EMBEDDED SYSTEM FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

SMARTFLOW – Smart Flood-Level Observation and Warning

GROUP PA-9

Alfonsus Tanara Gultom	**2306267126**
Jonathan Matius Weni	**2306161896**
Siti Amalia Nurfaidah	**2306161851**
Wilman Saragih Sitio	**2306161776**
Xavier Daniswara	**2206030230**

PREFACE

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat, karunia, dan petunjuk-Nya sehingga proyek akhir ini dapat kami selesaikan dengan baik. Proyek ini merupakan hasil kerja keras, dedikasi, serta kolaborasi tim Kelompok PA9 yang terdiri dari kami Alfonsus Tanara Gultom, Jonathan Matius Weni, Siti Amalia Nurfaidah, Wilman Saragih Sitio, dan Xavier Daniswara.

SmartFlow adalah sistem pemantauan ketinggian air real-time berbasis *liquid level sensor* S8050 NPN. Saat air menyentuh sensor, transistor mengubah arus menjadi sinyal analog yang kemudian dikonversi oleh ADC, diproses mikrokontroler, dan dikomunikasikan melalui serial. Protokol SPI diimplementasikan dengan Arduino *master* membaca data sensor dan Arduino *slave* menampilkan hasil pada serial monitor, sementara LED indikator dan timer memastikan pembaruan data secara berkala.

Motivasi pembuatan SmartFlow muncul dari tingginya frekuensi bencana banjir di berbagai wilayah. Kami yakin bahwa teknologi sederhana namun handal seperti ini dapat menjadi peringatan pertama sebelum bencana terjadi, memungkinkan setiap orang untuk mengambil tindakan mitigasi lebih cepat dan tepat.

Kami menyadari masih terdapat ruang untuk penyempurnaan, oleh karena itu kami sangat menghargai kritik, saran, dan masukan konstruktif demi pengembangan lebih lanjut. Semoga SmartFlow memberikan manfaat nyata dan menginspirasi inovasi di bidang sistem peringatan dini bencana.

Depok, 7 Mei, 2025

Group PA-19

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Bencana banjir merupakan salah satu permasalahan serius yang kerap melanda berbagai wilayah di Indonesia, terutama pada saat musim penghujan. Dampak yang ditimbulkan tidak hanya terbatas pada kerugian material seperti rusaknya infrastruktur dan harta benda, tetapi juga mengganggu aktivitas masyarakat dan mengancam keselamatan jiwa. Salah satu penyebab utama yang memperparah dampak banjir adalah keterlambatan dalam memperoleh informasi mengenai kenaikan permukaan air.

Dalam banyak kasus, tidak adanya sistem pemantauan yang akurat dan responsif menyebabkan keterlambatan dalam pengambilan keputusan mitigasi, seperti evakuasi atau pengamanan aset penting. Oleh karena itu, diperlukan sebuah sistem yang mampu melakukan pemantauan ketinggian air secara *real-time*, serta memberikan peringatan dini ketika air mencapai batas berbahaya.

Seiring berkembangnya teknologi, khususnya dalam bidang *Internet of Things* (IoT) dan *embedded systems*, kini semakin memungkinkan untuk membangun sistem pemantauan lingkungan yang cerdas, efisien, dan terjangkau. Salah satu pendekatan yang dapat digunakan adalah memanfaatkan sensor ketinggian air berbasis mikrokontroler, yang mampu mendeteksi perubahan permukaan air secara otomatis dan mengirimkan data secara *real-time* kepada pengguna.

1.2 PROPOSED SOLUTION

Berdasarkan permasalahan tersebut, proyek akhir ini bertujuan untuk mengembangkan sebuah sistem yang diberi nama **Smart Flood-Level Observation and Warning** (SmartFlow), yaitu sistem pemantauan level air secara real-time yang mampu memberikan peringatan dini terhadap potensi banjir. Sistem ini dirancang dengan pendekatan modular menggunakan mikrokontroler Arduino sebagai pusat pemrosesan, serta memanfaatkan teknologi sensor dan komunikasi data untuk memperoleh dan menyajikan informasi ketinggian air secara efisien.

Komponen utama dari sistem ini adalah *liquid level sensor module* berbasis transistor NPN S8050, yang bekerja dengan prinsip konduktivitas air. Ketika permukaan air menyentuh pin sensor, arus listrik akan mengalir dan mengaktifkan transistor di dalam modul. Aktivasi ini menghasilkan sinyal analog yang proporsional terhadap level air, dan sinyal tersebut kemudian dibaca oleh ADC (*Analog-to-Digital Converter*) pada Arduino.

Data yang diperoleh kemudian diproses oleh Arduino dan ditampilkan melalui serial monitor dalam bentuk informasi tekstual. Untuk memastikan efisiensi komunikasi antar perangkat, sistem ini menerapkan protokol SPI (*Serial Peripheral Interface*), di mana Arduino berperan sebagai *master* yang membaca data dari sensor, dan *slave* sebagai unit yang menampilkan hasil pengukuran secara serial. Pendekatan ini tidak hanya meningkatkan efisiensi pembacaan data, tetapi juga membuka peluang pengembangan sistem multipoint di masa mendatang.

Sistem SmartFlow juga dilengkapi dengan indikator LED untuk memberikan visualisasi status level air dalam tiga kategori yakni rendah, sedang, dan tinggi, sehingga pengguna dapat segera mengenali kondisi secara visual lewat indikator yang ada. Selain itu, digunakan *timer* internal untuk memperbarui data dalam interval waktu tertentu, memastikan pemantauan dilakukan secara periodik. Ketika level air mencapai ambang batas yang telah ditentukan, sistem akan *interrupt* akan *triggered* sebagai mekanisme peringatan otomatis, memungkinkan pengguna mengambil tindakan lebih cepat lewat mekanisme audio.

Secara keseluruhan, SmartFlow dirancang sebagai solusi yang sederhana, terjangkau, namun efektif untuk mendeteksi potensi banjir secara lebih dini. Sistem ini diharapkan dapat membantu masyarakat atau instansi terkait dalam upaya mitigasi bencana dengan menyediakan informasi yang cepat dan akurat. Kedepannya, sistem ini juga memiliki potensi untuk dikembangkan lebih lanjut, misalnya dengan menambahkan modul komunikasi nirkabel, tampilan UI/UX berbasis web atau aplikasi, serta integrasi ke dalam sistem peringatan bencana berskala luas.

