TUGAS AKHIR – TE145561

PERANCANGAN ANTARMUKA *OPERATOR CONTROL UNIT* RHINO ROBOT DENGAN MENGGUNAKAN *DEVELOPMENT BOARD* LATTEPANDA

Livian Tjandra
NRP. 10311500000056

Pembimbing
Dr. Ir. Achmad Affandy, DEA.
Imam Arifin, S.T., M.T.
M. Fajar Adityo, S.T.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – TE145561

DESIGNING INTERFACE OF RHINO ROBOT'S OPERATOR CONTROL UNIT USING DEVELOPMENT BOARD LATTEPANDA

Livian Tjandra
NRP. 10311500000056

Advisor
Dr. Ir. Achmad Affandy, DEA.
Imam Arifin, S.T., M.T.
M. Fajar Adityo, S.T.

AUTOMATION ELECTRONIC ENGINEERING DEPARTMENT
Vocational Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2018

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul:

"PERANCANGAN ANTARMUKA *OPERATOR CONTROL UNIT* RHINO ROBOT DENGAN MENGGUNAKAN *DEVELOPMENT BOARD* LATTEPANDA"

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya saya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 11 Juli 2018

Livian Tjandra
NRP. 10311500000056

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

PERANCANGAN ANTARMUKA *OPERATOR CONTROL UNIT* RHINO ROBOT DENGAN MENGGUNAKAN *DEVELOPMENT BOARD* LATTEPANDA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Achmad Affandi DEA	Imam Arifin, S.T., M.T.
NIP. 196510141990021001	NIP. 197302222002121001

SURABAYA
JUNI, 2018

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

PERANCANGAN ANTARMUKA OCU RHINO ROBOT DENGAN MENGGUNAKAN *DEVELOPMENT BOARD* LATTEPANDA DI PT. BHIMASENA RESEARCH AND DEVELOPMENT

TUGAS AKHIR

Disusun oleh:

Livian Tjandra NRP. 10311500000056

Menyetujui,

Kepala *Human Resources Department*,

Fadli Tirmissi
NIK. 020492016

Pembimbing Perusahaan,

M. Fajar Adityo, S.T.
NIK. 120322016

Chief Executive Officer,

BHIMASENA Research & Development

Dipl. -Ing. Aris Budiyarto

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

PERANCANGAN ANTARMUKA *OPERATOR CONTROL UNIT* RHINO ROBOT DENGAN MENGGUNAKAN *DEVELOPMENT BOARD* LATTEPANDA

Livian Tjandra
10311500000056

Pembimbing I : Dr. Ir. Achmad Affandi, DEA
Pembimbing II : Imam Arifin, S.T., M.T.
Pembimbing III: M. Fajar Adityo, S.T.

ABSTRAK

Rhino Robot bekerja berdasarkan kendali yang diberikan oleh pengguna melalui sebuah perangkat bernama *Operator Control Unit* (OCU). Saat ini, perangkat yang telah dikembangkan berupa sebuah laptop *toughbook* yang dilengkapi dengan *gamepad* sebagai kendalinya. Bentuk seperti ini dirasa menyusahkan pengguna, karena pada OCU terdapat sejumlah tombol yang tidak digunakan dalam mengoperasikan robot. Selain itu, juga diperlukan penggunaan kombinasi tombol pada *gamepad* untuk menentukan menu operasi yang dibutuhkan.

Oleh karena itu, dilakukan pengembangan lanjutan untuk perangkat kendali ini. Rancangan yang digunakan antara lain menjadikan OCU memiliki sebuah panel *input* khusus yang berisi sejumlah tombol dan *joystick.* Dimana tombol-tombol tersebut memiliki fungsi khusus untuk mengoperasikan robot. Tombol-tombol kemudian diletakkan sesuai fungsinya terhadap proses kendali robot, dengan bantuan visualisasi berupa gambar dari Rhino Robot. Penelitian ini membahas mengenai proses pembuatan antarmuka dari tiap-tiap tombol. Selain itu, juga membahas mengenai proses pembuatan antarmuka LED yang merupakan indikator dari sejumlah tombol. Proses penyusunan memanfaatkan *co-processor* Arduino pada LattePanda, dengan Visual Studio sebagai IDE penyusun program.

Hasil pengujian dari sejumlah tombol pada panel menggunakan program yang telah disusun, menunjukkan bahwa program telah mampu menjalankan fungsi antarmuka dari masing-masing tombol dan LED indikatornya.

Kata kunci : Rhino Robot, OCU, Antarmuka, LattePanda

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

DESIGNING INTERFACE OF RHINO ROBOT'S OPERATOR CONTROL UNIT USING DEVELOPMENT BOARD LATTEPANDA

Livian Tjandra
10311500000056

Advisor I : Dr. Ir. Achmad Affandi, DEA
Advisor II : Imam Arifin, S.T., M.T.
Advisor III : M. Fajar Adityo, S.T.

ABSTRACT

Rhino Robot work based on the control given by the user through a control device named Operator Control Unit (OCU). Currently, device that has been developed are form laptop toughbook with gamepad as controller. This design were inconvenience user, because there are batch of buttons which are not used to operate the robot. Moreover, using combination keys in gamepad to choose menu operation are required.

Therefore, further development for the control device of Rhino Robot is do. The design were used is in OCU consist of an panel input that contains a several buttons and joystick which have a specific function to operate Rhino Robot. The buttons will be placed according to its function to the robot control process, using visualization of Rhino Robot to help user understanding the function of each buttons. This study will discuss the process to making interface of each buttons. In addition, it also discusses the process how to make an interface of LED which used to buttons indicator. The compilation process utilizes the existing LattePanda's Arduino co-processor using Visual Studio to create a program.

After testing of the several buttons using program that have been create, it is known that the program have been able to perform each function of buttons and LED in input panel.

***Keywords** : Rhino robot, OCU, Interface, LattePanda*

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

KATA PENGANTAR

Segala puji syukur penulis ucapkan atas kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat, taufik, serta hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang telah dilaksanakan di PT Bhimasena Research and Development.

Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Diploma 3 pada Departemen Teknik Elektro Otomasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember (ITS) Surabaya dengan judul:

PERANCANGAN ANTARMUKA *OPERATOR CONTROL UNIT* RHINO ROBOT DENGAN MENGGUNAKAN *DEVELOPMENT BOARD* LATTEPANDA

Dalam proses pengerjaan, penulis telah banyak mendapat bantuan, dukungan, dan bimbingan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis mengucapkan terimakasih kepada beberapa pihak yang terlibat. Mulai dari orang tua dan seluruh anggota keluarga yang selalu memberikan semangat dan doa kepada penulis. Bapak Ir. Joko Susila, M.T., selaku Ketua Jurusan Departemen Teknik Elektro Otomasi ITS Surabaya yang telah mengantarkan penulis hingga bisa melaksanakan magang di PT Bhimasena Research and Development. Bapak Imam Arifin, S.T., M.T., selaku ketua Laboratorium Sistem Komputer dan Otomasi serta dosen pembimbing yang telah memberikan arahan dan bimbingannya, serta telah mengantarkan penulis untuk dapat melaksanakan magang di PT Bhimasena Research and Development. Bapak Dr. Ir. Achmad Affandy, DEA, selaku dosen pembimbing yang selalu memberikan arahan dan bimbingannya. Bapak Aris Budiyarto, Dipl.-Ing., selaku CTO di PT Bhimasena Research and Development yang telah memberikan izin kepada penulis untuk menyelesaikan Tugas Akhir di PT Bhimasena Research and Development. Bapak Fadli Tirmissi, selaku HRD yang bertanggung jawab atas seluruh kegiatan penulis di PT Bhimasena Research and Development. Mas M. Fajar Adityo, S.T. yang telah membimbing penulis selama proses pengerjaan Tugas Akhir berlangsung. Manajemen dan staf PT Bhimasena Research and Development yang memberikan kesempatan dan membantu penulis selama proses magang berlangsung. Teman-teman magang di PT Bhimasena Research and Development yang selalu menemani dan memberikan dukungannya kepada penulis selama pengerjaan Tugas

Akhir. Hingga pihak-pihak yang tidak dapat disebutkan satu persatu oleh penulis yang telah membantu selama proses pengerjaan Tugas Akhir maupun magang berlangsung.

Penulis menyadari bahwa laporan Tugas Akhir ini belum sempurna, oleh karena itu saran dan masukan sangat diharapkan untuk perbaikan di masa yang akan datang. Semoga laporan ini bermanfaat bagi pembaca dan masyarakat pada umumnya.

Surabaya, 11 Juli 2018

Livian Tjandra
10311500000056

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

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

BAB I
PENDAHULUAN

1.1 Latar Belakang

PT Bhimasena Research and Development merupakan sebuah perusahaan yang bergerak dibidang penelitian dan pengembangan. Produk yang dikembangkan antara lain kendaraan udara, laut, maupun darat yang berorientasi kemiliteran. Penelitian untuk masing-masing kendaraan dilakukan pada divisi yang berbeda. Salah satunya yaitu Rhino Robot, sebuah divisi yang bertugas untuk melakukan penelitian tentang robot EOD (*Explosive Ordnance Disposal*).

Secara global, robot EOD merupakan robot yang dikembangkan untuk menjalankan misi *explosive, ordnance,* dan *disposal.* Sebuah misi untuk melakukan pengintaian, pemindahan, dan penanganan benda-benda berbahaya, seperti benda yang mengandung bahan peledak [1]. Tujuan utama dari pengembangan robot ini yaitu untuk menggantikan manusia dalam misi tersebut [2].

Dalam menjalankan perannya, robot ini memanfaatkan bagian penyusunnya, yaitu *arm, chassis, flipper,* dan *gripper. Arm* (lengan robot) berfungsi untuk mengambil objek yang menjadi sasaran melalui subbagian bernama *gripper* (penggenggam). Sedangkan *chasis* berfungsi sebagai kaki dari robot, menjadikannya dapat bergerak dari satu tempat ke tempat lainnya. Selain itu, pada *chasis* juga dilengkapi dengan *flipper* yang berfungsi untuk membantu pergerakan robot ketika menaiki maupun menuruni tangga. Untuk dapat mengoperasikan robot ini, dibutuhkan sebuah perangkat yang mampu mengirimkan perintah kendali.

Secara spesifik, robot EOD yang dikembangkan oleh PT Bhimasena Research and Development diberi nama Rhino Robot, sesuai dengan divisi yang mengembangkannya. Bentuk dari robot ini seperti robot EOD pada umumnya, yang terdiri dari *arm* dan *chasis* yang telah dilengkapi dengan *gripper* dan *flipper.* Perangkat kendali yang digunakan untuk mengirimkan perintah operasi kepada Rhino Robot bernama *Operator Control Unit* (OCU) [3].

Hingga saat ini, perangkat yang telah dikembangkan berupa sebuah laptop *toughbook* dan *gamepad*. Laptop digunakan untuk menampilkan *Graphical User Interface* (GUI) yang berisi menu-menu pengoperasian robot. Sedangkan *gamepad* digunakan untuk

melakukan pemilihan menu pada GUI. Bentuk perangkat seperti ini dirasa menyulitkan pengguna, karena pada laptop terdapat sejumlah tombol yang tidak digunakan selama proses pengoperasian robot. Selain itu, pengguna juga harus memahami fungsi dari tiap-tiap tombol pada *gamepad* dan cara melakukan kombinasinya. Akan tetapi pada praktiknya, sebagian besar dari pengguna menginginkan perangkat yang praktis dan mudah dioperasikan.

Oleh karena itu, dilakukan pengembangan lanjutan mengenai perangkat kendali dari Rhino Robot. Dimana bentuk baru dari OCU Rhino Robot ini terdiri dari sebuah panel yang memiliki *input* berupa tombol-tombol dan *joystick*. Kemudian dihubungkan dengan LCD untuk menampilkan GUI. Perancangan yang dilakukan dimulai dari desain *hardware* panel, seperti peletakan tombol-tombol dan komponen yang diperlukan. Dilanjutkan dengan perancangan *software* antarmuka dari tombol-tombol pada panel *input*. GUI dari OCU versi lama dilakukan perombakan halaman yang ditampilkan, sesuai kebutuhan. Untuk dapat menjalankan semua perancangan yang dibuat, dibutuhkan sebuah *power distribution* yang dapat menyalurkan daya ke masing-masing komponen penyusun OCU.

Perancangan dari panel *input* OCU dilakukan untuk menjadikan tombol-tombol pada panel dapat berfungsi sesuai kebutuhan dalam pengoperasian robot. Diletakan sesuai fungsi kendalinya dengan bantuan visualisasi gambar dari Rhino Robot. Dimana sejumlah dari tombol-tombol tersebut dilengkapi dengan LED indikator. Masing-masing tombol memiliki peran yang berbeda dalam memberikan perintah untuk mengoperasikan robot.

Untuk dapat menjalankan fungsi dari tombol-tombol pada panel *input*, dibutuhkan proses antarmuka. Metode yang digunakan untuk dapat menjalankan fungsi antarmuka dari tombol-tombol beserta LED indikatornya memanfaatkan sebuah *development board* bernama LattePanda yang telah terintegrasi *co-processor* Arduino [4]. Antarmuka tersebut dibuat dengan menggunakan bahasa pemrograman C# yang beroperasi diatas *framework.NET* yang terdapat pada *software* Visual Studio.

Penelitian yang berfokus pada perancangan antarmuka panel *input* OCU Rhino Robot ini dilakukan untuk dapat menjalankan fungsi operasi dari masing-masing tombol yang digunakan. Selain itu juga untuk menjalankan fungsi dari LED yang dirancang sebagai indikator tombol.

1.2 Permasalahan

Pengembangan dari OCU Rhino Robot dilakukan sedemikian rupa hingga terdapat sebuah panel *input,* yang tersusun dari sejumlah tombol. Untuk dapat mengendalikan robot, tombol-tombol tersebut harus mampu beroperasi dengan baik. Selain itu juga dibutuhkan sebuah indikator untuk mengetahui apakah tombol sedang digunakan atau tidak. Metode yang dapat digunakan untuk menjadikan tombol dan LED indikator dapat berfungsi sesuai kebutuhan yaitu dengan melakukan penyusunan fungsi antarmuka dari masing-masing tombol dan LED.

1.3 Batasan Masalah

Perancangan antarmuka dari masing-masing tombol dan LED pada panel *input* disusun menggunakan *co-processor* Arduino yang ada pada *development board* LattePanda. Dimana program yang digunakan untuk menjalankan fungsi antarmuka disusun pada *software* Visual Studio, dengan bahasa pemrograman C#.

1.4 Tujuan

Penelitian ini bertujuan untuk mengoperasikan tombol-tombol pada panel *input* sesuai dengan fungsi yang dibutuhkan. Serta memudahkan pengguna untuk mengetahui keadaan dari masing-masing tombol, apakah tengah digunakan atau tidak.

1.5 Metodologi Penelitian

Penelitian ini dilakukan secara bertahap, dimulai dari persiapan hingga penyusunan laporan. Pada tahap persiapan, proses yang dilakukan yaitu studi literatur mengenai LattePanda, *board* penyusun OCU, dan pemrograman C# pada Visual Studio. Selain itu, untuk mendukung pemahaman implementasi masing-masing *board* penyusun OCU, perlu diketahui cara kerja dari komunikasi serial dan *multiplexer*.

Setelah materi yang dipersiapkan dirasa sudah cukup membantu, maka dilakukan perancangan sistem. Pada tahap ini, kegiatan yang dilakukan yaitu merancang program untuk menjalankan fungsi antarmuka dari masing-masing tombol dan LED indikator.

Perancangan program disusun pada *software* Visual Studio dengan menggunakan bahasa pemrograman C#.

Tahap yang dilakukan setelah perancangan sistem terselesaikan yaitu pengujian. Pada tahap ini, pengujian yang dilakukan bertujuan untuk mengetahui ketepatan fungsi kerja dari program terhadap kegunaan tiap-tiap *board* penyusun OCU. Mulai dari ketepatan fungsi antarmuka masing-masing tombol pada panel *input,* hingga ketepatan fungsi dari masing-masing LED indikator. Setelah pengujian terselesaikan, kemudian dilakukan analisis terhadap hasil yang didapatkan.

Bagian terakhir yang dilakukan yaitu penyusunan laporan. Bagian ini disusun dengan melakukan pengumpulan data hasil pengujian, analisis, dan perancangan sistem. Selain itu, juga dilakukan pengumpulan referensi yang dijadikan acuan selama proses pengerjaan.

1.6 Sistematika Laporan

Dari hasil penelitian yang telah dilakukan, kemudian disusun sebuah laporan yang berisi penjelasan dari awal mula dilakukannya penelitian hingga hasil yang didapatkan. Penyusunan dari laporan ini sendiri dilakukan berdasarkan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Penyusunan pendahuluan dari laporan ini diletakkan pada bab satu. Isi pembahasannya diantaranya penjelasan dari latar belakang, permasalahan, batasan masalah, tujuan, metodologi penelitian, serta sistematika laporan dari penelitian ini.

BAB II ANTARMUKA OCU RHINO ROBOT

Bab dua disusun untuk menjelaskan teori dasar yang digunakan secara singkat dan jelas, dimana materi yang disampaikan merupakan acuan dari perancangan sistem. Selain itu, materi-materi tersebut juga berhubungan dengan proses perancangan antarmuka untuk OCU Rhino Robot. Pembahasa dimulai dari penggunaan prosesor, bahasa pemrograman, dan IDE yang digunakan, hingga metode-metode pendukung.

BAB III PERANCANGAN ANTARMUKA OCU RHINO ROBOT MENGGUNAKAN LATTEPANDA

Bagian perancangan sistem ini membahas tentang konsep dasar dari Tugas Akhir yang dikerjakan, yaitu perancangan program untuk menjalankan fungsi antarmuka dari panel *input* OCU Rhino Robot. Dimana penyusunan antarmuka tersebut dilakukan pada LattePanda menggunakan bahasa pemrograman C# yang ada pada Visual Studio.

BAB IV HASIL PENGUJIAN DAN ANALISIS

Pada pembahasan bab pengujian sistem, dimulai dari prosedur pengujian hingga hasil yang didapatkan. Selain itu juga dilakukan analisis hasil dari pengujian.

BAB V PENUTUP

Pembahasan terakhir laporan ini berisi tentang kesimpulan yang didapatkan dari hasil perancangan antarmuka OCU Rhino Robot.

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

BAB II
ANTARMUKA OCU RHINO ROBOT

Rhino Robot merupakan sebuah robot yang dikembangkan dari hasil adaptasi Robot EOD, berfungsi untuk menggantikan manusia dalam menangani benda-benda berbahaya. Untuk mengoperasikan robot ini, dibutuhkan sebuah perangkat kendali berama *Operator Control Unit* (OCU). Secara umum, perangkat ini tersusun dari sejumlah tombol-tombol yang berfungsi sebagai pilihan menu kendali. Untuk membuat sebuah OCU, ada beberapa tahap yang harus dilakukan. Diantaranya terdapat perancangan *hardware, software,* GUI, dan *power distribution,* seperti bagan pada Lampiran 1.

Penelitian ini berfokuskan pada perancangan *software* dari OCU Rhino Robot. Fungsi dari perancangan ini diantaranya yaitu untuk menjadikan tombol-tombol yang ada pada panel *input* dapat beroperasi sesuai perannya masing-masing. Penyusunan dilakukan pada IDE bernama Visual Studio menggunakan bahasa pemrograman C#. Mikrokontroler yang digunakan untuk menjadikannya dapat berkoordinasi dengan perancangan *hardware* adalah Arduino Leonardo yang terdapat pada *development board* LattePanda.

2.1 Robot EOD

Dalam dekade terakhir, sistem robot telah dikembangkan untuk misi *Explosive Ordnance Disposal* (EOD). Sebuah misi yang bertujuan untuk menggantikan peran manusia dalam menangani benda-benda berbahaya. Memungkinkan robot untuk melakukan misi yang sebelumnya hanya dilakukan oleh manusia menggunakan pakaian khusus ledakan [2]. Hasil pengembangan robot pada misi ini biasa dikenal dengan nama Robot EOD.

Secara general, Robot EOD yaitu robot yang dapat menggantikan manusia untuk mengintai, memindahkan, dan menangani peledak atau benda berbahaya lainnya secara langsung [1]. Keuntungan utama menggunakan robot ini adalah mengurangi risiko terjadinya cedera pada manusia. Saat ini, robot telah mampu melintasi berbagai medan, mengumpulkan dan menghancurkan bahan peledak tertentu, serta memberikan kemampuan pengintaian. Salah satu hasil pengembangan dapat dilihat pada Gambar 2.1 [4].

Gambar 2.1 Contoh Hasil Pengembangan Robot EOD

Dalam menjalankan perannya, robot ini memanfaatkan bagian penyusunnya, seperti *arm, chassis, flipper,* dan *gripper. Arm* (lengan robot) berfungsi untuk mengambil objek yang menjadi sasaran melalui subbagian bernama *gripper* (penggenggam). Sedangkan *chasis* berfungsi sebagai kaki dari robot, menjadikannya dapat bergerak dari satu tempat ke tempat lainnya. Selain itu, pada *chasis* juga dilengkapi dengan *flipper* yang berfungsi untuk membantu pergerakan robot ketika menaiki maupun menuruni tangga. Untuk dapat mengoperasikan robot ini, dibutuhkan sebuah perangkat yang mampu mengirimkan perintah kendali.

