JCONES Vol. 8, No. 2 (2019) 1-12

Journal of Control and Network Systems

Situs Jurnal : http://jurnal.stikom.edu/index.php/jcone

# RANCANG BANGUN *MOBILE ROBOT VISION* PENGANTAR MAKANAN PADA SEBUAH RESTORAN

**Muhammad Faris Akbar** [1] **Susijanto Tri Rasmana** [2] **Harianto** [3]
Program Studi/Jurusan Teknik Komputer
Institut Bisnis dan Informatika Stikom Surabaya
Jl. Raya Kedung Baruk 98 Surabaya, 60298s
Email: 1)15410200070@stikom.edu 2)susyanto@stikom.edu, 3)harianto@stikom.edu

**Abstrak:** Di kafe klasik, restoran dan hotel, pelanggan menghadapi banyak masalah karena banyaknya pekerjaan pelayan pada pelanggan, tidak tersedianya pelayan dan pemesanan makanan masih secara manual. Kekurangan ini dapat ditangani dengan menggunakan sebuah sistem otomatisasi restoran seperti mengganti pelayan dengan menggunakan *Mobile Robot*. Pada riset sebelumnya *Mobile Robot* yang digunakan atau yang diteliti rata-rata masih menggunakan sensor Photodioda untuk membaca garis yang dibuat sebagai rute. Oleh karena itu, pada Tugas Akhir ini telah mengembangkan pembacaan garis rute menggunakan kamera Webcam sebagai komponen utama dalam pengolahan citra serta menggunakan logika *maze mapping* untuk penyelesaian masalah dalam menentukan rute dalam setiap destinasinya, sehingga pengembangan ini menciptakan istilah robot baru yaitu *Mobile Robot Vision*. Hasil pengujian yang didapatkan pada Tugas Akhir ini berupa robot dapat melaksanakan setiap tujuan melayani satu meja atau dua meja baik dengan beban atau tanpa beban, dengan waktu tempuh rata-rata dan kecepatan rata-rata dipengaruhi beban yang dibawa, semakin berat beban semakin lama waktu robot berjalan dan semakin berkurang kecepatan robot dan keberhasilan rata-rata didapatkan hingga 100%.

**Kata kunci:** *Mobile Robot*, *Maze Mapping*, robot, robot pengantar makanan, pengolahan citra, restoran.

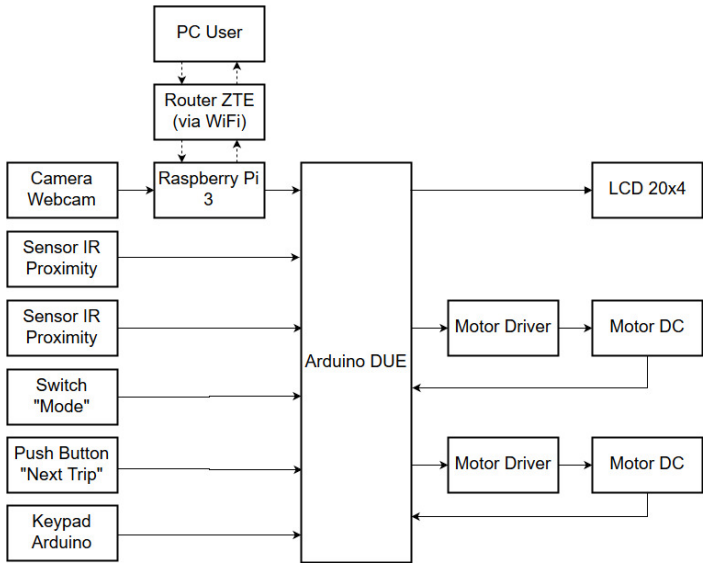
## PENDAHULUAN

Pelanggan adalah nyawa dari setiap bisnis yang dibangun. Bisnis tidak bisa tumbuh dengan besar tanpa adanya dukungan dan dorongan dari pelanggan yang mempercayakan kebutuhan mereka kepada bisnis tersebut. Salah satu cara mendapatkan kepercayaan dari pelanggan yakni pelayanan pengantaran hidangan atau makanan dengan cepat dan tepat oleh pelayan.

Di kafe klasik, restoran dan hotel, pelanggan menghadapi banyak masalah karena banyaknya pekerjaan pelayan pada pelanggan, tidak tersedianya pelayan dan pemesanan makanan secara manual. Kekurangan ini dapat ditangani dengan menggunakan sebuah sistem otomatisasi restoran seperti menggunakan "Pelayan Robot" yang membantu untuk mengantarkan makanan atau minuman. (Asif, 2015)

Robot yang biasanya digunakan dalam aktivitas kehidupan sehari-hari adalah *Mobile Robot*. *Mobile Robot* adalah robot yang memiliki aktuator roda atau kaki untuk memindahkan robot ke setpoint. Salah satunya adalah robot pembantu. Ini memiliki tujuan untuk membuat pelanggan memesan menu dengan mudah dan mempermudah pekerjaan bagi staf dan pemilik restoran. (Safii, 2017)

Pada penelitian sebelumnya yang dilakukan oleh (Safii, 2017) dalam jurnal yang berjudul "Mobile Robot Pembawa Peralatan Makan Kotor Otomatis Pada Sebuah Restoran" dan (Asif, 2015) dalam jurnalnya yang berjudul "Waiter Robot - Solution to Restaurant Automation". Pada kedua jurnal tersebut menjelaskan robot yang digunakan adalah robot *line tracer* atau *line follower* yang memakai Photodioda sebagai sensor pendeteksi garis yang digunakan untuk rute robot.




Gambar 1. Blok diagram keseluruhan

Sedangkan kelemahan dari sensor garis ini yaitu sensor hanya dapat mengerti jarak antara robot dengan garis, itupun dengan ketelitian yang terbatas, tergantung dari jumlah sensor yang dipakai dan peletakannya. Hal ini menyebabkan robot *line follower* tidak mengetahui besar simpangan antara robot dengan garis seperti ketika membaca garis perempatan ataupun persimpangan, robot mendeteksi garis berdasarkan garis yang terlebih dahulu terdeteksi oleh sensor dan juga kurangnya ketelitian dalam membaca posisi garis, sehingga terkadang robot bingung dalam pemilihan garis yang dilewati saat di perempatan ataupun di persimpangan. Salah satu solusi untuk mengatasi masalah tersebut yaitu dengan mengganti sensor pendeteksi garis yang memakai Photodioda dengan menggunakan kamera Webcam.

Kamera Webcam sebagai *vision* (penglihatan) dapat membaca garis dengan memanfaatkan warna dari garis, sehingga robot dapat berjalan mengikuti warna jalur yang digunakan sebagai rute serta robot dapat mengantarkan makanan pada meja makan pelanggan dengan rute yang sesuai menggunakan algoritma *Maze Mapping* pada tujuan yang telah dipilih sebelumnya dengan keypad, tombol *push button* untuk melanjutkan destinasi dan *switch mode* untuk menentukan robot maju menuju destinasi atau mundur kembali ke posisi awal (*home*).

## METODE PENELITIAN

Pada bagian ini pembahasan terbagi menjadi 2 bagian seperti berikut:

### *A. Perancangan Perangkat Keras*

Pada pembahasan ini, dijelaskan mengenai Blok Diagram, desain rancang bangun robot, dan bentuk hasil dari rancang bangun. Berikut penjelasannya:

Pada blok diagram pada gambar 1 terbagi menjadi 4 bagian yakni input, pengolahan citra, pengolahan data, dan output. Input terdiri dari kamera Webcam yang masuk ke Raspberry Pi 3 untuk membaca garis, sensor IR Proximity untuk mendeteksi adanya halangan di depan robot, switch sebagai "mode" untuk "maju" atau "balik", push button sebagai "next trip" untuk melanjutkan perjalanan dari meja pertama ke meja kedua, dan keypad arduino sebagai input nomor meja yang dituju oleh robot.

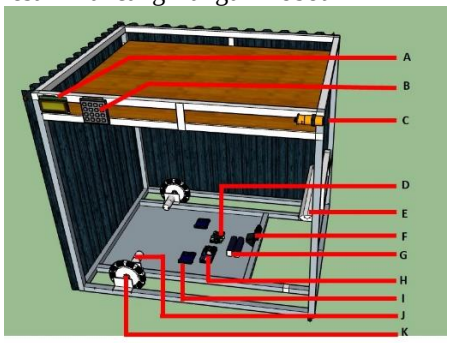
Pada bagian pengolahan citra terdapat Raspberry pi 3, Router ZTE dan PC User. Pertama PC user akses Raspberry Pi 3 dengan menggunakan via Wifi yang oleh Router ZTE sebagai *access point* dengan aplikasi Remote Desktop Connection. Dengan memasukkan alamat IP dari Raspberry Pi 3, lalu masukkan username dan password. Kemudian buka aplikasi Terminal lalu ketikkan sintaks untuk akses file OpenCV, lalu *compile* file dan di-*running* untuk menjalankan OpenCV 2.1.0. Setelah menjalankan program OpenCV, data output dari kamera berupa citra kemudian diolah dalam beberapa metode. Lalu nilai yang telah diolah

dikirim ke Arduino DUE melalui PORT_USB_Programming.

Pada bagian pengolahan data terdapat pada Arduino DUE sebagai pengolahnya menerima beberapa data input yakni dari Raspberry Pi 3 berupa nilai koordinat dan nilai dari karakter abjad yang dikirim secara serial, sensor IR Proximity berupa sinyal digital high atau low, keypad arduino berupa nilai digital 8 input, Switch "Mode" dan Push Button "Next Trip" yang difungsikan pada program *Maze Mappping*.

Pada bagian output terdapat Motor Driver EMS 30A H-Bridge yang bersambungan dengan Motor DC PG-45, dan LCD 20x4. Motor Driver sebagai pengatur gerak dari Motor DC PG-45 dengan menerima sinyal PWM yang diberikan oleh Arduino DUE, Motor DC PG-45 mempunyai *rotary encorder* sabagai input Arduino DUE menerima jumlah pulse selama motor berputar. LCD 20x4 menampilkan nomor meja 1 dan meja 2, status kamera serta mode yang digunakan pada robot.

2. Desain Rancang Bangun Robot




Gambar 2. Desain rancang bangun robot

Keterangan :
A = LCD 20x4
B = Keypad Arduino
C = Sensor IR Proximity
D = Raspberry Pi 3
E = Lampu
F = Kamera Webcam
G = Baterai 12V dan 24V
H = Arduino DUE
I = EMS 30A H-Bridge
J = Motor DC 24V
K = Roda

3. Interface Rancang Bangun Robot


Gambar 3. Tombol interface robot

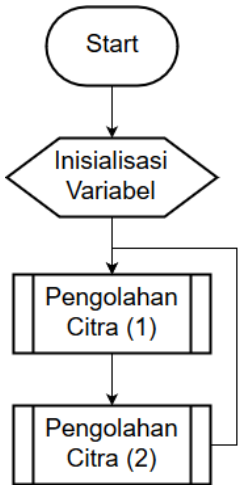
Keterangan:
a. Switch hijau (Tombol Mode) digunakan untuk menentukan robot bergerak menggunakan mode "Maju" atau "Balik".
b. Switch merah (Tombol Baterai 12/24 V) digunakan untuk menyalakan robot.
c. Push Button hijau (Tombol Next Trip) digunakan untuk memberikan robot aksi melanjutkan destinasi meja berikutnya sesuai dengan inputan.
d. Push Button kuning (Tombol Emergency/RESET) digunakan untuk mereset robot ketika dalam keadaan error atau memulai kembali proses pengantaran makanan.


Gambar 4. Tampilan Display LCD dan Keypad Arduino

***B. Perancangan Perangkat Lunak***

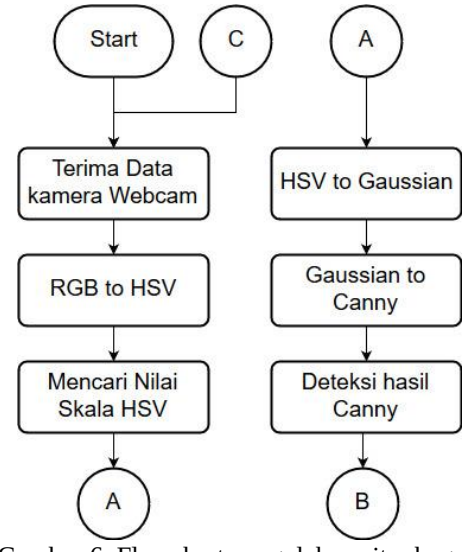
Pada pembahasan ini, dijelaskan mengenai program pengolahan citra pada Raspberry Pi 3, program pengolahan data pada Arduino DUE, dan denah meja. Berikut penjelasannya:

1. Pengolahan Citra pada Raspberry Pi 3



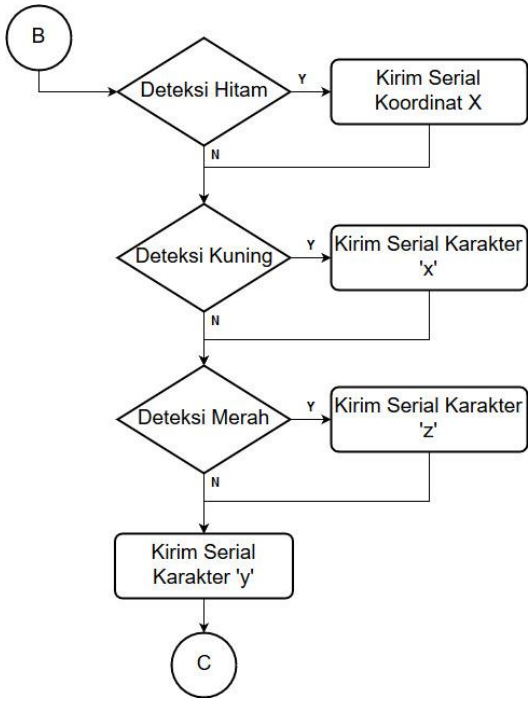
Gambar 5. Flowchart keseluruhan Raspberry Pi 3

Pada gambar 5 flowchart yang digunakan sangatlah detail dan rinci sehingga perlu penjabaran pada tiap simbol-simbol flowchart yang digunakan. Pada awal mula ketika Raspberry Pi 3 menyala maka memulai inisialisasi variabel lalu kemudian lanjut program Proses Pengolahan Citra (1) pada gambar 6, lalu setelah itu lanjut ke program Proses Pengolahan Citra (2) pada gambar 7.



Gambar 6. Flowchart pengolahan citra bag. 1

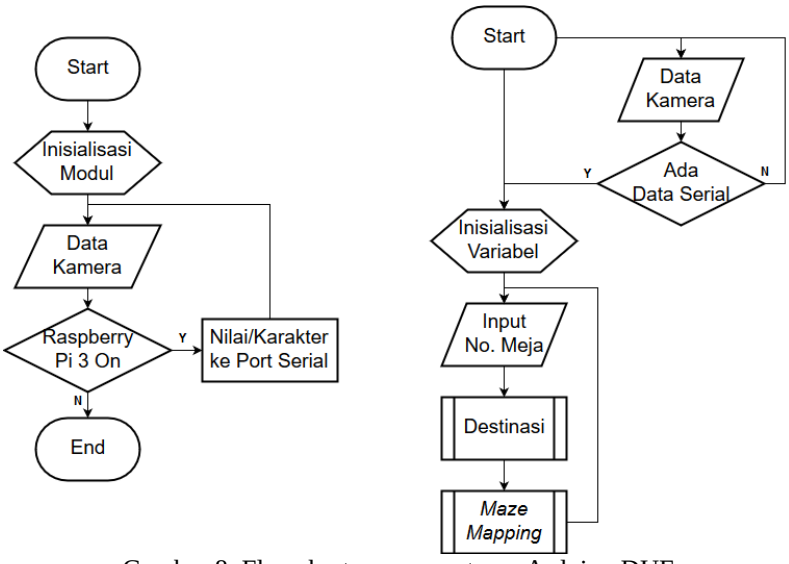
Setelah melakukan inisialisasi variabel, Raspberry Pi 3 menerima data kamera berupa citra dalam format RGB. Kemudian citra yang ditangkap dikonversi dari RGB ke HSV, lalu mencari nilai HSV nya dalam bentuk skala. Citra yang telah didapatkan nilai HSV-nya dikonversi untuk menghilangkan *noise* dengan cara meng-blurkan menggunakan metode *Gaussian-blur* untuk mendapatkan citra yang bersih. (Wibisono, 2011) Setelah di-blurkan citra dideteksi tepi garisnya dengan metode *Canny Edge Detection*, setelah mendapatkan tepi garisnya, citra selanjutnya dicari bentuk lingkarannya untuk menandakan objek berdasarkan warna HSV yang telah ditentukan sebelumnya dan dibatasi pembacaannya dengan hasil citra *Edge Detection*. Dengan metode *Hough Circle Transform* didapatkan bentuk lingkaran dari citra yang ditangkap. (Nalwan, 2010)



Gambar 7. Flowchart pengolahan citra bag. 2

Setelah objek didapatkan maka masuk pada kondisi dari masing-masing warna HSV objek yang dideteksi, pada percabangan pertama kondisi warna hitam jika IYA mendeteksi warna maka objek tersebut diambil nilai koordinat X-nya lalu dikirim ke Arduino DUE.

Jika TIDAK maka lanjut ke percabangan kedua yakni deteksi warna kuning, jika IYA maka kirim data serial karakter ‘x’ ke Arduino DUE. Jika TIDAK maka lanjut ke percabangan ketiga yakni deteksi warna merah, jika IYA maka kirim data serial karakter ‘z’ ke Arduino DUE. Jika TIDAK mendeteksi ketiga warna tersebut maka kirim serial karakter ‘y’ ke Arduino DUE dan kembali ke program Pengolahan Citra bagian 1.




Gambar 8. Flowchart program utama Arduino DUE

2. Pengolahan Data pada Arduino DUE

Pada pembahasan materi disini menjelaskan program Arduino yang dibagi menjadi 3 bagian yakni flowchart program utama, flowchart program cabang dan flowchart sub-cabang.

a. Flowchart Program Utama

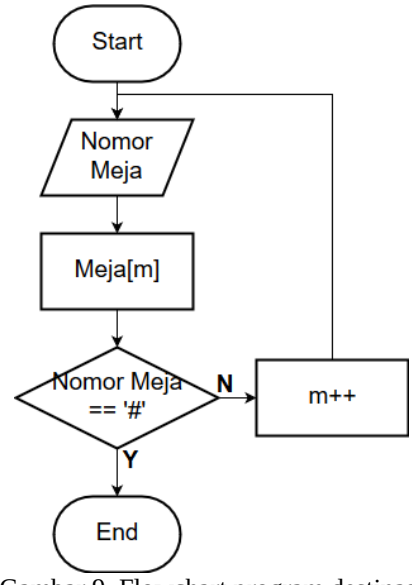
Pada gambar 8 terdapat dua program yang berjalan yakni program untuk Raspberry Pi 3 yang dijelaskan secara umum (penjelasan rinci telah dijelaskan pada bagian Pengolahan Citra) dan program utama Arduino DUE yang dijelaskan secara umum. Pertama Raspberry Pi 3 memulai inisialisasi modul berupa kamera Webcam, disisi lain Arduino DUE memulai inisialisasi variabel dan menunggu terima data kamera. Hasil tangkapan kamera dikirim ke Raspberry Pi 3 lalu diolah dan dikirim ke Arduino DUE, Arduino DUE menerima nilai data kamera lalu inputkan nomor meja yang dituju dan masuk pada program cabang Destinasi pada gambar 9, setelah selesai lanjut menjalankan program cabang *Maze Mappping* pada gambar 10 dan kembali melanjutkan dari awal menjalankan program utama.

b. Flowchart Program Cabang

Program cabang merupakan program yang dijalankan di program utama Arduino DUE. Terdiri dari 2 program, yakni program Destinasi dan *Maze Mapping*.

(1) Program Destinasi

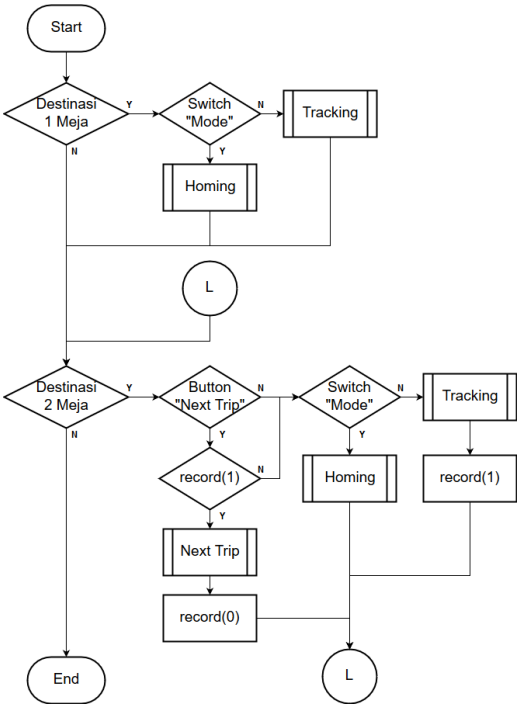
Pada gambar 9, program dimulai dengan mengiputkan nomor meja dari keypad lalu masuk pada variabel array, kemudian cek kondisi apabila nomor TIDAK diinputkan karakter '#' maka variabel array bertambah satu untuk input meja selanjutnya. Jika IYA maka program langsung diakhiri dan menuju ke program selanjutnya.




Gambar 9. Flowchart program destinasi

(2) Program *Maze Mapping*

Pada gambar 10 program terdapat 2 kondisi yang dijalankan yaitu ketika menjalankan kondisi destinasi 1 meja dan destinasi 2 meja. Jika kondisi pertama yakni destinasi 1 meja terpenuhi maka program menjalankan program sub-cabang Tracking bila tidak ada penekanan Switch "Mode". Jika terdapat




Gambar 10. Flowchart program *Maze Mapping*

penekanan maka menjalankan program sub-cabang Homing karena tujuan destinasi meja hanya satu.

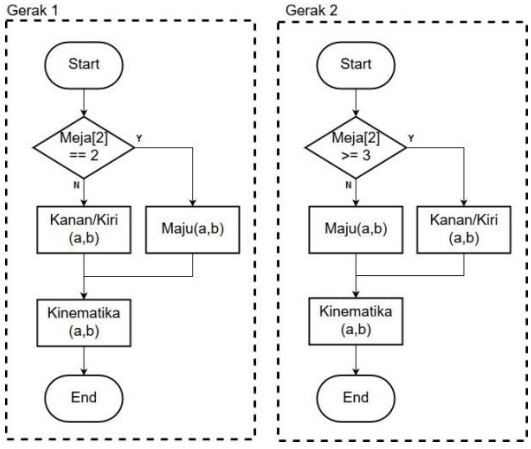
Bila yang terpenuhi yakni destinasi 2 meja, maka program menjalankan program sub-cabang Tracking bila tidak ada penekanan Push Button "Next Trip" dan Switch "Mode". Jika kondisi Tracking terpenuhi maka mengaktifkan sebuah variabel untuk menjalankan program Next Trip. Setelah menjalankan program Next Trip maka variabel sebelumnya dinonaktifkan, penekanan Switch "Mode" untuk menjalankan program Homing setelah mencapai destinasi meja kedua

c. Flowchart Program Sub-Cabang

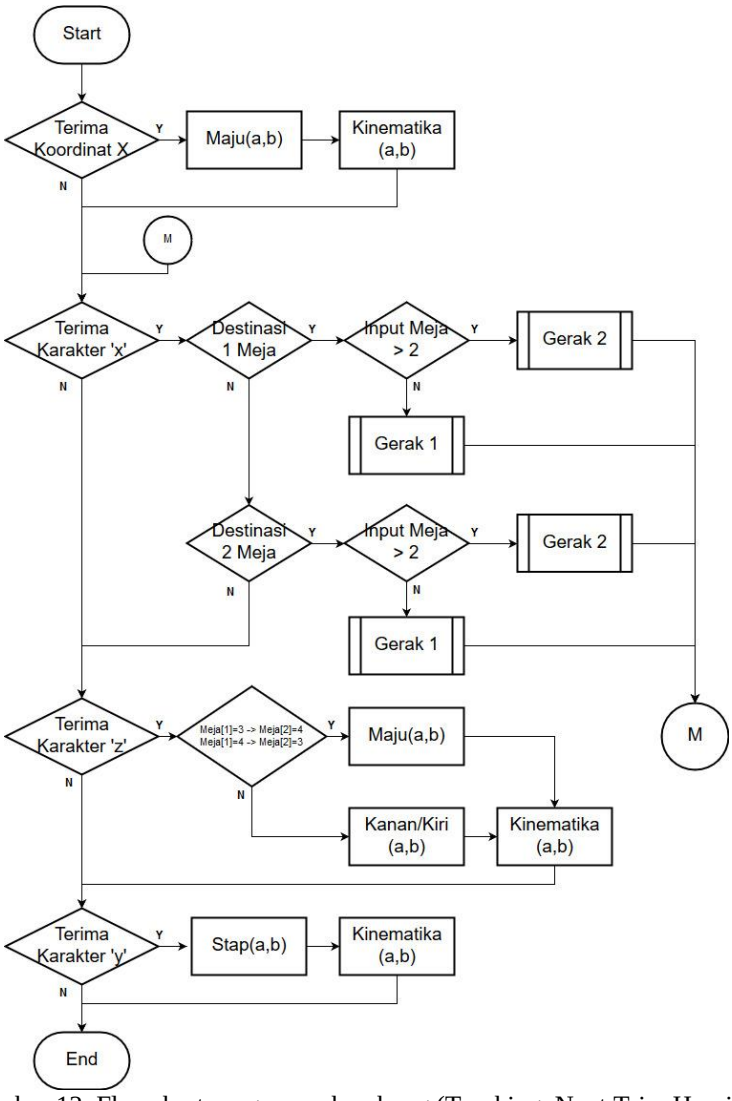
Pada gambar 12, program sub-cabang merupakan program yang dijalankan di program cabang, terutama program cabang *Maze Mapping* yang terdiri dari tiga program sub-cabang yakni Tracking, Next Trip, dan Homing. Dalam flowchart yang dicantumkan terdapat 4 kondisi yaitu:

(a) Kondisi pertama menerima nilai koordinat x atau bisa diartikan jika kamera mendeteksi adanya garis berwarna hitam maka program menjalankan perintah maju dan bergerak dengan kinematika menyesuaikan paracentimeter yang telah dibuat.

