maka sebuah animasi transisi antar *view* akan secara otomatis diputar dan ditampilkan melalui layar perangkat kepada pengguna. Animasi tersebut salah satunya bergantung kepada mekanisme *push* atau *pop* yang dilakukan oleh sistem. Apabila mekanisme *push* yang dilakukan, maka sistem akan melakukan instansiasi sebuah objek *view* yang berasal dari

class UIViewController berdasarkan *Storyboard ID* yang telah ditentukan pada *view* selanjutnya terlebih dahulu. Setelah itu, maka sistem akan memindahkan posisi *view* pengguna sekarang ke posisi *view* selanjutnya, yang sebenarnya merupakan objek *view* yang telah diinstansiasi sebelumnya selagi menampilkan animasi *segue* mendorong *view* ke kiri yang terlihat layaknya transisi ke halaman selanjutnya bagi pengguna. Apabila mekanisme *pop* yang dilakukan, maka sistem akan menampilkan animasi *segue* mendorong ke *view* kanan yang terlihat layaknya transisi ke halaman sebelumnya bagi pengguna.

BAB 7 PENGUJIAN

7.1 Pengujian Fungsional

Pengujian fungsional merupakan pengujian yang dilakukan untuk mengetahui apakah sistem yang telah diimplementasikan telah sesuai dengan kebutuhan fungsional yang telah didefinisikan diawal. Kebutuhan sistem yang akan digunakan dalam pengujian ini berasal kebutuhan fungsional yang sebelumnya telah dituliskan pada Tabel 4.2. Metode pengujian fungsional ini dilakukan dengan secara *black box*, dimana metode tersebut lebih menekankan kepada kesesuaian antara *input* dengan *output* sistem yang diinginkan tanpa membahas algoritme fungsi yang digunakan.

7.1.1 Penentuan Kasus Uji

Pengujian sistem dilakukan untuk mengetahui kesesuaian antara kebutuhan fungsional dengan implementasi sistem yang telah dibuat. Setiap kebutuhan fungsional akan diuji dengan menggunakan sejumlah kasus uji. Kasus uji yang baik mampu merepresentasikan seluruh kemungkinan jalur yang dapat dilakukan oleh pengguna terhadap sistem. Sebuah kasus uji terdiri atas kode kasus uji, nama kasus uji, objek yang diuji, tujuan pengujian, prosedur pengujian, dan hasil yang diharapkan dari pengujian. Adapun kasus uji yang ditentukan oleh peneliti ditujukan pada Tabel 7.1 sampai dengan Tabel 7.9 dibawah ini:

Tabel 7.1 Kasus Uji Pindah ke *Tab* Kedua dari *Tab* Pertama

Kode Kasus Uji	TESTCASE-HCB-001
Nama Kasus Uji	Pindah ke *tab* kedua dari *tab* pertama
Objek Uji	Kebutuhan Fungsional "Memindahkan posisi halaman / tab" (SRS-HCB-1.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat memindahkan posisi *tab* dari posisi *tab* pertama ke posisi *tab* kedua.
Prosedur Pengujian	1. Aktor membuka aplikasi *Health Communication Board*. 2. Aktor melakukan pergerakan kepala ke *axial left – long* sebanyak satu kali dari *tab* pertama.
Hasil yang Diharapkan	Sistem memindahkan posisi *tab* dari posisi *tab* pertama ke posisi *tab* kedua.

Tabel 7.2 Kasus Uji Pindah ke *Tab* Pertama dari *Tab* Kedua

Kode Kasus Uji	TESTCASE-HCB-002
Nama Kasus Uji	Pindah ke *tab* pertama dari *tab* kedua
Objek Uji	Kebutuhan Fungsional "Memindahkan posisi halaman / tab" (SRS-HCB-1.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat memindahkan posisi *tab* dari posisi *tab* kedua ke posisi *tab* pertama.
Prosedur Pengujian	1. Aktor membuka aplikasi *Health Communication Board*. 2. Aktor melakukan pergerakan kepala ke *axial left – long* sebanyak satu kali dari *tab* pertama. 3. Aktor melakukan pergerakan kepala *axial right – long* sebanyak satu kali dari *tab* kedua.
Hasil yang Diharapkan	Sistem memindahkan posisi *tab* dari posisi *tab* kedua ke *tab* pertama.

Tabel 7.3 Kasus Uji Pindah Kursor ke Kanan

Nomor Kasus Uji	TESTCASE-HCB-003
Nama Kasus Uji	Pindah kursor ke kanan
Objek Uji	Kebutuhan Fungsional "Memindahkan posisi kursor" (SRS-HCB-2.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat memindahkan posisi kursor ke arah kanan di dalam *tab* pertama.
Prosedur Pengujian	1. Aktor membuka aplikasi *Health Communication Board*. 2. Aktor melakukan pergerakan kepala *axial right – short* di dalam *tab* pertama.
Hasil yang Diharapkan	Sistem memindahkan posisi kursor ke arah kanan di dalam *tab* pertama.