Robot EOD yang dikembangkan oleh PT Bhimasena Research and Development diberi nama Rhino Robot, dengan bagian penyusun seperti Robot EOD pada umumnya. Perangkat kendali yang digunakan untuk mengirimkan perintah ke robot ini bernama *Operator Control Unit* (OCU).

2.2 *Operator Control Unit* (OCU)

Pada praktiknya, robot EOD masih membutuhkan perhatian dan kendali secara langsung dan konstan dari pengguna. Hal tersebut dapat diatasi dengan menggunakan sebuah perangkat kendali bernama *Operator Control Unit* (OCU) [5]. Gambar 2.2 merupakan salah satu contohnya.

OCU biasanya dilengkapi dengan sebuah pengendali berupa *joystick* dan sejumlah tombol, yang berfungsi untuk menggerakkan robot sesuai dengan yang diinginkan [6]. Perancangan eksternal (panel *input*) dari OCU dilakukan dengan menggunakan prinsip kerja dari *user interface* [7].

Gambar 2.2 Contoh dari Operator Control Unit

2.3 Antarmuka

Sebuah titik, wilayah, atau permukaan dimana dua zat atau benda berbeda bertemu, digunakan secara metafora sebagai pembatas antar benda tersebut, biasa disebut dengan antarmuka (*interface*). Bentuk kerja dari antarmuka yaitu menghubungkan dua atau lebih benda pada suatu titik atau batasan yang terbagi, atau menyiapkan kedua benda untuk tujuan tertentu [8].

Dalam disiplin ilmu komputer, antarmuka mempelajari teknik-teknik menghubungkan komputer dengan peralatan elektronika lainnya. Dalam hubungannya dengan perangkat lunak, antarmuka dapat diartikan sebagai sistem operasi yang digunakan untuk menghubungkan antar perangkat mikroprosesor agar dapat berkomunikasi dengan pengguna. Sedangkan pada konteks perangkat keras, antarmuka berarti komponen elektronika yang menghubungkan atau mengkomunikasikan prosesor dengan komponen atau perangkat lain dalam suatu sistem.

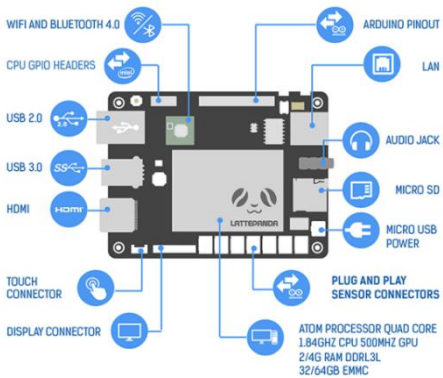
Antarmuka pengguna (*user interface*) merupakan bentuk tampilan operasi grafis yang berhubungan langsung dengan pengguna (*user*) [9]. Memiliki fungsi untuk menghubungkan pengguna dengan sistem, untuk membantu mengarahkan alur penelusuran masalah hingga sistem dapat digunakan.

2.4 LattePanda

Modul pengembangan pertama yang dapat menjalankan versi lengkap dari Windows 10 bernama LattePanda, memiliki bentuk fisik seperti pada Gambar 2.3. Modul ini beroperasi secara *turbocharged*

dengan sebuah prosesor *Intel Quad Core 1.8* GHz, yang dilengkapi memori sebesar 32-64 GB dan 2-4 GB RAM. Didukung dengan *co-processor* Arduino yang bertajuk *plug and play* serta pin GPIO (*General-Purpose Input/Output*), menjadikannya mampu mengakses 5V sensor dan aktuator standar.

LattePanda memiliki konektivitas yang baik dengan tiga buah *port* USB yang terdiri dari satu buah USB versi 3.0 dan dua buah USB versi 2.0, serta telah terintegrasi dengan WiFi dan Bluetooth versi 4.0. *Port* USB tersebut dapat digunakna untuk menghubungkan beberapa perangkat, seperti *flash drive, mouse* dan *keyboard*, serta *webcam*. Selain itu, terdapat soket kartu SD yang mendukung penyimpanan ekstra dari kartu mini SD. LattePanda juga dapat dihubungkan dengan perangkat pengeras suara melalui *jack audio* 3.5 mm [1].



Gambar 2.3 LattePanda

Dalam pengaksesan Arduino GPIO, diperlukan *library* LattePanda.Firmata yang merupakan *open-source library* dari Firmata. *Library* ini disediakan oleh *development board* LattePanda sesuai dengan aplikasi Windows yang telah dikembangkan pada Visual Studio [10]. Fitur-fitur yang dapat diakses berdasarkan *library* ini diantaranya:

- Membaca dan menulis pada pin digital.
- Membaca input pin analog.
- Mengendalikan motor servo.
- Mengirim dan menerima data dari *device* melalui terminal I2C.

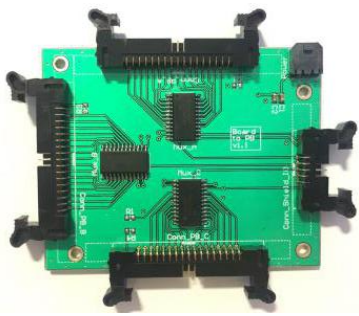
Firmata merupakan sebuah protokol yang digunakan untuk melakukan komunikasi dengan sebuah mikrokontroler melalui software pada PC. Protokol tersebut dapat diimplementasikan di firmware pada sejumlah arsitektur dari mikrokontroler sebaik software yang sepaket dengan PC. Protokol firmata secara teoritis dapat diimplementasikan pada platform mikrokontroler apa saja. Hingga saat ini, implementasi paling lengkap yaitu pada Arduino.

2.5 *Board* OCU

Untuk menjadikan OCU beroperasi sesuai dengan keinginan pengguna, digunakan beberapa *board* penyusun yang bekerja sesuai kebutuhan. Dimana *board-board* tersebut dapat menjadikan LattePanda mampu memproses sejumlah 45 tombol dan sejumlah LED. *Board* yang digunakan diantaranya *board push button, board* LED, dan *board* ATMega.

2.5.1 *Board Push Button*

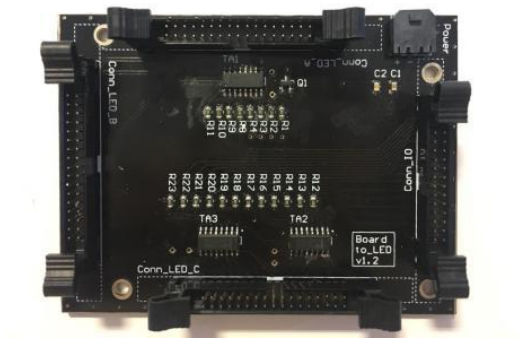
Pengaplikasian *board push button* ditujukan untuk menangani penggunaan pin mikrokontroler. Hal ini dikarenakan jumlah input tombol yang digunakan sebanyak 45 buah. Cara kerja yang dilakukan oleh *board* ini sesuai dengan komponen utama penyusunnya, yaitu multiplexer. IC yang digunakan pada *board* ini yaitu CD4067B [11], memiliki kemampuan menampung 16 jalur data *input*. Secara keseluruhan, *board* ini dapat digunakan untuk 48 data *input*. Cara menyambungkan *board* ini dengan LattePanda maupun masing-masing tombol digunakan beberapa konektor, yang tampak seperti Gambar 2.4.



Gambar 2.4 Bentuk Fisik dari *Board Push Button*

2.5.2 *Board* LED

Dalam perancangan panel *input* OCU, *board* LED digunakan untuk menangani *input* LED yang mengharuskannya menggunakan pin mikrokontroler dalam jumlah sedikit. *Board* ini bekerja dengan memanfaatkan fungsi dari rangkaian *switch transistor* yang sudah disediakan pada *board*, dimana menjadikan 1 *input* data yang diterima dapat digunakan untuk sejumlah LED. Berfungsi untuk menghubungkan LattePanda dengan LED yang berperan sebagai indikator pada panel *input,* memiliki bentuk fisik seperti pada Gambar 2.5.

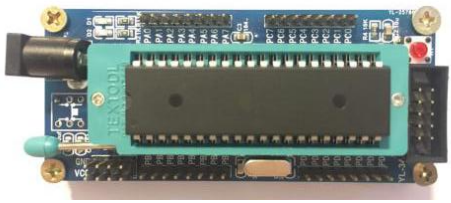


Gambar 2.5 Bentuk Fisik dari *Board* LED

2.5.3 *Board* ATMega16

Pada perancangan panel *input* OCU, *board* ATMega16 berfungsi untuk menjalankan komunikasi *Universal Synchronous Asynchronous Reciever Transmiter* (USART) antara LattePanda dengan LED. Komunikasi ini bertujuan untuk mengirimkan informasi dari keadaan tombol pada panel *input* yang sudah diolah menuju LED indikator. *Board* yang digunakan tampak seperti pada Gambar 2.6.

USART harus diinisialisasi terlebih dahulu sebelum komunikasi dapat berlangsung. Prosesnya terdiri dari pengaturan *baud rate* dan *frame format*, serta pengaktifan pengirim atau penerima. Untuk *interrupt* menjalankan operasi USART, global *interrupt flag* (penanda) sebaiknya dibersihkan dan *interrupt global* dinonaktifkan (*disable*) ketika inisialisasi dilakukan.

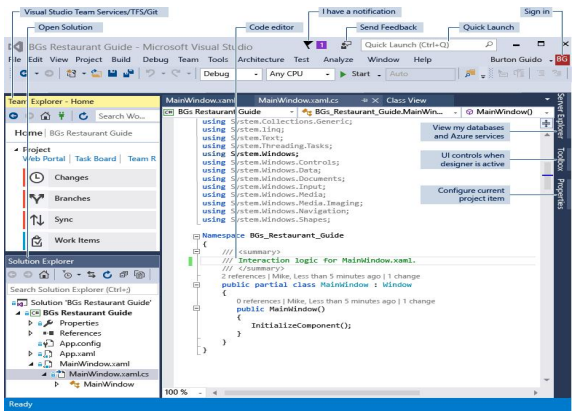


Gambar 2.6 Bentuk Fisik dari *Board* ATMega16

2.6 Visual Studio

Microsoft Visual Studio merupakan *Integrated Development Environment* (IDE) dengan fitur lengkap untuk Android, iOS, Windows, web, dan cloud, dengan tampilan layar utama seperti pada Gambar 2.7. Visual Studio dapat digunakan untuk membuat:

- Aplikasi dan permainan yang tidak hanya dapat berjalan pada Windows, tapi juga pada Android dan iOS.
- *Websites* dan *web services* yang bekerja diatas ASP.NET, JQuery, AngularJS, dan beberapa kerangka kerja lainnya.
- Aplikasi dan perangkat seperti Azure, Office, Sharepoint, Hololens, Kinect, dan Internet of Things.
- Permainan dan aplikasi grafis intensif untuk berbagai perangkat Windows, seperti Xbox dan DirectX.



Gambar 2.7 Tampilan Visual Studio IDE

Visual Studio secara *default* dapat digunakan untuk C#, C, C++, JavaScript, F#, dan Visual Basic yang bekerja dan terintegrasi secara baik dengan aplikasi pihak ketiga seperti Unity melalui ekstensi Visual Studio Tools untuk Unity dan Apache Cordova melalui Visual Studio Tools untuk Apache Cordova [12].

2.7 Pemrograman C#

Framework.NET adalah suatu komponen Windows yang terintegrasi. Dibuat dengan tujuan untuk mendukung pengembangan berbagai macam jenis aplikasi. *Framework* ini juga dapat mejalankan berbagai macam aplikasi generasi mendatang termasuk pengembangan aplikasi *web services* XML. Sebagai salah satu sarana untuk dapat memenuhi tujuan tersebut, maka dibuatlah berbagai macam bahasa pemrograman yang dapat digunakan dan dapat berjalan di atas platform Framework .NET. Bahasa-bahasa tersebut yaitu C#, VB.NET, J#, Perl.NET dan lain-lain [13].

C# (*c-sharp*) adalah bahasa pemrograman baru yang diciptakan oleh Microsoft seiring dengan pembuatan Framework .NET. *Chief Architect* dalam pembuatan C# adalah Anders Hejlsberg yang sebelumnya berperan dalam pembuatan Borland Delphi dan Turbo Pascal. Bahasa pemrograman ini berhasil distandarisasi oleh ECMA pada Desember 2001 dengan *compiler* C# buatan Microsoft dan *compiler* dari proyek Mono [14].

Pembuatan C# didasarkan kepada bahasa C dan C++, maka aspek-aspek seperti *statements, expression, operators*, dan sebagainya dapat ditemui pada C#, tetapi dengan berbagai perbaikan yang membuat bahasanya menjadi lebih sederhana. Didukung dengan penghilangan beberapa hal yang bersifat kompleks yang terdapat dalam Java dan C++, seperti *macro, templates, multiple inheritance* dan *virtual base classes.* Dikarenakan bekerja diatas *framework.NET*, maka C# bisa digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis Windows (*desktop*), web, dan *web services*.

Program C# diorganisasikan oleh *namespace,* dapat digunakan sebagai sistem organisasi internal dalam program. Selain itu, juga dapat diorganisasikan secara eksternal, yaitu menjadikannya dapat diakses secara publik menggunakan kata kunci *using*. Menurut ECMA, *class* adalah struktur data yang berisi data (*constant* dan *field*), fungsi

(*method, property, event, indexer, operator, constructor, destructor,* dan *static constructor*), serta class dalam class. Dalam C#, semua program dibangun diatas *class*.

Satu bentuk iterasi khusus pada C# yang tidak berasal dari C adalah *foreach*. Bentuk ini merupakan adaptasi dari *for each* yang ada pada Visual Basic. *Statement foreach* digunakan untuk menelusuri suatu *collection*, misalnya *array* seperti pada Gambar 2.8.

```
using System;
namespace org.gotdotnet.otak
{
	class ContohForeach
	{
		public static void Main()
		{
			string[] drives = System.Environment.GetLogicalDrives();
			foreach (string drive in drives)
			{
				Console.WriteLine("drive "+"drive");
			}
			Console.ReadLine();
		}
	}
}
```

Gambar 2.8 *Listing Program Foreach*

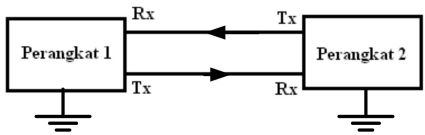
Berdasar *listing program* pada Gambar 2.8, dapat diketahui bahwa variabel *drive* secara implisit bertipe *read only*. Hal ini menandakan bila ada *statement* yang berusaha mengubah nilai dari variabel, maka *compiler* C# akan menampilkan pesan kesalahan sewaktu melakukan kompilasi.

2.8 Komunikasi Serial

Suatu proses pengiriman data secara serial (data dikirim satu persatu secara berurutan) disebut dengan komunikasi serial. Hal tersebut menjadikan komunikasi serial lebih lambat daripada komunikasi paralel. Agar komunikasi serial dapat bekerja dengan baik, data *byte* harus diubah ke dalam bit-bit serial [15].

Komunikasi serial dapat menghubungkan mikrokontroler dengan peralatan lainnya seperti yang ada pada Gambar 2.9. *Port* serial pada mikrokontroler terdiri atas dua pin, yaitu RxD dan TxD. RxD

berfungsi untuk menerima data dari komputer atau peralatan lainnya, sedangkan TxD berfungsi untuk mengirim data dari mikrokontroler ke komputer atau peralatan lainnya.



Gambar 2.9 Komunikasi Serial Mode Asinkron

Komunikasi serial mengenal dua buah metode, yaitu sinkron dan asinkron. Metode sinkron mengirimkan beberapa *byte* atau karakter (biasa disebut blok data atau *frame*) sebelum meminta konfirmasi apakah data sudah diterima dengan baik atau tidak. Sementara metode asinkron mengirim satu *byte* setiap pengiriman, dengan tidak dibutuhkannya konfirmasi penerimaan data. Proses pengiriman data serial tersebut dapat dilakukan menggunakan UART (*Universal Asynchronous Reciever Transmiter*) dan USART (*Universal Synchronous Asynchronous Reciever Transmiter*).

Universal Asynchronous Receiver-Transmitter (UART) merupakan sebuah protokol komunikasi yang umum digunakan dalam pengiriman data serial antara *device* satu dengan yang lainnya. Dalam pengiriman data, *clock* antara pengirim dan penerima harus sama, karena paket data dikirim tiap bit mengandalkan *clock* tersebut. Inilah salah satu keuntungan model *asynchronous* dalam pengiriman data, karena dengan hanya satu kabel transmisi maka data dapat dikirimkan. Namun kelemahan model ini yaitu dalam hal kecepatan dan jarak transmisi. Karena semakin cepat dan jauh, membuat paket-paket bit data menjadi terdistorsi sehingga data yang dikirim atau diterima bisa mengalami eror.

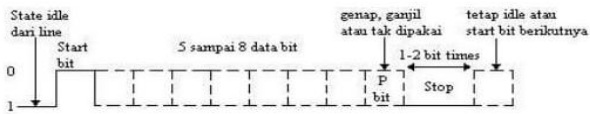
Asynchronous memungkinkan transmisi mengirim data tanpa sang pengirim harus mengirimkan sinyal *clock* ke penerima. Untuk melakukan sinkronisasi, pengirim dan penerima harus mengatur parameter waktu di awal (bit *start*) dan bit khusus ditambahkan untuk setiap data yang digunakan.

Universal Synchronous-Asynchronous Receiver Transmitter (USART) merupakan komunikasi yang memiliki fleksibilitas tinggi, yang dapat digunakan untuk melakukan transfer data baik antar

mikrokontroler maupun dengan modul-modul eksternal termasuk PC yang memiliki fitur UART.

Sebuah USART bekerja dengan menerima data paralel dari *Central Processing Unit* (CPU), mengubahnya menjadi data serial untuk ditransmisikan ke *port* serial. Demikian pula, menerima data serial dari *port* serial, mengkonversi ke paralel data dan mengirimkannya ke CPU. USART mirip UART, karena masing-masing mendukung dan memberikan komunikasi serial. Namun, UART hanya mendukung komunikasi serial *asynchronous*. Komunikasi ini memungkinkan transmisi data baik secara *syncrhronous* maupun *asyncrhronous*, sehingga dengan memiliki USART pasti kompatibel dengan UART.

Komunikasi serial asinkron berbeda dengan serial sinkron, yaitu tidak adanya sinyal *clock* yang menyinkronkan komunikasi data, hal yang digunakan untuk sinkronisasi adalah protokol komunikasi data seperti pada Gambar 2.10 dan kecepatan transfer yang dinyatakan dengan *bit per second* (bps) atau dikenal dengan nama *baud rate*.



Gambar 2.10 Protokol Komunikasi Serial Asinkron

Berdasarkan Gambar 2.10, diketahui bahwa protokol komunikasi serial asinkron terdiri dari beberapa elemen penyusun yang terdiri dari:

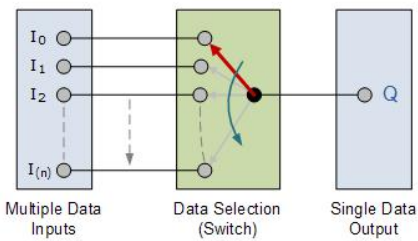
1. *Idle state* : adalah kondisi tidak terjadinya komunikasi data dan jalur data berlogika 1 secara terus menerus (*marking*).
2. Bit *start* : selalu bernilai 0, ketika komunkasi UART (serial asinkron) akan diberikan, terlebih dahulu dimulai dengan pemberian bit *start*. Fungsinya sebagai pemicu (tanda) kepada penerima (RxD) bahwa akan ada data yang diberikan oleh pemancar (TxD) dan juga akan memicu *clock* pada *reciever* sehingga disinkronkan dengan *clock* pada *transmitter*.
3. Bit data : adalah data yang akan dikirimkan secara UART dimulai dari LSB (bit ke 0) hingga MSB (bit terakhir). Penentuan banyaknya bit harus sama antara pemancar dengan penerima.

4. *Parity* : berfungsi sebagai pengecekan kesalahan data yang dikirim. *Parity* bisa bernilai ganjil, genap, dan *parity* tidak digunakan.
5. Bit *stop* : selalu bernilai 1, berfungsi sebagai akhir dari komunikasi data dan kemudian masuk pada *idle state*. Pengiriman data selanjutnya dapat dilakukan setelah bit *stop* diberikan.

2.9 *Multiplexer* (MUX)

Multiplexing merupakan istilah umum yang digunakan untuk mendeskripsikan operasi pengiriman satu maupun lebih dari sinyal analog dan digital. Pengiriman dilakukan melalui jalur transmisi umum dalam waktu dan kecepatan yang berbeda, sedangkan perangkat yang digunakan untuk menjalankan fungsi tersebut disebut dengan *multiplexer*.

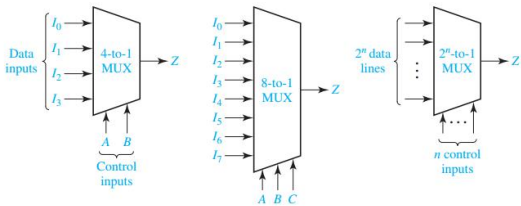
MUX atau MPX yang merupakan singkatan dari *multiplexer* adalah rangkaian logika kombinasional yang didesain untuk mengatur satu dari beberapa jalur input untuk menjadi sebuah jalur output hasil dari penerapan logika kontrol. *Multiplexer* beroperasi seperti beberapa saklar yang bergerak berputar sangat cepat untuk menghubungkan atau mengatur beberapa jalur input yang bernama '*channel'* satu per satu menuju output seperti yang ada para Gambar 2.11 [16].



