**TUGAS AKHIR – TE141599**

# PEMBIDIK OTOMATIS KUBAH SENJATA UNTUK KENDARAAN DARAT TANPA AWAK

**Fandi Afrizal**
**NRP 2213100175**

**Dosen Pembimbing**
**Ronny Mardiyanto, S.T., M.T., Ph.D.**
**Dr. Ir. Djoko Purwanto, M.Eng.**

**DEPARTEMEN TEKNIK ELEKTRO**
**Fakultas Teknologi Elektro**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2017**

**FINAL PROJECT – TE141599**

# AUTOMATIC AIMING WEAPON TURRET FOR UNMANNED GROUND VEHICLE

**Fandi Afrizal**
**NRP 2213100175**

**Supervisor**
**Ronny Mardiyanto, S.T., M.T., Ph.D.**
**Dr. Ir. Djoko Purwanto, M.Eng.**

**ELECTRICAL ENGINEERING DEPARTMENT**
**Faculty of Electrical Technology**
**Sepuluh Nopember Institute of Technology**
**Surabaya 2017**

**PEMBIDIK OTOMATIS KUBAH SENJATA UNTUK KENDARAAN DARAT TANPA AWAK**

# TUGAS AKHIR

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

Menyetujui

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Ronny Mardiyanto, ST., MT., Ph.D. | Dr. Ir. Djoko Purwanto, M.Eng. |
| NIP. 198101182003121003 | NIP. 196512111990021002 |

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO

**SURABAYA**
**JULI, 2017**

**PEMBIDIK OTOMATIS KUBAH SENJATA UNTUK KENDARAAN DARAT TANPA AWAK**

**Nama : Fandi Afrizal**
**Pembimbing I : Ronny Mardiyanto, S.T., M.T., Ph.D.**
**Pembimbing II : Dr. Ir. Djoko Purwanto, M.Eng.**

# ABSTRAK

Pembidik otomatis kubah senjata adalah sebuah sistem pembidik yang dapat mengikuti pergerakan target. Sistem ini akan dipergunakan untuk kepentingan militer, terutama akan dipasang pada kendaraan darat tanpa awak. Sistem pembidik otomatis ini merupakan inovasi pada sistem persenjataannya dimana pada umumnya sistem pembidik yang terpasang pada kendaraan darat tanpa awak hanya dapat dioperasikan secara manual. Pada Tugas Akhir ini, kami membuat sistem pembidik otomatis kubah senjata dengan memasang sebuah senjata softgun yang pergerakannya dikendalikan oleh dua buah motor servo untuk bergerak pan dan tilt. Kendaraan tanpa awak yang digunakan menggunakan prinsip differensial steering. Sistem ini diengkapi dengan dua kamera, (1) digunakan untuk melacak target dan (2) digunakan untuk navigasi. Metode *tracking* yang digunakan adalah metode *lucas kanade.* Metode ini membutuhkan satu kamera untuk menentukan posisi koordinat yang akan dikonversikan dalam gerak aktuator. Pemasangan Kamera dan senjata menggunakan satu robot yang dapat digerakkan sesuai keinginan user. Penguncian posisi target tetap dilakukan saat UGV bergerak. Aktuator yang menggerakkan dudukan senjata ini menggunakan servo yang dapat diatur sudutnya. Hasil pengujian yang dilakukan pada Tugas Akhir ini adalah sistem dapat mengunci posisi target yang dipilih dan dapat mengarahkan senjata ke posisi tersebut dengan memasukkan kontroller PI dalam sistem ini. Ketepatan terbaik adalah menggunakan kontrol proportional dengan nilai 0.04 dan 0.08 dan kontrol integral dengan nilai 0.005 dan 0.025 yaitu dengan *rise time* sekitar 2 detik, *overshoot* sekitar 10 pixel dan *error steady state* sekitar 1 pixel. Pengujian ketepatan didapatkan *error* pada pengujian dengan jarak 170 dan 290cm dari robot. Pada pengujian menggunakan intensitas cahaya didapatkan *error* dengan nilai dibawah 15 lux dan diatas 2000 lux. Jika dipenguruhi oleh faktor kecepatan, sistem akan *error* ketika kecepatan diatas 200 pwm.
Kata kunci: *UGV*, *tracking camera*, teknologi senjata

*Halaman ini sengaja dikosongkan*

***AUTOMATIC AIMING WEAPON TURRET FOR UNMANNED GROUND VEHICLE***


| | |
|---|---|
| ***Name*** | **: Fandi Afrizal** |
| ***Supervisor*** | **: Ronny Mardiyanto, S.T., M.T., Ph.D.** |
| ***Co-Supervisor*** | **: Dr. Ir. Djoko Purwanto, M.Eng.** |


## *ABSTRACT*


The automatic weapon dome shoot is a sifting system that can follow the target movement. This system will be used for military purposes, especially will be installed on land vehicles without waking up. This automatic sighting system is a system in an armament system wherein a built-in surveillance system on a ground vehicle can only be operated manually. In this final project, we make weapon system by making gun gun which is controlled by two servo motors to move pan and tilt. Unmanned vehicles that use steering difference principles. The system is equipped with two cameras, (1) used to track the target and (2) used for navigation. The tracking method used is the lucas kanade method. This requires a camera to determine the position of the coordinates to be converted in actuator motion. Installation Cameras and weapons use one robot that can be moved as per the user's wish. The target lock position remains when UGV is in motion. The actuator that drives this weapon stand uses an adjustable servo angle. The results of the tests performed on this Final Project is able to lock the position of the selected target and can direct to that position by entering the PI controller in this system. The best precision is to use proportional controls with values of 0.04 and 0.08 and integral controls with values of 0.005 and 0.025 ie with a rise time of about 2 seconds, an overshoot of about 10 pixels and a steady state error of about 1 pixel. Testing the accuracy of error speed in testing with distance of 170 and 290cm from robot. In the experiment use light intensity with errors below 15 lux and above 2000 lux. If it is triggered by a speed factor, the system will error at speeds above 200 pwm.



Key words: UGV, Tracking Camera, Weapon Technology

*Halaman ini sengaja dikosongkan*

# KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas kasih dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir ini dengan judul :

**Pembidik Otomatis Kubah Senjata Untuk Kendaraan Darat Tanpa Awak**

Tugas Akhir ini merupakan persyaratan dalam menyelesaikan pendidikan program Strata-Satu di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dibuat berdasarkan teori-teori yang didapat selama mengikuti perkuliahan, dan pengarahan dosen pembimbing dari awal hingga akhir pengerjaan Tugas Akhir ini.

Pada kesempatan ini, penulis ingin berterima kasih kepada pihak-pihak yang membantu pembuatan tugas akhir ini, khususnya kepada:

1. Bapak, Ibu, serta seluruh keluarga yang memberikan dukungan baik moril maupun materiil.
2. Ronny Mardiyanto, S.T., M.T.,Ph.D. selaku dosen pembimbing 2 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
3. Dr. Ir. Djoko Purwanto, M.Eng. selaku dosen pembimbing 1 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
4. Yohan, dan Pak Dhe yang turut serta membantu mengatasi permasalahan yang dihadapi ketika mengerjakan Tugas Akhir
5. Sahabat A206 Pandu, Hazbi, Komang, Mas zaman, Mas iwin serta seluruh warga S2 yang telah memberikan hiburan serta kenangan selama pengerjaan Tugas Akhir
6. Sahabat “Ngompek” Gendut, jember, Wek, Feris, Jatu, Sotob, Kimbum, Gaza, Erpan, Latip, Dapok, Datuk, Bintang, Subur, dan Rawon yang telah memberikan warna selama perkuliahan
7. Sahabat “Aliens” bulek, bintang, topa, datuk, chisa, bunga, farid, dll
8. Toto, Albert, Della, KL yang selalu menjadi inspirasi baru dalam menjalani perkuliahan.
9. Teman-teman laboratorium Elektronika yang tidak dapat disebutkan satu-persatu, telah membantu proses pengerjaan tugas akhir ini.

Penulis sadar bahwa Tugas Akhir ini belum sempurna dan masih banyak hal yang dapat diperbaiki. Saran, kritik dan masukan dari semua pihak sangat membantu penulis untuk pengembangan lebih lanjut.

Terakhir, penulis berharap Tugas Akhir ini dapat memberikan manfaat bagi banyak pihak. Penulis juga berharap Tugas Akhir ini dapat membantu pengembangan aplikasi *image processing*.

Surabaya, 1 Juni 2017

Penulis

# DAFTAR ISI

# TABLE OF CONTENTS

# DAFTAR GAMBAR

*Halaman ini sengaja dikosongkan*

# DAFTAR TABEL

*Halaman ini sengaja dikosongkan*

# BAB I
# PENDAHULUAN

## 1.1. Latar Belakang

Teknologi persenjataan dalam dunia militer telah menjadi hal yang terus dikembangkan dalam beberapa dekade ini. Perkembangannya sudah berkembang dengan pesat, baik peralatan tempur udara, darat maupun laut. Didukung dari berbagai aspek guna memudahkan militer dalam melakukan pertahanan Negara maupun penyerangan terhadap Negara lain. Tank anti baja dan panser merupakan peralatan tempur yang sering digunakan dalam medan pertempuran darat. Dalam pertempuran darat dibutuhkan alat tempur yang dapat mendukung di berbagai medan, anti peluru serta memiliki sistem persenjataan yang mematikan. Kedua peralatan ini mempunyai kemampuan yang sangat bagus dalam berbagai medan dan dilengkapi dengan peralatan tembak yang cukup memadai.

Saat ini senjata yang berada di atas tank / panser masih dioperasikan secara manual oleh manusia. Dengan pengoperasian secara manual ini memiliki beberapa keungulan dan kekurangan. Keungulannya antara lain adalah, senapan dapat digerakkan secara bebas sesuai dengan keingginan operator, memiliki pergerakan yang lincah dan cara pengoperasianya tidak jauh beda dengan senapan yang biasanya digunakan oleh para prajurit. Adapun beberapa kekuranganya, karena senapan ini berada dibagian atas tank / panser untuk mengoperasikanya operator harus berada pada bagian atas tank / panser. Hal ini menyebabkan kurang terjaminya keselamatan dari operator, baik dari serangan musuh atau terhadap goncangan saat melewati medan yang sangat berat.

Sehingga perlu dicarikan solusi yang lebih efisien pada masalah yang diangkat dan dijelaskan, dengan menerapkan teknologi dalam bidang pengolahan citra digital dan otomasi. Dalam mendeteksi/mengenali objek dapat digunakan kamera sebagai sensor yang bisa menggantikan peranan indra penglihat pada manusia. Objek yang ditangkap oleh sebuah sensor kamera dapat diolah menjadi sebuah informasi. Ada banyak metode yang dapat digunakan dalam proses pengolahan citra oleh kamera. Salah satu tekniknya adalah dengan menggunakan metode *motion tracking* dengan algoritma *lucas kanade.* Algoritma *Lucas Kanade* merupakan salah satu algoritma *Motion Tracking* yang melakukan pencarian berdasarkan warna. *Lucas Kanade*

dalam melakukan *tracking* dengan menggunakan *Optical Flow* dimana dapat mencari arah pergerakan objek yang akan ditracking dengan baik.

Pada tugas akhir ini akan dibangun sebuah sistem yang akan digunakan untuk menggerakkan senapan secara otomatis dalam mengikuti targetnya. Sistem ini dapat menangkap objek bergerak dengan menggunakan kamera pada sebuah senjata bersama turret, yang akan dikrontrol dengan kamera/webcam. Kamera digunakan untuk mendeteksi target yang bergerak berdasarkan warna tertentu kemudian informasi diolah oleh Raspberry dengan menggunakan metode *Lucas Kanade*. Hasil dari pengolahan citra tersebuat dijadikan sebuah informasi yang akan dikirim kesebuah motor servo yang akan menggerakkan arah sasaran tembak mengikuti target yang ditangkap oleh kamera, sehingga tembak dapat bergerak secara otomatis mengikuti sasarannya. Dengan demikian servo akan mempertahankan posisinya ketika UGV (*unmanned ground vehicle*) digerakkan bersama dengan turret diatasnya. Beberapa metode telah dikembangkan untuk mengarahkan senjata secara otomatis, seperti penggunaan kamera pada senjata otomatis atau sensor untuk mendeteksi target. Namun, metode-metode tersebut sejauh ini hanya digunakan untuk senjata yang statis atau diam.

Oleh karena itu, diajukanlah proposal tugas akhir ini dengan harapan agar sistem ini dapat membantu dalam proses pengarahan senjata otomatis. Sehingga, pengarahan senjata otomatis menjadi lebih efektif dan efisien. Semakin efektif senjata maka akan semakin memudahkan operator dalam menjalankan kerjanya. Semakin efisien senjata maka akan semakin optimal dari segi fungsinya yang salah satunya untuk mengurangi jumlah korban jiwa yang timbul dalam peperangan.

## 1.2. Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Robot dapat mendeteksi objek yang akan jadi sasarannya
2. Raspberry Pi dapat mengoperasikan kamera dan servo agar dapat melakukan object tracking terhadap pola dan warna
3. Mengintegrasikan kamera dengan Raspberry Pi
4. Robot dapat dikendalikan dari jarak jauh atau nirkabel
5. Pengukuran posisi target berdasarkan data yang didapatkan dari kamera.
6. Penguncian senjata secara otomatis.

## 1.3. Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah

1. Sudut Maksimal yang dapat dijangkau servo adalah $45^o$-$135^o$
2. Perancangan sistem ini menggunakan kamera yang hanya menerima panjang gelombang cahaya tampak.
3. Sistem ini bekerja pada lokasi dengan sistem pencahayaan yang tepat, tidak ada *backlight* dan tidak terlalu redup.
4. Memiliki jarak maksimal target pada jarak tertentu
5. Target dipilih secara manual di *frame* kamera kiri.

## 1.4. Tujuan

Tujuan yang ingin dicapai dalam perancangan ini adalah:

1. Target berupa manusia dapat diketahui posisi nya dari kamera yang terpasang pada senjata.
2. Target berupa manusia dapat dikunci oleh kamera melalui perbedaan kontras cahaya
3. Sistem dapat mengartikan data posisi target berupa manusia untuk mengarahkan senjata.
4. Sistem dapat mengarahkan senjata ke target berupa manusia secara otomatis.
5. Sistem dapat mempertahankan arah bidikan ke target berupa manusia ketika *UGV* bergerak
6. Robot dapat dikontrol dengan menggunakan jaringan wireless

## 1.5. Sistematika Penulisan

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat terbagi menjadi lima bab dengan sistematika penulisan sebagai berikut:

- BAB I : PENDAHULUAN

  Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, sistematika penulisan, dan relevansi.
- BAB II : TINJAUAN PUSTAKA DAN TEORI PENUNJANG

  Bab ini menjelaskan tentang teori penunjang dan *literature* yang dibutuhkan dalam pengerjakan tugas akhir ini. Dasar teori yang menunjang meliputi *tracking, optical flow, pulse width modulation*, Mikrokontroler Arduino, *driver* motor L298N, dan Motor DC Brush.

Bagian ini memaparkan mengenai beberapa teori penunjang dan beberapa literatur yang berguna bagi pembuatan Tugas Akhir ini.

- BAB III : PERANCANGAN SISTEM

  Bab ini menjelasakan tentang perencanaan sistem baik perangkat keras (*hardware*) maupun perangkat lunak (*software*) untuk sistem penguncian posisi target dan pengarahan senjata secara otomatis.

- BAB IV : PENGUJIAN

  Pada bab ini akan menjelaskan hasil uji coba sistem beserta analisanya.

- BAB V : PENUTUP

  Bagian ini merupakan bagian akhir yang berisikan kesimpulan yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran untuk pengembangan lebih lanjut.

## 1.6. Relevansi dan Manfaat

Tugas Akhir ini dapat dikembangkan menjadi lebih berguna, *tracking* object secara otomatis hanya pada gerakan rotasi dan dikembangkan menjadi translasi serta kontrol robot dapat pula dikembangkan menjadi berbasis web sehingga ruang lingkup kontrol dapat lebih luas sehingga memudahkan operator dalam mengontrol robot. Akan tetapi kesemuanya itu tidak dibahas dalam skripsi ini.

Manfaat yang ingin didapat dari tugas akhir ini adalah:

1. Membantu dalam pertahanan di dunia militer.
2. Membantu dalam mengurangi korban manusia dalam peperangan ataupun sengketa yang melibatkan dunia militer.

# BAB II
# TINJAUAN PUSTAKA DAN TEORI PENUNJANG

Teori penunjang dalam bab ini menjelaskan tentang teori penunjang yang berhubungan dengan keseluruhan sistem yang akan dibuat pada tugas akhir ini. Sedangkan tinjauan pustaka dalam bab ini menjelaskan tentang sistem-sistem yang berhubungan dengan tugas akhir ini dan pernah diimplementasikan oleh penulis-penulis sebelumnya.

## 2.1. Sistem *UGV* yang pernah ada

UGV adalah perangkat mekanik yang dioperasikan baik itu secara manual maupun otomatis di atas permukaan tanah untuk membawa atau mengangkut sesuatu tanpa adanya kontak secara langsung oleh manusia.[1] UGV merupakan salah satu bagian dari pengembangan UAV (Unmanned Aerial Vehicle) dan ROUV (Remotely Operated Underwater Vehicle). UGV dikembangkan untuk kebutuhan sipil dan militer yang sulit dijangkau dan berbahaya bagi manusia.[2]

Berdasarkan cara kerjanya ada dua kelas umum UGV, yaitu :

1. Teleoperated[3] Sebuah teleoperated UGV adalah sebuah kendaraan yang dikendalikan oleh operator manusia di lokasi yang jauh melalui hubungan komunikasi. Contoh teleoperated UGV dapat dilihat pada UGV yang bekerja dengan bahan peledak dan melumpuhkan bom.
2. Otonom[4] UGV otonom adalah robot yang bergerak secara otomatis dan biasanya memiliki kemampuan untuk :
   - Memperoleh informasi tentang kondisi lingkungan sekitarnya.
   - Bekerja untuk jangka waktu yang panjang tanpa intervensi manusia.
   - Mampu berjalan dari titik A ke titik B tanpa bantuan navigasi manusia.
   - Dapat menghindari sesuatu yang berbahaya, tergantung algoritma yang diberikan.
   - Dapat mendeteksi objek-objek tertentu, tergantung sensor yang digunakan.

Dalam penelitian yang lain sistem UGV yang digunakan adalah dengan menggunakan GPS dan kompas. Sensor ini digunakan untuk menentukan posisi target yang akan dituju. Sensor GPS kemudian memberikan informasi koordinat posisi UGV dan posisi yang akan dituju. Perbedaan nilai koordinat ini nantinya akan dijadikan error yang kemudian

digunakan untuk menggerakkan UGV pada lokasi tertentu secara otomatis. Derajat belok yang dilakukan UGV di lakukan *feedback* dengan menggunakan sensor kompas.

Prinsipnya adalah sama yaitu tentang tracking, namun terdapat perbedaan pada sensor dan aktuator nya. Pada Tugas Akhir yang penulis buat kali ini, proses *tracking* dengan mode otomatis menggunakan kamera dan aktuator nya menggunakan servo. Peran UGV hanya pada proses kontrol manual saja.

## 2.2. Metode Tracking yang pernah ada

Dalam melakukan proses *tracking* terdapat beberapa metode yang dapat digunakan. Pemilihan metode disesuaikan dengan kondisi yang dibutuhkan dari sistem. Kondisi dapat ditunjang dari segi kecepatan respon aktuator, ukuran objek yang akan dideteksi, dan tingkat presisi dari bidik senjata.

### 2.2.1. Mean Shift Tracking

*Tracking* objek .secara *real time* merupakan tugas penting dalam penglihatan komputer, dan sudah banyak algoritma telah diajukan untuk mengatasi kesulitan yang timbul dari *noise.* Dalam penelitian yang lain digunakan algortima mean shift yaitu dengan menghitung nilai rata-rata piksel warna dari sebuah objek. Mean shift pada awalnya ditemukan oleh Fukunaga dan Hostetler. Bradski kemudian memodifikasi dan mengembangkan algortima CAMSHIFT sebagai pendeteksi wajah. Meskipun tidak kuat, algoritma CAMSHIFT [5], sebagai skema pelacakan berbasis pergeseran paling awal, sebenarnya dapat menangani berbagai jenis pergerakan objek. Pada CAMSHIFT, momen bobot gambar ditentukan oleh model target yang digunakan untuk memperkirakan skala (juga disebut daerah) dan orientasi objek.

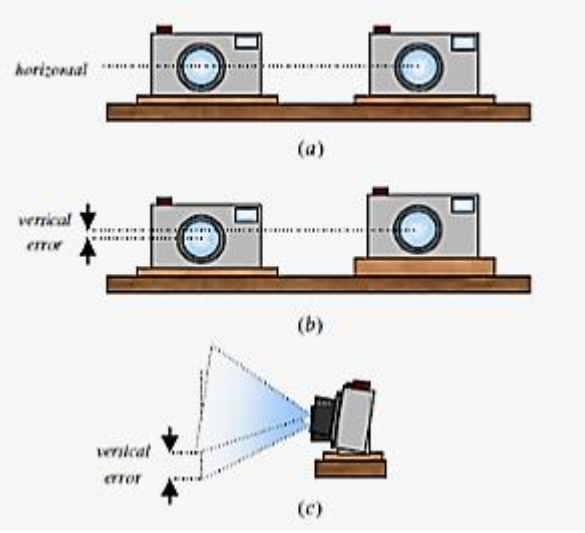
### 2.2.2. Sterero Vision

Dalam penelitian yang lain menggunakan metode stereo vision. Stereo vision menggunakan dua citra yang diambil dalam waktu yang hampir bersamaan pada suatu area dan arah yang sama. Dengan menggunakan stereo vision, jarak objek-objek yang ada di

kedua citra bisa diukur dengan menggunakan beberapa metode pengukuran jarak yang berdasarkan stereo vision, seperti metode triangulasi. Untuk melakukan pengukuran jarak objek menggunakan metode triangulasi, ada syarat yang harus dipenuhi, yaitu:

- Kamera harus benar-benar sejajar dan mengarah pada arah yang sama.seperti pada Gambar 1.
- Pengambilan dua citra harus bersamaan

Peletakan dua kamera yang tidak sejajar akan menyebabkan posisi horisontal dari dua citra yang diambil oleh dua kamera tersebut tidak sama. Hal ini akan menyebabkan proses pengukuran jarak dengan metode triangulasi tidak dapat dilakukan. Namun, pada praktiknya, peletakan dua kamera agar benar-benar sejajar memang sangat sulit untuk dilakukan. Untuk mengatasi hal tersebut, dibutuhkan beberapa proses komputasi yang dapat membuat posisi horisontal dari dua citra menjadi sama, yaitu Stereo Calibration dan Stereo Rectification [x].




**Gambar 2.1** Peletakan kamera sterero yang sejajar (a) dan peletakan kamera stereo dengan kesalahan vertikal (b) dan (c). [6]

## 2.3. Citra Digital

Untuk menampilkan sebuah gambar pada layar-layar digital, gambar tersebut harus melalui proses digitalisasi citra untuk diubah menjadi citra digital. Proses-proses tersebut meliputi proses *sampling* dan proses *quantization*. Proses *sampling* membagi gambar menjadi beberapa kotak kecil. Proses *quantization* memberi nilai kecerahan dari setiap kotak tersebut. Nilai hasil *quantization* juga sering disebut nilai kedalaman.

