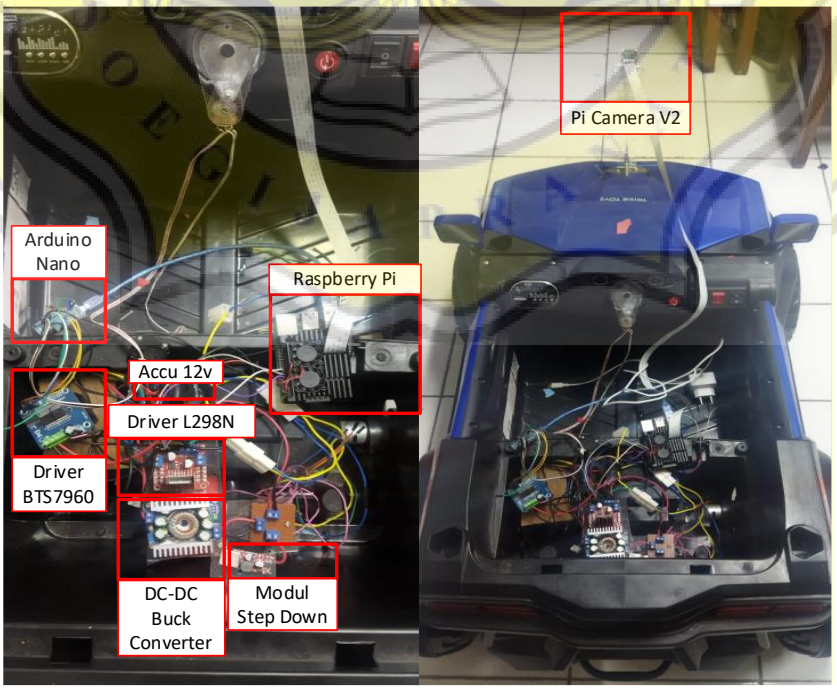
BAB IV
HASIL DAN PEMBAHASAN

4.1. Pendahuluan

Pada bab ini akan dijelaskan hasil hasil *prototype autonomous car*, program, dan pengujian alat yang dilakukan melalui beberapa proses pengujian yaitu dengan proses *color filtering*, pengujian fungsionalitas dan yang terakhir adalah pengujian deteksi jalur. Tujuan utama dari beberapa proses pengujian ini adalah agar sistem *self-driving* menggunakan algoritma *HSV* mendapat hasil yang maksimal.

4.2. Prototype Alat

Dalam penelitian ini, menghasilkan bentuk *hardware* yang digunakan untukTugas Akhir. Hasil *hardware* ini sesuai dengan perancangan alat pada Bab III. Gambar 4.1 menunjukkan *hardware* dari AGV.



Gambar 4.1 *Hardware Autonomous Car*

4.3. Program

Berikut ini merupakan pembahasan program untuk sistem pergerakan *autonomous car* saat bergerak mengikuti lintasan.

main.py

```
1  import cv2
2  import numpy as np
3  import serial
4  import RPi.GPIO as GPIO
5
```

Langkah pertama, import *library* yang dibutuhkan. Fungsi ***import*** adalah untuk memasukkan *library* diataranya ada *cv2* serial, import sendiri adalah fungsi untuk memasukkan program tersendiri yang sebelumnya dibuat.

```
6  if __name__ == '__main__':
7      GPIO.setwarnings(False)
8      kernel = np.ones((5,5),np.uint8)
9      cap = cv2.VideoCapture(0)
10     panjang=320
11     lebar=240
12     cap.set(3,panjang)
13     cap.set(4,lebar)
14     in1 = 24
15     in2 = 23
16     en = 25
17     GPIO.setmode(GPIO.BCM)
18     GPIO.setup(in1,GPIO.OUT)
19     GPIO.setup(in2,GPIO.OUT)
20     GPIO.setup(en,GPIO.OUT)
21     GPIO.output(in1,GPIO.LOW)
22     GPIO.output(in2,GPIO.LOW)
23     p=GPIO.PWM(en,1000)
24     kondisinya=0
25     p.start(100)
26     final=0
27     p.ChangeDutyCycle(100)
28     ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=0.05)
```

Dilihat dari program diatas adalah bagian program untuk inisialisasi alamat dan *setting* komunikasi serial untuk arduino nano.

```
30
31      def nothing(x):
32          pass
33      cv2.namedWindow('HueComp')
34      cv2.namedWindow('SatComp')
35      cv2.namedWindow('ValComp')
36      cv2.namedWindow('closing')
37      cv2.namedWindow('tracking')
38
39      cv2.createTrackbar('hmin', 'HueComp',168,179,nothing)
40      cv2.createTrackbar('hmax', 'HueComp',179,179,nothing)
41
42      cv2.createTrackbar('smin', 'SatComp',40,255,nothing)
43      cv2.createTrackbar('smax', 'SatComp',255,255,nothing)
44
45      cv2.createTrackbar('vmin', 'ValComp',26,255,nothing)
46      cv2.createTrackbar('vmax', 'ValComp',129,255,nothing)
47
48      def sortSecond(val):
49          return val[1]
```

Pada program diatas ini adalah program untuk menampilkan *windows* dan *setting* nilai algoritma *HSV* yang memiliki fungsi untuk mendeteksi lintasan agar *autonomous car* dapat bergerak secara mengikuti jalur.

main.py

```
50      while True:
51          nilai=""
52          if(ser.in_waiting>0):
53              while ser.in_waiting:
54                  nilai =str(ser.readline())
55                  nilai=nilai.replace(r"\r\n","")
56                  nilai=nilai.replace("b","")
57                  nilai=nilai.replace("'","")
58                  if(nilai!="" ):
59                      akhir=nilai
60          if(int(akhir)>200 and int(akhir)<580):
61              final=int(akhir)
62          print(final)
```

Program diatas adalah progam untuk mententukan nilai maksimal sudut manuver yang berfungsi menentukan posisi tengah *autonomous car*.

main.py

```
        buzz = 0
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)

        ser.write(b"m\n")
        _, frame = cap.read()
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        hue,sat,val = cv2.split(hsv)
        hmn = cv2.getTrackbarPos('hmin','HueComp')
        hmx = cv2.getTrackbarPos('hmax','HueComp')


        smn = cv2.getTrackbarPos('smin','SatComp')
        smx = cv2.getTrackbarPos('smax','SatComp')


        vmn = cv2.getTrackbarPos('vmin','ValComp')
        vmx = cv2.getTrackbarPos('vmax','ValComp')
        hthresh = cv2.inRange(np.array(hue),np.array(hmn),np.array(hmx))
        sthresh = cv2.inRange(np.array(sat),np.array(smn),np.array(smx))
        vthresh = cv2.inRange(np.array(val),np.array(vmn),np.array(vmx))
        tracking = cv2.bitwise_and(hthresh,cv2.bitwise_and(sthresh,vthresh))
```

```

        dilation = cv2.dilate(tracking,kernel,iterations = 1)
        #dilation1 = cv2.dilate(kebalikan,kernel,iterations = 1)
        closing = cv2.morphologyEx(dilation, cv2.MORPH_CLOSE, kernel)
        closing = cv2.GaussianBlur(closing,(5,5),0)
        #closing1 = cv2.morphologyEx(dilation1, cv2.MORPH_CLOSE, kernel)
        #closing1 = cv2.GaussianBlur(closing1,(5,5),0)
        contours, hierarchy = cv2.findContours(closing,
        cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_TC89_L1)
        result = frame.copy()
        polygonelist = []
        kanan=[]
        kiri=[]
        kananrev=[]
        kirirev=[]
        xkanan=0
        xkiri=0
        ykanan=0
        ykiri=0
```

Pada program diatas adalah program yang ada didalem *windows* yang ditampilkan saat program berjalan.

main.py

```
106     #print("baru")
107     for cntr in contours:
108         perimeter = cv2.arcLength(cntr,True)
109         epsilon = 0.005*cv2.arcLength(cntr,True)
110         approx = cv2.approxPolyDP(cntr,epsilon,True)
111         #print(approx)
112         polygonelist.append(approx)
113     cv2.drawContours(frame, polygonelist, -1, (0, 255, 0), 2)
114     for x in range(0,len(polygonelist)):
115         for a in range(0,len(polygonelist[x])):
116             for b in range(0,len(polygonelist[x][a])):
117                 if(x==0):
118                     kanan.append(polygonelist[x][a][b])
119                     kananrev.append(polygonelist[x][a][b])
120                 elif(x==1):
121                     kiri.append(polygonelist[x][a][b])
122                     kirirev.append(polygonelist[x][a][b])
123     #for k in range(0,len(kanan)):
124     #    print(kanan[k][1])
125     kanan.sort(key=sortSecond)
126     kiri.sort(key=sortSecond)
127     kananrev.sort(key=sortSecond,reverse=True)
128     kirirev.sort(key=sortSecond,reverse=True)
```

