REAL TIME SYSTEM AND INTERNET OF THINGS FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

WATER WAKER

GROUP IP-X/P-X/R-X

RAKHA ARGYA ZAHRAN	**2106733925**
ALIEFYA FIKRI IHSANI	**2106733843**
GEMILANG BAGAS RAMADHANI	**2006535205**
AZZAH AZKIYAH ANGELI S	**2106731390**

PREFACE

In an era where mornings are defined by hectic schedules and relentless busyness, the concept of a gentle, yet effective, wake-up call has become an aspiration for many. Thus, we set out to create an innovative solution that incorporates a distinctive feature, which is **the water spraying mechanism.** Unlike regular alarms, the “Water Waker” isn't just about sound. It's a clever system that connects to the internet to spray water when the alarm goes off. Think of it as having a smart assistant for waking up, but instead of a beep, it uses water to gently get you from sleep.

This report is made to explain the processes for making Water Waker and its implementation on IoT components as well as the integration of its settings and handles through Arduino Cloud. Our aspiration with the “Water Waker” project is to demonstrate the transformative potential of IoT in daily activities, especially in redefining morning wake-up. Throughout this report, we will delve into the intricate mechanisms and technical aspects of this innovative system, illustrating how it promises to revolutionize the way we start our day.

Depok, December 10, 2023

Group IP-X/P-X/R-X

TABLE OF CONTENTS

CHAPTER 1 ..... **3**
INTRODUCTION **3**
1.1 PROBLEM STATEMENT 4
1.2 PROPOSED SOLUTION 4
1.3 ACCEPTANCE CRITERIA 5
1.4 ROLES AND RESPONSIBILITIES 5
1.5 TIMELINE AND MILESTONES 5
CHAPTER 2 ..... **6**
IMPLEMENTATION **6**
2.1 HARDWARE DESIGN AND SCHEMATIC 7
2.2 SOFTWARE DEVELOPMENT 7
2.3 HARDWARE AND SOFTWARE INTEGRATION 8
CHAPTER 3 ..... **8**
TESTING AND EVALUATION **9**
3.1 TESTING 9
3.2 RESULT 9
3.3 EVALUATION 10
CHAPTER 4 ..... **11**
CONCLUSION **11**

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Many college students experience issues with waking up on time in the mornings, leading to lateness for class. The snooze function on many alarm clocks contributes to this problem. Students often hit snooze repeatedly, falling back asleep for hours past their intended wake up time. A more effective solution is needed to help students wake up on time.

1.2 PROPOSED SOLUTION

The proposed solution is the Water Waker - an alarm clock supplemented with a water spraying function to help wake up users. Taking inspiration from the refreshing sensation of splashing water on one's face in the morning, the Water Waker will spray the user's face with a brisk burst of water when the alarm goes off. This leverages the shocking and alerting effect of sudden water exposure to reliably rouse people from sleep.

The Water Waker consists of a microcontroller to control system logic, a beeper for audio alarms, a water pump and nozzle for water spraying, and a mini tank to hold water. When activated in the morning, it sprays a quick shot of water to the user's face to induce a startle response that fully wakes them up.

1.3 ACCEPTANCE CRITERIA

The acceptance criteria of this project are as follows:

1. Ability to wake users up through a combination of alarm noise and water stimulus
2. Water reaches user's face in bed
3. User-friendly interface to set alarms
4. Durable enclosure to protect components
5. Portable and battery-powered
6. Affordable components

Meeting these criteria will validate the Water Waker's effectiveness as an alarm clock for heavy sleepers. The wake-up reliability, intuitive controls, protective housing, portability, and low cost provide key indicators of a successful and usable product.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Role 1	Completing codes, Circuits, Purchase Components.	Rakha Argya Zahran
Role 2	Alarm Timer Codes, Circuits, Purchase Components, and Reports.	Aliefya Fikri Ihsani
Role 3	Circuit Implementation, Blynk Integration, Alarm Timer Codes.	Gemilang Bagas Ramadhani
Role 4	Blynk Integration Trial, Reports.	Azzah Azkiyah Angeli Syahwa

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

Insert Gantt Chart here. The Gantt Chart should consist of date interval for:

No	Task	Day	December				
			I	II	III	IV	V
		M		4	11	18	25
		T		5	12	19	26
		W		6	13	20	27
		TH		7	14	21	28
		F	1	8	15	22	29
		S	2	9	16	23	30
		S	3	10	17	24	31
1	Hardware Design completion						
2	Software Development						
3	Integration and Testing of Hardware and Software						
4	Final Product Assembly and Testing						

Fig 1. Gantt Chart

a) Hardware Design completion: A milestone indicating the date when the hardware design for the embedded system is finalized, including schematic.
b) Software Development: The date when the development of the user-created assembly code (software) begins, focusing on specific tasks and functionalities.
c) Integration and Testing of Hardware and Software: A milestone indicating when the hardware and software components are integrated and tested together to ensure proper functionality.
d) Final Product Assembly and Testing: A milestone marking when the final system product is assembled, tested, and verified to meet the acceptance criteria.

CHAPTER 2

IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC

The hardware design process of our group consists of several stages. At the initial stage, our group designed the schematic mock up on the Wokwi site. The components are ESP32, buzzer, servo, and LCD 1602a with I2C module attached to it.

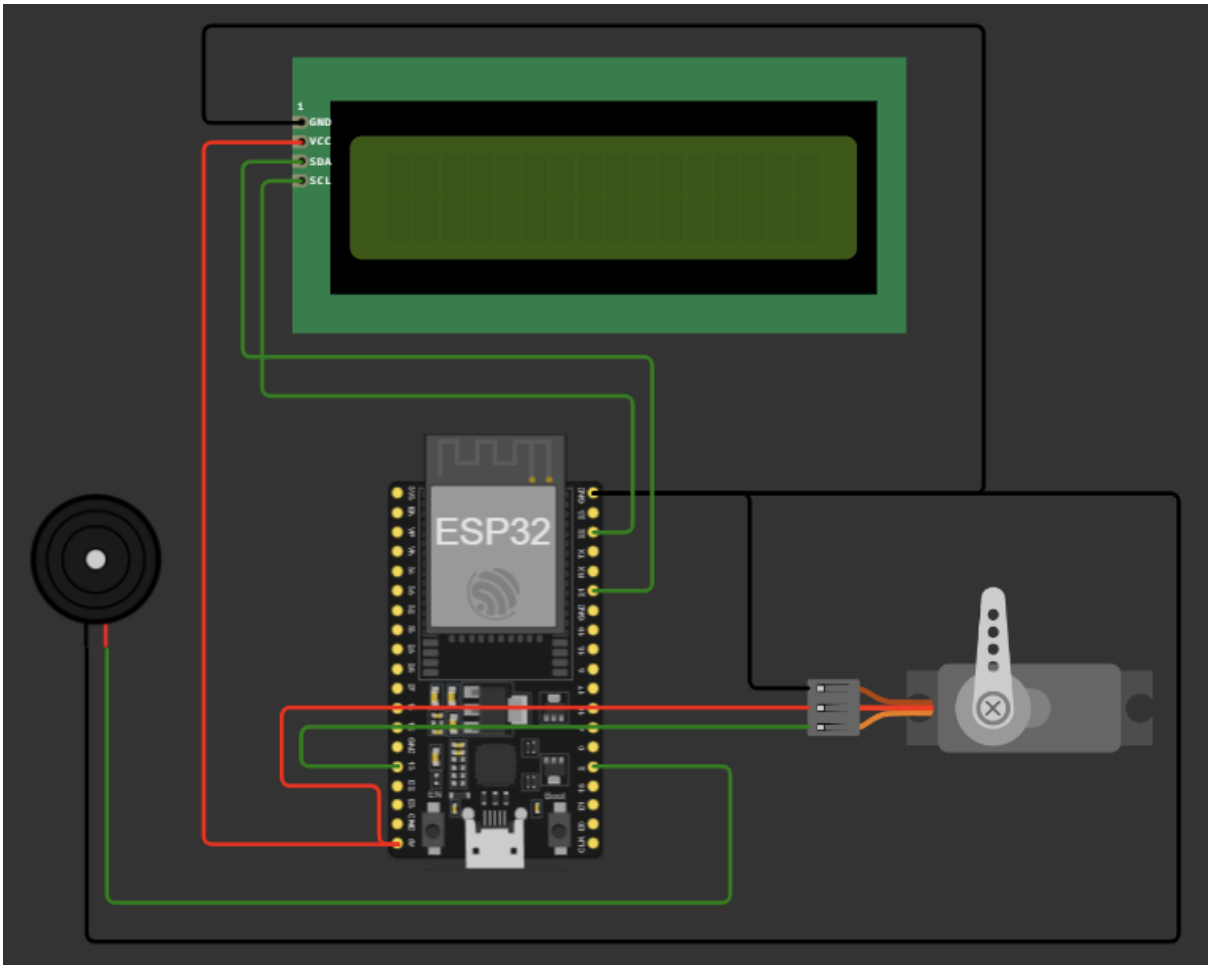


Fig 2. Wokwi Prototype

As visualized in the schematic/wiring diagram above, our prototype uses four signal pins, and two power pins. Pin D2 is connected to a buzzer to trigger it when the event occurs. Pin D21 and D22, works as medium to transmit SDA (Serial Data) and SCL (Serial Clock), respectively, in an I2C communication between ESP32 and 1602a LCD. Last, Pin D13 works to transmit PWM signal to the servo, so that servo can pull the spray trigger.

2.2 SOFTWARE DEVELOPMENT

The software development phase of the Water Waker project involved the creation of a program that would control the hardware components of the Water Waker, including the microcontroller, the beeper, and the water pump. The software was developed using the Arduino programming language, which is based on C/C++. The program was designed to interact with the hardware components, controlling when the beeper would sound and when the water pump would activate to spray water.

The software development process began with the creation of a flowchart to outline the logic of the program. This flowchart detailed the sequence of events that would occur when the alarm was set to go off, including the activation of the beeper and the water pump.

