BAB III

METODE PENELITIAN

Metode penelitian merupakan penjelasan dari metode-metode yang digunakan pada penelitian ini.

3.1 Metode Pengembangan

Pada penelitian Tugas Akhir ini dilakukan pendeteksian obyek berupa *cradle* (tempat tidur bayi pada KRPAI 2016) dengan *mobile robot* yang menggunakan kamera sebagai mata robot. *Mobile robot* akan mencari *cradle* pada setiap ruangan lapangan KRPAI 2016 dan ketika *mobile robot* mendeteksi sebuah *cradle* pada salah satu ruangan, maka *mobile robot* memberi penanda atau indikator bahwa *mobile robot* benar-benar mendeteksi adanya *cradle* pada salah satu ruangan lapangan.

Proses pengolahan citra dilakukan pada Raspberry yang sudah terinstall OpenCV, maka Raspberry bertugas untuk mendeteksi *cradle* dan mengirim data ke sebuah mikrokontroler untuk mengatur pergerakan *mobile robot* atau semua sensor-sensor yang terpasang pada *mobile robot*. Komunikasi antara Raspberry sebagai pengolarahan citra dan Arduino sebagai mikrokontroler pengendali pada *mobile robot* menggunakan komunikasi serial.

3.2 Prosedur Penelitian

Prosedur penelitian yang digunakan pada pengerjaan Tugas Akhir ini adalah:

1. Studi literatur

   Pada proses pengerjaan penelitian ini terdapat dua proses prancangan yang dilakukan yaitu, proses perancangan perangkat keras dan lunak.

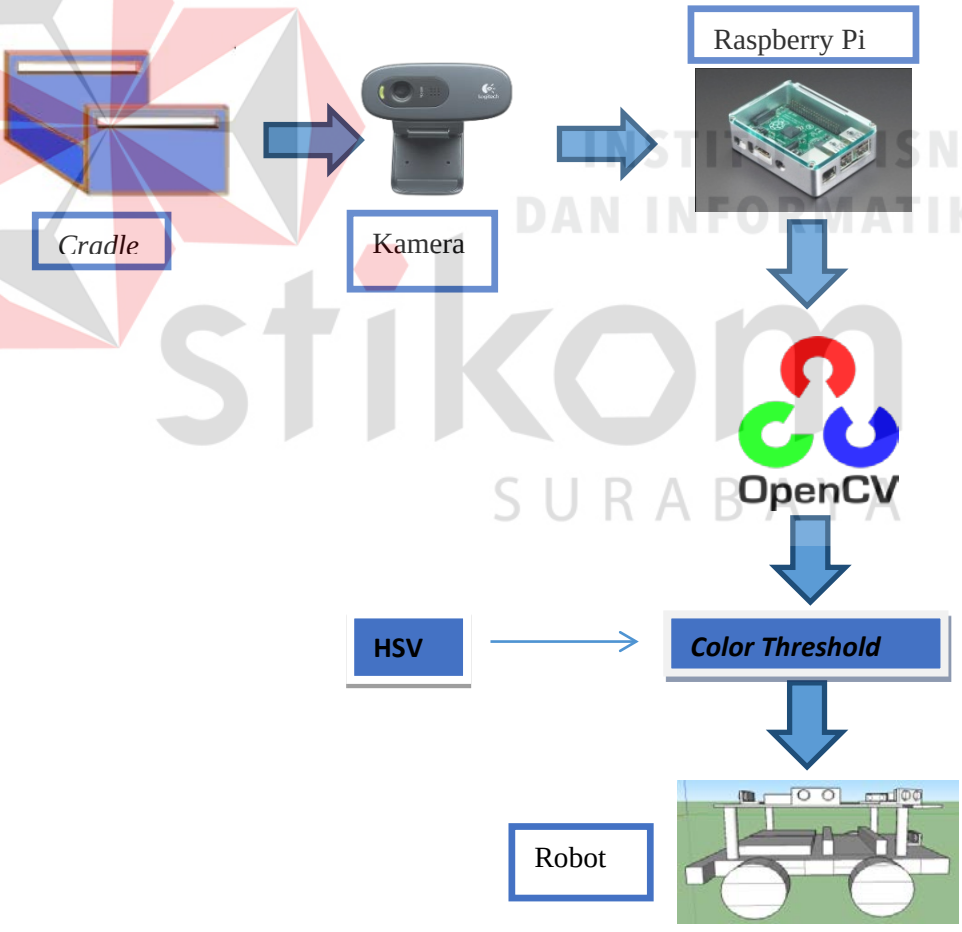
2. Tahapan perancangan dan pengembangan sistem

   Dalam pengerjaan pengembangan sistem, terdapat beberapa langkah rancangan sistem yang diambil antara lain:

   a. Membuat *flowchart* pada proses sistem secara keseluruhan.
   b. Melakukan perancangan perangkat keras yang meliputi:
      1. Merancang rangkaian elektronika yang digunakan pada penelitian.
      2. Melakukan percobaan tentang cara penggunaan Raspberry sebagai *image processing* serta sensor-sensor lainnya seperti limit *switch*, PING, *adjustable* Inframerah, sensor garis dan *device* yang digunakan pada penelitian.
      3. Merancang mekanik pada *mobile robot* KRPAI 2016.
   c. Melakukan perancangan perangkat lunak yang meliputi:
      1. Membuat program *Color Threshold* pada Raspberry agar dapat mendeteksi *cradle*.
      2. Membuat program untuk menggerakan motor pada Arduino agar dapat berjalan.
      3. Membuat program pembacaan sensor-sensor seperti PING, limit *switch, adjustable* inframerah serta sensor garis pada Arduino.

3.3 Diagram Blok Sistem

Dari penelitian ini terdapat proses yang akan dijalankan, yaitu proses pendeteksian *cradle* dengan menggunakan *image processing* melalui Raspberry dan pembacaan sensor-sensor serta mengatur pergerakan motor melalui Arduino sebagai mikrokontroller. *Mobile robot* memberi sebuah indikator ketika *mobile robot* dapat mendeteksi sebuah *cradle*. Pencarian *cradle* dilakukan pada setiap ruangan lapangan KRPAI 2016. Diagram blok penelitian dapat dilihat pada Gambar 3.1

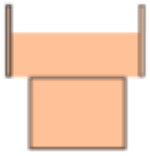


Gambar 3.1 Diagram Blok Penelitian

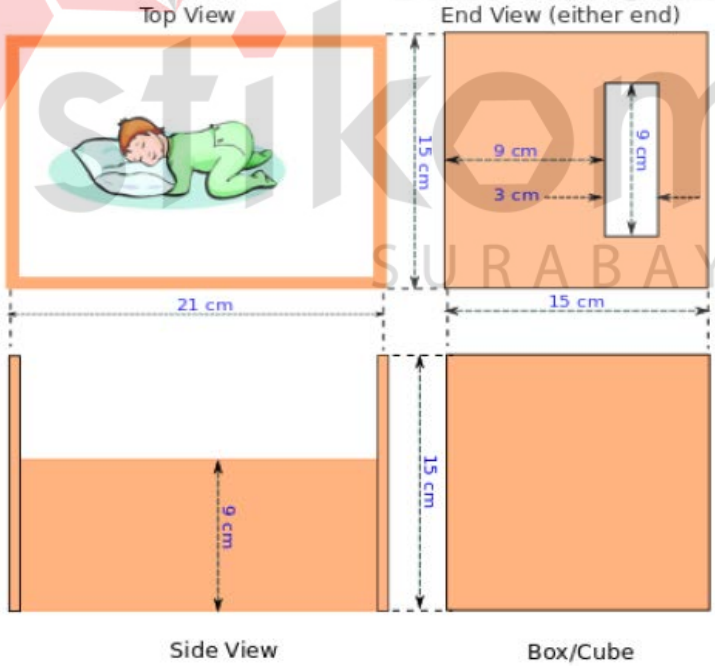
Dari Gambar 3.1 dapat dijelaskan proses alur kerja robot sebagai berikut:

3.3.1 *Cradle* (Tempat Tidur Bayi)

Cradle bisa juga disebut tempat tidur si bayi. *Cradle* terbuat dari *styrofoam*. *Cradle* diletakkan di atas kotak (kubus) dengan ukuran 15 cm. Tempat dudukan *cradle* ditunjukkan oleh Gambar 3.2, sedangkan untuk detail ukuran *cradle* seperti pada Gambar 3.3.