1.3 OBJECTIVE

Tujuan dari proyek ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan sistem pemantauan level air berbasis mikrokontroler Arduino yang mampu membaca data level air secara *real-time* menggunakan sensor liquid level
2. Menggunakan modul sensor berbasis transistor NPN (S8050) untuk mendeteksi perubahan level air dan mengkonversinya menjadi sinyal analog yang dapat dibaca oleh ADC pada Arduino.
3. Menerapkan komunikasi data dengan protokol SPI, di mana Arduino bertindak sebagai *master* untuk membaca data sensor dan *slave* untuk menampilkan informasi secara serial.
4. Menyediakan indikator visual berbasis LED yang menunjukkan status level air (rendah, sedang, tinggi) agar pengguna dapat dengan cepat memahami kondisi lingkungan dan indikator audio lewat *buzzer* (saat level tinggi).
5. Mengintegrasikan fitur *interrupt* dan *timer* untuk memberikan peringatan secara berkala serta melakukan pembaruan data dalam interval waktu yang terjadwal.
6. Menghasilkan sistem peringatan banjir yang sederhana, murah, dan mudah diimplementasikan sebagai solusi awal mitigasi bencana banjir di lingkungan masyarakat.

1.4 ROLES AND RESPONSIBILITIES

Peran dan tanggung jawab masing-masing anggota kelompok adalah sebagai berikut:

Roles	Responsibilities	Person
Code designer, perancang rangkaian simulasi dan dokumentasi	● Membahas dan merancang ide proyek ● Melengkapi kode *master* dan membuat rancangan proteus ● Dokumentasi dan laporan	Wilman Saragih Sitio
Code designer, perancang rangkaian simulasi dan dokumentasi	● Membahas dan merancang ide proyek ● Membuat base *main code* untuk UART ● Pengembangan rangkaian pada Proteus ● Dokumentasi laporan	Xavier Daniswara
Code designer dan dokumentasi	● Membahas dan merancang ide proyek ● Membuat *base* main code untuk SPI ● Dokumentasi laporan	Siti Amalia Nurfaidah
Leader dan *hardware implementation*	● Memberikan ide dan merancang ide proyek ● Mengimplementasikan pada rangkaian fisik ● Dokumentasi	Jonathan Matius Weni Gerimu
Dokumentasi, PPT, dan README	● Dokumentasi laporan ● PPT ● README	Alfonsus Tanara Gultom

Table 1. Roles and Responsibilities

CHAPTER 2

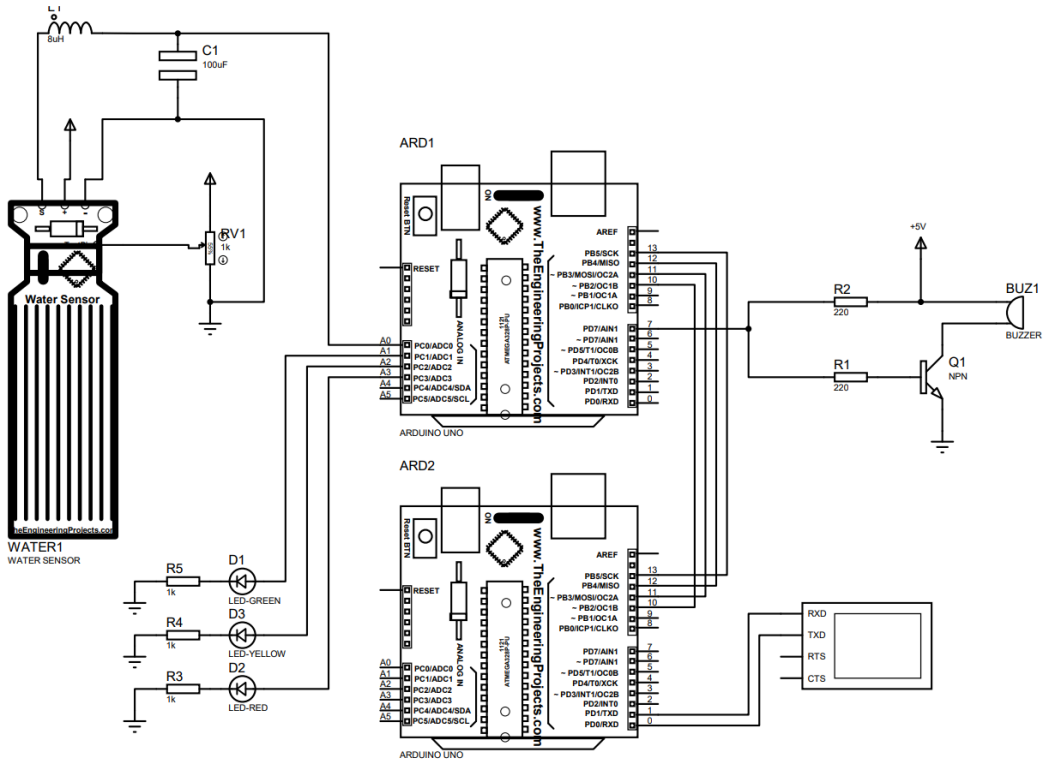
IMPLEMENTATION

2.1 EQUIPMENT

Tools yang digunakan dalam project ini adalah sebagai berikut:

- Visual Studio Code (IDE)
- Proteus
- Arduino IDE
- Github

2.2 HARDWARE DESIGN AND SCHEMATIC



Gambar 1. Implementasi Rangkaian

Proyek ini menggunakan dua buah Arduino, dengan Arduino pertama dikonfigurasikan sebagai *Master* dan Arduino kedua sebagai *Slave*, *liquid level sensor module*, rangkaian LED yang berwarna hijau, kuning, dan merah untuk memberi peringatan berdasarkan level ketinggian air tertentu, dan *buzzer*. *Liquid level sensor module* dihubungkan ke Port C pada PC0 sebagai input pada *Master* Arduino, lalu PC1 hingga PC3 digunakan sebagai *output* untuk memberikan peringatan visual menggunakan LED berwarna hijau, kuning, dan merah. Port D pada PD7 digunakan sebagai *output* untuk memberikan peringatan audio dengan *buzzer*. Port B dari PB2 hingga PB5 dihubungkan dari *Master* Arduino untuk berkomunikasi ke *Slave* Arduino dengan protokol SPI.

