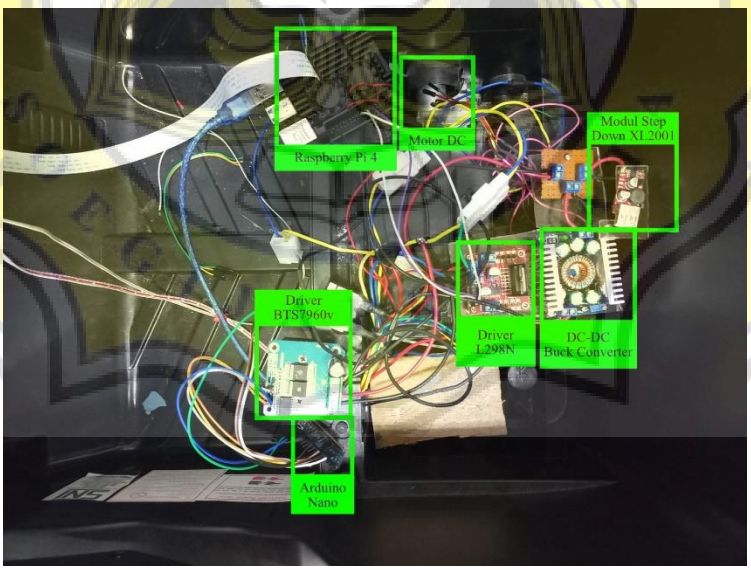
BAB IV
HASIL DAN PEMBAHASAN

4. 1. Pendahuluan

Bab ini akan dijelaskan hasil dari *prototype autonomous car*, program, dan pengujian alat untuk penelitian deteksi lintasan yang digunakan pada *autonomous car*. Uji hasil dan pembahasan ini akan menyajikan hasil prototype Alat, Program yang digunakan, pengujian *color filtering*, dan Pengujian Kecepatan Putaran roda.

4. 2. Prototype Alat

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



Gambar 4. 1 *Hardware autonomous car*

4. 3. Program

Berikut ini merupakan pembahasan program untuk sistem pergerakan *autonomous car* pada 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 yang sebelumnya telah 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)
```

Program diatas berfungsi sebagai pembacaan serial real time video yang diimplementasikan pada *autonomous car* dimana program diatas ini berisi

penambahan prosesing sebagai pengenalan maupun deteksi lintasan *street mark* dan *street background* tampilan hasil deteksi ini akan ditampilkan dalam layar monitor. Sehingga jalur terdeteksi dan pembacaan akan langsung ditampilkan pada monitor.

```
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]
```

Program pada gambar diatas ini adalah program deteksi HSV yang memiliki fungsi untuk menetapkan nilai dari *hue*, *saturation,* dan *value*, sehingga deteksi lintasan dapat lebih mudah dilakukan.

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 tingkat keakurasian pada suatu sudut yang telah ditentukan yang merupakan titik awal penempatan *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
```

main.py

```
        ykiri=0
        #print("baru")
        for cntr in contours:
            perimeter = cv2.arcLength(cntr,True)
            epsilon = 0.005*cv2.arcLength(cntr,True)
            approx = cv2.approxPolyDP(cntr,epsilon,True)
            #print(approx)
            polygonelist.append(approx)
        cv2.drawContours(frame, polygonelist, -1, (0, 255, 0), 2)
        for x in range(0,len(polygonelist)):
            for a in range(0,len(polygonelist[x])):
                for b in range(0,len(polygonelist[x][a])):
                    if(x==0):
                        kanan.append(polygonelist[x][a][b])
                        kananrev.append(polygonelist[x][a][b])
                    elif(x==1):
                        kiri.append(polygonelist[x][a][b])
                        kirirev.append(polygonelist[x][a][b])
        #for k in range(0,len(kanan)):
        #    print(kanan[k][1])
        kanan.sort(key=sortSecond)
        kiri.sort(key=sortSecond)
        kananrev.sort(key=sortSecond,reverse=True)
        kirirev.sort(key=sortSecond,reverse=True)
```

Program diatas adalah progam kontrol motor DC yang berfungsi untuk melakukan pergerakan pada *autonomous car* untuk mententukan antara berbelok kanan atau kiri. Pergerakan ini ditentukan nilai HSV yang sebelumnya telah dilakukan.

