



**EMBEDDED SYSTEM FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA**

GoVan (Go I.V.A.N)
Integrated Vehicle for Autonomous Navigation

GROUP PA-7

Adhikananda Wira Januar	2306267113
Daffa Sayra Firdaus	2306267151
Laura Fawzia Sambowo	2306260145
Muhammad Hilmi Al Muttaqi	2306267082

**PROGRAM STUDI TEKNIK KOMPUTER
UNIVERSITAS INDONESIA**

2025

PREFACE

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan laporan praktikum ini yang berjudul “GoVan ((Go I.V.A.N - *Integrated Vehicle for Autonomous Navigation*))”. Laporan ini disusun sebagai bagian dari tugas akhir praktikum Sistem Embedded di Fakultas Teknik Universitas Indonesia.

Laporan ini bertujuan untuk mendokumentasikan hasil dari perancangan dan implementasi sistem mobil mini otomatis (*GoVan*) yang mampu bergerak secara mandiri, menghindari rintangan, serta menampilkan status melalui berbagai indikator visual dan audio. Proyek ini merupakan bentuk penerapan konsep-konsep mikrokontroler dan sistem embedded dalam pengembangan solusi berbasis otomasi sederhana, yang potensial untuk diadaptasi dalam dunia nyata, seperti pengantaran paket, dokumen, atau logistik skala kecil di lingkungan kantor, hotel, atau restoran.

Ucapan terima kasih kami sampaikan kepada Bapak F. Astha Ekadiyanto selaku dosen pengampu, para asisten praktikum *Digital Laboratory* (Digilab) yang telah membimbing dengan sabar, serta seluruh rekan satu tim dan teman-teman yang telah memberikan dukungan dalam penyelesaian proyek ini.

Kami menyadari bahwa laporan ini masih memiliki kekurangan. Namun, kami telah berupaya sebaik mungkin agar laporan ini dapat menjadi dokumentasi yang bermanfaat, baik bagi kami sebagai mahasiswa maupun bagi para pembaca yang ingin memahami aplikasi mikrokontroler dalam sistem otomatisasi sederhana. Kritik dan saran yang membangun sangat kami harapkan demi penyempurnaan di masa mendatang.

Depok, 15 Mei, 2025

Group PA-7

TABLE OF CONTENTS

PREFACE.....	2
TABLE OF CONTENTS.....	3
LIST OF TABLES.....	4
LIST OF FIGURES.....	4
CHAPTER 1	
INTRODUCTION.....	5
1.1 PROBLEM STATEMENT.....	5
1.3 ACCEPTANCE CRITERIA.....	7
1.4 ROLES AND RESPONSIBILITIES.....	9
1.5 TIMELINE AND MILESTONES.....	10
CHAPTER 2	
IMPLEMENTATION.....	11
2.1 HARDWARE DESIGN AND SCHEMATIC.....	11
A. Hardware Design.....	11
B. Schematic.....	13
2.2 SOFTWARE DEVELOPMENT.....	13
A. Flowchart.....	15
B. Code (main.S).....	16
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	29
A. Code Integration (Hardware).....	30
CHAPTER 3	
TESTING AND EVALUATION.....	37
3.1 TESTING.....	37
3.2 RESULT.....	38
3.3 EVALUATION.....	41
CHAPTER 4	
CONCLUSION.....	42
REFERENCES.....	43
APPENDICES.....	44

LIST OF TABLES

Table 1.1 Roles and Responsibilities.....	9
Table 1.2 Timeline.....	10
Table 2.1 Components.....	12

LIST OF FIGURES

Fig 2.1 Flowchart GoVan.....	14
Fig. 3.1 Forward movement.....	37
Fig. 3.2 Left avoidance.....	38
Fig. 3.3 Emergency stop.....	38
Fig. 3.4 Sensor delay.....	39
Fig. 3.5 Buzzer.....	39

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Di era otomatisasi dan efisiensi kerja yang semakin dibutuhkan, pengantaran barang ringan seperti dokumen, makanan, atau paket internal menjadi aktivitas yang memerlukan solusi yang lebih cerdas dan efisien. Dalam lingkungan kerja seperti **kantor**, **restoran**, atau **hotel**, pengantaran manual oleh staf manusia masih menjadi metode utama, yang sering kali memakan waktu, mengganggu alur kerja, dan menambah beban operasional. Padahal, aktivitas ini bersifat repetitif dan dapat dengan mudah diotomatisasi menggunakan teknologi robotika sederhana.

Kebutuhan akan sistem pengantaran otomatis yang terjangkau, mudah diprogram, dan mampu bernavigasi secara mandiri semakin mendesak. Namun, hingga saat ini, belum tersedia banyak solusi edukatif maupun praktikal yang memanfaatkan teknologi mikrokontroler sederhana untuk menangani kebutuhan tersebut. Terlebih, tidak banyak pengembangan prototipe yang mengintegrasikan navigasi otonom berbasis sensor dengan logika pengambilan keputusan secara real-time menggunakan bahasa *Assembly*, yang dikenal menantang namun penting dalam dunia embedded system.

Adapun permasalahan yang diangkat dalam proyek ini adalah:

- Bagaimana merancang sebuah sistem pengantaran ringan otomatis yang mampu bernavigasi secara otonom menggunakan sensor sederhana untuk menghindari rintangan?
- Bagaimana membangun integrasi antara perangkat keras (motor, sensor, indikator) dan perangkat lunak yang dapat mengatur logika navigasi, penanganan rintangan, serta kondisi darurat?

- Bagaimana mengimplementasikan keseluruhan sistem menggunakan bahasa pemrograman Assembly dalam platform mikrokontroler sebagai bagian dari pengembangan sistem tertanam edukatif?

1.2 PROPOSED SOLUTION

Untuk menjawab permasalahan dalam proses pengantaran barang ringan di lingkungan kerja, proyek ini mengusulkan sebuah prototipe mobil mini otomatis yang dirancang untuk bergerak secara mandiri dan menghindari hambatan menggunakan sensor serta logika navigasi berbasis mikrokontroler. Proyek ini merepresentasikan langkah awal menuju sistem pengantaran cerdas yang dapat digunakan dalam skala kecil, seperti di kantor untuk mengantar dokumen antar divisi, atau di restoran dan hotel untuk mengirimkan makanan maupun barang ke pelanggan.

Mobil mini ini akan dilengkapi dengan empat sensor inframerah yang berfungsi sebagai pendeteksi hambatan dari empat arah berbeda (depan, kiri, kanan, dan belakang). Dengan memanfaatkan data dari sensor-sensor ini, sistem dapat menentukan arah gerak secara otomatis untuk menghindari rintangan di jalurnya. Pengambilan keputusan dilakukan melalui logika pemrograman berurutan (*polling*) berdasarkan prioritas arah, memastikan kendaraan dapat terus bergerak tanpa tabrakan.

Pengontrol utama sistem menggunakan mikrokontroler Arduino Uno, dengan program ditulis dalam bahasa *Assembly AVR*. Komponen dan bahasa pemrograman tersebut memungkinkan mahasiswa untuk memahami dan menerapkan konsep dasar pemrograman tingkat rendah, interupsi eksternal, kontrol motor dengan PWM, serta komunikasi antar komponen seperti LCD dan *buzzer*. Selain itu, sistem dilengkapi tombol darurat (*emergency stop*) yang dapat segera menghentikan seluruh fungsi

navigasi saat dibutuhkan, memberikan unsur keselamatan yang penting dalam proyek robotik.

Indikator visual (LCD) dan indikator suara (*buzzer*) juga digunakan untuk memberikan umpan balik secara real-time terhadap status sistem, seperti arah pergerakan, deteksi hambatan, atau aktivasi mode darurat. Dengan menggabungkan beberapa jenis perangkat input dan output secara terstruktur, proyek ini tidak hanya menawarkan solusi praktis terhadap masalah pengantaran, tetapi juga memberikan media pembelajaran interaktif bagi mahasiswa untuk menguasai dasar-dasar sistem tertanam (*embedded system*).

1.3 ACCEPTANCE CRITERIA

Kriteria keberhasilan proyek ini adalah sebagai berikut:

1. Mobil mini bergerak maju secara mandiri sebagai perilaku dasar yang membuatnya berkemampuan untuk melaju secara otomatis ke depan sebagai aksi default. Hal ini memastikan pengiriman barang atau dokumen dapat berlangsung tanpa hambatan saat kondisi jalur bebas, mengurangi kebutuhan intervensi manusia secara terus-menerus.
2. Navigasi cerdas dengan sensor *infrared* multi-arah untuk mendeteksi hambatan dengan bekal empat sensor inframerah yang dipasang di sisi depan, kanan, kiri, dan belakang, GoVan mampu mendeteksi keberadaan rintangan secara *real-time*. Dengan mekanisme pengecekan berurutan dari arah depan, kanan, kiri, hingga belakang, sistem dapat menentukan jalur terbaik untuk menghindari hambatan dan melanjutkan pergerakan dengan lancar. Ini memberikan fleksibilitas dan responsivitas dalam menghadapi lingkungan dinamis.
3. Pengendalian kecepatan motor dengan presisi menggunakan sinyal PWM yang dikendalikan oleh *timer*, kecepatan motor DC dapat disesuaikan dengan kebutuhan, memberikan kontrol yang halus dan

stabil pada pergerakan mobil. Kecepatan yang terjaga ini mengurangi risiko kecelakaan dan membuat navigasi menjadi lebih aman dan dapat diprediksi.

4. Keberadaan tombol *emergency stop* yang dihubungkan ke *interrupt* eksternal memungkinkan penghentian motor secara instan ketika diperlukan. Ketika tombol ini diaktifkan, *buzzer* dan LED akan memberikan sinyal peringatan, menjamin keamanan tidak hanya bagi perangkat, tapi juga bagi lingkungan sekitarnya.
5. Status operasi GoVan ditampilkan melalui layar LCD yang informatif, serta didukung dengan indikator *buzzer* dan lampu LED yang memberikan tanda suara dan cahaya secara *real-time*. Sistem ini memudahkan pengguna untuk memahami kondisi dan respons kendaraan tanpa perlu pemeriksaan manual.
6. Kolaborasi dua mikrokontroler melalui komunikasi I2C untuk memisahkan fungsi antara pengontrol utama yang mengatur navigasi dan pengontrol tambahan yang mengelola indikator membuat sistem ini modular dan efisien. Komunikasi menggunakan protokol I2C memastikan sinkronisasi data yang cepat dan akurat antara kedua mikrokontroler, mendukung operasi yang terkoordinasi dengan baik.
7. Monitoring kondisi kendaraan secara *real-time* melalui komunikasi serial dengan informasi status kendaraan juga dapat diteruskan melalui komunikasi serial ke perangkat eksternal yang akan memungkinkan pemantauan dan analisis data secara langsung.