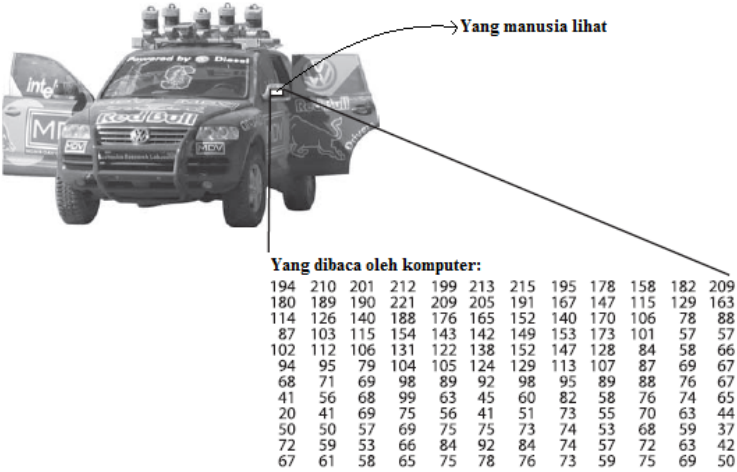
Hasil dari proses digitalisasi citra adalah sebuah matriks yang berisi nilai-nilai riil. Elemen-elemen dari matriks tersebut disebut *picture element* atau sering disbut *pixel*. Sebuah *pixel* mempunyai dua property yaitu koordinat posisi dari *pixel* tersebut dan nilai dari *pixel* tersebut[7]. Sehingga, sebuah *pixel* dapat dinyatakan sebagai fungsi dua dimensi $f(x, y)$ dengan titik asal $x$ dan $y$ ada di pojok kiri atas dari citra. Misalkan, sebuah *pixel* dinyatakan $f(2, 3) = 5$, berarti *pixel* tersebut berada di posisi $x = 2$ dan $y = 3$ dari pojok kiri atas citra dengan tingkat kecerahan 5. Gambar 2.1 menunjukkan citra yang dilihat oleh manusia merupakan kumpulan matriks data yang dibaca oleh komputer. Dari gambar tersebut bisa dilihat bahwa, pada potongan citra bagian spion mobil memiliki nilai kecerahan sebesar 194 pada koordinat $x = 0$ dan $y = 0$. Sedangkan pada koordinat $x = 1$ dan $y = 0$, kecerahan citra adalah 210.

Kedalaman *pixel* sering disebut juga kedalaman warna. Citra digital yang memiliki kedalaman *pixel* n bit disebut juga citra n-bit. Dalam pemrosesan citra digital, citra digital bisa dibedakan berdasarkan batas nilai kecerahan atau nilai kedalaman dari suatu citra. Beberapa jenis citra berdasarkan kedalaman citra, di antaranya adalah citra warna atau sering disebut sebagai citra RGB (*Red Green Blue*), citra *grayscale*, dan citra biner.

### 2.3.1. Citra Warna

Citra warna adalah citra yang setiap *pixel*-nya ditentukan oleh tiga nilai masing-masing untuk komponen merah, biru, dan hijau[9]. Komponen warna lebih dikenal sebagai kanal. Sehingga, nilai dalam satu *pixel*-nya terdapat 3 kanal yang mewakili nilai dari warna merah, warna hijau, dan warna biru.

Kombinasi nilai dari ketiga warna tersebut akan membentuk suatu warna tertentu. Gambar 2.2 menunjukkan warna-warna yang dihasilkan dari kombinasi nilai warna merah, hijau, dan biru.




**Gambar 2.2** Konsep *pixel* [6]

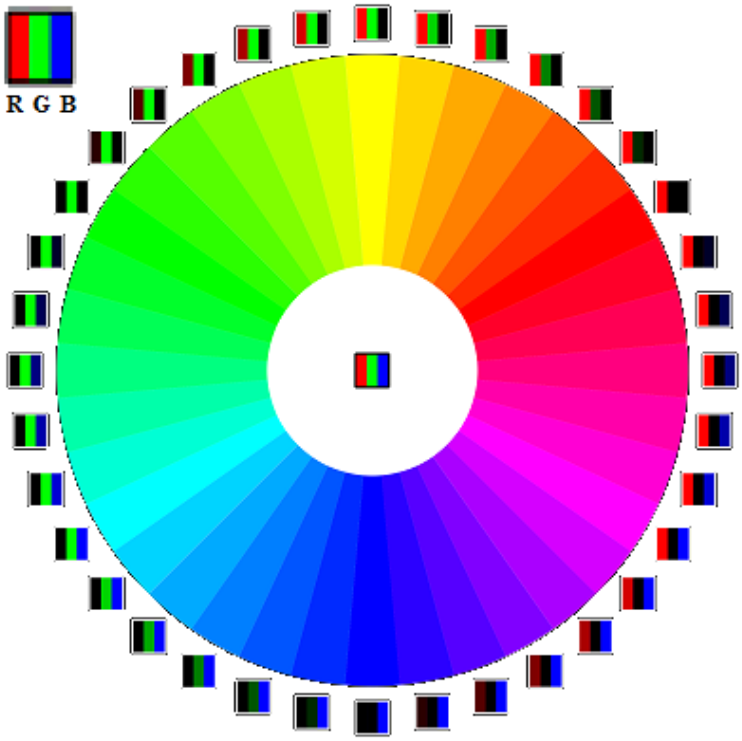
Nilai dari masing-masing komponen merah, hijau, dan biru memiliki rentang tertentu, tergantung tipe data yang dipakai. Jika tipe data tiap kanal warna tersebut berupa tipe data 8 bit, rentang nilai setiap kanal berada pada rentang nilai antara 0 sampai 255. Jika tipe datanya 16 bit, rentang nilai setiap kanal adalah antara 0 sampai 65535. Jumlah bit yang lebih banyak akan memberi variasi warna yang lebih banyak pula. Tipe data 8 bit per kanal akan memberi variasi warna sebanyak 16777216 warna, sedangkan tipe data 16 bit per kanal akan membentuk 281474976710656 jenis warna. Akan tetapi, dalam proses pengolahan citra, jumlah bit yang lebih sedikit memberi keuntungan pengolahan citra yang lebih cepat.

### 2.3.2. Citra *Grayscale*

Citra *grayscale* disebut juga citra intensitas atau citra *gray level*[6]. Berbeda dengan citra warna yang terdiri dari tiga kanal warna, citra *grayscale* hanya terdiri satu kanal saja. Jika kanal tersebut merupakan data 8 bit, akan ada 256 *gray level*, sehingga setiap *pixel* akan memiliki nilai antara 0 sampai 255. Nilai 0 merupakan warna hitam dan 255 merupakan warna putih. Di antara 0 dan 255 merupakan warna abu-abu tergantung kecerahan

dari *pixel* tersebut. Semakin tinggi nilainya, semakin cerah *pixel* tersebut. Gambar 2.3 menunjukkan rentang *gray level* dari hitam yang bernilai 0 hingga putih yang bernilai 255 dan identik dengan terang.

Tipe citra ini sangat sering digunakan di dalam pengolahan citra. Hal ini dikarenakan jumlah datanya yang tidak terlalu besar




**Gambar 2.3** Kombinasi warna RGB [6]


**Gambar 2.4** Rentang *gray level* [6]


**Gambar 2.5** Citra RGB dan Citra *Grayscale* [6]

yang akan mempercepat proses pengolahan data, tapi data *grayscale* tidak menghilangkan informasi penting dari citra. Oleh karena itu, jika citra asli yang didapatakan berupa citra warna dan akan dilakukan proses pengolahan citra,
citra tersebut biasanya dikonversi dahulu ke citra *grayscale* dengan cara mencari nilai rata-rata dari nilai-nilai dari kanal-kanal merah, hijau, dan biru. Gambar 2.4 menunjukkan perbandingan citra warna dan citra *grayscale*.

### 2.3.3. Citra Biner

Citra biner merupakan *array* logika yang hanya berisi 0 dan 1, yang diartikan hitam dan putih[10]. Untuk citra biner yang normal, objek biasanya ditandai dengan warna hitam. Sedangkan warna putih menunjukkan *background*.

Citra biner berasal dari citra *grayscale* yang di-*threshold*. Sebagai contoh, suatu citra *grayscale* di-*threshold* menggunakan nilai *threshold* 100, maka *pixel* yang memiliki nilai *gray level* kurang dari 100 akan diberi nilai 0 (warna hitam), sedangkan *pixel* yang memiliki nilai *gray level* di atas 100 akan diberi nilai 255 (warna putih). Gambar 2.5 menunjukkan hasil konversi menjadi citra biner.

Walaupun citra warna lebih disukai banyak orang, namun citra biner tetap dipakai, terutama dalam pemrosesan citra. Hal ini dikarenakan kebutuhan memori citra biner kecil sehingga waktu

**Gambar 2.6** Citra Biner [15]

pemrosesan lebih cepat dibandingkan dengan citra *grayscale* ataupun warna.

## 2.4. *Optical Flow Lucas-Kanade*

Menurut Szeliski (2011: 360) optical flow adalah vektor-vektor yang menjelaskan perpindahan setiap piksel pada sebuah wilayah gambar. Vektor ini dihitung dari pengaruh brightness. Fitur ini banyak digunakan pada segmentasi pergerakan objek dan aplikasi tracking. Salah satu metode optical flow yang sering digunakan adalah optical flow dengan metode Lucas Kanade.

Pelacakan dengan algoritma Lucas-Kanade menggunakan titik pada fitur sebelumnya untuk menentukan posisi titik fitur pada frame selanjutnya.

Menurut Baker & Iain (2004: 2-6) Algoritma Lucas-Kanade berfungsi untuk menemukan sebuah template gambar T(x) didalam sebuah gambar I(x) dimana x = (x,y)z adalah vektor dari koordinat piksel pada gambar. Jika algoritma Lucas-Kanade digunakan untuk melacak

**Gambar 2.7** Ilustrasi tracking titik [17]

gambar dari t=1 sampai t=2, template T(x) merupakan sub-bagian yang di ekstrak (misal: ukuran gambar 64x64 piksel) dari gambar t=1 dan I(x) adalah gambar di t=2.

Perpindahan sebuah template pada gambar T(x) ke gambar I(x) dapat ditulis dengan persamaan berikut:

$$W(x;p)= \begin{bmatrix} \boldsymbol{x+p1} \\ \boldsymbol{x+p2} \end{bmatrix} \quad (1)$$

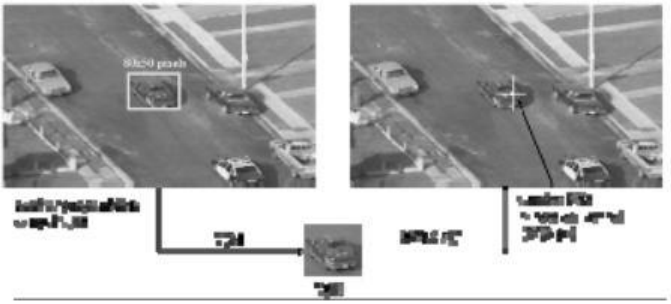
Dimana W(x;p) merupakan fungsi warp parameter, sedangkan x dan p merupakan vektor optical flow. Nilai p dapat berubah-ubah tergantung dari parameter dimana p=(p1,p2,….,pn)T.

Perpindahan affine pada template secara 3 dimensi pada gambar dapat ditulis sebagai berikut:

$$W(x;p)= \begin{bmatrix} (\mathbf{1}+\mathbf{p1}).\boldsymbol{x} & + & \boldsymbol{p3}.\boldsymbol{y} & + & \boldsymbol{p5} \\ \boldsymbol{p2}.\boldsymbol{x} & + & (\mathbf{1}+\boldsymbol{p4}).\boldsymbol{y} & + & \boldsymbol{p6} \end{bmatrix} = \begin{bmatrix} \mathbf{1}+\boldsymbol{p1} & \boldsymbol{p3} & \boldsymbol{p5} \\ \boldsymbol{p2} & \mathbf{1}+\boldsymbol{p4} & \boldsymbol{p6} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \mathbf{1} \end{bmatrix} \quad (2)$$

Dimana terdapat 6 parameter p=(p1,p2,p3,p4,p5,p6)T pada ruang 3 dimensi. Secara umum, nilai dari parameter n dapat berubah-ubah besarnya dan W(x;p) dapat berubah-ubah secara kompleks.

Tujuan dari algoritma Lucas-Kanade adalah untuk meminimalisasi kesalahan penemuan template gambar T(x) di dalam sebuah gambar I(x) yang digambarkan pada persamaan berikut:



**Gambar 2.8** Pemetaan gambar T(x) pada gambar I(W(x;p)) [10]

$$\sum_x \left[I\left(W(x;p)\right) - T(x)\right]^2 \quad (3)$$

Pencarian nilai p pada gambar tidak linear, oleh karena itu Lucas-Kanade mengasumsikan nilai p telah diketahui dan diperbaharui dengan nilai Δp secara iteratif dan menambahkan persamaan menjadi:

$$\sum_x \left[I\left(W(x;p+\Delta p)\right) - T(x)\right]^2 \quad (4)$$

dimana nilai p akan diperbaharui dengan persamaan:

$$p \leftarrow p + \Delta p \quad (5)$$

Dua persamaan diatas akan diiterasi secara terus - menerus sampai nilai estimasi dari parameter p menjadi konvergen. Secara umum tes untuk ke-konvergenan dilihat dari apakah standar vektor Δp dibawah nilai threshold $\varepsilon$, $\|\Delta p\| \leq \varepsilon$.

Ekspansi Taylor orde pertama digunakan untuk melinearisasikan persamaan (4) karena perpindahan jarak template sangatlah kecil, sehingga hasil orde pertama deret Taylor adalah:

$$\sum_x \left[I\left(W(x;p)\right) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x)\right]^2 \quad (6)$$

dimana dalam persamaan ini $\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$ merupakan gradien koordinat frame pada gambar I ke koordinat frame pada gambar T dan nilai jacobian dari W oleh persamaan berikut:

$$\frac{\partial W}{\partial p} = \begin{pmatrix} \frac{\partial Wx}{\partial p1} & \frac{\partial Wx}{\partial p2} & \cdots & \cdots & \frac{\partial Wx}{\partial pn} \\ \frac{\partial Wx}{\partial p1} & \frac{\partial Wx}{\partial p2} & \cdots & \cdots & \frac{\partial Wx}{\partial pn} \end{pmatrix} \quad (7)$$

Untuk menemukan jalan koordinat posisi template pada gambar T ke gambar I maka digunakan metode steepest descent yaitu sehingga menambahkan persamaan menjadi:

$$2\sum_x \left[\nabla \mathrm{I} \frac{\partial W}{\partial p}\right]^T \left[I\big(W(x;p)\big) + \nabla \mathrm{I} \frac{\partial W}{\partial p} \Delta \mathrm{p} - T(x)\right] \quad (8)$$

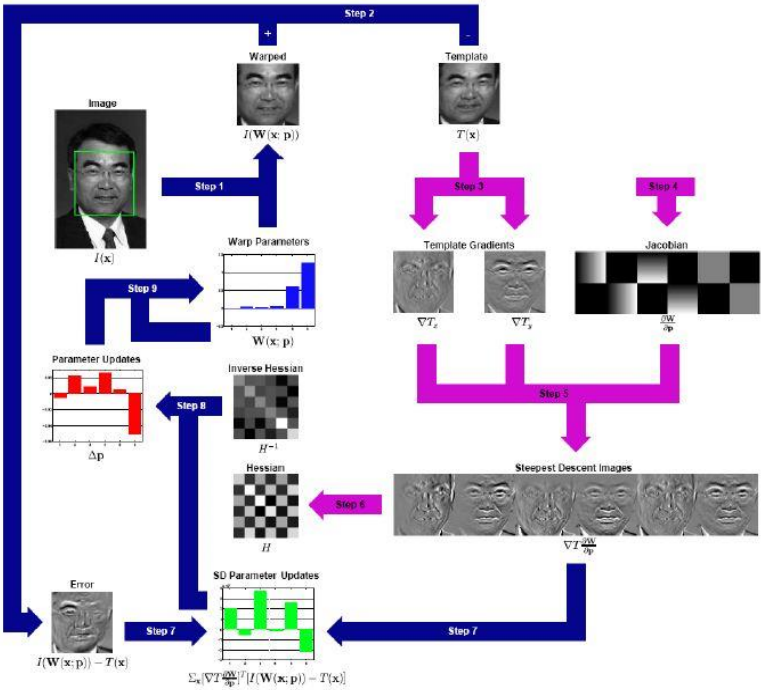
Untuk menentukan nilai Δp maka dipergunakan matriks Hessian yang dapat ditemukan melalui persamaan berikut:

$$H = \sum_x \left[\nabla \mathrm{I} \frac{\partial W}{\partial p}\right]^T \left[\nabla \mathrm{I} \frac{\partial W}{\partial p}\right] \quad (9)$$

Dan nilai Δp dapat ditemukan dengan persamaan berikut:

$$\Delta p = H^{-1} \sum_x \left[\nabla \mathrm{I} \frac{\partial W}{\partial p}\right]^T [T(x) - I\big(W(x;p)\big)] \quad (10)$$

Sehingga dengan demikian posisi jendela pendeteksian dapat ditentukan pada gambar I(W(x;p+ $\Delta p$)).




**Gambar 2.9** Algoritma optical flow Lucas-Kanade (Szeliski, 2011: 353) [10]

Algortima optical flow Lucas-Kanade adalah sebagai berikut:

1. Temukan template dengan perkiraan pada gambar I dengan menggunakan W(x;p) untuk menghitung I(W(x;p)).
2. Hitung nilai kesalahan T(x)-I(W(x;p)).
3. Tentukan nilai gradien $\nabla I$ dengan W(x;p).
4. Hitung nilai Jacobian $\frac{\partial W}{\partial p}$ pada (x;p).
5. Hitung nilai steepest descent $\nabla \mathrm{I} \frac{\partial W}{\partial p}$ .
6. Hitung nilai matriks Hessian dengan menggunakan persamaan (9).
7. Hitung persamaan $\sum_x \left[\nabla \mathrm{I} \frac{\partial W}{\partial p}\right]^T \left[T(x) - I\big(W(x;p)\big)\right]$.
8. Hitung nilai $\Delta p$.
9. Update parameter .
10. Ulangi algoritma hingga $||\Delta \mathrm{p}|| \leq \varepsilon$.

Dalam penerapannya algoritma Lucas-Kanade menggunakan enam parameter untuk penghitungan, yaitu:

1. Image<Gray,byte> prev,
   Gambar awal yang akan digunakan untuk penghitungan metode Lucas-Kanade.
2. Image<Gray,byte> curr,
   Gambar yang didapat saat ini untuk menghitung posisi objek yang akan dilacak pada gambar berikutnya.
3. PointF[] prevFeatures,
   Array dari titik yang perlu untuk dilacak.
4. Size winSize,
   Ukuran jendela pencari pada setiap level.
5. Int Level
   Nilai maksimal dari level Lucas-Kanade.
6. McvTermCriteria
   Menspesifikan kapan iterasi untuk setiap titik harus berhenti dilakukan.

## 2.5. *Pulse Width Modulation*

Dasar *pulse width modulation* (PWM) secara luas digunakan di dalam aplikasi elektronika daya untuk pengaturan pengkonversian daya (DC/DC, DC/AC, dll.)[11]. Secara sederhana, PWM merupakan sinyal yang lebar pulsa yang bernilai "HIGH" dalam satu periode mewakili suatu tegangan DC, tergantung nilai *duty cycle*. *Duty cycle* merupakan perbandingan lama waktu sinyal bernilai "HIGH" dengan satu periode. Gambar 2.9 menggambarkan *duty cycle* dari PWM.

Selain digunakan dalam pengaturan pengkonversian daya, ada beberapa aplikasi lain dari PWM. Contoh aplikasi umum yang lain adalah pengendalian kecepatan motor DC, pengendalian motor servo, pengaturan nyala terang LED dan lain sebagainya.

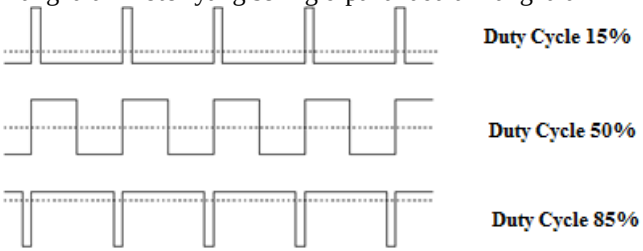
## 2.6. Mikrokontroler Arduino Nano

Mikrokontroler Arduno nano diciptakan dengan basis mikrokontroller ATmega328. Mikrokontroler jenis Arduino nano kurang lebih memiliki fungsi yang sama dengan Arduino uno begitu pula dengan jumlah pin yang dimiliki nya. Keunggulan menggunakan Arduino nano adalah dari segi ukurannya yang mungil sehingga memudahkan penempatan elektroniknya. Arduino nano memiliki tegangan kerja 5VDC. Selain itu Arduino juga dilengkapi 14 pin digital I/O serta 8 pin input. *Clock speed* yang dimiliki pun juga cukup mendukung dalam perancangan sistem Tugas Akhir ini yaitu sebesar 16MHz. ATmega328 memiliki flash memory sebesar 32 Kb, 2 Kb memory pada SRAM serta 1 Kb pada EEPROM. Setiap pin pada mikrokontroller dapat memberikan atau menerima arus maksimum sebesar 40 mA. Disamping itu beberapa pin memiliki fungsi khusus seperti serial TX, RX yang digunakan untuk menerima dan mengirimkan data serial. Kemudian external interrupt yaitu pada pin 2 dan pin 3 dapat dikonfigurasi untuk memicu sebuah interupsi pada nilai yang rendah, meningkat, atau menurun, atau perubahan nilai

Berdasarkan chip ATmega328, *board* mikrokontroler ini memiliki USB serial yang digunakan untuk melakukan *upload* program. Hal ini merupakan kelebihan menggunakan Arduino nano dengan *user interface* yang mudah dan menyenangkan. Gambar 2.10 merupakan gambar dari *Board Arduino Nano* yang sudah dilengkapi dengan kabel *downloader* tipe untuk memasukkan program dari komputer ke mikrokontroler Arduino Nano.

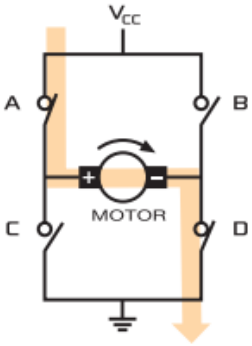
## 2.7. *Driver* motor

*Driver* motor merupakan rangkaian yang memperkuat arus yang akan masuk ke motor dari pengontrol atau mikrokontroler sehingga arus yang masuk ke motor cukup besar untuk menggerakkan motor. Rangkaian motor yang sering dipakai adalah rangkaian *H-Bridge*.




**Gambar 2.10** Duty cycle pada PWM [11]

**Gambar 2.11** *Board* Arduino Nano 3.0 [12]




**Gambar 2.12** Skematik sederhana *H-Bridge*[11]

Rangkaian *H-Bridge* namanya berasal dari rangkaian *full-bridge* yang ditunjukkan oleh gambar 2.11[11]. Dengan rangkaian ini, arah gerak motor bisa diatur. Jika saklar A dan saklar D disambungkan dan saklar B dan saklar C tidak disambungkan, arus akan mengalir dari tegangan sumber ke sisi kiri motor. Hal ini akan membuat gerakan motor searah jarum jam. Sebaliknya, jika saklar yang tersambung adalah saklar B dan saklar C, arus akan mengalir dari tegangan sumber ke sisi kanan motor. Hal ini akan membuat motor bergerak berlawanan arah jarum jam.

Dalam perancangan sistem Tugas Akhir ini digunakan driver motor L298N yang prinsip kerjanya sama dengan prinsip kerja *H-Bridge* namun sudah dirancang menjadi sebuah modul yang kecil dan ringkas. Arus maksimal yang dapat dilewatkan melalui driver L298N ini adalah 2A. Motor yang digunakan dalam perancangan membutuhkan arus kurang dari 2A sehingga penggunaan driver L298N ini sangat dimungkinkan dan terbilang aman. Gambar 2.12 merupakan bentuk dari driver motor L298N.

## 2.8. Kontrol PID

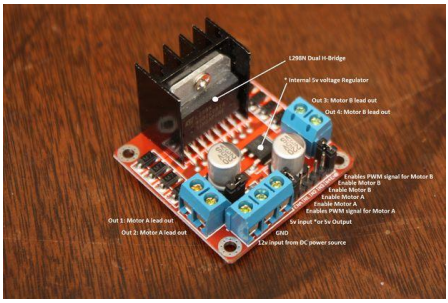
Kontroler *Proportional-Integral-Derivative* (PID) secara luas digunakan di dalam sistem pengaturan industri karena pengurangan jumlah parameter-parameter yang akan di-*tuning*[13]. Kontrol PID menghasilkan sinyal kontrol yang sebanding dengan sinyal *error* (aksi proporsional), sebanding dengan total sinyal *error* (aksi integral), dan sebanding dengan turunan dari kesalahan yang sekarang dengan kesalahan yang sebelumnya (aksi *derivative*). Sinyal *error* merupakan selisih antara *set point* dengan nilai keluaran aktual. Kontrol PID dapat ditulis menjadi persamaan (2.22) berikut ini,

$$u(t) = Kp.e(t) + Kd.\frac{d}{dt}e(t) + Ki.\int e(t)dt \qquad (2.22)$$

dengan $u(t)$ merupakan sinyal kontrol PID dan $e(t)$ merupakan sinyal *error*. $Kp$, $Ki$, dan $Kd$ masing-masng merupakan koefisien Untuk sistem diskrit, persamaan (2.23) di atas dimodifikasi menjadi,

$$u[n] = Kp.e[n] + Kd.(e[n] - e[n-1]) + Ki.\sum e[n] \quad (2.23)$$

Nilai Kp, Ki, dan Kd pada Tugas Akhir ini akan dicari menggunakan *tuning* manual. Sinyal kontrol yang akan dihasilkan berupa PWM untuk menggerakkan motor agar mengarah ke posisi yang diinginkan.




**Gambar 2.13** Driver motor L298N

## 2.9. Komunikasi Serial

Salah satu hal terpenting ketika menggunakan dua *processor* dalam satu sistem adalah komunikasi serial. Komunikasi serial memungkinkan dua *processor* untuk saling bertukar data hanya dengan menggunakan satu kawat konduktor saja. Hal ini dikarenakan data yang dikirimkan berupa deretan bit dari data yang dikirimkan.

Komunikasi serial terbagi menjadi dua jenis, yaitu *universal asynchronous receiver-transmitter* (UART) dan *universal synchronous asynchronous receiver-transmitter* (USART)[14]. USART dapat melakukan komunikasi asinkron dan sinkron, sedangkan UART hanya dapat melakukan komunikasi asinkron saja. Komunikasi sinkron mengharuskan dua divais yang berkomunikasi harus memiliki sistem *clock* yang sama. Sedangkan komunikasi asinkron tidak mengharuskan sistem *clock* dari dua divais berasal dari sistem *clock* yang sama.

## 2.10. OpenCV

OpenCV merupakan suatu *library* dari *computer vision* yang *open source* (gratis digunakan baik untuk urusan akademik ataupun komersil) dan tersedia di *http://SourceForge.net/projects/opencvlibrary*[15]. *Library* ini bisa digunakan dalam bahasa C, C++, Python, dan beberapa bahasa pemrograman lain. Dengan *library* OpenCV, program yang dibuat bisa digunakan untuk mengambil citra hingga mengolah citra. *Library* ini juga memungkinkan *programmer* untuk melakukan manipulasi video ataupun *real-time* video. Gambar 2.14 merupakan logo OpenCV yang terdapat di situs OpenCV yaitu *http://opencv.org*. Di dalam situs tersebut


**Gambar 2.14** Logo OpenCV[15]

terdapat pula hasil dokumentasi beberapa orang tentang pengolahan citra menggunakan OpenCV.

## 2.11. Object Tracking dengan dudukan servo pan/tilt

Object Tracking dengan dudukan servo pan/tilt merupakan webcam yang disematkan dua buah servo Pan dan Tilt terintegrasi dengan OpenCV Framework pada komputer melalui USB yang berfungsi mengikuti satu titik yang diklik pada sebuah objek yang bergerak. Alat ini dibuat oleh Zag Grad sebagai salah satu contoh aplikasi produk dari servo bracket yang dijual di Sparkfun Electronics. Komputer terhubung melalui USB dengan Arduino Uno dalam memerintah motor servo untuk bergerak, begitu pula dengan webcam yang digunakan untuk mengambil citra. Kemampuan dari kamera ini terbataskan pada tempat tertentu dan kekurangan dalam kemudahan instalasi karena menggunakan komputer (PC) sebagai pemroses utama.

*Halaman ini sengaja dikosongkan*

# BAB III
# PERANCANGAN SISTEM

Metodologi yang dikerjakan pada tugas akhir ini adalah sebagai berikut:

1. **Studi Literatur**

   Dalam hal ini bahan-bahan referensi yang berhubungan dengan materi yang akan dibahas dikumpulkan dari semua buku-buku atau internet. Adapun buku yang dijadikan studi literature adalah "*Learning opencv", "Pyramidal Inmplementation of The Lucas-Kanade Feature Tracker".* Selain itu juga diambil beberapa referensi dari internet yaitu Wikipedia dan journal.

2. **Perancangan *Hardware***

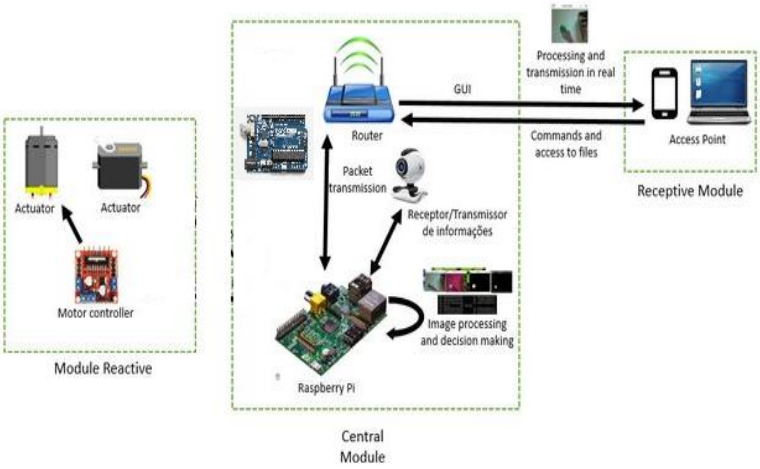
   Perancangan *hardware*, secara umum, meliputi kamera untuk *tracking*, dan senjata. Posisi kamera dan senjata terpasang menjadi satu kesatuan sehingga saat kamera melakukan *tracking* akan sejajar dengan arah yang dibidik oleh senjata.

3. **Perancangan *Software***

   Perangkat lunak dirancang dengan pembuatan source code yang meliputi penentuan target, pengukuran jarak target, perintah kalibrasi jarak target otomatis dan perintah untuk mengarahkan senjata secara otomatis. Untuk pemilihan target, dilakukan oleh user secara manual. Untuk metode tracking target dengan kamera akan dicoba beberapa metode dan akan dipilih metode terakurat dan yang tidak terganggu oleh lingkungan. Selanjutnya, dilakukan pengukuran posisi target terhadap senjata dengan menggunakan metode pengukuran besar piksel dari objek sehingga senjata bisa diarahkan ke target berdasarkan besar kecilnya piksel target terhadap senjata.

4. **Pengujian Sistem**

   Pengujian alat dilakukan untuk menentukan keandalan dari sistem yang telah dirancang. Pengujian dilakukan untuk melihat apakah *software* dan *hardware* dapat bekerja secara baik. Untuk pengujian dapat dilakukan dalam tiga tahap. Pertama adalah pengujian sistem untuk senjata yang diam. Kedua adalah pengujian sistem untuk senjata yang bergerak. Terakhir adalah pengujian system untuk senjata yang bergerak dalam intensitas cahaya yang berbeda.




**Gambar 3.1** Diagram *tracking object*

5. **Analisa**

   Analisa dilakukan terhadap hasil dari pengujian sehingga dapat ditentukan karakteristik dari software dan hardware yang telah dibuat. Apabila karakteristik dari software dan hardware masih belum sesuai, maka perlu dilakukan perancangan ulang

6. **Penyusunan Laporan Tugas Akhir**

   Proses terakhir adalah membuat dokumentasi pelaksanaan tugas akhir yang meliputi dasar teori, proses perancangan, pembuatan, dan pengujian aplikasi.

Pada bagian ini akan dijelaskan mengenai perancangan sistem baik dari perancangan software hingga perancangan hardware yang meliputi perancangan kendaraan darat tanpa awak yang dapat bergerak secara translasi beserta perancangan gerak senjata yang bergerak secara rotasi baik gerak vertikal maupun horizontal. Dalam bab ini juga akan dijelaskan bagaimana proses pengolahan gambar yang dilakukan pada *SOC (system on chip )* beserta proses komunikasi nya dengan mikrokontroler yang akan mengontrol pergerakan robot maupun pergerakan senjata. *System on chip* memiliki banyak macam dalam pengembangannya namun dalam penelitian kali ini digunakan Raspberry Pi 3 dengan beberapa alasan yaitu daya yang tidak dibutuhkan kecil,

ukurannya kecil, dan cukup memadai untuk dilakukan pengolahan citra didalamnya.

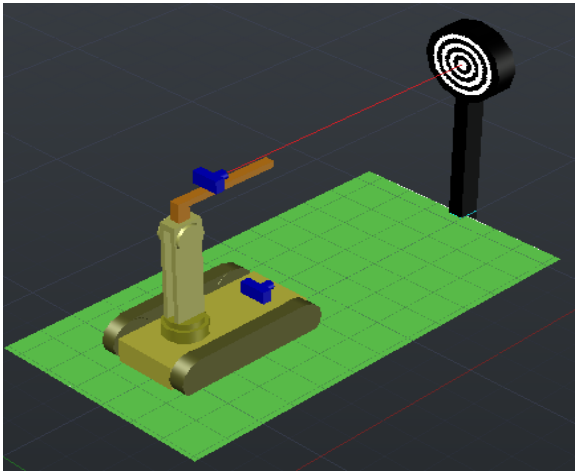
Gambar 3.1 di atas menunjukkan penyusunan kamera tracker yang dikontrol oleh arduino yang berkomunikasi serial dengan open cv pada System On Chip. Posisi antara kamera dan senjata harus berada pada level yang sama dipandang dari bidang horizontal atau vertikal. Untuk senjata, digunakan dua servo agar senjata dapat digerakkan dalam dua derajat kebebasan. Dua kamera dan senjata tersebut dihubungkan dengan processing unit agar bisa dikontrol secara otomatis. Terakhir, dudukan sistem diberikan roda agar sistem senjata dapat digerakkan oleh user. Pergerakannya menggunakan kontrol jarak jauh yang dilengkapi dengan kamera untuk proses navigasi. Proses navigasi robot digerakkan secara manual oleh user.

Pada pengerjaan tugas akhir ini, proses pengolahan citra digunakan untuk mengetahui posisi target terhadap kamera. Posisi tersebut kemudian dikurangkan dengan titik tengah dari frame sehingga dihasilkan sebuah eror. Informasi nilai eror inilah yang digunakan oleh mikrokontroller untuk menggerakkan *platform* senjata yang telah terpasang servo. Senjata dikontrol oleh mikrokontroller agar mengarah pada target tersebut. Pada sistem ini dibutuhkan *operator* yang bertugas untuk mengarahkan senjata serta memilih target. Prosesnya dapat dilakukan menggunakan *remote control* sehingga bisa dilakukan dari jarak jauh. Gambar 3.2 mengilustrasikan sistem persenjataan yang akan dibuat. Seperti terlihat pada gambar, peletakan kamera dipasang diatas senjata sehingga pergerakan senjata juga diikuti pergerakan kamera. *Platform* senjata bisa berotasi secara horizontal dan vertikal dengan batasan sudut tertentu. Sedangkan kendaraan daratnya dapat bertranslasi maju, mundur serta berbelok. Dari gambar tersebut juga bisa dilihat bahwa senjata mengarah pada target yang berada dalam pengamatan kamera. Dalam sistem juga dipasang dua kamera yang pertama pada senjata dan yang kedua pada kendaraan darat. Kamera pertama digunakan sebagai sistem *tracking* guna melakukan fungsi *lock target*, sedangkan kamera kedua digunakan untuk navigasi sehingga memudahkan *operator* dalam mengendalikan kendaraan darat dari jarak jauh.

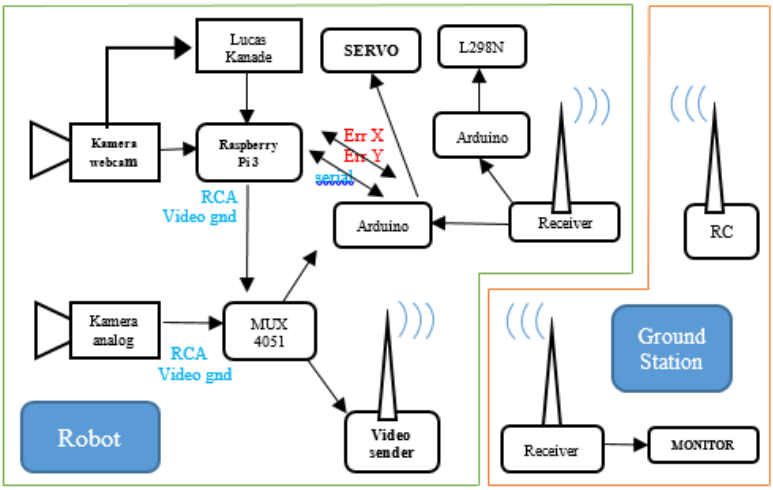
Dalam perencanaannya, sistem ini diharapkan dapat diletakkan untuk mempertahankan daerah-daerah perbatasan atau daerah yang butuh pertahanan. Peletakan *platform* yang dapat bergerak secara rotasi dan translasi memungkinkan persenjataan ini mampu dalam melakukan pengejaran target serta penguncian posisi. Sistem ini lebih difungsikan

*observer* bagi *operator*, karena senjata hanya akan bergerak ketika diperintahkan bergerak oleh operator. Saat dijalankan, sistem ini dapat menampilkan citra dari kamera Raspberry maupun kamera navigasi namun secara bergantian. Saat *operator* mengamati dari kamera dan terdapat sesuatu yang mencurigakan, *operator* dapat mengarahkan senjata pada target kemudian menarget objek tersebut dengan cara memilih objek di citra kamera Raspberry. Jika target bergerak senjata akan selalu mengikuti target yang telah dipilih. Jika target bergerak menjauh kendaraan darat dapat dijalankan bersamaan dengan kondisi senjata yang selalu mengikuti target. Jika dirasa sudah tepat sasaran, sistem ini juga diperintahkan untuk menembak target.

Diagram blok dari sistem ini ditunjukkan seperti pada Gambar 3.3. Secara garis besar, sistem terbagi menjadi empat sub sistem, yaitu *Tracking object*, pengarahan senjata ke target, kontrol manual oleh *remote control*, dan proses pengiriman video. Pada bagian pengarahan senjata secara otomatis dilakukan pengolahan citra menggunakan opencv pada raspberry. Selain bagian tersebut, semua proses dilakukan pada mikrokontroller Arduino nano. Dalam pemrosesan citra didapatkan nilai eror yang dibutuhkan Arduino untuk melakukan control PID pada servo. Sehingga, dibutuhkan komunikasi serial antara raspberry dan mikrokontroller Arduino.



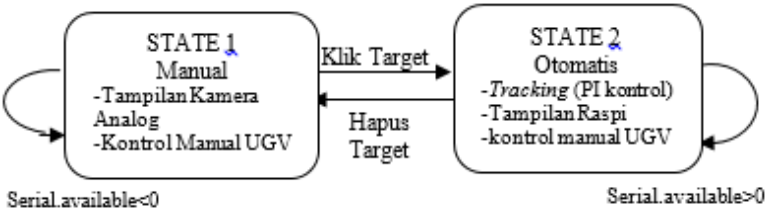
**Gambar 3.2** Ilustrasi sistem




**Gambar 3.3** Diagram Blok Sistem

Untuk mempermudah *operator*, diberikan beberapa fitur tambahan dalam sistem ini yaitu dengan memberikan kontrol pemilihan serta penghapusan titik objek dapat melalui *remote control.* Selain itu program pada raspberry sudah di setting *autorun* mulai saat awal booting. Sehingga operator tidak perlu menggunakan keyboard dan mouse lagi untuk running program dan melakukan klik titik objek.

Dalam menjalankan sebuah sistem diperlukan *Finite State Machine* untuk melakukan algortima sistem dalam lingkup keseluruhan. Perlu diatur proses state yang digunakan yaitu state manual dan otomatis. Dalam sistem otomatis ini pergerakan *turret* senjata di program untuk pergerakan secara otomatis menggunakan pemrosesan citra dan melakukan *tracking* objek. Namun pergerakan pada UGV tetap menggunakan kontrol manual menggunakan *remote control.* Adapun kontrol yang manjadikan sistem mengalami perubahan dari sistem manual ke otomatis adalah dengan melakukan perintah pemilihan titik target. Setelah titik target telah ditentukan, Arduino akan menerima pengiriman serial dari Raspberry Pi yang menandakan bahwa titik target sudah dipilih. Kemudian Arduino selanjutnya akan memproses kontrol servo otomatis dengan mengkonversi nilai *error* menjadi nilai sudut servo. Sistem akan dikembalikan pada mode manual jika dilakukan perintah penghapusan titik target. Ketika titik target dihilangkan, maka pengiriman serial dari

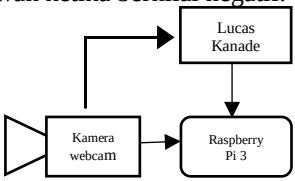



**Gambar 3.4** *Finite State Machine*

Raspberry ke Arduino terhenti. Sehingga Arduino akan mengartikan sistem berada pada mode manual. Adapun gambar Finite State Machine dari sistem terdapat pada Gambar 3.4.

## 3.1. *Tracking Object*

Pada tahap ini, semua proses pengolahan citra dilakukan pada SOC atau raspberry seperti tergambar dalam Gambar 3.5. Dalam melakukan proses *tracking* digunakan pelacakan titik fitur menggunakan algoritma Lucas-Kanade untuk menentukan posisi titik fitur pada frame sebelumnya ke frame selanjutnya. Tujuan dari algortima Lucas-Kanade adalah untuk menemukan sebuah template citra T(x) didalam sebuah citra I(x) dimana x merupakan $x=(x,y)^T$ adalah vector dari koordinat piksel pada citra. Dengan diketahuinya koordinat piksel, maka dapat diketahui pula nilai error dengan menggunakan rumus error=setpoint-(x,y). Dari pengurangan nilai koordinat tengah (setpoint) dengan koordinat piksel maka diperolehlah nilai error. Pada sumbu x jika nilai eror nya bernilai minus, maka koordinat piksel sedang berada di bagian kanan setpoint sedangkan ketika nilai errornya bernilai positif maka koordinat piksel sedang berada di bagian kiri setpoint. Sama halnya dengan yang terjadi pada sumbu y yaitu ketika bernilai positif maka mendapati posisi diatas dan akan mendapati posisi dibawah ketika bernilai negatif.




**Gambar 3.5** Pengolahan citra oleh Raspberry

## 3.2. *Preprocessing* Citra

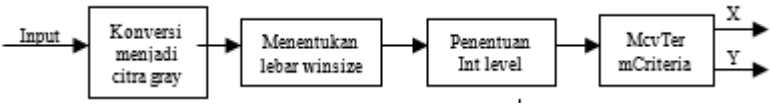
Agar dapat mengetahui posisi target dengan mudah, citra yang diambil dari kamera harus diolah terlebih dahulu yaitu dengan pengkonversian citra menjadi *grayscale* atau citra biner seperti pada Gambar 3.5. Pengkorversian citra menjadi *grayscale* bertujuan untuk memepercepat proses-proses berikutnya karena *grayscale* hanya memiliki lebar data sebesar 8 bit. Dalam proses ini, digunakan *library* OpenCV, yaitu `cvtColor` dengan koefisien `BGR2GRAY`.

Proses selanjutnya adalah pendeteksian titik fitur oleh citra berdasarkan tekstur area pendeteksian. Nilai tekstur yang baik ditentukan dengan nilai eigen pada area tersebut, semakin tinggi nilai eigen akan semakin baik tekstur yang dimiliki. Selain itu nilai *window size* juga mempengaruhi kualitas tracking. Semakin besar nilainya maka titik fitur tidak akan mudah lepas dengan pergerakan yang besar namun akan berpindah-pindah didalam area *window*, sedangkan ketika nilainya semakin kecil titik fitur akan tepat dalam satu koordinat namun tidak dapat menerima pergerakan yang besar. Ketika dilakukan pergerakan yang besar, titik fitur akan hilang dari proses *tracking*. Integer level adalah nilai maksimal dari level lucas-kanade. Kemudian proses terakhir adalah dengan menspesifikkan kapan iterasi untuk setiap titik harus berhenti dilakukan. Dengan itu diperolehlah titik koordinat piksel (x,y).

## 3.3. Penghitungan Sudut Posisi Motor *Platform* Senjata

Untuk menggerakkan senjata agar mengarah ke target, sudut vertikal dan horisontal dibutuhkan. Sehingga, dibutuhkan konversi dari koordinat kartesian menjadi posisi sudut vertikal dan horisontal seperti pada Gambar 3.6.

Dari ilustrsi tersebut, persamaan sudut horisontal dan sudut vertikal bisa didapatkan dari persamaan (3.1) dan (3.2) berikut:




**Gambar 3.6** Diagram Blok Pengukuran Posisi Target

$$\theta_h = \sqrt{{err_x}^2 + {err_y}^2} \quad (3.1)$$

$$\theta_h = \sqrt{(X_{target} - X_{set\ point})^2 + (Y_{target} - Y_{set\ point})^2} \quad (3.2)$$

Proses ini dilakukan di dalam Raspberry dalam bentuk perangkat lunak.

*Syntax* dari proses ini terdapat dalam mikrokontroller Arduino dan Raspberry. Pada Raspberry dilakukan proses pengolahan nilai error sedangkan pada Arduino dilakukan pengkonversian nilai error menjadi nilai sudut. Berikut ini *syntax* yang berkaitan dengan penghitungan sudut referensi motor servo.