main.py

```
        for k in range (0,2):
            if(k==0):
                xkanan=kanan[k][0]
                xkiri=kiri[k][0]
                ykiri=kanan[k][1]
                ykanan=kiri[k][1]
            elif(k==1):
                if(kanan[k][0]<xkanan):
                    xkananatas=kanan[k][0]
                    ykananatas=kanan[k][1]
                else:
                    xkananatas=xkanan
                    ykananatas=ykanan
                if(kiri[k][0]>xkiri):
                    xkiriatas=kiri[k][0]
                    ykiriatas=kiri[k][1]
                else:
                    xkiriatas=xkiri
                    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 didalam lintasan dan tidak keluar lintasan pada jalur *street mark* yang telah ditentukan. Pada program diatas dijelaskan ada program HIGH dan LOW program ini sendiri untuk mengatur suatu kecepatan motor DC.

main.py

```
205         cv2.putText(frame, posisi, (0,20), cv2.FONT_HERSHEY_SIMPLEX,
206                     0.5, (255, 0, 0), 2, cv2.LINE_AA)
207         cv2.circle(frame, (int(panjang/2),lebar), radius=2, color=(0, 0, 255), thickness=10)
208         cv2.line(frame, (int(panjang/2),0), ((int(panjang/2),lebar)), (255, 0, 0), 2)
209         cv2.line(frame, (xkananatas,ykananatas), ((int(panjang/2),lebar)), (0, 0, 255), 1)
210         cv2.line(frame, (xkiriatas,ykiriatas), ((int(panjang/2),lebar)), (0, 0, 255), 1)
211         cv2.line(frame, (xkananbawah,ykananbawah), ((int(panjang/2),lebar)), (0, 0, 255), 1)
212         cv2.line(frame, (xkiribawah,ykiribawah), ((int(panjang/2),lebar)), (0, 0, 255), 1)
213         #cv2.line(frame, (xkanantengan,ykanantengan), ((int(panjang/2),lebar)), (0, 0, 255),
214         #cv2.line(frame, (xkiritengan,ykiritengan), ((int(panjang/2),lebar)), (0, 0, 255), 1)
215         #cv2.imshow('HueComp',hthresh)
216         #cv2.imshow('SatComp',sthresh)
217         #cv2.imshow('ValComp',vthresh)
218         cv2.imshow('closing',closing)
219         cv2.imshow('tracking',frame)
220         k = cv2.waitKey(5) & 0xFF
221         if k == 27:
222             break
223
224     cap.release()
225
226     cv2.destroyAllWindows()
227
```

Program diatas adalah progam yang berfungsi untuk menampilkan trackbar yang berfungsi untuk mengatur Hue, Saturation dan Value.

```
int RPWM = 11;
int LPWM = 3;
int L_EN = 2;
int R_EN = 12;
String msg;

unsigned long interval = 1000; // the time we need to wait
unsigned long previousMillis = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(RPWM, OUTPUT);
  pinMode(LPWM, OUTPUT);
  pinMode(R_EN, OUTPUT);
  pinMode(L_EN, OUTPUT);

  digitalWrite(R_EN, HIGH);
  digitalWrite(L_EN, HIGH);
}
```

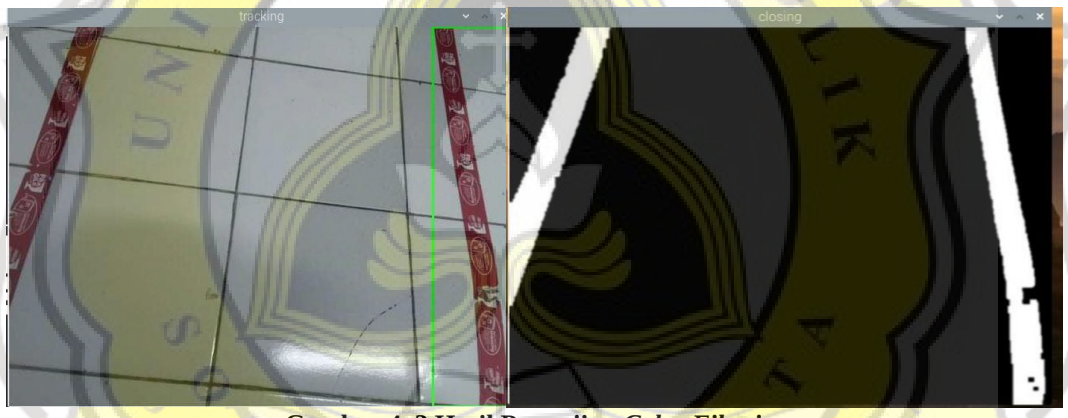
Sedangkan program yang digunakan pada aplikasi Arduino ide seperti yang ditampilkan diatas, berfungsi untuk penempatan alamat dan juga penempatan pin yang terpasang pada Arduino untuk dapat disambungkan ke driver motor L298N.