1.4 ROLES AND RESPONSIBILITIES

Peran dan tanggung jawab yang diberikan kepada anggota kelompok adalah sebagai berikut:

Roles	Responsibilities	Person
Anggota	Laporan Akhir, PPT, <i>Software Development</i> (Set delay untuk I2C sesuai rumus), <i>Integration & Testing</i> , <i>Final Product & Testing</i>	Adhikananda Wira Januar (2306267113)
Ketua Kelompok	PPT, <i>Software Development</i> (Set Handler Secara keseluruhan, <i>Error Handling</i> , <i>Interrupt</i>), <i>Hardware Implementation</i> , <i>Integration & Testing</i> , <i>Final Product & Testing</i>	Daffa Sayra Firdaus (2306267151)
Anggota	Laporan Akhir, PPT, <i>Software Development</i> (Fungsionalitas Gerak Mobil secara umum), <i>Integration & Testing</i> , <i>Final Product & Testing</i>	Laura Fawzia Sambowo (2306260145)
Anggota	Laporan Akhir, PPT, <i>Software Development</i> (Pembuatan LCD (I2C)), <i>Integration & Testing</i> , <i>Final Product & Testing</i>	Muhammad Hilmi Al Muttaqi (2306267082)

Table 1.1 Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

a) Hardware Design completion

Desain hardware untuk sistem tertanam selesai, termasuk skematiknya.

b) Software Development

Pengembangan kode *assembly* yang dibuat pengguna (*software*) dimulai, dengan fokus pada tugas dan fungsionalitas tertentu.

c) Integration and Testing of Hardware and Software:

Penggabungan komponen *hardware* dan *software* dan diuji bersama untuk memastikan fungsionalitas yang tepat.

d) Final Product Assembly and Testing

Produk sistem akhir dirakit, diuji, dan diverifikasi memenuhi kriteria penerimaan.

TASK	Mei								
	10	11	12	13	14	15	16	17	18
Hardware Design Completion									
Software Development									
Integration & Testing									
Final Product & Testing									

Table 1.2 Timeline

CHAPTER 2

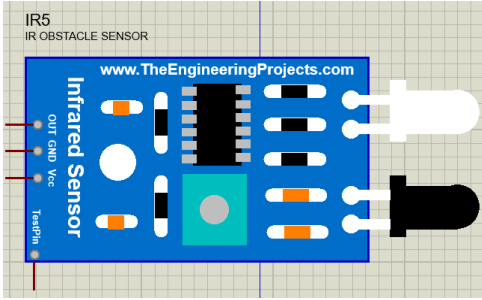
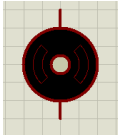
IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC

Proyek ini menggunakan beberapa komponen penting dalam perancangan perangkat kerasnya, yaitu:

- 4 *Infrared Sensor* untuk mendeteksi obstruksi dari empat arah,
- 2 *DC Motor* untuk menggerakkan roda,
- Monitor LCD sebagai *visual indicator*,
- *Button* untuk *emergency stop*,
- *Buzzer* sebagai *auditory indicator*,
- Arduino Uno sebagai otak dari sistem,
- L298N *Motor Driver* untuk mengendalikan motor.

A. Hardware Design

Components	Picture
Infrared sensor	
DC motor	

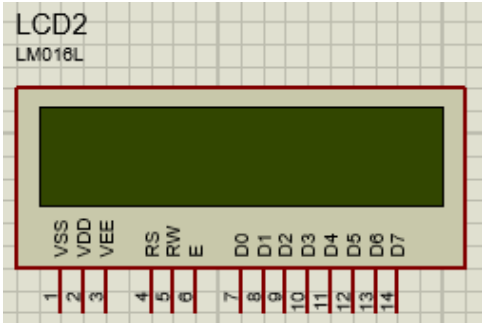


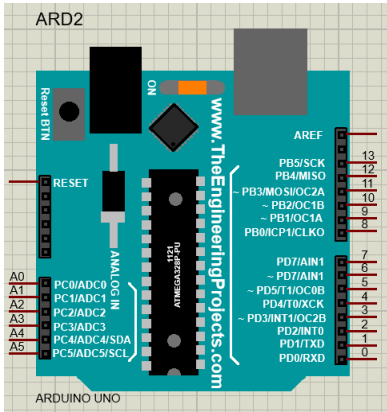
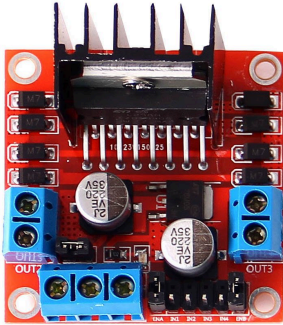
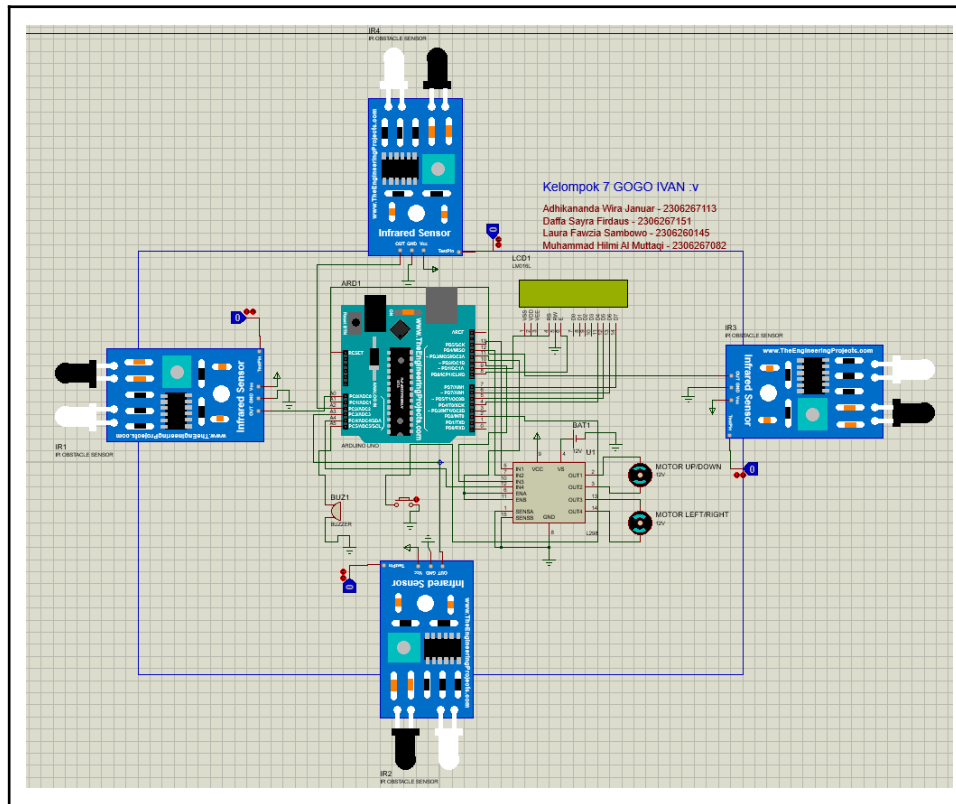
LCD monitor	
Button	
Buzzer	
Arduino Uno	
L298N Motor Driver	

Table 2.1 Components

B. Schematic



2.2 SOFTWARE DEVELOPMENT

Perangkat lunak untuk proyek ini mengimplementasikan sistem kendali otomatis pada mini mobil (GoVan) berbasis mikrokontroler (Arduino). Prototipe ini dirancang untuk bergerak maju secara *default*, menghindari rintangan dengan mengecek arah alternatif (kanan/kiri/belakang) menggunakan sensor IR. Logika pengecekan *obstacle* dilakukan secara berurutan (*polling*) dengan prioritas: depan → kiri → kanan → belakang.

Perangkat lunak ditulis dalam bahasa Assembly AVR, dengan menggunakan AVR Studio IDE untuk pengembangan. *Loop* program utama dirancang untuk menangani inisialisasi sistem, pembacaan sensor, pengambilan keputusan arah, serta penanganan kondisi darurat.

Komponen-komponen kunci dari perangkat lunak ini mencakup inisialisasi tampilan LCD untuk menampilkan status arah dan sensor,

pembacaan input digital dari sensor IR (inframerah) untuk mendeteksi hambatan di sekitar mobil, serta penggunaan interupsi eksternal untuk mengaktifkan mode berhenti darurat saat tombol ditekan.

Selain itu, perangkat lunak mengelola logika navigasi berdasarkan data sensor, mengontrol motor untuk bergerak maju, mundur, berbelok, atau berhenti, serta mengaktifkan *buzzer* sebagai peringatan saat terdapat hambatan. Sistem juga dapat beralih arah secara otomatis apabila terdeteksi hambatan di jalur saat ini.

Secara keseluruhan, perangkat lunak ini difokuskan untuk menciptakan sistem navigasi otonom yang responsif dan aman, dengan memanfaatkan fitur-fitur mikrokontroler seperti polling, interrupt, PWM, dan komunikasi I2C. Prototipe ini juga mengintegrasikan multi device (*sensor, actuator, indicator*) secara terstruktur untuk memastikan *reliabilitas* pergerakan.

A. Flowchart

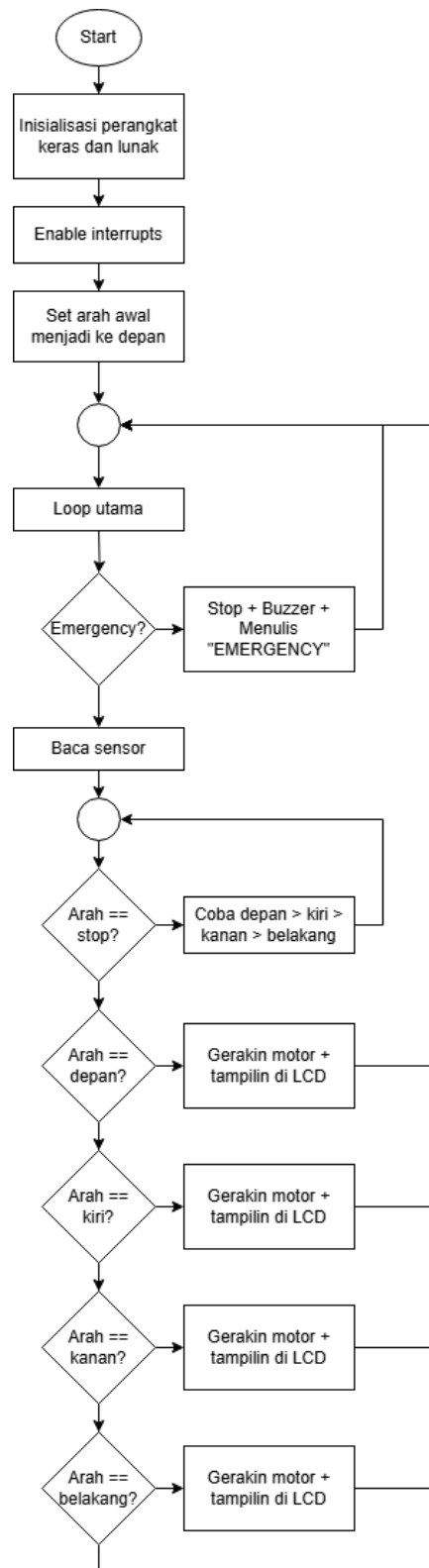


Fig 2.1 Flowchart GoVan

B. Code (main.S)

```
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;-----
; Fungsi-fungsi global
.global LCD_write
.global init_car
.global run_car

; Vektor interupsi
.global INT0_vect
;=====
LCD_write:
    ; Atur hanya pin nibble tinggi PORTD sebagai output untuk data LCD
    ; Jaga nibble rendah (termasuk PD2) tersedia untuk penggunaan lain
    IN    R16, DDRD      ; Baca nilai DDRD saat ini
    ORI   R16, 0xF0      ; Atur hanya nibble tinggi (PD4-PD7) sebagai
output
    OUT   DDRD, R16      ; Atur port D sebagai output untuk data

    ; Atur semua pin PORTB sebagai output untuk kontrol LCD
    LDI   R16, 0xFF
    OUT   DDRB, R16      ; Atur port B sebagai output untuk perintah

    ; Inisialisasi pin kontrol LCD
    CBI   PORTB, 0        ; EN = 0
    RCALL delay_ms        ; Tunggu LCD menyala

    ; Inisialisasi LCD
    RCALL LCD_init

    ; Kembali ke pemanggil - kontrol mobil akan menangani tampilan
    RET
;=====
LCD_init:
    LDI   R16, 0x33        ;inisialisasi LCD untuk data 4-bit
    RCALL command_wrt      ;kirim ke register perintah
    RCALL delay_ms
    LDI   R16, 0x32        ;inisialisasi LCD untuk data 4-bit
    RCALL command_wrt
    RCALL delay_ms
    LDI   R16, 0x28        ;LCD 2 baris, matriks 5x7
    RCALL command_wrt
    RCALL delay_ms
    LDI   R16, 0x0C        ;tampilan ON, kursor OFF
    RCALL command_wrt
    LDI   R16, 0x01        ;bersihkan LCD
    RCALL command_wrt
    RCALL delay_ms
    LDI   R16, 0x06        ;geser kursor ke kanan
    RCALL command_wrt
    RET
;=====
command_wrt:
    MOV   R27, R16
    ANDI  R27, 0xF0        ;masking nibble rendah & simpan nibble tinggi

    ; Pertahankan nibble rendah dari PORTD
    IN    R17, PORTD      ;baca nilai PORTD saat ini
    ANDI  R17, 0x0F        ;simpan hanya nibble rendah
    OR    R27, R17        ;gabungkan dengan nibble tinggi dari perintah
    OUT   PORTD, R27      ;output ke port D dengan mempertahankan nibble
rendah

    CBI   PORTB, 1        ;RS = 0 untuk perintah
    SBI   PORTB, 0        ;EN = 1
    RCALL delay_short     ;perlebar pulsa EN
    CBI   PORTB, 0        ;EN = 0 untuk pulsa H-ke-L
```



```

RCALL delay_us          ;tunda dalam mikrodetik
;-----
MOV R27, R16
SWAP R27                ;tukar nibble
ANDI R27, 0xF0          ;masking nibble rendah & simpan nibble tinggi

; Pertahankan nibble rendah dari PORTD
IN R17, PORTD           ;baca nilai PORTD saat ini
ANDI R17, 0x0F          ;simpan hanya nibble rendah
OR R27, R17             ;gabungkan dengan nibble tinggi dari perintah
OUT PORTD, R27          ;output ke port D dengan mempertahankan nibble
rendah

SBI PORTB, 0            ;EN = 1
RCALL delay_short       ;perlebar pulsa EN
CBI PORTB, 0            ;EN = 0 untuk pulsa H-ke-L
RCALL delay_us          ;tunda dalam mikrodetik
RET

;=====
data_wrt:
MOV R27, R16
ANDI R27, 0xF0          ;masking nibble rendah & simpan nibble tinggi

; Pertahankan nibble rendah dari PORTD
IN R17, PORTD           ;baca nilai PORTD saat ini
ANDI R17, 0x0F          ;simpan hanya nibble rendah
OR R27, R17             ;gabungkan dengan nibble tinggi dari data
OUT PORTD, R27          ;output ke port D dengan mempertahankan nibble
rendah

SBI PORTB, 1            ;RS = 1 untuk data
SBI PORTB, 0            ;EN = 1
RCALL delay_short       ;perlebar pulsa EN
CBI PORTB, 0            ;EN = 0 untuk pulsa H-ke-L
RCALL delay_us          ;tunda dalam mikrodetik
;-----
MOV R27, R16
SWAP R27                ;tukar nibble
ANDI R27, 0xF0          ;masking nibble rendah & simpan nibble tinggi

; Pertahankan nibble rendah dari PORTD
IN R17, PORTD           ;baca nilai PORTD saat ini
ANDI R17, 0x0F          ;simpan hanya nibble rendah
OR R27, R17             ;gabungkan dengan nibble tinggi dari data
OUT PORTD, R27          ;output ke port D dengan mempertahankan nibble
rendah

SBI PORTB, 0            ;EN = 1
RCALL delay_short       ;perlebar pulsa EN
CBI PORTB, 0            ;EN = 0 untuk pulsa H-ke-L
RCALL delay_us          ;tunda dalam mikrodetik
RET

;=====
delay_short:
NOP
NOP
RET

;-----
delay_us:
LDI R20, 90
13: RCALL delay_short
DEC R20
BRNE 13
RET

;-----
delay_ms:
LDI R21, 40
14: RCALL delay_us
DEC R21
BRNE 14

```

```

        RET
;=====
delay_seconds:      ;subrutin loop bersarang (tunda maksimal 3.11 detik)
        LDI    R20, 255    ;penghitung loop luar
15: LDI    R21, 255    ;penghitung loop tengah
16: LDI    R22, 20      ;penghitung loop dalam untuk memberikan tunda 0.25 detik
17: DEC    R22          ;kurangi loop dalam
        BRNE 17          ;loop jika tidak nol
        DEC    R21          ;kurangi loop tengah
        BRNE 16          ;loop jika tidak nol
        DEC    R20          ;kurangi loop luar
        BRNE 15          ;loop jika tidak nol
        RET            ;kembali ke pemanggil
;-----

;=====
; Fungsi-fungsi Kontrol Mobil
;=====

; Variabel untuk status mobil
.section .data
        car_direction: .byte 0 ; 0=berhenti, 1=maju, 2=mundur, 3=kiri, 4=kanan
        front_status: .byte 0 ; 0=aman, 1=bahaya
        back_status: .byte 0
        left_status: .byte 0
        right_status: .byte 0
        emergency_stop: .byte 0 ; 0=operasi normal, 1=berhenti darurat
        prev_direction: .byte 0 ; Simpan arah sebelumnya sebelum berhenti darurat

.section .text

;=====
; Inisialisasi komponen mobil (motor, sensor, buzzer)
;=====
init_car:
        ; Simpan register yang digunakan
        PUSH    R16
        PUSH    R17

        ; Inisialisasi LCD terlebih dahulu - ini akan mengatur PORTD dan PORTB
        untuk LCD
        RCALL LCD_write

        ; Sekarang inisialisasi PORTC untuk sensor IR dan buzzer
        LDI    R16, 0x20          ; Atur PC5 sebagai output untuk buzzer, PC0-PC3
        sebagai input
        OUT    DDRC, R16

        ; Aktifkan resistor pull-up untuk input sensor
        LDI    R16, 0x0F          ; Atur pull-up PC0-PC3
        OUT    PORTC, R16

        ; Konfigurasi INT0 (PD2) untuk tombol darurat
        ; Atur PD2 sebagai input dengan pull-up
        CBI    DDRD, 2            ; Atur PD2 sebagai input
        SBI    PORTD, 2          ; Aktifkan pull-up pada PD2

        ; Konfigurasi INT0 untuk dipicu pada falling edge (penekanan tombol)
        LDI    R16, (1<<ISC01)   ; Falling edge dari INT0 menghasilkan interupsi
        STS    EICRA, R16

        ; Aktifkan interupsi INT0
        LDI    R16, (1<<INT0)
        OUT    EIMSK, R16

        ; Inisialisasi emergency_stop ke 0 (operasi normal)
        LDI    R16, 0
        STS    emergency_stop, R16

        ; Aktifkan interupsi global