Desain dan implementasi dari proyek SmartFlow menggunakan komponen *liquid level sensor module* untuk mengukur ketinggian atau level dari air, ditambah dengan peringatan visual dari LED dan peringatan audio dari *buzzer*. Terdapat beberapa mode, di mana LED berwarna hijau akan menyala jika air menutupi 0–33% dari keseluruhan sensor liquid level, LED kuning akan menyala jika air menutupi 34–66% dari keseluruhan sensor, dan LED merah jika air menutupi 67% hingga menutupi keseluruhan sensor.

Proyek ini terdapat timer untuk membaca ulang sensor setiap beberapa detik untuk melakukan pembaharuan data secara otomatis. Selanjutnya juga terdapat interrupt yang akan memicu sistem peringatan ketika level air melebihi batas tertentu. Modul I2C/SPI juga digunakan untuk memfasilitasi komunikasi dari perangkat Master yang menerima data level ketinggian air dan Slave yang menampilkan data ke terminal serial monitor menggunakan protokol SPI.

Tujuan dari SmartFlow ini adalah untuk melakukan pemantauan level air dan

2.3 SOFTWARE DEVELOPMENT

Kode assembly ini ditulis untuk mikrokontroler AVR dan digunakan untuk mengimplementasikan Sistem Sensor Level Cairan yang membaca level ketinggian air menggunakan ADC, menampilkan level melalui LED, mengaktifkan alarm/buzzer saat *threshold* terlampaui, dan mengirim data ke perangkat *slave* melalui komunikasi SPI.

2.3.1 BAGIAN PERANGKAT MASTER

1. **Pengaturan Awal dan Inisialisasi**

   Pengaturan Awal dan Inisialisasi Kode ini ditulis dalam bahasa assembly AVR dan dieksekusi pada mikrokontroler AVR. Pada bagian awal kode diawali dengan mendefinisikan beberapa konfigurasi pin I/O dan menginisialisasi komponen-komponen yang diperlukan. *Vektor interrupt* diatur untuk vektor Reset yang nanti akan mengarahkan eksekusi ke fungsi main, dan alamat untuk vektor *interrupt* Timer1 *Compare Match* A yang mengarahkan eksekusi ke `TIMER1_COMPA_ISR`.

2. **Fungsi Main**

   Fungsi main dimulai dengan pengaturan beberapa pin sebagai *output* yakni mulai dari PC1, PC2, dan PC3 sebagai pin *output* untuk indikator LED level ketinggian air dari sensor dengan PD7 sebagai pin *output* untuk *buzzer* sebagai mekanisme alarm. Selanjutnya, program memanggil beberapa *subroutine* inisialisasi:

   - `SPI_Init` untuk mengatur komunikasi SPI sebagai *Master*
   - `ADC_Init` untuk mengatur pembacaan sensor level air
   - `Timer1_Init` untuk mengatur *timer* yang akan mengatur waktu pembacaan sensor
   - Pengaktifan *global interrupt* dengan perintah SEI

3. **Main Loop**

   Setelah inisialisasi, program masuk ke *loop* utama yaitu `main_loop`, yang akan berjalan terus-menerus selama sistem beroperasi. Di dalam *loop* ini terjadi beberapa proses:

   - Pembacaan nilai sensor level air menggunakan ADC dengan memanggil `Read_ADC`
   - Penskalaan hasil ADC dari 10-bit menjadi 8-bit untuk transmisi dengan `Scale_ADC_Result`
   - Pemeriksaan *threshold* alarm untuk level air dengan `Check_Alarm_Threshold`
   - Pembaruan indikator LED berdasarkan level air melalui `Update_LED_Indicators`

- Pengiriman nilai ADC yang telah diskalakan ke *slave* melalui SPI dengan `SPI_Transmit`

4. **Timer**

   Timer1 diatur dalam mode CTC dengan *prescaler* 1024. Ketika penghitung mencapai nilai *compare*, akan terjadi *interrupt* `TIMER1_COMPA_ISR`. Dalam ISR ini, status register disimpan dan *timer* di-reset. Setelah itu, status register dipulihkan dan program kembali dari interrupt dengan RETI, memungkinkan `main_loop` memproses pembacaan ADC berikutnya secara berkala. Kemudian *global interrupt* diaktifkan dengan perintah SEI.

5. **Pemrosesan ADC**

   Pembacaan sensor level air dilakukan melalui ADC dengan beberapa tahap:

   - ADC diinisialisasi untuk membaca dari *channel* 0 atau PC0 dengan tegangan referensi AVCC
   - Pada fungsi `Read_ADC`, konversi dimulai dan program menunggu hingga konversi selesai
   - Hasil 10-bit diambil untuk diproses
   - Hasil diskalakan menjadi 8-bit untuk memudahkan transmisi dan pemrosesan

6. **Monitoring**

   Berdasarkan hasil pembacaan ADC, *subroutine* akan membandingkan nilai diskalakan dengan threshold maksimum. Jika nilai melebihi batas, *buzzer* diaktifkan pada pin PD7. Selanjutnya, akan menyalakan salah satu LED pada PC1 - PC3 sesuai kategori level air:

   - LED PC1 untuk level rendah (<85)
   - LED PC2 untuk level sedang (85–169)
   - LED PC3 untuk level tinggi (≥170)

   Dengan mekanisme tersebut, sistem memberikan *feedback* visual dan audio instan berdasarkan kondisi level ketinggian.

7. **SPI**

   Sistem menggunakan protokol SPI untuk mengirimkan data ke perangkat *slave*. Program menginisialisasi SPI sebagai *master* dengan menyetel pin MOSI (PB3), SCK (PB5), dan SS (PB2) sebagai *output*. Ketika mengirim data, `SPI_Transmit` *slave* diaktifkan dengan SS *low*, kemudian *write* nilai level yang sudah diskalakan ke register SPDR kemudian menunggu hingga bit SPIF di SPSR menandakan transmisi selesai, dan setelah itu mengembalikan SS ke level tinggi untuk menonaktifkan *slave*.

2.3.1 BAGIAN PERANGKAT SLAVE

1. **Pengaturan Awal dan Inisialisasi**

   Kode perangkat *slave* ditulis dalam bahasa assembly AVR dan dieksekusi pada mikrokontroler yang berperan sebagai penerima data dari *master*. Program diawali dengan pemanggilan *subroutine* `UART_Init` untuk konfigurasi komunikasi serial dan `Send_Welcome_Message` untuk menampilkan pesan pembuka di terminal. Selanjutnya, *subroutine* `SPI_Slave_Init` dijalankan untuk mengatur pin yang digunakan SPI sebagai *slave* dan mengaktifkan modul SPI. Setelah itu, sistem masuk ke *loop* utama yaitu `slave_loop` untuk terus-menerus menerima dan memproses data dari *master*.

2. **Inisialisasi SPI sebagai Slave**

   Inisialisasi SPI dilakukan dengan mengatur pin PB4 (MISO) sebagai *output* agar dapat mengirim data ke *master*. Bit SPE pada register SPCR diset untuk mengaktifkan SPI dalam mode *slave*. Status awal SPI dibaca dari register SPSR dan SPDR, meskipun hanya untuk sinkronisasi awal. Setelah konfigurasi ini, *slave* siap menerima data dari *master* kapanpun tersedia.

3. **Loop Utama Slave**

   Dalam `slave_loop`, dua *subroutine* utama dipanggil secara berulang, yaitu `SPI_Receive` dan `Process_Data`. *Subroutine* `SPI_Receive` akan memeriksa apakah ada data masuk dari *master* dengan melihat bit SPIF pada register SPSR.

Bila ada data baru, nilai dari SPDR akan disimpan ke register R18, dan flag R20 di-set menandakan data baru tersedia untuk diproses.

4. **Pemrosesan Data**

   Jika flag R20 menunjukkan ada data baru, maka *subroutine* `Process_Data` akan menyalin data tersebut ke register yaitu R19. Kemudian akan memanggil `Update_Terminal_Display` untuk menampilkan informasi level air ke terminal. Dalam tampilan terminal, pesan deskriptif dan nilai ketinggian air akan ditampilkan dalam format persentase, lengkap dengan status level (rendah, sedang, atau tinggi) berdasarkan nilai yang diterima.

5. **Tampilan Level Air di Terminal**

   Fungsi `Update_Terminal_Display` bertugas menampilkan informasi level air pada terminal. Proses dimulai dengan menampilkan label *Water Level*, lalu memanggil subrutin `Send_Decimal_Value` untuk mengubah nilai 8-bit menjadi format persentase. Setelah itu, subrutin `Send_Status_Message` akan menampilkan status level air berdasarkan rentang nilai berikut:

   - Jika nilai < 33 → *Low Level*
   - Jika nilai 33–65 → *Medium Level*
   - Jika nilai > 66 → *High Level*

   Tampilan kemudian diakhiri dengan karakter *carriage return* dan *line feed* sebagai penanda akhir baris.

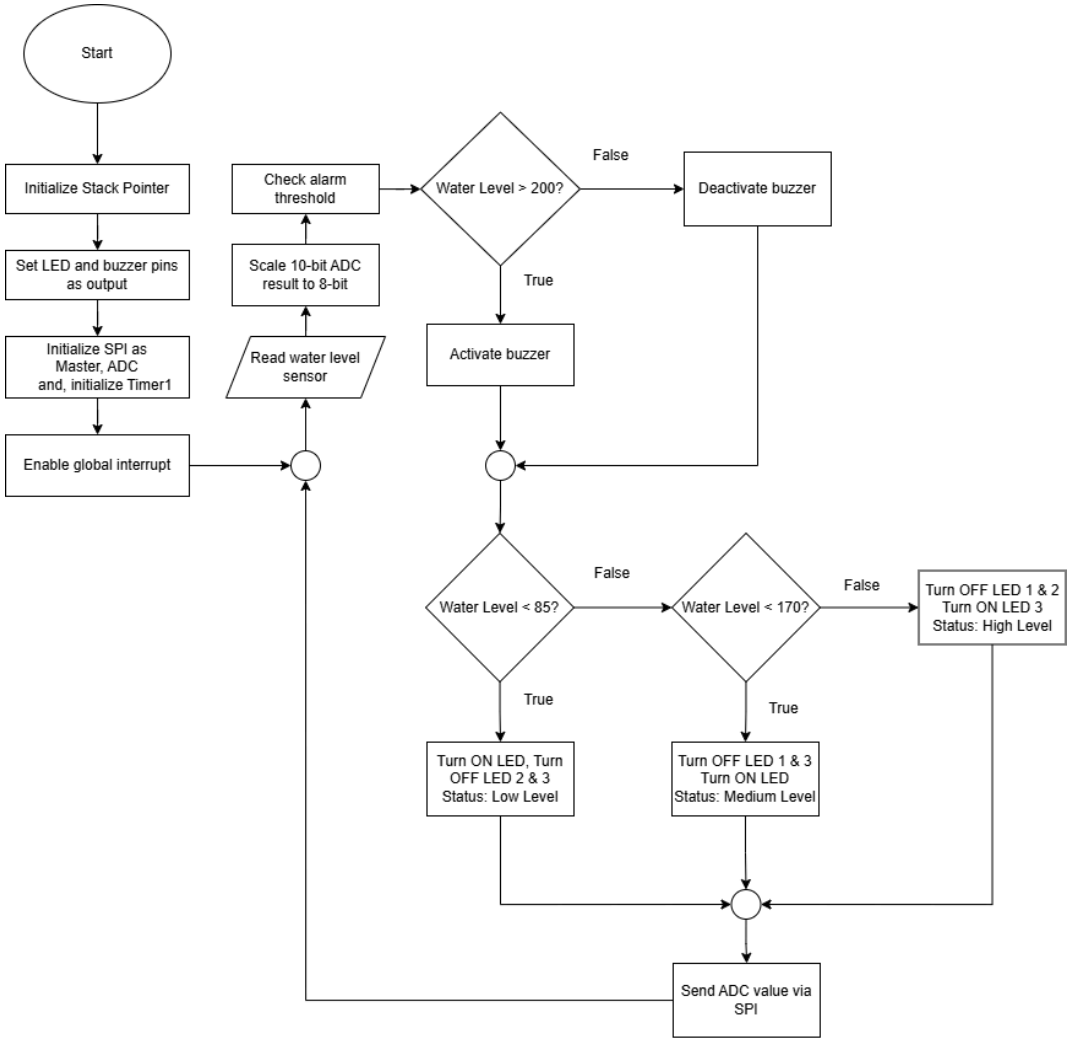
6. **Konversi Desimal ke Persentase**

   Konversi nilai dari register R19 ke format persentase dilakukan dengan mengalikan nilai tersebut dengan 100, lalu membaginya kembali sesuai skala maksimum, misalnya 255. Nilai hasil konversi kemudian dikirim satu per satu sebagai karakter ASCII ke terminal seperti ratusan, puluhan, satuan, dan simbol persen. Fungsi ini memastikan bahwa level air dapat dibaca secara intuitif oleh user dalam bentuk persentase.

