TUGAS AKHIR – TE 141599

SISTEM OTOMATISASI PELACAKAN OBJEK ASTRONOMI MENGGUNAKAN TELESKOP BERDASARKAN STELLARIUM

Afif Aulia Rahman
NRP 2210100072

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Ir. Tasripan, MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT – TE 141599

ASTRONOMICAL OBJECT TRACKING AUTOMATION SYSTEM USING TELESCOPE BASED ON STELLARIUM

Afif Aulia Rahman
NRP 2210100072

Promotor
Dr. Muhammad Rivai, ST., MT.
Ir. Tasripan, MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

SISTEM OTOMATISASI PELACAKAN OBJEK ASTRONOMI MENGGUNAKAN TELESKOP BERDASARKAN STELLARIUM

TUGAS AKHIR

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

Menyetujui:

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Muhammad Rivai, ST., MT.	Ir. Tasipan, MT.
NIP.196904261994031003	NIP. 196204181990031004

SURABAYA
JUNI, 2016

SISTEM OTOMATISASI PELACAKAN OBJEK ASTRONOMI MENGGUNAKAN TELESKOP BERDASARKAN STELLARIUM

Nama : **Afif Aulia Rahman**
Pembimbing I : **Dr. Muhammad Rivai, ST., MT.**
Pembimbing II : **Ir. Tasripan, MT.**

ABSTRAK

Stellarium merupakan software simulasi gambar 3D yang dapat menampilkan benda-benda angkasa secara detail, sehingga pengamat astronomi dapat mengetahui koordinat suatu objek angkasa melalui data yang disajikan oleh *stellarium*. Akan tetapi pencarian objek berdasarkan koordinatnya jika dilakukan pengamatan secara manual menggunakan teleskop akan membutuhkan waktu yang cukup lama. Oleh karena itu diperlukan suatu sistem yang secara otomatis dapat menggerakkan teleskop berdasarkan data-data yang diperoleh melalui *stellarium*.

Sistem tersebut dapat dibangun dengan mengintegrasikan arduino, kamera, sensor potensiometer, motor servo dan motor dc. *Stellarium* mengirimkan data ke komputer berupa koordinat objek. Kemudian di komputer koordinat tersebut dikonversi dalam bentuk derajat dan dikirim ke arduino. Dengan metode *fuzzy logic* arduino akan menggerakkan motor dc menuju objek berdasarkan data derajat dan sensor potensiometer. Kamera akan menangkap gambar *real-time* melalui teleskop dan ditampilkan ke layar monitor komputer. Kemudian komputer melakukan *image processing* yaitu *autofocus* dengan motor servo untuk memperjelas gambar dan *tracking* dengan menentukan titik tengah objek. Saat benda bergerak, kamera akan meng-*update* koordinat objek dan menggerakkan teleskop menuju koordinat baru objek tersebut. Berdasarkan pengujian, kesalahan hasil konversi sensor derajat *azimuth* rata-rata sebesar 0.6 derajat, sedangkan untuk *altitude* rata-rata sebesar 0.6 derajat.

Kata kunci: *fuzzy logic, image processing*, motor servo, motor dc

ASTRONOMICAL OBJECT TRACKING AUTOMATION SYSTEM USING TELESCOPE BASED ON STELLARIUM

Name **: Afif Aulia Rahman**
1st Advisor **: Dr. Muhammad Rivai, ST., MT.**
2nd Advisor **: Ir. Tasripan, MT.**

ABSTRACT

Stellarium is a simulation software that can display 3D images of celestial bodies in detail, so that observers can determine astronomical coordinates of a celestial object through the data presented by Stellarium. However, if object searching is observed manually, it will require considerable time. Therefore we need a system that can automatically move the telescope based on data obtained through Stellarium.

The system can be constructed by integrating arduino, camera, potentiometer sensors, servo motors and dc motors. Stellarium transmit data to a computer in the form of the object coordinates. Then in the computer, coordinate is converted in the form of degrees and sent to arduino. By using fuzzy logic method, arduino will drive the dc motor toward the object based on the degrees data and potentiometer sensor. The camera will capture real-time images through the telescope and show it to a computer screen. Then the computer perform image processing that is autofocus with servo motors to clarify the picture and tracking by determining the midpoint of the object. When the object is moving, the camera will update the coordinates of the object and move the telescope to the new coordinates of the object. Based on testing, fault degrees azimuth sensor conversion results by an average of 0.6 degrees, while the altitude by an average of 0.6 degrees.

Keywords: *fuzzy logic, image processing, servo motor, dc motor*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pesatnya perkembangan ilmu pengetahuan dan teknologi saat ini sangat membantu salah satunya dalam bidang astronomi, sehingga dikembangkanlah software yang merupakan salah satu media pembelajaran digital dalam bidang astronomi. *Stellarium* merupakan software simulasi gambar 3D yang dapat menampilkan benda-benda angkasa secara detail berikut juga data astronominya.

Selain itu, peran teleskop dalam bidang astronomi juga sangat berguna sebagai alat pengamatan dan observasi benda-benda angkasa secara langsung. Teleskop merupakan alat paling penting dalam pengamatan astronomi. Teleskop memperbesar ukuran sudut benda, dan juga kecerahannya[1]. Untuk melakukan pengamatan, terlebih dahulu harus mengetahui letak benda yang diamati. Sebagian besar teleskop yang digunakan oleh masyarakat umum belum memiliki perangkat *tracking object* jika benda bergerak.

Oleh karena itu diperlukan suatu sistem teleskop yang dapat melakukan pelacakan objek astronomi secara otomatis. *Inisialisai* sistem dengan pengambilan *database* objek dari *software stellarium* berupa koordinat benda yang diamati. Kemudian dengan koordinat tersebut arduino akan menggerakkan motor beserta teleskop menuju target. Dengan mengunakan kamera yang terpasang pada teleskop dapat dilakukan *tracking object* jika mengalami perubahan posisi.

Dengan sistem tersebut maka akan memudahkan proses pembelajaran bagi para pemula dan yang memiliki hobby astronomi. Dan juga para astronom professional akan terbantu dalam mengamati suatu objek tanpa harus melakukan perubahan teleskop secara manual.

1.2 Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini adalah:

a. Bagaimana pengambilan koordinat pada stellarium
b. Bagaimana mengatur pergerakan motor berdasarkan data sensor
c. Bagaimana mengatur *autofocus* teleskop
d. Bagaimana melacak objek saat bergerak

1.3 Tujuan

Penelitian pada tugas akhir ini bertujuan untuk mendesain sistem yang secara otomatis dapat melacak pergerakan objek astronomi melalui teleskop berdasarkan koordinat yang diperoleh melalui software *stellarium*.

1.4 Batasan Masalah

Dalam pengerjaan tugas akhir, permasalahan di atas dibatasi dengan asumsi sebagai berikut :

a. Objek benda langit yang akan diamati tidak terhalang oleh benda lain

1.5 Metodologi

Langkah-langkah yang dikerjakan pada Tugas Akhir ini adalah sebagai berikut :

a. Studi Literatur.
   - Mempelajari sistem koordinat benda langit.
   - Mempelajari pemrograman menggunakan *Arduino* dan *Visual Studio*.

b. Perencanaan dan pembuatan perangkat keras
   - Menyediakan mikrokontroler *Arduino*.
   - Merancang rangkaian pendukung untuk pergerakan teleskop.

c. Perencanaan dan pembuatan perangkat lunak, meliputi :
   - Perencanaan dan pembuatan program untuk sensor derajat pada *Arduino*.
   - Perencanaan dan pembuatan program fuzzy logic untuk mengatur kecepatan motor dc pada teleskop.
   - Perencanaan dan pembuatan program *tracking* dan *autofocus* teleskop pada *visual studio*.

d. Integrasi perangkat keras dengan perangkat lunak
   - Penggabungan sistem hardware yang mengambil data dari pembacaan sensor dan software yang digunakan untuk mengatur gerakan teleskop.

e. Pengujian dan analisa
   - Pada tahap ini dilakukan pengujian terhadap sistem yang telah dibuat. Dengan parameter keberhasilannya adalah dapat melakukan *tracking* secara otomatis.
   - Apabila hasil yang diperoleh dari tahap pengujian terjadi ketidaksesuaian pada alat yang telah dibuat atau mungkin kurang memuaskan secara bentuk dan ketelitian penyelesaian, maka perlu diadakan evaluasi pada software dan hardware serta sistem secara keseluruhan. Kemudian dilakukan pengujian ulang sampai parameter keberhasilan telah dicapai.

f. Penulisan buku Tugas Akhir

1.6 Sistematika Penulisan

Dalam buku Tugas Akhir ini, pembahasan mengenai sistem yang dibuat terbagi lima bab dengan sistematika sebagai berikut:

❖ BAB I : PENDAHULUAN

Pada bagian ini menjelaskan beberapa sub bagian yang antara lain berisi Latar Belakang, Permasalahan, Tujuan, Batasan Masalah, Metodologi, Sistematika Penulisan, dan Relevansi penulisan Tugas Akhir ini.

❖ BAB II : TEORI PENUNJANG

Pada bagian ini berisi tentang landasan teori yang digunakan dalam pelaksanaan Tugas akhir yang meliputi Teleskop, Sistem Koordinat Langit, *Stellarium*, Mikrokontroller Arduino, Fuzzy Logic, Motor DC, Motor Servo, Kamera USB, *Image Processing*. Bagian ini memaparkan tentang beberapa teori penunjang dan beberapa literatur yang berguna bagi pembuatan Tugas Akhir ini.

❖ BAB III : PERANCANGAN SISTEM

Pada bagian ini akan menjelaskan tentang disain sistem baik *hardware* maupun *software* dan tiap-tiap modul yang digunakan untuk menyusun sistem secara keseluruhan.

❖ BAB IV: PENGUJIAN ALAT

Pada bagian 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 pengembangannya.

1.7 Relevansi

Pembuatan Sistem Otomatisasi Pelacakan Objek Astronomi Menggunakan Teleskop Berdasarkan Stellarium merupakan salah satu langkah pengembangan dari teleskop otomatis dalam bidang astronomi, sehingga dapat membantu perkembangan ilmu astronomi dalam kemampuan teleskop untuk menemukan dan mengikuti pergerakan benda langit secara otomatis.

BAB II
TEORI PENUNJANG

2.1 Teleskop

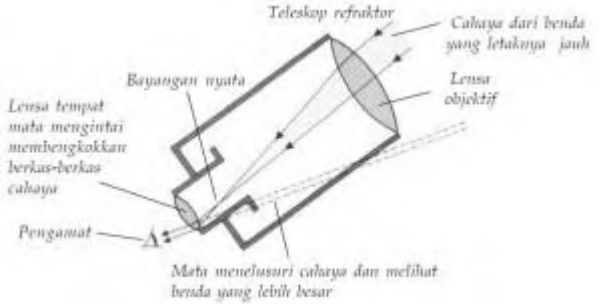
Teleskop atau teropong adalah sebuah instrumen pengamatan yang berfungsi mengumpulkan radiasi elektromagnetik dan sekaligus membentuk citra dari benda yang diamati. Teleskop merupakan alat paling penting dalam pengamatan astronomi. Teleskop merupakan jenis peralatan yang digunakan untuk membantu penginderaan jauh guna mengamati keberadaan benda-benda yang ada di angkasa. Dengan demikian, kita bisa melihat posisi sebuah benda di angkasa yang tidak bisa dilihat dengan menggunakan mata langsung[1].

Galileo diakui menjadi yang pertama dalam menggunakan teleskop untuk maksud astronomis. Galileo, walaupun bukan penemu teleskop, dia mengembangkan teleskop menjadi instrumen yang penting dan dapat digunakan. Galileo merupakan orang pertama yang meneliti ruang angkasa dengan teleskop, dan ia membuat penemuan-penemuan yang mengguncang dunia, di antaranya satelit-satelit Jupiter, fase Venus, bercak matahari, struktur permukaan bulan, dan bahwa galaksi Bimasakti terdiri dari sejumlah besar bintang-bintang individu[1].

Dalam pengertian teropong bintang juga dijelaskan bahwa teropong ini menggunakan dua buah lensa positif. Dimana masing-masing lensa berfungsi sebagai lensa obyektif dan lensa okuler. Inilah yang membedakan antara teropong bintang dengan mikroskop. Pada teropong bintang, jarak fokus lensa obyektif lebih besar daripada jarak fokus lensa okuler. Secara garis besar, teropong sendiri dibagi ke dalam dua kategori besar. Yang pertama adalah teropong bias, yaitu jenis teropong yang tersusun dari beberapa lensa. Sedangkan jenis kedua adalah teropong pantul, yaitu jenis teropong yang disusun dari beberapa cermin serta lensa.

2.1.1 Teleskop Pembias (Keplerian)

Teropong bias terdiri atas dua lensa cembung, yaitu sebagai lensa objektif dan okuler. Sinar yang masuk ke dalam teropong dibiaskan oleh lensa. Oleh karena itu, teropong ini disebut teropong bias. Teleskop pembias terdiri dari dua lensa konvergen (lensa cembung) yang berada pada ujung-ujung berlawanan dari tabung yang panjang, seperti diilustrasikan pada **Gambar 2.1**.