Tabel 7.4 Kasus Uji Pindah Kursor ke Kiri

Nomor Kasus Uji	TESTCASE-HCB-004
Nama Kasus Uji	Pindah kursor ke kiri
Objek Uji	Kebutuhan Fungsional "Memindahkan posisi kursor" (SRS-HCB-2.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat memindahkan posisi kursor ke arah kiri di dalam *tab* pertama.
Prosedur Pengujian	1. Pengguna membuka aplikasi *Health Communication Board*. 2. Pengguna melakukan pergerakan kepala *axial left – short* di dalam *tab* pertama.
Hasil yang Diharapkan	Sistem memindahkan posisi kursor pengguna ke arah kiri di dalam *tab* pertama.

Tabel 7.5 Kasus Uji Pindah Kursor ke Atas

Nomor Kasus Uji	TESTCASE-HCB-005
Nama Kasus Uji	Pindah kursor ke atas
Objek Uji	Kebutuhan Fungsional "Memindahkan posisi kursor" (SRS-HCB-2.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat memindahkan posisi kursor ke arah atas.
Prosedur Pengujian	1. Pengguna membuka aplikasi *Health Communication Board*. 2. Pengguna melakukan pergerakan kepala *extension – short* di dalam *tab* pertama.
Hasil yang Diharapkan	Sistem memindahkan posisi kursor pengguna ke arah atas di dalam *tab* pertama.

Tabel 7.6 Kasus Uji Pindah Kursor ke Bawah

Nomor Kasus Uji	TESTCASE-HCB-006
Nama Kasus Uji	Pindah kursor ke bawah
Objek Uji	Kebutuhan Fungsional "Memindahkan posisi kursor" (SRS-HCB-2.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat memindahkan posisi kursor ke arah bawah di dalam *tab* pertama.
Prosedur Pengujian	1. Pengguna membuka aplikasi *Health Communication Board*. 2. Pengguna melakukan pergerakan kepala *flexion – short* di dalam *tab* pertama.
Hasil yang Diharapkan	Sistem memindahkan posisi kursor pengguna ke arah bawah di dalam *tab* pertama.

Tabel 7.7 Kasus Uji Pilih Aksi Pengguna

Nomor Kasus Uji	TESTCASE-HCB-007
Nama Kasus Uji	Pilih aksi pengguna
Objek Uji	Kebutuhan Fungsional "Memilih aksi" (SRS-HCB-3.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat memutar suara aksi pengguna yang dipilih.
Prosedur Pengujian	1. Pengguna membuka aplikasi *Health Communication Board*. 2. Pengguna mengarahkan posisi kursor pada aksi yang diinginkan. 3. Pengguna melakukan pergerakan kepala *flexion – long* pada aksi yang diinginkan.
Hasil yang Diharapkan	Sistem memutar suara terhadap aksi pengguna yang dipilih melalui *speaker* perangkat.

Tabel 7.8 Kasus Uji Memilih Ikon Karakter

Nomor Kasus Uji	TESTCASE-HCB-008
Nama Kasus Uji	Memilih ikon karakter
Objek Uji	Kebutuhan Fungsional "Menyusun kata" (SRS-HCB-4.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat menyimpan masukan karakter yang dipilih pengguna dalam proses menyusun suatu kata di dalam *tab* keempat.
Prosedur Pengujian	1. Pengguna membuka aplikasi *Health Communication Board*. 2. Pengguna berpindah posisi *tab* hingga berada pada *tab* keempat. 3. Pengguna memindahkan kursor ke arah ikon karakter yang diinginkan. 4. Pengguna melakukan pergerakan kepala *flexion – long* pada ikon karakter yang diinginkan.
Hasil yang Diharapkan	Sistem menampilkan karakter yang dipilih pengguna pada label yang berada di dalam *tab* keempat.

Tabel 7.9 Kasus Uji Hapus Karakter

Nomor Kasus Uji	TESTCASE-HCB-009
Nama Kasus Uji	Hapus karakter
Objek Uji	Kebutuhan Fungsional "Menghapus karakter" (SRS-HCB-5.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat menghapus karakter terakhir yang terlihat pada label yang berada pada *tab* keempat.
Prosedur Pengujian	1. Pengguna membuka aplikasi *Health Communication Board*. 2. Pengguna berpindah posisi *tab* hingga berada pada *tab* keempat. 3. Pengguna memindahkan kursor ke arah ikon karakter yang diinginkan.

	4. Pengguna melakukan pergerakan kepala *flexion – long* pada ikon karakter yang diinginkan. 5. Apabila karakter yang diinginkan telah terlihat pada label yang berada di dalam *tab* keempat, pengguna melakukan pergerakan kepala *extension – long* (jika masih belum terlihat, ulangi tahap no. 4 kembali).
Hasil yang Diharapkan	Sistem menghapus karakter terakhir yang terlihat pada label yang berada di dalam *tab* keempat.