- Penghitungan sudut horizontal

```
Int Ax = Point.x – Set_point.x; // DALAM RASPBERRY PI
```

`Ax` merupakan selisih antara posisi target dengan posisi set point dalam sumbu x. Kemudian nilai error dikirimkan kepada Arduino menggunakan komunikasi serial dan dikonversikan menjadi nilai sudut di dalam Arduino.

```
P = Ax * Kp;                    // DALAM ARDUINO
I += Ax * Ki;
D = Kd * (Ax - err);
err = Ax;
pwm = P + I + D;
myservo.write((int)pwm);
```

`pwm` merupakan nilai derajat yang akan di *write* dalam servo. Nilai nya akan merepresentasikan nilai derajat dari servo horizontal. `err` merupakan nilai error sumbu x sebelum diolah dalam kontroller PID. Dalam syntax diberikan batasan nilai maksimal dan minimal karena motor servo yang digunakan memiliki derajat maksimal 180

- Penghitungan sudut vertical

```
Int Ay = Point.y – Set_point.y;      //DALAM RASPBERRY
```

`Ay` merupakan selisih antara posisi target dengan posisi set point dalam sumbu y. Kemudian nilai error dikirimkan kepada Arduino menggunakan komunikasi serial dan dikonversikan menjadi nilai sudut di dalam Arduino.

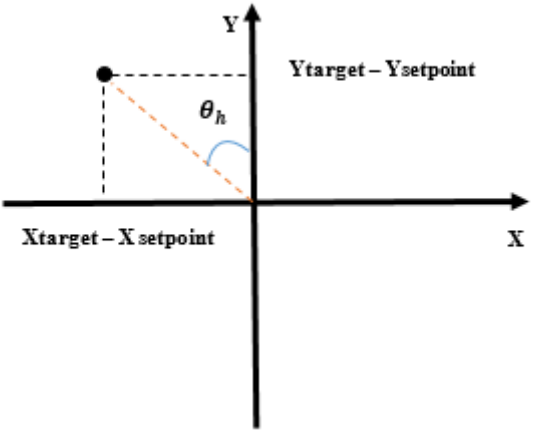
```
P = Ay * Kp;                    // DALAM ARDUINO
I += Ay * Ki;
D = Kd * (Ay - err);
```

```
err2 = Ay;
pwm2 = P + I + D;
myservo.write((int)pwm2);
```

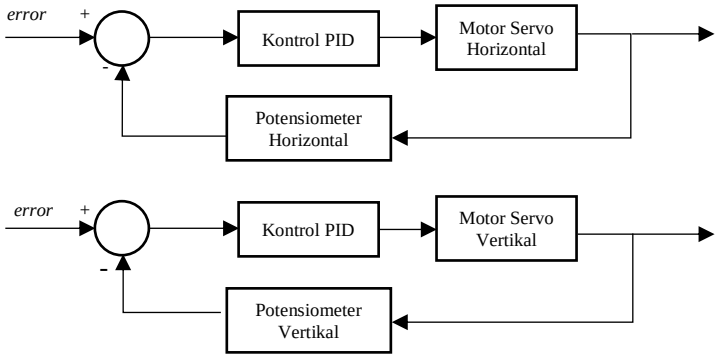
`pwm2` merupakan nilai derajat yang akan di *write* dalam servo. Nilai nya akan merepresentasikan nilai derajat dari servo vertikal. `err2` merupakan nilai error sumbu y sebelum diolah dalam kontroller PID. Dalam syntax diberikan batasan nilai maksimal dan minimal karena motor servo yang digunakan memiliki derajat maksimal 180.

## 3.4. Pengarahan Senjata ke Target

Pada bagian ini, semua proses dilakukan oleh mikrokontroller Arduino. Nilai eror yang dihasilkan oleh pemrosesan citra pada raspberry dikirimkan melalui komunikasi serial kepada Arduino. Dari sinilah Arduino mengkonversi nilai eror tersebut menjadi sudut vertikal dan sudut horizontal. Proses pengkonversian *error* menjadi sudut motor servo menggunakan kontrol PID. Kontrol PID akan menentukan berapa nilai sudut yang akan menjadi input bagi motor servo. Arduino akan menghasilkan sinyal PWM untuk menggerakkan motor servo. Pembacaan posisi aktual motor servo dilakukan oleh potensiometer untuk menentukan titik awal servo sebelum melakukan proses *tracking*. Diagram blok dari sistem ada pada Gambar 3.7.




**Gambar 3.7** Ilustrasi Konversi Koordinat Kartesian menjadi Sudut Motor




**Gambar 3.8** Diagram Blok Pengarah Senjata

### 3.4.1. Kontrol PID

Kontrol PID dalam Tugas Akhir ini merupakan keluaran nilai PWM untuk mengontrol motor servo ke posisi yang diinginkan. Dengan melakukan *tuning* Kp, Ki, dan Kd secara manual, maka akan diperoleh respon motor servo cukup cepat dan meminimumkan timbulnya osilasi. Proses *tuning* PID dilakukan secara coba-coba atau *trial and error*, sehingga diperolehlah hasil yang cukup bagus dengan hanya menggunakan nilai Ki = 0.005. Dengan diberikannya nilai Kp dan Kd hanya akan menimbulkan osilasi.

Penggunaaan kontrol PID di dalam Tugas Akhir ini diimplementasikan pada mikrokontroller Arduino nano. Algoritma yang digunakan untuk mengontrol servo horizontal dan vertikal adalah sama termasuk *threshold* kecepatan maksimum dan minimum. Selain itu diberikan juga threshold nilai error minimal untuk menghindari motor menjadi panas akibat bergetar. Motor bergetar diakibatkan oleh perubahan sudut yang kecil.

Proses kontrol PID dari sistem ini meliputi pencarian nilai *error* dilanjutkan dengan pencarian nilai *integral error* kemudian dilanjutkan dengan *derivative* nilai *error.* Ketiga nilai variable tersebut merupakan nilai *proportional, integral,* dan *derivative* yang kemudian nilainya dijumlahkan dan digunakan sebagai nilai PWM untuk mengontrol servo. Algoritma ini diimplementasikan dengan menggunakan *syntax* yang tertulis berikut ini.

- Mencari *proportional error*

```
Ax = point.x-setpoint.x; //pada raspberry
P = kp * Ax; // pada arduino
Ay = point.y-setpoint.y; //pada raspberry
P2 = kp2 * Ay; // pada arduino
```

- Mencari *integral error*

```
I += ki * Ax;
I2 += ki2 * Ay;
```

- Mencari *derivative error*

```
D = kd * (Ax - err);
err = Ax;
D2 = kd2 * (Ay - err);
Err2 = Ay;
```

- Keluaran kontrol PID

```
pwm = P + I + D;
if(pwm > 135) pwm = 135;
if(pwm < 45) pwm =45;
```

*syntax* ini difungsikan untuk memberikan *threshold* nilai minimal dan maksimal dari pwm. Pwm merupakan nilai dari motor servo dalam bentuk derajat.

Detil perhitungan ketika nilai error sebesar 100 (piksel) yaitu

I = (ki * Ax) + I
I = (0.005 * 100) + 0.5 = 1
I2 = (0.005 * 99) + 0.495 = 0.990
I3 = (0.005 * 98.01) + 0.49005 = 0.9801

Begitu seterusnya sehingga didapatkan error Ax = 0. Dalam proses perhitungan terlihat sangat lama dengan nilai decimal yang begitu kecil. Namun dikarenakan proses dalam mikrokontroller sangat cepat sehingga proses iterasi dalam sistem akan sesuai dengan yang dibutuhkan. Sistem hanya menggunakan integral kontrol sehingga proses perhitungannya hanya sampai disini.

Dalam pola algoritma kontroller PID juga terdapat *error* yang berfungsi sebagai pengontrolan output saat ini dengan output sebelumya:

$$PID = K_p\, e(t) + K_i \int e(t)\, dt + K_d \frac{de(t)}{dt}$$

Jika diumpamakan PID = Un

$$U_n = K_p e_n + K_i \sum e_n n_i = 0\ dt + K_d\ (e_n - e_n - 1dt)$$

Persamaan matematis untuk kontrol *proporsional*:

$$U(t) = K_p . e(t)$$

Dalam laplace menjadi:

$$\frac{U(s)}{E(s)} = K_p$$

Persamaan matematis untuk kontrol *integral*:

$$U(t) = K_i \int_o^t e(t)dt \qquad (37)$$

Dalam laplace :

$$\frac{U(s)}{E(s)} = \frac{K_i}{s}$$

Secara umum kontrol PI ditransformasi dalam laplace menjadi:

$$\text{PI} = K_p + \frac{K_i}{s}$$

Masukan ke pengendali PI pada robot dalam tugas akhir ini ada 2 sinyal *error*. Dua sinyal *error* yang mengandung *error* sumbu x, dan *error* sumbu y akan menghasilkan sudut perubahan $\theta_h$ yang diukur dari proses pengolahan citra menggunakan kamera. Untuk nilai e(t) dari sensor kamera didapatkan dengan persamaan sebagai berikut:

$$\text{e(t)} = \text{error} - \text{setpoint}$$

Untuk penggunaan kontrol *proporsional* (P) saja, pergerakan *turret* mengalami osilasi karena tidak ada yang digunakan untuk mengurangi osilasi pada *closed loop transfer function* yang ditunjukkan pada persamaan berikut.

$$P = Kp * e(t)$$

Untuk responnya yang masih berosilasi, maka perlu ditambahkan kontrol *integral* menjadi kontrol *proporsional integral* (PI) untuk membuat servo menjadi stabil tanpa osilasi. Dengan menambahkan nilai $K_i$, maka *transfer function* menjadi sebagai berikut.

$$PI = Kp * e(t) + Ki * \sum e * \Delta t$$

Untuk perancangan kontrol PI sudah bisa digunakan untuk menstabilkan robot. Karena robot memerlukan pergerakan dalam kecepatan tinggi untuk melakukan *tracking* terhadap titik target sehingga robot memerlukan *response time* sekecil mungkin dan dilakukan dengan menambahkan kontrol *integral.*

Dalam perancangan sistem kontrol PID kali ini tidak didefinisikan besar waktu sampling yang digunakan. Sehingga waktu samplingnya mengikuti lama waktu program dalam satu kali *loop*. Ketika *time sampling* tidak didefiniskan, maka nilainya akan berubah-ubah seiring jalannya program. Namun setelah di lihat dengan menggunakan *millis()* pada Arduino nilai *time sampling*nya adalah 50ms.

### 3.4.2. Motor Servo

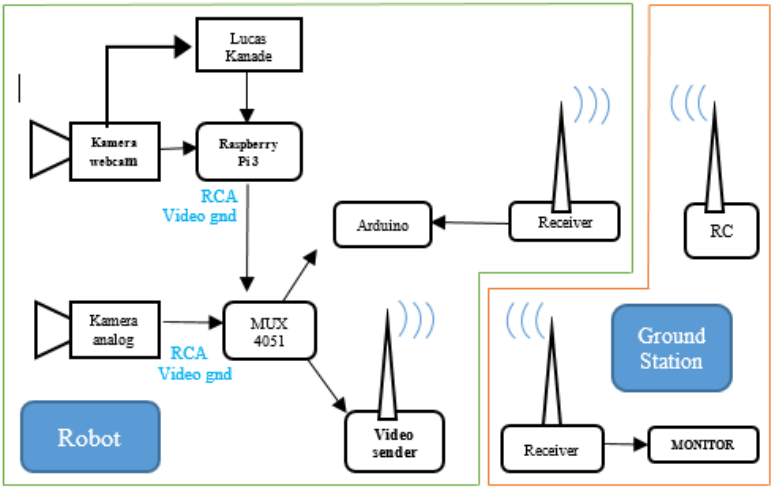
Pada Tugas Akhir ini menggunakan motor servo MG996R sebanyak 2 buah. Satu motor digunakan untuk menggerakkan motor penggerak vertikal sedangkan motor yang lain sebagai motor penggerak horizontal. Gambar 3.8 menunjukkan bentuk dari motor servo MG996R.

Torsi yang dimiliki dari motor servo ini 9.4kgf.cm (4.8V), 11 kgf.cm (6V) yang artinya motor dapat mengangkat beban maksimal hingga 9.4 Kg dengan kebutuhan tegangan suplai sebesar 4.8V. Jika dikehendaki membutuhkan torsi yang lebih besar hingga maksimal 11 Kg, maka tegangan suplai yang dibutuhkan juga lebih besar hingga 6V. Motor servo ini bekerja pada rentang tegangan 4.8V – 7.2 V yang berarti diluar nilai rentang tersebut servo tidak akan bekerja atau akan rusak.

## 3.5. Pengiriman Video

Pada bagian ini terdapat proses pengiriman berupa video yang bertujuan untuk memudahkan operator dalam melakukan kontrol robot. Video yang dapat ditampilkan nantinya merupakan tampilan layar raspberry berisi program tracking object yang telah di run sejak saat booting serta tampilan video dari kamera analog yang digunakan untuk navigasi UGV (unmanned ground vehicle). Gambar 3.9 merupakan blok diagram dari proses pengiriman video.


**Gambar 3.9** Motor Servo MG996R [16]




**Gambar 3.10** Blok Diagram Proses Pengiriman Video

## 3.6. *RCJoy Remote Control*

Remote control merupakan interface antara user (operator) dengan mesin. Sehingga sebelum masuk ke sistem perlu dilakukan pembahasan terlebih dahulu mengenai remote control karena semua proses akan berhubungan dengan user interface ini. Pada Gambar 3.10 dan Gambar 3.11 merupakan rancangan GUI yang telah dibuat untuk mengkonfigurasikan RCJoy remote control. Konfigurasi yang dibuat

melalui GUI digunakan untuk mendefinisikan channel pada receiver sehingga joystick dapat dikontrol channel 1 untuk gerak vertikal, channel 2 untuk gerak horizontal serta channel 3 untuk mode.

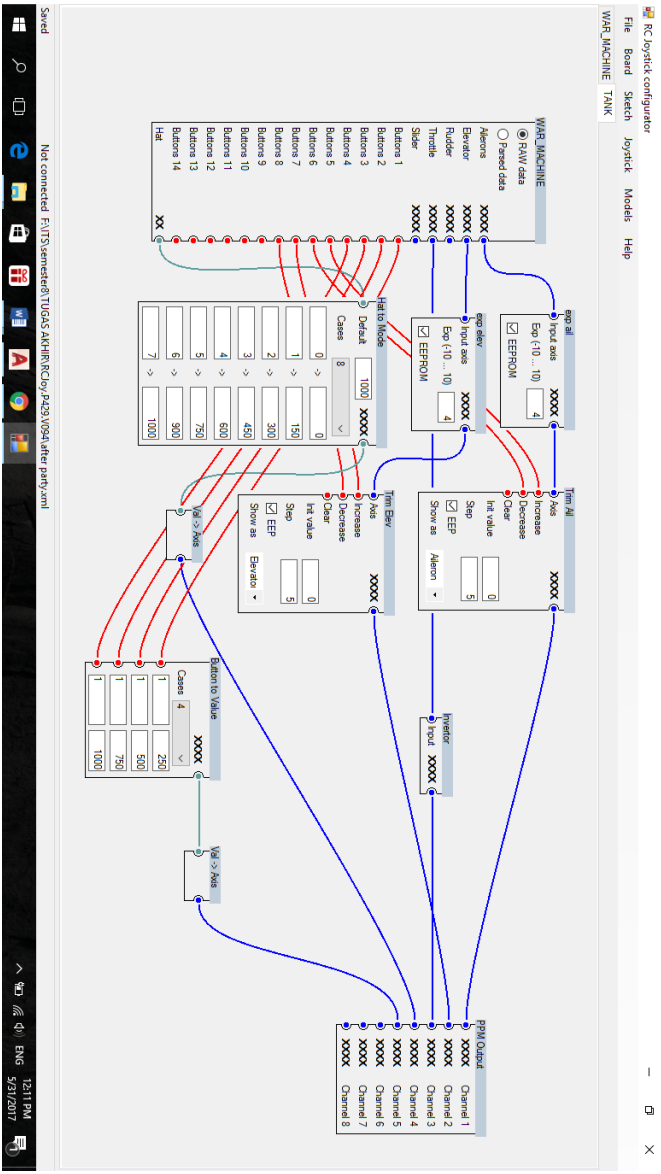
Gambar 3.10 merupakan GUI utama yang digunakan dalam remote control. Pada blok paling kiri adalah konfigurasi joystick yang memiliki 5 axis node, 14 tombol node tipe button, 1 hat node yang berisi nilai. Pada blok paling kanan adalah keluaran PPM dari 8 saluran input. Dapat dilihat bahwa terdapat dua blok eksponen, satu untuk control ailerons satu untuk elevator. Kedua blok eksponen memiliki 1 input dan 1 node output link tipe axis. Dalam salah satu blok eksponen juga mencentang kotak centang "EEPROM". Hal tersebut artinya tingkat eksponen blok ini bervarisasi dan bisa diubah melalui GUI. Kemudian dua blok selanjutnya adalah trimmer yang berfungsi untuk memangkas. Keduanya mengambil exp yang sesuai sebagai masukan dan beberapa tombol joystick untuk mengurangi dan manambahi. Setiap klik tombol itu akan mengubah nilai pemangkas. Juga kedua pemangkas memiliki kotak centang "EEP" ditandai, dan ini berarti nilai pemangkas akan disimpan dalam memori / EEPROM dan dipulihkan pada board berikutnya.

Gambar 3.11 merupakan GUI pendukung yang bertujuan untuk melakukan mode yang berbeda dari kontrol UGV dan servo. Pada konfigurasi GUI ini menggunakan mode throttle dalam menjalankan UGV. Jadi ketika throttle dinaikkan pada kecepatan tertentu, UGV akan berjalan dalam kecepatan konstan. Fungsi ailerons dan elevator nantinya hanya menjadi logic untuk maju, mundur serta berbelok saja. Namun sistem dirasa lebih *friendly use* pada konfigurasi yang pertama dibandingkan dengan konfigurasi kedua.

## 3.7. Kontrol Manual Oleh *Remote Control*

Pada bagian ini semua proses kontrol dilakukan secara manual menggunakan *remote*. Pembacaan nilai receiver yang dilakukan oleh mikokontroller Arduino diolah untuk dijadikan acuan dalam mengontrol nilai PWM. Terdapat 2 macam motor yang akan dikontrol dalam mode manual ini, yang pertama adalah motor servo untuk menggerakkan senjata dan yang kedua adalah motor *power window* yang digunakan untuk menggerakan UGV (*unmanned ground vehicle*). Dalam menggerakkan motor *power window* ini dibutuhkan bantuan driver motor L298N seperti yang ditunjukkan pada Gambar 3.12.

**Gambar 3.11** GUI Utama

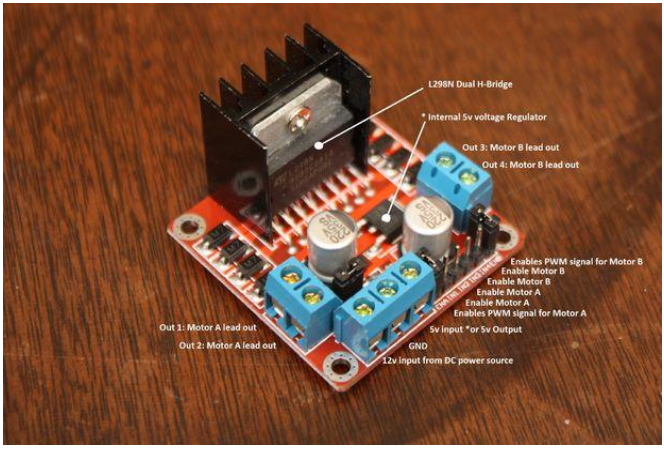



**Gambar 3.12** GUI pendukung

### 3.7.1. *Driver* Motor

Pada Tugas Akhir ini digunakan *driver* motor L298N sebanyak 1 buah. Dalam 1 driver tersebut terdiri dari *dual-bridge* sehingga sudah dapat menggerakkan 2 motor sekaligus. Pada *single-bridge* yang pertama digunakan untuk menggerakkan motor penggerak kiri dan *single-bridge* lainnya digunakan untuk menggerakkan motor penggerak kanan. Prinsip dalam menggerakkan UGV ini menggunakan konsep *differential steering* sehingga keadaan lurus, mundur maupun berbelok dipengaruhi oleh perbedaan putaran yang dihasilkan antara motor kanan dan motor kiri. Gambar 3.12 menunjukkan bentuk dari *driver* motor L298N.

Tegangan suplai yang dapat digunakan ada pada rentang 7-35V, sedangkan dalam tugas akhir ini digunakan sebesar 12 Volt. Terdapat 3 input lain, yaitu IN1, IN2, ENA, IN3, IN4, dan ENB. IN1 dan IN2 merupakan pengatur arah gerak motor yang menentukan kutub positif dan negatif dari OUT1 dan OUT2. Sedangkan ENA merupakan input sinyal PWM yang berfungsi untuk mengatur kecepatan motor. Demikian halnya dengan IN3, IN4, dan ENB memiliki fungsi yang sama dengan IN1, IN2, dan ENA perbedaannya hanya pada output yang diatur. Pada kali ini output yang diatur adalah OUT3, dan OUT4.




**Gambar 3.13** *Driver* Motor L298N

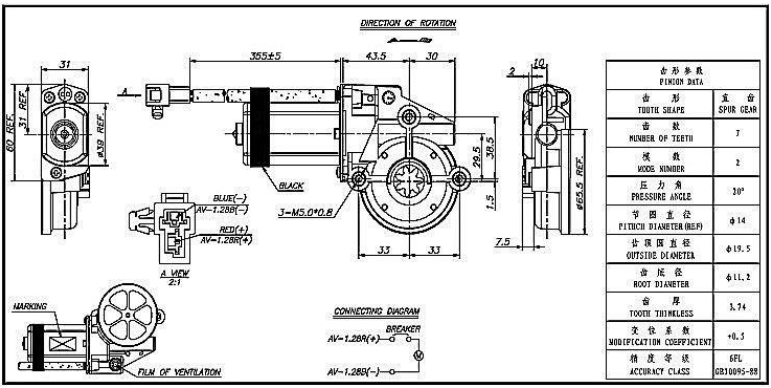
### 3.7.2. Motor DC

Penggerak dari UGV (*unmanned ground vehicle*) berupa motor DC yang biasanya digunakan sebagai *power window* pada mobil. Tujuan digunakannya motor ini agar mendapatkan torsi yang besar dan RPM yang kecil. Dalam melakukan proses *tracking* tidak perlu melakukan pergerakan yang besar agar titik fitur pada proses pengolahan citra tidak mudah hilang. Gambar 3.13 merupakan bentuk dari motor DC *power window.*

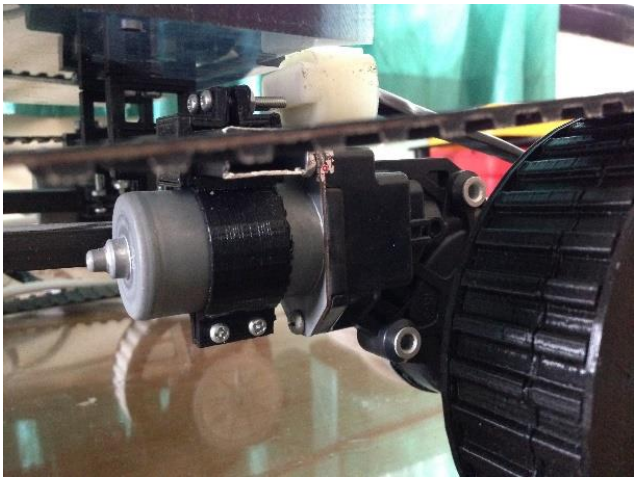
Tegangan suplai yang digunakan adalah 12VDC. Arah pergerakan motor ditentukan oleh pemberian tegangan DC pada kedua input motor. Torsi yang dihasilkan dari motor ini cukup besar yaitu 30Kg.cm yang artinya motor ini dapat menggerakkan beban maksimal hingga 30Kg. Gambar 3.13 dan Gambar 3.14 memberikan gambaran motor DC yang digunakan pada tugas akhir ini beserta peletakannya.

## 3.8. Fitur Pendukung

Terdapat beberapa fitur pendukung untuk mengoptimalisasi kerja dari sistem agar berjalan dengan baik. Adapun beberapa fitur pendukung dari sistem ini diantaranya adalah komunikasi serial antara raspberry dan mikrokontroller Arduino, dan berbagai macam mode yang dapat dikontrol melalui *remote control*.




**Gambar 3.14** Motor DC *power window* [17]



**Gambar 3.15** Motor DC Penggerak UGV

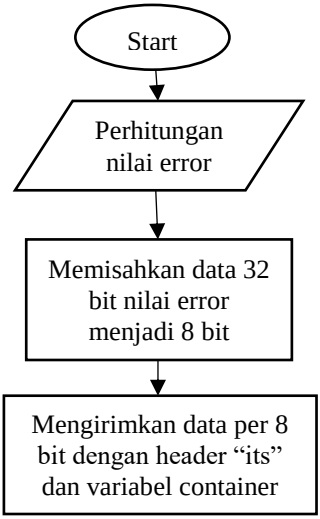
### 3.8.1. Komunikasi Serial

Perancangan komunikasi serial merupakan komponen penting di dalam sistem ini. Hal ini dikarenakan informasi yang telah diolah di dalam raspberry berupa nilai *error* yang dibutuhkan mikrokontroller untuk mengkonversinya dalam nilai PWM harus dikirimkan. Sistem ini terdiri dari dua *processor* sehingga perancangan komunikasi serial terbagi menjadi dua, yaitu bagian penerimaan di mikokontroller dan bagian pengiriman dari raspberry. Pengaturan komunikasi serial yang dipakai adalah

- *Baud rate* : 9600 bps
- Jumlah data tiap pengiriman : 8 bit
- *Stop* bit : 1 bit
- *Parity* bit : tidak ada
- *Flow control* : tidak ada

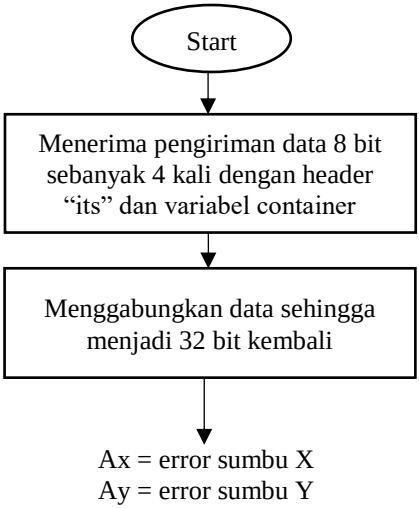
#### *3.8.1.1. Pengiriman Data dari Raspberry*

*Syntax* pengaturan pengiriman dari Raspberry agar sesuai dengan aturan komunikasi serial adaalah sebagai berikut:




#### *3.8.1.2. Penerimaan Data pada Arduino Nano*

Pengaturan komunikasi serial pada Arduino Nano agar dapat menerima data dari komputer adalah sebagai berikut:



### 3.8.2. Mode *Remote Control*

Untuk memfungsikan beberapa tombol dalam *remote rcjoy* perlu dilakukan pendefinisian program untuk menjalankan beberapa mode perintah yang akan dilakukan. Terdapat 6 mode perintah untuk melakukan pengontrolan sistem senjata melalui *remote control.* Terdapat 2 mode secara makro yaitu mode manual dan otomatis serta ditambah dengan 4 mode tambahan untuk mendukung sistem *tracking.* Adapun keenam mode yang digunakan adalah

1. Mode Gerak servo
   Merupakan kontrol manual dari *remote* untuk menggerakkan senjata.
2. Mode Gerak Kontrol Tembak
   Merupakan kontrol manual untuk menggerakkan servo pada pelatuk senjata, sehingga ketika servo bergerak senjata akan menembakkan peluru plastik.
3. Mode Klik Titik
   Merupakan kontrol untuk melakukan klik titik fitur pada proses *tracking.* Ketika di lakukan proses klik maka titik fitur akan selalu muncul ditengah-tengah frame.
4. Mode Hapus Titik
   Merupakan kontrol untuk melakukan penghapusan terhadap titik fitur karena kesalahan. Dengan menghapus titik fitur maka secara otomatis proses *tracking* akan berhenti dan servo akan mengarahkan senjata pada posisi 0.
5. Mode Gerak Otomatis
   Merupakan mode gerak servo secara otomatis yaitu proses komunikasi serial antara raspberry dengan mikokontroller Arduino dengan mengirimkan nilai *error.* Proses selanjutnya adalah dengan mengkonversi nilai *error* menjadi nilai PWM untuk mengontrol nilai sudut servo baik horizontal maupun vertikal.
6. Mode Gerak UGV
   Merupakan kontrol manual untuk menggerakkan UGV. Dengan demikian UGV dapat digerakkan maju, mundur serta berbelok sesuai keinginan *operator.*

# BAB IV
# PENGUJIAN DAN ANALISA

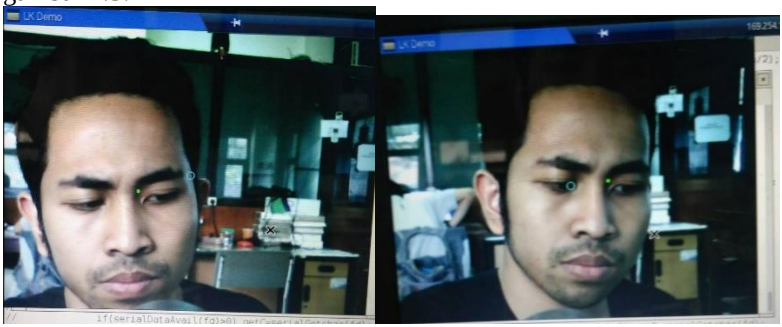
Pengujian sistem yang telah dirancang terdiri dari beberapa tahap bagian, dimulai dari pengujian secara perangkat keras dan perangkat lunak. Tujuan dari pengujian ini adalah untuk mengetahui apakah tujuan dalam perancangan sistem pada Tugas Akhir ini telah terlaksana atau tidak. Pengujian pada bab ini terdiri dari pengujian *tracking object*, pengujian kontrol manual.

## 4.1. Pengujian *Tracking object*

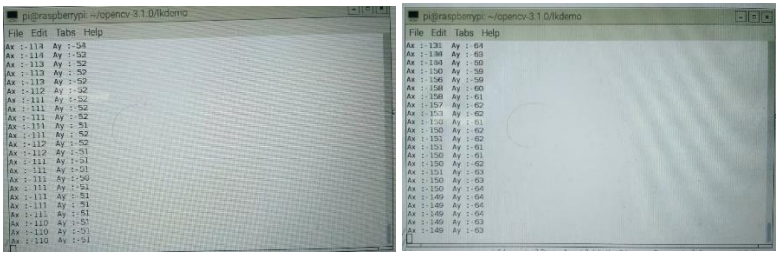
Pengujian ini dilakukan untuk mengetahui berapa besar ketelitian dalam melakukan *tracking* titik fitur. Ketelitian ini kerap berpengaruh terhadap proses kontroller nantinya. Semakin bagus ketelitian *tracking* titik fitur, semakin lembut pergerakan motor servo dalam memposisikan titik fitur ke posisi tengah. Gambar 4.1 menunjukkan hasil *tracking* dengan menggunakan algoritma lucas-kanade. Gambar 4.2 merupakan keluaran nilai Ax dan Ay yang dimonitor melalui komunikasi serial.

## 4.2. Realisasi Desain UGV dan *turret* senjata

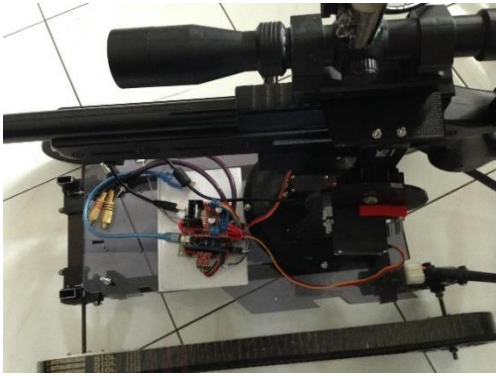
Rangka UGV menggunakan bahan PLA yang dicetak menggunakan 3D *printing*. Bentuk dan desain dibuat menggunakan software *auto cad* dengan ukuran 55X35 cm. Realisasi desain robot ditunjukkan pada gambar 4.3.




**Gambar 4.1** Hasil *tracking* pada titik hijau




**Gambar 4.2** Serial Monitor Nilai Ax dan Ay

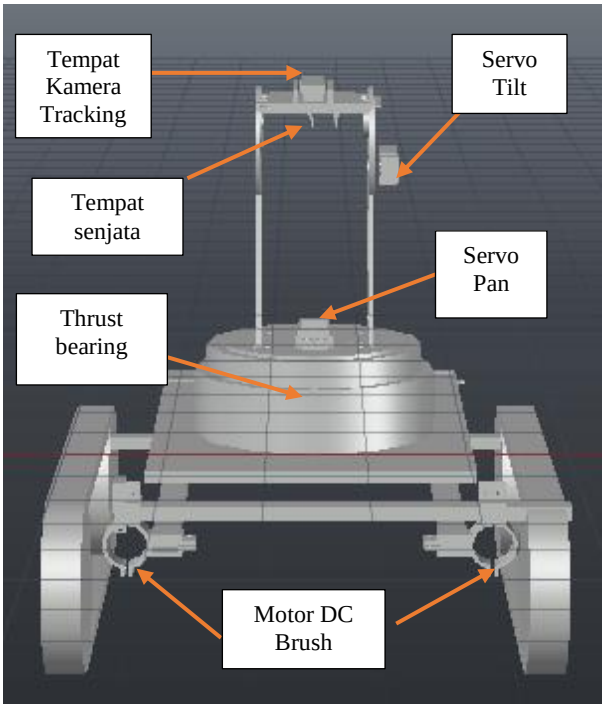


**Gambar 4.3** Peletakan Rangkaian Elektronik

Selain itu desain untuk *turret* senjata didesain dengan ukuran tinggi 25 cm dengan tujuan untuk memudahkan gerak bebas dari senjata. Dengan menggunakan tinggi tersebut senjata dapat dimiringkan hingga sudut 45 derajat secara vertikal. Sehingga kemungkinan senjata melakukan *tracking object* yang posisinya berada diatas robot atau mungkin dibawah robot masih bisa terjadi. Gambar 4.4 merupakan desain UGV dan Gambar 4.5 merupakan desain turret senjata.

### 4.2.1. Perancangan Kontruksi Mekanik

Dalam menjalankan sistem diperlukan mekanik untuk menunjang pergerakan dari robot. Bentuk dan desain dari mekanik perlu dirancang dan di desain sesuai dengan ukuran senjata laras panjang. Desain dasar *turret* menggunakan *thrust bearing* untuk memudahkan pergerakan dari

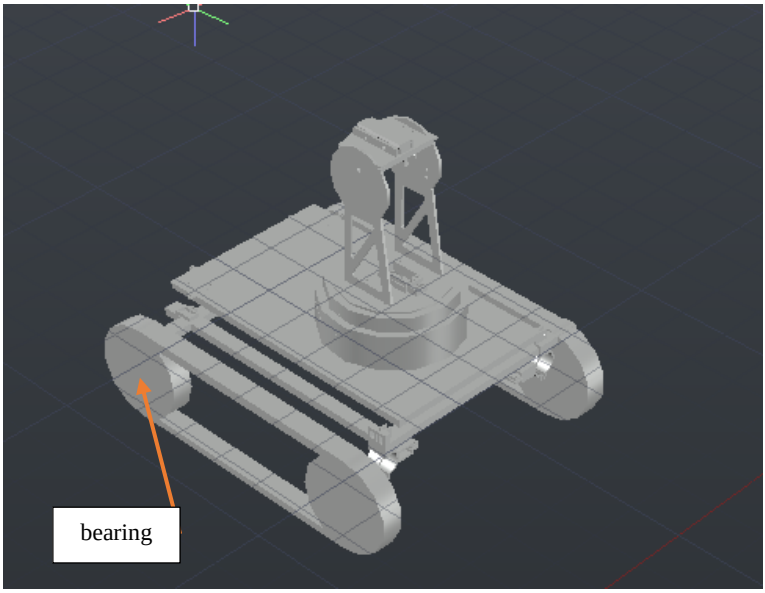



**Gambar 4.4** Konstruksi Mekanik

turret dan mengurangi gesekan antar plastik. Hal tersebut berfungsi untuk pergerakan senjata secara horizontal.

Pada desain mounting senjata dan kamera diberikan bearing untuk memperlancar pergerakan senjata dalam gerak vertikal. Kemudian pada perancangan roda dalam kendaraan darat tanpa awak menggunakan 2 *bearing* untuk menghindari roda goyah. Adapun desain konstruksi mekanik terlihat pada Gambar 4.4 dan desain 3 dimensi dari *autocad* terlihat pada Gambar 4.5.

Pada desain 3 dimensi dilihat dari sudut pandang belakang kiri robot. Pemasangan bearing hanya dipasang pada roda depan saja, karena roda depan hanya membutuhkan pergerakan bebas yang dikopel dari pergerakan roda belakang. Roda belakang digerakkan oleh motor DC




**Gambar 4.5** Desain 3 Dimensi

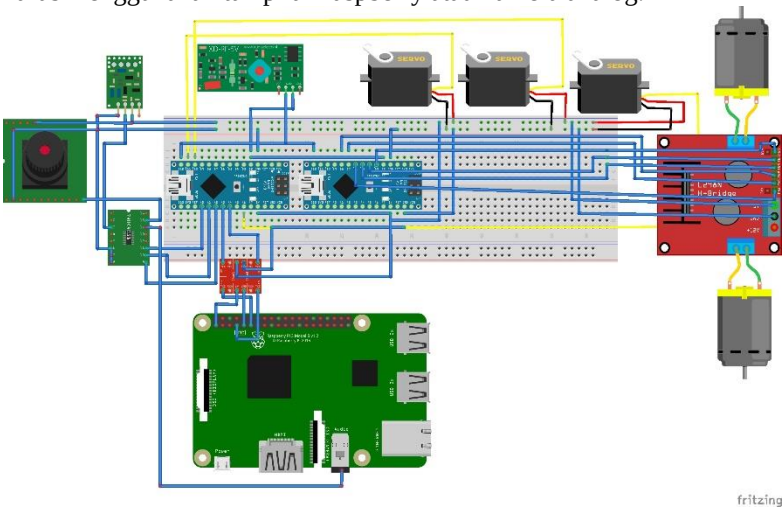
yang diberikan as roda yang di las dalam *gearbox* motor power window. Sehingga pergerakan motor diikuti pergerakan roda belakang serta depan.

## 4.3. Pengujian Perangkat Keras

Dalam melakukan pengujian perangkat keras dilakukan dengan menguji hasil output dari setiap rangkain elektronika. Pengujian yang dilakukan meliputi pengujian rangkaian *buck converter* yang digunakan sebagai supply, pengujian *remote control*, pengujian rangkaian multiplekser 4051. Pengujian dilakukan untuk memenuhi batasan-batasan masalah pada tugas akhir ini.

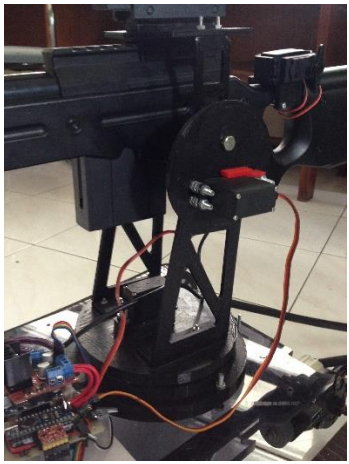
Pada Gambar 4.6 merupakan skema rangkaian yang dibuat dalam Tugas Akhir ini. Terdapat 2 arduino untuk melakukan kontrol mode otomatis dan mode manual. Pada Arduino bagian kiri merupakan mikrokontroller yang mengatur proses mode otomatis, seluruh pergerakan servo, dan *switch* tampilan monitor. Sedangkan pada Arduino bagian kanan mengatur proses kontrol manual pada UGV. Data pada receiver diparalel untuk 2 mikrokontroller Arduino agar dapat dilakukan proses pembacaan nilai receiver di kedua Arduino. *Switch* tampilan

monitor menggunakan multiplekser 4051 sehingga dapat diatur kapan harus menggunakan tampilan raspberry atau kamera analog.

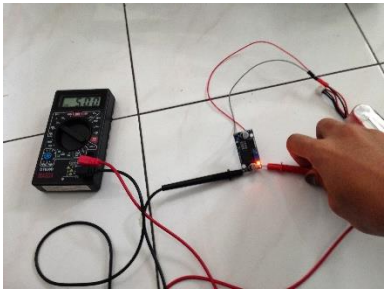



**Gambar 4.6** Skema Rangkaian sistem

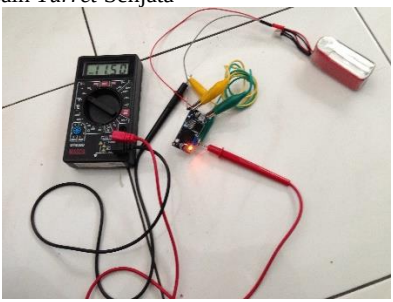
**Gambar 4.7** Desain UGV (*unmanned ground vehicle*)



**Gambar 4.8** Desain *Turret* Senjata



**Gambar 4.9** Tegangan Output



**Gambar 4.10** Tegangan Input

### 4.3.1. Pengujian Suplai *buck converter*

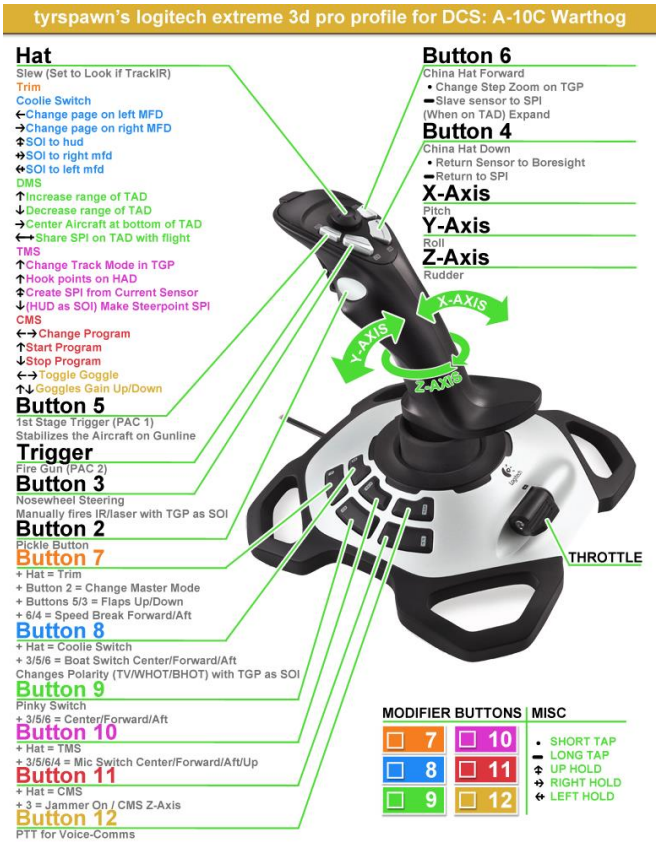
Sebuah sistem tidak akan bisa berjalan tanpa sebuah suplai. Rangkaian suplai ini merupakan rangkaian yang penting dalam sebuah sistem karena digunakan sebagai sumber daya dari seluruh sub elektrik. Rangkaian ini berfungsi untuk mengkonversikan tegangan lipo 3 sel yang memiliki tegangan 12.6 VDC menjadi tegangan 5 VDC. Rangkaian ini digunakan untuk mensuplai tegangan dari 3 motor servo.

Untuk mendapatkan tegangan 5VDC pada rangkaian *buck converter*, diperlukan memutar *multitune* hingga mendapatkan nilai

output tegangan seperti pada Gambar 4.6. Tegangan input sebesar 12.6 VDC yang ditunjukkan Gambar 4.7 digunakan untuk mensuplai motor DC *power window*, kamera analog, dan *transmitter* video. Dalam pengujian sempat menggunakan IC regulator 7805, namun penggunaanya tidak tahan lama dan mudah panas hingga putus sehingga diputuskan untuk menggunakan *buck converter*.

### 4.3.2. Pengujian *remote control*

Pada pengujian kali ini adalah mengenai *remote control*. Pengujian ini dirasa penting karena perlu dilakukan kalibrasi untuk




**Gambar 4.11** Logitech extreme 3D Pro Joystick [18]

menghasilkan pembacaan *remote* yang tepat. Receiver yang digunakan pada Tugas Akhir ini memiliki 8 channel input yang dapat dibaca mikrontroller, namun hanya digunakan 3 channel untuk mengontrol UGV dan *turret* senjata. Channel pertama digunakan untuk y axis, channel kedua untuk x axis, dan channel ketiga digunakan untuk mode. Gambar 4.8 merupakan jenis *remote control* yang dipakai

**Tabel 4.2** Hasil Pengujian Nilai Pulsa dari *receiver* channel 1 dan 2

| Nilai Pulsa (ms) | Posisi *remote* (cm) | |
|---|---|---|
| | X | Y |
| 867 | 1.9987 | 2.0076 |
| 950 | 1.3323 | 1.3191 |
| 1100 | 0.6667 | 0.6565 |
| 1275 | -0.0001 | 0.0012 |
| 1400 | -0.6678 | -0.6656 |
| 1550 | -1.3345 | -1.3451 |
| 1733 | -2.0012 | -2.0001 |

**Tabel 4.3** Hasil Pengujian Nilai Pulsa *receiver* channel 3

| Nilai pulsa (ms) | Tombol yang ditekan |
|---|---|
| +-1680 | Tombol *trigger* |
| +-1590 | Tombol 2 |
| +-1480 | Tombol 3 |
| +-1280 | Tombol 4 |
| +-1097 | Tombol 3 + 4 |
| +-1380 | Tombol 3 + 2 |
| +-1440 | Tombol 3 + *trigger* |
| +-1150 | Tombol 4 + 2 |
| +-1240 | Tombol 4 + *trigger* |
| +-1550 | Tombol 2 + *trigger* |
| +-910 | Tombol 4 + 3 + 2 |
| +-1000 | Tombol 4 + 3 + 1 |
| +-1110 | Tombol 4 + 2 + 1 |
| +-1310 | Tombol 3 + 2 + 1 |

Dari data yang diperoleh pada Tabel 4.2 dan Tabel 4.3, digunakan sebagai acuan dalam melakukan kontrol manual oleh *remote control.* Channel 1 dan channel 2 digunakan untuk mengatur nilai PWM dengan mengkonversi nilai pulsa receiver yang dibaca oleh Arduino menjadi nilai PWM. Nilai PWM ini nantinya dapat digunakan untuk mengontrol motor DC *power window* atau motor servo. Kemudian channel 3 pada receiver digunakan untuk mode yang dibutuhkan dalam sistem yang telah dirancang. Terdapat 14 perbedaan nilai pulsa di setiap tombol yang ditekan beserta kombinasinya. Dalam sistem Tugas Akhir yang dibuat hanya menggunakan 6 mode yang terdiri dari 2 mode utama dan 4 mode pendukung.