Gambar 3.2 Dudukan *Cradle*



Gambar 3.3 Detail Ukuran *Cradle*

Sumber : (Trinity College. Fire-Fighting Home Robot Contest 2016 Rules)

3.3.2 Kamera

Fungsi kamera secara umum adalah mengambil suatu gambar dari obyek, yang akan dibiaskan melalui lensa pada sensor CCD dan sensor BSI-CMOS kemudian direkam dan disimpan dalam format digital. Tetapi fungsi kamera yang dipasang pada *mobile robot* difungsikan sebagaimana mata manusia. Kamera mencari berupa obyek *cradle* dan ketika *cradle* terlihat oleh kamera, maka *mobile robot* memberikan penanda atau *indicator*. Berikut adalah cuplikan program untuk memanggil fungsi kamera dan mengatur resolusi *pixel*nya:

```
CvSize size640x480 = cvSize(640, 480);

      CvCapture* p_capWebcam;
p_capWebcam = cvCaptureFromCAM(1);

      cvNamedWindow("Original", CV_WINDOW_AUTOSIZE);
      cvNamedWindow("Processed", CV_WINDOW_AUTOSIZE);
```

3.3.3 HSV (*Hue Saturation Value*)

Untuk mentransformasikan dari RGB ke HSV, diasumsikan koordinat-koordinat R, G, B [0,1] adalah berurutan merah, hijau, biru dalam ruang warna RGB, dengan *max* adalah nilai maksimum dari nilai *red, green, blue* dan *min* adalah nilai minimum dari nilai *red, green, blue.* Pengkonversian nilai RGB ke HSV Untuk dapat merubah nilai RGB menjadi nilai HSV dapat menggunakan teori Travis, sebagai berikut:

$$H = \begin{cases} 0, jika\_Max = Min \\ \frac{G-B}{Max-Min} x60, ifR = Max \\ 120 + \frac{B-R}{Max-Min} x60, ifG = Max \\ 240 + \frac{R-G}{Max-Min} x60, ifB = Max \end{cases}$$

$$S = \frac{Max - Min}{Max}$$

$$V = Max$$

Rumus di atas menghasilkan nilai *value* dan *saturation* dalam jangkauan RGB [0,1]. Dikalikan dahulu dengan 255 untuk memperoleh nilai dengan jangkauan RGB [0,255]. Misalnya ingin ditransformasikan RGB (65, 27, 234) ke dalam bentuk HSV, maka langkahnya adalah sebagai berikut:

Setiap nilai RGB (65, 27, 234) diubah dalam jangkauan [0,1] dengan membagi setiap nilai dengan 255:

$$Menjadi \left(\frac{65}{255}, \frac{27}{255}, \frac{234}{255}\right) = (0.255, 0.106, 0.918)$$

RGB (0.255, 0.106, 0.918) ini yang akan ditransformasikan ke bentuk HSV.

max = nilai B (blue) = 0.918,

min = nilai G (green) = 0.106,

max – min = 0.918 – 0.106 = 0.812.

$$h\ (hue) = 60 \times \left(\frac{R-G}{Max-Min} + 4\right), karena\ Max = B\ Blue.$$

$$= 60 \times \left(\frac{0.255 - 0.106}{0.812} + 4\right) = 251$$

$$v\ (value) = Max = 0.918$$

$$s\ (saturation) = \frac{Max - Min}{v}, karena\ Max <> Min$$

$$= \frac{0.812}{0.918} = 0.885$$

Sehingga nilai RGB (65, 27, 234) ditransformasikan menjadi HSV (251, 0.885, 0.918).

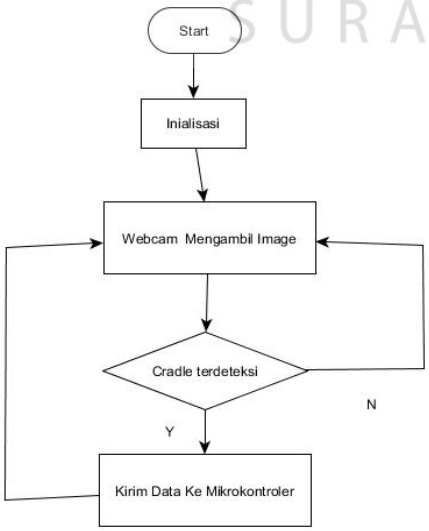
Berikut merupakan cuplikan program HSV (*Hue-Saturation-Value*):

```
int H_MIN = 0;
int H_MAX = 256;
int S_MIN = 0;
int S_MAX = 256;
int V_MIN = 0;
int V_MAX = 256;
using namespace cv;

const string trackbarWindowName = "Trackbars";
void on_trackbar(int, void*)
{}

void createTrackbars(){
        namedWindow(trackbarWindowName, 0);
        char TrackbarName[50];
        printf(TrackbarName, "H_MIN", H_MIN);
        printf(TrackbarName, "H_MAX", H_MAX);
        printf(TrackbarName, "S_MIN", S_MIN);
        printf(TrackbarName, "S_MAX", S_MAX);
        printf(TrackbarName, "V_MIN", V_MIN);
        printf(TrackbarName, "V_MAX", V_MAX);
createTrackbar("H_MIN", trackbarWindowName, &H_MIN, H_MAX,
on_trackbar);
       createTrackbar("H_MAX", trackbarWindowName, &H_MAX, H_MAX,
on_trackbar);
       createTrackbar("S_MIN", trackbarWindowName, &S_MIN, S_MAX,
on_trackbar);
       createTrackbar("S_MAX", trackbarWindowName, &S_MAX, S_MAX,
on_trackbar);
       createTrackbar("V_MIN", trackbarWindowName, &V_MIN, V_MAX,
on_trackbar);
       createTrackbar("V_MAX", trackbarWindowName, &V_MAX, V_MAX,
on_trackbar);
}
cvCvtColor(p_imgOriginal, p_imgHSV, CV_BGR2HSV);
```

