YAŞAR UNIVERSITY

COMP 3328: Embedded Systems PROJECT

NANOGROW: SMART PLANT MONITORING SYSTEM

Efe ilhan

20070001007

TERM PROJECT ADVISOR: AssISt.PROF.(PHD) ÖMER ÇETİN

ENGINEERING FACULTY

COMPUTER ENGINEERING PROGRAM

APPROVAL PAGE

I acknowledge and declare that I have personally worked in accordance with academic ethical principles during the preparation of this project. I accept and undertake that if the contrary situation is detected in any way, my project work will not be accepted and will not be evaluated, and I will be subject to sanctions announced by the university administration.

|  |  |
| --- | --- |
| Student: | Signature: |
| Efe İlhan  20070001007  23/05/2024 | ....................... |
|  |  |
|  |  |
|  |  |
|  |  |

ABSTRACT

NANOGROW: SMART PLANT MONITORING SYSTEM

İlhan, Efe  
COMP 3328: Embedded Systems Project Term Project,  
Advisor: Assist. Prof.(PhD) Ömer ÇETİN  
May 2024

The Smart Plant Monitoring System aims to revolutionize indoor plant care by providing an automated solution for monitoring and managing plant health. By leveraging real-time data on environmental factors like temperature, humidity, and light levels, users can maintain optimal growing conditions for their plants with ease. Moreover, the system will incorporate advanced features such as automated watering and proactive notification alerts for timely maintenance interventions. Through this innovative approach, plant owners can ensure the longevity and vitality of their indoor greenery, enhancing both aesthetic appeal and overall well-being.

**Keywords:** Smart Plant Monitoring System, indoor plants, automated solution, real-time data, environmental conditions, temperature, humidity, light levels, optimal growing conditions, automated watering, notification alerts, maintenance interventions.

**TABLE OF CONTENTS**

[APPROVAL PAGE iii](#_Toc160717760)

[ABSTRACT v](#_Toc160717761)

[TABLE OF CONTENTS vii](#_Toc160717762)

[1. CHAPTER: PROBLEM DEFINITION 1](#_Toc160717763)

[2. CHAPTER: THE GOAL OF THE PROJECT 3](#_Toc160717764)

[3. CHAPTER: HARDWARE 4](#_Toc160717767)

[3.1. Microprocessor / Microcontroller 4](#_Toc160717769)

[3.2. Diagram Defination of Project 4](#_Toc160717770)

[3.3. Other Input / Output Devices Used within the Scope of the Project 5](#_Toc160717771)

[3.4. Cost of Project 6](#_Toc160717772)

[4. CHAPTER: SOFTWARE DEVELOPMENT 7](#_Toc160717773)

[4.1. Functions 7](#_Toc160717774)

[4.2. Constants & Variables 7](#_Toc160717776)

[5. CHAPTER: CODE 9](#_Toc160717777)

[5.2. Arduino Code 9](#_Toc160717779)

[5.3. WOKWI diagram.json File 15](#_Toc160717780)

[5.4. WOKWI libraries.txt File 16](#_Toc160717781)

[5.5. Other WOKWI Files 16](#_Toc160717782)

[6. CHAPTER: PROJECT RUNTIME SCREENSHOT IMAGES 19](#_Toc160717783)

[7. CHAPTER: CONCLUSIONS AND FUTURE WORK 21](#_Toc160717784)

[8. REFERENCES 23](#_Toc160717785)

# CHAPTER: PROBLEM DEFINITION

In this section, we provide an overview of the development of the Smart Plant Monitoring System, its intended usage area, features, ease of use, and accessibility for users.

Indoor plant care has traditionally relied on manual methods, often resulting in suboptimal conditions and maintenance challenges. With the growing importance of greenery in indoor spaces, there is a pressing need for a modern, automated solution to monitor and manage plant health effectively.

The Smart Plant Monitoring System addresses this need by offering a comprehensive platform for real-time monitoring and management of indoor plants. By leveraging advanced sensor technology, the system provides users with valuable insights into environmental conditions such as temperature, humidity, and light levels. This data enables users to ensure optimal growing conditions for their plants, leading to healthier and more vibrant greenery.

Key features of the Smart Plant Monitoring System include:

1. Real-time data monitoring: Users can access up-to-date information on environmental conditions affecting their plants, allowing for timely interventions and adjustments.
2. Automated watering: The system includes automated watering capabilities, ensuring that plants receive the right amount of moisture to thrive.
3. Notification alerts: Users receive notifications when maintenance tasks are required, such as watering or adjusting environmental settings, enhancing ease of use and proactive plant care.
4. User-friendly interface: The system boasts an intuitive interface that is easy to navigate, making it accessible to users of all levels of expertise.

Access to the Smart Plant Monitoring System is facilitated through a web-based platform, allowing users to monitor and manage their plants from any internet-enabled device. This accessibility ensures that users can stay connected with their plants, even when they are away from home or office environments.

In the following chapters, we delve deeper into the development process of the Smart Plant Monitoring System, including data analysis, algorithm selection, and performance evaluation. We also discuss the implications of our findings and offer insights for future research and development in this field.

# CHAPTER: THE GOAL OF THE PROJECT

In this chapter, we outline the overarching objectives and goals of the Smart Plant Monitoring System project, detailing its intended usage area, features, ease of use, and accessibility for users.

The Smart Plant Monitoring System aims to revolutionize indoor plant care by providing an automated and intelligent solution for monitoring and managing plant health. With the rise of indoor greenery in various settings such as homes, offices, and public spaces, there is a growing need for a modern approach to plant care that can ensure optimal growing conditions and minimize maintenance efforts.

Key goals of the project include:

1. Development of an automated monitoring system: The project seeks to develop a system that can continuously monitor environmental conditions such as temperature, humidity, and light levels in real-time.
2. Provision of actionable insights: The system will analyze collected data to provide users with actionable insights and recommendations for optimizing plant health and growth.
3. Integration of automated maintenance features: The project aims to incorporate features such as automated watering and fertilization, reducing the need for manual intervention and streamlining plant care processes.
4. User-friendly interface: The system will feature an intuitive and user-friendly interface that is accessible to users of all levels of expertise, making it easy to navigate and operate.
5. Accessibility across devices: Users will be able to access the Smart Plant Monitoring System through various devices, including smartphones, tablets, and computers, ensuring convenience and flexibility in plant management.

By achieving these goals, the Smart Plant Monitoring System aims to empower users to maintain healthy and thriving indoor plants with minimal effort and expertise required. In the following chapters, we will delve deeper into the development process of the system, including data collection and analysis, algorithm development, and system testing and evaluation. Additionally, we will discuss the potential impact of the system on indoor plant care practices and its implications for future research and development in the field.

# CHAPTER: HARDWARE



## Microprocessor / Microcontroller

## *The Arduino Uno has been chosen for the Smart Plant Monitoring System project due to its widespread use and versatility. With ample input/output pins, stable performance, and ease of programming, the Uno provides an ideal platform for integrating sensors and actuators needed for plant health monitoring. Its popularity and user-friendly interface simplify development, making it accessible to users of all levels of expertise. Overall, the Arduino Uno offers a reliable and efficient solution for powering the hardware component of the Smart Plant Monitoring System.*

## Diagram Defination of Project

metin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Figure 2.1.** Image Description

Source: Author.

## Other Input / Output Devices Used within the Scope of the Project

*All display / input / output / sensor hardware in the WOKWI emulator environment is listed below. What the devices planned to be used within the scope of the project will be used for will be stated in the area next to them. Options whose intended use is left blank will be deleted from the table.*

|  |  |  |
| --- | --- | --- |
| **Type** | **I/O Device** | **Purpose of usage** |
| *Display* | *LED/ RGB LED* | Visual alerts or status indicators |
| *Display* | *SSD1306 OLED Display* |  |
| *Display* | *LCD (16x2 veya 20x4)* | Display real-time environmental data and plant status |
| *Display* | *TFT-LCD (Touch Screen optional)* |  |
| *Display* | *LED Dot Matrix* |  |
| *Display* | *Seven Segment Display* |  |
| *Display* | *LED Bar Graph* |  |
| *Input* | *Slide Switch* |  |
| *Input* | *Analog Joystick* |  |
| *Input* | *DIP Switch* |  |
| *Input* | *Load Cell* |  |
| *Input* | *IR Receiver / Remote* |  |
| *Input* | *Rotary Encoder* |  |
| *Input* | *Keypad* | User input for setting thresholds and parameters |
| *Input* | *Potentiometer (Slider or Rounded)* |  |
| *Sensor* | *Ultrasonic Distance Sensor* |  |
| *Sensor* | *DHT-22 (Temprature and humidity)* | Monitor environmental conditions |
| *Sensor* | *PIR Motion* |  |
| *Sensor* | *LDR Photoresistor* | Detect ambient light levels |
| *Sensor* | *Accelerometer & Gyroscope* |  |
| *Sensor* | *DS1307 RTC* |  |
| *Output* | *Buzzer* |  |
| *Output* | *Servo* | Control automated watering mechanism |
| *Output* | *Bipolar Stepper Motor (opt. with driver)* |  |
| *Output* | *Biaxial Stepper Motor* |  |
| *Output* | *Relay Module* |  |
| *Output* | *DPDT Relay* |  |
| *Misc* | *MicroSD Card Reader* |  |
| *Sensor* | *Soil Moisture Sensor* | Measure soil moisture levels |

## Cost of Project

**.1x DHT22 , 70.60 TL**

**1x Photoresistor 1.0 TL**

# 1x 4X4 Membran Keypad 35.30 TL

**1x** [**Arduino 2x16 LCD**](https://www.amazon.com.tr/Arduino-2x16-LCD-Ekran-Mod%C3%BCll%C3%BC/dp/B09Z71T7Z2/ref=sr_1_2?adgrpid=145225920634&dib=eyJ2IjoiMSJ9.-5ZS8i035-LpUZ9fNoQuxqCyRRBO8dNCYNmrkYCzFKnmVqMScnhYo-kvILZZwtWDFtQexxYvgnFb5Im_6afD3L2Rqf5Ie-RLtX-hbjuQbuiBncwk06EMtOz0nsz5fn0UX6PFVMDN8MvFOvwfE_lCPUxH8k8375iw8C1Cgbpml3CTPeoOxTK9GgQw_1BtoKABgSdRvWwR7YZ8NMlrhN4D-9USQvjebqEBUdjoO9enx8ng6jHD5cLEyHqJ5GExWlD74R_MOtzCO1B8qE0nTP46QlfBiJ3y0YSqzKd1ZKZPCzM.-eZtJEy3Wrh4fag2JEgRTvkIxNr94uw0cmIBE09Oyn8&dib_tag=se&hvadid=678425496923&hvdev=c&hvlocphy=1012783&hvnetw=g&hvqmt=b&hvrand=16856577968815233178&hvtargid=kwd-125188097&hydadcr=4713_2264181&keywords=16x2+lcd&qid=1716644182&sr=8-2)[**141,46TL**](https://www.amazon.com.tr/Arduino-2x16-LCD-Ekran-Mod%C3%BCll%C3%BC/dp/B09Z71T7Z2/ref=sr_1_2?adgrpid=145225920634&dib=eyJ2IjoiMSJ9.-5ZS8i035-LpUZ9fNoQuxqCyRRBO8dNCYNmrkYCzFKnmVqMScnhYo-kvILZZwtWDFtQexxYvgnFb5Im_6afD3L2Rqf5Ie-RLtX-hbjuQbuiBncwk06EMtOz0nsz5fn0UX6PFVMDN8MvFOvwfE_lCPUxH8k8375iw8C1Cgbpml3CTPeoOxTK9GgQw_1BtoKABgSdRvWwR7YZ8NMlrhN4D-9USQvjebqEBUdjoO9enx8ng6jHD5cLEyHqJ5GExWlD74R_MOtzCO1B8qE0nTP46QlfBiJ3y0YSqzKd1ZKZPCzM.-eZtJEy3Wrh4fag2JEgRTvkIxNr94uw0cmIBE09Oyn8&dib_tag=se&hvadid=678425496923&hvdev=c&hvlocphy=1012783&hvnetw=g&hvqmt=b&hvrand=16856577968815233178&hvtargid=kwd-125188097&hydadcr=4713_2264181&keywords=16x2+lcd&qid=1716644182&sr=8-2)

**1x Arduino UNO 993.89 TL**

**4x Servo 176.50 TL**

# CHAPTER: SOFTWARE DEVELOPMENT

From actual observed data, all previous observations have been generated from an Enterprise Resource Planning (ERP) application of a Liner shipping company. The collected data has been studied in two steps: Data Description and Data Analysis.

In the data description step, all potential and collectable features of the models have been determined and all data has been collected. After generating the dataset with determined features, data has been finalized for the training runs of machine learning algorithms.

In the data analysis step, the collected data has been merged and features have been classified for converting the elements as an operable input for the ML models. Data cleansing methods have been applied to ensure the dataset is clean and includes no missing or wrong data. Subsequently, because of the large size of the dataset outlier observations have been deleted according to the specified feature observation limits.

After completing the data analysis step, a total of 3.040.909 observations have been prepared for the two-class classification ML model studies.

## Functions



The big data has been prepared from the database (DB) management software, Oracle

## Constants & Variables

In the data analysis step, initial data has been gathered from the mentioned liner company’s ERP software. The SQL query has been constructed according to the determined feature inputs and according to the features past damage situations and observations have been tagged with 0 and 1 indicating “No Damage” and “Damage” respectively. The SQL transformation of this classification is presented in Figure 4.2.

l

# CHAPTER: CODE



## Arduino Code

1. #include <avr/io.h>
2. #include <util/delay.h>
3. #include <stdlib.h> // atoi fonksiyonu için
4. #include <ctype.h> // isdigit fonksiyonu için
5. #include <string.h> // memset fonksiyonu için
6. #include "necesarry.h"
7. #include "i2c.h"
8. #include "lcd.h"
9. #include "dht.h"
10. #include "keypad.h"
11. #define PHOTORESISTOR\_PIN 0
12. void moveServo(int angle) {
13. int pulseWidth = Map(angle, 0, 180, 1000, 2000
14. DigitalWrite(11, HIGH);
15. \_delay\_us(pulseWidth);
16. DigitalWrite(11, LOW);
17. \_delay\_us(20);
18. }
19. bool en\_soldaki\_bit\_1\_mi(uint16\_t sayi) {
20. return (sayi & 0x8000) != 0;
21. }
22. char key;
23. char data[6]; // Negatif işaret için ekstra bir karakter
24. uint16\_t humidity;
25. uint16\_t temperature;
26. int16\_t temperature\_upper\_limit = 30; // Örneğin 30.0 derece
27. int16\_t temperature\_lower\_limit = 15; // Örneğin 15.0 derece
28. uint16\_t humidity\_upper\_limit = 70; // Örneğin %70
29. uint16\_t humidity\_lower\_limit = 30; // Örneğin %30
30. uint16\_t lux\_upper\_limit = 800; // Örneğin 800 lux
31. uint16\_t lux\_lower\_limit = 200; // Örneğin 200 lux
32. float lux;
33. void setup() {
34. adc\_init();
35. I2C\_init();
36. LCD\_init();
37. keypad\_init();
38. PinMode(11, OUTPUT); // Servo pini çıkış olarak ayarla
39. LCD\_main\_screen();
40. \_delay\_ms(1000);
41. }
42. void loop() {
43. Request();
44. Response();
45. Receive\_all\_datas();
46. humidity = (I\_RH << 8) + D\_RH;
47. temperature = ((I\_Temp << 8) | D\_Temp);
48. lux = read\_lux\_value();
49. uint8\_t calculated\_sum = I\_RH + D\_RH + I\_Temp + D\_Temp;
50. int16\_t temperature\_real;
51. if (en\_soldaki\_bit\_1\_mi(temperature)) {
52. temperature = temperature & 0x7FFF;
53. temperature\_real = temperature \* (-1);
54. } else {
55. temperature\_real = temperature;
56. }
57. float real\_humidity = humidity / 10.0;
58. float real\_temperature = temperature\_real / 10.0;
59. key = keypad\_get\_key();
60. if (CheckSum != calculated\_sum) {
61. LCD\_set\_cursor(0, 0);
62. LCD\_print("Error");
63. } else {
64. int tmp = humidity / 10;
65. itoa(tmp, data, 10);
66. LCD\_set\_cursor(11, 0);
67. LCD\_print(data);
68. LCD\_print(".");
69. tmp = humidity % 10;
70. itoa(tmp, data, 10);
71. LCD\_print(data);
72. LCD\_print("                ");
73. tmp = temperature\_real / 10;
74. itoa(tmp, data, 10);
75. LCD\_set\_cursor(7, 1);
76. LCD\_print(data);
77. LCD\_print(".");
78. tmp = temperature % 10;
79. itoa(tmp, data, 10);
80. LCD\_print(data);
81. LCD\_print("                ");
82. }
83. if (key == 'S') {
84. show\_limits\_and\_set();
85. LCD\_main\_screen();
86. }
87. if (real\_temperature < temperature\_lower\_limit || real\_temperature > temperature\_upper\_limit) {
88. moveServo(90);
89. \_delay\_ms(1000);
90. }
92. // Nem kontrolü
93. if (real\_humidity < humidity\_lower\_limit || real\_humidity > humidity\_upper\_limit) {
94. moveServo(180);
95. \_delay\_ms(1000);
96. }
97. // Işık kontrolü
98. if (lux < lux\_lower\_limit || lux > lux\_upper\_limit) {
99. moveServo(120);
100. \_delay\_ms(1000);
101. }
102. \_delay\_ms(100);
103. }
104. void adc\_init() {
105. ADMUX |= (1 << REFS0);
106. ADMUX &= ~(1 << REFS1);
107. ADCSRA |= (1 << ADPS2) | (1 << ADPS0);
108. ADCSRA |= (1 << ADEN);
109. }
110. uint16\_t adc\_read(uint8\_t ch) {
111. ADMUX = (ADMUX & 0xF8) | (ch & 0x07);
112. ADCSRA |= (1 << ADSC);
113. while (ADCSRA & (1 << ADSC));
114. return ADC;
115. }
116. float read\_lux\_value() {
117. const float GAMMA = 0.7;
118. const float RL10 = 50;
119. int analogValue = adc\_read(PHOTORESISTOR\_PIN);
120. float voltage = analogValue / 1024.0 \* 5;
121. float resistance = 2000 \* voltage / (1 - voltage / 5);
122. float lux = pow(RL10 \* 1e3 \* pow(10, GAMMA) / resistance, (1 / GAMMA));
123. return lux;
124. }
125. void show\_limits\_and\_set() {
126. LCD\_clear();
127. LCD\_set\_cursor(0, 0);
128. LCD\_print("1:Temp 2:Hum");
129. LCD\_set\_cursor(0, 1);
130. LCD\_print("3:Lux ESC:Exit");
131. while (1) {
132. key = keypad\_get\_key();
133. if (key == '1' || key == '2' || key == '3') {
134. set\_limits(key);
135. return;
136. } else if (key == 'ESC') {
137. return;
138. }
139. }
140. }
141. void set\_limits(char key) {
142. uint16\_t \*upper\_limit;
143. uint16\_t \*lower\_limit;
144. char \*param\_name;
145. int lower\_limit\_range, upper\_limit\_range;
146. if (key == '1') {
147. upper\_limit = &temperature\_upper\_limit;
148. lower\_limit = &temperature\_lower\_limit;
149. param\_name = "Temp";
150. lower\_limit\_range = -40; // -40.0 derece
151. upper\_limit\_range = 80; // 80.0 derece
152. } else if (key == '2') {
153. upper\_limit = &humidity\_upper\_limit;
154. lower\_limit = &humidity\_lower\_limit;
155. param\_name = "Hum";
156. lower\_limit\_range = 0; // %0
157. upper\_limit\_range = 100; // %100
158. } else if (key == '3') {
159. upper\_limit = &lux\_upper\_limit;
160. lower\_limit = &lux\_lower\_limit;
161. param\_name = "Lux";
162. lower\_limit\_range = 0; // 0 lux
163. upper\_limit\_range = 10000; // 10000 lux
164. } else {
165. return;
166. }
167. LCD\_clear();
168. LCD\_set\_cursor(0, 0);
169. LCD\_print(param\_name);
170. LCD\_print(" limits");
171. set\_limit("Upper", upper\_limit, lower\_limit\_range, upper\_limit\_range);
172. set\_limit("Lower", lower\_limit, lower\_limit\_range, upper\_limit\_range);
173. if (\*lower\_limit > \*upper\_limit) {
174. LCD\_clear();
175. LCD\_set\_cursor(0, 0);
176. LCD\_print("Error: Lower > Upper");
177. \_delay\_ms(2000);
178. }
179. }
180. void set\_limit(const char\* limit\_type, uint16\_t\* limit, int lower\_limit\_range, int upper\_limit\_range) {
181. char key;
182. int pos = 0;
183. bool is\_negative = false;
184. memset(data, 0, sizeof(data));
185. LCD\_clear();
186. LCD\_set\_cursor(0, 0);
187. LCD\_print(limit\_type);
188. LCD\_print(" limit:");
189. while (1) {
190. LCD\_set\_cursor(0, 1);
191. LCD\_print(data);
192. key = keypad\_get\_key();
193. if ((isdigit(key) || (key == '-' && pos == 0)) && pos < sizeof(data) - 1) {
194. if (key == '-') {
195. is\_negative = true;
196. }
197. data[pos++] = key;
198. data[pos] = '\0';
199. } else if (key == 'D' && pos > 0) {
200. if (data[pos - 1] == '-') {
201. is\_negative = false;
202. }
203. data[--pos] = '\0';
204. LCD\_set\_cursor(0, 1);
205. LCD\_print("                ");
206. } else if (key == 'E') {
207. int input\_value = atoi(data);
208. if (input\_value >= lower\_limit\_range && input\_value <= upper\_limit\_range) {
209. \*limit = input\_value;
210. LCD\_clear();
211. LCD\_set\_cursor(0, 0);
212. LCD\_print("Limit set to:");
213. LCD\_set\_cursor(0, 1);
214. LCD\_print(data);
215. \_delay\_ms(2000);
216. break;
217. } else {
218. LCD\_clear();
219. LCD\_set\_cursor(0, 0);
220. LCD\_print("Invalid value");
221. \_delay\_ms(2000);
222. LCD\_clear();
223. LCD\_set\_cursor(0, 0);
224. LCD\_print(limit\_type);
225. LCD\_print(" limit:");
226. memset(data, 0, sizeof(data));
227. pos = 0;
228. }
229. } else if (key == 'ESC') {
230. return;
231. }
232. }
233. }

## WOKWI diagram.json File

1. {
2. "version": 1,
3. "author": "Anonymous maker",
4. "editor": "wokwi",
5. "parts": [
6. { "type": "wokwi-arduino-uno", "id": "uno", "top": -66.6, "left": -58.2, "attrs": {} },
7. {
8. "type": "wokwi-dht22",
9. "id": "dht1",
10. "top": -220.5,
11. "left": -120.6,
12. "attrs": { "temperature": "21.4", "humidity": "41.5" }
13. },
14. {
15. "type": "wokwi-lcd1602",
16. "id": "lcd1",
17. "top": -224,
18. "left": 284,
19. "attrs": { "pins": "i2c" }
20. },
21. {
22. "type": "wokwi-membrane-keypad",
23. "id": "keypad1",
24. "top": -482,
25. "left": 15.2,
26. "attrs": {
27. "keys": [ "1", "2", "3", "Del", "4", "5", "6", "-", "7", "8", "9", "M", "Slc", "0", "Ent", "Esc" ],
28. "fontSize": "8px"
29. }
30. },
31. { "type": "wokwi-servo", "id": "servo1", "top": -2, "left": 278.4, "attrs": {} },
32. {
33. "type": "wokwi-photoresistor-sensor",
34. "id": "ldr1",
35. "top": 185.6,
36. "left": -95.2,
37. "attrs": {}
38. }
39. ],
40. "connections": [
41. [ "dht1:GND", "uno:GND.1", "white", [ "v9.6", "h201.6" ] ],
42. [ "lcd1:GND", "uno:GND.1", "black", [ "h-86.4", "v134.4", "h-124.8" ] ],
43. [ "lcd1:VCC", "uno:5V", "red", [ "h-19.2", "v422.5", "h-124.8" ] ],
44. [ "lcd1:SDA", "uno:A4.2", "green", [ "h-57.6", "v173", "h-163.2" ] ],
45. [ "lcd1:SCL", "uno:A5.2", "green", [ "h-38.4", "v163.5", "h-182.4" ] ],
46. [ "uno:8", "keypad1:R2", "green", [ "v0" ] ],
47. [ "uno:7", "keypad1:R3", "green", [ "v0" ] ],
48. [ "uno:5", "keypad1:C1", "green", [ "v0" ] ],
49. [ "uno:4", "keypad1:C2", "green", [ "v0" ] ],
50. [ "uno:3", "keypad1:C3", "green", [ "v0" ] ],
51. [ "uno:2", "keypad1:C4", "green", [ "v0" ] ],
52. [ "servo1:V+", "uno:5V", "green", [ "h-86.4", "v201.7", "h-220.8" ] ],
53. [ "uno:6", "keypad1:R4", "green", [ "v0" ] ],
54. [ "uno:9", "keypad1:R1", "green", [ "v0" ] ],
55. [ "dht1:SDA", "uno:10", "green", [ "v19.2", "h249.7" ] ],
56. [ "dht1:VCC", "uno:5V", "red", [ "v278.4", "h278.4" ] ],
57. [ "ldr1:VCC", "uno:5V", "red", [ "h0" ] ],
58. [ "ldr1:GND", "uno:GND.2", "black", [ "h0" ] ],
59. [ "ldr1:AO", "uno:A0", "green", [ "h0" ] ],
60. [ "servo1:PWM", "uno:11", "green", [ "h-201.6", "v-57.4", "h-134.4" ] ],
61. [ "servo1:GND", "uno:GND.3", "black", [ "h-67.2", "v96", "h-96", "v-9.6" ] ]
62. ],
63. "dependencies": {}
64. }

## WOKWI libraries.txt File

# Wokwi Library List

# See https://docs.wokwi.com/guides/libraries

NONE

## Other WOKWI Files

**File Name: dht.h**

#define DHT22\_PIN 2

uint8\_t c=0,I\_RH,D\_RH,I\_Temp,D\_Temp,CheckSum;

void Request(){

 DDRB |= (1<<DHT22\_PIN);

 PORTB &= ~(1<<DHT22\_PIN);

 \_delay\_ms(20);

 PORTB |= (1<<DHT22\_PIN

}

void Response(){

 DDRB &= ~(1<<DHT22\_PIN);

 while(PINB & (1<<DHT22\_PIN));

 while((PINB & (1<<DHT22\_PIN))==0);

 while(PINB & (1<<DHT22\_PIN));

 }

uint8\_t Receive\_data() /\* receive data \*/ {

for (int q=0; q<8; q++) {

 while((PINB & (1<<DHT22\_PIN)) == 0);

 \_delay\_us(30);

 if(PINB & (1<<DHT22\_PIN

 c = (c<<1)|(0x01);

 else

 c = (c<<1); while(PINB & (1<<DHT22\_PIN));

 }

return c;

}

void Receive\_all\_datas(){

  I\_RH=Receive\_data(); /\* store first eight bit in I\_RH \*/

  D\_RH=Receive\_data(); /\* store next eight bit in D\_RH \*/

  I\_Temp=Receive\_data(); /\* store next eight bit in I\_Temp \*/

  D\_Temp=Receive\_data(); /\* store next eight bit in D\_Temp \*/

  CheckSum=Receive\_data();/\* store next eight bit in CheckSum \*/

}

**File Name: i2c.h**

#define F\_CPU 16000000UL

#define SCL\_CLOCK  100000L

void I2C\_init(void) {

    TWSR = 0x00;

    TWBR = ((F\_CPU/SCL\_CLOCK) - 16) / 2;

    TWCR = (1<<TWEN);

}

void I2C\_start(void) {

    TWCR = (1<<TWSTA) | (1<<TWEN) | (1<<TWINT);

    while (!(TWCR & (1<<TWINT)));

}

void I2C\_stop(void) {

    TWCR = (1<<TWSTO) | (1<<TWINT) | (1<<TWEN);

}

void I2C\_write(uint8\_t data) {

    TWDR = data;

    TWCR = (1<<TWINT) | (1<<TWEN);

    while (!(TWCR & (1<<TWINT))); //

}

**File Name: lcd.h**

#define LCD\_ADDR 0x27

void LCD\_send\_nibble(uint8\_t data) {

    data |= 0x08;

    I2C\_start();

    I2C\_write(LCD\_ADDR << 1);

    I2C\_write(data | 0x04);

    I2C\_write(data & ~0x04);

    I2C\_stop();

    \_delay\_us(50);

}

void LCD\_send\_byte(uint8\_t data, uint8\_t rs) {

    uint8\_t high\_nibble = (data & 0xF0) | rs;

    uint8\_t low\_nibble = ((data << 4) & 0xF0) | rs;

    LCD\_send\_nibble(high\_nibble);

    LCD\_send\_nibble(low\_nibble);

}

void LCD\_send\_command(uint8\_t command) {

    LCD\_send\_byte(command, 0x00);

    \_delay\_ms(2);

}

void LCD\_send\_data(uint8\_t data) {

    LCD\_send\_byte(data, 0x01);

}

void LCD\_init(void) {

    \_delay\_ms(50);

    \_delay\_ms(5);

    LCD\_send\_command(0x03);

    \_delay\_us(150);

    LCD\_send\_command(0x03);

    LCD\_send\_command(0x02);

    LCD\_send\_command(0x28);

    LCD\_send\_command(0x0C);

    LCD\_send\_command(0x01);

    \_delay\_ms(2);

    LCD\_send\_command(0x06);

void LCD\_set\_cursor(uint8\_t col, uint8\_t row) {

    uint8\_t address = (row == 0) ? col : (col + 0x40);

    LCD\_send\_command(0x80 | address);

}

void LCD\_print(char \*str) {

    while (\*str) {

        LCD\_send\_data(\*str++);

    }

}

void LCD\_clear(void) {

    LCD\_send\_command(0x01);

    \_delay\_ms(2);

}

void LCD\_main\_screen(void){

  LCD\_clear();

  LCD\_print("Humidity =");

  LCD\_set\_cursor(0, 1);

  LCD\_print("Temp =");

}

**File Name: keypad.h**

#define ROWS 4

#define COLS 4

// Tuş takımının bağlantılarını temsil eden dizi

char keys[ROWS][COLS] = {

    {'1','2','3','D'},

    {'4','5','6','-'},

    {'7','8','9','M'},

    {'S','0','E','P'}

};

// Satırların pinleri

byte rowPins[ROWS] = { 9, 8, 7, 6 }; // Arduino'da bağlı olan pinler

// Sütunların pinleri

byte colPins[COLS] = { 5, 4, 3, 2 }; // Arduino'da bağlı olan pinler

void keypad\_init(void) {

    for (int i = 0; i < ROWS; i++) {

        PinMode(rowPins[i], OUTPUT);

        DigitalWrite(rowPins[i], HIGH);

    }

    for (int i = 0; i < COLS; i++) {

        PinMode(colPins[i], INPUT\_PULLUP);

    }

}

char keypad\_get\_key(void) {

    for (uint8\_t row = 0; row < ROWS; row++) {

        for (int i = 0; i < ROWS; i++) {

            DigitalWrite(rowPins[i], HIGH);

        }

        DigitalWrite(rowPins[row], LOW);

        \_delay\_us(1);

        for (uint8\_t col = 0; col < COLS; col++) {

            if (!DigitalRead(colPins[col])) {

                \_delay\_ms(50);

                while (!DigitalRead(colPins[col]));

                return keys[row][col];

            }

        }

    }

    return '\0';

}

**File Name: necesarry.h**

void DigitalWrite(uint8\_t pin, uint8\_t value) {

    if (pin >= 0 && pin <= 7) {

        if (value == LOW) {

            PORTD &= ~(1 << pin);

        } else if (value == HIGH) {

            PORTD |= (1 << pin);

        }

    } else if (pin >= 8 && pin <= 13) {

        pin -= 8;

        if (value == LOW) {

            PORTB &= ~(1 << pin);

        } else if (value == HIGH) {

            PORTB |= (1 << pin);

        }

    } else if (pin >= 14 && pin <= 19) {

        pin -= 14;

        if (value == LOW) {

            PORTC &= ~(1 << pin);

        } else if (value == HIGH) {

            PORTC |= (1 << pin);

        }

    }

}

uint8\_t DigitalRead(uint8\_t pin) {

    if (pin >= 0 && pin <= 7) {

        return (PIND & (1 << pin)) ? 1 : 0;

    } else if (pin >= 8 && pin <= 13) {

        pin -= 8;

        return (PINB & (1 << pin)) ? 1 : 0;

    } else if (pin >= 14 && pin <= 19) {

        pin -= 14;

        return (PINC & (1 << pin)) ? 1 : 0;

    }

    return 0;

}

void PinMode(uint8\_t pin, uint8\_t mode) {

    if (pin >= 0 && pin <= 7) { // Port D

        if (mode == INPUT) {

            DDRD &= ~(1 << pin);

            PORTD &= ~(1 << pin);

        } else if (mode == OUTPUT) {

            DDRD |= (1 << pin);

        } else if (mode == INPUT\_PULLUP) {

            DDRD &= ~(1 << pin);

            PORTD |= (1 << pin);

        }

    } else if (pin >= 8 && pin <= 13) {

        pin -= 8;

        if (mode == INPUT) {

            DDRB &= ~(1 << pin);

            PORTB &= ~(1 << pin);

        } else if (mode == OUTPUT) {

            DDRB |= (1 << pin);

        } else if (mode == INPUT\_PULLUP) {

            DDRB &= ~(1 << pin);

            PORTB |= (1 << pin);

        }

    } else if (pin >= 14 && pin <= 19)

        pin -= 14;

        if (mode == INPUT) {

            DDRC &= ~(1 << pin);

            PORTC &= ~(1 << pin);

        } else if (mode == OUTPUT) {

            DDRC |= (1 << pin);

        } else if (mode == INPUT\_PULLUP) {

            DDRC &= ~(1 << pin);

            PORTC |= (1 << pin);

        }

    }

}

long Map(long x, long in\_min, long in\_max, long out\_min, long out\_max) {

    return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min;

}

# CHAPTER: PROJECT RUNTIME SCREENSHOT IMAGES

|  |  |
| --- | --- |
|  |  |
| Home Screen | Slc screen |
|  |  |
| Limit selection | Error Message |

# CHAPTER: CONCLUSIONS AND FUTURE WORK

In conclusion, the development and implementation of the Smart Plant Monitoring System have demonstrated the potential and effectiveness of using IoT and automation in plant care. The project successfully integrated various sensors to monitor temperature, humidity, and light levels, and utilized these inputs to manage the health of indoor plants through automated responses such as watering and alert notifications.

The key findings and achievements of this project are as follows:

1. **Real-Time Environmental Monitoring**: The system continuously monitors the environmental conditions, providing real-time data that ensures optimal growing conditions for indoor plants.
2. **Automated Plant Care**: The implementation of automated watering based on soil moisture levels has proven to reduce manual intervention, thereby simplifying plant care routines.
3. **User Notifications**: The system effectively notifies users when maintenance is required, ensuring that plants receive timely care and attention.

By leveraging the capabilities of IoT and automation, the Smart Plant Monitoring System has not only improved the health and longevity of indoor plants but also provided a convenient and efficient solution for plant enthusiasts and gardeners.

To improve the Smart Plant Monitoring System, several enhancements can be made. Adding sensors for soil pH and nutrients would provide more comprehensive plant health data. Improving the machine learning models will enhance the system's ability to predict plant needs accurately. Developing a mobile app will allow users to control and monitor their plants remotely. Adapting the system for larger gardens or greenhouses will increase its applicability. Implementing renewable energy sources like solar power will make the system more sustainable. Additionally, incorporating a user feedback feature will help continuously refine and improve the system based on user experiences.

These improvements will make the system more efficient and user-friendly.

# REFERENCES

Cetin, O. (s.y.). Cmp3328: Real-Time Systems. Erişim tarihi: 25 Mayıs 2024, <http://omercetin.com.tr/~courses/comp3328/?i=3>

Chris--A. (n.d.). Keypad. GitHub. Retrieved May 25, 2024, from <https://github.com/Chris--A/Keypad>

Rickman, J. (n.d.). LiquidCrystal\_I2C. GitHub. Retrieved May 25, 2024, from <https://github.com/johnrickman/LiquidCrystal_I2C>

Rickman, J. (n.d.). LiquidCrystal\_I2C. Retrieved May 25, 2024, from <https://github.com/johnrickman/LiquidCrystal_I2C>

Wokwi. (n.d.). Wokwi Photoresistor Sensor. Retrieved May 25, 2024, from <https://docs.wokwi.com/parts/wokwi-photoresistor-sensor>