Tabel 7.10 Kasus Uji Memutar Suara Kata yang Telah Disusun

Nomor Kasus Uji	TESTCASE-HCB-009
Nama Kasus Uji	Memutar suara kata yang telah disusun
Objek Uji	Kebutuhan Fungsional "Memutar suara kata" (SRS-HCB-6.0)
Tujuan Pengujian	Memastikan bahwa sistem dapat memutar suara terhadap kata yang disusun pengguna di dalam *tab* keempat.
Prosedur Pengujian	1. Pengguna membuka aplikasi *Health Communication Board*. 2. Pengguna berpindah posisi *tab* hingga berada pada *tab* keempat. 3. Pengguna memindahkan kursor ke arah ikon karakter yang diinginkan. 4. Pengguna melakukan pergerakan kepala *flexion – long* pada ikon karakter yang diinginkan. Apabila pengguna ingin menyusun sebuah kata, maka ulangi tahap 3 dan 4 hingga kata yang diinginkan berhasil ditampilkan pada label yang berada di dalam *tab* keempat. 5. Pengguna memindahkan kursor ke ikon bertuliskan "OK". 6. Pengguna melakukan pergerakan kepala *flexion – long* pada ikon bertuliskan "OK".
Hasil yang Diharapkan	Sistem memutar suara terhadap kata yang disusun pengguna melalui *speaker* perangkat.

7.1.2 Hasil Pengujian Fungsional

Pengujian fungsional sistem dilakukan berdasarkan kasus uji yang telah ditentukan sebelumnya. Secara keseluruhan, pengujian fungsionalitas sistem ini telah memberikan hasil akhir yang *valid*, yang berarti bahwa seluruh fungsionalitas sistem yang telah dikembangkan telah bekerja sesuai dengan hasil yang diharapkan oleh peneliti. Adapun hasil pengujian tersebut secara rinci ditujukan pada Tabel 7.11.

Tabel 7.11 Hasil Pengujian Fungsional

NO	Kode Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status
1.	TESTCASE-HCB-001	Sistem memindahkan posisi *tab* dari posisi *tab* pertama ke posisi *tab* kedua	Sistem berhasil memindahkan posisi *tab* dari posisi *tab* pertama ke posisi *tab* kedua	*Valid*
2.	TESTCASE-HCB-002	Sistem memindahkan posisi *tab* dari posisi *tab* kedua ke *tab* pertama	Sistem berhasil memindahkan posisi *tab* dari posisi *tab* kedua ke *tab* pertama	*Valid*
3.	TESTCASE-HCB-003	Sistem memindahkan posisi kursor ke arah kanan di dalam *tab* pertama	Sistem berhasil memindahkan posisi kursor ke arah kanan di dalam *tab* pertama	*Valid*
4.	TESTCASE-HCB-004	Sistem memindahkan posisi kursor ke arah kiri di dalam *tab* pertama	Sistem berhasil memindahkan posisi kursor ke arah kiri di dalam *tab* pertama	*Valid*
5.	TESTCASE-HCB-005	Sistem memindahkan posisi kursor ke arah atas di dalam *tab* pertama	Sistem berhasil memindahkan posisi kursor ke arah atas di dalam *tab* pertama	*Valid*
6.	TESTCASE-HCB-006	Sistem memindahkan posisi kursor ke arah bawah di dalam *tab*	Sistem berhasil memindahkan posisi kursor ke arah	*Valid*

		pertama	bawah di dalam *tab* pertama	
7.	TESTCASE-HCB-007	Sistem memutar suara terhadap aksi pengguna yang dipilih melalui *speaker* perangkat	Sistem berhasil memutar suara terhadap aksi pengguna yang dipilih melalui *speaker* perangkat	*Valid*
8.	TESTCASE-HCB-008	Sistem menampilkan karakter yang dipilih pengguna pada label yang berada di dalam *tab* keempat	Sistem berhasil menampilkan karakter yang dipilih pengguna pada label yang berada di dalam *tab* keempat	*Valid*
9.	TESTCASE-HCB-009	Sistem menghapus karakter terakhir yang terlihat pada label yang berada di dalam *tab* keempat	Sistem berhasil menghapus karakter terakhir yang terlihat pada label yang berada di dalam *tab* keempat	*Valid*
10.	TESTCASE-HCB-010	Sistem memutar suara terhadap kata yang disusun pengguna melalui *speaker* perangkat	Sistem berhasil memutar suara terhadap kata yang disusun pengguna melalui *speaker* perangkat	*Valid*

7.2 Pengujian Non Fungsional

Pengujian non fungsional merupakan pengujian yang dilakukan untuk mengetahui kualitas serta ke sistem yang telah diimplementasikan. Pengujian ini juga bertindak sebagai pendukung pengujian fungsional yang dilihat dari aspek kelayakan penggunaan sistem. Adapun parameter yang akan digunakan dalam proses pengujian non fungsional ini adalah parameter *usability,* yaitu parameter yang digunakan dalam menguji tingkat kepuasan pengguna dalam hal penggunaan sistem yang telah dikembangkan.