3.3.4 *Color Threshold*

Color Filtering adalah metode yang digunakan untuk mencari suatu warna pada sebuah *image*. Setelah warna yang dicari dapat ditemukan pada *image*, maka proses selanjutnya bisa dilakukan. Pada dasarnya pencarian ini menggunakan kombinasi dari komponen warna *Red*, *Green*, dan *Blue* yang terdapat pada *image*.

Nilai dari masing-masing komponen didapat dari hasil beberapa kali percobaan. Nilai dari masing-masing komponen warna kemudian dijadikan *filter* yang merupakan penentu sebuah warna diloloskan atau tidak. Keluaran dari metode *color Thershold* adalah *grayscale image*. *Grayscale image* adalah sebuah gambar yang hanya memiliki 2 gradasi warna yaitu hitam dan putih, dimana warna putih pada gambar merupakan representasi *pixel* dari warna yang lolos seleksi. Sedangkan, warna hitam pada gambar merupakan representasi *pixel* dari warna yang tidak lolos seleksi. *Flowchart Color Threshold* dapat dilihat pada Gambar 3.4



Gambar 3.4 *Flowchart Color Threshold*

Flowchart pada Gambar 3.4 menjelaskan tentang bagaimana proses pendeteksia *cradle* dengan menggunakan *color threshold*. Pertama adalah proses inisialisasi. Kemudian kamera *webcam* melakukan pencarian obyek berupa *cradle*. Ketika *cradle* terdeteksi oleh kamera, maka program akan mengirimkan data ke mikrokontroler dan jika *cradle* belum terdeteksi maka Webcam akan terus melakukan proses pendeteksian *cradle*. Berikut merupakan contoh program untuk *Color Threshold* dan untuk menandai bahwa *cradle* dapat dideteksi:

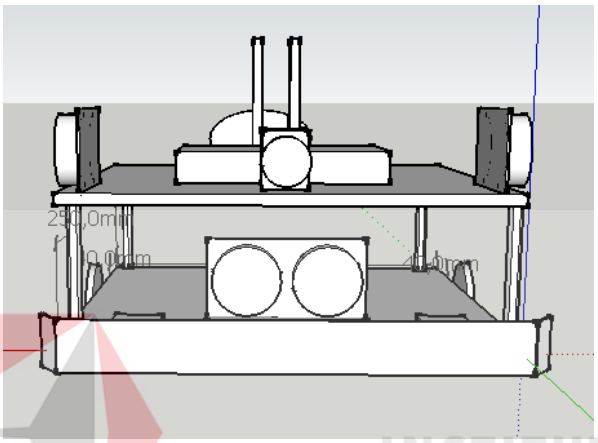
```
cvSmooth(p_imgProcessed, p_imgProcessed, CV_GAUSSIAN, 9, 9);

            p_seqCircles = cvHoughCircles(p_imgProcessed, p_strStorage,
CV_HOUGH_GRADIENT, 2, p_imgProcessed->height / 4, 100, 50, 10, 210);

            if (p_seqCircles->total == 1)
            {
                  p_fltXYRadius = (float*)cvGetSeqElem(p_seqCircles, 1);
                  //printf (" ono cak /n ");
                  printf("ball position x = %f, y = %f, r = %f \n",
p_fltXYRadius[0], p_fltXYRadius[1], p_fltXYRadius[2]);
                  cvCircle(p_imgOriginal, cvPoint(cvRound(p_fltXYRadius[0]),
cvRound(p_fltXYRadius[1])), 3, CV_RGB(0, 255, 0), CV_FILLED);
                  cvCircle(p_imgOriginal, cvPoint(cvRound(p_fltXYRadius[0]),
cvRound(p_fltXYRadius[1])), cvRound(p_fltXYRadius[2]), CV_RGB(255, 0, 0), 3);
            }
```


3.4 Perancangan Mekanik Robot

Mekanik dari *mobile robot* yang digunakan pada penelitian ini terbuat dari bahan plat sebagai *base* 1 atau paling bawah. Sedangkan, untuk *base* 2 terbuat dari cetakan PCB berbentuk persegi panjang yang telah didesain dan dirancang khusus untuk kepentingan penelitian sistem *mobile robot* ini. Tujuan *mobile robot* didesain sedemikian rupa agar seluruh komponen elektronika bisa terpasang dengan baik pada *mobile robot*, mulai dari rangkaian mikrokontroler, *motor driver*, sensor photodioda, PING, *limit Switch*, *adjustable infrared*, LCD

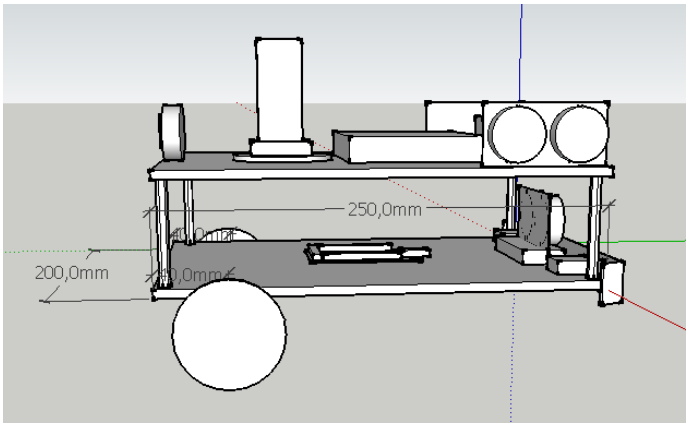
dan baterai untuk *mobile robot*. *Mobile robot* juga mempunyai sebuah indikator jika *mobile robot* dapat mendeteksi adanya *cradle*.

Pada Gambar 3.5 sampai 3.8 merupakan desain *mobile robot* dilihat dari beberapa sisi.



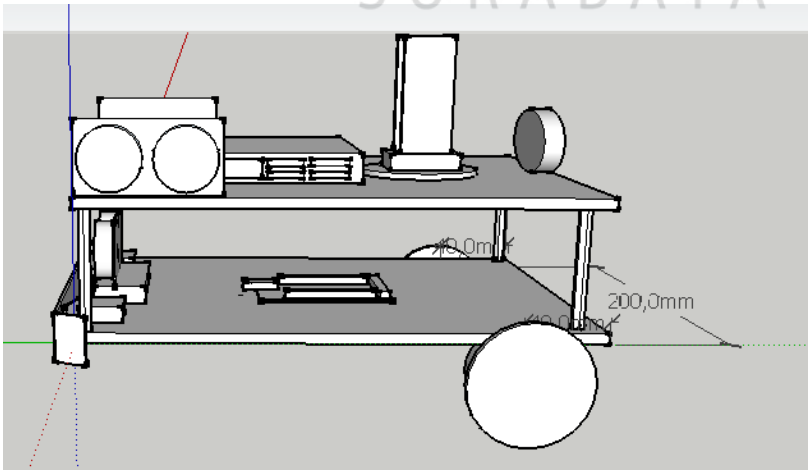
Gambar 3.5 *Mobile robot* Tampak Depan

Gambar 3.5 menunjukkan posisi *mobile robot* menghadap ke depan. Pada bagian depan robot terdapat beberapa komponen seperti sensor *ultrasonic*, yaitu PING dan sensor *Adjustable Infrared*. Sensor PING digunakan untuk memberi jarak antara *mobile robot* dengan dinding agar *mobile robot* tidak sampai menabrak dinding ketika berjalan. Sensor *adjustable infrared* digunakan untuk mendeteksi adanya benda (*furniture*) di depan *mobile robot* yang menghalangi jalan *mobile robot*.



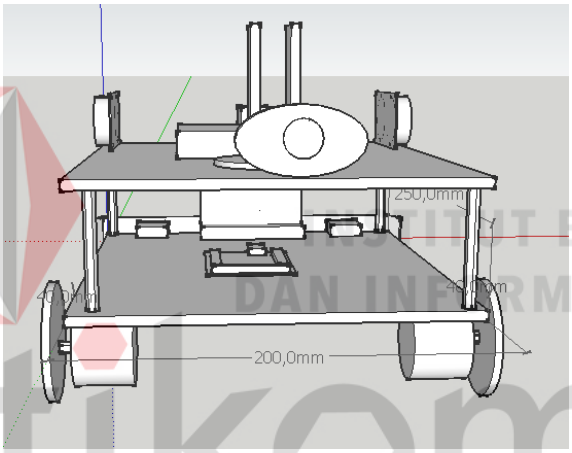
Gambar 3.6 *Mobile robot* Tampak Samping Kanan

