Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρονικής και Υπολογιστών

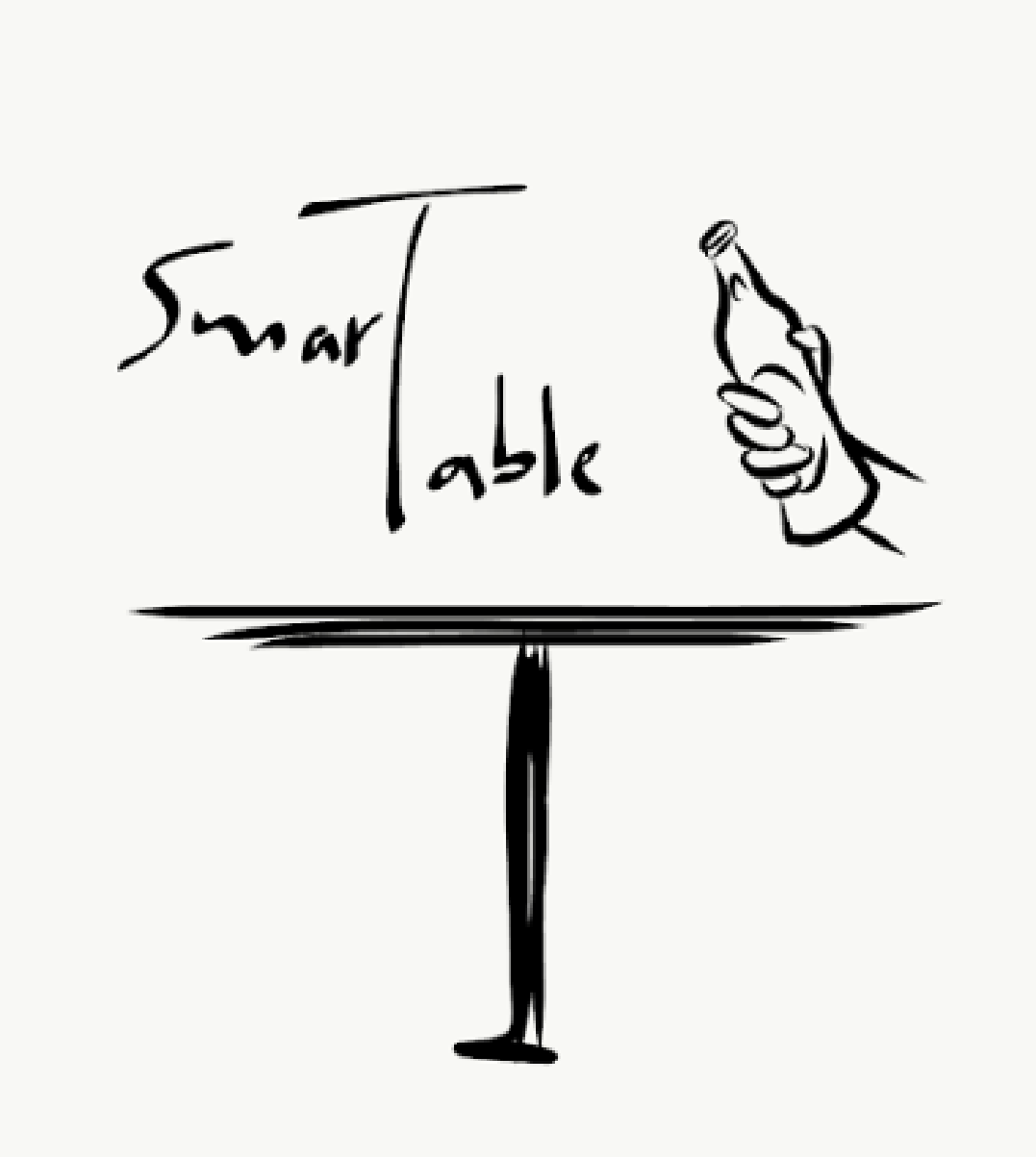
ΕΡΓΑΣΙΑ ΣΤΙΣ ΕΦΑΡΜΟΓΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΔΙΑΤΑΞΕΩΝ

των

ΜΑΝΟΥΣΑΡΙΔΗ ΙΩΑΝΝΗ ΑΕΜ 8855
ΜΥΡΣΙΝΙΑ ΑΝΤΩΝΗ ΑΕΜ 8873

Εξάμηνο 8^{ο} – Μάιος 2019

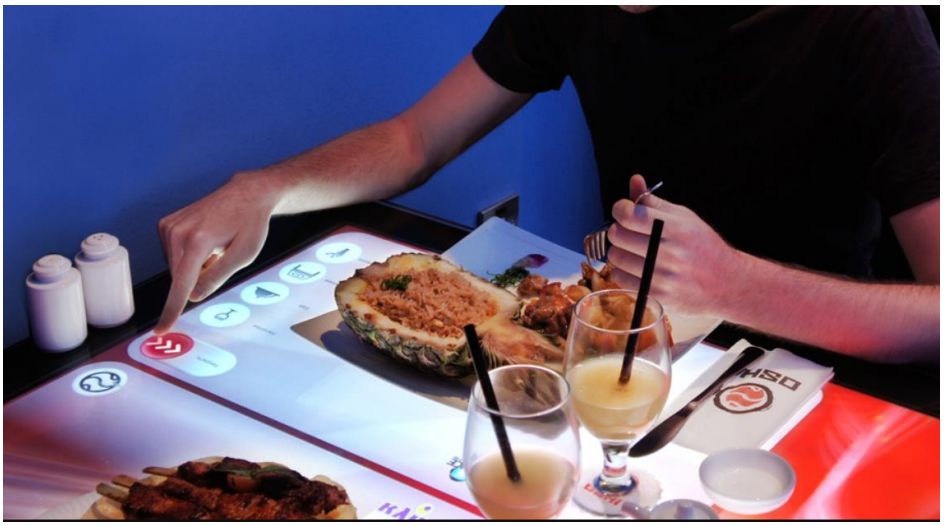
SmarTable



Περιεχόμενα

1. Περιγραφή εφαρμογής

Η εφαρμογή αποσκοπεί στη δημιουργία ενός έξυπνου εστιατορίου/μαγαζιού. Σε κάθε τραπέζι του μαγαζιού τοποθετούνται τα Arduino-πομποί. Αυτοί συλλέγουν πληροφορίες και τις αποστέλλουν σε ένα κεντρικό ώστε να εξυπηρετούνται πιο γρήγορα και αποτελεσματικά οι πελάτες.



Αρχικά, ο κάθε πομπός-Arduino έχει ένα μετρητή θερμοκρασίας. Μόλις, η θερμοκρασία ανέβει πάνω από ένα συγκεκριμένο όριο, ανοίγει αυτόματα ο ανεμιστήρας ή το κλιματιστικό του τραπεζιού, προκειμένου να ανακουφιστεί ο πελάτης από την ζέστη. Στα πλαίσια της εργασίας, η ενεργοποίηση του ανεμιστήρα/κλιματιστικού αντιστοιχεί στο LED 6.

Έπειτα, υπάρχει ένας αισθητήρας φωτεινότητας-ldr. Ουσιαστικά, ο συγκεκριμένος αισθητήρας χρησιμοποιείται για να αντιλαμβάνεται πότε έχει σκοτεινιάσει και να ανάβει το ηλεκτρικό κερί του τραπεζιού. Στα πλαίσια της εργασίας, το κερί αντιστοιχεί στο LED 9.

Στη συνέχεια, δίνεται η δυνατότητα του πελάτη να παραγγείλει με έξυπνο, γρήγορο και εύκολο τρόπο. Πατώντας το κουμπί που αντιστοιχεί στο pin 5, ανοίγει την LCD. Έπειτα, γυρνώντας το ποτενσιόμετρο, εμφανίζονται στην LCD οι διαθέσιμες επιλογές του μενού. Εφόσον, αποφασίσει τι επιθυμεί, μπορεί αρχικά με το κουμπί που αντιστοιχεί στο pin 3, να το επιλέξει και κατόπιν να το παραγγείλει.

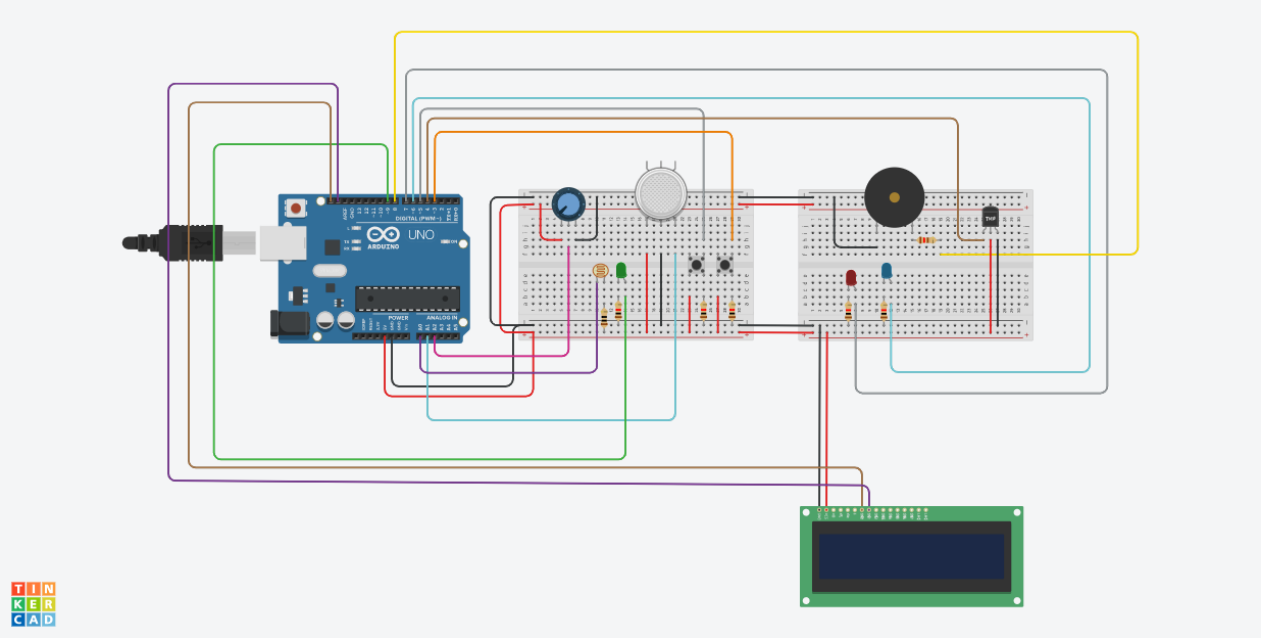
Περαιτέρω, στο τραπέζι υπάρχει ένας ανιχνευτής καπνού. Μόλις ένας πελάτης καπνίσει αυτός ενεργοποιεί το buzzer και αναβοσβήνει το κόκκινο LED.

Τέλος, όλες οι πληροφορίες σχετικά με το αν έχει παραγγείλει κάτι ο χρήστης, αν καπνίζει, η φωτεινότητα και η θερμοκρασία του τραπεζιού, αποστέλλονται στον κεντρικό κόμβο. Από εκεί ο κεντρικός κόμβος μπορεί να διαβάζει τις πληροφορίες αυτές για το κάθε τραπέζι και να ανταποκρίνεται ανάλογα προς την καλύτερη εξυπηρέτηση των πελατών.

2. Κύκλωμα εφαρμογής

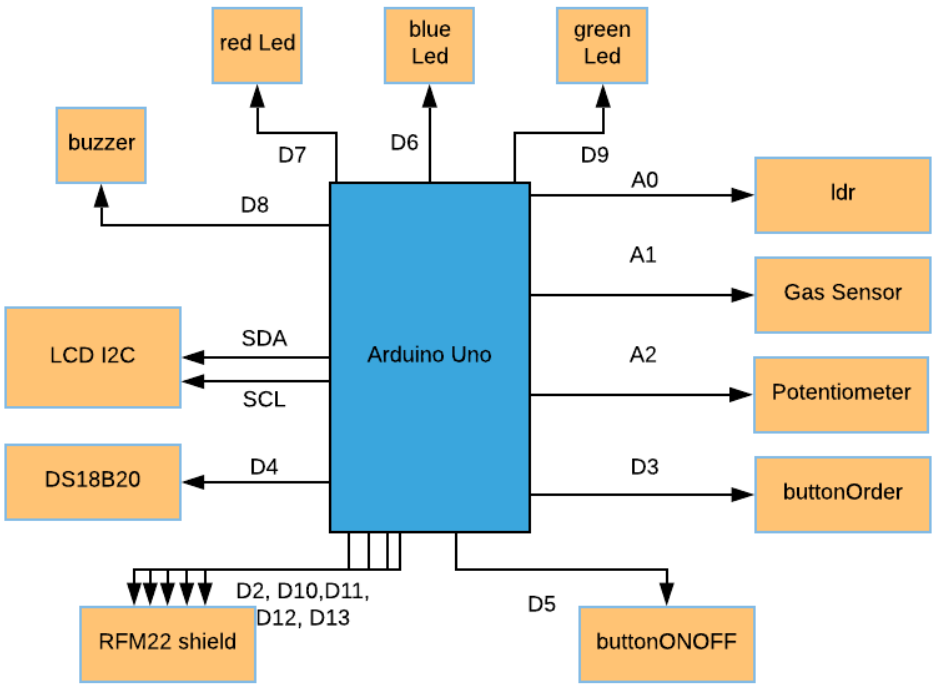
2.1 Hardware design

Παρακάτω φαίνονται οι συνδέσεις του κυκλώματος της εφαρμογής. Απουσιάζει το RFM-22 shield από το σχήμα, το οποίο τοποθετείται ακριβώς από πάνω από το Arduino.



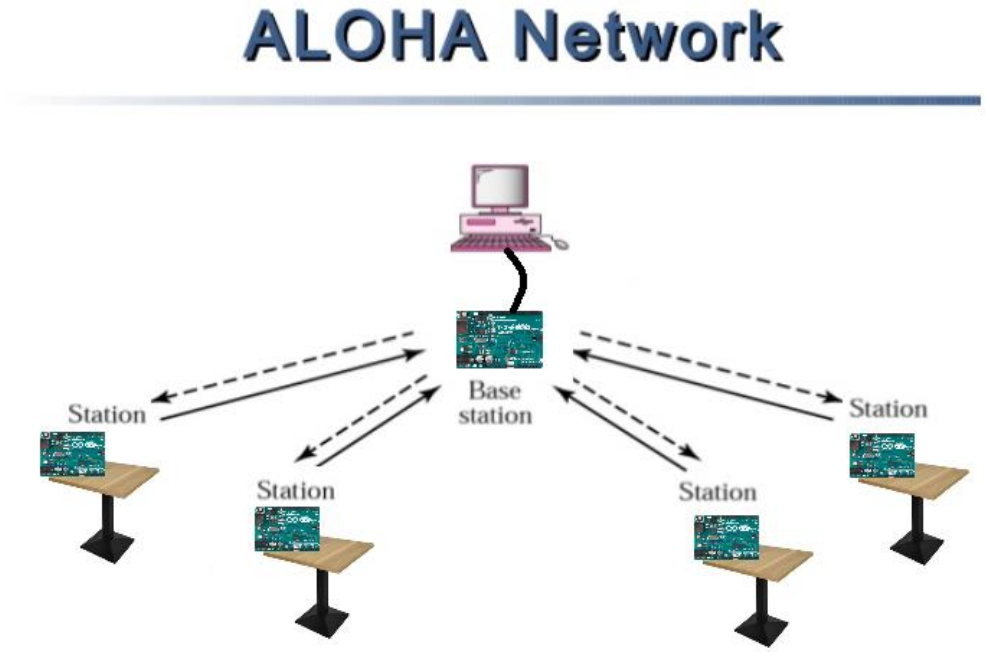
2.2 Diagram design

Εδώ παρουσιάζονται οι ακριβείς συνδέσεις των pin του Arduino με τις διάφορες περιφερειακές συσκευές.



3. Ανάλυση σχεδίασης του δικτύου

Στα πλαίσια της εργασίας, χρησιμοποιήθηκε το πρωτόκολλο ALOHA, ή αλλιώς το pure ALOHA. Σε αυτό, όταν ο χρήστης έχει κάποιο πακέτο να στείλει, το στέλνει αμέσως. Αν γίνει σύγκρουση με κάποιο άλλο πακέτο, τότε περιμένει για κάποιο τυχαίο χρονικό διάστημα (ALOHA_DELAY) και επανεκπέμπει το πακέτο.



3.1 Υλοποίηση δικτύου

Για την υλοποίηση του Wireless Sensor Network (WSN) χρησιμοποιήθηκε η ασύρματη κάρτα δικτύου RFM-22, προσαρτημένη σε ένα shield για μεγαλύτερη ευκολία στη χρήση. Συγκεκριμένα, κάθε πομπός και δέκτης συνδεδεμένο πάνω του ένα RFM22-shield.

3.2 Ανάλυση κριτηρίων χωρητικότητας, κάλυψης και καναλιών

Στην εφαρμογή μας θέλουμε να δίνουμε τη δυνατότητα να εξυπηρετούνται όσο το δυνατόν περισσότερα τραπέζια. Το κάθε τραπέζι πρέπει να έχει έναν πομπό-Arduino.

- Έστω λ η μέση τιμή των πακέτων ανά πομπό ανά μονάδα χρόνου (πακέτα/χρόνο), όπου λ = 4 πακέτα/ second.
- Έστω k το πλήθος των τραπεζιών, όπου θεωρούμε ότι το k = 20 σε ένα μικρό μαγαζί.
- Άρα ο συνολικός ρυθμός παραγωγής πακέτων r στο δίκτυο είναι kλ, r = kλ = 80 πακ/sec.
- Αν τ είναι η διάρκεια κάθε πακέτου (10ms στο παράδειγμα μας), τότε το κανάλι θα είναι κατειλημμένο για rτ, με rτ = 0.8 erlangs
- Αν μπορούσαμε να χωρέσουμε ακριβώς τα πακέτα το ένα μετά το άλλο, τότε θα ικανοποιούσαμε οριακά τη σχέση rτ=1, ή διαφορετικά ο μέγιστος ρυθμός μετάδοσης θα ήταν r=1/τ = 100 πακέτα το δευτερόλεπτο.
- Με το ALOHA, μπορώ να αξιοποιήσω το πολύ το 18.6% του διαθέσιμου καναλιού (χρόνου), διαφορετικά θα έχω τόσα collisions που θα στέλνω λιγότερα αληθινά data.

  rt = 0.8 erl
  max (για το ALOHA) = 0.186

  $$\frac{0.8}{0.186} = 4.301 \text{ Άρα χρειαζόμαστε 5 κανάλια τουλάχιστον}$$

Εδώ να σημειωθεί ότι θα μπορούσε να χρησιμοποιηθεί κάποιο άλλο πρωτόκολλο για τη μείωση των αριθμών των καναλιών όπως είναι το slotted ALOHA ή το non persistent CSMA.

Στο slotted ALOHA, έχουμε συγχρονισμό και οι εκπομπές των πακέτων γίνονται μόνο στην αρχή των slots και επομένως επιτρέπεται η αξιοποίηση του 36.8% του διαθέσιμου καναλιού. Αυτό έχει ως άμεση συνέπεια να χρειαζόμαστε 3 κανάλια τουλάχιστον αφού:

$$\frac{0.8}{0.368} = 2.174$$

Στο non persistent CSMA ακολουθείται η εξής διαδικασία:

1. Ακούω το κανάλι
2. Αν διαθέσιμο (ready terminal), στέλνω αμέσως
3. Αν δεν πάρω απάντηση, ξεκινώ τυχαία καθυστέρηση και ξανακούω το κανάλι μετά την καθυστέρηση (Βήμα 1)

Αν στο δίκτυο μας χρησιμοποιήσουμε αυτό το πρωτόκολλο με χρόνο καθυστέρησης διάδοσης τ =10^{-5} sec και μέση διάρκεια αποστολής των πακέτων όπως και πάνω 10^{-2}sec, τότε το για το α ισχύει ότι

$$\alpha = \frac{\tau}{T} = \frac{10^{-5}}{10^{-2}} = 0.001$$

Αν το G = 0.5 (για G = 0.5 έχω μέγιστο S = 0.184 στο ALOHA), τότε έχουμε ότι:

$$S = \frac{Ge^{-aG}}{G(1+2a)+e^{-aG}} = 0.333 \Rightarrow \frac{0.8}{0.333} = 2.402\,,$$ άρα θα χρειαστούμε 3 κανάλια

- Τα μηνύματα που στέλνονται είναι ένα String 36 χαρακτήρων από κάθε τραπέζι που ισοδυναμεί με bits=288 bits (θεωρώντας ότι κάθε χαρακτήρας αποτελείται από 8 bits). Ως αποτέλεσμα θα θέλουμε ρυθμό

$$B = \frac{288}{0.1} = 2880\ bps = 2.88\ kbps$$

- Στην περίπτωση της δικιάς μας εφαρμογής πρέπει σε κάθε τραπέζι του μαγαζιού να τοποθετηθεί και από μία συσκευή Arduino-πομπός. Αυτό που μας απασχολεί είναι η ελάχιστη ισχύς του πιο απομακρυσμένου κόμβου, P_{min}, καθώς και η απόσταση αυτού του κόμβου, r_{max}. Γι' αυτήν έχουμε ότι απαιτείται κάλυψη του χειρότερου/πιο απομακρυσμένου τραπεζιού για το 95% του χρόνου. Άρα:

$$\text{erf}\left(\frac{\gamma}{\sigma\sqrt{2}}\right) = 2 \cdot 0.95 - 1 \Rightarrow \frac{\gamma}{\sigma\sqrt{2}} = 1.156 \Rightarrow \gamma = 1.644\sigma = 16.44dB$$

Θεωρώντας σ=10dB και:

$$P_{sens} = -130dBm + 10log_{10}(B) = -95.406$$

θα πρέπει στον πιο απομακρυσμένο δέκτη (απόσταση $r_{max} = 50m$)να εξασφαλιστεί ισχύς λήψης:

$$P_{\min} = P_{\text{sens}} + \gamma = -95.406 + 16.44 = -61.5dBm$$

$$r_{max} = \sqrt{\frac{10^{-6}}{10^{-6.15}}\,25^2} = 29.67m$$

(Τα νούμερα είναι ενδεικτικά)

4. Ψευδοκώδικας κόμβων

Στη συνέχεια παρατίθεται ψευδοκώδικας για την περιγραφή της εφαρμογής.

4.1 Transmitter Node

Include libraries

Initialize and set variables
Initialize Network Card and HW resources

Loop_forever

// Step-1
Measure Ldr
If ldr Value < 50
    Turn on candle LED

//Step-2
Measure gas
If gas value > 150
    Turn on red LED and hit buzzer

//Step-3
Measure Temperature
If temperature value > 30
    Turn on fan LED

//Step-4
If button ONOFF is pressed
    Turn on or off LCD respectively

//Step-5
If potentiometer value changed or select button pressed (interrupt)
    Update LCD info accordingly

//Step-6
Try to send data
If success
    Go to Step1
Else if failed
    Create a random number and store it in ALOHA_delay
    Initial_time = get current time
    While (get current time – Initial_time < ALOHA_delay)
        Do steps 1 to 5
    Go to step-6 and try to send the data again

End_Loop_forever

4.2 Receiver Node

Include libraries

Initialize and set variables
Initialize Network Card
Set to accept data from many sources

Loop_forever
Read received data
End_Loop_forever

4.3 Transmitter-Helper Node

Include libraries

Initialize and set variables
Initialize Network Card

Loop_forever
Send Random data
End_Loop_forever

5. Κώδικας

"Talk is cheap. Show me the code" Linus Torvalds

5.1 Ανάλυση Κώδικα

5.1.1 Transmitter Code

Στην αρχή του κώδικα γίνονται include οι απαραίτητες βιβλιοθήκες και αρχικοποιούνται οι μεταβλητές και το hardware που θα χρειαστούν. Αυτά αφορούν την κάρτα δικτύου, το ldr, τον MQ2, τα διάφορα LED, το DS18B20, το buzzer, τα buttons και την LCD.

void setup()

Η setup() ουσιαστικά κάνει όλες τι αρχικοποιήσεις σχετικά με τα pin εισόδου/ εξόδου, την κάρτα δικτύου, ενώ τέλος θέτει σε όλες τις πιθανές εξόδους τιμές LOW και στις λογικές μεταβλητές τιμές false, έτσι ώστε να αποφευχθούν ανεπιθύμητες αρχικές συνθήκες. Επίσης, αρχικοποιεί και ένα interrupt για το button που θα κάνει select και order ένα προϊόν.

void loop()

Η δομής της είναι πολύ απλή και ουσιαστικά καλεί 6 συναρτήσεις με τη σειρά που είναι οι εξής:

- regulateCandle()
- checkSmoke()
- checkTemperature()
- checkLCD()
- ONOFFButtonPressed()
- sendData()

void checkLCD()
Ελέγχει αν η οθόνη πρέπει να ναι ανοιχτή ή όχι. Στην πρώτη περίπτωση καλεί την printLCD(), ενώ στη δεύτερη σβήνει την οθόνη και μηδενίζει τις μεταβλητές orderState, productID και θέτει σε τιμή false την productOrderFlag.

printLCD()
Ανάλογα με την τιμή του ποτενσιόμετρου και την τιμή της μεταβλητής orderState, προβάλλει στην LCD το αντίστοιχο προϊόν μαζί με την κατάσταση: τίποτα αν δεν έχει επιλεχθεί κάτι, selected – αν επιλέχθηκε κάτι- , ordered –αν ο πελάτης παρέγγειλε κάτι.

regulateCandle()
Ελέγχει την φωτεινότητα και ανάλογα με το όριο που έχει οριστεί ανάβει το candle LED ή όχι.

checkSmoke()
Ο MQ-2 παίρνει μετρήσεις και αν έχει ξεπεραστεί το όριο της επιτρεπτής τιμής καπνού, τότε καλεί τη συνάρτηση hitAlarm(). Ειδάλλως θέτει σε τιμή LOW το danger Led, σε τιμή false τη μεταβλητή gasDanger και απενεργοποιεί τον ήχο από το buzzer μέσω της noTone().

hitAlarm()
Ουσιαστικά η συνάρτηση αυτή αναβοσβήνει το κόκκινο LED και χτυπά το buzzer σε τακτά χρονικά διαστήμα για όση διάρκεια το gas value υπερβαίνει το επιτρεπτό όριο.

checkTemperature()
Ελέγχει την θερμοκρασία και ανάβει το LED του ανεμιστήρα αν τιμή ξεπερνά το όριο που έχει οριστεί.

ONOFFButtonPressed()
Ρυθμίζει τη λειτουργία του on – off button για να ανοιγοκλείνει ο χρήστης την οθόνη.

orderButtonPressed()
Καλείται όταν γίνεται interrupt με το button order και ανανεώνει την τιμή του orderState. Το pin για το interrupt πρέπει να είναι το pin 3 αναγκαστικά αφού το 2 χρησιμοποιείται από το shield της RFM22.

sendData()
Η συνάρτηση αυτή είναι και η βασική συνάρτηση για την υλοποίηση της επικοινωνίας μέσω του πρωτοκόλλου ALOHA. Στην αρχή όταν έχει δεδομένα τα αποστέλλει κατευθείαν. Αν η αποστολή δεν είναι επιτυχής, τότε γίνεται μία επανάληψη μέχρι να περάσει ένα χρονικό διάστημα Α. Στο χρονικό διάστημα αυτό Α καλείται συνεχώς η συνάρτηση checkAll(). Μόλις, παρέλθει το χρονικό διάστημα Α, γίνεται ξανά προσπάθεια αποστολής του πακέτου. Η sendData() στέλνει όσα πακέτα έχουν οριστεί από την μεταβλητή max_counter.

checkAll()
Η δομής της είναι πολύ απλή και ουσιαστικά καλεί 5 συναρτήσεις με τη σειρά που είναι οι εξής:

- regulateCandle()
- checkSmoke()
- checkTemperature()
- checkLCD()
- ONOFFButtonPressed()

** Τα κουμπιά για λόγους ταχύτητας θα είχαν γίνει και τα δύο με interrupt, ωστόσο το ένα από τα δύο διαθέσιμα pins δεν είναι διαθέσιμο εξαιτίας του shield της κάρτας δικτύου.

5.1.2 Receiver Code

Ο κώδικας αυτός είναι μία απλή υλοποίηση του reveiver για το πρωτόκολλο ALOHA, με τη κάρτα δικτύου RFM22 , όπως διδάχθηκε στο μάθημα Εφαρμογές Τηλεπικοινωνιακές Διατάξεις του Α.Π.Θ. το εαρινό εξαμήνο του 2019. Ο κώδικας απλά διαβάζει τις τιμές που δέχεται και τις τυπώνει.

5.1.3 Transmitter- helper Code

Απλή υλοποίηση ενός transmitter, ως βοηθητικός για να αποστέλλει random data, καθώς δεν μπορούσαν να υλοποιηθούν και άλλοι κόμβοι λόγω των περιορισμών του hardware.

5.2 GitHub

Ο κώδικες και όλα τα απαραίτητα αρχεία βρίσκονται στο παρακάτω repo στο GitHub:

https://github.com/imanousar/AUTh/tree/master/Wireless%20Sensor%20Network

6. Video

6.1 Promotion video

Στον παρακάτω σύνδεσμο μπορείτε να βρείτε το promotion video της εφαρμογής μας.

https://www.youtube.com/watch?v=ae23E55At5Y

6.2 Demo video

Στον παρακάτω σύνδεσμο μπορείτε να βρείτε το demo video της εφαρμογής μας.

https://www.youtube.com/watch?v=B45d_8eJRw4