7.2.1 Pengujian *Usability*

Pengujian *usability* merupakan pengujian yang dilakukan untuk mengetahui tingkat kemudahan penggunaan dari sebuah sistem. Pengujian ini dilakukan dengan cara menguji sistem yang telah dikembangkan secara langsung

kepada sejumlah responden. Responden akan diminta untuk mengisi sebuah kuisioner berisi sejumlah pernyataan yang terkait dengan kenyamanan dan kemudahan penggunaan sistem. Pernyataan yang akan digunakan di dalam kuisioner penelitian kali ini berasal dari penelitian sebelumnya yang berjudul "*Toward Standard Usability Questionnaires for Handheld Augmented Reality*", dimana penelitian tersebut membahas pengukuran aspek usabilitas pada penggunaan teknologi *augmented reality* di sebuah perangkat bergerak. Sejumlah pernyataan yang ada pada penelitian tersebut akan diambil dan disesuaikan dengan konteks pada penelitian kali ini yang berfokus pada pemanfaatan pergerakan kepala sebagai kontrol aplikasi perangkat bergerak. Pernyataan yang ada di dalam kuisioner akan dibagi menjadi dua buah bagian, yaitu bagian pertama yang berisi lima buah pernyataan terkait masalah persepsi penggunaan aplikasi, dan bagian kedua yang berisi enam buah pernyataan terkait masalah ergonomi penggunaan aplikasi. Adapun daftar pernyataan kuisioner ditujukan pada Tabel 7.12. Skala penilaian yang digunakan dalam menilai kuisioner tersebut adalah skala likert, dimana penguji menggunakan lima buah skor dengan rentang 1 sampai dengan 5, dimana tiap skor yang ada memiliki keterangan sebagai berikut:

1 = Sangat tidak setuju
2 = Tidak setuju
3 = Netral
4 = Setuju
5 = Sangat setuju

Responden kemudian akan diminta untuk menjawab pernyataan yang ada di dalam kuisioner sesuai dengan tingkat kepuasan mereka dalam menggunakan sistem dengan cara mengisikan salah satu dari lima skala *likert* yang disediakan di dalam kuisioner.

Tabel 7.12 Daftar Pernyataan Kuisioner

NO	Daftar Pernyataan Kuisioner
Pernyataan Terkait Masalah Persepsi Penggunaan Aplikasi	
1.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha otot tubuh.
2.	Saya berpikir banyaknya informasi yang ditampilkan pada layar perangkat bergerak sudah sesuai.

3.	Saya berpikir bahwa informasi yang ditampikan pada layar perangkat mudah untuk dibaca.
4.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat sudah jelas.
5.	Saya berpikir bahwa kata dan ikon yang ditampilkan pada layar perangkat mudah untuk dibaca.
Pernyataan Terkait Masalah Ergonomi Penggunaan Aplikasi	
6.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha fisik.
7.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha mental.
8.	Saya merasa bahwa penggunaan aplikasi ini nyaman digunakan di kepala saya.
9.	Saya berpikir bahwa penggunaan aplikasi ini bersifat natural.
10.	Saya tidak merasakan kepala saya lelah setelah menggunakan aplikasi ini.
11.	Saya berpikir bahwa pengoperasian aplikasi ini simpel dan tidak rumit.

7.2.2 Analisis Hasil Pengujian *Usability*

Berdasarkan kuisioner yang telah disebarkan kepada lima buah responden, akan dilakukan analisis seputar hasil kuisioner yang didapatkan dengan menggunakan metode skala likert (detil jawaban kuisioner responden terlampir pada Lampiran 1 sampai dengan 5). Seluruh proses perhitungan skala likert yang digunakan pada penelitian ini mengacu pada proses perhitungan yang telah digunakan pada penelitian sebelumnya yang berjudul "Rancang Bangun Aplikasi TV *Online* Berbasis iOS" (Asfarina 2016). Rumus perhitungan total skor per pernyataan kuisioner ditunjukkan pada Persamaan 6.1 dan rumus perhitungan indeks per pernyataan kuisioner ditunjukkan pada Persamaan 6.2. Adapun klasifikasi tingkat kepuasan pengguna yang ditentukan oleh peneliti ditujukan pada Tabel 7.13, kemudian perhitungan indeks kepuasan pengguna ditujukan pada Tabel 7.14, dan hasil akhir pengujian parameter *usability* ditujukan pada Tabel 7.15.

Keterangan:

- Total skor per pernyataan = (1 x jumlah skor Sangat Tidak Setuju) + (2 x jumlah skor Tidak Setuju) + (3 x jumlah skor Netral) + (4 x jumlah skor Setuju) + (5 x jumlah skor Sangat Setuju) **(Persamaan 6.1)**

- Z = Skor *likert* tertinggi x Jumlah responden.
- Indeks per pernyataan (%) = (Total Skor / Z) x 100 **(Persamaan 6.2)**

Tabel 7.13 Klasifikasi Tingkat Kepuasan Pengguna

NO	Interval Indeks Kepuasan Pengguna	Tingkat Kepuasan Pengguna
1.	0 sampai dengan 20%	Sangat tidak memuaskan
2.	21 sampai dengan 40%	Tidak memuaskan
3.	41 sampai dengan 60%	Netral
4.	61 sampai dengan 80%	Memuaskan
5.	81 sampai dengan 100%	Sangat memuaskan

Tabel 7.14 Perhitungan Indeks Kepuasan Pengguna

NO	Daftar Pernyataan Kuisioner	Nilai / Skor					Total Skor	Index (%)
	Pernyataan Terkait Masalah Persepsi Penggunaan Aplikasi	1	2	3	4	5		
1.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha otot tubuh.	0	0	1	3	1	20	80%
2.	Saya berpikir banyaknya informasi yang ditampilkan pada layar perangkat bergerak sudah sesuai.	0	0	2	3	0	18	72%
3.	Saya berpikir bahwa informasi yang ditampikan pada layar perangkat mudah untuk dibaca.	0	0	1	4	0	19	76%
4.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat sudah jelas.	0	0	1	3	1	20	80%
5.	Saya berpikir bahwa kata dan ikon yang ditampilkan pada layar	0	0	1	3	1	20	80%

	perangkat mudah untuk dibaca.							
Pernyataan Terkait Masalah Ergonomi Penggunaan Aplikasi								
6.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha fisik.	0	0	2	2	1	19	76%
7.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha mental.	0	0	3	1	1	18	72%
8.	Saya merasa bahwa penggunaan aplikasi ini nyaman digunakan di kepala saya.	0	1	4	0	0	14	56%
9.	Saya berpikir bahwa penggunaan aplikasi ini bersifat natural.	0	0	3	2	0	17	68%
10.	Saya tidak merasakan kepala saya lelah setelah menggunakan aplikasi ini.	0	1	2	2	0	16	64%
11.	Saya berpikir bahwa pengoperasian aplikasi ini simpel dan tidak rumit.	0	0	0	4	1	21	84%

Tabel 7.15 Hasil Akhir Pengujian Parameter *Usability*

NO	Aspek Penilaian	Rata - rata Skor	Rata - rata Persentase	Tingkat Kepuasan Pengguna Akhir
1.	Persepsi pengguna dalam menggunakan aplikasi	23,4	77,6%	Memuaskan
2.	Ergonomi penggunaan aplikasi	17,5	72%	Memuaskan
Rata - rata		20,45	74,8%	Memuasakan

Pada pernyataan kuisioner yang membahas seputar persepsi penggunaan aplikasi, indeks kepuasan tertinggi didapatkan dari pernyataan yang berkaitan dengan kejelasan penyampaian informasi serta antarmuka yang ditampilkan pada layar perangkat, yaitu sebesar 80%. Hal tersebut menunjukkan bahwa rata – rata responden sudah merasa puas dengan desain antarmuka serta tata cara penyampaian informasi yang diberikan oleh sistem melalui layar perangkat kepada pengguna. Indeks kepuasan terendah didapatkan pada pernyataan yang berkaitan dengan kesesuaian kuantitas informasi yang ditampilkan pada layar perangkat, yaitu sebesar 72%. Hal tersebut menunjukkan rata – rata responden merasa kurang puas terhadap kuantitas informasi yang ditampilkan oleh sistem pada layar perangkat. Dengan begitu, perlu diadakan penelitian lebih lanjut seputar penentuan kuantitas informasi yang sesuai untuk ditampilkan di dalam sebuah aplikasi perangkat bergerak yang menerapkan konsep *augmented reality* di dalamnya.

Pada pernyataan kuisioner yang membahas seputar ergonomi penggunaan aplikasi, indeks kepuasan tertinggi didapatkan dari pernyataan yang berkaitan dengan kemudahan pengoperasian aplikasi, yaitu sebesar 84%. Hal tersebut menunjukkan bahwa pengguna sudah merasa puas dengan cara pengoperasian aplikasi yang simpel dan tidak rumit. Indeks kepuasan terendah didapatkan pada pernyataan yang berkaitan dengan kenyamanan penggunaan aplikasi di kepala pengguna, yaitu sebesar 54%. Peneliti memiliki hipotesis bahwa alasan pengguna memberikan indeks kepuasan yang rendah pada pernyataan ini dikarenakan penggunaan jenis *Head-Mounted Display* (HMD) yang dirasa kurang nyaman ketika digunakan pada kepala responden atau dikarenakan berat total perangkat (HMD dan ponsel) yang dirasa terlalu berat ketika dikenakan pada kepala responden untuk waktu yang lama. Dengan begitu, perlu diadakan penelitian lebih lanjut seputar penentuan jenis dan kriteria HMD yang nyaman digunakan pada kepala pengguna serta pemilihan ponsel yang memiliki berat yang lebih ringan untuk digunakan sebagai tampilan dari aplikasi yang dikembangkan bagi pengguna.

Hasil pengujian parameter *usability* secara keseluruhan menunjukkan rata – rata hasil sebesar 74,8%, dengan detil kepuasan pengguna dalam hal persepsi penggunaan aplikasi sebesar 77,6% dan kepuasan ergonomi penggunaan aplikasi sebesar 72%. Hal tersebut mengartikan bahwa sistem yang dikembangkan memberikan hasil yang memuaskan kepada pengguna. Namun begitu, untuk meningkatkan indeks kepuasan pengguna terhadap aplikasi yang telah dikembangkan, maka aplikasi ini perlu diuji kembali menggunakan sejumlah parameter kepuasan pengguna lainnya. Selain itu, aplikasi ini juga perlu diujikan kepada lebih banyak responden selagi menambal sejumlah kekurangan yang telah ditemukan pada hasil pengujian parameter *usability* kali ini, baik itu dari segi fungsional, persepsi maupun ergonomi aplikasi.

BAB 8 PENUTUP

8.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan sebelumnya oleh peneliti, maka dapat diambil kesimpulan penelitian sebagai berikut:

1. Rancangan aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS (*Head Movement Control System*) berbentuk sebuah tampilan kamera pada layar perangkat bergerak yang di dalamnya ditambahkan sejumlah elemen grafik berupa sejumlah tampilan ikon komunikasi berisikan aksi pengguna tertentu. Aplikasi ini mengimplementasikan konsep *augmented reality*, dimana selain dapat melihat dunia sekitarnya, pengguna juga dapat berinteraksi dengan sejumlah elemen yang ditampilkan di dalam tampilan dunia tersebut. Tipe interaksi yang digunakan pada rancangan aplikasi ini adalah *mediated interaction*, dimana pengguna menggunakan bantuan gerakan kepala dalam berinteraksi dengan aplikasi.
2. Implementasi metode kendali pergerakan kepala pada aplikasi *Health Communication Board* berbasis kendali pergerakan linear HEMOCS (*Head Movement Control System*) digunakan pengguna pada saat melakukan perpindahan antar *tab*, memindahkan kursor, memilih aksi pengguna, memilih karakter, menghapus karakter, serta memutar suara dari kata yang telah disusun oleh pengguna. Metode tersebut dilakukan dengan mengambil data pergerakan yang terjadi pada perangkat lalu mengecek pergerakan tersebut terhadap sejumlah batasan dan kondisi yang telah ditentukan, seperti halnya pengecekan terhadap batasan sudut pergerakan kepala minimum yang harus dilakukan pengguna sebelum sebuah pergerakan dapat dibaca oleh sistem, serta pengecekan batas pergerakan kepala dalam menentukan jenis pergerakan kepala durasi pendek (*short*) atau panjang (*long*). Hasil akhir metode tersebut digunakan dalam menentukan aksi yang akan dijalankan sistem ke depan.
3. Besaran tingkat usabilitas aplikasi yang dilihat dari sudut pandang persepsi penggunaan aplikasi menunjukkan indeks kepuasan pengguna sebesar 77,6%, sedangkan jika dilihat dari sudut pandang ergonomi penggunaan aplikasi menunjukkan indeks kepuasan pengguna sebesar 72%. Dengan rata – rata indeks kepuasan sebesar 74,8%, maka sistem dapat dikatakan bekerja dengan hasil yang memuaskan pada pengguna.

8.2 Saran

Adapun saran bagi pengembangan sistem yang selanjutnya adalah sebagai berikut:

1. Fitur untuk menetapkan tingkat sensitivitas pembacaan pergerakan kepala pengguna secara dinamis dapat dikembangkan pada penelitian yang selanjutnya.
2. *Activity Daily Language* (ADL) pasien yang didapatkan dari dokter dapat digunakan sebagai subtitusi aksi pengguna yang berasal dari *Vidatak EZ Picture Board*.
3. Pengembangan aplikasi selanjutnya dapat ditambahkan dengan fitur Bahasa Indonesia yang diterapkan pada aksi pengguna yang digunakan di dalam sistem.

DAFTAR PUSTAKA

Arai, K. 2013. "Wearable Computing System with Input-Output Devices Based on Eye-Based Human Computer Interaction Allowing Location Based Web Services." *International Journal of Advanced reasearch in Artificial Intelligence* 34-39.

Tolle, H., A. Pinandito, E.M Adams J, dan K. Arai. 2015. "Virtual Reality Game Controlled with User's Head and Body Movement Detection Using Smartphone Sensors." *ARPN Journal Engineering and Applied Sciences* 10: 9776-9782.

Arai, Kohei, Herman Tolle, dan Akihiro Serita. 2013. "Mobile Device Based 3D Image Display Depending on User's Action and Movement." *International Journal of Advanced Research in Artificial Intelligence* 71-78.

Safii, Imam. 2016. *Pengembangan Metode Pendeteksian Pergerakan Kepala Berbasis Sensor Internal pada Perangkat Bergerak Berbasis iOS.* Malang: Universitas Brawijaya.

Rusyda, Rezqi Andini. 2016. *Perancangan Kontrol Keyboard Menggunakan Head Movement Control System Dengan Pergerakan Linier pada Perangkat Bergerak Berbasis iOS.* Malang: Universitas Brawijaya.

Bachand, Michael, dan Adam Michela. 2015. *Airbnb Nerds.* 11 September. Diakses Oktober 9, 2016. http://nerds.airbnb.com/tvos-focus-engine/.

2016. *Livescience.* 1 Oktober. http://www.livescience.com/40103-accelerometer-vs-gyroscope.html.

Apple. 2016. *Apple.* 3 Oktober. https://developer.apple.com/reference/coremotion.

Allan, A. 2011. *Basic Sensor in iOS.* O'Reilly Media.

