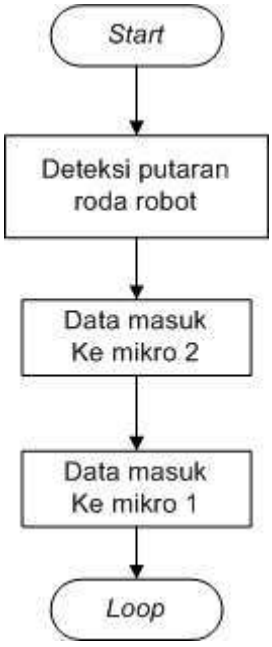
BAB III

ANALISIS DAN PERANCANGAN SISTEM

III.1 Analisis Sistem

Sistem rangkaian pada perangkat lunak sebelumnya merupakan proses berkesinambungan (*looping*), dimana memiliki rangkaian elektronik berbasis mikrokontroler berupa sistem kontrol yang menangani seluruh proses dari proses akuisisi data berupa data perubahan posisi dan proses pengendalian gerakan robot melalui aktuator. Aktuator sendiri adalah komponen penggerak seperti motor *DC* yang akan memindahkan posisi robot itu sendiri terhadap lingkungan yang dalam hal ini adalah *environment*, yang pada akhirnya informasi ini dijadikan masukan bagi sistem kontrol. Gambar III.1 adalah bagan sistem perangkat lunak pergerakkan roda robot mobil pada umumnya



Gambar III.1 Bagan sistem perangkat lunak pergerakkan roda robot mobil pada umumnya

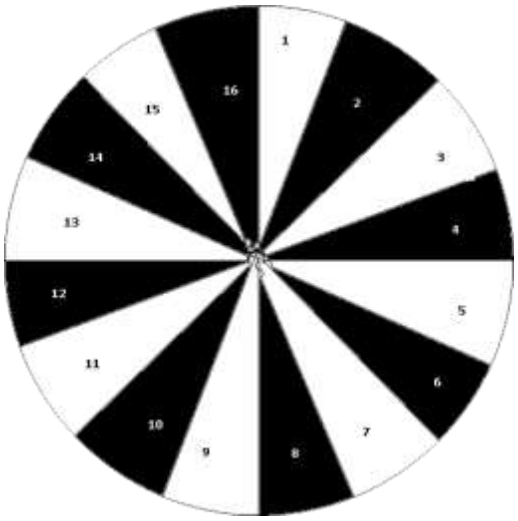
III.1.1 Analisis Masalah

Sistem pada perangkat lunak yang telah terdefinisikan akan memudahkan dalam perencanaan perancangan rangkaian. Keluaran atau hasil dari sistem adalah kecepatan putaran motor *DC* cenderung stabil, artinya pada saat robot melewati lintasan berupa tanjakkan, umumnya sinyal *Pulse Width Modulation* pada motor *DC* akan menyempit sehingga robot cenderung berjalan lambat atau tidak kuat bergerak. Perlunya prediksi gangguan berupa data historian sebelum menyelesaikan *looping* sistem. Sehingga nilai banyaknya putaran yang dikehendaki dapat dicapai.

III.1.2 Analisis Algoritma Konvensional

Dalam keadaan nyata membutuhkan media atau alat yang dapat membaca atau mendeteksi putaran yang sedang berjalan. Media atau alat ini juga dapat membantu dalam mengatur putaran yang sedang berlangsung agar menghasilkan putaran yang konstan. Alat ini bernama *optocoupler,* alat ini nantinya juga akan membantu jika pada penerapan algoritma *Model Predicitve Control* diterapkan pada sisi hardware. Kondisi awal *optocoupler* akan mendeteksi putaran motor *DC*. Data yang terbaca oleh sensor berupa tegangan digital, kemudian diproses oleh mikrokontroler. Data tersebut dibandingkan dengan set point, untuk menjadi data keluaran sinyal *Pulse Width Modulation*. Jika data lebih kecil dari set point, maka sinyal *Pulse Width Modulation* tersebut diperlebar. Setelah itu, sinyal tersebut akan menggerakkan motor *DC* sehingga robot dapat berjalan dengan kecepatan tetap atau stabil.

Pada penerapan ke dalam sistem atau dalam hal ini berupa algoritma yang nantinya akan disimpan dalam mikrokontroler pada robot mobil beroda membutuhkan bantuan alat yang bernama *optocoupler* yang berfungsi agar dapat membaca bahwa telah terjadi putaran pada roda robot. Namun perlu ada pembacaan pada algoritma yang tertanam pada robot berupa menghitung putaran yang sedang terjadi. Dengan melakukan pengidentifikasian akan dimisalkan berupa piringan yang diberi warna selang – seling hitam dan putih seperti pada gambar III.2 piringan hitam putih. Piringan tersebut dapat dipasangkan pada roda robot. Jadi jika pada roda robot mengalami pergerakkan, piringan tersebut juga ikut berputar mengikuti putaran yang sama dengan roda pada robot mobil.



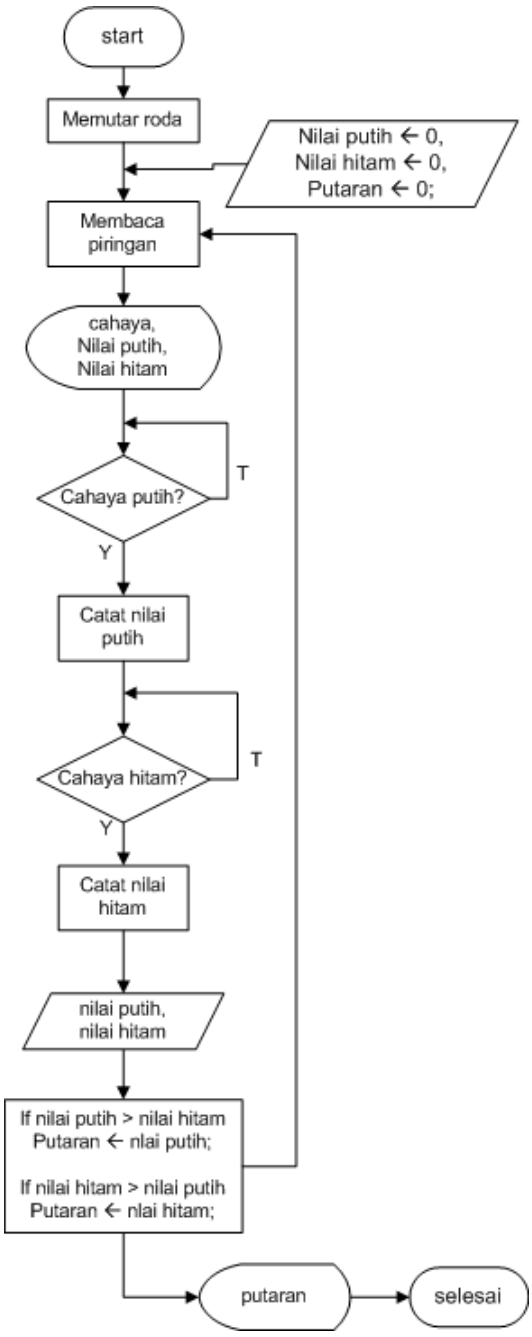
Gambar III.2 Piringan hitam - putih

Piringan hitam putih berguna jika pada saat membutuhkan parameter yang berkaitan dengan putaran roda. Semakin banyak jumlah warna hitam dan putih pada piringan, semakin teliti pula dalam hal pencatatan putaran. Dikarenakan semakin besar nilai pembaginya. Misal mengunakan piringan dengan jumlah

hitam dan putih 16, maka ketelitian minimumnya merupakan kelipatan 1/16 atau 0,0625. Jika jumlah piringan tersebut dibandingkan dengan piringan yang memiliki jumlah warna hitam dan putihnya lebih sedikit misal 4 atau 8 maka ketelitian minimum perhitungan waktu untuk hasil putaran menjadi sebesar 1/4 atau 0,25 hingga 1/8 atau 0,125.

Tabel III.1 *Pseudocode* membaca piringan hitam putih pada proses membaca putaran

```
Procedure hitung putaran()
deklarasi :
      putih, hitam, putaran, t, Rps := integer;
      timer := time;
deskripsi :
      putih := 0;
      hitam := 0;
      putaran := 0;
      t := 0;
      rps := 0;
      t := TimeToInt(timer);
If putih < 1 then //piringan putaran pertama
            putih += 1;
      Else If (putih >= 1 AND putih <= 15) then
      //piringan putih
            putih += 2;
      Else If putih > 15 then //looping
            putih ← 1;
  End If;
  If hitam <= 16 then //piringan hitam
            hitam += 2;
      Else If hitam > 16 then //looping
            hitam ← 2;
  End If;
  If hitam == 16 then //per satu kali putaran
      putaran += 1;
  End If;
  /* pencatatan timer per 1 detik */
  If (t - (t - 1)) == 1 then
      If putih > hitam then
            Write putih;
            Rps ← putaran + (putih / 16);
      Else If hitam > putih then
            Write hitam;
            Rps ← putaran + (hitam / 16);
      Else If putih == hitam then
            Rps ← null;
      End If;
  End If;
```



Gambar III.3 Flowchart saat membaca piringan pada proses membaca putaran

Kelebihan :

1. Waktu untuk mengeksekusi relatif lebih cepat.
2. Tidak membutuhkan banyak variabel untuk melakukan pergerakan roda.

Kekurangan :

1. Jika dengan kondisi lintasan berubah, dalam hal ini menanjak. Kecepatan putaran roda akan tidak stabil. Dikarenakan belum adanya kontrol yang dapat menjaga nilai banyaknya putaran pada roda.
2. Waktu tempuh akan lebih lama pada kondisi lintasan menanjak.
3. Tidak adanya variabel yang menampung untuk mengkontrol putaran roda.

III.1.3 Analisis Algoritma *Model Predictive Control*

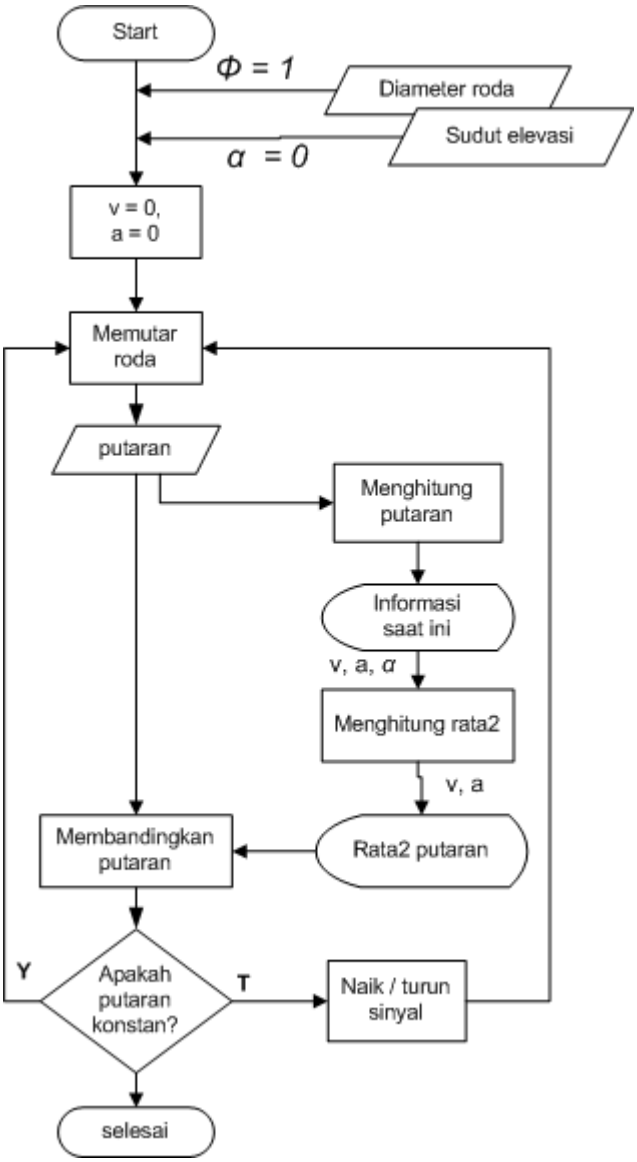
Proses pada algoritma *Model Predictive Control* yang digunakan pertama kali dilakukan robot mobil beroda adalah *start* dimana akan menjadi masukan untuk *optocoupler*. *Optocouler* berfungsi untuk membaca putaran roda robot, putaran ini yang akan menjadi masukan pada mikrokontroller untuk kemudian dilakukan proses selanjutnya berupa perbaikan agar nilai putaran dapat sesuai dengan yang diinginkan, berupa putaran yang konstan.

Pada gambar III.4 *flowchart* implementasi algoritma *Model Predictive Control* akan menjelaskan proses kerja rangkaian untuk mengatur kecepatan roda robot, yaitu:

1. Menentukan terlebih dahulu diameter roda, dan sudut elevasi yang akan disimulasikan.
2. Proses kerja selanjutnya ialah memutar roda, dengan masukan diameter roda dan sudut yang akan dilewati.

3. Saat roda telah berputar, maka akan ada proses pembacaan putaran roda dengan menghasilkan data berupa kecepatan, percepatan atau akselerasi dan informasi berupa sudut yang sedang dilalui.
4. Setelah mendapatkan kecepatan dan percepatan atau akselerasi. Akan dilakukan proses perhitungan yang menghasilkan rata – rata yang diambil dari kecepatan maupun percepatan yang telah dilalui sebelumnya.
5. Selama roda melakukan perputaran, nilai dari kecepatan sekarang akan dibandingkan dengan kecepatan rata – rata yang telah dihasilkan pada proses sebelumnya.
6. Dalam proses perbandingan putaran, akan didapat bahwa putaran yang sedang terjadi sekarang apakah konstan atau tidak. Jika putaran yang dinilai adalah konstan maka proses akan kembali(*looping*) pada proses memutar roda. Karena mempertahankan putaran yang ada agar putaran yang dihasilkan konstan. Namun jika dinilai putaran yang ada belum konstan sesuai dengan rata – rata putaran sebelumnya, maka akan melewati proses naik dan turunkan sinyal.
7. Proses dinyatakan selesai jika tidak terjadi putaran roda. Namun proses akan terus menerus berulang dimana putaran roda terus terjadi.

Pada perangkat lunak simulasi algoritma *Model Predictive Control* tidak akan membahas bagaimana cara menambahkan dan mengurangi sinyal agar putaran roda berubah. Karena proses tersebut ada pada level *hardware* berupa *Pulse Width Modulation* yang memang di luar dari bahasan algoritma *Model Predictive Control* itu sendiri.



Gambar III.4 Flowchart implementasi algoritma MPC

Pada proses naik dan menurunkan sinyal *Pulse Width Modulation* berupa proses di luar algoritma *Model Predictive Control* namun juga turut dijelaskan pada flowchart gambar III.4 di atas karena juga dapat membantu menjelaskan proses yang sedang berjalan, jika algoritma *Model Predictive Control* ini diterapkan pada proses pergerakkan putaran roda pada level hardware berupa

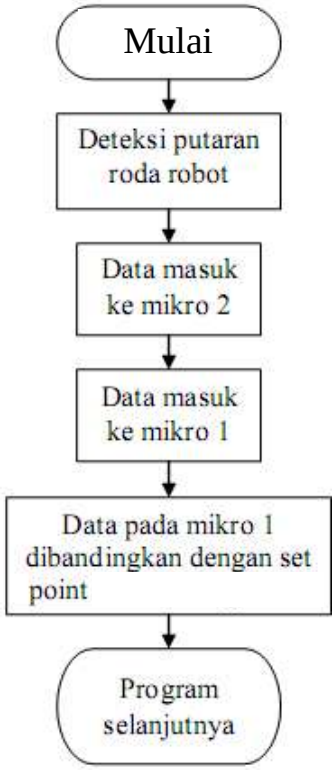
Pulse Width Modulation. *Pulse Width Modulation* akan mengatur cepat atau lambatnya robot berjalan dengan cara jika nilai *Pulse Width Modulation* yang berupa sinyal, lebih kecil daripada set *point*, maka pada mikrokontroller akan memperlebar sinyal *Pulse Width Modulation*, begitu juga sebaliknya. Jika nilai *Pulse Width Modulation* yang berupa sinyal, lebih besar daripada set *point*, maka pada mikrokontroller akan mempersempit sinyal. Sehingga robot dapat berjalan stabil. Sinyal yang dimaksud adalah berupa frekuensi listrik pada motor DC. Sedangkan set *point* yang dimaksud didapat dari sinyal rata-rata yang diperoleh saat kecepatan hampir konstan atau memliki percepatan yang tidak berubah.

Kelebihan :

1. Kecepatan putaran roda akan relatif stabil dalam kondisi lintasan yang berubah, dalam hal ini menanjak. Karena adanya penambahan kontrol pada putaran yang sedang terjadi.
2. Waktu tempuh relatif lebih cepat pada kondisi lintasan menanjak.
3. Terdapat variabel yang menampung untuk mengkontrol putaran roda.

Kekurangan :

1. Adanya jeda waktu(*laging*) untuk mengeksekusi pergerakan roda.
2. Perlunya variabel tambahan untuk menjaga kondisi pergerakan roda agar tetap stabil.



Gambar III.5 Bagan sistem perangkat lunak pergerakkan roda robot mobil setelah implementasi algoritma MPC

Pada gambar III.5 Bagan sistem perangkat lunak pergerakkan roda robot mobil setelah implementasi algoritma *Model Predictive Control* program pada mikro 2 berisi implementasi algoritma *Model Predictive Control* yang dirancang untuk menghitung data masukan dari roda yang berputar, lalu mengirimkannya ke mikro 1 secara paralel. Program pada mikro 1 berisi implementasi algoritma *Model Predictive Control* yang dirancang untuk membandingkan masukan serial dari sistem minimum mikrokontroller dengan masukan dari mikro 2 lalu mengatur lebar sinyal *Pulse Width Modulation*, kemudian data dikirim ke motor driver sehingga roda robot dapat berjalan stabil.

III.1.3.1 Analisis Keterkaitan Algoritma *Model Predictive Control* dengan metode *Pulse Width Modulation*

Pada gambar III.4 sebelumnya yang membahas mengenai flowchart algoritma *Model Predicitve Control* namun belum menjelaskan bagaimana proses atau cara menaikan meupun menurunkan sinyal agar memperoleh hasil putaran yang mendekati dengan yang diharapkan. Dengan membahas keterkaitan atau relasi antara algoritma *Model Predictive Control* dengan metode *Pulse Width Modulation* diharapkan dapat memperjelas bagaimana cara kerja maupun sistem yang akan digunakan pada simulasi perputaran roda robot mobil.

Analisis proses akan memberikan pengaturan kontrol proses dalam terminologi *Model Predictive Control*. *Input* dan *Output* proses digolongkan dalam 4 katagori yang berbeda. Penggolongan ini berdasarkan penggunaannya dalam pengontrolan proses[3], yaitu :

a. *Manipulated Variabel* (MV) – input proses yang diatur untuk menjaga output proses tetap pada setpoint.

   Berupa hasil putaran sebelumnya, untuk menjadi patokan atau set *point* agar mendapatkan perbaikan.

b. *Controller Variabel* (CV) – output proses yang ingin dijaga pada harga setpoint.

   Hasil putaran saat itu, putaran yang didapat akan disimpan untuk kemudian dibandingkan dengan putaran berikutnya.

c. *Disturbance Variabel* (DV) – input proses yang memberi efek terhadap proses dan output variabel tertentu.

Dapat berupa sudut elevasi pada lintasan, diameter roda pada robot mobil, dan bobot mobil.

d. *Constraint* (AV) – output proses yang harus dijaga dalam suatu definisi batas jangkauan operasi yang berlawanan dengan set *point*.

Proses input dan output dapat juga didefinisikan sebagai *teroptimisasi* jika pada proses input atau output tersebut terdapat sebuah pertimbangan ekonomis atau kinerja untuk menekan proses variabel dalam arah yang diinginkan hingga beberapa *constraint* (batasan) proses menjadi aktif. Dalam hal ini yaitu *Pulse Width Modulation (PWM),* dengan melihat kembali pada gambar III.4 flowchart algoritma *Model Predicitve Control* yaitu proses menaikan atau menurunkan sinyal. Agar didapatkan hasil putaran yang mendekati konstan. Jika pada simulasi algoritma *Model Predicitve Control* dengan cara melakukan penambahan kecepatan namun sedikit-sedikit atau tidak terlalu signifikan dalam melakukan perubahan kecepatan.

III.1.3.2 Analisis Perhitungan yang Berhubungan Dengan Keadaan Riil

Pada perancangan perangkat lunak simulasi algoritma *Model Predictive Control* memerlukan analisis mana-mana saja perhitungan yang dapat mewakili dengan keadaan sebenarnya. Begitupun juga pula pada algoritma *Model Predictive Control* yang akan di*embed* pada perangkat lunak simulasi ini. Jika melihat kasus lintasan yang menjadi bahasan pada tugas akhir skripsi berupa tanjakan, maka pasti ada gaya ke belakang robot mobil beroda karena benda tersebut mempunyai berat, dengan memiliki persamaan yaitu :

$$F_{blk} = m.g.\cos \alpha$$

F_{blk} = gaya ke belakang sebuah benda

m = bobot benda

g = gravitasi

α = sudut elevasi tanjakan

sebagai contoh, diketahui :

m = 1kg

g = 32.6

$\alpha = 15^{0}$

F_{blk} = 1 x 32.6 x 15

F_{blk} = 489 *Newton*

jika diambil dalam kondisi nyata, maka akan ada gaya gesek, yang arahnya melawan arah benda, dengan rumus gaya gesekan :

$$f_k = \mu_k.m.g.\sin \alpha$$

f_k = gaya gesek

μ_k = koefisien gesek (0,2)

m = massa benda

g = gravitasi

α = sudut elevasi tanjakan

namun dapat pula pada simulasi algoritma *Model Predictive Control* tidak dikerjakan gaya apapun, atau memang pada algoritma *Model Predictive Control* tidak memperhitungkan gaya yang terjadi. Karena memang tidak ada masukan berupa tenaga atau usaha agar robot mobil beroda dapat melaju dan memang tidak ada alat atau media yang dapat terpasang dan menghitung usaha yang sedang terjadi. Karena hal ini diluar cakupan kontrol algoritma *Model Predictive Control*. Perhitungan percepatan laju roda dapat diwakili oleh perubahan sudut, dengan rumus percepatan benda yaitu :

$$a = \frac{-F_{blk} - f_k}{m} = -g(\cos\alpha + \mu_k \sin\alpha)$$

a = percepatan

g = gravitasi

α = sudut elevasi tanjakan

μ_k = koefisien gesek (0,2)

maka kecepatan akhir mobil tersebut bisa didapat menggunakan rumus GLBB :

$v_t = v_0 - a.t$ atau $v_t^2 = v_0^2 - 2.g.s$ dimana s = h . sin α

Vt = kecepatan pada waktu tertentu

V_0 = kecepatan awal

s = jarak; h = tinggi

t = waktu

Dalam kondisi nyata bobot merupakan faktor yang dapat memengaruhi cepat atau lambatnya sebuah benda dapat melaju. Namun dapat pula bobot sebuah benda tidak memengaruhi dalam perhitungan pada simulasi. Dengan asumsi benda atau robot mobil beroda pada perangkat lunak simulasi adalah benda titik. Hal ini dapat diterapkan pada simulasi berupa piksel, dengan menganggap masukan pada perhitungan diwakili oleh piksel yang sedang dilalui. Selanjutnya faktor yang kurang lebih juga memengaruhi perhitungan adalah diameter roda. Jika akan memasukkan diameter roda didalam perhitungan perangkat lunak simulasi, maka langkah selanjutnya harus memperhitungkan torsi perputaran roda dengan anggapan sebagai berikut :

1. Roda robot mobil yang akan disimulasikan ada empat buah atau berapapun dengan ukuran yang sama pada masing-masing roda.
2. Roda dianggap berbentuk silinder dengan jari-jari yang diwakilkan oleh R.
3. Satu-satunya gaya yang bekerja pada setiap roda hanyalah gaya gesekan :

   $f_k = \mu_k.N = \mu_k.M.g.\sin \alpha$, dengan M adalah massa total benda

4. Setiap roda mendapatkan beban yang sama dari bobot robot mobil, atau setiap roda mendapatkan bobot mobil dibagi 4, jadi :

$f_k = \mu_k.1/4\ N = ¼\ \mu_k.M.g.\sin\alpha$

Jadi jika memperhitungkan semua torsi yang bekerja pada keempat roda, menjadi:

$$\Sigma\tau = I.\alpha$$

$$4.f_k.R = 4.(\frac{1}{2}mR^2).\frac{a}{R}$$

$$(-\frac{1}{4}.\mu_k.M.g.\sin\alpha).R = \frac{1}{2}mR^2.\frac{a}{R}$$

$$a = -\frac{1}{2}.\frac{M}{m}.\mu_k.g.\sin\alpha$$

Sisanya untuk mencari kecepatan robot mobil beroda dapat menggunakan rumus GLBB seperti di bawah ini :

$v_t = v_0 - a.t$ atau $v_t^2 = v_0^2 - 2.g.s$ dimana s = h sin α

Vt = kecepatan pada waktu tertentu

V_0 = kecepatan awal

α = sudut elevasi

t = waktu

ternyata jika melibatkan ukuran diameter roda, massa total mobil dan massa roda harus diperhitungkan selain gaya gesek dan sudut elevasinya. Hal ini dapat menjadikan rumit, karena harus memperhitungkan titik berat benda dan jarak titik pusat kedua roda. Maka untuk memudahkan pembacaan informasi serta menjalankan yang hanya dapat berhubungan dengan algoritma yang berada pada pergerakan roda robot mobil dibuatkanlah analisis yang berisi parameter namun

dari luar sistem perputaran roda untuk kemudian diterapkan pada perangkat lunak simulasi.

III.1.3.2 Analisis Parameter Eksternal yang Digunakan pada Simulasi

Pada gambar 3.4 flowchart implementasi algoritma *Model Predictive Control* terdapat simbol – simbol masukan berupa data yang memerlukan penjelasan yaitu berupa faktor perhitungan lain yang berkaitan dengan putaran roda, salah satunya yaitu perhitungan kecepatan berdasarkan besar kecilnya diameter pada roda robot. Berikut merupakan rumus yang dapat menghasilkan kecepatan dengan diketahui di awal dari banyaknya putaran per detik, dan diameter roda.

$$V = \varphi * \pi * rps$$

Keterangan :

V = *Velocity* atau Kecepatan

φ = d*iameter* atau diameter roda

π = konstanta dengan nilai 22/7 atau 3.14

rps = *rotational per second* atau putaran per detik

sebagai contoh, diketahui :

φ = 1,8 cm

π = 22/7 atau 3.14

rps = 2 rps

dengan menggunakan rumus

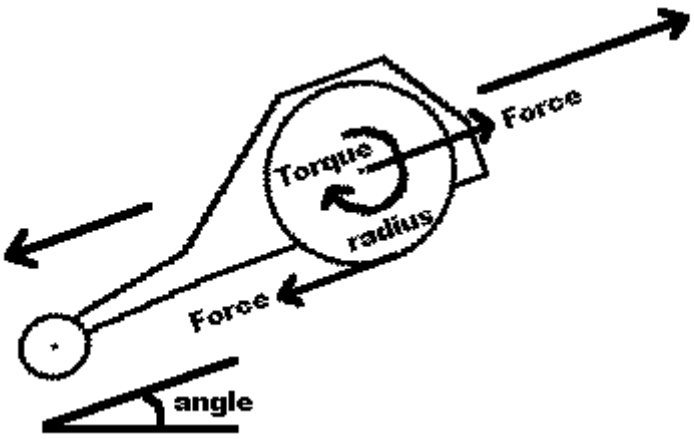
$V = \varphi * \pi * rps$

maka :

V = 1,8 x 3,14 x 2

V = 11,304 cm/s

Untuk melakukan perhitungan mengukur perubahan putaran yang terjadi pada roda robot juga dapat berdasarkan sudut elevasi dengan menggunakan rumus perhitungan seperti berikut.