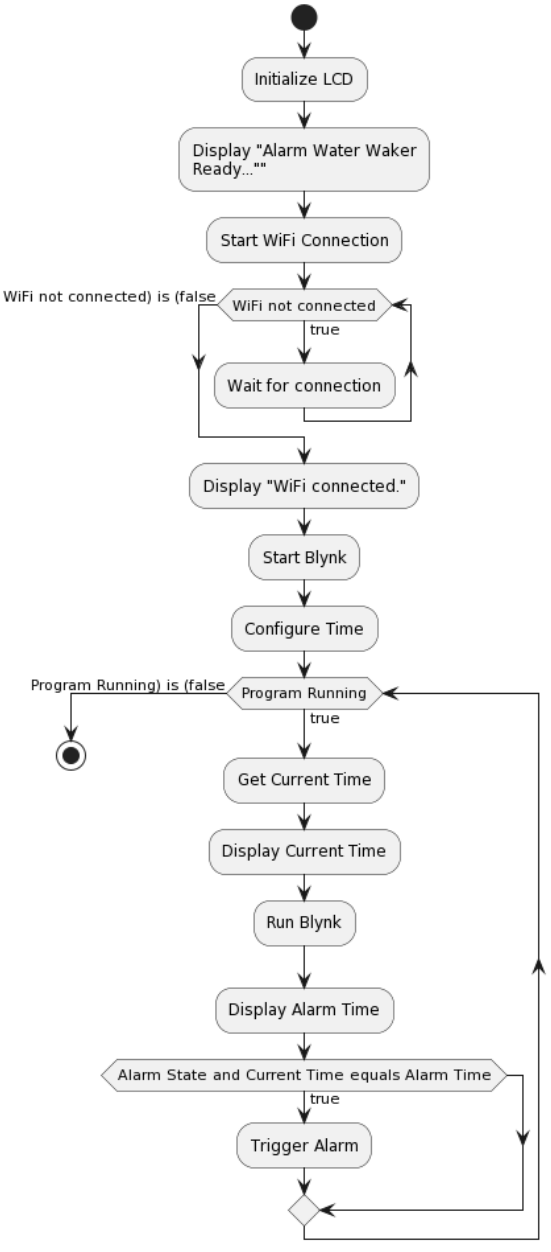


Fig 3. Flowchart

Once the flowchart was completed, the coding process began. The code was written to follow the logic outlined in the flowchart, ensuring that the hardware components would function as intended. The code was then tested and debugged to ensure its functionality and reliability. The software development phase also included the creation of a user-friendly interface for setting the alarm. This interface was designed to be intuitive and easy to use, allowing users to easily set the time for the alarm to go off.

```
#define BLYNK_TEMPLATE_ID "TMPL6Updquyaz"
#define BLYNK_TEMPLATE_NAME "FINPRO"
#define BLYNK_AUTH_TOKEN "Azv3PAq5DkeLW9R-4rqmZ6OnQqAaxCP9"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h> // Library for servo control

const char* ssid = "Wokwi-GUEST";
const char* password = "";
const int address = 0x27; // LCD I2C address
LiquidCrystal_I2C lcd(address, 16, 2); // Initialize LCD
String timeStamp;
String formattedDate;
String dayStamp;
Servo servo; // Initialize servo
char auth[] = BLYNK_AUTH_TOKEN;
const int servoPin = 13; // Servo motor pin
const int buzzerPin = 2; // Buzzer pin
const long SPRAY_DURATION = 2000; // 2 seconds
#define VIRTUAL_PIN_HOUR V1
#define VIRTUAL_PIN_MINUTE V2
#define VIRTUAL_PIN_alarmHour V3
#define VIRTUAL_PIN_alarmMinute V4
#define VIRTUAL_PIN_alarm_On_Off V5

const long timerInterval = 1000;
int alarmState = 1;
int alarmMinute;
```

```
int alarmHour;
int hour;
int minute;
int repeat;

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

void setup() {
  lcd.init(); // Initialize LCD
  lcd.backlight(); // Turn on backlight
  lcd.setCursor(0, 0); // Set cursor position
  lcd.print("Alarm Water Waker"); // Display text
  lcd.setCursor(0, 1);
  lcd.print("Ready...");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0, 0); // Set cursor position
  lcd.print("Connecting to "); // Display text
  lcd.setCursor(0, 1);
  lcd.print(ssid);
  delay(1000);
  lcd.clear();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    lcd.setCursor(0, 1);
    lcd.print(".");
  }
  lcd.clear();
