TUGAS AKHIR – TM 141585

RANCANG BANGUN *SELF-DRIVING REMOTE CONTROL CAR* DENGAN METODE DISKRIT MODE KONTROLER LIMA POSISI BERBASIS *GLOBAL POSITIONING SYSTEM (GPS)*

Moh. Akrom Wafaiq
NRP. 2112 100 119

Dosen Pembimbing
Arif Wahjudi, St., Mt., Phd.

Jurusan Teknik Mesin
Fakultas Teknologi Industri
Institute Teknologi Sepuluh Nopember Surabaya
Surabaya 2016

FINAL PROJECT – TM 141585

DESIGN SELF-DRIVING CAR WITH REMOTE CONTROL DISCRETE CONTROLLER MODE FIVE POSITION BASED ON GLOBAL POSITIONING SYSTEM (GPS)

Moh. Akrom Wafaiq
NRP. 2112 100 119

Academic Advisor
Arif Wahjudi, St., Mt., Phd.

Department Mechanical Engineering
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

LEMBAR PENGESAHAN

RANCANG BANGUN *SELF-DRIVING REMOTE CONTROL CAR* DENGAN METODE DISKRIT MODE KONTROLER LIMA POSISI BERBASIS *GLOBAL POSITIONING SYSTEM (GPS)*

TUGAS AKHIR
Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Program Studi S-1 Jurusan Teknik Mesin
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Oleh:
MOHAMMAD AKROM WAFAIQ
Nrp. 2112 100 119

Disetujui oleh Tim Penguji Tugas Akhir :
1. Arif Wahjudi, S.T, M.T, Ph.D.....(Pembimbing)
   NIP. 197303222001121001
2. Prof. Dr. Ing. Ir. I Made Londen Batan, M.Eng
   .....(Penguji I)
   NIP. 195811061986011001
3. Ir. Bambang Pramujati, M.Sc. Eng, Ph.D.....(Penguji II)
   NIP. 196912031994031001
4. Ibu Dinny Harnaniy, S.T, M.Sc.....(Penguji III)
   NIP. 2100201405001

SURABAYA
Juli 2016

RANCANG BANGUN *SELF-DRIVING REMOTE CONTROL CAR* DENGAN METODE DISKRIT MODE KONTROLER LIMA POSISI BERBASIS *GLOBAL POSITIONING SYSTEM (GPS)*

Nama : **Moh. Akrom Wafaiq**
NRP : **2112100119**
Jurusan : **Teknik Mesin FTI-ITS**
Dosen Pembimbing : **Arif Wahjudi ST,MT,Ph.D**

Abstrak

Semakin cepat perkembangan teknologi otomotif dan komunikasi dewasa ini membuat banyak perubahan diseluruh aspek kehidupan umat manusia. Dibidang otomotif sudah dikembangkan *autonomous car* yang merupakan mobil tanpa pengemudi, dibuat dengan teknologi yang sangat canggih seperti lidar, video camera, position estimator, dan radar, sehingga tak heran biaya produksinya sangat mahal. Dibidang komunikasi *smartphone* telah menjadi kebutuhan pokok dikarenakan memiliki fungsi yang bermacam-macam sepertihalnya *chatting*, sms, hingga sistem navigasi dalam berkendara. Berdasarkan fakta-fakta ini maka muncul gagasan mengenai *self-driving car* dengan memanfaatkan Google Maps pada *smartphone* sebagai navigasi dengan menggunakan proses *image processing*.

Pada tugas akhir ini dilakukan pembuatan *self-driving remote control car* dengan *input* berupa pergeseran sudut antara pointer pada *Google Maps* dan jalan yang dilalui. Dalam pembuatan *self-driving remote control car* dilakukan studi literatur dan dilakukan pemilihan komponen yang kemungkinan mendukung kinerja prototipe yang akan dibuat, kemudian dilakukan perancangan sistem kendali dengan mengkomunikasikan seluruh perangkat terhadap arduino sebagai kontroler, kemudian dilakukan perakitan prototipe mobil. Pada tahap ujicoba, *input* berupa sudut yang berasal dari proses image processing antara sudut pointer *google maps* dan sudut jalan.

Sistem kendali yang dibangun mampu mengendalikan mobil remot kontrol dengan kondisi lurus, kiri, dan kanan sesuai nilai sudut akhir yang diperoleh dari hasil *image processing*.

Kata kunci: *image processing, smartphone, image processing.*

DESIGN SELF-DRIVING CAR WITH REMOTE CONTROL DISCRETE CONTROLLER MODE FIVE POSITION BASED ON GLOBAL POSITIONING SYSTEM (GPS)

Name	***: Moh. Akrom Wafaiq***
NRP	***: 2112100119***
Departement	***: Teknik Mesin FTI-ITS***
Academic Advisor	***: Arif Wahjudi ST,MT,Ph.D***

Abstract

Rapid development of automotive technology and communication today makes a lot of changes to all aspects of human life. Automotive has been developed which is an autonomous car without a driver's car, made with advanced technologies such as lidar, video camera, position estimator, and radar, so no wonder the production costs are very expensive. In communications smartphones have become a primiery needs because heve a lot of functions like chat, sms, to the navigation system in driving. Based on these facts, it came the idea of a self-driving car by using Google Maps on a smartphone as a navigation using image processing.

In this final assignment of making a self-driving remote control car with input of the shift angle between the pointer on Google Maps and roads. Design and manufacture of self-driving remote control car carried out literature studies and selection of components that may support the performance of the prototype, and then to design a control system to communicate all devices to Arduino as controller, then assembly of prototypes. In the trial phase, input is angle from image processing between the angle pointer google maps and street.

Control systems are built to control remote control car with a straight condition, turn left, and turn right according to the final angle values, obtained from the image processing.

Keywords*: image processing, smartphone, Google Maps.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Teknologi kendaraan *autonomous car* ataupun mobil otomatis sebenarnya sudah sejak lama dikembangkan. Prototipe pertama bahkan telah ada sekitar tahun 1980. Prototipe ini mampu menempuh jarak hingga 100 km pada jalan kosong dengan alat bantu berupa kamera, dari sini kemudian muncul beberapa proyek pada kurun waktu 80-an hingga sekarang dengan berbagai pengembangannya. Sekarang ini banyak pengembang seperti Audi, Hyundai, Honda, Tesla, dan lain-lain telah berlomba-lomba mengembangkan *autonomous car* dengam berbagai inovasi yang berbeda. Pada umumnya autonomous car dibuat dengan teknologi yang sangat canggih seperti lidar, video camera, position estimator, dan radar, sehingga tak heran biaya produksinya sangat mahal. Sebagai gambaran, Honda Accord ADAS yang ditunjukkan gambar 1.1 dijual dengan harga USD 46.500,00 atau setara Rp 600.000.000,00 dengan nilai tukar rupiah terhadap dolar pada 14 maret 2016. Sehingga dengan harga yang sangat mahal, *autonomous car* hanya mampu dinikmati oleh kalangan atas saja.

Di jaman sekarang ini *smartphone* telah menjadi kebutuhan pokok bagi semua orang dari berbagai golongan. Tidak terkecuali Indonesia penggunaan *smartphone* beberapa tahun ini telah mengalami kenaikan yang tajam dikarenakan *smartphone* memiliki fungsi yang bermacam-macam seperti halnya untuk berfoto, game, video, sms, chatting, mempresentasikan pekerjaan, hingga sebagai sistem navigasi. Sistem navigasi yang ada pada *smartphone* biasanya

menggunakan *global positioning system (GPS)* yang mampu menampilkan posisi kendaraan dan jalan beserta kondisinya.

Gambar 1.1. Honda accord ADAS dengan sistem auto pilot penuh
Sumber : www.engadget.com

Pengolahan citra (*image processing*) adalah setiap bentuk pengolahan sinyal dimana input adalah gambar seperti foto, video bingkai, ataupun *real time*, sedangkan output dari pengolahan gambar dapat berupa gambar atau sejumlah karakteristik atau parameter yang berkaitan dengan gambar. Pengolahan citra ini luas sekali penggunaanya seperti pengukuran, pembacaan *bar code*, pengolahan citra sinar X untuk mammografi hingga mampu mengenali sasaran peluru kendali melalui sensor visual.

Berdasarkan fakta-fakta yang telah diuraikan diatas, dalam tugas akhir ini kami mencoba melakukan penelitian dengan memanfaatkan informasi *computer vision* dari *GPS smart phone* untuk mengendalikan pergerakan kendaraan pada jalur yang diinginkan.

1.2 Rumusan Masalah

Permasalahan yang dihadapi dalam penyelesaian tugas akhir ini adalah, bagaimana merancang dan membangun sistem pengendali kendaraan berdasarkan informasi sudut antara kendaraan dan jalan pada GPS.

1.3 Tujuan

Tugas akhir ini bertujuan untuk merancang dan membangun sistem pengendali kendaraan berdasarkan informasi sudut antara kendaraan dan jalan pada GPS.

1.4 Batasan Masalah

Agar dalam penyelesaian tugas akhir ini dapat lebih terarah, maka pembahasan penulisan ini dibatasi pada ruang lingkup pembahasan sebagai berikut:

1. Menggunakan mikrokontroler Arduino Uno sebagai pengolah data hasil *image processing.*
2. Mobil yang digunakan adalah mobil remot kontrol.
3. Pengendlian menggunakan prinsip pengendali diskrit-open loop.
4. Dalam tugas akhir ini, fokus pada pembuatan alat dan kemampuan mobil mengikuti *input* berupa sudut antara pointer segitiga dan jalan pada *GPS*.

1.5 Manfaat Penelitian

Tugas akhir ini diharapkan dapat memberikan manfaat sebagai berikut:

1 Memberikan kontribusi nyata pada pengembangan teknologi perancangan sistem pengendalian, demi terciptanya kemajuan teknologi nasional khususnya di bidang otomotif.

2 Prototipe mobil yang telah berfungsi dapat digunakan untuk penelitian lebih lanjut.

BAB II
DASAR TEORI

2.1 Mobil Otomatis (*Autonomous Car*)

Kemajuan teknologi telah merubah berbagai tatanan dalam banyak aspek kehidupan. Salah satu dampaknya adalah pada industri otomotif terutama pada peningkatan kenyamanan dan keamanan untuk kosumen. Perkembangan teknologi otomotif ini mampu membantu memberikan keputusan pada pengemudi dalam berbagai keadaan, seperti halnya sistem alaram benturan, *adaptive cruis control* (ACC), sistem penjaga jalur, hingga teknologi yang mampu membuat mobil parkir dengan sendirinya atau biasa disebut *advance driver assistance system* [2].

NHTSA telah mengklasifikasikan kedalam lima tingkatan untuk membantu menejelaskan automatisasi kendaran di jalan raya yaitu [2]:

a) Level 0 (Tanpa kendali otomatis)
   Pengemudi mengendalikan seluruh sistem kemudi utama secara manual seperti pengereman, kendali kemudi, dan kendali kecepatan sepanjang waktu, dan juga bertanggung jawab penuh mengamati keadaan jalan untuk keselamatan perjalanan.
b) Level 1 (automatisasi pada fungsi yang khusus)
   Automatisasi pada tahap ini menyangkut satu atau beberapa fungsi kontrol yang khusus. Jika beberapa fungsi bekerja secara otomatis fungi tersebut akan bekerja secara bebas antara satu dan yang lainnya. Pada tahap ini pengemudi tetap memegang kendali penuh untuk proses

mengemudi yang aman, tapi dapat juga mengaktifkan otomatisasi pada beberapa fungsi kendali tertentu seperti sistem pengereman dinamis pada saat keadaan darurat.

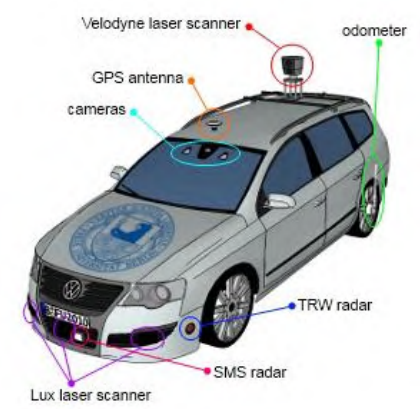
c) Level 2 (Penggabungan fungsi otomatis)
Pada tahapan ini automatisasi telah mampu mengendalikan setidaknya dua buah sistem pengendali yang didesain untuk bekerja bersama-sama, untuk mebantu pengemudi megendalikan sistem kemudi tersebut.

d) Level 3 (Auomatisasi kemudi terbatas)
Otomatisasi kendaraan pada tahap ini adalah pengemudi secara sempurna mengontrol secara otomatis dalam keadaan lalu lintas tertentu atau keadaan lingkungan tertentu. Keadaan ini benar-benar sukar bagi kendaraan untuk mengamati perubahan yang menyebabkan peralihan ke mode pengemudi manual. Pengemudi pada tahap level 3 masih sesekali dapat mengendalikan kendaraan secara manual dengan waktu peralihan ke mode automatis cukup cepat sehinga tidak menggangu kenyamanan.

e) Level 4 (Automatisasi kemudi penuh)
Kendaraan pada tahap ini didesain untuk menunjukkan seluruh fungsi pengendali otomatisasi dan juga pengamatan kondisi jalan raya selama perjalanan, dan juga desain kendaraan ini dapat melakukan kerja secara otomatis ketika pengemudi telah menetapkan tempat tujuan.

Seperti yang telah dijelaskan diatas, *autonomous car* merupakan pengembangan mobil tahap tebaru pada klasifikasi NHTSA, dalam hal ini *autonomous car* adalah suatu mekanisme otomatisasi pada kendaraan dengan tingkat keamanan yang tinggi sebagai sarana transportasi tanpa campur tangan pengemudi. Kendaraan ini mampu mengenali

dan mengidentifikasi keadaan lingkungan sekitarnya dengan bantuan berbagai peralatan canggih yang terpasang, lalu memproses data-data tersebut untuk menghasilkan respon ataupun tanggapan berupa manuver yang perlu dilakukan kendaraan pada saat di jalan raya dengan banyaknya tingkat kejadian [2].