7. **Komunikasi UART**

Seluruh komunikasi tampilan ke terminal dilakukan melalui UART. Inisialisasi UART dilakukan dengan mengatur *baud rate* 9600 melalui register UBRR0L, mengaktifkan *transmitter* dengan bit TXEN0 pada UCSR0B, serta memilih mode data 8-bit dengan *setting* pada UCSR0C. Seluruh karakter dikirim melalui fungsi `Send_Char` yang menunggu hingga *buffer* transmisi siap sebelum mengirim karakter ke UDR0. Ini menjamin komunikasi serial yang stabil dan sinkron dengan terminal.

Langkah-langkah yang dijalankan dan diimplementasikan dalam program ini adalah sebagai berikut:

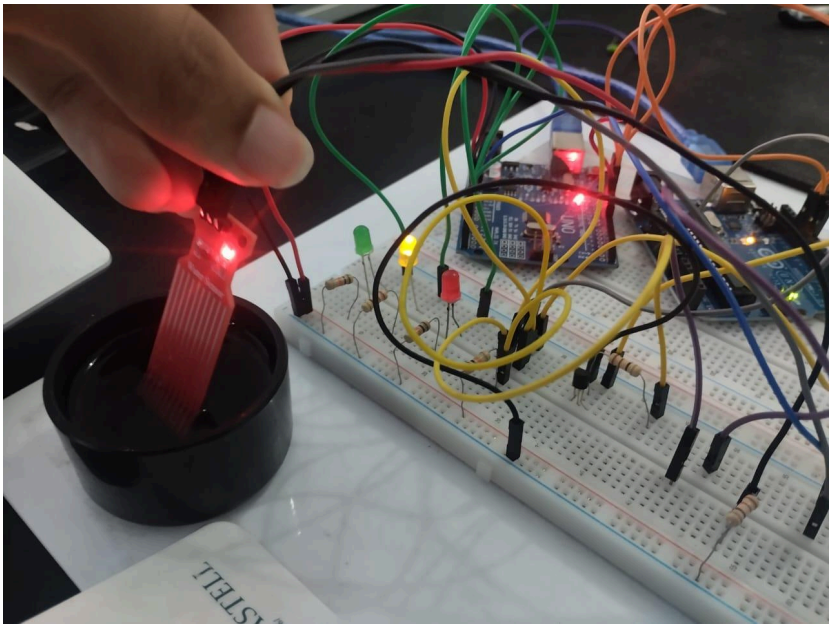


Gambar 2. Flowchart *master device*

2.3 HARDWARE AND SOFTWARE INTEGRATION

Sistem SmartFlow mengintegrasikan dua mikrokontroler Arduino yang dikonfigurasi sebagai *Master* dan *Slave*. Komponen utama dalam sistem ini meliputi *liquid level sensor module*, LED indikator berwarna hijau, kuning, dan merah, serta *buzzer* sebagai alarm. *Liquid level sensor* terhubung ke port input pada Arduino *Master*, sedangkan LED dan *buzzer* digunakan sebagai *output* visual dan auditori. Untuk komunikasi antarmikrokontroler, digunakan protokol SPI yang menghubungkan *Master* ke *Slave* melalui pin PB2 hingga PB5. Desain ini memungkinkan sistem untuk mendeteksi dan mengklasifikasikan level ketinggian air, serta memberikan respons *real-time* berdasarkan level air yang terdeteksi.

Di sisi *software*, program ditulis menggunakan bahasa assembly untuk mikrokontroler AVR. Perangkat lunak pada Arduino *Master* menangani inisialisasi ADC untuk pembacaan sensor, konfigurasi SPI sebagai *Master*, serta pengaturan *timer* untuk pembacaan berkala. Data hasil ADC yang sudah diskalakan akan dibandingkan dengan *threshold* untuk menentukan level air, mengaktifkan LED indikator, dan mengirim data ke *Slave* melalui SPI. *Software* pada Arduino *Slave* kemudian menerima data tersebut dan menampilkannya secara deskriptif di terminal melalui komunikasi UART, termasuk nilai persentase dan status level air (*Low, Medium, High*). Integrasi antara komponen *hardware* dan *software* ini memungkinkan sistem bekerja secara otomatis, efisien, dan responsif terhadap perubahan level air, sehingga memberikan solusi *monitoring* yang efektif.



Gambar 3. Implementasi Rangkaian Fisik

Implementasi hardware Sistem Sensor Level Cairan memanfaatkan dua mikrokontroler AVR yang terpasang pada breadboard sebagai media perangkaian. Unit Master berperan sebagai pengendali utama dengan modul sensor level air yang terhubung ke ADC0 untuk pembacaan analog. Master juga dilengkapi tiga LED indikator yang terhubung ke PC1 (hijau), PC2 (kuning), dan PC3 (merah) untuk menunjukkan level cairan, serta buzzer pada PD7 sebagai alarm ketika level cairan melebihi batas aman. Komunikasi antar mikrokontroler menggunakan protokol SPI dengan MOSI (PB3), SCK (PB5), dan SS (PB2) yang menghubungkan Master ke Slave.