  // Print local IP address and start web server
  delay(1000);
  lcd.setCursor(0, 0);
  lcd.print("WiFi connected.");
  lcd.setCursor(0, 1);
  lcd.print("IP address: ");
  lcd.print(WiFi.localIP());
  lcd.clear();
// Initialize a NTPClient to get time
  timeClient.begin();
  timeClient.setTimeOffset(25200);
  Blynk.begin(auth, ssid, password);
}
```

```
void loop() {
  while(!timeClient.update()) {
    timeClient.forceUpdate();
  }

  formattedDate = timeClient.getFormattedDate();

  int splitT = formattedDate.indexOf("T");
              timeStamp       =       formattedDate.substring(splitT+1,
formattedDate.length()-1);

  dayStamp = formattedDate.substring(0, splitT);
  // Extract hour and minute from timeStamp string
  String hourStr = timeStamp.substring(0, 2);
  String minuteStr = timeStamp.substring(3, 5);
  // Convert hour and minute to integers
  hour = hourStr.toInt();
  minute = minuteStr.toInt();

  // Display hour and minute on LCD with period in between
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(dayStamp);
  lcd.setCursor(6, 1);
  lcd.print(hour);
  lcd.print(".");
  lcd.print(minute);
  Blynk.run();
  Blynk.virtualWrite(VIRTUAL_PIN_HOUR, hour);
  Blynk.virtualWrite(VIRTUAL_PIN_MINUTE, minute);
  if (alarmState && hour == alarmHour && minute == alarmMinute) {
    for(int repeat = 0; repeat < 6; repeat++){
      lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Wake Up!");
  lcd.setCursor(0, 1);
  lcd.print(hour);
  lcd.print(minute);
  servo.attach(servoPin); // Attach servo to pin
  servo.write(180);
  delay(SPRAY_DURATION);
  servo.detach();
```

```
  // Play the sound after the servo movement completes
  tone(buzzerPin, 1000);
  delay(SPRAY_DURATION);
  noTone(buzzerPin);
  delay(SPRAY_DURATION);
    }
  }
  delay(1000);
}


BLYNK_WRITE(VIRTUAL_PIN_alarmHour){
  int pinValue = param.asInt();
  alarmHour = pinValue;
}
BLYNK_WRITE(VIRTUAL_PIN_alarmMinute){
  int pinValue = param.asInt();
  alarmMinute = pinValue;
}
BLYNK_WRITE(VIRTUAL_PIN_alarm_On_Off){
  int pinValue = param.asInt();
  if (pinValue == 1){
    alarmState = 1;
  } else {
    alarmState = 0;
  }
}
```