Gambar 2.11 Ilustrasi Cara Kerja MUX

MUX memiliki sekelompok data *input* dan sekelompok *input* kendali (*selector*) yang digunakan untuk memilih salah satu dari data *input* dan ditransmisikan ke saluran *output*. Gambar 2.12 menunjukkan tipe-tipe MUX, diantaranya MUX 4-ke-1, MUX 8-ke-1, dan MUX 2^n-ke-1. MUX 4-ke-1 bekerja seperti saklar empat posisi

yang hanya mentransmisikan salah satu dari empat input ke output. Dua *selector* (A dan B) dibutuhkan untuk memilih satu dari keempat input tersebut. Jika *selector* AB=00, output yang dihasilkan adalah I_0; dengan serupa, *selector* dengan nilai 01, 10, dan 11 memberikan output I1, I2, dan I3 secara berurutan.



Gambar 2.12 Tipe-tipe MUX

Serupa dengan sebelumnya, MUX 8-ke-1 memilih satu dari kedelapan data *input* menggunakan tiga *selector*. Saat *selector* ABC bernilai 011, output yang dihasilkan adalah I3, dan output lainnya didapatkan dengan cara yang sama. Secara umum, MUX dengan n *selector* dapat digunakan untuk memilih salah satu dari 2^n input data [17].

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

BAB III
PERANCANGAN ANTARMUKA OCU RHINO ROBOT MENGGUNAKAN LATTEPANDA

Proses yang harus dilalui pada perancangan OCU dari Rhino Robot ini dilakukan secara bertahap. Diantaranya dilakukan perancangan panel OCU, yang terdiri dari perancangan *hardware* dan *software*, bertujuan untuk dapat mengoperasikan tombol-tombol pada panel *input*. Selain itu juga ada optimalisasi GUI dari versi sebelumnya, berfungsi untuk mengurangi jumlah halaman yang tidak digunakan. Serta adanya perancangan *power distribution* untuk memberikan *supply* tegangan pada masing-masing komponen yang digunakan dalam proses perancangan OCU.

Penelitian ini fokus terhadap perancangan *software* dari panel *input* OCU Rhino Robot. Hal yang dilakukan antaranya yaitu menyusun program-program antarmuka yang dapat menjalankan fungsi dari masing-masing tombol pada panel *input* dan LED indikatornya. Komponen yang dibutuhkan dalam perancangan ini diantaranya yaitu sebuah IDE yang dapat beroperasi pada *co-processor* Arduino dari *development board* LattePanda. *Software* tersebut yaitu Visual Studio dengan bahasa pemrograman C#. Pemilihan LattePanda sebagai prosesornya didasarkan pada bentuk dari OCU versi lama yang memiliki tampilan seperti laptop pada umumnya. LattePanda yang digunakan beroperasi pada Windows 10, yang merupakan sistem operasi dari laptop pada umumnya. Selain itu, prosesor pendukung dari LattePanda yaitu Arduino, yang merupakan mikrokontroler yang umum digunakan oleh pada *developer*.

Perancangan antarmuka yang disusun meliputi pembuatan program untuk menjalankan fungsi operasi dari 45 tombol dan sejumlah LED yang ada pada panel *input*.

3.1 Rancangan Sistem

Antarmuka dari OCU Rhino Robot dirancang dengan tujuan akhir mampu menjalankan semua fungsi dari masing-masing tombol pada panel *input.* Dimana tombol-tombol tersebut berjumlah 45 buah. Rincian fungsi dari tombol-tombol tersebut diantaranya terdapat pada Tabel 3.1.

Tabel 3.1 Rincian Fungsi Antarmuka Tombol-tombol Panel *Input*

No.	Nama Tombol	Keterangan
1.	Tombol menu *toggle*	Berfungsi untuk menjalankan fungsi dari menu *toggle* yang telah diatur, yaitu dapat memunculkan menu *pop-up* yang tersedia pada halaman operasi utama
2.	Tombol menu *enter*	Berfungsi untuk menjalankan fungsi dari menu *enter* yang telah diatur, yaitu menjadikan halaman tampilan berganti sesuai dengan menu yang dipilih
3.	Tombol menu *back*	Berfungsi untuk menjalankan fungsi dari menu *back* yang telah diatur, yaitu menjadikan halaman tampilan kembali ke tampilan yang sebelumnya
4.	Tombol menu *down*	Berfungsi untuk menjalankan fungsi dari menu *direction down* yang telah diatur, yaitu dapat memindahkan kursor ke menu yang ada di bawahnya
5.	Tombol menu *up*	Berfungsi untuk menjalankan fungsi dari menu *direcction up* yang telah diatur, yaitu dapat memindahkan kursor ke menu yang ada di atasnya
6.	Tombol menu *left*	Berfungsi untuk menjalankan fungsi dari menu *direcction left* yang telah diatur, yaitu dapat memindahkan kursor ke menu yang ada di kirinya
7.	Tombol menu *right*	Berfungsi untuk menjalankan fungsi dari menu *direcction right* yang telah diatur, yaitu dapat memindahkan kursor ke menu yang ada di kanannya
8.	Tombol menu pose *preset* 1	Berfungsi untuk menjalankan pose preset1 yang telah diatur
9.	Tombol menu pose *preset* 2	Berfungsi untuk menjalankan pose preset2 yang telah diatur
10.	Tombol menu pose *preset* 3	Berfungsi untuk menjalankan pose preset3 yang telah diatur
11.	Tombol menu pose *preset* 4	Berfungsi untuk menjalankan pose preset4 yang telah diatur
12.	Tombol menu pose *preset* 5	Berfungsi untuk menjalankan pose preset5 yang telah diatur
13.	Tombol menu pose *preset* 6	Berfungsi untuk menjalankan pose preset6 yang telah diatur
14.	Tombol menu *speed control*	Berfungsi untuk menjalankan menu yang dapat mengatur kecepatan gerak dari robot, mulai dari *slow*, normal, hingga *fast*

Tabel 3.1 Rincian Fungsi Antarmuka Tombol-tombol Panel *Input* (Lanjutan)

No.	Nama Tombol	Keterangan
15.	Tombol *shoulder counter clockwise*	Berfungsi untuk menjalankan menu operasi *shoulder* yang berputar berlawanan arah dengan jarum jam
16.	Tombol *shoulder clockwise*	Berfungsi untuk menjalankan menu operasi *shoulder* yang berputar searah dengan jarum jam
17.	Tombol *elbow counter clockwise*	Berfungsi untuk menjalankan menu operasi *elbow* yang berputar berlawanan arah dengan jarum jam
18.	Tombol *elbow clockwise*	Berfungsi untuk menjalankan menu operasi *elbow* yang berputar searah dengan jarum jam
19.	Tombol *wrist counter clockwise*	Berfungsi untuk menjalankan menu operasi *wrist* yang berputar berlawanan arah dengan jarum jam
20.	Tombol *wrist clockwise*	Berfungsi untuk menjalankan menu operasi *wrist* yang berputar searah dengan jarum jam
21.	Tombol *camera pan counter clockwise*	Berfungsi untuk menjalankan menu operasi *camera pan* yang berputar berlawanan arah dengan jarum jam
22.	Tombol *camera pan clockwise*	Berfungsi untuk menjalankan menu operasi *camera pan* yang berputar searah dengan jarum jam
23.	Tombol *camera tilt counter clockwise*	Berfungsi untuk menjalankan menu operasi *camera tilt* yang berputar berlawanan arah dengan jarum jam
24.	Tombol *camera tilt clockwise*	Berfungsi untuk menjalankan menu operasi *camera tilt* yang berputar searah dengan jarum jam
25.	Tombol *turntable counter clockwise*	Berfungsi untuk menjalankan menu operasi *turntable* yang berputar berlawanan arah dengan jarum jam
26.	Tombol *turntable clockwise*	Berfungsi untuk menjalankan menu operasi *turntable* yang berputar searah dengan jarum jam
27.	Tombol *gripper rotation counter clockwise*	Berfungsi untuk menjalankan menu operasi *gripper rotation* yang berputar berlawanan arah dengan jarum jam
28.	Tombol *gripper rotation clockwise*	Berfungsi untuk menjalankan menu operasi *gripper rotation* yang berputar searah dengan jarum jam
29.	Tombol *gripper grasp open*	Berfungsi untuk menjalankan menu operasi *gripper grasp* membuka
30.	Tombol *gripper grasp close*	Berfungsi untuk menjalankan menu operasi *gripper grasp* menutup

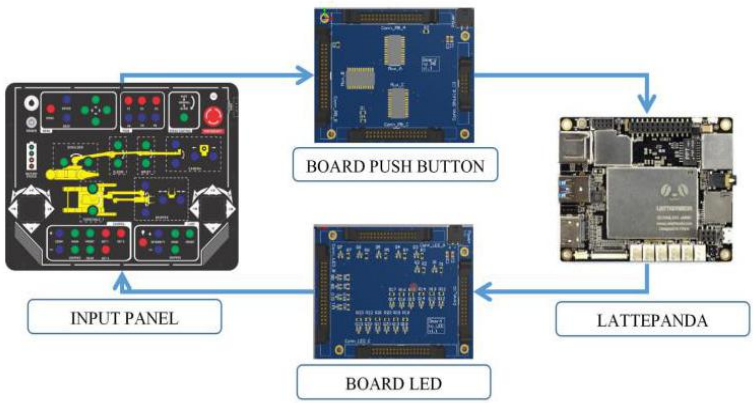
Tabel 3.1 Rincian Fungsi Antarmuka Tombol-tombol Panel *Input* (Lanjutan)

No.	Nama Tombol	Keterangan
31.	Tombol aksesoris *camera zoom in*	Berfungsi untuk menjalankan menu aksesoris *camera zoom in,* yaitu untuk menjadikan kamera dapat memperbesar jangkauan
32.	Tombol aksesoris *camera zoom out*	Berfungsi untuk menjalankan menu aksesoris *camera zoom out,* yaitu untuk menjadikan kamera dapat memperkecil jangkauan
33.	Tombol aksesoris *camera main*	Berfungsi untuk menjalankan menu aksesoris *camera main,* yaitu untuk menjadikan kamera utama menyala
34.	Tombol aksesoris *camera gripper*	Berfungsi untuk menjalankan menu aksesoris *camera gripper,* yaitu untuk menjadikan kamera penggenggam menyala
35.	Tombol aksesoris *camera front*	Berfungsi untuk menjalankan menu aksesoris *camera front,* yaitu untuk menjadikan kamera depan menyala
36.	Tombol aksesoris *camera rear*	Berfungsi untuk menjalankan menu aksesoris *camera rear,* yaitu untuk menjadikan kamera belakang menyala
37.	Tombol aksesoris *camera set* 1	Berfungsi untuk menjalankan menu aksesoris *camera set* 1, yaitu berisi pengaturan jumlah kamera yang digunakan
38.	Tombol aksesoris *camera set* 2	Berfungsi untuk menjalankan menu aksesoris *camera set* 2, yaitu berisi pengaturan jumlah kamera yang digunakan
39.	Tombol aksesoris *camera set* 3	Berfungsi untuk menjalankan menu aksesoris *camera set* 3, yaitu berisi pengaturan jumlah kamera yang digunakan
40.	Tombol *lamp speaker toggle*	Berfungsi untuk menjalankan menu aksesoris *lamp speaker toggle*, yaitu memunculkan bagian untuk mengatur kecerahan lampu
41.	Tombol *lamp speaker increase*	Berfungsi untuk menjalankan menu aksesoris *lamp speaker increase*, yaitu menjadikan lampu semakin terang
42.	Tombol *lamp speaker decrease*	Berfungsi untuk menjalankan menu aksesoris *lamp speaker decrease*, yaitu menjadikan lampu semakin redup
43.	Tombol *lamp main*	Berfungsi untuk menjalankan menu aksesoris *lamp main*, yaitu menjadikan lampu utama dari robot menyala
44.	Tombol *lamp gripper*	Berfungsi untuk menjalankan menu aksesoris *lamp gripper,* yaitu menjadikan lampu penggenggam dari robot menyala
45.	Tombol *lamp front*	Berfungsi untuk menjalankan menu aksesoris *lamp front*, yaitu menjadikan lampu depan dari robot menyala

3.2 Diagram Alur Sistem

Antarmuka dari OCU Rhino Robot versi baru ini dirancang sedemikian rupa guna menjadikan tombol-tombol pada panel *input* dan LED indikator bekerja sesuai dengan yang diharapkan. Perancangan disusun seperti diagram alur pada Gambar 3.1, dimana tombol-tombol panel *input* yang dirancang pada *board push button* mengirimkan statusnya ke LattePanda untuk dieksekusi oleh program yang telah disusun. Informasi tersebut kemudian dikirimkan menggunakan *board* ATMega16 ke *board* LED untuk menyalakan LED indikatornya.

Gambar 3.1 Diagram Alur Sistem Antarmuka OCU

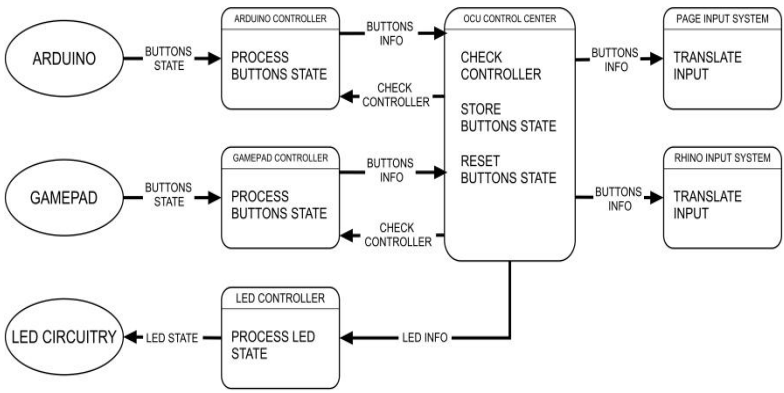


Gambar 3.2 Ilustrasi Perancangan Antarmuka OCU Rhino Robot

Berdasarkan diagram alur sistem pada Gambar 3.1, dapat diilustrasikan bahwa perancangan antarmuka OCU dengan menggunakan *development board* LattePanda ini tampak seperti pada Gambar 3.2. Panel *input* disambungkan dengan *board push button* yanga telah terhubung dengan masing-masing tombol pada panel *input.* Kemudian mengirimkan informasi dari keadaan tombol ke LattePanda untuk diolah melalui program yang telah disusun dan disalurkan ke *board* LED untuk mengirimkan informasi ke LED indikator.

3.3 Perancangan Program

Untuk dapat menjalankan fungsi antarmuka dari panel *input* OCU Rhino Robot, disusun sejumlah program pada sebuah IDE bernama Visual Studio. Penyusunan dilakukan dengan menggunakan bahasa pemrograman C#, penggunaan bahasa ini didasarkan pada sifatnya yang memudahkan penulis dalam melakukan penyusunan program. Selain itu, bahasa ini juga telah digunakan untuk menyusun fungsi antarmuka dari perangkat OCU versi sebelumnya. Program dirancang sedemikian rupa hingga dapat menjalankan fungsi antarmuka tombol-tombol pada panel *input* dan LED indikator. Yaitu menjadikan tombol dapat mengoperasikan sistem dari OCU, serta menjadikan LED sebagai indikator dari sejumlah tombol. Fungsi dari LED ini sendiri yaitu untuk memudahkan pengguna mengetahui tombol mana saja yang sedang digunakan.



Gambar 3.3 Diagram Alir Data Program Antarmuka

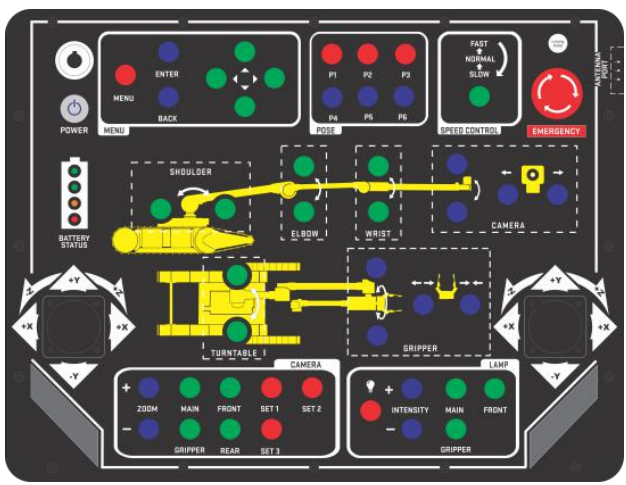
Berdasarkan Gambar 3.3 yang berisi diagram alir data dari program antarmuka OCU Rhino Robot, dapat diketahui bahwa data *input* diperoleh melalui dua perangkat kendali, yaitu Arduino dan *gamepad*. Data yang dikirim oleh kedua perangkat tersebut berupa status dari tombol-tombol yang kemudian diproses pada *control handler.* Pemrosesan ini dilakukan pada file program yang berbeda, fungsinya untuk memudahkan proses penyusunan program. Informasi-informasi hasil pemrosesan status tombol kemudian

ditampung pada *control center* sebelum diterjemahkan oleh *input system* untuk menjalankan fungsi antarmuka. Selain itu, *control center* juga berfungsi untuk mengirimkan informasi ke *file program* bernama *LEDController.cs* untuk menjalankan fungsi antarmuka dari LED indikator.

3.2.1 *Control Handler*

Program bagian *control handler* disusun berdasarkan perangkat kendali yang digunakan. Diantaranya terdapat *GamepadController.cs* dan *ArduinoController.cs*. Pada bagian ini, program menjalankan fungsinya untuk melakukan pemeriksaan koneksi dari masing-masing perangkat kendali. Setelah konektivitas perangkat diketahui, dilakukan pendeklarasian dari keadaan tombol. Pendeklarasian tersebut dilakukan dengan dibuatnya metode untuk melakukan pembaruan dari keadaan tombol dan mengembalikan nilai tombol pada keadaan *default.*

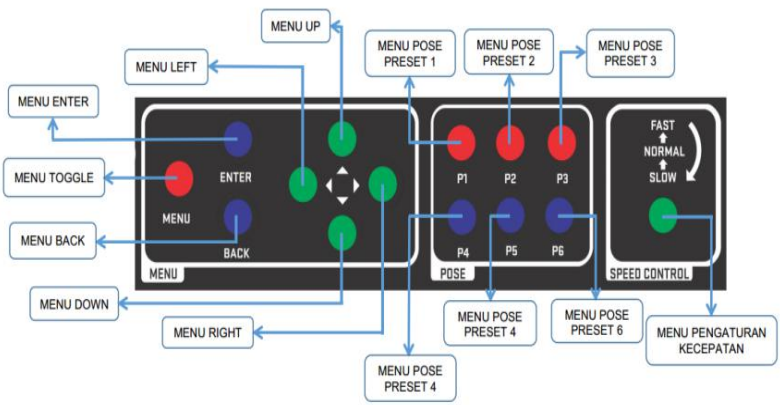
Penyusunan program *GamepadController.cs* ditujukan untuk perangkat kendali yang menggunakan *gamepad.* Program disusun dengan menggunakan sebuah *namespace* yang dapat menangani data-data yang dikirimkan, yaitu SlimDX.XInput. Sebuah *namespace* yang dapat mendeklarasikan keadaan dari kontroler Xbox 360.



Gambar 3.4 Panel *Input*

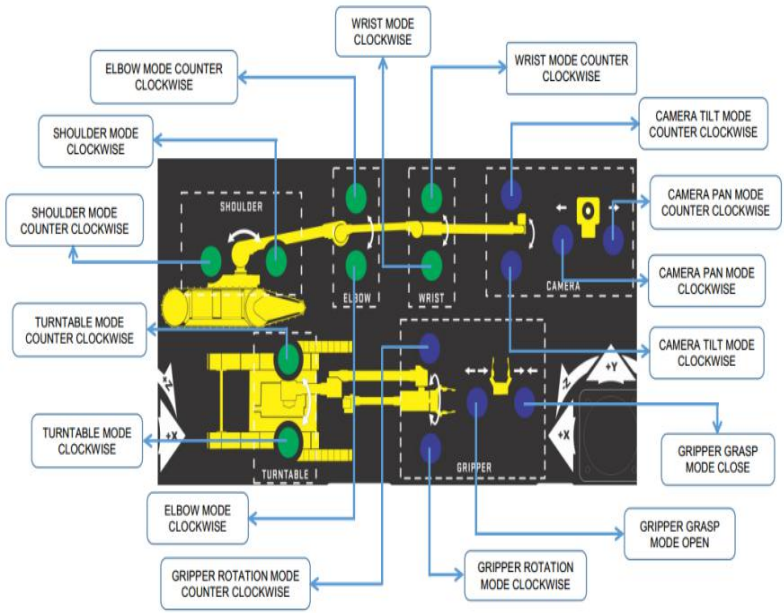
Perangkat kendali yang menggunakan Arduino (panel *input*), berfungsi untuk melakukan eksekusi dari informasi yang dikirim oleh tombol-tombol pada panel *input.* Fungsi ini diproses pada *file* program bernama *ArduinoController.cs*. Dikarenakan panel *input* melakukan pemrosesan data untuk 45 buah tombol, maka dibutuhkan sebuah penanganan khusus yang dapat mengakses sejumlah *input* tersebut dengan penggunaan pin LattePanda yang sedikit. Oleh karena itu, digunakan *board push button* dengan 3 buah MUX yang dapat menampung *input* lebih dari 1. Masing-masing MUX tersebut dapat berfungsi untuk memproses 16 data *input.* Pengelompokan tombol-tombol dilakukan berdasarkan letaknya pada panel yang memiliki tampilan seperti pada Gambar 3.4. Sesuai dengan penggunaan MUX sebagai metode untuk *handling input* tombol-tombol, maka program yang disusun juga harus mengimplementasikan cara kerja dari MUX. Dimana harus ada inisialisasi dari *selector* masing-masing MUX dan pendeklarasian untuk masing-masing keadaan tombol pada panel *input.*

MUX pertama digunakan untuk memproses data *input* dari kelompok tombol yang terletak di bagian atas panel. Pada kelompok ini MUX melakukan pemrosesan data untuk tombol-tombol menu, pose, dan *speed control*. Bagian ini terdiri dari 14 tombol, dimana 7 tombol merupakan tombol menu, 6 tombol untuk tombol pose, dan 1 tombol untuk *speed control.* Nama-nama dan letak dari tombol tersebut seperti yang ada pada Gambar 3.5.