```

```

SEI

; Re-inisialisasi PORTB untuk kontrol motor sambil mempertahankan pin LCD
; PB0 (E) dan PB1 (RS) digunakan oleh LCD
; PB5 (IN1), PB4 (IN2), PB3 (ENA) untuk motor maju/mundur
IN    R16, DDRB          ; Baca nilai DDRB saat ini
ORI   R16, 0x38          ; Atur PB3, PB4, PB5 sebagai output sambil
mempertahankan bit lainnya
OUT   DDRB, R16

; Konfigurasi PB2 (IN3) untuk kontrol motor kiri/kanan
; PB2 (IN3), PC4/A4 (IN4) untuk motor kiri/kanan
; Bit yang lebih tinggi dari PORTD digunakan untuk data LCD
IN    R16, DDRB          ; Baca nilai DDRB saat ini
ORI   R16, 0x04          ; Atur PB2 sebagai output sambil mempertahankan bit
lainnya
OUT   DDRB, R16

; Konfigurasi PC4 (A4/IN4) untuk kontrol motor kiri/kanan
IN    R16, DDRC          ; Baca nilai DDRC saat ini
ORI   R16, 0x10          ; Atur PC4 sebagai output sambil mempertahankan bit
lainnya
OUT   DDRC, R16

; Inisialisasi Timer2 untuk PWM pada PB3 (OC2A)
LDI   R16, (1<<WGM20)|(1<<WGM21)|(1<<COM2A1) ; Mode Fast PWM, mode
non-inverting
STS   TCCR2A, R16
LDI   R16, (1<<CS21)      ; Prescaler = 8
STS   TCCR2B, R16
LDI   R16, 200            ; Atur siklus tugas PWM (0-255)
STS   OCR2A, R16

; Tampilkan pesan awal
RCALL display_status

; Kembalikan register
POP   R17
POP   R16
RET

;=====
; Fungsi operasi utama mobil
;=====
; Handler interupsi INT0 untuk tombol darurat
INT0_vect:
; Simpan register yang digunakan
PUSH  R16

; Toggle status emergency_stop
LDS   R16, emergency_stop
CPI   R16, 0
BRNE  emergency_to_normal

; Normal -> Darurat: Simpan arah saat ini dan berhenti
LDS   R16, car_direction
STS   prev_direction, R16
LDI   R16, 1
STS   emergency_stop, R16
RCALL motor_stop
RCALL buzzer_on
RJMP  int0_exit

emergency_to_normal:
; Darurat -> Normal: Kembalikan arah sebelumnya
LDI   R16, 0
STS   emergency_stop, R16
RCALL buzzer_off

; Kembalikan arah sebelumnya dan mulai bergerak lagi

```

```

    LDS    R16, prev_direction
    STS    car_direction, R16

    ; Periksa arah mana yang akan dikembalikan dan mulai motor sesuai
    CPI    R16, 0          ; Jika arah sebelumnya adalah berhenti, tetap
berhenti
    BREQ   restore_done

    CPI    R16, 1          ; Maju?
    BRNE   restore_check_backward
    RCALL   motor_forward
    RJMP   restore_done

restore_check_backward:
    CPI    R16, 2          ; Mundur?
    BRNE   restore_check_left
    RCALL   motor_backward
    RJMP   restore_done

restore_check_left:
    CPI    R16, 3          ; Kiri?
    BRNE   restore_check_right
    RCALL   motor_left
    RJMP   restore_done

restore_check_right:
    CPI    R16, 4          ; Kanan?
    BRNE   restore_done
    RCALL   motor_right

restore_done:
    RJMP   int0_exit

int0_exit:
    ; Kembalikan register
    POP    R16
    RETI

run_car:
    ; Simpan register yang digunakan
    PUSH   R16
    PUSH   R17

    ; Atur arah awal ke maju
    LDI    R16, 1
    STS    car_direction, R16
    RCALL   motor_forward

car_loop:
    ; Periksa apakah dalam mode berhenti darurat
    LDS    R16, emergency_stop
    CPI    R16, 1
    BRNE   normal_operation
    JMP     emergency_display

normal_operation:

    ; Operasi normal - baca sensor dan lanjutkan
    ; Baca semua sensor IR
    RCALL   read_sensors

    ; Periksa arah saat ini dan sensor yang sesuai
    LDS    R16, car_direction

    CPI    R16, 0          ; Berhenti?
    BRNE   not_stopped    ; Jika tidak berhenti, lanjutkan pemeriksaan

    ; Ketika berhenti, periksa semua arah untuk melihat apakah kita dapat
bergerak lagi
    RCALL   check_front    ; Coba bergerak maju jika memungkinkan

```

```

    RCALL check_left      ; Coba belok kiri jika memungkinkan
    RCALL check_right     ; Coba belok kanan jika memungkinkan
    RCALL check_back      ; Coba bergerak mundur jika memungkinkan
    RJMP do_update_display ; Jika semua arah memiliki hambatan, hanya perbarui
tampilan

not_stopped:
    CPI R16, 1           ; Maju?
    BRNE not_forward
    LDS R16, front_status
    CPI R16, 1           ; Hambatan?
    BRNE buzzer_check_done ; Tidak ada hambatan, lanjutkan

    ; Penanganan hambatan di depan
    RCALL handle_forward_obstacle
    RJMP do_update_display

not_forward:
    CPI R16, 2           ; Mundur?
    BRNE not_backward
    LDS R16, back_status
    CPI R16, 1           ; Hambatan?
    BRNE buzzer_check_done ; Tidak ada hambatan, lanjutkan

    ; Penanganan hambatan di belakang
    RCALL handle_backward_obstacle
    RJMP do_update_display

not_backward:
    CPI R16, 3           ; Kiri?
    BRNE not_left
    LDS R16, left_status
    CPI R16, 1           ; Hambatan?
    BRNE buzzer_check_done ; Tidak ada hambatan, lanjutkan

    ; Penanganan hambatan di kiri
    RCALL handle_left_obstacle
    RJMP do_update_display

not_left:
    CPI R16, 4           ; Kanan?
    BRNE buzzer_check_done
    LDS R16, right_status
    CPI R16, 1           ; Hambatan?
    BRNE buzzer_check_done ; Tidak ada hambatan, lanjutkan

    ; Penanganan hambatan di kanan
    RCALL handle_right_obstacle
    RJMP do_update_display

buzzer_check_done:
    ; Lanjutkan untuk memperbarui tampilan
    RJMP do_update_display

; Fungsi penanganan untuk hambatan di berbagai arah
handle_forward_obstacle:
    RCALL buzzer_on      ; Nyalakan buzzer untuk hambatan di depan
    RCALL check_left     ; Coba temukan arah alternatif
    RCALL check_right
    RCALL check_back
    RCALL buzzer_off     ; Matikan buzzer
    ; Jika kita sampai di sini, semua arah memiliki hambatan
    RCALL motor_stop
    LDI R16, 0
    STS car_direction, R16
    RET

handle_backward_obstacle:
    RCALL buzzer_on      ; Nyalakan buzzer untuk hambatan di belakang
    RCALL check_left     ; Coba temukan arah alternatif

```

```

RCALL check_right
RCALL check_front
RCALL buzzer_off          ; Matikan buzzer
; Jika kita sampai di sini, semua arah memiliki hambatan
RCALL motor_stop
LDI R16, 0
STS car_direction, R16
RET

handle_left_obstacle:
RCALL buzzer_on          ; Nyalakan buzzer untuk hambatan di kiri
RCALL check_right        ; Coba temukan arah alternatif
RCALL check_front
RCALL check_back
RCALL buzzer_off          ; Matikan buzzer
; Jika kita sampai di sini, semua arah memiliki hambatan
RCALL motor_stop
LDI R16, 0
STS car_direction, R16
RET

handle_right_obstacle:
RCALL buzzer_on          ; Nyalakan buzzer untuk hambatan di kanan
RCALL check_left         ; Coba temukan arah alternatif
RCALL check_front
RCALL check_back
RCALL buzzer_off          ; Matikan buzzer
; Jika kita sampai di sini, semua arah memiliki hambatan
RCALL motor_stop
LDI R16, 0
STS car_direction, R16
RET

emergency_display:
; Dalam mode berhenti darurat, hanya perbarui tampilan dan lanjutkan loop
RCALL display_status

; Tampilkan pesan "EMERGENCY" atau indikator khusus
; Ini ditangani dalam fungsi display_status

; Tunda kecil
LDI R17, 2
emergency_delay_loop:
RCALL delay_ms
DEC R17
BRNE emergency_delay_loop

; Lanjutkan loop utama
RJMP car_loop

do_update_display:
; Perbarui LCD dengan status saat ini
RCALL display_status

; Tunda kecil
LDI R17, 2
delay_loop:
RCALL delay_ms
DEC R17
BRNE delay_loop

; Lanjutkan loop utama
RJMP car_loop

run_car_exit:
; Kembalikan register dan kembali (hanya dicapai jika kita secara eksplisit
melompat ke sini)
POP R17
POP R16
RET

```

```

;=====
; Periksa arah untuk hambatan dan ubah arah jika diperlukan
;=====
check_left:
    LDS    R16, left_status
    CPI    R16, 1          ; Hambatan di kiri?
    BREQ   check_left_exit ; Ya, keluar

    ; Tidak ada hambatan di kiri, belok kiri
    LDI    R16, 3
    STS    car_direction, R16
    RCALL  motor_left
    RCALL  display_status

    ; Tunda kecil untuk memungkinkan belok
    LDI    R17, 10
left_delay:
    RCALL  delay_ms
    DEC    R17
    BRNE   left_delay

    ; Kembali ke loop utama
    RJMP   car_loop

check_left_exit:
    RET

check_right:
    LDS    R16, right_status
    CPI    R16, 1          ; Hambatan di kanan?
    BREQ   check_right_exit ; Ya, keluar

    ; Tidak ada hambatan di kanan, belok kanan
    LDI    R16, 4
    STS    car_direction, R16
    RCALL  motor_right
    RCALL  display_status

    ; Tunda kecil untuk memungkinkan belok
    LDI    R17, 10
right_delay:
    RCALL  delay_ms
    DEC    R17
    BRNE   right_delay

    ; Kembali ke loop utama
    RJMP   car_loop

check_right_exit:
    RET

check_back:
    LDS    R16, back_status
    CPI    R16, 1          ; Hambatan di belakang?
    BREQ   check_back_exit ; Ya, keluar

    ; Tidak ada hambatan di belakang, mundur
    LDI    R16, 2
    STS    car_direction, R16
    RCALL  motor_backward
    RCALL  display_status

    ; Tunda kecil untuk memungkinkan pergerakan
    LDI    R17, 10
back_delay:
    RCALL  delay_ms
    DEC    R17
    BRNE   back_delay

```

```

; Kembali ke loop utama
RJMP car_loop

check_back_exit:
RET

check_front:
LDS R16, front_status
CPI R16, 1 ; Hambatan di depan?
BREQ check_front_exit ; Ya, keluar

; Tidak ada hambatan di depan, maju
LDI R16, 1
STS car_direction, R16
RCALL motor_forward
RCALL display_status

; Tunda kecil untuk memungkinkan pergerakan
LDI R17, 10
front_delay:
RCALL delay_ms
DEC R17
BRNE front_delay

; Kembali ke loop utama
RJMP car_loop

check_front_exit:
RET

;=====
; Fungsi kontrol motor
;=====
motor_forward:
; Atur IN1=HIGH, IN2=LOW untuk maju
SBI PORTB, 5 ; IN1 = HIGH
CBI PORTB, 4 ; IN2 = LOW

; Hentikan motor kiri/kanan saat bergerak maju
CBI PORTB, 2 ; IN3 = LOW (PB2)
CBI PORTC, 4 ; IN4 = LOW (PC4/A4)
RET

motor_backward:
; Atur IN1=LOW, IN2=HIGH untuk mundur
CBI PORTB, 5 ; IN1 = LOW
SBI PORTB, 4 ; IN2 = HIGH

; Hentikan motor kiri/kanan saat bergerak mundur
CBI PORTB, 2 ; IN3 = LOW (PB2)
CBI PORTC, 4 ; IN4 = LOW (PC4/A4)
RET

motor_left:
; Hentikan motor maju/mundur saat berbelok ke kiri
CBI PORTB, 5 ; IN1 = LOW
CBI PORTB, 4 ; IN2 = LOW

; Atur motor kiri/kanan untuk berbelok ke kiri
SBI PORTB, 2 ; IN3 = HIGH (PB2)
CBI PORTC, 4 ; IN4 = LOW (PC4/A4)
RET

motor_right:
; Hentikan motor maju/mundur saat berbelok ke kanan
CBI PORTB, 5 ; IN1 = LOW
CBI PORTB, 4 ; IN2 = LOW

; Atur motor kiri/kanan untuk berbelok ke kanan
CBI PORTB, 2 ; IN3 = LOW (PB2)

```



```

SBI   PORTC, 4           ; IN4 = HIGH (PC4/A4)
RET

motor_stop:
; Hentikan motor maju/mundur
CBI   PORTB, 5           ; IN1 = LOW
CBI   PORTB, 4           ; IN2 = LOW

; Hentikan motor kiri/kanan
CBI   PORTB, 2           ; IN3 = LOW (PB2)
CBI   PORTC, 4           ; IN4 = LOW (PC4/A4)
RET

;=====
; Fungsi pembacaan sensor
;=====
read_sensors:
; Baca sensor depan (PC2)
IN     R16, PINC
ANDI   R16, 0x04         ; Masking untuk PC2
CPI     R16, 0           ; Jika 0, hambatan terdeteksi (active low)
BREQ   front_obstacle
LDI     R16, 0           ; Tidak ada hambatan
RJMP   store_front
front_obstacle:
LDI     R16, 1           ; Hambatan terdeteksi
store_front:
STS     front_status, R16

; Baca sensor belakang (PC3)
IN     R16, PINC
ANDI   R16, 0x08         ; Masking untuk PC3
CPI     R16, 0           ; Jika 0, hambatan terdeteksi (active low)
BREQ   back_obstacle
LDI     R16, 0           ; Tidak ada hambatan
RJMP   store_back
back_obstacle:
LDI     R16, 1           ; Hambatan terdeteksi
store_back:
STS     back_status, R16

; Baca sensor kiri (PC0)
IN     R16, PINC
ANDI   R16, 0x01         ; Masking untuk PC0
CPI     R16, 0           ; Jika 0, hambatan terdeteksi (active low)
BREQ   left_obstacle
LDI     R16, 0           ; Tidak ada hambatan
RJMP   store_left
left_obstacle:
LDI     R16, 1           ; Hambatan terdeteksi
store_left:
STS     left_status, R16

; Baca sensor kanan (PC1)
IN     R16, PINC
ANDI   R16, 0x02         ; Masking untuk PC1
CPI     R16, 0           ; Jika 0, hambatan terdeteksi (active low)
BREQ   right_obstacle
LDI     R16, 0           ; Tidak ada hambatan
RJMP   store_right
right_obstacle:
LDI     R16, 1           ; Hambatan terdeteksi
store_right:
STS     right_status, R16

RET

;=====
; Fungsi kontrol buzzer
;=====

```

```

buzzer_on:
    SBI    PORTC, 5          ; Nyalakan buzzer (PC5)
    RET

buzzer_off:
    CBI    PORTC, 5          ; Matikan buzzer (PC5)
    RET

;=====
; Fungsi tampilan
;=====
display_status:
    ; Bersihkan LCD
    LDI    R16, 0x01
    RCALL  command_wrt
    RCALL  delay_ms

    ; Periksa apakah dalam mode berhenti darurat
    LDS    R16, emergency_stop
    CPI    R16, 1
    BRNE   disp_normal_mode
    JMP    disp_emergency

disp_normal_mode:
    ; Tampilkan arah pada baris pertama
    LDS    R16, car_direction

    ; Gunakan serangkaian perbandingan dan lompatan alih-alih cabang
    CPI    R16, 0
    BRNE   disp_check_forward
    JMP    disp_stop

disp_check_forward:
    CPI    R16, 1
    BRNE   disp_check_backward
    JMP    disp_forward

disp_check_backward:
    CPI    R16, 2
    BRNE   disp_check_left
    JMP    disp_backward

disp_check_left:
    CPI    R16, 3
    BRNE   disp_check_right
    JMP    disp_left

disp_check_right:
    CPI    R16, 4
    BRNE   disp_not_right
    JMP    disp_right

disp_not_right:
    JMP    disp_unknown

disp_emergency:
    ; Tampilkan "EMERGENCY" pada baris pertama
    LDI    R16, 'E'
    RCALL  data_wrt
    LDI    R16, 'M'
    RCALL  data_wrt
    LDI    R16, 'E'
    RCALL  data_wrt
    LDI    R16, 'R'
    RCALL  data_wrt
    LDI    R16, 'G'
    RCALL  data_wrt
    LDI    R16, 'E'
    RCALL  data_wrt
    LDI    R16, 'N'

```

```

        RCALL data_wrt
        LDI  R16, 'C'
        RCALL data_wrt
        LDI  R16, 'Y'
        RCALL data_wrt
        RJMP disp_sensors

disp_stop:
        LDI  R16, 'S'
        RCALL data_wrt
        LDI  R16, 'T'
        RCALL data_wrt
        LDI  R16, 'O'
        RCALL data_wrt
        LDI  R16, 'P'
        RCALL data_wrt
        RJMP disp_sensors

disp_forward:
        LDI  R16, 'D'
        RCALL data_wrt
        LDI  R16, 'E'
        RCALL data_wrt
        LDI  R16, 'P'
        RCALL data_wrt
        LDI  R16, 'A'
        RCALL data_wrt
        LDI  R16, 'N'
        RCALL data_wrt
        RJMP disp_sensors

disp_backward:
        LDI  R16, 'B'
        RCALL data_wrt
        LDI  R16, 'E'
        RCALL data_wrt
        LDI  R16, 'L'
        RCALL data_wrt
        LDI  R16, 'A'
        RCALL data_wrt
        LDI  R16, 'K'
        RCALL data_wrt
        LDI  R16, 'A'
        RCALL data_wrt
        LDI  R16, 'N'
        RCALL data_wrt
        LDI  R16, 'G'
        RCALL data_wrt
        RJMP disp_sensors

disp_left:
        LDI  R16, 'K'
        RCALL data_wrt
        LDI  R16, 'I'
        RCALL data_wrt
        LDI  R16, 'R'
        RCALL data_wrt
        LDI  R16, 'I'
        RCALL data_wrt
        RJMP disp_sensors

disp_right:
        LDI  R16, 'K'
        RCALL data_wrt
        LDI  R16, 'A'
        RCALL data_wrt
        LDI  R16, 'N'
        RCALL data_wrt
        LDI  R16, 'A'
        RCALL data_wrt

```

```

LDI R16, 'N'
RCALL data_wrt
RJMP disp_sensors

disp_unknown:
LDI R16, '?'
RCALL data_wrt

disp_sensors:
; Pindah ke baris kedua
LDI R16, 0xC0
RCALL command_wrt
RCALL delay_ms

; Tampilkan sensor depan (D:)
LDI R16, 'D'
RCALL data_wrt
LDI R16, ':'
RCALL data_wrt
LDS R16, front_status
CPI R16, 0 ; Periksa apakah 0 atau 1
BREQ front_zero
LDI R16, '0' ; Muat ASCII '0' (tidak ada hambatan untuk active low)
RJMP front_display

front_zero:
LDI R16, '1' ; Muat ASCII '1' (hambatan untuk active low)
front_display:
RCALL data_wrt

; Tampilkan spasi
LDI R16, ' '
RCALL data_wrt

; Tampilkan sensor belakang (B:)
LDI R16, 'B'
RCALL data_wrt
LDI R16, ':'
RCALL data_wrt
LDS R16, back_status
CPI R16, 0 ; Periksa apakah 0 atau 1
BREQ back_zero
LDI R16, '0' ; Muat ASCII '0' (tidak ada hambatan untuk active low)
RJMP back_display

back_zero:
LDI R16, '1' ; Muat ASCII '1' (hambatan untuk active low)
back_display:
RCALL data_wrt

; Tampilkan spasi
LDI R16, ' '
RCALL data_wrt

; Tampilkan sensor kiri (K:)
LDI R16, 'K'
RCALL data_wrt
LDI R16, ':'
RCALL data_wrt
LDS R16, left_status
CPI R16, 0 ; Periksa apakah 0 atau 1
BREQ left_zero
LDI R16, '0' ; Muat ASCII '0' (tidak ada hambatan untuk active low)
RJMP left_display

left_zero:
LDI R16, '1' ; Muat ASCII '1' (hambatan untuk active low)
left_display:
RCALL data_wrt

; Tampilkan spasi
LDI R16, ' '
RCALL data_wrt

```

```

; Tampilkan sensor kanan (N:)
LDI R16, 'N'
RCALL data_wrt
LDI R16, ':'
RCALL data_wrt
LDS R16, right_status
CPI R16, 0 ; Periksa apakah 0 atau 1
BREQ right_zero
LDI R16, '0' ; Muat ASCII '0' (tidak ada hambatan untuk active low)
RJMP right_display
right_zero:
LDI R16, '1' ; Muat ASCII '1' (hambatan untuk active low)
right_display:
RCALL data_wrt

RET

```

2.3 HARDWARE AND SOFTWARE INTEGRATION

Dalam kode yang kami buat, integrasi antara perangkat keras dan perangkat lunak digunakan untuk mengendalikan mobil mini otomatis secara mandiri. Perangkat keras yang digunakan meliputi sensor IR, motor DC, *buzzer*, *button* darurat, dan LCD. Sedangkan perangkat lunak berupa program dalam bahasa Assembly AVR yang mengatur logika navigasi dan respon sistem terhadap kondisi lingkungan.