Gambar 3.6 menunjukkan posisi *mobile robot* menghadap ke samping kanan. Pada bagian samping kanan *mobile robot* terdapat beberapa komponen seperti sensor *ultrasonic* PING, motor *driver*, dan roda kanan *mobile robot*. Sensor PING digunakan untuk memberi jarak antara *mobile robot* dengan dinding agar *mobile robot* tidak sampai menabrak dinding ketika berjalan. Motor *driver* digunakan untuk mengontrol pergerakan dan kecepatan motor DC. Roda digunakan agar *mobile robot* bisa bergerak.



Gambar 3.7 *Mobile robot* Tampak Samping Kiri

Gambar 3.7 menunjukkan posisi robot menghadap ke samping kiri. Pada bagian samping kiri *mobile robot* terdapat beberapa komponen seperti sensor *ultrasonic* PING, motor *driver*, dan roda kiri *mobile robot*. Sensor PING digunakan untuk memberi jarak antara *mobile robot* dengan dinding agar *mobile robot* tidak sampai menabrak dinding ketika berjalan. Motor *driver* digunakan untuk mengontrol pergerakan dan kecepatan motor DC. Roda digunakan agar *mobile robot* bisa bergerak.



Gambar 3.8 *Mobile robot* Tampak Belakang

Gambar 3.8 menunjukkan posisi robot menghadap ke belakang. Pada bagian samping kiri *mobile robot* terdapat kamera yang digunakan untuk mendeteksi *cradle*.

3.4.1 Dimensi Robot

Setelah semua komponen tambahan dipasang pada *mobile robot*, berikut merupakan dimensi keseluruhan dari *mobile robot*:

1. Panjang *Mobile robot* : 25 cm

2. Lebar *Mobile robot* : 20 cm
3. Tinggi *Mobile robot* : 27 cm

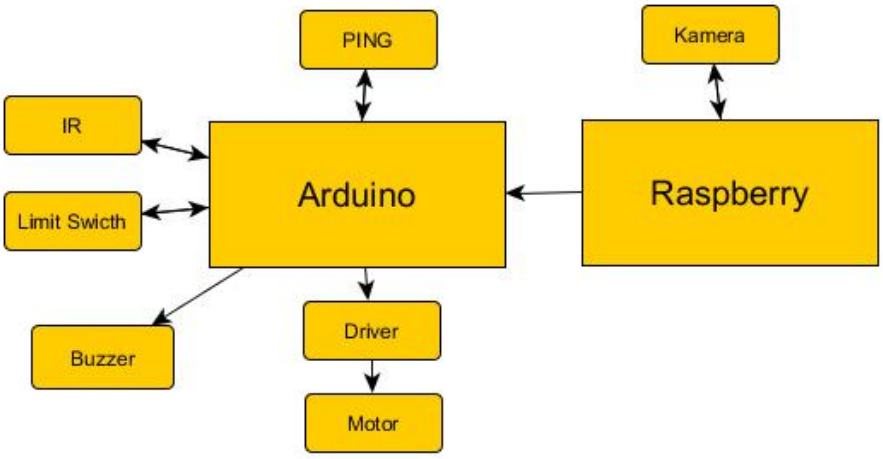
3.4.2 Struktur Material Robot

Bahan material yang digunakan dalam penelitian ini untuk merancang dan membuat *mobile robot* adalah sebagai berikut:

a. Bagian *base* 1
    1. Plat
    2. Mur dan Baut
b. Bagian *base* 2
    1. PCB
    2. Mur dan Baut
c. Bagian dari Perangkat Robot
    1. Motor DC 12 volt
    2. Roda Bebas
    3. Penghalang *limit switch*

3.4.3 Perancangan Mikrokontroller Arduino

Bagian elektronika *mobile robot*:

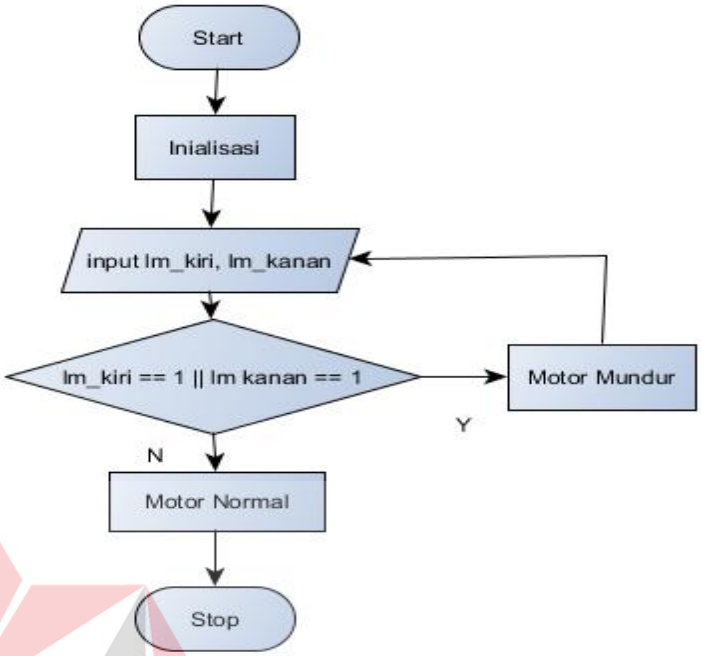


Gambar 3.9 Diagram Robot

Pada Gambar 3.9 Arduino sebagai mikrokontroler digunakan untuk mengatur sensor-sensor yang terpasang pada *mobile robot*, seperti sensor ultrasonik PING, *Adjustable infrared*, *limit switch*, motor *driver*, serta komunikasi antara Arduino dengan Raspberry menggunakan komunikasi serial. Pada sisi komunikasi, Arduino berfungsi hanya sebagai penerima saja dan tidak bisa mengirim kembali atau memberi *feedback* ke Raspberry.

3.4.4 *Limit Switch*

Limit switch berkerja ketika terkena tekanan atau sentuhan dari sebuah benda. Misalnya pada robot KRPAI, ketika robot menabrak atau menyentuh dinding, maka robot akan memberikan tegangan pada mikrokontroler sebagai indikator bahwa robot menabrak atau menyentuh dinding. *Flowchat* penggunaan *limit switch* pada penelitian ini ditunjukkan oleh Gambar 3.10.



Gambar 3.10 *Flowchart Limit Switch*

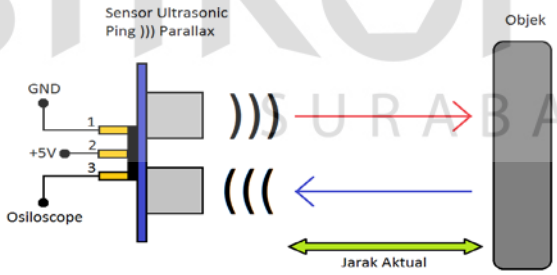
Fungsi *limit switch* pada robot KRPAI adalah untuk meminimalisir kerusakan pada *base* robot saat terjadi benturan dengan dinding ataupun benda yang lainnya. *Flowchart* pada Gambar 3.10 menjelaskan bahwa setelah proses inisialisasi dilakukan proses selanjutnya adalah ketika salah satu atau kedua *limit switch* bernilai 1 maka robot akan mundur, tetapi jika tidak robot berjalan normal. Perubahan nilai dari *limit switch* terjadi karena *mobile robot* menabrak dinding sehingga *limit switch* menjadi tertekan oleh dinding.

Robot KRPAI mempunyai 2 buah *limit switch* dengan cara kerja *limit switch* sebagai berikut. Berikut merupakan program *limit switch* yang diterapkan pada *mobile robot*:

```
int swit1=digitalRead(lim_swit);
 if(swit1 == LOW)
 {   analogWrite(PWMka,40);
  analogWrite(PWMki,40);
  mundur_kanan();
  mundur_kiri();
  delay(500);}
```

3.4.5 Sensor Ultrasonic PING

Sensor ultrasonic adalah sebuah sensor yang memanfaatkan pancaran gelombang ultrasonic. Sensor ultrasonic terdiri dari *transmitter* dan *receiver*. *Transmitter* berfungsi pemancar gelombang, sedangkan *receiver* berfungsi sebagai penerima gelombang. Pembacaan sensor ultrasonik PING ditunjukkan pada Gambar 3.11



Gambar 3.11 Pembacaan Sensor Ultrasonic PING

Nilai pembacaan jarak pada sensor ultrasonic PING diperoleh dari rumus berikut ini:

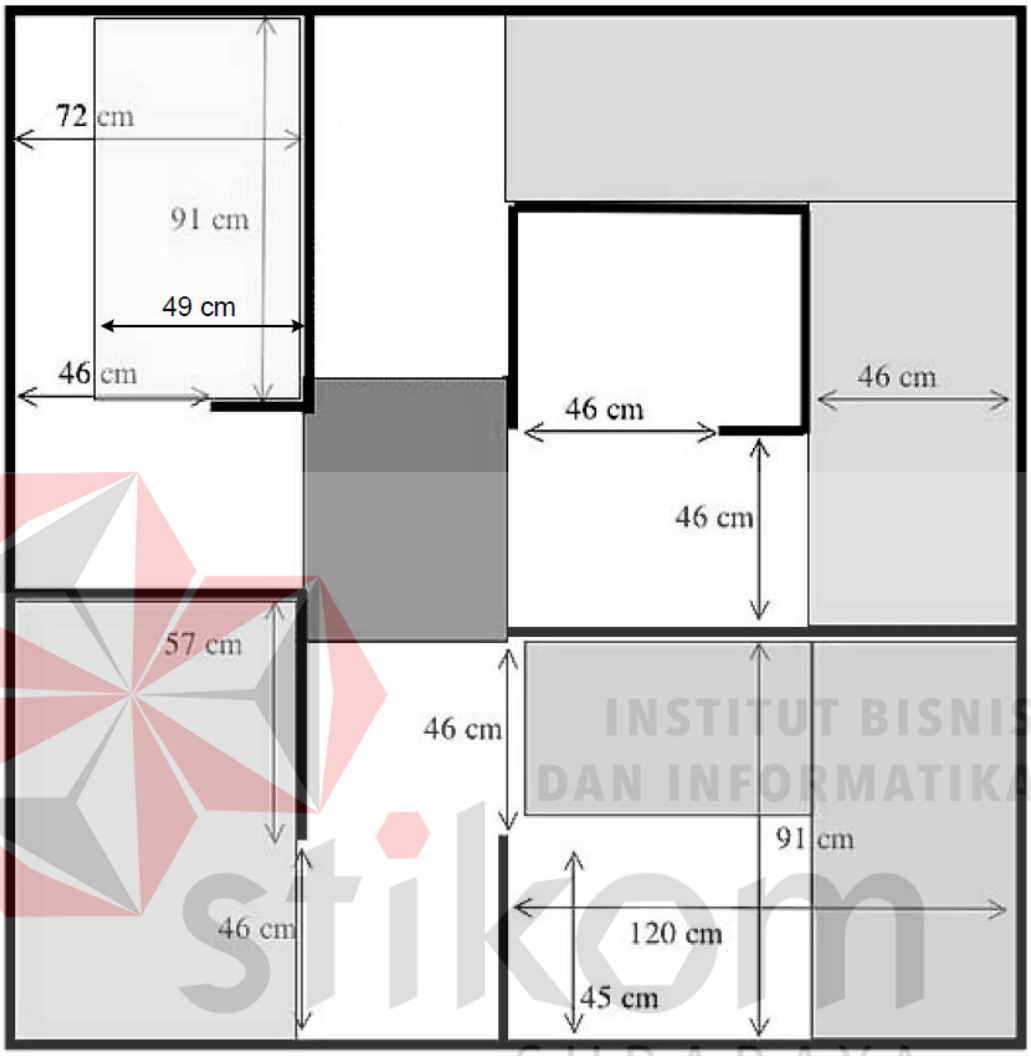
Jarak (cm) = Lama Waktu Pantul (uS) / 29.034 / 2

Rumus jarak didapat dari pembagian lama waktu pantul dengan kecepatan gelombang ultrasonik dan dibagi 2 karena pada saat pemantulan terjadi dua kali jarak tempuh antara sensor dengan obyek, yaitu pada saat gelombang dipancarkan dari *transmitter* ke obyek dan pada saat gelombang memantul ke *receiver* ultrasonik. Berikut adalah contoh program sensor ultrasonic PING:

```
const int pingPin = 42;
const int pingPin2 = 48;
int ping1()
{  long duration, cm; pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW); delayMicroseconds(2);
digitalWrite(pingPin, HIGH); delayMicroseconds(5);
digitalWrite(pingPin, LOW); pinMode(pingPin, INPUT);
 duration = pulseIn(pingPin, HIGH);
 cm = duration / 29 / 2;
 dtp1=cm;}
void loop()
{ping1();}
```


3.5 Lapangan Uji Coba Robot

Lapangan uji coba digunakan untuk melakukan percobaan pendeteksian *cradle* di setiap ruangan. Lapangan menggunakan konsep seperti pada pertandingan robot KRPAI beroda, yang memiliki 4 ruangan seperti yang ditunjukkan oleh Gambar 3.12



Gambar 3.12 Lapangan Uji Coba