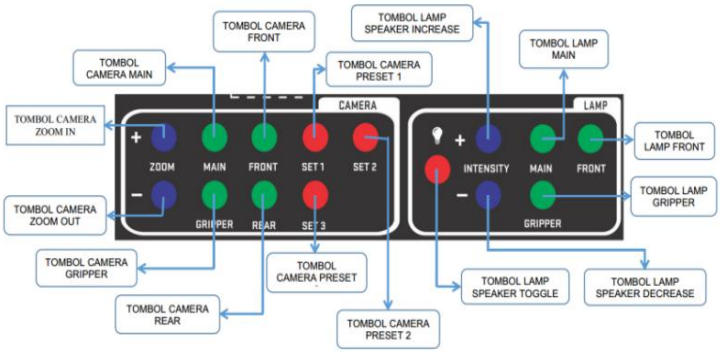
Gambar 3.5 Kelompok Tombol MUX1

Bagian tengah dari panel merupakan kelompok untuk tombol yang diakses oleh MUX kedua. Kelompok ini berisi tombol-tombol yang berfungsi mengirimkan perintah operasi untuk pergerakan robot. Diantaranya tombol untuk mengatur *turntable, shoulder, elbow, wrist, camera tilt, camera pan, gripper rotation,* dan *gripper grasp.* Tombol untuk masing-masing menu tersebut terdapat dua pilihan, yaitu untuk memutar searah jarum jam dan sebaliknya. Nama dan letak dari tombol-tombol pada kelompok ini diantaranya seperti yang ada pada Gambar 3.6.



Gambar 3.6 Kelompok Tombol MUX2

MUX terakhir berfungsi melakukan pemrosesan untuk kelompok tombol yang terletak pada panel bagian bawah. Tombol yang tergabung dalam kelompok ini merupakan tombol aksesoris dari robot, yaitu *camera* dan *lamp*. Untuk menu aksesoris *camera,* terdapat 9 tombol, dan 6 tombol untuk menu aksesoris *lamp.* Letak dan penamaan dari masing-masing tombol kelompok ini dapat dilihat pada Gambar 3.7



Gambar 3.7 Kelompok Tombol MUX3

Berdasarkan data nama tombol pada masing-masing kelompok MUX pada Gambar 3.5, 3.6, dan 3.7, terdapat beberapa tombol yang memiliki nama cukup panjang. Oleh karena itu, digunakan kode nama untuk tiap-tiap tombol dengan tujuan memudahkan proses penyusunan program. Daftar kode nama tombol-tombol tersebut dapat dilihat pada Tabel 3.2.

Tabel 3.2 Daftar Nama Tombol yang Digunakan pada Panel *Input*

Data	Multiplexer 1	Multiplexer 2	Multiplexer 3
0	Menu_Toggle	Turntable_CC	Camera_Zoom_In
1	Menu_Enter	Turntable_C	Camera_Zoom_Out
2	Menu_Back	Shoulder_CC	Camera_Main
3	Menu_Left	Shoulder_C	Camera_Gripper
4	Menu_Up	Elbow_CC	Camera_Front
5	Menu_Down	Elbow_C	Camera_Rear
6	Menu_Right	Wrist_CC	Camera_Preset1
7	Pose_Preset1	Wrist_C	Camera_Preset2
8	Pose_Preset2	Camera_Tilt_CC	Camera_Preset3
9	Pose_Preset3	Camera_Tilt_C	Lamp_Speaker_Toggle
10	Pose_Preset4	Camera_Pan_CC	Lamp_Speaker_Increase
11	Pose_Preset5	Camera_Pan_C	Lamp_Speaker_Decrease
12	Pose_Preset6	Gripper_Rotation_CC	Lamp_Main
13	Speed_Control	Gripper_Rotation_C	Lamp_Gripper
14	-	Gripper_Grasp_CC	Lamp_Front
15	-	Gripper_Grasp_C	-

Selain proses pengelompokan tombol-tombol pada panel yang perlu diperhatikan, *ArduinoController.cs* juga membutuhkan suatu fungsi yang dapat mengetahui keadaan dari *port* serial LattePanda. Hal ini dikarenakan pada *namespace* LattePanda.Firmata belum terdapat fungsi yang dapat menjalankan perintah tersebut. Oleh karena itu, dibuat suatu fungsi tambahan pada *file Arduino.cs,* yang merupakan *listing program* dari *namespace* LattePanda.Firmata. Fungsi tambahan tersebut kemudian diberi nama *IsConnected()*, seperti yang tampak pada Gambar 3.8. Fungsi ini berisi perintah untuk melakukan pengecekan apakah *port serial* dari LattePanda yang digunakan telah terhubung atau belum.

```
public bool IsConnected()
{
    if (_serialPort!=null)
    {
        return _serialPort.IsOpen;
    }
    else
    {
        return false;
    }
}
```

Gambar 3.8 *Listing Program* dari Fungsi IsConnected()

3.2.2 *Control Center*

Informasi yang telah diproses pada *control handler* kemudian ditampung pada *control center.* Selama informasi tersebut berada pada *control center,* program akan melakukan pemeriksaan terhadap masing-masing perangkat kendali. Fungsi dari perintah ini yaitu untuk mengetahui perangkat mana yang sedang aktif atau tengah digunakan. Setelahnya, program menyimpan informasi yang dikirim oleh *control handler.* Kemudian dilakukan pendeklarasian untuk mengubah nilai informasi yang berupa status dari tombol-tombol pada keadaan *default.*

Saat pemeriksaan perangkat kendali dilakukan, dibutuhkan sebuah metode untuk menangani kemungkinan adanya perangkat tambahan yang terhubung. Metode tersebut berisi perintah untuk mengetahui keadaan dari perangkat. *Listing program* yang digunakan untuk menjalankan metode ini seperti yang ada pada Gambar 3.9.

```
foreach (OCU_CONTROLLER item in controller)
{
    if (item != OCU_CONTROLLER.ARDUINO)
    {
        secondarycontroller.Device = item;
        break;
    }
}
if (controller.Contains(secondarycontroller.Device))
{
    if (!secondarycontroller.IsActive)
    {   secondarycontroller.IsActive = true; }
}
else
{
    if(secondarycontroller.IsActive)
    { secondarycontroller.IsActive = false; }
}
```

Gambar 3.9 Listing Program untuk Perangkat Kendali Tambahan

Untuk dapat mengakses *listing program* pada Gambar 3.9*,* diperlukan sebuah *class* tambahan bernama AuxiliaryControl. *Class* tersebut berisi pendeklarasian untuk keadaan-keadaan yang mungkin terjadi pada perangkat kendali tambahan. Mulai dari fungsi untuk mengetahui keadaan perangkat sedang aktif atau tidak hingga mode kendali yang digunakan oleh perangkat tersebut.

Selain melakukan proses dari data *input* yang diterima, *control center* juga memberikan perintah kepada *board* LED. Proses yang dijalankan disusun pada sebuah *file* program bernama *LEDController.cs*. Perintah-perintah yang dieksekusi dimulai dari pemeriksaan konektivitas *port* serial yang digunakan. Selanjutnya yang dilakukan yaitu melakukan pendeklarasian untuk metode pembaruan data sesuai dengan informasi terbaru yang dikirimkan oleh perangkat kendali, maupun metode untuk mengembalikan data pada keadaan *default.*

Pada proses pengiriman data, diperlukan penanganan untuk mengatur nilai dari tiap-tiap bit, atau bisa disebut dengan pengaturan keadaan dari masing-masing LED. Hal tersebut dilakukan karena masing-masing bit data yang dikirimkan membawa informasi atau perintah untuk masing-masing LED. Penanganan tersebut terdiri dari dua metode, diantaranya untuk mengatur nilai bit hingga bernilai satu (1) yang berarti pengaturan agar LED menyala dan mengembalikan

nilai bit menjadi nol (0) untuk mengatur LED agar keadaannya menjadi padam.

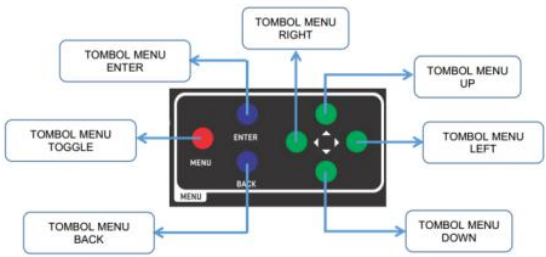
Proses pengaturan nilai bit hingga bernilai 1 digunakan operasi logika OR. Operasi dilakukan antara nilai data pembawa bit dengan variabel yang telah dideklarasikan. Variabel yang dideklarasikan untuk masing-masing bit diberi nama awalan *mask* yang kemudian diikut nama masing-masing bit. Pengembalian nilai bit menjadi 0, digunakan operasi AND, yaitu antara data pembawa bit dengan hasil pembalikan nilai dari *mask*. Untuk melakukan pembalikan nilai variabel ini digunakan operasi *bitwise complement*. Akan tetapi penggunaan operasi bilangan ini pada Visual Studio menyebabkan terjadinya salah pembacaan nilai, yaitu hasil akhir dari data yang dikirimkan bernilai negatif. Oleh karena itu, digunakan fungsi *unchecked* pada proses AND. Perintah-perintah tersebut diakses menggunakan komunikasi USART dari *board* ATMega16 melalui sebuah protokol yang telah dibuat, dengan daftar-daftar data sesuai pada Tabel 3.3.

Tabel 3.3 Protokol USART LattePanda dengan ATMega16

Byte_Index	Contain	Info	Bit_Index
0	Header	0x40 (‘@’)	n/a
1	Data_Size	2	n/a
2	Status	~	7
		Inv.Kin	6
		Normal	5
		Emergency	4
	Menu	~	3
		~	2
		Menu Status	1
		Menu Toggle	0
3	Cam_Select	Rear	7
		Front	6
		Gripper	5
		Main	4
	Lamp_Select	Front	3
		Gripper	2
		Main	1
		Status	0
4	CRC(XOR)	XOR_Status	n/a

3.2.3 *Input System*

Proses selanjutnya yang dilakukan yaitu menerjemahkan informasi yang telah dikirimkan untuk dapat memproses perintah yang diinginkan. Proses ini dilakukan pada *input system*, dengan penerjemahan informasi yang dilakukan pada dua *file* program berbeda, yaitu *PageInputSystem.cs* dan *RhinoInputSystem.cs*.



Gambar 3.10 Tombol untuk *PageInputSystem.cs*

Pada *file* program *PageInputSystem.cs* proses yang dilakukan yaitu menerjemahkan informasi yang berhubungan dengan halaman operasi (tampilan). Diantaranya yaitu informasi dari tombol-tombol untuk mengakses tampilan maupun LED yang perlu diatur terlebih dahulu pada tampilan. Daftar tombol yang digunakan pada proses ini yaitu sesuai dengan Gambar 3.10.

Sedangkan *file RhinoInputSystem.cs* berfungsi untuk menerjemahkan fungsi-fungsi yang berhubungan dengan robot. Perintah-perintah tersebut terdiri dari tombol dan LED pada *panel input* selain yang diakses pada *file PageInputSystem.cs*. Pada proses ini juga terdapat beberapa penanganan khusus untuk pengaksesan fungsi-fungsi tertentu. Penanganan ini dilakukan dikarenakan adanya penggunaan kombinasi tombol, diantaranya:

a. Fungsi *ToggleBrake*

   Pada fungsi ini nilai yang dikembalikan diantaranya sesuai dengan perintah-perintah eksekusi yang diberikan, diantaranya:

   - Jika keadaan dari tombol *joystick* kanan pada *panel input* baru ditekan dan tertahan hingga lima *cycle* berikutnya, maka data yang dikembalikan bernilai *true*.

- Sedangkan jika terdapat perangkat kendali tambahan dan tidak berada pada mode NONE, serta keadaan dari tombol *back* pada *gamepad* menunjukkan bahwa baru ditekan, maka data yang dikembalikan yaitu keadaan dari tombol *back* pada *gamepad*.
- Selain keadaan tersebut, data yang dikembalikan bernilai *false*.

b. Fungsi *ToggleDrivetrainMode*,
Pada fungsi ini nilai yang dikembalikan diantaranya sesuai dengan perintah-perintah eksekusi yang diberikan, diantaranya:
- Jika keadaan dari tombol *joystick* kanan pada *panel input* baru ditekan dan tertahan hingga cycle berikutnya, maka data yang dikembalikan bernilai *true*.
- Sedangkan jika terdapat perangkat kendali tambahan dan berada pada mode *drivetrain,* serta keadaan dari tombol *joystick* kiri pada *gamepad* baru ditekan, maka data yang dikembalikan yaitu keadaan dari tombol *joystick* kiri pada *gamepad*.
- Selain keadaan tersebut, data yang dikembalikan bernilai *false*.

Metode yang digunakan untuk menjalankan kedua fungsi tersebut yaitu dengan digunakannya *switch-case*. Dimana *case* pertama berisi perintah *await*, berfungsi untuk menunggu apakah ada perintah yang diberikan atau tidak. *Case* kedua berisi perintah*hold*, yaitu untuk mengetahui apakah tombol yang ditekan kemudian ditahan atau tidak. Selanjutnya *case* ketiga atau yang terakhir berisi perintah untuk *toggled*, yaitu mengakses perintah yang diinginkan.

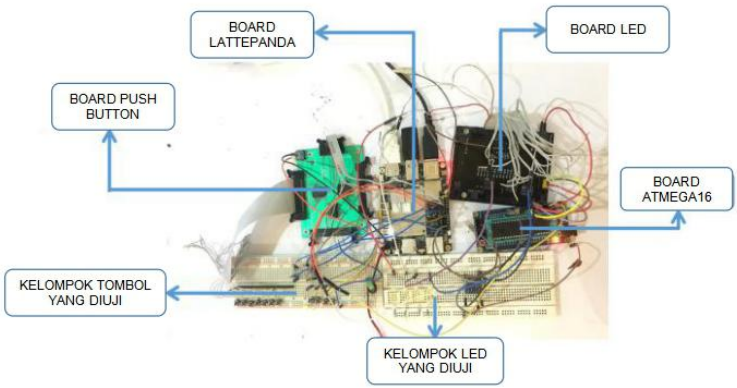
Selain menerjemahkan informasi untuk masing-masing tombol, pada *input system* juga terdapat proses *toggle* data untuk LED indikator. Sama seperti penerjemahan yang dilakukan pembagian, proses *toggle* data LED juga dilakukan pada dua *file* berbeda. Data yang diakses pada *PegeInputSystem.cs* diantaranya data untuk *emergency, menu toggle, menu status*, dan data untuk kamera. *File RhinoInputSystem.cs* bertujuan untuk mengakses data tombol *emergency, normal, inv. kin*, dan data untuk lampu.

Selain program-program diatas, pada perancangan antarmuka OCU ini diperlukan sebuah program tambahan berisi sekelompok enumerasi yang digunakan. Diantaranya enumerasi dari nama-nama tombol yang telah ditentukan, serta enumerasi dari *class* untuk

perangkat kendali tambahan. Program-program yang telah disusun memiliki alur kerja sesuai *flowchart* pada Lampiran 3.

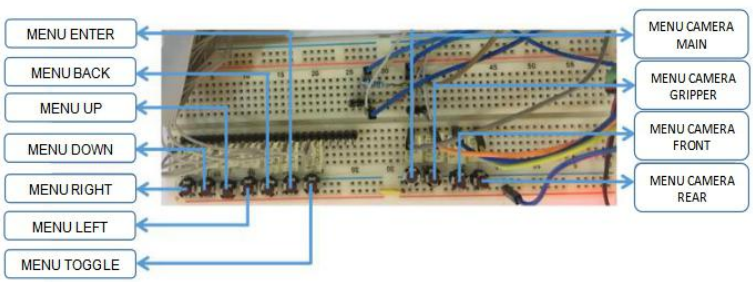
3.4 Metodologi Pengujian

Pengujian yang dilakukan menggunakan rangkaian seperti yang ada pada Gambar 3.11. Dimana LattePanda dihubungkan dengan ketiga *board* penyusun panel *input* OCU, yaitu *board push button, board* LED, dan *board* ATMega. *Board push button* kemudian dihubungkan secara langsung menuju tombol-tombol yang digunakan. Begitu juga dengan *board* LED, langsung dihubungkan dengan LED yang berperan sebagai indikator.



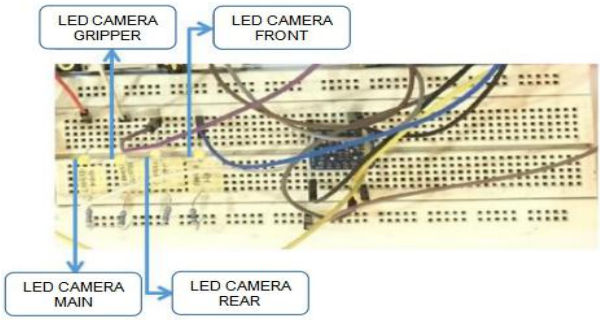
Gambar 3.11 Rangkaian Pengujian Sistem Antarmuka

Peletakan tombol-tombol yang digunakan pada proses pengujian tampak seperti pada Gambar 3.12.



Gambar 3.12 Tata Letak Tombol pada *Board* Pengujian

Rangkaian yang digunakan dalam proses pengujian fungsi LED dapat dilihat pada Gambar 3.13. Dimana dalam rangkaian ini dibutuhkan sebuah IC bernama level shifter yang berfungsi untuk menaikkan tegangan yang didapat dari LattePanda.



Gambar 3.13 Tata Letak LED pada *Board* Pengujian

3.5 *Set-up* Visual Studio pada LattePanda

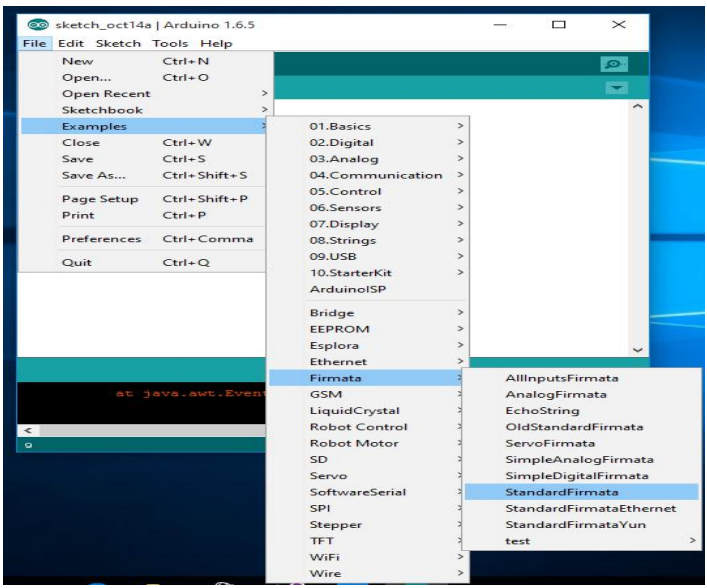
Untuk dapat mengakses *co-processor* Arduino yang ada pada LattePanda menggunakan Visual Studio diperlukan sebuah pengaturan [10]. Langkah pertama yang harus dilakukan yaitu *set-up* pada *development board* LattePanda (PC) dengan cara:

a. Mengunduh *software* Visual Studio pada PC.
b. Mengaktifkan '*developer mode*' pada sistem operasi PC, dengan langkah-langkah sesuai Gambar 3.14.
c. Mengunduh library LattePanda.Firmata pada website resmi LattePanda.

Gambar 3.14 Cara Mengaktifkan *Developer Mode*

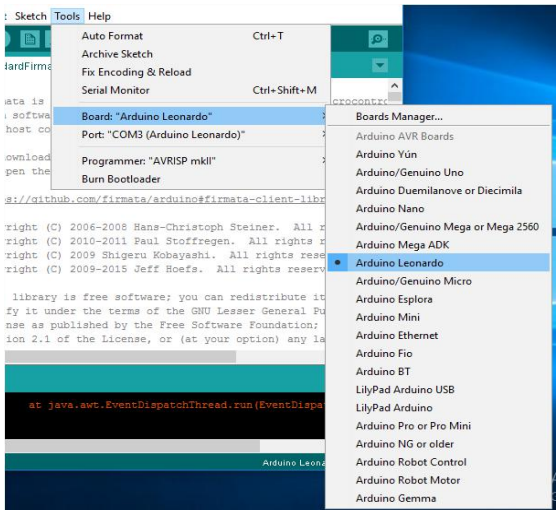
Setelah selesai mengganti mode PC, dilakukan *set-up software* Arduino yang ada pada *development board* LattePanda dengan cara:

a. *Software* Arduino yang terdapat pada PC dijalankan, kemudian digunakan library bernama ‘StandartFirmata’ pada menu “*File* - *Examples* - Firmata - StandartFirmata”. Langkah yang dilakukan sesuai pada Gambar 3.15.



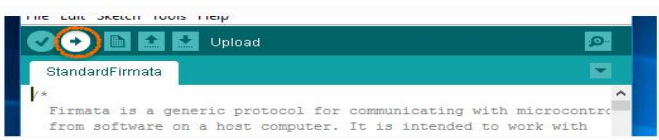
Gambar 3.15 Cara Membuka *File* StandartFirmata

b. Memilih jenis Arduino yang digunakan pada menu “*Tools* - *Board* - Arduino Leonardo”. Langkah yang perlu diambil sesuai Gambar 3.16.
c. Memilih *port* COM yang akan digunakan, dengan memilih menu “*Tools* - *Port* - COM” sesuai langkah yang ditunjukkan oleh Gambar 3.17.
d. Melakukan *upload file* program dari StandartFirmata yang telah tersedia dengan mengakses tombol anak panah sesuai pada Gambar 3.18.
e. *Upload file* telah selesai dilakukan apabila muncul keterangan seperti pada Gambar 3.19 di bagian kanan bawah *software* Arduino.



Gambar 3.16 Cara Memilih Tipe Arduino yang Digunakan

Gambar 3.17 Cara Memilih *Port* COM yang Digunakan

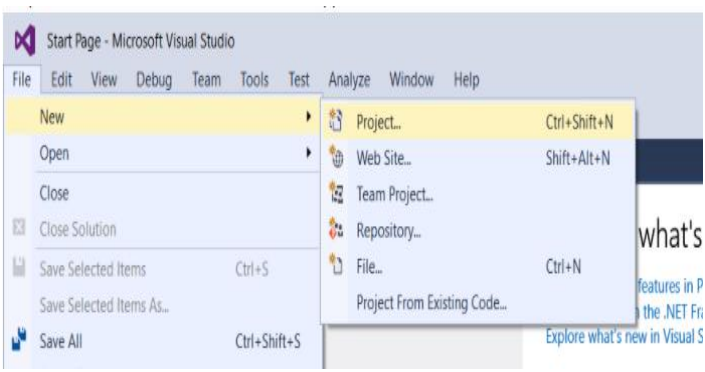


Gambar 3.18 Cara Melakukan *Upload File* pada Arduino

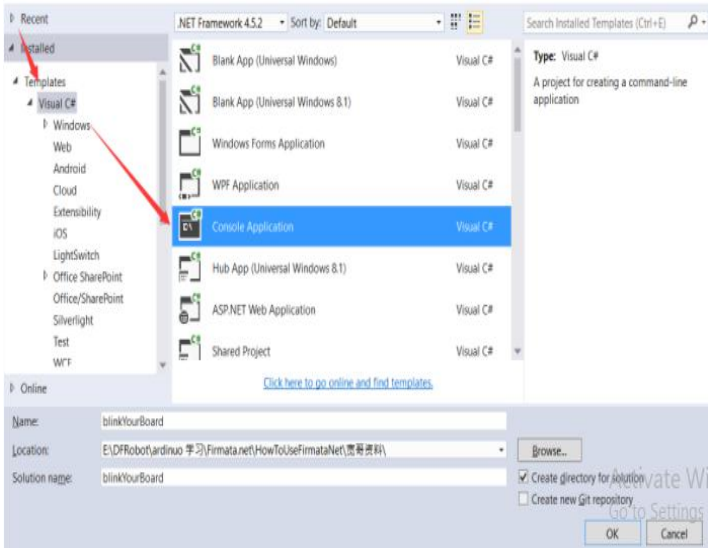
Gambar 3.19 Indikator *File* Selesai di-*upload*

Setelah semua langkah *set-up* terlaksana, maka dapat dilakukan pengujian program. Pengujian dapat dilakukan dengan menggunakan contoh *project* bernama 'blinkyourboard' yang telah tersedia pada *website* resmi LattePanda [4] dengan cara:

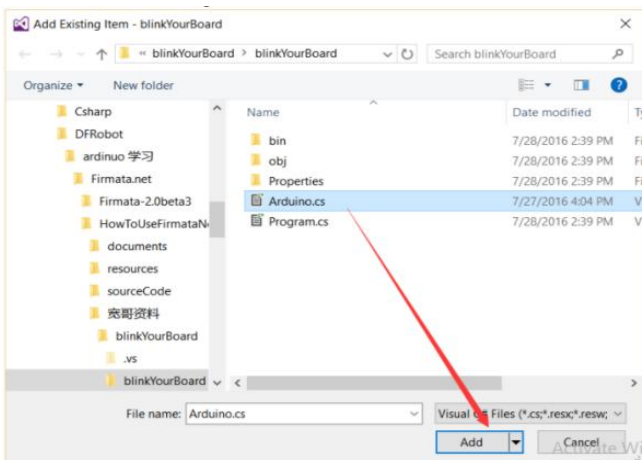
a. Menjalankan aplikasi *software* Visual Studio yang telah terpasang pada PC. Kemudian membuat *file project* baru dengan langkah-langkah sesuai pada Gambar 3.20 dan Gambar 3.21.
b. Menambahkan *library* LattePanda.Firmata yang telah diunduh pada *project* yang dibuat. Hal tersebut dapat dilakukan dengan menekan tombol *mouse* bagian kanan pada daerah kosong yang tersedia di jendela *solution explorer*. Langkah selanjutnya muncul jendela seperti pada Gambar 3.22 dan *file* yang akan digunakan dapat dipilih.
c. Menambahkan *listing program* pada *file project,* diantaranya:
    - Menambahkan dua baris *listing program* pada Gambar 3.23 sebelum *namespace* pada *file project. Listing program* yang pertama digunakan untuk mengatur *delay*, sedangkan yang kedua merupakan *namespace* dari *library* LattePanda.Firmata.
    - Tambahkan listing program dari Gambar 2.24 pada fungsi utama *file project*.
d. Hasil *project* yang telah dibuat tampak seperti pada Gambar 3.25, yang berisi *listing program* untuk menjalankan LED berkedip-kedip.



Gambar 3.20 Cara Membuat *Project* Baru pada Visual Studio



Gambar 3.21 Cara Membuat *Project* Baru pada Visual Studio



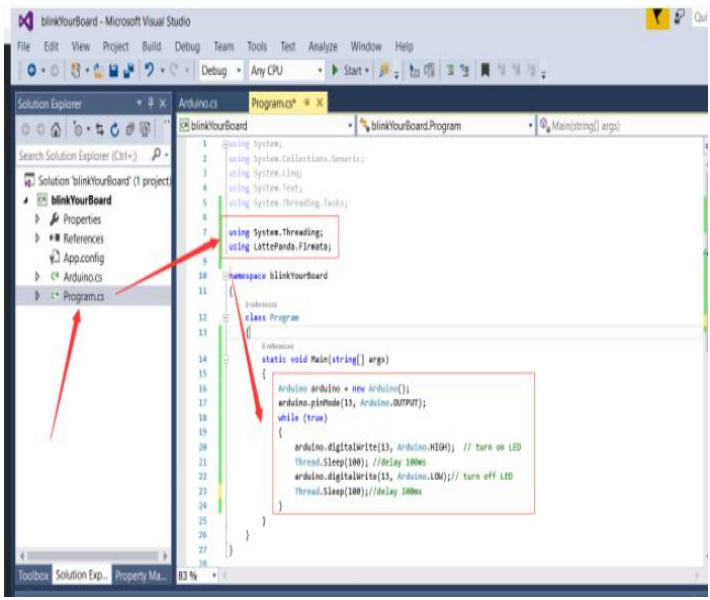
Gambar 3.22 Cara Menambahkan *File Library* pada Visual Studio

```
using System.Threading;
using LattePanda.Firmata;
```

Gambar 3.23 *Listing Program Blink Your Board*

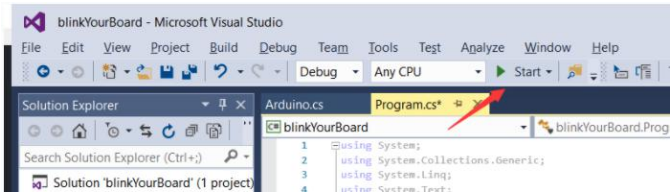
```
Arduino arduino = new Arduino();
arduino.pinMode (13, Arduino.OUTPUT);//pin 13 sebagai output
while (true)
{
        arduino.digitalWrite(13, Arduino.HIGH);//LED nyala
        Thread.Sleep(1000);//delay untuk 1 detik
        arduino.digitalWrite(13, Arduino.LOW);//LED mati
        Thread.Sleep(1000);//delay untuk 1 detik
}
```

Gambar 3.24 *Listing Program Blink Your Board*



Gambar 3.25 *File Project* untuk Program 'blinkyourboard'

Langkah selanjutnya dapat dijalankan file *project* yang telah dibuat dengan menghubungkan Arduino dan tekan tombol *start* pada Visual Studio sesuai dengan Gambar 3.26.



Gambar 3.26 Cara Menjalankan *Project* pada Visual Studio

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

BAB IV
HASIL PENGUJIAN SISTEM ANTARMUKA

Berdasarkan hasil dari perancangan sistem yang telah disusun, untuk mengetahui ketepatan kerjanya dilakukan pengujian. Dengan 2 proses pengujian berbeda berdasar pada hasil fungsi yang ingin diketahui, yaitu pengujian untuk antarmuka dari tombol-tombol pada panel *input* dan LED indikator.

4.1 Pengujian Antarmuka dari *Board Push Button*

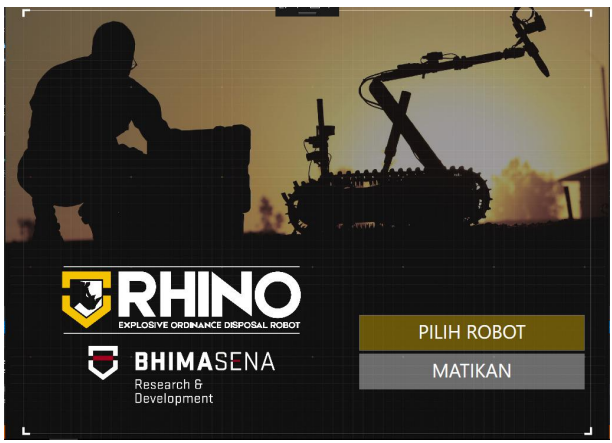
Tahap pengujian pertama yang dilakukan bertujuan untuk mengetahui ketepatan fungsi kerja antarmuka dari tombol-tombol pada panel *input.* Pengujian ini dilakukan secara parsial, diantaranya terhadap tombol-tombol menu dan tombol-tombol aksesoris kamera seperti yang ada pada Tabel 4.1. Hal tersebut dilakukan karena pada saat pengujian, robot dalam keadaan perbaikan, sehingga tombol-tombol untuk operasi robot tidak dapat dilakukan pengujian. Langkah yang dilakukan untuk mendapatkan hasil pengujian yaitu dengan mengamati tampilan dari halaman GUI saat tombol ditekan.

Tabel 4.1 Hasil Pengujian Fungsi Antarmuka Panel *Input*

No.	Fungsi antarmuka tombol	Hasil Pengujian	
		Berhasil	Tidak
1.	Tombol menu direction up	√	-
2.	Tombol menu direction *down*	√	-
3.	Tombol menu direction *left*	√	-
4.	Tombol menu direction *right*	√	-
5.	Tombol menu *enter*	√	-
6.	Tombol menu *toggle*	√	-
7.	Tombol aksesoris *camera main*	√	-
8.	Tombol aksesoris *camera rear*	√	-
9.	Tombol aksesoris *camera front*	√	-
10.	Tombol aksesoris *camera gripper*	√	-

Data yang terdapat pada Tabel 4.1 merupakan hasil dari pengujian secara keseluruhan. Data-data tersebut didapatkan dari hasil pengamatan ayng dilakuakn terhadap pengujian. Secara keseluruhan, pengujian dilakukan secara bertahap, dimulai dari pertama kali program dijalankan yang menampilakn GUI hingga penggunaan aksesoris kamera.

Langkah pengujian yang pertama kali dilakukan yaitu menjalankan program antarmuka. Program tersebut telah terhubung dengan GUI dari OCU yang berfungsi untuk menampilkan menu-menu kendali dari robot. Hasil dari pengujian didapatkan seperti yang tertampil pada Gambar 4.1. Dimana saat program pertama kali dijalankan, halaman GUI yang ditampilkan berisi 2 menu pilihan operasi, yaitu "pilih robot" dan "matikan". Pada tampilan ini, kursor secara *default* terletak pada menu "pilih robot".

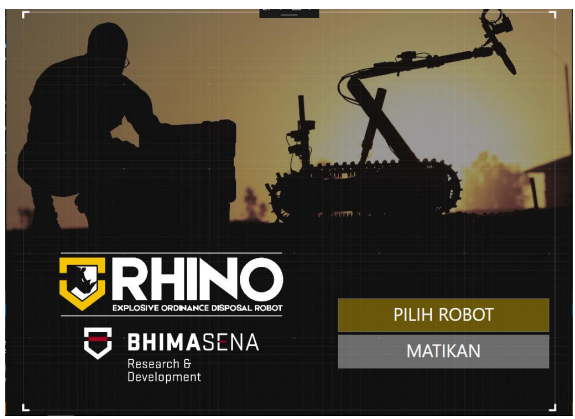


Gambar 4.1 Tampilan saat Program Dijalankan

Langkah selanjutnya yang dilakukan yaitu dengan menekan tombol menu *direction down.* Hasil dari pengujian tahap ini dapat dilihat pada Gambar 4.2, dimana ketika tombol ditekan, hasil yang didapatkan yaitu kursor berpindah ke menu yang terletak di bawahnya, yaitu menu matikan. Hal tersebut dapat diketahui karena sebelum tombol ditekan, kursor berada pada menu pilih robot, seperti yang ada pada Gambar 4.1.

Gambar 4.2 Tampilan saat Tombol Menu *Down* Ditekan

Pengujian selanjutnya yang dilakukan yaitu dengan menekan tombol menu *direction up.* Pada pengujian ini, hasil yang diharapkan yaitu kursor berpindah ke menu yang ada di atasnya. Pada pengujian ini, hasil yang ditampilkan tampak seperti Gambar 4.3. Kursor yang sebelumnya berada pada menu matikan, berpindah kembali ke menu pilih robot. Hal tersebut dikarenakan menu pilih robot berada di atas menu matikan.



Gambar 4.3 Tampilan saat Tombol Menu *Up* Ditekan

Setelah kursor kembali berada pada menu pilih robot, maka pengujian selanjutnya yang dilakukan yaitu dengan menekan tombol menu *enter*. Hasil pengujian tahap ini yaitu halaman tampilan berganti seperti yang ada pada Gambar 4.4.

Gambar 4.4 Tampilan saat Tombol Menu *Enter* Ditekan

Berdasar yang ditampilkan oleh Gambar 4.4, dapat diketahui bahwa setelah tombol menu *enter* ditekan, maka tampilan dari halaman GUI berubah ke halaman selanjutnya. Halaman tersebut digunakan untuk memasukkan identitas robot. Dimana kursor secara *default* berada pada bagian seri robot.

Saat berada pada halaman Masukkan ID Robot, pengujian tombol dilakukan 2 kali, yaitu untuk tombol menu *left* dan *right*. Pengujian yang pertama dilakukan yaitu untuk tombol menu *right*, dengan hasil seperti yang ditampilkan oleh Gambar 4.5. Dimana kursor yang awalnya berada pada bagian seri robot kemudian berpindah ke menu nomor robot. Menu ini digunakan untuk melakukan pengaturan identitas dari robot yang digunakan.

Akan tetapi jika tombol yang ditekan adalah tombol menu *left,* maka kursor dari menu seri robot berpindah ke menu hubungkan. Menu ini berfungsi untuk menghubungkan sistem operasi yang ditampilkan pada GUI dengan robot yang digunakan, tujuannya yaitu untuk mengirimkan kendali terhadap robot.

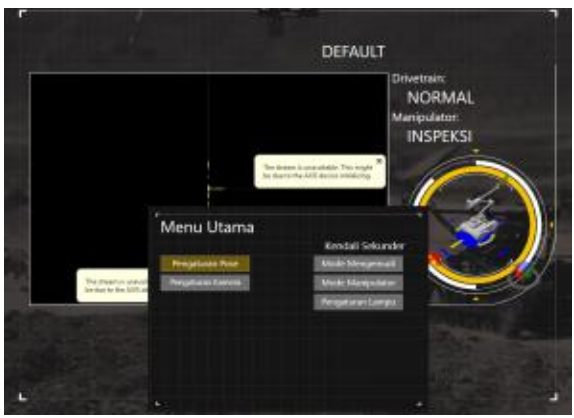
Gambar 4.5 Tampilan saat Ditekan Tombol Menu *Right*

Gambar 4.6 Tampilan saat Ditekan Tombol Menu *Left*

Setelah kursor berada pada menu hubungkan, maka pengujian selanjutnya yang dapat dilakukan yaitu pengujian tombol *enter*. Hasil yang didapatkan tampak seperti yang ada pada Gambar 4.7, dimana halaman yang ditampilkan berganti ke halaman selanjutnya, yaitu halaman operasi utama. Halaman ini digunakan untuk melakukan pemilihan operasi untuk robot. Menu-menu pilihan tersebut berada pada halaman selanjutnya yang ditampilkan melalui menu pop-up.

Gambar 4.7 Tampilan saat Ditekan Tombol Menu *Enter*

Langkah yang perlu dilakukan untuk menampilkan menu *pop-up* yang berisi menu-menu operasi dari robot, dapat dilakukan dengan menekan tombol menu *toggle*. Halaman yang ditampilkan saat tombol ditekan tampak seperti Gambar 4.8, yang awalnya tidak ada menu pop-up.



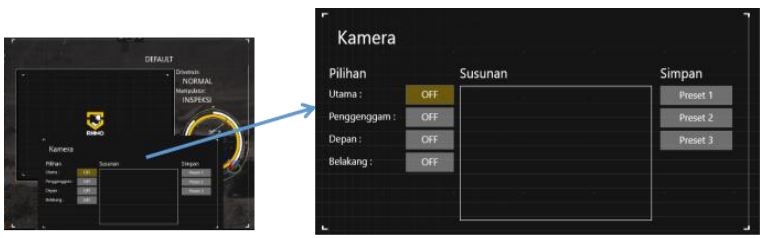
Gambar 4.8 Tampilan saat Ditekan Tombol Menu *Toggle*

Untuk melakukan pengujian selanjutnya, menu yang dipilih yaitu pengaturan kamera. Untuk dapat mengakses menu tersebut, kursor

harus dipindahkan ke bawah terlebih dahulu. Tombol yang digunakan untuk menyelesaikan pengujian ini yaitu menu *direction down,* dengan hasil yang ditampilkan seperti yang ada pada Gambar 4.9.

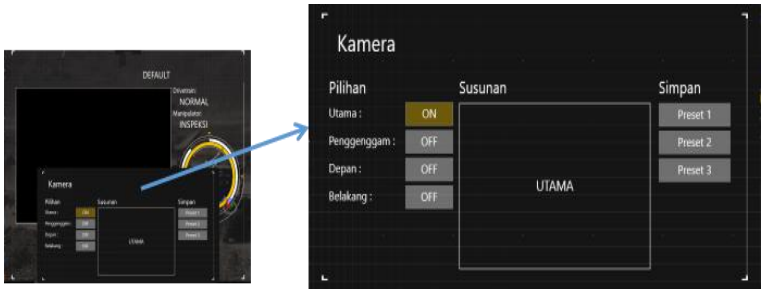
Gambar 4.9 Tampilan saat Tombol Menu *Down* Ditekan

Untuk melakukan pengujian pada menu operasi kamera ini, diperlukan pemilihan menu operasi ini, langkah yang dilakukan yaitu dengan menekan tombol menu *enter.* Hasil yang didapatkan setelah tombol menu *enter* ditekan tampak seperti Gambar 4.10, dimana menu yang ditampilkan terdapat 4 pilihan, yaitu utama (*main*), penggenggam (*gripper*), depan (*front*), dan belakang (*rear*).