Program diatas adalah progam untuk menentukan garis setiap sudut pada lintasan yang sudah diproses oleh *HSV* guna untuk menentukan garis tengah pada lintasan sehingga *autonomous car* dapat mempertahankan posisi pada tengah lintasan.

main.py

```
129         for k in range (0,2):
130             if(k==0):
131                 xkanan=kanan[k][0]
132                 xkiri=kiri[k][0]
133                 ykiri=kanan[k][1]
134                 ykanan=kiri[k][1]
135             elif(k==1):
136                 if(kanan[k][0]<xkanan):
137                     xkananatas=kanan[k][0]
138                     ykananatas=kanan[k][1]
139                 else:
140                     xkananatas=xkanan
141                     ykananatas=ykanan
142                 if(kiri[k][0]>xkiri):
143                     xkiriatas=kiri[k][0]
144                     ykiriatas=kiri[k][1]
145                 else:
146                     xkiriatas=xkiri
147                     ykiriatas=ykiri
```

main.py

```
149         for k in range (0,2):
150             if(k==0):
151                 xkanan=kananrev[k][0]
152                 xkiri=kirirev[k][0]
153                 ykiri=kananrev[k][1]
154                 ykanan=kirirev[k][1]
155             elif(k==1):
156                 if(kananrev[k][0]<xkanan):
157                     xkananbawah=kananrev[k][0]
158                     ykananbawah=kananrev[k][1]
159                 else:
160                     xkananbawah=xkanan
161                     ykananbawah=ykanan
162                 if(kiri[k][0]>xkiri):
163                     xkiribawah=kirirev[k][0]
164                     ykiribawah=kirirev[k][1]
165                 else:
166                     xkiribawah=xkiri
167                     ykiribawah=ykiri
168         misskiri=int(panjang/2)-xkiriatas
169         misskanan=xkananatas-int(panjang/2)
170         print(misskiri,misskanan,abs(misskanan-misskiri))
171         posisi="Kanan"
```

main.py

```
173         if(abs(misskanan-misskiri)<=75):
174             posisi="Tengah"
175         else:
176             if(misskiri>misskanan):
177                 posisi="Kiri"
178                 kondisinya=1
179                 GPIO.output(in1,GPIO.LOW)
180                 GPIO.output(in2,GPIO.HIGH)
181                 print("kiri")
182             else:
183                 posisi="Kanan"
184                 kondisinya=2
185                 GPIO.output(in1,GPIO.HIGH)
186                 GPIO.output(in2,GPIO.LOW)
187                 print("kanan")
```

main.py

```
188         if(kondisinya==1 and posisi=="Tengah"):
189             if(final>340 and final<450):
190                 GPIO.output(in1,GPIO.LOW)
191                 GPIO.output(in2,GPIO.LOW)
192             else:
193                 GPIO.output(in1,GPIO.HIGH)
194                 GPIO.output(in2,GPIO.LOW)
195                 print("Balik arah")
```

main.py

```
196         elif(kondisinya==2 and posisi=="Tengah"):
197             if(final>340 and final<450):
198                 GPIO.output(in1,GPIO.LOW)
199                 GPIO.output(in2,GPIO.LOW)
200             else:
201                 GPIO.output(in1,GPIO.LOW)
202                 GPIO.output(in2,GPIO.HIGH)
203                 print("Balik arah")
```

Program diatas adalah progam *autonomous car* dalam melakukan pergerakan agar *autonomous car* bergerak tetap dalam kondisi didalam lintasan atau berjalan ditengah dan tidak keluar lintasan pada jalur *Street Mark* pada program diatas dijelaskan ada program *HIGH* dan *LOW* program ini sendiri untuk mengatur suatu kecepatan motor DC.

main.py

```
        cv2.putText(frame, posisi, (0,20), cv2.FONT_HERSHEY_SIMPLEX,
                    0.5, (255, 0, 0), 2, cv2.LINE_AA)
        cv2.circle(frame, (int(panjang/2),lebar), radius=2, color=(0, 0, 255), thickness=10)
        cv2.line(frame, (int(panjang/2),0), ((int(panjang/2),lebar)), (255, 0, 0), 2)
        cv2.line(frame, (xkananatas,ykananatas), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        cv2.line(frame, (xkiriatas,ykiriatas), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        cv2.line(frame, (xkananbawah,ykananbawah), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        cv2.line(frame, (xkiribawah,ykiribawah), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        #cv2.line(frame, (xkanantengan,ykanantengan), ((int(panjang/2),lebar)), (0, 0, 255),
        #cv2.line(frame, (xkiritengan,ykiritengan), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        #cv2.imshow('HueComp',hthresh)
        #cv2.imshow('SatComp',sthresh)
        #cv2.imshow('ValComp',vthresh)
        cv2.imshow('closing',closing)
        cv2.imshow('tracking',frame)
        k = cv2.waitKey(5) & 0xFF
        if k == 27:
            break

    cap.release()

    cv2.destroyAllWindows()
```

Pada program diatas adalah yang berfungsi untuk menampilkan *trackbar* yang berfungsi untuk mengatur *Hue*, *Saturation* dan *Value*

4.4. Pengujian Color Filtering

Dengan melakukan kalibrasi *min* ke *max* dan *max* ke *min*. Pada percobaan ini kalibrasi *min* ke *max* diberikan nilai 0, dan pengujian kalibrasi *max* ke *min* dengan memberikan nilai *max* 179 untuk H , 255 untuk S dan V. Dengan cara menggeser nilai *max* atau *min* sampai terjadinya perubahan pada objek pola. Tabel 2 menunjukkan uji coba *threshold* H *(hue).* Tabel 3 menunjukkan uji coba *threshold* S *(saturation).* Tabel 4 menunjukkan uji coba *threshold* V *(value).*

Tabel 1. Uji Coba *Threshold Hue*

Kalibrasi	H *min*	H *max*	Warna Putih	Warna Hitam
Min ke *Max*	0	135	Terdeteksi	Tidak
Max ke *Min*	137	179	Tidak	Terdeteksi

Tabel 2. Uji Coba *Threshold Saturation*

Kalibrasi	S *min*	S *max*	Warna Putih	Warna Hitam
Min ke *Max*	0	70	Terdeteksi	Tidak
Max ke *Min*	72	255	Tidak	Terdeteksi

Tabel 3. Uji Coba *Threshold Value*

Kalibrasi	V *min*	V *max*	Warna Putih	Warna Hitam
Min ke *Max*	0	53	Terdeteksi	Tidak
Max ke *Min*	60	255	Tidak	Terdeteksi

Pada Tabel 1 sampai dengan Tabel 3 menggunakan metode *thresholding* dimana pada proses ini memerlukan setting nilai yang sesuai sehingga ditemukan nilai yang akurat. Kemudian dilakukan uji coba melalui perubahan nilai *min* ke *max* dan *max* ke *min*. Metode diperlukan untuk menentukan warna hitam maupun putih pada jalur.

4.5. Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan untuk menentukan efek dari intensitas cahaya yang berbeda. Pada pengujian ini mencoba dua kondisi yaitu di *indoor* dan di *outdoor*. Pada pengujian *indoor* menggunakan penerangan lampu TL sedangkan pada pengujian di *outdoor* langsung menggunakan penerangan dari sinar matahari tanpa hambatan apapun.

Tabel 4. Pengujian Fungsionalitas

Threshold Saturation	*Threshold Value*	Kondisi *Indoor*	Kondisi *Outdoor*
0-20	200-255	Jalur Tidak Terdeteksi	Jalur Tidak Terdeteksi
10-50	130-255	Jalur Terdeteksi	Terdeteksi Banyak *Noise*
70-255	53-106	Jalur Terdeteksi	Jalur Terdeteksi

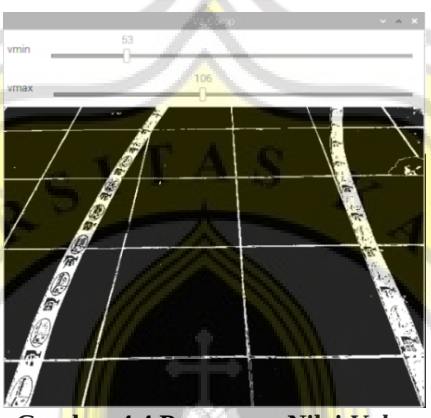
Tabel 4 menunjukan bahwa menetapkan nilai threshold yang dibawah kondisi lokasi yang berbeda dapat mempengaruhi persepsi proses filter warna yang dilakukan. Tetapi jika nilai *threshold saturation* adalah 70-255 dan nilai *threshold value* 53-106 maka sistem dapat beradaptasi dengan kondisi pencahayaan di *indoor* dan *outdoor*.

4.6. Pengujian Deteksi Jalur

Pada penelitian ini nilai H sebesar *Hmin* = 135 dan *Hmax* = 179, nilai S sebesar *Smin* = 70 dan *Smax* = 255, dan nilai V sebesar *Vmin* = 53 dan *Vmax* = 106, agar alat dapat mendeteksi jalur yang ditentukan. Pada proses uji coba ini menggunakan metode *thresholding* untuk membedakan warna hitam dan putih. Penentuan nilai *HSV* bisa dilihat pada Gambar 10, 11 dan 12.

Gambar 4.2 Penentuan Nilai *Hue*

Gambar 4.3 Penentuan Nilai *Saturation*



Gambar 4.4 Penentuan Nilai *Value*

4.7. Hasil Keluaran

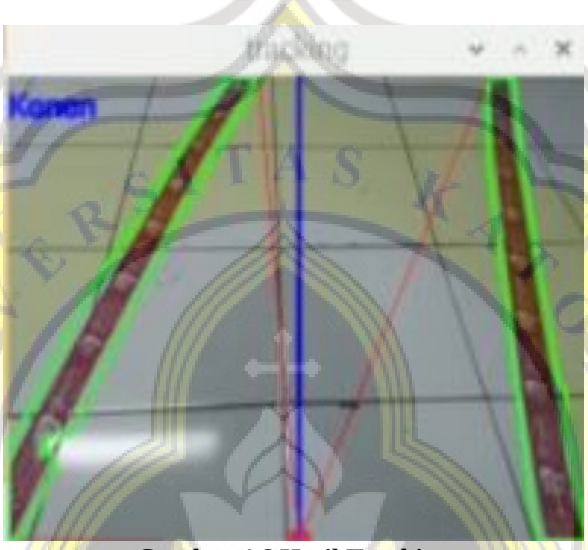
Hasil keluaran merupakan jendela hasil akhir dimana sudah melalui proses *thresholding* dan *gaussian filter* yang menghasilkan filtering warna yang bagus tanpa *noise.* Setelah melalui proses penentuan nilai *HSV* maka hasil akhir akan terlihat pada Gambar4.5.

Gambar 4.5 Hasil Keluaran

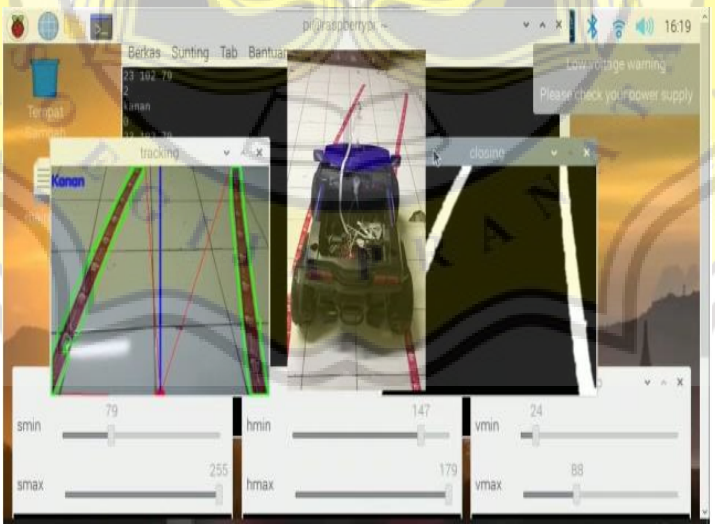
4.8. Hasil Percobaan *Self-Driving*

Pada tahap ini dilakukan uji coba *self-driving* menggunakan algoritma *HSV* terhadap jalur lintasan. Pada proses ini mobil dapat berjalan otomatis mengikuti jalur secara *real-*

time. Pergerakan mobil ditentukan dari proses tracking menggunakan sumbu (x,y) dimana memanfaatkan titik tengah yang berwarna biru untuk mengatur kondisi pergerakan mobil. Jika sudut garis merah lebih besar ke kanan maka mobil akan berbelok kanan. Jika sudut garis merah lebih besar ke kiri maka mobil akan berbelok kiri. Jika posisi sudut sama besarnya maka mobil akan berjalan lurus. Untuk hasil tracking dan percobaannya dapat diamati pada Gambar 14 dan 15.



Gambar 4.6 Hasil Tracking



Gambar 4.7 Hasil Percobaan *Self-Driving*