



ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
(Α.Π.Θ.)

ΗΥ0901 ΜΙΚΡΟΠΕΞΕΡΓΑΣΤΕΣ ΚΑΙ ΠΕΡΙΦΕΡΕΙΑΚΑ

Εργασία Arduino

Αντωνιάδης Δημήτριος (8462): akdimitri@auth.gr
Δημητριάδης Βασίλειος (ΑΕΜ): dimvasdim@auth.gr

14 Μαΐου 2019

Περιεχόμενα

1	Εισαγωγή.	2
2	Εργασία στον μικροελεγκτή <i>Arduino</i>.	3
2.1	Λειτουργία προγράμματος.	3
2.2	Υλικό και συνδεσμολογία.	5
2.3	Πηγαίος κώδικας.	6
2.4	Αποτελέσματα εργασίας του μικροελεγκτή <i>Arduino</i>	12
3	Εργασία στον μικροελεγκτή <i>Wemos D1 R2</i>.	13
3.1	Λειτουργία προγράμματος.	13
3.2	Υλικό και συνδεσμολογία.	15
3.3	Πηγαίος κώδικας.	16
3.3.1	<i>thermostat-Wifi.ino</i>	16
3.3.2	<i>Gsender.h</i>	23
3.3.3	<i>Gsender.cpp</i>	24
3.4	Αποτελέσματα εργασίας του μικροελεγκτή <i>Wemos D1 R2</i>	28

1 Εισαγωγή.

Το παρόν έγγραφο αποτελεί την αναφορά της εργασίας *Arduino* που πραγματοποιήθηκε στο πλαίσιο του μαθήματος *Μικροεπεξεργαστές και Περιφερειακά* του 8^{ου} εξαμήνου. Σκοπός της εργασίας ήταν η δημιουργία ενός έξυπνου θερμομέτρου-θερμοστάτη. Πιο συγκεκριμένα, με τη χρήση του μικροελεγκτή *Arduino* σε συνδυασμό με αισθητήρες θερμότητας και εγγύτητας έπρεπε να αναπτυχθεί ένα σύστημα το οποίο θα μετρούσε την θερμοκρασία του χώρου ανά τακτά χρονικά διαστήματα και θα την παρουσίαζε τοπικά μέσω μιας LCD οθόνης.

Παραπάνω έγινε μία σύντομη περιγραφή της εργασίας. Στην επόμενη (2^η) ενότητα παρουσιάζεται η λειτουργία του προγράμματος στον μικροελεγκτή *Arduino*. Εντός της δεύτερης ενότητας παρουσιάζονται ο τρόπος λειτουργίας του προγράμματος, τα εξαρτήματα που χρησιμοποιήθηκαν και η συνδεσμολογία τους. Ακόμη, στην ενότητα αυτή παρουσιάζεται ο κώδικας που χρησιμοποιήθηκε για την εκτέλεση του προγράμματος και τα αποτελέσματα των μετρήσεων. Στην τρίτη (3^η) ενότητα παρουσιάζεται η λειτουργία του προγράμματος στον μικροελεγκτή *Wemos D1 R2*. Ομοίως με τη δεύτερη ενότητα παρουσιάζονται οι αντίστοιχες πτυχές. Στην τέταρτη (4^η) ενότητα παρουσιάζονται τα συμπεράσματα της εργασίας αυτής και ο επίλογος.

Η δεύτερη υλοποίηση στον μικροελεγκτή *Wemos D1 R2* πραγματοποιήθηκε με σκοπό την εκπλήρωση του υποερωτήματος που αφορούσε την αποστολή e-mail για την άνοδο της θερμοκρασίας πάνω από μία ορισμένη τιμή. Και αυτό γιατί ο μικροελεγκτής *Arduino* που διαθέταμε δεν είχε τη δυνατότητα σύνδεσης στο διαδίκτυο. Ωστόσο, ο μικροελεγκτής *Wemos D1 R2* έχει μόλις 8 pins GPIO και για το λόγο αυτό στη δεύτερη υλοποίηση δεν περιλαμβάνονται οι ενέργειες αναφορικά με τα LEDs.

2 Εργασία στον μικροελεγκτή *Arduino*.

Η παρούσα ενότητα περιγράφει την υλοποίηση την εργασίας σε μικροελεγκτή *Arduino*.

2.1 Λειτουργία προγράμματος.

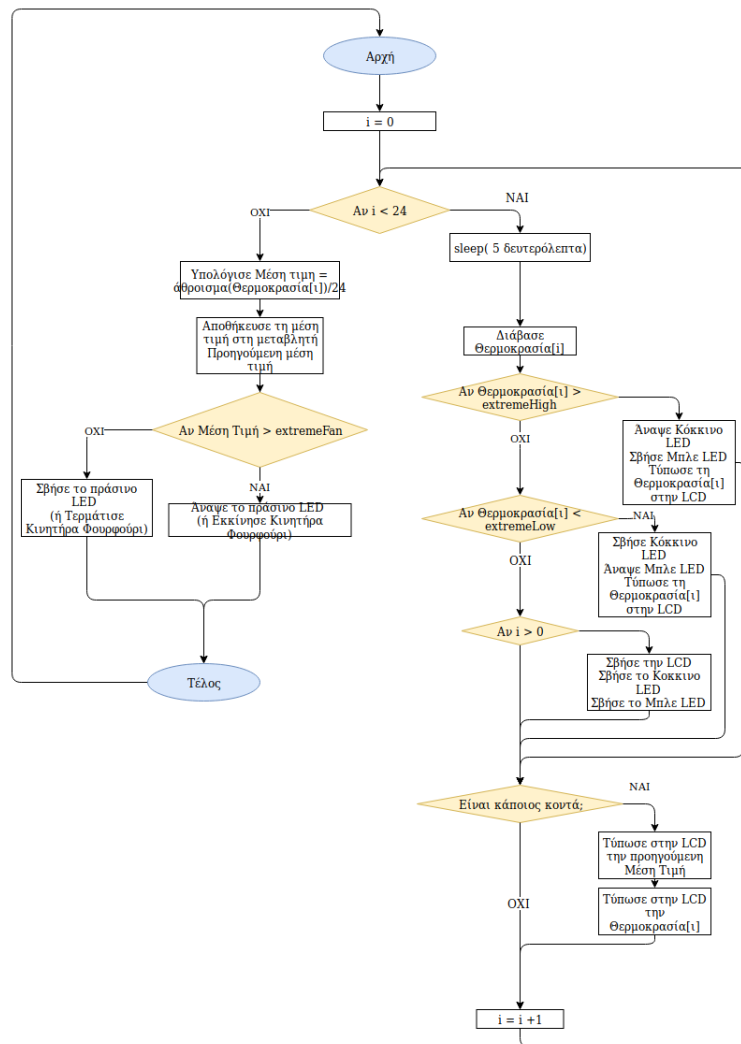
Στην υποενότητα αυτή παρουσιάζεται ένα αναλυτικό διάγραμμα ροής το οποίο περιγράφει τη λειτουργία του προγράμματος που εκτελεί ο μικροελεγκτής *Arduino*.

Το πρόγραμμα εκκινεί με έναν μετρητή ίσο με μηδέν. Κάθε 5 δευτερόλεπτα, το πρόγραμμα λαμβάνει τη τιμή του αισθητήρα θερμοκρασίας. Αν η τιμή αυτή είναι μεγαλύτερη από μία συγκεκριμένη τιμή(*extremeHigh*), τότε ανάβει το κόκκινο LED, αν η τιμή αυτή είναι μικρότερη από μία άλλη συγκεκριμένη τιμή(*extremeLow*) τότε ανάβει το μπλε LED. Οι παραπάνω ενέργειες ακολουθούνται από την εκτύπωση της τιμής της θερμοκρασίας στην οθόνη LCD. Στις 24 μετρήσεις, δηλαδή στα δύο λεπτά, υπολογίζεται η μέση τιμή της θερμοκρασίας των τελευταίων 24 μετρήσεων και αποθηκεύεται. Επιπλέον, η μέση τιμή τυπώνεται στη LCD οθόνη και το πρόγραμμα εκκινεί από την αρχή.

Η μέση τιμή παραμένει στην οθόνη για 10 δευτερόλεπτα. Αυτό επιτυγχάνεται με τον εξής τρόπο: αφού εκκινήσει το πρόγραμμα από την αρχή είναι ήδη τυπωμένη η προηγούμενη μέση τιμή, συνεπώς, όταν το πρόγραμμα εισέλθει στη δεύτερη επανάληψη, όπου $i = 1$, τότε θα έχουν περάσει και 10 δευτερόλεπτα, άρα καθαρίζεται και η LCD οθόνη.

Ακόμη, ύστερα από τον υπολογισμό της μέσης τιμής, ελέγχεται αν η τιμή αυτή είναι μεγαλύτερη από μία συγκεκριμένη τιμή(*extremeFan*). Αν είναι μεγαλύτερη τότε ανάβει το πράσινο LED.

Επιπρόσθετα, ο μικροελεγκτής εκτελεί μία ακόμη λειτουργία. Διαβάζει κάθε 5 δευτερόλεπτα και μέσω ενός αισθητήρα εγγύτητας την απόσταση του κοντινότερου αντικειμένου. Εάν, η τιμή αυτή είναι σχετικά μικρή, δηλαδή αν αντιληφθεί ότι κάποιος βρίσκεται κοντά, τότε τυπώνει στη LCD οθόνη την τελευταία μέτρηση και την τελευταία μέση τιμή που υπολογίστηκε.



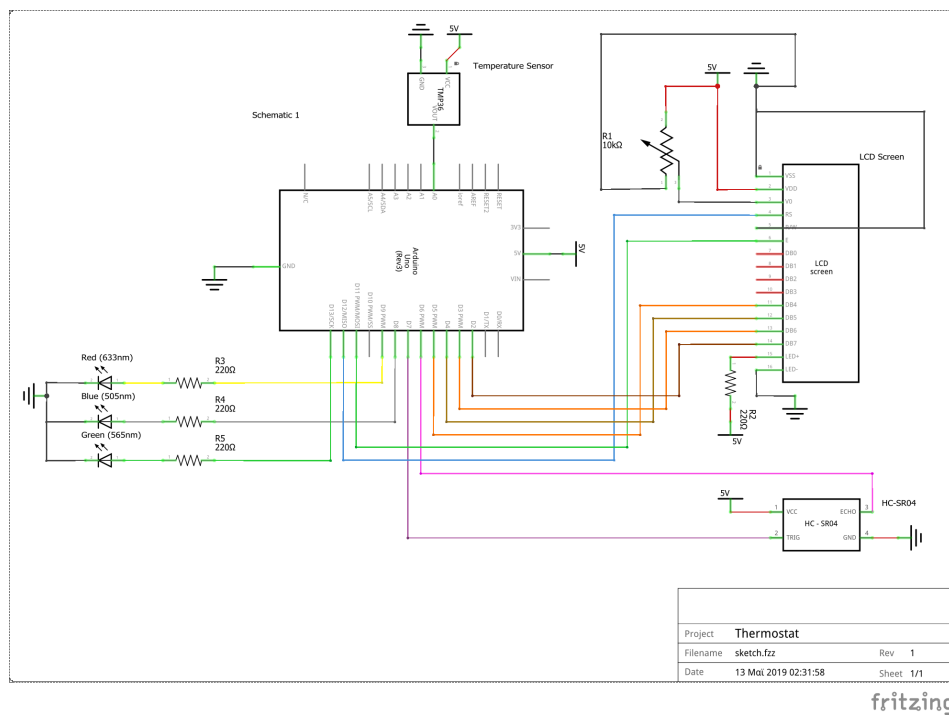
Σχήμα 1: Διάγραμμα Ροής προγράμματος στον μικροελεγκτή *Arduino*

2.2 Υλικό και συνδεσμολογία.

Για την υλοποίηση του παραπάνω προγράμματος χρησιμοποιήθηκαν τα εξής εξαρτήματα:

- Arduino UNO
- Αισθητήρας θερμότητας TMP36 [1]
- Οθόνη LCD διαστάσεων 16x2 JHD659 M10 1.1 [2]
- Αισθητήρας εγγύτητας HC-SR04 [3]
- Ποτενσιόμετρο 10 k Ω
- 3 LEDs

Τα παραπάνω εξαρτήματα συνδέθηκαν σύμφωνα με την εξής συνδεσμολογία:



Σχήμα 2: Συνδεσμολογία εξαρτημάτων με μικροελεγκτή *Arduino*.

Με βάση τη συνδεσμολογία αυτή υλοποιήθηκε και ο πηγαίος κώδικας που εκτελεί το πρόγραμμα.

2.3 Πηγαίος κώδικας.

Στην υποενότητα αυτή παρουσιάζεται ο πηγαίος κώδικας ο οποίος εκτελείται ακριβώς με τον ίδιο τρόπο τις λειτουργίες όπως αυτές περιγράφηκαν στο διάγραμμα ροής.

```
1  /*
2  * Authors: 1. Dimitrios Antoniadis
3  *           2. Vasileios Dimitriadis
4  *
5  * email:   1. akdimitri@auth.gr
6  *           2. dimvasdim@auth.gr
7  *
8  * University: Aristotle University of Thessaloniki (AUTH)
9  * Semester:   8th
10 * Subject:    Microprocessors and Peripherals
11 *
12 * Parts required:
13 * - one TMP36 temperature sensor
14 * - one LCD 16x2 JHD659 M10 1.1
15 * - one HC-SR04 proximity sensor
16 */
17
18 // include the library code:
19 #include <LiquidCrystal.h>
20
21 // functions declaration
22 void printLCD( float averageTemperature);
23 void checkExtremeFan( float averageTemperature);
24 int checkProximity();
25
26 // initialize the LCD function with the numbers of the interface pins
27 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
28
29 // global variables declaration
30 const int sensorPin = A0;
31 const int BLUE = 8;
32 const int RED = 9;
33 const int GREEN = 13;
34 const float extremeLow = 21.5;
35 const float extremeHigh = 22.5;
36 const float extremeFan = 23;
37 float temperature[24], averageTemperature;
38 float lastAverageTemperature = 0;
39 int i;
40 const int trigPin = 7;    // Trigger
41 const int echoPin = 6;    // Echo
42 unsigned long duration, cm, inches;
43 int isSomeoneClose = 0;
44
```

```

45 // setup function
46 void setup() {
47
48 //LEDS
49 // set the digital pin as output:
50 pinMode(BLUE, OUTPUT);
51 pinMode(RED, OUTPUT);
52 pinMode(GREEN, OUTPUT);
53 digitalWrite(RED, LOW);
54 digitalWrite(BLUE, LOW);
55 digitalWrite(GREEN, LOW);
56
57 //LCD
58 // set up the number of columns and rows on the LCD
59 lcd.begin(16, 2);
60 // Print a message to the LCD.
61 lcd.print("ARDUINO");
62 // set the cursor to column 0, line 1
63 // line 1 is the second row, since counting begins with 0
64 lcd.setCursor(0, 1);
65 // print to the second line
66 lcd.print("PROJECT");
67
68 // HC-SR04
69 pinMode(trigPin, OUTPUT);
70 pinMode(echoPin, INPUT);
71
72 // open a serial connection to display values
73 Serial.begin(9600);
74 }
75
76 void loop() {
77
78     averageTemperature = 0;
79
80     // 24*5 secs = 2 minutes
81     for( i = 0; i < 24; i++){
82         // delay 5 seconds = 5 * 1000 milliseconds
83         delay(5*1000);
84
85         // if 10 seconds elapsed since previous average temperature print
86         // clear LCD
87         if( i == 1){
88             lcd.clear();
89             lcd.setCursor(0, 0);
90             lcd.print("CALCULATING...");
91         }
92
93         // read temperature in Celsius

```



```

93 // read the value on AnalogIn pin 0 and store it in a variable
94 int sensorVal = analogRead(sensorPin);
95
96 // send the 10-bit sensor value out the serial port
97 Serial.print("sensor Value: ");
98 Serial.print(sensorVal);
99
100 // convert the ADC reading to voltage
101 float voltage = (sensorVal / 1024.0) * 5.0;
102
103 // Send the voltage level out the Serial port
104 Serial.print(", Volts: ");
105 Serial.print(voltage);
106
107 // convert the voltage to temperature in degrees C
108 // the sensor changes 10 mV per degree
109 // the datasheet says there's a 500 mV offset
110 // ((voltage - 500 mV) times 100)
111 Serial.print(", degrees C: ");
112 temperature[i] = (voltage - .5) * 100;
113 Serial.println(temperature[i]);
114
115 // check extreme values
116 if( temperature[i] < extremeLow && i > 0){
117     lcd.clear();
118     lcd.setCursor(0, 0);
119     lcd.print("WARNING");
120     lcd.setCursor(0, 1);
121     lcd.print("TEMP < ");
122     lcd.print(extremeLow);
123     digitalWrite(BLUE, HIGH);
124     digitalWrite(RED, LOW);
125 }
126 else if( temperature[i] > extremeHigh && i > 0){
127     lcd.clear();
128     lcd.setCursor(0, 0);
129     lcd.print("WARNING");
130     lcd.setCursor(0, 1);
131     lcd.print("TEMP > ");
132     lcd.print(extremeHigh);
133     digitalWrite(RED, HIGH);
134     digitalWrite(BLUE, LOW);
135 }
136 else{
137     // clear LEDS
138     digitalWrite(RED, LOW);
139     digitalWrite(BLUE, LOW);
140     if( i > 0);
141 }

```

```

142
143 // check if someone is close
144 isSomeoneClose = checkProximity();
145
146 if( isSomeoneClose == 1){
147     // clean up the screen before printing a new reply
148     lcd.clear();
149     // set the cursor to column 0, line 0
150     lcd.setCursor(0, 0);
151     // print some text
152     lcd.print("AVG. TEMP.");
153     lcd.print(lastAverageTemperature);
154     // move the cursor to the second line
155     lcd.setCursor(0, 1);
156     lcd.print("LAST TEMP. ");
157     lcd.print(temperature[i]);
158 }
159 }
160
161 // calculate average temperature
162 for( i = 0; i < 24; i++){
163     averageTemperature = averageTemperature + temperature[i];
164 }
165 averageTemperature = averageTemperature/24;
166
167 // store average temperature
168 lastAverageTemperature = averageTemperature;
169
170 // print average temperature
171 printLCD( averageTemperature);
172 Serial.print("Average Temperature: ");
173 Serial.println(averageTemperature);
174
175 //check for fan
176 checkExtremeFan( averageTemperature);
177
178 }
179
180 /*****
181 * Sub-routines *
182 *****/
183
184
185 /* temperatureFunction: this function is responsible for reading
186 *                        the analog value of temperature function,
187 *                        also is responsible for checking extreme
188 *                        temperature values.
189 */
190

```

```

191
192 /* printLCD(): this function is responsible for
193 *           printing average temperature to
194 *           LCD screen.
195 */
196 void printLCD( float averageTemperature){
197     // clean up the screen before printing a new reply
198     lcd.clear();
199     // set the cursor to column 0, line 0
200     lcd.setCursor(0, 0);
201     // print some text
202     lcd.print("AVG. TEMP.");
203     // move the cursor to the second line
204     lcd.setCursor(0, 1);
205     lcd.print(averageTemperature);
206 }
207
208 /* checkExtremeFan(): this function is responsible
209 *           for checking whether fan is needed
210 *           to be turned on
211 */
212 void checkExtremeFan( float averageTemperature){
213
214     // check fan
215     if( averageTemperature > extremeFan){
216         digitalWrite(GREEN, HIGH);
217     }
218     else{
219         digitalWrite(GREEN, LOW);
220     }
221 }
222
223
224 /* checkProximity(): this function is responsible for acknowledging
225 *           wheter
226 *           someone is close or not.
227 */
228 int checkProximity(){
229     // The sensor is triggered by a HIGH pulse of 10 or more
230     // microseconds.
231     // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
232     digitalWrite(trigPin, LOW);
233     delayMicroseconds(5);
234     digitalWrite(trigPin, HIGH);
235     delayMicroseconds(15);
236     digitalWrite(trigPin, LOW);
237
238     // Read the signal from the sensor: a HIGH pulse whose
239     // duration is the time (in microseconds) from the sending

```

```

238 // of the ping to the reception of its echo off of an object.
239 if( duration = pulseIn(echoPin, HIGH)){
240
241     // Convert the time into a distance
242     cm = (duration/2) / 29.1;    // Divide by 29.1 or multiply by
0.0343
243     inches = (duration/2) / 74;    // Divide by 74 or multiply by
0.0135
244
245     Serial.print(inches);
246     Serial.print("in, ");
247     Serial.print(cm);
248     Serial.print("cm");
249     Serial.println();
250     if( cm < 10){
251         return 1;
252     }
253     else{
254         return 0;
255     }
256 }
257 else{
258     Serial.println("Nobody Nearby");
259     return 0;
260 }
261 }

```

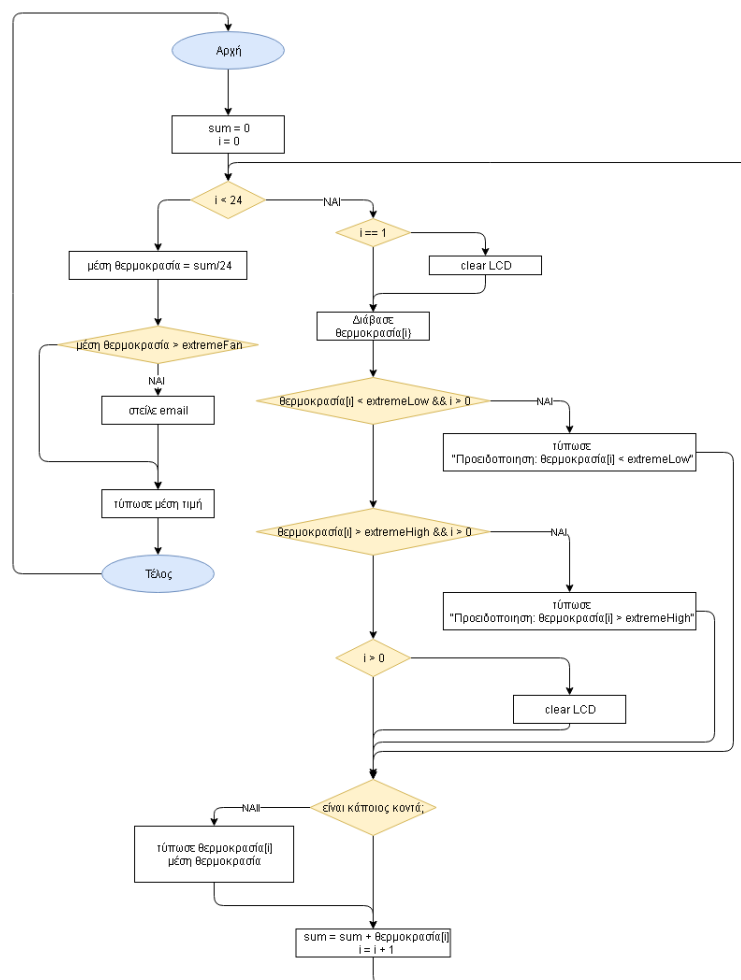
2.4 Αποτελέσματα εργασίας του μικροελεγκτή *Ardui- no*.

3 Εργασία στον μικροελεγκτή *Wemos D1 R2*.

Η παρούσα ενότητα περιγράφει την υλοποίηση την εργασίας σε μικροελεγκτή *Wemos D1 R2*.

3.1 Λειτουργία προγράμματος.

Στην υποενότητα αυτή παρουσιάζεται ένα αναλυτικό διάγραμμα ροής το οποίο περιγράφει τη λειτουργία του προγράμματος που εκτελεί ο μικροελεγκτής *Arduino*.



Σχήμα 3: Διάγραμμα Ροής στον μικροελεγκτή *Wemos D1 R2*

Το παραπάνω διάγραμμα περιγράφει τη λειτουργία του προγράμματος που εκτελείται στον μικροελεγκτή *Wemos D1 R2*. Ωστόσο, για να εκτελεσθεί η αποστολή του e-mail έχουν πραγματοποιηθεί και οι απαραίτητες ρυθμίσεις.

Για την αποστολή του e-mail χρησιμοποιήθηκε ο **SMTP Server** που παρέχει η *Google*. Πιο συγκεκριμένα, δημιουργήθηκε ο λογαριασμός mcu.2019.wemos@gmail.com. Οι πληροφορίες του λογαριασμού αυτού τοποθετήθηκαν στον πηγαίο κώδικα και το πρόγραμμα ρυθμίστηκε ώστε να εκτελούνται οι απαραίτητες ενέργειες.

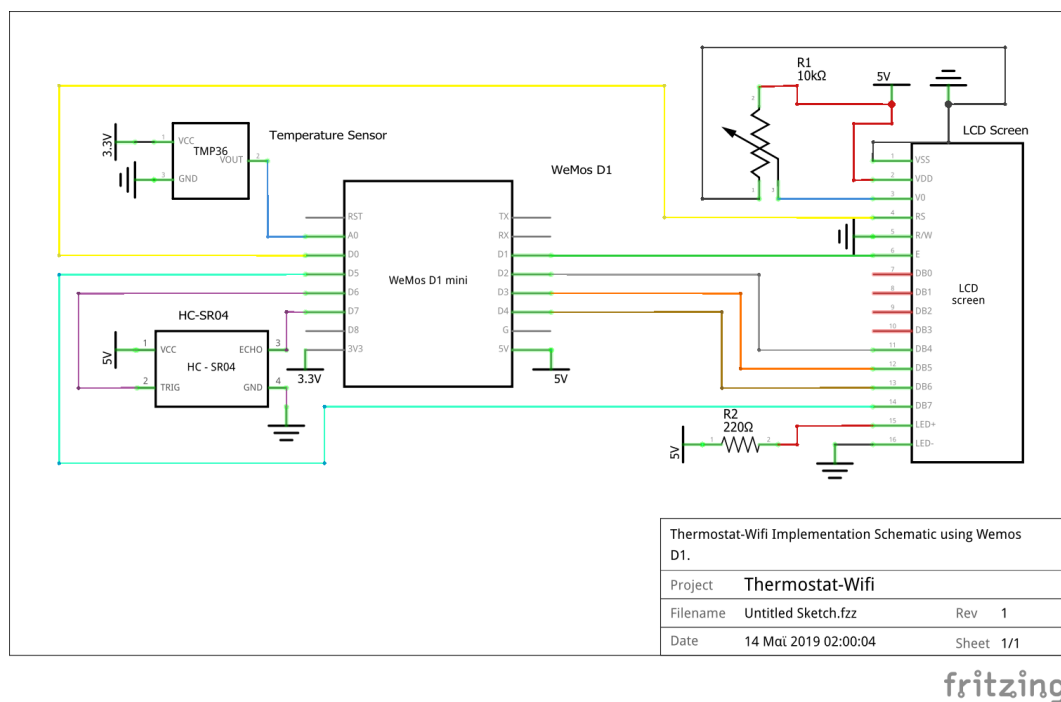
Το πρόγραμμα γενικά είναι παρόμοιο με αυτό που εκτελείται στον μικροελεγκτή *Arduino* με τη διαφορά ότι δεν περιλαμβάνονται στο πρόγραμμα αυτό τα LEDs.

3.2 Υλικό και συνδεσμολογία.

Για την υλοποίηση του παραπάνω προγράμματος χρησιμοποιήθηκαν τα εξής εξαρτήματα:

- Wemos D1 R2
- Αισθητήρας θερμοότητας TMP36 [1]
- Οθόνη LCD διαστάσεων 16ξ2 JHD659 M10 1.1 [2]
- Αισθητήρας εγγύτητας HC-SR04 [3]
- Ποτενσιόμετρο 10 kΩ

Τα παραπάνω εξαρτήματα συνδέθηκαν σύμφωνα με την εξής συνδεσμολογία:



Σχήμα 4: Συνδεσμολογία εξαρτημάτων με μικροελεγκτή *Wemos D1 R2*.

Με βάση τη συνδεσμολογία αυτή υλοποιήθηκε και ο πηγαίος κώδικας που εκτελεί το πρόγραμμα.

3.3 Πηγαίος κώδικας.

Στην υποενότητα αυτή παρουσιάζεται ο πηγαίος κώδικας ο οποίος εκτελείται ακριβώς με τον ίδιο τρόπο τις λειτουργίες όπως αυτές περιγράφηκαν στο διάγραμμα ροής.

Το πρόγραμμα περιλαμβάνει 3 αρχεία. Το αρχείο *thermostat-Wifi.ino* περιλαμβάνει το κύριο πρόγραμμα ενώ τα άλλα δύο αρχεία *Gsender.cpp*, *Gsender.h* υλοποιούν τις ρυθμίσεις για την επικοινωνία με τον **SMTP Server**.

3.3.1 *thermostat-Wifi.ino*

```
1 #include <ESP8266WiFi.h>
2 #include <LiquidCrystal.h>
3 #include "Gsender.h"
4
5 #pragma region Globals
6 const char* ssid = "Dimitris2"; // WIFI
7 const char* password = "69451070306945558718"; // WIFI
8 uint8_t connection_state = 0; //
9 uint16_t reconnect_interval = 10000; // If not
10 const int sensorPin = A0;
11 int sensorVal, i;
12 float sum, temperature[24], averageTemperature;
13 const float extremeLow = 21.5;
14 const float extremeHigh = 22.5;
15 const float extremeFan = 23;
16 const int trigPin = 12; // Trigger
17 const int echoPin = 13; // Echo
18 unsigned long duration, cm, inches;
19 int isSomeoneClose = 0;
20 #pragma endregion Globals
21
22 // initialize the LCD function with the numbers of the interface pins
23 // D0 -> GPIO16
24 // D1 -> GPIO5
25 // D2 -> GPIO4
26 // D3 -> GPIO0
27 // D4 -> GPIO2
28 // D5 -> GPIO14
29 // D6 -> GPIO12
```

```

30 // D7 -> GPIO13
31 LiquidCrystal lcd(16, 5, 4, 0, 2, 14);
32
33 uint8_t WiFiConnect(const char* nSSID = nullptr, const char*
    nPassword = nullptr)
34 {
35     static uint16_t attempt = 0;
36     Serial.print("Connecting to ");
37     if(nSSID) {
38         WiFi.begin(nSSID, nPassword);
39         Serial.println(nSSID);
40     } else {
41         WiFi.begin(ssid, password);
42         Serial.println(ssid);
43     }
44
45     uint8_t i = 0;
46     while(WiFi.status() != WL_CONNECTED && i++ < 50)
47     {
48         delay(200);
49         Serial.print(".");
50     }
51     ++attempt;
52     Serial.println("");
53     if(i == 51) {
54         Serial.print("Connection: TIMEOUT on attempt: ");
55         Serial.println(attempt);
56         if(attempt % 2 == 0)
57             Serial.println("Check if access point available or SSID
and Password\r\n");
58         return false;
59     }
60     Serial.println("Connection: ESTABLISHED");
61     Serial.print("Got IP address: ");
62     Serial.println(WiFi.localIP());
63     return true;
64 }
65
66 void Awaits()
67 {
68     uint32_t ts = millis();
69     while(!connection_state)
70     {
71         delay(50);
72         if(millis() > (ts + reconnect_interval) && !connection_state)
73         {
74             connection_state = WiFiConnect();
75             ts = millis();
76         }
77     }

```

```

76     }
77 }
78
79
80 /* printLCD(): this function is responsible for
81 *             printing average temperature to
82 *             LCD screen.
83 */
84 void printLCD( float averageTemperature){
85     // clean up the screen before printing a new reply
86     lcd.clear();
87     // set the cursor to column 0, line 0
88     lcd.setCursor(0, 0);
89     // print some text
90     lcd.print("AVG. TEMP.");
91     // move the cursor to the second line
92     lcd.setCursor(0, 1);
93     lcd.print(averageTemperature);
94 }
95
96
97 /* checkProximity(): this function is responsible for acknowledging
98 *                     wheter
99 *                     someone is close or not.
100 */
101 int checkProximity(){
102     // The sensor is triggered by a HIGH pulse of 10 or more
103     // microseconds.
104     // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
105     digitalWrite(trigPin, LOW);
106     delayMicroseconds(5);
107     digitalWrite(trigPin, HIGH);
108     delayMicroseconds(15);
109     digitalWrite(trigPin, LOW);
110
111     // Read the signal from the sensor: a HIGH pulse whose
112     // duration is the time (in microseconds) from the sending
113     // of the ping to the reception of its echo off of an object.
114     if( duration = pulseIn(echoPin, HIGH)){
115
116         // Convert the time into a distance
117         cm = (duration/2) / 29.1;    // Divide by 29.1 or multiply by
118         0.0343
119         inches = (duration/2) / 74;  // Divide by 74 or multiply by
120         0.0135
121
122         Serial.print(inches);
123         Serial.print("in, ");
124         Serial.print(cm);

```

```

121     Serial.print("cm");
122     Serial.println();
123     if( cm < 10){
124         return 1;
125     }
126     else{
127         return 0;
128     }
129 }
130 else{
131     Serial.println("Nobody Nearby");
132     return 0;
133 }
134 }
135
136
137 /* checkExtremeFan(): this function is responsible
138 *                      for checking whether fan is needed
139 *                      to be turned on
140 */
141 void checkExtremeFan(float averageTemperature){
142
143     // check fan
144     if( averageTemperature > extremeFan){
145         //digitalWrite(GREEN, HIGH);
146         Gsender *gsender = Gsender::Instance();    // Getting pointer to
class instance
147         String subject = "FAN = ON!";
148         String value = String(averageTemperature, DEC);
149         String message = String("FAN TURNED ON : " + value);
150         if(gsender->Subject(subject)->Send("mitsos1996@yahoo.com",
message)) {
151             Serial.println("Message send.");
152         } else {
153             Serial.print("Error sending message: ");
154             Serial.println(gsender->getError());
155         }
156     }
157 }
158 else{
159
160     //digitalWrite(GREEN, LOW);
161 }
162 }
163
164
165
166
167 void setup()

```

```

168 {
169     Serial.begin(115200);
170     connection_state = WiFiConnect();
171     if(!connection_state) // if not connected to WIFI
172         Awaits();         // constantly trying to connect
173
174
175     //LCD
176     // set up the number of columns and rows on the LCD
177     lcd.begin(16, 2);
178     // Print a message to the LCD.
179     lcd.print("ARDUINO");
180     // set the cursor to column 0, line 1
181     // line 1 is the second row, since counting begins with 0
182     lcd.setCursor(0, 1);
183     // print to the second line
184     lcd.print("PROJECT");
185
186
187     // HC-SR04
188     pinMode(trigPin, OUTPUT);
189     pinMode(echoPin, INPUT);
190 }
191
192 void loop(){
193     sum = 0;
194     for( i = 0; i < 24; i++){
195
196         // if 10 seconds elapsed since previous average temperature print
197         // clear LCD
198         if( i == 1){
199             lcd.clear();
200             lcd.setCursor(0, 0);
201             lcd.print("CALCULATING...");
202         }
203
204         // read the value on AnalogIn pin 0 and store it in a variable
205         sensorVal= analogRead(sensorPin);
206
207         // send the 10-bit sensor value out the serial port
208         Serial.print("sensor Value: ");
209         Serial.print(sensorVal);
210
211         // convert the ADC reading to voltage
212         float voltage = (sensorVal / 1024.0) * 3.1;
213
214         // Send the voltage level out the Serial port
215         Serial.print(", Volts: ");
216         Serial.print(voltage);

```

```

216
217 // convert the voltage to temperature in degrees C
218 // the sensor changes 10 mV per degree
219 // the datasheet says there's a 500 mV offset
220 // ((voltage - 500 mV) times 100)
221 Serial.print(", degrees C: ");
222 temperature[i] = (voltage - .5) * 100;
223 Serial.println(temperature[i]);
224
225 // check extreme values
226 if( temperature[i] < extremeLow && i > 0){
227     lcd.clear();
228     lcd.setCursor(0, 0);
229     lcd.print("WARNING");
230     lcd.setCursor(0, 1);
231     lcd.print("TEMP < ");
232     lcd.print(extremeLow);
233 }
234 else if( temperature[i] > extremeHigh && i > 0){
235     lcd.clear();
236     lcd.setCursor(0, 0);
237     lcd.print("WARNING");
238     lcd.setCursor(0, 1);
239     lcd.print("TEMP > ");
240     lcd.print(extremeHigh);
241     //digitalWrite(LED, HIGH);
242     //digitalWrite(LED, LOW);
243 }
244 else if( i > 0){
245     lcd.clear();
246 }
247
248
249 // check if someone is close
250 isSomeoneClose = checkProximity();
251
252 if( isSomeoneClose == 1){
253     // clean up the screen before printing a new reply
254     lcd.clear();
255     // set the cursor to column 0, line 0
256     lcd.setCursor(0, 0);
257     // print some text
258     lcd.print("AVG. TEMP.");
259     lcd.print(averageTemperature);
260     // move the cursor to the second line
261     lcd.setCursor(0, 1);
262     lcd.print("LAST TEMP.");
263     lcd.print(temperature[i]);
264 }

```

```
265
266     sum = sum + temperature[i];
267
268     delay(5*1000);
269
270 }
271
272     averageTemperature = sum/24;
273
274     checkExtremeFan( averageTemperature);
275
276     // print average temperature
277     printLCD( averageTemperature);
278     Serial.print("Average Temperature: ");
279     Serial.println(averageTemperature);
280
281 }
```

3.3.2 *Gsender.h*

```
1  /* Gsender class helps send e-mails from Gmail account
2  *   using Arduino core for ESP8266 WiFi chip
3  *   by Boris Shobat
4  *   September 29 2016
5  */
6  #ifndef G_SENDER
7  #define G_SENDER
8  #define GS_SERIAL_LOG_1          // Print to Serial only server
   response
9  //#define GS_SERIAL_LOG_2        // Print to Serial client commands
   and server response
10 #include <WiFiClientSecure.h>
11
12 class Gsender
13 {
14     protected:
15         Gsender();
16     private:
17         const int SMTP_PORT = 465;
18         const char* SMTP_SERVER = "smtp.gmail.com";
19         const char* EMAILBASE64_LOGIN = "
bWN1LjIwMTkud2Vtb3NAZ21haWwuY29t";
20         const char* EMAILBASE64_PASSWORD = "REBuKzAxOTk2";
21         const char* FROM = "mcu.2019.wemos@gmail.com";
22         const char* _error = nullptr;
23         char* _subject = nullptr;
24         String _serverResponse;
25         static Gsender* _instance;
26         bool AwaitSMTPResponse(WiFiClientSecure &client, const String
&resp = "", uint16_t timeOut = 10000);
27
28     public:
29         static Gsender* Instance();
30         Gsender* Subject(const char* subject);
31         Gsender* Subject(const String &subject);
32         bool Send(const String &to, const String &message);
33         String getLastResponse();
34         const char* getError();
35 };
36 #endif // G_SENDER
```


3.3.3 Gsender.cpp

```
1 #include "Gsender.h"
2 Gsender* Gsender::_instance = 0;
3 Gsender::Gsender() {}
4 Gsender* Gsender::Instance()
5 {
6     if (_instance == 0)
7         _instance = new Gsender;
8     return _instance;
9 }
10
11 Gsender* Gsender::Subject(const char* subject)
12 {
13     delete [] _subject;
14     _subject = new char[strlen(subject)+1];
15     strcpy(_subject, subject);
16     return _instance;
17 }
18 Gsender* Gsender::Subject(const String &subject)
19 {
20     return Subject(subject.c_str());
21 }
22
23 bool Gsender::AwaitSMTPResponse(WiFiClientSecure &client, const
    String &resp, uint16_t timeOut)
24 {
25     uint32_t ts = millis();
26     while (!client.available())
27     {
28         if(millis() > (ts + timeOut)) {
29             _error = "SMTP Response TIMEOUT!";
30             return false;
31         }
32     }
33     _serverResponse = client.readStringUntil('\n');
34     #if defined(GS_SERIAL_LOG_1) || defined(GS_SERIAL_LOG_2)
35     Serial.println(_serverResponse);
36     #endif
37     if (resp && _serverResponse.indexOf(resp) == -1) return false;
38     return true;
39 }
40
41 String Gsender::getLastResponse()
42 {
43     return _serverResponse;
44 }
```

```

45
46 const char* Gsender::getError()
47 {
48     return _error;
49 }
50
51 bool Gsender::Send(const String &to, const String &message)
52 {
53     WiFiClientSecure client;
54     client.setInsecure();
55     #if defined(GS_SERIAL_LOG_2)
56     Serial.print("Connecting to :");
57     Serial.println(SMTP_SERVER);
58     #endif
59     if(!client.connect(SMTP_SERVER, SMTP_PORT)) {
60         _error = "Could not connect to mail server";
61         return false;
62     }
63     if(!AwaitSMTPResponse(client, "220")) {
64         _error = "Connection Error";
65         return false;
66     }
67
68     #if defined(GS_SERIAL_LOG_2)
69     Serial.println("HELO friend:");
70     #endif
71     client.println("HELO friend");
72     if(!AwaitSMTPResponse(client, "250")){
73         _error = "identification error";
74         return false;
75     }
76
77     #if defined(GS_SERIAL_LOG_2)
78     Serial.println("AUTH LOGIN:");
79     #endif
80     client.println("AUTH LOGIN");
81     AwaitSMTPResponse(client);
82
83     #if defined(GS_SERIAL_LOG_2)
84     Serial.println("EMAILBASE64_LOGIN:");
85     #endif
86     client.println(EMAILBASE64_LOGIN);
87     AwaitSMTPResponse(client);
88
89     #if defined(GS_SERIAL_LOG_2)
90     Serial.println("EMAILBASE64_PASSWORD:");
91     #endif
92     client.println(EMAILBASE64_PASSWORD);
93     if (!AwaitSMTPResponse(client, "235")) {

```

```

94     _error = "SMTP AUTH error";
95     return false;
96 }
97
98 String mailFrom = "MAIL FROM: <" + String(FROM) + '>';
99 #if defined(GS_SERIAL_LOG_2)
100 Serial.println(mailFrom);
101 #endif
102 client.println(mailFrom);
103 AwaitSMTPResponse(client);
104
105 String rcpt = "RCPT TO: <" + to + '>';
106 #if defined(GS_SERIAL_LOG_2)
107 Serial.println(rcpt);
108 #endif
109 client.println(rcpt);
110 AwaitSMTPResponse(client);
111
112 #if defined(GS_SERIAL_LOG_2)
113 Serial.println("DATA: ");
114 #endif
115 client.println("DATA");
116 if (!AwaitSMTPResponse(client, "354")) {
117     _error = "SMTP DATA error";
118     return false;
119 }
120
121 client.println("From: <" + String(FROM) + '>');
122 client.println("To: <" + to + '>');
123
124 client.print("Subject: ");
125 client.println(_subject);
126
127 client.println("Mime-Version: 1.0");
128 client.println("Content-Type: text/html; charset=\\"UTF-8\\"");
129 client.println("Content-Transfer-Encoding: 7bit");
130 client.println();
131 String body = "<!DOCTYPE html><html lang=\\"en\\">" + message + "</
    html>";
132 client.println(body);
133 client.println(".");
134 if (!AwaitSMTPResponse(client, "250")) {
135     _error = "Sending message error";
136     return false;
137 }
138 client.println("QUIT");
139 if (!AwaitSMTPResponse(client, "221")) {
140     _error = "SMTP QUIT error";
141     return false;

```

```
142 }  
143 return true;  
144 }
```

3.4 Αποτελέσματα εργασίας του μικροελεγκτή *Wemos D1 R2*.

Αναφορές

- [1] TMP35/TMP36/TMP37 datasheet. <https://www.arduino.cc/en/uploads/Main/TemperatureSensor.pdf>.
- [2] JHD659 M10 1.1 LCD datasheet. <https://www.arduino.cc/documents/datasheets/LCDscreen.PDF>.
- [3] HC-SR04 datasheet. <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>.