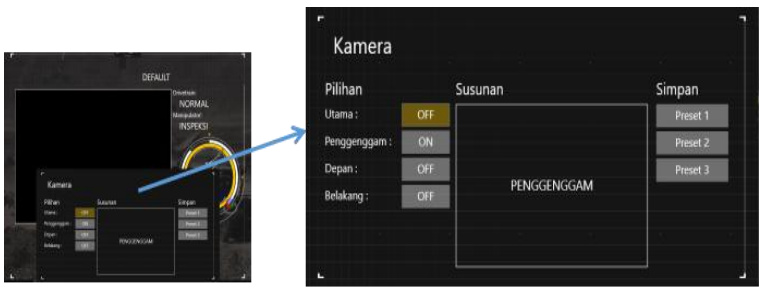
Gambar 4.10 Tampilan saat Tombol Menu *Enter* Ditekan

Pengujian pertama pada operasi aksesoris kamera yang dilakukan yaitu untuk tombol menu *camera main*. Hasil yang ditampilkan tampak seperti Gambar 4.11, dimana setelah tombol ditekan, keterangan dari kamera tersebut berubah menjadi *on* yang pada awalnya keadaanya adalah *off.*



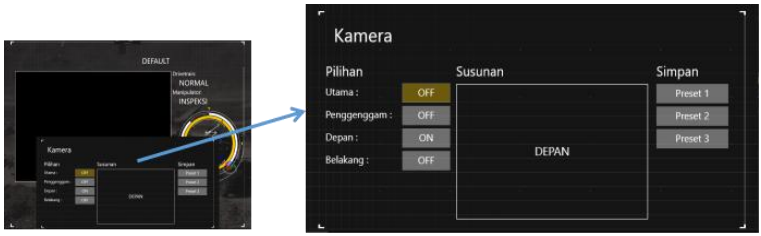
Gambar 4.11 Tampilan saat Tombol *Camera Main* Ditekan

Pengujian yang dilakukan untuk tombol aksesori *camera gripper* ini tidak berbeda dengan pengujian sebelumnya, yaitu hanya dengan menekan tombol dari menu aksesoris *camera gripper*. Hasil yang didapatkan setelah tombol ditekan, sesuai dengan yang ditampilkan oleh Gambar 4.12. Perbedaan hasil dari pengujian yang sebelumnya yaitu keadaan yang dihasilkan, yaitu keadaan dari kamera penggenggam berubah menjadi *on* dari yang sebelumnya adalah *off*.



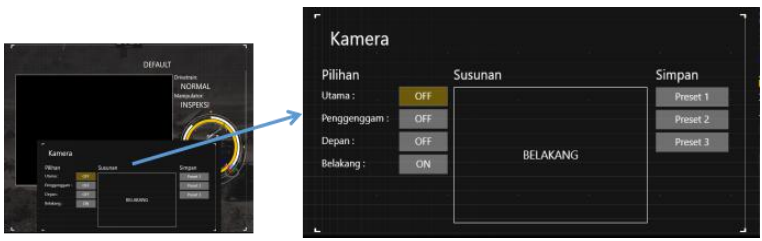
Gambar 4.12 Tampilan saat Tombol *Camera Gripper* Ditekan

Untuk tombol aksesoris *camera front,* pengujian yang dilakukan yaitu dengan menekan tombol dari menu aksesoris tersebut. Dengan hasil yang didapatkan yaitu seperti yang ada pada Gambar 4.13, dimana keadaaan yang berubah dari *off* menjadi *on* adalah menu pilihan depan. Hal tersebut menunjukkan jika menu aksesoris untuk *camera front* sedang digunakan, atau kamera yang dinyalakan yaitu kamera depan.



Gambar 4.13 Tampilan saat Tombol *Camera Front* Ditekan

Pengujian terakhir yang dilakukan yaitu untuk menu tombol aksesoris *camera front.* Dimana ketika tombol ini ditekan, hasil yang ditampilkan tampak seperti yang ada pada Gambar 4.14. Keadaan yang berubah yaitu pada pilihan belakang, dimana awalnya *off* kemudian berubah jadi *on*.



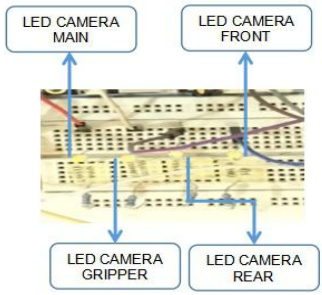
Gambar 4.14 Tampilan saat Tombol *Camera Front* Ditekan

Berdasar hasil pengujian yang telah dilakukan dari awal hingga akhir ini, dapat disimpilkan bahwa fungsi antarmuka dari tombol-tombol yang disusun telah mampu beroperasi sesuai dengan yang dibutuhkan.

4.2 Pengujian Antarmuka dari *Board* LED

Pada tahap pengujian antarmuka untuk LED indikator, langkah yang dilakukan yaitu mengamati keadaan LED dari tombol aksesoris kamera. Proses yang dilakukan yaitu melakukan pengaturan keadaan dari beberapa kamera melalui tombol-tombol menu yang telah disediakan. Berikut merupakan hasil pengamatan yang didapat saat pengujian:

Saat halaman tampilan dari GUI menunjukkan menu *pop-up,* maka manu pilihan yang diakses yaitu menu pengaturan kamera. Langkah pengujian pertama yang dilakukan yaitu tidak menekan satupun tombol menu aksesoris untuk kamera. Hasil dari pengujian yang berupa kondisi dari masing-masing LED indikator dapat dilihat pada Gambar 4.15.



Gambar 4.15 Kondisi LED Saat Tidak Ada Tombol yang Ditekan

Berdasarkan hasil pengujian pada Gambar 4.15, dapat diketahui jika tidak ada tombol yang ditekan, maka LED tidak menyala. Hal tersebut menunjukkan bahwa tombol tidak dalam keadaan aktif.

Pengujian selanjutnya yang dilakukan yaitu pada pilihan pengaturan kamera di halaman operasi, saat tombol yang ditekan adalah tombol menu kamera *rear*, maka LED yang menyala tampak seperti Gambar 4.16.

Gambar 4.16 Kondisi LED Saat Tombol Menu Kamera *Rear* Ditekan

Sesuai dengan hasil yang ditampilkan pada Gambar 4.16, dapat diketahui jika tombol menu untuk kamera depan ditekan (tombol menu kamera *rear*), maka LED yang berfungsi sebagai indikatornya

menyala. Hal tersebut menunjukkan bahwa tombol tersebut dalam keadaan aktif, atau sedang digunakan.

Setelah LED indikator dari menu kamera *rear* menyala, pengujian selanjutnya yang dilakukan yaitu dengan menekan tombol menu kamera *front*. Hal ini dilakukan guna mengetahui, apakah tombol dapat diakses dalam waktu yang bersamaan. Hasil yang ditampilkan dari pengujian LED tampak seperti Gambar 4.17.

Gambar 4.17 Kondisi LED Saat Tombol Menu Kamera *Front* Ditekan

Hasil yang ditunjukkan pada Gambar 4.17 menunjukkan bahwa jika ada penambahan tombol yang dipencet, maka LED indikator dari tombol tersebut juga menyala, tanpa mematikan LED indikator dari tombol lainnya. Hal tersebut membuktikan bahwa tombol yang sedang aktif, atau menu kamera yang digunakan berjumlah lebih dari 2.

---- Halaman ini sengaja dikosongkan ---

BAB V
PENUTUP

Perancangan *Operator Control Unit* (OCU) Rhino Robot versi baru ini disusun dalam beberapa tahap, dimulai dari perancangan *hardware* dan *software* panel *input*, optimalisasi GUI, dan perancangan *power distribution.* Panel *input* dari perangkat ini terdiri dari 45 buah tombol dan sejumlah LED sebagai indikatornya. Untuk dapat menjalankan fungsi dari masing-masing tombol tersebut, dibutuhkan perancangan *software* (program) antarmukanya. Penyusunan dilakukan pada IDE bernama Visual Studio dengan menggunakan bahasa pemrograman C#. Untuk dapat menghubungkan program yang telah disusun dengan perancangan *hardware,* yang menggunakan Arduino dari LattePanda digunakan sebuah protokol bernama LattePanda.Firmata.

Setelah sistem selesai disusun, kemudian dilakukan pengujian melalui dua tahap berbeda. Pengujian pertama dilakukan untuk mengetahui ketepatan fungsi antarmuka tombol-tombol pada panel *input.* Pengujian selanjutnya yaitu untuk mengetahui ketepatan fungsi antarmuka dari LED indikator. Hasil dari pengujian menunjukkan bahwa sistem yang disusun telah mampu menjalankan fungsi antarmuka yang diinginkan. Hal tersebut diketahui dari hasil pengujian yang menunjukkan saat tombol ditekan, maka fungsi antarmuka yang diperankan kemudian dijalankan. Selain itu, LED yang digunakan sebagai indikator dari beberapa tombol telah mampu berjalan sesuai fungsinya, dimana saat tombol ditekan, maka LED yang berperan sebagai indikatornya akan menyala.

Pada pengujian hasil dari perancangan antarmuka OCU Rhino Robot masih digunakan rangkaian pada *breadboard* dan *push button* biasa. Untuk dapat mengetahui keakuratan dari program yang disusun lebih baik digunakan rangkaian dan komponen yang sesungguhnya. Selain itu, juga penyusunan *listing program* yang masih berantakan, untuk memudahkan pembacaan program oleh pengguna, lebih baik pada pengembangan selanjutnya program disusun sedemikian rupa sehingga pengguna dapat membaca dengan mudah.

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

DAFTAR PUSTAKA

[1] F. Luqiao, et al, "A Single-Hand and Binocular Visual System for EOD Robot", Automation and Logistics, 2007 IEEE International Conference on, ISSN 2161-8151.

[2] M. W. Carey, et al, "Novel EOD Robot Design with Dexterous Gripper and Intuitive Teleoperation", World Automation Congres (WAC), 2012, ISSN 2154-4824.

[3] N. Checka, et al, "Handheld Operator Control Unit", Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on, ISSN 2167-2121.

[4] LattePanda, "Peripheral Device", [Online], Available: http://docs.lattepanda.com/content/getStarted/peripheral/.

[5] P. Duddu, 2014, "The top 5 military robots for explosive ordnance disposal", [Online], Available: https://www.army-technology.com/features/featuredetect-and-diffuse-the-top-5-military-robots-for-explosive-ordnance-disposal-4372678/.

[6] J. Crossman, R. Marinier, E. B. Olson, "A Hands-Off, Multi-Robot Display for Communicating Situation Awareness to Operators", Collaboration Technologies and Systems (CTS), 2012 International Conference on.

[7] P. Candela, et al, "Designing an Operator Control Unit for Cooperative Autonomous Unmanned Systems", Proc. SPIE Vol. 10195, 101950L, 2017.

[8] --, "Antarmuka", [Online], Available: https://id.wikipedia.org/wiki/Antarmuka.

[9] --, "Antramuka Pengguna", [Online], Available: https://id.wikipedia.org/wiki/Antarmuka_pengguna.

[10] LattePanda, "3 Steps to Access Pinouts", [Online], Available: http://docs.lattepanda.com/content/hardware/accessPinoutsFromVS/.

[11] Datasheet, "CD4067B, CD4097B Types", Texas Instruments, 2017.

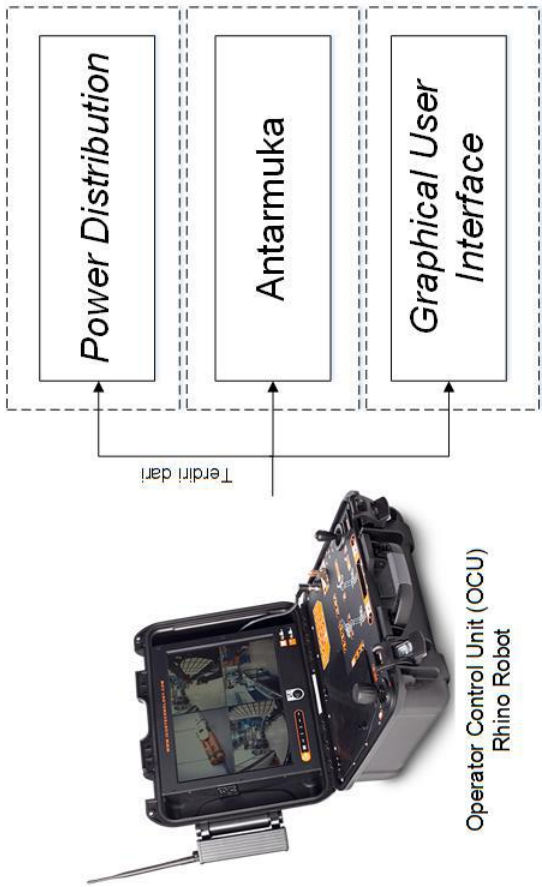
[12] Microsoft, 2018, "Visual Studio IDE", [Online], Available: https://msdn.microsoft.com/en-us/library/dn762121.aspx.

[13] K. Agus, dkk, "Pengenalan Bahasa C", Project Otak, Jakarta, 2004.

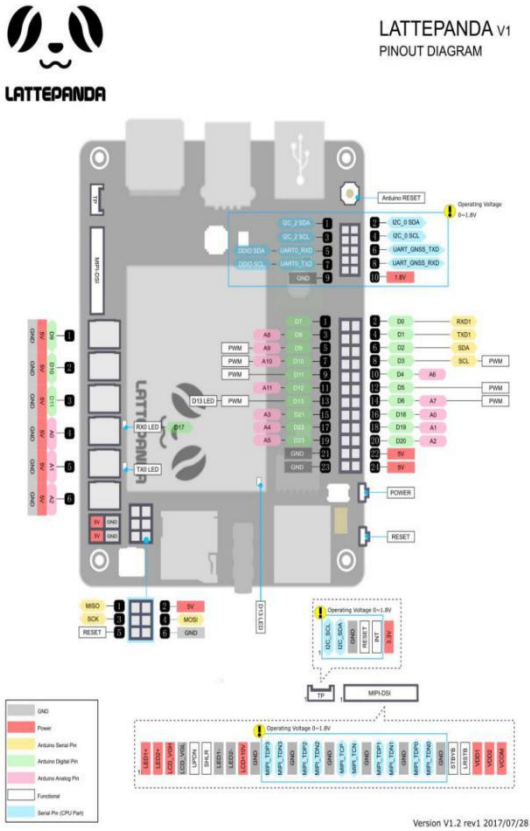
[14] R. Agro, “Mempelajari C#: Bahasa Pemrograman Modern”, 2002.
[15] Suyadi, “Komunikasi Serial dan Port Serial (COM)”, L200100015, Teknik Informatika UMS, 2012.
[16] Aspencore, “The Multiplexer”, [Online], Available: https://www.electronics-tutorials.ws/combination/comb_2.html.
[17] H. R. Charles, L. K. Larry, “Fundamentals of Logic Design”, Cengage Learning, 2010.

LAMPIRAN

Lampiran 1. Lingkup Pengerjaan dari Perancangan *Operator Control Unit* (OCU) Rhino Robot



Lampiran 2. GPIO LattePanda



Pinouts in area U1 are assigned to the X-Z8300 core.

Pinouts in area U2 are assigned to the ATmega32u4 core. Each of the 20 digital pins (A0 - A5, D0 - D13) in area U2 can be used as an input or output, each operating at 5 volts. Each pin can output or receive 40 mA and each has an internal pull-up resistor (disconnected by default) of 20-50k ohm.**Caution:** Exceeding 40mA on any I/O pin may cause permanent damage to the ATmega32u4.

Some pins have specialized functions:

- **Analog Inputs:** A0 - A5, A6 - A11 (on D4, D6, D8, D9, D10, and D12). The LattePanda has 12 analog inputs, labeled A0 through A11, all of which can also be used as digital I/O. Each pin has a 10 bit resolution (i.e. 1024 different values). By default they measure from ground to 5 volts.
- **Serial:** D0 (RX) and D1 (TX). Used to receive (RX) and transmit (TX) TTL serial data.
- **External Interrupts:** D3 (interrupt 0), D2 (interrupt 1), D0 (interrupt 2), D1 (interrupt 3) and D7 (interrupt 4). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM:** D3, D5, D6, D9, D10, and D13 provide 8-bit PWM output.
- **SPI:** D16 (MOSI), D14 (MISO), D15 (SCK).
- **LED:** D13 There is a built-in LED driven by digital pin 13.
- **TWI:** D2(SDA), D3(SCL). **Other pins on the board:**
- **Reset:** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Lampiran 3. *Datasheet* MUX

TEXAS INSTRUMENTS

Data sheet acquired from Harris Semiconductor
SCHS052B – Revised June 2003

CD4067B, CD4097B Types

CMOS Analog Multiplexers/Demultiplexers

High-Voltage Types (20-Volt Rating)

CD4067B – Single 16-Channel Multiplexer/Demultiplexer

CD4097B – Differential 8-Channel Multiplexer/Demultiplexer

■ CD4067B and CD4097B CMOS analog multiplexers/demultiplexers* are digitally controlled analog switches having low ON impedance, low OFF leakage current, and internal address decoding. In addition, the ON resistance is relatively constant over the full input-signal range.

The CD4067B is a 16-channel multiplexer with four binary control inputs, A,B,C,D, and an inhibit input, arranged so that any combination of the inputs selects one switch.

The CD4097B is a differential 8-channel multiplexer having three binary control inputs A, B, C, and an inhibit input. The inputs permit selection of one of eight pairs of switches.

A logic "1" present at the inhibit input turns all channels off.

The CD4067B and CD4097B types are supplied in 24-lead hermetic dual-in-line ceramic packages (F3A suffix), 24-lead dual-in-line plastic packages (E suffix), 24-lead small-outline packages (M, M96, and NSR suffixes), and 24-lead thin shrink small-outline packages (P and PWR suffixes).

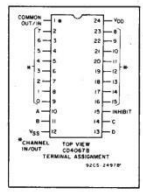
*When these devices are used as demultiplexers, the channel in/out terminals are the outputs and the common out/in terminals are the inputs.

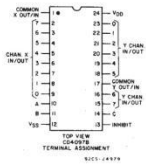
Features:

- Low ON resistance: 125 Ω (typ.) over 15 V_{p-p} signal-input range for $V_{DD}-V_{SS}=15$ V
- High OFF resistance: channel leakage of ±10 pA (typ.) @ $V_{DD}-V_{SS}=10$ V
- Matched switch characteristics: $R_{ON}=5$ Ω (typ.) for $V_{DD}-V_{SS}=15$ V
- Very low quiescent power dissipation under all digital-control input and supply conditions: 0.2 μW (typ.) @ $V_{DD}-V_{SS}=10$ V
- Binary address decoding on chip
- 5-V, 10-V, and 15-V parametric ratings
- 100% tested for quiescent current at 20 V
- Standardized symmetrical output characteristics
- Maximum input current of 1 μA at 18 V over full package temperature range; 100 nA at 18 V and 25°C
- Meets all requirements of JEDEC Tentative **Standard No. 13B**, "Standard Specifications for Description of 'B' Series CMOS Devices"

Applications:

- Analog and digital multiplexing and demultiplexing
- A/D and D/A conversion
- Signal gating





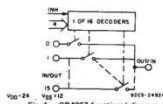


Fig. 1 – CD4067 functional diagram.

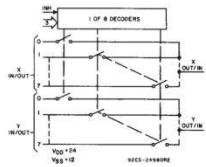


Fig. 2 – CD4097 functional diagram.

Recommended Operating Conditions at T_A = 25°C (Unless Otherwise Specified)

For maximum reliability, nominal operating conditions should be selected so that operation is always within the following ranges. Values shown apply to all types except as noted.

Characteristic	Min.	Max.	Units
Supply-Voltage Range (T_A=Full Package-Temp. Range)	3	18	V
Multiplexer Switch Input Current Capability	–	25	mA
Output Load Resistance	100	–	Ω

NOTE:

In certain applications, the external load-resistor current may include both V_{DD} and signal-line components. To avoid drawing V_{DD} current when switch current flows into the transmission gate inputs, the voltage drop across the bidirectional switch must not exceed 0.8 volt (calculated from R_{ON} values shown in ELECTRICAL CHARACTERISTICS CHART). No V_{DD} current will flow through R_L if the switch current flows into terminal 1 on the CD4067; terminals 1 and 17 on the CD4097.

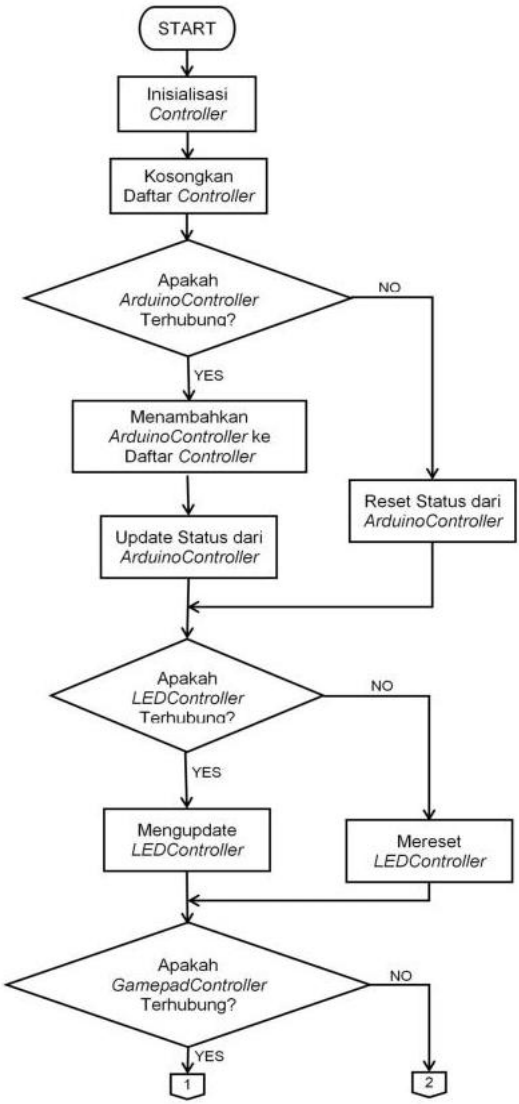
CD4067 TRUTH TABLE

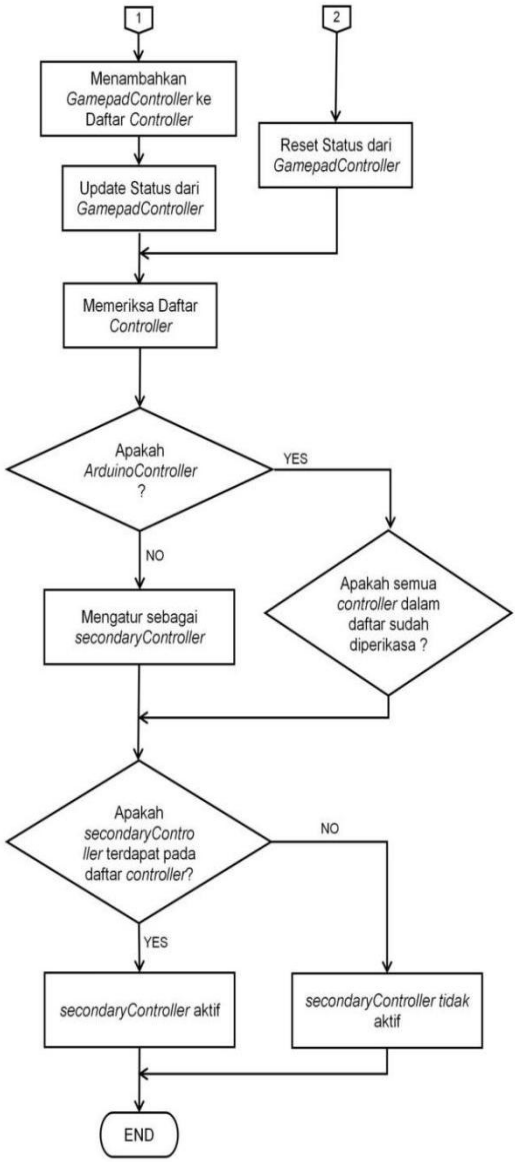
A	B	C	D	Inh	Selected Channel
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

CD4097 TRUTH TABLE

A	B	C	Inh	Selected Channel
X	X	X	1	None
0	0	0	0	0X, 0Y
1	0	0	0	1X, 1Y
0	1	0	0	2X, 2Y
1	1	0	0	3X, 3Y
0	0	1	0	4X, 4Y
1	0	1	0	5X, 5Y
0	1	1	0	6X, 6Y
1	1	1	0	7X, 7Y

Lampiran 4. *Flowchart* Program Antarmuka OCU Rhino Robot




1
2
Menambahkan GamepadController ke Daftar Controller
Reset Status dari GamepadController
Update Status dari GamepadController
Memeriksa Daftar Controller
Apakah ArduinoController ?
YES
NO
Apakah semua controller dalam daftar sudah diperikasa ?
Mengatur sebagai secondaryController
Apakah secondaryContro ller terdapat pada daftar controller?
NO
YES
secondaryController aktif
secondaryController tidak aktif
END

Lampiran 5. *Listing Program GamepadController.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Windows.Threading;
using SlimDX.XInput;

namespace RGUI_PS
{
    public class GamepadController
    {
        #region Initialization
        public GamepadController(UserIndex player)
        {
            gamepad = new Controller(player);
        }
        #endregion

        #region Field
        private Controller gamepad;
        private JOYSTICK leftJoystick, rightJoystick;
        private List<GAMEPAD_BUTTON> currentPressedButton = new
List<GAMEPAD_BUTTON>();
        private List<GAMEPAD_BUTTON> previousPressedButton = new
List<GAMEPAD_BUTTON>();
        #endregion

        #region Properties
        public JOYSTICK LeftJoystick { get { return leftJoystick; } }
        public JOYSTICK RightJoystick { get { return
rightJoystick; } }
        #endregion

        #region Methods
        public bool CheckConnection()
        {
```

```
            try
            {
                if (gamepad.IsConnected)
                    return true;
                else
                    return false;
            }
            catch (Exception)
            {
                return false;
            }

        }

        public bool isNewPressedButton(GAMEPAD_BUTTON button)
        {
            if (!previousPressedButton.Contains(button) &&
currentPressedButton.Contains(button))
                return true;
            else
                return false;
        }

        public bool isPressedButton(GAMEPAD_BUTTON button)
        {
            if (currentPressedButton.Contains(button))
                return true;
            else
                return false;
        }

        public void UpdateState()
        {
            //Analog Data
            leftJoystick.X = gamepad.GetState().Gamepad.LeftThumbX;
//range: -32768 to 32767
            leftJoystick.Y = gamepad.GetState().Gamepad.LeftThumbY;
//range: -32768 to 32767
            leftJoystick.Z =
gamepad.GetState().Gamepad.LeftTrigger; //range: 0 to 255
            rightJoystick.X =
gamepad.GetState().Gamepad.RightThumbX; //range: -32768 to 32767
            rightJoystick.Y =
gamepad.GetState().Gamepad.RightThumbY; //range: -32768 to 32767
            rightJoystick.Z =
gamepad.GetState().Gamepad.RightTrigger; //range: 0 to 255

            //Button Data
```

```
            GamepadButtonFlags buttonFlags = new
GamepadButtonFlags();//|Y|X|B|A|-|-|RS|LS|THUMB_R|THUMB_L|BACK|S
TART|DPAD_R|DPAD_L|DPAD_D|DPAD_U
            buttonFlags = gamepad.GetState().Gamepad.Buttons;

            previousPressedButton = currentPressedButton.ToList();
            currentPressedButton.Clear();

            if (((short)buttonFlags & 0x0001) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.DPAD_U);
            if (((short)buttonFlags & 0x0002) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.DPAD_D);
            if (((short)buttonFlags & 0x0004) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.DPAD_L);
            if (((short)buttonFlags & 0x0008) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.DPAD_R);
            if (((short)buttonFlags & 0x0010) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.START);
            if (((short)buttonFlags & 0x0020) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.BACK);
            if (((short)buttonFlags & 0x0040) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.THUMB_L);
            if (((short)buttonFlags & 0x0080) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.THUMB_R);
            if (((short)buttonFlags & 0x0100) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.LS);
            if (((short)buttonFlags & 0x0200) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.RS);
            if (((short)buttonFlags & 0x1000) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.A);
            if (((short)buttonFlags & 0x2000) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.B);
            if (((short)buttonFlags & 0x4000) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.X);
            if (((short)buttonFlags & 0x8000) != 0)
currentPressedButton.Add(GAMEPAD_BUTTON.Y);
        }

        public void ResetState()
        {
            //Analog Data
            leftJoystick.X = 0;
            leftJoystick.Y = 0;
            leftJoystick.Z = 0;
            rightJoystick.X = 0;
            rightJoystick.Y = 0;
            rightJoystick.Z = 0;
```

```
            //Button Data
            currentPressedButton.Clear();
            previousPressedButton = currentPressedButton.ToList();
        }
        #endregion
    }
}
```

Lampiran 6. *Listing program ArduinoController.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Windows.Threading;
using LattePanda.Firmata;

namespace RGUI_PS
{
    public class ArduinoController
    {
        #region Initialitation
        public ArduinoController()
        {
            Connect();
        }
        #endregion

        #region Field
        private Arduino arduino;
        private JOYSTICK leftJoystick, rightJoystick;
        private const int dataJoystick = 512;
        private byte activeState = Arduino.HIGH;
        private List<ARDUINO_BUTTON> currentPressedButton = new
List<ARDUINO_BUTTON>();
        private List<ARDUINO_BUTTON> previousPressedButton = new
List<ARDUINO_BUTTON>();

        //Input Pin : Analog
        private const int lJoystick1 = 0; //X Axis
        private const int lJoystick2 = 1; //Y Axis
        private const int lJoystick3 = 2; //Z Axis
        private const int rJoystick1 = 5; //X Axis
        private const int rJoystick2 = 4; //Y Axis
        private const int rJoystick3 = 3; //Z Axis
```

```
        //Input Pin : Digital Buttons
        private const int MUXPin_Input1 = 4;
        private const int MUXPin_Input2 = 5;
        private const int MUXPin_Input3 = 6;

        //Output Pin : Digital Buttons
        private const int MUXPin_Select1 = 0; //Selector A
        private const int MUXPin_Select2 = 1; //Selector B
        private const int MUXPin_Select3 = 2; //Selector C
        private const int MUXPin_Select4 = 3; //Selector D

        //Input Pin : Priority Digital Buttons
        private const int Priority1 = 7; //Emergency Button
        private const int Priority2 = 9; //Left Joystick Button
        private const int Priority3 = 8; //Right Joystick Button
        #endregion

        #region Properties
        public JOYSTICK LeftJoystick { get { return leftJoystick; } }
        public JOYSTICK RightJoystick { get { return
rightJoystick; } }
        #endregion

        #region Methods
        public bool Connect()
        {
            try
            {
                arduino = new Arduino();
                arduino.pinMode(MUXPin_Select1, Arduino.OUTPUT);
                arduino.pinMode(MUXPin_Select2, Arduino.OUTPUT);
                arduino.pinMode(MUXPin_Select3, Arduino.OUTPUT);
                arduino.pinMode(MUXPin_Select4, Arduino.OUTPUT);
                arduino.pinMode(MUXPin_Input1, Arduino.INPUT);
                arduino.pinMode(MUXPin_Input2, Arduino.INPUT);
                arduino.pinMode(MUXPin_Input3, Arduino.INPUT);
                arduino.pinMode(Priority1, Arduino.INPUT);
                arduino.pinMode(Priority2, Arduino.INPUT);
                arduino.pinMode(Priority3, Arduino.INPUT);
                return true;
            }
            catch (Exception)
            {
                return false;
            }

        }
```

```
        public bool CheckConnection()
        {
            try
            {
                if (arduino.IsConnected())
                    return true;
                else
                    return false;
            }
            catch (Exception)
            {
                return false;
            }
        }

        public bool isNewPressedButton(ARDUINO_BUTTON button)
        {
            if (!previousPressedButton.Contains(button) &&
currentPressedButton.Contains(button))
                return true;
            else
                return false;
        }

        public bool isPressedButton(ARDUINO_BUTTON button)
        {
            if (currentPressedButton.Contains(button))
                return true;
            else
                return false;
        }

        public void UpdateState()
        {
            //Analog Data
            leftJoystick.X = (arduino.analogRead(lJoystick1) -
dataJoystick);
            leftJoystick.Y = (arduino.analogRead(lJoystick2) -
dataJoystick);
            leftJoystick.Z = (arduino.analogRead(lJoystick3) -
dataJoystick);
            rightJoystick.X = (arduino.analogRead(rJoystick1) -
dataJoystick);
            rightJoystick.Y = (arduino.analogRead(rJoystick2) -
dataJoystick);
            rightJoystick.Z = (arduino.analogRead(rJoystick3) -
dataJoystick);
```

```
            //Button Data
            //Digital Buttons
            previousPressedButton = currentPressedButton.ToList();
            currentPressedButton.Clear();

            //Data ke-0
            arduino.digitalWrite(MUXPin_Select1, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select2, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select3, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select4, Arduino.LOW);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.MENU_TOGGLE); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.TURNTABLE_CC); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_ZOOM_IN); }

            //Data ke-1
            arduino.digitalWrite(MUXPin_Select1, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select2, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select3, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select4, Arduino.LOW);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.MENU_ENTER); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.TURNTABLE_C); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_ZOOM_OUT); }

            //Data ke-2
            arduino.digitalWrite(MUXPin_Select1, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select2, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select3, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select4, Arduino.LOW);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.MENU_BACK); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.SHOULDER_CC); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_MAIN); }

            //Data ke-3
            arduino.digitalWrite(MUXPin_Select1, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select2, Arduino.HIGH);
```

```
            arduino.digitalWrite(MUXPin_Select3, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select4, Arduino.LOW);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.MENU_LEFT); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.SHOULDER_C); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_GRIPPER); }

            //Data ke-4
            arduino.digitalWrite(MUXPin_Select1, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select2, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select3, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select4, Arduino.LOW);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.MENU_UP); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.ELBOW_CC); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_FRONT); }

            //Data ke-5
            arduino.digitalWrite(MUXPin_Select1, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select2, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select3, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select4, Arduino.LOW);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.MENU_DOWN); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.ELBOW_C); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_REAR); }

            //Data ke-6
            arduino.digitalWrite(MUXPin_Select1, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select2, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select3, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select4, Arduino.LOW);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.MENU_RIGHT); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.WRIST_CC); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_PRESET1); }
```

```
            //Data ke-7
            arduino.digitalWrite(MUXPin_Select1, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select2, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select3, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select4, Arduino.LOW);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.POSE_PRESET1); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.WRIST_C); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_PRESET2); }

            //Data ke-8
            arduino.digitalWrite(MUXPin_Select1, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select2, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select3, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select4, Arduino.HIGH);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.POSE_PRESET2); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_TILT_CC); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_PRESET3); }

            //Data ke-9
            arduino.digitalWrite(MUXPin_Select1, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select2, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select3, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select4, Arduino.HIGH);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.POSE_PRESET3); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_TILT_C); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.LAMP_SPEAKER_TOGGLE); }

            //Data ke-10
            arduino.digitalWrite(MUXPin_Select1, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select2, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select3, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select4, Arduino.HIGH);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.POSE_PRESET4); }
```

```
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_PAN_CC); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.LAMP_SPEAKER_INCREASE);
 }

            //Data ke-11
            arduino.digitalWrite(MUXPin_Select1, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select2, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select3, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select4, Arduino.HIGH);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.POSE_PRESET5); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.CAMERA_PAN_C); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.LAMP_SPEAKER_DECREASE);
 }

            //Data ke-12
            arduino.digitalWrite(MUXPin_Select1, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select2, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select3, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select4, Arduino.HIGH);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.POSE_PRESET6); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.GRIPPER_ROTATION_CC); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.LAMP_MAIN); }

            //Data ke-13
            arduino.digitalWrite(MUXPin_Select1, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select2, Arduino.LOW);
            arduino.digitalWrite(MUXPin_Select3, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select4, Arduino.HIGH);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input1) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.SPEED_CONTROL); }
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.GRIPPER_ROTATION_C); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.LAMP_GRIPPER); }

            //Data ke-14
            arduino.digitalWrite(MUXPin_Select1, Arduino.LOW);
```

```
            arduino.digitalWrite(MUXPin_Select2, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select3, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select4, Arduino.HIGH);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.GRIPPER_GRASP_OPEN); }
            if (arduino.digitalRead(MUXPin_Input3) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.LAMP_FRONT); }

            //Data ke-15
            arduino.digitalWrite(MUXPin_Select1, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select2, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select3, Arduino.HIGH);
            arduino.digitalWrite(MUXPin_Select4, Arduino.HIGH);
            Thread.Sleep(10);
            if (arduino.digitalRead(MUXPin_Input2) == activeState)
{ currentPressedButton.Add(ARDUINO_BUTTON.GRIPPER_GRASP_CLOSE); }

            //Priority Digital Buttons
            if (arduino.digitalRead(Priority1) == Arduino.LOW)
{ currentPressedButton.Add(ARDUINO_BUTTON.EMERGENCY); }
            if (arduino.digitalRead(Priority2) == Arduino.LOW)
{ currentPressedButton.Add(ARDUINO_BUTTON.LEFT_JOYSTICK); }
            if (arduino.digitalRead(Priority3) == Arduino.LOW)
{ currentPressedButton.Add(ARDUINO_BUTTON.RIGHT_JOYSTICK); }
        }

        public void ResetState()
        {
            //Analog Data
            leftJoystick.X = 0;
            leftJoystick.Y = 0;
            leftJoystick.Z = 0;
            rightJoystick.X = 0;
            rightJoystick.Y = 0;
            rightJoystick.Z = 0;

            //Button Data
            currentPressedButton.Clear();
            previousPressedButton = currentPressedButton.ToList();
        }
        #endregion
    }
}
```

Lampiran 7. *Listing program OCUControlCenter.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SlimDX.XInput;
using LattePanda.Firmata;

namespace RGUI_PS
{
    public class OCUControlCenter
    {
        #region Initialization
        public OCUControlCenter()
        {
            controller = new List<OCU_CONTROLLER>();
            secondarycontroller = new AuxiliaryControl();
            gamepad = new GamepadController(UserIndex.One);
            arduino = new ArduinoController();
            led = new LEDController();
        }
        #endregion

        #region Field
        private static List<OCU_CONTROLLER> controller;
        private static AuxiliaryControl secondarycontroller;
        private static GamepadController gamepad;
        private static ArduinoController arduino;
        private static LEDController led;
        #endregion

        #region Properties
        public GamepadController Gamepad { get { return gamepad; } }
        public ArduinoController Arduino { get { return arduino; } }
        public AuxiliaryControl SecondaryController { get { return
secondarycontroller; } }
        public LEDController Led { get { return led; } }
        #endregion

        #region Methods
        public bool IsNoneConnected() { return (controller.Count ==
0) ? true : false; }
        public bool IsConnected(OCU_CONTROLLER Device) { return
controller.Contains(Device); }
```

```
public void CheckController()
{
    controller = new List<OCU_CONTROLLER>();

    #region Main Controller
    if (arduino.CheckConnection())
    {
        arduino.UpdateState();
        controller.Add(OCU_CONTROLLER.ARDUINO);
    }
    else
    {
        if (arduino.Connect())
        {
            arduino.UpdateState();
            controller.Add(OCU_CONTROLLER.ARDUINO);
        }
        else
        {
            arduino.ResetState();
        }
    }
    #endregion

    #region Secondary Controller
    if (gamepad.CheckConnection())
    {
        gamepad.UpdateState();
        controller.Add(OCU_CONTROLLER.GAMEPAD);
    }
    else
    {
        gamepad.ResetState();
    }

    foreach (OCU_CONTROLLER item in controller)
    {
        if (item != OCU_CONTROLLER.ARDUINO)
        {
            secondarycontroller.Device = item;
            break;
        }
    }
    if (controller.Contains(secondarycontroller.Device))
    {
        if (!secondarycontroller.IsActive)
        {
            secondarycontroller.IsActive = true;
```

```
            }
        }
        else
        {
            if(secondarycontroller.IsActive)
            {
                secondarycontroller.IsActive = false;
            }
        }
        #endregion

        #region LED Controller
        if (led.IsConnected)
        {
            led.UpdateState();
        }
        else
        {
            led.ResetState();
        }
        #endregion
    }
    #endregion
}
```

Lampiran 8. *Listing program LEDController.cs*

```
using System;
using System.Collections.Generic;
using System.IO.Ports;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RGUI_PS
{
    public class LEDController
    {
        #region Initialization
        public LEDController()
        {
            serialPort = new SerialPort();

            serialPort.PortName = "COM1";
            serialPort.BaudRate = 9600;
            serialPort.Parity = Parity.None;
            serialPort.DataBits = 8;
            serialPort.StopBits = StopBits.One;
            serialPort.Handshake = Handshake.None;
            serialPort.ReadTimeout = 100;
            serialPort.WriteTimeout = 500;

            serialPort.Open();

            byte[] buffer_byte = new byte[5];
            buffer_byte[0] = header;
            buffer_byte[1] = dataSize;
        }
        #endregion

        #region Fields
        private SerialPort serialPort;

        private byte[] buffer_byte;
        private byte operation_byte, setting_byte, CRC_byte;

        private const int bufferSize = 5;
        private const int dataSize = 2;
        private const byte header = 0x40;
        private const byte maskMenuToggle = 0x01;
        private const byte maskMenuStatus = 0x02;
        private const byte maskEmergency = 0x10;
```

```
private const byte maskNormal = 0x20;
private const byte maskIK = 0x40;
private const byte maskLampStatus = 0x01;
private const byte maskLampMain = 0x02;
private const byte maskLampGripper = 0x04;
private const byte maskLampFront = 0x08;
private const byte maskCamMain = 0x10;
private const byte maskCamGripper = 0x20;
private const byte maskCamFront = 0x40;
private const byte maskCamRear = 0x80;
#endregion

#region Properties
public bool IsConnected
{
    get
    {
        if (serialPort != null)
            return serialPort.IsOpen;
        else
            return false;
    }
}
#endregion

#region Methods
public void Connect()
{
    try
    {
        serialPort.Open();
    }
    catch (Exception)
    {

    }
}

public void UpdateState()
{
    buffer_byte[2] = operation_byte;
    buffer_byte[3] = setting_byte;

    CRC_byte = (byte) (operation_byte ^ setting_byte);
    buffer_byte[4] = CRC_byte;

    serialPort.Write(buffer_byte, 0, bufferSize);
}
```

```
        public void ResetState()
        {
            operation_byte = 0x00;
            setting_byte = 0x00;
            CRC_byte = 0x00;
        }

        public void SetBit_MenuToggle() { operation_byte |=
maskMenuToggle; }
        public void SetBit_MenuStatus() { operation_byte |=
maskMenuStatus; }
        public void SetBit_Emergency() { operation_byte |=
maskEmergency; }
        public void SetBit_Normal() { operation_byte |=
maskNormal; }
        public void SetBit_IK() { operation_byte |= maskIK; }
        public void SetBit_LampStatus() { setting_byte |=
maskLampStatus; }
        public void SetBit_LampMain() { setting_byte |=
maskLampMain; }
        public void SetBit_LampGripper() { setting_byte |=
maskLampGripper; }
        public void SetBit_LampFront() { setting_byte |=
maskLampFront; }
        public void SetBit_CamMain() { setting_byte |=
maskCamMain; }
        public void SetBit_CamGripper() { setting_byte |=
maskCamGripper; }
        public void SetBit_CamFront() { setting_byte |=
maskCamFront; }
        public void SetBit_CamRear() { setting_byte |=
maskCamRear; }

        public void ResetBit_MenuToggle() { operation_byte &=
unchecked((byte)~maskMenuToggle); }
        public void ResetBit_MenuStatus() { operation_byte &=
unchecked((byte)~maskMenuStatus); }
        public void ResetBit_Emergency() { operation_byte &=
unchecked((byte)~maskEmergency); }
        public void ResetBit_Normal() { operation_byte &=
unchecked((byte)~maskNormal); }
        public void ResetBit_IK() { operation_byte &=
unchecked((byte)~maskIK); }
        public void ResetBit_LampStatus() { setting_byte &=
unchecked((byte)~maskLampStatus); }
        public void ResetBit_LampMain() { setting_byte &=
unchecked((byte)~maskLampMain); }
```

```
        public void ResetBit_LampGripper() { setting_byte &=
unchecked((byte)~maskLampGripper); }
        public void ResetBit_LampFront() { setting_byte &=
unchecked((byte)~maskLampFront); }
        public void ResetBit_CamMain() { setting_byte &=
unchecked((byte)~maskCamMain); }
        public void ResetBit_CamGripper() { setting_byte &=
unchecked((byte)~maskCamGripper); }
        public void ResetBit_CamFront() { setting_byte &=
unchecked((byte)~maskCamFront); }
        public void ResetBit_CamRear() { setting_byte &=
unchecked((byte)~maskCamRear); }
        #endregion
    }
}
```

Lampiran 9. *Listing program PageInputSystem.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RGUI_PS
{
    public class PageInputSystem
    {
        #region Initialization
        public PageInputSystem(OCUControlCenter ControlCenter)
        {
            controlCenter = ControlCenter;
        }
        #endregion

        #region Fields
        private OCUControlCenter controlCenter;
        #endregion

        #region Properties
        public bool MenuToggle
        {
            get
            {
                return
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.RS)
                    ||
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.MENU_TOG
GLE));
            }
        }

        public bool MenuEnter
        {
            get
            {
                return
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.A)
                    ||
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.MENU_ENT
ER));
            }
        }
```

```
        public bool MenuBack
        {
            get
            {
                return
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.BACK)
                    ||
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.MENU_BAC
K));
            }
        }

        public bool MenuUp
        {
            get
            {
                return
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.DPAD_U)
                    ||
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.MENU_UP))
;
            }
        }

        public bool MenuDown
        {
            get
            {
                return
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.DPAD_D)
                    ||
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.MENU_DOW
N));
            }
        }

        public bool MenuRight
        {
            get
            {
                return
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.DPAD_R)
                    ||
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.MENU_RIG
HT));
            }
        }
```

```
        public bool MenuLeft
        {
            get
            {
                return
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.DPAD_L)
                    ||
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.MENU_LEF
T));
            }
        }
        #endregion

        #region Methods
        public void ToggleLED_Emergency(bool IsSet)
        {
            if (IsSet)
                controlCenter.Led.SetBit_Emergency();
            else
                controlCenter.Led.ResetBit_Emergency();
        }
        public void ToggleLED_MenuToggle(bool IsSet)
        {
            if (IsSet)
                controlCenter.Led.SetBit_MenuToggle();
            else
                controlCenter.Led.ResetBit_MenuToggle();
        }
        public void ToggleLED_MenuStatus(bool IsSet)
        {
            if (IsSet)
                controlCenter.Led.SetBit_MenuStatus();
            else
                controlCenter.Led.ResetBit_MenuStatus();
        }
        public void ToggleLED_CameraMain(bool IsSet)
        {
            if (IsSet)
                controlCenter.Led.SetBit_CamMain();
            else
                controlCenter.Led.ResetBit_CamMain();
        }
        public void ToggleLED_CameraGripper(bool IsSet)
        {
            if (IsSet)
                controlCenter.Led.SetBit_CamGripper();
            else
```

```
                controlCenter.Led.ResetBit_CamGripper();
            }
            public void ToggleLED_CameraFront(bool IsSet)
            {
                if (IsSet)
                    controlCenter.Led.SetBit_CamFront();
                else
                    controlCenter.Led.ResetBit_CamFront();
            }
            public void ToggleLED_CameraRear(bool IsSet)
            {
                if (IsSet)
                    controlCenter.Led.SetBit_CamRear();
                else
                    controlCenter.Led.ResetBit_CamRear();
            }
            #endregion
        }
    }
```

Lampiran 10. *Listing program RhinoInputSystem.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RGUI_PS
{
    public class RhinoInputSystem
    {
        #region Initialization
        public RhinoInputSystem(OCUControlCenter ControlCenter)
        {
            controlCenter = ControlCenter;
        }
        #endregion

        #region Fields
        private OCUControlCenter controlCenter;
        private List<TimeToggleSpeed> timetogglespeed = new
List<TimeToggleSpeed>();
        private TimeToggleSpeed brakestate;
        private TimeToggleSpeed drivetrainstate;
        private TimeToggleSpeed firedisruptorstate;
        private const int timehold = 5;
        private const int buttonispressed = 3;
        private int timeawaitbrake = 0;
        private int buttonpressed = 0;
        #endregion

        #region Properties
        public bool PosePreset1
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.POSE_PRE
SET1); }
        }
        public bool PosePreset2
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.POSE_PRE
SET2); }
        }
        public bool PosePreset3
        {
```

```
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.POSE_PRE
SET3); }
        }
        public bool PosePreset4
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.POSE_PRE
SET4); }
        }
        public bool PosePreset5
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.POSE_PRE
SET5); }
        }
        public bool PosePreset6
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.POSE_PRE
SET6); }
        }

        public bool LampMain
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.LAMP_MAI
N); }
        }
        public bool LampGripper
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.LAMP_GRI
PPER); }
        }
        public bool LampFront
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.LAMP_FRO
NT); }
        }
        public bool LampSpeakerToggle
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.LAMP_SPE
AKER_TOGGLE); }
        }
        public bool LampSpeakerIncrease
```

```
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.LAMP_SPE
AKER_INCREASE); }
        }
        public bool LampSpeakerDecrease
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.LAMP_SPE
AKER_DECREASE); }
        }

        public bool SpoolerExtend
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.SPOOLER_
EXTEND); }
        }
        public bool SpoolerCollect
        {
            get { return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.SPOOLER_
COLLECT); }
        }

        public short DrivetrainX
        {
            get
            {
                if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.DRIVETRAIN) &&
(controlCenter.Gamepad.LeftJoystick.X != 0))
                    return
(short)controlCenter.Gamepad.LeftJoystick.X;
                else
                    return
(short)controlCenter.Arduino.RightJoystick.X;
            }
        }

        public short DrivetrainY
        {
            get
            {
                if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.DRIVETRAIN) &&
(controlCenter.Gamepad.LeftJoystick.Y != 0))
```

```
                return
(short)controlCenter.Gamepad.LeftJoystick.Y;
            else
                return
(short)controlCenter.Arduino.RightJoystick.Y;
        }
    }

    public short DrivetrainFlipper
    {
        get
        {
            if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.DRIVETRAIN) &&
(controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.LS)))
                return byte.MaxValue;
            else if
((controlCenter.SecondaryController.CurrentMode ==
ControlMode.DRIVETRAIN) &&
(controlCenter.Gamepad.LeftJoystick.Z != 0) &&
(!controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.LS)))
                return
(short)controlCenter.Gamepad.LeftJoystick.Z;
            else
                return
(short)controlCenter.Arduino.RightJoystick.Z;
        }
    }

    public short ManipulatorX
    {
        get
        {
            if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.LeftJoystick.X != 0))
                return
(short)controlCenter.Gamepad.LeftJoystick.X;
            else
                return
(short)controlCenter.Arduino.RightJoystick.X;
        }
    }

    public short ManipulatorY
    {
        get
        {
```

```
            if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.LeftJoystick.Y != 0))
                return
(short)controlCenter.Gamepad.LeftJoystick.Y;
            else
                return
(short)controlCenter.Arduino.RightJoystick.Y;
        }
    }

    public short ManipulatorZ
    {
        get
        {
            if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(!controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.DPAD_U))
&&
(controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.DPAD_D)))
                return short.MinValue;
            else if
((controlCenter.SecondaryController.CurrentMode ==
ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.DPAD_U)))
                return short.MaxValue;
            else
                return
(short)controlCenter.Arduino.LeftJoystick.Z;
        }
    }

    public bool ToggleManipulatorMode
    {
        get
        {
            if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.THUMB_L)
))
                return
controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.THUMB_L)
;
            else
                return
controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.LEFT_JOY
STICK);
        }
```

```
        }

        public bool ToggleDrivetrainMode
        {
            get
            {
                bool MainValue = false;
                bool SecondaryValue = false;

                switch (drivetrainstate)
                {
                    case TimeToggleSpeed.AWAIT:
                        {
                            if
(controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.RIGHT_J
OYSTICK))
                            {
                                drivetrainstate =
TimeToggleSpeed.HOLD;
                            }
                            break;
                        }
                    case TimeToggleSpeed.HOLD:
                        {
                            if
(controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.RIGHT_JOYS
TICK))
                            {
                                timeawaitbrake++;
                                if (timeawaitbrake >= timehold)
                                {
                                    drivetrainstate =
TimeToggleSpeed.AWAIT;
                                    timeawaitbrake = 0;
                                }
                            }

                            else
                            {
                                drivetrainstate =
TimeToggleSpeed.TOGGLED;
                                timeawaitbrake = 0;
                            }
                            break;
                        }
                    case TimeToggleSpeed.TOGGLED:
                        {
                            MainValue = true;
```

```
                        drivetrainstate = TimeToggleSpeed.AWAIT;
                        break;
                    }
                default:
                    break;
            }

            if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.DRIVETRAIN) &&
(controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.THUMB_L)
))
                SecondaryValue =
controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.THUMB_L)
;

            return MainValue || SecondaryValue;
        }
    }

    public bool ToggleBrake
    {
        get
        {
            bool MainValue = false;
            bool SecondaryValue = false;

            switch (brakestate)
            {
                case TimeToggleSpeed.AWAIT:
                    {
                        if
(controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.RIGHT_J
OYSTICK))
                        {
                            brakestate = TimeToggleSpeed.HOLD;
                        }
                        break;
                    }
                case TimeToggleSpeed.HOLD:
                    {
                        if
(controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.RIGHT_JOYS
TICK))
                        {
                            timeawaitbrake++;
                            if (timeawaitbrake == timehold)
                            {
```

```
                                brakestate =
TimeToggleSpeed.TOGGLED;
                            }
                        }
                        else
                        {
                            brakestate = TimeToggleSpeed.AWAIT;
                            timeawaitbrake = 0;
                        }
                        break;
                    }
                case TimeToggleSpeed.TOGGLED:
                    {
                        MainValue = true;
                        brakestate = TimeToggleSpeed.AWAIT;
                        break;
                    }
                default:
                    break;
            }

            if
(controlCenter.SecondaryController.CurrentMode !=
ControlMode.NONE)
                SecondaryValue =
controlCenter.Gamepad.isNewPressedButton(GAMEPAD_BUTTON.BACK);

            return MainValue || SecondaryValue;
        }
    }

    public bool ToggleSpeed
    {
        get
        {
            if
((controlCenter.SecondaryController.CurrentMode !=
ControlMode.NONE) &&
(controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.Y)))
                return
controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.Y);
            else
                return false;
        }
    }

    public bool FireDisruptor
    {
```

```
            get
            {
                bool MainValue = false;
                bool SecondaryValue = false;

                switch (firedisruptorstate)
                {
                    case TimeToggleSpeed.AWAIT:
                        {
                            if
(controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.DISRUPT
OR_TRIGGER))
                            {
                                firedisruptorstate =
TimeToggleSpeed.HOLD;
                            }
                            break;
                        }
                    case TimeToggleSpeed.HOLD:
                        {
                            if
(controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.DISRUPTOR_
TRIGGER))
                            {
                                if
(controlCenter.Arduino.isNewPressedButton(ARDUINO_BUTTON.DISRUPT
OR_FIRE))
                                {
                                    buttonpressed++;
                                }
                                if (buttonpressed==buttonispressed)
                                {
                                    firedisruptorstate =
TimeToggleSpeed.TOGGLED;
                                }
                            }
                            else
                            {
                                firedisruptorstate =
TimeToggleSpeed.AWAIT;
                                buttonpressed = 0;
                            }
                            break;
                        }
                    case TimeToggleSpeed.TOGGLED:
                        {
                            MainValue = true;
```

```
                        firedisruptorstate =
TimeToggleSpeed.AWAIT;
                        break;
                    }
                default:
                    break;
            }

            return MainValue || SecondaryValue;
        }
    }

    public bool TurntableCC
    {
        get { return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.TURNTABLE_C
C); }
    }
    public bool TurntableC
    {
        get { return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.TURNTABLE_C)
; }
    }
    public bool ShoulderCC
    {
        get { return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.SHOULDER_CC)
; }
    }
    public bool ShoulderC
    {
        get { return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.SHOULDER_C)
; }
    }
    public bool ElbowCC
    {
        get { return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.ELBOW_CC); }
    }
    public bool ElbowC
    {
        get { return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.ELBOW_C); }
    }

    public bool GripperRotationCC
```

```
        {
            get
            {
                if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.DPAD_L)))
                    return
controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.DPAD_L);
                else
                    return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.GRIPPER_ROT
ATION_CC);
            }
        }

        public bool GripperRotationC
        {
            get
            {
                if((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.DPAD_R)))
                    return
controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.DPAD_R);
                else
                    return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.GRIPPER_ROT
ATION_C);
            }
        }

        public bool GripperGraspOpen
        {
            get
            {
                if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.RS)))
                    return
controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.RS);
                else
                    return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.GRIPPER_GRA
SP_OPEN);
            }
        }

        public bool GripperGraspClose
```

```
        {
            get
            {
                if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.RightJoystick.Z == byte.MaxValue))
                    return true;
                else
                    return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.GRIPPER_GRA
SP_CLOSE);
            }
        }

        public bool WristCC
        {
            get
            {
                if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.LS)))
                    return
controlCenter.Gamepad.isPressedButton(GAMEPAD_BUTTON.LS);
                else
                    return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.WRIST_CC);
            }
        }

        public bool WristC
        {
            get
            {
                if ((controlCenter.SecondaryController.CurrentMode
== ControlMode.MANIPULATOR) &&
(controlCenter.Gamepad.LeftJoystick.Z == byte.MaxValue))
                    return true;
                else
                    return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.WRIST_C);
            }
        }

        public bool CameraTiltCC
        {
            get
            {
```

```
            if
((controlCenter.SecondaryController.CurrentMode !=
ControlMode.NONE) && (controlCenter.Gamepad.RightJoystick.Y ==
short.MaxValue))
                return true;
            else
                return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.CAMERA_TILT
_CC);
        }
    }

    public bool CameraTiltC
    {
        get
        {
            if
((controlCenter.SecondaryController.CurrentMode !=
ControlMode.NONE) && (controlCenter.Gamepad.RightJoystick.Y ==
short.MinValue))
                return true;
            else
                return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.CAMERA_TILT
_C);
        }
    }

    public bool CameraPanCC
    {
        get
        {
            if
((controlCenter.SecondaryController.CurrentMode !=
ControlMode.NONE) && (controlCenter.Gamepad.RightJoystick.X ==
short.MaxValue))
                return true;
            else
                return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.CAMERA_PAN_
CC);
        }
    }

    public bool CameraPanC
    {
        get
        {
```

```
            if
((controlCenter.SecondaryController.CurrentMode !=
ControlMode.NONE) && (controlCenter.Gamepad.RightJoystick.X ==
short.MinValue))
                return true;
            else
                return
controlCenter.Arduino.isPressedButton(ARDUINO_BUTTON.CAMERA_PAN_
C);
        }
    }
    #endregion

    #region Methods
    public void ToggleLED_OpEmergency(bool IsSet)
    {
        if (IsSet)
            controlCenter.Led.SetBit_Emergency();
        else
            controlCenter.Led.ResetBit_Emergency();
    }
    public void ToggleLED_OpNormal(bool IsSet)
    {
        if (IsSet)
            controlCenter.Led.SetBit_Normal();
        else
            controlCenter.Led.ResetBit_Normal();
    }
    public void ToggleLED_OpIK(bool IsSet)
    {
        if (IsSet)
            controlCenter.Led.SetBit_IK();
        else
            controlCenter.Led.ResetBit_IK();
    }
    public void ToggleLED_LampStatus(bool IsSet)
    {
        if (IsSet)
            controlCenter.Led.SetBit_LampStatus();
        else
            controlCenter.Led.ResetBit_LampStatus();
    }
    public void ToggleLED_LampMain(bool IsSet)
    {
        if (IsSet)
            controlCenter.Led.SetBit_LampMain();
        else
            controlCenter.Led.ResetBit_LampMain();
```

```
        }
        public void ToggleLED_LampGripper(bool IsSet)
        {
            if (IsSet)
                controlCenter.Led.SetBit_LampGripper();
            else
                controlCenter.Led.ResetBit_LampGripper();
        }
        public void ToggleLED_LampFront(bool IsSet)
        {
            if (IsSet)
                controlCenter.Led.SetBit_LampFront();
            else
                controlCenter.Led.ResetBit_LampFront();
        }
        #endregion
    }
}
```

Lampiran 11. *Listing program Common.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace RGUI_PS
{
    #region Controller
    public enum OCU_CONTROLLER
    {
        ARDUINO = 1,
        GAMEPAD,
        NONE
    }

    public enum DrivetrainMode
    {
        NORMAL,
        TERBALIK
    }

    public enum ManipulatorMode
    {
        INSPEKSI,
        PENGGENGGAM
    }

    public enum ControlMode
    {
        NONE,
        DRIVETRAIN,
        MANIPULATOR
    }

    public enum TimeToggleSpeed
    {
        AWAIT,
        HOLD,
        TOGGLED
    }

    public enum GAMEPAD_BUTTON
```

```
{
    DPAD_U = 1,
    DPAD_D,
    DPAD_L,
    DPAD_R,
    START,
    BACK,
    THUMB_L,
    THUMB_R,
    LS,
    RS,
    A,
    B,
    X,
    Y
}

public enum ARDUINO_BUTTON
{
    EMERGENCY = 1,
    LEFT_JOYSTICK,
    RIGHT_JOYSTICK,

    MENU_TOGGLE,
    MENU_ENTER,
    MENU_BACK,
    MENU_UP,
    MENU_DOWN,
    MENU_LEFT,
    MENU_RIGHT,
    POSE_PRESET1,
    POSE_PRESET2,
    POSE_PRESET3,
    POSE_PRESET4,
    POSE_PRESET5,
    POSE_PRESET6,
    SPEED_CONTROL,

    TURNTABLE_CC,
    TURNTABLE_C,
    SHOULDER_CC,
    SHOULDER_C,
    ELBOW_CC,
    ELBOW_C,
    WRIST_CC,
    WRIST_C,
    CAMERA_TILT_CC,
    CAMERA_TILT_C,
```

```
    CAMERA_PAN_CC,
    CAMERA_PAN_C,
    GRIPPER_ROTATION_CC,
    GRIPPER_ROTATION_C,
    GRIPPER_GRASP_OPEN,
    GRIPPER_GRASP_CLOSE,

    CAMERA_ZOOM_IN,
    CAMERA_ZOOM_OUT,
    CAMERA_MAIN,
    CAMERA_GRIPPER,
    CAMERA_FRONT,
    CAMERA_REAR,
    CAMERA_PRESET1,
    CAMERA_PRESET2,
    CAMERA_PRESET3,
    LAMP_SPEAKER_TOGGLE,
    LAMP_SPEAKER_INCREASE,
    LAMP_SPEAKER_DECREASE,
    LAMP_MAIN,
    LAMP_GRIPPER,
    LAMP_FRONT,

    DISRUPTOR_TRIGGER,
    DISRUPTOR_FIRE,
    SPOOLER_EXTEND,
    SPOOLER_COLLECT
}

public struct JOYSTICK
{
    public int X;
    public int Y;
    public int Z;
}
#endregion

public class AuxiliaryControl
{
    #region Initialitation
    public AuxiliaryControl()
    {
        device = OCU_CONTROLLER.NONE;
        temp = ControlMode.DRIVETRAIN;
        currentMode = ControlMode.NONE;
    }
    #endregion
```

```
#region Field
private OCU_CONTROLLER device;
private bool isActive;
private ControlMode temp;
private ControlMode currentMode;
private ControlMode previousMode;
#endregion

#region Properties
public bool IsActive
{
    get { return isActive; }
    set
    {
        if (value == true)
            currentMode = temp;
        else
        {
            temp = currentMode;
            currentMode = ControlMode.NONE;
        }
        isActive = value;
    }
}

public OCU_CONTROLLER Device
{
    get { return device; }
    set
    {
        device = value;
    }
}

public ControlMode CurrentMode
{
    get { return currentMode; }
    set
    {
        if (value != currentMode)
            previousMode = currentMode;
        currentMode = value;
    }
}

public void BackToPreviousMode()
{
    if (isActive)
```

```
                currentMode = previousMode;
            }
            #endregion
        }
    }
```

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

RIWAYAT PENULIS

Livian Tjandra lahir di kota Jombang, provinsi Jawa Timur pada tanggal 13 Juni 1998. Penulis merupakan anak pertama dari pasangan Kastono dan Lilik Mahmudah. Setelah menamatkan pendidikan dari Sekolah Dasar hingga Sekolah Menengah Atas di daerah asal pada tahun 2015, penulis kemudian melanjutkan pendidikannya di Institut Teknologi Sepuluh Nopember. Selama menempuh pendidikan di perguruan tinggi, penulis tinggal di Keputih II C No. 22, Sukolilo.

E-mail : livlivian98@gmail.com