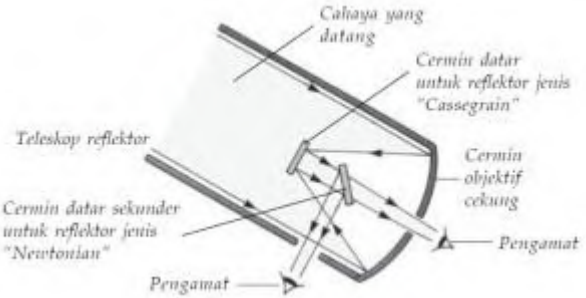
Gambar 2.1 Pembentukan bayangan pada teropong bias[16].

Cahaya yang diterima oleh lensa objektif kemudian akan difokuskan ke satu titik fokus, selanjutnya cahaya tersebut akan ditangkap oleh lensa okulernya untuk kemudian diteruskan ke detektor. Teleskop ini baik untuk digunakan pada saat mengamati cahaya bintang yang redup dari Bumi. Namun teleskop ini memiliki beberapa kelemahan antara lain[1]:

1. Terjadinya abrasi kromatis pada lensa objektifnya, yaitu komponen cahaya bintang memiliki panjang fokus yang berbeda-beda. Hal ini mengakibatkan cahaya bintang menjadi agak kabur. Namun kelemahan ini dapat diminimalisir dengan menggunakan dua lensa (cekung dan cembung) yang memiliki indeks bias yang berbeda menjadi lensa objektifnya.
2. Lensa adalah benda yang terbuat dari cairan yang sebenarnya terlambat membeku. Ketika teleskop ditegakkan, maka bagian lensa yang paling atas akan mengalir ke arah bawah secara perlahan - lahan. Hal ini menyebabkan mutu bayangan menjadi kurang bagus.
3. Sulitnya membuat lensa dengan diameter yang besar. Karena semakin besar diameter lensa, maka akan semakin sulit untuk membuat lensa yang homogen, sehingga cahaya bintang tidak terdistorsi oleh turbulensi atmosfer.

Benda yang diamati terletak di titik jauh tak hingga, sehingga bayangan yang dibentuk oleh lensa objektif tepat berada pada titik fokusnya. Bayangan yang dibentuk lensa objektif merupakan benda bagi lensa okuler. Lensa okuler berfungsi sebagai lup.

2.1.2 Teleskop Pemantul (Newtonian)



Gambar 2.2 Pembentukan bayangan pada teropong pantul[16]

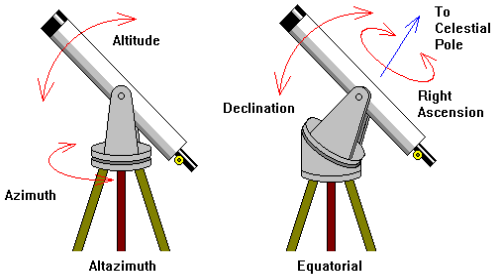
Karena jalannya sinar di dalam teropong dengan cara memantul maka teropong ini dinamakan teropong pantul. **Gambar 2.2** menunjukkan bagaimana cara teropong pantul bekerja. Pada teropong pantul, cahaya yang datang dikumpulkan oleh sebuah cermin melengkung yang besar. Cahaya tersebut kemudian dipantulkan ke mata pengamat oleh satu atau lebih cermin yang lebih kecil.

Namun teleskop ini memiliki beberapa kelemahan antara lain[1]:

1. Kelemahan dari teleskop pantul ini adalah, bayangan yang dihasilkan tidak terlalu tajam karena berasal dari proses pemantulan, tidak seperti pada teleskop bias yang bayangannya berasal dari pembiasan.
2. Terjadinya abrasi sferis, yaitu cahaya yang datang di samping teleskop, tidak sama dengan cahaya yang datang di tengah teleskop. Ini membuat mutu bayangan menjadi sedikit terganggu.

Agar teleskop astronomi menghasilkan bayangan yang terang dari bintang-bintang yang jauh, cermin objektif harus besar untuk memungkinkan cahaya masuk sebanyak mungkin. Dan memang, diameter objektif merupakan parameter yang paling penting untuk teleskop astronomi, yang merupakan alasan mengapa teleskop yang paling besar dispesifikasikan dengan menyebutkan diameter objektifnya.

2.1.3 *Mounting* teleskop



Gambar 2.3 Jenis *mounting* pada teleskop[17]

Mounting atau yang lebih familiar dikenal dengan "dudukan teleskop" terbagi dalam 2 jenis yaitu jenis mounting equatorial dan jenis mounting altazimuth [1]. **Gambar 2.3** menunjukkan peletakan setiap jenis *mounting* teleskop. Mounting Equatorial bekerja menggunakan 3 buah sumbu yaitu sumbu RA, Deklinasi dan Equator. Sedang mounting altazimuth menggunakan 2 buah sumbu yaitu sumbu X atau altitude (atas bawah) dan Y atau azimuth (kanan kiri).

Setiap jenis *mounting* memiliki kelebihan dan kekurangannya masing-masing. Untuk jenis altazimuth pembuatan *mounting* nya jauh lebih mudah dibanding mounting equatorial dan lebih murah, tetapi kekurangannya teleskop harus diatur berdasarkan kedua sumbu azimuth (kiri dan kanan) serta sumbu ketinggian (atas dan bawah) untuk mengikuti bintang saat bumi berputar. Sedangkan untuk jenis equatorial karena satu sumbu putarnya (*polar axis*) sejajar dengan sumbu putar bumi, maka teleskop hanya perlu diatur berdasarkan sumbu putar ini. Tetapi kekurangannya jenis *mounting* ini lebih sulit dan mahal dalam pembuatannya.

2.2 Sistem Koordinat Langit

Untuk menyatakan letak suatu benda langit diperlukan suatu tata koordinat yang dapat menyatakan secara pasti kedudukan benda langit tersebut. Tata koordinat tersebut terdiri dari tata koordinat horison, tata koordinat ekuator, tata koordinat ekliptika dan tata koordinat galaktik.

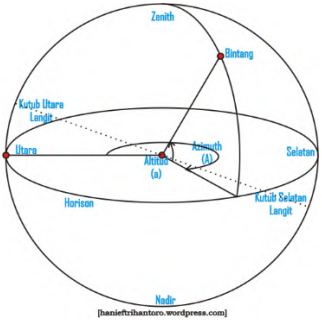
Tata koordinat horison dan tata koordinat ekuator merupakan sistem koordinat yang paling sering digunakan dalam astronomi[3].

Tiap-tiap tata koordinat tentunya memiliki cara penggunaan sistem yang berbeda serta terdapatnya berbagai macam keuntungan dan kelemahan dalam penggunaan sistem tersebut. Dengan demikian penggunaan suatu sistem koordinat bergantung pada hasil yang diinginkan, apakah hasil yang didapat ingin digunakan untuk waktu sesaat atau untuk waktu yang lama dan dapat dipakai secara universal.

2.2.1 Koordinat Horison

koordinat ini adalah tata koordinat yang paling sederhana dan paling mudah dipahami. Tetapi tata koordinat ini sangat terbatas, yaitu hanya dapat menyatakan posisi benda langit pada satu saat tertentu, untuk saat yang berbeda tata koordinat ini tidak dapat memberikan hubungan yang mudah dengan posisi benda langit sebelumnya. Karena itu menyatakan saat benda langit pada posisi itu sangat diperlukan dan tata koordinat lain diperlukan agar dapat memberikan hubungan dengan posisi sebelum dan sesudahnya[3].

Bola langit dapat dibagi menjadi dua bagian sama besar oleh satu bidang yang melalui pusat bola itu, menjadi bagian atas dan bagian bawah. Bidang itu adalah bidang horizontal yang membentuk lingkaran horison pada permukaan bola, dan bagian atas adalah letak benda-benda langit yang tampak, dan bagian bawahnya adalah letak dari benda-benda langit yang tidak terlihat saat itu. Horison adalah bidang datar yang menjadi pijakan pengamat, yang menjadi batas antara belahan langit yang dapat diamati dengan yang tidak dapat diamati.



Gambar 2.4 Tata koordinat horizon[18]

Ordinat-ordinat dalam tata koordinat horizon adalah[3]:

4. Bujur suatu bintang dinyatakan dengan azimut (Az). Azimut umumnya diukur dari utara ke arah timur sampai pada proyeksi bintang itu di horizon. Besarnya azimuth adalah dari 0 derajat hingga 360 derajat.
5. Lintang suatu bintang dinyatakan dengan altitude (alt), yang diukur dari proyeksi bintang di horizon ke arah bintang itu menuju ke zenit. Tinggi bintang diukur 0° – 90° jika arahnya ke atas (menuju zenit) dan 0° – -90° jika arahnya ke bawah. Apabila sebuah bintang baru terbit atau tenggelam, ketinggiannya dari horison adalah 0 derajat. Dan bintang yang berada di zenith memiliki altitud 90 derajat.

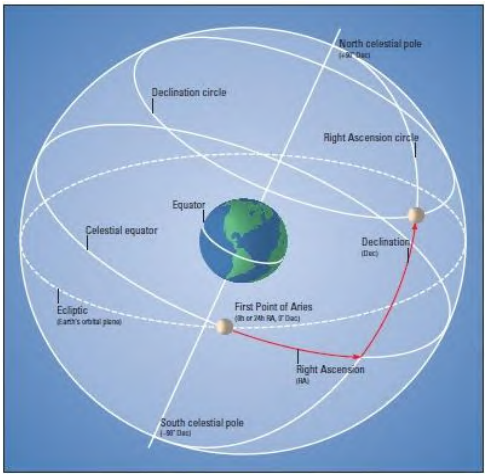
Apabila sebuah bintang baru terbit atau tenggelam, ketinggiannya dari horison adalah 0 derajat. Dan bintang yang berada di zenith memiliki altitud 90 derajat. **Gambar 2.4** menunjukkan ordinat dalam koordinat horizon.

2.2.2 Koordinat Ekuator

Tata koordinat ini merupakan salah satu tata koordinat yang sering digunakan dalam astronomi. Sistem koordinat ini dapat menyatakan letak benda langit dalam skala waktu relatif panjang. Sekalipun perubahan unsur-unsur koordinatnya relatif kecil terhadap waktu[3].

Dalam setiap pembahasan sistem koordinat benda langit, setiap benda langit selalu dipandang terproyeksi pada suatu bidang bola khayal yang digambarkan sebagai bola langit. Bola yang memuat bidang khayal tersebut disebut bola langit. Ukuran bola Bumi diabaikan terhadap bola langit sehingga setiap pengamat di muka Bumi dianggap berada di pusat bola langit.

Pada koordinat ekuatorial terdapat pula ekuator langit, kutub langit, dan belahan utara serta selatan. Ekuator langit merupakan perpanjangan dari ekuator bumi, hingga menyentuh bola langit. Sedangkan kutub-kutub langit juga sejajar dengan kutub-kutub yang dimiliki oleh bumi. Garis lintang dan garis bujur pada koordinat ekuatorial adalah Deklinasi (Declination (Dec)) dan Aksensiorekta (Right Ascension (RA)). **Gambar 2.5** menunjukkan ordinat dalam koordinat ekuator.



Gambar 2.5 Tata koordinat equator[19]

Deklinasi memiliki prinsip yang identik dengan garis lintang. Nilai dari deklinasi diperoleh dengan menghitung besarnya sudut yang berpangkal pada ekuator langit. Bermula dari ekuator, nilai deklinasi berkisar antara 0 derajat hingga 90 derajat pada kutub-kutub langit. Jika benda langit berada di belahan utara, maka deklimasi akan bernilai positif, dan sebaliknya jika berada pada belahan selatan maka akan bernilai negatif. Untuk penulisannya, walaupun bernilai positif, tanda (+) tetap disertakan. Misalnya benda langit yang berada pada kutub utara langit, maka deklinasinya adalah +90 derajat begitu pula dengan benda yang berada pada kutub selatan langit akan bernilai -90 derajat[4].

Aksensiorekta yang identik dengan garis bujur juga memiliki prinsip penghitungan yang tidak jauh berbeda. Jika nilai pada garis bujur berpangkal pada titik di Greenwich London, maka nilai aksensiorekta berpangkal dari titik Aries atau Vernal Equinox. Titik vernal equinox merupakan perpotongan antara ekuator langit dengan ekliptika, sehingga pada saat itu di khatulistiwa, matahari benar-benar tebit dari timur dan tenggelam tepat di barat. Berpangkal pada titik vernal equinox tersebut, nilai aksensiorekta dihitung berlawanan dengan arah jam (ke arah timur), dengan satuan jam atau derajat. Aksensiorekta bernilai 0 derajat hingga 360 derajat atau 0-24jam dengan tiap jam mewakili 15 derajat**[**4].

2.3 *Stellarium*

Stellarium merupakan proyek software "Open Source" yang mencoba membuat sebuah komputer menjadi planetarium virtual. Stellarium berlisensi GNU General Public License. Dengan lisensi tersebut pengguna bebas memakai, saling berbagi copy software, bahkan memodifikasinya karena disediakannya kode programnya. **Gambar 2.6** menunjukkan tampilan *stellarium.*

Selain berjalan di Linux, Stellarium juga dapat diinstall di Windows dan Macintosh. Salah satu keunggulan lain Stellarium adalah dapat berjalan di spesifikasi komputer yang relatif rendah.

Stellarium dapat menghitung koordinat benda-benda langit dan menampilkannya pada posisi pandangan virtual yang sama seperti saat seorang pengamat melihat pemandangan di langit. Karena itu Stellarium sangat cocok sebagai sarana edukasi untuk mengajarkan astronomi.

Selain itu Stellarium dapat difungsikan sebagai pengendali pada teleskop yang dilengkapi sistem kontrol rotator sehingga menemukan sebuah benda langit akan jauh lebih mudah.

Antar muka Stellarium adalah pemandangan sebuah lokasi pengamatan langit yang biasanya sebuah padang rumput dengan hutan disekelilingnya atau lahan pertanian yang luas. Pemandangan lokasi ini bisa kita ubah-ubah, bahkan kita bisa membuat pemandangan lokasi daerah pengamatan hasil olahan sendiri.

Gambar 2.6 Tampilan *Stellarium*

Ada banyak fitur yang terdapat dalam Stellarium. Untuk fitur langit, terdapat lebih dari 600 ribu bintang yang terdapat dalam katalog Hipparcos dan katalog Tycho-2. Hipparcos merupakan kepanjangan dari High Precision Parallax Collecting Satellite (Satelit Pengkoleksi Paralak Presisi Tinggi), sebuah proyek dari Badan Antariksa Eropa (European Space Agency – ESA) untuk pengukuran paralaks dan gerak diri bintang. Sedangkan Tycho-2 merupakan bagian dari satelit Hipparcos yang mengkoleksi bintang dengan tingkat presisi yang rendah[5].

Fitur langit lainnya adalah tambahan katalog dari 210 juta lebih bintang. Selain itu, ada juga asteroid dan ilustrasi rasi bintang dari 10 kebudayaan dunia, nebula dari katalog Messier, *Milky Way*, dilengkapi juga dengan atmosfer, sunrise, dan sunset yang nyata, serta planet-planet dalam tata surya beserta satelit-satelitnya.

Selain itu, ada juga fitur visualisasi berupa garis khatulistiwa, garis bujur, serta garis lintang, bintang berkedip, bintang jatuh, simulasi gerhana dan landscape atau dataran tempat pengamat berada.

Sedangkan Pengatur Dasar (Basic Control) memiliki fungsi untuk perbesaran (zoom), mengamati sekeliling dengan perputaran 360 derajat, pengaturan waktu yang bisa memajukan, memundurkan, dan mempercepat waktu, dan menyimpan lokasi pengamatan.

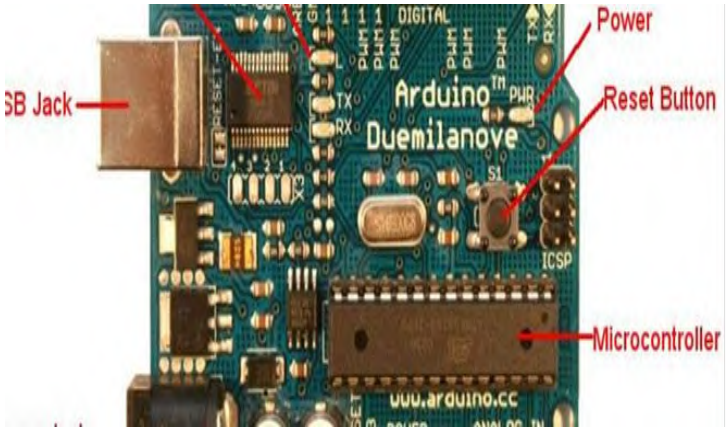
2.4 Mikrokontroler Arduino

Secara sederhana, mikrokontroler adalah chip yang menjadi otak dari sebuah rangkaian elektronika. Pada rangkaian mikrokontroler Arduino, chip ATmega328 adalah otaknya, yang mengatur komponen lain yang terhubung dengan nya (misal led, sensor2, motor).

Arduino sebenarnya adalah sebuah platform. Platform ini diciptakan untuk menyederhanakan proses rangkaian dan pemrograman mikrokontroler sehingga menjadi lebih mudah dipelajari[6].

Platform ini disusun pada sebuah software yang diberi nama Arduino IDE. Software inilah yang paling utama membantu menjembatani antara bahasa mesin yang begitu rumit sehingga menjadi bahasa logic yang lebih mudah dimengerti manusia. **Gambar 2.7** menunjukkan pin-pin pada arduino.

Board Arduino didesain untuk mempermudah pemrograman dan koneksi chip ATmega328 dengan komponen lainnya. Dengan board Arduino, baik itu Arduino UNO, Arduino Mega 2560, Arduino Nano, maupun Arduino Pro Mini, semuanya membantu untuk menyederhanakan proses membuat rangkaian mikrokontroler.



Gambar 2.7 Pin-pin Arduino[20]

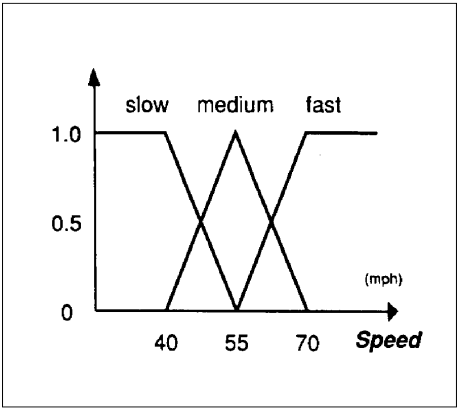
2.5 *Fuzzy Logic*

Logika fuzzy adalah peningkatan dari logika Boolean yang berhadapan dengan konsep kebenaran sebagian. Saat logika klasik menyatakan bahwa segala hal dapat diekspresikan dalam istilah biner (0 atau 1), logika fuzzy menggantikan kebeneran Boolean dengan tingkat kebenaran[7].

Logika fuzzy memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistik, konsep tidak pasti seperti ‘sedikit’, ‘lumayan’, dan ‘sangat’. Logika ini berhubungan dengan himpunan fuzzy dan teori kemungkinan.

Logika fuzzy dan logika probabilitas secara matematis sama, keduanya mempunyai nilai kebenaran yang berkisar antara 0 dan 1, namun secara konsep berbeda. Logika fuzzy berbicara mengenai ‘derajat kebenaran’, sedangkan logika probabilitas mengenai ‘probabilitas, kecenderungan’. Karena kedua hal itu berbeda, logika fuzzy dan logika probabilitas mempunyai contoh penerapan dalam dunia nyata yang berbeda.

Fungsi keanggotaan dari himpunan fuzzy dinyatakan dengan derajat keanggotaan suatu nilai dengan nilai tegasnya yang berkisa antara 0,0 sampai dengan 1,0. Jika A: himpunan fuzzy, µA : fungsi keanggotaan dan X : semesta, maka fungsi keanggotaan dalam suatu himpunan fuzzy dapat dinyatakan pada persamaan 2.1



Gambar 2.8 Contoh logika fuzzy

$$A = \{(x, \mu A(x)) \mid x \epsilon X\} \tag{2.1}$$

Gambar 2.8 menunjukkan contoh himpunan logika fuzzy. Fungsi keanggotaan suatu himpunan fuzzy dapat ditentukan dengan fungsi segitiga atau trapesium.

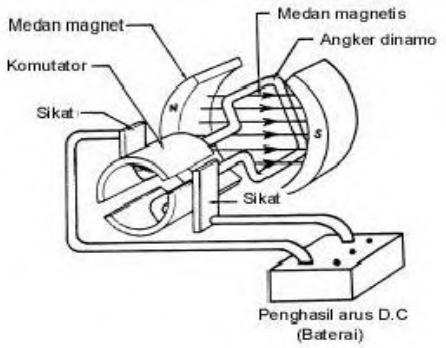
Fungsi segitiga:

$$\mu x = \begin{cases} 0, for\ x < a \\ \dfrac{x-a}{b-a}, for\ a \le x \le b \\ \dfrac{c-x}{c-b}, for\ b \le\ x \le c \\ 0, for\ x > c \end{cases} \tag{2.2}$$

Fungsi trapesium:

$$\mu x = \begin{cases} 0, for\ x < a \\ \dfrac{x-a}{b-a}, for\ a \le x \le b \\ 1, for\ b \le x \le c \\ \dfrac{d-x}{d-c}, for\ c \le\ x \le d \\ 0, for\ d \le x \end{cases} \tag{2.3}$$

2.6 Motor DC



Gambar 2.9 Motor DC

Motor DC adalah motor listrik yang memerlukan suplai tegangan arus searah pada kumparan medan untuk diubah menjadi energi gerak mekanik. Kumparan medan pada motor dc disebut stator (bagian yang tidak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar). Motor arus searah, sebagaimana namanya, menggunakan arus langsung yang tidak langsung/direct-unidirectional. Motor DC memiliki 3 bagian atau komponen utama untuk dapat berputar sebagai berikut.

Komponen Utama Motor DC[8]:

1. Kutub medan. Motor DC sederhana memiliki dua kutub medan, kutub utara dan kutub selatan. Garis magnetik energi membesar melintasi ruang terbuka diantara kutub-kutub dari utara ke selatan. Untuk motor yang lebih besar atau lebih komplek terdapat satu atau lebih elektromagnet.
2. Current Elektromagnet atau Dinamo. Dinamo yang berbentuk silinder, dihubungkan ke as penggerak untuk menggerakan beban. Untuk kasus motor DC yang kecil, dinamo berputar dalam medan magnet yang dibentuk oleh kutub-kutub, sampai kutub utara dan selatan magnet berganti lokasi.
3. Commutator. Komponen ini terutama ditemukan dalam motor DC. Kegunaannya adalah untuk transmisi arus antara dinamo dan sumber daya.

Bagian utama motor DC adalah stator dan rotor dimana kumparan medan pada motor dc disebut stator (bagian yang tidak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar)[8]. Bentuk motor paling sederhana memiliki kumparan satu lilitan yang bisa berputar bebas di antara kutub-kutub magnet permanen. Catu tegangan dc dari baterai menuju ke lilitan melalui sikat yang menyentuh komutator, dua segmen yang terhubung dengan dua ujung lilitan. Kumparan satu lilitan pada **Gambar 2.9** disebut angker dinamo. Angker dinamo adalah sebutan untuk komponen yang berputar di antara medan magnet.

2.7 Motor Servo

Motor servo adalah sebuah motor DC dengan sistem umpan balik tertutup di mana posisi rotor-nya akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri dari sebuah motor DC, serangkaian gear, potensiometer, dan rangkaian kontrol. Potensiometer berfungsi untuk menentukan batas sudut dari putaran servo. Sedangkan sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor servo.

Ada dua jenis motor servo, yaitu motor servo AC dan DC. Motor servo AC lebih dapat menangani arus yang tinggi atau beban berat, sehingga sering diaplikasikan pada mesin-mesin industri. Sedangkan motor servo DC biasanya lebih cocok untuk digunakan pada aplikasi-aplikasi yang lebih kecil. Dan bila dibedakan menurut rotasinya, umumnya terdapat dua jenis motor servo yang dan terdapat di pasaran, yaitu motor servo rotation 180^0 dan servo rotation continuous[9].

Gambar 2.10 Motor Servo

1. Motor servo standard (servo rotation 180^0) adalah jenis yang paling umum dari motor servo, dimana putaran poros outputnya terbatas hanya 90^0 kearah kanan dan 90^0 kearah kiri. Dengan kata lain total putarannya hanya setengah lingkaran atau 180^0.
2. Motor servo rotation continuous merupakan jenis motor servo yang sebenarnya sama dengan jenis servo standard, hanya saja perputaran porosnya tanpa batasan atau dengan kata lain dapat berputar terus, baik ke arah kanan maupun kiri.

Motor servo dikendalikan dengan mengirimkan pulsa melalui kabel control dengan variable lebar pulsa terkirim atau biasa disebut "Pulse Width Modulation (PWM)".

Ada minimum lebar pulsa dan maksimum lebar pulsa dan tingkat perulangan. Sebuah motor servo biasanya hanya dapat mengubah 90° di kedua arah untuk total 180° gerakan. Posisi netral motor didefinisikan sebagai posisi di mana servo memiliki jumlah yang sama dari potensi rotasi di kedua searah jarum jam atau berlawanan arah jarum jam arah. PWM yang dikirim ke motor menentukan posisi poros, dan berdasarkan durasi dari pulsa yang dikirim melalui kabel kontrol rotor akan berubah ke posisi yang diinginkan[10]. Gambar motor servo ditunjukkan pada **Gambar 2.10.**

2.8 Kamera *USB*

Kamera *usb* adalah sebutan kamera real-time (bermakna keadaan pada saat ini juga) yang gambarnya dapat di lihat secara langsung dan dapat dihubungkan ke komputer biasanya melalui *port usb* yang mendukung fasilitas PnP (Plug and Play). Pada umumnya kamera *usb* tidak membutuhkan kaset atau tempat penyimpanan data, data hasil perekaman yang didapat langsung ditransfer ke komputer.

Kamera *usb* bekerja seperti halnya sebuah kamera digital, hanya saja kamera ini didesain untuk komputer jadi tidak bisa di bawa ke mana-mana dan jauh lebih simple di banding kamera-kamera pada umumnya. **Gambar 2.11** merupakan contoh kamera usb.

Sebuah kamera *usb* sederhana terdiri dari[11]:

1. Sebuah lensa standar, dipasang di sebuah papan sirkuit untuk menangkap sinyal gambar.
2. Casing (cover), termasuk casing depan dan casing samping untuk menutupi lensa standar dan memiliki sebuah lubang lensa di casing depan yang berguna untuk memasukkan gambar.

Gambar 2.11 Kamera USB

3. Kabel, yang dibuat dari bahan yang fleksibel, salah satu ujungnya dihubungkan dengan papan sirkuit dan ujung satu lagi memiliki connector, kabel ini dikontrol untuk menyesuaikan ketinggian, arah dan sudut pandang web camera.
4. Sebuah kamera *usb* biasanya dilengkapi dengan software, software ini mengambil gambar-gambar dari kamera digital secara terus menerus ataupun dalam interval waktu tertentu dan menampilkannya ke layar monitor komputer.

2.9 *Image Processing*

Pengolahan gambar digital atau Digital Image Processing adalah Ilmu tentang proses memanipulasi dan mengolah informasi dari sebuah gambar. Suatu bidang yang berkembang sangat pesat sejalan dengan kemajuan teknologi pada industri saat ini. Perkembangan yang pesat seiring dengan munculnya teknologi komputer yang sanggup memenuhi suatu kecepatan proses dan kapasitas memori yang dibutuhkan oleh berbagai algoritma pengolahan citra. Sejak itu berbagai aplikasi mulai dikembangkan, yang secara umum dapat dikelompokkan ke dalam 2 bagian[12]:

1. Untuk memperbaiki kualitas dari gambar sehingga gambar dapat dilihat lebih jelas tanpa ada ketegangan pada mata, karena informasi penting diekstrak dari gambar yang dihasilkan harus jelas sehingga didapatkan hasil yang terbaik.

2. Mengolah informasi yang terdapat pada gambar (citra) untuk keperluan pengenalan suatu objek secara otomatis oleh suatu mesin. Bidang ini sangat erat hubungannya dengan ilmu pengenalan pola (pattern recognition), yang secara umum bertujuan untuk mengenali suatu objek dengan cara mengektraksi informasi penting yang terdapat dalam suatu image/citra.

2.9.1 Metode Konvolusi

Konvolusi citra adalah tehnik untuk menghaluskan suatu citra atau memperjelas citra dengan menggantikan nilai piksel dengan sejumlah nilai piksel yang sesuai atau berdekatan dengan piksel aslinya. Dengan adanya konvolusi, ukuran dari citra tetap sama, tidak berubah[13].

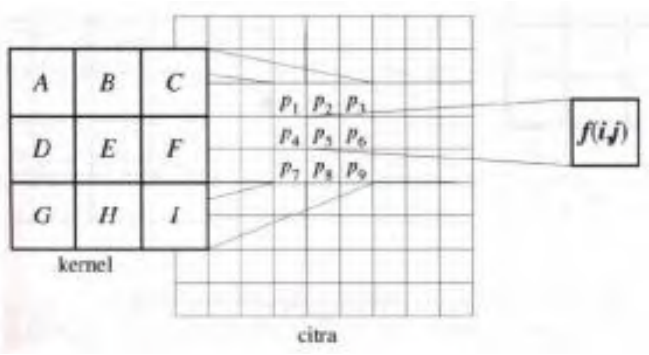
Konvolusi memiliki dua buah fungsi f(x) dan g(x) yang didefinisikan sebagai berikut:

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a) \, . \, g(x-a) da \qquad (2.4)$$

Untuk pengolahan citra, operasi yang dilakukan adalah diskrit karena nilai koordinat piksel merupakan nilai yang diskrit. Selanjutnya filter atau mask yang digunakan pada pengolahan citra biasanya berukuran terbatas, dalam artian bobot atau pengaruh dari titik-titik yang cukup jauh sudah tidak signifikan, sehingga dapat diabaikan (dianggap nol)[13].

Untuk fungsi dengan dua dimensi, operasi konvolusi didefinisikan sebagai berikut:
untuk fungsi diskrit:

$$h(x,y) = f(x,y) * g(x,y) = \sum_{a=-\infty}^{\infty} \sum_{a=-\infty}^{\infty} f(a,b) . g(x-a, y-b) \qquad (2.5)$$



Gambar 2.12 Ilustrasi konvolusi[13]

Fungsi penapis g(x,y) disebut juga konvolusi filter, konvolusi mask, konvolusi kernel, atau template. Dalam bentuk diskrit konvolusi kernel dinyatakan dalam bentuk matriks (umumnya matriks 3x3). Ukuran matriks ini biasanya lebih kecil dari ukuran citra. Setiap elemen matriks disebut koefisien konvolusi[13]. **Gambar 2.12** merupakan contoh ilustrasi metode konvolusi.

2.9.2 *Hough Transform*

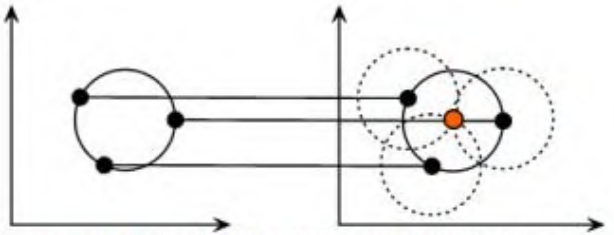
Hough Transform merupakan suatu teknik untuk menentukan lokasi suatu bentuk dalam citra. Dalam implementasinya, *Hough Transform* melakukan pemetaan terhadap titik-titik pada citra ke dalam parameter space berdasarkan suatu fungsi yang mendefinisikan bentuk yang ingin dideteksi. *Hough Transform* umumnya dugunakan untuk melakukan ekstrasi garis, lingkaran atau elips pada citra, namun dalam perkembangnnya, *Hough Transform* juga telah dapat dgunakan untuk melakukan ekstraksi bentuk-bentuk yang lebih kompleks[14].

Jika suatu gambar mengandung banyak titik, beberapa yang jatuh diperimeter lingkaran, maka tugas program pencarian untuk menemukan kembar tiga parameter (a,b,R) untuk menggambarkan lingkaran masing-masing.

Jika lingkaran dalam sebuah gambar R jari-jari diketahui, dan koordinat pusat adalah (a, b) maka persamaan titik anggota lingkaran dapat ditentukan dengan persamaan berikut pada ruang geometrik.

$$x = a + R \cos(\theta) \quad (2.6)$$
$$y = b + R \sin(\theta) \quad (2.7)$$

Gambar 2.13 menunjukkan proses *hough transform*. Lokus (a, b) poin di ruang parametrik kecepatan parameter pada jari-jari lingkaran R berpusat di (x, y). Titik pusat sejati akan umum untuk semua kalangan parameter, dan dapat ditemukan dengan array akumulasi Hough[14].

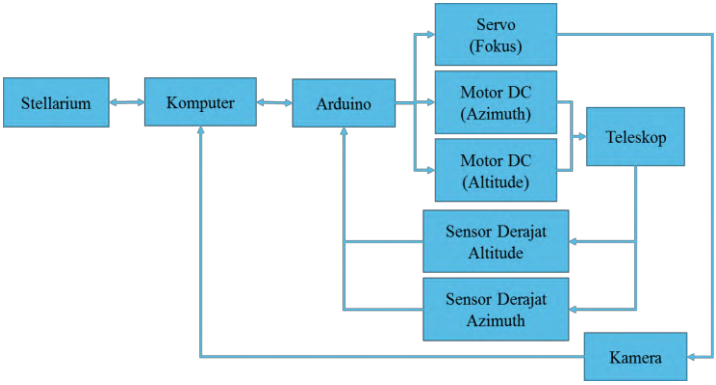


Gambar 2.13 Ruang geometri dan ruang parameter lingkaran[14]

BAB III
PERANCANGAN SISTEM

Pada bab ini dibahas tentang perancangan sistem secara keseluruhan dimulai dari penjelasan skenario kerja sistem yang sesuai dengan tujuan dari tugas akhir sampai pada perancangan perangkat keras dan perangkat lunak dari sistem.

Pada tugas akhir ini sistem merupakan teleskop *reflector* (*Newtonian*) yang dapat dikontrol pergerakannya melalui software, yaitu *stellarium*, yang telah ter*install* pada perangkat komputer serta dapat mengikuti objek benda langit ketika bergerak. Teleskop dilengkapi dengan mikrokontroller *Arduino* sebagai pengatur gerakan motor dc dan motor servo. Dua buah motor dc sebagai penggerak derajat azimuth dan altitude teleskop, dua buah potensiometer sebagai sensor derajat azimuth dan altitude, sebuah kamera yang dapat mengambil gambar langsung dari cermin sekunder teleskop dan satu buah motor servo untuk mengatur jarak *focus* kamera. Diagram blok keseluruhan sistem ditampilkan pada **Gambar 3.1**.



Gambar 3.1 Diagram Blok Sistem

Cara kerja keseluruhan sistem adalah:

a. *Stellarium* mengirim koordinat *RA* dan *DEC* ke komputer.
b. Komputer melakukan konversi koordinat *RA* dan *DEC* menjadi koordinat *Azimuth* dan *Altitude*.
c. Komputer mengirim koordinat *Azimuth* dan *Altitude ke* arduino.

d. Arduino menggerakkan motor dc berdasarkan koordinat hasil konversi dan mengatur kecepatan motor dc dengan metode *fuzzy logic* berdasarkan *rule based* yang telah dibuat sebelumnya.
e. Motor dc dan potensiometer dihubungkan oleh sebuah *gear*. Maka saat motor berputar, potensiometer juga akan ikut berputar. Arduino akan membandingkan nilai konversi derajat potensiometer dengan nilai derajat koordinat *azimuth* dan *altitude*. Jika nilai derajat potensiometer dan derajat koordinat *azimuth* dan *altitude* telah sama, motor akan berhenti. Jika nilai nilai perbandingan masih berbeda, motor akan terus berputar.
f. Saat teleskop sudah menghadap koordinat yang dituju, komputer mengirim perintah ke arduino yang kemudian akan menggerakkan motor servo untuk mengatur *focus* kamera.
g. Penentuan *focus* kamera dilakukan dengan *image processing* melalui metode konvolusi. Komputer akan membandingkan nilai hasil konvolusi dengan nilai *threshold* yang telah ditentukan. Jika nilai hasil konvolusi sama dengan nilai *threshold* maka kamera sudah dalam keadaan fokus. Sedangkan jika kurang dari nilai *threshold* maka kamera belum dalam keadaan focus, sehingga motor servo akan terus berputar agar kamera menjadi focus.
h. Setelah proses penentuan focus selesai, kamera menangkap gambar *real-time* dari teleskop dan menampilkannya pada layar monitor komputer. Pada komputer kemudian dilakukan proses *tracking* dengan *image processing* melalui metode *hough transform* terhadap gambar yang ditangkap kamera.
i. Dengan metode *hough transform* komputer akan menandai benda langit yang berbentuk lingkaran dan menentukan titik tengahnya.
j. Saat benda langit yang ditangkap kamera bergerak, maka titik tengan objek akan bergeser. Komputer akan memerintahkan arduino untuk menggerakkan motor dc agar titik tengah objek berada pada tengah kamera, sehingga objek bisa terus diikuti pergerakannya.

3.1 Perancangan Perangkat Keras

Perancangan perangkat keras sistem meliputi pemasangan motor dc dan potensiometer untuk gerakan *azimuth* dan *altitude* teleskop, kamera

dan motor servo sebagai pengatur jarak *focus* kamera, *driver* motor dc sebagai pengatur kecepatan dan arah putaran motor dc, *display* lcd untuk menampilkan nilai derajat dan nilai ADC sensor serta koneksi sensor potensiometer dengan arduino.

3.1.1 Motor DC dan Sensor Potensiometer

Motor yang digunakan sebagai penggerak teleskop pada tugas akhir ini adalah dua buah motor dc yang dapat bergerak pada 2 arah secara berlawanan. Arah pergerakan motor ditentukan oleh pemberian tegangan DC pada kedua input motor. Sedangkan potensiometer digunakan sebagai sensor sudut derajat *azimuth* dan *altitude* teleskop.

Motor dc dan potensiometer dihubungkan dengan sebuah *gear* untuk menggerakkan teleskop berdasarkan koordinat *azimuth* dan *altitude*. Ketika motor dc berputar, *gear* juga akan ikut berputar, akan memutar sensor derajat potensiometer.

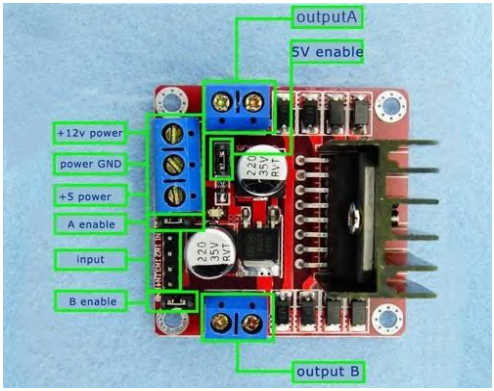
3.1.2 Kamera dan Motor Servo

Kamera berfungsi untuk menangkap gambar dari cermin sekunder teleskop dan menampilkannya pada layar monitor komputer. Motor servo digunakan untuk mengatur jarak fokus pada kamera. Arah pergerakan motor servo ditentukan oleh pemberian sinyal oleh arduino pada pin control motor servo. Bentuk kamera ditunjukkan pada **Gambar 3.2**.

Penempatan kamera dengan meletakkan ujung kamera pada lensa okuler teleskop yang langsung menghadap cermin sekunder, sehingga kamera dapat mengambil gambar langsung dari teleskop.

Gambar 3.2 Bentuk kamera

3.1.3 *Driver* Motor DC



Gambar 3.3 Koneksi pin *driver* Motor DC

Rangkaian ini berfungsi untuk mengatur gerakan kedua motor untuk mengarahkan arah gerakan teleskop sesuai dengan kendali baik secara manual dari pengguna ataupun kendali otomatis berdasarkan informasi dari kamera. Rangkaian pengendali motor yang digunakan pada tugas akhir ini adalah rangkaian *H-Bridge*. Kecepatan motor diatur dengan menginputkan PWM pada pin *enable driver* motor. **Gambar 3.3** merupakan *driver* motor dc yang digunakan.

Output A dihubungkam dengan motor untuk gerakan *altitude* teleskop, sedangkan output B dihubungkan dengan motor untuk gerakan *azimuth* teleskop. *Enable* A dihubungkan ke pin 10 PWM arduino dan *Enable* B dihubungkan ke pin 11 PWM arduino. Pin input dihubungkan ke pin 2, 3, 4, dan 5 pada arduino yang digunakan agar motor dc dapat bergerak pada 2 arah secara berlawanan. Pin 12V dan GND dihubungkan ke *power supply* 12V. Sedangkan pin 5V sebagai output yang dapat digunakan sebagai *supply* untuk arduino.

3.1.4 *Display* LCD

Penggunaan *display* lcd dimaksudkan untuk menampilkan data nilai *binary* potensiometer, nilai hasil konversi derajat potensiometer, dan nilai derajat yang akan dituju oleh teleskop. Koneksi LCD menggunakan I2C sehingga dapat menghemat penggunaan pin pada mikrokontroller.

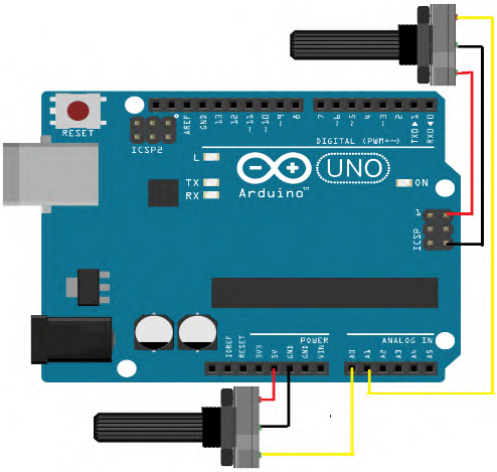
Pin I2C lcd untuk *power* dan *ground* dihubungkan dengan pin +5V dan GND pada arduino. Sedangkan untuk mengirimkan data I2C lcd yaitu SDA dan SCL dihubungkan dengan pin analog in A4 dan A5 pada arduino. I2C lcd memiliki *address* 0x27

3.1.5 Koneksi Sensor Potensiometer

Arduino memiliki 14 digital pin input / output (atau biasa ditulis I/O, dimana 6 pin diantaranya dapat digunakan sebagai output PWM), 6 pin input analog, menggunakan crystal 16 MHz, koneksi USB, jack, header ICSP dan tombol reset.

Pada sistem ini mikrokontroller arduino merupakan pusat proses data dari sensor potensiometer untuk mengubah nilai biner ADC potensiometer menjadi nilai derajat. Sensor potensiometer memiliki 3 pin yaitu pin tegangan +5V, pin GND, dan pin data. Pin 5V dan GND sensor dihubungkan ke pin 5V dan GND mikrokontroller arduino. Pin sensor data untuk derajat *altitude* dihubungkan ke pin arduino *analog in* A0, dan untuk pin sensor data derajat *azimuth* dihubungkan ke pin arduino *analog in* A1. Koneksi pin sensor ditunjukkan pada **Gambar 3.4**.

Pin *analog in* digunakan untuk mengubah data analog berupa tegangan pada sensor menjadi data digital yang kemudian ditampilkan pada layar lcd.



Gambar 3.4 Koneksi pin sensor dengan arduino

3.2 Perancangan Perangkat Lunak

Perangkat lunak yang dirancang yaitu perangkat lunak pada mikrokontroler dan pada komputer. Perangkat lunak pada arduino dirancang untuk melakukan proses konversi ADC sensor, komunikasi dengan komputer, *fuzzy logic*, mengontrol kecepatan dan arah motor dc dan pengaturan dari motor servo. Kemudian Perancangan Perangkat lunak pada komputer meliputi *interface* pada visual studio, pengambilan koordinat dari *stellarium* dan proses pengolahan citra dari kamera.

3.2.1 Perangkat Lunak Pada Arduino

Bahasa pemrograman Arduino menggunakan bahasa pemrograman C/C++ sebagai dasarnya. Namun dalam penulisannya terdapat sedikit perbedaan dengan bahasa C/C++. Perbedaan yang paling terlihat adalah struktur bahasa yang digunakan pada bahasa Arduino. Pada Arduino, fungsi utama dipisahkan menjadi dua bagian yaitu fungsi *setup*() dan fungsi *loop*() untuk memudahkan pembacaan.

Dalam diagram blok sistem pada **Gambar 3.1** dapat dilihat bahwa ada 3 tugas yang dilakukan oleh mikrokontroller yaitu membaca tegangan analog sensor potensiometer, mengatur motor dc untuk bergerak kearah koordinat yang dituju, dan mengatur putaran motor servo untuk fokus kamera. Untuk menggerakkan teleskop, arduino akan mendapatkan perintah dari komputer berupa nilai derajat yang akan dituju. Kemudian arduino mengirim sinyal pulsa konstan high ke input driver motor dc untuk menentukan arah putaran motor dc. Pada saat motor berputar, arduino akan terus membaca tegangan keluaran sensor dan membandingkannya dengan nilai derajat yang dituju.

3.2.1.1 Fungsi konversi binary ke derajat

Fungsi ini mengubah nilai pembacaan sensor potensiometer yang berupa nilai binary ADC menjadi nilai derajat. Nilai pembacaan ADC arduino mulai dari 0 sampai maksimum 1023. Untuk minimum derajat azimuth yaitu 0 derajat pada posisi menghadap utara. Sedangkan untuk minimum derajat altitude yaitu 90 derajat pada posisi berdiri tegak. Nilai ADC yang terbaca pada posisi-posisi tersebut kemudian akan dikonversi menjadi nilai derajat.

3.2.1.2 Perhitungan Resolusi ADC

Arduino memiliki resolusi ADC sebesar 10 bit, dengan meggunakan Vref sebesar 5V. Nilai resistansi potensiometer yang digunakan sebesar 10k. Nilai biner maksimum ADC sebesar 2^n-1, yaitu 2^{10}-1 sama dengan 1023, sehingga:

$$1LSB = \frac{Vref}{(2^n - 1)} = \frac{5}{(2^{10} - 1)} = \frac{5}{1023} = 4.8\text{mV}$$

Pada posisi 90 derajat dan 0 derajat, nilai ADC sensor yang terbaca untuk *azimuth* adalah 278 dan 158. Untuk mendapatkan tegangan yang terbaca pada posisi tersebut maka nilai tegangan 1 LSB x nilai ADC yang terbaca, maka:
Tegangan posisi 90º azimuth = 4.8mV x 336 = 1.61V
Tegangan posisi 0º azimuth = 4.8mV x 172 = 0.82V
untuk mendapatkan nilai tegangan setiap perubahan 1º diperoleh dengan cara:
tegangan setiap perubahan 1º = (1.61V – 0.82V) / 90 – 0, sehingga didapat sekitar 8.7mV/1º untuk azimuth.

Sedangkan pada posisi 90 derajat dan 0 derajat, nilai ADC sensor yang terbaca untuk *altitude* adalah 408 dan 572. Untuk mendapatkan tegangan yang terbaca pada posisi tersebut maka nilai tegangan 1 LSB x nilai ADC yang terbaca, maka:
Tegangan posisi 90º altitude = 4.8mV x 408 = 1.95V
Tegangan posisi 0º altitude = 4.8mV x 574 = 2.75V
untuk mendapatkan nilai tegangan setiap perubahan 1º diperoleh dengan cara:
tegangan setiap perubahan 1º = (1.95V – 2.74V) / 90 – 0, sehingga didapat sekitar 0.87mV/1º untuk altitude.

3.2.1.3 Pengaturan Motor DC

Untuk menentukan arah putaran motor dc, arduino mengirimkan sinyal pulsa konstan ke masing masing pin *input driver* motor dc. Sedangkan untuk mengatur kecepatannya, dengan mengirimkan sinyal pwm pada pin *enable driver* motor dc.

Penentuan nilai pwm yang diberikan untuk mengatur kecepatan motor dc diperoleh dengan metode fuzzy logic. Terdapat

tiga fungsi yang mengatur motor dc, yaitu fungsi putaran *clockwise*, fungsi putaran *counter clockwise*, dan fungsi *off*.

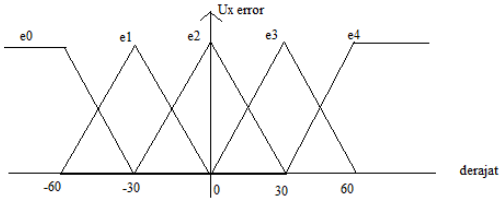
3.2.1.4 *Fuzzy Logic*

Logika fuzzy menggunakan rule base untuk dapat mengklasifikasikan himpunan data-data yang ada. Pada tugas akhir ini klasifikasi yang dilakukan adalah nilai error derajat (e) dan perubahan error derajat (Δe) sebagai input pada logika fuzzy. Error derajat (e) dibagi menjadi lima kelompok, ditunjukkan pada **Gambar 3.5**, yaitu:

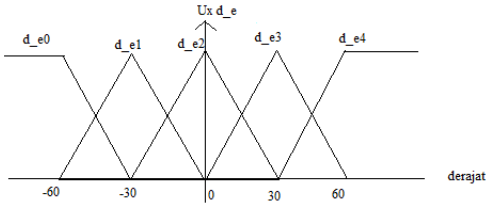
1. e0 : < -30º
2. e1 : 0 - (-60º)
3. e2 : -30 - 30º
4. e3 : 0 - 60º
5. e4 : > 30º

Sedangkan pada perubahan error derajat (Δe) dikelompokkan menjadi 5, ditunjukkan pada **Gambar 3.6,** yaitu:

1. d_e0 : < -30º
2. d_e1 : 0 - (-60º)
3. d_e2 : -30 - 30º
4. d_e3 : 0 - 60º
5. d_e4 : > 30º



Gambar 3.5 Himpunan fuzzy input 1



Gambar 3.6 Himpunan fuzzy input 2

Hasil output yang digunakan adalah nilai PWM yang akan dikirim ke *enable* driver motor DC untuk mengatur kecepatan motor DC. **Tabel 3.1** merupakan *rule base fuzzy* yang telah dibuat.

Tabel 3.1 *Rule Based Fuzzy Logic*[21]

	d_e0	d_e1	d_e2	d_e3	d_e4
e0	ncepat	ncepat	ncepat	nsedang	zero
e1	ncepat	nsedang	nlambat	zero	plambat
e2	ncepat	nlambat	zero	plambat	pcepat
e3	nlambat	zero	plambat	psedang	pcepat
e4	zero	psedang	pcepat	psedang	pcepat

3.2.2 Perangkat Lunak Pada Komputer

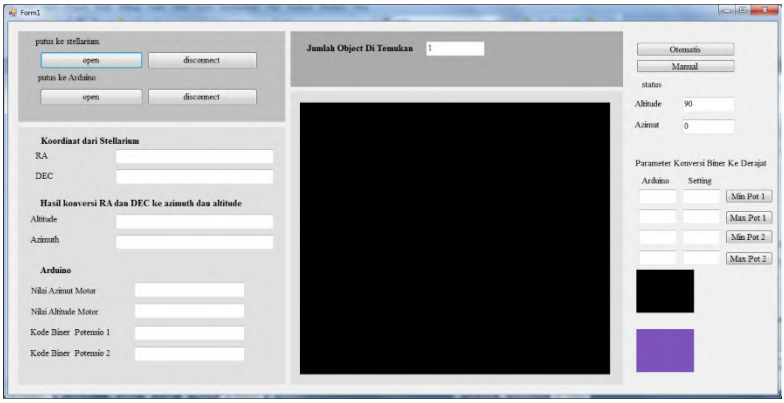
Sistem perangkat lunak yang akan dirancang pada komputer adalah pembuatan *interface* pada visual studio, pengambilan koordinat dari *stellarium*, konversi koordinat RA dan Dec ke *Azimuth* dan *Altitude*, komunikasi serial arduino dengan komputer dan proses pengolahan citra dari kamera. *Software* yang digunakan adalah *Microsoft visual studio* dengan bahasa C#.

3.2.2.1 *Interface* pada visual studio

Pembuatan *interface* pada visual studio bertujuan sebagai penghubung antara *software stellarium* dengan arduino.

Terdapat dua mode, yaitu mode otomatis dan mode manual. Dimana mode otomatis akan menggerakkan teleskop berdasarkan koordinat yang diperoleh dari *stellarium*. Sedangkan mode manual akan menggerakkan teleskop berdasarkan nilai derajat yang diinputkan sendiri oleh pengguna (*user*) pada kolom altitude dan azimuth.

Kolom koordinat dari stellarium akan menampilkan koordinat RA dan Dec yang diperoleh dari stellarium. Kolom hasil konversi merupakan nilai hasil konversi dari koordinat RA dan Dec menjadi azimuth dan altitude. Kolom arduino digunakan untuk memonitoring nilai hasil konversi sensor dan nilai pembacaan ADC sensor potensiometer. **Gambar 3.7** merupakan tampilan *interface* program yang dibuat pada *visual studio*.



Gambar 3.7 Tampilan *interface* pada visual studio

3.2.2.2 Pengambilan koordinat dari *stellarium*

Pengambilan data koordinat dari *stellarium* dilakukan melalui *Serial Command Protocol*, yaitu berupa symbol dan huruf. Saat inisialisasi pertama kali *stellarium* akan mengirimkan *command protocol* ke komputer berupa:

:GD# (*Get Declination Telescope*), returns: sDD*MM# atau sDD*MM'SS#
:GR# (*Get Right Ascension Telescope*), returns: HH:MM.T# atau HH:MM:SS#

yang artinya *Stellarium* meminta koordinat Dec (*Declination*) dan RA (*Right Ascencion*) awal teleskop. Kemudian komputer akan mengirimkan koordinat awal teleskop. Saat memilih objek yang akan dituju oleh teleskop, *stellarium* akan mengirimkan lagi *command protocol* ke komputer berupa:

:SdsDD*MM# (*Set Target Object Declination*), returns: 1 (Dec Accepted) atau 0 (Dec invalid)
:SrHH:MM:SS# (*Set target object RA*), returns: 1 (RA Accepted) atau 0 (RA invalid)

Dimana sDD, MM merupakan nilai Dec yang akan dituju oleh teleskop dan HH, MM, SS merupakan nilai RA yang akan dituju oleh teleskop.

3.2.2.3 Konversi koordinat RA dan Dec ke *Azimuth* dan *Altitude*

Konversi koordinat RA dan Dec ke *azimuth* dan *altitude* diperlukan karena jenis teleskop yang digunakan yaitu jenis altazimuth, dimana perputarannya berdasarkan koordinat *azimuth* dan *altitude*. Ada 3 poin yang diperlukan untuk melakukan konversi, yaitu:

1. Nilai RA dan Dec, didapatkan dari *stellarium*
2. Waktu, didapatkan dari komputer
3. *Lattitude* (Garis lintang) dan *Longitude* (Garis bujur), didapatkan dari internet, contoh *google maps*.

Setelah didapatkan nilai yang diperlukan, kemudian melakukan konversi menjadi decimal dengan satuan derajat. Dengan langkah sebagai berikut[22]:

1. RA dinyatakan dalam satuan hours, minutes, seconds. Untuk mengubah RA menjadi nilai decimal dengan satuan derajat maka 15*(hours + (minutes/60) + (seconds/3600)).
2. Dec dinyatakan dalam satuan degrees, minutes, seconds. Untuk mengubah Dec menjadi nilai decimal dalam satuan derajat maka degrees + (minutes/60) + (seconds/3600).
3. Waktu dinyatakan dalam jam, menit, detik dalam UT (*Universal* Time). Waktu diubah kedalam bentuk desimal, nilai jam + (menit/60) + (detik/3600).
4. *Lattitude* (Garis lintang) dan *Longitude* (Garis bujur) yang didapatkan dari *google maps* yang sudah dalam decimal dengan satuan derajat.

Kemudian menghitung jumlah hari sejak J2000. J2000 merupakan referensi awal perhitungan hari yaitu sejak 1 januari 2000 AD jam 12.00 UT (Universal Time). Langkah-langkah untuk menghitung jumlah hari sejak J2000 sebagai berikut [22]:

1. Membagi waktu hasil konversi dengan 24.
2. Dari **Tabel 3.2** pada table A, cari jumlah hari sejak awal tahun hingga awal bulan pengamatan dilakukan. Normal year berjumlah 365 hari, sedangkan leap year (tahun kabisat) berjumlah 366 hari.

Tabel 3.2 Table untuk menentukan jumlah hari[22]

Tabel A Jumlah hari sampai awal bulan			Tabel B Jumlah hari sejak J2000 sampai awal tahun			
Bulan	Normal year	Leap Year	Tahun	Hari	Tahun	Hari
Jan	0	0	1998	-731.5	2010	3651.5
Feb	31	31	1999	-366.5	2011	4016.5
Mar	59	60	2000	-1.5	2012	4381.5
Apr	90	91	2001	364.5	2013	4747.5
Mei	120	121	2002	729.5	2014	5112.5
Jun	151	152	2003	1094.5	2015	5477.5
Jul	181	182	2004	1459.5	2016	5842.5
Ags	212	213	2005	1825.5	2017	6208.5
Sep	243	244	2006	2190.5	2018	6573.5
Okt	273	274	2007	2555.5	2019	6938.5
Nop	304	305	2008	2920.5	2020	7303.5
Des	334	335	2009	3286.5	2021	7669.5

Contoh jika pengamatan dilakukan pada bulan agustus, maka jumlah harinya 212 hari untuk normal year atau 213 hari untuk leap year.

3. Tanggal saat pengamatan dilakukan.
4. Dari **Tabel 3.2** pada table B, cari jumlah hari sejak J2000 hingga awal tahun pengamatan dilakukan. Contoh jika pengamatan dilakukan pada tahun 2007 maka jumlah harinya 2555.5 hari.
5. Menjumlahkan semua hasil yang didapat, jumlah hari sejak J2000 = poin 1 + poin 2 + poin 3 + poin 4

Kemudian menghitung LST (Local Sidereal Time) dengan rumus:

$$LST = 100.46 + 0.985647 * d + longitude + 15 * UT \quad (3.1)$$

dimana, d adalah jumlah hari sejak J2000.
longitude adalah garis bujur
UT adalah waktu hasil konversi

nilai LST harus diantara 0 sampai 360 derajat. Jika nilai LST diluar range tersebut maka perlu dijumlahkan atau dikurangi dengan kelipatan 360.

Lalu menghitung HA (Hour Angle) dengan rumus:

$$HA = LST - RA \quad (3.2)$$

dimana, LST adalah Loca Sidereal Time
RA adalah RA hasil konversi

nilai HA harus diantara 0 sampai 360 derajat. Jika HA bernilai negative, maka perlu dijumlahkan dengan 360.

Setelah didapatkan nilai hasil konversi, LST dan HA, selanjutkan menentukan azimuth dan altitude dengan rumus:

$$\sin(ALT) = \sin(DEC)*\sin(LAT)+\cos(DEC)*\cos(LAT)*\cos(HA) \quad (3.3)$$
$$ALT = \text{asin}(ALT) \quad (3.4)$$

$$\cos(A) = \sin(DEC) - \sin(ALT)*\sin(LAT) / \cos(ALT)*\cos(LAT) \quad (3.5)$$
$$A = \text{acos}(A) \quad (3.6)$$

jika sin(HA) bernilai negative, maka azimuth = A. Sedangkan jika sin(HA) bernilai positif, maka azimuth = 360 – A.

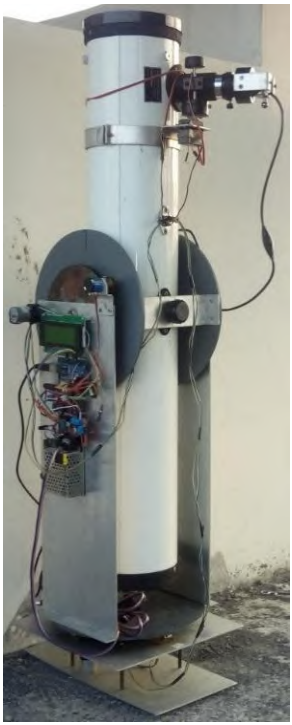
3.2.2.4 Komunikasi serial arduino dengan komputer

Komunikasi serial arduino dengan komputer bertujuan untuk mengirimkan nilai hasil konversi koordinat RA dan Dec. Nilai tersebut akan digunakan sebagai set point konversi derajat sensor potensiometer.

Pengiriman data azimuth dan altitude ke arduino dilakukan secara bergantian. Data nilai altitude akan dikirim dengan format “nilai altitude”-motor2+. Sedangkan data nilai azimuth dikirim dengan format “nilai azimuth”-motor1+.

BAB IV
PENGUJIAN ALAT

Pada tugas akhir ini, pengujian dilakukan untuk mengetahui kinerja sistem secara keseluruhan. Pengujian sistem ini terdiri atas pengujian perangkat keras, pengujian perangkat lunak serta pengujian keseluruhan sistem. **Gambar 4.1** menampilkan tampilan mekanik keseluruhan sistem.



Gambar 4.1 Mekanik Keseluruhan sistem

4.1 Pengujian perangkat keras

Pengujian perangkat keras meliputi pengujian driver motor dc untuk menentukan arah putaran motor dc dan pengujian konversi pembacaan nilai derajat sensor potensiometer.

4.1.1 Pengujian driver motor dc

Pengujian yang dilakukan pada bagian ini adalah pengujian untuk mengatur arah putaran motor dc berdasarkan sensor potensiometer dengan menggunakan driver motor dc. Pengujian driver motor dilakukan dengan cara pada pin *input* di beri salah satu logika dan pin *enable* diberi logika High.

Saat inisialisasi kedua pin *enable* pada *driver* akan tetap aktif, sehingga untuk memberhentikan motor dc dengan memberikan sinyal *low* pada masing-masing *input*. Berikut program inisialisai motor.

```
void inisialMotor(){
  pinMode(mtr1, OUTPUT);
  pinMode(mtr2, OUTPUT);
  pinMode(mtr3, OUTPUT);
  pinMode(mtr4, OUTPUT);
  pinMode(enable1, OUTPUT);
  pinMode(enable2, OUTPUT);

  digitalWrite(enable1, HIGH);
  digitalWrite(enable2, HIGH);
}

void motor_1_cw(){
  digitalWrite(mtr1, LOW);
  digitalWrite(mtr2, HIGH);
}

void motor_1_ccw(){
  digitalWrite(mtr1, HIGH);
  digitalWrite(mtr2, LOW);
}

void motor_1_off(){
  digitalWrite(mtr1, LOW);
  digitalWrite(mtr2, LOW);
}

void motor_2_cw(){
```

```
  digitalWrite(mtr3, LOW);
  digitalWrite(mtr4, HIGH);
}

void motor_2_ccw(){
  digitalWrite(mtr3, HIGH);
  digitalWrite(mtr4, LOW);
}

void motor_2_off(){
  digitalWrite(mtr3, LOW);
  digitalWrite(mtr4, LOW);
}
```

Mtr1 sampai mtr4 dihubungkan ke pin input *driver* motor dc, enable1 dan enable2 dihubungkan ke pin *enable driver* motor dc.

Program untuk mengatur arah putaran motor dc sebagai berikut, untuk azimuth:

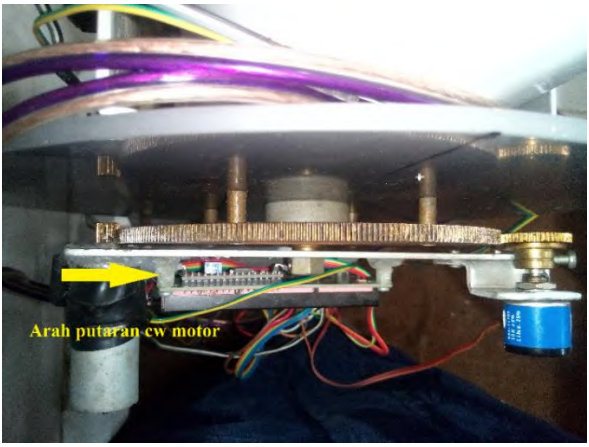
```
float hasil_1 = derajat_forMove_1 - derajat_now_1 ;
    if( hasil_1 > -1 ) motor_1_ccw();
    else motor_1_cw();
```

untuk altitude:

```
float hasil_2 = derajat_forMove_2 - derajat_now_2 ;
    if( hasil_2 > -1 ) motor_2_ccw();
    else motor_2_cw();
```

derajat_now merupakan nilai derajat posisi teleskop sekarang yang didapat dari sensor potensiometer, dan derajat_forMove merupakan nilai derajat yang menjadi tujuan teleskop yang didapat dari komputer.

Dari pengujian saat set derajat altitude (derajat_forMove_2) sebesar 45 derajat dan posisi derajat altitude teleskop sekarang (derajat_now_2) pada 90 derajat, sehingga hasil_2 < -1 maka arah putaran motor dc akan cw. Karena putaran motor dc *clockwise*, teleskop yang semula pada posisi tegak 90 derajat akan bergerak turun.



Gambar 4.2 Arah putaran *clockwise* pada motor dc altitude

Dari hasil pengujian *driver* motor dc, arah putaran motor dc berdasarkan sensor derajat potensiometer sudah sesuai yang diharapkan dengan kecepatan motor dc yang konstan seperti yang ditunjukkan **Gambar 4.2**.

4.1.2 Pengujian sensor derajat potensiometer

Pada pengujian pembacaan konversi derajat oleh sensor potensiometer dilakukan dengan kalibrasi secara manual. Kalibrasi dilakukan dengan cara mengarahkan teleskop secara manual pada posisi 0 dan 90 derajat untuk masing-masing koordinat *azimuth* dan *altitude*. Kemudian melihat nilai biner ADC yang terbaca pada kedua posisi tersebut pada display lcd. Nilai kalibrasi yang diperoleh untuk azimuth yaitu:

Min_biner = 172

Max_biner = 336

dan untuk altitude yaitu:

Min_biner = 574

Max_biner = 410

berikut program konversinya:

```
void degree_::setParameter(float min_derajat, float max_derajat, float min_biner, float max_biner ){
```

```
    data[ 0 ] = min_derajat ;
    data[ 1 ] = max_derajat ;
    data[ 2 ] = min_biner;
    data[ 3 ] = max_biner;
    data[ 4 ] = data[ 0 ] - data[ 0 ];
    data[ 5 ] = data[ 1 ] - data[ 0 ];
    data[ 6 ] = data[ 2 ] - data[ 2 ];
    data[ 7 ] = data[ 3 ] - data[ 2 ];
    data[ 9 ] = data[ 5 ] - data[ 4 ];
    data[ 10 ] = data[ 7 ] - data[ 6 ];
    data[ 11 ] = data[ 9 ] / data[ 10 ];
  }

  float degree_::Bin2Degree( float biner ){
    return ( data[ 11 ]  * ( biner - data[ 2 ] ) ) + data[ 0 ] ;
  }
```

Fungsi pertama untuk menentukan nilai parameter yang akan digunakan oleh fungsi kedua melakukan konversi dari nilai binary ADC menjadi nilai derajat. Nilai min_derajat dan max_derajat sudah ditentukan sebesar 0 dan 90. Nilai kalibrasi yang diperoleh kemudian dimasukkan kedalam min_biner dan max_biner.

Hasil pengujian konversi nilai ADC dilakukan dengan cara 15 kali pengambilan data dengan nilai derajat yang berbeda-beda, sehingga diketahui nilai error derajat. Posisi awal untuk azimuth adalah 0 derajat, sedangkan untuk altitude 90 derajat. Hasil pengujian ditampilkan pada **Tabel 4.1** dan **Tabel 4.2**.

Tabel 4.1 Hasil pengujian pembacaan konversi derajat *azimuth*

No.	Set derajat (derajat)	Hasil konversi (derajat)	Error (derajat)
1	0	0.8	-0.8
2	10	10.8	-0.8
3	25	25.8	-0.8
4	30	30.5	-0.5

No.	Set derajat (derajat)	Hasil konversi (derajat)	Error (derajat)
5	45	45.3	-0.3
6	60	60.5	-0.5
7	75	75.6	-0.6
8	90	89.7	0.3
9	135	134.5	0.5
10	180	179.5	0.5
11	200	199.5	0.5
12	220	219.3	0.7
13	270	269.2	0.8
14	310	309.2	0.8
15	360	359.3	0.7

Tabel 4.2 Hasil pengujian pembacaan konversi derajat *altitude*

No.	Set derajat (derajat)	Hasil konversi (derajat)	Error (derajat)
1	90	89.5	0.5
2	85	85.5	-0.5
3	81	81.2	-0.2
4	79	79.7	-0.7
5	75	75.7	-0.7
6	66	66.9	-0.9
7	55	55.3	-0.3
8	45	45.9	-0.9

No.	Set derajat (derajat)	Hasil konversi (derajat)	Error (derajat)
9	39	39.7	-0.7
10	33	33.9	-0.9
11	30	30.8	-0.8
12	25	25.6	-0.6
13	15	15.5	-0.5
14	5	5.7	-0.7
15	0	0.4	-0.4

Dari hasil pengujian pembacaan hasil konversi sensor derajat potensiometer untuk derajat azimuth dan derajat altitude didapatkan kesalahan absolut untuk derajat azimut rata-rata sebesar 0.6 derajat. Sedangkan kesalahan absolut untuk derajat altitude rata-rata sebesar 0.6 derajat.

4.2 Pengujian perangkat lunak

Pengujian perangkat lunak meliputi pengujian hasil konversi RA dan Dec menjadi azimuth dan altitude dan pengujian gerakan teleskop berdasarkan koordinat stellarium

4.2.1 Pengujian konversi RA dan Dec

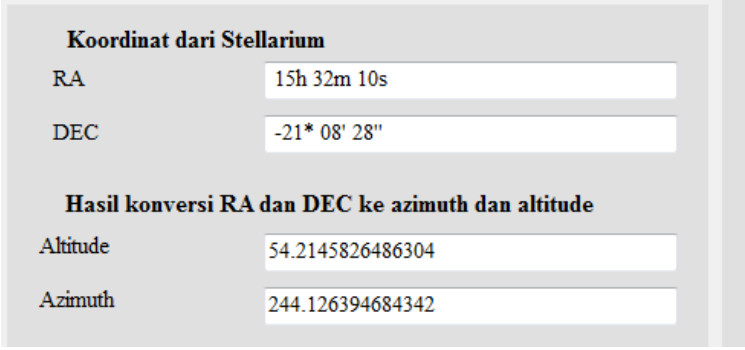
Pengujian ini dilakukan utuk mengetahui hasil konversi koordinat RA dan Dec dari *stellarium* menjadi koordinat azimuth dan altitude. Nilai hasil konversi akan ditampilkan pada *interface* visual studio. Untuk mengetahui apakah hasil konversi tersebut benar atau salah, maka harus ada nilai pembanding yang akurat. Nilai pembanding tersebut dapat dilihat pada tampilan stellarium.

Ketika memilih suatu objek, misalkan Mars, maka akan muncul pada sebelah kiri data koordinat planet tersebut. **Gambar 4.3** menunjukkan tampilan data mars pada *stellarium.* Mars memiliki koordinat RA = 15h32m10.63s, Dec = -21º08’28.3”, azimuth = 244º, dan altitude = 54º. Nilai azimuth dan altitude pada stellarium tersebut

yang akan menjadi nilai pembanding untuk hasil konversi RA dan Dec.

Dari hasil pengujian yang ditunjukkan pada **Gambar 4.4** dan **Gambar 4.5**, diketahui hasil konversi koordinat RA dan Dec pada komputer telah sesuai dengan nilai pembanding pada stellarium. Sedangkan pembacaan sensor memiliki error sebesar 1°.

Gambar 4.3 Tampilan data mars pada stellarium



Gambar 4.4 Koordinat mars dan hasil konversinya

Gambar 4.5 Tampilan nilai derajat mars pada lcd

4.2.2 Pengujian teleskop berdasarkan stellarium

Pada pengujian ini teleskop akan digerakkan menuju objek yang dipilih melalui *stellarium*. Arah azimuth 0 derajat teleskop harus dihadapkan pada arah utara 0 derajat, sehingga teleskop dapat tepat menghadap objek. Pengujian dilakukan dengan menggerakkan teleskop ke arah bulan dan planet mars.

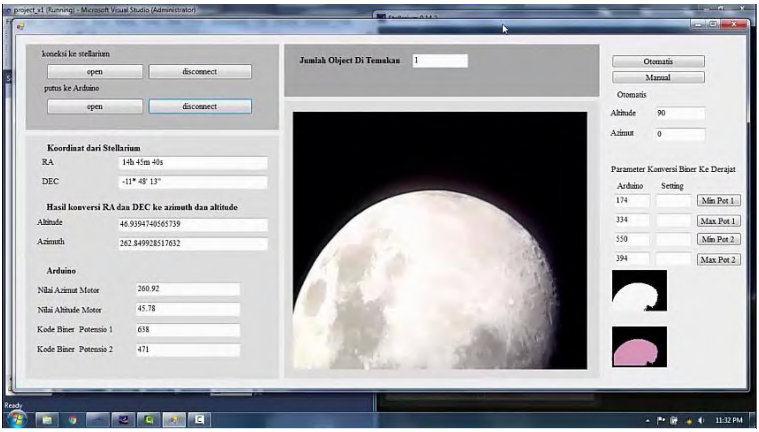
Kamera usb dipasang menghadap ke arah lensa okuler teleskop. Lensa okuler yang digunakan pada teleskop memiliki *focal length* 20mm. Sedangkan *focal length* teleskop 900 mm. Pemilihan lensa okuler dengan ukuran *focal length* 20 mm dimaksudkan untuk mendapatkan pembesaran teleskop yang lebih kecil, sehingga objek yang memiliki jarak lebih dekat dengan pengamat seperti bulan dapat ditampilkan dengan ukuran penuh.

Pembesaran teleskop diperoleh dengan rumus:

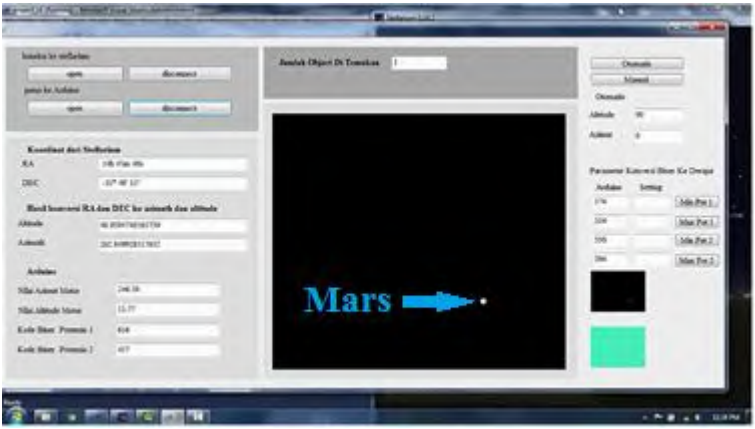
$$\text{Pembesaran} = \frac{focal\ length \text{ teleskop}}{focal\ length \text{ lensa okuler}} \tag{4.1}$$

$$= \frac{900\ mm}{20\ mm}$$

$$= 45 \text{ kali}$$



Gambar 4.6 Tampilan bulan yang diperoleh oleh teleskop



Gambar 4.7 Tampilan mars yang diperoleh oleh teleskop

Dari **Gambar 4.6** dan **Gambar 4.7** dapat dilihat bahwa bulan dan planet mars tidak tepat berada pada posisi tengah layar. Hal ini salah satunya dikarenakan *error* yang terdapat pada pembacaan sensor potensiometer. Selain itu perancangan antara motor dc dan gear penghubung dengan sensor potensiometer dapat mempengaruhi pengambilan gambar oleh teleskop.

LAMPIRAN

Lampiran A

Listing Program Arduino

```
//==================== class untuk pengubahan parameter biner menjadi degree =====//
class degree_{
  public :
   float data[12] ;
  public :
   void setParameter( float min_derajat , float max_derajat , float min_biner , float max_biner  );
   float Bin2Degree( float biner );
};

void degree_::setParameter(float min_derajat , float max_derajat , float min_biner , float max_biner ){
   //====================> set nilai maximal dan minimal derajat
   data[ 0 ] = min_derajat ;  //===> set nilai minimal derajat
   data[ 1 ] = max_derajat ;  //===> set nilai maximal derajat
   data[ 2 ] = min_biner   ;  //===> set nilai minimal biner putaran
   data[ 3 ] = max_biner   ;  //===> set nilai maximal biner putaran
   //====================> lakukan langkah untuk meng null kan nilai minimal dan maximal
   //====> contoh :
   //      nilai minimal : 10
   //      nilai maximal : 290
   //====> lakukan langkah untuk meng null kan
   //      nilai minimal baru = 10 - 10 = 0
   //      nilai maximal baru = 290 - 10 = 280
   data[ 4 ] = data[ 0 ] - data[ 0 ]; //====> nilai minimal derajat yang baru
   data[ 5 ] = data[ 1 ] - data[ 0 ]; //====> nilai maximal derajat yang baru
   data[ 6 ] = data[ 2 ] - data[ 2 ]; //====> nilai minimal biner yang baru
   data[ 7 ] = data[ 3 ] - data[ 2 ]; //====> nilai maximal biner yang baru
```

```
    //====> lakukan perhitungan untuk mengetahui perbedaan nilai maximal dan minimal
    //====> atau bisa di sebut dengan normalisasi
    data[ 9 ] = data[ 5 ] - data[ 4 ]; //====> nilai perbedaan minimal dan maximal derajat
    data[ 10 ] = data[ 7 ] - data[ 6 ]; //===> nilai perbedaan minimal dan maximal biner

    data[ 11 ] = data[ 9 ] / data[ 10 ]; //====> hitung nilai
  }

  float degree_::Bin2Degree( float biner ){
   return ( data[ 11 ]  * ( biner - data[ 2 ] ) ) + data[ 0 ] ;
  }

  degree_ degreeMotor_1 ;
  degree_ degreeMotor_2 ;
  //=============================================================================//

  //==================== parameter derajat motor 2 ============//

  boolean motor_1_still_move_1 = false  ;
  boolean afterKalkulasi_1 = false  ;
  float derajat_now_1 = 0.0 ;
  float derajat_forMove_1 = 0.0 ;

  boolean motor_2_still_move_2 = false  ;
  boolean afterKalkulasi_2 = false  ;
  float derajat_now_2 = 0.0 ;
  float derajat_forMove_2 = 0.0 ;

  int sns_motor_1 = 1 ;
  int sns_motor_2 = 0 ;
  //=============================================================================//
```

```
    //============== set degree parameter =============//
      degreeMotor_1.setParameter(  0.00  ,  90.00  ,
read_(addr_min_motor_1) , read_(addr_max_motor_1) );
      degreeMotor_2.setParameter(  0.00  ,  90.00  ,
read_(addr_min_motor_2) , read_(addr_max_motor_2) );

    //======================== fungsi untuk mengawasi
pergerakan motor 2 ========//

    void trackingMoveMantMotor1(){
      if( motor_1_still_move_1 ){
        derajat_now_1  = readDerajat_x1(sns_motor_1);
        if( !afterKalkulasi_1 ){
          float hasil_1 = derajat_forMove_1 - derajat_now_1 ;
          if( hasil_1 > -1 ) motor_1_ccw();
          else motor_1_cw();
          afterKalkulasi_1 = true ;
        }
        if( abs( derajat_forMove_1 - derajat_now_1 ) < 0.8 ){
motor_1_off(); motor_1_still_move_1 = false ; }
      }
      else motor_1_off();
    }
    void trackingMoveMantMotor2(){
      if( motor_2_still_move_2 ){
        derajat_now_2  = readDerajat_x2(sns_motor_2);
        if( !afterKalkulasi_2 ){
          float hasil_2 = derajat_forMove_2 - derajat_now_2 ;
          if( hasil_2 > -1 ) motor_2_ccw();
          else motor_2_cw();
          afterKalkulasi_2 = true ;
        }
        if( abs( derajat_forMove_2 - derajat_now_2 ) < 0.8 ){
motor_2_off(); motor_2_still_move_2 = false ; }
      }
      else motor_2_off();
    }
```

```
//===== fungsi untuk kendali motor======//
void inisialMotor(){
  //===> set motor sebagai output
  pinMode(mtr1, OUTPUT);
  pinMode(mtr2, OUTPUT);
  pinMode(mtr3, OUTPUT);
  pinMode(mtr4, OUTPUT);
  pinMode(enable1, OUTPUT);
  pinMode(enable2, OUTPUT);

  // aktifkan motor
  digitalWrite(enable1, HIGH);
  digitalWrite(enable2, HIGH);
}
void motor_1_cw(){
  digitalWrite(mtr1, LOW);
  digitalWrite(mtr2, HIGH);
}
void motor_1_ccw(){
  digitalWrite(mtr1, HIGH);
  digitalWrite(mtr2, LOW);
}
void motor_1_off(){
  digitalWrite(mtr1, LOW);
  digitalWrite(mtr2, LOW);
}
void motor_2_cw(){
  digitalWrite(mtr3, LOW);
  digitalWrite(mtr4, HIGH);
}
void motor_2_ccw(){
  digitalWrite(mtr3, HIGH);
  digitalWrite(mtr4, LOW);
}
void motor_2_off(){
  digitalWrite(mtr3, LOW);
  digitalWrite(mtr4, LOW);
}
```

Lampiran B

Listing Program Visual Studio

// Program konversi //

```
class cls_RaDecToAltAz
    {
        double RA = 250.425; // 16 h 41.7 m * 15
        double Dec = 36.46667; // 35 ° 30 m
        public double Lat = -7.2915137;
        public double Long = 112.7967808;

        public double Alt = 0;
        public double Az = 0;

        public void Calculate(double RA, double Dec,
double Lat, double Long)
        {
            Calculate(RA, Dec, Lat, Long,
DateTime.UtcNow);
        }

        public void Calculate(double RA, double Dec)
        {
            Calculate(RA, Dec, this.Lat, this.Long,
DateTime.UtcNow);
        }

        public void Calculate(double RA, double Dec,
double Lat, double Long, DateTime Date)
        {
            // Day offset and Local Siderial Time
            double dayOffset = (Date - new
DateTime(2000, 1, 1, 12, 0, 0,
DateTimeKind.Utc)).TotalDays;
            double LST = (100.46 + 0.985647 * dayOffset
+ Long + 15 * (Date.Hour + Date.Minute / 60d) + 360) %
360;

            // Hour Angle
            double HA = (LST - RA + 360) % 360;
```

```
            // HA, DEC, Lat to Alt, AZ
            double x = Math.Cos(HA * (Math.PI / 180)) *
Math.Cos(Dec * (Math.PI / 180));
            double y = Math.Sin(HA * (Math.PI / 180)) *
Math.Cos(Dec * (Math.PI / 180));
            double z = Math.Sin(Dec * (Math.PI / 180));

            double xhor = x * Math.Cos((90 - Lat) *
(Math.PI / 180)) - z * Math.Sin((90 - Lat) * (Math.PI /
180));
            double yhor = y;
            double zhor = x * Math.Sin((90 - Lat) *
(Math.PI / 180)) + z * Math.Cos((90 - Lat) * (Math.PI /
180));

            double az = Math.Atan2(yhor, xhor) * (180 /
Math.PI) + 180;
            double alt = Math.Asin(zhor) * (180 /
Math.PI);

            this.Alt = alt; this.Az = az;
        }
    }
```

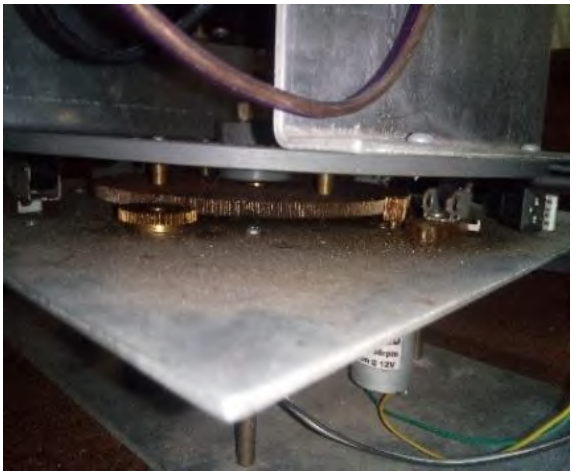
// program ambil data stellarium //

```
void sp_DataReceived(object sender,
SerialDataReceivedEventArgs e)
        {
            msg.msg += _serialPort.ReadExisting();

            if (msg.msg.IndexOf(":GR#") > -1)
            {
                _serialPort.Write("00:00:00#");
                msg.msg = "";
            }
            else if (msg.msg.IndexOf(":GD#") > -1)
            {
                int[] arr = new int[] { 43, 48, 48,
223, 48, 48, 58, 48, 48, 35 };
                String buff = "";
```

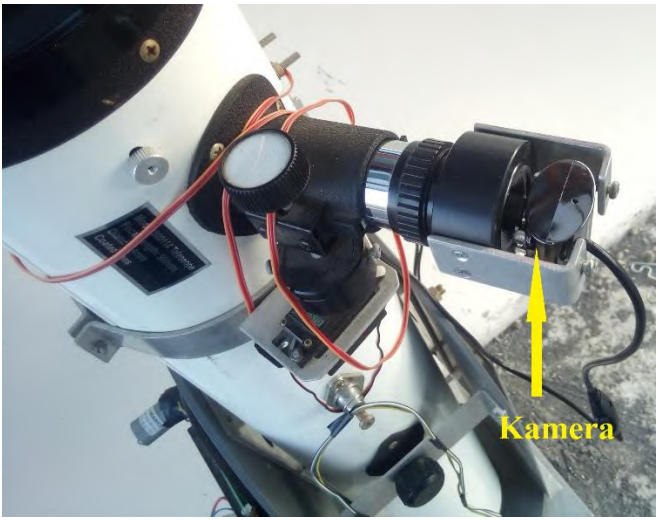
```
                for (int i = 0; i < arr.Length; i++)
                {
                    buff += (char)arr[i];
                }
                _serialPort.Write(buff);
                msg.msg = "";
            }
            else if (msg.msg.IndexOf(":Sr") > -1)
            {
                _serialPort.Write("1");
                msg.cor += msg.msg;
                msg.msg = "";
            }
            else if (msg.msg.IndexOf(":Sd") > -1)
            {
                _serialPort.Write("1");
                msg.cor += msg.msg;
                this.parse.parse(msg.cor);

                cordinat = new
cls_corStar(this.parse.RDHH, this.parse.RDMM,
this.parse.RDSS, this.parse.DECHH, this.parse.DECMM,
this.parse.DECSS);

                msg.cor = "";
                msg.msg = "";
            }
```

Lampiran C

Pemasangan Motor DC dan Potensiometeruntuk gerakan *altitude*



Pemasangan Motor DC dan Potensiometer untuk gerakan *azimuth*



Penempatan kamera pada teleskop

Hasil konversi sensor derajat potensiometer

BAB V
PENUTUP

Setelah dilakukan rangkaian kegiatan perancangan sistem dan pengujian alat penulis memperoleh kesimpulan dan memberikan beberapa saran sebagai berikut

5.1 Kesimpulan

Dalam Tugas Akhir ini dirancang sebuah sistem berupa teleskop yang dapat melakukan pelacakan objek astronomi secara otomatis. *Inisialisai* sistem dengan pengambilan *database* objek dari *software stellarium* berupa koordinat benda yang diamati. Kemudian berdasarkan koordinat tersebut teleskop akan bergerak secara otomatis menuju objek yang dipilih pada *stellarium*. Teleskop dilengkapi dengan kamera usb untuk mengambil gambar dari lensa okuler teleskop dan kemudian ditampilkan pada layar monitor komputer.

Berdasarkan beberapa pengujian yang dilakukan, kesimpulan yang diperoleh dalam Tugas Akhir ini adalah perubahan kecil pada nilai ADC sensor dapat mempengaruhi konversi nilai derajat teleskop. Perubahan tersebut mengakibatkan nilai error konversi rata-rata sebesar 0.6 derajat, sehingga dapat mempengaruhi pengambilan gambar oleh kamera. Perancangan motor dc dan gear penghubung dengan sensor potensiometer yang kurang akurat dapat menjadi faktor penyebab teleskop tidak dapat menangkap gambar objek yang diinginkan karena error pada putaran motor dc dan gear. Selain itu penempatan posisi teleskop pada sudut 0 derajat azimuth harus tepat dihadapkan pada arah utara 0 derajat agar teleskop tepat menghadap posisi yang dituju.

5.2 Saran

Beberapa saran yang dapat penulis berikan untuk pengembangan Tugas Akhir adalah sebagai berikut :

1. Diperlukan sensor yang lebih akurat sehingga nilai derajat dapat dikonversi dengan presisi.
2. Dibutuhkan perancangan hardware antara motor dc dan sensor potensiometer yang lebih baik agar didapatkan pembacaan sensor yang lebih akurat.

DAFTAR PUSTAKA

[1] Wikipedia, 2016. “Teleskop”. http:/id.wikipedia.org/wiki/Teleskop

[2] Rendydarma, 2011. “Peralatan Pengamatan Bintang”. http://astro-rendydarma.blogspot.co.id/2011/11/peralatan-pengamatan-bintang.html

[3] Irawan Dian, 2011, “sistem dan tata koordinat benda langit”, http://fisika-astronomy.blogspot.co.id/2012/11/sistem-dan-tata-koordinat-benda-langit.html

[4] Latifah W. Erni, 2011, “Sistem Koordinat Equatorial”, http://kafeastronomi.com/sistem-koordinat-equatorial.html.

[5] Stellarium, ___. http://www.stellarium.org.

[6] _____, ___. “Apakah Arduino itu”. http://ecadio.com/apakah-arduino-itu

[7] Wikipedia, 2014. “Logika Fuzzy”. https://id.wikipedia.org/wiki/Logika_fuzzy

[8] Elektronika Dasar, 2012. “Teori Motor DC Dan Jenis-Jenis Motor DC”. http://elektronika-dasar.web.id/teori-motor-dc-dan-jenis-jenis-motor-dc/

[9] Unik, Kado, 2015. “Motor Servo (Komponen Jemuran Otomatis)”. http://msaipudin.blogspot.co.id/2016/03/motor-servo-komponen-jemuran-otomatis.html

[10] Admin, 2014. “bagaimana itu? – motor servo dan cara kerjanya”. http://soerip.com/Dongeng/?p=63

[11] _____, 2015. “Pengertian Webcam beserta fungsinya”. http://kom-swb.blogspot.co.id/2012/10/pengertian-webcam-beserta-fungsinya.html

[12] Wahyudi, Agung, 2010. “Penerapan Algoritma”. Universitas Indonesia.

[13] Gazali Wikaria, Soeparno Haryono, Ohliati Jenny,” Penerapan Metode Konvolusi Dalam Pengolahan Citra Digital”, Binus University.

[14] Riwinoto. “Penggunaan Algoritma Hough Tranforms Untuk Deteksi Bentuk Lingkaran pada Ruang 2D”. Politeknik Batam.

[15] Rosebrock Adrian, 2015,”Blur detection with OpenCV”, http://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/.

[16] _____, ___. “Teropong”. http://www.rumus-fisika.com/2015/02/teropong.html

[17] Dutch, Steven, 2010. “Telescope Mounts”. https://www.uwgb.edu/dutchs/AstronNotes/NeedToKnow.HTM
[18] _____, 2009. “Koordinat Horison (Alt-Azimuth)”. http://duniaastronomi.com/2009/02/koordinat-horison-alt-azimuth/
[19] X, The, 2010. “Jangan lewatkan Autumnal Equinox tanggal 23 September 2010”. http://www.faktailmiah.com/2010/09/22/jangan-lewatkan-autumnal-equinox-tanggal-23-september-2010.html
[20] _____, 2014. “What is Arduino”. http://lide.uhk.cz/prf/ucitel/slegrja1/chemduino/arduino.htm
[21] Neema D.D, Patel R.N, Thoke A.S, 2011. “Speed Control of Induction Motor using Fuzzy Rule Base”, International Journal of Computer Applications.
[22] Burnett Keith, 1998, “Converting RA and DEC to ALT and AZ”, http://www.stargazing.net/kepler/altaz.html.

BIODATA PENULIS

Penulis bernama lengkap Afif Aulia Rahman. Lahir di Mataram pada tanggal 4 Desember 1990, merupakan anak pertama dari tiga bersaudara. Penulis memulai pendidikan formal di SDN 1 Mataram yang kemudian dilanjutkan di SMPN 1 Mataram. Pendidikan menengah atas penulis ditempuh di SMA Assalaam Solo. Pada tahun 2010, penulis melanjutkan pendidikan S1 di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember dengan konsentrasi pada bidang studi Elektronika. Selama perkuliahan, penulis aktif mengikuti kegiatan kampus seperti kepanitiaan dan menjadi asisten laboratorium elektronika dasar dan rangkaian listrik.