```
void loop() {
  //  forward
  unsigned long currentMillis = millis();
  if ((unsigned long)(currentMillis - previousMillis) >= interval) {
    int nilai_sensor = analogRead(A5);
    Serial.println(nilai_sensor);
    previousMillis = millis();
  }

  if (Serial.available() > 0) {
    while (Serial.available() > 0) {
      msg += (char)Serial.read();
    }

    if (msg == "m") {
      analogWrite(LPWM, 0);
      analogWrite(RPWM, 30);
      delay(1000);
      msg = "";
//      Serial.print("oke");
    }
    else {
      analogWrite(LPWM, 250);
      analogWrite(RPWM, 250);
      msg = "";
    }
  }
```

Pada program arduino diatas berfungsi sebagai serial connector, apabila program mulai *looping* maka akan mengaktifkan *serial connector* antara Arduino nano, Raspberry Pi 4, dan juga sebagai pengatur PWM.

4. 4. Pengujian Color Filtering

Tahap ini akan dilakukan proses pengujian deteksi jalur dengan menggunakan metode *Region of Interest* untuk membedakan antara *street mark* dengan *sreet background*, dan juga menggunakan metode HSV *color filtering* dengan cara mengubah citra RGB menjadi citra *grayscale*, selanjutnya gambar *grayscale* dirubah menjadi gambar *threshold* menjadi warna hitam (0), dan juga putih (255) tanpa adanya gradasi. Pengujian terhadap penentuan nilai *threshold hue, saturation,* dan *value* dilakukan untuk membedakan *street mark* dan *street background* Sesuai yang ditampilkan pada Gambar 4.2 dibawah



Gambar 4. 2 Hasil Pengujian *Color Filtering*

Nilai threshold *hue, saturation,* dan *value* yang memiliki ketetapan nilai (0), dan nilai maksimal (179) untuk *hue*, dan (255) untuk nilai *saturation* dan *value*. Setelah itu gambar *threshold* akan diproses menggunakan fungsi menjadi putih seperti *street mark* yang telah dibuat, Dan pada data nilai saturation didapati hasil bahwa threshold saturation mendeteksi objek hitam lebih maksimal dibandingkan dengan objek putih. Sedangkan pada data nilai value didapati hasil bahwa threshold value mendeteksi objek putih maupun objek hitam pada nilai minimal. Berdasarkan hasil yang diperoleh pada Tabel 1 dibawah tentang kalibrasi dan pengujian *threshold*

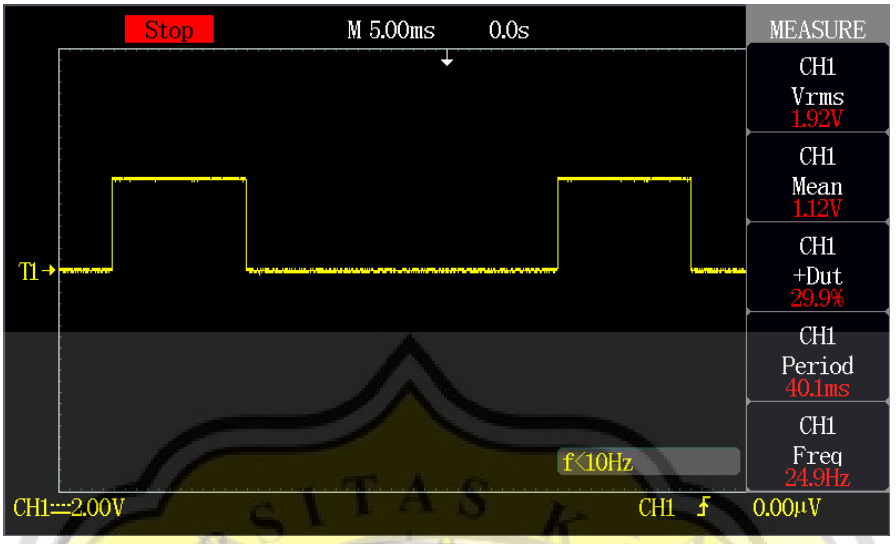
terhadap nilai hue, saturation, dan value didapati hasil bahwa nilai hue mampu mendeteksi objek putih dan objek hitam secara maksimal.

Tabel 1 Pengujian *Color Filtering*

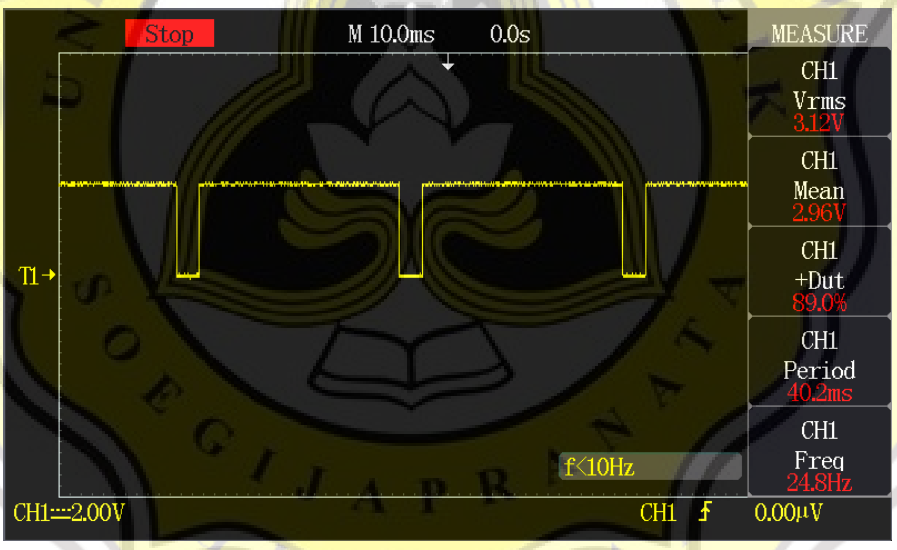
Parameter	Nilai Min	Nilai Max	Objek Putih	Objek Hitam
Hue	155	179	Terdeteksi	Terdeteksi
Saturation	79	255	Sebagian	Terdeteksi
Value	44	94	Sebagian	Sebagian

4. 5. Pengujian Kecepatan Putaran Roda Berdasarkan Input PWM

Pada pengujian ini bertujuan untuk mengatahui kecepatan *autonomous car* secara linear, sekaligus untuk mengetahui kemampuan kecepatan pada *autonomous car* baik pada saat jalur lurus dan jalur berbelok. Kecepatan putaran, dan kecepatan linear dipengaruhi oleh nilai PWM yang ditentukan, motor DC yang digunakan dan jumlah yang digunakan, diameter roda serta berat pada *autonomous car* itu sendiri. Pengujian ini dilakukan menggunakan tachometer, yang kecepatan putaran (RPM) rodanya telah menggunakan tachometer. Pada pengujian ini, *autonomous car* memiliki berat 4,8 kg, dan memiliki jari-jari roda 14 cm. Pengujian kecepatan putaran roda dilakukan dengan *dutycycle* yang berbeda, dimulai dengan nilai pwm *dutycyle* yang paling rendah yaitu 30% dan hasil sinyal keluaran yang dihasilkan ditampilkan pada Gambar 4.3 dibawah ini. Dan nilai pwm *dutycyle* yang paling tinggi yaitu 90%, hasil sinyal keluaran ditampilkan pada Gambar 4.4 dibawah ini.



Gambar 4. 3 PWM *duty cycle* 30%



Gambar 4. 4 PWM *duty cycle* 90%

Pada nilai pwm *dutycyle* 30% didapati hasil kecepatan putarannya ialah 16,2 Rad/s dan kecepatan linearnya yaitu 2,26 m/s. Dan pada nilai *dutycyle* 60% didapati hasil kecepatan putarannya adalah 37,17 Rad/s dan kecepatan linearnya adalah 5,2 m/s. Pada pengujian yang terakhir nilai pwm *dutycyle* dinaikkan menajadi 90% yang menyebabkan kecepatan putarnnya menjadi jauh lebih besar yaitu 53,56 Rad/s dan

kecepatan linearnya sebesar 7,49 m/s. Sehingga dari data yang diperoleh pada Tabel 3 didapatkan nilai rata-rata 36,01 Rad/s dan kecepatan linearnya sebesar 5,03 m/s. Hasil pengukuran kecepatan putar roda dan kecepatan linear ditampilkan pada Tabel 2 dibawah.

Tabel 2 Hasil Pengujian Kecepatan Putar Roda

PWM Dutycycle (%)	Kecepatan Putaran (RPM)	Kecepatan Putaran (Rad/s)	Kecepatan Linear (m/s)
30	155	16,2	2,26
50	255,7	26,75	3,74
60	355,4	37,17	5,2
80	443,8	46,4	6,49
90	512,2	53,56	7,49
Rata-rata	344,42	36,01	5,03