FINAL PROJECT REPORT DEPARTMENT OF ELECTRICAL ENGINEERING UNIVERSITY OF INDONESIA

PARKING LOT SENSOR

GROUP 22

Ibnu Zaky Fauzi	**2306161870**
Muhammad Ikhsan Kurniawan	**2306210784**
Muhammad Raditya Alif Nugroho	**2306212745**
Wiellona Darlene Oderia Saragih	**2306264396**

PREFACE

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan laporan proyek akhir yang berjudul "Parking Lot Sensor" ini dengan baik dan tepat waktu. Laporan ini disusun sebagai salah satu syarat untuk memenuhi tugas pada mata kuliah Praktikum Sistem Embedded pada program studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

Di tengah kemajuan teknologi yang pesat, kebutuhan akan sistem otomatis yang dapat mempermudah aktivitas manusia semakin meningkat. Salah satu inovasi yang sangat membantu adalah sistem sensor parkir. Melalui proyek ini, kami merancang dan mengimplementasikan sistem Parking Lot Sensor yang mampu mendeteksi ketersediaan slot parkir secara otomatis menggunakan sensor ultrasonik dan Arduino sebagai unit pengendali.

Kami menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kami sangat terbuka terhadap kritik dan saran yang membangun demi perbaikan di masa yang akan datang. Semoga laporan ini dapat memberikan manfaat serta menjadi referensi bagi pengembangan teknologi di bidang sistem parkir otomatis. Akhir kata, kami mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dan bantuan dalam menyelesaikan proyek ini.

Depok, May 16, 2025

Group 22

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Di era modern saat ini, peningkatan jumlah kendaraan pribadi menjadi tantangan tersendiri dalam pengelolaan ruang publik terutama pada area parkir. Keterbatasan lahan parkir dan kurangnya informasi real-time mengenai ketersediaan slot parkir seringkali menyebabkan kemacetan dan pemborosan waktu bagi pengemudi yang mencari tempat parkir.

Salah satu solusi yang dapat diterapkan untuk mengatasi permasalahan ini adalah dengan menggunakan sistem sensor parkir otomatis yang mampu mendeteksi dan menghitung jumlah slot parkir yang tersedia secara real-time. Sistem ini akan membantu pengemudi untuk mengetahui jumlah ruang parkir yang masih kosong tanpa harus berkeliling area parkir sehingga lebih efisien dan terorganisir.

Berdasarkan hal tersebut, kami tertarik untuk merancang dan membangun sistem sensor parkir berbasis Arduino yang mampu menghitung secara otomatis jumlah slot parkir yang tersedia di suatu area. Sistem ini akan memanfaatkan sensor ultrasonik HC-SR04 pada setiap slot untuk mendeteksi apakah slot tersebut terisi atau kosong. Data dari sensor akan diproses oleh Arduino Master dan ditampilkan pada monitor virtual sebagai informasi jumlah slot kosong yang tersedia.

Proyek ini bertujuan untuk menciptakan sistem parkir yang efisien dan mudah diakses, yang dapat digunakan di berbagai lokasi seperti pusat perbelanjaan, gedung perkantoran, dan fasilitas umum lainnya. Dengan sistem ini, diharapkan pengelolaan tempat parkir dapat menjadi lebih teratur dan pengemudi dapat menghemat waktu bagi para pengemudi mobil.

1.2 PROPOSED SOLUTION

Proyek ini dirancang untuk memantau ketersediaan slot parkir secara otomatis menggunakan sensor ultrasonik HC-SR04 yang terpasang pada tempat parkir. Sensor ini akan mendeteksi keberadaan kendaraan dengan cara memancarkan gelombang ultrasonik dan mengukur waktu pantulan dari objek untuk menentukan jarak. Sistem terdiri dari dua unit Arduino, yaitu Arduino Slave dan Arduino Master. Arduino Master bertugas membaca data dari sensor ultrasonik yang mewakili satu slot parkir. Jika sensor mendeteksi kendaraan pada suatu slot, maka informasi ini dikirimkan ke Arduino Slave melalui komunikasi serial. Arduino Slave kemudian mengatur indikator LED sebagai penanda visual. Jika slot parkir

kosong, maka LED hijau akan menyala. Sebaliknya, jika slot terisi oleh kendaraan, maka LED merah akan menyala. Selain itu, sistem juga dilengkapi dengan tampilan monitor virtual untuk menunjukkan jumlah slot kosong secara real-time. Dengan sistem ini, pengguna dapat mengetahui status tempat parkir dengan lebih mudah dan efisien tanpa harus memeriksa satu per satu secara manual.

1.3 ACCEPTANCE CRITERIA

Tujuan dari proyek ini adalah:

1. Akurasi Deteksi Slot Parkir : Sistem dapat secara akurat mendeteksi apakah sebuah slot parkir sedang kosong atau terisi kendaraan menggunakan sensor ultrasonik HC-SR04.
2. Indikator Visual yang Responsif : LED hijau menyala saat slot parkir kosong, dan LED merah menyala saat slot parkir terisi, tanpa adanya keterlambatan atau kesalahan tampilan.
3. Tampilan Jumlah Slot Tersedia : Jumlah slot parkir yang tersedia ditampilkan secara real-time pada monitor virtual atau display yang digunakan dengan pembaruan otomatis setiap kali ada perubahan status pada salah satu slot.
4. Skalabilitas Sistem : Sistem dapat dengan mudah dikembangkan untuk menambah jumlah slot parkir tanpa perubahan besar pada perangkat lunak atau perangkat keras.
5. Keamanan dan Keandalan Operasional : Sistem bekerja stabil dalam jangka waktu lama tanpa gangguan/error pada sensor, komunikasi, atau indikator.

1.4 ROLES AND RESPONSIBILITIES

Berikut adalah peran dan tanggung jawab yang diberikan untuk tiap anggota kelompok:

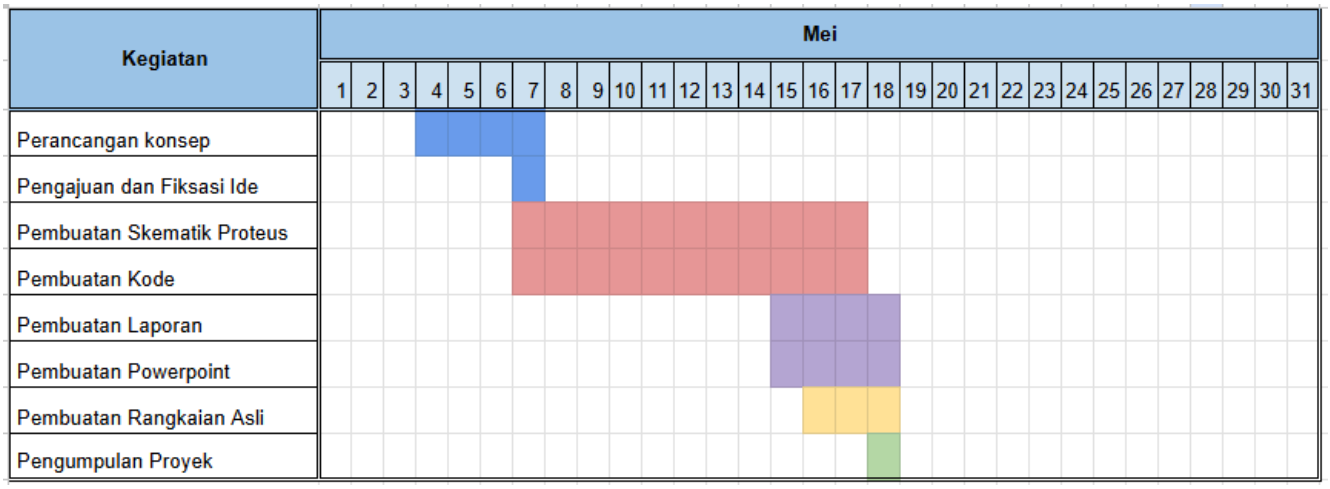
Roles	Responsibilities	Person
Role 1	● Laporan bab 1, 3, dan 4 ● Finishing kode ● PPT ● Support system	Ibnu Zaky Fauzi
Role 2	● Pengembangan kode lebih lanjut ● Rangkaian asli ● PPT ● Support system	Muhammad Ikhsan Kurniawan

Role 3	• Laporan bab 1.5 dan bab 2 • Pengembangan kode lebih lanjut • README • PPT • Support system	Muhammad Raditya Alif Nugroho
Role 4	• Rangkaian skematik proteus • Rangkaian asli • Dasar dari kode • README • PPT • Support system	Wiellona Darlene Oderia Saragih

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

Gantt Chart untuk timeline pengerjaan proyek akhir



CHAPTER 2

IMPLEMENTATION

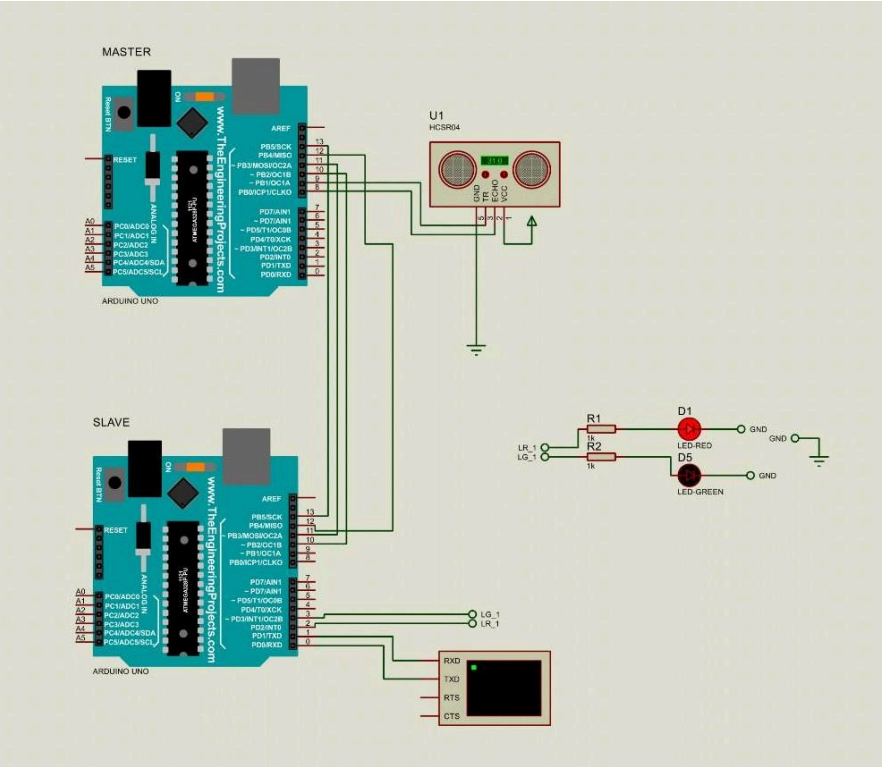
2.1 HARDWARE DESIGN AND SCHEMATIC

Parking lot sensor ini memerlukan beberapa jenis komponen untuk menjalankan fungsinya dengan benar. Dalam keseluruhan sistem ini, kami menggunakan dua arduiono yang berperan sebagai master dan slave. Master bertanggung jawab atas sensor jarak HC-SR04, sementara slave bertanggung jawab terhadap output ke LED dan LCD Display

Sensor HC-SR04 digunakan untuk mendeteksi keberadaan mobil di dalam slot parkirnya. Sensor akan mendeteksi jarak dari benda yang ada di depannya. Jika jaraknya berada di bawah 30 cm, maka dianggap ada mobil yang mengisi parkir. Jika jaraknya berada di atas 30 cm, maka dianggap tidak ada mobil yang mengisi parkir. Berdasarkan nilai 30 cm yang dibandingkan, sistem akan menentukan LED mana yang dinyalakan. Jika jarak < 30 cm, maka LED Hijau akan menyala, menandakan slot parkir kosong. Jika jarak > 30 cm, maka LED Merah akan menyala, menandakan slot parkir terisi. Komponen yang digunakan dalam proyek ini adalah sebagai berikut:

- Arduino Uno dengan Microcontroller ATMega328P
- Sensor jarak HC-SR04
- Breadboard
- LED Merah dan Hijau
- Kabel Jumper
- Resistor
- LCD

Untuk mengendalikan sistem ini, kami menggunakan kedua arduino. Arduino master yang terhubung ke HC-SR04 akan mengirimkan informasi jarak ke arduino slave. Arduino slave akan mengolah informasi yang diberikan dan menentukan LED yang menyala. Dengan menggunakan dua Arduino, sistem ini memungkinkan pengukuran jarak yang lebih akurat dan responsif. Penggunaan LED akan memberikan respons yang mempermudah pengguna dalam menentukan keadaan slot parkir



2.2 SOFTWARE DEVELOPMENT

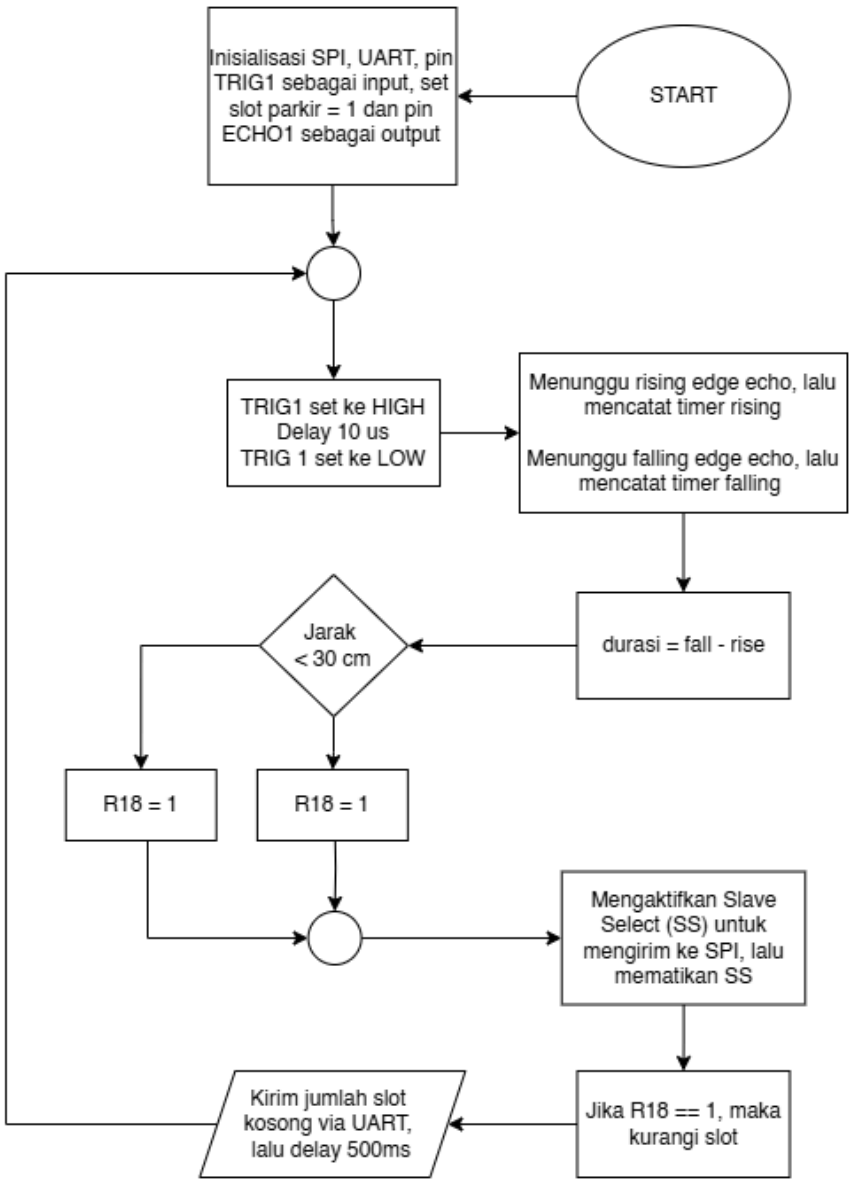
Pada proses pengembangan software ini, kami membuat kode yang sesuai dengan materi yang sudah dipelajari selama praktikum. Proyek ini melibatkan penggunaan sensor HC-SR04 untuk membaca jarak, serta LED Merah dan LED Hijau yang dikendalikan berdasarkan jarak yang terdeteksi.

Selain itu, kami juga mengimplementasikan komunikasi antara dua Arduino, yaitu Arduino master dan Arduino slave. Arduino master bertugas untuk menerima data dari sensor HC-SR04, mengukur jarak, dan membandingkannya dengan batas yang telah ditentukan. Selanjutnya, hasil pengukuran tersebut dikirimkan ke Arduino slave. Arduino slave akan mengaktifkan salah satu LED sesuai dengan jarak yang diterima:

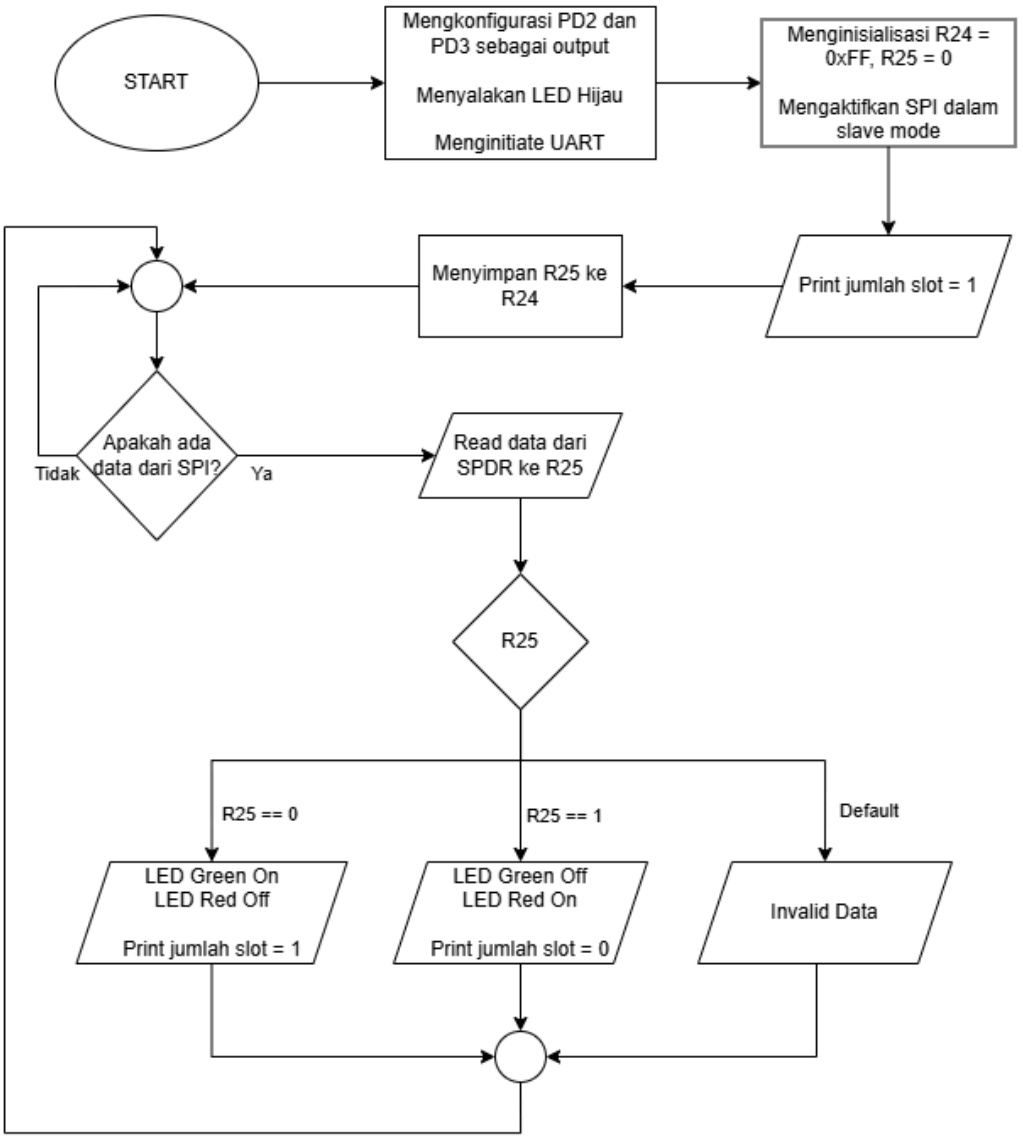
- LED Merah menyala dan hijau mati untuk jarak kurang dari 30 cm.
- LED Merah mati dan hijau menyala untuk jarak lebih dari 30 cm.

Sebelum membuat kode untuk rangkaian, pertama-tama kami membuat *flowchart* untuk menjadi gambaran dari jalannya program. Flowchart ini akan sangat membantu dalam merancang logika dari keseluruhan program, dan memastikan bahwa respons yang didapat bisa sesuai dengan perkiraan

A. Flowchart master.S



B. Flowchart slave.s



2.3 HARDWARE AND SOFTWARE INTEGRATION

Proses integrasi antara hardware dan software melibatkan arduino yang berperan sebagai pengendali, dan komponen-komponen seperti sensor dan LED yang berperan sebagai input dan output. Kode master dan slave yang sudah dibuat diupload ke arduino agar data dari sensor bisa diproses dan diolah sebagai output ke LED. Dengan adanya integrasi ini, tercipta sistem yang dapat menampilkan status slot parkir dan dapat mempermudah pengalaman pengguna dalam mencari parkir

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

Pada proyek ini akan dilakukan pengujian, pengujian sendiri dilakukan untuk memastikan sistem yang sudah dibuat bekerja dengan semestinya dan melakukan pengecekan output dari sistem berdasarkan input yang berbeda-beda.

- Menyalakan sistem: : Menyalakan sistem bertujuan untuk menguji apakah sistem dapat dinyalakan saat sistem dalam keadaan mati. Ketika sistem dinyalakan, maka sensor akan masuk kedalam mode reading untuk mengecek apakah ada objek atau tidak didepannya
- Pengujian sensor jarak: Ketika sistem sudah dinyalakan, sensor jarak HC-SR04 akan mulai membaca dan memberikan input kepada microcontroller. Pengukuran jarak dibagi ke dalam dua kategori yaitu jarak kategori jauh dan dekat, jika jauh maka led hijau akan menyala dan menandakan bahwa parkiran masih memiliki slot kosong, jika dekat maka led merah akan menyala dan menandakan bahwa slot parkir sudah terisi. Berikut adalah kode rangkaian yang dibuat pada Arduino IDE

- Master Arduino Code

```
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;-----------------------
.global main
;=================================================================
main:
.equ SCK, 5    ; PB5 (pin 13)
.equ MOSI, 3   ; PB3 (pin 11)
.equ SS, 2     ; PB2 (pin 10)
.equ TRIG1, 1  ; PB1 (pin 9)
.equ ECHO1, 0  ; PB0 (pin 8)

.equ TOTAL_SPOTS, 1   ; Only one parking spot

;-----------------------------------------------------------------
    ; Initialize UART for Serial Monitor first
    RCALL uart_init
```

```
    ; Initialize R19 and R26 for status comparison (status change
detection)
    LDI R19, 0xFF        ; Initial current status (invalid to force first
message)
    LDI R26, 0           ; Initial previous status

    ; Print welcome message to serial monitor
    LDI ZL, lo8(welcome_msg)
    LDI ZH, hi8(welcome_msg)
    RCALL print_string
    RCALL print_newline

    ; Initialize SPI as master
    LDI R17, (1<<MOSI)|(1<<SCK)|(1<<SS)
    OUT DDRB, R17        ; Set MOSI, SCK, SS as output
    LDI R17, (1<<SPE)|(1<<MSTR)|(1<<SPI2X)|(1<<SPR0)
    OUT SPCR, R17        ; Enable SPI as master, fsck=fosc/8, SPI mode 0

    ; Initial test message
    LDI ZL, lo8(spi_initialized_msg)
    LDI ZH, hi8(spi_initialized_msg)
    RCALL print_string
    RCALL print_newline
;-----------------------------------------------------------------
    ; Set trigger pin as output and echo pin as input
    SBI DDRB, TRIG1     ; PB1 as output (Trigger1)
    CBI DDRB, ECHO1     ; PB0 as input (Echo1)

    ; Send a test command to verify SPI is working
    LDI ZL, lo8(sending_test_msg)
    LDI ZH, hi8(sending_test_msg)
    RCALL print_string
    RCALL print_newline

    ; Send test value 0 (LED free status)
    LDI R18, 0
    RCALL send_spi

    ; Send test value 1 (LED occupied status)
    LDI R18, 1
    RCALL send_spi

    ; Send test value 0 again (LED free status)
    LDI R18, 0
    RCALL send_spi
```

```
    ; Initialize available spot counter
    LDI R24, TOTAL_SPOTS   ; Start with all spots available

;-----------------------------------------------------------
main_loop:
    ; Store previous status in R26 for comparison later
    MOV R26, R19  ; Save the previous status

    ; Generate trigger pulse for sensor
    SBI PORTB, TRIG1       ; Set trigger high
    RCALL delay_10us       ; Wait 10 microseconds
    CBI PORTB, TRIG1       ; Set trigger low

    ; Measure echo pulse
    RCALL check_echo_PB0

    ; Check if spot is occupied
    CPI R28, 30            ; Compare with threshold (30cm)
    BRSH spot1_free        ; If distance >= 30cm, spot is free

    ; Spot is occupied (distance < 30cm)
    LDI R18, 1             ; Spot occupied code = 1
    RJMP spot1_status_set

spot1_free:
    LDI R18, 0             ; Spot free code = 0

spot1_status_set:
    ; Save status value for later use
    MOV R19, R18

    ; Only print info if the status has changed
    CP R19, R26
    BREQ skip_status_print  ; Skip printing if status hasn't changed

    ; Print separator for clarity
    LDI ZL, lo8(separator_msg)
    LDI ZH, hi8(separator_msg)
    RCALL print_string
    RCALL print_newline

    ; Print change notification
    LDI ZL, lo8(status_change_msg)
    LDI ZH, hi8(status_change_msg)
```

```
    RCALL print_string
    RCALL print_newline

    ; Print distance to serial monitor
    LDI ZL, lo8(distance_msg)
    LDI ZH, hi8(distance_msg)
    RCALL print_string

    MOV R18, R28          ; Copy distance value to R18
    RCALL print_number    ; Print the distance value

    LDI R18, 'c'          ; Print 'cm'
    RCALL send_char
    LDI R18, 'm'
    RCALL send_char
    RCALL print_newline

    ; Print spot status based on R19
    CPI R19, 1
    BREQ print_occupied

    ; Print free status
    LDI ZL, lo8(free_msg)
    LDI ZH, hi8(free_msg)
    RCALL print_string
    RCALL print_newline
    RJMP status_printed

print_occupied:
    ; Print occupied status
    LDI ZL, lo8(occupied_msg)
    LDI ZH, hi8(occupied_msg)
    RCALL print_string
    RCALL print_newline

status_printed:
    ; Debug print the value being sent via SPI
    LDI ZL, lo8(sending_msg)
    LDI ZH, hi8(sending_msg)
    RCALL print_string
    MOV R18, R19
    RCALL print_number
    RCALL print_newline

skip_status_print:
```

```
      ; Send status to slave via SPI
    MOV R18, R19          ; Restore value to R18 for SPI
    RCALL send_spi        ; Send to slave

    ; Calculate available spots and send to serial monitor
    LDI R24, TOTAL_SPOTS   ; Start with all spots available

    ; Check spot 1
    CPI R19, 1
    BRNE send_to_monitor
    DEC R24               ; Decrease available spots if occupied

send_to_monitor:
    ; Only print available spots if status changed
    CP R19, R26
    BREQ skip_available_print

    ; Print available spots message
    LDI ZL, lo8(available_msg)
    LDI ZH, hi8(available_msg)
    RCALL print_string

    ; Send available spots count to serial monitor
    MOV R18, R24
    RCALL print_number    ; Print available spots count
    RCALL print_newline

skip_available_print:
    RCALL delay_1000ms    ; Longer delay before next loop
    RJMP main_loop

;===============================================================
send_spi:
; Function to send data via SPI
; Input: R18 = data to send
    PUSH R19
    PUSH R20

    ; Only print debug messages if status changed
    CP R19, R26
    BREQ skip_spi_debug_msg

    ; Debug message
    MOV R20, R18          ; Preserve data byte
    LDI ZL, lo8(spi_sending_msg)
```

```
    LDI ZH, hi8(spi_sending_msg)
    RCALL print_string
    MOV R18, R20
    RCALL print_number
    RCALL print_newline
    RJMP continue_spi

skip_spi_debug_msg:
    MOV R20, R18          ; Preserve data byte

continue_spi:
    CBI PORTB, SS         ; Enable slave device (SS low)

    OUT SPDR, R20         ; Transmit byte to slave

spi_wait:
    IN R19, SPSR
    SBRS R19, SPIF        ; Wait for byte transmission
    RJMP spi_wait         ; to complete

    SBI PORTB, SS         ; Disable slave device (SS high)

    ; Only print debug messages if status changed
    CP R19, R26
    BREQ skip_spi_complete_msg

    ; Debug message
    LDI ZL, lo8(spi_complete_msg)
    LDI ZH, hi8(spi_complete_msg)
    RCALL print_string
    RCALL print_newline

skip_spi_complete_msg:
    POP R20
    POP R19
    RET
;===================================================================
check_echo_PB0:
; Function to measure echo pulse width on PB0
    LDI R20, 0b00000000
    STS TCCR1A, R20       ; Timer 1 normal mode
    LDI R20, 0b11000101   ; Set for rising edge detection &
    STS TCCR1B, R20       ; prescaler=1024, noise cancellation ON

wait_rise_pb0:
```

```
    IN R21, TIFR1
    SBRS R21, ICF1
    RJMP wait_rise_pb0    ; Loop until rising edge is detected

    LDS R16, ICR1L        ; Store count value at rising edge

    OUT TIFR1, R21        ; Clear flag for falling edge detection
    LDI R20, 0b10000101
    STS TCCR1B, R20       ; Set for falling edge detection

wait_fall_pb0:
    IN R21, TIFR1
    SBRS R21, ICF1
    RJMP wait_fall_pb0    ; Loop until falling edge is detected

    LDS R28, ICR1L        ; Store count value at falling edge

    SUB R28, R16          ; Count diff R28 = R28 - R16
    OUT TIFR1, R21        ; Clear flag for next sensor reading
    RET
;============================================================
uart_init:
; Initialize UART for 9600 baud rate
    LDI R17, 103          ; For 9600 baud @16MHz
    STS UBRR0L, R17
    LDI R17, 0
    STS UBRR0H, R17

    LDI R17, (1<<TXEN0)   ; Enable transmitter
    STS UCSR0B, R17

    LDI R17, (1<<UCSZ01)|(1<<UCSZ00) ; 8-bit data
    STS UCSR0C, R17
    RET
;============================================================
send_char:
; Send single character in R18 to UART
    PUSH R19
wait_tx_ready:
    LDS R19, UCSR0A
    SBRS R19, UDRE0       ; Wait if buffer is full
    RJMP wait_tx_ready

    STS UDR0, R18         ; Send data
    POP R19
```

```
    RET
;=================================================================
print_string:
; Print null-terminated string pointed to by Z register
    PUSH R18
print_string_loop:
    LPM R18, Z+           ; Load character from program memory
    CPI R18, 0            ; Check for null terminator
    BREQ print_string_done
    RCALL send_char       ; Send character
    RJMP print_string_loop
print_string_done:
    POP R18
    RET
;=================================================================
print_number:
; Print number in R18 (0-255)
    PUSH R18
    PUSH R19

    CPI R18, 100          ; Check if number >= 100
    BRLO tens_digit       ; If less than 100, skip to tens digit

    ; Print hundreds digit
    LDI R19, '0'          ; Start with '0'
hundreds_loop:
    CPI R18, 100          ; Check if >= 100
    BRLO hundreds_done    ; If < 100, done with hundreds
    SUBI R18, 100         ; Subtract 100
    INC R19               ; Increment digit
    RJMP hundreds_loop    ; Continue loop
hundreds_done:
    PUSH R18              ; Save remainder
    MOV R18, R19          ; Move hundreds digit to R18
    RCALL send_char       ; Print hundreds digit
    POP R18               ; Restore remainder

tens_digit:
    CPI R18, 10           ; Check if number >= 10
    BRLO ones_digit       ; If less than 10, skip to ones digit

    ; Print tens digit
    LDI R19, '0'          ; Start with '0'
tens_loop:
    CPI R18, 10           ; Check if >= 10
```

```
    BRLO tens_done          ; If < 10, done with tens
    SUBI R18, 10            ; Subtract 10
    INC R19                 ; Increment digit
    RJMP tens_loop          ; Continue loop
tens_done:
    PUSH R18                ; Save remainder
    MOV R18, R19            ; Move tens digit to R18
    RCALL send_char         ; Print tens digit
    POP R18                 ; Restore remainder

ones_digit:
    ; Print ones digit
    SUBI R18, -'0'          ; Convert to ASCII
    RCALL send_char         ; Print ones digit

    POP R19
    POP R18
    RET
;=============================================================
print_newline:
; Print newline and carriage return
    PUSH R18
    LDI R18, 0x0D           ; Carriage return
    RCALL send_char
    LDI R18, 0x0A           ; Line feed
    RCALL send_char
    POP R18
    RET
;=============================================================
delay_10us:
; 10 microsecond delay using Timer0
    CLR R20
    OUT TCNT0, R20          ; Initialize timer0 with count=0
    LDI R20, 20
    OUT OCR0A, R20          ; OCR0 = 20
    LDI R20, 0b00001010
    OUT TCCR0B, R20         ; Timer0: CTC mode, prescaler 8

delay_10us_loop:
    IN R20, TIFR0
    SBRS R20, OCF0A         ; If OCF0=1, skip next instruction
    RJMP delay_10us_loop    ; Else, loop back & check OCF0 flag

    CLR R20
    OUT TCCR0B, R20         ; Stop timer0
```

```
    LDI R20, (1<<OCF0A)
    OUT TIFR0, R20        ; Clear OCF0 flag
    RET
;================================================================
delay_ms:
; Short delay (milliseconds)
    LDI R21, 10
d1: LDI R22, 50
d2: LDI R23, 50
d3: DEC R23
    BRNE d3
    DEC R22
    BRNE d2
    DEC R21
    BRNE d1
    RET
;================================================================
delay_500ms:
; Longer delay (500 milliseconds)
    LDI R21, 100
d4: LDI R22, 100
d5: LDI R23, 100
d6: DEC R23
    BRNE d6
    DEC R22
    BRNE d5
    DEC R21
    BRNE d4
    RET

;================================================================
delay_1000ms:
; Longer delay (1000 milliseconds)
    RCALL delay_500ms
    RCALL delay_500ms
    RET
;================================================================

; String constants
welcome_msg:
    .ascii "Parking System Master Started"
    .byte 0
distance_msg:
    .ascii "Distance: "
```

```
    .byte 0
raw_sensor_msg:
    .ascii "Raw sensor value: "
    .byte 0
free_msg:
    .ascii "Spot FREE"
    .byte 0
occupied_msg:
    .ascii "Spot OCCUPIED"
    .byte 0
available_msg:
    .ascii "Available spots: "
    .byte 0
sending_msg:
    .ascii "Sending status to slave: "
    .byte 0
spi_initialized_msg:
    .ascii "SPI initialized as master"
    .byte 0
spi_sending_msg:
    .ascii "SPI sending byte: "
    .byte 0
spi_complete_msg:
    .ascii "SPI transmission complete"
    .byte 0
sending_test_msg:
    .ascii "Sending test SPI commands on startup"
    .byte 0
checking_sensor_msg:
    .ascii "Checking ultrasonic sensor"
    .byte 0
separator_msg:
    .ascii "----------------------------------------"
    .byte 0
status_change_msg:
    .ascii "STATUS CHANGED!"
    .byte 0
;=============================================================
```

- Slave Arduino Code

```
#define __SFR_OFFSET 0x00
#include "avr/io.h"
;-----------------------
.global main
;=============================================================
```

```
main:
;-------------------------------------------------------------
    ; Configure PORTD for outputs (LEDs)
    LDI R17, 0b00001100   ; PD2-PD3 as outputs (for spot 1 LEDs)
    OUT DDRD, R17

    ; Initialize all LEDs - turn on green LED initially
    LDI R17, 0b00001000   ; PD3 (green LED) ON, PD2 (red LED) OFF
    OUT PORTD, R17

    ; Initialize UART for serial monitor
    RCALL uart_init

    ; Initialize R24 and R25 for status comparison
    LDI R24, 0xFF         ; Initial previous value (invalid to force first
message)
    LDI R25, 0            ; Default current value

    ; Define total parking spots
    LDI R23, 1            ; Total parking spots (currently 1)

    ; Show initial available slots (start with all free)
    LDI ZL, lo8(available_slots_msg)
    LDI ZH, hi8(available_slots_msg)
    RCALL print_string
    MOV R18, R23          ; Copy total spots to R18 (initially all
available)
    RCALL print_number    ; Print available slots count
    RCALL print_newline

    ; Enable SPI as slave
    LDI R17, (1<<SPE)
    OUT SPCR, R17
;-------------------------------------------------------------
wait_data:
    ; Store previous data for comparison
    MOV R24, R25          ; Save previous value to R24

    ; Wait for SPI data
    IN R18, SPSR
    SBRS R18, SPIF        ; Wait for byte reception
    RJMP wait_data        ; to complete

    ; Get the received data
    IN R25, SPDR          ; Get received byte from data register to R25
```

```
    ; Only update display if the data changed
    CP R25, R24
    BREQ skip_status_update

    ; Process received data
    CPI R25, 0            ; Compare with 0 (spot free)
    BREQ set_spot_free
    CPI R25, 1            ; Compare with 1 (spot occupied)
    BREQ set_spot_occupied
    RJMP wait_data        ; Invalid data, just wait for next

skip_status_update:
    ; Clear SPI interrupt flag
    IN R18, SPSR          ; Read SPSR to clear SPIF
    RJMP wait_data

;---------------------------------------------------------------
set_spot_free:
    ; Turn off red LED (PD2)
    CBI PORTD, 2
    ; Turn on green LED (PD3)
    SBI PORTD, 3

    ; Calculate available slots and display
    LDI ZL, lo8(available_slots_msg)
    LDI ZH, hi8(available_slots_msg)
    RCALL print_string
    MOV R18, R23          ; Copy total spots to R18 (all available when
free)
    RCALL print_number    ; Print available slots count
    RCALL print_newline

    ; Clear SPI interrupt flag
    IN R18, SPSR          ; Read SPSR to clear SPIF
    RJMP wait_data

;---------------------------------------------------------------
set_spot_occupied:
    ; Turn on red LED (PD2)
    SBI PORTD, 2
    ; Turn off green LED (PD3)
    CBI PORTD, 3

    ; Calculate available slots and display
```

```
    LDI ZL, lo8(available_slots_msg)
    LDI ZH, hi8(available_slots_msg)
    RCALL print_string
    CLR R18               ; 0 slots available when occupied
    RCALL print_number    ; Print available slots count
    RCALL print_newline

    ; Clear SPI interrupt flag
    IN R18, SPSR          ; Read SPSR to clear SPIF
    RJMP wait_data

;================================================================
; UART Functions
;================================================================
uart_init:
; Initialize UART for 9600 baud rate
    LDI R17, 103          ; For 9600 baud @16MHz
    STS UBRR0L, R17
    LDI R17, 0
    STS UBRR0H, R17

    LDI R17, (1<<TXEN0)   ; Enable transmitter
    STS UCSR0B, R17

    LDI R17, (1<<UCSZ01)|(1<<UCSZ00) ; 8-bit data
    STS UCSR0C, R17
    RET
;================================================================
send_char:
; Send single character in R18 to UART
    PUSH R19
wait_tx_ready:
    LDS R19, UCSR0A
    SBRS R19, UDRE0       ; Wait if buffer is full
    RJMP wait_tx_ready

    STS UDR0, R18         ; Send data
    POP R19
    RET
;================================================================
print_string:
; Print null-terminated string pointed to by Z register
    PUSH R18
print_string_loop:
    LPM R18, Z+           ; Load character from program memory
```

```
    CPI R18, 0            ; Check for null terminator
    BREQ print_string_done
    RCALL send_char       ; Send character
    RJMP print_string_loop
print_string_done:
    POP R18
    RET
;============================================================
print_number:
; Print number in R18 (0-255)
    PUSH R18
    PUSH R19

    CPI R18, 100          ; Check if number >= 100
    BRLO tens_digit       ; If less than 100, skip to tens digit

    ; Print hundreds digit
    LDI R19, '0'          ; Start with '0'
hundreds_loop:
    CPI R18, 100          ; Check if >= 100
    BRLO hundreds_done    ; If < 100, done with hundreds
    SUBI R18, 100         ; Subtract 100
    INC R19               ; Increment digit
    RJMP hundreds_loop    ; Continue loop
hundreds_done:
    PUSH R18              ; Save remainder
    MOV R18, R19          ; Move hundreds digit to R18
    RCALL send_char       ; Print hundreds digit
    POP R18               ; Restore remainder

tens_digit:
    CPI R18, 10           ; Check if number >= 10
    BRLO ones_digit       ; If less than 10, skip to ones digit

    ; Print tens digit
    LDI R19, '0'          ; Start with '0'
tens_loop:
    CPI R18, 10           ; Check if >= 10
    BRLO tens_done        ; If < 10, done with tens
    SUBI R18, 10          ; Subtract 10
    INC R19               ; Increment digit
    RJMP tens_loop        ; Continue loop
tens_done:
    PUSH R18              ; Save remainder
    MOV R18, R19          ; Move tens digit to R18
```

```
    RCALL send_char       ; Print tens digit
    POP R18               ; Restore remainder

ones_digit:
    ; Print ones digit
    SUBI R18, -'0'        ; Convert to ASCII
    RCALL send_char       ; Print ones digit

    POP R19
    POP R18
    RET
;===================================================================
print_newline:
; Print newline and carriage return
    PUSH R18
    LDI R18, 0x0D         ; Carriage return
    RCALL send_char
    LDI R18, 0x0A         ; Line feed
    RCALL send_char
    POP R18
    RET
;===================================================================

; String constants
welcome_msg:
    .ascii "Parking Spot Monitor Slave Started"
    .byte 0
received_msg:
    .ascii "Received data from master: "
    .byte 0
free_msg:
    .ascii "Spot status: FREE (Green LED ON)"
    .byte 0
occupied_msg:
    .ascii "Spot status: OCCUPIED (Red LED ON)"
    .byte 0
invalid_msg:
    .ascii "Invalid data received"
    .byte 0
pins_config_msg:
    .ascii "DDRD configuration: "
    .byte 0
init_led_msg:
    .ascii "Initial LED state (PORTD): "
    .byte 0
```

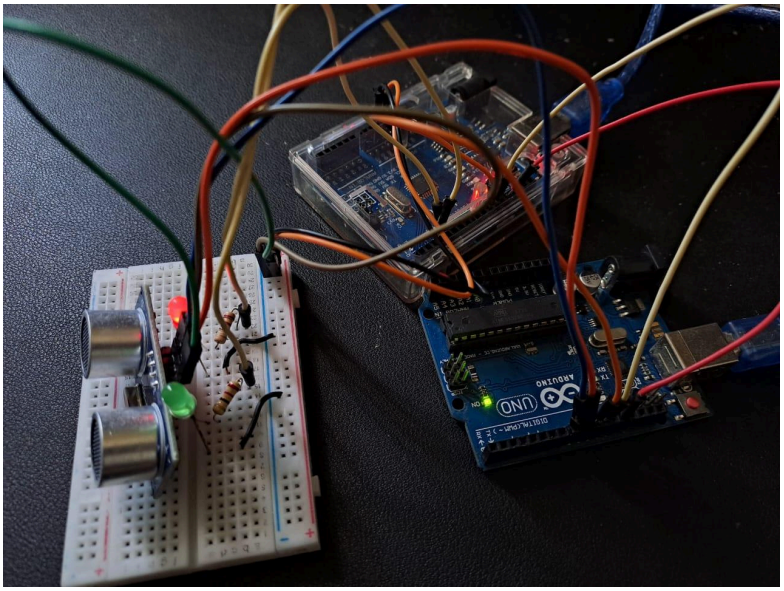
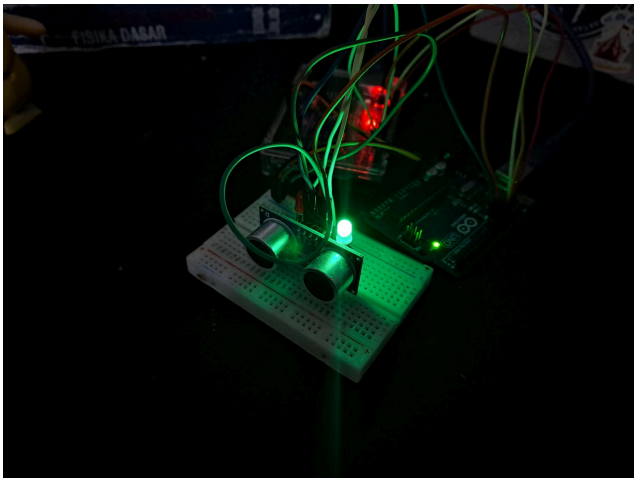
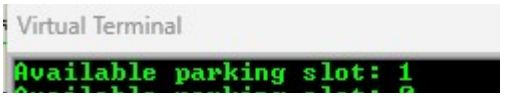
```
new_led_msg:
    .ascii "Updated LED state (PORTD): "
    .byte 0
setting_free_msg:
    .ascii "Setting LEDs for FREE state (Red OFF, Green ON)"
    .byte 0
setting_occupied_msg:
    .ascii "Setting LEDs for OCCUPIED state (Red ON, Green OFF)"
    .byte 0
waiting_msg:
    .ascii "Waiting for data from master..."
    .byte 0
available_slots_msg:
    .ascii "Available parking slot: "
    .byte 0
;=============================================================
```

3.2 RESULT

Hasil dari pengujian membuktikan bahwa sistem dapat bekerja sesuai fungsinya pada kondisi-kondisi tertentu. Sistem dapat menghasilkan output yang sesuai ketika jarak objek

dekat atau jauh dari sensor. LED berfungsi sebagai indikator dapat mati ataupun menyala secara otomatis berdasarkan input dari sensor jarak HC-SR04.

Ketika objek terdeteksi berada dalam jarak kurang dari 30 cm, LED merah akan menyala menandakan bahwa slot parkir masih kosong dan ketika objek terdeteksi berada dalam jarak lebih dari 30 cm, LED hijau akan menyala dan satu buzzer akan aktif. Berdasarkan pengujian yang dilakukan, sistem *Parking Lot Sensor* memenuhi kriteria yang harus dipenuhi. Karena memenuhi kriteria, maka sistem *Parking Lot Sensor* dapat membantu pengemudi mobil dalam mengetahui slot parkir yang tersedia dalam sebuah tempat sehingga para pengendara dapat menghemat waktu tanpa harus berkeliling area parkir sehingga lebih efisien dan terorganisir.

Foto Rangkaian

Hasil
 

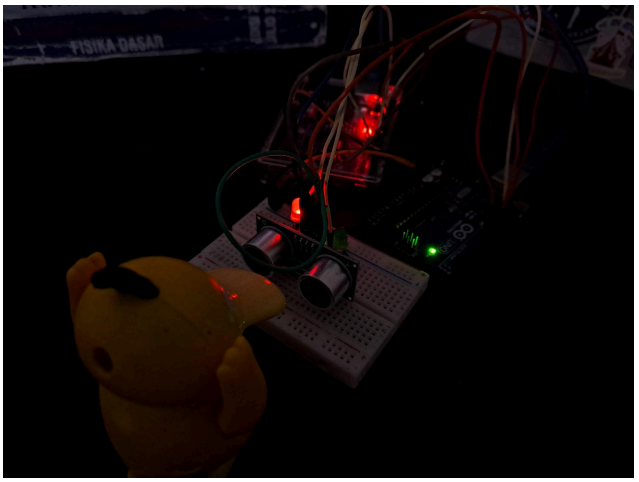
	Available parking slot: 0

Table 2. Testing Result

3.3 EVALUATION

Berdasarkan hasil percobaan, sistem parkir otomatis berbasis Arduino dengan komunikasi master-slave yang kami kembangkan mampu menjalankan fungsinya dengan cukup baik. Sensor ultrasonik HC-SR04 yang terpasang pada board master dapat membaca jarak kendaraan secara periodik, kemudian hasilnya dikirim melalui SPI ke board slave untuk ditampilkan. Proses komunikasi antar board berjalan lancar dan status parkir dapat ditentukan secara otomatis tanpa intervensi pengguna karena sistem aktif begitu diberi daya.

Selama pengujian, sistem mampu merespons perubahan jarak dengan memberikan output berupa indikator LED atau tampilan yang sesuai. Namun, terdapat beberapa kendala yang menyebabkan performa tidak selalu konsisten. Beberapa di antaranya adalah delay pada pembacaan sensor dan tampilan LED yang terkadang terlambat menyala. Hal ini kemungkinan disebabkan oleh koneksi kabel yang kurang stabil, noise pada sinyal ultrasonik, serta pembacaan sensor yang sensitif terhadap fluktuasi jarak kecil.

CHAPTER 4

CONCLUSION

Perangkat “Parking Lot Sensor” cocok untuk digunakan agar pengemudi dapat dengan mengetahui slot parkir yang tersedia dalam sebuah tempat sehingga para pengendara dapat menghemat waktu tanpa harus berkeliling area parkir sehingga lebih efisien dan terorganisir.

Sensor HC-SR04 digunakan untuk mengukur jarak objek dari sensor menggunakan gelombang ultrasonik. Dalam proyek ini, sensor HC-SR04 berfungsi untuk mendeteksi keberadaan kendaraan. Terdapat 2 buah kondisi dimana apakah jarak mobil dan sensor lebih dari 30 cm atau kurang dari 30 cm. Jika jarak lebih dari 30 cm maka device akan menyalakan led hijau untuk menandakan bahwa slot parkir masih tersedia, sementara jika jarak mobil dan sensor kurang dari 30 cm, maka device akan menyalakan led merah untuk menandakan bahwa slot parkir sudah tidak tersedia.

Link Repository Github:

https://github.com/wiellona/Parking-Lot-Sensor

REFERENCES

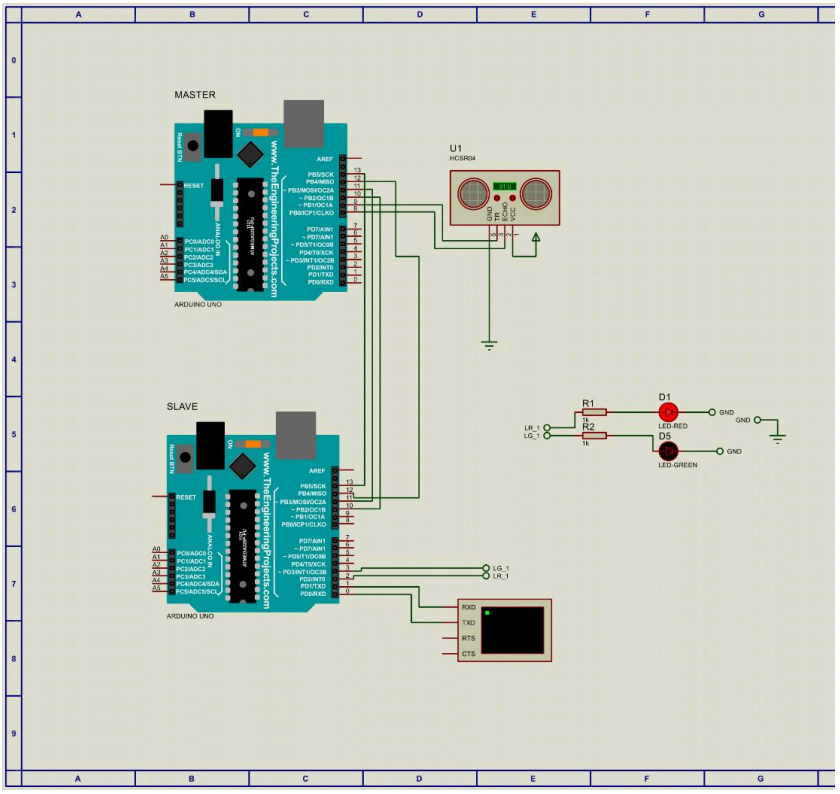
[1] Digilab, In MODUL 1-9, Lab Module. Universitas Indonesia, 2024.

[2] A. E. Dwiputra, H. Khoswanto, R. Sutjiadi, and R. Lim, "IoT-Based Car's Parking Monitoring System," MATEC Web of Conferences, vol. 164, p. 01002, 2018, doi: https://doi.org/10.1051/matecconf/201816401002.

[3] H. Purwanto and B. Prasetio, "Microcontroller Based Parking Lot Monitoring System Prototype," International Journal of Research and Applied Technology, vol. 2, no. 1, pp. 132–141, Mar. 2022, doi: https://doi.org/10.34010/injuratech.v2i1.6742.

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation