



ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ (Α.Π.Θ.)

ΗΥ0901 ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ ΚΑΙ ΠΕΡΙΦΕΡΕΙΑΚΑ

Εργασία Arduino

Αντωνιάδης Δημήτριος (8462): *akdimitri@auth.gr*
Δημητριάδης Βασίλειος (8404): *dimvasdim@auth.gr*

6 Ιουνίου 2019

Περιεχόμενα

1 Εισαγωγή.	2
2 Εργασία στον μικροελεγκτή <i>Arduino</i>.	2
2.1 Λειτουργία προγράμματος.	2
2.2 Τλικό και συνδεσμολογία.	4
2.3 Αποτελέσματα εργασίας του μικροελεγκτή <i>Arduino</i>	5
3 Εργασία στον μικροελεγκτή <i>Wemos D1 R2</i>.	7
3.1 Λειτουργία προγράμματος.	7
3.2 Τλικό και συνδεσμολογία.	9
3.3 Αποτελέσματα εργασίας του μικροελεγκτή <i>Wemos D1 R2</i>	10
4 Επίλογος.	12

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 τη θερμοκρασία του χώρου ανά τακτά χρονικά διαστήματα και θα την παρουσίαζε τοπικά μέσω μιας LCD οινόνης.

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 μέρος του κώδικα 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 (ESP8266)*. 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 (ESP8266)* 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 διαθέτει 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.

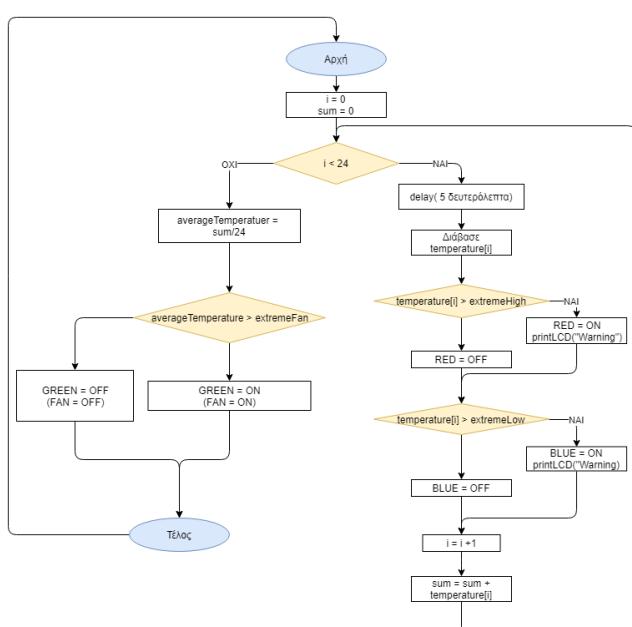
Ο συνολικός κώδικας μαζί με τις μετρήσεις, τις εικόνες, και άλλα σχετικά αρχεία περιλαμβάνονται στον εξής σύνδεσμο:

<https://github.com/akdimitri/Microprocessors-and-Peripherals-Arduino-Project.git>

2 Εργασία στον μικροελεγκτή *Arduino*.

2.1 Λειτουργία προγράμματος.

Το πρόγραμμα που εκτελείται στον μικροελεγκτή *Arduino* μετράει κάθε 5 δευτερόλεπτα τη θερμοκρασία του περιβάλλοντος. Αν η θερμοκρασία αυτή είναι μεγαλύτερη από μία συγκεκριμένη τιμή (**extremeHigh**) ή αν είναι μικρότερη από μία άλλη συγκεκριμένη τιμή (**extremeLow**) τότε τυπώνεται το αντίστοιχο μήνυμα στην οινόνη LCD. Επιπλέον, εάν η θερμοκρασία που διαβάζει ο μικροελεγκτής είναι μεγαλύτερη από τη θερμοκρασία **extremeHigh** τότε ανάβει το **κόκκινο** LED. Αντιστοίχως, όταν η θερμοκρασία είναι μικρότερη από τη θερμοκρασία **extremeLow**, τότε ανάβει το **μπλε** LED.



Σχήμα 1: Διάγραμμα ροής κυρίως προγράμματος.

Κάθε 24 μετρήσεις της θερμοκρασίας περιβάλλοντος, δηλαδή κάθε 5 δευτερόλεπτα, υπολογίζεται η μέση τιμή των 24 αυτών μετρήσεων. Η μέτρηση αυτή τυπώνεται στην οινόνη LCD και παραμένει για δέκα (10) δευτερόλεπτα στην οινόνη και δεν μπορεί να τυπωθεί κάτι άλλο κατά τη διάρκεια αυτών των δέκα δευτερολέπτων (παρά μόνο στην περίπτωση που κάποιος πλησιάσει κοντά στον αισθητήρα εγγύτητας). Αν η μέση τιμή είναι μεγαλύτερη από μία συγκεκριμένη τιμή **extremeFan**, ανάβει το **πράσινο** LED, το οποίο προσδομιώνει την ενεργοποίηση του ανεμιστήρα. Το διάστημα των δέκα δευτερολέπτων τηρείται με τη συνθήκη $i > 0$ το οποίο επιτρέπει τη νέα εκτύπωση στην οινόνη ύστερα από 2 επαναλήψεις των 5 δευτερολέπτων, δηλαδή ύστερα από 10 δευτερόλεπτα. Η συνήκη αυτή δε φαίνεται στο διάγραμμα για να μη γίνει πολύπλοκο και δυσνόητο το διάγραμμα αυτό.

Επιπλέον, το πρόγραμμα αυτό περιλαμβάνει και την υλοποίηση ενός **Overflow Interrupt Vector** του **Timer2**. Συγκεκριμένα, το Interrupt αυτό χρησιμοποιείται για τη μέτρηση της απόστασης του κοντινότερου αντικείμενου από τον αισθητήρα εγγύτητας κάθε 0.5 δευτερόλεπτα. Αν η απόσταση αυτή είναι μικρότερη από μία συγκεκριμένη τιμή (10 cm) τότε τυπώνεται στην οινόνη LCD η προηγούμενη μέση τιμή της θερμοκρασίας και η τελευταία μέτρηση της

θερμοκρασίας που πραγματοποιήθηκε.

To Arduino διαθέτει έναν **ATmega328P** και επομένως για την υλοποίηση του Interrupt Vector του TIMER2 πρέπει να γίνουν οι κατάλληλες ρυθμίσεις, όπως φαίνονται παρακάτω στο κομμάτι κώδικα που ακολουθεί.

Αρχικά, στη συνάρτηση `setup()` στον καταχωρητή **TIMSK2** το bit 0 τίθεται σε 1 προκειμένου να ενεργοποιηθεί το TIMER2 OVERFLOW INTERRUPT VECTOR. Στη συνέχεια τα bits 0,1,2 του καταχωρητή **TCCR2B** τίθενται σε 1 προκειμένου να ορισθεί ο prescaler ίσος με `clk/1024`.

O ATmega328P λειτουργεί στα 16MHz. Με τον prescaler ίσο με 1024 ο TIMER2 λειτουργεί στα 15 kHz. Ο TIMER2 πραγματοποιεί OVERFLOW στο 255 εφόσον αυτός είναι 8 bits. Επομένως, overflow πραγματοποιείται κάθε 0,016 δευτερόλεπτα. [4]

Εμείς επιθυμούμε να πραγματοποιούμε έλεγχο κάθε μισό λεπτό, για το λόγο αυτό χρησιμοποιείται μία μεταβλήτη counter η οποία ελέγχει πόσες φορές έχει πραγματοποιηθεί overflow. Όταν counter == 30, δηλαδή κάθε 0.49152 δευτερόλεπτα, εκτελείται ο έλεγχος εγγύτητας.

```
1 /* setup(): This function is executed only once at the POWER ON
2  *          or at RESET
3  */
4 void setup() {
5
6     ...
7
8     /* Enable TIMER2 OVERFLOW INTERRUPT */
9     TCCR2A = 0;
10    TCCR2B = 0;
11    TCNT2 = 0;
12    TIMSK2 = (TIMSK2 & B11111110) | 0x01;           // Arduino UNO has an ATmega328P CPU
13    TCCR2B = (TCCR2B & B11111000) | 0x07;           // Thus, ENABLE bit TIMSK:0 and SET prescaler to clk
14    /1024
15
16    sei();
17}
18
19 /* TIMER2 OVERFLOW INTERRUPT VECTOR: this function checks periodically every 0.5 seconds
20  *          if someone is nearby. Then it prints average Temperature
21  *          and the latest temperature measurment.
22 */
23 ISR(TIMER2_OVF_vect){
24     counter++;
25     //Serial.println("TIMER OVERFLOW: " + counter);
26     if( counter == 30){
27         String line1, line2;
28         //Serial.print("INTERRUPT: ");
29         // check if someone is close
30         isSomeoneClose = checkProximity();
31         counter = 0;
32         if( isSomeoneClose == 1){
33             line1 = String("AVG: " + String( averageTemperature, 3));
34             if(i == 0)
35                 line2 = String("Last TEMP: " + String(temperature[24], 3));
36             else
37                 line2 = String("Last TEMP: " + String(temperature[i-1], 3));
38
39             printLCD( line1, line2);
40         }
41     }
42 }
```

Η συνάρτηση `loop()` είναι υπεύθυνη για την εκτέλεση του κύριου προγράμματος και εκτελεί τις λειτουργίες που περιγράφονται στο διάγραμμα ροής (Σχήμα 1) παραπάνω.

```
1 /* loop(): this function is the main function. It executes
2  *          continuously.
3  */
4 void loop() {
5
6     sum = 0;                                // this variable holds the sum of temperatures
7
8     Serial.println("New Measurement");
```

```

9   for( i = 0; i < 24; i++){
10    delay(5*1000);
11    temperature[ i ] = readTemperature();
12    sum = sum + temperature[ i ];
13    checkExtremeValues( temperature[ i ] , i );
14  }
15
16 /* Calculate Average Temperature */
17 averageTemperature = sum/24;
18 String line1 = String("Avg. Temp.");
19 String line2 = String(averageTemperature, 3);
20 printLCD( line1, line2 );
21 Serial.println("\n\n Average Temperature: " + String(averageTemperature, 3));
22
23 //check for fan
24 checkExtremeFan( averageTemperature );
25
26 }

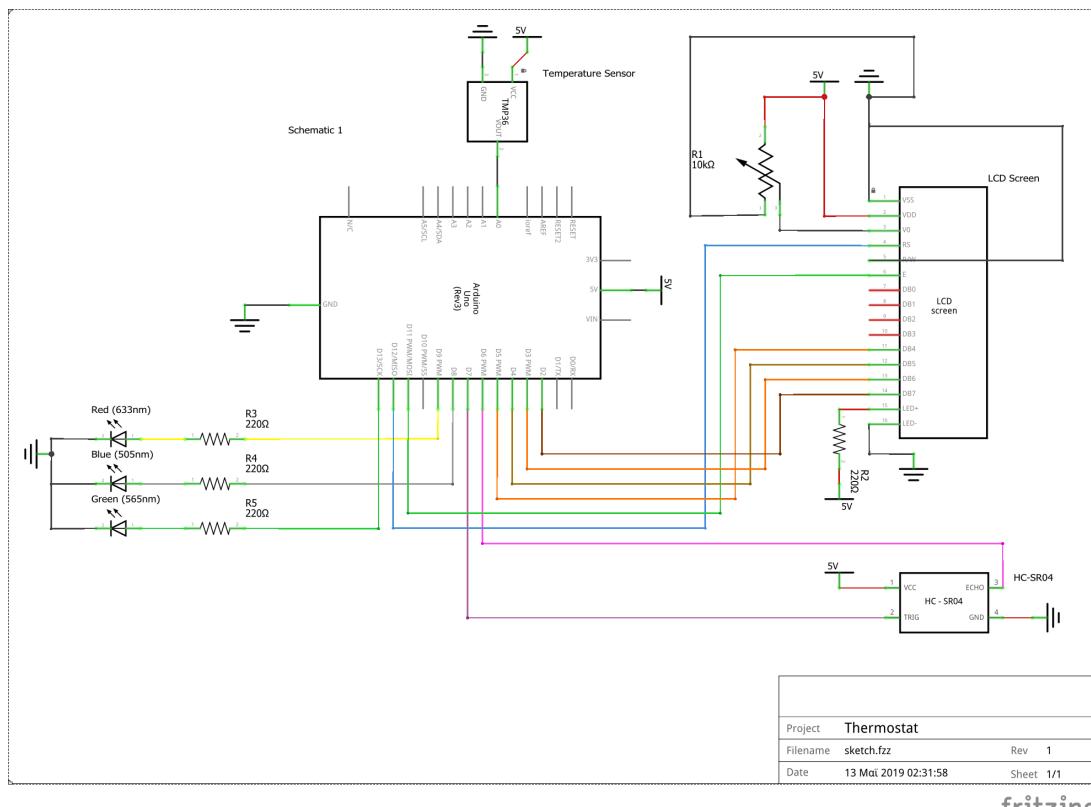
```

2.2 Υλικό και συνδεσμολογία.

Για την υλοποίηση του παραπάνω προγράμματος χρησιμοποιήθηκαν τα εξής εξαρτήματα:

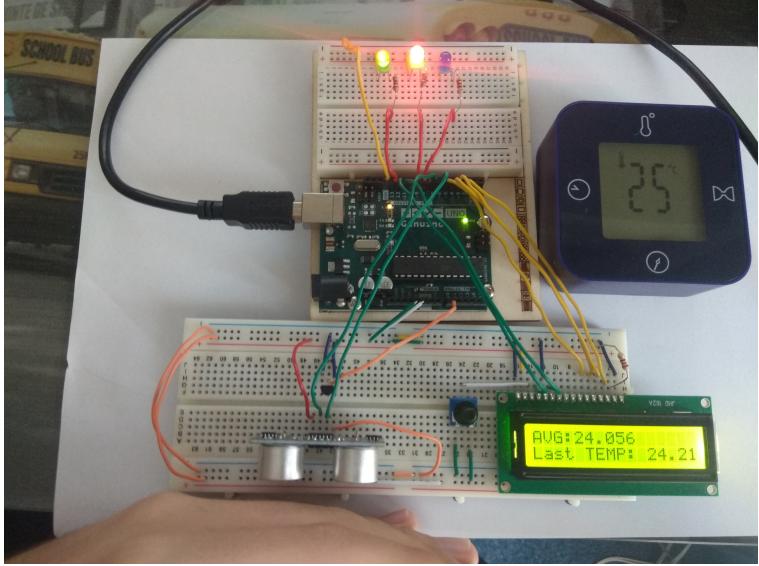
- Arduino UNO
- Αισθητήρας θερμότητας TMP36 [1]
- Ουδόνη LCD διαστάσεων 16x2 JHD659 M10 1.1 [2]
- Αισθητήρας εγγύτητας HC-SR04 [3]
- Ποτενσιόμετρο 10 kΩ
- 3 LEDs

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.



Σχήμα 6: Έλεγχος εγγύτητας.

Στην εικόνα αριστερά φαίνεται η ορθή λειτουργία του TIMER2 και του έλεγχου εγγύτητας. Ο μικροελεγκτής έχει αντιληφθεί ότι βρίσκεται αντικείμενο κοντά και τυπώνει την τελευταία μέση τιμή που υπολογίσθηκε και την τελευταία μέτρηση.

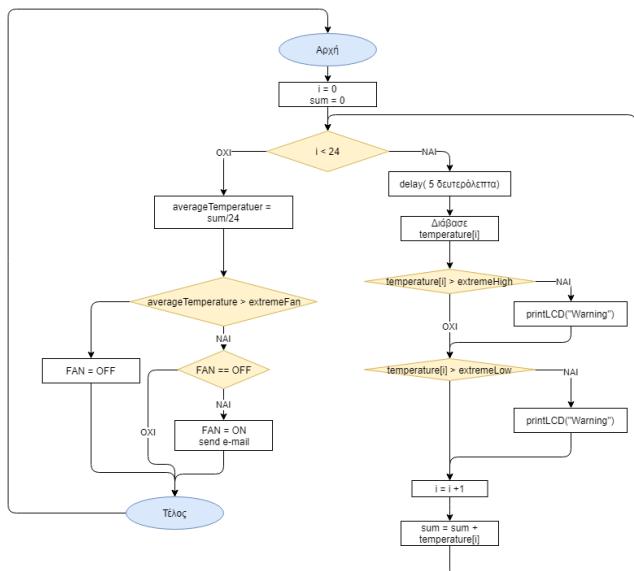
Σημείωση: στη φωτογραφία αριστερά περιλαμβάνεται και ένα ψηφιακό θερμόμετρο το οποίο πιστοποιεί την ορθότητα των μετρήσεων.

3 Εργασία στον μικροελεγκτή Wemos D1 R2.

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

3.1 Λειτουργία προγράμματος.

Το πρόγραμμα που εκτελείται στον μικροελεγκτή Wemos D1 R2 μετράει κάθε 5 δευτερόλεπτα τη θερμοκρασία του περιβάλλοντος. Αν η θερμοκρασία αυτή είναι μεγαλύτερη από μία συγκεκριμένη τιμή (**extremeHigh**) ή αν είναι μικρότερη από μία τιμή (**extremeLow**) τότε τυπώνεται το αντίστοιχο μήνυμα στην οθόνη LCD.



Σχήμα 7: Διάγραμμα ροής κυρίως προγράμματος.

πό τον αισθητήρα εγγύτητας. Αν η απόσταση αυτή είναι μικρότερη από μία συγκεκριμένη μέση τιμή της θερμοκρασίας και η τελευταία μέτρηση της θερμοκρασίας που πραγματοποιήθηκε.

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. Στο αρχείο **Gsender.h** συμπληρώθηκαν τα κατάλληλα στοιχεία όπως φαίνονται παρακάτω.

Κάθε 24 μετρήσεις της θερμοκρασίας περιβάλλοντος, δηλαδή κάθε 5 δύο (2) λεπτά, υπολογίζεται η μέση τιμή των 24 αυτών μετρήσεων. Η μέτρηση αυτή τυπώνεται στην οθόνη LCD και παραμένει για δέκα (10) δευτερόλεπτα στην οθόνη και δεν μπορεί να τυπωθεί κάτι άλλο κατά τη διάρκεια αυτών των δέκα δευτερολέπτων (παρά μόνο στην περίπτωση που κάποιος πλησιάσει κοντά στον αισθητήρα εγγύτητας). Αν η μέση τιμή είναι μεγαλύτερη από μία συγκεκριμένη τιμή **extremeFan**, τότε αποστέλλεται e-mail το οποίο πληροφορεί τον παραλήπτη για την ενεργοποίηση του αισθητήρα. Το διάστημα των δέκα δευτερολέπτων τηρείται με τη συνθήκη $i > 0$ το οποίο επιτρέπει τη νέα εκτύπωση στην οθόνη ύστερα από 2 επαναλήψεις των 5 δευτερολέπτων, δηλαδή ύστερα από 10 δευτερόλεπτα.

Επιπλέον, το πρόγραμμα αυτό περιλαμβάνει και την υλοποίηση ενός Software Interrupt με τη χρήση της βιβλιοθήκης *Ticker.h*. Η βιβλιοθήκη αυτή κάνει χρήση των *itInterrupts* του ESP8266 για να προσεγγίσει τη λειτουργία ενός Timer Overflow Interrupt. Έτσι κατασκευάζεται μία συνάρτηση η οποία να καλείται κάθε 0.5 δευτερόλεπτα και να διακόπτει το κυρίως πρόγραμμα για να εκτελέσει κάποιες λειτουργίες. Συγκεκριμένα, αυτή χρησιμοποιείται για τη μέτρηση της απόστασης του κοντινότερου αντικείμενου από τον αισθητήρα εγγύτητας. Συγκεκριμένα, αυτή χρησιμοποιείται για τη μέτρηση της απόστασης του κοντινότερου αντικείμενου από τον αισθητήρα εγγύτητας. Αν η απόσταση αυτή είναι μικρότερη από μία συγκεκριμένη μέση τιμή (10 cm) τότε τυπώνεται στην οθόνη LCD η προηγούμενη μέση τιμή της θερμοκρασίας και η τελευταία μέτρηση της θερμοκρασίας που πραγματοποιήθηκε.

```
1 const int SMTP_PORT = 465;
2 const char* SMTP_SERVER = "smtp.gmail.com";
3 const char* EMAILBASE64_LOGIN = "e-mail encoded in BASE64";
4 const char* EMAILBASE64_PASSWORD = "password encoded in BASE64";
5 const char* FROM = "mcu.2019.wemos@gmail.com";
```

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

Ο παρακάτω κώδικας, παρουσιάζει μέρος του συνολικού κώδικα και περιγράφει πως υλοποιείται ο **software interrupt**. Στη συνάρτηση *setup()* δηλώνεται ότι κάθε 0.5 δευτερόλεπτα θα διακόπτεται η ροή του προγράμματος για να εκτελεσθεί η συνάρτηση *interruptVector()*. Η συνάρτηση αυτή αν αναγνωρίσει ότι κάποιο αντικείμενο βρίσκεται σε απόσταση κάτω των 10 εκατοστών τότε τυπώνει στην LCD οθόνη την τελευταία μέτρηση που πραγματοποιήθηκε σύγουρα (i-1) και την τελευταία μέση τιμή.

```

1  /* include Libraries */
2 #include <Ticker.h>
3 .
4 .
5 .
6 /* Global Variable Declaration */
7 Ticker INTERRUPT;
8 .
9 .
10 .
11 /* interruptVector(): this function checks periodically every 0.5 seconds
12 * if someone is nearby. Then it prints average Temperature
13 * and the latest temperature measurement.
14 */
15 void interruptVector(){
16
17 // check if someone is close
18 isSomeoneClose = checkProximity();
19 String line1, line2;
20 if( isSomeoneClose == 1){
21     line1 = String("AVG: " + String( averageTemperature, 3));
22     if(i == 0)
23         line2 = String("Last TEMP: " + String(temperature[24], 3));
24     else
25         line2 = String("Last TEMP: " + String(temperature[i-1], 3));
26
27     printLCD( line1, line2);
28 }
29 }
30 .
31 .
32 .
33 /* setup() function */
34 void setup(){
35
36     /* Setup ticker function */
37     INTERRUPT.attach( 0.5, interruptVector);
38
39 }
```

Το κυρίως πρόγραμμα εκτελείται στη συνάρτηση void loop(). Το διάγραμμα στο Σχήμα 7 περιγράφει τη συνάρτηση αυτή. Η συνάρτηση checkExtremeValues() είναι υπεύθυνη για τον έλεγχο των ακραίων τιμών extremehigh, extremeLow και η συνάρτηση checkExtremeFan() είναι υπεύθυνη για την ενεργοποίηση του ανεμιστήρα και την αποστολή του e-mail.

```

1 /* loop(): this function is the main function. It executes
2 * continuously.
3 */
4 void loop(){
5     sum = 0;                                // this variable holds the sum of temperatures
6
7     /* 2 minutes Loop */
8     Serial.println("\n\n\nNew Measurement");
9     for( i = 0; i < 24; i++){
10         delay(5*1000);
11         temperature[i] = readTemperature();
12         sum = sum + temperature[i];
13         checkExtremeValues( temperature[i], i);
14     }
15
16     /* Calculate Average Temperature */
17     averageTemperature = sum/24;
18     String line1 = String("Avg. Temp.");
19     String line2 = String(averageTemperature, 3);
20     printLCD( line1, line2);
21     Serial.println("\n\n\n Average Temperature: " + String(averageTemperature, 3));
22
23     //check for fan
24     checkExtremeFan( averageTemperature);
25
26 }
```

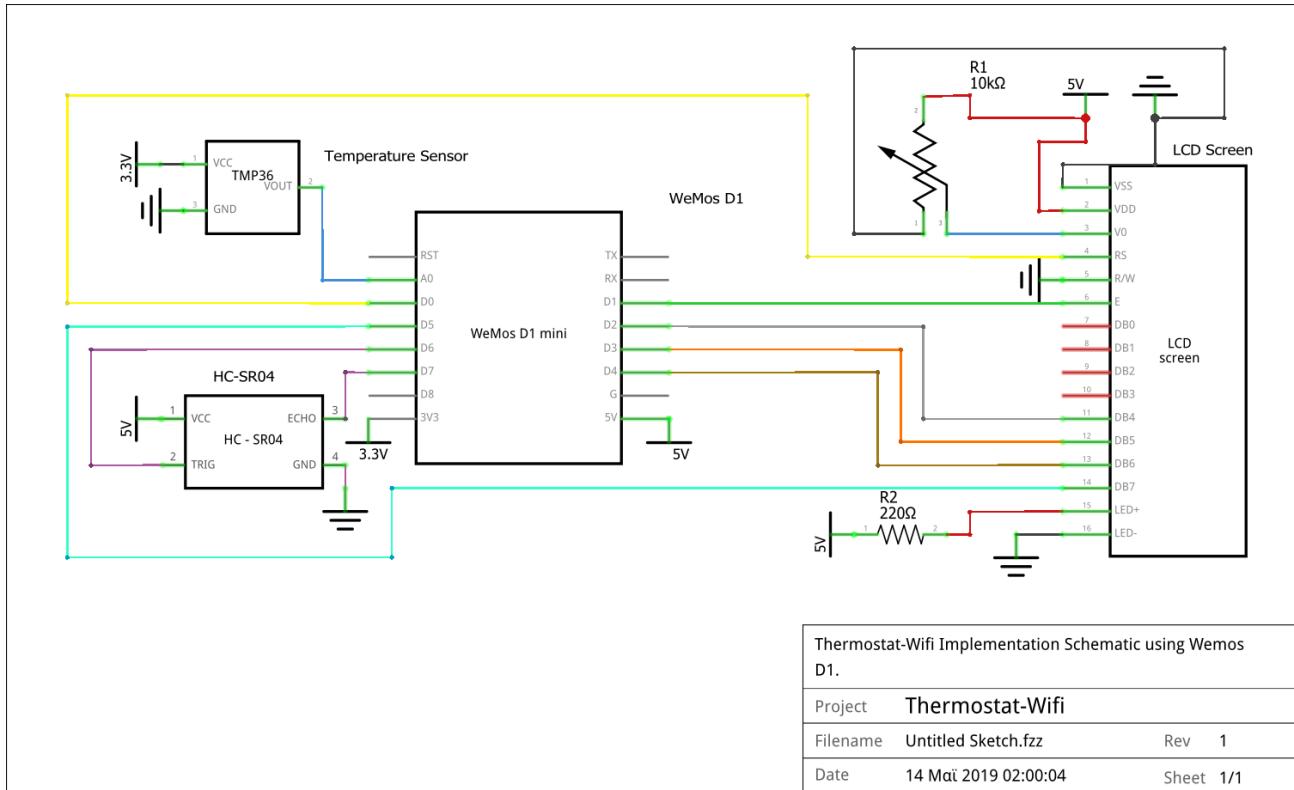
Το πρόγραμμα αποτελείται από 3 αρχεία. Το αρχείο *thermostat-Wifi.ino* περιλαμβάνει το κύριο πρόγραμμα ενώ τα άλλα δύο αρχεία *Gsender.cpp*, *Gsender.h* υλοποιούν τις ρυθμίσεις για την επικοινωνία με τον **SMTP Server**. Τα δύο τελευταία αρχεία πάρθηκαν από τη σελίδα [5] και τροποποιήθηκαν σε πρόσφατες ρυθμίσεις διότι δεν ήταν πλέον λειτουργικά.

3.2 Υλικό και συνδεσμολογία.

Για την υλοποίηση του παραπάνω προγράμματος χρησιμοποιήθηκαν τα εξής εξαρτήματα:

- Wemos D1 R2(ESP8266)
- Αισθητήρας θερμότητας TMP36 [1]
- Οθόνη LCD διαστάσεων 16x2 JHD659 M10 1.1 [2]
- Αισθητήρας εγγύτητας HC-SR04 [3]
- Ποτενσιόμετρο 10 kΩ

Τα παραπάνω εξαρτήματα συνδέθηκαν σύμφωνα με την εξής συνδεσμολογία:



Σχήμα 8: Συνδεσμολογία εξαρτημάτων με μικροελεγκτή *Weemos D1 R2*.

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

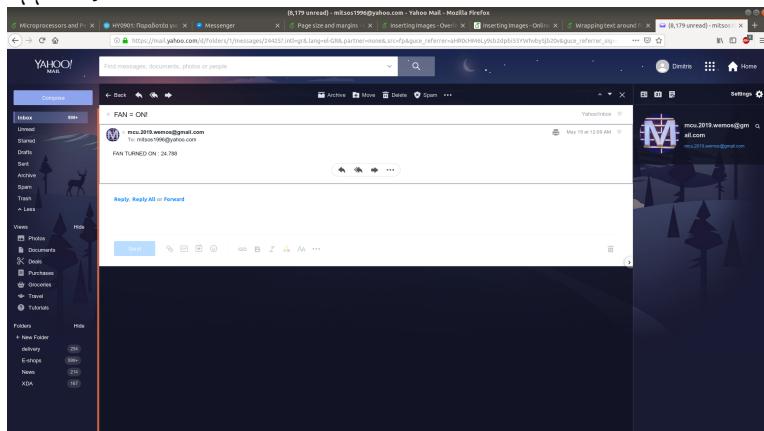
Σημείωση: στο σχηματικό περιέχεται ο μικροελεγκτής Wemos D1 mini, ο οποίος, ωστόσο, είναι πανομοιότυπος με το Wemos D1 R2, με τη διαφορά ότι είναι πιο μικρό το μέγεθος του.

3.3 Αποτελέσματα εργασίας του μικροελεγκτή Wemos D1 R2.

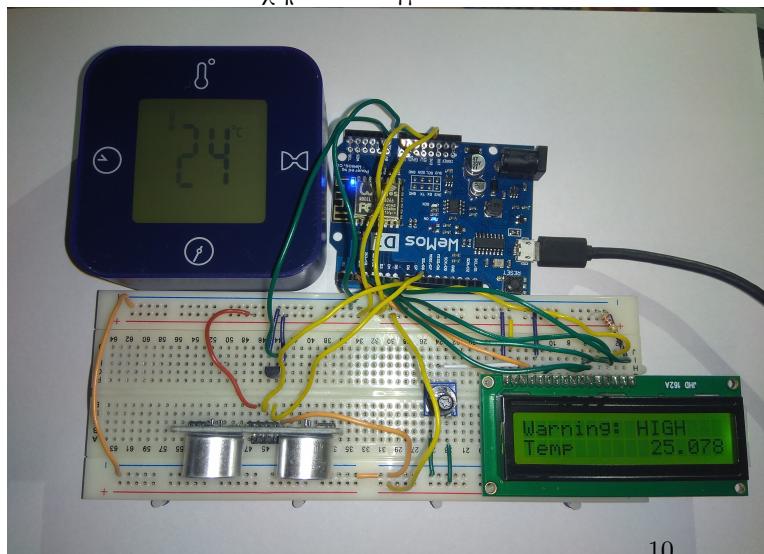
The screenshot displays two terminal windows showing the serial communication log. The logs show various sensor readings and control commands related to the Wemos D1 R2 microcontroller setup. Key entries include:

- Temperature sensor readings: "Temp: 24.773", "Temp: 24.775", "Temp: 24.776".
- Humidity sensor readings: "Humidity: 41.32", "Humidity: 41.33".
- Fan control: "Fan ON", "Fan OFF".
- Sensor threshold crossings: "Fan turned ON", "Fan turned OFF".
- Timestamps indicating when the fan state changed. For example, "Fan turned ON at 2019-05-10T12:09:00Z" and "Fan turned OFF at 2019-05-10T12:09:54Z".

Σχήμα 9: Παράθυρα σειριακής επικοινωνίας χατά την εκτέλεση του προγράμματος.



Σχήμα 10: Ληφθέν email.



Αριστερά στις δύο εικόνες φαίνεται η εκτέλεση του προγράμματος από την αρχή. Σημειώνεται ότι η τιμή **extremeFan** τέθηκε ίση με 23. Συνεπώς, για μέση τιμή θερμοκρασίας 23 και πάνω ενεργοποιείται ο ανεμιστήρας (**bool FAN**) και αποστέλλεται e-mail.

Στις δύο εικόνες αριστερά, αρχικά φαίνεται ότι επιτυγχάνεται η σύνδεση στο WiFi και τυπώνεται η IP που λαμβάνει ο μικροελεγκτής. Στη συνέχεια εκκινεί η διαδικασία μετρήσεων. Όλες οι εκτυπώσεις στην ουδόνη είναι αριθμημένες. Πραγματοποιούνται 24 μετρήσεις. Κάθε μέτρηση εκτυπώνει τον αριθμό που μετράει ο ADC Converter. Η μέτρηση αυτή κυμαίνεται από 0 έως 1023. Η μέτρηση αυτή μετατρέπεται σε Volts και αντίστοιχα τυπώνεται η θερμοκρασία. [1]

Μόλις πραγματοποιηθούν 24 μετρήσεις τυπώνεται η μέση τιμή. Στη συνέχεια ελέγχεται αν πρέπει να ενεργοποιηθεί ο ανεμιστήρας. Εφόσον ο ανεμιστήρας είναι απενεργοποιημένος και η μέση τιμή είναι μεγαλύτερη από extremeFan τότε ενεργοποιείται ο ανεμιστήρας και αποστέλλεται το αντίστοιχο e-mail.

Αριστερά των εκτυπώσεων υπάρχει το αντίστοιχο timestamp. Τη χρονική στιγμή **00:09**, δηλαδή 12:09 π.μ. αποστέλλεται το σχετικό e-mail.

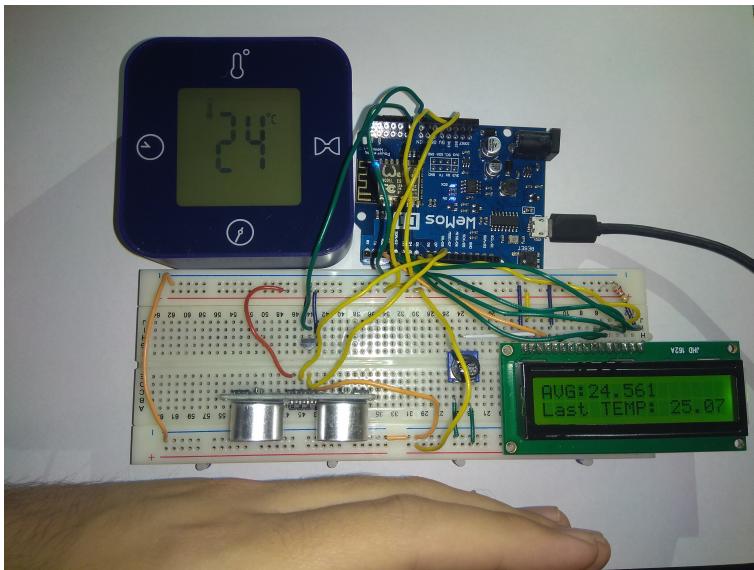
Στην επόμενη εικόνα φαίνεται το e-mail λήφθηκε στις **12:09** π.μ. Επίσης το e-mail περιλαμβάνει και τη μέση τιμή που υπολογίσθηκε. Το e-mail ενημερώνει τον παραλήπτη ότι ο ανεμιστήρας ενεργοποιήθηκε.

Στο Σχήμα 9, επιπλέον, φαίνεται ότι ενώ είναι ήδη ενεργοποιημένος ο ανεμιστήρας δεν αποστέλλεται νέο e-mail.

Ακόμη, ανάμεσα στις εκτυπώσεις, υπάρχουν και δύο εκτυπώσεις οι οποίες πληροφορούν ότι κάποιος βρίσκεται κοντά στον αισθητήρα εγγύτητας γεγονός που πυροδοτεί την εκτύπωση της προηγούμενης μέσης τιμής και της τελευταίας τιμής που διαβάστηκε.

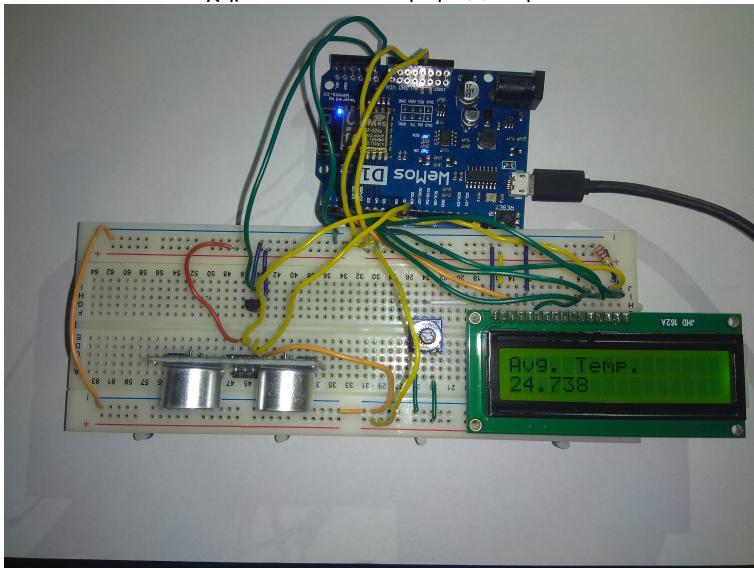
Όλες οι μετρήσεις έχουν μεταξύ τους 5 δευτερόλεπτα όπως επιβεβαιώνεται και από το Σχήμα 9. Ακόμη, οι δύο εκτυπώσεις που πληροφορούν ότι κάποιος βρίσκεται κοντά έχουν μεταξύ τους χρονική απόσταση 0.5 δευτερόλεπτα, γεγονός που επιβεβαιώνει ότι λειτουργεί σωστά η συγκεκριμένη λειτουργία. Το γεγονός αυτό επιβεβαιώνεται και από το Σχήμα 12

Σημείωση: στη φωτογραφία αριστερά περιλαμβάνεται και ένα ψηφιακό θερμόμετρο το οποίο πιστοποιεί την ορθότητα των μετρήσεων.



Στο Σχήμα 13 φαίνεται η λειτουργία κατά την οποία εκτυπώνεται η μέση τιμή της θερμοκρασίας ύστερα από 24 μετρήσεις (2 λεπτά).

Σχήμα 12: Ειδοποίηση εγγύτητας.



Σχήμα 13: Ειδοποίηση μέσης τιμής.

4 Επίλογος.

Συνοψίζοντας, στην αναφορά αυτή παρουσιάσθηκαν δύο υλοποιήσεις που απαντούν πλήρως τα ζητούμενα της εργασίας. Η πρώτη αποτελεί πλήρη λύση χωρίς την υλοποίηση του προαιρετικού ερωτήματος αναφορικά με την αποστολή του e-mail. Η δεύτερη υλοποίηση, εμπεριέχει το προαιρετικό ερώτημα της αποστολής του e-mail, ωστόσο λόγο περιορισμένου αριθμού GPIOs δεν περιλαμβάνει τα LEDs.

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>.
- [4] AVR. ATmega328P Datasheet. http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.
- [5] Boris Shobat. ESP8266 GMail Sender. <https://www.instructables.com/id/ESP8266-GMail-Sender/>.