Gambar 2.1 Sensor dan alat bantu pada *autonomous car* (thedailyomnivore.net)

Autonomous car memiliki perangkat canggih untuk mengenali dan memonitor keadaan lingkungan disekitarnya dengan menggunakan beberapa perangkat canggih seperti radar, *Global Position System (GPS)* atau yang lainnya seperti yang ditunjukkan gambar 2.1. Sistem *autonomous car* yang canggih mampu mengenali serta menentukan jalur yang cocok untuk dilalui oleh kendaraan dalam mencapai tujuan transportasi, serta mampu mengenali hambatan yang akan dilalui oleh kendaraan, seperti jalur menanjak, jalur berkelok-

kelok, dan sebagainya. Beberapa kendaraan yang memiliki *autonomous car* bahkan dapat memperbaharui data peta yang telah tersimpan di dalam basis data mereka berdasarkan masukkan dari sensor-sensor yang ada, seperti penambahan jalan kecil yang dilalui oleh kendaraan tersebut dimana jalan kecil biasanya tidak terlihat di dalam peta elektronik.

Keuntungan *autonomous car* adalah mengurangi *human error*, di Amerika sendiri, lebih dari 30.000 orang terbunuh tiap tahun dalam kecelakaan, sekitar 2,5 juta luka-luka, dan sebagian besar kecelakaan ini disebabkan *human error* (Choi et al., 2008). Maka dari itu *autonomous driving* sendiri sudah mulai diterapkan pada kendaraan-kendaraan seperti pesawat dan mobil. Sistem *autonomous* pada pesawat dinamakan *autopilot*. Saat sistem *autopilot* ini diaktifkan, pesawat akan dengan otomatis menjaga ketinggian serta kecepatan pesawat agar tetap stabil tanpa harus dikendalikan langsung oleh pilot. *Autopilot* ini juga dapat memberikan saran jalur yang dapat dilalui pesawat berdasarkan peraturan penerbangan internasional. Memasuki tahun 2014, mulai banyak perusahaan yang menerapkan *autonomous driving* pada mobil, yang dinamakan *autonomous car*. *Autonomous car* mampu mengoperasikan dirinya sendiri tanpa campur tangan manusia, tetapi karena masih dalam tahap pengembangan, *autonomous car* ini baru mampu dioperasikan pada jalan-jalan yang kosong atau masih sepi dari kendaraan. *Autonomous car* juga memiliki fitur keamanan yang akan menjaga pengendaranya dari hal-hal yang tidak diinginkan, jadi walaupun *autonomous car* ini sedang dikendalikan langsung oleh pengendara, sistem *autonomous car* tetap berjalan, seperti sensor yang mendeteksi keberadaan kendaraan yang ada disekitar *autonomous car*, sensor yang mendeteksi keberadaan garis jalan agar mobil

tidak berada dalam jalur yang salah, sensor yang mendeteksi keadaan pengendara (pengendara sedang mengantuk atau tidak) dan lain-lain [2].

2.2 Definisi Sistem

Sistem merupakan perpaduan dari beberapa unsur yang bekerja bersama-sama dan melakukan tujuan tertentu dan tidak terbatas pada sistem fisik saja. Konsep sistem dapat digunakan pada gejala-gejala yang abstrak dan dinamis seperti yang dijumpai dalam ekonomi. Sehingga, dapat dikatakan bahwa sistem harus dapat di definisikan untuk dapat menyatakan sistem fisik, biologi, ekonomi, dan sebagainya.

Menurut Jerry Bank beberapa sisitem dapat dibedakan dari beberapa sudut pandang yang berbeda diantaranya adalah sebagai berikut [5]:

1. Dari sudut pandang tingkat kepastian sistem

   Dari sudut pandang tingkat kepastian sistem contohnya adalah sistem deterministik dan sistem stokastik. Model simulasi dikatakan deterministik jika dalam model tersebut mengandung komponen probabilitas yang pasti. Kebalikannya model stokastik adalah model yang kemungkinan perubahannya acak.

2. Dari sudut pandang perubahan sistem yang berlangsung

   Dari sudut pandang perubahan sistem yang berlangsung contohnya adalah sistem sistem dinamis dan sistem statis. Model statis merupakan representasi dari sebuah sistem pada waktu tertentu, sedangkan model

dinamis menggambarkan suatu sistem yang lambat laun terjadi tanpa batas waktu.

3. Dari sudut pandang sifat perubahan sistem

   Dari sudut pandang sifat perubahan sistem contohnya adalah sistem kontinu dan sistem diskrit. Dalam simulasi sistem kontinyu, perubahan keadaan suatu sistem akan berlangsung terus menerus seiring dengan perubahan waktu. Sebagai contoh adalah perubahan debit air dalam sebuah tangki reservoir yang dilubangi bagian bawahnya. Akan tetapi untuk simulasi sistem diskrit, perubahan keadaan sistem hanya akan berlangsung pada sebagian titik perubahan waktu, seperti perubahan sistem yang terjadi pada suatu sistem manufaktur.

Dewasa ini perkembangan sistem pengendali telah berperan besar dalam lahirnya penemuan teknologi baru tentang konsep dan prinsip kerja dari sistem terutama pada sektor industri. Tak jarang sesekali indutri-industri pun pada akhirnya bekerja sama dengan pihak perguruan tinggi untuk dapat melaksanakan proyek penelitiannya ataupun sekedar membuat proyek simulator, dari sistem yang sebenarnya untuk mempermudah proses pengecekan dilapangan agar struktur sistem yang sedang beroperasi terlihat rapi dan teratur, sehingga apabila terlihat adanya suatu kerusakan ataupun kesalahan dapat langsung diketahui dan diperbaiki. Aplikasi sistem kontrol dapat dibuat simulasi, seperti simulasi kedudukan satelit, simulasi sistem navigasi pesawat terbang dan kapal laut, simulasi pengendalian tinggi permukaan bendungan, dan yang terbaru adalah simulasi *autonomus car*.

2.3 Sistem Kontrol

Sistem kontrol biasanya terdiri dari beberapa subsistem dan proses yang berkumpul dengan tujuan mendapatkan output dan juga peforma yang diinginkan ketika diberikan *input* yang khusus. Gambar 2.2 menunjukkan sistem kontrol dalam bentuk yang sederhana, dimana *input* menggambarkan sebuah *output* yang diinginkan. Sistem kontrol juga berarti proses pengaturan ataupun pengendalian terhadap satu atau beberapa besaran *(variabel, parameter)* sehingga menghasilkan suatu nilai atau dalam suatu jangkauan nilai *(range)* tertentu [1].

Gambar 2.2 Deskripsi singkat sistem kontrol
(Nise,Norman;2011)

Mesin industri bukan satu-satunya sistem kontrol otomatis, sistem ini juga terdapat di alam. Di dalam tubuh manusia ada banyak sekali sistem kontrol, seperti otak mengatur seluruh fungsi organ dalam tubuh manusia. Dalam keadaan tegang, adrenalin kita meningkat bersamaan dengan detak jantung, menyebabkan oksigen lebih banyak dialirkan ke dalam sel tubuh dari pada keadaan normal. Mata mengikuti sebuah benda yang bergerak untuk menjaganya dalam jarak pandang, dan tangan memegang dan meletakkan benda pada tempat yang diinginkan [1].

Di dalam dunia industri, dituntut suatu proses kerja yang aman dan berefisiensi tinggi untuk menghasilkan produk dengan kualitas dan kuantitas yang baik serta dengan waktu

yang telah ditentukan. Otomatisasi sangat membantu dalam banyak hal antara lain kelancaran operasional, keamanan, investasi, lingkungan, ekonomi (biaya produksi), mutu produk, dan lain-lain.

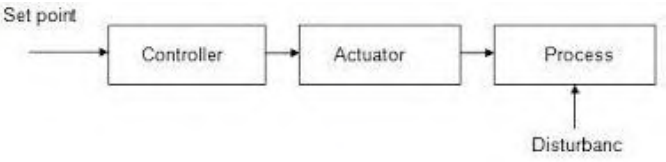
Ada banyak proses yang harus dilakukan untuk menghasilkan suatu produk sesuai standar, sehingga terdapat parameter yang harus dikontrol atau di kendalikan antara lain tekanan (*pressure*), aliran (*flow*), suhu (*temperature*), ketinggian (*level*), kerapatan (*intensity*). Gabungan kerja dari berbagai alat-alat kontrol dalam proses produksi dinamakan sistem pengontrolan proses (*process control system*). Sedangkan semua peralatan yang membentuk sistem pengontrolan disebut pengontrolan instrumentasi proses (*process control instrumentation*).

2.4 *Open-loop System*

Sistem kontrol loop terbuka (*open-loop system*) merupakan sisitem kontrol yang *output* tidak berubah meskipun ada suatu gangguan dalam prosesnya, dan *output* tidak dapat digunakan sebagai umpan balik untuk di lakukan proses koreksi kembali. Secara umum loop terbuka dimulai terlebih dahulu dengan sebuah subsistem atau disebut *input transcducer* yang merubah bentuk *input* yang digunakan oleh kontroler. Kontroler melakukan sebuah proses ataupun perencanaan tanpa adanya umpan balik yang ditunjukkan gambar 2.3. Terkadang kontrol loop terbuka tidak dapat melakukan tugas sesuai yang diharapkan dikarenakan tidak ada umpan balik atau sebuah proses koreksi [1].

Contoh dari sistem loop terbuka adalah operasi mesin cuci. Penggilingan pakaian, pemberian sabun, dan pengeringan yang bekerja sebagai operasi mesin cuci tidak akan berubah

(hanya sesuai dengan yang diinginkan seperti semula) walaupun tingkat kebersihan pakaian (sebagai keluaran sistem) kurang baik akibat adanya faktor-faktor yang kemungkinan tidak diprediksikan sebelumnya. Diagram kotak yang ditunjukkan gambar 2.3 memberikan alur proses loop terbuka.



Gambar 2.3 Blok diagram sistem kontrol loop terbuka (tneutro.net)

Contoh lainnya adalah sistem mekanik yang tersusun atas sebuah masa, pegas, dan peredam dengan gaya konstan yang diberikan pada masa. Semakin besar gaya semakin besar pula perpindahannya. Posisi sistem akan berubah dengan adanya gangguan, seperti ketika ditambah lagi gaya tambahan yang bekerja pada masa, dan sistem tidak mampu mendeteksi ataupun mengkoreksi gangguan. Sistem kontrol loop terbuka ini memang lebih sederhana, murah, dan mudah dalam desainnya, akan tetapi akan menjadi tidak stabil dan seringkali memiliki tingkat kesalahan yang besar bila diberikan gangguan dari luar [1].

2.5 Sistem Kontrol Diskrit (Diskontinyu)

Variabel diskrit adalah sesuatu yang memiliki hanya satu nilai pada rentang tetentu atau diasumsikan memiliki nilai yang telah ditentukan. Jenis variabel diskrit yang paling umum pada sistem pemrograman adalah biner yang berarti memiliki dua kemungkinan yaitu 0 dan 1 [4].

Dalam kontrol diskrit, parameter diskrit dan variabel diskrit suatu sistem dirubah pada saat keadaan (*state*) dan kejadian (*event*), dengan memakai program *logic thinking*. Perubahan tersebut dilakukan baik karena kondisi sistem telah selesai diubah atau karena waktu tertentu telah dicapai. Misalnya sistem pada lampu lalu lintas dan sistem pada *vending machine*. Maka dari itu semakin mutakhir sebuah sistem kontrol otomatis, umumnya tersusun atas *continuous state system* dan DES (*discrete event system*). Sistem seperti ini terdapat pada penerbangan dan *autonomous car* [3].

Berdasarkan kedua hal tersebut Jery Bank membaginya menjadi dua jenis perubahan yaitu[5]:

1. Perubahan gerak-kejadian (*event drive changes*)

   Perubahan gerak-kejadian dilakukan oleh kontroler sebagi respon terhadap beberapa kejadian yang telah menyebabkan keadaan sistem berubah. Perubahan dapat terjadi pada saat awal operasi atau akhir operasi. Contoh: level pengurangan bahan plastik pada corong isi mesin cetak injeksi, menarik saklar batas bawah hingga terbuka, dan memulai pengisian bahan plastik baru kedalam corong isi. Bila level pengisian plastik baru kedalam corong isi mencapai level tertentu maka saklar batas atas akan ditarik

sehingga katup akan menutp yang mengakibatkan aliran bahan plastik kedalam corong berhenti.

2. Peubahan gerak-waktu (*time drive changes*)

   Perubahan gerak-waktu dilakukan oleh sistem kontrol sebagi respon terhadap pencapaian kondisi tertentu, dalam waktu tertentu atau dalam selang waktu tertentu setelah terjadi kondisi tertentu. Seperti pada perubahan gerak-kejadian, perubahan biasanya terjadi dari memulai sesuatu atau menghentikan sesuatu, dan waktu saat perubahan terjadi merupakan sesuatu yang utama. Contoh: dalam pabrik dengan waktu memulai dan waktu mengakhiri untuk kerja sift dan istirahat bagi semua pekerja, jam bengkel distel untuk membunyikan bel dalam saat tertentu selama hari tersebut untuk menyatakan waktu mulai dan berhenti kerja.

2.6 Mode Multiposisis

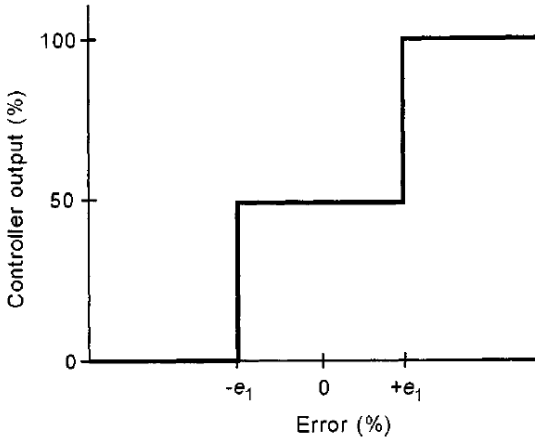
Mode multiposisi merupakan salah satu mode kontroler diskontinyu yang merupakan perluasan dari metode kontroler dua posis atau mode on-of, mode multi posisi ini digunakan dalam upaya untuk mengurangi siklus pengulangan, *overshoot, undershoot* yang menjadi sifat mode kontroler dua posisi. Kenyataannya, untuk mengurangi sifat diatas pada kontroler dua posisi biasanya menggunakan metode lain dikarenakan *output* tidak sesuai keinginan, mode ini ditunjukkan pada persamaan 2.1.

$$p = p_i \quad e_p > |e_i| \; i = 1, 2, 3 \ldots, n \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.1)$$

Error yang melebihi ketentuan dari ± e_i, dari kontroler *output* akan diubah kembali menjadi nilai baru pi, contoh yang sering digunakan adalah kontrol dengan tiga posisi yang ditunjukkan persamaan 2.2 [6].