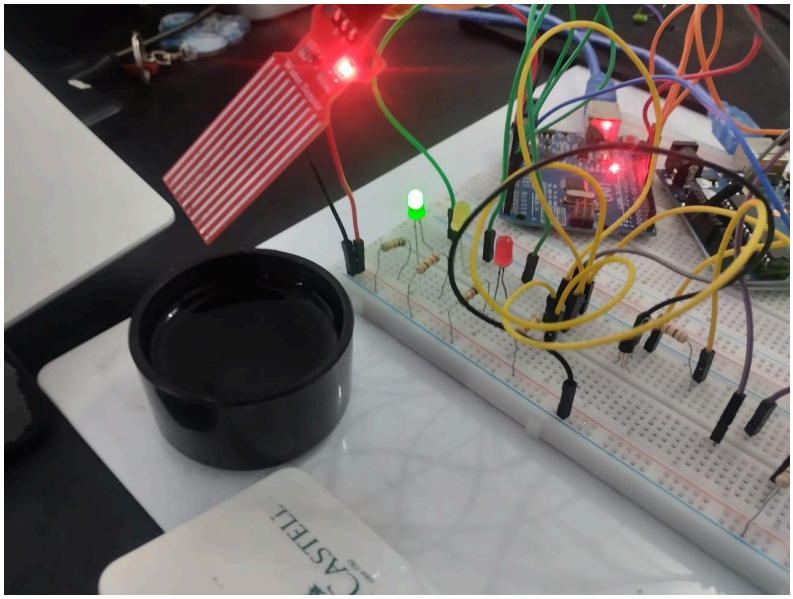
Unit Slave berfungsi sebagai monitor yang menerima data level cairan dari Master melalui SPI dan menampilkannya dalam bentuk persentase melalui komunikasi UART ke komputer. Koneksi UART dilakukan melalui pin TX dan RX yang dihubungkan ke terminal komputer menggunakan konverter UART-USB. Dengan rangkaian pada breadboard dimungkinkan sistem bekerja secara terintegrasi, di mana Master melakukan pembacaan sensor dan menampilkan indikasi visual dan auditori, sementara Slave mengonversi data untuk memberi informasi numerik akurat tentang level cairan kepada *user*. Sistem ini memberikan monitoring yang jelas dengan kombinasi indikator visual, peringatan suara, dan *interface* pengguna melalui terminal serial.

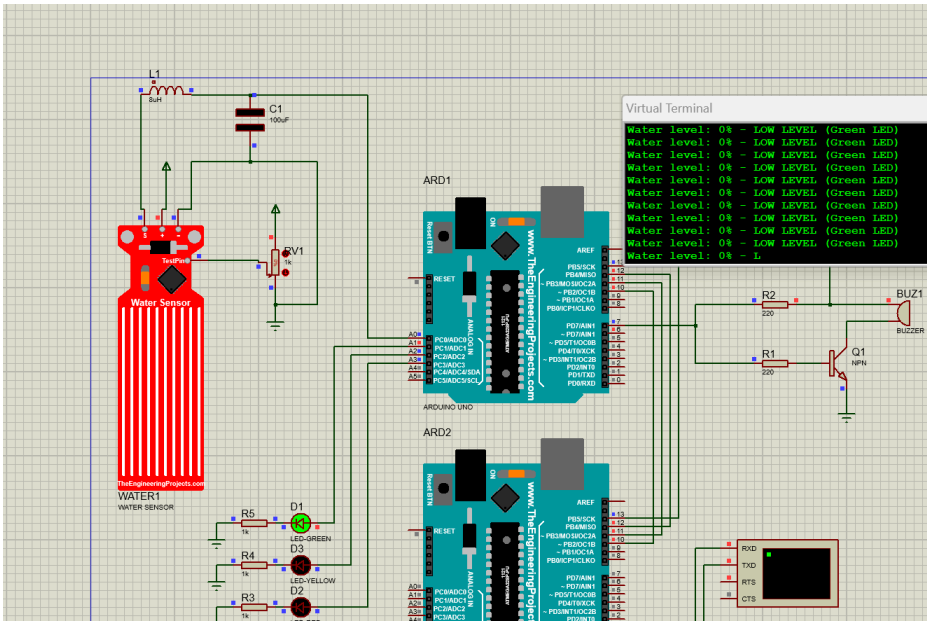
CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

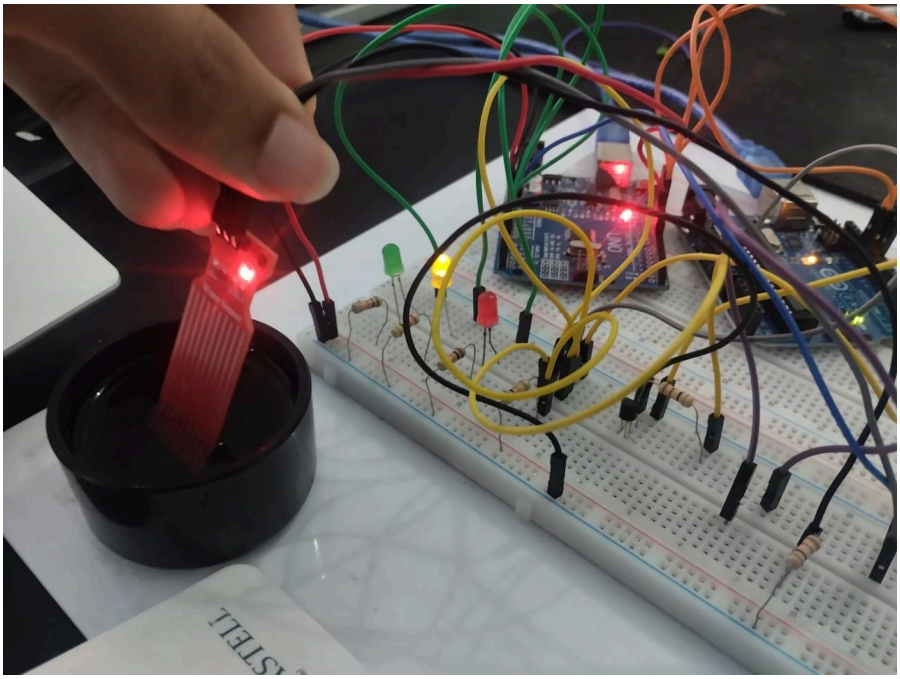
Level ketinggian rendah:

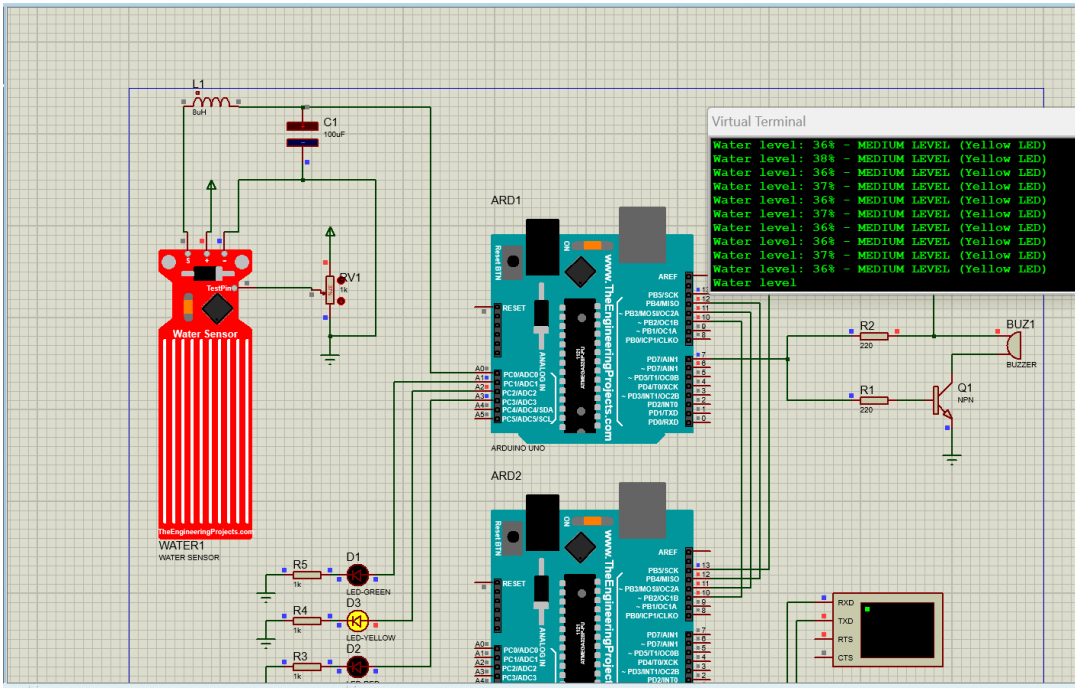




Gambar 4. Testing pada level ketinggian low

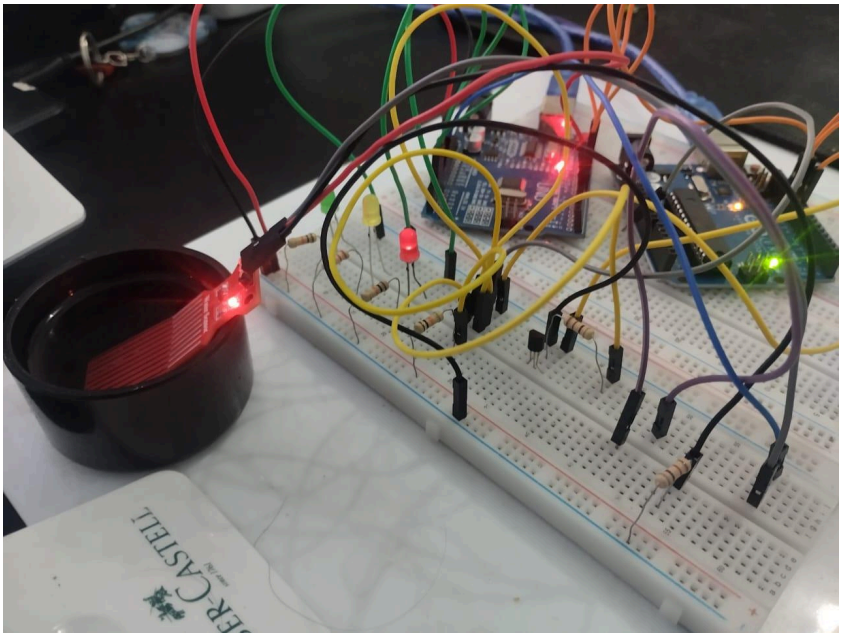
Level ketinggian sedang:

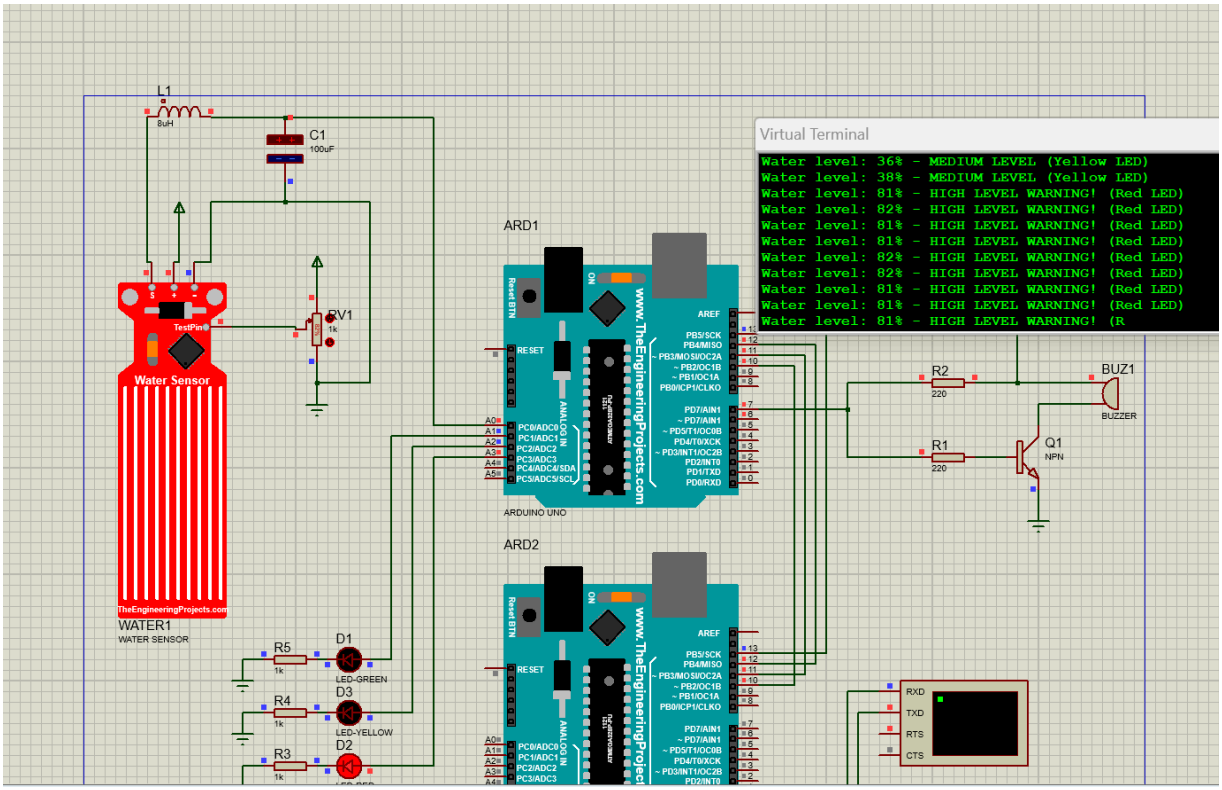




Gambar 5. Testing pada level ketinggian medium

Level ketinggian tinggi/maksimum:





Gambar 6. Testing pada level ketinggian high

3.2 RESULT

Hasil simulasi pada software Proteus menunjukkan bahwa sistem SmartFlow bekerja sesuai dengan batasan persentase level air yang telah ditentukan. Pengujian dilakukan menggunakan test pin yang dikendalikan melalui potensiometer, dengan simulasi nilai sebesar 0% untuk kategori Low, 37% untuk Medium, dan 82% untuk High. Dalam simulasi tersebut, LED indikator menyala sesuai dengan level air yang diatur melalui potensiometer. Sistem juga menyalakan LED dengan tepat berdasarkan nilai yang dimasukkan dan serial monitor yang disimulasikan pada Proteus juga berhasil menampilkan persentase level air dengan benar sesuai dengan nilai input yang diuji.

Pada pengujian rangkaian fisik, sistem juga menunjukkan hasil yang sesuai dalam segi visual dan suara. Sensor level air yang dimasukkan ke dalam air mampu memicu LED indikator (hijau, kuning, dan merah) secara akurat berdasarkan kedalaman sensor di dalam air. LED menyala sesuai dengan batasan level air yang telah ditetapkan. Namun, pada tampilan serial monitor melalui Arduino IDE, nilai persentase level air yang ditampilkan belum cukup akurat dengan kedalaman sebenarnya dari sensor saat dimasukkan ke dalam air.

3.3 ANALYSIS AND EVALUATION

Berdasarkan hasil pengujian, dapat dianalisis bahwa ketidaksesuaian nilai persentase level air pada serial monitor disebabkan oleh proses konversi data yang berjalan pada kode program di Arduino Slave (file Slave.S). Data level air yang diterima dari Master berupa nilai digital dengan rentang maksimal 255, sedangkan output sensor level air analog sebenarnya dapat mencapai nilai sekitar 440. Oleh karena itu, diperlukan proses pemetaan atau konversi agar nilai dari sensor tersebut dapat direpresentasikan dalam rentang 0 hingga 100 persen. Dalam implementasinya, pembuatan kode konversi ini menghadapi kesulitan untuk menghasilkan nilai persentase yang akurat dan sesuai dengan kondisi nyata sensor. Kesulitan ini membuat nilai persentase yang tampil pada serial monitor belum sepenuhnya mencerminkan kedalaman air yang sebenarnya meskipun indikator LED pada Master telah berfungsi dengan baik sesuai batasan level air.

CHAPTER 4

CONCLUSION

SmartFlow berhasil dirancang dan diimplementasikan dengan memanfaatkan dua mikrokontroler Arduino yang dikonfigurasi sebagai Master dan Slave. Pada sistem ini, Arduino Master berfungsi sebagai pengendali utama yang melakukan pembacaan sensor level air melalui ADC dan mengatur indikator LED berwarna hijau, kuning, dan merah sesuai dengan level air yang terdeteksi. Selain itu, Master juga mengendalikan buzzer sebagai alarm yang aktif ketika level ketinggian melebihi batas aman. Pengaturan pin-pin pada Arduino dilakukan dengan tepat untuk mendukung fungsi input dan output, termasuk komunikasi SPI yang menghubungkan Master dengan Slave. Pada sisi Slave, mikrokontroler menerima data level air dari Master melalui protokol SPI, kemudian menampilkan nilai persentase dan status level air secara deskriptif pada serial monitor menggunakan komunikasi UART.

Penggunaan interrupt dan timer pada Master memungkinkan pembacaan sensor dan pengiriman data dilakukan secara berkala dan responsif terhadap perubahan level air. Sistem ini mampu bekerja secara otomatis dengan integrasi hardware dan software yang mendukung monitoring level air secara real-time melalui indikator visual, alarm suara, dan tampilan numerik di terminal komputer. Meski demikian, masih terdapat kendala pada proses konversi nilai analog sensor menjadi persentase yang tepat di program Slave, sehingga nilai persentase yang tampil belum sepenuhnya akurat. Namun secara keseluruhan, sistem telah berhasil menjalankan fungsi utama monitoring level air dengan baik.

Kedepannya, penggunaan sensor dengan rentang pembacaan yang lebih besar dan bahan sensor yang lebih tahan kondisi lingkungan akan memungkinkan implementasi SmartFlow pada skala lebih luas, khususnya untuk monitoring di daerah rawan bencana. Dengan pengembangan tersebut, sistem ini dapat menjadi solusi efektif dalam memberikan peringatan dini dan pengawasan level air secara real-time guna mendukung keselamatan masyarakat.

REFERENCES

[1] jameswilson, “Water Sensor Library For Proteus - The Engineering Projects,” The Engineering Projects, Jul. 13, 2020. https://www.theengineeringprojects.com/2020/07/water-sensor-library-for-proteus.html (accessed May 19, 2025).

[2] Last Minute Engineers, “In-Depth: How Water Level Sensor Works and Interface it with Arduino,” Last Minute Engineers, Nov. 29, 2019. https://lastminuteengineers.com/water-level-sensor-arduino-tutorial/ (accessed May 19, 2025).

[3] Adminninjatech, “SIMPLE WATER LEVEL INDICATOR in Atmega32 using ATMEL STUDIO & PROTEUS,” Ninja Tech, Dec. 16, 2022. https://ninjatech.live/simple-water-level-indicator-in-atmega32-using-atmel-studio-proteus (accessed May 19, 2025).

[4] “How to Build a Liquid Level Sensor Circuit with an AVR,” Learningaboutelectronics.com, 2025. https://www.learningaboutelectronics.com/Articles/AVR-liquid-level-sensor-circuit.php (accessed May 19, 2025).