Gambar III.6 **P**arameter lain yang dapat membantu informasi saat simulasi

$$A = V^2 * \sin((\alpha * \pi) / 180)$$

Keterangan :

A = *Acceleration* atau Percepatan

V = *Velocity* atau Kecepatan

α = *angle* atau sudut elevasi

π = *phi*, konstanta dengan nilai 22/7 atau 3.14

sebagai contoh, diketahui :

V = 1,8

$\alpha = 15^0$

π = 22/7 atau 3.14

dengan menggunakan rumus

$$A = V^2 * \sin((\alpha * \pi) / 180)$$

maka :

$$A = V^2 * \sin((\alpha * \pi) / 180)$$

$$A = 1{,}8^2 \text{ x sin } ((15 \text{ x } 3{,}14) / 180)$$

$$A = 3{,}24 \text{ x sin } (47{,}1/180)$$

$$A = 3{,}24 \text{ x sin } 0{,}26$$

$$A = 0{,}015 \text{ cm/s}^2$$

III.1.3.3 Analisis Korelasi Satuan yang Digunakan pada Simulasi

Analisis satuan yang digunakan pada kejadian sebenarnya adalah satuan yang umum digunakan dan sesuai dengan *environment* atau hal-hal sekitarnya, seperti halnya dengan penggunaan satuan untuk kecepatan benda adalah cm/s. Menggunakan sentimeter per detik dikarenakan *environment* sekelilingnya lintasan yang dilalui tidak lebih jauh dari satuan meter bahkan kilometer. Jarak yang dilalui tidak mencapai angka puluhan meter ataupun kilometer, maka tepat dan efisien dengan menggunakan satuan cm/s saja.

Dalam penerapan kedalam simulasi perangkat lunak. Didalam layar yang digunakan juga bisa dikatakan tidak efisien jika menggunakan satuan cm/s atau sentimeter per detik. Hal ini bisa dianalisis dengan melihat layar yang digunakan pada saat simulasi memiliki lebar yang terbatas. Maka agar satuan kecepatan pada saat simulasi tepat dan efisien jika menggunakan satuan piksel, sesuai dengan karakteristik layar. Satuan kecepatan yang akan digunakan menggantikan satuan jarak pada kondisi nyata diwakili dengan satuan piksel per detik.

Tabel III-2 Korelasi Satuan antara Riil dengan Simulasi

Satuan	**Kondisi *real***	**Layar**
Jarak	Sentimeter	Piksel
Kecepatan	Cm / s	Piksel / detik
Percepatan	Cm / s^2	$Piksel / detik^2$

Dengan layar antarmuka, dan spesifikasi yang dimiliki oleh perangkat keras yang digunakan pada pembangunan perangkat lunak memiliki karakteristik sebagai berikut :

Resolusi : 1366 x 768 piksel

Kedalaman : 96 *Pixel Per Inch* (*PPI*)

Pada Kedalaman 96 *PPI* dengan artian bahwa setiap 1 *inch* pada layar memiliki 96 piksel, namun satuan jarak pada perangkat lunak yang akan digunakan ialah sentimeter. Perlu adanya perubahan satuan agar penggunaan jarak pada layar dapat mendekati keadaan nyata.

1 *inch* ≈ 2,54 sentimeter

1 *inch* ≈ 96 piksel

$$\frac{96\ piksel}{2.54\ sentimeter} = 37{,}8 \text{ piksel per sentimeter}$$

Maka didapat kedalaman layar dengan 37,8 *PPC* atau setara dengan 96 *PPI.* Didalam penggunaan perangkat lunak perhitungan jarak, kecepatan, dan percepatan sudah dapat menggunakan satuan sentimeter, namun diharuskan memberikan pembagi pada setiap hasil piksel dengan nilai 37,8.

III.1.4 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan bagian penting lainnya agar simulasi perangkat lunak dapat berjalan dengan baik. Untuk mengetahui elemen-elemen yang berhubungan dengan sistem yang sedang berjalan.

III.1.4.1 Kebutuhan Karakteristik Perangkat Keras Pembangun Simulasi

Perangkat keras pendukung dalam membangun simulasi perangkat lunak ini membutuhkan seperangkat komputer dengan spesifikasi minimum :

1. *Processor* 1.7 *Ghz*
2. *RAM* 1 *GB.*
3. *Harddisk* 80 *GB.*
4. Monitor dengan resolusi minimum 1366 x 768 piksel

III .1.4.2 Kebutuhan Karakteristik Perangkat Lunak Pembangun Simulasi

Perangkat lunak pendukung dalam membangun perangkat lunak simulasi algoritma *Model Predictive Control* adalah :

1. Sistem operasi *windows xp sp2*.
2. *Borland delphi* 7.

III.1.4.3 Kebutuhan Karakteristik Perangkat Keras Pengguna Simulasi

Perangkat keras pendukung agar dapat menggunakan simulasi perangkat lunak algoritma *Model Predictive Control* kurang lebih membutuhkan seperangkat komputer dengan spesifikasi minimum :

1. *Processor intel* 1.7 *Ghz*.
2. *RAM* 512 *MB*.
3. *Harddisk 20GB.*
4. Monitor dengan resolusi minimum 1366 x 768 piksel

III.1.4.4 Kebutuhan Karakteristik Perangkat Lunak Pengguna Simulasi

Perangkat lunak pendukung agar dapat menggunakan simulasi perangkat lunak algoritma *Model Predictive Control* dengan spesifikasi minimum sistem operasi *windows xp sp2*.

III.1.4.5 Karakteristik Pengguna

Dalam menggunakan simulasi perangkat lunak dibutuhkan pengguna yang sudah berpengalaman dalam menggunakan komputer atau yang telah mengikuti *training* simulasi perangkat lunak ini dengan karakteristik sebagai berikut :

Tabel III.3 Karakteristik Pengguna

Pengguna	**Tanggung Jawab**	**Hak Akses**	**Tingkat Pendidikan**
Umum	--	berinteraksi dengan perangkat lunak	minimal setara SMA

III.1.5 Analisis Kebutuhan Fungsional

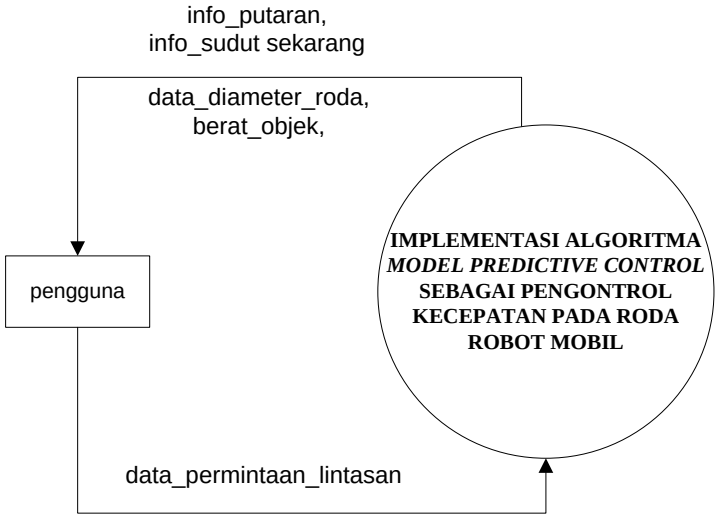
Analisis kebutuhan fungsional ini meliputi *Entity Relationship Diagram(ERD)* dan Data *Flow Diagram* (*DFD*). *Entity Relationship Diagram* menjelaskan kardinalitas antara entitas – enitas dan menjelaskan pula relasi antar entitas – entitas yang berada pada simulasi perangkat lunak algoritma *Model Predictive Control*. Data *Flow* Diagram berikut ini merupakan alat yang dapat menggambarkan suatu proses di dalam sistem secara terstruktur dan jelas, serta dokumentasi dari perangkat lunak yang baik. *DFD* suatu level dapat diturunkan menjadi *level* berikutnya jika memang terdapat proses yang masih dapat dijelaskan.

III.1.5.1 ERD Perangkat Lunak Implementasi Algoritma MPC

Pembangunan perangkat lunak implementasi algoritma *Model Predictive Control* tidak menghasilkan tabel. Data akan ada jika proses perhitungan dilakukan yaitu disaat proses menghitung kecepatan, percepatan dan sudut lintasan.

III.1.5.2 Diagram Konteks Perangkat Lunak Implementasi Algoritma MPC

Untuk menggambarkan analisis perangkat lunak secara umum digunakan alat bantu yaitu diagram konteks dan *data flow diagram*. Diagram konteks dari perangkat lunak simulasi algoritma *Model Predictive Control* yang akan dibangun adalah sebagai berikut :



Gambar III.8 Diagram Konteks implementasi algoritma *Model Predicitve Control*

III.1.5.3 DFD Level 1

Pada DFD Level 1 akan dibagi menjadi tiga garis besar proses utama, yaitu pemilihan lintasan, pemilihan diameter roda, dan simulasi.

1. Pemilihan lintasan

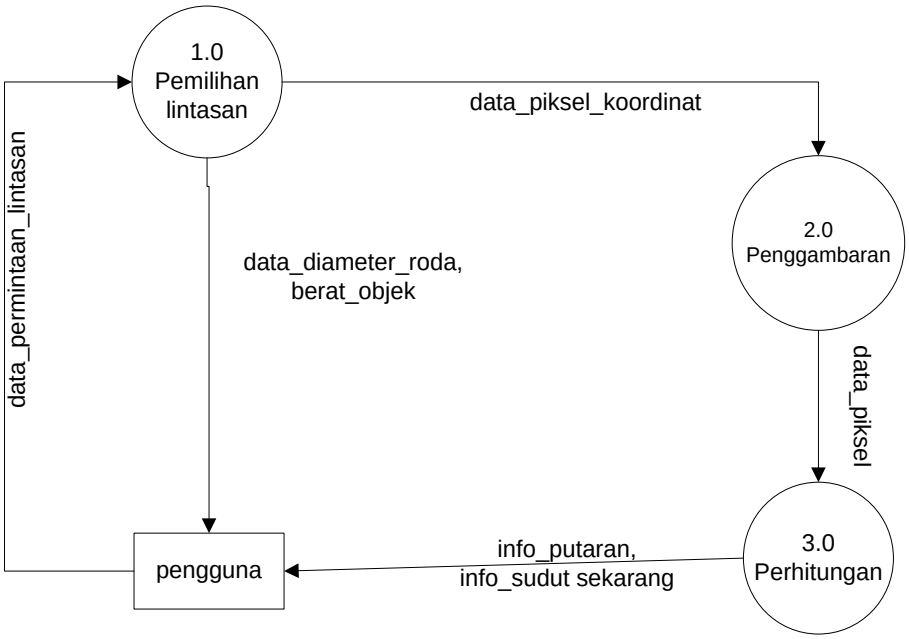
   Dimana dengan memilih lintasan, dengan sekaligus pengguna memilih sudut yang akan diperlihatkan pada simulasi perangkat lunak.

2. Penggambaran

   Menginputkan jika akan melakukan perubahan diameter roda, dan melakukan perubahan bentuk berupa letak titik koordinat pada kanvas.

3. Perhitungan

   Dalam proses ini akan menampilkan perubahan pergerakan setelah diameter roda diatur, dan memilih lintasan agar mendapatkan informasi yang terjadi dari perbandingan antara algoritma *Model Predictive Control* dengan algoritma robot konvensional. hasil berupa kecepatan, akselerasi, dan sudut yang sedang dilaluiakan ditampilkan sebagai bahan informasi.



Gambar III.9 DFD Level 1

Berikut adalah spesifikasi proses dari DFD untuk menjelaskan perilaku – perilaku yang ada didalam DFD secara lebih rinci. Berikut spesifikasi proses pada simulasi perangkat lunak algoritma *Model Predictive Control* yang dapat dilihat pada tabel III.4:

Tabel III.4 Spesifikasi Proses DFD level 1

no	**Spesifikasi**	**Keterangan**
1	No	1.0
	Nama	Pemilihan lintasan
	Input	data_permintaan_lintasan
	Output	data piksel koordinat, data_diameter_roda, berat_objek
	Logika proses	- Pengguna akan memilih lintasan berdasarkan nama file:sudut_algoritma.*coaster* - Pengguna mendapatkan data diameter roda dan berat objek.
2	No	2.0
	Nama	Penggambaran

no	Spesifikasi	Keterangan
	Input	data_piksel_koordinat
	Output	data_piksel
	Logika proses	Setelah pemilihan lintasan berdasarkan file yang dibuka, akan melakukan proses penggambaran pada kanfas.
3	No	3.0
	Nama	Perhitungan
	Input	data_piksel
	Output	Info_putaran, info_sudut_sekarang
	Logika proses	Setiap perubahan pergerakan piksel akan dihitung, baik dari sudut elevasi yang sedang dilalui, kecepatan pergerakan setiap satuan waktu.

III.1.5.5 Kamus Data

Kamus data dapat membantu mendefinisikan dengan lengkap data yang mengalir diantara proses, penyimpanan data, dan entitas luar. Data yang mengalir tersebut dapat berupa masukan untuk sistem atau hasil dari proses sistem. Kamus data dibuat berdasarkan arus data yang mengalir pada konteks diagram dan DFD.

Tabel III.5 Kamus data pada simulasi perangkat lunak implementasi algoritma *MPC*

No	Data	Keterangan
1	Nama	data_permintaan_lintasan
	Struktur data	file berekstensi *.coaster* dengan format penamaan file sudut_algoritma.coaster
2	Nama	data_diameter_roda
	Struktur data	*cart*y
	Deskripsi	berasal dari pemilihan lintasan yang sudah tersimpan
3	Nama	berat_objek
	Struktur data	*mass*
	Deskripsi	berasal dari pemilihan lintasan yang sudah tersimpan
4	Nama	data_piksel_koordinat

No	Data	Keterangan
	Struktur data	x, y
	Deskripsi	titik satu-satu sebelum memulai penggambaran, baik penggambaran lintasan maupun roda robot mobil
5	Nama	data_piksel
	Struktur data	x, y
	Deskripsi	titik saat perhitungan piksel, digunakan saat menghitung sudut, kecepatan dan percepatan
6	Nama	info_putaran
	Struktur data	v,v*max*,v*min*,a,a*max*,a*min*,
	Deskripsi	hasil perhitungan piksel sementara saat roda robot mobil bergerak
7	Nama	info_sudut_sekarang
	Struktur data	*theta*
	Deskripsi	hasil perhitungan piksel sementara saat roda robot mobil bergerak

III.2 Perancangan Sistem

Perancangan merupakan tahap lanjutan setelah tahap analisis perangkat lunak dimana pada perancangan perangkat lunak digambarkan rancangan perangkat lunak yang akan dibangun sebelum dilakukan pengkodean kedalam bahasa pemrograman.

III.2.2 Perancangan Data

Perancangan data terdiri dari perancangan diagram relasi berdasarkan analisis basis data dan struktur tabel yang dibangun dalam sistem.

III.2.2.1 Diagram Relasi

Pada pembangunan simulasi perangkat lunak algoritma *Model Predictive Control* tidak ditemui adanya tabel data hasil perhitungan yang memiliki relasi khusus. Dikarenakan menghasilkan satu tabel hasil perhitungan saja dimana isi

dari tabel tersebut hanya berisi dan jika hanya pada pengguna telah melakukan perhitungan kecepatan, percepatan, dan sudut lintasan.

III.2.2.2 Struktur Tabel

Tabel III.6 adalah struktur tabel berdasarkan analisis basis data. Meskipun pada simulasi perangkat lunak tidak menyimpan struktur tabel, namun ada hasil perhitungan yang dimasukan kedalam perangkat lunak maka pada tabel III.6 akan menjelaskan tipe data dari hasil perhitungan simulasi perangkat lunak.

Tabel III.6 struktur tabel simulasi perangkat lunak

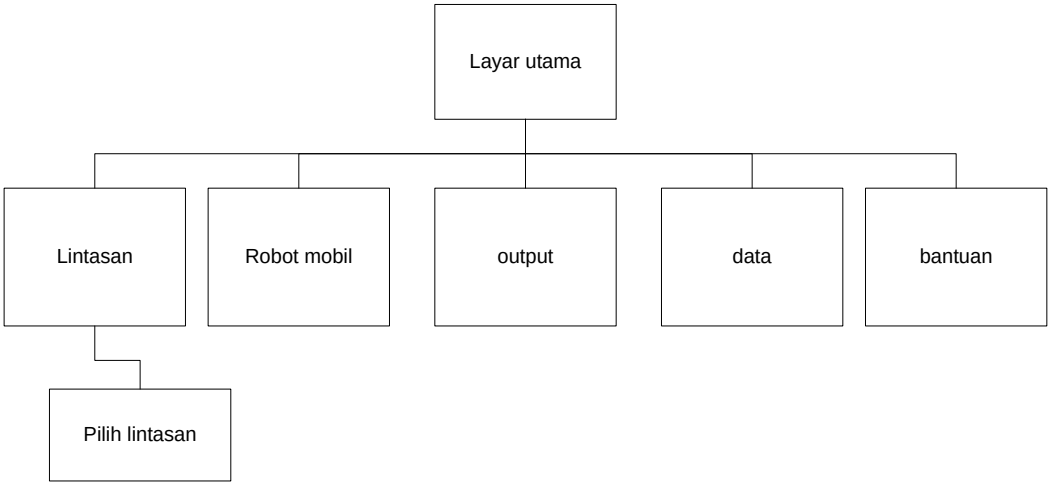
Nama field	Tipe	Panjang	Keterangan
x	*float*		
y	*float*		
mass	*float*		bobot objek
*car*ty	*float*	[2 .. 20]	*default* = 18, diameter roda
v,v*max*,v*min*	*float*		kecepatan
a,a*max*,a*min*	*float*		percepatan
theta	*float*		sudut
dist	*float*		jarak
time	*float*		*temporary* waktu

III.2.2.3 Perancangan Arsitektur

Perancangan arsitektur terdiri dari perancangan struktur menu, perancangan antarmuka dan jaringan semantik pada sistem yang dirancang.

III.2.2.3.1 Perancangan Struktur Menu

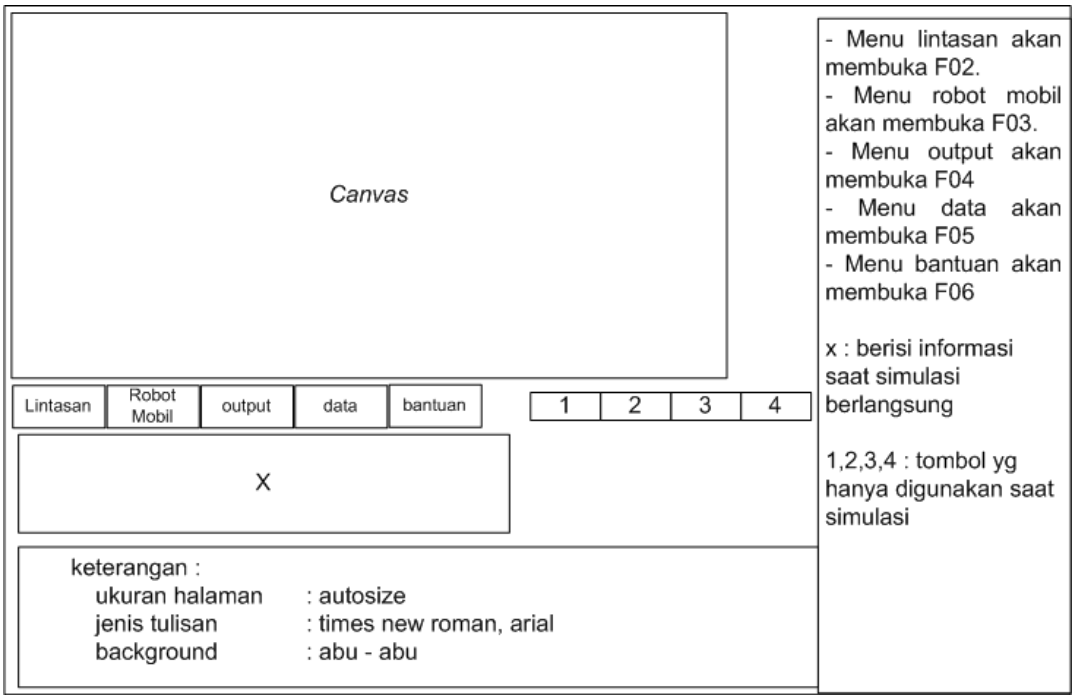
Perancangan struktur menu yang terdapat pada perangkat lunak simulasi algoritma *Model Predictive Control* dapat dijelaskan pada gambar III.10 perancagan struktur menu.



Gambar III.10 Perancangan Struktur Menu

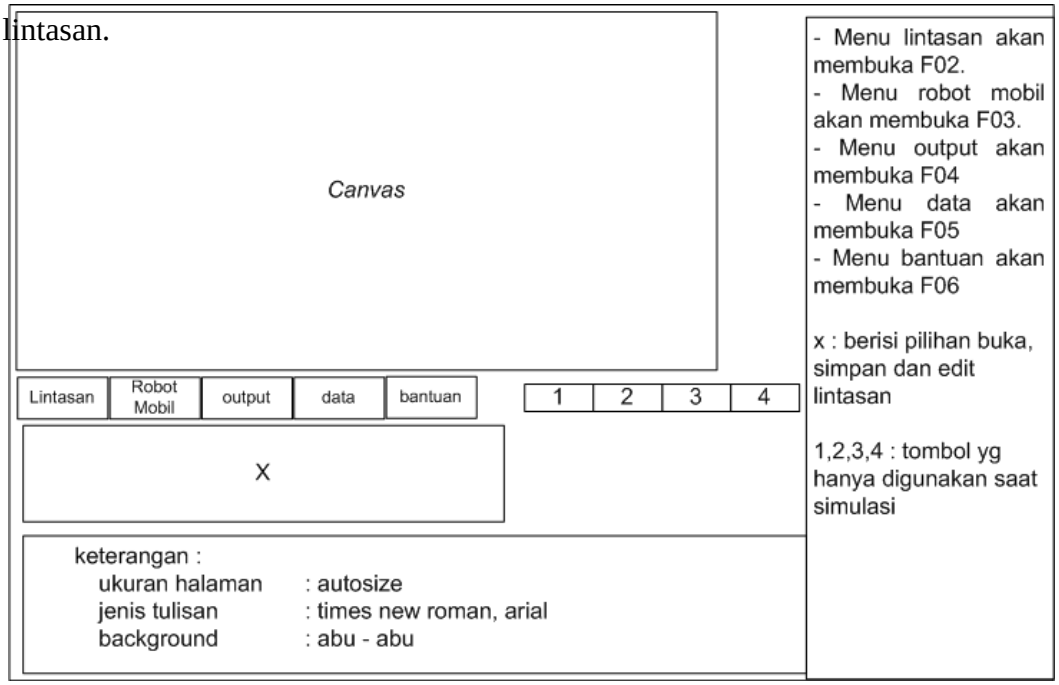
III.2.2.3.2 Perancangan Antarmuka

F01 Antarmuka layar utama. Gambar III.11 adalah perancangan antarmuka layar utama.



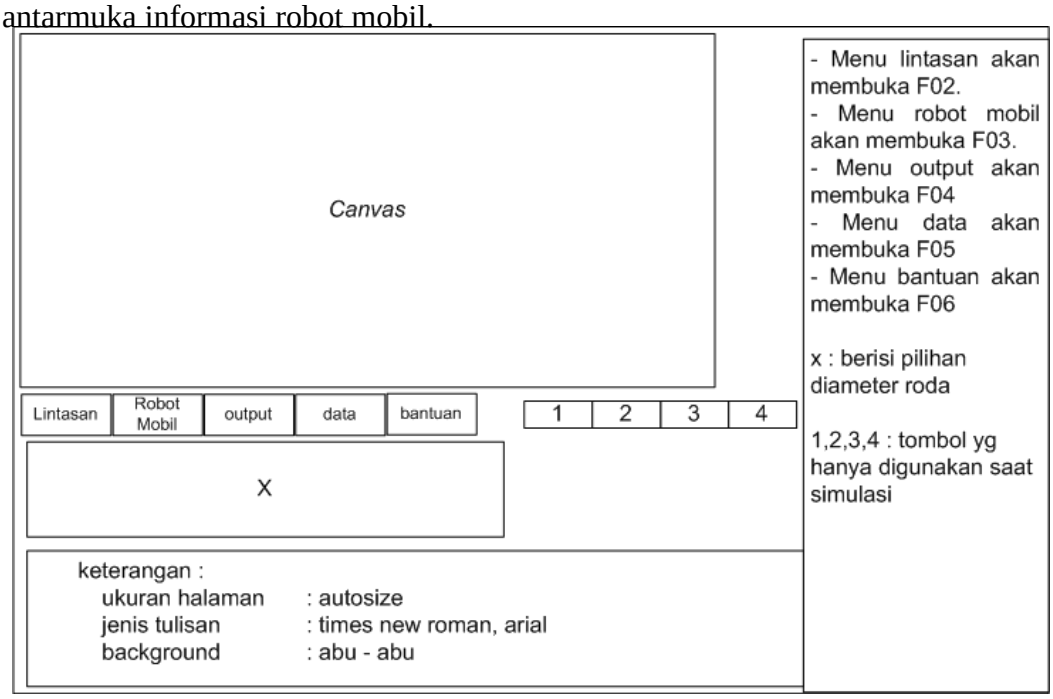
Gambar III.11 Antarmuka layar utama

F02 Menu pilih lintasan. Gambar III.12 adalah perancangan antarmuka pilih lintasan.



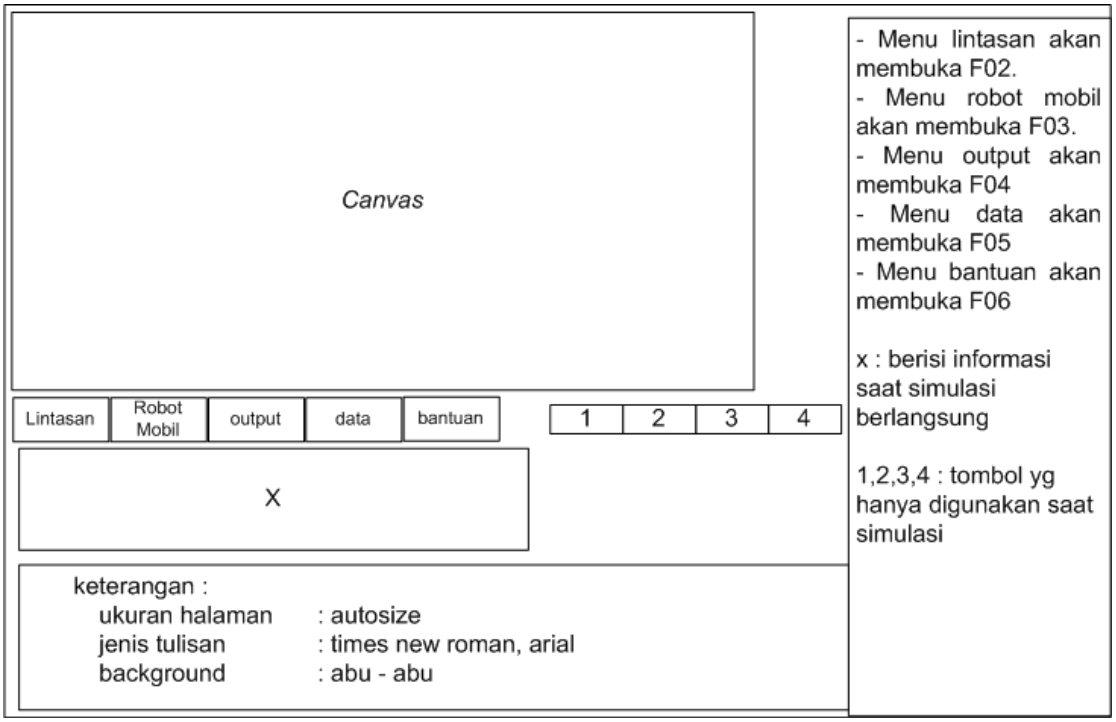
Gambar III.12 Antarmuka pilih lintasan

F03 Antarmuka infomasi robot mobil. Gambar III.13 adalah perancangan antarmuka informasi robot mobil.



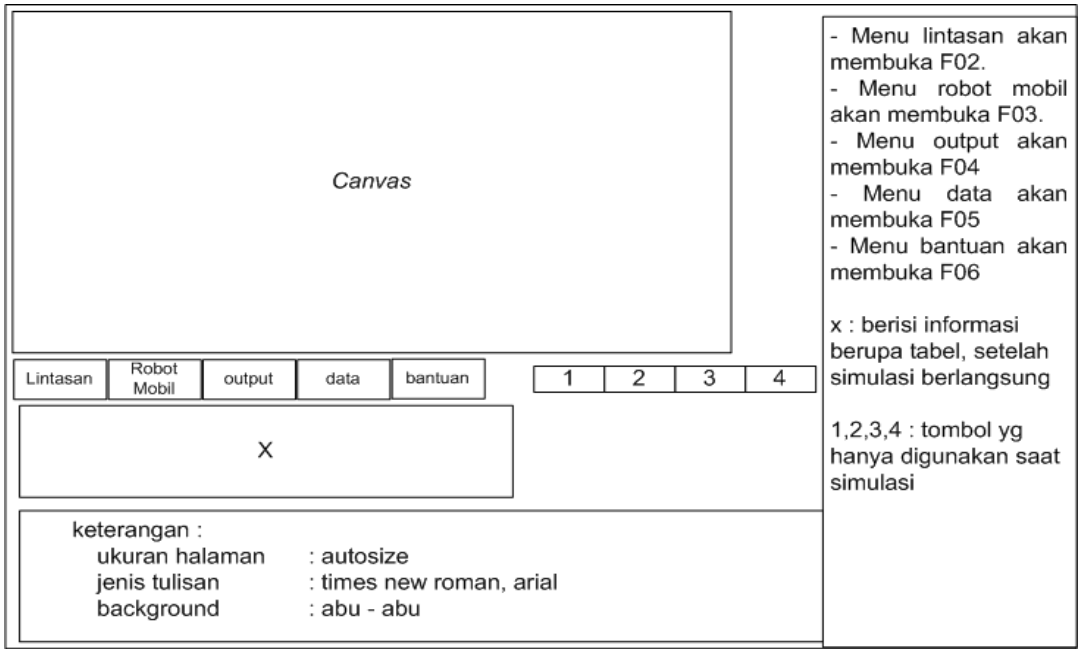
Gambar III.13 Antarmuka informasi robot mobil

F04 Antarmuka hasil *output* dari *canvas* yang sedang digunakan. Gambar III.14 adalah perancangan antarmuka *output* berupa data *realtime*.



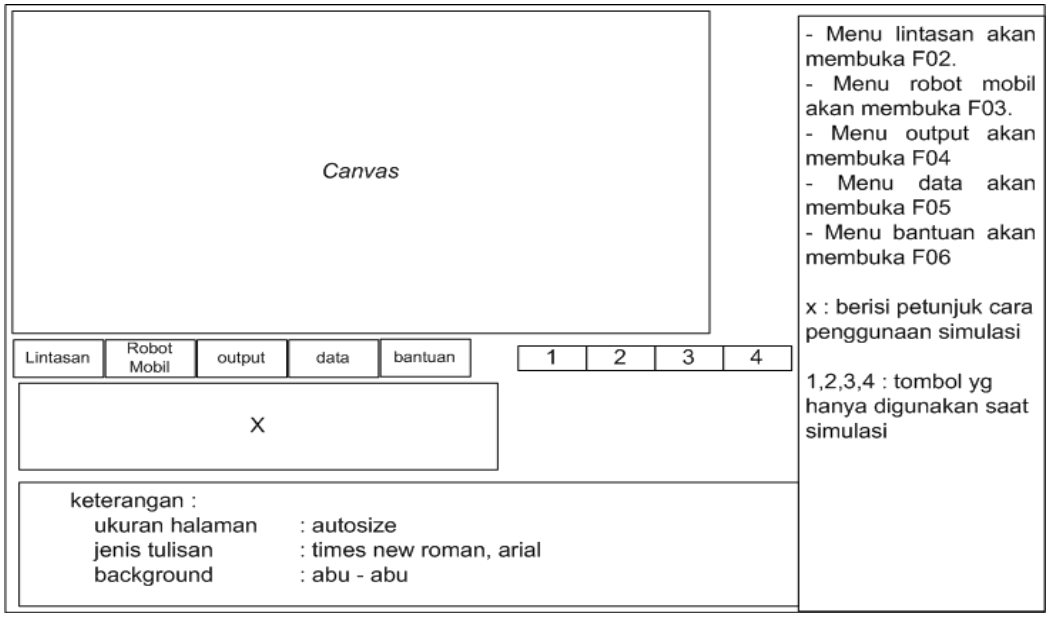
Gambar III.14 Antarmuka *output*

F05 Antarmuka Menu Data. Gambar III.15 adalah perancangan antarmuka menu data.



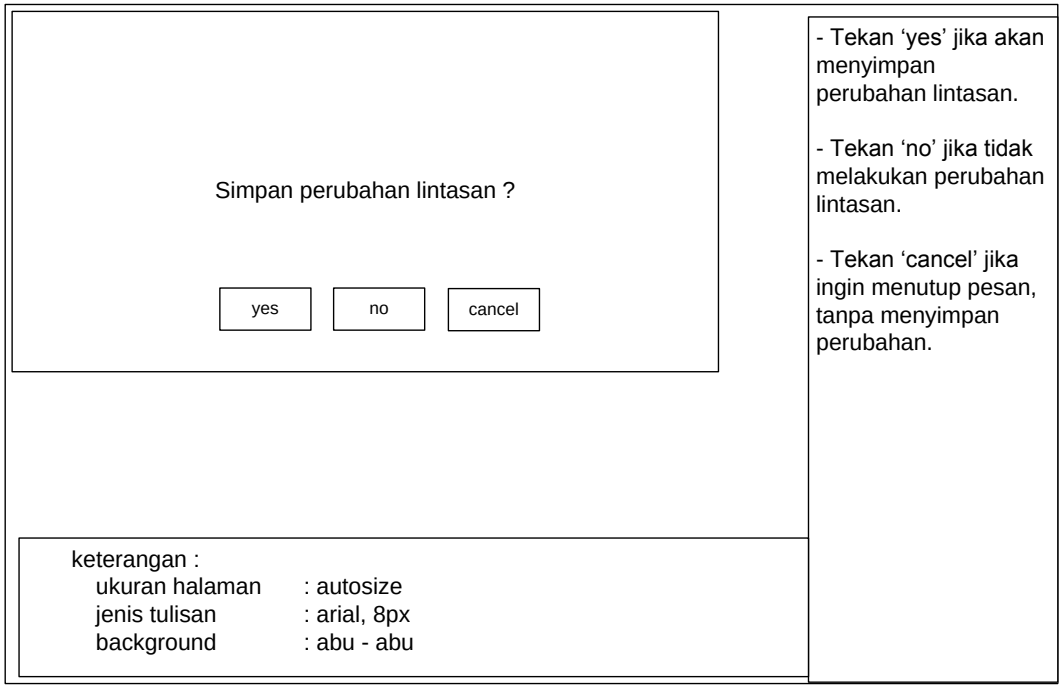
Gambar III.15 Antarmuka menu data

F06 Antarmuka Menu Bantuan. Gambar III.16 adalah perancangan antarmuka menu bantuan.



Gambar III.16 Antarmuka Bantuan

M01 Antarmuka pesan simpan perubahan. Gambar III.17 adalah perancangan antarmuka pesan simpan perubahan lintasan.



Gambar III.17 Antarmuka pesan simpan perubahan

III.2.2.3.3 Jaringan Semantik

Dengan keterangan :

F01 : layar utama

F02 : antarmuka pilih lintasan

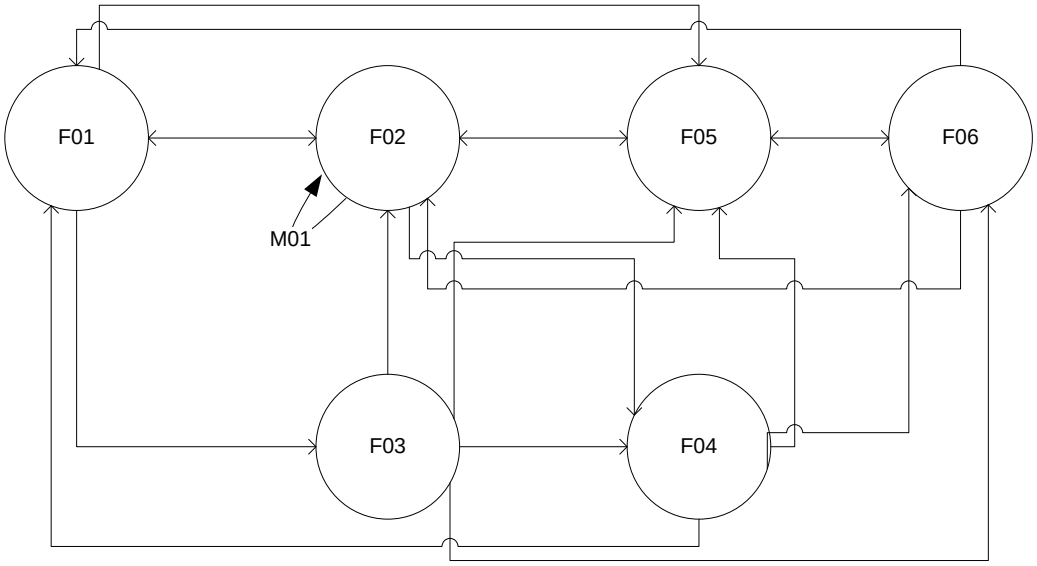
F03 : antarmuka informasi robot mobil

F04 : antarmuka informasi output dari kanvas

F05 : antarmuka menu data

F06 : antarmuka menu bantuan

M01 : pesan simpan perubahan



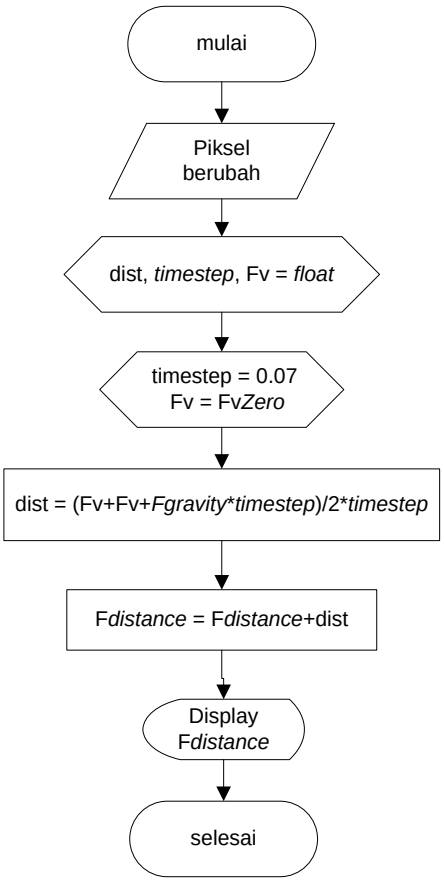
Gambar III.18 Jaringan semantik perangkat lunak

III.2.3 Perancangan Prosedural

Perancangan prosedural merupakan perancangan yang dilakukan untuk menetapkan detail algoritma yang akan dinyatakan ke dalam suatu perangkat lunak. Adapun perancangan prosedural untuk perangkat lunak yang akan dinyatakan ke dalam suatu perangkat lunak.

III.2.3.1 Tahap Perhitungan Jarak

Gambar III.19 adalah algoritma yang digunakan untuk flowchart perhitungan jarak.



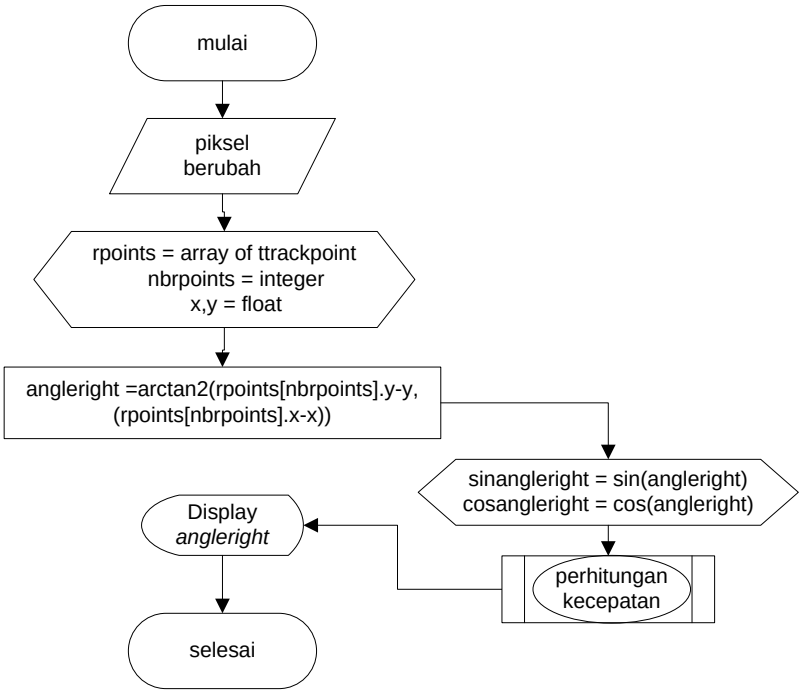
Gambar III.19 flowchart algoritma perhitungan jarak

Keterangan gambar III.19:

Dist : jarak perubahan piksel Timestep : waktu berubah Fv : kecepatan awal Fgravity : nilai grafitasi Fdistance : total jarak

III.2.3.2 Tahap Perhitungan Sudut

Gambar III.20 adalah algoritma yang digunakan untuk flowchart perhitungan sudut.



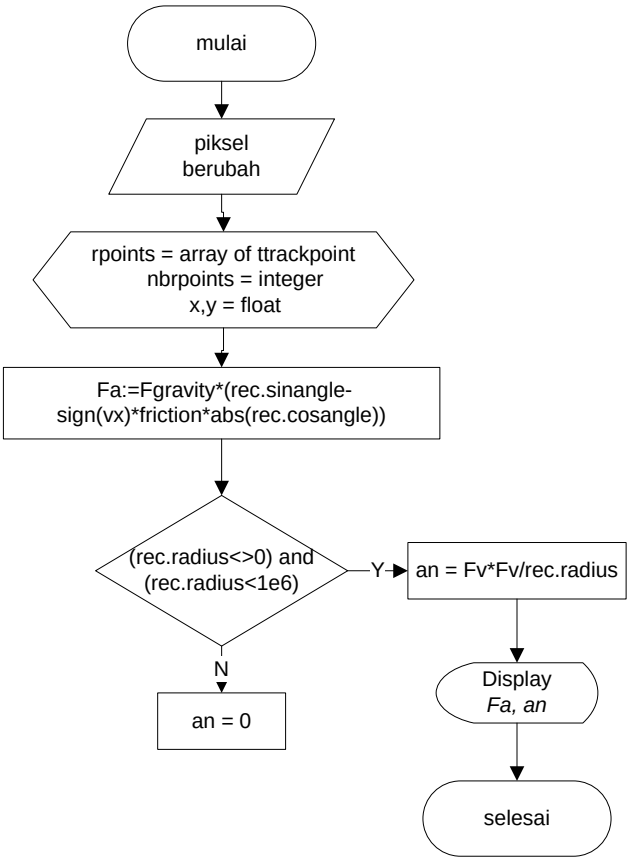
Gambar III.20 flowchart algoritma perhitungan sudut

Keterangan gambar III.20 :

Rpoints : array untuk kumpulan titik objek
Nbrpoints : titik objek bergerak
x,y : titik-ttik koordinat
arctan2 : fungsi untuk mengetahui sudut titik koordinat dengan fungsi tan
angleright : sudut objek sebelah kanan
sin(angleright) : fungsi untuk mengetahui titik koordinat dengan fungsi sinus
cos(angleright) : fungsi untuk mengetahui titik koordinat dengan fungsi cosinus

III.2.3.3 Tahap perhitungan percepatan

Gambar III.21 adalah algoritma yang digunakan untuk flowchart perhitungan percepatan.



Gambar III.21 flowchart algoritma perhitungan percepatan

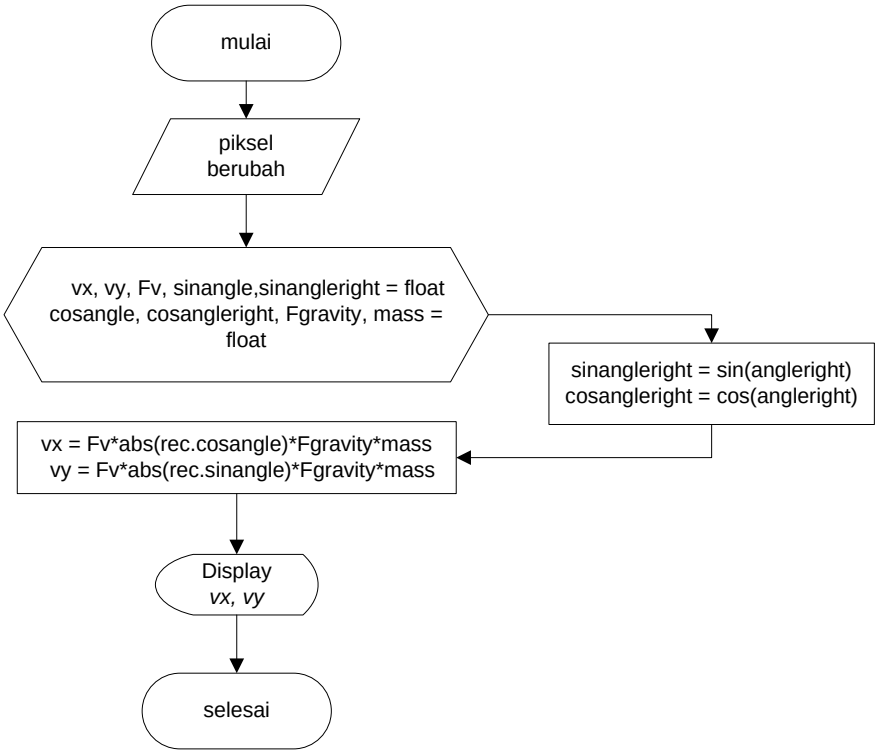
Keterangan gambar III.21 :

Rpoints : array untuk kumpulan titik objek
Nbrpoints : titik objek bergerak
x,y : titik-ttik koordinat
arctan2 : fungsi untuk mengetahui sudut titik koordinat dengan fungsi tan
angleright : sudut objek sebelah kanan
sin(angleright) : fungsi untuk mengetahui titik koordinat dengan fungsi sinus

cos(angleright) : fungsi untuk mengetahui titik koordinat dengan fungsi cosinus

III.2.3.4 Tahap perhitungan kecepatan

Gambar III.22 adalah algoritma yang digunakan untuk flowchart perhitungan kecepatan.



Gambar III.22 flowchart algoritma perhitungan kecepatan

Keterangan gambar III.22 :

Mass : bobot objek bergerak Fv : kecepatan awal Fgravity : nilai grafitasi angleright : sudut objek sebelah kanan sin(angleright) : fungsi untuk mengetahui titik koordinat dengan fungsi trigonometri *sinus* cos(angleright) : fungsi untuk mengetahui titik koordinat dengan fungsi trigonometri *cosinus*