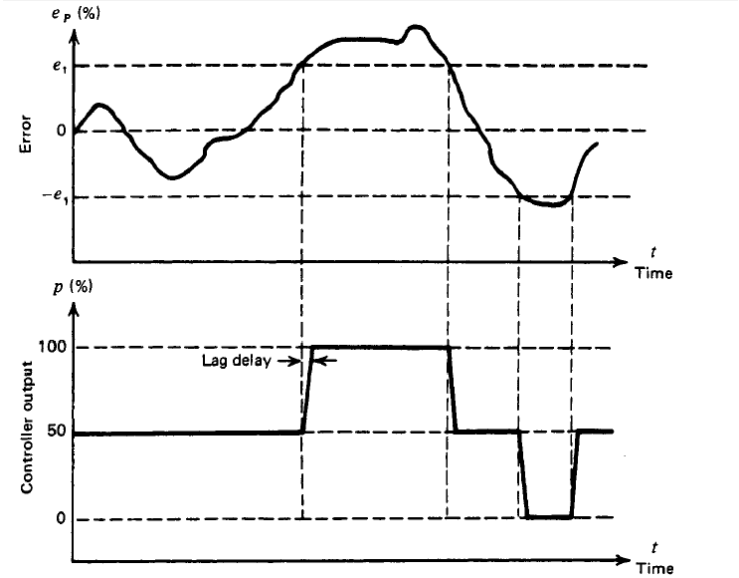
$$p = \begin{cases} 100 & e_p > e_2 \\ 50\ , & -e_{p1} < e_p < e_2 \\ 0 & e_p < -e_1 \end{cases} \quad \text{.....(2.2)}$$

Selama *error* diantara e_2 dan e_1 dari set point, kontroler akan selalu berada tetap di angka seting tertentu, yang diperlihatkan oleh outpu sebesar 50%, jika *error* melebihi set point sebesar e_1 atau lebih, *output* akan meningkat hingga 100%. Jika kurang dari set point e_1 atau lebih, maka output kontroler akan berkurang hingga nol.



Gambar 2.4 Kerja mode kontroler tiga posisi
(Johnson, Curtis; 2003)

Gambar 2.4 menggambarkan mode secara grafis. Beberapa titik netral yang sementara, biasanya muncul pada beberapa perubahan point. Tapi tidak berdasarkan desain. Oleh sebab itu bagian tersebut tidak ditampilkan. Tipe mode kontrol ini biasanya membutuhkan elemen pengontrol yang lebih rumit karena harus memiliki lebih dari dua pengaturan [6].



Gambar 2.5 Kerja kontroler tiga posisi
(Johnson, Curtis; 2003)

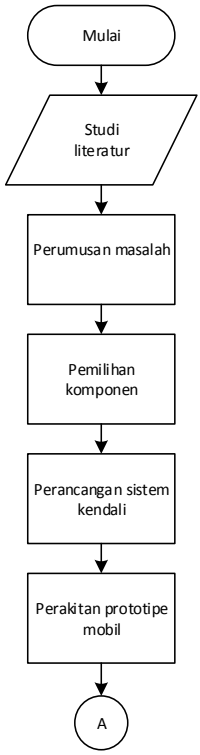
Gambar 2.5 menunjukkan grafik dari variabel dinamis dan pengaturan elemen kontrol terhadap waktu pada kasus mode kontroler tiga posisi. Perubahan pada penyetingan kontrol elemen dicatat sebagai variable yang berubah terhadap dua titik jalan. Pada grafik ini waktu yang ditetapkan untuk kontrol elemen untuk berubah dari satu posisi ke posisi yang

lain juga ditampilkan. Lihat *overshoot* dan *undershoot* dari *error* disekitar *setpoint* atas dan *setpoint* bawah. Ini terjadi pada kedua hal tersebut, yaitu jeda waktu proses dan kontrol ditunjukkan oleh batas waktu yang dibutuhkan untuk elemen kontrol mencapai pengaturan yang baru [6].

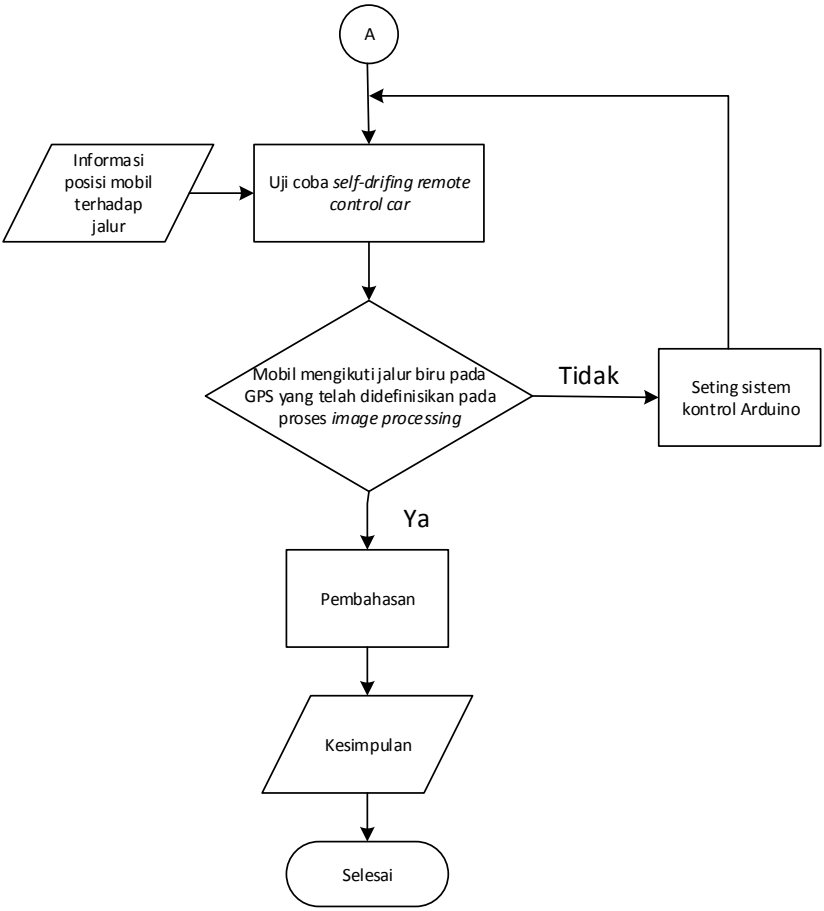
BAB III
METODE PENELITIAN

3.1 Diagram Alir Metodelogi Penelitian

Metodelogi penelitian ini menggambarkan langkah-langkah yang akan dilakukan dalam penelitian. Secara umum langkah-langkah yang akan dilakukan dalam penelitian ini ditunjukkan dalam diagram alir pada gambar 3.1 dan gabar 3.2.



Gambar 3.1 Diagram alir penelitian



Gambar 3.2 Diagram alir penelitian

3.2 Tahap Perancangan

Berdasarkan diagram alir penelitian pada gambar 3.1 dan gambar 3.2, berikut adalah penjelasan dari langkah-langkah penelitian secara detail:

1. Studi literatur
   Studi literatur dilakukan sebagai tahap awal dan dasar dalam memulai proposal tugas akhir. Pada tahap ini juga dilakukan pengumpulan data terkait dengan perancangan.

2. Perumusan masalah
   Langkah ini dilakukan untuk menentukan permasalahan yang akan dikaji dan dicari solusi terbaiknya. Dalam tugas akhir ini rumusan masalahnya adalah bagaimana merancang dan membangun sistem pengendali kendaraan berdasarkan informasi GPS berupa sudut antara kendaraan dan jalan.

3. Pemilihan komponen dan pembuatan alat.
   Pada tahap ini dilakukan pemilihan komponen-komponen yang sesuai terkait dengan perancangan sistem kendali otomatis berdasarkan mode kontroler multi posisi. Informasi lengkap mengenai perangkat keras penelitian tersebut adalah sebagai berikut:

   a. Laptop
      Laptop dengan spesifikasi memory 4GB RAM, intel® Core ™ i5. Laptop digunakan untuk mengolah masukan image dari kamera.

b. Handphone
   Handphone merupakan sebuah perangkat bantu yang digunakan untuk menjalankan aplikasi GPS.
c. Web camera
   Web camera merupakan perangkat yang digunakan untuk menangkap image dari *handphone*.
d. Arduino Uno
   Arduino merupakan sebuah pengendali mikro single-board yang bersifat open-source, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang.
e. Motor driver L298N
   Motor driver merupakan sebuah perangkat yang dapat mengendalikan arah putaran dan kecepatan motor DC.
f. Motor DC
   Motor DC merupakan jenis motor listrik yang bekerja menggunakan sumber tegangan DC. Motor DC menggunakan arus langsung, digunakan pada penggunaan khusus dimana diperlukan penyalaan torsi yang tinggi atau percepatan yang tetap.
g. Mobil remot kontrol
   Mobil remot control ini merupakan salah satu mainan anak-anak yang menyerupai mobil dengan laju dan beloknya dapat dikontrol dengan *controller* dengan sekala 1:10.
h. LCD
   LCD merupakan perangkat yang digunakan untuk menampilkan program.

4. Perancangan sistem kendali.
   Pada tahap ini dilakukan perancangan logika berpikir yang ada pada software Arduino, difokuskan pada kontrol radius belok pada roda depan dan kontrol motor DC bagian belakang.

5. Perakitan prototipe mobil
   Pada perakitan prototipe mobil ini dilakukan penyusunan seluruh komponen yang telah dipilih, kemudian mengkomunikasikan motor driver dengan Arduino. Mengkomunikasika motor driver dengan Arduino merupakan tahapan untuk menentukan besar PWM sehingga kecepatan dan sudut belok motor DC dapat diatur.

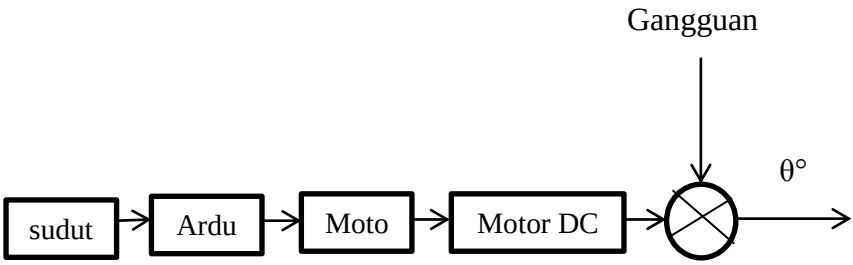
6. Mengkomunikasikan sudut hasil *image processing* yang masuk arduino dengan motor DC.
   Mengkomunikasika arduino dengan motor DC merupakan tahap berikutnya yang harus dilakukan untuk membuktikan hubungan antara besar sudut yang diperoleh dari pembacaan laptop sama dengan radius belok (θ) roda prototipe mobil.

7. Uji coba alat
   Uji coba alat ini dilakukan untuk mengetahui respon kontroler yang telah dirancang dan komponen pendukung, apakah telah terkomunikasi dengan baik.

3.3 Blok Diagram Sistem

Sistem kontrol yang digunakan dalam kendali otomatis berdasarkan metode image processing *GPS* adalah sistem diskrit *open loop* mode lima posisi dimana *input* berupa gambar diberikan oleh kamera untuk diolah oleh laptop melalui software *visula studio* 2012, dengan *output* berupa sudut akhir (α) sehingga

dapat dikomunikasikan dengan motor driver melalui Arduino. Setelah mendapatkan perintah dari Arduino, motor driver memberikan perintah pada motor DC depan dan motor DC belakang, motor DC depan akan bergerak sesuai dengan arah yang diberikan dan roda depan akan berbelok sebesar sudut (θ), dan nantinya sudut (θ) akan di *feedback* kan lagi untuk menentukan arah mobil berikutnya sedangkan motor DC belakang berputar secara konstan sehingga mobil remot kontrol bergerak maju. Sistem kerja motor DC depan dapat dilihat pada blok diagram yang ditunjukkan gambar 3.3.



