

# ΤΕΧΝΙΚΗ ΑΝΑΦΟΡΑ

## ΕΦΑΡΜΟΓΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΔΙΑΤΑΞΕΩΝ



ΝΕΝΟΣ ΔΗΜΗΤΡΗΣ ΑΕΜ 9012

ΚΟΥΤΖΙΑΜΠΑΣΗΣ ΔΗΜΗΤΡΗΣ ΑΕΜ 9012

ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ 2019-2020

## Περιεχόμενα

Περιγραφή του προβλήματος.....	3
Ανάλυση σχεδίασης του δικτύου.....	3
Υλοποίηση.....	4
1. Αισθητήρας θερμοκρασίας για το χώρο του κοτετσιού .....	4
2. Αισθητήρας υγρασίας-θερμοκρασίας για το χώρο των αυγοαποθηκών .....	6
3. Αισθητήρας δύναμης στις ταΐστρες .....	9
4. Επικοινωνία πομπών με τον κεντρικό σταθμό .....	12
5. Λήψη δεδομένων από το δέκτη (κεντρικό σταθμό) .....	16
Πρακτικότητα .....	21
Εναλλακτικές Σχεδίασης .....	21

## Περιγραφή του προβλήματος

Στην παρούσα εργασία στόχος μας είναι να φτιάξουμε ένα σύστημα αυτόματης σίτισης πουλερικών και επώασης των αυγών τους κάτω από τις κατάλληλες συνθήκες.

Το σύστημα αυτό μπορεί να εφαρμοστεί για μεγάλο αριθμό πουλερικών κάθε είδους, καθώς θα αποτελείται από μεγάλο αριθμό εκκολαπτικών μηχανών και χώρων αποθήκευσης τροφής. Διευκολύνει τη χειρωνακτική εργασία, καθώς τα πάντα εκτελούνται αυτόματα και έτσι ο άνθρωπος έχει κυρίως ρόλο ελεγκτή. Ιδιαίτερα συμβάλλει στη δημιουργία κατάλληλων συνθηκών για την εκκόλαψη των αυγών, με αποτέλεσμα τα ποσοστά θνησιμότητας να μειώνονται στο ελάχιστο σε σχέση με το φυσικό τρόπο επώασής τους. Η θερμοκρασία περιβάλλοντος του κοτετσιού ελέγχεται και σταθεροποιείται διότι βελτιώνει άμεσα την ωοτοκία.

### Η εργασία αυτή περιέχει τα εξής:

Δημιουργούμε μια μονάδα που τοποθετείται μέσα στην εκκολαπτική μηχανή και διαθέτει αισθητήρες υγρασίας-θερμοκρασίας, οι οποίοι θα ελέγχουν αν οι συνθήκες είναι κατάλληλες. Αυτή θα επικοινωνεί με τον κεντρικό σταθμό, ενημερώνοντας τον τακτικά, ώστε αν έχουμε διαφορετική ένδειξη από την επιθυμητή, να γίνεται αυξομείωση θερμοκρασίας και υγρασίας. Επίσης θα υπάρχει μια μονάδα που ελέγχει την ποσότητα τροφής των πουλερικών μέσω ενός αισθητήρα βάρους, ώστε να παρέχει τροφή όποτε είναι απαραίτητο. Τέλος, θα υπάρχουν αισθητήρες θερμοκρασίας σε διάφορα μέρη του κοτετσιού, προκειμένου να επικρατεί η ιδανική για τις κότες θερμοκρασία.

## Ανάλυση σχεδίασης του δικτύου

Για να υλοποιήσουμε το σύστημά μας θα θεωρήσουμε ότι το δίκτυο λαμβάνει χώρα σε κοτέτσι με έκταση ακτίνας 500 μέτρων. Επιλέξαμε τυπική απόκλιση 10dB και επιθυμούμε κάλυψη 95%. Αν υποθέσουμε ότι οι κόμβοι της μονάδας της μηχανής και οι κόμβοι στο χώρο του κοτετσιού στέλνουν κάθε 10 λεπτά, τότε θα στέλνουν 0,166 πακέτα/sec. Επίσης η κόμβοι της διανομής τροφής στέλνουν με ρυθμό 0,75 πακέτα/sec. Αν υποθέσουμε ότι η διάρκεια κάθε πακέτου είναι 20 msec και θέλουμε να εξυπηρετεί συνολικά 240 κόμβους (200 κόμβοι στις μηχανές εκκόλαψης, 15 κόμβοι στο χώρο του κοτετσιού και 25 στους χώρους σίτισης) με το πρωτόκολλο ALOHA ( $\max=0.186$ ) ο αριθμός των καναλιών που θα χρειαστούμε είναι 6.

$$\text{Αριθμός Καναλιών} = \frac{\left(215 * \frac{0.166}{s} + 25 * \frac{0.75}{s}\right) * 0.02}{0,186} = 5.85 = 6$$

## Υλοποίηση

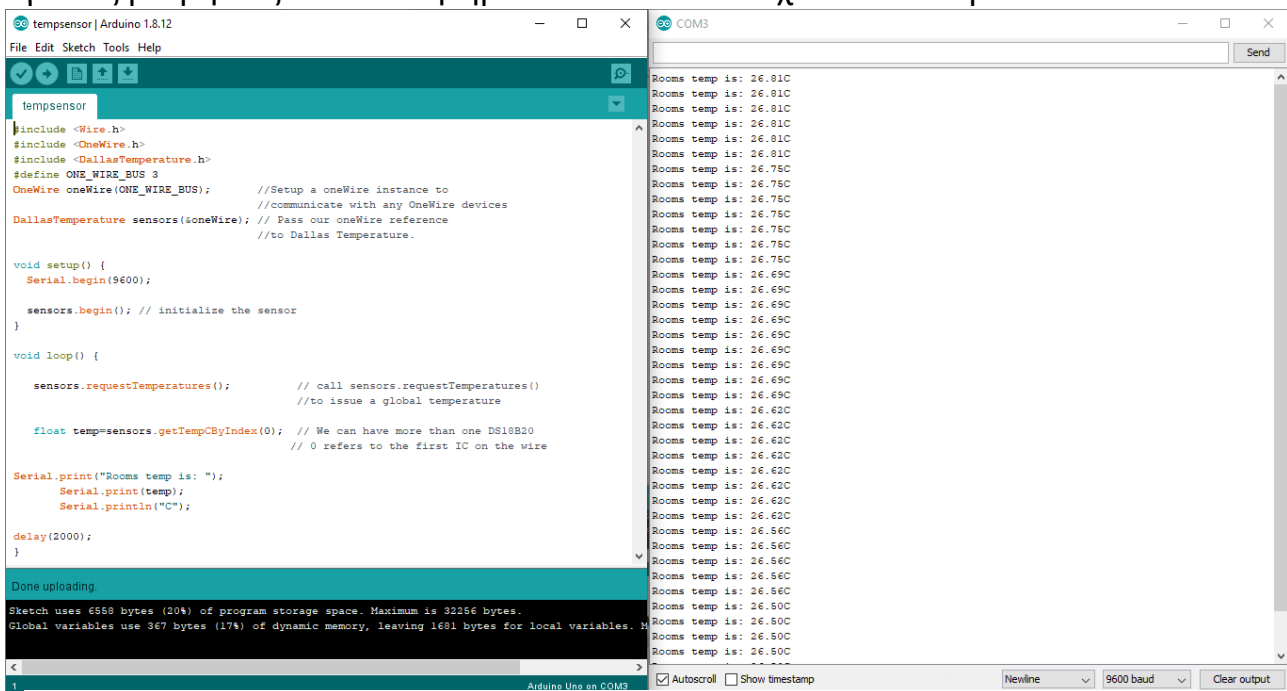
Για την υλοποίηση της ιδέας μας χρησιμοποιήθηκαν τέσσερα Arduino Boards σε συνδυασμό με τέσσερις RFM22 Transceivers. Επιπρόσθετα, χρειάστηκαν αισθητήρες θερμοκρασίας, υγρασίας – θερμοκρασίας και δύναμης.

Στον πρώτο κόμβο του Arduino τοποθετήθηκε ο αισθητήρας θερμοκρασίας, στο δεύτερο ο αισθητήρας υγρασίας-θερμοκρασίας και στον τρίτο ο αισθητήρας δύναμης. Αυτοί οι 3 κόμβοι είναι οι πομποί που στέλνουν τις μετρήσεις στο δέκτη που είναι ο κεντρικός σταθμός του συστήματος μας και είναι υπεύθυνος να ελέγξει αν οι τιμές που λαμβάνει είναι οι επιθυμητές και αν δεν είναι να δράσει ανάλογα.

Η επικοινωνία μεταξύ των κόμβων γίνεται εφικτή μέσω του πρωτοκόλλου επικοινωνίας δικτύων ALOHA. Παρακάτω αναλύεται ξεχωριστά η λειτουργία και η εφαρμογή του κάθε στοιχείου.

### 1. Αισθητήρας θερμοκρασίας για το χώρο του κοτετσιού

Προκειμένου να μετρηθεί η θερμοκρασία στους διάφορους χώρους κοτετσιού , χρησιμοποιήθηκαν αισθητήρες θερμοκρασίας.. Παρακάτω παρατίθεται ο κώδικας για τις μετρήσεις των αισθητήρων και τα αντίστοιχα αποτελέσματα.



```
tempensor | Arduino 1.8.12
File Edit Sketch Tools Help

tempensor

#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 3
OneWire oneWire(ONE_WIRE_BUS); //Setup a oneWire instance to
//communicate with any OneWire devices
DallasTemperature sensors(&oneWire); // Pass our oneWire reference
//to Dallas Temperature.

void setup() {
  Serial.begin(9600);

  sensors.begin(); // initialize the sensor
}

void loop() {
  sensors.requestTemperatures(); // call sensors.requestTemperatures()
  //to issue a global temperature

  float temp=sensors.getTempCByIndex(0); // We can have more than one DS18B20
  // 0 refers to the first IC on the wire

  Serial.print("Rooms temp is: ");
  Serial.print(temp);
  Serial.println("C");

  delay(2000);
}

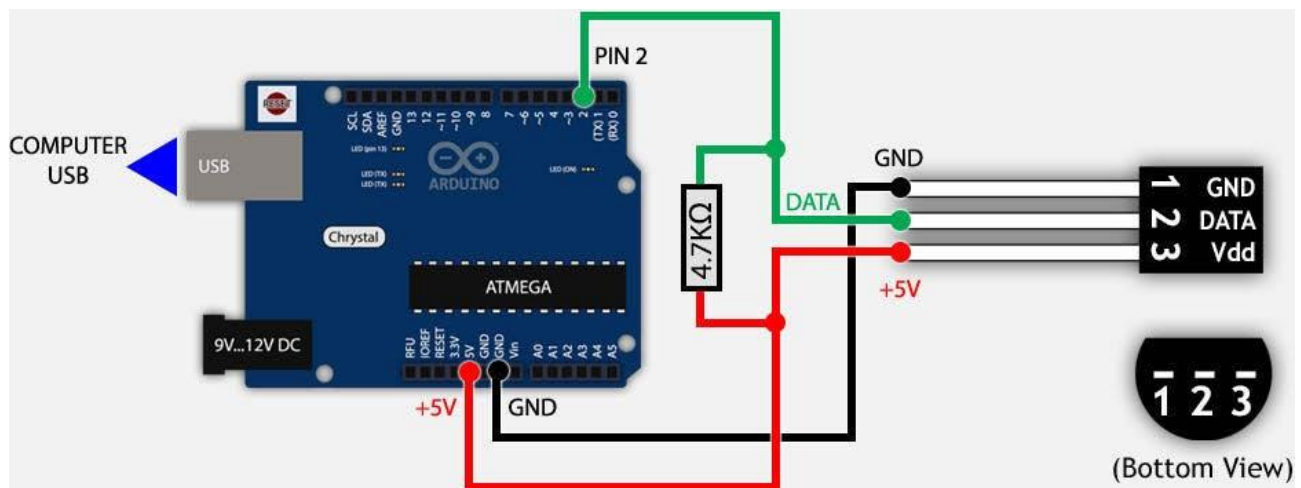
Done uploading
Sketch uses 6550 bytes (20%) of program storage space. Maximum is 32256 bytes.
Global variables use 367 bytes (17%) of dynamic memory, leaving 1681 bytes for local variables.

COM3
Rooms temp is: 26.81C
Rooms temp is: 26.81C
Rooms temp is: 26.81C
Rooms temp is: 26.81C
Rooms temp is: 26.81C
Rooms temp is: 26.81C
Rooms temp is: 26.75C
Rooms temp is: 26.75C
Rooms temp is: 26.75C
Rooms temp is: 26.75C
Rooms temp is: 26.75C
Rooms temp is: 26.75C
Rooms temp is: 26.75C
Rooms temp is: 26.69C
Rooms temp is: 26.69C
Rooms temp is: 26.69C
Rooms temp is: 26.69C
Rooms temp is: 26.69C
Rooms temp is: 26.69C
Rooms temp is: 26.62C
Rooms temp is: 26.62C
Rooms temp is: 26.62C
Rooms temp is: 26.62C
Rooms temp is: 26.62C
Rooms temp is: 26.62C
Rooms temp is: 26.62C
Rooms temp is: 26.56C
Rooms temp is: 26.56C
Rooms temp is: 26.56C
Rooms temp is: 26.56C
Rooms temp is: 26.56C
Rooms temp is: 26.56C
Rooms temp is: 26.50C
Rooms temp is: 26.50C
Rooms temp is: 26.50C
Rooms temp is: 26.50C
Rooms temp is: 26.50C
Autoscroll Show timestamp Newline 9600 baud Clear output
```

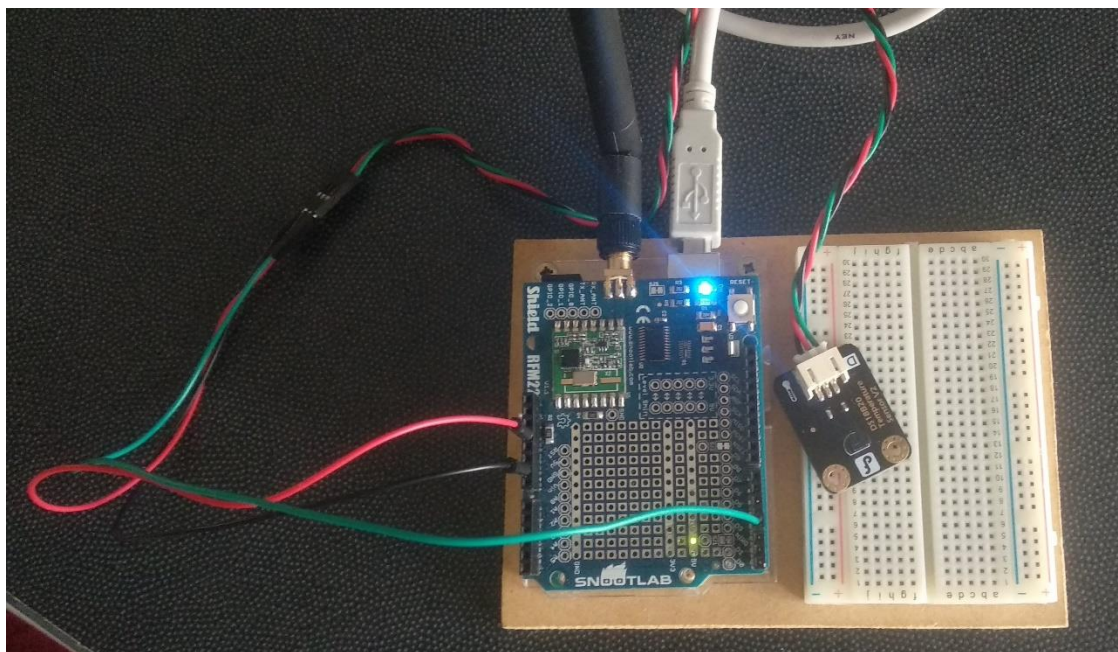
Εικόνα 1-Κώδικας και αποτελέσματα αισθητήρα θερμοκρασίας

Από τα παραπάνω προέκυψε το συμπέρασμα ότι οι τιμές δεν έχουν ακραίες μεταβολές σε σύντομο χρόνο. Έτσι, αποφασίστηκε να λαμβάνονται μετρήσεις από τους αισθητήρες κάθε 10 λεπτά, οι οποίες θα στέλνονται στο δέκτη.

Οι επιμέρους συνδεσμολογίες που ακολουθήθηκαν για τους αισθητήρες φαίνονται παρακάτω μαζί με τον τρόπο που υλοποιήθηκαν στην πράξη.



Εικόνα 2- Συνδεσμολογία αισθητήρα θερμοκρασίας



Εικόνα 3- Συνδεσμολογία αισθητήρα θερμοκρασίας στην πράξη

Το σύστημα παραπάνω επιστρέφει της τιμές που φαίνονται στην εικόνα 1 με σκοπό έπειτα να σταλούν στον κεντρικό σταθμό.

Η βασική λειτουργικότητα του DS18B20 είναι ο αισθητήρας θερμοκρασίας απευθείας σε ψηφιακό.

Για να ξεκινήσει μια μέτρηση θερμοκρασίας και μετατροπή A-σε-D, ο κύριος πρέπει να εκδώσει μια εντολή Μετατροπή T. Μετά τη μετατροπή, τα προκύπτοντα θερμικά δεδομένα αποθηκεύονται στον καταχωρητή θερμοκρασίας 2-byte στη μνήμη του scratchpad και το DS18B20 επιστρέφει στην κατάσταση αδράνειας. Εάν το DS18B20 τροφοδοτείται από εξωτερική τροφοδοσία, τότε ο κύριος μπορεί να παρέχει χρονοθυρίδες ανάγνωσης δίπλα στην εντολή Μετατροπή T. Ο αισθητήρας θα αντιδρά με παροχή 0 αν και η αλλαγή θερμοκρασίας βρίσκεται σε βελτίωση και αντιδρά με παροχή 1 αν και η αλλαγή θερμοκρασίας έχει γίνει.

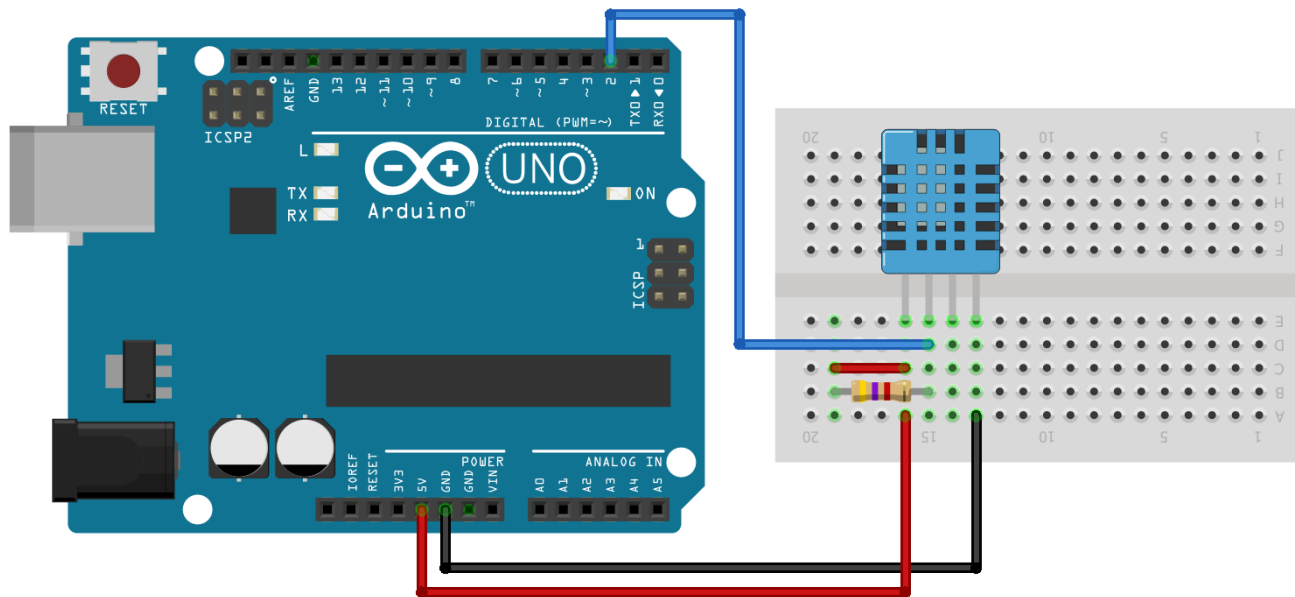
Προκειμένου να βεβαιωθούμε ότι οι συνθήκες που επικρατούν στις αποθήκες των αυγών είναι ιδανικές, χρησιμοποιούμε αισθητήρες υγρασίας-θερμοκρασίας. Η υγρασία και η θερμοκρασία είναι ιδιαίτερα σημαντικές για την επώαση του αυγού και όταν είναι κατάλληλες μειώνουν σημαντικά τα ποσοστά θνησιμότητας. Παρακάτω είναι ο κώδικας για τις μετρήσεις των αισθητήρων και τα αποτελέσματα που μας δίνουν.

Εικόνα 4 – Κώδικας και αποτελέσματα αισθητήρα υγρασίας θερμοκρασίας

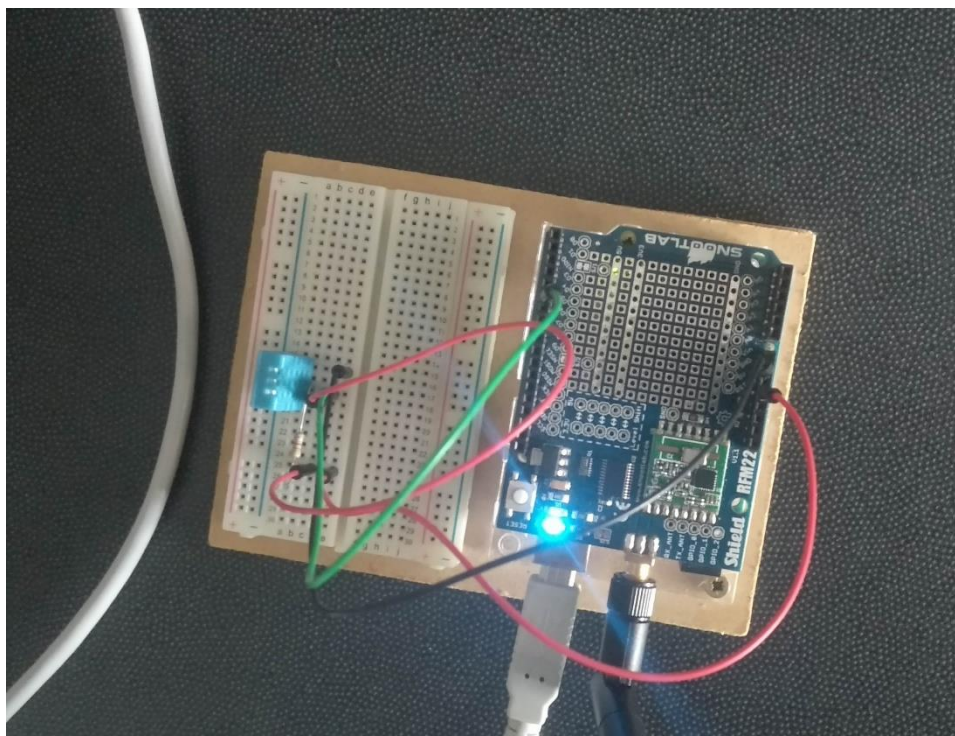


Παρατηρούμε ότι οι τιμές δεν έχουν ούτε ακραίες ούτε συχνές μεταβολές. Έτσι, αποφασίστηκε να λαμβάνονται μετρήσεις από τους αισθητήρες κάθε 10 λεπτά και να στέλνονται στο δέκτη.

Οι συνδεσμολογία που ακολουθήθηκε είναι:



Εικόνα 5- Συνδεσμολογία αισθητήρα υγρασίας-θερμοκρασίας

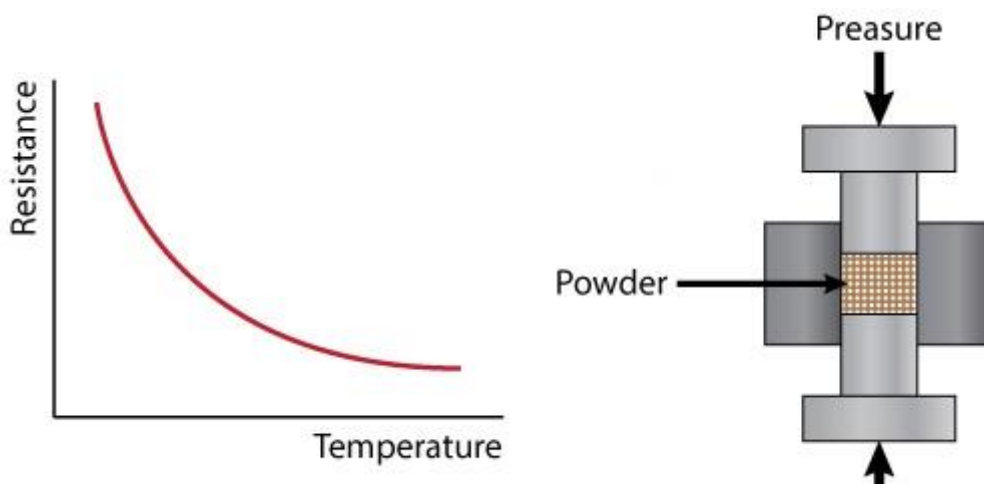


Εικόνα 6- Συνδεσμολογία αισθητήρα υγρασίας-θερμοκρασίας στην πράξη

Το σύστημα παραπάνω επιστρέφει της τιμές που φαίνονται στην εικόνα 4 με σκοπό έπειτα να σταλούν στον κεντρικό σταθμό. Ο αισθητήρας DHT11 χρησιμοποιήθηκε για

να μετράει τη θερμοκρασία και την υγρασία. Η θερμοκρασία μετριέται με τη βοήθεια ενός θερμίστορ NTC ( αρνητικού συντελεστή θερμοκρασίας θερμίστορ). Αυτά τα θερμίστορ συνήθως κατασκευάζονται με ημιαγωγούς, κεραμικά και πολυμερή. Η αντίσταση της συσκευής είναι αντιστρόφως ανάλογη της θερμοκρασίας και ακολουθεί μια υπερβολική καμπύλη. Η θερμοκρασία με τη χρήση NTC βρίσκεται με την εξίσωση Steinhart Hart.

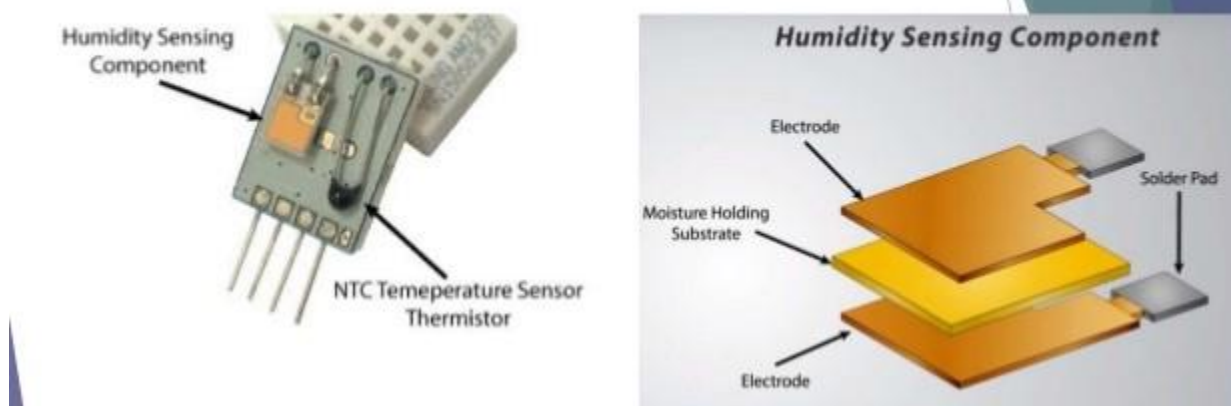
$$T = \frac{1}{A + B \ln(R) + C [\ln(R)]^3}$$



Εικόνα 7- Λειτουργία NTC

Η υγρασία ανιχνεύεται χρησιμοποιώντας μια αντίσταση που εξαρτάται από την υγρασία. Έχει δύο ηλεκτρόδια και μεταξύ αυτών υπάρχει ένα υπόστρωμα που συγκρατεί και διατηρεί την υγρασία. Η αγωγιμότητα και συνεπώς η αντίσταση αλλάζει με την μεταβαλλόμενη υγρασία.

## DHT 11 Working Principle



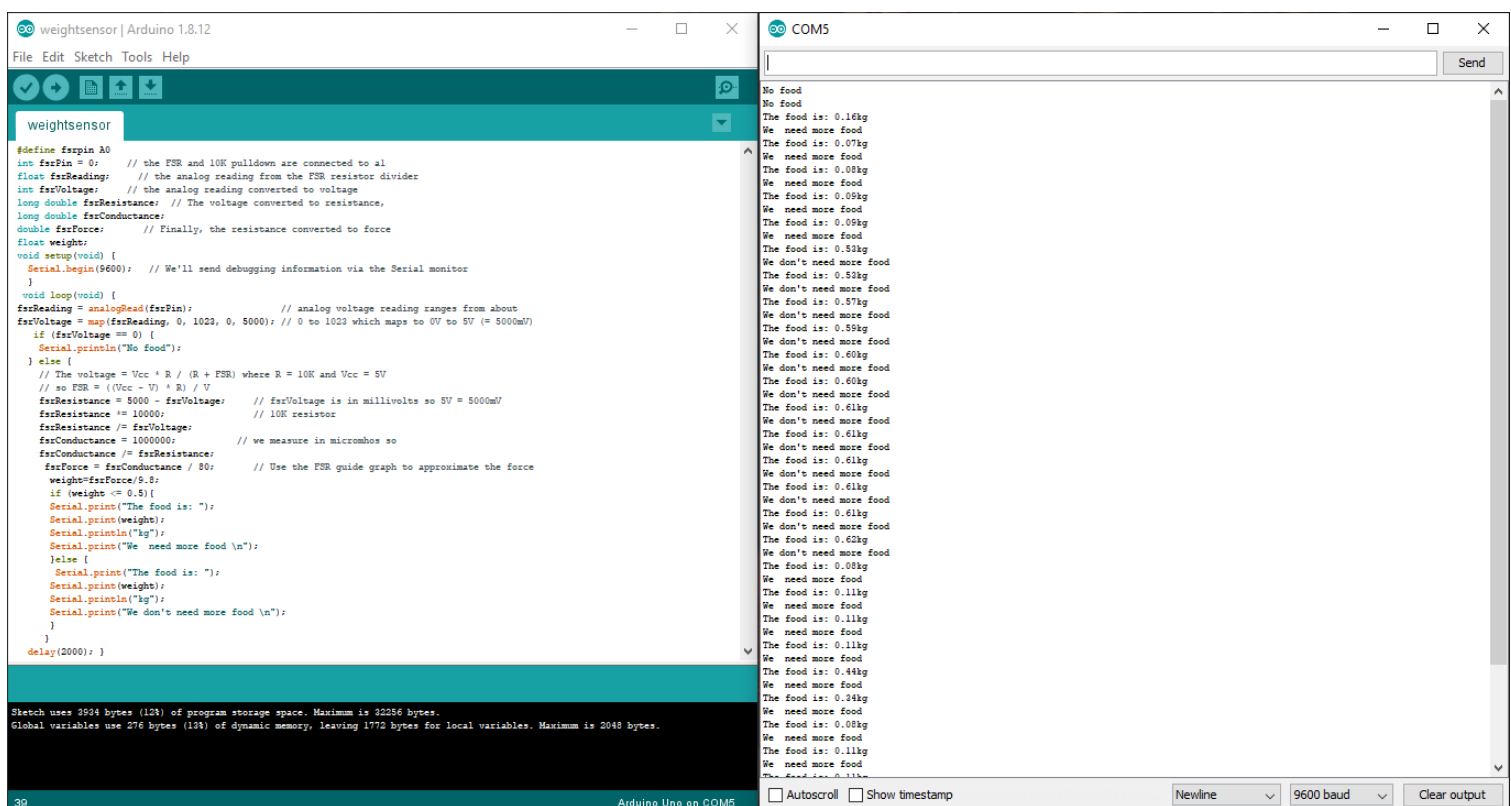
Εικόνα 8 – Λειτουργία ανίχνευσης υγρασίας



Οι μεταβολές επεξεργάζονται από ένα ολοκληρωμένο τοποθετημένο στην άλλη πλευρά της πλακέτας που υπολογίζει τις τιμές και μπορεί να τις μεταδώσει σε έναν μικροελεγκτή χρησιμοποιώντας μόνο μία γραμμή δεδομένων.

### 3. Αισθητήρας δύναμης στις ταΐστρες

Προκειμένου να ελέγχουμε το βάρος του φαγητού στις ταΐστρες χρησιμοποιήθηκαν αισθητήρες δύναμης. Παρακάτω παρατίθεται ο κώδικας για τις μετρήσεις των αισθητήρων και τα αντίστοιχα αποτελέσματα.



The screenshot displays the Arduino IDE interface. The left pane shows the 'weightsensor' sketch, which defines pins, reads analog data from an FSR sensor, and calculates force in grams. The right pane shows the serial monitor output, which prints messages like 'No food', 'The food is: 0.16kg', and 'We need more food' based on the sensor readings. The bottom status bar indicates the sketch is for an 'Arduino Uno en COM5'.

```
#define fsrcPin A0
int fsrcPin = 0;
float fsrcReading; // the analog reading from the FSR resistor divider
int fsrcVoltage; // the analog reading converted to voltage
long double fsrcResistance; // The voltage converted to resistance,
long double fsrcConductance;
double fsrcForce; // Finally, the resistance converted to force
float weight;

void setup(void) {
  Serial.begin(9600); // We'll send debugging information via the Serial monitor
}

void loop(void) {
  fsrcReading = analogRead(fsrcPin); // analog voltage reading ranges from about
  fsrcVoltage = map(fsrcReading, 0, 1023, 0, 5000); // 0 to 1023 which maps to 0V to 5V (= 5000mV)
  if (fsrcVoltage == 0) {
    Serial.println("No food");
  } else {
    // The voltage = Vcc * R / (R + FSR) where R = 10K and Vcc = 5V
    // so FSR = ((Vcc - V) * R) / V
    fsrcResistance = 5000 - fsrcVoltage; // fsrcVoltage is in millivolts so 5V = 5000mV
    fsrcResistance *= 10000; // 10K resistor
    fsrcResistance /= fsrcVoltage;
    fsrcConductance = 1000000; // we measure in microhmhos so
    fsrcConductance /= fsrcResistance;
    fsrcForce = fsrcConductance / 80; // Use the FSR guide graph to approximate the force
    weight = fsrcForce / 9.8;
    if (weight <= 0.5) {
      Serial.print("The food is: ");
      Serial.print(weight);
      Serial.println("kg");
      Serial.print("We need more food \n");
    } else {
      Serial.print("The food is: ");
      Serial.print(weight);
      Serial.println("kg");
      Serial.print("We don't need more food \n");
    }
  }
  delay(2000);
}
```

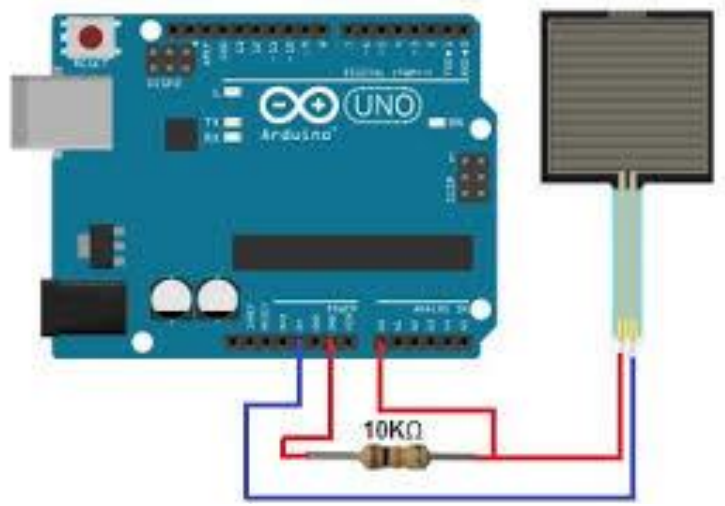
Serial Monitor Output:

```
No food
No food
The food is: 0.16kg
We need more food
The food is: 0.07kg
We need more food
The food is: 0.08kg
We need more food
The food is: 0.09kg
We need more food
The food is: 0.09kg
We need more food
The food is: 0.53kg
We don't need more food
The food is: 0.53kg
We don't need more food
The food is: 0.57kg
We don't need more food
The food is: 0.59kg
We don't need more food
The food is: 0.60kg
We don't need more food
The food is: 0.60kg
We don't need more food
The food is: 0.61kg
We don't need more food
The food is: 0.61kg
We don't need more food
The food is: 0.61kg
We don't need more food
The food is: 0.61kg
We don't need more food
The food is: 0.62kg
We don't need more food
The food is: 0.08kg
We need more food
The food is: 0.11kg
We need more food
The food is: 0.11kg
We need more food
The food is: 0.11kg
We need more food
The food is: 0.44kg
We need more food
The food is: 0.34kg
We need more food
The food is: 0.08kg
We need more food
The food is: 0.11kg
We need more food
The food is: 0.11kg
```

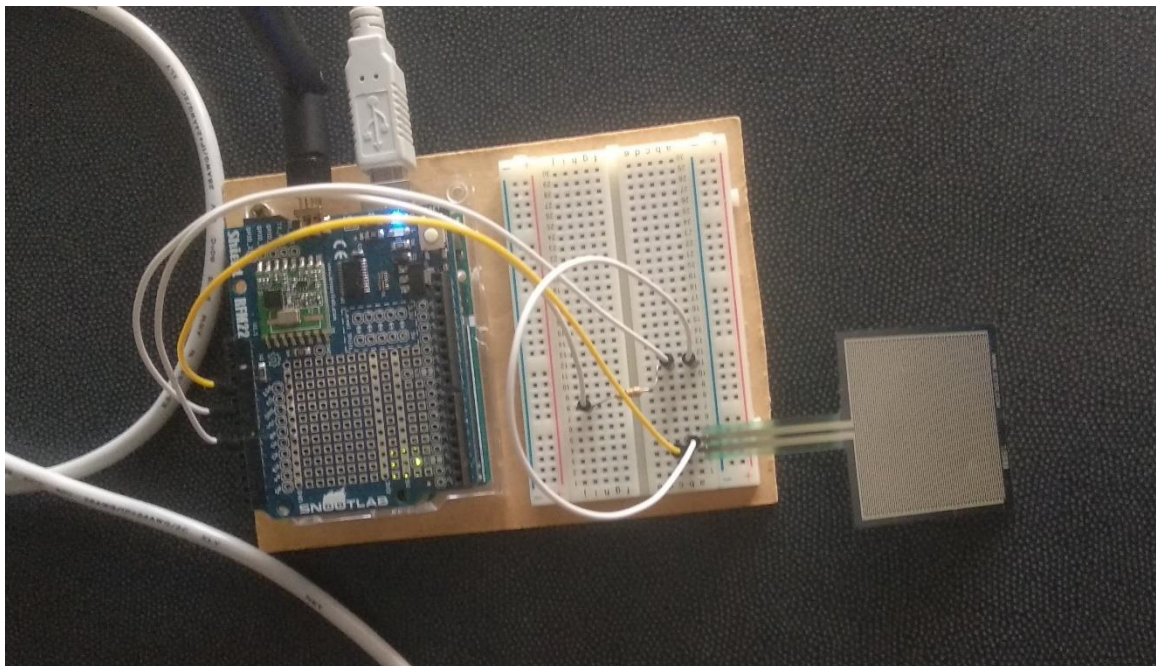
Εικόνα 9 - Κώδικας και αποτελέσματα αισθητήρα δύναμης

Επειδή η κατανάλωση φαγητού από τις κότες είναι απρόβλεπτη και δεν μπορούμε να κάνουμε υποθέσεις αποφασίστηκε να λαμβάνουμε 0.75 πακέτα/second, τα οποία θα στέλνονται στο κεντρικό σταθμό.

Παρακάτω αναπαρίσταται η συνδεσμολογία που προτείνεται για τον αισθητήρα και ο τρόπος που υλοποιήθηκε στην πραγματικότητα:



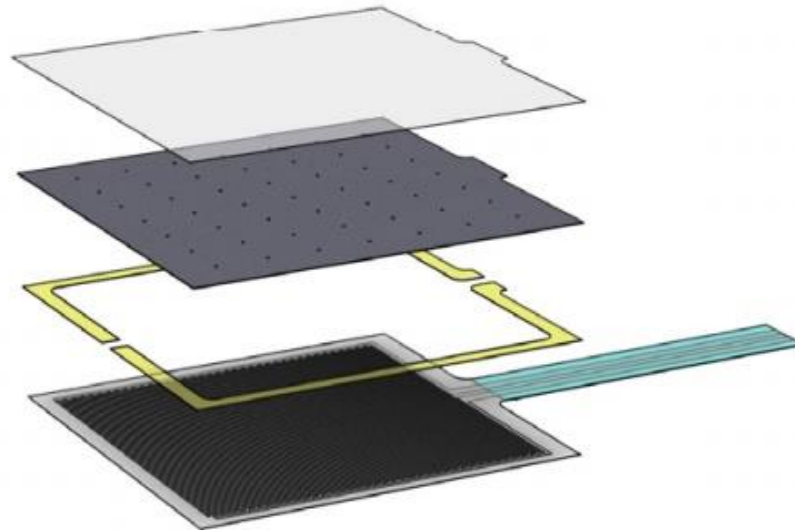
Εικόνα 10- Συνδεσμολογία αισθητήρα δύναμης



Εικόνα 11- Συνδεσμολογία αισθητήρα δύναμης στην πράξη

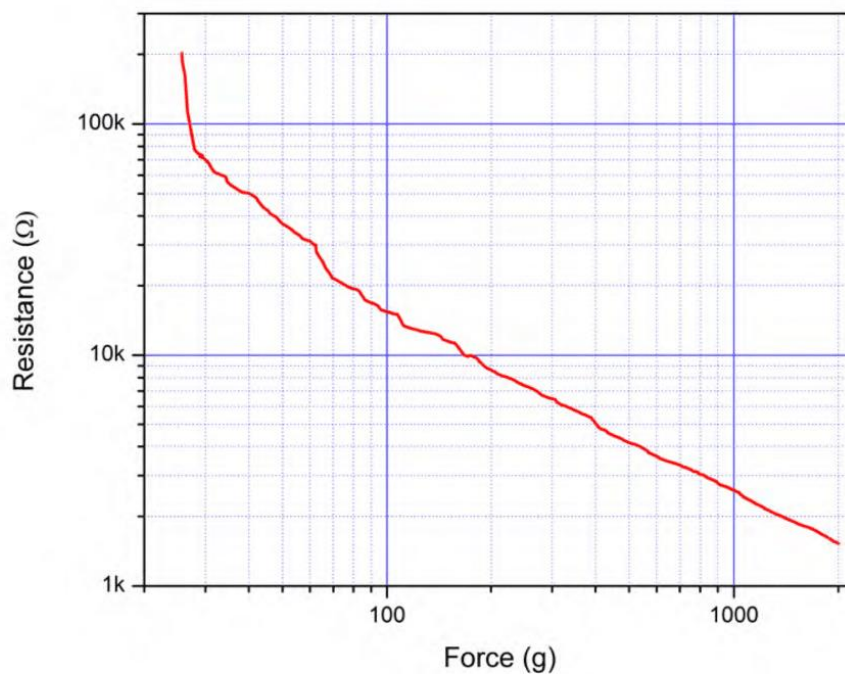
Η αντίσταση ενός FSR εξαρτάται από την πίεση που ασκείται στην περιοχή ανίχνευσης. Όσο μεγαλύτερη πίεση ασκείτε, τόσο χαμηλότερη είναι η αντίσταση. Το εύρος αντίστασης είναι στην πραγματικότητα αρκετά μεγάλο: > 10 MΩ (χωρίς πίεση) έως ~ 200 Ω (μέγιστη πίεση). Τα περισσότερα FSR μπορούν να ανιχνεύσουν δύναμη στην περιοχή από 100 g έως 10 kg.

Ένα FSR αποτελείται από δύο μεμβράνες και ένα συγκολλητικό απόστασης. Οι αγωγικές μεμβράνες διαχωρίζονται από ένα λεπτό διάκενο αέρα όταν δεν ασκείται πίεση. Μία από τις μεμβράνες περιέχει δύο ίχνη που εκτείνονται από την ουρά στην περιοχή ανίχνευσης (το στρογγυλό μέρος). Αυτά τα ίχνη είναι υφασμένα μαζί, αλλά δεν αγγίζουν. Η άλλη μεμβράνη επικαλύπτεται με αγωγίμο μελάνι. Όταν πιέζετε τον αισθητήρα, το μελάνι κλείνει τα δύο ίχνη μαζί με μια αντίσταση που εξαρτάται από την πίεση.



Εικόνα 12- Κατασκευή FSR

Το παρακάτω γράφημα εμφανίζει την καμπύλη αντίστασης έναντι δύναμης για τον αισθητήρα FSR-Square. Τα δεδομένα απεικονίζονται σε λογαριθμικές κλίμακες, υπάρχει μια τεράστια πτώση στην αντίσταση όταν ασκείται μικρή ποσότητα πίεσης. Μετά από αυτό, η αντίσταση είναι αντιστρόφως ανάλογη με την εφαρμοζόμενη δύναμη. Περίπου 10 kg (δεν φαίνεται στο γράφημα) ο αισθητήρας είναι κορεσμένος και μια αύξηση της ισχύος αποδίδει μικρή έως καθόλου μείωση στην αντίσταση.



Εικόνα 13- Γραφική παράσταση αντίστασης-δύναμης

Η διαδικασία μετατροπής της τιμής εξόδου( Vout) σε δύναμη και μετά σε βάρος είναι η εξής:

```
// The voltage = Vcc * R / (R + FSR) where R = 10K and Vcc = 5V
//so FSR = ((Vcc - V) * R) / V
fsrResistance = 5000 - fsrVoltage;      // fsrVoltage is in millivolts so 5V = 5000mV
fsrResistance *= 10000;                 // 10K resistor
fsrResistance /= fsrVoltage;
fsrConductance = 1000000;              // we measure in micromhos so
fsrConductance /= fsrResistance;
fsrForce = fsrConductance / 80;         // we use the FSR guide graph to approximate the force
weight=fsrForce/9.8;                   // and from the force we calculate the weight
```

Εικόνα 14- Κώδικας μετατροπής τάσης σε βάρος

#### 4. Επικοινωνία πομπών με τον κεντρικό σταθμό

Βασικός σκοπός αυτού του τμήματος είναι να στέλνει τα δεδομένα που λαμβάνουν οι αισθητήρες στον κεντρικό σταθμό, προκειμένου αυτός να ελέγξει άμα επικρατούν κατάλληλες συνθήκες και να λάβει τα απαραίτητα μέτρα. Οι μετρήσεις αυτές γίνονται συνεχώς και στέλνονται στο δέκτη κάθε δέκα λεπτά από τους αισθητήρες-πομπούς θερμοκρασίας και θερμοκρασίας-υγρασίας, ενώ κάθε 45 λεπτά στέλνει στον δέκτη ο αισθητήρας-πομπός της δύναμης.

Αρχικά, φορτώνονται οι απαραίτητες βιβλιοθήκες για την επικοινωνία (πχ. RF22, RF22 Router).

Ο κάθε αισθητήρας αποτελεί ένα κόμβο που είναι ουσιαστικά ένας πομπός και διαθέτει τη δική του ταυτότητα η οποία είναι ο αριθμός του συγκεκριμένου Arduino που τον υποστηρίζει. Ανάλογα της ταυτότητας αυτής καλείται και ο constructor που δημιουργεί τη διεύθυνση του. Οι συναρτήσεις setup() και loop() έχουν παρόμοια λειτουργία σε όλους τους πομπούς.

##### **void setup():**

Στο κομμάτι αυτό τοποθετείται κώδικας αρχικοποιήσεων, που τρέχει μόνο μια φορά. Στην προκειμένη περίπτωση, το τμήμα του setup χρησιμοποιείται για την εκκίνηση του Serial Monitor ώστε να εκτυπώνονται επιθυμητές τιμές και για την ρύθμιση της επικοινωνίας (πχ ισχύς εκπομπής, δημιουργία δρόμου διασύνδεσης ανάμεσα σε πομπούς και δέκτη). Η μόνη διαφορά είναι ότι στους αισθητήρες θερμοκρασίας και θερμοκρασίας-υγρασίας έχουμε και εντολή εκκίνησης των αισθητήρων. Πολύ σημαντικός είναι και ο συντονισμός στην προκαθορισμένη συχνότητα των πομπών, στην οποία συντονίστηκε και ο κεντρικός σταθμός (δέκτης) για να επιτευχθεί η επικοινωνία.

## void loop():

Το συγκεκριμένο τμήμα ενός κώδικα Arduino επαναλαμβάνεται διαρκώς. Από τη στιγμή που όλοι οι πομποί είναι διαρκώς σε λειτουργία, στο τμήμα αυτό υλοποιούνται οι διεργασίες που εκτελούνται συνεχώς και επαναληπτικά. Όλα τα τμήματα του κώδικά μας τα έχουμε υλοποιήσει μέσα στη loop.

Ο στόχος είναι να λάβουμε τη μέτρηση των αισθητήρων και να τις στείλουμε στον κεντρικό σταθμό. Επειδή οι περισσότερες μεταβλητές είναι τύπου float τις μετατρέπουμε σε int με πολλαπλασιασμό είτε με 100 ή με 1000 (θα είναι δουλειά του δέκτη να τις ξαναμετατρέψει σε float, ούτως ώστε να λάβουμε την πραγματική τιμή). Για να ελέγξουμε αν τα πακέτα στέλνονται στο δέκτη χρησιμοποιούμε μία if συνθήκη η οποία αν δε βγει αληθής δημιουργείται τυχαίος αριθμός αναμονής για να γίνει νέα προσπάθεια αποστολής του πακέτου. Παρακάτω είναι οι κώδικες για τον κάθε πομπό.

```
void setup() {  
    Serial.begin(9600);  
    if (!rf22.init())  
        Serial.println("RF22 init failed");  
  
    if (!rf22.setFrequency(431.0))  
        Serial.println("setFrequency Fail");  
    rf22.setTxPower(RF22_TXPOW_20DBM);  
    rf22.setModemConfig(RF22::GFSK_Rb125Fd125);  
    rf22.addRouteTo(DESTINATION_ADDRESS, DESTINATION_ADDRESS);  
    sensors.begin(); // initialize the sensor  
    randomSeed(2000); //start randNumber  
}
```

Εικόνα 15- Συνάρτηση setup του πομπού θερμοκρασίας

```
void setup() {  
    Serial.begin(9600);  
    if (!rf22.init())  
        Serial.println("RF22 init failed");  
  
    if (!rf22.setFrequency(431.0))  
        Serial.println("setFrequency Fail");  
    rf22.setTxPower(RF22_TXPOW_20DBM);  
    rf22.addRouteTo(DESTINATION_ADDRESS, DESTINATION_ADDRESS);  
    rf22.setModemConfig(RF22::GFSK_Rb125Fd125);  
    dht.begin(); // Setup sensor  
    randomSeed(2000); //start randNumber  
}
```

Εικόνα 16- Συνάρτηση setup του πομπού υγρασίας-θερμοκρασίας



```

void setup() {
  Serial.begin(9600);
  if (!rf22.init())
    Serial.println("RF22 init failed");

  if (!rf22.setFrequency(431.0))
    Serial.println("setFrequency Fail");
  rf22.setTxPower(RF22_TXPOW_20DBM);
  rf22.setModemConfig(RF22::GFSK_Rb125Fd125);
  rf22.addRouteTo(DESTINATION_ADDRESS, DESTINATION_ADDRESS);

  randomSeed(2000); //start randNumber
}

```

Εικόνα 17-Συνάρτηση setup του πομπού δύναμης

```

void loop() {

  sensors.requestTemperatures(); // call sensors.requestTemperatures() to issue a global temperature
  float temp=sensors.getTempCByIndex(0); // We can have more than one DS18B20
                                          // 0 refers to the first IC on the wire

  Serial.print("Rooms temp is: ");
  Serial.print(temp);
  Serial.println("C");
  // Get Measurement

  int temp2=(int) (temp*100);

  char data_read[RF22_ROUTER_MAX_MESSAGE_LEN];
  uint8_t data_send[RF22_ROUTER_MAX_MESSAGE_LEN];
  memset(data_read, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
  memset(data_send, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
  sprintf(data_read, "%d", temp2);

  data_read[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
  memcpy(data_send, data_read, RF22_ROUTER_MAX_MESSAGE_LEN);
}

```

---

Εικόνα 18-Συνάρτηση loop πομπού θερμοκρασίας

Στην παρακάτω εικόνα (εικόνα 19), επειδή θέλουμε να στείλουμε 2 τιμές στο δέκτη από τον κόμβο υγρασίας-θερμοκρασίας χρησιμοποιούμε ένα κοινό string στο οποίο τυπώνονται οι 2 float τιμές ως εξής:

```

sprintf(data_read, "%d %d", tempEgg2, humiEgg2);

```

Οι τιμές χωρίζονται από το χαρακτήρα του κενού (" ") που μετά τη λήψη του μηνύματος χρησιμοποιείται ως token, το οποίο θα εξηγηθεί παρακάτω. Κατά τ' άλλα, ακολουθείται ο βασικός κώδικας αποστολής.

---

```
void loop() {

float humiEgg = dht.readHumidity();           // Read the humidity in %
float tempEgg = dht.readTemperature();        // Read the temperature as Celsius
    if (isnan(humiEgg) && isnan(tempEgg)) {
        Serial.println("Failed to read from DHT sensor!");
    }

// Get Measurement
int humiEgg2=(int) (humiEgg);
int tempEgg2=(int) (tempEgg*100);

char data_read[RF22_ROUTER_MAX_MESSAGE_LEN];
uint8_t data_send[RF22_ROUTER_MAX_MESSAGE_LEN];
memset(data_read, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
memset(data_send, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);

sprintf(data_read, "%d %d", tempEgg2, humiEgg2);

data_read[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
memcpy(data_send, data_read, RF22_ROUTER_MAX_MESSAGE_LEN);
```

Εικόνα 19-Συνάρτηση loop πομπού υγρασίας-θερμοκρασίας

```
void loop()
{
    fsrReading = analogRead(fsrPin); // analog voltage reading ranges from about 0 to 1023 which maps to 0V to 5V (= 5000mV)
    fsrVoltage = map(fsrReading, 0, 1023, 0, 5000);
    if (fsrVoltage == 0) {
        Serial.println("No food");
    }
    else {
        // The voltage = Vcc * R / (R + FSR) where R = 10K and Vcc = 5V
        //so FSR = ((Vcc - V) * R) / V
        fsrResistance = 5000 - fsrVoltage; // fsrVoltage is in millivolts so 5V = 5000mV
        fsrResistance *= 10000;           // 10K resistor
        fsrResistance /= fsrVoltage;
        fsrConductance = 1000000;        // we measure in micromhos so
        fsrConductance /= fsrResistance;
        fsrForce = fsrConductance / 80;   // we use the FSR guide graph to approximate the force
        weight=fsrForce/9.8;              // and from the force we calculate the weight
    }
    delay(7000);

// Get Measurement

int weight2 =(int) (weight*1000);
char data_read[RF22_ROUTER_MAX_MESSAGE_LEN];
uint8_t data_send[RF22_ROUTER_MAX_MESSAGE_LEN];
memset(data_read, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
memset(data_send, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
sprintf(data_read, "%d", weight2);

data_read[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
memcpy(data_send, data_read, RF22_ROUTER_MAX_MESSAGE_LEN);
```

Εικόνα 20-Συνάρτηση loop πομπού δύναμης

Στα παραπάνω τα προς αποστολή δεδομένα τυπώνονται σε έναν πίνακα τύπου char με διάσταση RF22\_ROUTER\_MAX\_MESSAGE\_LEN που έχει προκαθοριστεί στη βιβλιοθήκη RF22 Router , ισούται με 50 και αντιστοιχεί στο μέγιστο μήκος ενός πακέτου που αποστέλλεται. Ο πίνακας αυτός χρησιμοποιείται ως ενδιάμεσος, για μετατροπή του εκάστοτε τύπου που θέλουμε να στείλουμε σε char. Ύστερα, το αποτέλεσμα αυτής της μετατροπής τυπώνεται με τη σειρά του σε έναν πίνακα τύπου unsigned integer που είναι κατάλληλος για αποστολή. Η αποστολή επιχειρείται με sendtoWait() σε συγκεκριμένη διεύθυνση και λαμβάνεται ως επιτυχής αν δεν προκύψει error. Στα πλαίσια του πρωτοκόλλου ALOHA, η διαδικασία επαναλαμβάνεται για συγκεκριμένο μέγιστο πλήθος προσπαθειών αποστολής. Το παρακάτω κομμάτι κώδικα ελέγχει την επιτυχία αποστολής των πακέτων στο δέκτη και είναι ίδιο και για τους 3 πομπούς.

```
successful_packet = false;
while (!successful_packet)
{
    if (rf22.sendtoWait(data_send, sizeof(data_send), DESTINATION_ADDRESS) != RF22_ROUTER_ERROR_NONE)
    {
        Serial.println("sendtoWait failed");
        randomNumber=random(200,max_delay);
        //Serial.println(randomNumber);
        delay(randomNumber);
    }
    else
    {
        successful_packet = true;
        Serial.println("sendtoWait Succesful");
    }
}

delay(20000);
```

Εικόνα 21-Συνάρτηση loop πομπών για έλεγχο αποστολής πακέτων

## 5. Λήψη δεδομένων από το δέκτη (κεντρικό σταθμό)

Ο βασικός σκοπός αυτού του τμήματος είναι να δέχεται συνεχώς πληροφορίες από τους αισθητήρες και να ελέγχει αν οι τιμές που έχει λάβει είναι οι επιθυμητές. Αν δεν είναι, τότε αυτός πρέπει να πάρει τα απαραίτητα μέτρα (π.χ πτώση θερμοκρασίας στις αυγοαποθήκες- άνοιγμα λάμπας για αύξηση θερμοκρασίας, έλλειψη φαγητού- προσθήκη φαγητού στις ταΐστρες κτλ). Η αναπαράσταση των μέτρων αποκατάστασης των συνθηκών έχει γίνει με LED, που το καθένα αντιστοιχεί σε μια λειτουργία. Συγκεκριμένα, όταν μια τιμή έχει ξεπεράσει τα επιθυμητά όρια τότε ανάβει ένα LED, που σημαίνει ότι έχει ξεκινήσει η διαδικασία αποκατάστασης των συνθηκών.

Φορτώνονται οι απαραίτητες βιβλιοθήκες για την επικοινωνία (πχ. RF22, RF22 Router).

### **void setup():**

Όπως συνηθίζεται σε κώδικες σε Arduino, στο κομμάτι αυτό τοποθετείται κώδικας που τρέχει μόνο μια φορά. Στην προκειμένη περίπτωση, το τμήμα του setup χρησιμοποιείται για την εκκίνηση του Serial Monitor ώστε να εκτυπώνονται επιθυμητές τιμές και για τη ρύθμιση της επικοινωνίας. Ύψιστης σημασίας είναι ο συντονισμός σε συγκεκριμένη συχνότητα μεταξύ του δέκτη και όλων των πομπών ώστε να επιτυγχάνεται η επικοινωνία μεταξύ τους. Λαμβάνεται μια αρχική τιμή του χρόνου συστήματος ώστε να μετρηθούν οι μέρες που βρίσκονται τα αυγά στις αποθήκες. Τέλος, καθορίζονται τα led που αντιστοιχούν σε κάθε κόμβο-αισθητήρα.

```
void setup() {
  Serial.begin(9600);
  pinMode(led5, OUTPUT); // LED connected to weight sensor
  pinMode(led6, OUTPUT); // LED connected to Humidity-Temperature sensor(Humidity part)
  pinMode(led7, OUTPUT); //LED connected to Humidity-Temperature sensor(Temperature part)
  pinMode(led8, OUTPUT); //LED connected to Temperature sensor

  if (!rf22.init())
    Serial.println("RF22 init failed");

  if (!rf22.setFrequency(431.0))
    Serial.println("setFrequency Fail");
  rf22.setTxPower(RF22_TXPOW_20DBM);

  rf22.setModemConfig(RF22::GFSK_Rb125Fd125);

  // Manually define the routes for this network
  rf22.addRouteTo(SOURCE_ADDRESS1, SOURCE_ADDRESS1);
  rf22.addRouteTo(SOURCE_ADDRESS2, SOURCE_ADDRESS2);
  rf22.addRouteTo(SOURCE_ADDRESS3, SOURCE_ADDRESS3);
}
```

Εικόνα 22-Συνάρτηση Setup του δέκτη

### **void loop():**

Το συγκεκριμένο τμήμα ενός κώδικα Arduino επαναλαμβάνεται διαρκώς. Σε αυτό το σημείο έχουν γίνει οι αρχικοποιήσεις των μεταβλητών των μετρήσεων που δέχεται ο κεντρικός σταθμός και τοποθετήθηκαν εδώ έτσι ώστε σε κάθε νέα μέτρηση να γίνεται μηδενισμός της προηγούμενης τιμής. Ακολουθεί ένα τμήμα κώδικα που αφορά τη λήψη των μετρήσεων, όπου εκτελείται ο βασικός κώδικας λήψης που θα αναλυθεί αργότερα.

```

void loop() {

    int roomtemp1=0; //Value we get from Source2 (room temperature multiplied by 100)
    float roomtemp2=0; //Variable that we will use to divide roomtemp1 with 100 and get the actual temperature
    int weight1 = 0; //Value we get from Source1 (food weight multiplied by 1000)
    float weight2 = 0; //Variable that we will use to divide weight1 with 1000 and get the actual weight of food
    float egghumidity1 = 0; //Value we get from Source3 (egg humidity)
    float eggtemp1 = 0; //Value we get from Source3 (egg temperature multiplied by 100)
    float eggtemp2 = 0; //Variable that we will use to divide eggtemp1 with 100 and get the actual temperature of eggs

    uint8_t buf[RF22_ROUTER_MAX_MESSAGE_LEN];
    char incoming[RF22_ROUTER_MAX_MESSAGE_LEN];
    memset(buf, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
    memset(incoming, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
    uint8_t len = sizeof(buf);
    uint8_t from;

```

Εικόνα 23-Συνάρτηση loop του δέκτη 1<sup>ο</sup> μέρος

Στη συνέχεια η λήψη και ο έλεγχος επιτυχίας της πραγματοποιούνται μέσω της `recvfromAck()`. Τα δεδομένα που καταφθάνουν αποθηκεύονται σε έναν πίνακα τύπου `integer`, μήκους 50. Μετά, στο τελευταίο στοιχείο τοποθετείται η τιμή `NULL` προκειμένου ο πίνακας να μετατρέψει επιτυχώς σε έναν ενδιάμεσο πίνακα `char`. Εν τέλει, ο πίνακας αυτός μετατρέπεται στον επιθυμητό τύπο. Τονίζεται ότι μέσω μιας μεταβλητής `from` ο κώδικας είναι σε θέση να γνωρίζει τον αποστολέα του πακέτου πληροφορίας. Επίσης χρησιμοποιείται η συνάρτηση `atoi()` για τη μετατροπή του `char` σε `integer` και στη συνέχεια σε `float` διαιρώντας με το 100 ή 1000 για να έχουμε ακρίβεια δεκαδικού. Ανάλογα τον πομπό από όπου ήρθε το μήνυμα ο κώδικας του δέκτη διαθέτει 3 `if` όπου σε κάθε μια έχει τα όρια των επιθυμητών τιμών και εφόσον τα περάσει τότε ανοίγει ένα `led` ανάλογα με το ποια μέτρηση έχει ακραία τιμή και ανάβει για 7 δευτερόλεπτα. Όταν ανάβει ένα `led` δείχνει την διεργασία αποκατάστασης της ακραίας τιμής που έλαβε ο δέκτης. Η πρώτη `if` (`if==22`) αφορά μετρήσεις που λαμβάνει ο δέκτης για την θερμοκρασία του χώρου του κοτετσιού και αυτή ιδανικά θα πρέπει να κυμαίνεται ανάμεσα στους 15-25 βαθμούς Celsius όπου σε αυτές τις θερμοκρασίες έχουμε αύξηση της ωοτοκίας. Αν ανάψει το `led` το σημαίνει ότι ανάβει ένα κλιματιστικό για να ψύξει ή να θερμάνει το χώρο για να έχουμε ξανά τις ιδανικές συνθήκες.

```

if (rf22.recvfromAck(buf, &len, &from))
{
    Serial.print("Got request from : ");
    Serial.println(from, DEC);

    if (from==22) //with this if statement, depending on the source that sent us values, receiver will take respective actions,22 is Temperature sensor
    {
        buf[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
        memcpy(incoming, buf, RF22_ROUTER_MAX_MESSAGE_LEN);
        int roomtemp1=atoi((char*)incoming);
        float roomtemp2=roomtemp1/100.0; //Here we convert our incoming to the actual room temperature

        Serial.print("Rooms temp is: ");
        Serial.print(roomtemp2);
        Serial.println("C");

        if (roomtemp2<15|| roomtemp2>25) //critical room temperature
        {
            Serial.println("Room temperature is critical");
            digitalWrite(8,HIGH); //turn on warning LED
            delay(7000);
            digitalWrite(8,LOW); //turn off warning LED
            delay(500);
        }else{
            Serial.println("Room temperature is fine");
        }
    }
}

```

Εικόνα 24-Συνάρτηση loop 2<sup>ο</sup> μέρος, 1<sup>η</sup> `if` για θερμοκρασία χώρου



Στη συνέχεια έχουμε την if (if==24) για τις μετρήσεις που λαμβάνουμε από τον αισθητήρα της δύναμης. Όταν το βάρος του φαγητού στις ταϊστρες κατέβει κάτω από 0,5 kg τότε σημαίνει ότι θα ανάψει το αντίστοιχο led και θα πρέπει να γεμίσει με φαγητό η αντίστοιχη ταϊστρα.

```
else if (from==24) //24 is weight sensor
{

    buf[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
    memcpy(incoming, buf, RF22_ROUTER_MAX_MESSAGE_LEN);
    int weight1=atoi((char*)incoming);
    float weight2=weight1/1000.0; //Here we convert our incoming to the actual food weight
    Serial.print("Region's weight is: ");
    Serial.print(weight2);
    Serial.println("kg");
    if (weight2<0.5) //Too little food
    {
        Serial.println("Need food");
        digitalWrite(5,HIGH);
        delay(7000);
        digitalWrite(5,LOW);
        delay(500);

    }else{

        Serial.println("Food is fine");
    }
}
```

Εικόνα 25-Συνάρτηση loop 3<sup>ο</sup> μέρος , 2<sup>η</sup> if για βάρος φαγητού

Τέλος έχουμε τις μετρήσεις που λαμβάνει ο κεντρικός σταθμός για την υγρασία-θερμοκρασία που υπάρχει στις αποθήκες που είναι τα αυγά. Εδώ ο πομπός στέλνει και τις δύο τιμές μαζί, τυπώνονται σε ένα string με κενά ανάμεσα στις τιμές. Το parsing του string υλοποιείται με την εντολή που συνηθίζεται σε τέτοιες περιπτώσεις: strtok. Ορίζεται ως token το κενό και κάθε φορά που εντοπίζεται, λαμβάνεται ένα ξεχωριστό τμήμα από το συνολικό string. Τα τμήματα αυτά έπειτα μετατρέπονται σε float αριθμούς μέσω των εντολών atof. Σημειώνεται ότι μετά την επιτυχή λήψη του συνολικού string, έχουμε ακόμη μια if μια για θερμοκρασία όπου τα επιθυμητά όρια είναι 37-38 βαθμούς Celsius και μία για την υγρασία όπου τα επιθυμητά όρια είναι 50%-60%, εφόσον τα ξεπεράσει ανάβει το αντίστοιχο led.

```

else if (from==26)      //26 is Humidity-Temperature sensor
{

    buf[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
    memcpy(incoming, buf, RF22_ROUTER_MAX_MESSAGE_LEN);
    const char d[2]=" "; //We create a string with two values
    char *token;
    token=strtok(incoming,d); //We separate the string values
    float eggtempl=atof(token);
    token=strtok(NULL,d);
    float egghumidityl=atof(token);
    float eggtemp2=eggtempl/100.0; //Here we convert our incoming to the actual egg temperature

    Serial.print("Region's temperature is: ");
    Serial.print(eggtemp2);
    Serial.println(" C");

    Serial.print("Region's humidity is: ");
    Serial.print(egghumidityl);
    Serial.println("%");

    if (eggtemp2<37 || eggtemp2>38) //Critical temperature for the eggs
    {
        Serial.println("Egg temperature critical");

        digitalWrite(7,HIGH);
        delay(7000);
        digitalWrite(7,LOW);
        delay(500);

    if (eggtemp2<37 || eggtemp2>38) //Critical temperature for the eggs
    {
        Serial.println("Egg temperature critical");

        digitalWrite(7,HIGH);
        delay(7000);
        digitalWrite(7,LOW);
        delay(500);

    }else{
        Serial.println("Egg temperature is fine");
    }
    if (egghumidityl<50 || egghumidityl>60) // critical humidity for the eggs
    {
        Serial.println("Humidity critical");
        digitalWrite(6,HIGH);
        delay(7000);
        digitalWrite(6,LOW);
        delay(500);
    }else{
        Serial.println("Humidity is fine");
    }
}

}

Serial.println(" "); //one line prints for easier serial reading

    delay(1000);

```

Εικόνα 26-Συνάρτηση loop 4<sup>ο</sup> μέρος , 3<sup>η</sup> if για υγρασία-θερμοκρασία

Στις παραπάνω περιγραφές πολλές φορές αναφέρθηκαν οι βασικοί κώδικες επικοινωνίας: αποστολής και λήψης. Οι κώδικες αυτοί δεν είναι άλλοι από τους κώδικες πομπού για την αποστολή (Tx με ALOHA) και δέκτη για τη λήψη (Rx με ALOHA) .

## Πρακτικότητα

Οι λόγοι για τους οποίους η πρόταση μας χαρακτηρίζεται πρακτική είναι οι εξής :

Κάθε κοτέτσι θα έχει συνήθως έναν δέκτη οπότε δε θα υπάρχουν θέματα στην επικοινωνία. Ακόμα όμως κ για ένα πολύ μεγάλο κοτέτσι όπου είναι πιθανό να χρειαστούν περισσότεροι του ενός δέκτες, αυτοί θα έχουν διαφορετική συχνότητα μεταξύ τους οπότε δεν θα υπάρχει σύγχυση επικοινωνίας.

Η εφαρμογή που αναπτύσσεται είναι απλή στη χρήση και συμβατή με κινητά τηλέφωνα, γεγονός που κάνει πολύ εύκολο τον έλεγχο του κοτετσιού από απόσταση, αν ο χρήστης-κτηνοτρόφος επιθυμεί να δει σε τι κατάσταση βρίσκεται το κοτέτσι οποιαδήποτε στιγμή, ενώ αν έχουμε κάποια ακραία τιμή ενημερώνεται με σχετικό μήνυμα τόσο όταν συμβεί αυτό, όσο και όταν αποκατασταθεί.

Με την εφαρμογή αυτή οι χρήστες έχουν πρόσβαση στα δεδομένα των συνθηκών του κοτετσιού, γεγονός που μπορεί να τους οδηγήσει σε σημαντικά συμπεράσματα σχετικά με το κοτέτσι τους (π.χ αν χρειάζεται συχνά το σύστημα να αυξάνει τη θερμοκρασία στα αυγά προκειμένου να σταθεροποιηθεί, πιθανώς χρειάζεται προσθήκη κάποιας καλύτερης μόνωσης).

## Εναλλακτικές Σχεδίασης

Στον τομέα των ασύρματων δικτύων αισθητήρων (Wireless Sensor Networks-WSN) είναι γνωστό ότι υπάρχουν πολύ πιο αποδοτικά πρωτόκολλα καθώς και η επιλογής ανάμεσα σε Αποφυγή των Collision (Collision Avoidance- CA) και Ανίχνευση Collision (Collision Detection-CD). Στην προκειμένη περίπτωση, επιλέξαμε ALOHA με Collision Avoidance. Πρόκειται για ένα αρκετά απλό πρωτόκολλο που όμως δεν είναι εξίσου αποδοτικό με τα **CSMA** πρωτόκολλα.

Συνεπώς, θα είχε νόημα να μιλήσουμε για εναλλακτική σχεδίαση με τη χρήση διαφορετικού πρωτόκολλου. Στη συνέχεια αναλύεται ως εναλλακτική το **non-persistent CSMA** πρωτόκολλο και το πώς θα μπορούσε να χρησιμοποιηθεί στην περίπτωσή μας μιας που συμφέρει τόσο σε απλότητα όσο και σε καλή απόδοση.

Η βασική ροή σε ένα Non-Persistent CSMA πρωτόκολλο είναι η εξής: Ο αποστολέας ελέγχει το κανάλι και αν αυτό είναι διαθέσιμο (IDLE), προχωράει στην αποστολή του επιθυμητού πακέτου πληροφορίας. Αν όχι, περιμένει (πχ για ένα τυχαίο χρονικό διάστημα) και επαναλαμβάνει τον έλεγχο. Η διαδικασία συνεχίζεται μέχρι την επίτευξη της αποστολής. Πρακτικά, μια απλή διάταξη που θα μπορούσε να χρησιμεύσει για υλοποίηση collision detection/avoidance είναι η εξής:

- Ένας αισθητήρας υπέρυθρων
- Ένας κινητήρας servo

Στην πράξη, ο κινητήρας θα περιστρέφει τον αισθητήρα υπέρυθρων που θα λαμβάνει συνεχώς δεδομένα και θα τα απεικονίζει μέσω του προγράμματος Proccessing. Ο κώδικας του Arduino θα πρέπει να εκτελεί 2 απλές λειτουργίες, εμφωλευμένες η μια στην άλλη: περιστροφή του κινητήρα και λήψη δεδομένων από τον αισθητήρα.

Επίσης, θα μπορούσαμε να αντικαταστήσουμε τον αισθητήρα δύναμης με ένα hx711 module και ένα Load Cell, μιας και ο force sensor μετράει σε Ohm και για τη μετατροπή σε δύναμη απαιτούνται αρκετές πράξεις οι οποίες είναι μάλιστα διαφέρουν αναλόγως την τιμή (όπως είδαμε κ παραπάνω η γραφική παράσταση Resistance( $\Omega$ ) - Force(g) έχει αρκετά περίεργη μορφή).

Τέλος, μία ακόμα εναλλαγή θα μπορούσε να ήταν η χρήση επιπλέον συναρτήσεων στο δέκτη, αντί της υλοποίησης ολόκληρου του κώδικα μέσα στη loop(). Ο λόγος για τον οποίο αυτό θα ήταν ωφέλιμο είναι ότι οι επιπλέον συναρτήσεις θα συνέβαλλαν στη συντήρηση του κώδικα, στην καλύτερη οργάνωση και την ευκολότερη κατανόησή του.