2016. *Burak Kanber, Engineer.* Januari. Diakses Oktober 9, 2016. https://www.burakkanber.com/blog/google-cardboard-stereoscopic-phones-and-the-future-of-augmented-reality/.

2016. *VIDATAK Innovation in Patient Communication.* 12 12. http://www.vidatak.com/ezboards.html.

Hegarty, Paul. 2013. *Developing Applications for iOS.* California.

Marc Ericson C. Santos, Jarkko Polvi, Takafumi Taketomi, Goshiro Yamamoto, Christian Sandor, Hirokazu Kato. 2015. "Toward Standard Usability Questionnaires for Handheld Augmented Reality." 11.

Kurniawan, Tri Astoto. 2012. *Rekayasa Kebutuhan - Konsep.* Malang.

Asfarina. 2016. "Rancang Bangun Aplikasi TV Online Berbasis iOS." 93.

Tolle, Herman, dan Kohei Arai. 2016. "Design of Head Movement Controller System (HEMOCS) for Control Mobile Application through Head Pose

Movement Detection." *International Journal of Interactive Mobile Technologies (iJIM)* 10: 24-28.

Tolle, Herman, Ismiarta Aknuranda, Mahardeka Tri Ananta, Komang Candra Brata, dan Hanifah Muslimah Az-Zahra. 2016. "Design of keyboard input control for mobile application using Head Movement Control (HEMOCS)." *International Conference on Advanced Computer Science and Information Systems (ICACSIS)* 518-523.

Kochan, Stephen G. 2014. *Programming in Objective-C Sixth Edition.* US.

BAKER, STEPHAN. 2016. *IOS DEVICE MOTION: APPS WITH ATTITUDE.* 19 May. Diakses May 26, 2017. https://www.forestgiant.com/articles/ios-core-motion/.

LAMPIRAN

KUISIONER USABILITAS APLIKASI HEALTH COMMUNICATION BOARD

Nama : [illegible]
Umur : 21 tahun
Jenis Kelamin : (L) / P

NO	Daftar Pernyataan Kuisioner	Nilai *				
		1	2	3	4	5
	Pernyataan Terkait Masalah Persepsi Penggunaan Aplikasi					
1.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha otot tubuh.				✓	
2.	Saya berpikir banyaknya informasi yang ditampilkan pada layar perangkat bergerak sudah sesuai.				✓	
3.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat mudah untuk dibaca.				✓	
4.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat sudah jelas.					✓
5.	Saya berpikir bahwa kata dan ikon yang ditampilkan pada layar perangkat mudah untuk dibaca.				✓	
	Pernyataan Terkait Masalah Ergonomi Penggunaan Aplikasi					
6.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha fisik.				✓	
7.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha mental.				✓	
8.	Saya merasa bahwa penggunaan aplikasi ini nyaman digunakan di kepala saya.			✓		
9.	Saya berpikir bahwa penggunaan aplikasi ini bersifat natural.				✓	
10.	Saya tidak merasakan kepala saya lelah setelah menggunakan aplikasi ini.			✓		
11.	Saya berpikir bahwa pengoperasian aplikasi ini simpel dan tidak rumit.				✓	

* 1 = Sangat tidak setuju
2 = Tidak setuju
3 = Netral
4 = Setuju
5 = Sangat setuju

Saran:

Semangat ujian

Lampiran 1 - Hasil Kuisioner Usabilitas Aplikasi (Responden 1)

KUISIONER USABILITAS APLIKASI HEALTH COMMUNICATION BOARD

Nama : [illegible]
Umur : 21 tahun
Jenis Kelamin : L / P

NO	Daftar Pernyataan Kuisioner	Nilai *				
		1	2	3	4	5
	Pernyataan Terkait Masalah Persepsi Penggunaan Aplikasi					
1.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha otot tubuh.					✓
2.	Saya berpikir banyaknya informasi yang ditampilkan pada layar perangkat bergerak sudah sesuai.				✓	
3.	Saya berpikir bahwa informasi yang ditampikan pada layar perangkat mudah untuk dibaca.			✓		
4.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat sudah jelas.				✓	
5.	Saya berpikir bahwa kata dan ikon yang ditampilkan pada layar perangkat mudah untuk dibaca.			✓		
	Pernyataan Terkait Masalah Ergonomi Penggunaan Aplikasi					
6.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha fisik.					✓
7.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha mental.					✓
8.	Saya merasa bahwa penggunaan aplikasi ini nyaman digunakan di kepala saya.			✓		
9.	Saya berpikir bahwa penggunaan aplikasi ini bersifat natural.			✓		
10.	Saya tidak merasakan kepala saya lelah setelah menggunakan aplikasi ini.				✓	
11.	Saya berpikir bahwa pengoperasian aplikasi ini simpel dan tidak rumit.				✓	

* 1 = Sangat tidak setuju
2 = Tidak setuju
3 = Netral
4 = Setuju
5 = Sangat setuju

Saran:

Mungkin bisa dengan model stereoskopik agar pengguna tidak merasa pusing.

