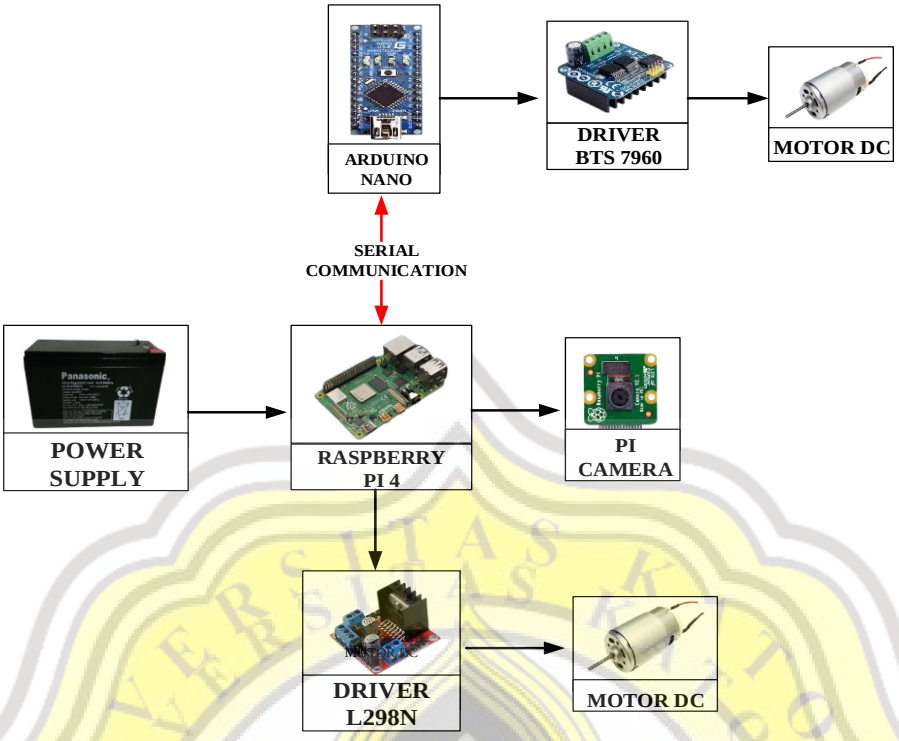
BAB III
PERANCANGAN ALAT

3.1. Pendahuluan

Bab ini akan menjelaskan perihal *wiring diagram* pada *autonomous car,* proses kinerja *autoonomous car* yang dijelaskan dengan *flowchart*, kemudian pola lintasan yang dipakai *autonomous car*, dan yang terakhir membuat metodologi penelitian.

3.2. Wiring Diagram

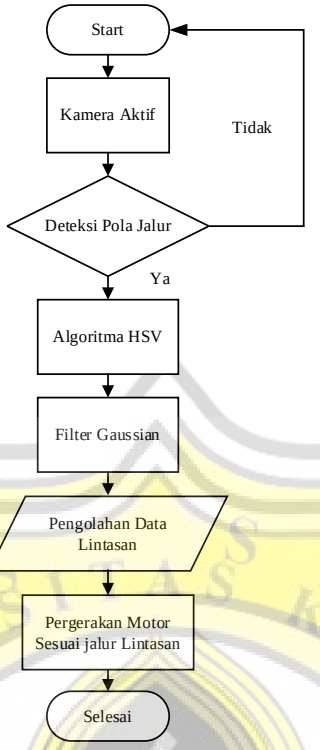
Pada proses *wiring* yang dilakukan seperti pada Gambar 3.1 yang perlu difokuskan yaitu komponen yang digunakan pada saat perakitan *autonomous car*. Kontroler dan mikrokontroler yang di pakai adalah mini computer Raspberry Pi 4, Arduino Nano, BTS7960 dan Driver L298N. Driver L298N digunakan untuk mengontrol motor DC sebagai penggerak roda pada *autonomous car*. Untuk penggerak *steering* dikontrol menggunakan BTS7960 yang terintegrasi langsung dengan program Arduino Nano. Kamera Raspberry juga di kontrol oleh Raspberry Pi 4 tersebut. Beberapa komponen tersebut akan dikontrol menggunakan Raspberry Pi 4 sebagai kontroler paling utama. Dalam perancangan prototipe pada *autonomous car* menggunakan catu daya berupa *accu* dengan tegangan 12V kapasitas 12AH sebagai sumber tegangan Raspberry Pi. Perancangan skematik ini meminimalisir apabila adanya terjadi kesalahan pemasangan komponen dan mempermudah ketika merakit komponen ini hingga menjadi *autonomous car.*



Gambar 3.1 *Wiring Diagram*

3.3. Proses Kerja *Autonomous Car*

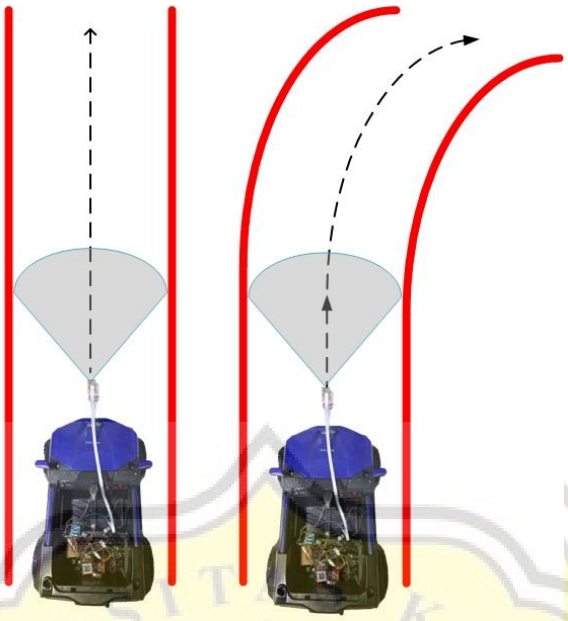
Langkah awal pada *flowchart* kamera aktif dalam bentuk video. Selanjutnya kamera mendeteksi pola jalur. Jika pola tidak terdeteksi maka program tidak berjalan dan akan kembali ke proses awal. Jika pola terdeteksi, maka proses algoritma *HSV* bekerja dimana *HSV* ini mampu mengolah ataupun memfilter warna yang dibutuhkan selanjutnya *filter gaussian* untuk menghaluskan atau menghilangkan noise pada proses deteksi garis tepi jalur agar hasil lebih baik. Kemudian proses selanjutnya ketika sudah berhasil di filter, hasil dari pembacaan akan diinterkoneksikan dengan motor penggerak yang akan bergerak sesuai dengan lintasan yang sudah diproses oleh kamera. Untuk melihat *flowchart* proses kerja *autonomous car* dapat dilihat pada Gambar 3.2.



Gambar 3.2 *Flowchart* Proses Kinerja *Autonomous Car*

3.4. Pola Lintasan

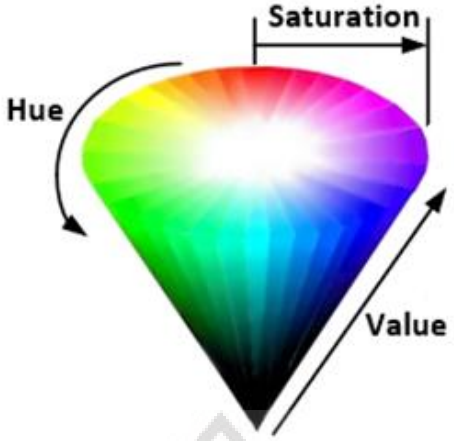
Rute yang digunakan pada *autonomous car* ini di tunjukkan pada Gambar 3.3. *Autonomous car* ini diuji dengan kondisi yang berbeda yaitu di *indoor* dan *outdoor* dan pola lintasan yang digunakan pada penelitian ini Menggunakan 2 jalur dengan kondisi berbelok dan kondisi lurus. Langkah yang perlu diperhatikan adalah proses kalibrasi nilai pada *threshold* karena sangat mempengaruhi jalannya *autonomous car* tersebut. Selain itu langkah yang perlu diperhatikan adalah posisi peletakan *autonomous car*, karena pola jalur harus masuk dalam frame kamera. Jika tidak terlihat oleh kamera maka algoritma *HSV* tidak bisa berjalan. Gambar 3.3 menunjukkan pola lintasan *autonomous car*.



Gambar 3.3 Pola Lintasan

3.5. Algoritma *HSV*

HSV merupakan suatu algoritma untuk menentukan warna yang diinginkan. Dalam penelitian ini tingkat akurasi dan kemampuan mengenali jalur menggunakan teknik filter warna *HSV* sangat cocok [10]. *HSV* merupakan sistem untuk mendeteksi tepi garis lintasan melalui proses gambar dari Raspberry Pi *camera v2* menggunakan proses *color filtering*. Keuntungan menggunakan algoritma ini adalah warna yang ditangkap lebih mendekati oleh penglihatan mata manusia [11]. Algoritma *HSV* sangat baik dalam mengidentifikasi warna primer, dan *HSV* toleran dengan perubahan intensitas cahaya. Ini adalah kelebihan algoritma *HSV* dibandingkan ruang warna lainnya [12]. Ruang warna algoritma *HSV* memiliki komposisi yang merupakan kombinasi dari warna primer biru, merah dan hijau ditunjukkan seperti Gambar 3.4.



Gambar 3.4 Ruang Warna Algoritma *HSV*

HSV memiliki tiga karakteristik pokok, yaitu *Hue*, *Saturation* dan *Value* yang mempunyai nilai yang berbeda [13]. Jenis tampilan warna hue merupakan lingkaran dengan sudut 360°. Nilai *hue* dirubah pada 0 sampai 255, dengan nilai 0 menjadi warna merah. *Saturation* menggambarkan intensitas warna yang mencakup nilai 0 sampai 255. Jika nilai *saturation* kurang akan menunjukan warna abu-abu. *Value* mencakup nilai 0 sampai 255, dengan 0 menjadi warna gelap dan nilai 255 menjadi warna cerah [14].

Nilai *HSV* diperoleh dari konversi nilai *RGB*. Konversi *RGB* dapat dijelaskan pada persamaan berikut [15]:

$$h = tan\left[\frac{3(g-b)}{(r-g)+(r-b)}\right] \quad (1)$$

$$s = 1 - \left[\frac{min(r,g,b)}{v}\right] \quad (2)$$

$$v = \frac{r+g+b}{3} \quad (3)$$

Pada persamaan (1)-(3) ketika S = 0 bahwa nilai H belum bisa ditentukan. Maka harus normalisasi nilai *RGB* lebih awal, persamaannya seperti berikut:

$$R = \frac{r}{r+g+b} \quad (4)$$

$$G = \frac{g}{r+g+b} \quad (5)$$

$$B = \frac{b}{r + g + b} \tag{6}$$

Dengan nilai normalisasi ini, maka persamaan konversi nilai dari *RGB* ke nilai *HSV* adalah seperti berikut:

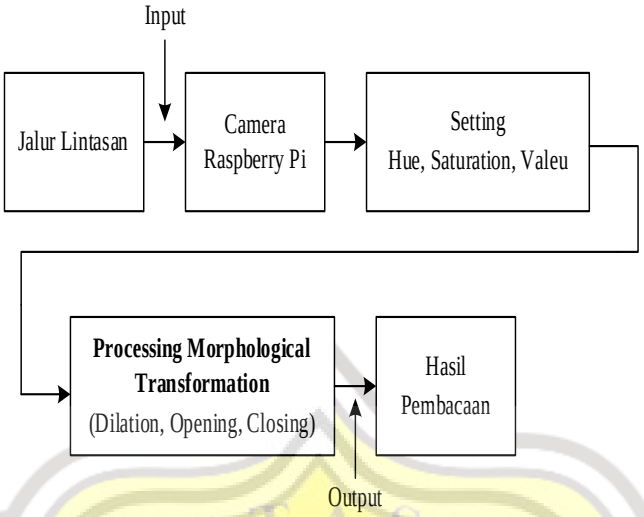
$$v = max\ (R, G, B) \tag{7}$$

$$S = \begin{cases} 0 & jika\ v > 0 \\ v - \frac{\min(R,G,B)}{v} & jika\ v = 0 \end{cases} \tag{8}$$

$$H = \begin{cases} 0 & jika\ s = 0 \\ \frac{60 \text{ x } (\text{g} - \text{b})}{S \text{ x } \text{V}} & jika\ v = R \\ 60 \text{ x } \left[2 + \frac{(\text{b} - \text{r})}{S \text{ x } \text{V}}\right] & jika\ v = G \\ 60 \text{ x } \left[4 + \frac{(\text{r} - \text{g})}{S \text{ x } \text{V}}\right] & jika\ v = B \end{cases} \tag{9}$$

$$H = H + 360 \qquad jika\ H < 0 \tag{10}$$

Sistem yang digunakan pada *HSV* terdiri dari jalur lintasan. Setelah jalur lintasan yang akan dideteksi oleh *camera* Raspberry Pi. Setelah kamera mendeteksi maka akan dilanjutkan dengan proses kalibrasi nilai *threshold* pada *HSV*. *Setting* nilai *threshold* difungsikan untuk menentukan deteksi pola lintasan. Kemudian akan diproses lagi dengan menggunakan *morphologi transformation* dengan melalui proses *dilation*, *opening* dan *closing*. Setelah semua proses dilalui maka hasil akan terlihat dilayar monitor. Diagram blok *HSV* bisa dilihat pada Gambar 3.5.



Gambar 3.5 Diagram Blok *HSV*

3.6. Filter Gaussian

Filter gaussian untuk menghaluskan atau menghilangkan *noise* saat mendeteksi garis tepi jalur dengan mengubah gambar input menjadi gambar abu abu atau blur [16]. Gambar blur diperoleh dengan melakukan *konvolusi kernel gaussian* ditunjukan pada persamaan (11), (12).

$$G(x,y) = F(x,y) * H(x,y) \tag{11}$$

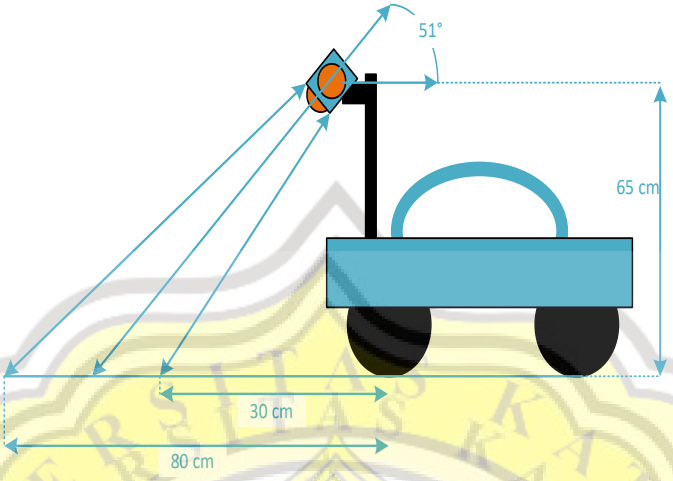
$$H(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{X^2+Y^2}{2\delta^2}} \tag{12}$$

Pada persamaan (11) dan (12), G(x,y) mewakili gambar yang dihaluskan, F(x,y) mewakili gambar input, dan H(x,y) mewakili *filter gaussian* yang dipilih.

3.7. Desain Penempatan Kamera *Autonomous Car*

Sensor yang dipakai dalam penelitian ini adalah Raspberry Pi *camera v2* diletakkan di bagian depan mobil dengan menggunakan tempat buatan tangan berbahan dasar akrilik. Tempat kamera Raspberry Pi ini diletakkan 510 terhadap lantai dengan tinggi 65 cm di atas permukaan lantai. Pemasangan ini bertujuan agar kamera Raspberry Pi dapat

mengidentifikasi objek yaitu berupa jalur lintasan. Gambar penempatan kamera bisa dilihat pada Gambar 3.6.



Gambar 3.6 Penempatan Kamera