ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΦΑΡΜΟΓΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΔΙΑΤΑΞΕΩΝ

ΟΜΑΔΑ DUCKLINGS

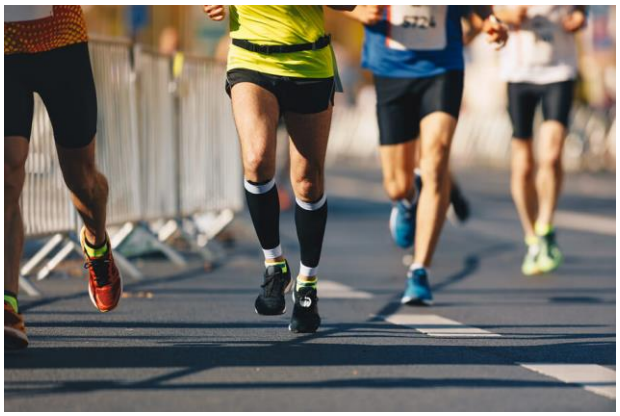
BIORUN

Αναστάσιος Μουρατίδης 9040
Χριστίνα Παρδάλη 9039

ΠΕΡΙΕΧΟΜΕΝΑ

1. Περιγραφή προβλήματος και προτεινόμενη λύση

Οι μαραθώνιοι αγώνες είναι μεγάλες διοργανώσεις που λαμβάνουν χώρα σε μήκος 42,2 χιλιομέτρων και παίρνουν μέρος εκατοντάδες άνθρωποι. Κατά τη διάρκεια ενός μαραθωνίου μπορούν να προκύψουν διάφορα προβλήματα υγείας στους αθλητές, μιας και πρόκειται για έντονη σωματική άθληση η οποία εκτελείται σε ένα αρκετά μεγάλο χρονικό διάστημα και κάποιες φορές κάτω από ιδιαίτερες καιρικές συνθήκες.



Παρατηρήσαμε, λοιπόν, ότι απουσιάζει ένα σύστημα παρακολούθησης της υγείας των αθλητών αλλά και των περιβαλλοντικών συνθηκών στις οποίες αθλούνται. Γι αυτό, λοιπόν, προτείνουμε την εφαρμογή μας, ***BioRun***, η οποία θα παρέχει στους διοργανωτές ενημέρωση σε πραγματικό χρόνο για την υγεία των αθλητών και τις καιρικές συνθήκες σε όλα τα σημεία της διαδρομής. Επίσης, οι αθλητές θα ενημερώνονται με τα αντίστοιχα μηνύματα διαλείμματος σε περίπτωση υπερβολικής αύξησης των παλμών τους, πολλής ζέστης ή έντονης βροχής. Η ζωντανή ενημέρωση που παρέχεται στους διοργανωτές θα βοηθήσει στην άμεση παροχή βοήθειας σε όσους την χρειαστούν, και θα αποφευχθούν τελικά πιο σοβαρά ατυχήματα.

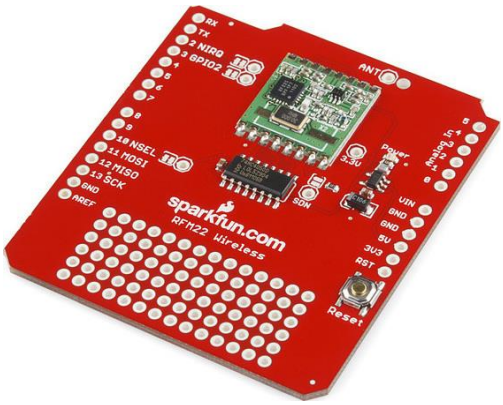
Συγκεκριμένα η εφαρμογή μας στηρίζεται στη δημιουργία ενός ασύρματου δικτύου αισθητήρων. Κάθε κόμβος του δικτύου αντιπροσωπεύει έναν αθλητή και αποτελείται από τρεις αισθητήρες, έναν αισθητήρα μέτρησης παλμών, έναν αισθητήρα θερμοκρασίας και έναν αισθητήρα μέτρησης βροχής. Ιδανικά, ο

κάθε αθλητής θα έχει ένα wearable με τους προαναφερθέντες αισθητήρες ώστε να μην παρεμποδίζεται η άσκηση του.

Τα μηνύματα από τους αισθητήρες των αθλητών αποστέλλονται σε έναν κεντρικό κόμβο(σταθμός βάσης) στον οποίο έχουν πρόσβαση οι διοργανωτές του μαραθωνίου, χρησιμοποιώντας το πρωτόκολλο ασύρματης επικοινωνίας ALOHA.

2. Ανάλυση σχεδίασης Δικτύου

Το ασύρματο δίκτυο αισθητήρων αποτελείται από 4 Arduino Uno Boards,4 RFM22 Transceiver Modules και 4 κεραίες ώστε να επιτυγχάνεται η επικοινωνία μεταξύ των κόμβων. Στην παρακάτω εικόνα φαίνονται τα εξαρτήματα που χρησιμοποιήθηκαν:



Εικόνα 1: RFM22 Wireless Card

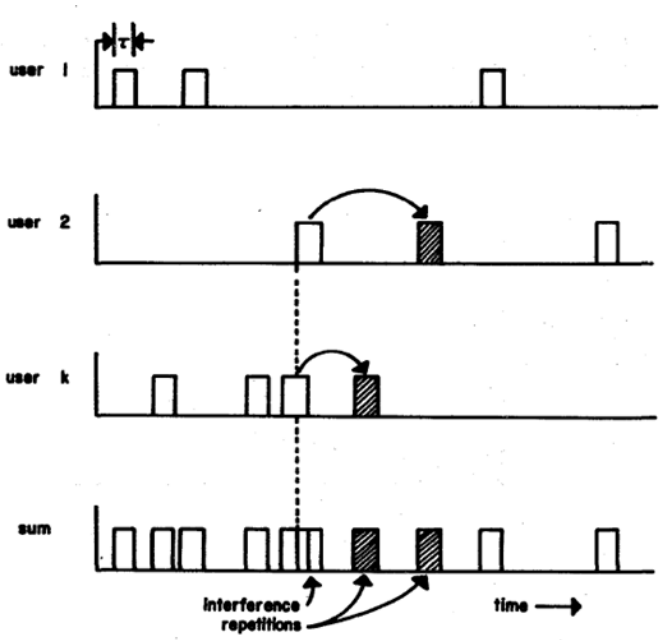
Εικόνα 2: Arduino Uno Board

Εικόνα 3: Πανκατευθυντική κεραία

Το πρωτόκολλο που θα χρησιμοποιηθεί για την μετάδοση δεδομένων από τους 3 κόμβους προς τον σταθμό βάσης είναι το ALOHA. Είναι ένα πρωτόκολλο ανταγωνισμού που επιτρέπει την κοινή χρήση του μέσου μετάδοσης από όλους τους σταθμούς του δικτύου. Ο κάθε σταθμός μπορεί να στείλει δεδομένα σε οποιαδήποτε χρονική στιγμή και το βασικό χαρακτηριστικό αυτού του πρωτοκόλλου είναι η απουσία της διαδικασίας ανίχνευσης φέροντος. Αυτό σημαίνει πως ο σταθμός στέλνει δεδομένα χωρίς προηγουμένως να ελέγξει εάν το κανάλι είναι άδειο ή εάν εκείνη τη χρονική στιγμή στέλνει κάποιος άλλος σταθμός.

Έτσι είναι πολύ πιθανό να λάβει χώρα εκπομπή δεδομένων από κάποιο σταθμό, την ώρα που στο κανάλι κινείται το πακέτο δεδομένων κάποιου άλλου χρήστη. Στην περίπτωση αυτή λαμβάνει χώρα μία σύγκρουση που οδηγεί στην αμοιβαία καταστροφή των πακέτων. Εάν ο σταθμός που έστειλε το πακέτο καταλάβει ότι αυτό έχει καταστραφεί, περιμένει για ένα τυχαίο διάστημα, και μετά το στέλνει εκ νέου.

Η διαδικασία αυτή επαναλαμβάνεται συνεχώς, μέχρι τελικά η μετάδοση του πακέτου να είναι επιτυχής. Ο τρόπος λειτουργίας του πρωτοκόλλου ALOHA φαίνεται στην παρακάτω εικόνα:

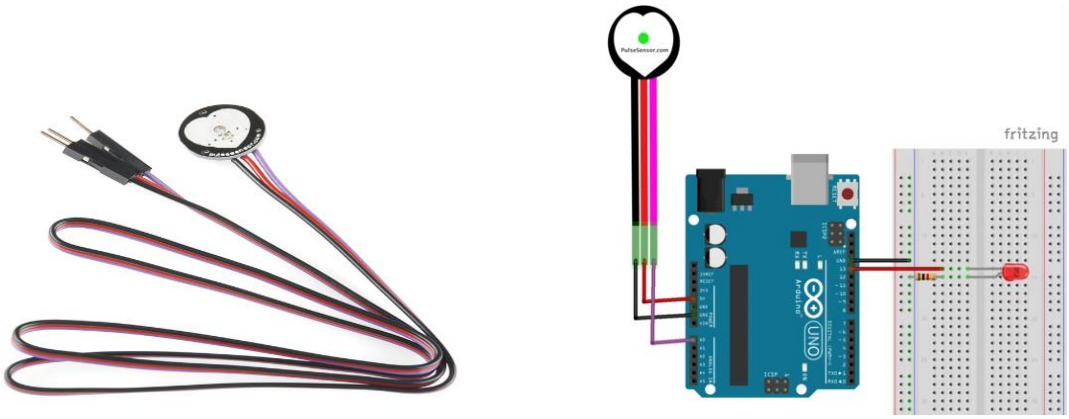


Εικόνα 4: Πρωτόκολλο ALOHA

Στο δίκτυο μας, ο ένας κόμβος αποτελεί τον σταθμό βάσης που συγκεντρώνει τις πληροφορίες από τους άλλους τρεις και τις εμφανίζει στην οθόνη. Από τους υπόλοιπους τρεις κόμβους ο ένας διαθέτει τους αισθητήρες μέτρησης ενώ οι άλλοι δύο παράγουν τυχαίες τιμές στα επιθυμητά διαστήματα. Συγκεκριμένα, οι αισθητήρες που χρησιμοποιήθηκαν για τη δημιουργία του δικτύου είναι:

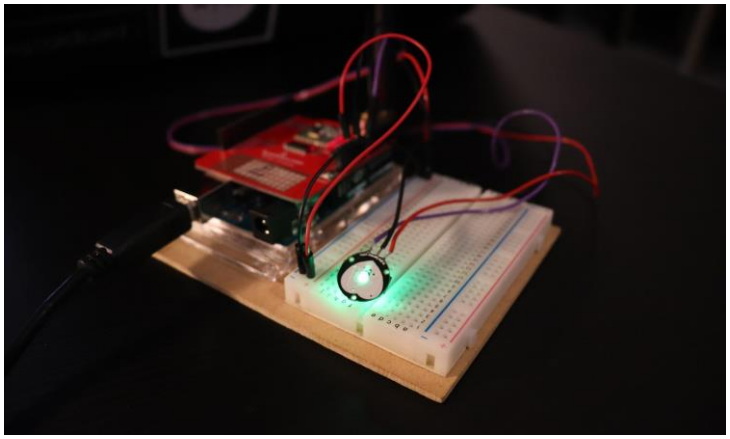
2.1 Pulse Sensor SEN-11574

Χρησιμοποιώντας τον αισθητήρα αυτό μετράμε τον καρδιακό ρυθμό των αθλητών. Η συνδεσμολογία του αισθητήρα στο board φαίνεται στην παρακάτω εικόνα. Στη συνδεσμολογία του δικτύου μας χρησιμοποιήσαμε το ενσωματωμένο led του Arduino και όχι ένα εξωτερικό.



Εικόνα 5: Pulse Sensor και Συνδεσμολογία του

Η πειραματική διάταξη του αισθητήρα και οι τιμές που εμφανίζει είναι οι εξής:



```
♥  A HeartBeat Happened !
BPM: 36
♥  A HeartBeat Happened !
BPM: 39
♥  A HeartBeat Happened !
BPM: 43
♥  A HeartBeat Happened !
BPM: 50
♥  A HeartBeat Happened !
BPM: 60
♥  A HeartBeat Happened !
BPM: 75
```

Εικόνα 6: Κύκλωμα και αποτελέσματα Pulse Sensor

Ορισμένα σημεία του κώδικα του αισθητήρα φαίνονται παρακάτω:

- Αρχικά, συμπεριλαμβάνουμε την βιβλιοθήκη PulseSensorPlayground.h και δημιουργούμε τόσο τις κατάλληλες μεταβλητές όσο και το αντικείμενο.

```
#define USE_ARDUINO_INTERRUPTS true     // Set-up low-level interrupts for most acurate BPM math.
#include <PulseSensorPlayground.h>

//  Variables and object for Pulse Sensor
const int PulseWire = 0;       // PulseSensor PURPLE WIRE connected to ANALOG PIN 0
const int LED13 = 13;          // The on-board Arduino LED, close to PIN 13.
int Threshold = 550;           // Determine which Signal to "count as a beat" and which to ignore.

PulseSensorPlayground pulseSensor;
```

- Στη συνέχεια, στη συνάρτηση setup αρχικοποιούμε τα χαρακτηριστικά του αντικειμένου και ελέγχουμε την ορθή λειτουργία του.

```
void setup() {

  Serial.begin(9600);
  // Configure the PulseSensor object, by assigning our variables to it.
  pulseSensor.analogInput(PulseWire);
  pulseSensor.blinkOnPulse(LED13);
  pulseSensor.setThreshold(Threshold);

  // Check if Pulse Sensor object is created
  if (pulseSensor.begin()) {
     Serial.println("Pulse Sensor Object created");
```

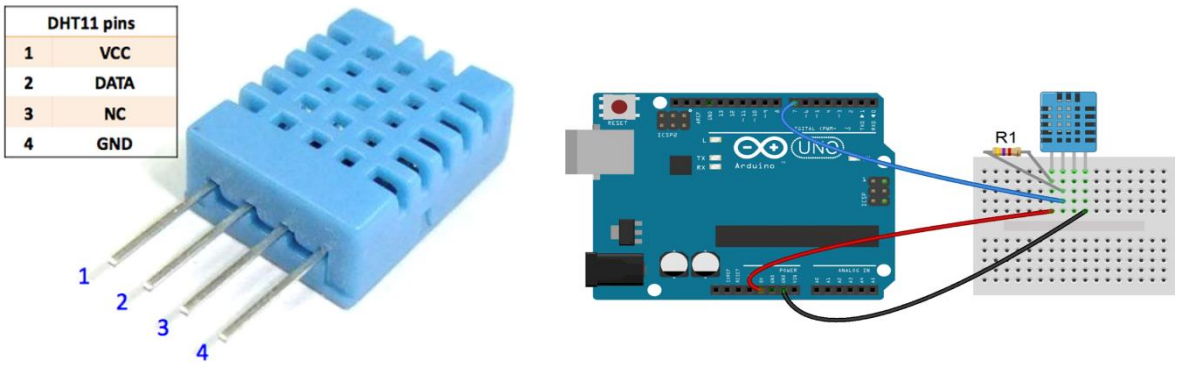
- Τέλος,μέσα στη συνάρτηση loop παίρνουμε την μέτρηση καρδιακών παλμών,εμφανίζουμε το ανάλογο μήνυμα και αποθηκεύουμε τη μέτρηση σε μια μεταβλητή.

```
void loop() {
  delay(2000);
  int myBPM = pulseSensor.getBeatsPerMinute();  // Get BPM Measurement

 if (pulseSensor.sawStartOfBeat()) {            // Constantly test to see if "a beat happened".
  Serial.println("♥  A HeartBeat Happened ! ");
  Serial.print("BPM: ");
  Serial.println(myBPM);
 }
 //4 Measurements to be sent
  int SensorValue1 = myBPM;
```

2.2 Temperature and Humidity Sensor DHT11

Με τον αισθητήρα αυτό οι αθλητές θα έχουν την δυνατότητα να βλέπουν την θερμοκρασία και το ποσοστό υγρασίας που επικρατούν στον χώρο που τρέχουν καθ'όλη τη διάρκεια των 42 χιλιομέτρων.



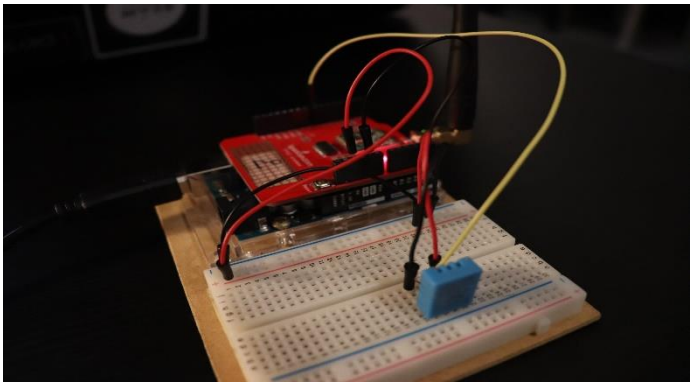
Εικόνα 7: Αισθητήρας DHT11 και η Συνδεσμολογία του

Μερικά χαρακτηριστικά του αισθητήρα είναι:

- Εύρος Υγρασίας: $20 - 90\%\ RH$
- Ακρίβεια Υγρασίας: $\pm 5\%\ RH$
- Εύρος Θερμοκρασίας: $0 - 50\ ^{\circ}C$
- Ακρίβεια Θερμοκρασίας: $\pm 2\%\ ^{\circ}C$

- Τάση λειτουργίας: $3V$ *έως* $5.5V$

Η πειραματική διάταξη του αισθητήρα και οι τιμές που εμφανίζει είναι οι εξής:



```
Temperature = 24.00
Humidity = 44.00
Temperature = 24.00
Humidity = 44.00
Temperature = 24.00
Humidity = 44.00
Temperature = 24.00
Humidity = 43.00
Temperature = 25.00
Humidity = 43.00
```

Εικόνα 8: Κύκλωμα και αποτελέσματα DHT11 Sensor

Ορισμένα σημεία του κώδικα του αισθητήρα φαίνονται παρακάτω:

- Αρχικά, συμπεριλαμβάνουμε την βιβλιοθήκη DHT.h και κάνουμε define το DHTPIN και DHTTYPE.

```
#include <RF22.h>
#include <SPI.h>
#include <RF22Router.h>
#include "DHT.h"

#define DHTPIN 7
#define SOURCE_ADDRESS 2 //Node 2
#define DESTINATION_ADDRESS 10
#define DHTTYPE DHT11
```

- Έπειτα, δημιουργούμε ένα αντικείμενο τύπου dht.

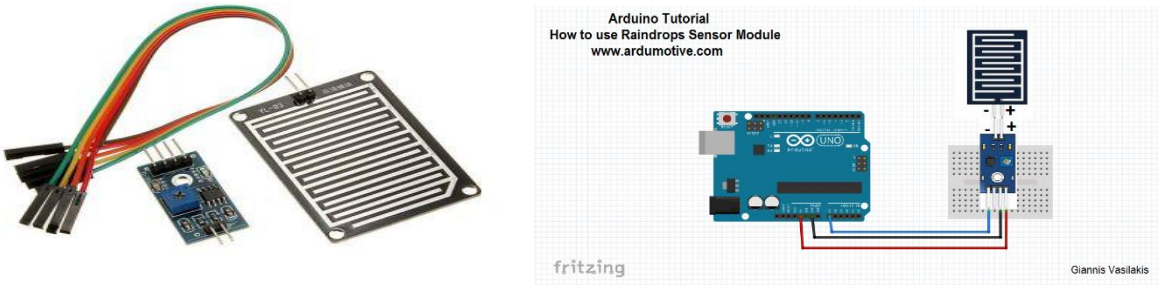
```
// Object for Temperature-Humidity Sensor
DHT dht(DHTPIN, DHTTYPE);
```

- Τέλος, παίρνουμε μέσω του αντικειμένου τις μετρήσεις θερμοκρασίας και υγρασίας και τις αποθηκεύουμε σε δύο μεταβλητές.

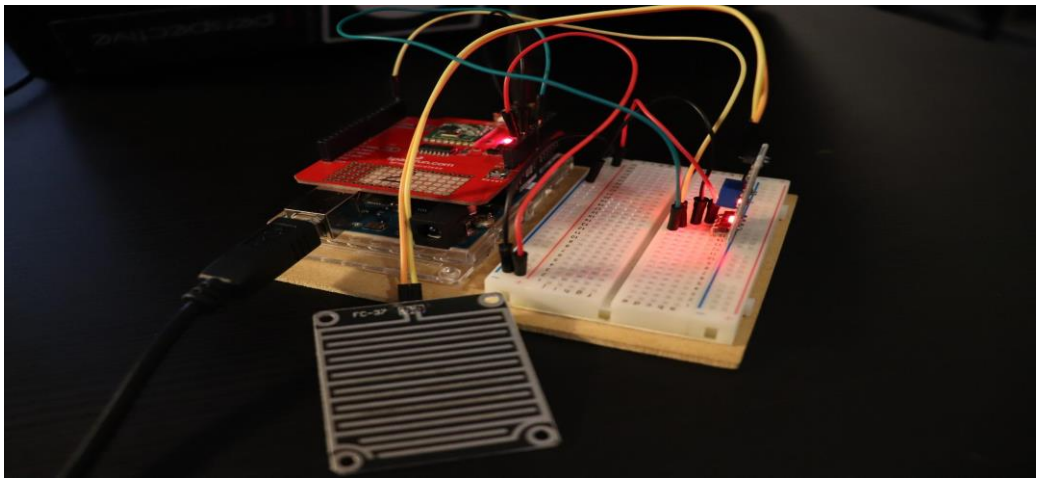
```
int SensorValue3 = dht.readTemperature();
int SensorValue4 = dht.readHumidity();
```

2.3 Rain Sensor

Οι ενδείξεις του συγκεκριμένου αισθητήρα θα ενημερώνουν τους αθλητές για την πιθανότητα βροχής ώστε να αναζητήσουν το κοντινότερο σημείο για στάση. Οι ενδείξεις που μπορεί να εμφανίσει στην οθόνη είναι Raining, Rain Warning και Not Raining.



Εικόνα 9: Rain Sensor και η Συνδεσμολογία του



Εικόνα 10: Κύκλωμα Rain Sensor

Ορισμένα σημεία του κώδικα του αισθητήρα φαίνονται παρακάτω:

- Αρχικά, δημιουργούμε τις απαραίτητες global μεταβλητές.

```
// Variables initialized for Rain Sensor
const int sensorMin = 0 ;
const int sensorMax = 1024;
```

- Έπειτα, κάτω από τη συνάρτηση loop δημιουργούμε τη συνάρτηση RainMeasure() που υπολογίζει την πιθανότητα βροχής.

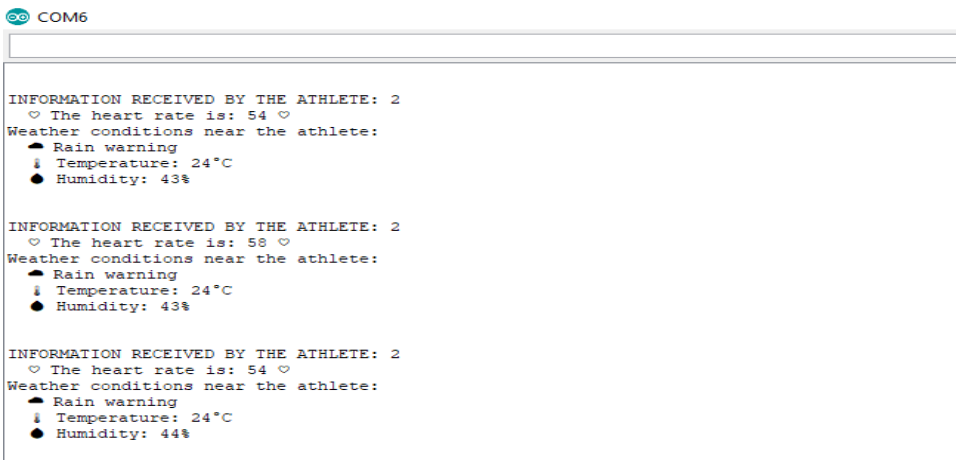
```
}

//Function to get the Rain Sensor Measurement
int RainMeasure(){
  int sensorReading=analogRead(A1);
  int range=map(sensorReading,sensorMin,sensorMax,0,3);
  switch (range){
  case 0:    // Sensor getting wet
    Serial.println("It's raining");
    break;
  case 1:    // Sensor getting wet
    Serial.println("Rain Warning");
    break;
  case 2:    // Sensor dry - To shut this up delete the " Serial.println("Not Raining"); " below.
    Serial.println("Not Raining");
    break;
  }
  return range;
}
```

- Τέλος, μέσα στη συνάρτηση loop καλούμε τη συνάρτηση RainMeasure και αποθηκεύουμε το αποτέλεσμα σε μία μεταβλητή.

```
int SensorValue2 = RainMeasure();
```

Συνδέοντας τους τρεις προαναφερθέντες αισθητήρες στο κόμβο 2 του δικτύου μας, τα δεδομένα αποστέλλονται στον σταθμό βάσης και εμφανίζονται στην οθόνη όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 11: Οθόνη σταθμού βάσης

3. Θεωρητική Ανάλυση

3.1 Ανάλυση κριτηρίων χωρητικότητας και αριθμός καναλιών

Για την θεωρητική ανάλυση του δικτύου μας θεωρούμε ότι στο μαραθώνιο δρόμο συμμετέχουν 200 αθλητές. Κάθε αθλητής-κόμβος παράγει 1 πακέτο/s και η διάρκεια του πακέτου είναι 20ms. Με βάση τα στοιχεία αυτά ο απαιτούμενος αριθμός καναλιών χρησιμοποιώντας το πρωτόκολλο ALOHA είναι :

Αριθμός καναλιών $N = \frac{200 * \left(1\ {}^{\pi\alpha\kappa\acute{\varepsilon}\tau o}/_{s}\right) * 0.02}{0.186} \approx 22$

3.2 Ανάλυση κριτηρίων κάλυψης

Σκοπός μας είναι να επιλέξουμε τον ελάχιστο αριθμό σταθμών βάσης και να τους τοποθετήσουμε κατάλληλα στην υπάρχουσα γεωμετρία έτσι ώστε να έχουμε καθυστέρηση μικρότερη των 200ms.
Τα απαραίτητα στοιχεία για την ανάλυση του δικτύου με το πρωτόκολλο ALOHA είναι:

- Αριθμός αθλητών-κόμβων 200
- Κάθε κόμβος στέλνει 1 πακέτο/s με διάρκεια πακέτου 20ms
- Μήκος του πακέτου που αποστέλλεται: $45\ bytes$, δηλαδή $45 * 8 = 360\ bits$
- Χρόνος διάδοσης $\tau = 0.001\ ms$
- Τυχαία καθυστέρηση $X = 100ms$
- Β ρυθμός μετάδοσης δικτύου
- Ισχύς σε απόσταση $100m\ -50dBm$
- Απόσβεση ισχύος συναρτήσει της δεύτερης δύναμης της απόστασης
- Διασπορά σ κανονικής κατανομής $10\ dBm$
- Επιτυχής κάλυψη 95% του χρόνου

$$S = \lambda * T = (200 * 1)\frac{\pi\alpha\kappa\acute{\varepsilon}\tau\alpha}{s} * 0.02 = 4$$

Επομένως, ακόμα κι αν τέλεια κατανομή της κίνησης στο πεδίο του χρόνου, θα χρειαζόμασταν τουλάχιστον 4 κανάλια.

Ο ρυθμός μετάδοσης του δικτύου υπολογίζεται $B = \frac{360}{0,02} = 18000\ bps$.

Η ισχύς P_{min} υπολογίζεται από τον τύπο $P_{min} = -130dBm + 10 * log_{10}(B) = -130dBm + 10 * log_{10}(18000) = -87.45\ dBm$

Για το πρωτόκολλο ALOHA με το συγκεκριμένο a από τους τύπους και τα σχετικά γραφήματα, η μέγιστη εξυπηρετούμενη κίνηση S ανά συχνοτικό κανάλι που εξασφαλίζει τη ζητούμενη μέγιστη καθυστέρηση των $200\ ms$ είναι 0.18
Οπότε χρειαζόμαστε $\frac{4}{0.18} = 22.22$, δηλαδή τουλάχιστον 23 συχνοτικά κανάλια και με δεδομένο ότι κάθε σταθμός εξυπηρετεί $N_{max} = 10$ κανάλια, θα τοποθετήσουμε τουλάχιστον 3 σταθμούς.
Έχουμε: $\frac{P_{r_0}}{P_{min}} = \frac{r_{max}^n}{r_0^n}$, για $n = 2,\ r_0 = 100m,,\ P_{r_0} = -60dBm$

Υπολογίσαμε παραπάνω πως για τον συγκεκριμένο ρυθμό μετάδοσης είναι

$$P_{sensitivity} = -87.45\ dBm.$$

Βέβαια, λόγω διαλείψεων, χρειαζόμαστε περισσότερη ισχύ στον δέκτη. Για κάλυψη 0.95 έχουμε

$$q = 0.95 \rightarrow 2q - 1 = 0.9$$

Από τους πίνακες της κανονικής κατανομής για το όρισμα της erf, προκύπτει

$$\gamma = 1.15 * \sqrt{2} * 12 = 19.51\ dBm.$$

Έτσι,

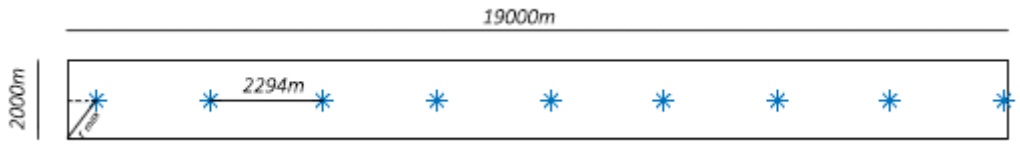
$$P_{min} = P_{sensitivity} + \gamma = -70.41\ dBm.$$

Τελικά, λύνοντας ως προς r_{max}, έχουμε

$$r_{max} = \sqrt{\frac{10^{-6}}{10^{-7.119}} * 100^2} = 1147m$$

Ο μαραθώνιος είναι περίπου 42 χιλιόμετρα, και επιλέγουμε την ακόλουθη γεωμετρία για την αναπαράστασή του, δηλαδή ένα ορθογώνιο με περίμετρο $42000m$ και πλευρές $19000m\ και\ 2000m$. Υπολογίζουμε την θέση του πρώτου σταθμού (από αριστερά), έτσι ώστε να καλύπτει το άκρο της διαδρομής

$d = \sqrt{1147^2 - 1000^2} = 561.79m$. Τελικά για παροχή κάλυψης σε όλο το μήκος της διαδρομής χρειάζονται 9 σταθμοί που απέχουν μεταξύ τους $2294m$, και τοποθετούνται όπως φαίνονται στο σχήμα.



Εικόνα 12: Τοποθέτηση σταθμών βάσης του δικτύου

4. Επεξήγηση κώδικα

Ο κώδικας που δημιουργήσαμε για το ασύρματο δίκτυο αποτελείται από 4 διαφορετικά αρχεία .ino, 3 για τους Transmitters και 1 για τον Receiver.
Εκτός από την απαραίτητη ανάλυση που έγινε νωρίτερα για τους κώδικες των επιμέρους αισθητήρων , παρακάτω υπάρχουν ορισμένα σημεία που χρήζουν προσοχής στους ολοκληρωμένους κώδικες:

1. Στους κώδικες των Transmitters ορίζουμε τα στοιχεία της ασύρματης κάρτας δηλώνοντας ως συχνότητα λειτουργίας 431MHz, ισχύ εκπομπής 20dBm και είδος διαμόρφωσης GFSK. Με τα ίδια χαρακτηριστικά λειτουργεί και ο Receiver του δικτύου.

```
// Set frequency
if (!rf22.setFrequency(431.0)){
   Serial.println("setFrequency Fail");}
// Set power and modulation
rf22.setTxPower(RF22_TXPOW_20DBM);
rf22.setModemConfig(RF22::GFSK_Rb125Fd125);
```

2. Στους κόμβους 5 και 7 παράγουμε τυχαίους αριθμούς σε ορισμένα διαστήματα ώστε να προσομοιώνουν τις τιμές των πραγματικών αισθητήρων. Συγκεκριμένα, χρησιμοποιούμε την συνάρτηση random(minValue,MaxValue) όπως φαίνεται και στον παρακάτω κώδικα:

```
void loop() {

  int SensorValue1= random(50,200);
  int SensorValue2= random(0,2);
  int SensorValue3=random(0,50);
  int SensorValue4=random(20,80);
```

3. Για την αποστολή των δεδομένων από τους Transmitters αποφασίσαμε να χρησιμοποιήσουμε ένα πακέτο με τα δεδομένα και των 3 αισθητήρων μιας και το μήκος του πακέτου της ασύρματης κάρτας επαρκούσε για όλους τους αισθητήρες. Έτσι , οι τιμές των 3 αισθητήρων αποτυπώνονται σε ένα string και έχουν ως διαχωριστικό το σύμβολο του κενού ώστε ο Receiver να μπορέσει να ξεχωρίσει τις τιμές.

```
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 %d %d", SensorValue1,SensorValue2,SensorValue3,SensorValue4);
data_read[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
memcpy(data_send, data_read, RF22_ROUTER_MAX_MESSAGE_LEN);
successful_packet = false;
```

Στον Receiver αντίστοιχα, διαχωρίζουμε τα δεδομένα των 3 αισθητήρων χρησιμοποιώντας τις συναρτήσεις strtok και atoi όπως φαίνεται στον παρακάτω κώδικα:

```
const char s[3] = " " ;
char *token ;
token = strtok(incoming,s);
int BPM = atoi(token);
token = strtok(NULL,s);
int Rain = atoi(token);
token=strtok(NULL,s);
int Temp = atoi(token);
token = strtok(NULL,s);
int Hum = atoi(token);
```

4. Μέσα στη συνάρτηση void loop() του κόμβου 2 που έχει τους αισθητήρες εισάγουμε ένα delay(2000) έτσι ώστε οι αισθητήρες να παίρνουν τις σωστές μετρήσεις στα κατάλληλα διαστήματα. Για την ομοιόμορφη λειτουργία του δικτύου εισάγουμε την ίδια καθυστέρηση και στους άλλους δύο κόμβους του δικτύου.

5. Όσον αφορά στον υπολογισμό του ποσοστού επιτυχίας, μετράμε τον αριθμό τόσο των επιτυχημένων εκπομπών όσο και των επανεκπομπών που χρειάστηκαν λόγω σύγκρουσης των πακέτων. Υπολογίζουμε το Success Ratio ως το κλάσμα των επιτυχημένων εκπομπών προς το άθροισμα των επιτυχημένων εκπομπών και των επανεκπομπών. Στην περίπτωση του δικτύου μας, εξαιτίας του μικρού αριθμού των κόμβων αλλά και των πακέτων που στέλνουν, το ποσοστό επιτυχίας ήταν 100%. Στο πλαίσιο πειραμάτων με τις καθυστερήσεις που εισάγαμε στους κόμβους, παρατηρούμε ότι όταν αφαιρούμε την καθυστέρηση των 2 δευτερολέπτων από τους κόμβους που αποστέλλουν τις ψευδοτυχαίες τιμές, υπάρχουν μερικές συγκρούσεις πακέτων, αλλά αρκετά σπάνια. Έτσι, το ποσοστό επιτυχίας παραμένει υψηλό κοντά στο 97%. Η καθυστέρηση που εισάγεται στον κόμβο με τους αισθητήρες δεν θα μπορούσε να παραλειφθεί μιας και θα εμποδίζεται η ορθή ανάγνωση των τιμών από τους αισθητήρες.
6. Όσον αφορά στον υπολογισμό της διεκπεραιωτικής ικανότητας (throughput), μετράμε τον χρόνο πετυχημένης αποστολής ενός πακέτου σε έναν κόμβο και το μέγεθος του πακέτου. Οπότε υπολογίζουμε το throughput ως το λόγο του μεγέθους του πακέτου ως προς το χρόνο πετυχημένης αποστολής του. Το throughput των πομπών είναι περίπου

5. Ψευδοκώδικες

Receiver

Include libraries
Define nodes of network
Initialize an object RF22

Setup():

Open the Serial Port

Define RF22 characteristics

Add all the routes

Loop():

Initialize RF22 parameters

Allocate the memory to receive the data

If(receive_packet=true)

{

Receive data and send acknowledgement

Print the node that sent the message

Split the packet in the four measurements

Print measurements

}

Transmitter 2

Include libraries

Define pins of sensors

Define nodes of network

Initialize variables and object of Pulse Sensor

Initialize variables of Rain Sensor

Initialize object of Temperature-Humidity Sensor

Initialize object of RF22

Initialize variables for Aloha Protocol

Initialize variables for success rate and throughput

Setup():

Open serial port

Configure the PulseSensor Object

Define RF22 characteristics

Add all the routes

Create the random seed

Loop():

Wait for 2000ms

Get measurements from 3 sensors
Load Measurements in one packet
If(Get Measurement)
{
Flag_sent=FALSE;
While(!Flag_sent)
{
Try to send
If(ack)
{
Flag_Sent=true;
Increase successfulTrasmissions;
}
Else
{
Increase retransmissions;
Delay=Rand(uniform,0,max);
Wait(for Delay);
}
}
}
Calculate Success Ratio
Calculate Throughput
}

RainMeasure():

Get measurement of Rain Sensor
Return measurement

Transmitters 5,7

Include libraries
Define nodes of network
Initialize variables and object of Aloha Protocol and RF22 library
Initialize variables for success rate and throughput

Setup():

Open serial port
Define RF22 characteristics
Add all the routes
Create the random seed
Loop():
Wait for 200ms
Get random measurements
If(Get Measurement)
{
Flag_sent=FALSE;
While(!Flag_sent)
{
Try to send
If(ack)
{
Flag_Sent=true;
Increase successfulTrasmissions;
}
Else
{
Increase retransmissions;
Delay=Rand(uniform,0,max);
Wait(for Delay);
}
}
}

Calculate Success Ratio
Calculate Throughput
}

6. Πρακτικότητα και εναλλακτικές σχεδίασης

Σκοπός της εφαρμογής BioRun είναι να παρέχει χρήσιμες πληροφορίες τόσο στους μαραθωνοδρόμους όσο και στους διοργανωτές των μαραθωνίων δρόμων.

Η πρακτικότητα του δικτύου έγκειται στο γεγονός ότι οι αθλητές θα μπορούν να διαθέτουν ένα wearable καθ'όλη τη διάρκεια του δρόμου ώστε να ενημερώνονται live για τις καιρικές συνθήκες και να λαμβάνουν τις ανάλογες αποφάσεις. Αν για παράδειγμα λάβουν μήνυμα ότι η θερμοκρασία ανεβαίνει ή η βροχή αυξάνεται μπορούν να σταματήσουν στο κοντινότερο σημείο που έχει οριστεί από την διοργάνωση είτε για διάλειμμα είτε για παροχή βοήθειας. Η διοργάνωση από την πλευρά της θα έχει τη δυνατότητα να λαμβάνει τους παλμούς κάθε αθλητή και έτσι να παρακολουθεί την υγεία τους.

Σε επόμενο στάδιο, θα μπορούσε να ληφθεί υπόψη ο φυσιολογικός ρυθμός των παλμών ανά ηλικία σύμφωνα με τον παρακάτω πίνακα:

Ηλικία	Στόχος	Μέγιστοι παλμοί
20	100-170 bpm	200 bpm
30	95-162 bpm	190 bpm
40	93-157 bpm	180 bpm
45	90-153 bpm	175 bpm
50	88-149 bpm	170 bpm
55	85-145 bpm	165 bpm
60	83-140 bpm	160 bpm
65	80-136 bpm	155 bpm
70	75-128 bpm	150 bpm

Εικόνα 13: Παλμοί ανά ηλικία

Στο wearable του αθλητή θα υπάρχει η δυνατότητα εμφάνισης του αριθμού παλμών του και σε περίπτωση που είναι εκτός των φυσιολογικών ορίων για την ηλικία του και την ένταση της άσκησης, ένα μήνυμα θα τον προτρέπει να σταματήσει για σύντομο χρονικό διάστημα. Τα ίδια στοιχεία θα εμφανίζονται και στον κεντρικό σταθμό βάσης έτσι ώστε οι διοργανωτές να μπορούν να επέμβουν σε περίπτωση έκτακτης ανάγκης.

Εικόνα 14: Wearable

Ως επέκταση του υπάρχοντος δικτύου μπορούν να προστεθούν στα wearable των ασθενών περισσότεροι βιοϊατρικοί αισθητήρες όπως αισθητήρας πίεσης αίματος και αισθητήρας μυογραφήματος. Επιπλέον, μπορούν να έχουν ενσωματωμένο ένα GPS Module έτσι ώστε η διοργάνωση να παρακολουθεί τους αθλητές και σε περίπτωση έκτακτης ανάγκης όπως υπερβολικής αύξησης παλμών ένα ασθενοφόρο να μεταβεί στην ακριβή τοποθεσία.

Εικόνα 15: EMG Sensor

Εικόνα 16: GPS Module

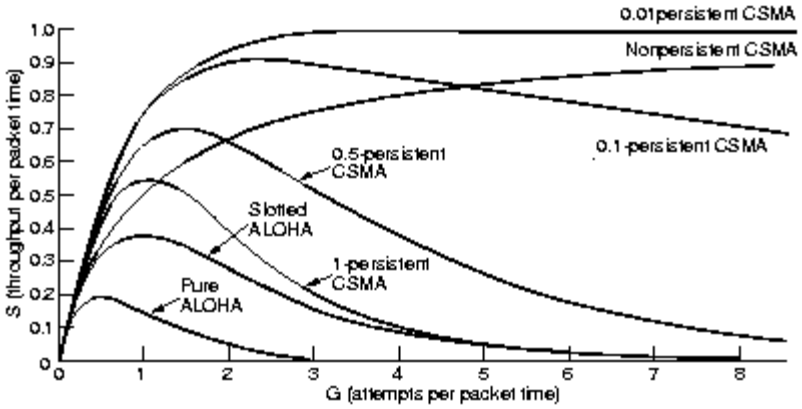
Ακόμη μία χρήσιμη προσθήκη στην εφαρμογή θα ήταν η αποστολή όλων των δεδομένων στο cloud και η αποθήκευση τους μέσω μιας ασύρματης κάρτας με Wifi. Έτσι, οι αθλητές θα μπορούν να βλέπουν στατιστικές τιμές από τα δεδομένα των αισθητήρων και να κρατούν το δικό τους προσωπικό αρχείο από μαραθώνιους δρόμους που συμμετέχουν. Ενδεικτικά, μπορεί να χρησιμοποιηθεί το παρακάτω Wifi Module:

Εικόνα 17: Wifi Module

Επιπλέον, το δίκτυο της εφαρμογής BioRun μπορεί να χρησιμοποιηθεί όχι μόνο σε μαραθώνιους δρόμους αλλά και σε ορειβατικούς ή και αναρριχητικούς αγώνες. Ειδικά σε αυτές τις περιπτώσεις όπου οι διαδρομές είναι δύσβατες και η ανάγκη παρακολούθησης των αθλητών αυξάνεται, η εφαρμογή BioRun μπορεί να αποβεί ιδιαίτερα χρήσιμη.

Τέλος, όσον αφορά το πρωτόκολλο επικοινωνίας του ασύρματου δικτύου αισθητήρων, χρησιμοποιούμε το ALOHA, το οποίο αξιοποιεί το πολύ το 18.6% του διαθέσιμου καναλιού για αποστολή δεδομένων. Θα μπορούσαμε να χρησιμοποιήσουμε το Slotted-ALOHA που αξιοποιεί το διπλάσιο ποσοστό του

διαθέσιμου καναλιού σε σχέση με το απλό ALOHA. ΄Άλλο ένα πρωτόκολλο που μπορεί να εφαρμοστεί στο δίκτυό μας είναι CSMA.



Εικόνα 18: Σύγκριση MAC Protocols