### 4.3.3. Pengujian Multiplekser 4051

Dalam melakukan proses transmisi video perlu dilakukan pemilihan dari video manakah yang akan ditampilkan dalam layar. Terdapat 2 macam video yang akan ditampilkan pada layar, yaitu yang pertama tampilan dari raspberry dan yang kedua adalah tampilan dari kamera analog pada UGV. Untuk itu diperlukanlah rangkaian multiplekser 4051 yang berfungsi sebagai selektor. Pin selektor dari 4051 nantinya dihubungkan dengan pin pada mikrokontroller Arduino sehingga proses selektornya dapat diproses di dalam Arduino. Gambar 4.9 merupakan rangkaian dari Multiplekser.

### 4.3.4. Pengujian menampilkan raspberry pada layar monitor

Pada bagian ini dilakukan pengujian menampilkan tampilan raspberry menggunakan kabel RCA. Perlu dilakukan beberapa pengaturan pada raspberry agar tampilan dapat ditampilkan melalui kabel RCA. Pengaturan *default* dari raspberry adalah dengan menggunakan kabel HDMI.

## 4.4. *Tuning* PID dan Pengujian Kontrol *Platform* Senjata Berdasarkan *error* Sumbu X dan Sumbu Y

Di dalam pengujian ini, nilai Kp, Ki, dan Kd diubah-ubah dan dicari agar pergerakan motor servo senjata berosilasi seminimum mungkin ketika diberi suatu *setpoint*. Nilai *setpoint* merupakan titik tengah dari frame yang telah dirancang sebelumnya. Nilai PID dicari secara manual

hingga mendapatkan hasil terbaik dengan rise time tercepat, overshoot dan error steady state terkecil. Percobaan secara manual dengan cara menaikkan nilai Kp secara perlahan dan melihat respon osilasinya, kemudian dengan menaikkan nilai Kd dan melihat respon berkurangnya osilasi, dan yang terakhir adalah dengan menaikkan nilai Ki dengan melihat nilai error steady state nya.

**Tabel 4.4** Hasil Tuning PID untuk Motor Penggerak Horisontal

| No | Kp | Ki | Kd | Perubahan Sudut | Sudut Maksimum | Simpangan Osilasi | Lama Osilasi |
|---|---|---|---|---|---|---|---|
| 1. | 0.1 | 0.0 | 0.0 | 80$^o$ | 120$^o$ | Sangat Besar | >10s |
| 2. | 0.01 | 0.0 | 0.0 | 60$^o$ | 110$^o$ | Besar | 2s |
| 3. | 0.01 | 0.0 | 0.1 | 50$^o$ | 110$^o$ | Besar | 2s |
| 4. | 0.01 | 0.0 | 0.01 | 30$^o$ | 105$^o$ | Kecil | 1s |
| 5. | 0.01 | 0.0 | 0.001 | 5$^o$ | 85$^o$ | Tidak ada | - |
| 6. | 0.0 | 0.01 | 0.01 | 30$^o$ | 110$^o$ | Besar | 5s |
| 7. | 0.0 | 0.005 | 0.01 | 40$^o$ | 110$^o$ | Besar | 5s |
| 8. | 0.0005 | 0.0005 | 0.01 | 10$^o$ | 95$^o$ | Sangat kecil | <1s |
| 9. | 0.04 | 0.0025 | 0.0 | 5$^o$ | 91$^o$ | Sangat kecil | <1s |

Tabel 4.4 menunjukkan pengujian nilai Kp, Ki, dan Kd dengan *tuning* manual untuk motor penggerak horisontal *platform* kamera. Hasilnya adalah, nilai Kp, Ki, dan Kd terbaik adalah masing-masing 0.0, 0.0005, dan 0.0. Dengan demikian, kontroler terbaik untuk motor jenis yang dipakai pada TA ini adalah kontroler I. Selain itu, nilai sudut maksimum dibatasi 91%. Dengan nilai-nilai ini, ketika motor mencapai *setpoint*, motor tidak bergetar.

Untuk motor penggerak vertikal, nilai Kp, Ki, dan Kd yang dipakai sama dengan yang dipakai untuk motor penggerak horisontal, seperti yang tertera pada Tabel 4.5. Dengan nilai ini, osilasi tidak terjadi.

Setelah nilai Kp, Ki, dan Kd serta sudut maksimum untuk kedua motor servo *platform* senjata didapatkan, sistem diuji untuk menggerakkan motor vertikal dan motor horisontal bersama-sama dengan

*setpoint* sudut horisontal dan vertikal yang diatur melalui komputer. Hasil pengujian tertera pada Tabel 4.6. Dari pengamatan yang dilakukan, sistem memiliki respon yang cukup cepat dan osilasi hampir tidak terjadi ketika senjata sudah ada di posisi *setpoint*.

**Tabel 4.5** Hasil Tuning PID untuk Motor Penggerak Vertikal

| No | Kp | Ki | Kd | Perubahan Sudut | Sudut Maksimum | Simpangan Osilasi | Lama Osilasi |
|---|---|---|---|---|---|---|---|
| 1. | 0.1 | 0.0 | 0.0 | 80$^o$ | 120$^o$ | Sangat Besar | >10s |
| 2. | 0.01 | 0.0 | 0.0 | 60$^o$ | 110$^o$ | Besar | 2s |
| 3. | 0.01 | 0.0 | 0.1 | 50$^o$ | 110$^o$ | Besar | 2s |
| 4. | 0.01 | 0.0 | 0.01 | 30$^o$ | 105$^o$ | Kecil | 1s |
| 5. | 0.01 | 0.0 | 0.001 | 5$^o$ | 85$^o$ | Tidak ada | - |
| 6. | 0.0 | 0.01 | 0.01 | 30$^o$ | 110$^o$ | Besar | 5s |
| 7. | 0.0 | 0.005 | 0.01 | 40$^o$ | 110$^o$ | Besar | 5s |
| 8. | 0.001 | 0.00005 | 0.01 | 10$^o$ | 95$^o$ | Sangat kecil | <1s |
| 9. | 0.08 | 0.005 | 0.0 | 5$^o$ | 91$^o$ | Sangat kecil | <1s |

**Tabel 4.6** Pengujian Pengarahan Senjata Melalu Komputer

| No. | Sudut Horisontal | Sudut Vertikal | Keterangan |
|---|---|---|---|
| 1. | 10$^o$ | 10$^o$ | Respon kedua motor cepat dan osilasi sangat kecil di akhir |
| 2. | -50$^o$ | -15$^o$ | Respon kedua motor cepat dan osilasi sangat kecil di akhir |
| 3. | 70$^o$ | 20$^o$ | Respon kedua motor cepat dan osilasi sangat kecil di akhir |
| 4. | -70$^o$ | 0$^o$ | Respon kedua motor cepat dan osilasi sangat kecil di akhir |

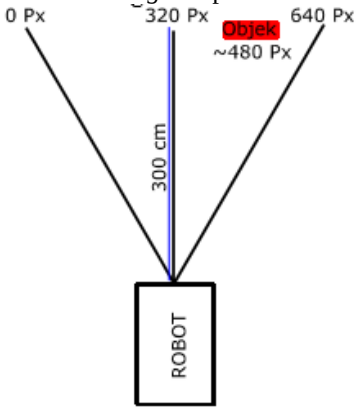
## 4.5. Pengujian Kontrol Bidik

Pengujian ini dilakukan untuk melihat respon pergerakan *turret* senjata kekanan, kekiri, keatas dan kebawah dalam mengarahkan senjata. Pengujian ini berhubungan dengan kontrol integral robot. Pengujian ini juga berhubungan dengan ketepatan robot dan mengarahkan senjata. Pengujian ini dilakukan dengan cara menghitung error antara titik target dengan titik set point dalam satuan pixel.

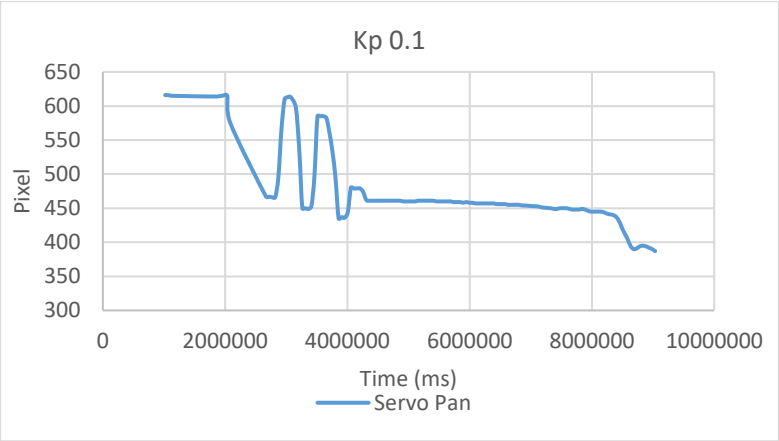
Pengujian ini dilakukan dengan cara meletakkan robot pada jarak tertentu dengan objek yaitu 100 cm. Kemudian benda diletakkan di sudut tertentu dari robot. Ilustrasi pengujian dapat dilihat di Gambar 4.9. Benda diletakkan dalam posisi tertentu sehingga pada kamera benda terlihat pada posisi sumbu tertentu. Robot dan benda juga ditempatkan pada ruangan dengan pencahayaan tetap. Hal ini untuk meminimalisir pengaruh pencahayaan dan perubahan latar belakang (*background*) percobaan dilakukan sebanyak 7 kali dengan posisi benda yang berbeda kemudian di jalankan proses *tracking.*

Dalam pengujian ini didapatkan tiga jenis data yaitu *rising time, overshoot,* dan *error steady state. Rise time* adalah waktu yang dibutuhkan saat pertama kali *tracking* dilakukan hingga mencapai titik yang diinginkan.

*Rise time* bisa dianalogikan sebagai kecepatan *turret* mengarahkan senjata saat pertama kali titik *tracking* didapatkan. *Overshoot* adalah error



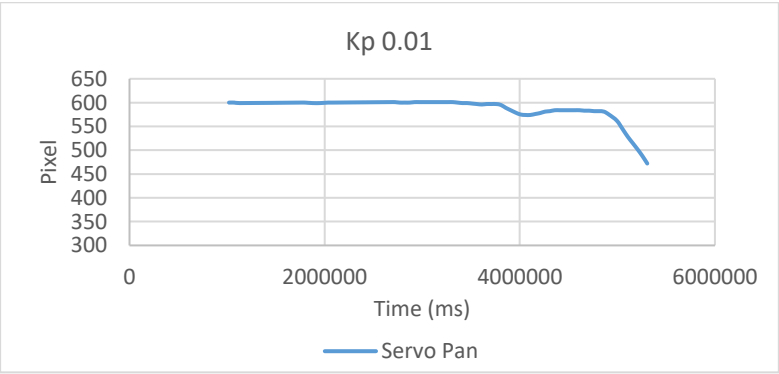

**Gambar 4.12** Ilustrasi peletakan benda dari robot. [19]



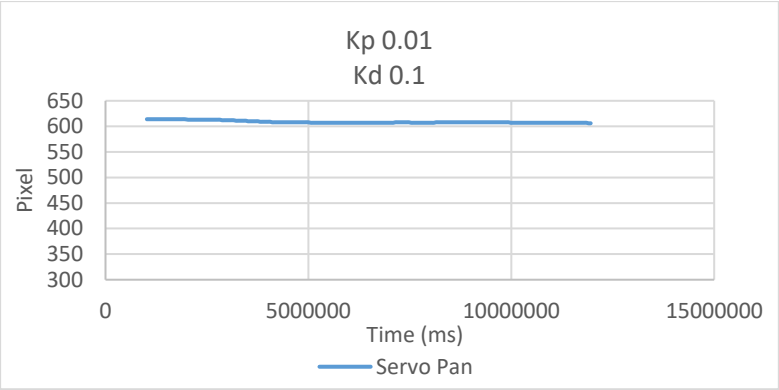

**Gambar 4.13** Grafik Pengaruh respon *rise time* sumbu x

terbesar saat melakukan *tracking. Overshoot* biasanya merupakan error pertama kali dalam *tracking*. *Error steady state* adalah error yang dihasilkan saat sistem sudah mencapai titik *steady state*. Pada pengujian ini diambil data pengaruh posisi benda terhadap kecepatan *rise time*, *overshoot*, dan *error steady state*. Nilai *rise time* berdasarkan tuning PID secara manual didapatkan hasil respon yang berbeda-beda. Berikut ini adalah data *rise time* yang didapatkan dari sumbu x.

Pada *tuning* PID menggunakan *proportional* dengan nilai Kp 0.1 didapatkan osilasi yang besar tidak pada rentang set point yaitu 320 pixel. Sehingga perlu dikurangkan nilai Kp menjadi 0.01untuk mengurangi osilasi



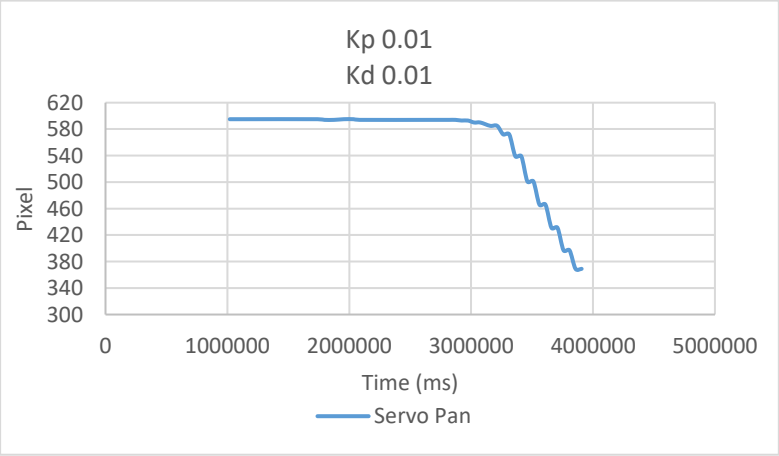

**Gambar 4.14** Grafik Pengaruh respon *rise time* sumbu x



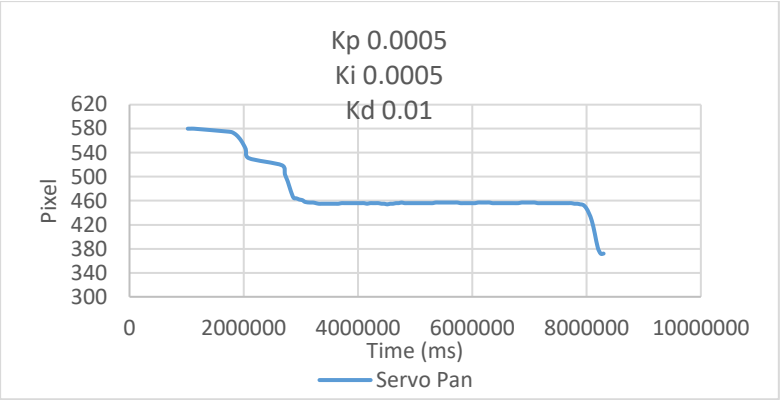

**Gambar 4.13** Grafik Pengaruh respon *rise time* sumbu x

Ketika respon ditambahkan nilai Kd terlihat pada Gambar 4.13 tidak terjadi aksi dalam mendekati set point. Sistem servo hanya berhenti pada posisi 600 pixel saja. Sehingga ditarik kesimpulan bahwa peran *derivative* adalah memperlambat gerak sistem menuju set point.

Peran *derivative* dalam *tuning* sebelumya terlalu besar nilai hambatan sistem menuju set point, sehingga perlu dikurangi nilai Kd agar sistem dapat bergerak menuju set point. Pengurangan nilai Kd sebesar 0.1 diubah menjadi 0.01. Terlihat pada Gambar 4.14 bahwa sistem mengalami perubahan mendekati set point, namun sistem masih belum memenuhi syarat mempertahankan nilai steady state.



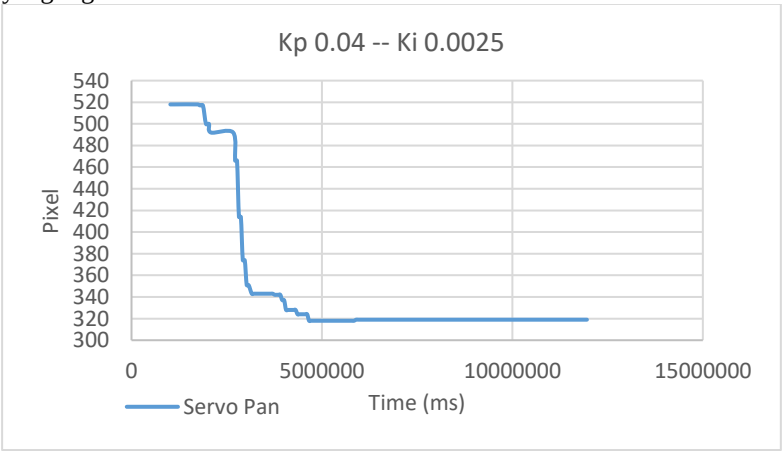

**Gambar 4.14** Grafik Pengaruh respon *rise time* sumbu x



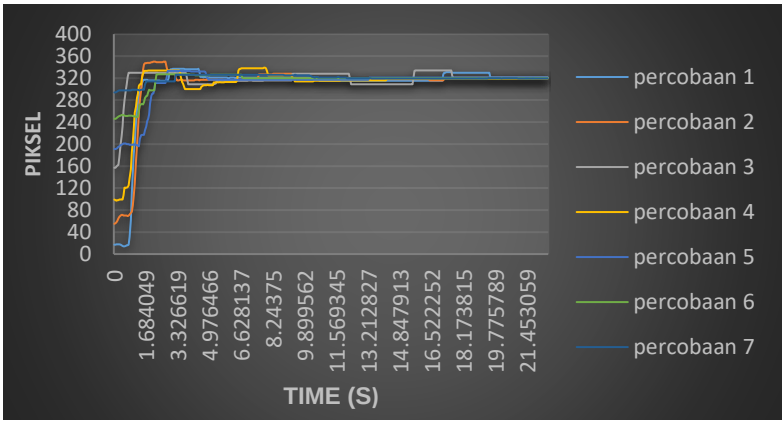

**Gambar 4.15** Grafik Pengaruh respon *rise time* sumbu x

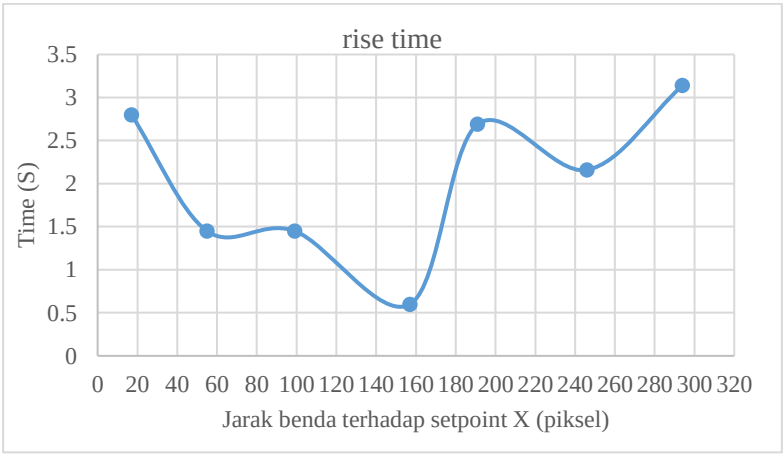
Pada *tuning* kali ini menggunakan nilai Kp 0.0005, Ki 0.0005, dan Kd 0.01 respon sistemnya sudah mulai mendekati set point yang diinginkan. Namun sistem juga belum mempertahankan posisi steady state. Sehingga perlu dilakukan perubahan lagi terhadap nilai Kp, Ki maupun Kd.

Pada *tuning* yang terakhir didapatkan hasil yang cukup bagus dengan rise time sekitar 3 detik dan tidak mengalami overshoot ketika input yang diberikan adalah sebesar 600 pixel. Nilai *tuning* kali ini yaitu hanya menggunakan Kp sebesar 0.04 dan Ki 0.0025 sehingga kontroller yang digunakan adalah kontroller PI.



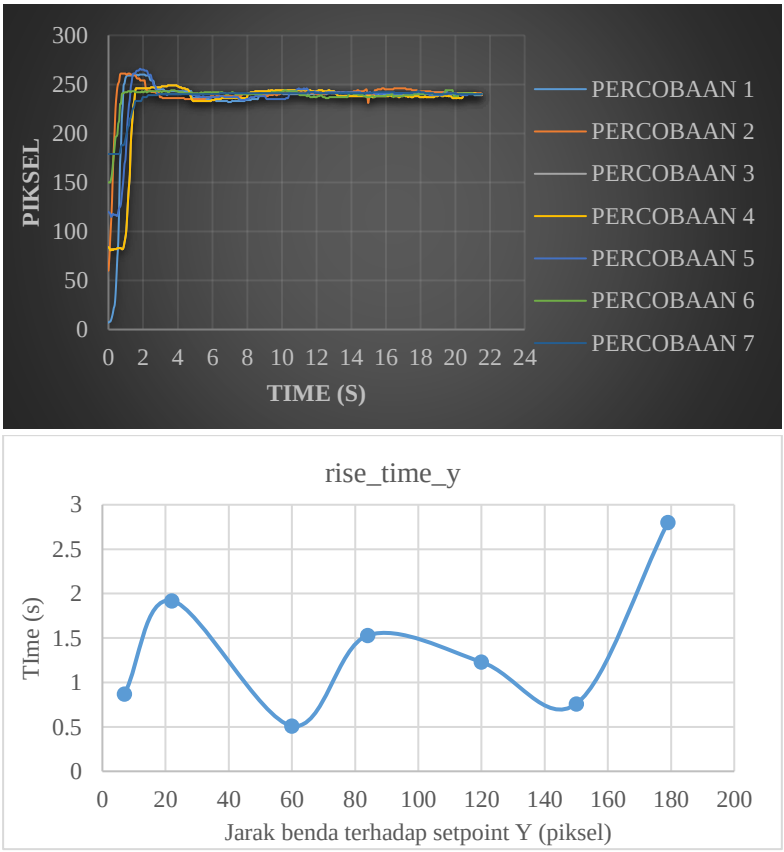

**Gambar 4.16** Grafik Pengaruh respon *rise time* sumbu x



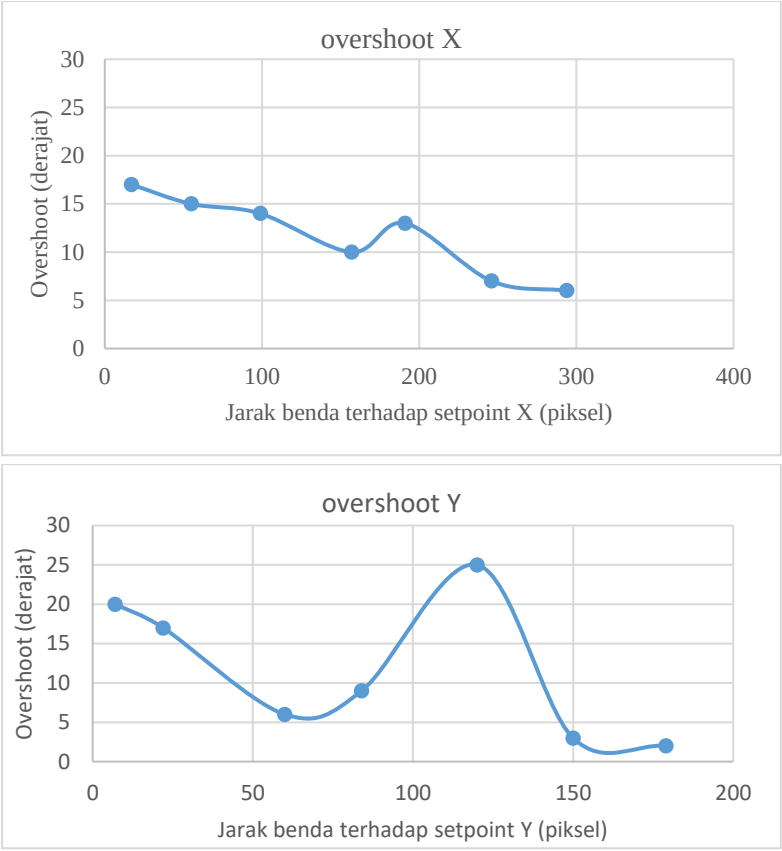

**Gambar 4.10** Grafik data pengaruh posisi benda terhadap *rise time* sumbu x. Data domain waktu (atas). Data setelah diolah menyajikan *rise time* setiap percobaan (bawah).