Gambar 3.3 Blok diagram sistem kendali mobil otomatis

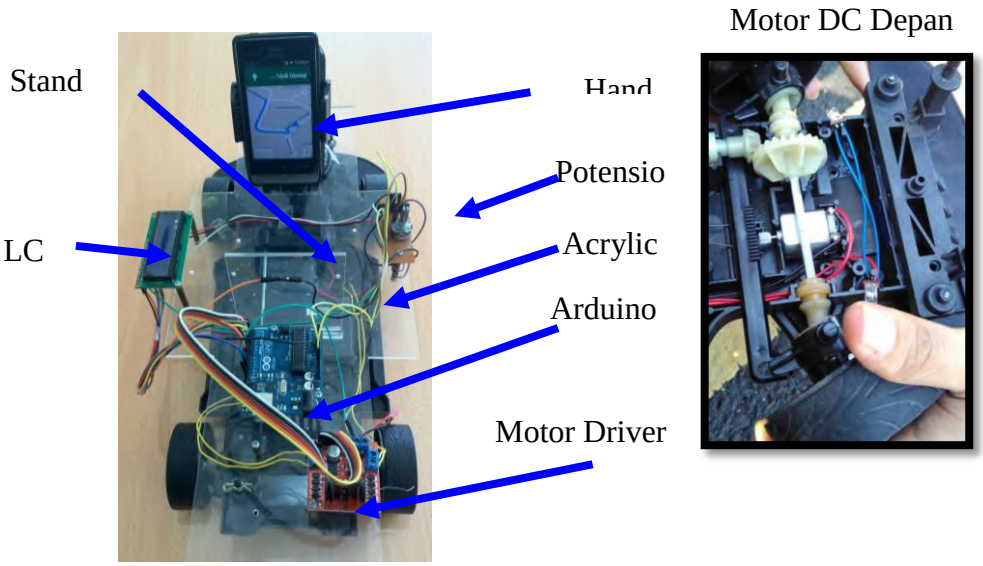
BAB IV
PERANCANGAN PROTOTIPE DAN SISTEM KENDALI *SELF-DRIVING REMOTE CONTROL CAR*

4.1 Konfigurasi Sistem Mekanik *Self-Driving Remote Control Car*

Self-driving remote control car merupakan suatu konstruksi mobil remot kontrol kendali otomatis yang memungkinkan untuk bergerak mengikuti jalur yang ditunjukkan oleh GPS. Sistem *self-driving remote control car* dibuat dari mobil remot kontrol yang beloknya tidak lagi dikendalikan oleh remot kontrol dan diganti dengan mikro kontroler. Biasanya mobil remot kontrol akan digerakkan oleh operator baik itu kecepatannya ataupun arah beloknya. Dalam tugas akhir ini, *self-driving remote control car* akan bekerja secara otomatis dengan inputan berupa *image* sudut antara pointer pada Google Maps dan jalan. *Image* yang menjadi inputan kemudian diolah dengan menggunakan *software* untuk dicari tahu selisih sudutnya. Setelah didapat selisih sudutnya kemudian dilakukan komunikasi ke mikrokontroller. Mikrokontroller yang mendapatkan informasi kemudian memberi perintah kepada motor DC untuk bergerak, sehingga *Self-driving remote control car* mampu bergerak sesuai dengan tujuan yang telah ditentukan.

Secara umum konstruksi dari *Self-driving remote control car* sangatlah sederhana, terdiri dari mobil remot kontrol yang terdiri dari rangka utama, motor DC belakang dan motor DC depan. Konstruksi *Self-driving remote control car* yang digunakan dalam tugas akhir ini ditunjukkan pada gambar 4.1. Gambar 4.1 menunjukkan bagian-bagian utama pada konstruksi *Self-driving remote control car*. Rangka utama berupa bodi mobil remot kontrol berfungsi sebagai tempat terpasangnya mekanisme gerak maju ataupun mundur motor DC belakang, dan tempat terpasangangnya mekanisme belok roda bagian depan beserta *gear box*-nya. *Acrylic* pada *self-driving remote control car* yang terpasang pada atas rangka mobil digunakan sebagai pondasi

untuk tempat mikro kontroller, *motor driver,* webcam dan *phone holder*.



Gambar 4.1 Konstruksi *self-driving remote control car.*

4.2 Konfigurasi Sistem Pengendali *Self-Driving Remote Control Car*

Pengaturan sudut dan arah belok dilakukan dengan menggunakan motor DC bagian depan, sedangkan untuk pengaturan arah laju mobil remot kontrol kedepan ataupun kebelakang dengan motor DC bagian belakang. Untuk mengontrol motor ini digunakan perangkat-perangkat pendukung seperti laptop, *handphone*, mikrokontroller, *motor driver*, dan program pendukung. Dalam mengoperasikan Arduino digunakan beberapa fungsi program yang telah disediakan dalam *library function* arduino diantaranya adalah sebagai berikut:

a. *Include* <studio.h>

Library stdio.h adalah pustaka pada bahasa C++ yang digunakan untuk operasi *input-output* (stdio = Standar *Input* dan *Output*). Tanpa menggunakan *library* ini maka perintah-perintah *input* dan *output* tidak dapat dieksekusi.

b. *Include* <math.h>

Library math.h merupakan sebuah pustaka yang disediakan untuk melakukan perhitungan secara matematika.

c. *Include* <stdlib.h>

Library stdlib.h kepanjangan dari standar *library* yang merupakan fungsi *header* merupakan fungsi untuk menggunakan standar pustaka dalam program.

d. *Include* <SoftwareSerial.h>

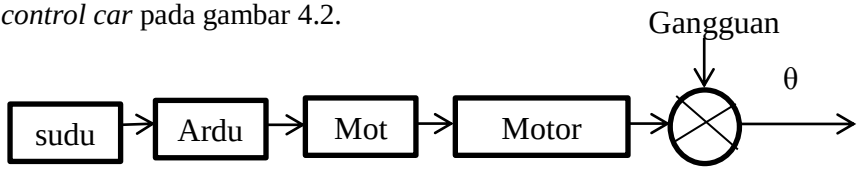
Include SoftwareSerial.h berfungsi untuk memanggil *library* komunikasi serial pada *software* arduino.

e. *Include* <LiquidCrystal.h>

Include <LiquidCrystal.h> merupakan pustaka khusus untuk LCD, dengan menggunakan pustaka ini pemrograman LCD menjadi sangat mudah.

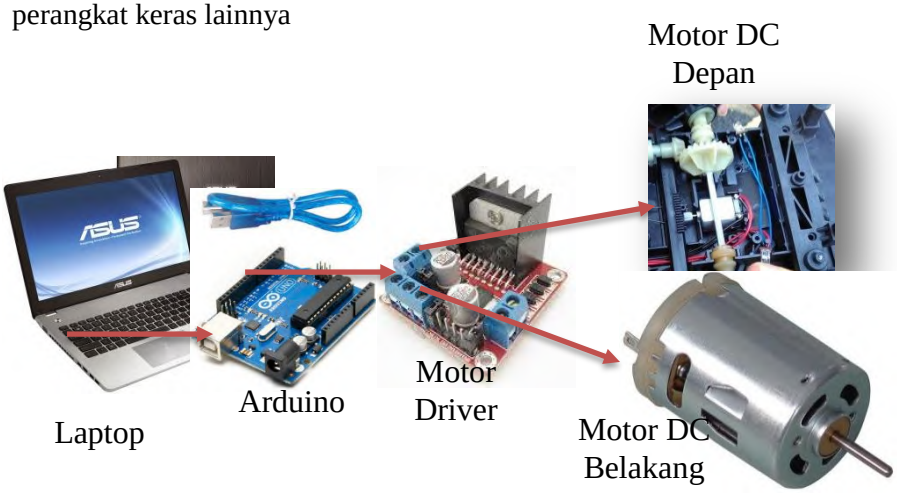
Dalam penggunaan *library-library* tersebut akan terlihat peran dan fungsinya ketika dikonfigurasikan dengan perangkat keras pendukungnya.

Berikut adalah skema sistem kendali *self-driving remote control car* pada gambar 4.2.



Gambar 4.2 Skema sistem kendali *self-driving remote control car*.

Gambar 4.3 menunjukkan konfigurasi perangkat keras sistem yang saling terhubung antara satu perangkat keras dengan perangkat keras lainnya



Gambar 4.3 Konfigurasi sistem kendali *self-driving remote control car* dan perangkat kerasnya

Gambar 4.2 menunjukkan bahwa sistem kontrol yang digunakan dalam *self-driving remote control car* adalah sistem *close loop* dimana *input* berupa gambar yang ditangkap oleh kamera untuk diolah oleh *PC* melalui *software* sehingga dapat dikomunikasikan dengan mikrokontroller pada *self-driving remote control car* melalui *minimum system.* Setelah mendapatkan perintah dari *minsys, self-driving remote control car* akan bergerak sesuai dengan perintah yang diberikan. Ketika mobil melaju dan roda depan yang berbelok akan dikoreksi oleh posisi mobil remot yang tampak pada layar *handphone*. Gambar baru diproses dengan *image proccesing* pada laptop dan memberikan *feedback* selisih sudut aktual dari *pointer* Google Maps dan jalan apakah *self-driving remote control car* telah

melaju sesuai dengan jalur yang telah ditetapkan atau tidak. Sementara konfigurasi sistem kontrol *self-driving remote control car* dan perangkat kerasnya dapat dilihat pada gambar 4.3. Diketahui bahwa mikrokontroller yang digunakan adalah Arduino jenis uno yang sekaligus merupakan minimum sistem yang sederhana. Minimun sistem inilah yang kemudian memberikan perintah kepada *motor driver* yang terletak di atas *acrylic* bagian belakang sebagai pengatur voltase yang masuk ke motor DC depan dan motor DC belakang.

4.3 Pemilihan Komponen

Proses pembuatan perangkat keras (*hardware*) unit kontrol untuk alat *self-driving remote control car* dengan dua motor DC ini memerlukan beberapa macam komponen. Komponen yang digunakan dalam tugas akhir ini memiliki spesifikasi seperti pada tabel berikut ini.

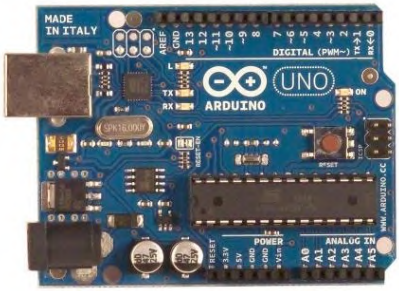
1. Mikrokontroller Arduino Uno

   Mikrokontroler adalah komputer mikro dalam satu chip tunggal. Mikrokontroler memadukan CPU, ROM, RWM, I/O paralel, I/O seri, *counter-timer*, dan rangkaian *clock* dalam satu chip. Dengan kata lain, mikrokontroler adalah suatu alat elektronika digital yang mempunyai masukan dan keluaran serta kendali dengan program yang bisa ditulis dan dihapus dengan cara khusus. Cara kerja mikrokontroler adalah membaca dan menulis data. Spesifikasi dari Arduino yang digunakan ditunjukkan table 4.1.

Tabel 4.1 Mikrokontroller Arduino Uno

Microcontroller	Arduino Uno
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V

Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 Kb (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm



Gambar 4.4 Arduino Uno Rev3.

Arduino uno merupakan papan mikrokontroler berbasis ATmega328 yang ditunjukkan gambar 4.4. Mikrokontroller ini beroperasi pada tegangan sebesar 5 volt untuk setiap komponen yang diaktifkannya. Agar arduino bekerja optimal baik dalam menggerakkan komponen motor maupun

menyalakan lcd, maka direkomendasikan untuk memberikan tegangan *input*an sebesar 7 hingga 12 volt. Akan tetapi jika komponen yang hendak dijalankan lebih dari 2 jenis misal dua motor, lcd dan led maka diperlukan tegangan yang lebih, biasanya diberi batasan inputan maksimal 20 volt. Arduino uno memiliki 14 digital pin *input/output*, dimana 6 pin digunakan sebagai *output* PWM, 6 pin input analog, dengan *clock speed* sebesar 16 MHz resonator keramik. Memiliki *connector* USB dan *jack* catu daya eksternal sebagai *input*an tegangan tambahan.

2. LCD (*Liquid Crystal Display*)

LCD adalah salah satu komponen elektronika yang berfungsi untuk menampilkan suatu data, baik karakter, huruf maupun grafik yang ditunjukkan gambar 4.5. LCD (*Liquid Cristal Display*) adalah salah satu jenis display elektronik yang dibuat dengan teknologi CMOS *logic* yang bekerja dengan tidak menghasilkan cahaya tetapi memantulkan cahaya yang ada di sekelilingnya terhadap *front-lit* atau mentransmisikan cahaya dari *back-lit*.

Gambar 4.5 LCD 16x2.

Proses pembacaan data pada register perintah biasa digunakan untuk melihat status *busy* dari LCD atau membaca *Address Counter*. RS diatur pada logika 0 untuk akses ke Register Perintah, R/W diatur pada logika 1 yang menunjukkan proses pembacaan data. 4 bit *nibble* tinggi dibaca dengan diawali pulsa logika 1 pada E Clock dan kemudian 4 bit *nibble* rendah dibaca dengan diawali pulsa

logika 1 pada E *Clock*. Untuk Mode 8 bit *interface*, pembacaan 8 bit (*nibble* tinggi dan rendah) dilakukan sekaligus dengan diawali sebuah pulsa logika 1 pada E *Clock*.

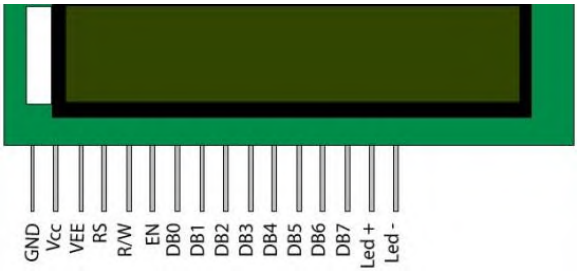
Berikut adalah fitur LCD yang digunakan:

a. Terdiri dari 16 karakter dan 2 baris.
b. Mempunyai 192 karakter tersimpan.
c. Terdapat karakter generator terprogram.
d. Dapat dialamati dengan mode 4-bit dan 8-bit.
e. Dilengkapi dengan *back light*.

Berikut adalah spesifikasi kaki LCD 16x2 yang ditunjukkan table 4.2

Tabel 4.2 Spesifikasi kaki LCD 16x2.

PIN	**Deskripsi**
GND	Ground
2	RS
Ground	RW
4	EN
13	D7
Resistor	VEE
7	D4
8	D5
12	D6



Gambar 4.6 Pin pada LCD 16x2.

Gambar 4.6 menunjukkan lcd 16x2 memiliki 16 pin. Dimana tiap-tiap pin memiliki peran dan fungsi yang berbeda. Pin 1 merupakan pin yang harus dihubungkan dengan *ground* pada mikrokontroller. Pin 2 adalah pin Vcc, bagian ini memiliki fungsi yaitu sebagai penyuplai tegangan. Secara umum LCD membutuhkan tegangan sebesar 5V (4.7V – 5.3V). Pin 3 merupakan pin yang berfungsi sebagai pengatur kontras pada lcd, biasanya pin ini juga dihubungkan dengan variabel resistor. Pin 4 pada lcd berfungsi untuk menangkap sinyal ketika register data yang diterima *high* atau *low*. Pin 5 pada lcd berfungsi meneruskan perintah berupa register data *high* atau *low*, jika data *high* maka akan dibaca namun bila data low maka akan diterjemahkan sebagai perintah tulis oleh pin 5. Pin 6 merupakan pin yang berfungsi untuk mengirimkan sinyal data ke pin data saat diberikan *pulse* dari tinggi ke rendah. Pin 7 sampai pin 14 merupakan pin data dengan mode 8-bit. Pin data ini berfungsi sebagai penerima sinyal *input* atau *output* data sesuai dengan kebutuhan, LCD pada tugas akhir ini digunakan untuk menampilkan nilai sudut akhir dan mengetahui apakah data nilai sudut akhir yang telah terproses telah terbaca oleh Arduino.

3. Kabel *Jumpper*

Kabel merupakan salah satu komponen yang dibutuhkan dalam perancangan sistem kendali *self-driving remote control car*. Untuk menghubungkan antara LCD dengan mikrokontroller maka diperlukanlah kabel *jumper* sebagai penghubungnya. Kabel yang khusus ujung pinnya disesuaikan dengan lubang-lubang breadboard, ujungnya agak kaku dan tengahnya lentur atau lemas seperti kabel biasa, berikut pada gambar 4.7 merupakan gambar dari kabel *jummper*.

Gambar 4.7 Kabel *Jumper Male Female.*

Secara umum kabel *jumpper* yang digunakan adalah *male to female, female to female dan male to male.* Kabel ini memiliki ukuran panjang yang berbeda-beda, mulai dari 15 cm, 20 cm hingga 30 cm. Selain itu ujung dari kabel *jumpper* juga beragam, ada yang pipih menempel ada pula yang runcing namun tidak melekat satu sama lain.

4. Motor Driver L298N

Motor *driver* L298N yang ditunjukkan pada gambar 4.8 adalah jenis IC driver motor yang dapat mengendalikan arah putaran dan kecepatan motor DC ataupun motor stepper. Mampu mengeluarkan *output* tegangan untuk motor DC dan motor stepper sebesar 50 volt. IC l298 terdiri dari transistor-transistor logik (TTL) yang memudahkan dalam menentukkan arah putaran suatu motor DC dan motor stepper. Dapat mengendalikan 2 untuk motor DC namun pada hanya dapat mengendalikan 1 motor stepper. Penggunaannya paling sering untuk robot line follower. Bentuknya yang kecil memungkinkan dapat meminimalkan pembuatan robot *line follower*. Dalam tugas akhir ini motor driver digunakan untuk mengatur besar voltase dan arah putaran motor DC sehingga

mampu mengatur laju dan arah dari *self-driving remote control car*.

Gambar 4.8 Motor driver L298N.

Tabel 4.3 menunjukkan spesifikasi dari *motor driver* L298N yang digunakan.

Tabel 4.3 Spesifikasi *motor driver* L298N.

PIN	**Deskripsi**
in1	Enable Motor A
in2	Enable Motor A
in3	Enable Motor B
in4	Enable Motor B
GND	Ground
5v	Vcc
EnA	Enable PWM signal for motor A
EnB	Enable PWM signal for motor B
Spesification	Deskripsi
Logical voltage	5V Drive voltage
Logical	Max 2 Ampere

current	
Max power	25W
Dimension	43 x 43 x 26 mm
Weight	26 gram

Berdasarkan *datasheet motor driver* L298N pada tabel 4.3 diatas maka dapat diketahui bahwa modulasi dari motor driver tersebut adalah digital dengan catu daya 5 volt dan maksimal arus yang bias masuk adalah 2 ampere.

5. *Power Supply*

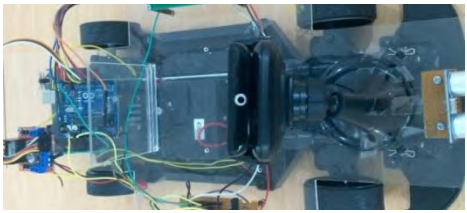
Power Supply atau dalam bahasa indonesia disebut dengan catu daya adalah suatu alat listrik yang dapat menyediakan energi listrik untuk perangkat listrik ataupun elektronika lainnya yang ditunjukkan pada gambar 4.9. Oleh karena itu, *power supply* kadang-kadang disebut juga dengan istilah *Electric Power Converter*. Pada tugas akhir ini digunakan 1 buah baterai lipo dengan 3 cell dengan voltase 11.1 Volt dan kapasitas minimum 2800 mAh, dimana baterai lipo digunakan sebagai *power supply* motor driver yang digunakan untuk menggerakkan motor DC belakang dan motor DC depan pada *self-driving remote control car.*

Gambar 4.9 Baterai lipo.

4.4 Perakitan Alat

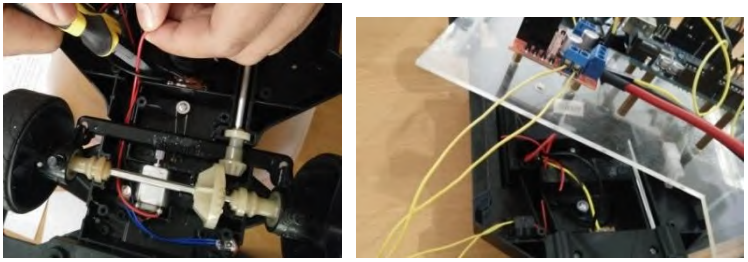
Dalam pembuatan alat *Self-driving remote control car* ada beberapa tahapan yang dilakukan untuk merakit seluruh komponen yang telah dipilih yaitu:

1. Membuat papan dari akrilik.
   Pembuatan papan dari akrilik bertujuan untuk tempat peletakan komponen elektrik seperti Arduino, Motor Driver L298N, LCD, dan lain-lain yang terlihat pada gambar 4.10.



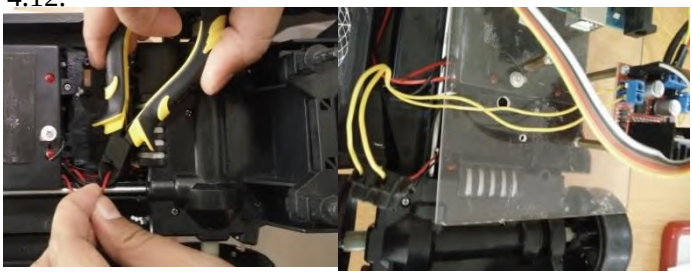
Gambar 4.10 Papan akrilik.

2. Memotong kabel pada motor DC depan dari radio kontroler.
   Kabel motor DC depan dipotong karena motor DC depan tidak lagi menerima perintah dari radio kontroler tetapi menerima perintah melalui mikro kontroler Arduino. Kabel dari motor DC depan dihubungkan ke Motor Driver L298N karena Arduino tidak dapat langsung mengatur putaran motor DC depan yang terlihat pada gambar 4.11.



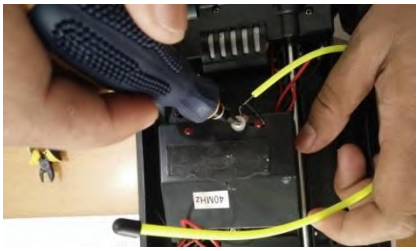
Gambar 4.11 Kabel positif dari motor DC depan dipotong.

3. Memotong kabel pada motor DC belakang dari radio kontroler.
   Kabel motor DC belakang dipotong karena motor DC belakang tidak lagi menerima perintah dari radio kontroler tetapi menerima perintah melalui mikro kontroler Arduino. Kabel dari motor DC belakang dihubungkan ke Motor Driver L298N karena Arduino tidak dapat langsung mengatur putaran motor DC depan yang terlihat pada gambar 4.12.



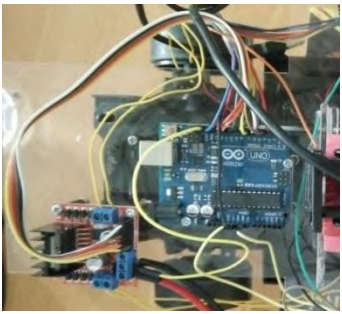
Gambar 4.12 Kabel dari motor DC belakang dipotong.

4. Melepas antena *reciver.*
   Antena pada mobil remot kontrol dilepas karena sudah tidak digunakan lagi yang ditunjukkan gambar 4.13. Kontroler beralih dari radio kontroler ke Arduino.



Gambar 4.13 Antena penerima sinyal dilepas.

5. Menghubungkan motor driver ke Arduino.
   Motor driver dihubungkan ke Arduino seperti yang ditunjukkan gambar 4.14. Meskipun motor driver merupakan perangkat sinyal digital, namun tidak semua pin bisa digunakan, hanya pin khusus yang mengatur PWM dengan simbol ” ~ ” didepan nomornya.

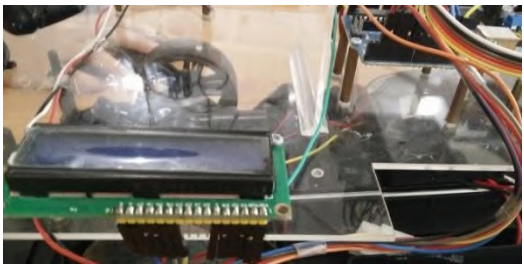


Gambar 4.14 Motor driver L298N dihubungkan di pin PWM Arduino.

6. Menghubungkan baterai lipo dengan motor driver.
   Motor driver memerlukan catu daya sebesar 5 volt dan arus sebesar 2 amper untuk menggerakkan dua buah motor DC. Pada kutub negatif dihubungkan ke pin ground pada Arduino. Batrei lipo 3 cell ini dipilih karena memiliki masa pakai yang lebih lama.

Gambar 4.15 Baterai lipo dengan motor driver dihubungkan.

7. Menghubungkan LCD dengan Arduino.
   LCD dihubungkan dengan Arduino bertujuan untuk mengetahui apakah data pebacaan dari Visual Studio telah terbaca atau belum pada Arduino yang terlihat pada gambar 4.16. Pin pada LCD yang dihubungkan ke Arduino yang ditunjukkan pada gambar 4.16 sesuai dengan table 4.2 sedangkan resistor yang digunakan adalah variabel resistor sehingga dapat diatur kontras dari LCD dengan memutar variabel resistor.



Gambar 4.16 Menghubungkan LCD dan Arduino.

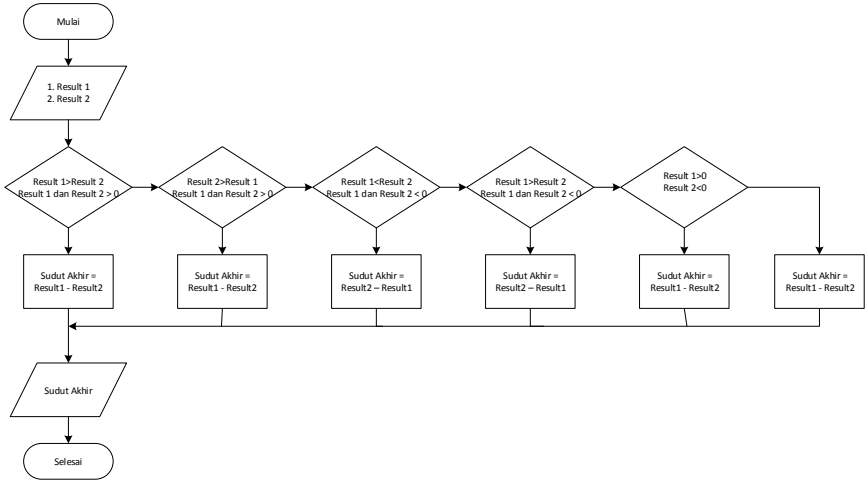
8. Memasang *holder smart phone* di bagian depan *stand* kamera.
   Holder dipasang pada bagian tempat peletakan kamera, holder yang digunakan adalah *holder smart phone* yang terlihat pada gambar 4.17.

Gambar 4.17 *Holder* untuk memegang *smart phone.*

Seluruh rangkaian *Self-driving remote control car* dapat dilihat pada gambar 4.1, namun *Self-driving remote control car* belum mampu bekerja karena dibutuhkan kode pemrograman untuk membuatnya bergerak sesuai dengan yang diinginkan, kode pemrograman ini akan dibahas pada subab berikutnya.

4.5 Penentuan dan Pengujian Nilai Sudut Akhir

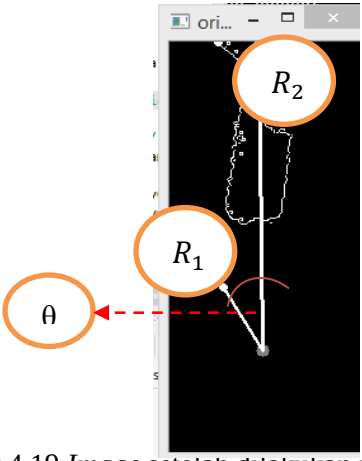
Teori persamaan garis lurus memegang peranan penting karena bagian ini merupakan ilmu dasar yang akan diimplentasikan pada program aplikasi. Secara sederhana persamaan garis lurus dapat didefinisikan sebagai sebuah garis lurus dimana posisinya ditentukan oleh sebuah persamaan dan apabila persamaan tersebut digambarkan pada bidang kartesius maka akan menghasilkan sebuah garis yang lurus, salah satu contoh persamaan garis lurus adalah y = mx + c. Pada tugas akhir ini digunakan sistem garis lurus, sistem garis lurus yang digunakan melalui titik (0,0) sehingga menghasilkan sebuah sudut terhadap garis inisial.



Gambar 4.18 *Flowchart* penentuan nilai sudut akhir.

Berdasarkan *flowchart* yang ditunjukkan gambar 4.18, untuk memperoleh sudut akhir dicari selisih sudut antara result 1 yang merupakan sudut dari pointer dengan result 2 yang merupakan sudut jalan. Sehingga untuk mendapatkan sudut akhir dilakukan pengurangan dengan menggunakan 6 buah kondisi dalam dua buah kuadran, yaitu kuadran 1 dan kuadran 2 pada bidang kartesian yang ditunjukkan gambar 4.19.

Dari gambar 4.19 ditunjukkan dalam enam buah kondisi, sehingga muncul enam buah persamaan dan enam buah konstrein yang ditunjukkan pada persamaan berikut ini.



Gambar 4.19 *Image* setelah dilakukan pemrosesan sudut akhir pada Visual Studio.

Dimana :

R_1 = Hasil sudut dari pembacaan segitiga.

R_2 = Hasil sudut dari pembacaan jalan.

θ = Selisih sudut antara result 1 dan result 2.

Kondisi 1

Sudut akhir = result 1 – result 2 $\begin{cases} Result\ 1 > result\ 2 \\ Result\ 1\ dan\ result\ 2 > 0 \end{cases}$

Kondisi 2

Sudut akhir = result 1 – result 2 $\begin{cases} Result\ 1 < result\ 1 \\ Result\ 1\ dan\ result\ 2 > 0 \end{cases}$

Kondisi 3

Sudut akhir = result 2 – result 1 $\begin{cases} Result\ 1 < result\ 2 \\ Result\ 1\ dan\ result\ 2 < 0 \end{cases}$

Kondisi 4

Sudut akhir = result 2 – result 1 $\begin{cases} Result\ 1 > result\ 2 \\ Result\ 1\ dan\ result\ 2 < 0 \end{cases}$

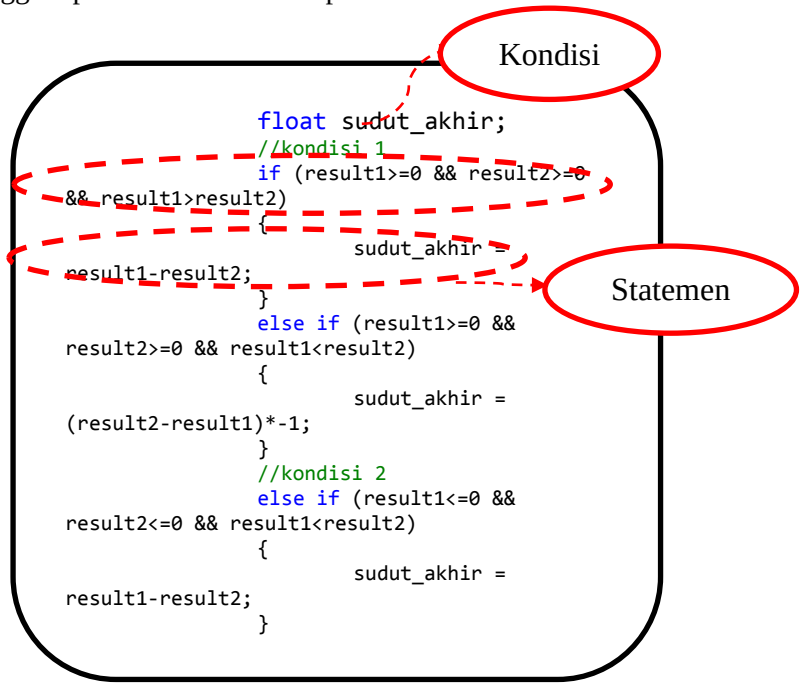
Kondisi 5

Sudut akhir = result 1 – result 2 $\begin{cases} Result\ 1 > 0 \\ Result\ 2 < 0 \end{cases}$

Kondisi 6

Sudut akhir = result 1 – result 2 $\begin{cases} Result\ 1 < 0 \\ Result\ 2 > 0 \end{cases}$

Dari persamaan diatas kita rubah kedalam bahasa C++ pada visual studio dengan mnambahkan float sudut_akhir pada *header* sehingga diperoleh nilai desimal pada nilai sudut akhir.



Gambar 4.20 Program sudut akhir pada visual studio.

Dalam pemrograman pada visual studio menggunakan if dan else if yang artinya apabila kondisi 1 bernilai benar statemen 1 dijalankan. Apa bila kondisi 1 bernilai salah dan kondisi 2 bernilai benar maka statemen 2 dijalankan, if....else if dipilih dikarenakan lebih berguna untuk memilih salah satu dari banyak kondisi sudut akhir yang akan di komunikasikan dengan Arduino seperti yang ditunjukkan gambar 4.20.

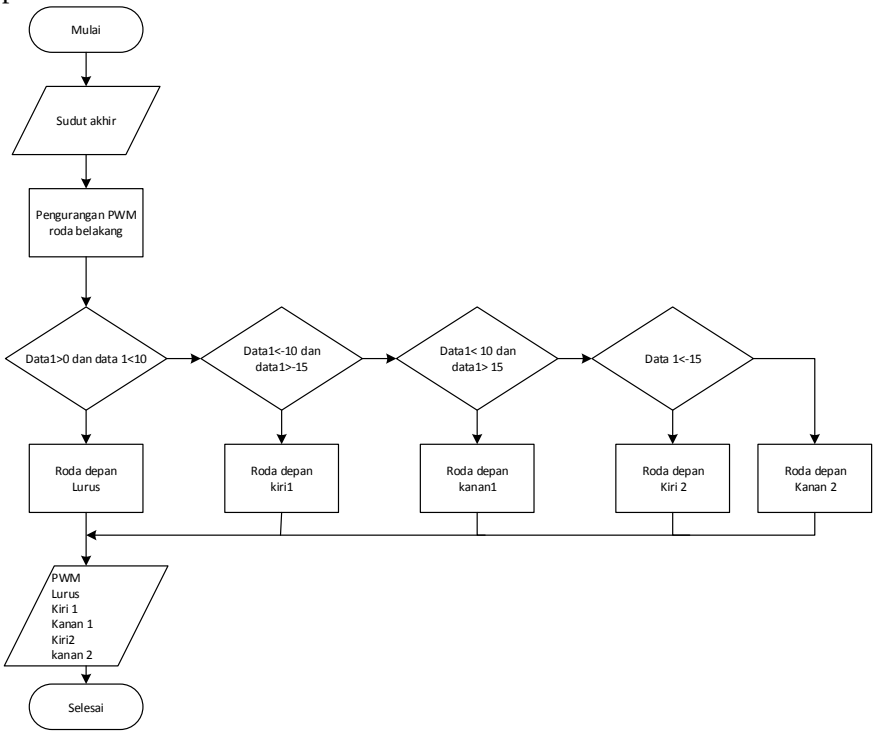
4.6 Penentuan Radius Belok pada Sistem Pengendali *Self-Driving Remote Control Car.*

Dari perhitungan pada gambar 4.19 akan diperoleh nilai positif ataupun negatif untuk sudut akhir, yang nantinya ketika bernilai positif mobil akan berbelok kesebelah kanan dan ketika bernilai negatif mobil akan berbelok kekiri. Nilai inilah yang nantinya akan akan dikomunikasikan dengan Arduino uno.

Untuk program pada Arduino digunakan mode kontroler proporsional dan menggunakan lima posisi sebagai set poin yang ditunjukkan pada *flow chart* pada gambar 4.21. Nilai yang masuk berupa sudut akhir merupakan nilai yang diperoleh dari flowchart pada gambar 4.18 dimana nilai sudut akhir ini ditentukan dengan kejadian-kejadian yang mungkin terjadi pada saat pemrosesan gambar bisa bernilai positif ataupun sebaliknya.

Gambar 4.21 menjelaskan program pada Arduino yang melakuan dua buah pengontrolan. Kontrol pertama mengatur kecepatan mobil dimana seiring bertambahnya waktu PWM mobil akan mengalami penurunan yang ditampilkan pada LCD, hal ini dilakukan karena pada awalnya mobil perlu torsi besar untuk bergerak, sehingga dibutuhkan maksimal PWM sebesar 255 dan pada saat mobil telah melaju sekitar 250 mikrodetik terjadi penurunan PWM mencapai 140, angka 140 ini didapatkan dari beberapa percobaan di jalan agar memudahkan dalam pengambilan data dan pengamatan terhadap *self-driving remote control car* yang melaju dijalan, dikarenakan *self-driving remote control car* terhubung dengan kabel sepanjang 3 meter yang merupakan kabel arduino dan kabel kamera sepanjang 2 meter

yang keduanya terhubung dengan laptop yang dibawa oleh penulis.



Gambar 4.21 *Flowchart* kendali Arduino.

Kode pemrograman dari degradsi kecepatan dan *display* dari degradasi PWM ditunjukkan gambar 4.22 dibawah ini.

```
for(int i=255;i>150;i--){
  kecepatan=i;
  delay(250);
  lcd.setCursor(5,1);
  lcd.print(i);
  maju();
}
```

Gambar 4.22 Kode pemrograman kendali Arduino roda belakang.

Kontrol kedua mengatur radius belok dan arah belok mobil, dalam pengontrolan ini ditentukan 5 posisi dalam arah belok mobil yaitu lurus, kiri 1, kanan 1, kiri 2, dan kanan 2. Dalam pemrograman ini kita buat dulu enam buah void setup untuk menentukan nilai PWM yang akan diberikan pada masing-masing kejadian seperti halnya lurus, kiri 1, kanan 1, kiri 2, dan kanan 2, *set up* ini yang nantinya akan di disesuaikan dengan pin-pin yang ada pada motor driver yang ditunjukkan gambar 4.23.

```
void lurus(){
  analogWrite(ena_de,0);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  delay(50);
  }

void kiri1(){
  analogWrite(ena_de,120);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  delay(50);
  }
void kiri2(){
  analogWrite(ena_de,200);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
```

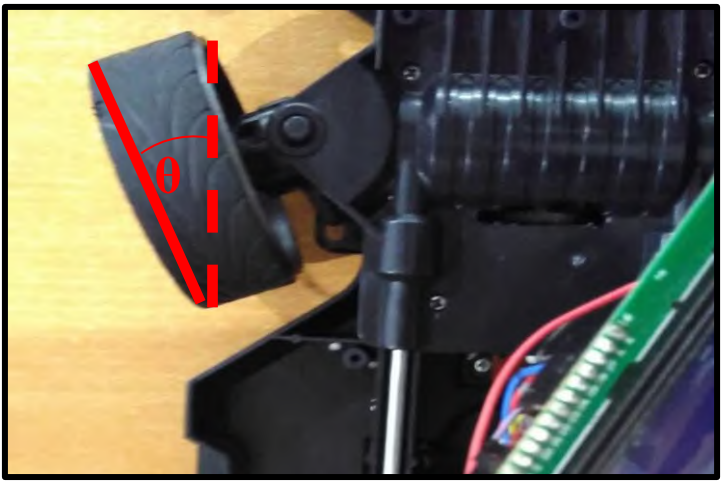
```
void kanan1(){
    analogWrite(ena_de,120);
    digitalWrite(in2,HIGH);
    digitalWrite(in1,LOW);
    delay(50);
  }
void kanan2(){
    analogWrite(ena_de,200);
    digitalWrite(in2,HIGH);
    digitalWrite(in1,LOW);
    delay(50);
  }
void maju(){
  analogWrite(ena_be,kecepatan);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
}
```

Gambar 4.23 Kode pemrograman kendali Arduino void setup.

High dan *low* merupakan nilai sinyal yang dikirimkan ke motor driver yang menyebabkan motor driver berputar kearah kanan ataupun kearah sebaliknya, sedangkan nilai PWM ini berpengaruh terhadap besar sudut belok roda dengan nilai PWM dari 0 hingga 225, penentuan nilai PWM ini dilakukan dengan melakukan ujicoba pada mobil yang rodanya menyentuh meja sehingga gaya geseknya telah diperhitungkan seperti yang ditunjukkan gambar 4.24 dan data hubungan antara PWM dan besar sudut belok mobil dapat dilihat di lampiran.

Kontrol kedua mengatur radius belok dan arah belok mobil, dalam pengontrolan ini ditentukan 5 posisi dalam arah belok mobil yaitu lurus, kiri 1, kanan 1, kiri 2, dan kanan 2. Kondisi

roda mobil lurus terjadi ketika nilai θ sebesar 0° kondisi ini diperoleh ketika PWM roda depan kita tentukan sebesar 0, kondisi lurus ini terjadi saat nilai sudut akhir dari pembacaan *image processing* berada pada rentang -10° hingga 10° sehingga dalam jangkauan nilai ini mobil akan terus berjalan lurus dijalan dengan meniadakan gangguan berupa kemiringan jalan ataupun kerikil yang mampu menyebabkan mobil belok dengan sendirinya. Kondisi roda mobil berbelok kiri 1 terjadi ketika nilai θ sebesar 17° kondisi ini diperoleh ketika PWM roda depan kita tentukan sebesar 120, kondisi belok kiri 1 ini terjadi saat nilai sudut akhir dari pembacaan *image processing* berada pada rentang -11° hingga -15°. Kondisi roda mobil berbelok kanan 1 terjadi ketika nilai θ sebesar 17° kondisi ini diperoleh ketika PWM roda depan kita tentukan sebesar 120, kondisi belok kanan 1



Gambar 4.24 Sudut belok roda mobil depan (θ).

terjadi saat nilai sudut akhir dari pembacaan *image processing* berada pada rentang 10° hingga 15°. Kondisi roda mobil berbelok

kiri 2 terjadi ketika nilai θ sebesar 35° kondisi ini diperoleh ketika PWM roda depan kita tentukan sebesar 200, kondisi belok kiri 2 ini terjadi saat nilai sudut akhir dari pembacaan *image processing* berada pada rentang kurang dari -15°. Kondisi roda mobil berbelok kanan 2 terjadi ketika nilai θ sebesar 35° kondisi ini diperoleh ketika PWM roda depan kita tentukan sebesar 200, kondisi belok kanan 2 ini terjadi saat nilai sudut akhir dari pembacaan *image processing* berada pada rentang lebih dari 15°. Pembagian posisi ini dapat dilakukan posisi yang lebih banyak.

```
    lcd.print("    ");

    if(datal>=0 && datal<=10 ){
      lurus();
      delay(10);
    }
    else if(datal<0 && datal>=-10){
      lurus();
      delay(10);
    }
    if(datal>10 && datal<=15 ){
      kanan1();
      delay(10);
    }
    else if(datal<-10 && datal>=-15){
      kiri1();
      delay(10);
    }
    if(datal>15 && datal<=53 ){
      kanan2();
      delay(10);
    }
    else if(datal<-15 && datal>=-53){
      kiri2();
      delay(10);
    }
```

Gambar 4.25 Kode pemrograman kendali Arduino *if statement conditional.*

Namun dalam prakteknya menyebabkan kelambatan dari pembacaan Arduino terhadap pembacaan Visula Studio sehingga pada display LCD akan terlihat kalu sudut akhir yang ditampilkan tidak sesuai dengan pembacaan sudut akhir terbaru pada Visual Studio.

Dalam pembuatan kode pemrograman ini kita tentukan dulu void-void untuk tiap-tiap posisi sehingga muncullah 5 buah void yaitu void lurus, void kiri 1, void kanan 1, void kiri 2, dan void kanan 2. Setelah kita tentukan void-void ini barulah dilakukan beberapa pengkondisian if....else if untuk mengaktifkan masing-masing void setup yang terlihat pada gambar 4.25 dibawah ini

4.7 Komunikasi Sistem *Self-driving Remote Control Car* dengan Kamera

Berdasarkan hasil perhitungan nilai sudut akhir yang diperoleh pada perhitungan data pada gambar 4.19, selanjutnya akan dilakuakan komunikasi sistem *self-driving remote control car* dengan kamera. Untuk dapat mengkomunikasikan kedua buah *hardware* ini, maka dilakukanlah penggabungan *software* antara Visual Studio dengan Arduino.

Berikut adalah alur komunikasinya yang dapat dilihat pada gambar 4.26.

Gambar 4.26 Komunikasi antara kamera dengan *self-driving remote control car*.

Kamera dalam sistem ini merupakan penangkap gambar yang bertujuan untuk mengidentifikasi sudut pada pointer, sudut jalan, dan sudut akhir yang merupakan selisih antara sudut pointer dan jalan. Data-data berupa sudut akhir merupakan data yang dikomunikasikan dengan Arduino

Sedangkan pada program Arduino nilai sudut akhir dipanggil dengan program Serial.read seperti program yang ditunjukkan pada gambar 4.27

```
void cek(){
     while (Serial.available() > 0) {
     data1 = Serial.read();
```

Gambar 4.27 Program pembacaan sudut akhir dari Visual Studio.

BAB V
PENGUJIAN

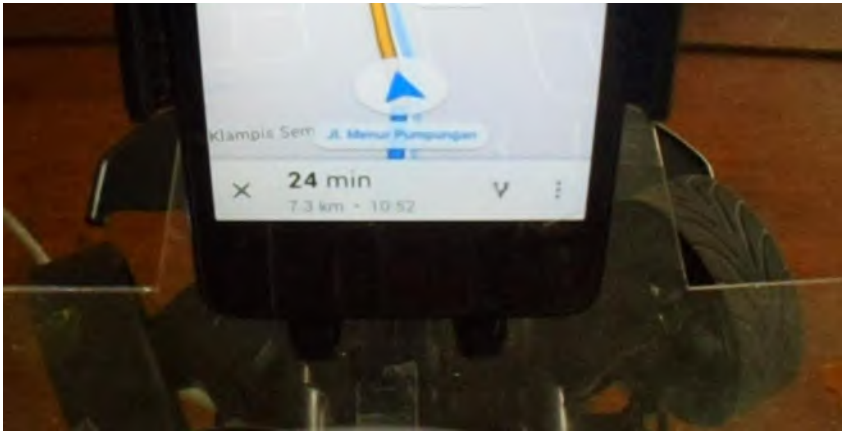
Dalam pengujian belok *Self-driving remote control car*, radius dan arah belok roda depan direkam apakah telah sesuaikan dengan tampilan nilai sudut akhir yang tertera pada LCD atau belum, dengan dua keadaan yaitu *indoor* dan *outdoor*. Sehingga dapat dilihat seperti apa pergerakan dari *self-driving remote control car* apakah pergerakannya sesuai dengan jalur yang telah ditunjukkan oleh Google Maps atau tidak. Pengujian *self-driving remote control car* pada ruang tertutup dapat dilihat pada gambar 5.1 sedangkan pengujian pada ruang terbuka dapat dilihat pada gambar 5.2 dengan menggunakan alat bantu berupa kain penutup. Kain penutup digunakan untuk mengurangi gangguan dari intensitas cahaya matahari yang terlalu terik.

5.1 Pengujian Dalam Ruangan Tertutup

Pada pengujian dalam ruang tertutup dilaksanakan di Laboratorium Otomasi Industri Teknik Mesin ITS, menggunakan aplikasi *video recorder* untuk Android, dari *video recorder* Google Maps hasil dari navigasi yang dijalankan sepeda motor dijadikan sebagai *video capture*, diperoleh hasil yang lebih baik pada pengujian dalam ruangan tertutup dikarenakan gangguan yang disebabkan oleh cahaya dan getaran dapat dikurangi, dan juga tidak membutuhkan kabel tambahan untuk pengambilan data sehingga respon mobil dapat diamati dengan lebih mudah.

Dari pengujian ini didapatkan beberapa kondisi respons roda mobil terlambat berbelok dikarenakan adanya salah pembacaan nilai sudut akhir, kesalahan ini terjadi pada belokan dengan sudut 90°, putar balik, dan belokan beruntun yang saling berdekatan. Respons lambat roda mobil yang ditunjukkan video 1 detik ke-39 dikarenakan motor DC tidak mampu langsung membaca nilai sudut akhir, sehingga dibuat void-void yang ditunjukkan gambar 4.23 untuk merubah PWM ke nilai sudut belok roda yang sesuai nilai sudut akhir. Berbeda dengan motor

servo yang mampu menterjemahkan nilai sudut akhir yang dibaca oleh Arduino menjadi radius belok roda secara langsung.



Gambar 5.1 Pengujian *self-driving remote control car* pada ruang tertutup.

5.2 Pengujian Dalam Ruangan Terbuka

Pada pengujian dalam ruang terbuka dilakukan di Perumahan Manyar Kartika, diperoleh hasil yang kurang maksimal dikarenakan banyaknya gangguan diantaranya adalah cahaya, kualitas lintasan yang bergelombang menyebabkan roda susah berbelok sesuai perintah kode pemrograman, dan kesalahan dalam pengambilan data. Untuk mengatasi masalah cahaya digunakan penutup dari kain hitam untuk mengurangi intensitas cahaya dari lingkungan sekitar.

Dalam penerapan *self-driving remote control car* masih terdapat pergerakan roda depan yang dinilai tidak benar, karena alat tidak bergerak mengikuti arah pointer dan jalan pada Google Maps yang ditangkap oleh kamera. Sehingga terlihat mobil remot

control berbelok lebih dulu saat mencapai belokan hal ini juga disebabkan pembacaan program, pada result 2 yang terbaca adalah sejauh 8 meter didepan kendaraan.

Gambar 5.2 Pengujian *self-driving remote control car* pada ruang terbuka.

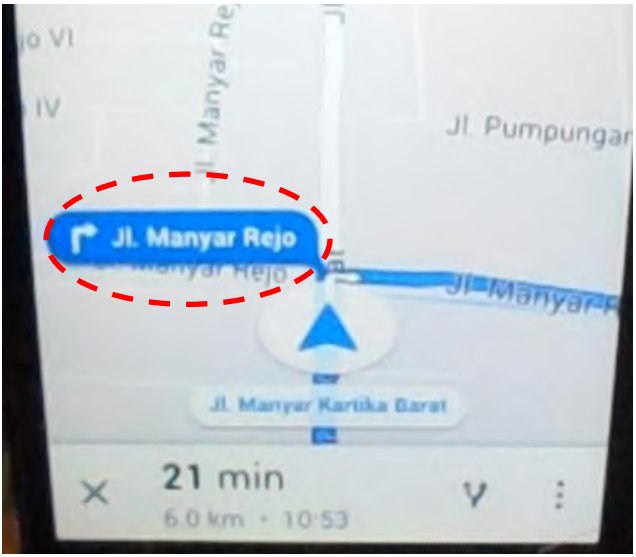
Beberapa faktor yang menjadi pengaruh dari kesalahan ini diantaranya adalah:

1. Faktor kamera yang kurang baik membuat inputan nilai untuk sudut akhir menjadi berubah-ubah terhadap kondisi sebenarnya, dikarenakan kemampuan kamera menterjemahkan warna terbatas dan posisi kamera yang mudah berubah.

2. Kecepatan pembacaan pada mikrokontroler tidak sama dengan pembacaan image processing pada laptop, dikarenakan kecepatan pengolahan data (*clock speed*)

pada laptop sebesar 2.2 GHz melebihi kecepatan daripada Arduino sebesar 16 MHz.

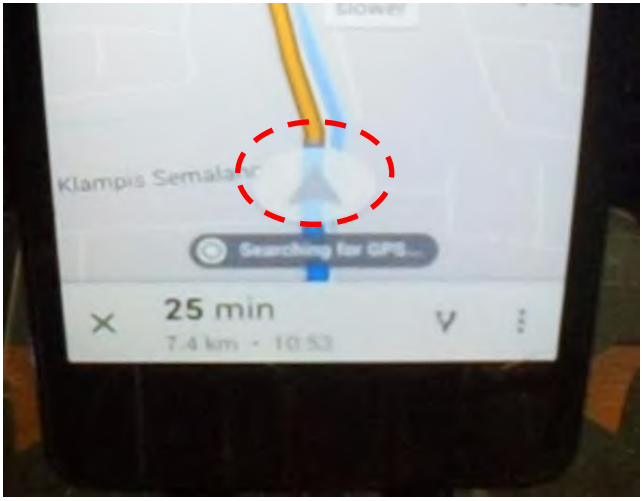
3. Pada Google Maps terbaru 9.27.2 muncul *pop-up* nama jalan yang ditandai garis putus-putus merah pada gambar 5.3, Hal ini terjadi ketika navigasi jalan menunjukkan akan berbelok kesebelah kanan atau kesebelah kiri menyebabkan adanya kesalahan pembacaan sudut belok, dikarenakan centroid result 2 berubah.



Gambar 5.3 *Pop-up* nama jalan

4. Pada Google Maps terkadang pointer akan berubah menjadi abu-abu atau terjadi perubahan warna yang ditandai garis putus-putus merah pada gambar 5.4. Hal ini terjadi ketika kita berhenti beberapa saat ataupun ketika koneksi terputus dikarenakan satelit tidak mampu membaca posisi kita, sehingga mengakibatkan kesalahan

pembacaan nilai sudut akhir dikarenakan nilai resul 1 *error* .



Gambar 5.4 Warna pointer berubah

(Halaman ini sengaja dikosongkan)

Lampiran

Program Sudut Akhir

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/core/core.hpp>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>

using namespace cv;
using namespace std;

int HueL_jalan=0;
int HueM_jalan=255;

int HueL_pointer=0;
int HueM_pointer=255;

int SaturationL_jalan=0;
int SaturationM_jalan=255;

int SaturationL_pointer=0;
int SaturationM_pointer=255;

int ValueL_jalan=0;
int ValueM_jalan=255;

int ValueL_pointer=0;
int ValueM_pointer=255;

int BlurL_jalan=0;
```

```
int BlurM_jalan=255;

int BlurL_pointer=0;
int BlurM_pointer=255;

int Ero_jalan=0;
int Dil_jalan=0;

int Ero_pointer=0;
int Dil_pointer=0;

int baris1=0,kolom1=0;
int baris2=0,kolom2=0;
int baris3=0,kolom3=0;

Mat frame;
Mat frame3;
Mat hsv2;
Mat thr2;

Mat hsv3;
Mat thr3;
Mat thr;
Mat thr4;

float sudut_akhir;

int main()
{
	HANDLE hSerial = CreateFile(L"COM3",
GENERIC_READ | GENERIC_WRITE, 0, 0,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
	if (hSerial !=INVALID_HANDLE_VALUE)
	{
		printf("Port opened! \n");
```

```
            DCB dcbSerialParams;
          GetCommState(hSerial,&dcbSerialParams);
          dcbSerialParams.BaudRate = CBR_9600;
          dcbSerialParams.ByteSize = 8;
          dcbSerialParams.Parity = NOPARITY;
          dcbSerialParams.StopBits = ONESTOPBIT;
          SetCommState(hSerial, &dcbSerialParams);
      }
      else
      {
            if(GetLastError() ==
ERROR_FILE_NOT_FOUND)
            {
                  printf("Serial port doesn't
exist! \n");
            }
            printf("Error while setting up
serial port! \n");
      }

      char outputChars[] = "c";
      DWORD btsIO;

      VideoCapture cap(1);

      //create trackbar
      namedWindow("control_jalan",CV_WINDOW_NORM
AL);
      namedWindow("control_pointer",CV_WINDOW_NO
RMAL);

      cvCreateTrackbar("HueL_jalan","control_jal
an",&HueL_jalan,255);
      cvCreateTrackbar("HueM_jalan","control_jal
an",&HueM_jalan,255);
```

```
      cvCreateTrackbar("HueL_pointer","control_p
ointer",&HueL_pointer,255);
      cvCreateTrackbar("HueM_pointer","control_p
ointer",&HueM_pointer,255);

      cvCreateTrackbar("SaturationL_jalan","cont
rol_jalan",&SaturationL_jalan,255);
      cvCreateTrackbar("SaturationM_jalan","cont
rol_jalan",&SaturationM_jalan,255);

      cvCreateTrackbar("SaturationL_pointer","co
ntrol_pointer",&SaturationL_pointer,255);
      cvCreateTrackbar("SaturationM_pointer","co
ntrol_pointer",&SaturationM_pointer,255);

      cvCreateTrackbar("ValueL_jalan","control_j
alan",&ValueL_jalan,255);
      cvCreateTrackbar("ValueM_jalan","control_j
alan",&ValueM_jalan,255);

      cvCreateTrackbar("ValueL_pointer","control
_pointer",&ValueL_pointer,255);
      cvCreateTrackbar("ValueM_pointer","control
_pointer",&ValueM_pointer,255);

      cvCreateTrackbar("Erode_jalan","control_ja
lan",&Ero_jalan,100);
      cvCreateTrackbar("Dilate_jalan","control_j
alan",&Dil_jalan,100);
      cvCreateTrackbar("BlurL_jalan","control_ja
lan",&BlurL_jalan,255);

      cvCreateTrackbar("Erode_pointer","control_
pointer",&Ero_pointer,100);
```

```
	cvCreateTrackbar("Dilate_pointer","control
_pointer",&Dil_pointer,100);
	cvCreateTrackbar("BlurL_pointer","control_
pointer",&BlurL_pointer,255);

	for(;;)
	{
		cap >> frame;

		//Crop windows

		Rect myROI(325,50,150,180);
		Mat frame2 = frame(myROI);

		Mat frame3(frame2.clone());

		Rect ROIrect(10,70,125,100);

		Mat fillROI = frame2(ROIrect);

		fillROI = Scalar(0);

		Mat inverseFill(frame3.clone());

		Mat inverseMask(inverseFill.size(),
CV_8UC1,Scalar(1));

		Mat inverseMaskROI =
inverseMask(ROIrect);

		inverseMaskROI = Scalar(0);

		inverseFill.setTo(Scalar(0),
inverseMask);
```

```
            //Jalan Biru Bintang Program


      cvtColor(frame2,hsv2,CV_BGR2HSV_FULL);


      inRange(hsv2,Scalar(HueL_jalan,SaturationL
_jalan,ValueL_jalan),Scalar(HueM_jalan,Saturatio
nM_jalan,ValueM_jalan),thr2);

      erode(thr2,thr2,getStructuringElement(MORP
H_CROSS, Size(Ero_jalan+1,Ero_jalan+1),
Point(Ero_jalan,Ero_jalan)));

      dilate(thr2,thr2,getStructuringElement(MOR
PH_CROSS,
Size(Dil_jalan+1,Dil_jalan+1),Point(Dil_jalan,Di
l_jalan)));
            GaussianBlur(thr2,thr2,
Size((BlurL_jalan+1),
(BlurL_jalan+1)),BlurM_jalan,BlurM_jalan);

            //morphological opening (remove
small objects from the foreground)
            erode(thr2, thr2,
getStructuringElement(MORPH_ELLIPSE, Size(5,5))
);
            dilate(thr2,thr2,
getStructuringElement(MORPH_ELLIPSE, Size(5,5))
);


            Canny(thr2,thr2,100,200,3);

            //Scanning Pixel
            for(int j=thr2.rows-1;j>=0;j--)
```

```
            {
                  for(int i=thr2.cols-1; i>=0;i-
-)
                  {
                        int scan1 =
thr2.at<uchar>(j,i);
                        if(scan1>253)
                        {
                              baris1=j;
                              kolom1=i;
                        }
                  }
            }

      circle(thr2,Point(kolom1,baris1),2,Scalar(
255,255,255),2,8 );

            for(int j=thr2.rows-1;j>=0;j--)
            {
                  for(int i=0;i<thr2.cols;i++)
                  {
                        int scan2 =
thr2.at<uchar>(j,i);
                        if(scan2>253)
                        {
                              baris2=j;
                              kolom2=i;
                        }
                  }
            }

      circle(thr2,Point(kolom2,baris2),2,Scalar(
255,255,255),2,8 );
```

```
      circle(thr2,Point((kolom1+kolom2)/2,(baris
1+baris2)/2),2,Scalar(255,255,255),2,8 );

            //Segitiga bintang program


      cvtColor(inverseFill,hsv3,CV_BGR2HSV_FULL)
;


      inRange(hsv3,Scalar(HueL_pointer,Saturatio
nL_pointer,ValueL_pointer),Scalar(HueM_pointer,S
aturationM_pointer,ValueM_pointer),thr3);

      erode(thr3,thr3,getStructuringElement(MORP
H_CROSS, Size(Ero_pointer+1,Ero_pointer+1),
Point(Ero_pointer, Ero_pointer)) );

      dilate(thr3,thr3,getStructuringElement(MOR
PH_CROSS,
Size(Dil_pointer+1,Dil_pointer+1),Point(Dil_poin
ter, Dil_pointer)) );
            GaussianBlur(thr3,thr3,
Size((BlurL_pointer+1),
(BlurL_pointer+1)),BlurM_pointer,BlurM_pointer);

            //morphological opening (remove
small objects from the foreground)
            erode(thr3, thr3,
getStructuringElement(MORPH_ELLIPSE, Size(5,5))
);
            dilate(thr3,thr3,
getStructuringElement(MORPH_ELLIPSE, Size(5,5))
);
```

```
            Canny(thr3,thr3,100,200,3);

            //centroid
            float sumx = 0;
            float sumy = 0;
            float num_pixel = 0;
            for(int x=0; x<thr3.cols; x++)
            {
                  for(int y=0; y<thr3.rows; y++)
                  {
                        int val =
thr3.at<uchar>(y,x);
                  if( val >= 253)
                        {
                              sumx += x;
                              sumy += y;
                              num_pixel++;
                        }
                  }
            }

            Point p(sumx/num_pixel,
sumy/num_pixel);

            Moments m = moments(thr3, false);
            Point p1(m.m10/m.m00, m.m01/m.m00);



            //scanning pixel
            for(int j=thr3.rows-1;j>=0;j--)
            {
                  for(int i=thr3.cols-1; i>=0;i-
-)
```

```
                {
                        int scan3 =
thr3.at<uchar>(j,i);
                        if(scan3>=253)
                        {
                              baris3=j;
                              kolom3=i;
                        }
                }
          }



     circle(thr4,Point(kolom3,baris3),2,Scalar(
255, 255, 255 ),2,8 );
          circle(thr4, p, 5, Scalar(128,0,0),-
1);
          line(thr4, Point(sumx/num_pixel,
sumy/num_pixel) ,Point(kolom3,baris3), Scalar(
255,255,255 ),  2, 8 );


     circle(thr2,Point(kolom3,baris3),2,Scalar(
255, 255, 255 ),2,8 );
          circle(thr2, p, 5, Scalar(128,0,0),-
1);

          //garis segitiga
          line(thr2, Point(sumx/num_pixel,
sumy/num_pixel) ,Point(kolom3,baris3), Scalar(
255,255,255 ),  2, 8 );
          line(thr3, Point(sumx/num_pixel,
sumy/num_pixel) ,Point(kolom3,baris3), Scalar(
255,255,255 ),  2, 8 );
```

```
            //Garis Jalan
            line(thr2, Point(sumx/num_pixel,
sumy/num_pixel),Point((kolom1+kolom2)/2,(baris1+
baris2)/2),Scalar( 255,255,255 ),2,8);

            //Perhitungan Sudut
            float result1;

            float x1 = (kolom3-sumx/num_pixel);
            float y1 = (baris3-sumy/num_pixel);

            result1 = atan(x1/y1) * 180/3.14;
//kiri positif, kanan negatif

            float result2;

            float x2 = ((kolom1+kolom2)/2-
sumx/num_pixel);
            float y2 = ((baris1+baris2)/2-
sumy/num_pixel);

            result2 = atan(x2/y2)*180/3.14;
//kiri positif, kanan negatif

            //kondisi 1
            if (result1>=0 && result2>=0 &&
result1>result2)
            {
                  sudut_akhir = result1-result2;
            }
            else if (result1>=0 && result2>=0 &&
result1<result2)
            {
                  sudut_akhir = (result2-
result1)*-1;
```

```
            }
            //kondisi 2
            else if (result1<=0 && result2<=0 &&
result1<result2)
            {
                  sudut_akhir = result1-result2;
            }
            else if (result1<=0 && result2<=0 &&
result1>result2)
            {
                  sudut_akhir = (result2-
result1)*-1;
            }
            //kondisi 3
            else if (result1>=0 && result2<=0)
            {
                  sudut_akhir = result1-result2;
            }
            //kondisi 4
            else if (result1<=0 && result2>=0)
            {
                  sudut_akhir = result1-result2;
            }


            //printf("%f\n",result1);
            //printf("%f\n",result2);
            printf("%f\n",sudut_akhir);
            //printf("%f %f
%f\n",result1,result2,sudut_akhir);


            //delay
            //for(int i=0; i<10000;i++)
            //{
```

```
                    //for(int i=0; i<10000;i++){}
            //}


            outputChars[0] = sudut_akhir;
            WriteFile(hSerial, outputChars,
strlen(outputChars), &btsIO, NULL);

            //imshow("original",frame);
            //imshow("original2",frame2);
            //imshow("original3",inverseMask);
            imshow("original4",frame3);
            imshow("original5",inverseFill);
            imshow("original6",thr2);
            //imshow("original7",thr3);
            imshow("original8",thr3);

            waitKey(1);
      }
      CloseHandle(hSerial);
      return 0;
}
```

Program Arduino

```
#include <SoftwareSerial.h>
SoftwareSerial portOne(10,11);
#include <LiquidCrystal.h>
int in1=10;
int in2=11;
int ena_de=6;
int in3=3;
int in4=9;
int ena_be=5;
int kecepatan=255;
int sign=0;
LiquidCrystal lcd(2, 4, 7, 8, 12, 13);
int pos = 0;
int data1,data2;

void henti (){
  analogWrite(ena_be,0);
  digitalWrite(in3,LOW);
  digitalWrite(in4,LOW);
  delay(50);
}

void lurus(){
  analogWrite(ena_de,0);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  delay(50);
  }

void kiri1(){
  analogWrite(ena_de,120);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
```

```
  delay(50);
  }

void kiri2(){
  analogWrite(ena_de,200);
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  delay(50);
  }

void kanan1(){
    analogWrite(ena_de,120);
    digitalWrite(in2,HIGH);
    digitalWrite(in1,LOW);
    delay(50);
  }

void kanan2(){
    analogWrite(ena_de,200);
    digitalWrite(in2,HIGH);
    digitalWrite(in1,LOW);
    delay(50);
  }

void mundur(){
  analogWrite(ena_be,200);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
}

void maju(){
  analogWrite(ena_be,kecepatan);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
}
```

```
void setup(){
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(ena_de,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
  pinMode(ena_be,OUTPUT);

  lcd.begin(16, 2);
  Serial.begin(9600);
  lcd.setCursor(0,0);
  lcd.print("Sudut Akhir=    ");
  lcd.setCursor(0,1);
  lcd.print("PWM=    ");
  portOne.begin(9600);

  for(int i=255;i>200;i--){
    kecepatan=i;
    delay(250);
    lcd.setCursor(5,1);
    lcd.print(i);
    maju();
  }
}

void cek(){
  while (Serial.available() > 0) {
    data1 = Serial.read();

    if(data1>60){
      data1=data1-255;
    }
    else{
      data1=data1;
```

```
}
lcd.setCursor(13,0);
lcd.print(data1);
lcd.print("    ");

if(data1>=0 && data1<=10 ){
  lurus();
  delay(10);
  //maju();
  //delay(1000);
}
else if(data1<0 && data1>=-10){
  lurus();
  delay(10);
 // maju();
  //delay(1000);
}
if(data1>10 && data1<=15 ){
  kanan1();
  delay(10);
  //maju();
  //delay(1000);
}
else if(data1<-10 && data1>=-15){
  kiri1();
  delay(10);
 // maju();
  //delay(1000);
}
if(data1>15 && data1<=53 ){
  kanan2();
  delay(10);
  //maju();
  //delay(1000);
}
```

```
    else if(data1<-15 && data1>=-53){
      kiri2();
      delay(10);
     // maju();
      //delay(1000);
    }
    }
}
void loop(){
  cek();

  }
```

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Sistem kendali yang dibangun mampu mengendalikan mobil remot kontrol dengan kondisi lurus, kiri, dan kanan sesuai nilai sudut akhir yang diperoleh dari hasil *image processing*.

6.2 Saran

Untuk meningkatkan kemampuan belok mobil diperlukan motor yang lebih responsif dikarenakan motor DC mengalami keterlambatan dalam pembacaan nilai sudut akhir yang diproses oleh Visual Studio, dalam hal ini motor servo memiliki kemampuan bergeser sejauh 1° dengan kesalahn penyimpangan 0.97°.

DAFTAR PUSTAKA

[1] Nise, Norman., 2011, "**Control System Engginering Sixth Edition**", California State University, Pomona.

[2] Anderson, James, 2014, "**Autonomous Vehicle Technology**", RAND Corporation.

[3] J. Borestein, 1996, "Sensor and Metodhs for Mobile Robot Positioning".

[4] Blank, Leland, 1982, "**Statistical Procedures for Engginering, Management, and Science International Student Edition**", Texas A&M University,

[5] Banks, J., J.S. Carson, and B.L. Nelson, 1996, "**Discrete-Event System Simulation**", Prentice Hall, New Jersey.

[6] Johnson, Curtis, 2003, "**Process Control Instrumentation Technology Seventh Edition**", University of Houston, New Jersey.

BIODATA PENULIS

Mohammad Akrom Wafaiq lahir di Kabupaten Nganjuk, 24 Maret 1993. Putra Pertama dari pasangan Dra. Djariyah dan Drs Hamim Thohari. Penulis Menyelesaikan pendidikan formal di TK Diponegoro, SDN 2 Tanjunganom, SMP Negri 1 Kediri, dan SMA Negeri 2 Jombang pada tahun 2012. Selepas itu penulis melanjutkan studi di S1-Teknik Mesin, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.

Selama menempuh pendidikan di ITS penulis mengambil konsentrasi bidang studi Manufaktur dan menjadi anggota serta Kordinator Lab Perancangan dan Pengembangan Produk. Selama kuliah penulis berusaha menjadi Aktivis baik di dalam kelas maupun di luar kelas. Di luar kelas penulis aktif berorganisasi ditingkat kampus mulai dari Badan Eksekutif Mahasiswa sampai menjadi Sarjana LKMM dengan menyelesaikan LKMM TD. Serta untuk di tingkat Nasional aktif pada mengikuti perlombaan riset ilmiah serta untuk kegiatas sosial penulis juga pernah menjadi peserta bakti kampus didaerah tertinggal di Kabupaten Mojokerto. Di dalam kelas atau di bidang akademik penulis pernah meraih medali emas Nasional pada Pekan Ilmiah Mahasiswa Nasional (PIMNAS) 27 di Universitas Diponegoro Jawa Tengah. Adapun motto hidup penulis ada *" Do the best, be good, then you will be the Winner".*