Lampiran 2 - Hasil Kuisioner Usabilitas Aplikasi (Responden 2)

KUISIONER USABILITAS APLIKASI HEALTH COMMUNICATION BOARD

Nama : Dwi Putra P.
Umur : 20 tahun
Jenis Kelamin : L / P

NO	Daftar Pernyataan Kuisioner	Nilai *				
		1	2	3	4	5
	Pernyataan Terkait Masalah Persepsi Penggunaan Aplikasi					
1.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha otot tubuh.			✓		
2.	Saya berpikir banyaknya informasi yang ditampilkan pada layar perangkat bergerak sudah sesuai.				✓	
3.	Saya berpikir bahwa informasi yang ditampikan pada layar perangkat mudah untuk dibaca.				✓	
4.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat sudah jelas.				✓	
5.	Saya berpikir bahwa kata dan ikon yang ditampilkan pada layar perangkat mudah untuk dibaca.					✓
	Pernyataan Terkait Masalah Ergonomi Penggunaan Aplikasi					
6.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha fisik.			✓		
7.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha mental.			✓		
8.	Saya merasa bahwa penggunaan aplikasi ini nyaman digunakan di kepala saya.			✓		
9.	Saya berpikir bahwa penggunaan aplikasi ini bersifat natural.			✓		
10.	Saya tidak merasakan kepala saya lelah setelah menggunakan aplikasi ini.			✓		
11.	Saya berpikir bahwa pengoperasian aplikasi ini simpel dan tidak rumit.				✓	

* 1 = Sangat tidak setuju
2 = Tidak setuju
3 = Netral
4 = Setuju
5 = Sangat setuju

Saran:

Gunakan konsep stereoscopic agar pengalaman nyaman saat menggunakan aplikasi.

Lampiran 3 - Hasil Kuisioner Usabilitas Aplikasi (Responden 3)

KUISIONER USABILITAS APLIKASI HEALTH COMMUNICATION BOARD

Nama : Ahmad Leo Yudano
Umur : 22 tahun
Jenis Kelamin : (L) / P

NO	Daftar Pernyataan Kuisioner	Nilai *				
		1	2	3	4	5
	Pernyataan Terkait Masalah Persepsi Penggunaan Aplikasi					
1.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha otot tubuh.				✓	
2.	Saya berpikir banyaknya informasi yang ditampilkan pada layar perangkat bergerak sudah sesuai.			✓		
3.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat mudah untuk dibaca.				✓	
4.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat sudah jelas.			✓		
5.	Saya berpikir bahwa kata dan ikon yang ditampilkan pada layar perangkat mudah untuk dibaca.				✓	
	Pernyataan Terkait Masalah Ergonomi Penggunaan Aplikasi					
6.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha fisik.			✓		
7.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha mental.			✓		
8.	Saya merasa bahwa penggunaan aplikasi ini nyaman digunakan di kepala saya.			✓		
9.	Saya berpikir bahwa penggunaan aplikasi ini bersifat natural.				✓	
10.	Saya tidak merasakan kepala saya lelah setelah menggunakan aplikasi ini.		✓			
11.	Saya berpikir bahwa pengoperasian aplikasi ini simpel dan tidak rumit.					✓

* 1 = Sangat tidak setuju
2 = Tidak setuju
3 = Netral
4 = Setuju
5 = Sangat setuju

Saran:

Lampiran 4 - Hasil Kuisioner Usabilitas Aplikasi (Responden 4)

KUISIONER USABILITAS APLIKASI HEALTH COMMUNICATION BOARD

Nama : Sigoni & Fanisa
Umur : 23 tahun
Jenis Kelamin : L / P

NO	Daftar Pernyataan Kuisioner	Nilai *				
		1	2	3	4	5
	Pernyataan Terkait Masalah Persepsi Penggunaan Aplikasi				✓	
1.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha otot tubuh.				✓	
2.	Saya berpikir banyaknya informasi yang ditampilkan pada layar perangkat bergerak sudah sesuai.			✓		
3.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat mudah untuk dibaca.				✓	
4.	Saya berpikir bahwa informasi yang ditampilkan pada layar perangkat sudah jelas.				✓	
5.	Saya berpikir bahwa kata dan ikon yang ditampilkan pada layar perangkat mudah untuk dibaca.				✓	
	Pernyataan Terkait Masalah Ergonomi Penggunaan Aplikasi					
6.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha fisik.				✓	
7.	Saya berpikir bahwa berinteraksi dengan aplikasi ini tidak membutuhkan banyak usaha mental.			✓		
8.	Saya merasa bahwa penggunaan aplikasi ini nyaman digunakan di kepala saya.		✓			
9.	Saya berpikir bahwa penggunaan aplikasi ini bersifat natural.			✓		
10.	Saya tidak merasakan kepala saya lelah setelah menggunakan aplikasi ini.				✓	
11.	Saya berpikir bahwa pengoperasian aplikasi ini simpel dan tidak rumit.				✓	

* 1 = Sangat tidak setuju
2 = Tidak setuju
3 = Netral
4 = Setuju
5 = Sangat setuju

Saran:

Lampiran 5 - Hasil Kuisioner Usabilitas Aplikasi (Responden 5)