2.3 HARDWARE AND SOFTWARE INTEGRATION

The integration of the hardware and software components was a critical next step after the completion of the individual designs. The purpose of the integration phase was to combine the physical electronic components and the control software in order to test the full system functionality. The integration process began by first loading the Arduino program onto the ESP32 microcontroller. This allowed the ESP32 to run the software logic and control the operation of the attached hardware components.

With the software loaded, each hardware component was then connected to the ESP32 via the designed pin connections outlined in the schematic diagram. This included the wiring of the buzzer to Pin D2, the LCD screen via I2C to Pins D21 and D22, the servo motor to Pin D13, and the power rails to the specified pins. Once wired, debugging and troubleshooting of the system began to identify and resolve any integration issues. This involved verifying the correct transmission of signals between the ESP32 and each component, tweaking timing configurations, and testing connectivity across all parts of the system.

Upon full integration without errors, overall validation testing was conducted. This included running through various usage flow scenarios such as setting alarm times, activating alarms, checking system responses, and measuring functionality metrics. Tests were repeated iteratively as issues were incrementally resolved. The completion of the integration phase resulted in a fully assembled Water Waker prototype, with the software successfully interfaced to drive the hardware components in unison. This allowed testing and refinement of the complete alarm system to meet the final performance and reliability criteria.

With the foundation of the integrated system now established, work shifted to finalizing the enclosure, battery, and portability elements to produce the complete end-user product.

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

The functionality testing of the Water Waker project aimed to ensure that the system met the acceptance criteria outlined in the project proposal. This involved testing the alarm noise, the water spraying mechanism, and the user interface for setting alarms.

3.1.1 Alarm Noise Testing

The alarm noise was tested by setting the alarm and verifying that the noise was loud enough to wake a person. The beeper was connected to Pin D2 on the ESP32 microcontroller, and the alarm noise was activated using the Arduino programming language. The volume and duration of the alarm noise were adjusted to ensure that it was effective in waking up users.

3.1.2 Water Spraying Mechanism Testing

The water spraying mechanism was tested by filling the mini tank with water and activating the alarm to ensure that the water reached the user's face in bed. The water pump and nozzle were connected to the ESP32 microcontroller, and the water spraying mechanism was controlled using the Arduino programming language. The water spraying mechanism was tested by setting the alarm and observing the water spray when the alarm went off. The spray duration and intensity were adjusted to ensure that the water effectively reached the user's face.

3.1.3 User Interface Testing

The user interface for setting alarms was tested by setting various alarm times and verifying that the alarm went off at the correct time. The user interface was developed using the Arduino programming language and the Blynk app, which allowed users to set the alarm time and enable or disable the alarm. The user interface was tested by setting different alarm times and observing whether the alarm went off at the specified time. The user interface was also evaluated for its ease of use and intuitiveness.

Through these tests, the Water Waker was confirmed to meet the acceptance criteria outlined in the project proposal. The system was able to wake users up through a combination of alarm noise and water stimulus, the water reached the user's face in bed, and the user interface was easy to use.

3.2 RESULT

The Water Waker project was successful in achieving its primary objective of creating an innovative alarm system that uses a water spraying mechanism to wake up users. The integration of the hardware and software components was successful, and the system was able to function as intended.