(b) Kondisi kedua ketika menerima karakter 'x' atau kamera sedang mendeteksi garis berwarna kuning maka program menyesuaikan destinasi meja yang dipilih. Jika destinasi hanya satu meja saja maka menjalankan perintah sesuai input meja yang dimasukkan yakni: Jika meja yang diinputkan nomor 1 dan 2 maka menjalankan program Gerak 1 dan jika meja yang diinputkan diatas nomor 2 yakni nomor 3 atau 4, maka menjalankan program Gerak 2. Program Gerak 1 dan Gerak 2 menjalankan perintah sesuai kondisi (karena kondisi destinasi tidak mendapati variabel 'Meja[2]' diinputkan nilai maka menjalankan kondisi TIDAK pada masing-masing kondisi pada tiap program) dan parameter yang diinputkan.
Sedangkan, jika destinasi meja yang dipilih dua maka menjalankan perintah sesuai dengan nomor inputan yang diberikan, Jika meja yang diinputkan nomor 1 atau 2 maka menjalankan program Gerak 1, dan jika meja yang diinputkan diatas nomor 2 yakni nomor 3 atau 4, maka menjalankan program Gerak 2. Program Gerak 1 dan Gerak 2 menjalankan perintah sesuai kondisi (karena kondisi destinasi mendapati variabel 'Meja[2]' diinputkan nilai maka menjalankan kondisi YA pada masing-masing kondisi pada tiap program) dan parameter yang diinputkan. Penjelasan lengkap dapat dilihat pada gambar 11. (Akbar, 2013)




Gambar 11. Flowchart program gerak



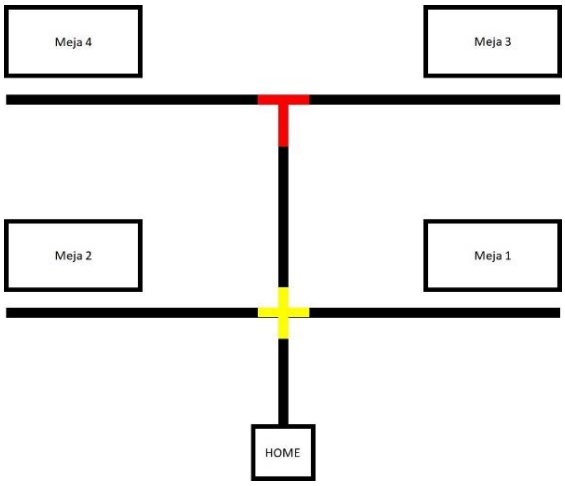

Gambar 12. Flowchart program sub-cabang (Tracking, Next Trip, Homing)

(c) Kondisi ketiga ketika menerima karakter ‘z’ atau kamera mendeteksi warna merah maka program menjalankan perintah sesuai kondisi, jika kondisi meja yang diinputkan TIDAK dalam keadaan dari meja 3 ke meja 4 atau sebaliknya, maka menjalankan Putar kanan/kiri sesuai parameter yang diinputkan. Dan jika kondisi meja yang diinputkan YA dalam keadaan dari meja 3 ke meja 4 atau sebaliknya maka menjalankan perintah Maju.

(d) Kondisi keempat ketika menerima karakter ‘y’ atau kamera mendeteksi warna selain hitam, kuning dan merah maka program menjalankan perintah stap atau stop, lalu program selesai kembali ke program cabang *Maze Mapping* di Arduino DUE. (Akbar, 2013)

3. Denah Meja

Pada gambar 13 jumlah meja yang digunakan ada 4, terdiri jalur lurus dengan warna hitam, jalur perempatan dan jalur pertigaan.



Gambar 13. Denah Meja

## HASIL DAN PEMBAHASAN

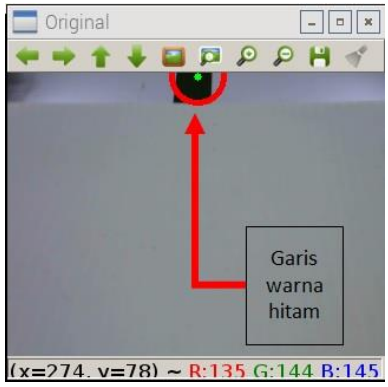
Pada pembahasan kali ini, membahas tentang pengujian *Mobile Robot Vision* berupa uji kamera deteksi garis, uji destinasi 1 meja dan destinasi 2 meja.

***A. Pengujian Kamera Deteksi Garis***

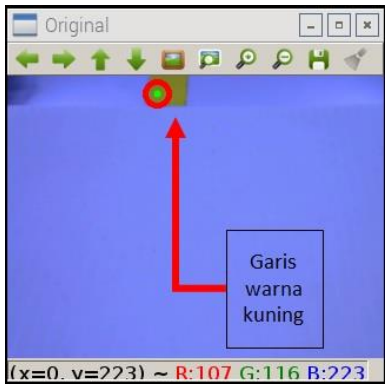
Pengujian ini digunakan untuk mengetahui apakah kamera dapat berfungsi dengan baik sehingga dapat digunakan untuk mendeteksi warna objek yang ditangkap. Peralatan yang dibutuhkan untuk pengujian ini yaitu *Mobile Robot Vision* (terutama Raspberry Pi 3), kamera Webcam Logitech C270, Laptop, Program OpenCV 2.1.0. berikut tahap-tahap untuk melakukan pengujian sebagai berikut:

1. Menangkap citra dengan format RGB

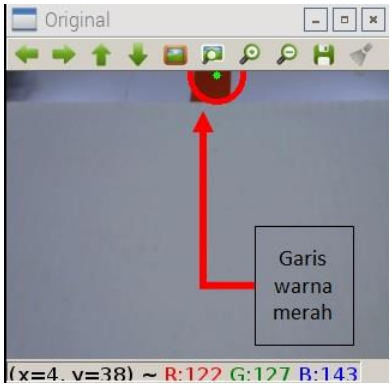
Pada tahap pertama kamera menangkap citra RGB yang kemudian ditampilkan. Berikut gambar citra RGB yang ditangkap.



Gambar 14. Citra RGB warna hitam



Gambar 15. Citra RGB warna kuning



Gambar 16. Citra RGB warna merah

2. Konversi dari bentuk RGB to HSV

Setelah mendapatkan citra RGB langkah selanjutya yakni mengkonversi format dari RGB ke HSV, sebelumnya dicari terlebih dahulu nilai HSV dalam bentuk skala yang ditangkap yakni warna HSV hitam, kuning dan merah. Berikut tabel warna HSV yang didapatkan dari masing-masing warna.

Tabel 1. Nilai skala HSV warna hitam

| **Hitam** | |
|---|---|
| **Nama Variabel** | **Nilai Skala** |
| Nilai Minimal *Hue* (H_MIN) | 0 |
| Nilai Maksimal *Hue* (H_MAX) | 255 |
| Nilai Minimal *Saturation* (S_MIN) | 0 |
| Nilai Maksimal *Saturation* (S_MAX) | 255 |
| Nilai Minimal *Value* (V_MIN) | 0 |
| Nilai Maksimal *Value* (V_MAX) | 68 |

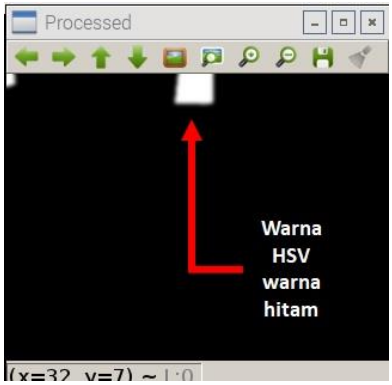
Tabel 2. Nilai skala HSV warna kuning

| Kuning | |
|---|---|
| **Nama Variabel** | **Nilai Skala** |
| Nilai Minimal *Hue* (H_MINz) | 20 |
| Nilai Maksimal *Hue* (H_MAXz) | 57 |
| Nilai Minimal *Saturation* (S_MINz) | 50 |
| Nilai Maksimal *Saturation* (S_MAXz) | 255 |
| Nilai Minimal *Value* (V_MINz) | 80 |
| Nilai Maksimal *Value* (V_MAXz) | 195 |

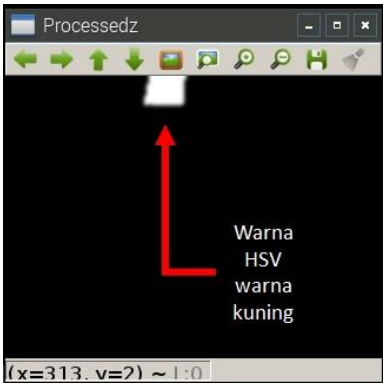
Tabel 3. Nilai skala HSV warna merah

| Merah | |
|---|---|
| **Nama Variabel** | **Nilai Skala** |
| Nilai Minimal *Hue* (H_MINr) | 0 |
| Nilai Maksimal *Hue* (H_MAXr) | 51 |
| Nilai Minimal *Saturation* (S_MINr) | 181 |
| Nilai Maksimal *Saturation* (S_MAXr) | 255 |
| Nilai Minimal *Value* (V_MINr) | 73 |
| Nilai Maksimal *Value* (V_MAXr) | 113 |

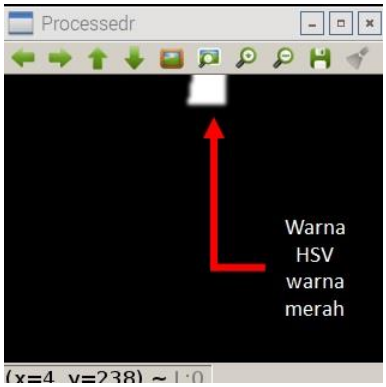
Lalu setelah mendapatkan nilai HSV-nya, maka langkah selanjutnya dikonversi ke citra HSV. Lalu disimpan dalam memori Raspberry Pi 3.



Gambar 17. Citra HSV warna hitam



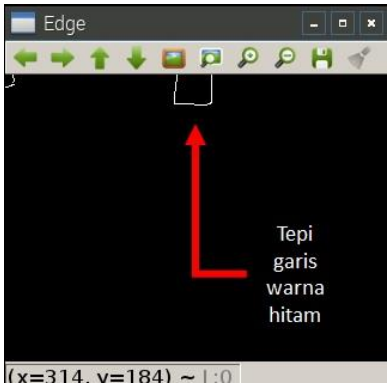
Gambar 18. Citra HSV warna kuning



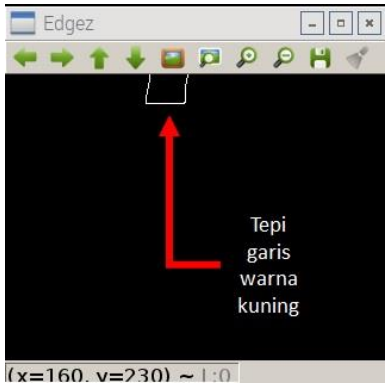
Gambar 19. Citra HSV warna merah

3. Menghilangkan *noise* dan deteksi tepi garis

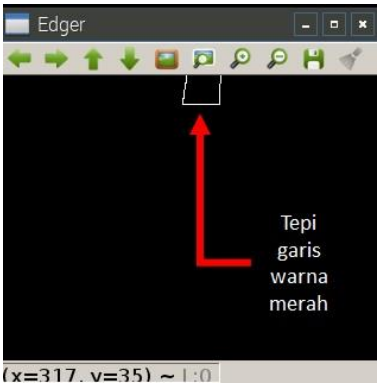
Pada tahap ini, citra HSV yang telah disimpan di dalam memori dihilangkan *noise*-nya dengan metode *Gaussian-blur* dan mendeteksi tepi garis dengan metode *Canny Edge Detection*.



Gambar 20. Citra *Edge* warna hitam



Gambar 21. Citra *Edge* warna kuning



Gambar 22. Citra *Edge* warna merah

Kemudian setelah mendapatkan tepi garisnya, citra selanjutnya dicari bentuk lingkarannya untuk menandakan objek berdasarkan warna HSV yang telah ditentukan sebelumnya dan dibatasi pembacaannya dengan hasil citra *Edge Detection*, dengan metode *Hough Circle Transform* seperti pada gambar 14, gambar 15 dan gambar 16.

### *B. Pengujian Destinasi 1 Meja*

Pengujian ini digunakan untuk mengetahui waktu tempuh robot dari posisi start (*home*) ke meja dan kembali lagi ke posisi semula, kecepatan robot dalam melakukan perjalanan berangkat hingga kembali serta persentase keberhasilan robot dalam menuntaskan destinasinya. Dengan diketahuinya kesemua itu, dapat kita lihat rata-rata waktu tempuh, kecepatan, dan keberhasilan robot. Peralatan yang dibutuhkan untuk pengujian ini yakni *Mobile Robot Vision*, Laptop, Program Arduino IDE dan Stopwatch. Berikut tabel 4 *rule* untuk menentukan persentase keberhasilan serta tabel pengujian masing-masing meja.

