Τεχνική αναφορά

Εφαρμογές τηλεπικοινωνιακών διατάξεων

Σωτήρης Μόσχος 9030

Γεράσιμος Παπακώστας 8890

Ακαδημαϊκό έτος : 2019-2020

Περιεχόμενα

Περιγραφή του προβλήματος

Στη συγκεκριμένη εργασία στόχος μας είναι να αναπτύξουμε ένα δίκτυο ,το οποίο θα ενημερώνει ανά τακτά χρονικά διαστήματα τα δασαρχεία και τα πυροσβεστικά σώματα , αν και εφόσον αυτά είναι διαθέσιμα , για τις περιβαλλοντικές συνθήκες και τον κίνδυνο εμφάνισης πυρκαγιάς κοντινών δασικών εκτάσεων.

Ο άνθρωπος εξαρτάται από τα δάση. Είναι σημαντικά για την ζωή του και την αναψυχή του. Επιτελούν όμως και σημαντικές λειτουργίες. Ενισχύουν τη βιοποικιλότητα και το τοπίο, ρυθμίζουν το κλίμα , προστατεύουν το έδαφος. Τα δάση, πιθανώς να είναι ο πιο σημαντικός φυσικός πόρος στην Ελλάδα. Ακόμη, φιλοξενούν μεγάλο αριθμό θηλαστικών, πουλιών, ερπετών και αμφιβίων.

Για το λόγο αυτό , θεωρούμε ότι το δίκτυο αυτό, που βοηθά στην πρόληψη των πυρκαγιών στα δάση καθίσταται απαραίτητο.

Αρχικά , δημιουργούμε έναν “φάρο” , ένα μηχάνημα που είναι τοποθετημένο σε ένα συγκεκριμένο σημείο της κάθε δασικής περιοχής και διαθέτει αισθητήρες θερμοκρασίας, υγρασίας, υγρασίας εδάφους και ανίχνευσης πυρκαγιάς. Ο “φάρος” , δέχεται μια ένδειξη διαθεσιμότητας από το εκάστοτε δασαρχείο – πυροσβεστικό σώμα και αποστέλλει πληροφορίες , ανά 15 λεπτά. Σε περίπτωση που κάποιο δασαρχείο – πυροσβεστικό σώμα σταματάει να είναι διαθέσιμο , αποστέλλει μια ένδειξη μη διαθεσιμότητας και έτσι ο “φάρος” σταματάει να του παρέχει πληροφορίες.

Τοποθετώντας τον κατάλληλο αριθμό “φάρων” σε μια δασική έκταση καταφέρνουμε να λαμβάνουμε περιβαλλοντικές ενδείξεις για κάθε πιθανή περιοχή , με αποτέλεσμα να εστιάσουμε την προσοχή μας σε αυτές που εμφανίζουν τη μεγαλύτερη πιθανότητας εμφάνισης πυρκαγιά επιτυγχάνοντας έτσι τον στόχο μας , της πρόληψης της , ενώ σε περίπτωση πρόκλησης πυρκαγιάς , επιτυγχάνουμε την γρήγορη αντιμετώπισή της.

Ανάλυση σχεδίασης δικτύου

Για την ανάπτυξη της ιδέας μας υποθέτουμε ότι το δίκτυο μας λαμβάνει χώρα σε μια δασική έκταση , για παράδειγμα το Σέιχ Σου.

Επιλέξαμε τυπική απόκλιση σ = 10 dB με σκοπό να έχουμε επαρκή κάλυψη 95%.

Υποθέτοντας ότι οι κόμβοι του φάρου στέλνουν πληροφορίες ανά 15 λεπτά , θα αποστέλλουν 0.25 πακέτα/second.

Επίσης ο κόμβος του δασαρχείου – σώματος αποστέλλει ένδειξη διαθεσιμότητας με ρυθμό 1 πακέτα/second.

Η διάρκεια του κάθε πακέτου είναι 20 ms , και θέλουμε να εξυπηρετεί 100 κόμβους , με βάση το πρωτόκολλο ALOHA (max = 0.186).

Άρα έχουμε:

$$\text{Κανάλια} = \frac{\left(100*\frac{1}{s}+1*\frac{0.25}{s}\right)*0.02s}{0.186} = 10.77 = 11$$

Περιοχή κάλυψης δικτύου

Υλοποίηση

Για την υλοποίηση χρησιμοποιήθηκαν 2 Arduino Boards σε συνδυασμό με 2 RFM22 transceivers, αισθητήρες θερμοκρασίας, υγρασίας, υγρασίας εδάφους και φωτιάς.

Στον πρώτο κόμβο του Arduino που είναι ο “φάρος” (ένα μηχάνημα που είναι τοποθετημένο σε ένα συγκεκριμένο σημείο της κάθε δασικής περιοχής), βρίσκονται όλοι οι αισθητήρες καθώς και ο ένας από τους δύο transceivers για την επιτυχή επικοινωνία με τους σταθμούς.

Στον δεύτερο κόμβο του Arduino, που υλοποιεί τον πυροσβεστικό σταθμό - δασαρχείο, τοποθετήθηκε ένα κουμπί, ένα LED καθώς και ο δεύτερος transceiver.

Η επικοινωνία μεταξύ των κόμβων πραγματοποιείται μέσω του πρωτοκόλλου επικοινωνίας δικτύων ALOHA .Παρακάτω ακολουθεί η αναλυτική λειτουργία και εφαρμογή του κάθε στοιχείου.

Εντοπισμός σταθμού από τον φάρο.

Προκειμένου να αναγνωρίσει ο φάρος την διαθεσιμότητα ενός σταθμού, είναι συνεχώς σε ετοιμότητα να δεχτεί ένα μήνυμα από τον σταθμό. Το μήνυμα αυτό δεν είναι άλλο από τον αριθμό των φορών που έχει πατηθεί το κουμπί, τον οποίο αριθμό ο σταθμός στέλνει συνεχώς. Ο τρόπος που αντιλαμβάνεται την διαθεσιμότητα ο φάρος εξηγείται παρακάτω. Ο beacon(φάρος) θα μπορεί να γνωρίζει την ταυτότητα του σταθμού ώστε να μπορεί να δέχεται ανά πάσα στιγμή την διαθεσιμότητα του.

Για να υλοποιήσουμε το μήνυμα του σταθμού σκεφτήκαμε πως όταν το κουμπί έχει πατηθεί μονό αριθμό φορών το LED είναι αναμμένο. Αντίθετα όταν το κουμπί έχει πατηθεί ζυγό αριθμό φορών το LED σβήνει. Έτσι ,όταν ο σταθμός είναι διαθέσιμος ο beacon λαμβάνει έναν μονό αριθμό από τον συγκεκριμένο σταθμό, και έναν ζυγό όταν δεν είναι διαθέσιμος.

Όπως κάθε σταθμός διαθέτει ένα SOURCE_ADDRES, μοναδικό σε αυτόν, έτσι και κάθε κομμάτι δασικής έκτασης διαθέτει συγκεκριμένο DESTINATION_ADDRESS, αλλά και τη δική της συχνότητα επικοινωνίας για να επιτυγχάνεται η ανταλλαγή πληροφοριών χωρίς παρεμβολές από άλλες εκτάσεις.

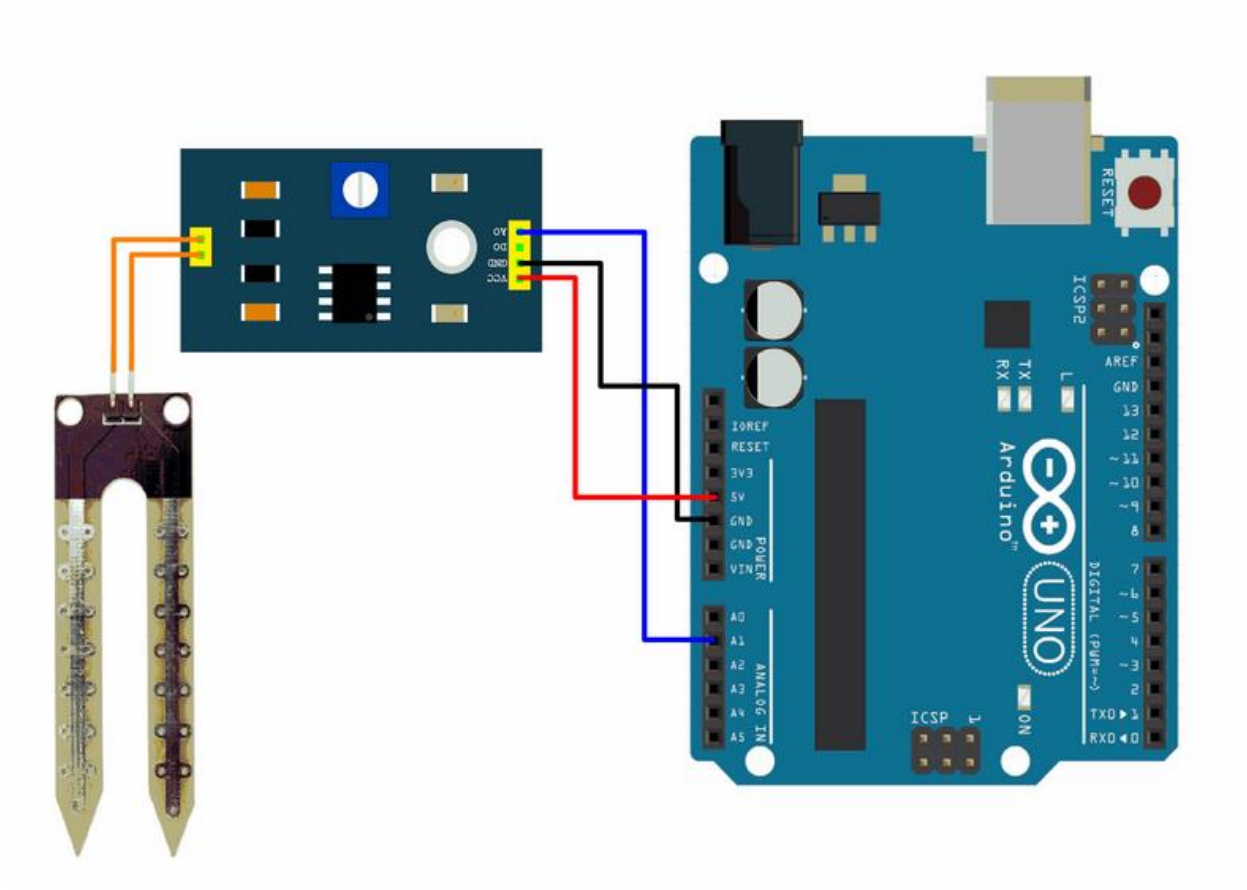
Αισθητήρας θερμοκρασίας, υγρασίας, υγρασίας εδάφους και φωτιάς στον beacon.

Προκειμένου να υλοποιηθεί ο έλεγχος της κατάστασης που επικρατεί στην δασική έκταση χρησιμοποιήθηκαν αισθητήρες θερμοκρασίας, υγρασίας, υγρασίας εδάφους και φωτιάς. Παρακάτω παρατίθεται ο κώδικας για τις μετρήσεις των αισθητήρων.

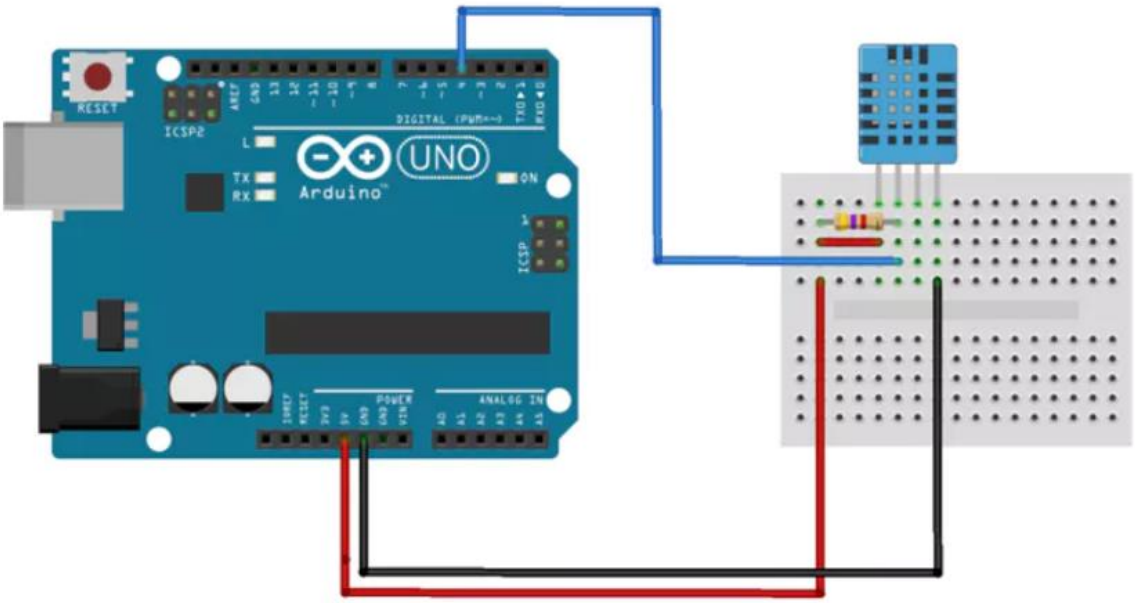
```
/*SOIL*/
  int sensorPin = A1;
  int sensorValue2;
  sensorValue2 = analogRead(sensorPin);
  float voltage3 = (sensorValue2/124.0) * 5.0;
  float soil = (voltage3 - 0.05) * 100;
  int soil2 = (int)(soil*100);
  /*TEMPERATURE-HUMIDTIY*/
  int chk = DHT.read11(DHT11_PIN);
  float voltage2=DHT.temperature;
  float temperature=voltage2;

  int temperature2=(int)(temperature);

  float voltage1=DHT.humidity;
  float humidity=voltage1;
  int humidity2=(int)(humidity);

  /*FLAME SENSOR*/
  int sensorReading = analogRead(A0);
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);
  // range value:
  switch (range) {
  case 0:    // A fire closer than 1.5 feet away.
    s=0;
    break;
    case 1:    // A fire between 1-3 feet away.
    s=1;
    break;
    case 2:    // No fire detected.
    s=2;
     break;
  }
```

Κώδικας αισθητήρων

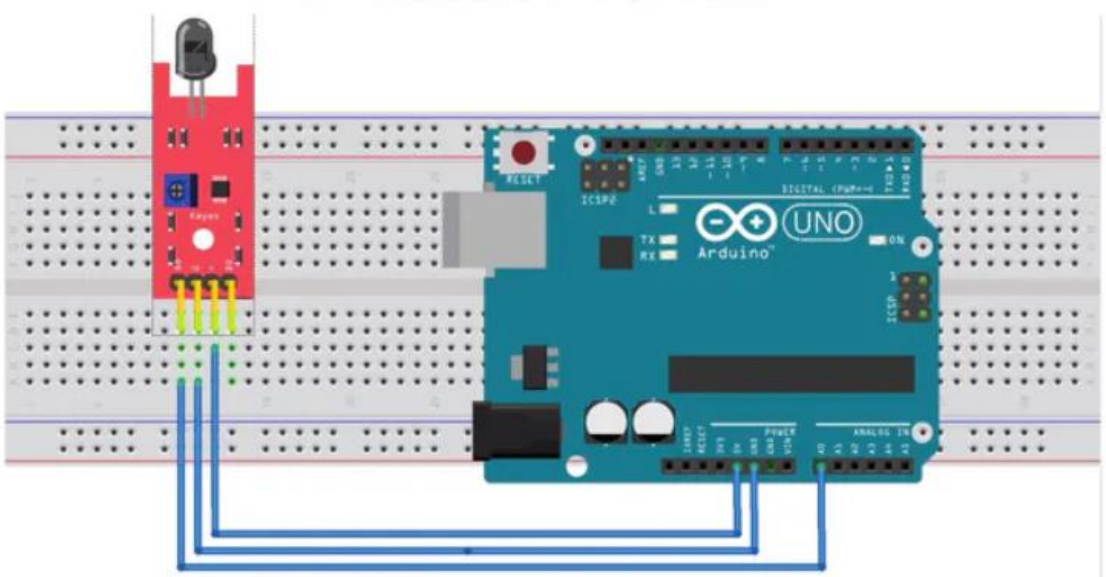
Από τα αποτελέσματα των μετρήσεων προέκυψε το συμπέρασμα ότι οι τιμές δεν μεταβάλλονται σχεδόν καθόλου σε σύντομο χρονικό διάστημα. Μόνο ο αισθητήρας φωτιάς άλλαζε ανάλογα με το αν υπάρχει φωτιά ή όχι. Παρόλα αυτά θεωρήσαμε ότι για να μπορέσει να υπάρξει δυνατότητα πυρκαγιάς θα πρέπει και οι άλλες μετρήσεις να είναι αρκετά υψηλές. Έτσι αποφασίσαμε να λαμβάνονται μετρήσεις από τους αισθητήρες κάθε 15 λεπτά ,οι οποίες και στέλνονται στους σταθμούς μέσω μιας συνάρτησης. Ακολουθούν οι συνδεσμολογίες των αισθητήρων:



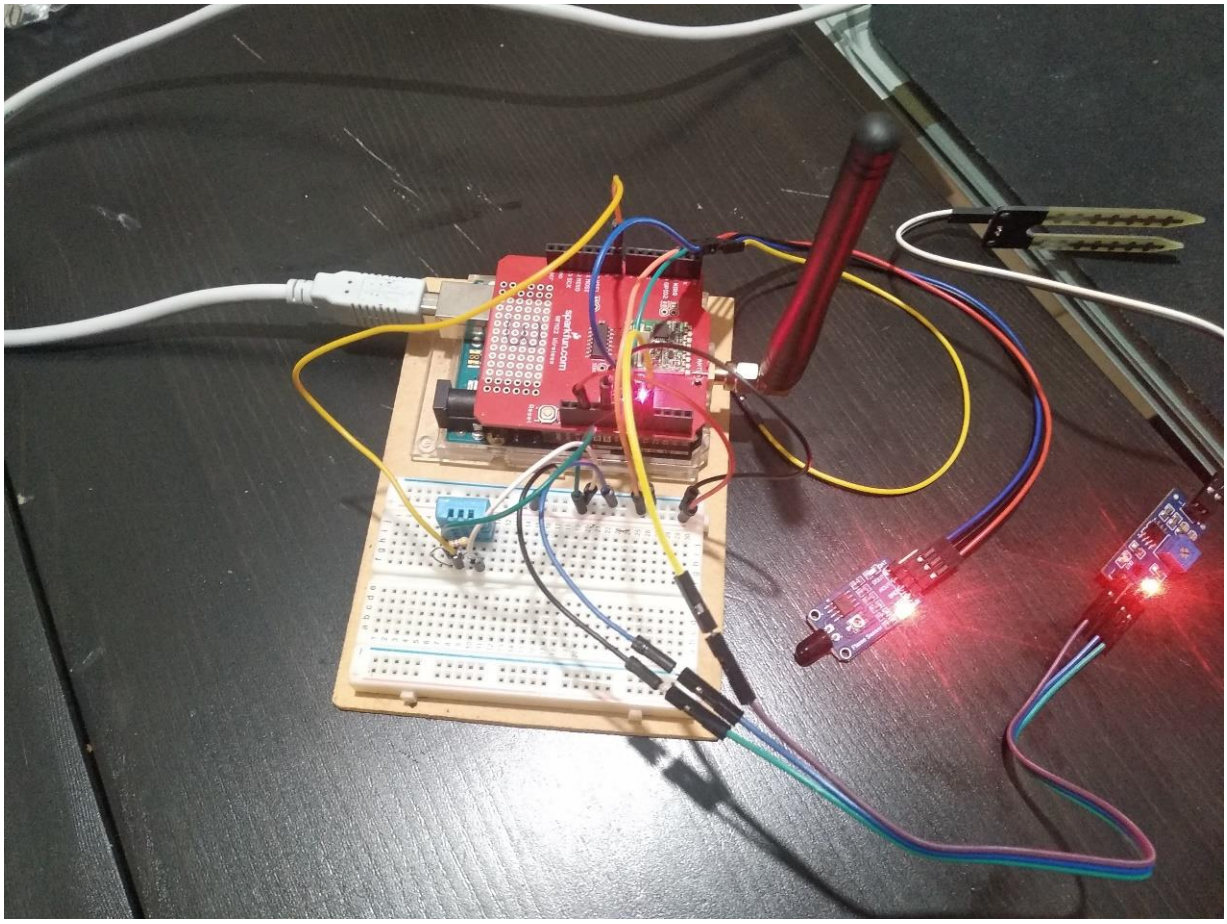
Συνδεσμολογία αισθητήρα υγρασίας εδάφους



Συνδεσμολογία αισθητήρα θερμοκρασίας-υγρασίας



Συνδεσμολογία αισθητήρα φωτιάς



Συνδεσμολογία αισθητήρων beacon

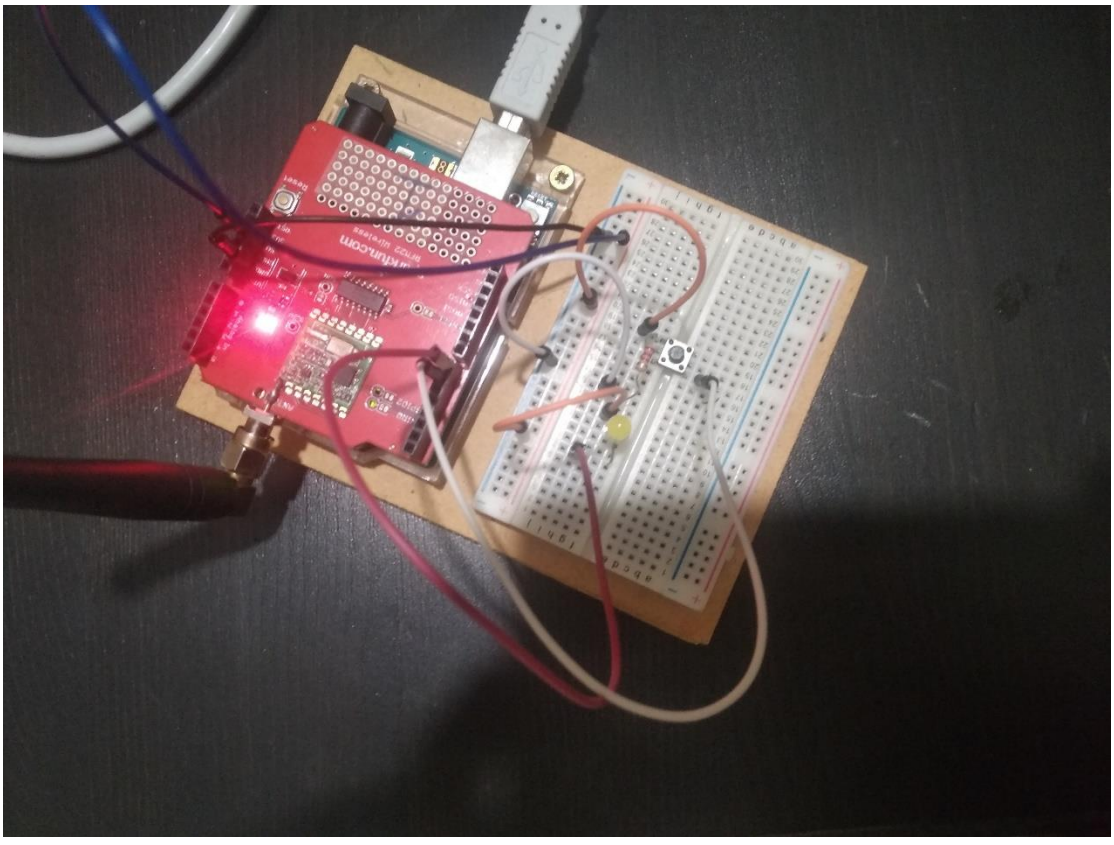
Συνδεσμολογία σταθμού

Για την διαθεσιμότητα ή μη του σταθμού σκεφτήκαμε πως το πιο απλό και εύχρηστο είναι να μπαίνει σε κατάσταση δέκτη, ανάλογα με τον αριθμό των φορών που έχει πατηθεί ένα κουμπί. Κάθε φορά που πατιέται το κουμπί αυξάνεται ο αριθμός (counter) κατά 1. Δεδομένου ότι ο counter αρχικοποιείται στο 0 (σβηστό το LED, άρα ο σταθμός δεν είναι διαθέσιμος), κάθε φορά που ο σταθμός θα γίνεται διαθέσιμος, ο counter θα παίρνει μονές τιμές (couter mod 2 !=0) και ζυγές όταν θα είναι μη διαθέσιμος (counter mod 2 ==0). Για τα παραπάνω αναπτύχθηκε το παρακάτω τμήμα κώδικα:

```
void loop() {

    buttonNew=digitalRead(btnpin);
    if(buttonOld==0 && buttonNew==1){

        if(LEDstate==0){
          digitalWrite(ledpin,HIGH);
          LEDstate=1;
          counter=counter+1;

          Serial.println("Available Station");

    }
      else{
          digitalWrite(ledpin,LOW);
          LEDstate=0;
          counter=counter+1;

          Serial.println("Unavailable Station");
        }

  }
  buttonOld=buttonNew;
```

Κώδικας για τον σταθμό

Ακολουθεί η συνδεσμολογία του σταθμού:



Συνδεσμολογία σταθμού

Σύνδεση beacon.

Η αναλυτική περιγραφή της υλοποίησης του προγράμματος ξεκινά με τον κώδικα που προσομοιώνει το κομμάτι του beacon.

Βασικός σκοπός αυτού του κώδικα είναι να δέχεται τον αριθμό των φορών που έχει πατηθεί το κουμπί από κάθε σταθμό, να μετράει χρονικά διαστήματα 15 λεπτών και όταν αυτά συμπληρωθούν να δέχεται πληροφορίες από τους αισθητήρες για τις συνθήκες που επικρατούν στην δασική έκταση. Κατόπιν οι πληροφορίες αυτές αποστέλλονται σε όλους τους διαθέσιμους σταθμούς.

Αρχικά φορτώνονται οι απαραίτητες βιβλιοθήκες για την επικοινωνία (RF22,RF22 Router) αλλά και βιβλιοθήκες για τους αισθητήρες. Ο beacon κάθε δασικής έκτασης διαθέτει την δική του ταυτότητα που συνεπάγεται της ταυτότητας του συγκεκριμένου Arduino που τον υποστηρίζει. Ανάλογα αυτής της ταυτότητας καλείται και ο constructor που δημιουργεί τη διεύθυνση του. Επιπλέον, αρχικοποιούνται global μεταβλητές για να είναι ορατές σε όλη την έκταση του προγράμματος που απαρτίζεται από διάφορες συναρτήσεις:

void setup():

Στο κομμάτι αυτό τοποθετείται κώδικας αρχικοποιήσεων, που τρέχει μόνο μια φορά. Στον προκείμενη περίπτωση, η setup χρησιμοποιείται για την εκκίνηση του Serial Monitor ώστε να εκτυπώνονται τιμές και για την ρύθμιση της επικοινωνίας (ισχύς εκπομπής , δημιουργία δρόμου διασύνδεσης ανάμεσα σε σταθμό με beacon κτλ). Πολύ σημαντικό είναι ο συντονισμός σε συγκεκριμένη συχνότητα που διαφέρει ανά δασική έκταση ώστε να επιτυγχάνεται η επικοινωνία χωρίς παρεμβολές. Επίσης, λαμβάνεται μια αρχική τιμή του χρόνου συστήματος ώστε να μετρηθεί το πρώτο διάστημα 15 λεπτών. Από εκεί και πέρα ο χρόνος αναφοράς θα ξανά αρχικοποιείται κάθε φορά που περνάει το παραπάνω χρονικό διάστημα.

void loop():

Από τη στιγμή που ο beacon είναι διαρκώς σε λειτουργία, στο τμήμα αυτό υλοποιούνται οι διεργασίες που εκτελούνται συνεχώς και επαναληπτικά. Κάποια τμήματα υλοποιούνται αυτούσια ενώ κάποια άλλα σε συναρτήσεις που καλούνται επαναληπτικά όταν ικανοποιούνται κάποιες προϋποθέσεις. Ο λόγος που χρησιμοποιήθηκαν επιπλέον συναρτήσεις είναι για την καλύτερη οργάνωση , εύκολη κατανόηση αλλά και συντήρηση του κώδικα.

Κατά τη διαρκή επανάληψη στόχος είναι να λαμβάνεται η κατάσταση διαθεσιμότητας από τους σταθμούς χρησιμοποιώντας τον βασικό κώδικα λήψης που εξηγούμε σε επόμενο στάδιο. Η σκέψη μας είναι η εξής: Στο τμήμα των global μεταβλητών έχουν δηλωθεί μεταβλητές που δηλώνουν το πλήθος των διαθέσιμων σταθμών καθώς και ένας int πίνακας τιμών με μήκος όσο και το μέγιστο πλήθος των διαθέσιμων σταθμών. Ο πίνακας αυτός αρχικοποιείται στο 0. Κάθε φορά που λαμβάνεται επιτυχώς μια τιμή από τον αριθμό των φορών που έχει πατηθεί το κουμπί, ο beacon είναι σε θέση να γνωρίζει ποιος την έστειλε. Ο κώδικας προελαύνει τη θέση που αντιστοιχεί στον αποστολέα στον πίνακα. Αν το κελί είναι αρχικοποιημένο στο 0 σημαίνει πως ο σταθμός δεν ήταν διαθέσιμος προηγουμένως. Στο κελί αντικαθίσταται η τιμή του κουμπιού. Αν ο αριθμός που έλαβε ο beacon είναι μονός, τότε αυξάνεται ο αριθμός των διαθέσιμων σταθμών κατά 1, διαφορετικά μειώνεται κατά 1. Έτσι, ο beacon μπορεί να γνωρίζει πόσες επιτυχείς αποστολές συνθηκών περιβάλλοντος πρέπει να πραγματοποιήσει, όταν έρθει η ώρα. Σημειώνεται ότι σε κάθε επανάληψη του loop πρέπει να ληφθεί ακριβώς ένας αριθμός. Αυτό σημαίνει ότι ο έλεγχος παραμένει μέσω ενός while στο κομμάτι του receive ώστε να ληφθεί επιτυχώς ένας αριθμός.

```
 boolean flag=false;
while(flag==false){
 if (rf22.recvfromAck(buf, &len, &from))
 {
   flag=true;
   buf[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
   memcpy(incoming, buf, RF22_ROUTER_MAX_MESSAGE_LEN);
   Serial.print("got request from : ");
   Serial.println(from, DEC);
   received_value = atoi((char*)incoming);

   station=atoi((char*)from);//ποιος σταθμός έστειλε το σήμα

   if(received_value %2 !=0 && present[station-1]==0)
   {
    availableStations=availableStations+1;
    present[station-1]=received_value;



   }else if (received_value %2 ==0 && present[station-1]!=0){
     present[station-1]=0;
     availableStations=availableStations-1;
   }
```

Λήψη και αναγνώριση αριθμού

Ο beacon είναι σε θέση να εξυπηρετήσει μια λήψη ανά εκτέλεση του loop. Οι πληροφορίες που στέλνουν οι σταθμοί δε χάνονται στο ενδιάμεσο, μιας που λόγω του πρωτόκολλου ALOHA οι αποστολείς κάνουν επαναλαμβανόμενες προσπάθειες αποστολής. Στο τελευταίο μέρος του loop και άρα πριν την επανεκτέλεση του τοποθετείται ο έλεγχος του χρονικού διαστήματος των 15 λεπτών. Λαμβάνεται ο τρέχων χρόνος συστήματος και συγκρίνεται με μια αρχική τιμή, που αρχικοποιήθηκε για πρώτη φορά στην setup(). Αν έχει περάσει το επιθυμητό χρονικό διάστημα και οι διαθέσιμοι σταθμοί είναι μεγαλύτεροι του 0 καλείται η συνάρτηση environment (availablestations) και αφού αυτή επιστρέψει, η τιμή αναφοράς αντικαθίσταται με την τρέχουσα ώστε να μετρηθεί εκ νέου το χρονικό διάστημα των 15 λεπτών.

void environment (availablestations):

Η συνάρτηση αυτή είναι υπεύθυνη για την λήψη μετρήσεων από τους αισθητήρες, την ειδοποίηση των σταθμών ότι ήρθε η ώρα να δεχτούν τις διαθέσιμες μετρήσεις και την αποστολή των μετρήσεων αυτών. Αρχικά, χρησιμοποιώντας τον βασικό κώδικα αποστολής, μεταδίδεται το μήνυμα ”measurements time”. Για τη μετάδοση χρησιμοποιείται συγκεκριμένος αριθμός προσπαθειών βάσει του πρωτοκόλλου ALOHA. Ο συνολικός αριθμός επιτυχών αποστολών του μηνύματος πρέπει να ταυτίζεται με τον συνολικό αριθμό διαθέσιμων σταθμών, ο υπολογισμός του οποίου αναφέρθηκε παραπάνω.

Αφού ειδοποιηθούν όλοι οι διαθέσιμοι σταθμοί, στόχος είναι με τον ίδιο τρόπο να αποσταλούν οι πληροφορίες από τους αισθητήρες. Χρησιμοποιείται και πάλι ο βασικός κώδικας αποστολής σε συνδυασμό με τη μέθοδο που αναφέρθηκε προηγουμένως. Σημαντικό είναι το γεγονός ότι δεν χρησιμοποιούνται 4 βρόγχοι αποστολής, ένας για κάθε συνθήκη περιβάλλοντος. Ο λόγος είναι ότι αυτό θα εισήγαγε αρκετή καθυστέρηση και σύγχυση στα δεδομένα που φτάνουν στους δέκτες. Για παράδειγμα αυτός που έλαβε την θερμοκρασία θα πρέπει να περιμένει να την λάβουν και όλοι οι υπόλοιποι ενώ αν κάτι δεν δουλέψει σωστά μπορεί να λάβει ξανά την θερμοκρασία και να την θεωρήσει σαν υγρασία. Χρησιμοποιείται λοιπόν ένα κοινό string όπου τυπώνονται οι 4 int τιμές:

sprintf(data_read, " %d %d %d %d",temperature2 ,humidity2 ,soil2 ,s)

Οι τιμές χωρίζονται από το χαρακτήρα του κενού (” ”) που μετά τη λήψη του μηνύματος χρησιμοποιείται ως token(εξηγείται στο τμήμα του σταθμού).Ο βασικός κώδικας αποστολής είναι ίδιος.

```
 sprintf(data_read, " %d %d %d %d",temperature2 ,humidity2 ,soil2 ,s);
 memcpy(data_send, data_read, RF22_ROUTER_MAX_MESSAGE_LEN);
 data_read[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';


for (counter2=0;counter2<availableStations;counter2++)
 {
   successful_packet=false;
   while(!successful_packet){
       if(rf22.sendtoWait(data_send,sizeof(data_send),DESTINATION_ADDRESS)!=RF22_ROUTER_ERROR_NONE)//r
        {
         //Serial.println("sendtoWait failed");
         randNumber=random(200,max_delay);//περίμενε για ένα τυχαίο χρόνο
         //Serial.println(randNumber);
         delay(randNumber);
        }
       else
        {
         successful_packet=true;
         //Serial.println("sendtoWait Succesful");
         success=success+1;

         }

    }
    if(x=success)
    {    RF22Router rf22(DESTINATION_ADDRESS2);
         rf22.addRouteTo(SOURCE_ADDRESS2,SOURCE_ADDRESS2);
         rf22.addRouteTo(SOURCE_ADDRESS3,SOURCE_ADDRESS3);
         break;
    }
```

Εκτύπωση και αποστολή συνθηκών σε κοινό string

Όταν διεκπεραιωθεί η λειτουργία της συνάρτησης, ο έλεγχος επιστρέφει στην loop().

Σύνδεση σταθμού.

Σε αυτό το τμήμα θα σχολιασθεί ο κώδικας που προσομοιώνει το κομμάτι του σταθμού.

Όπως έχει ήδη αναφερθεί, βασικός σκοπός αυτού του τμήματος είναι να αποστέλλει τον αριθμό των φορών που έχει πατηθεί το κουμπί αλλά και να δέχεται πληροφορίες για τις συνθήκες που επικρατούν στις δασικές εκτάσεις.

Όπως και στον φάρο, φορτώνονται οι βιβλιοθήκες για την επικοινωνία. Ανάλογα την ταυτότητα του συγκεκριμένου Arduino (που αποτελεί την ταυτότητα συγκεκριμένου σταθμού) καλείται και ο constructor που δημιουργεί τη διεύθυνση του. Ακόμα, αρχικοποιούνται global μεταβλητές για να είναι ορατές σε όλη την έκταση του προγράμματος.

void setup():

Στο κομμάτι αυτό τοποθετείται ο κώδικας αρχικοποιήσεων, που τρέχει μόνο μια φορά. Στον προκείμενη περίπτωση, η setup χρησιμοποιείται για την εκκίνηση του Serial Monitor ώστε να εκτυπώνονται τιμές και για την ρύθμιση της επικοινωνίας (ισχύς εκπομπής ,δημιουργία δρόμου διασύνδεσης ανάμεσα σε σταθμό με beacon κ.τ.λ.). Πολύ σημαντικό είναι ο συντονισμός σε προκαθορισμένη συχνότητα που συντονίστηκε και ο beacon για να επιτευχθεί η επικοινωνία.

void loop():

Από τη στιγμή που ο σταθμός είναι διαρκώς σε λειτουργία, στο τμήμα αυτό υλοποιούνται οι διεργασίες που εκτελούνται συνεχώς και επαναληπτικά. Κάποια τμήματα υλοποιούνται αυτούσια ενώ κάποια άλλα σε συναρτήσεις που καλούνται επαναληπτικά όταν ικανοποιούνται κάποιες προϋποθέσεις. Ο λόγος που χρησιμοποιήθηκαν επιπλέον συναρτήσεις είναι για την καλύτερη οργάνωση , εύκολη κατανόηση αλλά και συντήρηση.

Κατά τη διαρκή επανάληψη ελέγχεται ο αριθμός των φορών που έχει πατηθεί το κουμπί και στέλνει αυτό τον αριθμό μέσω της συνάρτησης transmit_availability(counter). Όσο ο σταθμός δεν είναι διαθέσιμος βρίσκεται μόνιμα σε κατάσταση πομπού. Μόλις ο σταθμός γίνει διαθέσιμος στέλνει τον μονό αριθμό και γίνεται δέκτης.

```
void loop() {

    buttonNew=digitalRead(btnpin);
    if(buttonOld==0 && buttonNew==1){

        if(LEDstate==0){
          digitalWrite(ledpin,HIGH);
          LEDstate=1;
          counter=counter+1;

          Serial.println("Available Station");

    }
      else{
          digitalWrite(ledpin,LOW);
          LEDstate=0;
          counter=counter+1;

          Serial.println("Unavailable Station");
        }

  }
  buttonOld=buttonNew;


    transmit_availability(counter);
```

Λήψη αριθμού και αποστολή αυτού

voids transmit_availability (int counter):

Στο σημείο αυτό ο έλεγχος μεταφέρεται από την loop() σε αυτή την συνάρτηση που είναι υπεύθυνη για τη μετάδοση του αριθμού των φορών που έχει πατηθεί το κουμπί στον beacon για τους σκοπούς που έχουν ήδη αναλυθεί. Πρόκειται για τον βασικό κώδικα μετάδοσης ALOHA που θα αναλυθεί στην επόμενη ενότητα. Μετά την επιτυχή μετάδοση, ο έλεγχος επιστρέφει στην loop().

void environment_receive():

Αφού σταλεί ο αριθμός(counter),ο έλεγχος επιστρέφει στην loop() και ακολουθεί ένα τμήμα κώδικα που αφορά τη λήψη των συνθηκών περιβάλλοντος. Πιο συγκεκριμένα, γίνεται έλεγχος αν το κουμπί έχει πατηθεί μονό αριθμό. Αν ισχύει η συνθήκη τότε ο σταθμός μπαίνει σε κατάσταση δέκτη και περιμένει το μήνυμα “measurements time” που αποστέλλεται από τον beacon κάθε 15 λεπτά. Αξίζει να αναφερθεί το γεγονός ότι κατά την αναμονή του μηνύματος γίνεται συνέχεια έλεγχος για το αν το κουμπί ξαναπατηθεί. Αν πατηθεί και ο σταθμός γίνει μη διαθέσιμος τότε ξαναμπαίνει σε κατάσταση πομπού και γίνεται break της while().

```
if(counter%2!=0){
RF22Router rf22(DESTINATION_ADDRESS);
rf22.addRouteTo(SOURCE_ADDRESS,SOURCE_ADDRESS);

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;
boolean flag=false;
/* ΠΕΡΙΜΕΝΕ ΜΕΧΡΙ ΝΑ ΛΑΒΕΙΣ ΤΟ msg ΕΠΙΒΕΒΑΙΩΣΗΣ.
 * ΣΕ ΠΕΡΙΠΤΩΣΗ ΠΟΥ ΘΕΛΗΣΕΙΣ ΝΑ ΓΙΝΕΙΣ ΠΑΛΙ ΠΟΜΠΟΣ
 * ΠΡΙΝ ΛΑΒΕΙΣ ΤΙΣ ΜΕΤΡΗΣΕΙΣ
 * ΚΑΝΕ ΕΛΕΓΧΟ ΓΙΑ ΤΟ ΚΟΥΜΠΙ ΚΑΙ ΚΑΝΕ BREAK ΤΗΝ WHILE
 * ΑΦΟΥ ΠΡΩΤΑ ΞΑΝΑΓΙΝΕΙΣ ΠΟΜΠΟΣ
*/
while(flag=false){

  buttonNew=digitalRead(btnpin);
  if(buttonOld==0 && buttonNew==1){
    if(LEDstate==0){
      digitalWrite(ledpin,HIGH);
      LEDstate=1;
      counter=counter+1;
      Serial.println("Available Station");
                  }
  else{
      digitalWrite(ledpin,LOW);
      LEDstate=0;
      counter=counter+1;
      Serial.println("Unavailable Station");
      }
}
 buttonOld=buttonNew;
 if(counter%2==0){
   RF22Router rf22(SOURCE_ADDRESS2);
   rf22.addRouteTo(DESTINATION_ADDRESS2,DESTINATION_ADDRESS2);
  break;
    }
```

Αφού συμπληρωθεί το χρονικό διάστημα των 15 λεπτών και ο σταθμός παραμείνει διαθέσιμος, τότε λαμβάνει τα δεδομένα από τον beacon και τα συγκρίνει μέσω της εντολής strcmp() με το μήνυμα. Αν είναι το επιθυμητό μήνυμα τότε καλείται η συνάρτηση environment_receive().

```
        if(rf22.recvfromAck(buf,&len,&from)){
          buf[RF22_ROUTER_MAX_MESSAGE_LEN-1]='\0';
          memcpy(incoming,buf,RF22_ROUTER_MAX_MESSAGE_LEN);
          Serial.print("got request from:");
          Serial.println(from,DEC);
          if(!strcmp(incoming,msg)){
          environment_receive();
          }
        }
       }
      }

}
```

Σύγκριση ληφθέντος μηνύματος με επιθυμητή τιμή

Μόλις ληφθεί το μήνυμα με τις πληροφορίες περιβάλλοντος σειρά έχει το parsing του string.Το parsing υλοποιείται μέσω της εντολής strtok . Ορίζεται ως token το κενό και κάθε φορά που εντοπίζεται, λαμβάνεται ένα ξεχωριστώ τμήμα από το string. Τα τμήματα αυτά ύστερα μετατρέπονται σε integer μέσω της atoi.

```
  while (flag==false){
   if (rf22.recvfromAck(buf, &len, &from))
{
  flag=true;
  buf[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
  memcpy(incoming, buf, RF22_ROUTER_MAX_MESSAGE_LEN);
  Serial.print("got request from : ");
  Serial.println(from, DEC);
  const char d[3]=" ";
  char *token;

  token=strtok(incoming,d);
 float temperature2=atof(token);
  token=strtok(NULL,d);
 float humidity2=atof(token);
  token=strtok(NULL,d);
 float soil2=atof(token);
  token=strtok(NULL,d);
 float flame2=atof(token);
  Serial.print("Region's temperature is: ");
  Serial.println(temperature2);

 Serial.print("Region's humidity is: ");
  Serial.println(humidity2);
  Serial.print("Ground's humidity of the region is: ");
 Serial.println(soil2/100);
  if(flame2==2){
    Serial.println("No fire");

    }else if(flame2==1){
     Serial.println("Fire between 1-3 feet away");
      }else{
          Serial.println("Fire closer than 1.5 feet away");
        }


  delay(3000);
```

Parsing του string συνθηκών και εκτύπωση αυτών

Πριν το τέλος της συνάρτησης, μεσολαβεί μια καθυστέρηση 3 δευτερολέπτων μιας που το αμέσως επόμενο βήμα είναι ο έλεγχος και η αποστολή για το πόσες φορές έχει πατηθεί το κουμπί στην loop(), τη στιγμή που είναι πιθανόν ο beacon να αποστέλλει ακόμα συνθήκες περιβάλλοντος στους υπόλοιπους σταθμούς.

Βασικοί κώδικες επικοινωνίας

Ξεκινώντας με τον βασικό κώδικα αποστολής, 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.

Ύστερα, σε ό,τι αφορά τον βασικό κώδικα λήψης, η διαδικασία είναι ακριβώς αντίστροφη. Η λήψη και ο έλεγχος επιτυχίας της πραγματοποιούνται μέσω της recvfromAck() . Τα δεδομένα που καταφθάνουν αποθηκεύονται σε έναν πίνακα τύπου unsigned integer, μήκους 50. Στη συνέχεια, στο τελευταίο στοιχείο τοποθετείται η τιμή NULL προκειμένου ο πίνακας να μετατρέπει επιτυχώς σε έναν ενδιάμεσο πίνακα char. Εν τέλει, ο πίνακας αυτός μετατρέπεται στον επιθυμητό τύπο δεδομένων όπως αναφέρθηκε παραπάνω. Τονίζεται ότι μέσω μιας μεταβλητής from ο κώδικας είναι σε θέση να γνωρίζει τον αποστολέα του πακέτου πληροφορίας.

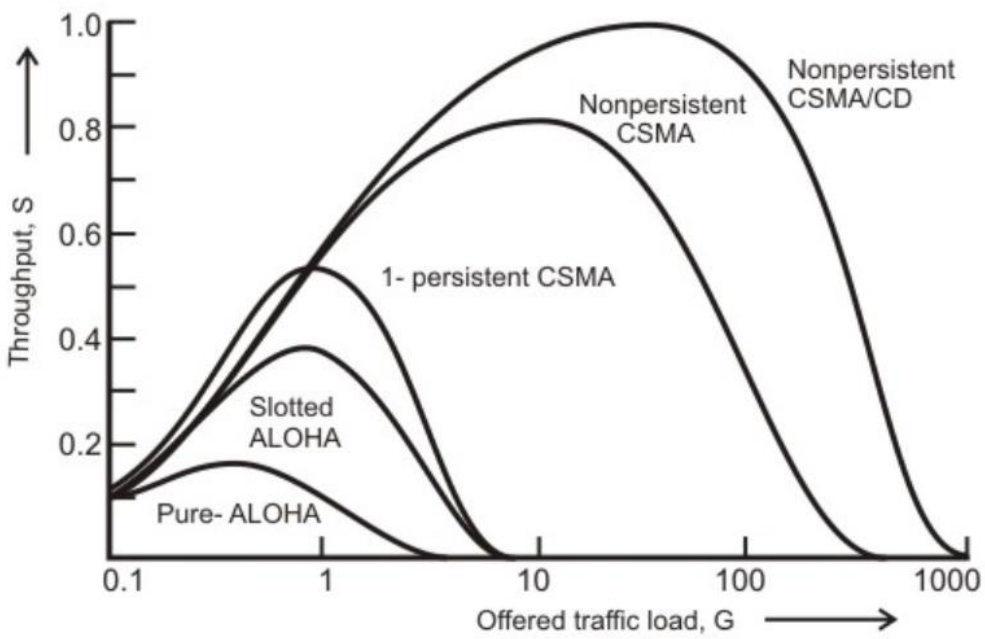
Εναλλακτικές Σχεδίασης

Για την καλύτερη επεξεργασία και μετάδοση των πληροφοριών είναι δυνατή η χρήση διαφορετικών microcontroller boards με ενσωματωμένο Wi-Fi module , έτσι ώστε τα δεδομένα να “ανεβαίνουν” απευθείας σε μια βάση , με αποτέλεσμα την ευκολότερη επεξεργασία τους και την απλοϊκότερη επικοινωνία μεταξύ πομπού και δέκτη.

Τέτοια modules είναι οι ESP8266 , ESP32 , ESP8285.

Επίσης , με τη χρήση κάποιου GPS θα μπορούσε να γίνει ο ακριβής προσδιορισμός της περιοχής που έχει ξεσπάσει κάποια πυρκαγιά. Όμως , με αυτό τον τρόπο αυξάνεται το κόστος.

Όσον αφορά την επικοινωνία στο δίκτυο μας , θα μπορούσε να γίνει με χρήση διαφορετικού πρωτοκόλλου πέρα από το ALOHA , όπως για παράδειγμα τα CSMA πρωτόκολλα , τα οποία είναι πολύ πιο αποδοτικά όπως φαίνεται στο παρακάτω διάγραμμα.



Πρακτικότητα

Λόγοι για τους οποίους η πρότασης μας χαρακτηρίζεται πρακτική:

- Σε κάθε κομμάτι δασικής έκτασης ο φάρος έχει διαφορετική συχνότητα , οπότε δεν υπάρχουν παρεμβολές στην επικοινωνία.
- Το κόστος κατασκευής του φάρου αλλά και του σταθμού είναι αρκετά χαμηλό.
- Η χρήση από το σταθμό είναι απλή , όποτε δε χρειάζεται κάποια ιδιαίτερη εξοικείωση.