Seperti yang terlihat pada Gambar 4.10 bahwa waktu *rise time* sumbu x berkisar diantara 0.5 sampai 3.5 detik. Pada data tersebut terlihat bahwa jarak benda terhadap set point memiliki pengaruh yang siginifikan terhadap *rise time.* Hal tersebut dikarenakan robot menggunakan kontrol integral.



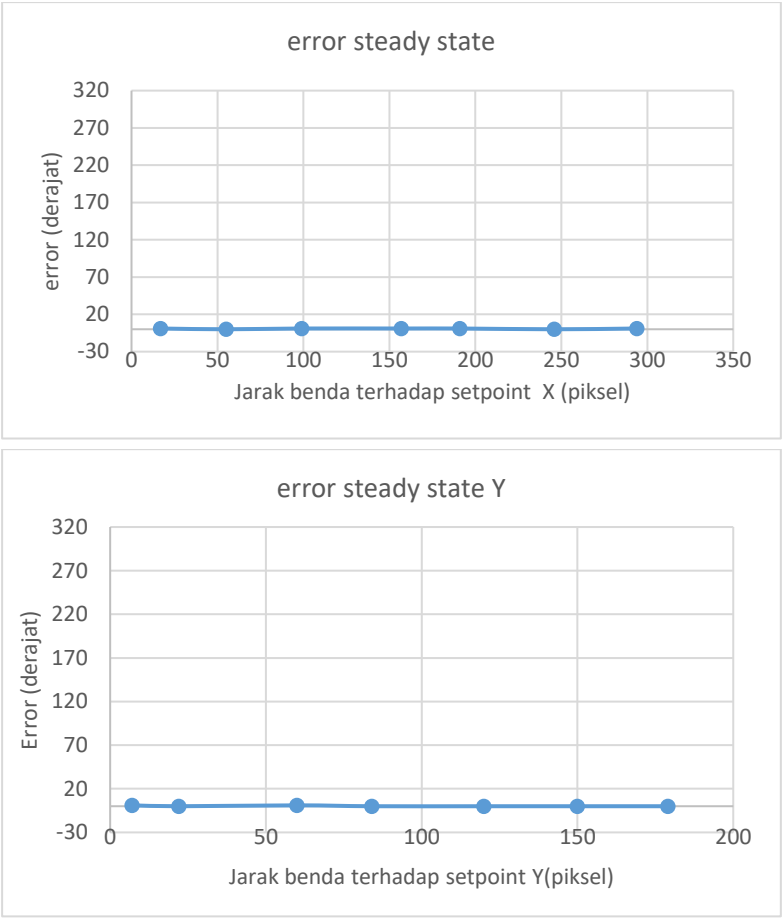

**Gambar 4.11** Grafik data pengaruh posisi benda terhadap *rise time* sumbu y. Data domain waktu (atas). Data setelah diolah menyajikan *rise time* setiap percobaan (bawah).

Seperti yang terlihat pada Gambar 4.11 bahwa waktu *rise time* sumbu y berkisar diantara 0.5 sampai 3 detik. Pada data tersebut terlihat bahwa jarak benda terhadap set point memiliki pengaruh yang siginifikan terhadap *rise time.* Hal tersebut dikarenakan robot menggunakan kontrol integral.



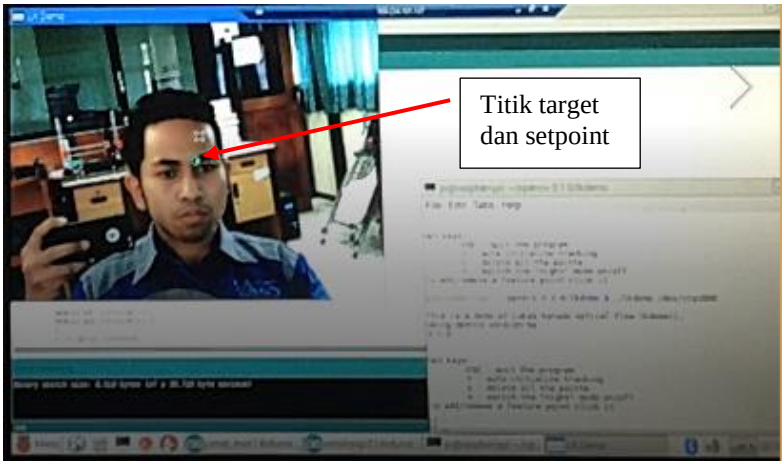

**Gambar 4.12** Grafik data pengaruh posisi benda terhadap *overshoot*. Data setelah diolah menyajikan *overshoot* setiap percobaan sumbu x (atas) dan sumbu y (bawah).

Pada grafik Gambar 4.12 nampak bahwa rentang *overshoot* pada sumbu x adalah 5 pixel sampai dengan 20 pixel, sedangkan pada sumbu y adalah 0 pixel sampai 25 pixel. Posisi benda dapat mempenagruhi *overshoot* seiring dengan semakin jauh jarak benda terhadap *set point* maka *overshoot* semakin kecil. Namun hal tersebut hanya berlaku pada sumbu x, sedangkan pada sumbu y juga mengalami perubahan namun tidak menampilkan grafik linear.




**Gambar 4.13** Grafik data pengaruh posisi benda terhadap *eror steady state.* Data setelah diolah menyajikan *error steady state* setiap percobaan sumbu x (atas) dan sumbu y (bawah).

Pada grafik Gambar 4.13 nampak bahwa rentang *error steady state* pada sumbu x dan y adalah 0 pixel sampai dengan 1 pixel. Dari percobaan didapatkan nilai *error* terhadap *set point* sangat kecil. Percobaan dilakukan dengan mempertahankan posisi titik target berada pada *set*




**Gambar 4.14** Tampilan sistem dalam melakukan *tracking*

*point* selama 30 detik. Hingga pada detik ke 30 *error steady state* yang dihasilkan berkisar antara 0 pixel dan 1 pixel saja.

## 4.6. Pengujian Ketepatan

Pengujian ini dilakukan untuk melihat kemampuan robot dalam mengarahkan senjata secara otomatis. Pengujian ini dilakukan dengan cara meletakkan robot dengan jarak tertentu dari objek dalam satuan cm. Kemudian robot diperintahkan untuk mengarahkan senjata sesuai input yang dimasukkan. Input merupakan jarak titik target dari set point dalam satuan pixel. Input dimasukkan nilai dari 50 cm hingga 650 cm. percobaan kali ini juga dilakukan dengan intensitas cahaya yang sama untuk mengurangi pengaruh pencahayaan dan latar belakang yang berbeda

Terlihat dari Gambar 4.14 antara titik target dengan set point berhimpit menjadi satu. Titik tebal berwarna hijau adalah titik target, sedangkan lingkaran berwarna biru merupakan set point.

Tabel 4.7 merupakan hasil pengujian robot mengarahkan benda secara otomatis dengan jarak tertentu. Dengan menggunakan sistem yang telah dirancang dan target yang telah ditentukan robot bekerja secara optimal hingga jarak 650 cm. Hanya pada jarak 170 cm dan 290 cm terjadi kegagalan sebanyak satu kali dari 7 percobaan. Meski demikian tingkat ketepatan masih menunjukkan nilai diatas 80%.

**Tabel 4.7** Pengujian Ketapatan sistem mengarahkan senjata

| No | Jarak Benda dari Robot (cm) | Percobaan ke – (1 = tepat, 0 = gagal) | | | | | | | Ketepatan Sistem mengarahkan senjata secara otomatis (%) |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 1 | 50 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 2 | 110 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 3 | 170 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 85.71428571 |
| 4 | 230 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 5 | 290 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 85.71428571 |
| 6 | 350 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 7 | 410 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 8 | 470 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 9 | 530 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 10 | 590 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 11 | 650 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |

## 4.7. Pengujian Ketepatan dengan Cahaya yang berbeda-beda

Metode yang sama pada pengujian sebelumnya juga dapat digunakan untuk menggali data ketepatan *tracking* dengan mempertimbangkan besar intensitas cahaya. Perbedaan intensitas cahaya ini penting untuk dilakukan percobaan karena perubahan latar belakang (*background*) juga dapat mempengaruhi proses *tracking.*

Ketepatan *tracking* dapat dilihat dengan meletakkan robot pada pencahayaan yang berbeda-beda. Kemudian dilakukan pencatatan keberhasilan percobaan yang dilakukan selama 7 kali. Pada percobaan kali ini jarak benda dari robot dibuat tetap yaitu sebesar 100cm. Terlihat pada Gambar 4.7 merupakan proses mengidentifikasi besar intensitas cahaya menggunakan lux meter.

Dari data terlihat bahwa dengan intensitas cahaya yang berbeda-beda ketepatan *tracking* masih terbilang bagus. Namun ketika intensitas cahaya rendah prosentase ketepatan yang dihasilkan juga rendah. Selain itu ketika intensitas cahaya tinggi prosentase ketepatan menurun. Selain itu prosentase ketepatannya 100% yaitu berhasil dari 7 percobaan berturut-turut.


**Gambar 4.15** Pengujian intensitas cahaya menggunakan Lux meter

**Tabel 4.8** Pengujian Ketapatan dengan cahaya berbeda

| No | Jarak Benda dari Robot (cm) | Besar Intensitas Cahaya pada lingkungan (lux) | Percobaan ke - | | | | | | | Ketepatan Sistem mengarahkan senjata secara otomatis (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 1 | 100 | 5.2 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 42.85714286 |
| 2 | 100 | 10.9 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 57.14285714 |
| 3 | 100 | 15.1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 85.71428571 |
| 4 | 100 | 20.2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 5 | 100 | 25.6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 6 | 100 | 30.5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 7 | 100 | 100.3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 8 | 100 | 500.7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 9 | 100 | 1000.2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 100 |
| 10 | 100 | 1999.8 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 85.71428571 |

## 4.8. Pengujian Ketepatan saat UGV bergerak

**Tabel 4.9** Pengujian Ketapatan ketika UGV bergerak

| No | Jarak Benda dari Robot (cm) | kecepatan UGV (dalam pwm) | Ketepatan Sistem mengarahkan senjata saat UGV bergerak |
|---|---|---|---|
| 1 | 50 | 20 | Tepat |
| 2 | 110 | 40 | Tepat |
| 3 | 170 | 60 | Tepat |
| 4 | 230 | 80 | Tepat |
| 5 | 290 | 100 | Tepat |
| 6 | 350 | 120 | Tepat |
| 7 | 410 | 140 | Tepat |
| 8 | 470 | 160 | Tepat |
| 9 | 530 | 180 | Tepat |
| 10 | 590 | 200 | Gagal |
| 11 | 650 | 220 | Gagal |

Metode yang sama pada pengujian sebelumnya juga dapat digunakan untuk menggali data ketepatan tracking dengan mempertimbangkan pergerakan dari UGV. Kecepatan dari UGV juga merupakan pengaruh dalam proses *tracking.* Semakin cepat UGV bergerak, titik target akan semakin mudah hilang, karena algoritma lucas kanade tidak dapat mendeteksi pergerakan yang terlalu cepat. Hasil dari pengujian dapat dilihat dalam Tabel 4.9. Dari data terlihar bahwa sistem gagal melakukan *tracking* ketika kecepatan UGV mencapai 200 nilai PWM. Masalah yang dihadapi adalah *turret* tidak dapat mengejar titik target yang sudah di tentukan.

## 4.9. *Real Application*

Setelah pengujian-pengujian dilakukan, akurasi dari sistem tidak cukup baik. Hal-hal yang bisa meningkatkan akurasi pengarahan senjata ke target di antaranya adalah

1. Posisi kamera terhadap laras senjata harus sedekat mungkin agar sasaran bidikan oleh senjata menjadi lebih akurat.
2. Posisi pencahayaan yang baik sangat diperlukan dalam melakukan proses *tracking.* Karena algortima Lucas-kanade sangat membutuhkan pencahayaan yang bagus.

Dengan beberapa syarat di atas, aplikasi nyata (*real application*) dari sistem ini di antaranya adalah untuk

1. Pertahanan Diatas Tank

   Pemasangan sistem ini pada kendaraan darat tank atau panser menjadi aplikasi yang paling masuk akal untuk sistem ini. Karena adanya syarat-syarat di atas untuk meningkatkan akurasi pengarahan senjata, tentu pemasangan sistem ini oleh *engineer* akan lebih baik ketika ditempatkan di tank atau panser. Dengan demikian akan membuat *operator* lebih tenang dalam menjalankan sistem persenjataan ini serta dapat mencari tempat dengan pencahayaan yang cukup dengan bantuan monitor.

2. Pertahanan dalam melakukan mata-mata

   Penggunaaan sistem ini sebagai mata-mata akan masuk akal karena sistem geraknya yang dapat dikontrol jarak jauh. Selain itu juga dapat dilakukan penembakan target dari jarak jauh (*sniper*) karena dilengkapi dengan senjata laras panjang. Dengan kondisi ini para *operator* dapat melakukan penyerangan secara diam-diam tanpa sepengetahuan musuh. Sistem ini juga dilengkapi dengan UGV yang berbentuk tank sehingga dapat bergerak di segala medan darat. Penargetan musuh menggunakan menggunakan sistem ini masih masuk akal untuk diaplikasikan.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1. Kesimpulan

Kesimpulan yang bisa diambil oleh penulis dari tugas akhir ini adalah:

1. Proses *tracking* menggunakan algoritma Lucas-kanade dapat digunakan dengan mendeteksi titik fitur namun hanya dapat digunakan dalam intensitas cahaya antara 20 lux hingga 1000 lux
2. Metode *tracking optical flow* dapat dilakukan dengan proses pengiriman nilai *error* dari Raspberry kepada mikrokontroller Arduino yang kemudian dikonversi menjadi nilai sudut servo menggunakan kontroller I dengan nilai ki 0.025 dan 0.005
3. Pemasangan kamera yang berjarak 10cm diatas laras panjang senjata dapat menimbulkan *error* sebesar 10 cm dari titik target pada kamera. Meski demikian proses *tracking* masih tetap bisa jika target yang dipilih adalah manusia dengan jarak 650 cm dari kamera.
4. Robot dapat dikendalikan menggunakan *remote control* dengan mikrokontroller Arduino sebagai pemrosesan perintah mode robot
5. Posisi target dapat diketahui koordinat sumbu x dan sumbu y dari frame. Dari koordinat tersebut dapat dicari nilai *error* yang digunakan untuk kontrol PID
6. Algoritma Lucas-Kanade juga dapat melakukan *tracking* ketika kamera yang digerakkan. Jadi baik objek ataupun kamera yang bergerak tetap dapat melakukan *tracking* titik fitur.

## 5.2. Saran

Beberapa saran yang penulis bisa berikan untuk pengembangan tugas akhir adalah:

1. Metode *tracking* Lucas-kanade harus disertakan juga program untuk mengkalibrasi secara otomatis terhadap perbedaan intensitas cahaya.
2. Kamera yang dipakai sebaiknya memiliki resolusi dan fps lebih tinggi agar jarak yang terukur bisa lebih jauh dan *tracking* juga lebih akurat.

3. Jika ingin menambahkan metode *tracking*, bisa dicari metode *tracking* yang respon terhadap pergerakan yang cepat dan dapat tidak terpengaruh halangan.
4. Kamera yang digunakan memiliki mode *night vision* atau memiliki respon rentang panjang gelombang hingga infra merah sehingga bisa dicoba untuk kondisi gelap.

# DAFTAR PUSTAKA


[1] Goge, Douglas W., A Brief History of Unmanned Ground Vehicle (UGV) Development Efforts, Unmanned System Magazine, United States of America, 1995.

[2] ------, UGV, http://en.wikipedia.org/wiki/Unmanned_ ground_vehicle.

[3] Khatib, Oussama, Autonoumous Robotic System, Department of Computer Science, Universitas Stanford, Palo Alto USA, 2008.

[4] Forest, Alex dan Mustafa Konca, Autonoumous Cars and Society, Department of Social Science and Policy Studies, Worcester Polytechnic InstituteWorcester, 2007.

[5] Bradski, G. dan Kaehler, A., "Learning OpenCV Computer Vision with the OpenCV Library", O'Reilly, Sebastopol, 2008.

[6] Maulidi, A.U., "Rancang Bangun Sistem Pengukuran Posisi Target dengan Kamera Stereo untuk Pengarah Senjata Otomatis". Tugas Akhir. Institut Teknologi Sepuluh Nopember: Surabaya, 2016.

[7] Bradski, G. dan Kaehler, A., "Learning OpenCV Computer Vision with the OpenCV Library", O'Reilly, Sebastopol, 2008.

[8] Kumar, T. dan Verma, K., "A Theory Based on Conversion of RGB Image to Gray Image", International Journal of Computer Applications, vol. 7 - no. 2, pp. 7-10, September, 2010.

[9] Hermawati, F. A., "Pengolahan Citra Digital Konsep & Teori", Penerbit Andi, Yogyakarta, Bab 2, 2013.

[10] Bradski, G. dan Kaehler, A., "Learning OpenCV Computer Vision with the OpenCV Library", O'Reilly, Sebastopol, 2008.

[11] "Controlling DC Brush Motors with H-Bridge Driver ICs", ROHM Semiconductor, 2009.

[12] "Arduino Nano" < URL:http://arduino.cc>,2017.

[13] Basilio, J.C. dan Matos, S.R., "Design of PI and PID Controllers with Trancient Performance Specification", IEEE Transaction on Education, vol. 45 – no.4, pp. 364-370, November, 2002.

[14] Syahrul, "Pemrograman Mikrokontroler AVR Bahasa ASSEMBLY dan C", INFORMATIKA, Bandung, Bab 11, 2014.

[15] "OPENCV (Open Source Computer Version)" <URL:http://opencv.org>, 2016.

[16] "Servo MG966R"<URL:http://servo.org>,2017

[17] "motor DC power window"<URL:http://motordcbrush.org>,2017

[18] "logitech extreme 3d pro" <URL:http://logitech.com>,2017

[19] Qomaruzzaman, M., “Semi Autonomous Mobile Robot dengan Lengan untuk Mengambil Objek ”. Tugas Akhir. Institut Teknologi Sepuluh Nopember: Surabaya, 2015.

# LAMPIRAN

## Program Pada Raspberry

```
#include "opencv2/video/tracking.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <wiringPi.h>
#include <wiringSerial.h>
#include <string.h>

#include <iostream>
#include <fstream>
#include <ctype.h>
#include <stdio.h>

using namespace cv;
using namespace std;

static void help()
{
    // print a welcome message, and the OpenCV version
    cout << "\nThis is a demo of Lukas-Kanade optical flow
lkdemo(),\n"
         "Using OpenCV version %s\n" << CV_VERSION << "\n"
         << endl;

    cout << "\nHot keys: \n"
         "\tESC - quit the program\n"
         "\tr - auto-initialize tracking\n"
         "\tc - delete all the points\n"
         "\tn - switch the \"night\" mode on/off\n"
         "To add/remove a feature point click it\n" << endl;
}

Point2f point = Point2f(640/2,480/2);
bool addRemovePt = false;

static void onMouse( int event, int x, int y, int flags, void* param )
```

```
{
    if( event == CV_EVENT_LBUTTONDOWN )
    {
        addRemovePt = true;
    }
}

int main( int argc, char** argv )
{
    VideoCapture cap;
    TermCriteria
termcrit(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.03);
    Size subPixWinSize(10,10), winSize(31,31);

    const int MAX_COUNT = 500;
    bool needToInit = false;
    bool nightMode = false;
   /*
    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 &&
isdigit(argv[1][0])))
        cap.open(argc == 2 ? argv[1][0] - '0' : 0);
    else if( argc == 2 )
        cap.open(argv[1]);
*/

if (argc < 2){
printf("wrong arguments\n");
return 0;
}else{
cap.open(atoi(argv[1]));
//printf("%s\n", argv[2]);
}


    if( !cap.isOpened() )
    {
        cout << "Could not initialize capturing...\n";
        return 0;
    }

    help();
```

```
    namedWindow( "LK Demo", CV_WINDOW_FULLSCREEN );
    moveWindow( "LK Demo", 0, 0);
    setMouseCallback( "LK Demo", onMouse, 0 );

    Mat gray, prevGray, image;
    vector<Point2f> points[2];
wiringPiSetupPhys();
pinMode(11,INPUT);
pinMode(7,INPUT);
int fd = serialOpen(argv[1],9600);
if (fd<0) printf("error open serial\n");

    ofstream myfile;
    myfile.open ("example.txt");


    for(;;)
    {
if(digitalRead(11)==HIGH) addRemovePt=true;
        Mat frame;
        cap >> frame;
flip(frame,frame,1);
        if( frame.empty() )
            break;

        frame.copyTo(image);
circle(image, point, 7, Scalar(255,255,64), 0, 8);
        cvtColor(image, gray, CV_BGR2GRAY);
        if( nightMode )
            image = Scalar::all(0);

        if( needToInit )
        {
            // automatic initialization
            goodFeaturesToTrack(gray, points[1], MAX_COUNT, 0.01,
10, Mat(), 3, 0, 0.04);
            cornerSubPix(gray, points[1], subPixWinSize, Size(-1,-1),
termcrit);
            addRemovePt = false;
        }
```

```
        else if( !points[0].empty() )
        {
            vector<uchar> status;
            vector<float> err;
            if(prevGray.empty())
                gray.copyTo(prevGray);
            calcOpticalFlowPyrLK(prevGray, gray, points[0], points[1],
status, err, winSize,
                                 3, termcrit, 0, 0.001);
            size_t i, k;
            for( i = k = 0; i < points[1].size(); i++ )
            {
                long int time;
time = clock ();
if( addRemovePt )
                {
                    if( norm(point - points[1][i]) <= 5 )
                    {
                        addRemovePt = false;
                        continue;
                    }
                }

                if( !status[i] )
                    continue;

                points[1][k++] = points[1][i];
                circle( image, points[1][i], 3, Scalar(0,255,0), -1, 8);
// cout << "\nposisi \n" <<points[1][i]  << "\n"
//        << endl;
CvPoint point=points[1][i];
CvPoint setpoint=cvPoint(image.cols/2,image.rows/2);
char container [10]={};
unsigned char getC;

int Ax=point.x-setpoint.x;
//cout << "Ax :" << int(Ax) << "\n";
memcpy(container,&Ax,2);
int Ay=point.y-setpoint.y;
memcpy(container + 2,&Ay,2);
cout << "Ax :" << int(Ax) << "  Ay :" << int(Ay) << "\n";
```

```
serialPutchar(fd,'i');
serialPutchar(fd,'t');
serialPutchar(fd,'s');
serialPutchar(fd,container[0]);
serialPutchar(fd,container[1]);
serialPutchar(fd,container[2]);
serialPutchar(fd,container[3]);

myfile << Ax << "," << Ay << "," << setpoint.x << "," << setpoint.y <<
"," << point.x << "," << point.y << "," << time << endl;
            }
            points[1].resize(k);
}
        if( addRemovePt && points[1].size() < (size_t)MAX_COUNT )
        {
            vector<Point2f> tmp;
            tmp.push_back(point);
            cornerSubPix( gray, tmp, winSize, cvSize(-1,-1), termcrit);
            points[1].push_back(tmp[0]);
            addRemovePt = false;
        }

        needToInit = false;
        imshow("LK Demo", image);

        char c = (char)waitKey(10);
if( c == 27 )
{
             myfile.close();
break;
}
        switch( c )
        {
        case 'r':
            needToInit = true;
            break;
        case 'c':
            points[1].clear();
            break;
        case 'n':
```

```
            nightMode = !nightMode;
            break;
        default:
            ;
        }
if(digitalRead(7)==HIGH) points[1].clear();

        std::swap(points[1], points[0]);
        swap(prevGray, gray);
    }

    return 0;
}
```

## Program Pada Arduino 1

```
#include <Servo.h>
#include <stdio.h>
char tampil[500];

Servo myservo;
Servo mineservo;
Servo tembak;

//baca remot
double pwm_value[4];
int maju_mundur = 2;
int kanan_kiri = 12;
int mode = 13;
//int servo_pan = 11;
//int servo_tilt = 10;

//---------------------OTOMATIIIIIIIISSSS----------------------
int lock_target = A3;
int hapus_titik = A4;
char container[200];
char serial_data;
char serial_status;
float pwm=90;
float pwm2=90;
```

```
//int deg=0;
float P;
float P2;
float I;
float I2;
float D;
float D2;
float err;
float err2;
float kp = 0.000;
float kp2 = 0.000;
float ki = 0.0025;
float ki2 = 0.005;
float kd = 0.0;
float kd2 = 0.0;

long int time;
long int time_out;

int Ax;
//memcpy(&Ax,container,2);
int Ay;
//memcpy(&Ay,container+2,2);

void setup()
{
  myservo.attach(11);
  mineservo.attach(10);
  tembak.attach(6);
  pinMode(maju_mundur, INPUT);
  pinMode(kanan_kiri, INPUT);
  pinMode(mode, INPUT);
  pinMode(lock_target, OUTPUT);
  pinMode(hapus_titik, OUTPUT);
  //pinMode(servo_pan, OUTPUT);
  //pinMode(servo_tilt, OUTPUT);
  //pinMode(A1, OUTPUT);
  Serial.begin(9600);
  I = 90;
  I2 = 90;
}
```

```
void baca_remot()
{
  pwm_value[0] = pulseIn(maju_mundur, HIGH);
  pwm_value[1] = pulseIn(kanan_kiri, HIGH);
}

void baca_mode()
{
pwm_value[2] = pulseIn(mode, HIGH);
}

#define CH0_ATAS        1733.0
#define CH0_TENGAH      1275.0
#define CH0_BAWAH       867.0

#define CH1_ATAS        1738.0
#define CH1_TENGAH      1278.0
#define CH1_BAWAH       867.0

#define CH2_ATAS        874.0
#define CH2_TENGAH      1300.0
#define CH2_BAWAH       1708.0


//float out_remot_2;



float out_remot_servo_0 = 90;
float out_remot_servo_1 = 90;
float out_remot_servo_2;

//void olah_pwm_servo()
//{
//  if(pwm_value[0] >= (CH0_TENGAH+50)) out_remot_servo_0 =
((pwm_value[0] - (CH0_TENGAH+50))*90/(CH0_ATAS -
(CH0_TENGAH+50)))+90;
```

```
//  else if(pwm_value[0] <= (CH0_TENGAH-50)) out_remot_servo_0
= ((pwm_value[0]-50) - CH0_BAWAH)*90/((CH0_TENGAH-50) -
CH0_BAWAH);
//  else out_remot_servo_0 = 90;
//
//  if(pwm_value[1] >= (CH1_TENGAH+50)) out_remot_servo_1 =
((pwm_value[1] - CH1_TENGAH)*90/(CH1_ATAS -
CH1_TENGAH))+90;
//  else if(pwm_value[1] <= (CH1_TENGAH-50)) out_remot_servo_1
= ((pwm_value[1]-50) - CH1_BAWAH)*90/((CH1_TENGAH-50) -
CH1_BAWAH);
//  else out_remot_servo_1 = 90;
//
//  servo(out_remot_servo_0, out_remot_servo_1);
//}

void olah_pwm_servo()
{
  if(pwm_value[0] >= (CH0_TENGAH+100)) out_remot_servo_0 =
out_remot_servo_0 + 1;
  else if(pwm_value[0] <= (CH0_TENGAH-100)) out_remot_servo_0
= out_remot_servo_0 - 1;
  else out_remot_servo_0 = out_remot_servo_0;

  if(pwm_value[1] >= (CH1_TENGAH+100)) out_remot_servo_1 =
out_remot_servo_1 + 1;
  else if(pwm_value[1] <= (CH1_TENGAH-100)) out_remot_servo_1
= out_remot_servo_1 - 1;
  else out_remot_servo_1 = out_remot_servo_1;

  //servo(out_remot_servo_0, out_remot_servo_1);
}



void servo(signed int pos_tilt, signed int pos_pan)
{
  int s_pan;
  int s_tilt;

//  if(m_kanan>=0) {MOTOR_KANAN_CW;}
```

```
//  else {MOTOR_KANAN_CCW;}
//
//  if(m_kiri>=0) {MOTOR_KIRI_CW;}
//  else {MOTOR_KIRI_CCW;}

  s_pan = abs(pos_pan);
  s_tilt = abs(pos_tilt);

  if(s_pan>180) s_pan = 180;
  else if(s_pan<0) s_pan = 0;

  if(s_tilt>180) s_tilt = 180;
  else if(s_tilt<0) s_tilt = 0;

  //if(pwm_value[2]<1500&&pwm_value[2]>1400)
  //{
    myservo.write(s_pan);
    mineservo.write(s_tilt);
  //}
}

#define kamera_satu     digitalWrite(A0, LOW);digitalWrite(A1,
LOW);digitalWrite(A2,LOW)
#define kamera_dua      digitalWrite(A0, LOW);digitalWrite(A1,
LOW);digitalWrite(A2,HIGH)

#define MODE_GERAK_SERVO 0
#define MODE_GERAK_KONTROL_TEMBAK        1
#define MODE_GERAK_KLIK_TITIK        2
#define MODE_GERAK_HAPUS_TITIK        3
#define MODE_GERAK_OTOMATIS_1 4
#define MODE_GERAK_OTOMATIS_2 5
#define MODE_DIAM 6

int mode_gerak;

void mode_servo()
{
    baca_remot();
    kamera_dua;//kamera raspi
    olah_pwm_servo();
```

```
    servo(out_remot_servo_0, out_remot_servo_1);
    digitalWrite(lock_target,LOW);
    digitalWrite(hapus_titik,LOW);
    //Serial.print("servo \n");
    //sprintf(tampil,"SP: %4d %4d ST: %4d %4d MODE: %4d
\n",(int)pwm_value[0],(int)out_remot_servo_0,(int)pwm_value[1],(int)o
ut_remot_servo_1,(int)pwm_value[2]);
    //Serial.print(tampil);
    //motor(out_remot_0, out_remot_1);
}

void mode_kontrol_tembak()
{
     //deg = 90;
     kamera_dua;
     digitalWrite(lock_target,LOW);
     digitalWrite(hapus_titik,LOW);
     tembak.write(90);
     delay(1000);
     tembak.write(0);
     //Serial.print("tembak \n");


}

void mode_klik_titik()
{
     kamera_dua;
     digitalWrite(lock_target,HIGH);
     //Serial.print("klik \n");
}

void mode_hapus_titik()
{
    kamera_dua;
    digitalWrite(hapus_titik,HIGH);
    //Serial.print("hapus \n");
}

void mode_otomatis_1()
{
```

```
    kamera_satu;//kamera analog
    digitalWrite(lock_target,LOW);
    digitalWrite(hapus_titik,LOW);
    //Serial.print("otomatis \n");
    if(Serial.available()>14)
    {
     while(Serial.available())
     {
        time = millis();

        serial_data=Serial.read();

        if(serial_data == 'i' && serial_status == 0)serial_status++;
        else if(serial_data=='t'&&serial_status==1)serial_status++;
        else if(serial_data=='s'&&serial_status==2)serial_status++;
        else
if(serial_status==3){container[0]=serial_data;serial_status++;}
        else
if(serial_status==4){container[1]=serial_data;serial_status++;}
        else
if(serial_status==5){container[2]=serial_data;serial_status++;}
        else if(serial_status==6)
        {
        container[3]=serial_data;
        serial_status=0;

        memcpy(&Ax,container,2);
        memcpy(&Ay,container+2,2);
        Serial.flush();
        break;
        }
        else serial_status=0;
     }
     }
     //kontroller servo pan
     //if(Ax > 255) Ax = 255;
     //if(Ax < -255) Ax = -255;

     P = kp* Ax;
     I += ki * Ax;
     D = kd * ( Ax - err );
```

```
    if(I>135)I=135;
    else if(I<45) I = 45;

    err = Ax;

    pwm = P + I + D;

    if(pwm > 135) pwm = 135;
    if(pwm<45) pwm = 45;


    P2 = kp2* Ay;
    I2 += ki2 * Ay;
    D2 = kd2 * ( Ay - err2 );

    if(I2>135)I2=135;
    else if(I2<45) I2 = 45;

    err2 = Ay;

    pwm2 = P2 + I2 + D2;

    if(pwm2 > 135) pwm2 = 135;
    if(pwm2<45) pwm2 = 45;


    time_out = millis() - time;

    if(time_out>2000)
    {
       pwm = 90;
       pwm2 = 90;
       I = 90;
       I2 = 90;
    }

    //sprintf(tampil,"Ax : %4d Ay : %4d PAN : %4d TILT : %4d
\n",(int)Ax, (int)Ay, (int)pwm, (int)pwm2);
    //Serial.print(tampil);
    myservo.write((int)pwm);
```

```
      mineservo.write((int)pwm2);


      delay(10);

}

void mode_otomatis_2()
{
    kamera_dua;//kamera raspi
    digitalWrite(lock_target,LOW);
    digitalWrite(hapus_titik,LOW);
    //Serial.print("otomatis \n");
    if(Serial.available()>14)
    {
     while(Serial.available())
     {
        time = millis();

        serial_data=Serial.read();

        if(serial_data == 'i' && serial_status == 0)serial_status++;
        else if(serial_data=='t'&&serial_status==1)serial_status++;
        else if(serial_data=='s'&&serial_status==2)serial_status++;
        else
if(serial_status==3){container[0]=serial_data;serial_status++;}
        else
if(serial_status==4){container[1]=serial_data;serial_status++;}
        else
if(serial_status==5){container[2]=serial_data;serial_status++;}
        else if(serial_status==6)
        {
        container[3]=serial_data;
        serial_status=0;

        memcpy(&Ax,container,2);
        memcpy(&Ay,container+2,2);
        Serial.flush();
        break;
        }
        else serial_status=0;
```

```
}
}
//kontroller servo pan

P = kp* Ax;
I += ki * Ax;
D = kd * ( Ax - err );

if(I>135)I=135;
else if(I<45) I = 45;

err = Ax;

pwm = P + I + D;

if(pwm > 135) pwm = 135;
if(pwm<45) pwm = 45;

P2 = kp2* Ay;
I2 += ki2 * Ay;
D2 = kd2 * ( Ay - err2 );

if(I2>135)I2=135;
else if(I2<45) I2 = 45;

err2 = Ay;

pwm2 = P2 + I2 + D2;

if(pwm2 > 135) pwm2 = 135;
if(pwm2<45) pwm2 = 45;


time_out = millis() - time;

if(time_out>2000)
{
   //mode_servo();
   pwm = 90;
   pwm2 = 90;
   //pwm = out_remot_servo_1;
```

```
          //pwm2 = out_remot_servo_0;
          I = 90;
          I2 = 90;
       }

      myservo.write((int)pwm);
       mineservo.write((int)pwm2);

       delay(10);

}

void mode_diam()
{
digitalWrite(lock_target,LOW);
        digitalWrite(hapus_titik,LOW);
        kamera_satu;//kamera analog
//olah_pwm_remot();
//kinematik_motor(out_remot_0, out_remot_1);
        //Serial.print("UGV \n");
}

void demo()
{


  sprintf(tampil,"Ax : %4d Ay : %4d SP: %4d %4d ST: %4d %4d
MODE: %4d \n",(int)pwm, (int)pwm2,
(int)pwm_value[0],(int)out_remot_servo_0,(int)pwm_value[1],(int)out_
remot_servo_1,(int)pwm_value[2]);
  Serial.print(tampil);

  ///////////////// tentukan mode /////////////////
  baca_mode();
  //disesuaikan lebar pulsanya. G usah pakek pwm<... && pwm > ...
sudah bisa kyk gini. klo ada satu yang terpenuhi else if dibawahnya d
skip sama dia
  if(pwm_value[2] < 1300 && pwm_value[2] > 1200) mode_gerak =
MODE_GERAK_SERVO;
  else if(pwm_value[2] < 1640 && pwm_value[2] > 1620) mode_gerak
= MODE_GERAK_KONTROL_TEMBAK;
```

```
  else if(pwm_value[2] < 1100 && pwm_value[2] > 1000) mode_gerak
= MODE_GERAK_KLIK_TITIK;
  else if(pwm_value[2] < 1400 && pwm_value[2] > 1300) mode_gerak
= MODE_GERAK_HAPUS_TITIK;
  else if(pwm_value[2] < 1500 && pwm_value[2] > 1400) mode_gerak
= MODE_GERAK_OTOMATIS_1;
  else if(pwm_value[2] <= 1620 && pwm_value[2] > 1500)
mode_gerak = MODE_GERAK_OTOMATIS_2;
  else mode_gerak = MODE_DIAM;
  ////////////////////////////////////////////////


switch(mode_gerak)
{
case MODE_GERAK_SERVO       : mode_servo();break;
case MODE_GERAK_KONTROL_TEMBAK                :
mode_kontrol_tembak();break;
case MODE_GERAK_KLIK_TITIK       : mode_klik_titik();break;
case MODE_GERAK_HAPUS_TITIK       : mode_hapus_titik();
break;
case MODE_GERAK_OTOMATIS_1       : mode_otomatis_1();break;
              case MODE_GERAK_OTOMATIS_2       :
mode_otomatis_2();break;
case MODE_DIAM         : mode_diam();break;
}

}

void loop() {
  demo();

}
```

## Program Pada Arduino 2

```
#include <stdio.h>
char tampil[200];

//baca remot
double pwm_value[4];
```

```
int maju_mundur = 2;
int kanan_kiri = 12;
int mode = 13;
//int servo_pan = 11;
//int servo_tilt = 10;
//motor 1
int enA = 6;
int in1 = 7;
int in2 = 8;
//motor 2
int in3 = 5;
int in4 = 4;
int enB = 3;
//pancingan aja
int sign = 10;

void setup()
{
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
  pinMode(sign, OUTPUT);
  pinMode(maju_mundur, INPUT);
  pinMode(kanan_kiri, INPUT);
  pinMode(mode, INPUT);
  Serial.begin(9600);
}

#define
MOTOR_KANAN_CW      digitalWrite(in1,1);digitalWrite(in2,0)
#define
MOTOR_KANAN_CCW     digitalWrite(in1,0);digitalWrite(in2,1)
#define MOTOR_KIRI_CW       digitalWrite(in3,0);digitalWrite(in4,1)
#define MOTOR_KIRI_CCW      digitalWrite(in3,1);digitalWrite(in4,0)

void motor(signed int m_kiri, signed int m_kanan)
{
  int out_motor_kanan;
  int out_motor_kiri;
```

```
  if(m_kanan>=0) {MOTOR_KANAN_CW;}
  else {MOTOR_KANAN_CCW;}

  if(m_kiri>=0) {MOTOR_KIRI_CW;}
  else {MOTOR_KIRI_CCW;}

  out_motor_kanan = abs(m_kanan);
  out_motor_kiri = abs(m_kiri);

  if(out_motor_kanan>255)out_motor_kanan = 255;
  if(out_motor_kiri>255)out_motor_kiri = 255;

  analogWrite(enA, out_motor_kanan);
  analogWrite(enB, out_motor_kiri);
}

void baca_remot()
{
  pwm_value[0] = pulseIn(maju_mundur, HIGH);
  pwm_value[1] = pulseIn(kanan_kiri, HIGH);
}

void baca_mode()
{
pwm_value[2] = pulseIn(mode, HIGH);
}

#define CH0_ATAS       1733.0
#define CH0_TENGAH     1275.0
#define CH0_BAWAH      867.0

#define CH1_ATAS       1738.0
#define CH1_TENGAH     1278.0
#define CH1_BAWAH      867.0

#define CH2_ATAS       874.0
#define CH2_TENGAH     1300.0
#define CH2_BAWAH      1708.0

  float out_remot_0;
  float out_remot_1;
```

```
  float out_remot_2;

int olah_pwm_remot()
{

  if(pwm_value[0]<= (CH0_TENGAH-100)) out_remot_0 =
(pwm_value[0] - CH0_TENGAH)*255/(CH0_ATAS - CH0_TENGAH);
  else if(pwm_value[0]>= (CH0_TENGAH+100)) out_remot_0 = -
((pwm_value[0] - CH0_TENGAH)*255/(CH0_BAWAH -
CH0_TENGAH));
  else out_remot_0 = 0;

  if(pwm_value[1]>= (CH1_TENGAH+100)) out_remot_1 =
(pwm_value[1] - CH1_TENGAH)*255/(CH1_ATAS - CH1_TENGAH);
  else if(pwm_value[1]<= (CH1_TENGAH-100)) out_remot_1 = -
((pwm_value[1] - CH1_TENGAH)*255/(CH1_BAWAH -
CH1_TENGAH));
  else out_remot_1 = 0;

//  if(pwm_value[2] >= CH2_TENGAH) = ((pwm_value[2] -
CH2_TENGAH)*128/(CH2_ATAS - CH2_TENGAH))+128;
//  else out_remot_2 = (pwm_value[2] -
CH2_BAWAH)*128/(CH2_TENGAH - CH2_BAWAH);

  //if(pwm_value[2]>= CH2_TENGAH) out_remot_2 = (pwm_value[2]
- CH2_TENGAH)*1000/(CH2_ATAS - CH2_TENGAH)+1000;
  //else out_remot_2 = - ((pwm_value[2] -
CH2_BAWAH)*1000/(CH2_TENGAH - CH2_BAWAH))+1000;

  return 0;
}

void kinematik_motor(int maju, int belok)
{
  int _out_motor_kanan;
  int _out_motor_kiri;

  _out_motor_kanan = maju + belok;
  _out_motor_kiri = maju - belok;

  if(_out_motor_kanan>255) _out_motor_kanan =255;
```

```
  if(_out_motor_kiri>255) _out_motor_kiri =255;

  motor(_out_motor_kanan, _out_motor_kiri);
}

#define MODE_DIAM                    0
#define MODE_GERAK_OTOMATIS 1
#define MODE_GERAK_UGV               2


int mode_gerak;

void mode_otomatis()
{
      olah_pwm_remot();
kinematik_motor(out_remot_0, out_remot_1);
}

void mode_ugv()
{
olah_pwm_remot();
kinematik_motor(out_remot_0, out_remot_1);
      //Serial.print("UGV \n");
}

void mode_diam()
{
      digitalWrite(sign, HIGH);
}

void demo()
{


  sprintf(tampil,"MKA: %1d %1d %4d MKI: %1d %1d %4d MODE:
%4d %4d %4d
\n",digitalRead(in1),digitalRead(in2),(int)out_remot_0,digitalRead(in3)
,digitalRead(in4),(int)out_remot_1,(int)pwm_value[0],(int)pwm_value[
1],(int)pwm_value[2]);
  Serial.print(tampil);
```

```
  ///////////////// tentukan mode /////////////////
  baca_mode();
  //disesuaikan lebar pulsanya. G usah pakek pwm<... && pwm > ...
sudah bisa kyk gini. klo ada satu yang terpenuhi else if dibawahnya d
skip sama dia
  if(pwm_value[2] < 1300 && pwm_value[2] > 1200) mode_gerak =
MODE_DIAM;
  else if(pwm_value[2] < 1640 && pwm_value[2] > 1600) mode_gerak
= MODE_DIAM;
  else if(pwm_value[2] < 1100 && pwm_value[2] > 1000) mode_gerak
= MODE_DIAM;
  else if(pwm_value[2] < 1400 && pwm_value[2] > 1300) mode_gerak
= MODE_DIAM;
  else if(pwm_value[2] < 1500 && pwm_value[2] > 1400) mode_gerak
= MODE_GERAK_OTOMATIS;
  else mode_gerak = MODE_GERAK_UGV;
  ////////////////////////////////////////////////

  if(mode_gerak != MODE_DIAM)
  {
  baca_remot();
  olah_pwm_remot();
  }
switch(mode_gerak)
{
case MODE_DIAM       : mode_diam();break;
case MODE_GERAK_OTOMATIS       : mode_otomatis();break;
case MODE_GERAK_UGV       : mode_ugv();break;
}

}

void loop()
{
  demo();
}
```

# BIODATA PENULIS

Fandi Afrizal lahir di Surabaya pada 09 Februari 1995, yang merupakan anak tunggal dari pasangan Djoko dan Ping. Penulis menyelesaikan pendidikan dasar di SDN Dr Soetomo VII Surabaya dan dilanjutkan dengan pendidikan menengah di SMP Al-Hikmah Surabaya dan SMAN 5 Surabaya. Pada tahun 2013, penulis memulai pendidikan di jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah, penulis aktif membantu penyelenggaraan kegiatan dan aktif sebagai asisten laboratorium Elektronika Dasar.

Email:
fandiafrizal177@gmail.com

*Halaman ini sengaja dikosongkan*