Tabel 4. *Rule* keberhasilan destinasi 1 meja

| **Destinasi 1 Meja** | |
|---|---|
| **Untuk pilihan meja 1 atau 2** | |
| **Keterangan** | **Persentase(%)** |
| **Fase 1** | |
| *Home*ke Perempatan | 0 – 25 |
| Perempatan ke Meja | 26 – 50 |
| **Fase 2** | |
| Meja ke Perempatan | 51 – 75 |
| Perempatan ke *Home* | 76 – 100 |
| **Untuk pilihan meja 3 atau 4** | |
| **Keterangan** | **Persentase(%)** |
| **Fase 1** | |
| *Home*ke Perempatan | 0 – 10 |
| Perempatan ke Pertigaan | 11 – 25 |
| Pertigaan ke Meja | 26 – 50 |
| **Destinasi 1 Meja** | |
| **Untuk pilihan meja 1 atau 2** | |
| **Keterangan** | **Persentase(%)** |
| **Fase 2** | |
| Meja ke Pertigaan | 51 – 60 |
| Pertigaan ke Perempatan | 61 – 75 |
| Perempatan ke *Home* | 76 – 100 |

Tabel 5. Pengujian destinasi 1 meja

| **Kalkulasi 1 Meja** | | | | |
|---|---|---|---|---|
| **Tanpa Beban** | | | | |
| **Meja** | **1** | **2** | **3** | **4** |
| **Total Waktu (s)** | 40,18 | 36,51 | 51,10 | 51,23 |
| **Kecepatan (cm/s)** | 21,45 | 25,27 | 24,48 | 24,23 |
| **Keberhasilan (%)** | 96 | 85 | 93 | 94 |
| **Beban 1 Kg** | | | | |
| **Meja** | **1** | **2** | **3** | **4** |
| **Total Waktu (s)** | 43,11 | 43,10 | 57,27 | 56,30 |
| **Kecepatan (cm/s)** | 19,53 | 19,58 | 21,20 | 21,60 |
| **Keberhasilan (%)** | 100 | 100 | 100 | 100 |
| **Beban 2 Kg** | | | | |
| **Meja** | **1** | **2** | **3** | **4** |
| **Total Waktu (s)** | 45,07 | 45,06 | 59,34 | 58,26 |
| **Kecepatan (cm/s)** | 18,68 | 18,73 | 20,46 | 20,87 |
| **Keberhasilan (%)** | 100 | 100 | 100 | 100 |
| **Beban 3 Kg** | | | | |
| **Meja** | **1** | **2** | **3** | **4** |
| **Total Waktu (s)** | 49,93 | 49,92 | 64,20 | 63,12 |
| **Kecepatan (cm/s)** | 16,87 | 16,91 | 18,91 | 19,27 |
| **Keberhasilan (%)** | 100 | 100 | 100 | 100 |
| **Beban 4 Kg** | | | | |
| **Meja** | **1** | **2** | **3** | **4** |
| **Total Waktu (s)** | 0,00 | 0,00 | 0,00 | 0,00 |
| **Kecepatan (cm/s)** | 0,00 | 0,00 | 0,00 | 0,00 |
| **Keberhasilan (%)** | 0 | 0 | 0 | 0 |

Pada tabel 5 pengujian destinasi 1 meja input meja dilakukan dengan dua cara, yaitu robot melakukan destinasi tanpa beban dan dengan beban. Dari pengujian destinasi 1 meja tanpa adanya beban, menunjukkan memiliki rata-rata keberhasilan pada tiap meja berbeda-beda. Sedangkan pada pengujian dengan adanya beban menunjukkan mempunyai tingkat keberhasilan rata-rata 100%.

### *C. Pengujian Destinasi 2 Meja*

Pengujian ini digunakan untuk mengetahui waktu tempuh dari posisi start (*home*) ke meja pertama, waktu tempuh meja pertama ke meja kedua dan kembali lagi dari meja kedua ke posisi semula, kecepatan robot dalam melakukan perjalanan berangkat, menuju ke destinasi selanjutnya hingga kembali serta persentase keberhasilan robot dalam

menuntaskan destinasinya secara keseluruhan. Dengan diketahuinya kesemua itu, dapat kita lihat rata-rata waktu tempuh, kecepatan, dan keberhasilan robot. Peralatan yang dibutuhkan untuk pengujian ini yakni *Mobile Robot Vision*, Laptop, Program Arduino IDE dan Stopwatch. Berikut tabel 6 *rule* untuk menentukan persentase keberhasilan serta tabel pengujian masing-masing meja yang berpasangan.

Tabel 6. *Rule* keberhasilan destinasi 2 meja

| **Destinasi 2 Meja** | |
|---|---|
| **Untuk pilihan meja 1 dan/atau 2** | |
| **Keterangan** | **Persentase(%)** |
| **Fase 1** | |
| *Home*ke Perempatan | 0 - 15 |
| Perempatan ke Meja 1 | 16 - 33 |
| **Fase 2** | |
| Meja 1 ke Perempatan | 34 - 50 |
| Perempatan ke Meja 2 | 51 - 65 |
| **Fase 3** | |
| Meja 2 ke Perempatan | 66 - 83 |
| Perempatan ke *Home* | 84 - 100 |

| **Untuk pilihan meja 1/2 dan 3/4** | |
|---|---|
| **Keterangan** | **Persentase(%)** |
| **Fase 1** | |
| *Home*ke Perempatan | 0 - 15 |
| Perempatan ke Meja 1 | 16 - 33 |
| **Fase 2** | |
| Meja 1 ke Perempatan | 34 - 44 |
| Perempatan ke Pertigaan | 45 - 54 |
| Pertigaan ke Meja 2 | 55 - 65 |
| **Fase 3** | |
| Meja 2 ke Pertigaan | 66 - 78 |
| Pertigaan ke Perempatan | 79 - 89 |
| Perempatan ke *Home* | 90 - 100 |

| **Destinasi 2 Meja** | |
|---|---|
| **Untuk pilihan meja 3 dan/atau 4** | |
| **Keterangan** | **Persentase(%)** |
| **Fase 1** | |
| *Home*ke Perempatan | 0 - 13 |
| Perempatan ke Pertigaan | 14 - 23 |
| Pertigaan ke Meja 1 | 24 - 33 |
| **Fase 2** | |
| Meja 1 ke Pertigaan | 34 - 50 |
| Pertigaan ke Meja 2 | 51 - 65 |
| **Fase 3** | |
| Meja 2 ke Pertigaan | 66 - 83 |
| Pertigaan ke Perempatan | 66 - 83 |
| Perempatan ke *Home* | 84 - 100 |

| **Untuk pilihan meja 3/4 dan 1/2** | |
|---|---|
| **Keterangan** | **Persentase(%)** |
| **Fase 1** | |
| *Home*ke Perempatan | 0 - 13 |
| Perempatan ke Pertigaan | 14 - 23 |
| Pertigaan ke Meja 1 | 24 - 33 |

| **Untuk pilihan meja 3/4 dan 1/2** | |
|---|---|
| **Keterangan** | **Persentase(%)** |
| **Fase 2** | |
| Meja 1 ke Pertigaan | 34 - 44 |
| Pertigaan ke Perempatan | 45 - 54 |
| Perempatan ke Meja 2 | 55 - 65 |
| **Fase 3** | |
| Meja 2 ke Perempatan | 66 - 83 |
| Perempatan ke *Home* | 84 - 100 |

Tabel 7. Pengujian destinasi 2 meja

| **Kalkulasi 2 Meja** | | | | | | |
|---|---|---|---|---|---|---|
| **Tanpa Beban** | | | | | | |
| **Meja** | **1&2** | **1&3** | **1&4** | **2&3** | **2&4** | **3&4** |
| **Total Waktu (s)** | 56,76 | 70,93 | 55,59 | 60,80 | 68,97 | 68,44 |
| **Kecepatan (cm/s)** | 22,19 | 23,68 | 48,26 | 32,15 | 26,52 | 24,89 |
| **Keberhasilan (%)** | 92 | 91 | 90 | 79 | 88 | 96 |
| **Beban 1 Kg** | | | | | | |
| **Meja** | **1&2** | **2&1** | **2&3** | **3&2** | **3&4** | **4&3** |
| **Total Waktu (s)** | 60,23 | 62,44 | 80,17 | 81,08 | 73,88 | 75,15 |
| **Kecepatan (cm/s)** | 20,79 | 20,05 | 20,26 | 20,03 | 22,06 | 21,69 |
| **Keberhasilan (%)** | 100 | 100 | 100 | 100 | 100 | 100 |
| **Beban 2 Kg** | | | | | | |
| **Meja** | **1&2** | **2&1** | **2&3** | **3&2** | **3&4** | **4&3** |
| **Total Waktu (s)** | 63,17 | 65,38 | 83,11 | 84,02 | 76,82 | 78,09 |
| **Kecepatan (cm/s)** | 19,82 | 19,15 | 19,54 | 19,33 | 21,22 | 20,87 |
| **Keberhasilan (%)** | 100 | 100 | 100 | 100 | 100 | 100 |
| **Beban 3 Kg** | | | | | | |
| **Meja** | **1&2** | **2&1** | **2&3** | **3&2** | **3&4** | **4&3** |
| **Total Waktu (s)** | 70,46 | 72,67 | 90,40 | 91,31 | 84,11 | 85,38 |
| **Kecepatan (cm/s)** | 17,77 | 17,23 | 17,96 | 17,79 | 19,38 | 19,09 |
| **Keberhasilan (%)** | 100 | 100 | 100 | 100 | 100 | 100 |
| **Beban 4 Kg** | | | | | | |
| **Meja** | **1&2** | **2&1** | **2&3** | **3&2** | **3&4** | **4&3** |
| **Total Waktu (s)** | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| **Kecepatan (cm/s)** | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| **Keberhasilan (%)** | 0 | 0 | 0 | 0 | 0 | 0 |

Pada tabel 7 pengujian destinasi 2 meja juga dilakukan dengan mengujikan robot melakukan destinasi tanpa beban dan dengan beban. Pada pengujian destinasi 2 meja tanpa beban rata-rata keberhasilan pada tiap destinasi berbeda-beda, sedangkan untuk pengujian dengan beban rata-rata keberhasilan mencapai 100%. Secara keseluruhan *Mobile Robot Vision* ini sudah cukup untuk dimanfaatkan di restoran karena memiliki rata-rata keberhasilan untuk keseluruhan yakni sebesar 100%.

## KESIMPULAN

Berdasarkan hasil pengujian, dapat disimpulkan bahwa hasil dari proses pengujian kamera deteksi garis menunjukkan bahwa kamera berhasil membedakan deteksi garis warna hitam, warna kuning dan warna merah. Dengan menggunakan fitur dari OpenCV 2.1.0 dengan urutan tahapan yaitu menangkap citra RGB, konversi RGB to HSV, mencari nilai HSV

dengan skala, *Gaussian-blur*, *Canny Edge Detection*, dan *Hough Circle Transform*.

Nilai HSV yang didapatkan untuk warna hitam yakni H : 0-255, S : 0-255, V : 0-68. Warna kuning H : 20-57, S : 50-255, V : 80-195, dan warna merah H : 0-51, S : 181-255, V : 73-113.

Hasil dari proses pengujian destinasi 1 meja dan destinasi 2 meja menunjukkan bahwa semakin berat beban robot yang dibawa, rata-rata waktu tempuh yang dibutuhkan semakin banyak dan rata-rata kecepatan semakin berkurang hingga robot mencapai batas maksimalnya mengantarkan makanan yakni 3 Kg serta keberhasilan dengan adanya beban yang ada pada robot membuat berjalan stabil dan dapat menyelesaikan secara keseluruhan hingga 100%.

**Saran**

Pengembangan atau untuk penelitian selanjutnya adalah Untuk mekanik robot dapat diperbarui lebih efisiensi, agar pergerakan robot tidak menyeret dan menghindari selip. Penggunaan Mini PC di upgrade agar pengolahan citra bisa lebih cepat, tepat dan rinci. Penggunaan Dual-Kamera agar dapat memudahkan robot dalam membaca garis serta menghemat pergerakan robot, sehingga tidak berpacu pada satu kamera saja. Jika menggunakan 2 robot lebih, maka perlu pengembangan komunikasi antar robot itu sendiri. Penggunaan ban karet untuk mencegah ban selip ataupun menyeret ketika melakukan belokan atau putar balik. Menambah sistem kontrol atau sistem pengaturan pada robot sehingga robot dapat berjalan dengan lebih stabil. Untuk pengembangan selanjutnya penerapan robot dilakukan tanpa menggunakan garis sebagai jalur melainkan memahami letak meja yang dituju.

## DAFTAR PUSTAKA


Akbar, A. H. (2013). Penyelesaian Jalur Terpendek dengan menggunakan Maze Mapping Pada Line Maze. *Jurnalpa*.

Asif, M. (2015). Waiter Robot - Solution to Restaurant Automation. *MDSRC*.

Halim, S. (2007). *Merancang Mobile Robot Pembawa Objek Menggunakan OOPic-R*. Jakarta: PT. Elex Media Komputindo.

Nalwan, W. B. (2010). *Membuat Sendiri Robot Humanoid*. Jakarta: PT. Elex Media Komputindo.

Safii, C. A. (2017). Mobile Robot Pembawa Peralatan Makan Kotor Otomatis Pada Sebuah Restoran. *JCONES*, 160-167.

Wibisono, D. C. (2011). Metode Gaussian Smoothing Untuk Peningkatan Kualitas Citra Medis Yang Blur. *publikasi.dinus.ac.id*, 1-6.