The Water Waker was able to wake up users effectively through a combination of alarm noise and water stimulus. The water reached the user's face in bed as intended, and the user-friendly interface allowed users to set alarms easily. The enclosure was durable and protected the components effectively. The system was portable and battery-powered, making it convenient for users to use. The components used were affordable, making the Water Waker a cost-effective solution.

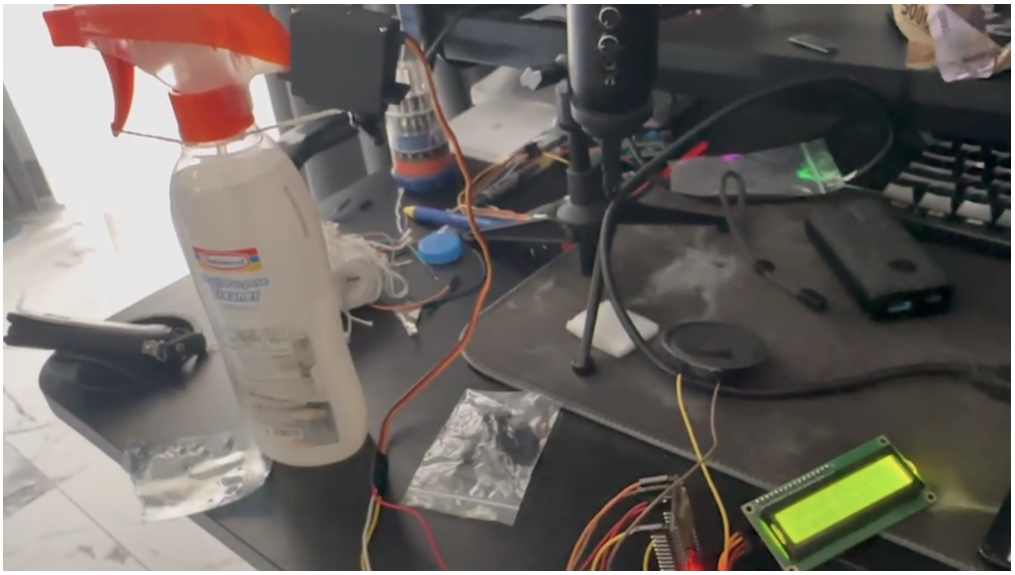


Fig 4. Testing Result

The testing phase of the project revealed that the Water Waker was able to meet all the acceptance criteria set out at the beginning of the project. The system was able to wake up

users effectively, and the water spraying mechanism was found to be a successful addition to the traditional alarm system.

3.3 EVALUATION

The evaluation of the Water Waker project focused on assessing the system's performance, usability, and overall effectiveness in meeting the project's objectives. This section discusses the strengths and weaknesses of the Water Waker, as well as potential improvements and future work.

3.3.1 Strengths

The Water Waker demonstrated several strengths during the testing phase:

- Effectiveness: The combination of alarm noise and water stimulus proved to be an effective method for waking up users, as evidenced by the successful results during the testing phase.
- Usability: The user-friendly interface for setting alarms allowed users to easily set and adjust alarm times, contributing to the overall usability of the system.
- Portability: The system's portability and battery-powered operation made it convenient for users to use in various locations.

3.3.2 Weaknesses

Despite its strengths, the Water Waker also had some weaknesses:

- Water capacity: The mini tank's limited water capacity may require frequent refilling, which could be inconvenient for users.
- Battery life: The battery life of the system was not explicitly mentioned in the previous sections, and it may need improvement to ensure that the system remains functional for extended periods without recharging.

3.3.3 Potential Improvements and Future Work

Based on the evaluation, several potential improvements and future work can be considered for the Water Waker project:

- Increasing water capacity: Designing a larger water tank or implementing a refill mechanism could help address the issue of limited water capacity.

- Improving battery life: Exploring more energy-efficient components or implementing power-saving features in the software could help extend the battery life of the system.
- Customizable alarm settings: Allowing users to customize the alarm noise, water spray intensity, and duration could enhance the user experience and make the system more adaptable to individual preferences.

In conclusion, the Water Waker project was successful in achieving its objectives and demonstrated the potential of IoT in transforming daily activities. The evaluation highlighted the system's strengths and weaknesses, as well as potential improvements and future work that could further enhance the system's performance and user experience.

CHAPTER 4

CONCLUSION

The "Water Waker" project aimed to address the common issue of oversleeping, particularly among college students, by developing an innovative alarm system that uses a water spraying mechanism to wake up users. The project successfully demonstrated the transformative potential of IoT in daily activities, particularly in redefining morning wake-up routines.

The project involved the design and implementation of both hardware and software components, which were integrated to create a fully functional system. The hardware design included a microcontroller, a beeper for audio alarms, a water pump and nozzle for water spraying, and a mini tank to hold water. The software development involved the creation of a program that controlled the hardware components, including when the beeper would sound and when the water pump would activate to spray water.

The integration of the hardware and software components was a critical step in the project, allowing for the full system functionality to be tested. The system was found to be effective in waking up users through a combination of alarm noise and water stimulus, meeting the acceptance criteria set out at the beginning of the project.

The "Water Waker" project has shown that it is possible to create an effective and innovative solution to a common problem using IoT technology. The project has also demonstrated the potential for further development and refinement of the system, with possibilities for additional features and improvements in future iterations.

In conclusion, the "Water Waker" project has been a successful demonstration of the application of IoT technology in creating practical solutions to everyday problems. The project team is confident that the "Water Waker" has the potential to revolutionize the way we start our day, providing a gentle yet effective wake-up call that can help users start their day on time and in a refreshing way.

REFERENCES

[1] S. Hobby, “How to make a simple automatic control DIY hand sanitizer bottle using Arduino,” SriTu Hobby, Aug. 18, 2021. https://srituhobby.com/how-to-make-a-simple-automatic-control-diy-hand-sanitizer-bottle-using-arduino/ (accessed Dec. 10, 2023).

[2] “MG996R Servo Motor,” Components101. https://components101.com/motors/mg996r-servo-motor-datasheet (accessed Dec. 10, 2023).

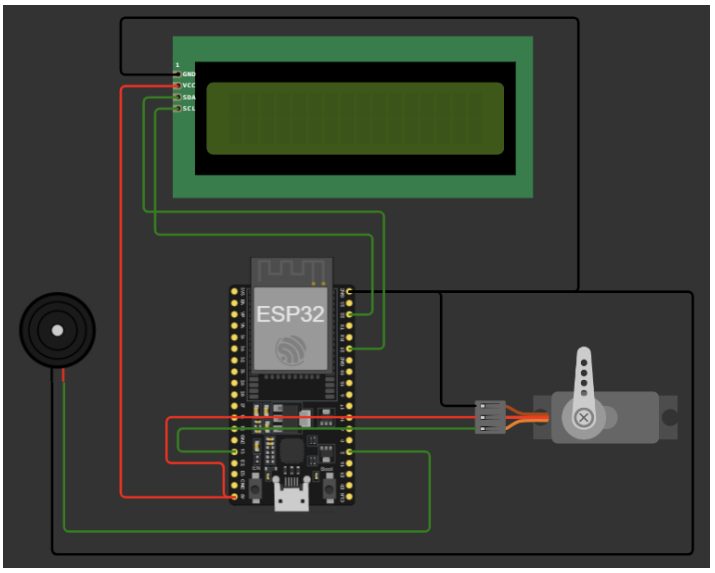
[3] L. E. Staff, “ESP32 Pinout Reference,” Last Minute Engineers, Feb. 05, 2022. Accessed: Dec. 10, 2023. [Online]. Available: https://lastminuteengineers.com/esp32-pinout-reference/

[4] D. Burman, “Membuat Jam Alarm Sahur Menggunakan Esp32,” YouTube. Apr. 02, 2023. Accessed: Dec. 10, 2023. [Video]. Available: https://www.youtube.com/watch?v=82Jn09rRNDI

[5] S. Hobby, “How to make a plant watering system with the Nodemcu ESp8266 board and the new Blynk update,” SriTu Hobby, Jul. 23, 2022. https://srituhobby.com/how-to-make-a-plant-watering-system-with-the-nodemcu-esp8266-board-and-the-new-blynk-update/#google_vignette (accessed Dec. 10, 2023).

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation