



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

**NoiseShield**

**(Sistem Pendekripsi Kebisingan Otomatis dengan Sensor Mikrofon)**

**GROUP 13**

<b>Audy Natalie Cecilia R</b>	<b>2306266962</b>
<b>M. Avicenna Raffaiz A.</b>	<b>2206062844</b>
<b>Muhammad Hilmy M.</b>	<b>2306267006</b>
<b>Muhammad Pavel</b>	<b>2306243363</b>

## PREFACE

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, atas segala rahmat dan karunia-Nya sehingga laporan proyek akhir ini dapat kami selesaikan dengan lancar. Laporan ini dibuat sebagai bagian dari penyelesaian mata kuliah Sistem Embedded pada Tahun Ajaran 2024/2025. Proyek yang kami kerjakan berjudul NoiseShield - Sistem Otomatis Deteksi Kebisingan Menggunakan Sensor Mikrofon, yang bertujuan untuk menghadirkan solusi dalam memantau tingkat suara di lingkungan sekitar secara otomatis dan real-time.

Sistem ini memanfaatkan sensor mikrofon sebagai perangkat input utama yang akan mengukur level kebisingan, kemudian data tersebut diolah oleh mikrokontroler dengan bahasa Assembly. Melalui penggeraan ini, kami memperoleh pengalaman praktis sekaligus pemahaman lebih mendalam tentang penerapan teori sistem embedded dan penanganan sinyal real-time. Dalam proses pengembangan, kami melakukan penelitian dan perancangan yang matang agar sistem dapat berfungsi secara optimal sesuai dengan tujuan yang telah ditetapkan.

Ucapan terima kasih kami tujuhan kepada para asisten laboratorium yang telah memberikan arahan, dukungan, dan bantuan teknis selama pelaksanaan proyek ini. Meskipun demikian, kami menyadari laporan ini masih belum sempurna dan mengharapkan kritik serta saran yang membangun untuk perbaikan di masa mendatang.

Depok, Mei 15, 2025

Group 13

## TABLE OF CONTENTS

<b>PREFACE.....</b>	<b>2</b>
<b>TABLE OF CONTENTS.....</b>	<b>3</b>
<b>CHAPTER 1</b>	
<b>INTRODUCTION.....</b>	<b>4</b>
1.1 PROBLEM STATEMENT.....	x
1.2 PROPOSED SOLUTION.....	x
1.3 ACCEPTANCE CRITERIA.....	x
1.4 ROLES AND RESPONSIBILITIES.....	x
<b>CHAPTER 2</b>	
<b>IMPLEMENTATION.....</b>	<b>x</b>
2.1 HARDWARE DESIGN AND SCHEMATIC.....	x
2.2 SOFTWARE DEVELOPMENT.....	x
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	x
<b>CHAPTER 3</b>	
<b>TESTING AND ANALYSIS.....</b>	<b>x</b>
3.1 TESTING.....	x
3.2 RESULT.....	x
3.3 EVALUATION.....	x
<b>CHAPTER 4</b>	
<b>CONCLUSION.....</b>	<b>x</b>
<b>REFERENCES.....</b>	<b>x</b>
<b>APPENDICES.....</b>	<b>x</b>
Appendix A: Project Schematic.....	x
Appendix B: Documentation.....	x

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 PROBLEM STATEMENT**

Kebisingan di lingkungan rumah sakit, khususnya di ruang perawatan bayi baru lahir (newborn), merupakan isu serius yang sering kali kurang mendapat perhatian. Bayi, terutama yang dirawat di ruang NICU (Neonatal Intensive Care Unit), memiliki sistem saraf yang masih sangat rentan terhadap gangguan suara. Paparan kebisingan secara terus-menerus dapat menyebabkan gangguan tidur, stres fisiologis, peningkatan detak jantung, serta gangguan pada perkembangan kognitif jangka panjang. Selain itu, suara tangisan bayi yang tidak segera terdeteksi bisa menjadi tanda adanya kebutuhan mendesak seperti rasa lapar, ketidaknyamanan, atau masalah medis yang memerlukan respons cepat dari tenaga kesehatan.

Sayangnya, dalam praktiknya, pemantauan kebisingan dan tangisan bayi masih sangat bergantung pada pengamatan langsung oleh perawat atau tenaga medis. Hal ini menjadi kurang efektif, terutama ketika jumlah tenaga medis terbatas dan tingkat kewaspadaan menurun di luar jam sibuk. Selain itu, rumah sakit belum banyak menerapkan sistem pemantauan akustik otomatis yang dapat membantu mengidentifikasi suara-suara kritis secara real-time. Tanpa sistem yang mendukung pemantauan suara secara kontinu dan akurat, risiko keterlambatan dalam merespons kebutuhan bayi meningkat, yang pada akhirnya dapat memengaruhi kualitas perawatan dan keselamatan pasien.

## **1.2 PROPOSED SOLUTION**

Untuk mengatasi permasalahan tersebut, kami mengembangkan sistem *NoiseShield*, yaitu alat pendeteksi kebisingan otomatis berbasis mikrokontroler Arduino Uno dan sensor suara KY-037. Sistem ini dirancang untuk mengukur intensitas suara di lingkungan sekitar dan memberikan sinyal peringatan saat tingkat kebisingan melebihi ambang batas tertentu. Sensor suara KY-037 digunakan karena memiliki sensitivitas tinggi terhadap perubahan suara, memungkinkan deteksi tangisan bayi atau suara bising lainnya secara real-time. Data yang ditangkap sensor akan diproses oleh Arduino, yang kemudian mengklasifikasikan kondisi menjadi tiga kategori: hening, normal, dan gaduh.

Sebagai bentuk umpan balik kepada petugas medis, sistem ini dilengkapi dengan tiga buah LED sebagai indikator visual, di mana masing-masing LED menyala sesuai dengan kategori kondisi suara yang terdeteksi. Dengan penggunaan LED ini, petugas medis dapat dengan cepat mengenali tingkat kebisingan di ruang perawatan. Threshold atau ambang batas suara dapat diatur ulang sesuai kebutuhan tiap ruang, menjadikan sistem ini fleksibel untuk digunakan di berbagai jenis ruang rawat, termasuk NICU yang memerlukan sensitivitas tinggi terhadap suara.

*NoiseShield* menawarkan solusi efektif dalam membantu pemantauan akustik di rumah sakit. Dengan sistem ini, proses deteksi tangisan bayi menjadi otomatis, cepat, dan tidak memerlukan pengawasan visual terus-menerus oleh petugas medis. Sistem ini tidak hanya mendukung efisiensi kerja tenaga kesehatan, tetapi juga berkontribusi terhadap peningkatan kualitas pelayanan serta keamanan bayi yang dirawat.

### **1.3 ACCEPTANCE CRITERIA**

Target kriteria yang ingin dicapai pada proyek berikut adalah:

1. Sistem mampu memantau tingkat kebisingan secara otomatis menggunakan Arduino Uno dan sensor suara KY-037.
2. Sistem dapat mengklasifikasikan kondisi suara menjadi tiga kategori: hening, kondusif, dan gaduh.
3. Sistem memberikan peringatan visual melalui LED saat tingkat kebisingan melebihi threshold.
4. Threshold kebisingan dapat disesuaikan agar sesuai dengan kebutuhan ruang rawat bayi.

### **1.4 ROLES AND RESPONSIBILITIES**

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

Roles	Responsibilities	Person
Paper, PPT, code writer	Membuat laporan, PPT, flowchart, Readme dan kode	Audy Natalie Cecilia R
Paper	Membuat laporan	M. Avicenna Raffaiz Adiharsa
Proteus schematic and the actual circuit designer, code writer	Merancang rangkaian schematic Arduino, sensor mikrofon, LED, agar dapat terhubung, kode dan laporan	Muhammad Hilmy Mahardika
Programmer Arduino	Memprogram Arduino dan laporan	Muhammad Pavel

*Table 1. Roles and Responsibilities*

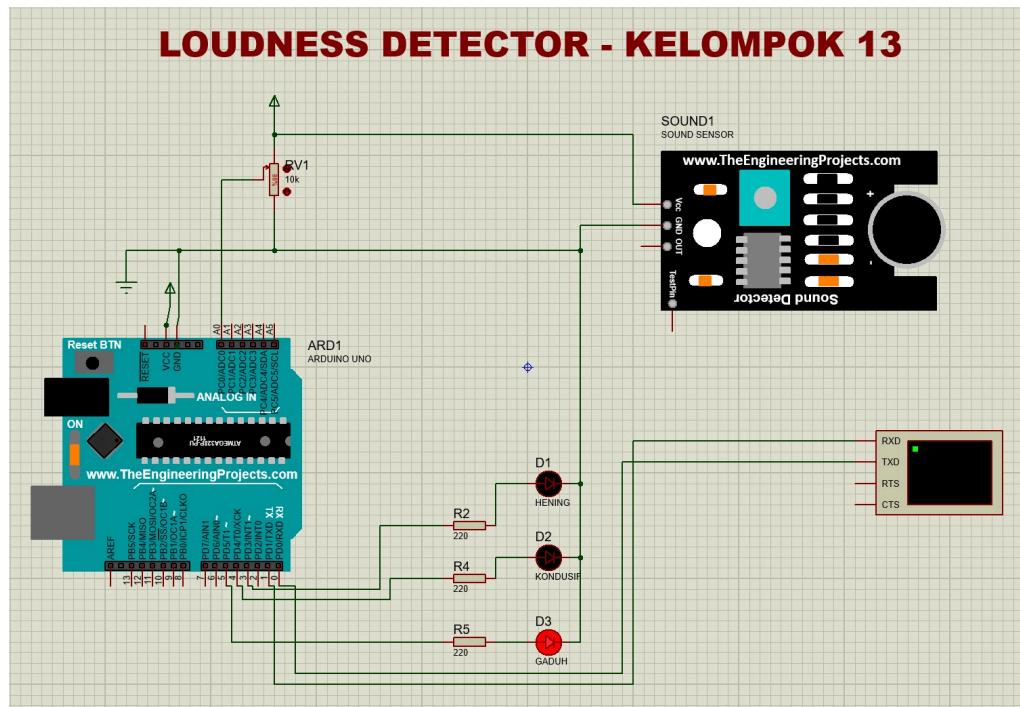
## 1.5 TIMELINE AND MILESTONES

ACTIVITIES	MEI												
	5	6	7	8	9	10	11	12	13	14	15	16	17
Project Idea Discussion													
Purchasing Required Hardware													
Designing in Proteus													
Writing Code													
Hardware and Software Integration													
Final Product Assembly and Testing													
Report Writing													

**Table 2. Timeline and Milestones**

## CHAPTER 2 IMPLEMENTATION

### 2.1 HARDWARE DESIGN AND SCHEMATIC



**Picture 1. Proteus Design**

Pembuatan alat *NoiseShield* dilakukan dengan terlebih dahulu membuat skema alat atau prototype pada software Proteus. Skema ini memudahkan dalam melakukan perangkaian alat pada rangkaian fisik, sehingga meminimalisir kerusakan atau kegagalan komponen akibat kesalahan dalam merangkai atau rangkaian yang belum terintegrasi dengan baik. Komponen yang dibutuhkan dalam mendesain dan merangkai skema beserta fungsinya pada rangkaian antara lain:

### **1. Arduino Uno**

Mikrokontroler berbasis ATmega328p yang berfungsi sebagai otak dari sistem loudness detector. Mikrokontroler ini bertugas memproses data dari sensor suara, mengatur logika pendektsian tingkat kebisingan, dan mengendalikan output berdasarkan pembacaan sensor. Arduino Uno dipilih karena memiliki pin digital dan analog yang cukup untuk menghubungkan berbagai komponen dalam sistem ini.

### **2. Sensor Suara (Sound Sensor)**

Sensor suara digunakan untuk mendekksi intensitas suara di lingkungan sekitar. Sensor ini memiliki mikrofon yang mengubah gelombang suara menjadi sinyal listrik yang dapat dibaca oleh Arduino melalui pin analog. Sensor ini juga dilengkapi dengan penguat operasional untuk memperkuat sinyal suara yang diterima, sehingga pembacaan menjadi lebih akurat.

### **3. LED Indikator (D1, D2, D3)**

Kami menggunakan tiga LED dengan warna berbeda sebagai indikator visual tingkat kebisingan:

- LED Hijau (D1): Menyala ketika tingkat kebisingan rendah atau normal
- LED Kuning (D2): Menyala ketika tingkat kebisingan sedang
- LED Merah (D3): Menyala ketika tingkat kebisingan tinggi dan telah melampaui batas yang ditentukan

#### **4. Resistor (R2, R3, R5)**

Resistor digunakan untuk membatasi arus yang mengalir ke LED, melindunginya dari kerusakan akibat arus berlebihan. Resistor yang digunakan memiliki nilai resistansi  $220\Omega$  untuk masing-masing LED.

#### **5. Potensiometer**

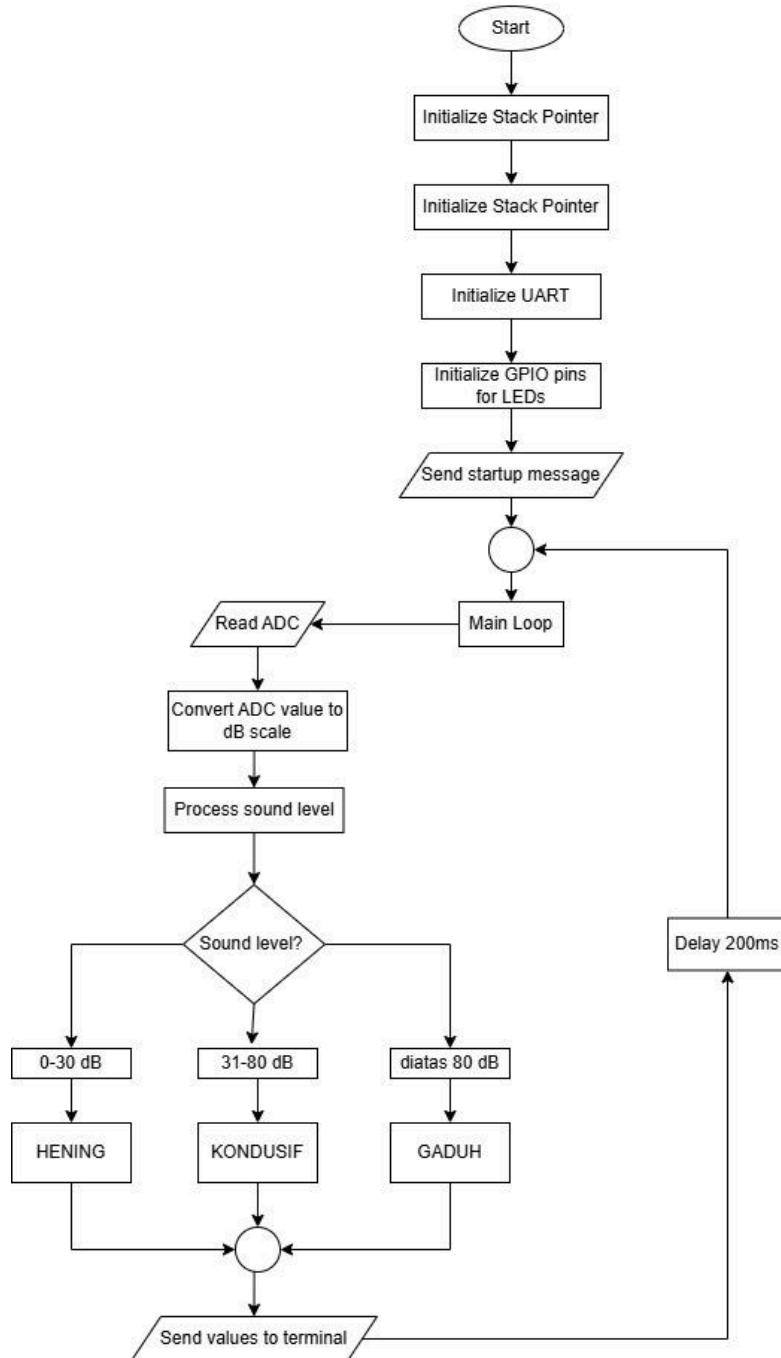
Potensiometer digunakan untuk mengatur tingkat sensitivitas sensor suara atau untuk menetapkan ambang batas kebisingan yang dapat diatur secara manual oleh pengguna. Dengan adanya potensiometer, sistem menjadi lebih fleksibel karena dapat disesuaikan dengan kebutuhan lingkungan yang berbeda-beda. Potensiometer terhubung ke pin A0 Arduino sehingga nilai resistansi yang berubah dapat dibaca dan diinterpretasikan oleh program.

#### **6. Virtual Terminal**

Virtual Terminal pada Proteus digunakan sebagai monitor untuk menampilkan nilai pembacaan tingkat kebisingan secara real-time. Terminal ini terhubung dengan Arduino melalui komunikasi serial, sehingga data pengukuran dapat ditampilkan dalam format teks pada layar virtual. Hal ini memudahkan kami untuk memantau, mengkalibrasi, dan menguji sistem deteksi kebisingan tanpa perlu menggunakan perangkat display fisik. Virtual Terminal juga memungkinkan pencatatan dan analisis data kebisingan selama periode waktu tertentu.

## 2.2 SOFTWARE DEVELOPMENT

FLOWCHART FINAL PROJECT MBD  
GROUP 13 - NoiseShield



Picture 2. Code Flowchart

- **Inisialisasi**

Proses inisialisasi ini memastikan semua komponen beroperasi dengan benar. Sistem dimulai dengan mengatur stack pointer untuk manajemen memori, dilanjutkan dengan konfigurasi UART pada 9600 baud untuk komunikasi serial, menyiapkan pin GPIO untuk LED indikator, dan menyiapkan ADC untuk membaca sensor mikrofon. Tahap ini juga mengirimkan pesan pembuka ke terminal untuk mengkonfirmasi bahwa sistem telah berhasil diinisialisasi dan siap untuk digunakan.

```
main:  
    ; Initialize stack pointer  
    ldi r16, hi8(RAMEND)  
    out SPH, r16  
    ldi r16, lo8(RAMEND)  
    out SPL, r16  
  
    ; Initialize UART (virtual terminal)  
    rcall uart_init  
  
    ; Initialize GPIO pins  
    rcall gpio_init  
  
    ; Initialize ADC  
    rcall adc_init  
  
    ; Send startup message  
    rcall send_startup_msg
```

- **Pemrosesan Level Suara**

Bagian ini merupakan otak dari sistem yang bertanggung jawab untuk mengklasifikasikan tingkat kebisingan ke dalam tiga kategori berdasarkan nilai desibel yang terukur. Sistem menggunakan dua nilai ambang batas: HENING\_THRESHOLD (30 dB) dan GADUH\_THRESHOLD (80 dB). Pembacaan di bawah 30 dB dikategorikan sebagai HENING, pembacaan di atas 80 dB dianggap GADUH, dan nilai di antaranya diklasifikasikan sebagai KONDUSIF. Pemrosesan ini memungkinkan sistem untuk memberikan informasi yang relevan tentang lingkungan akustik sekitar.

```

process_level:
    ; r16 = level dB (0-100)
    ; Mengembalikan r17 = kategori (0=HENING, 1=KONDUSIF, 2=GADUH)

    cpi r16, GADUH_THRESHOLD
    brsh is_gaduh

    cpi r16, HENING_THRESHOLD
    brsh is_kondusif

    ldi r17, 0      ; HENING (0-30 dB)
    ret

is_kondusif:
    ldi r17, 1      ; KONDUSIF (31-80 dB)
    ret

is_gaduh:
    ldi r17, 2      ; GADUH (>80 dB)
    ret

```

- **Kontrol Output LED**

Mengelola tampilan visual dari sistem dengan mengaktifkan LED yang sesuai berdasarkan kategori level suara yang terdeteksi. Sistem terlebih dahulu mematikan semua LED untuk menghindari keadaan ambigu, kemudian menyalakan LED yang sesuai dengan kategori saat ini: LED HENING untuk lingkungan yang tenang, LED KONDUSIF untuk tingkat kebisingan moderat, dan LED GADUH untuk lingkungan yang berisik. Pendekatan ini memberikan indikasi visual yang jelas dan langsung tentang kondisi kebisingan.

```

update_leds:
    ; r17 = kategori (0=HENING, 1=KONDUSIF, 2=GADUH)

    ; Matikan semua LED terlebih dahulu
    cbi PORTD, HENING_LED
    cbi PORTD, KONDUSIF_LED
    cbi PORTD, GADUH_LED

    ; Nyalakan LED yang sesuai
    cpi r17, 0
    brne not_hening_led
    sbi PORTD, HENING_LED
    ret

not_hening_led:
    cpi r17, 1
    brne not_kondusif_led

```

```

    sbi PORTD, KONDUSIF_LED
    ret

not_kondusif_led:
    sbi PORTD, GADUH_LED
    ret

```

- **Pembacaan ADC**

Proses ini melibatkan konversi sinyal analog dari sensor mikrofon menjadi nilai digital yang dapat diproses oleh mikroprosesor. Sistem memulai konversi ADC, menunggu hingga proses selesai, kemudian membaca hasil 10-bit dan mengkonversinya ke skala desibel 0-100 dB dengan metode pembagian sederhana (nilai ADC dibagi 10). Algoritma pembagian diimplementasikan menggunakan pengurangan berulang untuk memastikan kompatibilitas dengan arsitektur mikroprosesor dan mengoptimalkan performa.

```

read_adc:
    ; Mulai konversi
    lds r16, ADCSRA
    ori r16, (1<<ADSC)
    sts ADCSRA, r16

    ; Tunggu konversi selesai
adc_wait:
    lds r16, ADCSRA
    sbrc r16, ADSC
    rjmp adc_wait

    ; Baca hasil ADC
    lds r16, ADCL      ; Harus membaca ADCL terlebih dahulu
    lds r17, ADCH

    ; Konversi ADC 10-bit (0-1023) ke dB (0-100)
    ; Kita akan menggunakan rumus sederhana: dB = ADC / 10
    ; Gabungkan ADCL dan ADCH menjadi nilai 10-bit
    mov r24, r16        ; Byte rendah di r24
    mov r25, r17        ; Byte tinggi di r25

    ; Bagi dengan 10 dengan pengurangan berulang
    ldi r21, 10          ; Pembagi
    clr r16              ; Penghitung hasil
div10_loop:
    cp r24, r21          ; Bandingkan byte rendah dengan 10
    cpc r25, r1           ; Bandingkan byte tinggi dengan 0
    brlo div10_done       ; Jika kurang dari 10, selesai

    ; Kurangi 10
    subi r24, 10
    sbci r25, 0

```

```

; Tambah hasil
inc r16
rjmp div10_loop

div10_done:
    ret           ; Hasil di r16 (0-100)

```

- **Komunikasi Serial**

Modul ini bertanggung jawab untuk menyediakan umpan balik ke pengguna melalui terminal serial. Sistem mengirimkan informasi dalam format yang terstruktur, termasuk nilai desibel numerik dan kategori kebisingan dalam bentuk teks (HENING, KONDUSIF, atau GADUH). Data ini dikirim dengan format yang konsisten untuk memudahkan pemantauan dan analisis. Komunikasi serial ini sangat berguna untuk debugging, pengujian, dan pemantauan jangka panjang terhadap kinerja sistem.

```

send_to_terminal:
    ; r16 = level dB, r17 = kategori
    push r16
    push r17

    ; Cetak nilai dB - DIREVISI
    rcall print_decimal

    ; Cetak teks "dB "
    ldi r16, 'd'
    rcall uart_tx_char
    ldi r16, 'B'
    rcall uart_tx_char
    ldi r16, ' '
    rcall uart_tx_char

    ; Cetak kategori berdasarkan r17
    pop r17
    cpi r17, 0
    brne not_hening_print

    ; Cetak "HENING"
    rcall print_hening
    rjmp finish_print

not_hening_print:
    cpi r17, 1
    brne not_kondusif_print

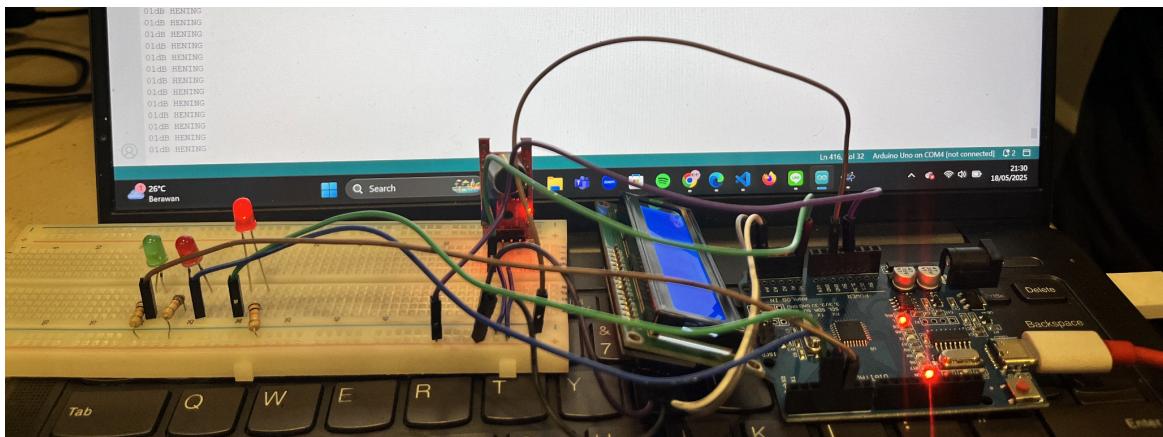
    ; Cetak "KONDUSIF"
    rcall print_kondusif

```

```
rjmp finish_print

not_kondusif_print:
    ; Cetak "GADUH"
    rcall print_gaduh
```

## 2.3 HARDWARE AND SOFTWARE INTEGRATION



*Picture 3. Rangkaian Asli yang teringetrasi*

Integrasi *hardware* dan *software* pada *NoiseShield* ini dilakukan dengan menggunakan mikrokontroler AVR sebagai pusat kendali utama yang mengkoordinasikan seluruh komponen sistem. Mikrokontroler ini berfungsi sebagai penghubung antara sensor mikrofon, LED indikator, dan komunikasi serial agar seluruh perangkat dapat bekerja secara sinkron sesuai program yang telah diatur. Sensor mikrofon yang terhubung ke pin ADC mikrokontroler mendekripsi tingkat kebisingan lingkungan dengan mengubah gelombang suara menjadi sinyal listrik. Tiga buah LED indikator (hijau, kuning, dan merah) yang terhubung ke pin digital mikrokontroler digunakan untuk menampilkan tiga kategori kebisingan: HENING, KONDUSIF, dan GADUH, masing-masing mewakili tingkat kebisingan yang berbeda.

Pada sisi *software*, mikrokontroler diprogram untuk membaca nilai analog dari sensor mikrofon secara berkala, mengkonversi data mentah ADC 10-bit menjadi nilai desibel, lalu mengklasifikasikannya berdasarkan ambang batas yang sudah ditentukan.

LED indikator akan menyala sesuai dengan kategori kebisingan: LED hijau untuk di bawah 30 dB (HENING), kuning untuk 31-80 dB (KONDUSIF), dan merah untuk di atas 80 dB (GADUH). Selain itu, sistem juga menggunakan komunikasi serial dengan baudrate 9600 untuk mengirim data kebisingan secara real-time ke komputer atau perangkat pemantauan lain, memungkinkan pemantauan jarak jauh dan analisis data. Integrasi *hardware* dan *software* ini menghasilkan sistem yang responsif dan efektif, memberikan umpan balik visual cepat sekaligus menyediakan data rinci untuk analisis lebih lanjut, dengan desain modular yang memudahkan pemeliharaan dan pengembangan di masa depan.

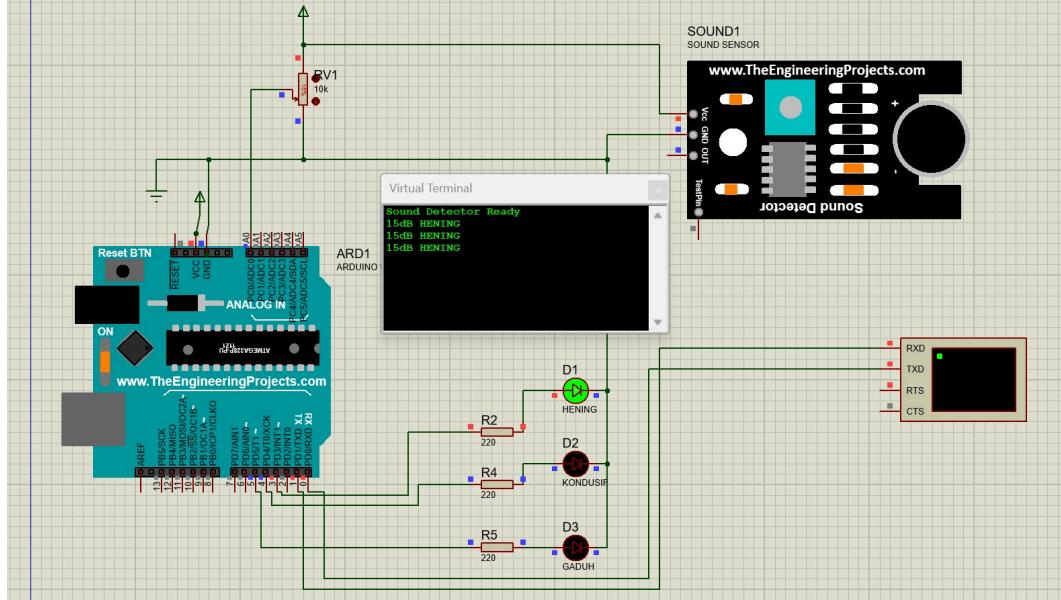
## **CHAPTER 3**

### **TESTING AND EVALUATION**

#### **3.1 TESTING**

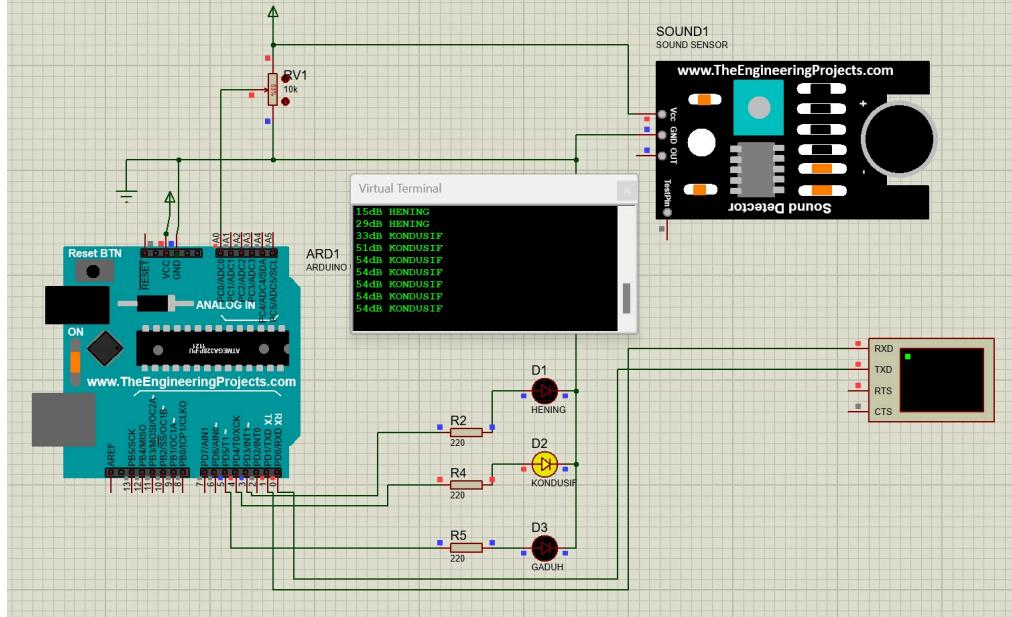
Kami melakukan testing pada rangkaian hardware dan juga melalui proteus. Kami memastikan apakah kinerja rangkaian berhasil atau tidak dan melakukan debugging jika ada yang dirasa masih kurang benar. Pada pengujian di proteus kami mendapatkan kendala karena sound sensor pada proteus tidak menghasilkan output analog seperti yang kami inginkan walaupun input sensor sudah menggunakan potentiometer, oleh karena itu kami menggunakan potentiometer untuk langsung input ke arduino pada pin A0. Untuk rangkaian asli kami menggunakan serial monitor pada Arduino IDE untuk melihat tampilannya dan juga testing pada sensor aslinya. Untuk sensor asli kami menggunakan KY-037 tetapi sepertinya sensor tersebut tidak sensitif walaupun kami sudah mengatur potentiometer pada module sensornya.

## LOUDNESS DETECTOR - KELOMPOK 13

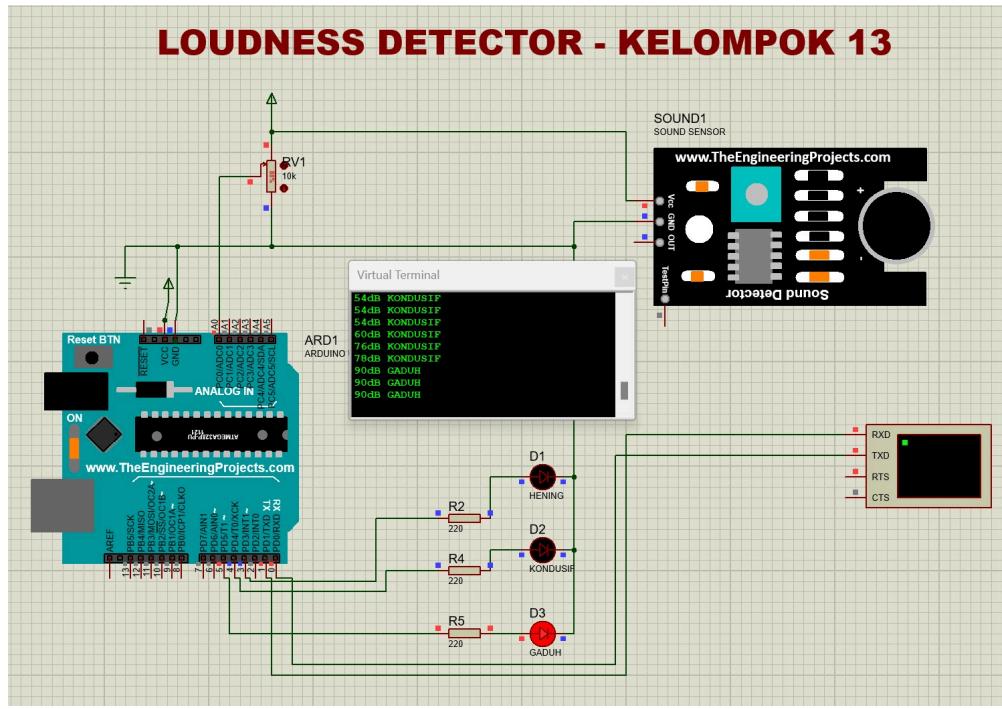


*Picture 4. Testing Hening*

## LOUDNESS DETECTOR - KELOMPOK 13



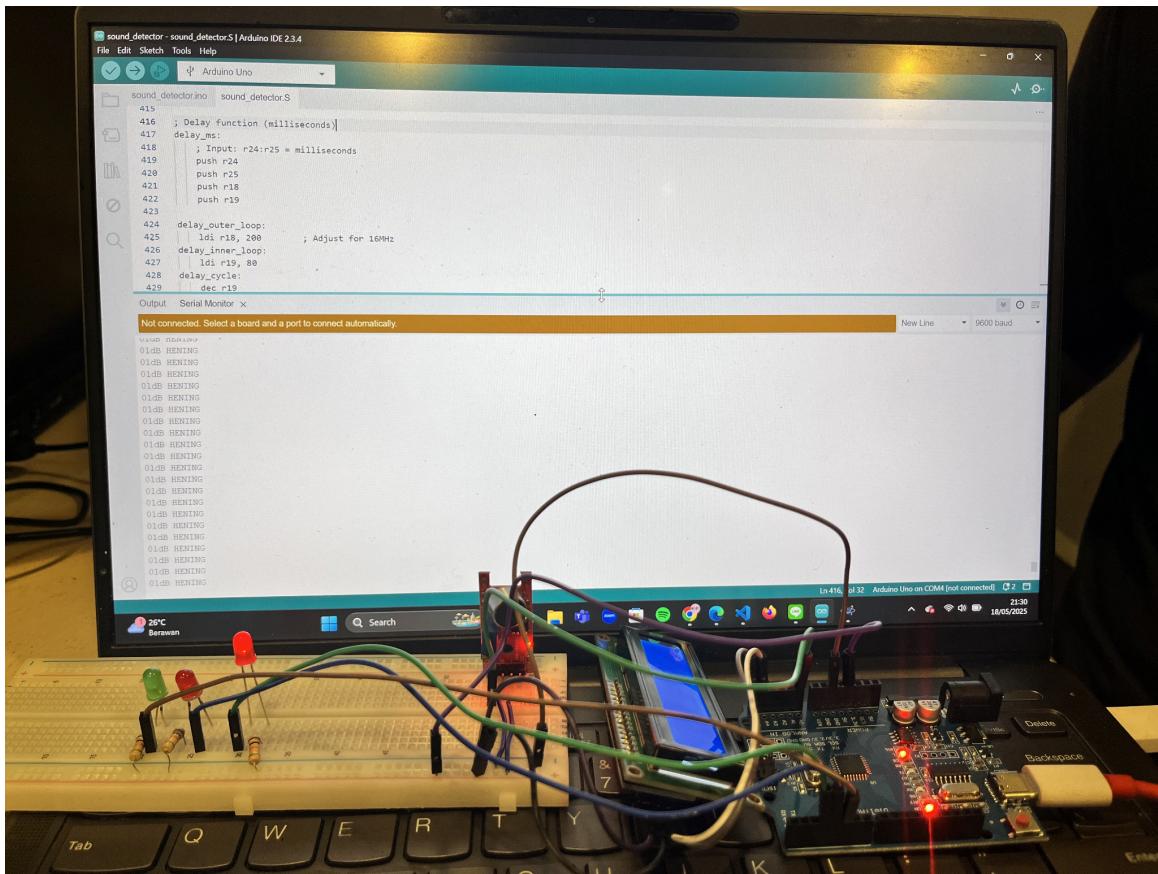
*Picture 5. Testing Kondusif*



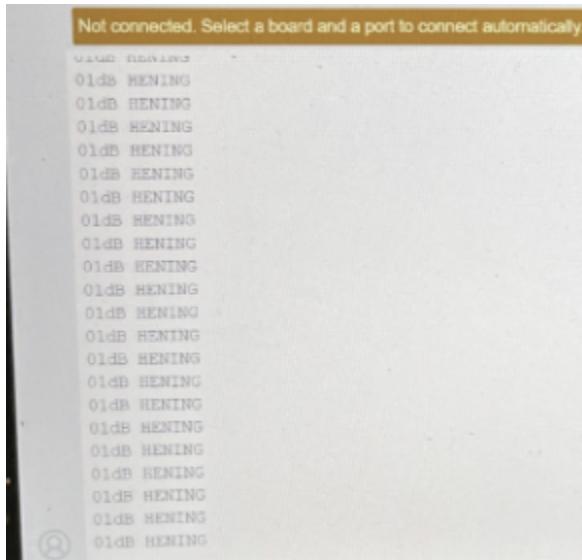
*Picture 6. Testing Gaduh*

### 3.2 RESULT

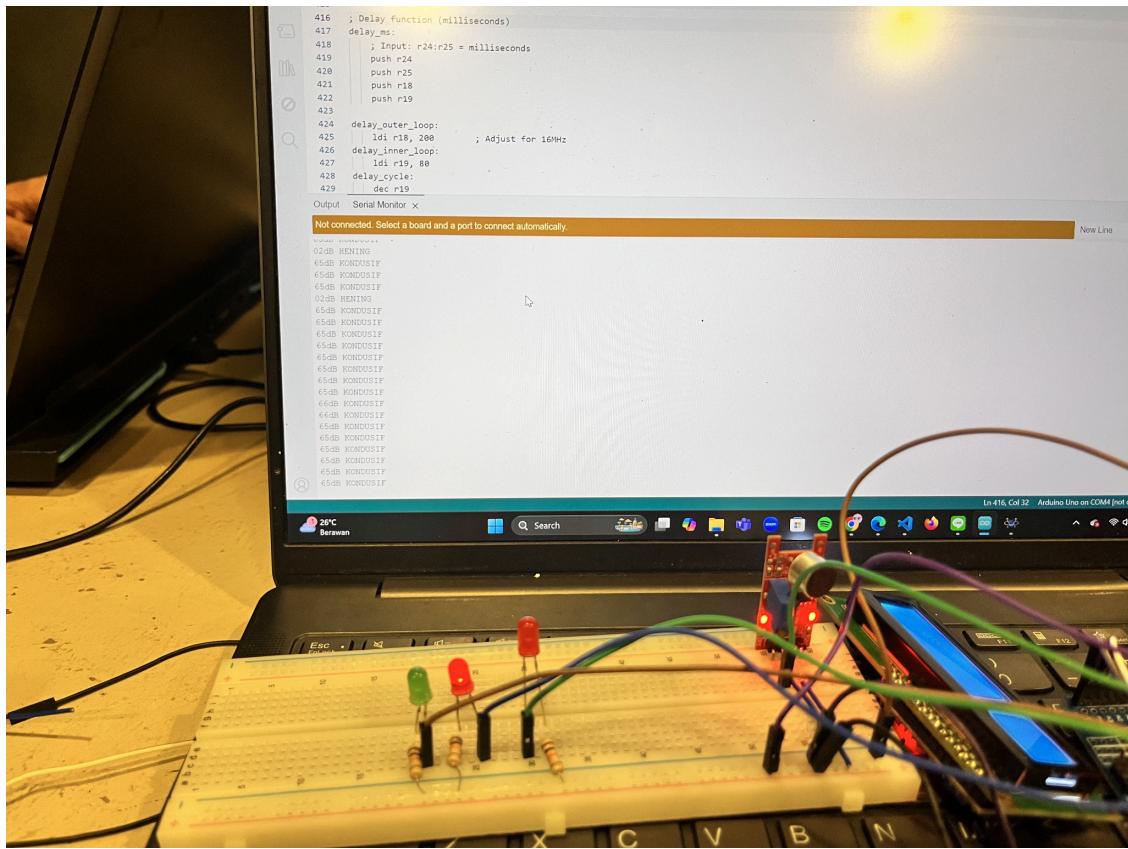
Hasil yang kami dapatkan setelah melakukan testing dengan menggunakan proteus dan rangkaian asli berhasil dengan ada beberapa catatan. Untuk hasil testing dengan proteus dengan input potentiometer berhasil dengan baik sesuai dengan yang kami inginkan dimana saat level desibel 0-30 db menghasilkan output “HENING” dengan ditandai juga LED D1 yang menyala, saat level desibell 31-79 db menghasilkan output “KONDUSIF” dengan ditandai LED D2 yang menyala, dan saat level desibel >30 menghasilkan output “GADUH” yang ditandai dengan LED D3 yang menyala. Untuk hasil testing pada rangkaian asli juga berjalan tetapi untuk mencapai level gaduh sulit dan led nya juga bekerja, hasil yang ditampilkan pada serial monitor juga sesuai dengan ketentuannya.



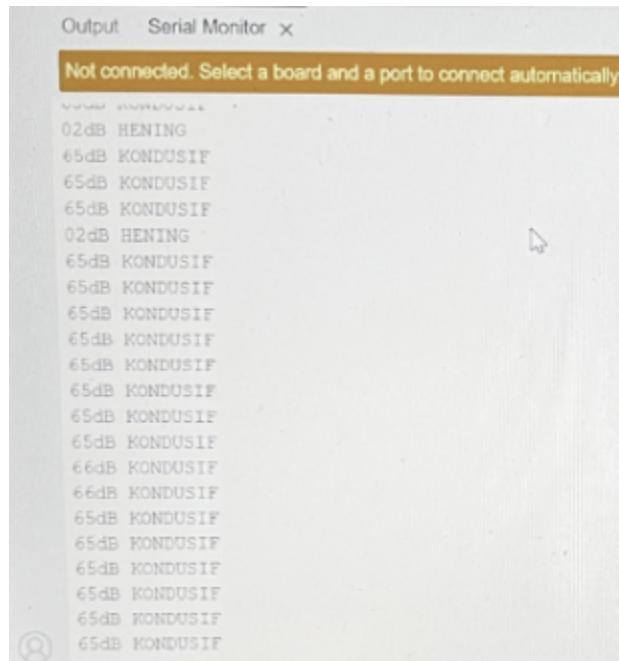
*Picture 7. Result Hening - Lampu kanan menyala*



*Picture 8. Close up Serial Monitor Hening*



*Picture 9. Result Kondusif - Lampu tengan menyala*



*Picture 10. Close up Serial Monitor Kondusif*

### **3.3 EVALUATION**

Evaluasi proyek NoiseShield dapat ditinjau dari dua aspek utama: proses penggerjaan dan hasil implementasi sistem. Selama proses pengembangan, tim kami berhasil menjalankan kolaborasi yang efektif dengan distribusi tanggung jawab yang sesuai keahlian masing-masing anggota. Komunikasi antar tim berjalan lancar memungkinkan integrasi komponen paper writing, perancangan rangkaian, dan pemrograman Arduino terlaksana dengan tepat waktu. Dari perspektif hasil teknis, sistem berhasil memenuhi tujuan utama dalam mendeteksi dan mengklasifikasikan tingkat kebisingan menjadi tiga kategori dengan indikator visual yang jelas. Pengujian simulasi di Proteus menunjukkan kinerja yang memuaskan dengan respons yang akurat terhadap variasi input. Meski demikian, implementasi fisik menghadapi kendala teknis khususnya pada sensitivitas sensor KY-037 yang kurang optimal ketika mendeteksi level suara kategori gaduh. Komunikasi serial berfungsi dengan baik menampilkan data kebisingan secara real-time melalui Serial Monitor. Untuk pengembangan masa depan, peningkatan kualitas dapat dilakukan melalui penggunaan sensor dengan sensitivitas lebih tinggi, penambahan fitur notifikasi nirkabel ke perangkat medis, serta kalibrasi yang lebih presisi untuk penerapan di lingkungan rumah sakit yang sebenarnya.

## **CHAPTER 4**

## **CONCLUSION**

Proyek NoiseShield telah berhasil menciptakan sistem pemantauan kebisingan yang efektif menggunakan Arduino Uno dan sensor suara, dengan kemampuan mengklasifikasikan kondisi akustik lingkungan ke dalam tiga kategori yang ditampilkan melalui indikator LED. Sistem ini memenuhi seluruh kriteria penerimaan yang ditetapkan, yaitu mampu memantau tingkat kebisingan secara otomatis, mengklasifikasikan kondisi suara menjadi tiga kategori, memberikan peringatan visual melalui LED saat kebisingan melebihi ambang batas, dan memiliki threshold yang dapat disesuaikan. Meskipun terdapat beberapa keterbatasan teknis pada sensitivitas sensor KY-037 dalam implementasi fisik, konsep dan desain NoiseShield telah membuktikan

potensinya sebagai solusi praktis untuk membantu tenaga medis dalam memantau kondisi akustik di ruang perawatan bayi. Dengan pengembangan lebih lanjut, terutama pada penggunaan sensor yang lebih sensitif dan algoritma pemrosesan suara yang lebih canggih, NoiseShield dapat menjadi alat yang sangat berguna dalam meningkatkan kualitas perawatan dan keselamatan pasien di lingkungan rumah sakit.

## REFERENCES

- [1]microchip, “AVR ® Instruction Set Manual AVR ® Instruction Set Manual,” 2021. Accessed: May 11, 2025. [Online]. Available: <https://ww1.microchip.com/downloads/en/DeviceDoc/AVR-InstructionSet-Manual-DS40002198.pdf>
- [2]Digilab, “Modul 8 SSF : SPI & I2C,” Google Docs, 2020. <https://docs.google.com/document/d/1CsIbwLVUrsKjZ3YhyF0J-gsGNU1RCQu7JTYMCx3cFY/edit?tab=t.0> (accessed May 11, 2025).
- [3]Digilab, “Modul 9 SSF : Sensor Interfacing,” Google Docs, 2019. <https://docs.google.com/document/d/14D8bETDw8x-BbeWWfg2QrjEE1WJA17kAZVDu79iCzCs/edit?tab=t.0> (accessed May 11, 2025).
- [4]Digilab, “Modul 4 SSF : Serial Port,” Google Docs, 2019. [https://docs.google.com/document/d/1rRWvBgL3Nsb\\_h10131A-1kiGQkrGGNLedYoeVIGR9zg/edit?tab=t.0](https://docs.google.com/document/d/1rRWvBgL3Nsb_h10131A-1kiGQkrGGNLedYoeVIGR9zg/edit?tab=t.0) (accessed May 11, 2025).
- [5]Digilab, “Modul 2 SSF: Introduction to Assembly & I/O Programming,” Google Docs, 2019. [https://docs.google.com/document/d/1s84Y1xrGyJwWXQJgdBQL6bTaFFZtiOsA4\\_3YGouP1CY/edit?tab=t.0](https://docs.google.com/document/d/1s84Y1xrGyJwWXQJgdBQL6bTaFFZtiOsA4_3YGouP1CY/edit?tab=t.0) (accessed May 11, 2025).

## APPENDICES

### Appendix A: Documentation



*Picture 11. Documentation*