Berikut integrasi yang kami gunakan pada proyek kali ini:

- **Inisialisasi**

Dilakukan di awal program untuk mengatur konfigurasi awal dari seluruh perangkat keras, seperti pengaturan port input/output, konfigurasi interupsi eksternal untuk tombol darurat, inisialisasi LCD, dan *timer* untuk PWM motor.

- **Loop utama**

Bagian inti program yang terus berjalan untuk membaca data dari sensor IR, mengevaluasi kondisi di sekitar mobil (depan, kiri, kanan, belakang), serta menentukan arah gerak berdasarkan logika penghindaran hambatan.

- **Pengambilan keputusan**

Program akan memutuskan arah gerak secara otomatis dengan

mempertimbangkan data dari sensor IR. Jika hambatan terdeteksi, mobil akan mencoba berbelok atau mundur, dan bila semua arah terhalang, mobil akan berhenti.

- **Mode darurat**

Ketika tombol darurat ditekan, interupsi akan menghentikan mobil dan menyalakan *buzzer* sebagai peringatan. Saat kondisi kembali normal, mobil melanjutkan pergerakan sesuai arah sebelumnya.

A. Code Integration (Hardware)

i. Inisialisasi LCD

```
LCD_write:
    ; Atur hanya pin nibble tinggi PORTD sebagai output untuk data LCD
    ; Jaga nibble rendah (termasuk PD2) tersedia untuk penggunaan lain
    IN    R16, DDRD          ; Baca nilai DDRD saat ini
    ORI   R16, 0xF0          ; Atur hanya nibble tinggi (PD4-PD7)
    sebagai output
    OUT   DDRD, R16          ; Atur port D sebagai output untuk data

    ; Atur semua pin PORTB sebagai output untuk kontrol LCD
    LDI   R16, 0xFF
    OUT   DDRB, R16          ; Atur port B sebagai output untuk
    perintah

    ; Inisialisasi pin kontrol LCD
    CBI   PORTB, 0           ; EN = 0
    RCALL delay_ms           ; Tunggu LCD menyala

    ; Inisialisasi LCD
    RCALL LCD_init

    ; Kembali ke pemanggil - kontrol mobil akan menangani tampilan
    RET

LCD_init:
    LDI   R16, 0x33          ;inisialisasi LCD untuk data 4-bit
    RCALL command_wrt        ;kirim ke register perintah
    RCALL delay_ms
    LDI   R16, 0x32          ;inisialisasi LCD untuk data 4-bit
    RCALL command_wrt
    RCALL delay_ms
    LDI   R16, 0x28          ;LCD 2 baris, matriks 5x7
    RCALL command_wrt
    RCALL delay_ms
    LDI   R16, 0x0C          ;tampilan ON, kursor OFF
    RCALL command_wrt
    LDI   R16, 0x01          ;bersihkan LCD
    RCALL command_wrt
    RCALL delay_ms
    LDI   R16, 0x06          ;geser kursor ke kanan
    RCALL command_wrt
    RET
```

ii. Inisialisasi Komponen Mobil

```
init_car:
    ; Simpan register yang digunakan
    PUSH R16
    PUSH R17

    ; Inisialisasi LCD terlebih dahulu - ini akan mengatur PORTD dan
    PORTB untuk LCD
    RCALL LCD_write

    ; Sekarang inisialisasi PORTC untuk sensor IR dan buzzer
    LDI R16, 0x20 ; Atur PC5 sebagai output untuk buzzer,
    PC0-PC3 sbg input
    OUT DDRC, R16

    ; Aktifkan resistor pull-up untuk input sensor
    LDI R16, 0x0F ; Atur pull-up PC0-PC3
    OUT PORTC, R16

    ; Konfigurasi INT0 (PD2) untuk tombol darurat
    ; Atur PD2 sebagai input dengan pull-up
    CBI DDRD, 2 ; Atur PD2 sebagai input
    SBI PORTD, 2 ; Aktifkan pull-up pada PD2

    ; Konfigurasi INT0 untuk dipicu pada falling edge (penekanan tombol)
    LDI R16, (1<<ISC01) ; Falling edge dari INT0 menghasilkan
    interupsi
    STS EICRA, R16

    ; Aktifkan interupsi INT0
    LDI R16, (1<<INT0)
    OUT EIMSK, R16

    ; Inisialisasi emergency_stop ke 0 (operasi normal)
    LDI R16, 0
    STS emergency_stop, R16

    ; Aktifkan interupsi global
    SEI

    ; Re-inisialisasi PORTB untuk kontrol motor sambil mempertahankan
    pin LCD
    ; PB0 (E) dan PB1 (RS) digunakan oleh LCD
    ; PB5 (IN1), PB4 (IN2), PB3 (ENA) untuk motor maju/mundur
    IN R16, DDRB ; Baca nilai DDRB saat ini
    ORI R16, 0x38 ; PB3, PB4, PB5 sbg output sambil
    mempertahankan bit lain
    OUT DDRB, R16

    ; Konfigurasi PB2 (IN3) untuk kontrol motor kiri/kanan
    ; PB2 (IN3), PC4/A4 (IN4) untuk motor kiri/kanan
    ; Bit yang lebih tinggi dari PORTD digunakan untuk data LCD
    IN R16, DDRB ; Baca nilai DDRB saat ini
    ORI R16, 0x04 ; Atur PB2 sebagai output sambil
    mempertahankan bit lainnya
    OUT DDRB, R16

    ; Konfigurasi PC4 (A4/IN4) untuk kontrol motor kiri/kanan
    IN R16, DDRC ; Baca nilai DDRC saat ini
    ORI R16, 0x10 ; Atur PC4 sebagai output sambil
    mempertahankan bit lainnya
    OUT DDRC, R16

    ; Inisialisasi Timer2 untuk PWM pada PB3 (OC2A)
    LDI R16, (1<<WGM20)|(1<<WGM21)|(1<<COM2A1) ; Mode Fast PWM, mode
    non-inverting
    STS TCCR2A, R16
    LDI R16, (1<<CS21) ; Prescaler = 8
```

```

STS    TCCR2B, R16
LDI    R16, 200          ; Atur siklus tugas PWM (0-255)
STS    OCR2A, R16

; Tampilkan pesan awal
RCALL  display_status

; Kembalikan register
POP    R17
POP    R16
RET

```

iii. Loop Utama

```

run_car:
; Simpan register yang digunakan
PUSH   R16
PUSH   R17

; Atur arah awal ke maju
LDI    R16, 1
STS    car_direction, R16
RCALL  motor_forward

car_loop:
; Periksa apakah dalam mode berhenti darurat
LDS    R16, emergency_stop
CPI    R16, 1
BRNE   normal_operation
JMP    emergency_display

normal_operation:

; Operasi normal - baca sensor dan lanjutkan
; Baca semua sensor IR
RCALL  read_sensors

; Periksa arah saat ini dan sensor yang sesuai
LDS    R16, car_direction

CPI    R16, 0          ; Berhenti?
BRNE   not_stopped    ; Jika tidak berhenti, lanjutkan pemeriksaan

; Ketika berhenti, periksa semua arah untuk melihat apakah kita
dapat bergerak lagi
RCALL  check_front     ; Coba bergerak maju jika memungkinkan
RCALL  check_left      ; Coba belok kiri jika memungkinkan
RCALL  check_right     ; Coba belok kanan jika memungkinkan
RCALL  check_back      ; Coba bergerak mundur jika memungkinkan
RJMP   do_update_display ; Jika semua arah memiliki hambatan, hanya
perbarui tampilan

not_stopped:
CPI    R16, 1          ; Maju?
BRNE   not_forward
LDS    R16, front_status
CPI    R16, 1          ; Hambatan?
BRNE   buzzer_check_done ; Tidak ada hambatan, lanjutkan

; Penanganan hambatan di depan
RCALL  handle_forward_obstacle
RJMP   do_update_display

not_forward:
CPI    R16, 2          ; Mundur?
BRNE   not_backward

```



```

LDS R16, back_status
CPI R16, 1 ; Hambatan?
BRNE buzzer_check_done ; Tidak ada hambatan, lanjutkan

; Penanganan hambatan di belakang
RCALL handle_backward_obstacle
RJMP do_update_display

not_backward:
CPI R16, 3 ; Kiri?
BRNE not_left
LDS R16, left_status
CPI R16, 1 ; Hambatan?
BRNE buzzer_check_done ; Tidak ada hambatan, lanjutkan

; Penanganan hambatan di kiri
RCALL handle_left_obstacle
RJMP do_update_display

not_left:
CPI R16, 4 ; Kanan?
BRNE buzzer_check_done
LDS R16, right_status
CPI R16, 1 ; Hambatan?
BRNE buzzer_check_done ; Tidak ada hambatan, lanjutkan

; Penanganan hambatan di kanan
RCALL handle_right_obstacle
RJMP do_update_display

buzzer_check_done:
; Lanjutkan untuk memperbarui tampilan
RJMP do_update_display

; Fungsi penanganan untuk hambatan di berbagai arah
handle_forward_obstacle:
RCALL buzzer_on ; Nyalakan buzzer untuk hambatan di depan
RCALL check_left ; Coba temukan arah alternatif
RCALL check_right
RCALL check_back
RCALL buzzer_off ; Matikan buzzer
; Jika kita sampai di sini, semua arah memiliki hambatan
RCALL motor_stop
LDI R16, 0
STS car_direction, R16
RET

handle_backward_obstacle:
RCALL buzzer_on ; Nyalakan buzzer untuk hambatan di belakang
RCALL check_left ; Coba temukan arah alternatif
RCALL check_right
RCALL check_front
RCALL buzzer_off ; Matikan buzzer
; Jika kita sampai di sini, semua arah memiliki hambatan
RCALL motor_stop
LDI R16, 0
STS car_direction, R16
RET

handle_left_obstacle:
RCALL buzzer_on ; Nyalakan buzzer untuk hambatan di kiri
RCALL check_right ; Coba temukan arah alternatif
RCALL check_front
RCALL check_back
RCALL buzzer_off ; Matikan buzzer
; Jika kita sampai di sini, semua arah memiliki hambatan
RCALL motor_stop
LDI R16, 0
STS car_direction, R16
RET

```

```

handle_right_obstacle:
    RCALL buzzer_on          ; Nyalakan buzzer untuk hambatan di kanan
    RCALL check_left        ; Coba temukan arah alternatif
    RCALL check_front
    RCALL check_back
    RCALL buzzer_off        ; Matikan buzzer
    ; Jika kita sampai di sini, semua arah memiliki hambatan
    RCALL motor_stop
    LDI    R16, 0
    STS    car_direction, R16
    RET

emergency_display:
    ; Dalam mode berhenti darurat, hanya perbarui tampilan dan lanjutkan
loop
    RCALL display_status

    ; Tampilkan pesan "EMERGENCY" atau indikator khusus
    ; Ini ditangani dalam fungsi display_status

    ; Tunda kecil
    LDI    R17, 2
emergency_delay_loop:
    RCALL delay_ms
    DEC    R17
    BRNE   emergency_delay_loop

    ; Lanjutkan loop utama
    RJMP   car_loop

do_update_display:
    ; Perbarui LCD dengan status saat ini
    RCALL display_status

    ; Tunda kecil
    LDI    R17, 2
delay_loop:
    RCALL delay_ms
    DEC    R17
    BRNE   delay_loop

    ; Lanjutkan loop utama
    RJMP   car_loop

run_car_exit:
    ; Kembalikan register dan kembali (hanya dicapai jika kita secara
    eksplisit melompat ke sini)
    POP    R17
    POP    R16
    RET

```

iv. Pengambilan Keputusan (*contohnya: check_left*)

```

check_left:
    LDS    R16, left_status
    CPI    R16, 1          ; Hambatan di kiri?
    BREQ   check_left_exit ; Ya, keluar

    ; Tidak ada hambatan di kiri, belok kiri
    LDI    R16, 3
    STS    car_direction, R16
    RCALL motor_left
    RCALL display_status

    ; Tunda kecil untuk memungkinkan belok

```

```

        LDI    R17, 10
left_delay:
        RCALL  delay_ms
        DEC    R17
        BRNE   left_delay

        ; Kembali ke loop utama
        RJMP   car_loop

check_left_exit:
        RET

```

v. Mode Darurat

```

INT0_vect:
        ; Simpan register yang digunakan
        PUSH   R16

        ; Toggle status emergency_stop
        LDS    R16, emergency_stop
        CPI    R16, 0
        BRNE   emergency_to_normal

        ; Normal -> Darurat: Simpan arah saat ini dan berhenti
        LDS    R16, car_direction
        STS    prev_direction, R16
        LDI    R16, 1
        STS    emergency_stop, R16
        RCALL  motor_stop
        RCALL  buzzer_on
        RJMP   int0_exit

emergency_to_normal:
        ; Darurat -> Normal: Kembalikan arah sebelumnya
        LDI    R16, 0
        STS    emergency_stop, R16
        RCALL  buzzer_off

        ; Kembalikan arah sebelumnya dan mulai bergerak lagi
        LDS    R16, prev_direction
        STS    car_direction, R16

        ; Periksa arah mana yang akan dikembalikan dan mulai motor sesuai
        CPI    R16, 0          ; Jika arah sebelumnya adalah berhenti,
        tetap berhenti
        BREQ   restore_done

        CPI    R16, 1          ; Maju?
        BRNE   restore_check_backward
        RCALL  motor_forward
        RJMP   restore_done

restore_check_backward:
        CPI    R16, 2          ; Mundur?
        BRNE   restore_check_left
        RCALL  motor_backward
        RJMP   restore_done

restore_check_left:
        CPI    R16, 3          ; Kiri?
        BRNE   restore_check_right
        RCALL  motor_left
        RJMP   restore_done

restore_check_right:
        CPI    R16, 4          ; Kanan?

```

```
    BRNE restore_done
    RCALL motor_right

restore_done:
    RJMP int0_exit

int0_exit:
    ; Kembalikan register
    POP R16
    RETI
```

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

Pengujian dilakukan untuk memastikan seluruh fungsi utama dari mobil mini Arduino GoVan dapat berjalan sesuai desain dan spesifikasi.

Pengujian meliputi beberapa skenario sebagai berikut:

- **Pengujian Pergerakan Default**

Mobil diuji berjalan maju secara otomatis pada kondisi jalur bebas hambatan. Pengujian ini bertujuan memverifikasi kontrol motor dan pengaturan PWM pada driver motor L298 sudah berjalan dengan baik.

- **Pengujian Deteksi Hambatan**

Sensor infrared yang terpasang pada empat sisi (depan, belakang, kiri, kanan) diuji untuk mendeteksi keberadaan hambatan. Ketika hambatan terdeteksi pada arah tertentu, sistem diharapkan mampu menghentikan pergerakan ke arah tersebut dan memilih jalur alternatif yang aman.

- **Pengujian Respons Tombol Darurat**

Fungsi tombol *emergency* yang terhubung ke interupsi eksternal INT0 diuji untuk memastikan mobil dapat berhenti secara instan pada saat tombol ditekan, serta dapat melanjutkan operasi normal saat tombol ditekan kembali.

- **Pengujian Respons Sensor dan Buzzer**

Pengujian dilakukan untuk mengamati waktu respons sensor infrared terhadap hambatan dan perilaku *buzzer* yang memberikan peringatan suara saat hambatan terdeteksi.

Pengujian dilakukan dengan pengamatan langsung, pencatatan nilai sensor, status motor, serta tampilan status pada layar LCD.

3.2 RESULT

Hasil pengujian:

- **Pergerakan Maju Otomatis Tanpa Hambatan**

Mobil berjalan maju secara lancar saat tidak terdapat hambatan di depan. Hal ini membuktikan bahwa modul motor dengan driver L298 dan pengaturan PWM sudah terkonfigurasi dan berfungsi dengan baik. Motor menerima sinyal sesuai arah pergerakan yang diinginkan.

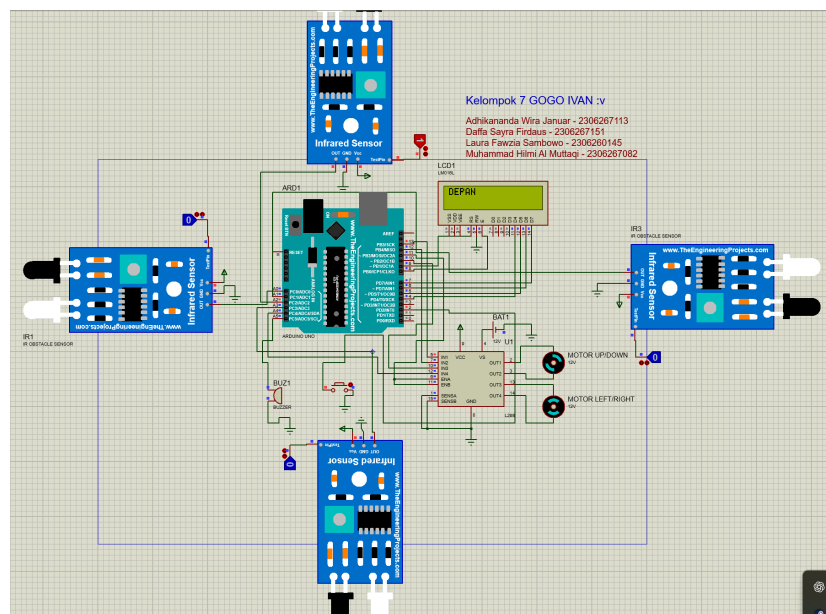


Fig. 3.1 Forward movement

- **Pergerakan Maju Otomatis Tanpa Hambatan**

Mobil berjalan maju secara lancar saat tidak terdapat hambatan di depan. Hal ini membuktikan bahwa modul motor dengan driver L298 dan pengaturan PWM sudah terkonfigurasi dan berfungsi dengan baik. Motor menerima sinyal sesuai arah pergerakan yang diinginkan.

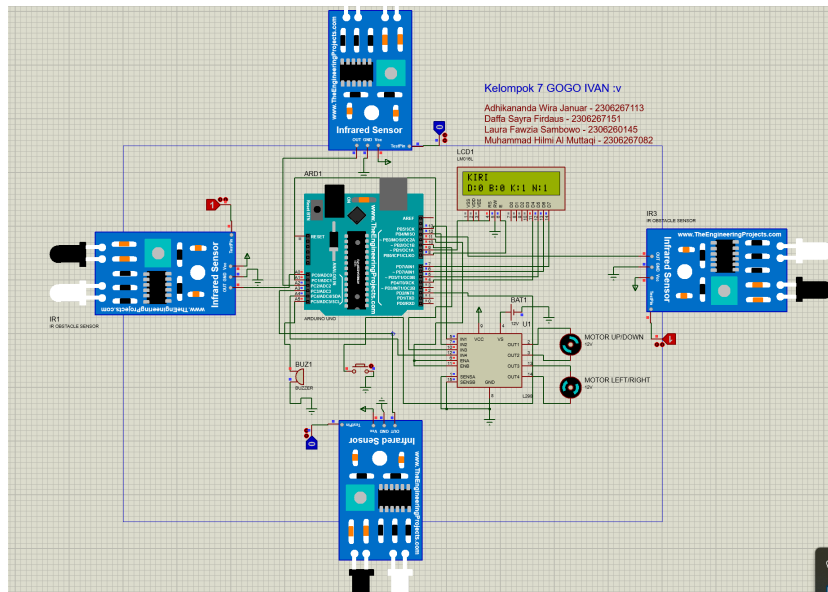


Fig. 3.2 Left avoidance

- **Fungsi Emergency Stop**

Saat tombol darurat ditekan, interupsi eksternal INT0 memicu handler yang segera menghentikan kedua motor dan mengaktifkan *buzzer* sebagai tanda kondisi darurat. Tombol darurat ini dapat ditekan kembali untuk melanjutkan operasi normal mobil, dimana arah pergerakan terakhir sebelum *emergency* disimpan dan diaktifkan ulang.

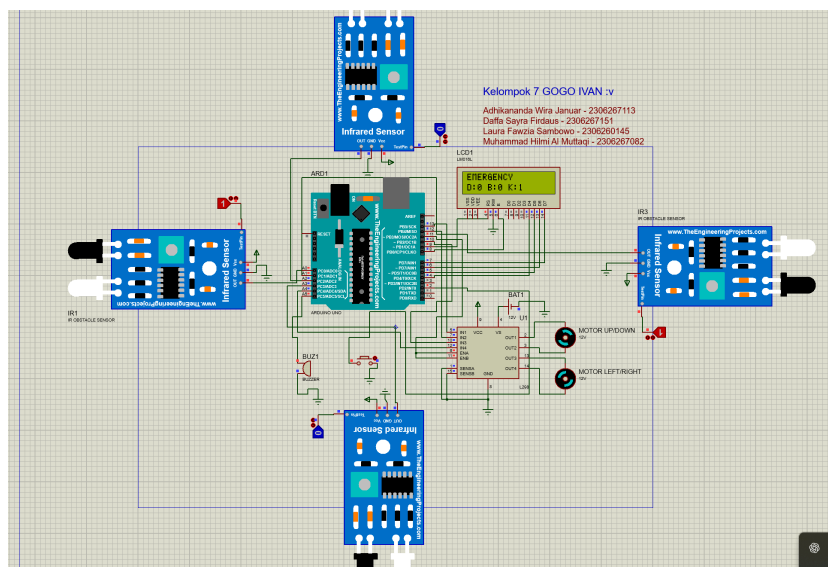


Fig. 3.3 Emergency stop

- **Delay Pembacaan Sensor Infrared**

Pengujian terhadap sinyal *output* sensor infrared menunjukkan adanya delay antara nilai tegangan *output* (V_{out}) dengan pembacaan pada testpin mikrokontroler. Delay ini menyebabkan nilai yang terbaca tidak langsung sama dengan kondisi aktual hambatan, sehingga berpotensi memperlambat reaksi mobil saat menghadapi rintangan.

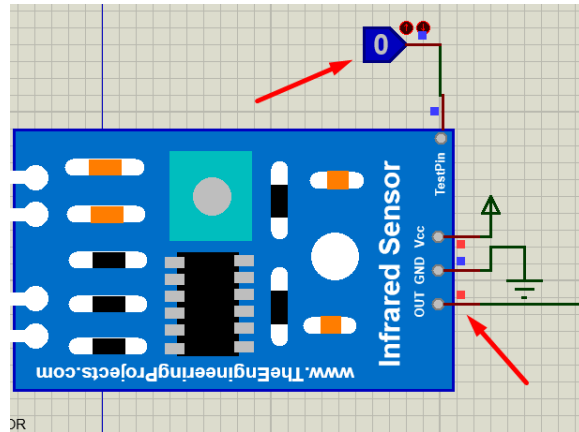


Fig. 3.4 Sensor delay

- **Kinerja Buzzer Saat Hambatan Terdeteksi**

Buzzer yang terhubung ke pin PC5 menyala terus-menerus selama hambatan terdeteksi dan hanya mati ketika mode *emergency* aktif atau motor berhenti. Kondisi ini kurang ideal karena suara *buzzer* yang terus menyala dapat mengganggu dan kurang efisien.

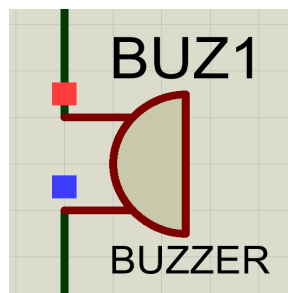


Fig. 3.5 Buzzer

3.3 EVALUATION

Mobil mini (GoVan) berhasil menunjukkan kemampuan bergerak maju secara mandiri dengan navigasi menggunakan sensor infrared multi-arrah yang efektif mendeteksi hambatan dari berbagai sisi. Mekanisme pengecekan berurutan memungkinkan sistem menentukan jalur alternatif dengan baik. Pengendalian kecepatan motor menggunakan sinyal PWM menghasilkan pergerakan yang halus dan stabil, sementara tombol *emergency stop* berfungsi efektif dengan aktivasi *buzzer* dan indikator sebagai tanda peringatan.

Meski demikian, terdapat keterbatasan pada delay pembacaan sensor infrared yang mempengaruhi respons terhadap hambatan secara cepat. Pengaturan buzzer yang saat ini menyala terus-menerus saat hambatan terdeteksi juga kurang efisien, sehingga diperlukan pengendalian yang lebih teratur atau penggunaan IC khusus. Sistem modular dengan komunikasi I2C dan monitoring real-time melalui komunikasi serial sudah mendukung operasi yang efisien, namun masih ada ruang pengembangan untuk meningkatkan performa dan kenyamanan pengguna secara keseluruhan.

CHAPTER 4

CONCLUSION

Proyek mobil mini Arduino GoVan telah berhasil memenuhi tujuan utama yang ditetapkan sejak awal, yaitu mampu bergerak maju secara otomatis dan mandiri serta melakukan navigasi dengan sensor infrared multi-arrah yang efektif dalam mendeteksi dan menghindari hambatan dari berbagai arah. Sistem pengendalian motor menggunakan sinyal PWM terbukti mampu menghasilkan pergerakan yang halus, stabil, dan dapat diprediksi, sehingga mendukung keamanan dan kelancaran operasi kendaraan. Selain itu, fitur *emergency stop* yang terhubung ke *interrupt* eksternal berfungsi dengan baik untuk menghentikan mobil secara instan, dengan aktivasi *buzzer* dan indikator sebagai tanda peringatan yang membantu meningkatkan aspek keselamatan.

Dalam pengujian GoVan, beberapa kendala ditemukan. *Delay* sensor infrared memperlambat respons terhadap hambatan, sehingga perlu optimasi sensor atau penggantian dengan yang lebih cepat. *Buzzer* yang terus menyala saat deteksi hambatan juga tidak efisien, sehingga kontrol *buzzer* teratur atau IC khusus disarankan. Sistem modular dengan I2C dan monitoring serial telah berjalan baik, namun peningkatan fungsi dan performa masih memungkinkan.

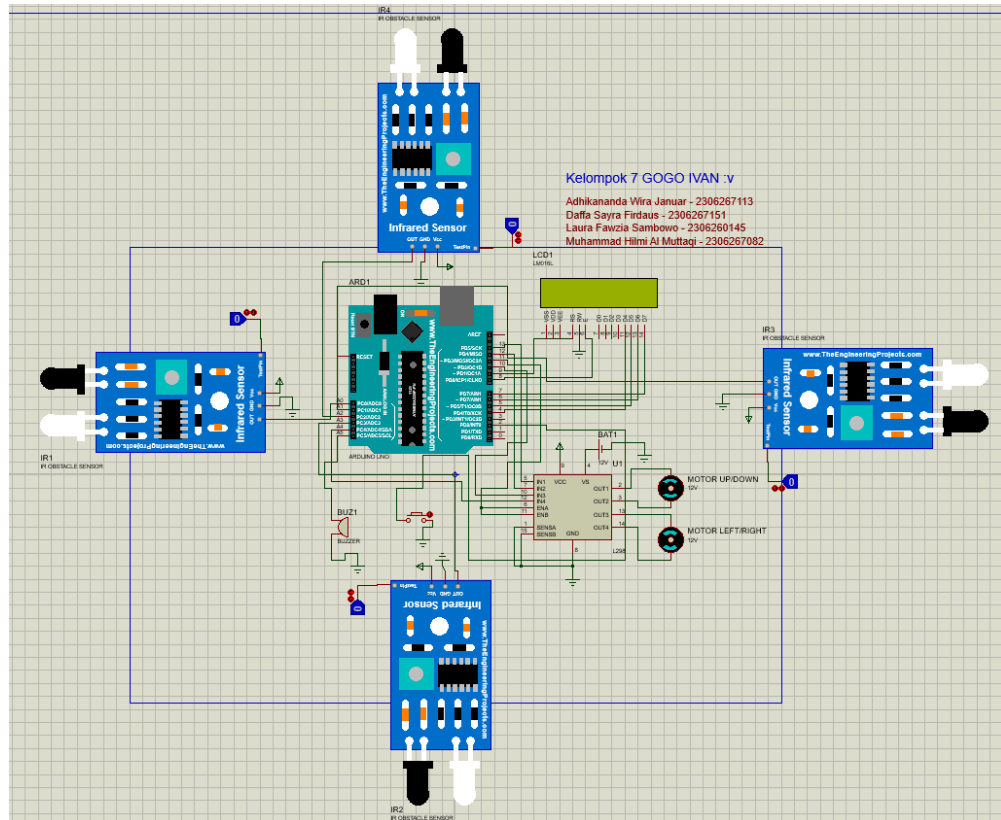
Secara keseluruhan, prototipe ini tidak hanya memberikan solusi praktis dalam pengantaran barang ringan secara otomatis di lingkungan kerja terbatas, tetapi juga berfungsi sebagai media pembelajaran yang efektif untuk memahami konsep sistem tertanam dan pemrograman tingkat rendah menggunakan bahasa *Assembly*. Dengan perbaikan dan pengembangan lebih lanjut, sistem ini berpotensi menjadi kendaraan pengantar yang lebih responsif, andal, dan nyaman digunakan dalam berbagai aplikasi nyata.

REFERENCES

- [1] Sistem Embedded dan Praktikum - Sistem Embedded 01, “Ch 16: OUTPUT: DC Motor & PWMFile,” EMAS2 UI. [Online]. Available: <https://emas2.ui.ac.id/mod/resource/view.php?id=2960947>. (Accessed: May. 16, 2025)
- [2] Anas Kuzechie, “Assembly via Arduino (part 5) - Programming LCD,” YouTube. [Online]. Available: <https://youtu.be/U8OF9N5rULw?si=V0nRJm9oe72xMXQ4>. (Accessed: May. 16, 2025)
- [3] GeeksforGeeks, “Branch Instructions in AVR Microcontroller,” GeeksforGeeks. [Online]. Available: <https://www.geeksforgeeks.org/branch-instructions-in-avr-microcontroller/>. (Accessed: May. 16, 2025)
- [4] HiBit, “How to Use the L298N Motor Driver Module,” HiBit. [Online]. Available: <https://www.hibit.dev/posts/89/how-to-use-the-l298n-motor-driver-module>. (Accessed: May. 16, 2025)
- [5] Circuit Digest, “Interfacing IR Sensor Module with Arduino,” Circuit Digest. [Online]. Available: <https://circuitdigest.com/microcontroller-projects/interfacing-ir-sensor-module-with-arduino>. (Accessed: May. 16, 2025)

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation

Link Repo: https://github.com/DAFFAsd/PA_MBD_Kelompok7