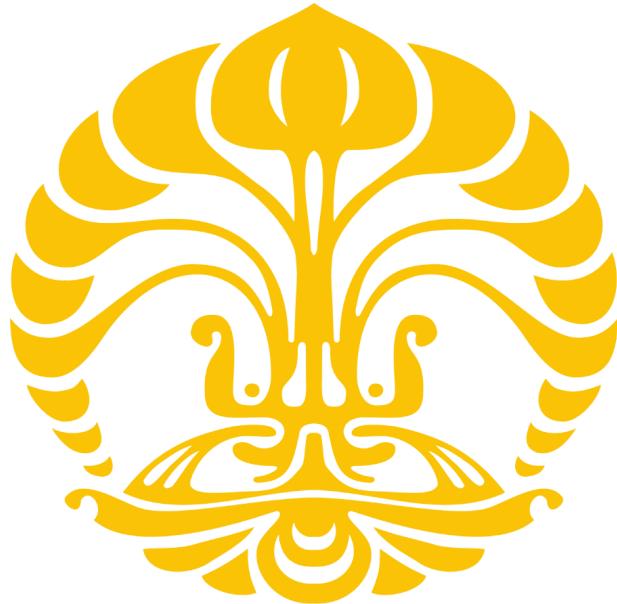


# **LAPORAN PROYEK AKHIR**

## **PRAKTIKUM SISTEM EMBEDDED**

**Air Quality Control**



**Disusun oleh:**

**Kelompok 16**

|                            |            |
|----------------------------|------------|
| Jonathan Frederick Kosasih | 2306225981 |
| Muhammad Rey Kafaka Fadlan | 2306250573 |
| Muhammad Rafli             | 2306250730 |
| Raddie Ezra Satrio Andaru  | 2306250693 |

**FAKULTAS TEKNIK**  
**PROGRAM STUDI TEKNIK KOMPUTER**  
**UNIVERSITAS INDONESIA**

**2025**

## KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga kami, kelompok 16, berhasil menyelesaikan tugas akhir Praktikum Sistem Siber Fisik dengan judul “Air Quality Control”.

Seiring dengan kemajuan zaman, teknologi terus berkembang dan telah menjadi bagian tak terpisahkan dari kehidupan sehari-hari. Mulai dari saat kita bangun pagi hingga tidur di malam hari, kehadiran teknologi selalu menyertai kita. Melalui proyek ini, kami berharap dapat mengaplikasikan modul-modul yang telah kami pelajari sebelumnya, sekaligus memberikan manfaat bagi masyarakat sekitar.

Kami, kelompok 16, mengucapkan terima kasih yang sebesar-besarnya kepada seluruh anggota kelompok atas kerja sama yang solid dalam mencari ide, merancang rangkaian, menulis kode, serta menyusun laporan. Berkat kerja sama tersebut, proyek ini dapat diselesaikan tepat waktu dengan hasil yang memuaskan. Kami juga menyampaikan apresiasi kepada asisten laboratorium yang telah memberikan bimbingan selama praktikum berlangsung. Tak lupa, kami ucapkan terima kasih kepada Ivan Yuantama Pradipta atas saran dan masukan yang sangat berguna bagi pengembangan proyek kami.

Depok, 18 Mei 2025

Kelompok 16

## DAFTAR ISI

|  |           |
|--|-----------|
| <b>KATA PENGANTAR.....</b>                             | <b>1</b>  |
| <b>DAFTAR ISI.....</b>                                 | <b>2</b>  |
| <b>BAB 1.....</b>                                      | <b>3</b>  |
| <b>PENDAHULUAN.....</b>                                | <b>3</b>  |
| 1.1 PERNYATAAN MASALAH.....                            | 3         |
| 1.2 SOLUSI YANG DIUSULKAN.....                         | 3         |
| 1.3 KRITERIA.....                                      | 4         |
| 1.4 PERAN ANGGOTA.....                                 | 4         |
| <b>BAB 2.....</b>                                      | <b>6</b>  |
| <b>IMPLEMENTASI.....</b>                               | <b>6</b>  |
| 2.1 DESAIN PERANGKAT KERAS DAN SKEMATIK RANGKAIAN..... | 6         |
| 2.2 PENGEMBANGAN SOFTWARE.....                         | 7         |
| 2.3 INTEGRASI HARDWARE DENGAN SOFTWARE.....            | 41        |
| <b>BAB 3.....</b>                                      | <b>42</b> |
| <b>UJI COBA DAN EVALUASI.....</b>                      | <b>42</b> |
| 3.1 UJI COBA RANGKAIAN.....                            | 42        |
| 3.2 HASIL UJI COBA.....                                | 46        |
| 3.3 EVALUASI.....                                      | 46        |
| <b>BAB 4.....</b>                                      | <b>48</b> |
| <b>KESIMPULAN.....</b>                                 | <b>48</b> |
| <b>REFERENSI.....</b>                                  | <b>49</b> |

## BAB 1

### PENDAHULUAN

#### 1.1 PERNYATAAN MASALAH

Di era modern saat ini, kualitas udara menjadi salah satu faktor penting yang memengaruhi kesehatan dan kenyamanan manusia, terutama di lingkungan perkotaan dan area industri. Namun, banyak lokasi yang masih mengalami pencemaran udara akibat polutan seperti partikel debu, gas berbahaya, dan bahan kimia, yang dapat menimbulkan risiko kesehatan serius bagi masyarakat.

Kurangnya sistem monitoring dan pengendalian kualitas udara secara real-time menyebabkan keterlambatan dalam mendeteksi perubahan kondisi udara yang berbahaya. Hal ini mengakibatkan tindakan pencegahan yang kurang efektif serta potensi dampak negatif jangka panjang terhadap kesehatan dan lingkungan.

Oleh karena itu, dibutuhkan sebuah sistem Air Quality Control yang mampu memantau kondisi udara secara kontinu dengan sensor yang sensitif, mengolah data secara akurat, dan memberikan peringatan serta kendali otomatis untuk menjaga kualitas udara tetap dalam batas aman. Sistem ini diharapkan dapat digunakan di berbagai lingkungan seperti rumah, kantor, pabrik, atau area publik untuk meningkatkan kualitas hidup dan mengurangi risiko pencemaran udara.

#### 1.2 SOLUSI YANG DIUSULKAN

Untuk mengatasi masalah ini, solusi yang kelompok 18 usulkan adalah dengan membuat sebuah proyek seperti “*Air Quality Control*” yang akan membantu masyarakat untuk memantau kualitas udara pada lingkungan di sekitar dengan akurat. Alat ini juga akan menggunakan Arduino dan akan menggunakan *assembly* sebagai bahasa pemrograman yang kami gunakan. Alat ini juga dilengkapi dengan sensor, seperti sensor kelembapan DHT11 untuk mengukur tingkat kelembapan, serta sensor dari MQ-series yang berfungsi untuk mendeteksi parameter kualitas udara. Seluruh data pengukuran akan ditampilkan secara real-time pada layar LCD yang telah terintegrasi dengan sistem. Selanjutnya, alat ini akan memberikan peringatan kepada pengguna melalui buzzer atau LED apabila kualitas udara terdeteksi melebihi ambang batas yang telah ditetapkan. Dengan adanya sistem peringatan ini, masyarakat dapat segera mengambil langkah yang diperlukan untuk menjaga kualitas

udara di sekitarnya. Data yang ditampilkan dari hasil pemantauan kualitas udara dapat membantu masyarakat dalam menyesuaikan kebiasaan sehari-hari mereka. Misalnya, dengan menghindari aktivitas luar ruangan saat polusi tinggi, memperbaiki sistem ventilasi di dalam rumah, atau mengambil tindakan untuk mengurangi sumber polusi di lingkungan sekitar.

Alat ini sangat dibutuhkan di berbagai wilayah, khususnya di kota-kota besar di seluruh dunia yang memiliki tingkat pencemaran udara tinggi. Dengan demikian, masyarakat perkotaan dapat memantau dan mengetahui kondisi kualitas udara di lingkungan tempat tinggal mereka.

Melalui solusi ini, kami berharap perangkat yang dikembangkan dapat memberikan dampak positif bagi para pengguna, khususnya dalam hal pemantauan kualitas udara di sekitar mereka. Dengan demikian, masyarakat dapat mengambil langkah preventif terhadap paparan polusi udara atau melakukan upaya untuk menguranginya. Berkat biaya produksi yang terjangkau dan pemanfaatan teknologi Arduino yang populer, kami juga berharap alat ini dapat digunakan secara luas dan memberikan manfaat yang besar bagi banyak orang.

### **1.3 KRITERIA**

Air Quality Control dapat berjalan bila memenuhi syarat berikut:

1. Sensor DHT11 dan MQ-2 mampu mengukur suhu, kelembapan, serta kandungan gas di lingkungan sekitar. Data hasil pengukuran tersebut kemudian diproses oleh Arduino dan ditampilkan pada serial monitor. Proses pengiriman data dan sinyal ini menggunakan protokol komunikasi agar informasi dapat dibaca dan dikirim secara efektif. Suhu, kelembapan dan data gas ditampilkan melalui serial monitor.
2. Buzzer akan aktif ketika kadar gas di lingkungan melebihi batas aman yang telah ditentukan dalam kode program.

### **1.4 PERAN ANGGOTA**

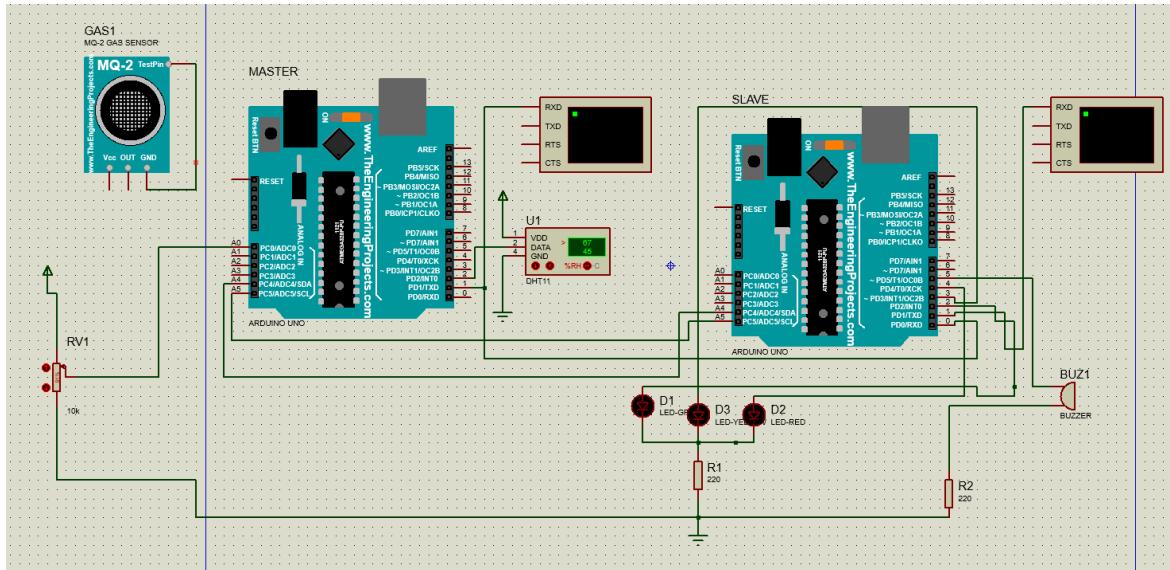
Peran dan tanggung jawab yang diberikan kepada anggota kelompok adalah sebagai berikut:

| <b>Anggota</b>             | <b>Peran</b>  | <b>Tanggung Jawab</b>  |
|----------------------------|---|--|
| Jonathan Frederick Kosasih | Menyusun Laporan,<br>Membuat desain<br>rangkaian dan kode | Membuat dan mengedit<br>laporan serta membuat<br>kode master |
| Muhammad Rey Kafaka Fadlan | Menyusun Laporan,<br>Membuat desain<br>rangkaian dan kode | Membuat dan mengedit<br>laporan serta membuat<br>kode master |
| Muhammad Rafli             | Menyusun Laporan,<br>Membuat desain<br>rangkaian dan kode | Membuat dan mengedit<br>laporan serta membuat<br>kode slave  |
| Raddie Ezra Satrio Andaru  | Menyusun Laporan,<br>Membuat desain<br>rangkaian dan kode | Membuat dan mengedit<br>laporan serta membuat<br>kode slave  |

## BAB 2

### IMPLEMENTASI

#### 2.1 DESAIN PERANGKAT KERAS DAN SKEMATIK RANGKAIAN



Gambar 1. Hasil Rangkaian

Beberapa komponen yang digunakan antara lain MQ-2 Gas Sensor, Arduino Uno, DHT11, LED, buzzer, serta kabel jumper. Berikut adalah penjelasan dari masing-masing komponen tersebut:

a. Arduino Uno

Board mikrokontroler yang menjadi pusat dan otak dari seluruh rangkaian. Terdapat 2 board pada rangkaian ini, satu digunakan sebagai master dan satu lainnya digunakan sebagai slave. Master akan menerima input dari sensor DHT11 dan MQ-2 untuk di-passing ke slave, sedangkan slave akan menerima data dari master lalu mengeluarkan output pada *serial monitor* dan sinyal untuk LED dan buzzer berdasarkan nilai dari input.

b. MQ-2 Gas Sensor

Sensor gas yang akan mendeteksi jenis gas, seperti Metana ( $\text{CH}_4$ ), Propana ( $\text{C}_3\text{H}_8$ ), Karbon Monoksida (CO), Butana ( $\text{C}_4\text{H}_{10}$ ), Hidrogen ( $\text{H}_2$ ), dan lain sebagainya. MQ-2 mempunyai dua pin utama yaitu VCC sebagai

sumber tegangan positif dan analog output voltage yang dapat memberikan output yang berkaitan dengan konsentrasi gas yang terdeteksi.

c. DHT11

Sensor digital yang digunakan untuk mengukur suhu dan kelembapan udara di lingkungan. Input yang diperoleh merupakan sinyal digital yang akan *di-passing* dari master board ke slave board.

d. LED

Komponen elektronika terbuat dari bahan semikonduktor yang akan mengeluarkan cahaya ketika dialiri arus listrik dengan cara mengkonversi energi listrik menjadi cahaya. Rangkaian ini menggunakan 3 warna LED untuk 3 tingkatan kualitas udara: hijau untuk kualitas baik, kuning untuk kualitas sedang, merah untuk kualitas buruk.

e. Buzzer

Komponen elektronika yang akan menghasilkan getaran dan suara ketika diberikan tegangan listrik. Buzzer pada rangkaian ini digunakan sebagai indikator kualitas udara buruk untuk memberikan peringatan dalam bentuk suara kepada pengguna.

f. Kabel Jumper

Kabel ini berfungsi untuk menghubungkan berbagai komponen, baik di breadboard maupun secara langsung. Ujung kabel tersebut akan mengkoneksikan pin atau port pada masing-masing komponen.

## 2.2 PENGEMBANGAN SOFTWARE

Source Code Master:

```
#define __SFR_OFFSET 0x00
#include "avr/io.h"

; Constants for sensor pins and thresholds
.set MQ2_PIN, 0          ; A0 = ADC0
.set DHT_PIN, 2           ; Digital pin 2
.set DHT_TYPE, 11         ; DHT11 type
.set AQ_GOOD, 341         ; Good air quality threshold (0-341)
.set AQ_MODERATE, 682     ; Moderate threshold (342-682)
```

```

; Improved far call macro using indirect jump with Z register
.macro CALL_FAR target
    push r16
    ldi r16, lo8(99f)
    push r16
    ldi r16, hi8(99f)
    push r16
    ldi ZL, lo8(\target)
    ldi ZH, hi8(\target)
    ijmp
99:
    pop r16
.endm

; Variables in data segment
.section .data
temperature:    .space 2
humidity:       .space 2
airQuality:     .space 2
airQualityStatus: .space 1
tempBuffer:      .space 2
humBuffer:       .space 2
rawValue:        .space 2

; UTILITY FUNCTIONS
uart_send:
uart_wait:
    lds r17, UCSR0A
    sbrs r17, UDRE0
    rjmp uart_wait
    sts UDR0, r16
    ret

delay_us:
delay_us_loop:
    nop
    nop

```

```
    nop
    dec r24
    brne delay_us_loop
    ret

delay_ms:
    push r18
    push r19
    push r20
    mov r18, r24
    or r18, r25
    breq delay_ms_done

delay_ms_loop:
    ldi r18, 20

delay_ms_loop1:
    ldi r19, 200

delay_ms_loop2:
    ldi r20, 3

delay_ms_loop3:
    dec r20
    brne delay_ms_loop3
    dec r19
    brne delay_ms_loop2
    dec r18
    brne delay_ms_loop1

    subi r24, 1
    sbci r25, 0

    brne delay_ms_loop
    tst r24
    brne delay_ms_loop

delay_ms_done:
```

```
pop r20
pop r19
pop r18
ret

.section .text

; Interrupt vector table
.org 0x0000
jmp main

; Main program
.global main
main:
    ; Stack initialization
    ldi r16, hi8(RAMEND)
    out SPH, r16
    ldi r16, lo8(RAMEND)
    out SPL, r16

    ; System initialization
    CALL_FAR init_serial
    CALL_FAR init_i2c
    CALL_FAR init_dht

    ; Initial delay
    ldi r24, lo8(2000)
    ldi r25, hi8(2000)
    CALL_FAR delay_ms

    ; Print startup message
    ldi ZL, lo8(init_message)
    ldi ZH, hi8(init_message)
    CALL_FAR print_string

; Main program loop
loop:
    CALL_FAR read_dht11
```

```

CALL_FAR read_mq2
CALL_FAR determine_air_quality
CALL_FAR send_data_to_slave
CALL_FAR print_data

; Wait between readings
ldi r24, lo8(2000)
ldi r25, hi8(2000)
CALL_FAR delay_ms

rjmp loop

; Initialize UART (9600 baud at 16MHz)
init_serial:
    ldi r16, 103
    sts UBRR0H, r1
    sts UBRR0L, r16

    ldi r16, (1<<TXEN0) | (1<<RXEN0)
    sts UCSR0B, r16

    ldi r16, (1<<UCSZ01) | (1<<UCSZ00)
    sts UCSR0C, r16
    ret

; Initialize I2C as master (100kHz at 16MHz)
init_i2c:
    ldi r16, 72
    sts TWBR, r16

    ldi r16, 0
    sts TWSR, r16

    ldi r16, (1<<TWEN)
    sts TWCR, r16
    ret

; Initialize DHT11 sensor

```

```

init_dht:
    sbi DDRD, DHT_PIN
    sbi PORTD, DHT_PIN
    ret

; Print null-terminated string from flash
print_string:
print_string_loop:
    lpm r16, Z+
    cpi r16, 0
    breq ps_done

    push ZL
    push ZH
    CALL_FAR uart_send
    pop ZH
    pop ZL
    rjmp print_string_loop

ps_done:
    ret

; Print 16-bit decimal value
print_decimal:
    subi r16, -'0'
    CALL_FAR uart_send
    ret

; Read temperature and humidity from DHT11
read_dht11:
    ; DHT11 start signal
    cbi PORTD, DHT_PIN
    ldi r24, lo8(20)
    ldi r25, hi8(20)
    CALL_FAR delay_ms

    sbi PORTD, DHT_PIN
    ldi r24, 40

```

```

CALL_FAR delay_us

cbi DDRD, DHT_PIN

; Set dummy values for demonstration
ldi r16, 50          ; Humidity = 50%
ldi r17, 0
sts humidity, r16
sts humidity+1, r17

ldi r16, 25          ; Temperature = 25°C
ldi r17, 0
sts temperature, r16
sts temperature+1, r17

ret

; Read MQ-2 sensor value
read_mq2:
    ; ADC setup and conversion
    ldi r16, (1<<REFS0)
    sts ADMUX, r16

    ldi r16, (1<<ADEN) | (1<<ADSC) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0)
    sts ADCSRA, r16

adc_wait:
    lds r16, ADCSRA
    sbrc r16, ADSC
    rjmp adc_wait

; Read and store ADC value
lds r16, ADCL
lds r17, ADCH

sts rawValue, r16
sts rawValue+1, r17

```

```

sts airQuality, r16
sts airQuality+1, r17

ret

; Determine air quality status based on thresholds
determine_air_quality:
    lds r16, airQuality
    lds r17, airQuality+1

    cpi r17, hi8(AQ_GOOD)
    brlo determine_good_air
    brne determine_moderate_check

    cpi r16, lo8(AQ_GOOD)
    brlo determine_good_air

determine_moderate_check:
    cpi r17, hi8(AQ_MODERATE)
    brlo determine_moderate_air
    brne determine_poor_air

    cpi r16, lo8(AQ_MODERATE)
    brlo determine_moderate_air

determine_poor_air:
    ldi r16, 2          ; Status 2 = Poor
    sts airQualityStatus, r16
    ret

determine_good_air:
    ldi r16, 0          ; Status 0 = Good
    sts airQualityStatus, r16
    ret

determine_moderate_air:
    ldi r16, 1          ; Status 1 = Moderate
    sts airQualityStatus, r16

```

```
ret

; Strings in program memory
init_message:
    .ascii "Master Arduino Initialized\r\n"
    .ascii "Monitoring with DHT11 and MQ-2 Gas Sensor\r\n"
    .byte 0

data_header:
    .byte 13, 10
    .ascii "----- Sensor Readings ----- \r\n"
    .byte 0

temp_str:
    .ascii "Temperature (DHT11): "
    .byte 0

celsius_str:
    .ascii " C\r\n"
    .byte 0

hum_str:
    .ascii "Humidity (DHT11): "
    .byte 0

percent_str:
    .ascii " %\r\n"
    .byte 0

air_str:
    .ascii "Air Quality (MQ-2): "
    .byte 0

raw_str:
    .ascii " Raw Value\r\n"
    .byte 0

status_str:
```

```

.ascii "Air Quality Status: "
.byte 0

good_str:
.ascii "GOOD (Green) - Range 0-340\r\n"
.byte 0

moderate_str:
.ascii "MODERATE (Yellow) - Range 341-682\r\n"
.byte 0

poor_str:
.ascii "POOR (Red) - Range 683-1023\r\n"
.byte 0

; Send sensor data to slave Arduino via I2C
send_data_to_slave:
ldi r24, 50
CALL_FAR delay_ms

; Start I2C transmission
ldi r16, (1<<TWINT) | (1<<TWSTA) | (1<<TWEN)
sts TWCR, r16

wait_start:
lds r16, TWCR
sbrs r16, TWINT
rjmp wait_start

; Send slave address (8) with write bit (0)
ldi r16, (8<<1) | 0
sts TWDR, r16

ldi r16, (1<<TWINT) | (1<<TWEN)
sts TWCR, r16

wait_addr:
lds r16, TWCR

```

```

sbrs r16, TWINT
rjmp wait_addr

; Send temperature data
lds r16, temperature+1
sts TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)
sts TWCR, r16

wait_temp_h:
lds r16, TWCR
sbrs r16, TWINT
rjmp wait_temp_h

lds r16, temperature
sts TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)
sts TWCR, r16

wait_temp_l:
lds r16, TWCR
sbrs r16, TWINT
rjmp wait_temp_l

; Send humidity data
lds r16, humidity+1
sts TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)
sts TWCR, r16

wait_hum_h:
lds r16, TWCR
sbrs r16, TWINT
rjmp wait_hum_h

lds r16, humidity
sts TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)

```

```

sts TWCR, r16

wait_hum_l:
lds r16, TWCR
sbrs r16, TWINT
rjmp wait_hum_l

; Send air quality data
lds r16, airQuality+1
sts TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)
sts TWCR, r16

wait_aq_h:
lds r16, TWCR
sbrs r16, TWINT
rjmp wait_aq_h

lds r16, airQuality
sts TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)
sts TWCR, r16

wait_aq_l:
lds r16, TWCR
sbrs r16, TWINT
rjmp wait_aq_l

; Send air quality status
lds r16, airQualityStatus
sts TWDR, r16
ldi r16, (1<<TWINT) | (1<<TWEN)
sts TWCR, r16

wait_status:
lds r16, TWCR
sbrs r16, TWINT
rjmp wait_status

```

```

; Send stop condition
ldi r16, (1<<TWINT) | (1<<TWEN) | (1<<TWSTO)
sts TWCR, r16

ldi r24, 50
CALL_FAR delay_ms
ret

; Print all sensor data to serial
print_data:
    ldi ZL, lo8(data_header)
    ldi ZH, hi8(data_header)
    CALL_FAR print_string

    ; Print temperature
    ldi ZL, lo8(temp_str)
    ldi ZH, hi8(temp_str)
    CALL_FAR print_string

    lds r16, temperature
    lds r17, temperature+1
    CALL_FAR print_decimal

    ldi ZL, lo8(celsius_str)
    ldi ZH, hi8(celsius_str)
    CALL_FAR print_string

    ; Print humidity
    ldi ZL, lo8(hum_str)
    ldi ZH, hi8(hum_str)
    CALL_FAR print_string

    lds r16, humidity
    lds r17, humidity+1
    CALL_FAR print_decimal

    ldi ZL, lo8(percent_str)

```

```

ldi ZH, hi8(percent_str)
CALL_FAR print_string

; Print air quality
ldi ZL, lo8(air_str)
ldi ZH, hi8(air_str)
CALL_FAR print_string

lds r16, airQuality
lds r17, airQuality+1
CALL_FAR print_decimal

ldi ZL, lo8(raw_str)
ldi ZH, hi8(raw_str)
CALL_FAR print_string

; Print air quality status with appropriate message
ldi ZL, lo8(status_str)
ldi ZH, hi8(status_str)
CALL_FAR print_string

lds r16, airQualityStatus
cpi r16, 0
brne check_moderate

ldi ZL, lo8(good_str)
ldi ZH, hi8(good_str)
CALL_FAR print_string
ret

check_moderate:
cpi r16, 1
brne print_poor

ldi ZL, lo8(moderate_str)
ldi ZH, hi8(moderate_str)
CALL_FAR print_string
ret

```

```
print_poor:  
    ldi ZL, lo8(poor_str)  
    ldi ZH, hi8(poor_str)  
    CALL_FAR print_string  
    ret
```

Kode assembly yang disajikan merupakan program untuk mikrokontroler berbasis arsitektur AVR, yang ditulis menggunakan bahasa assembly. Program tersebut mengandung beberapa fungsi utama sebagai berikut:

- **uart\_send:** Fungsi ini mengirim satu byte data ke UART. Ia memeriksa register status UART (UCSR0A) hingga buffer transmisi kosong (bit UDRE0 diset). Setelah buffer siap, data di-register r16 dikirim ke UDR0 untuk transmisi serial.
- **delay\_us:** Delay mikrodetik menggunakan loop pengurangan register r24 dan instruksi nop untuk menunda eksekusi. Fungsi ini digunakan untuk delay pendek yang presisi, seperti sinyal waktu kritis dalam komunikasi sensor digital seperti DHT11.
- **delay\_ms:** Membuat delay dalam milidetik dengan nested loop yang menghitung waktu lebih lama dari delay\_us. Register r24 dan r25 menyimpan nilai delay, kemudian fungsi mengulang loop bersarang untuk menghasilkan waktu tunggu yang sesuai kebutuhan sistem.
- **main:** Fungsi utama program yang melakukan inisialisasi stack, serial UART, I2C, dan sensor DHT11. Setelah inisialisasi, program masuk ke loop tak berujung membaca data sensor, menentukan kualitas udara, mengirim data ke slave I2C, dan mencetak hasil pembacaan ke serial monitor.
- **init\_serial:** Mengatur baud rate UART menjadi 9600 pada frekuensi 16 MHz, mengaktifkan transmitter dan receiver, serta mengatur format frame data 8 bit tanpa parity dan 1 stop bit, sehingga komunikasi serial dapat dilakukan dengan perangkat lain.
- **init\_i2c:** Menginisialisasi I2C sebagai master dengan kecepatan 100 kHz. Register TWBR di-set, prescaler TWSR diset nol, dan modul TWI diaktifkan dengan TWEN. Fungsi ini mempersiapkan mikrokontroler untuk komunikasi I2C dengan perangkat slave.

- **init\_dht:** Mengatur pin DHT11 sebagai output dan mengatur pin ke kondisi HIGH. Persiapan ini diperlukan agar mikrokontroler dapat mengirim sinyal start ke sensor DHT11 sebelum membaca data suhu dan kelembapan.
- **print\_string:** Mencetak string null-terminated dari memori program (flash) ke serial monitor. Fungsi membaca tiap karakter menggunakan instruksi lpm dari register Z, lalu mengirim karakter via uart\_send sampai menemukan karakter null sebagai tanda akhir string.
- **print\_decimal:** Mengirim karakter angka ASCII satu digit yang disimpan di r16 ke UART. Fungsi ini hanya mengirim satu byte, tanpa mengonversi nilai numerik menjadi string desimal lengkap, sehingga lebih cocok untuk debugging nilai kecil.
- **read\_dht11:** Melakukan sinyal start ke sensor DHT11 dengan menurunkan dan menaikkan pin DHT\_PIN, lalu mengkonfigurasi pin sebagai input untuk membaca data. Pada contoh ini, nilai dummy suhu 25°C dan kelembapan 50% langsung disimpan ke memori sebagai simulasi pembacaan sensor.
- **read\_mq2:** Menginisialisasi ADC untuk membaca nilai analog dari pin MQ2\_PIN (ADC0). Mengaktifkan ADC, memulai konversi, menunggu sampai selesai, lalu membaca nilai ADC 10-bit dan menyimpan hasilnya sebagai nilai kualitas udara dalam variabel airQuality.
- **determine\_air\_quality:** Membandingkan nilai airQuality dengan ambang batas AQ\_GOOD dan AQ\_MODERATE. Berdasarkan hasil perbandingan, status kualitas udara disimpan sebagai 0 (save), 1 (warning), atau 2 (danger) dalam airQualityStatus untuk dikendalikan LED dan buzzer.
- **send\_data\_to\_slave:** Mengirim data suhu, kelembapan, nilai analog kualitas udara, dan status kualitas udara ke perangkat slave melalui protokol I2C. Fungsi mengatur kondisi start, alamat slave, mengirim byte data secara berurutan, lalu mengakhiri dengan kondisi stop.
- **print\_data:** Menampilkan seluruh data sensor dengan label yang jelas ke serial monitor. Fungsi mencetak header, suhu, kelembapan, nilai kualitas udara, dan status dengan pesan deskriptif, sehingga memudahkan pemantauan kondisi lingkungan secara real-time.

Source Code Slave:

```

#define __SFR_OFFSET 0
#include <avr/io.h>

; Define constants
.equ GREEN_LED_PIN, 2
.equ YELLOW_LED_PIN, 3
.equ RED_LED_PIN, 4
.equ BUZZER_PIN, 5

.equ I2C_SLAVE_ADDR, 8

; SRAM variables
.section .data
temperature: .space 2
humidity: .space 2
airQuality: .space 2
airQualityStatus: .space 1
independentTest: .space 1
buffer: .space 10

; Code section
.section .text

; --- Reset Vector ---
.org 0x0000
rjmp main

; --- TWI (I2C) Interrupt Vector ---
.org 0x0028 ; For ATmega328p, TWI interrupt vector address is
0x28 (decimal 40)
rjmp TWI_vect

; Default interrupt handler (just reti)
.org 0x0040
default_interrupt:
reti

; --- Main program ---

```

```

.global main
main:
    ; Initialize stack pointer
    ldi r16, hi8(RAMEND)
    out SPH, r16
    ldi r16, lo8(RAMEND)
    out SPL, r16

    ; Clear independentTest flag
    ldi r16, 0
    sts independentTest, r16

    ; Initialize UART 9600 baud for debug
    rcall uart_init

    ; Initialize I2C slave
    rcall i2c_init

    ; Configure LED and buzzer pins as outputs
    ldi r16,
    (1<<GREEN_LED_PIN) | (1<<YELLOW_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_PIN)
    out DDRD, r16

    ; Set all outputs LOW initially
    ldi r16, 0
    out PORTD, r16

    ; Print initialization message
    ldi ZL, lo8(msg_init)
    ldi ZH, hi8(msg_init)
    rcall print_string

    ; Test all LEDs
    rcall testAllLEDs

main_loop:
    ; Check independentTest flag

```

```

lds r16, independentTest
cpi r16, 0
breq normal_operation

; Run independent LED test if flag set
rcall runIndependentLEDTest
rjmp main_loop_end

normal_operation:
    rcall updateIndicators
    rcall printData

main_loop_end:
    ; Delay 1 second
    ldi r16, 10
delay_loop:
    rcall delay_100ms
    dec r16
    brne delay_loop

    rjmp main_loop

; --- UART Initialization ---
uart_init:
    ; 16MHz, 9600 baud UBRR=103
    ldi r16, 0
    sts UBRR0H, r16
    ldi r16, 103
    sts UBRR0L, r16

    ; Enable transmitter
    ldi r16, (1<<TXEN0)
    sts UCSR0B, r16

    ; Frame format 8N1
    ldi r16, (1<<UCSZ01) | (1<<UCSZ00)
    sts UCSR0C, r16
    ret

```

```

; --- UART send byte ---
uart_send_byte:
wait_tx:
    lds r17, UCSR0A
    sbrc r17, UDRE0
    rjmp wait_tx
    sts UDR0, r16
    ret

; --- Print zero-terminated string ---
print_string:
    lpm r16, Z+
    cpi r16, 0
    breq print_done
    rcall uart_send_byte
    rjmp print_string
print_done:
    ret

; --- I2C Initialization ---
i2c_init:
    ldi r16, (I2C_SLAVE_ADDR<<1)
    sts TWAR, r16

    ldi r16, (1<<TWEN) | (1<<TWEA) | (1<<TWIE)
    sts TWCR, r16

    sei
    ret

; --- TWI interrupt service routine ---
TWI_vect:
    push r16
    in r16, SREG
    push r16

    push r17

```

```

push r18
push r19
push r20
push ZL
push ZH

lds r16, TWSR
andi r16, 0xF8

cpi r16, 0x60
breq twi_slaw_received
cpi r16, 0x80
breq twi_data_received

; Default: re-enable TWI
ldi r16, (1<<TWEN) | (1<<TWEA) | (1<<TWIE)
sts TWCR, r16
rjmp twi_exit

twi_slaw_received:
ldi r19, 0
sts buffer, r19
ldi r16, (1<<TWEN) | (1<<TWEA) | (1<<TWIE) | (1<<TWINT)
sts TWCR, r16
rjmp twi_exit

twi_data_received:
lds r16, TWDR
lds r19, buffer
ldi ZL, lo8(buffer)
ldi ZH, hi8(buffer)
add ZL, r19
adc ZH, r1
st Z, r16
inc r19
sts buffer, r19

cpi r19, 7

```

```

brne twi_continue_receiving

rcall process_received_data

twi_continue_receiving:
    ldi r16, (1<<TWEN) | (1<<TWEA) | (1<<TWIE) | (1<<TWINT)
    sts TWCR, r16

twi_exit:
    pop ZH
    pop ZL
    pop r20
    pop r19
    pop r18
    pop r17
    pop r16
    out SREG, r16
    pop r16
    reti

; --- Process received I2C data ---
process_received_data:
    ldi ZL, lo8(buffer)
    ldi ZH, hi8(buffer)

    ld r16, Z+
    ld r17, Z+
    sts temperature, r16
    sts temperature+1, r17

    ld r16, Z+
    ld r17, Z+
    sts humidity, r16
    sts humidity+1, r17

    ld r16, Z+
    ld r17, Z+
    sts airQuality, r16

```

```

sts airQuality+1, r17

ld r16, Z
sts airQualityStatus, r16
ret

; --- Update LED indicators ---
updateIndicators:
    lds r16, airQualityStatus
    in r17, PORTD
    andi r17,
~((1<<GREEN_LED_PIN) | (1<<YELLOW_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_
PIN))

    cpi r16, 0
    brne check_moderate
    ori r17, (1<<GREEN_LED_PIN)
    rjmp set_indicators

check_moderate:
    cpi r16, 1
    brne check_poor
    ori r17, (1<<YELLOW_LED_PIN)
    rjmp set_indicators

check_poor:
    cpi r16, 2
    brne set_indicators
    ori r17, (1<<RED_LED_PIN) | (1<<BUZZER_PIN)

set_indicators:
    out PORTD, r17
    ret

; --- Print received data ---
printData:
    ldi ZL, lo8(msg_data_header)
    ldi ZH, hi8(msg_data_header)

```

```
rcall print_string

ldi ZL, lo8(msg_temperature)
ldi ZH, hi8(msg_temperature)
rcall print_string

lds r16, temperature
lds r17, temperature+1
rcall print_decimal

ldi ZL, lo8(msg_celsius)
ldi ZH, hi8(msg_celsius)
rcall print_string

ldi ZL, lo8(msg_humidity)
ldi ZH, hi8(msg_humidity)
rcall print_string

lds r16, humidity
lds r17, humidity+1
rcall print_decimal

ldi ZL, lo8(msg_percent)
ldi ZH, hi8(msg_percent)
rcall print_string

ldi ZL, lo8(msg_air_quality)
ldi ZH, hi8(msg_air_quality)
rcall print_string

lds r16, airQuality
lds r17, airQuality+1
rcall print_decimal

ldi ZL, lo8(msg_raw_value)
ldi ZH, hi8(msg_raw_value)
rcall print_string
```

```
ldi ZL, lo8(msg_status)
ldi ZH, hi8(msg_status)
rcall print_string

lds r16, airQualityStatus
cpi r16, 0
brne check_status_1

ldi ZL, lo8(msg_status_good)
ldi ZH, hi8(msg_status_good)
rcall print_string
rjmp print_footer

check_status_1:
cpi r16, 1
brne check_status_2

ldi ZL, lo8(msg_status_moderate)
ldi ZH, hi8(msg_status_moderate)
rcall print_string
rjmp print_footer

check_status_2:
cpi r16, 2
brne status_unknown

ldi ZL, lo8(msg_status_poor)
ldi ZH, hi8(msg_status_poor)
rcall print_string
rjmp print_footer

status_unknown:
ldi ZL, lo8(msg_status_unknown)
ldi ZH, hi8(msg_status_unknown)
rcall print_string

mov r16, r0
ldi r17, 0
```

```

rcall print_decimal

ldi ZL, lo8(msg_status_unknown_end)
ldi ZH, hi8(msg_status_unknown_end)
rcall print_string

print_footer:
    ldi ZL, lo8(msg_footer)
    ldi ZH, hi8(msg_footer)
    rcall print_string
    ret

; --- Test LEDs ---
testAllLEDs:
    ldi ZL, lo8(msg_test_leds)
    ldi ZH, hi8(msg_test_leds)
    rcall print_string

    in r16, PORTD
    andi r16, ~((1<<YELLOW_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_PIN))
    ori r16, (1<<GREEN_LED_PIN)
    out PORTD, r16

    ldi ZL, lo8(msg_green_on)
    ldi ZH, hi8(msg_green_on)
    rcall print_string

    ldi r16, 10
    rcall delay_ms_loop

    in r16, PORTD
    andi r16, ~((1<<GREEN_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_PIN))
    ori r16, (1<<YELLOW_LED_PIN)
    out PORTD, r16

    ldi ZL, lo8(msg_yellow_on)
    ldi ZH, hi8(msg_yellow_on)
    rcall print_string

```

```

ldi r16, 10
rcall delay_ms_loop

in r16, PORTD
andi r16, ~(1<<GREEN_LED_PIN) | (1<<YELLOW_LED_PIN))
ori r16, (1<<RED_LED_PIN) | (1<<BUZZER_PIN)
out PORTD, r16

ldi ZL, lo8(msg_red_buzzer_on)
ldi ZH, hi8(msg_red_buzzer_on)
rcall print_string

ldi r16, 10
rcall delay_ms_loop

in r16, PORTD
andi r16,
~((1<<GREEN_LED_PIN) | (1<<YELLOW_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_
PIN))
out PORTD, r16

ldi ZL, lo8(msg_all_off)
ldi ZH, hi8(msg_all_off)
rcall print_string

ldi ZL, lo8(msg_test_complete)
ldi ZH, hi8(msg_test_complete)
rcall print_string

ret

; --- Independent LED test ---
runIndependentLEDTest:
    in r16, PORTD
    andi r16, ~(1<<YELLOW_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_PIN))
    ori r16, (1<<GREEN_LED_PIN)
    out PORTD, r16

```

```

ldi ZL, lo8(msg_ind_green)
ldi ZH, hi8(msg_ind_green)
rcall print_string

ldi r16, 20
rcall delay_ms_loop

in r16, PORTD
andi r16, ~((1<<GREEN_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_PIN))
ori r16, (1<<YELLOW_LED_PIN)
out PORTD, r16

ldi ZL, lo8(msg_ind_yellow)
ldi ZH, hi8(msg_ind_yellow)
rcall print_string

ldi r16, 20
rcall delay_ms_loop

in r16, PORTD
andi r16, ~((1<<GREEN_LED_PIN) | (1<<YELLOW_LED_PIN))
ori r16, (1<<RED_LED_PIN) | (1<<BUZZER_PIN)
out PORTD, r16

ldi ZL, lo8(msg_ind_red_buzzer)
ldi ZH, hi8(msg_ind_red_buzzer)
rcall print_string

ldi r16, 20
rcall delay_ms_loop

in r16, PORTD
andi r16,
~((1<<GREEN_LED_PIN) | (1<<YELLOW_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_
PIN))
out PORTD, r16

```

```

ldi ZL, lo8(msg_ind_all_off)
ldi ZH, hi8(msg_ind_all_off)
rcall print_string

ldi r16, 10
rcall delay_ms_loop
ret

; --- Delay loop (r16 * 100ms) ---
delay_ms_loop:
rcall delay_100ms
dec r16
brne delay_ms_loop
ret

; --- Delay ~100ms ---
delay_100ms:
ldi r18, 200
outer_loop:
ldi r19, 250
inner_loop:
ldi r20, 40
innermost_loop:
dec r20
brne innermost_loop
dec r19
brne inner_loop
dec r18
brne outer_loop
ret

; --- Print 16-bit decimal (simplified, prints raw bytes) ---
print_decimal:
mov r16, r0
rcall uart_send_byte
mov r16, r1
rcall uart_send_byte
ret

```

```
; --- Strings in flash ---
.section .rodata

msg_init:
    .ascii "Slave Arduino Initialized\r\nReady to receive data and
control indicators\r\n"
    .byte 0

msg_data_header:
    .ascii "\r\n----- Received Data ----- \r\n"
    .byte 0

msg_temperature:
    .ascii "Temperature: "
    .byte 0

msg_celsius:
    .ascii " C\r\n"
    .byte 0

msg_humidity:
    .ascii "Humidity: "
    .byte 0

msg_percent:
    .ascii " %\r\n"
    .byte 0

msg_air_quality:
    .ascii "Air Quality (Potentiometer): "
    .byte 0

msg_raw_value:
    .ascii " Raw Value\r\n"
    .byte 0

msg_status:
```

```

.ascii "Air Quality Status: "
.byte 0

msg_status_good:
.ascii "GOOD (Green LED ON) - Range 0-340\r\n"
.byte 0

msg_status_moderate:
.ascii "MODERATE (Yellow LED ON) - Range 341-682\r\n"
.byte 0

msg_status_poor:
.ascii "POOR (Red LED ON, Buzzer ON) - Range 683-1023\r\n"
.byte 0

msg_status_unknown:
.ascii "UNKNOWN "
.byte 0

msg_status_unknown_end:
.ascii ")\r\n"
.byte 0

msg_footer:
.ascii "-----\r\n"
.byte 0

msg_test_leds:
.ascii "Testing all LEDs and buzzer...\r\n"
.byte 0

msg_green_on:
.ascii "GREEN LED ON\r\n"
.byte 0

msg_yellow_on:
.ascii "YELLOW LED ON\r\n"
.byte 0

```

```

msg_red_buzzer_on:
    .ascii "RED LED and BUZZER ON\r\n"
    .byte 0

msg_all_off:
    .ascii "All indicators OFF\r\n"
    .byte 0

msg_test_complete:
    .ascii "LED test completed. Starting normal operation.\r\n"
    .byte 0

msg_ind_green:
    .ascii "IND TEST: GREEN LED ON\r\n"
    .byte 0

msg_ind_yellow:
    .ascii "IND TEST: YELLOW LED ON\r\n"
    .byte 0

msg_ind_red_buzzer:
    .ascii "IND TEST: RED LED and BUZZER ON\r\n"
    .byte 0

msg_ind_all_off:
    .ascii "IND TEST: All OFF\r\n"
    .byte 0

```

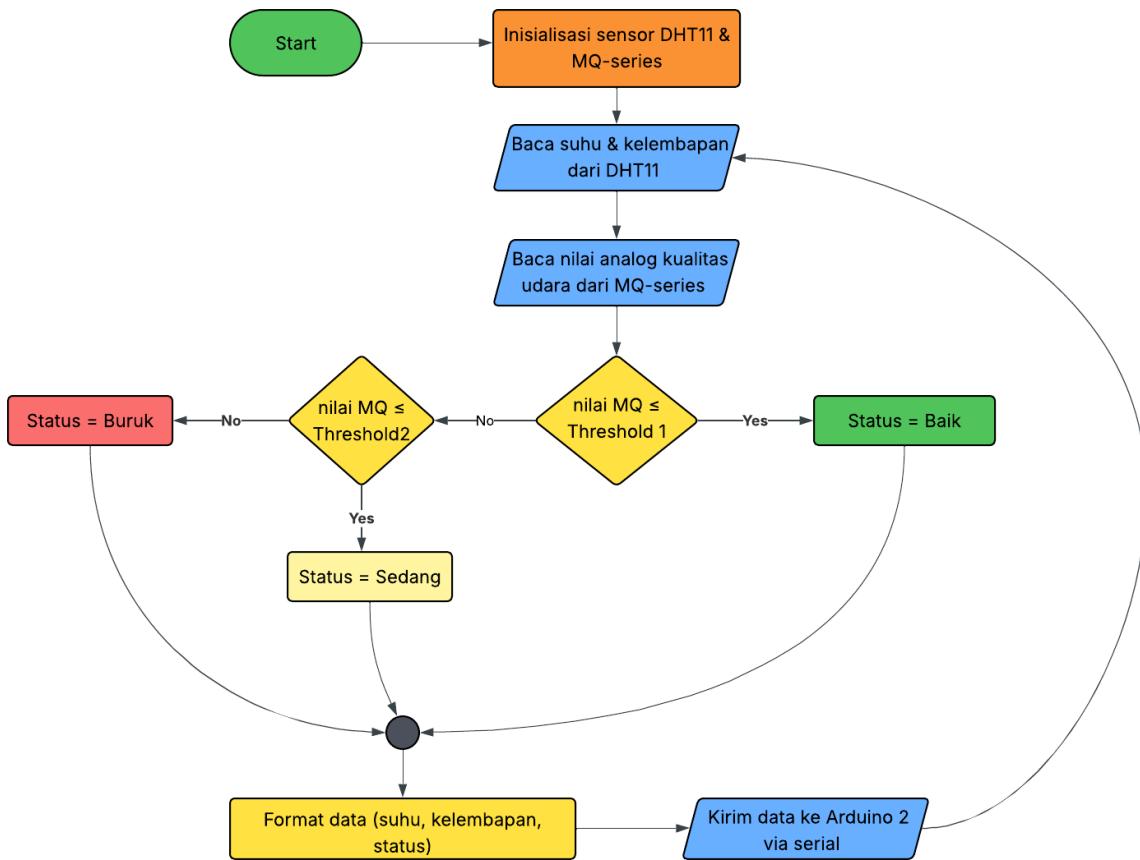
Kode assembly yang disajikan merupakan program untuk mikrokontroler berbasis arsitektur AVR, yang ditulis menggunakan bahasa assembly. Program tersebut mengandung beberapa fungsi utama sebagai berikut:

- **uart\_init:** Fungsi ini mengatur modul UART mikrokontroler untuk komunikasi serial dengan kecepatan 9600 baud. Fungsi ini mengkonfigurasi register UBRR untuk pengaturan baud rate, mengaktifkan transmitter, dan mengatur format frame data 8 bit, no parity, 1 stop bit (8N1). Ini memungkinkan pengiriman data serial untuk debugging.

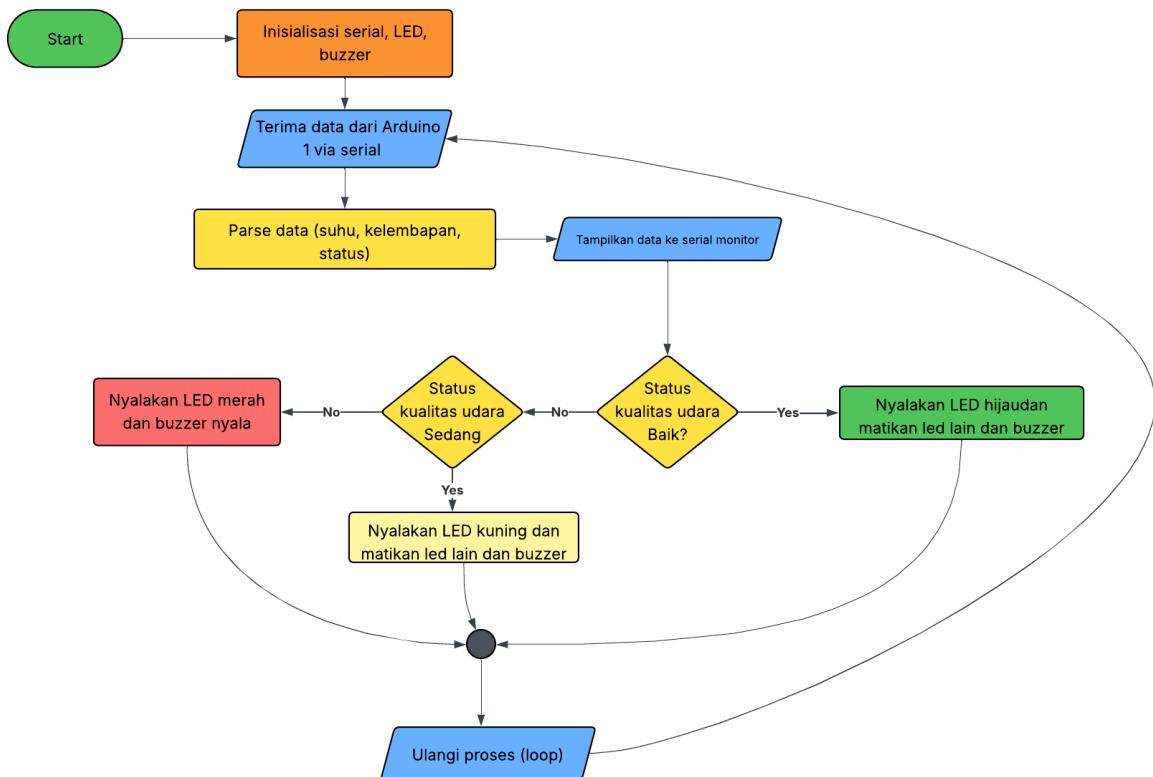
- **uart\_send\_byte:** Fungsi ini mengirim satu byte data melalui UART. Fungsi ini menunggu hingga buffer transmisi kosong (bit UDRE0 diset), kemudian memindahkan byte data ke register UDR0. Setelah itu, data dikirim melalui jalur serial ke perangkat penerima seperti komputer untuk monitoring.
- **print\_string:** Mencetak string karakter yang disimpan di memori flash ke serial monitor. Fungsi membaca tiap karakter dari alamat yang ditunjuk register Z hingga menemukan karakter null (0), lalu mengirim tiap byte tersebut lewat fungsi uart\_send\_byte secara berurutan untuk menampilkan teks di terminal.
- **i2c\_init:** Menginisialisasi modul TWI (I2C) mikrokontroler sebagai slave dengan alamat tertentu (dari konstanta I2C\_SLAVE\_ADDR). Mengaktifkan pengakuan alamat, interrupt TWI, dan modul TWI itu sendiri, sehingga siap menerima data dari master dan menjalankan interrupt handler saat data diterima.
- **TWI\_vect:** Interrupt service routine yang dijalankan ketika ada I2C (TWI). Fungsi ini memeriksa status TWI untuk mengetahui alamat diterima atau data diterima. Jika data diterima, disimpan di buffer dan setelah data lengkap, diteruskan ke proses pengolahan data. Setelah itu interrupt di-enable lagi.
- **process\_received\_data:** Fungsi ini mengekstrak data sensor (temperature, humidity, airQuality, airQualityStatus) dari buffer yang diterima lewat I2C. Data disimpan ke variabel global di SRAM untuk digunakan program utama mengendalikan LED, buzzer, dan mencetak data.
- **updateIndicators:** Berdasarkan nilai status kualitas udara, fungsi ini mengatur output PORTD untuk menyalakan LED hijau (aman), kuning (warning), atau merah dengan buzzer (danger). Semua indikator lain dimatikan, hanya LED dan buzzer yang sesuai kondisi dinyalakan untuk memberi sinyal visual dan suara.
- **printData:** Fungsi ini menampilkan data sensor dan status kualitas udara ke serial monitor dalam format yang jelas. Data suhu, kelembapan, dan kualitas udara ditampilkan bersama label dan status LED yang sedang aktif, sehingga pengguna bisa memantau kondisi secara langsung.
- **testAllLEDs:** Untuk pengujian perangkat output, fungsi ini menyalakan LED hijau, kuning, lalu merah dan buzzer secara bergantian dengan jeda delay. Tujuannya memastikan semua LED dan buzzer berfungsi dan tidak ada kerusakan perangkat keras sebelum digunakan.
- **runIndependentLEDTes:** Mirip testAllLEDs, namun ini dijalankan ketika flag independentTest di-set. Fungsi menyalakan LED dan buzzer satu per satu secara

independen untuk pengecekan manual, membantu debugging sistem apabila ada masalah pada indikator.

- **delay\_100ms & delay\_ms\_loop:** Fungsi delay ini menggunakan nested loop untuk membuat jeda sekitar 100ms per iterasi. delay\_ms\_loop memanggil delay\_100ms sebanyak nilai di register r16 kali, memungkinkan pembuatan delay variabel dengan satuan 100ms untuk mengatur timing pada program.
- **print\_decimal:** Fungsi sederhana yang mengirim nilai 16-bit (dua byte) ke UART. Data dikirim secara mentah byte per byte tanpa konversi ke bentuk angka desimal yang terbaca manusia. Fungsi ini berguna untuk debugging nilai variabel secara cepat tanpa format khusus.



**Gambar 2.** Flowchart arduino I master



**Gambar 3.** Flowchart arduino 2 slave

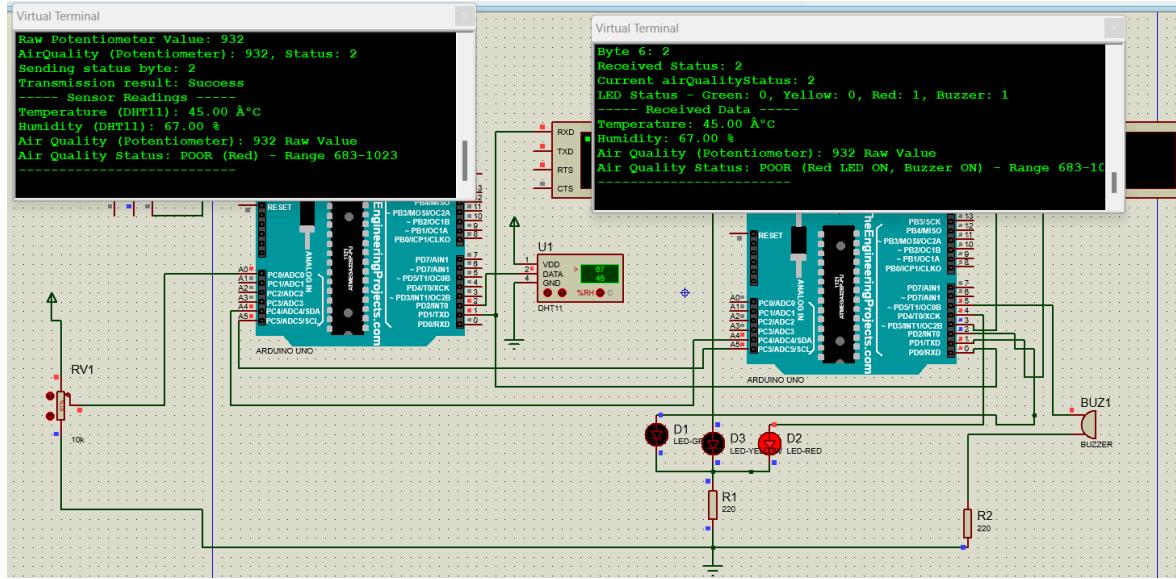
### 2.3 INTEGRASI HARDWARE DENGAN SOFTWARE

Jika melihat dari proteus, terdapat 2 arduino yang masing-masing berperan sebagai Master dan Slave. Pada kedua arduino tersebut terdapat dua buah serial monitor yang port RXD-nya tersambung ke port D1/TX (PD1) pada masing-masing arduino. Lalu pada arduino master tersambung dengan DHT11 dengan menggunakan port D2 (PD2) yang mengarah langsung ke DATA pada DHT11. Lalu pada port D14 (PC0) juga mengarah langsung ke Sensor Gas atau MQ-2. Lalu terdapat protokol komunikasi serial yaitu I2C yang memungkinkan kedua arduino ini saling bertukar data melalui dua kabel yang terhubung pada port D18 (PC4) dan D19 (PC5). Pada arduino slave terdapat tiga LED yang tersambung ke D2 (PD2) yang mengarah ke LED merah, lalu D3 (PD3) yang mengarah ke LED kuning, dan D4 (PD4) yang mengarah ke LED hijau. Lalu diport D5 (PD5) tersambung ke Buzzer yang berfungsi sebagai pemberitahuan. Sama seperti arduino master, arduino slave juga terdapat protokol komunikasi serial yaitu I2C yang tersambung pada port D18 (PC4) dan D19 (PC5).

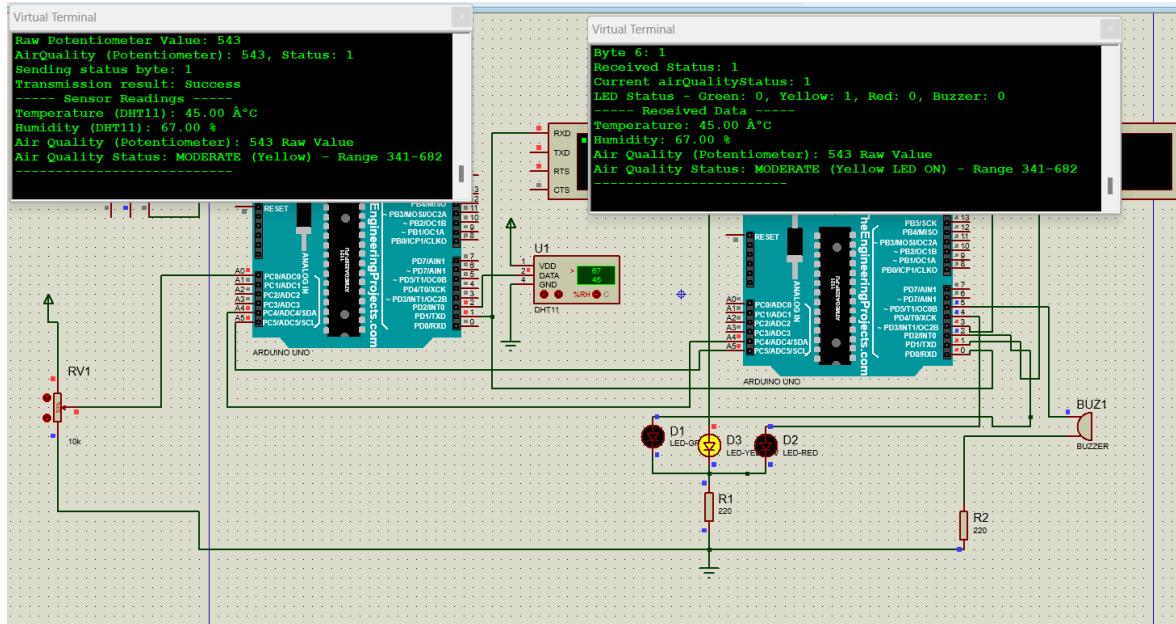
## BAB 3

### UJI COBA DAN EVALUASI

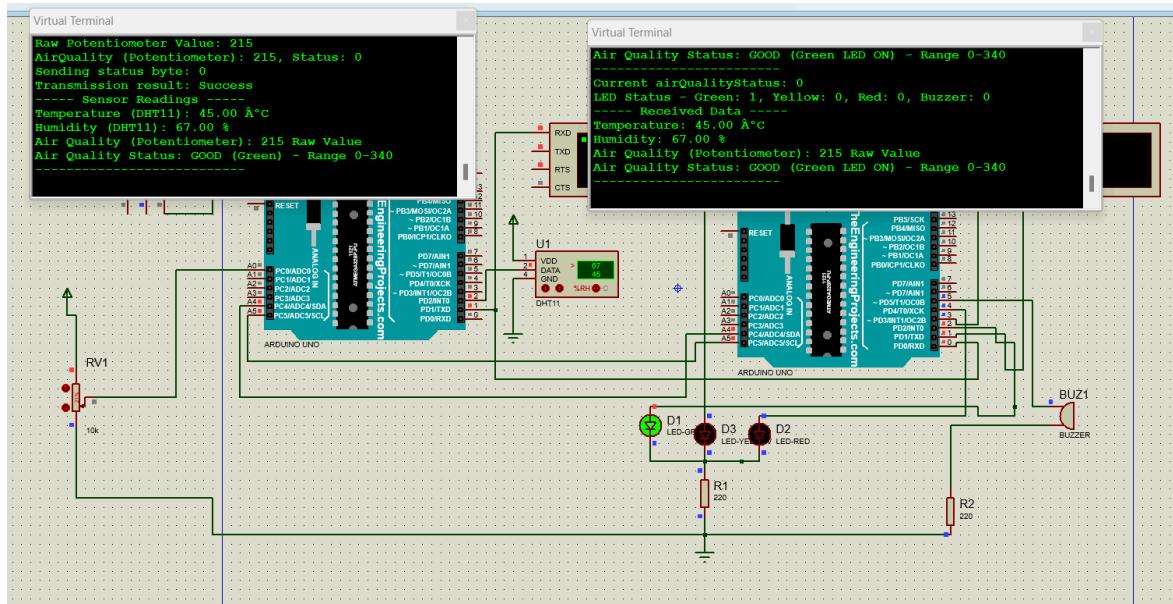
#### 3.1 UJI COBA RANGKAIAN



Gambar 4. Contoh LED Merah Menyala



Gambar 5. Contoh LED Kuning Menyala



**Gambar 6.** Contoh LED Hijau Menyala

Setelah proses perakitan dan penulisan program selesai, tahap percobaan dapat dimulai. Pada awalnya, percobaan dilakukan dalam kondisi normal untuk mengamati hasil pengukuran dan memastikan bahwa sensor DHT11 serta sensor gas MQ-2 berfungsi dengan baik dan data yang diperoleh tampil di Serial Monitor.

Selanjutnya, untuk memastikan buzzer beroperasi dengan benar, percobaan akan dilakukan dengan kode yang sudah dibuat sebagai berikut:

```
; --- Test LEDs ---
testAllLEDs:
    ldi ZL, lo8(msg_test_leds)
    ldi ZH, hi8(msg_test_leds)
    rcall print_string

    in r16, PORTD
    andi r16, ~((1<<YELLOW_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_PIN))
    ori r16, (1<<GREEN_LED_PIN)
    out PORTD, r16

    ldi ZL, lo8(msg_green_on)
```

```

ldi ZH, hi8(msg_green_on)
rcall print_string

ldi r16, 10
rcall delay_ms_loop

in r16, PORTD
andi r16, ~((1<<GREEN_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_PIN))
ori r16, (1<<YELLOW_LED_PIN)
out PORTD, r16

ldi ZL, lo8(msg_yellow_on)
ldi ZH, hi8(msg_yellow_on)
rcall print_string

ldi r16, 10
rcall delay_ms_loop

in r16, PORTD
andi r16, ~((1<<GREEN_LED_PIN) | (1<<YELLOW_LED_PIN))
ori r16, (1<<RED_LED_PIN) | (1<<BUZZER_PIN)
out PORTD, r16

ldi ZL, lo8(msg_red_buzzer_on)
ldi ZH, hi8(msg_red_buzzer_on)
rcall print_string

ldi r16, 10
rcall delay_ms_loop

in r16, PORTD
andi r16,
~((1<<GREEN_LED_PIN) | (1<<YELLOW_LED_PIN) | (1<<RED_LED_PIN) | (1<<BUZZER_
PIN))
out PORTD, r16

ldi ZL, lo8(msg_all_off)
ldi ZH, hi8(msg_all_off)

```

```
rcall print_string

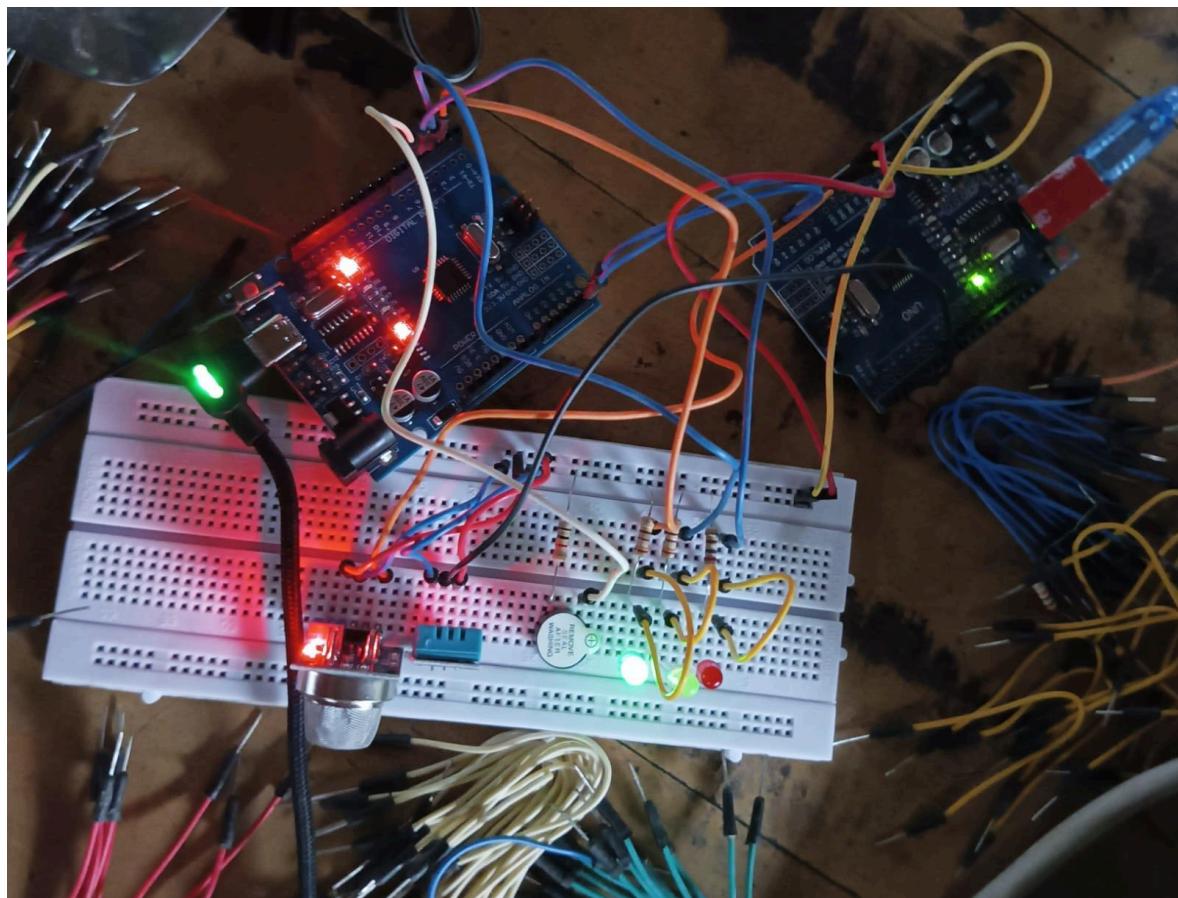
ldi ZL, lo8(msg_test_complete)
ldi ZH, hi8(msg_test_complete)
rcall print_string

ret
```

Dengan cara ini, buzzer akan aktif dan berbunyi ketika kode yang dibuat sudah dibaca oleh arduino, sehingga dapat memverifikasi fungsi alarm pada sistem ini.

### 3.2 HASIL UJI COBA

Berdasarkan percobaan yang telah dilakukan sebelumnya, diperoleh hasil sebagai berikut. Untuk mengetahui apakah sensor DHT11 dan MQ-2 berhasil menangkap data lingkungan serta mengirimkannya ke Arduino, dapat dilihat hasilnya pada uraian berikut:



*Gambar 7. Rangkaian yang Sudah Dibuat*

Selanjutnya, percobaan dilakukan untuk memastikan fungsi buzzer bekerja dengan baik. Pada tahap ini, batas ambang aman yang mengaktifkan buzzer diubah pada kode program. Setelah pengubahan tersebut, buzzer berhasil menyala, menandakan bahwa penyesuaian batas ambang yang diturunkan telah berhasil dan sistem alarm berfungsi sesuai harapan.

### 3.3 EVALUASI

Evaluasi proyek Air Quality Monitor menunjukkan bahwa alat ini mampu mengukur suhu dan kelembaban, serta mendeteksi keberadaan gas yang mudah terbakar seperti metana dan propana. Saat sistem dijalankan, data akan diproses dalam bentuk analog dan ditampilkan

pada serial monitor. Selain itu, ketika gas terdeteksi, maka akan diukur tingkat kelembapan dan suhunya, lalu dikirimkan ke output yang benar. Jika gas melebihi angka yang sudah disesuaikan oleh kode buzzer akan otomatis menyala sebagai tanda peringatan. Hasil pengujian menunjukkan keakuratan alat, terlihat dari tampilan suhu dalam satuan celcius dan kelembaban pada serial monitor, serta bunyi buzzer saat gas diberikan. Dengan sistem ini, masyarakat dapat memantau suhu, kelembaban, dan keberadaan gas di lingkungan mereka dengan mudah.

## **BAB 4**

### **KESIMPULAN**

Air Quality Control merupakan sebuah alat untuk memeriksa dan memantau suhu, kelembapan udara, dan lingkungan, serta memberikan peringatan suara ketika kualitas udara sudah terlalu buruk. Alat ini menggunakan sensor DHT11 yang berfungsi untuk mengambil data suhu dan kelembapan udara lingkungan dan sensor MQ-2 yang berfungsi untuk mendeteksi adanya gas-gas berbahaya pada udara lingkungan. Kedua sensor ini dipasang pada master Arduino untuk mengambil data dari lingkungan. Data yang diperoleh akan dikirimkan dari master Arduino ke slave Arduino untuk nanti diproses. Berdasarkan hasil pemrosesan input, slave Arduino akan mengeluarkan output nilai suhu dan kelembapan ke serial monitor pada sebuah interval untuk pemantauan. Selain itu, terdapat 3 LED dan buzzer sebagai visualisasi kualitas udara. Ketika kualitas udara memburuk, sistem akan memberikan peringatan melalui warna nyala LED dan suara buzzer.

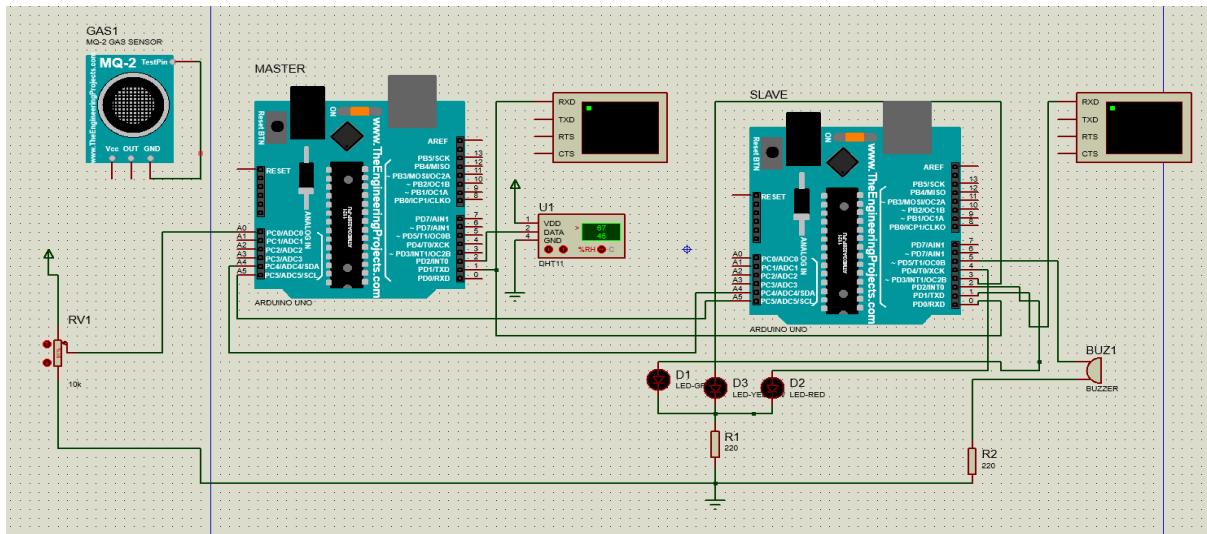
Proyek ini telah mengimplementasikan modul 2, 3, 4, 5, 6, 8, dan 9 pada praktikum Sistem Embedded selama semester ini. Dengan pembuatan proyek ini, Air Quality Control dapat membantu masyarakat memantau suhu dan kelembapan udara di suatu lingkungan, serta memberikan peringatan suara ketika kualitas udara sudah terlalu buruk. Menggunakan alat ini, masyarakat dapat melakukan tindakan preventif melalui pemantauan nilai suhu dan kelembapan udara, serta dengan warna LED yang menjadi indikasi kualitas udara. Selain itu, masyarakat dapat segera melakukan evakuasi ketika menerima peringatan dari suara buzzer. Harapan kami adalah alat ini dapat dipergunakan oleh masyarakat umum dengan baik dan apabila terdapat kesempatan di masa depan, alat ini dapat kami kembangkan lagi menjadi lebih bermanfaat.

## REFERENSI

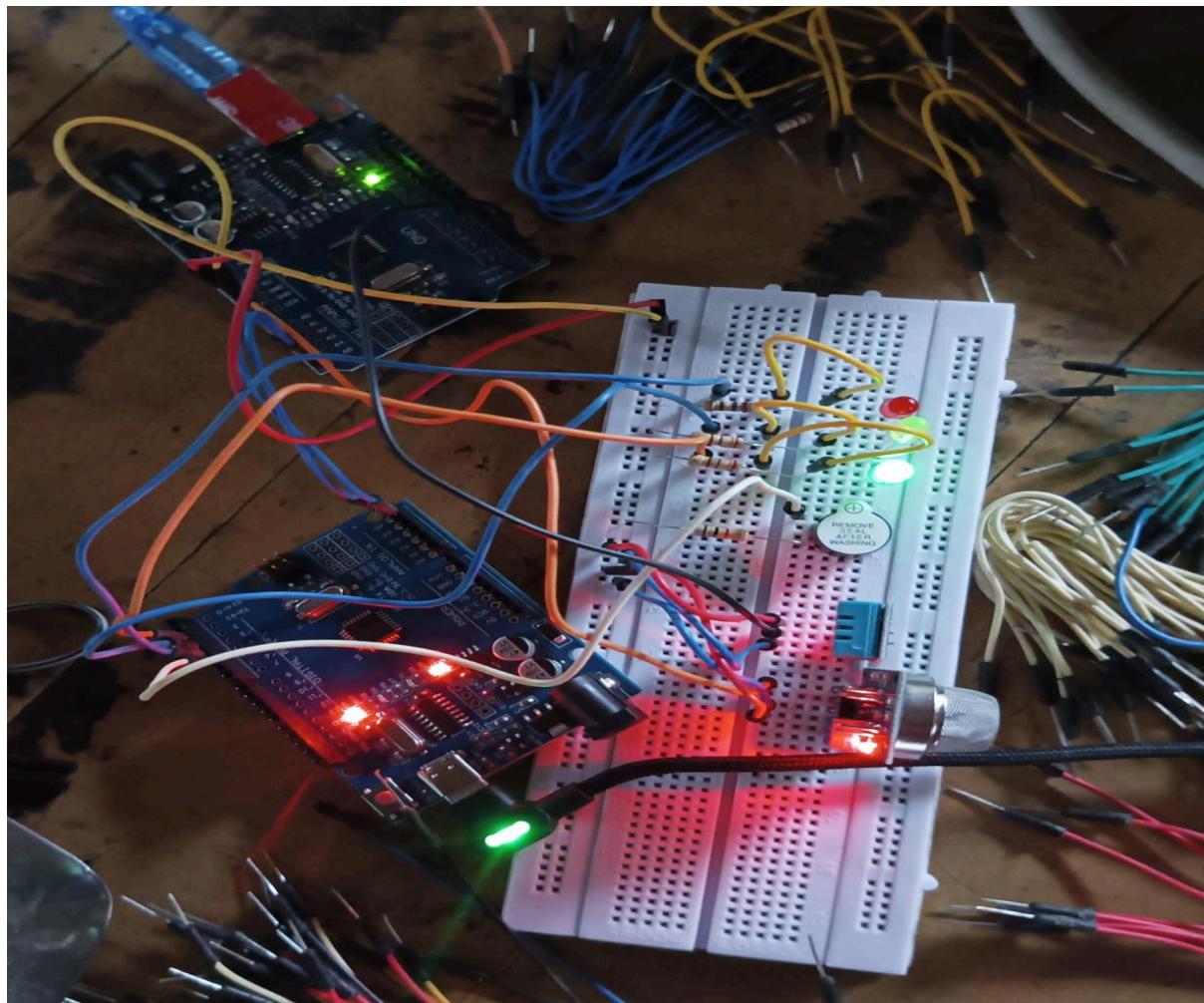
- [1] Electronicos Caldas. "DHT11 Datasheet." [Online]. Available: [https://www.electronicoscaldas.com/datasheet/DHT11\\_Aosong.pdf](https://www.electronicoscaldas.com/datasheet/DHT11_Aosong.pdf). [Accessed: May 14, 2025]
- [2] Mouser Electronics. "DHT11 Technical Data Sheet Translated Version." [Online]. Available: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>. [Accessed: May 12, 2025].
- [3] Last Minute Engineers, "Interface DHT11 Module With Arduino," [Online]. Available: <https://lastminuteengineers.com/dht11-module-arduino-tutorial/> [Accessed: May 14, 2025]
- [4] Scott Campbell. "How to Set Up the DHT11 Humidity Sensor on an Arduino." [Online]. Available: <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/> [Accessed: May 17, 2025]
- [5] Pololu. "MQ2 Gas Sensor Datasheet." [Online]. Available: <https://www.pololu.com/file/0J309/MQ2.pdf>. [Accessed: May 14, 2025]
- [6] Random Nerd Tutorials. "Guide for MQ-2 Gas/Smoke Sensor with Arduino." [Online]. Available: <https://randomnerdtutorials.com/guide-for-mq-2-gas-smoke-sensor-with-arduino/>. [Accessed: May 17, 2025].
- [7] Last Minute Engineers. "How MQ2 Gas/Smoke Sensor Works? & Interface it with Arduino." [Online]. Available: <https://lastminuteengineers.com/mq2-gas-sensor-arduino-tutorial/>. [Accessed: May 17, 2025].
- [8] Debashis Das, "How Does MQ-2 Flammable Gas and Smoke Sensor Work with Arduino?" [Online]. Available: <https://circuitdigest.com/microcontroller-projects/interfacing-mq2-gas-sensor-with-arduino>. [Accessed: May 17, 2025].

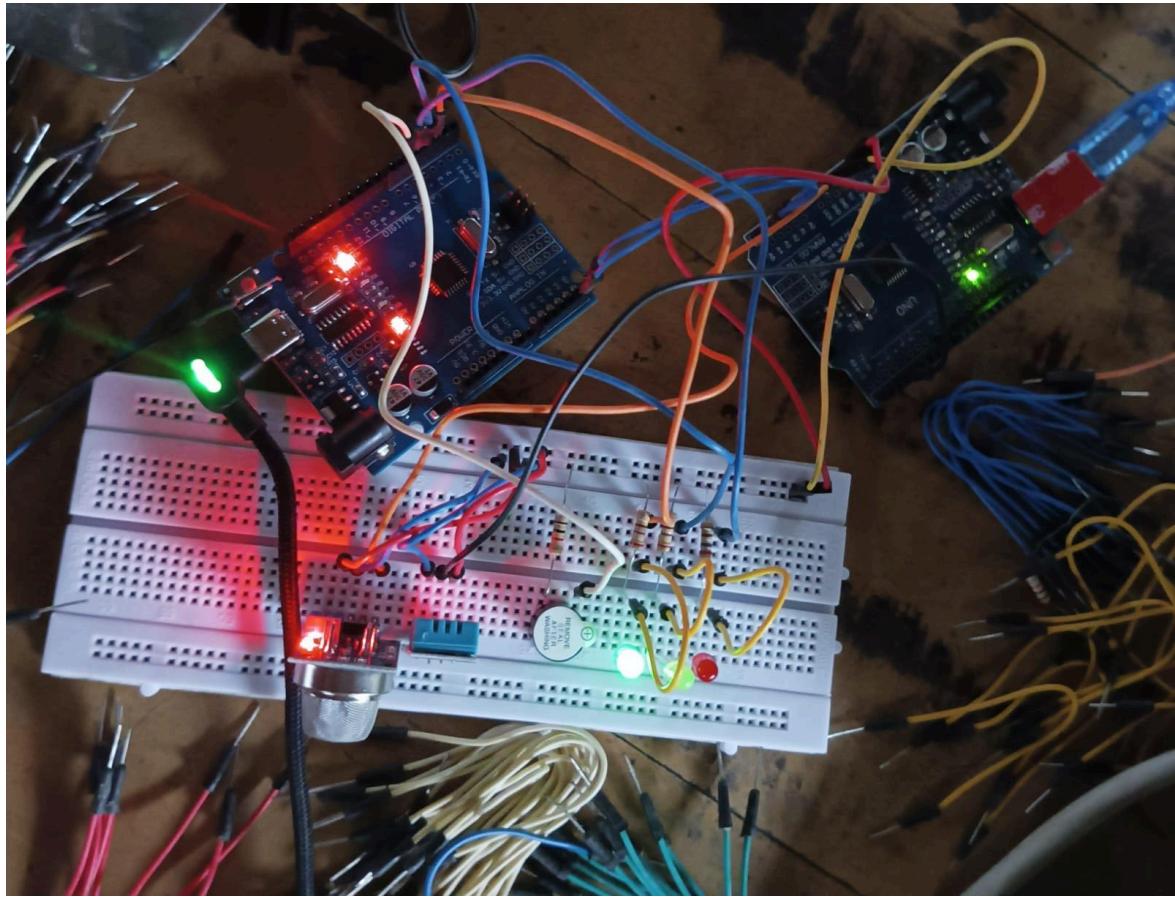
## LAMPIRAN

### Skematika Rangkaian Proteus



## Foto Rangkaian Fisik





## Dokumentasi Pengerajan Proyek

