Εφαρμογές Τηλεπικοινωνιακών Διατάξεων

Parking Made Faster

ΚΑΡΑΜΗΤΟΠΟΥΛΟΣ ΠΑΝΑΓΙΩΤΗΣ ΑΕΜ 9743 email: karamitopp@ece.auth.gr

ΣΑΡΙΔΑΚΗ ΜΑΡΙΑ-ΡΑΦΑΗΛΙΑ ΑΕΜ 9633 email: saridakm@ece.auth

ΣΙΔΗΡΟΠΟΥΛΟΣ ΛΕΩΝΙΔΑΣ ΑΕΜ 9818 email: leonsidi@ece.auth.gr

ΤΗΜΜΥ ΑΠΘ Ιούνιος 2022

Περιεχόμενα

1. Περιγραφή προβλήματος και προτεινόμενη λύση

Παρατηρήσαμε ότι στην πόλη μας, όπως και σε πολλές άλλες, υπάρχει πρόβλημα εύρεσης θέσης στάθμευσης, με αποτέλεσμα οι οδηγοί να σπαταλούν συχνά αρκετό χρόνο για το παρκάρισμα. Για αυτό, επιλέξαμε να σχεδιάσουμε ένα σύστημα parking σε επίπεδο πόλης. Συγκεκριμένα, σχεδιάσαμε ένα σύστημα με βάση το οποίο ελέγχονται οι ελεύθερες θέσεις σε κάθε parking μίας πόλης, ώστε οι οδηγοί να ενημερώνονται για αυτές από οθόνες σε κεντρικούς δρόμους της πόλης, ώστε να μειώνεται ο χρόνος αναζήτησης θέσης στάθμευσης, ενώ στο κάθε parking ξεχωριστά αξιοποιούνται διάφοροι αισθητήρες για τη βέλτιστη εμπειρία των οδηγών. Αν και η γενική ιδέα αφορά όλα τα parking μίας πόλης και οθόνες σε αρκετούς δρόμους της, σε επίπεδο demo υλοποιήσαμε το σύστημα μόνο με δύο ενδεικτικά parking, τα οποία στέλνουν τις απαραίτητες πληροφορίες για τις ελεύθερες θέσεις τους σε μία μόνο οθόνη.

2. Ανάλυση Σχεδίασης Δικτύου

Το ασύρματο δίκτυο αισθητήρων αποτελείται από 4 Arduino, 3 Sparkfun RFM22 και 3 κεραίες, ώστε να επικοινωνούν μεταξύ τους οι 3 κόμβοι.

Figure 1: Arduino Uno

Figure 2: RFM22 Wireless Card

Figure 2 Σπαστή Κεραία

Στο δίκτυο που υλοποιήθηκε ο ένας κόμβος αποτελεί τον σταθμό βάσης, στον οποίο συγκεντρώνονται τα δεδομένα από τους 2 άλλους σταθμούς και εμφανίζονται στην LCD οθόνη και στην οθόνη του υπολογιστή. Το 4ο board Arduino UNO χρησιμοποιείται τοπικά σε ένα από τα 2 κλειστά parking που υλοποιήθηκαν, αλλά μπορεί με την προσθήκη ασύρματης κάρτας και κεραίας να χρησιμοποιηθεί αυτόνομα ως κόμβος σε οποιοδήποτε υπαίθριο parking.

Για την μετάδοση των δεδομένων από τους 2 κόμβους προς τον τρίτο κόμβο που αποτελεί τον σταθμό βάσης χρησιμοποιήθηκε το πρωτόκολλο ALOHA. Το πρωτόκολλο αυτό είναι ένα πρωτόκολλο όπου οι κόμβοι ανταγωνίζονται για το μέσο. Ο κάθε κόμβος μόλις θέλει να στείλει κάτι, απλά το στέλνει και περιμένει αναγνώριση λήψης από τον σταθμό (acknowledgement). Αν δεν πάρει acknowledgement, σημαίνει ότι και κάποιος άλλος κόμβος έστειλε δεδομένα εκείνη την στιγμή και υπήρξε σύγκρουση (collision), σε αυτήν την περίπτωση, ξεκινά ένας τυχαίος timer (μια αντίστροφη μέτρηση), μόλις εκπνεύσει ο timer, στέλνει τα data. Αν υπάρξει και πάλι σύγκρουση/εις επαναλαμβάνεται η διαδικασία με νέο timer μέχρι να φτάσουν τα δεδομένα στον προορισμό τους. Το πρωτόκολλο αυτό δεν έχει την δυνατότητα ακρόασης του μέσου, έτσι συχνά μπορεί να υπάρχουν συγκρούσεις (collision).

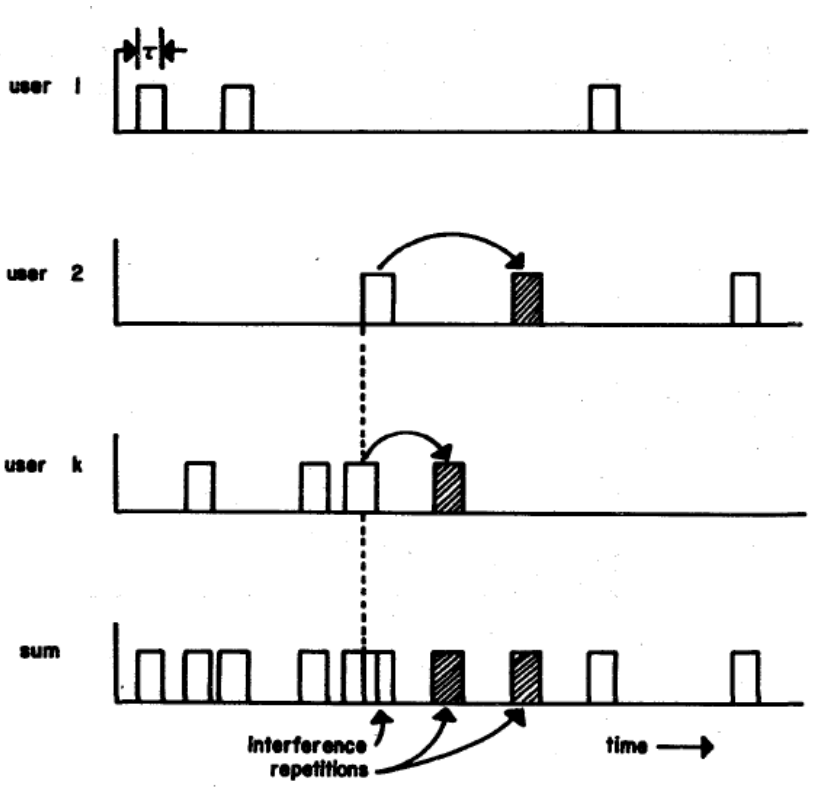


Figure 4 ALOHA communication multiplexing

2.1 Κόμβος – Transmitter(x2)

2.1.1 Servomotor(x2)

Χρησιμοποιήθηκαν 2 σερβοκινητήρες, ο ένας στην είσοδο του parking και ο άλλος στην έξοδο, προκειμένου να ανοιγοκλείνουν οι μπάρες του parking όταν εισέρχεται ή εξέρχεται κάποιο αυτοκίνητο.

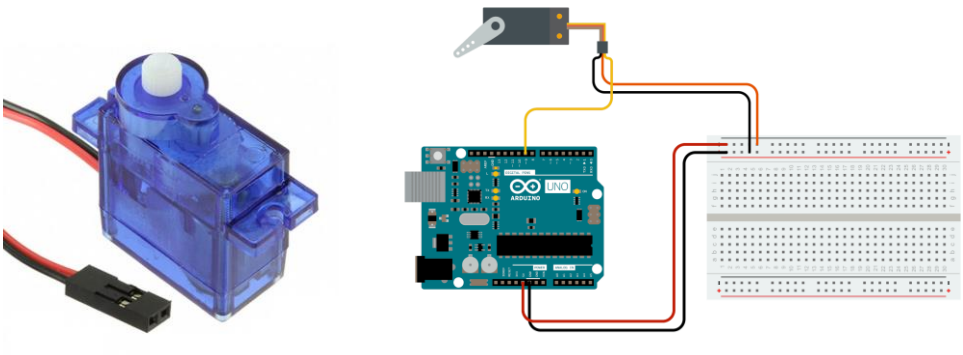


Figure 5 Servomotor και συνδεσμολογία

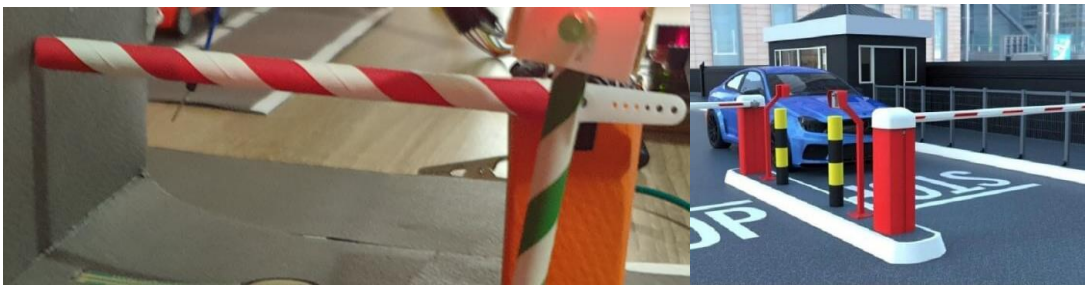


Figure 6 Μπάρα στην είσοδο/έξοδο των parking

Ορισμένα τμήματα του κώδικα για τον σερβοκινητήρα:

- Αρχικά, συμπεριλαμβάνεται η βιβλιοθήκη Servo.h, δημιουργούνται 2 μεταβλητές Servo s1, s2 και ορίζονται 2 pins ως Servopins.

```
#include <Servo.h>

int ServoPin = 3;
Servo s;
int ServoPin1 = 5;
Servo s1;
```

- Έπειτα στην συνάρτηση setup αρχικοποιούνται τα χαρακτηριστικά των s1 και s2. Ενώ οι σερβοκινητήρες ρυθμίζονται αρχικά να είναι στραμμένοι στις 0 μοίρες.

```
s.attach(ServoPin);
s1.attach(ServoPin1);
s1.write(0);
s.write(0);
```

- Τέλος στην συνάρτηση loop οι σερβοκινητήρες ρυθμίζονται να στραφούν κατά 90 μοίρες όταν εξέρχεται/εισέρχεται κάποιο όχημα και μετά επανέρχονται στην μηδενική γωνία.

```
s1.write(90);

...

s1.write(0);
```

2.1.2 Red and Green LEDs (x2)

Σε κάθε είσοδο/έξοδο των parking τοποθετήθηκε από ένα φανάρι (πράσινο – κόκκινο), προκειμένου να γίνεται πράσινο όταν ανοίξουν οι μπάρες για να μπει/βγει κάποιο όχημα και κόκκινο όταν οι μπάρες είναι κλειστές. Για κάθε φανάρι χρησιμοποιήθηκε ένα κόκκινο και ένα πράσινο led, στο καθένα συνδέθηκε εν σειρά μια αντίσταση 330 ή 220 Ω.

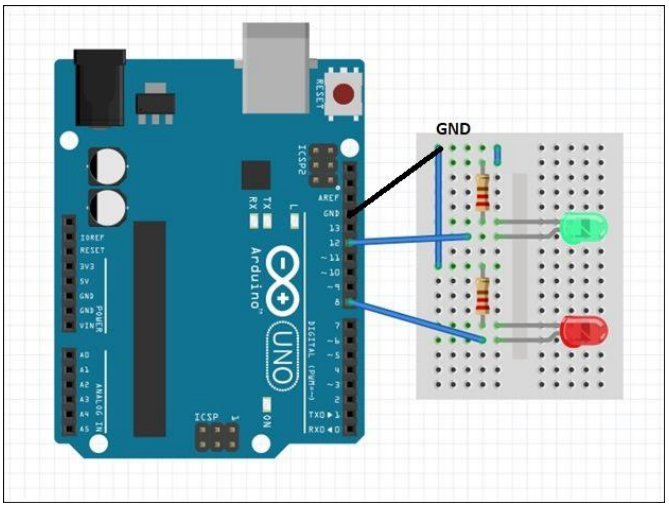


Figure 7 Συνδεσμολογία Φαναριών

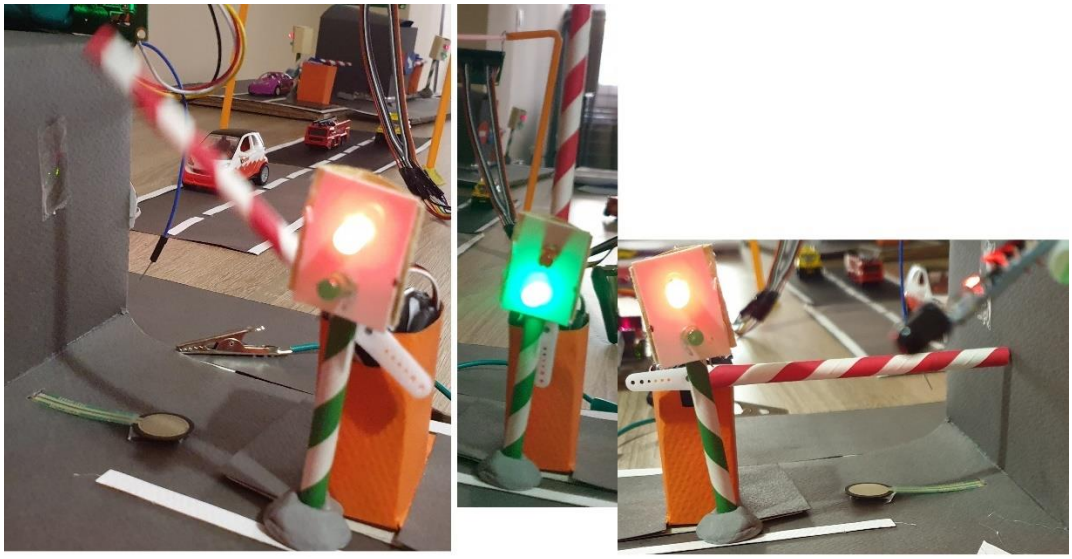


Figure 8 Φανάρια

Ορισμένα τμήματα του κώδικα για τα φανάρια:

- Ορίζονται αρχικά σε ποιά pin είναι συνδεδεμένοι οι άνοδοι των leds.

```
int eisodos_led_red = 7;
int eisodos_led_green = 8;

int exit_led_red = 6;
int exit_led_green = 9;     /
```

- Στην συνάρτηση setup ορίζονται τα παραπάνω pins ως output και ανάβουν τα κόκκινα φανάρια.

```
pinMode(eisodos_led_red, OUTPUT);
pinMode(eisodos_led_green, OUTPUT);
pinMode(exit_led_green, OUTPUT);
pinMode(exit_led_red, OUTPUT);

digitalWrite(eisodos_led_red, HIGH);
digitalWrite(exit_led_red, HIGH);
```

- Στην συνάρτηση loop αν είναι να εισέλθει/εξέλθει κάποιο όχημα, το φανάρι γίνεται πράσινο και αφού εισέλθει/εξέλθει το όχημα, το φανάρι γίνεται κόκκινο.

```
digitalWrite(exit_led_red, LOW);
digitalWrite(exit_led_green, HIGH);

...

digitalWrite(exit_led_red, HIGH);
digitalWrite(exit_led_green, LOW);
```

2.1.3 FSR (x2)

Επιπλέον κάτω από κάθε μπάρα τοποθετήθηκε ένας αισθητήρας δύναμης, προκειμένου η μπάρα να παραμένει ανοιχτή όσο βρίσκεται όχημα κάτω από αυτήν. Ο αισθητήρας αυτός χρησιμοποιήθηκε μόνο για το demo, στην πραγματικότητα θα μπορούσε να χρησιμοποιηθεί κάποιος αισθητήρας απόστασης ή κάποιος κατάλληλος αισθητήρας δύναμης/πίεσης.

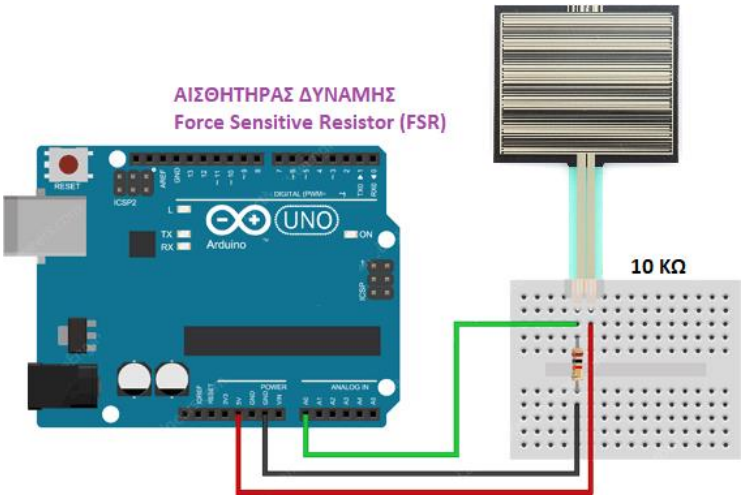


Figure 9 Συνδεσμολογία FSR

Ορισμένα τμήματα του κώδικα για τον FSR:

- Αρχικά ορίζονται τα pins που θα συνδεθούν τα 2 FSRs

```
int input_sensor = A2;
int exit_sensor = A1;
```

- Στην συνάρτηση loop διαβάζεται το σήμα της θύρας, όπου είναι συνδεδεμένος ο FSR. Όσο αυτό είναι μεγαλύτερο του 200, η μπάρα παραμένει ανοιχτή, αν γίνει μικρότερο του 200, η μπάρα κλείνει και το φανάρι γίνεται από πράσινο, κόκκινο.

```
while(analogRead(input_sensor) > 200){
      delay(1);
   }
 if(analogRead(input_sensor) < 200){
      s.write(0);


      digitalWrite(eisodos_led_red, HIGH);
      digitalWrite(eisodos_led_green, LOW);

 }
```

2.1.4 Touch sensor(x2)

Οι touch sensors χρησιμοποιηθήκαν ο ένας στην είσοδο του parking, και ο άλλος στην έξοδο, προκειμένου όταν ο οδηγός του αυτοκινήτου επιθυμεί να εισέλθει ή να εξέλθει από το χώρο στάθμευσης, να τους πιέζει για να ανοίξουν οι αντίστοιχες μπάρες και να γίνει πράσινο το αντίστοιχο φανάρι.

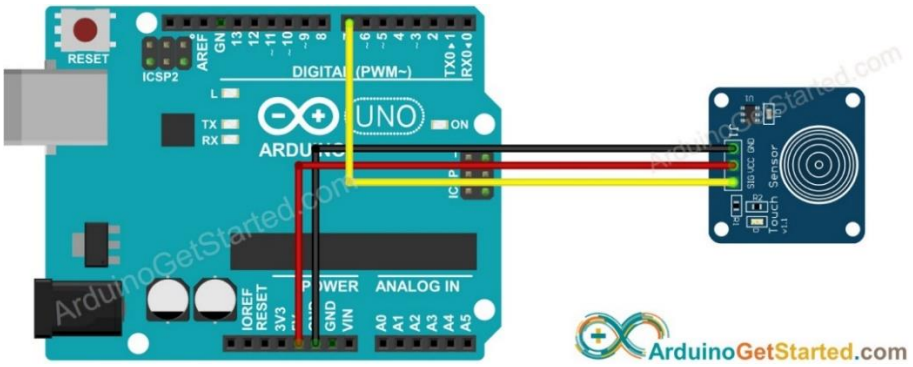


Figure 10 Συνδεσμολογία Touch Sensor

Ορισμένα τμήματα του κώδικα για τον Touch Sensor:

- Αρχικά ορίζεται το pin στο οποίο θα συνδεθεί ο ακροδέκτης sig του αισθητήρα.

```
int Touchsensor = 4;
```

- Στην συνάρτηση setup το pin αυτό ορίζεται ως input.

```
pinMode(Touchsensor, INPUT);
```

- Στην loop διαβάζεται η τιμή του σήματος. Αν είναι 1 και υπάρχει διαθέσιμη θέση στάθμευσης, μειώνονται οι συνολικές θέσεις στάθμευσης κατά 1, ανοίγει η μπάρα και το φανάρι γίνεται πράσινο προκειμένου να διέλθει το αυτοκίνητο.

```
if(digitalRead(Touchsensor)){
  plirotita += 1;
  if(plirotita > max_plirotita){
      plirotita = max_plirotita;
      free_slots = 0;


      lcd.clear();
      lcd.setCursor(4,0); //First line
      lcd.print("Parking" );
      Data = free_slots;
      Serial.print("Parking slots: ");
      Serial.println("P1: " + String(free_slots));

      lcd.setCursor(0,1); //First line
      lcd.print("Sorry, no spots");

      digitalWrite(eisodos_led_red, HIGH);
      digitalWrite(eisodos_led_green, LOW);
  }
  else{
```

```
 digitalWrite(eisodos_led_red, LOW);
 digitalWrite(eisodos_led_green, HIGH);

 s.write(90);
 delay(3000);

 lcd.clear();
 lcd.setCursor(1,0); //First line
 lcd.print("Parking slots" );
 free_slots = max_plirotita - plirotita;
 Data = free_slots;
 Serial.print("Parking slots: ");
 Serial.println("P1: " + String(free_slots));

 lcd.setCursor(0,1); //First line
 lcd.print("P1: " + String(free_slots));
   delay(3000);
while(analogRead(input_sensor) > 200){
     delay(1);
   }
 if(analogRead(input_sensor) < 200){
     s.write(0);


     digitalWrite(eisodos_led_red, HIGH);
     digitalWrite(eisodos_led_green, LOW);

 }
```

2.1.5 Flame sensor

Χρησιμοποιήθηκε ένας αισθητήρας φλόγας, προκειμένου αν υπάρξει φωτιά, να ενημερωθεί ο κεντρικός σταθμός (και μετέπειτα η πυροσβεστική), να ηχήσει η σειρήνα, και να ανοίξουν οι μπάρες προκειμένου να εκκενωθεί το parking.

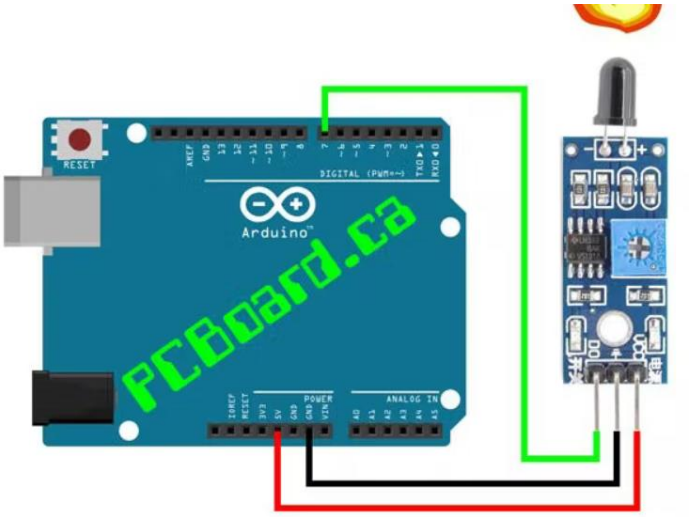


Figure 11 Συνδεσμολογία flame sensor

Ορισμένα τμήματα του κώδικα για τον flame Sensor:

- Μέσα στην συνάρτηση loop διαβάζεται η τιμή του σήματος στην θύρα A3. Αν είναι μικρότερη του 500, ανιχνεύθηκε φωτιά, ανοίγουν οι μπάρες, ηχεί η σειρήνα (buzzer) και στέλνονται τα κατάλληλα δεδομένα στον κεντρικό κόμβο.

```
if( analogRead(A3) < 500){
    digitalWrite(exit_led_green, HIGH);
    lcd.clear();
    lcd.setCursor(5,0);
    lcd.print("Warning");
    lcd.setCursor(0,1);
    lcd.print("Fire Fire Fire!");
    digitalWrite(eisodos_led_red, HIGH);
    digitalWrite(exit_led_green, HIGH);
    digitalWrite(exit_led_red, LOW);
    s1.write(90);
    s.write(90);

    for(int j = 300; j<700; j++)
        {
            tone(BUZZER, j);
            delay(10);
         }
    for(int j = 700; j>300; j--){
      tone(BUZZER, j);
      delay(10);
      }
      noTone(BUZZER);
      Serial.println("Stop");
      Serial.println("Detected Fire!");
      lcd.clear();
      lcd.setCursor(1,0);
      lcd.print("Warining! FIRE!");
      lcd.setCursor(1,1); //First line
      lcd.print("P1: " + String(free_slots));
      Data = 1999;
     }
```

2.1.6 Buzzer

Χρησιμοποιήθηκε ένα buzzer το οποίο με την χρήση κατάλληλων συναρτήσεων λειτουργεί ως σειρήνα σε περίπτωση έκτακτης ανάγκης (φωτιά).

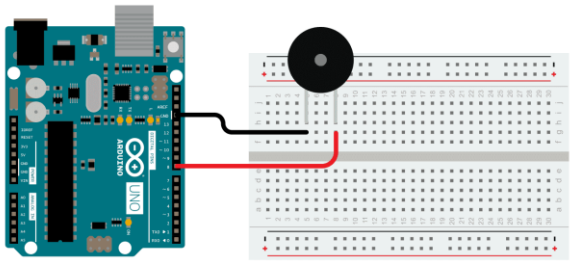


Figure 12 Συνδεσμολογία Buzzer

Ορισμένα τμήματα του κώδικα για το buzzer:

- Ορίζεται αρχικά το pin στο οποίο συνδέεται το buzzer

```
int BUZZER = 10;
```

- Στην setup το παραπάνω pin δηλώνεται ως output

```
pinMode(buzzer, OUTPUT);
```

- Στην loop σε περίπτωση που ανιχνευθεί φωτιά, μέσω του παρακάτω κώδικα ηχεί ως σειρήνα.

```
for(int j = 300; j<700; j++)
    {
        tone(BUZZER, j);
        delay(10);
     }
for(int j = 700; j>300; j--){
  tone(BUZZER, j);
  delay(10);
  }
  noTone(BUZZER);
```

2.1.7 LCD

Στο εξωτερικό του parking έχει τοποθετηθεί μια LCD με I2C στην οποία τυπώνονται οι διαθέσιμες θέσεις που υπάρχουν εντός του parking καθώς και μηνύματα έκτακτης ανάγκης (πχ φωτιά).

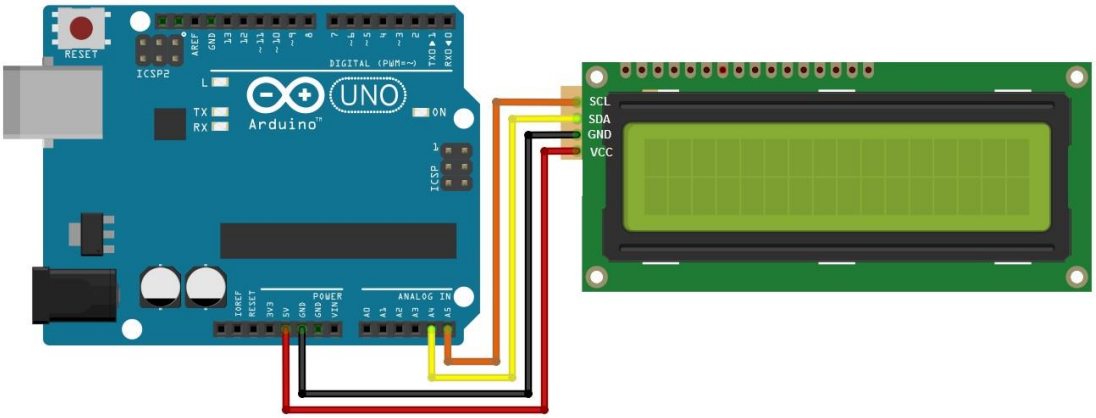


Figure 13 Συνδεσμολογία LCD i2c

Ορισμένα τμήματα του κώδικα για την LCD:

- Αρχικά συμπεριλαμβάνεται η βιβλιοθήκη LiquidCrystal_I2C.h και δημιουργήθηκε το αντικείμενο lcd

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
```

- Στην setup ενεργοποιείται η lcd, ανάβει το backlight, τοποθετείται ο cursor στην επιθυμητή θέση και τυπώνονται διάφορα μηνύματα (ομοίως και στην loop χωρίς να χρειάζεται να γίνει lcd.begin() και lcd.backlight()).

```
  lcd.begin();   // iInit the LCD for 16 chars 2 lines
  lcd.backlight();   // Turn on the backligt (try lcd.noBaklight()
lcd.setCursor(1,1); //First line
lcd.print("P1: " + String(free_slots));
```

2.1.8 Bluetooth- HC05

Επιπλέον χρησιμοποιήθηκε ένα Bluetooth Module HC05 και ένα smartphone (app Arduino bluetooth) ως τηλεχειριστήριο. Το τηλεχειριστήριο αυτό το κατέχει ο υπεύθυνος του parking προκειμένου να εκτελεί ορισμένες απαραίτητες λειτουργίες.

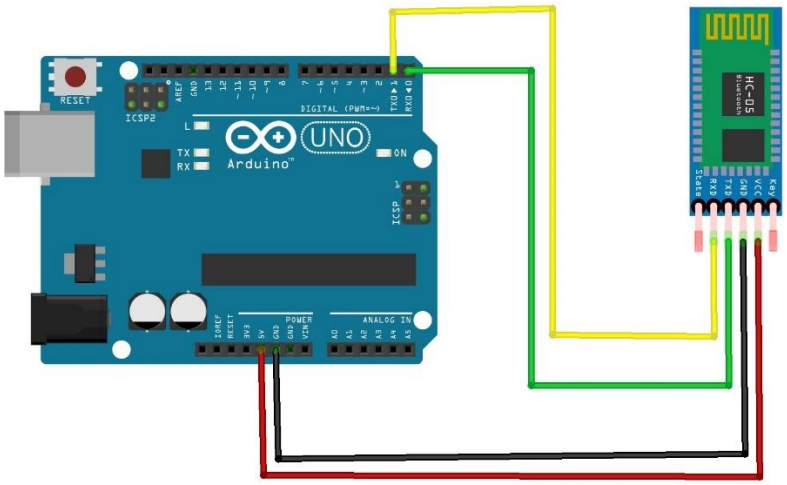


Figure 14 Συνδεσμολογία HC-05

Ορισμένα τμήματα του κώδικα για το HC05:

- Αρχικά συμπεριλαμβάνεται η βιβλιοθήκη SoftwareSerial.h και αρχικοποιείται ο χαρακτήρας incoming value στο 0.

```
#include <SoftwareSerial.h>

char Incoming_value = 0;
```

- Στην loop, αν η σειριακή είναι διαθέσιμη, ανατίθεται στην incoming_value η τιμή που διαβάζεται μέσω της σειριακής (αυτή που στέλνεται από το κινητό-τηλεχειριστήριο bluetooth). Καθώς η τιμή της incoming_value είναι 9 μπαίνει στο Control Mode, περιμένει για 4s. Στην συνέχεια:
    - ✓ Αν πληκτρολογηθεί το 0, τότε ζητάει να εισαχθεί το συνολικό πλήθος των θέσεων στάθμευσης που διαθέτει το parking.
    - ✓ Αν πληκτρολογηθεί το 1, τότε μειώνει κατά 1 τις ελεύθερες θέσεις στάθμευσης στο parking (πχ σε περίπτωση που χρειαστεί να γίνουν κάποιες εργασίες σε αυτήν την θέση)
    - ✓ Αν πληκτρολογηθεί το 2, τότε αυξάνει κατά 1 τις ελεύθερες θέσεις στάθμευσης στο parking (σε περίπτωση που έχουν τελειώσει οι εργασίες της παραπάνω περίπτωσης)

- ✓ Αν πληκτρολογηθεί το 3, τότε ελευθερώνει όλες τις θέσεις στάθμευσης, ώστε η διαθεσιμότητα που φαίνεται στις LCD να είναι η μέγιστη.
- ✓ Αν πληκτρολογηθεί το 4, τότε η διαθεσιμότητα θέσεων στάθμευσης γίνεται μηδενική.
- ✓ Αν πληκτρολογηθεί το 5, τότε ανοίγουν οι μπάρες προκειμένου να εισέλθει κάποιο όχημα του διαχειριστή/υπευθύνου του parking ή κάποιο ειδικό όχημα.
- ✓ Αν πληκτρολογηθεί το 6 κλείνουν οι μπάρες.

```
if(Serial.available() > 0)
{
  Incoming_value = Serial.read();
  Serial.print(Incoming_value);
  Serial.print("\n");

  while(Incoming_value == '9'){

    Serial.println("Contol Mode");
    delay(4000);
    Incoming_value = Serial.read();
    if(Incoming_value == '0'){
      Serial.println("max_plirotita = ???");
      max_plirotita = int(Serial.read()) - 48;
      Serial.println(String(max_plirotita));
      }
    else if(Incoming_value == '1'){
      plirotita += 1;
      Serial.println("Parking P1: " + String(free_slots));
      lcd.clear();
      lcd.setCursor(1,0); //First line
      lcd.print("Parking slots" );
      free_slots = max_plirotita - plirotita;
      Data = free_slots;
      Serial.print("Parking slots: ");
      Serial.println("P1: " + String(free_slots));
      lcd.setCursor(1,1); //First line
      lcd.print("P1: " + String(free_slots));
      }
    else if(Incoming_value == '2'){
      lcd.clear();
      lcd.setCursor(1,0); //First line
```

```
  lcd.print("Parking slots" );
  plirotita -= 1;
  free_slots = max_plirotita - plirotita;
  Data = free_slots;
  Serial.print("Parking slots: ");
  Serial.println("P1: " + String(free_slots));
  lcd.setCursor(1,1); //First line
  lcd.print("P1: " + String(free_slots));
  }
else if(Incoming_value == '3'){
  plirotita = 0;
  lcd.clear();
  lcd.setCursor(1,0); //First line
  lcd.print("Parking slots" );
  free_slots = max_plirotita - plirotita;
  Data = free_slots;
  Serial.print("Parking slots: ");
  Serial.println("P1: " + String(free_slots));
  lcd.setCursor(1,1); //First line
  lcd.print("P1: " + String(free_slots));

  }
else if(Incoming_value == '4'){
  plirotita = max_plirotita;
  lcd.clear();
  lcd.setCursor(1,0); //First line
  lcd.print("Parking slots" );
  free_slots = max_plirotita - plirotita;
  Data = free_slots;
  Serial.print("Parking slots: ");
  Serial.println("P1: " + String(free_slots));



     lcd.setCursor(0,1); //First line
     lcd.print("Sorry, no spots");
     }
   else if(Incoming_value == '5'){
      s.write(90);
      s1.write(90);
     }
   else if(Incoming_value == '6'){
      s.write(0);
      s1.write(0);
      digitalWrite(exit_led_red, HIGH);
      digitalWrite(exit_led_green, LOW);
     }
  }
}
```

2.1.9 Σειριακή επικοινωνία

Επιπλέον γίνεται χρήση της σειριακής επικοινωνίας, για να τυπώνονται στην οθόνη του υπολογιστή που … αν υπάρχει (**δεν** χρειάζεται να υπάρχει απαραίτητα), στο parking διάφορα μηνύματα για την διαθεσιμότητα των θέσεων κλπ.

- Στην setup γίνεται η ενεργοποίηση της σειριακής επικοινωνίας.

```
Serial.begin(9600);
```

- Έπειτα τόσο στην setup όσο και στην loop μέσω της εντολής Serial.print(); τυπώνονται στην οθόνη διάφορα μηνύματα.

```
plirotita = 0;
free_slots = max_plirotita - plirotita;
Serial.print("Parking slots: ");
Serial.println("P1: " + String(free_slots));
```

2.1.10 Επεξήγηση κώδικα πομπού

- Αρχικά συμπεριλαμβάνονται οι βιβλιοθήκες RF22.h και RF22Router.h, και αρχικοποιείται η διεύθυνση του πομπού, καθώς και η διεύθυνση προορισμού του δέκτη.

```
#include <RF22.h>
#include <RF22Router.h>
#define MY_ADDRESS 12
#define DESTINATION_ADDRESS 5
RF22Router rf22(MY_ADDRESS);
```

- Στην setup δηλώνονται τα στοιχεία της ασύρματης κάρτας.

  Συχνότητα: 431 MHz, Ισχύ εκπομπής: 20dBm, Διαμόρφωση: GFSK

```
if (!rf22.init())
  Serial.println("RF22 init failed");
// Defaults after init are 434.0MHz, 0.05MHz AFC pull-in, modulation FSK_Rb2_4Fd36
if (!rf22.setFrequency(431.0))
  Serial.println("setFrequency Fail");
rf22.setTxPower(RF22_TXPOW_20DBM);
//1,2,5,8,11,14,17,20 DBM
//rf22.setModemConfig(RF22::OOK_Rb40Bw335  );
rf22.setModemConfig(RF22::GFSK_Rb125Fd125);
//modulation

// Manually define the routes for this network
rf22.addRouteTo(DESTINATION_ADDRESS, DESTINATION_ADDRESS);
randomSeed(Data); //(μία μόνο φορά μέσα στην setup)
```

- Στην loop για την αποστολή των δεδομένων χρησιμοποιήθηκε ένα πακέτο που είτε θα είναι η διαθεσιμότητα του parking, είτε κάποιος 4ψήφιος κωδικός προβλήματος (πχ 1999-> όταν ανιχνευθεί φωτιά).

```
    Data = free_slots;
}
```

ή

```
lcd.print("Warining! FIRE!");
lcd.setCursor(1,1); //First line
lcd.print("P1: " + String(free_slots));
Data = 1999;
```

```
memset(data_read, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
memset(data_send, '\0', RF22_ROUTER_MAX_MESSAGE_LEN);
sprintf(data_read, "%d", Data);
data_read[RF22_ROUTER_MAX_MESSAGE_LEN - 1] = '\0';
memcpy(data_send, data_read, RF22_ROUTER_MAX_MESSAGE_LEN);
successful_packet = 0;

while (!successful_packet)
{

  if (rf22.sendtoWait(data_send, sizeof(data_send), DESTINATION_ADDRESS) != RF22_ROUTER_ERROR_NONE)
  {
    Serial.println("sendtoWait failed");
    randNumber=random(200,max_delay);
    Serial.println(randNumber);
    delay(randNumber);
  }
  else
  {
    successful_packet = 1;
    counter = counter + 1;
    Serial.println("sendtoWait Succesful");
  }
  Serial.println("Data = " + String(Data));
}
```

2.2 Σταθμός Βάσης-Receiver

2.2.1 LCD

Χρησιμοποιήθηκε μια LCD (χωρίς i2c πρωτόκολλο) στην οποία θα τυπώνονται μηνύματα πληροφόρησης των οδηγών για τις διαθέσιμες θέσεις στάθμευσης που υπάρχουν σε κάθε parking.

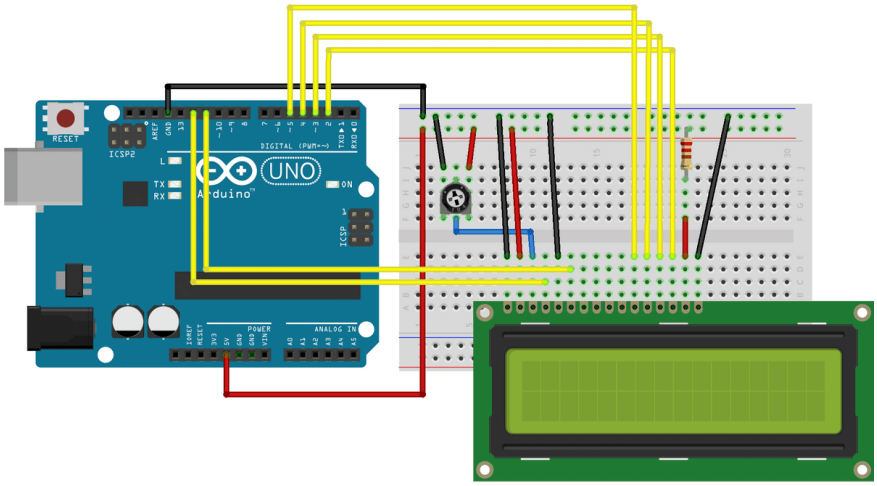


Figure 15 Συνδεσμολογία LCD

Ορισμένα τμήματα του κώδικα για την LCD (χωρίς I2C):

- Αρχικά συμπεριλαμβάνεται η βιβλιοθήκη LiquidCrystal_I2C.h στην συνέχεια ορίζουμε σε ποια pins έχει συνδεθεί.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 8);
```

- Στην setup ενεργοποιείται η lcd, τοποθετείται ο cursor στην επιθυμητή θέση και τυπώνονται διάφορα μηνύματα (ομοίως και στην loop, χωρίς να χρειάζεται να γίνει lcd.begin()).

```
lcd.begin(16, 2);   // Init the LCD for 16 chars 2 lines
lcd.setCursor(0,0); //First line of lcd

lcd.print("Parking slots" );
lcd.setCursor(0,1); //Second line of lcd
lcd.print("P1: " + String(free_slots_p1) + "  " + "P2: " + String(free_slots_p2));
```

2.2.2 Σειριακή επικοινωνία

Γίνεται χρήση της σειριακής επικοινωνίας, για να τυπώνονται στην οθόνη του κεντρικού υπολογιστή διάφορα μηνύματα για την διαθεσιμότητα των θέσεων στάθμευσης καθώς και πληροφορίες για το αν υπάρχει κάποιο πρόβλημα σε κάποιο parking (πχ φωτιά).

- Στην setup γίνεται η ενεργοποίηση της σειριακής επικοινωνίας.

```
Serial.begin(9600);
```

- Έπειτα τόσο στην setup όσο και στην loop μέσω της εντολής Serial.print() τυπώνονται στην οθόνη διάφορα μηνύματα.

```
else if(from == NODE_ADDRESS_2){
  free_slots_p2 = (received_value);

  if(received_value == 1999){
    free_slots_p2 = "0";
    Serial.println("Call 199");
    Serial.println("Call 199");
    Serial.println("Call 199");
    }
   else{
    Serial.print("P2: ");
    Serial.println(free_slots_p2);
    }
  }
```

2.2.3 Επεξήγηση κώδικα δέκτη

- Αρχικά συμπεριλαμβάνονται οι βιβλιοθήκες RF22.h και RF22Router.h, και αρχικοποιείται η διεύθυνση του δέκτη, καθώς και οι διευθύνσεις των πομπών.

```
#include <RF22.h>
#include <RF22Router.h>
#define MY_ADDRESS 5
#define NODE_ADDRESS_1 12
#define NODE_ADDRESS_2 13
RF22Router rf22(MY_ADDRESS);
```

- Στην setup δηλώνονται τα στοιχεία της ασύρματης κάρτας, όμοια με τον πομπό.

  Συχνότητα: 431 MHz, Ισχύ εκπομπής: 20dBm, Διαμόρφωση: GFSK

```
if (!rf22.init())
  Serial.println("RF22 init failed");
// Defaults after init are 434.0MHz, 0.05MHz AFC pull-in, modulation FSK_Rb2_4Fd36
if (!rf22.setFrequency(431.0))
  Serial.println("setFrequency Fail");
rf22.setTxPower(RF22_TXPOW_20DBM);
//1,2,5,8,11,14,17,20 DBM
rf22.setModemConfig(RF22::GFSK_Rb125Fd125);
//modulation

// Manually define the routes for this network
rf22.addRouteTo(NODE_ADDRESS_1, NODE_ADDRESS_1);
rf22.addRouteTo(NODE_ADDRESS_2, NODE_ADDRESS_2);
```

- Στην loop του δέκτη γίνεται η λήψη του πακέτου. Για την αποστολή των δεδομένων χρησιμοποιήθηκε ένα πακέτο από κάθε πομπό που είτε θα είναι η διαθεσιμότητα του parking, είτε κάποιος 4ψήφιος κωδικός προβλήματος (πχ 1999-> όταν ανιχνευθεί φωτιά).

```
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;

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);
  received_value = atoi((char*)incoming);
  Serial.println(received_value);
//   delay(1000);
 }
```

Ο διαχωρισμός των 2 ειδών δεδομένων στον δέκτη φαίνεται παρακάτω:

```
else if(from == NODE_ADDRESS_2){
  free_slots_p2 = (received_value);

  if(received_value == 1999){
    free_slots_p2 = "0";
    Serial.println("Call 199");
    Serial.println("Call 199");
    Serial.println("Call 199");
    }
   else{
    Serial.print("P2: ");
    Serial.println(free_slots_p2);
    }
  }

lcd.setCursor(0,1); //First line
lcd.print("P1: " + (free_slots_p1) + "   " + "P2: " + (free_slots_p2));
```

2.3 Τοπική χρήση ενός Arduino και πιθανή προέκταση

Σε ένα από τα δύο parking χρησιμοποιήθηκε ένα Arduino χωρίς ασύρματη κάρτα και κεραία, με πολλούς αισθητήρες απόστασης προκειμένου να πληροφορείται ο οδηγός εντός του parking μέσω ειδικών φωτεινών επιγραφών LED για το πού είναι η πρώτη ελεύθερη θέση στάθμευσης χωρίς να χρειάζεται να ψάχνει μέσα στο parking. Σε αυτό το σύστημα μπορεί μελλοντικά να προστεθεί και ασύρματη κάρτα με

κεραία αν το parking είναι ένα πραγματικό parking με πολλές θέσεις (και όχι 6 θέσεων που είναι το demo μας, οπότε οι 6 αισθητήρες που είναι τοποθετημένοι σε κάθε θέση συνδέθηκαν σε ένα μόνο arduino) είτε είναι ανοιχτό parking και δεν έχει μπάρες στην είσοδο/έξοδο προκειμένου να γίνεται γνωστό πόσες θέσεις είναι κενές και που είναι αυτές οι θέσεις.

2.3.1 HC-SR04 Ultrasonic Ranging Sensor (x6)

Σε κάθε θέση στάθμευσης (6 in Demo) είναι τοποθετημένος ένας αισθητήρας HC-SR04 προκειμένου να γνωστοποιείται ποιες θέσεις στάθμευσης είναι κενές.

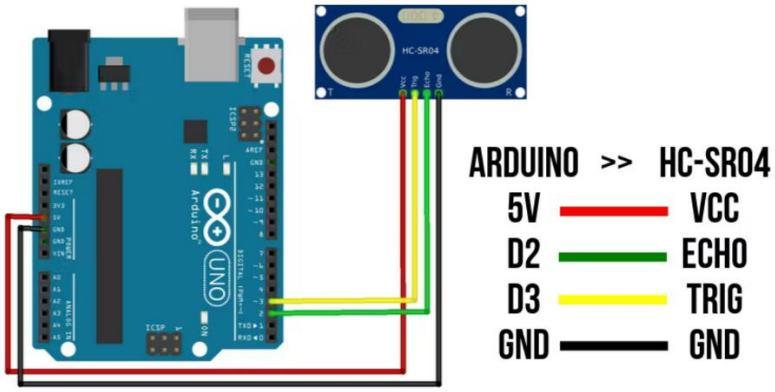


Figure 16 Συνδεσμολογία HC-SR04

Ορισμένα τμήματα του κώδικα για τον HC-SR04:

- Αρχικά δηλώνονται τα pin στα οποία είναι συνδεδεμένοι οι ακροδέκτες echo και trig, και δηλώνονται οι μεταβλητές duration, distance.

```
int echoPin1 = 10;
int trigPin1 = 9;
long duration1, distance1;
```

- Στην setup το pin που είναι συνδεδεμένος ο ακροδέκτης echo δηλώνεται input, ενώ αυτό που είναι συνδεδεμένος ο trig δηλώνεται ως output.

```
pinMode(echoPin1, INPUT);
pinMode(trigPin1, OUTPUT);
```

- Στην loop για να υπολογιστεί η απόσταση ακολουθείται η παρακάτω διαδικασία.

```
digitalWrite(trigPin1, LOW);
delayMicroseconds(2);
digitalWrite(trigPin1, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin1, LOW);
duration1 = pulseIn(echoPin1, HIGH);
distance1 = duration1 / 58.2;
```

2.3.2 Led matrix 8x8

Επιπλέον χρησιμοποιήθηκε ένας φωτεινός πίνακας led προκειμένου να πληροφορεί τον οδηγό για το που ακριβώς βρίσκεται η πρώτη διαθέσιμη θέση στάθμευσης στο parking.

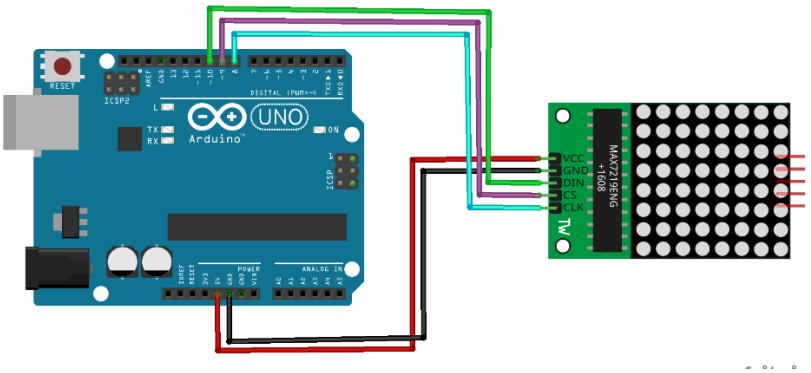


Figure 17 Συνδεσμολογία led matrix 8x8

Ορισμένα τμήματα του κώδικα για τον led matrix 8x8:

- Συμπεριλαμβάνεται η βιβλιοθήκη LedControl.h, ορίζονται σε ποια pins έχουν συνδεθεί οι τρεις ακροδέκτες του matrix.

```
#include <LedControl.h>
int DIN = 13;
int CS = 12;
int CLK = 11;
```

- Μέσω της σελίδας δημιουργούνται οι παρακάτω πίνακες οι οποίοι αντιστοιχούν με το ποια leds του πίνακα θα είναι αναμμένα κάθε φορά.

```
byte N1[8] = {0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x03};
byte N2[8] = {0x00,0x00,0x00,0x0f,0x0f,0x0c,0x0c,0x0c};
byte N3[8] = {0x0f,0x0f,0x0c,0x0c,0x0c,0x0c,0x0c,0x0c};
byte N4[8] = {0xfc,0xfc,0x0c,0x0c,0x0c,0x0c,0x0c,0x0c};
byte N5[8] = {0x3c,0x3c,0x2c,0xec,0xec,0x0c,0x0c,0x0c};
byte N6[8] = {0x3c,0x3c,0x2c,0x2c,0x2c,0x2c,0xec,0xec};
byte keno[8] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
LedControl lc = LedControl(DIN, CLK, CS, 0);
```

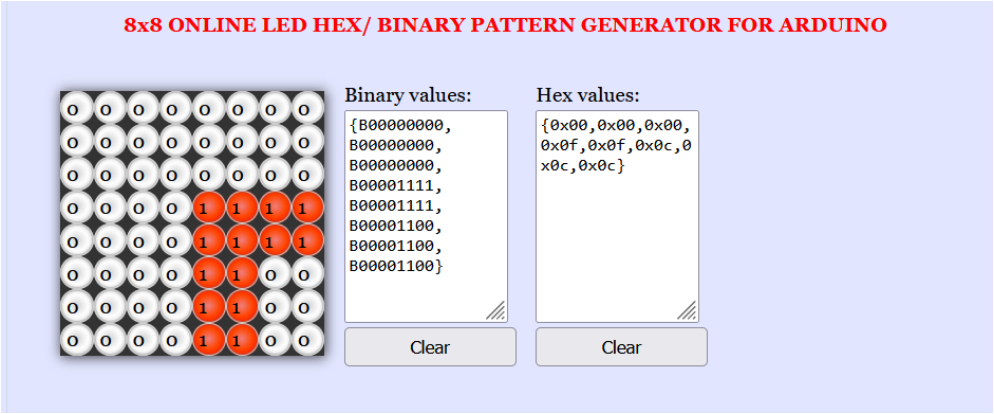


Figure 18 η ένδειξη που ανάβει όταν η 2η θέση του parking είναι η 1η ελεύθερη και οι κωδικοποίηση

- Μέσω του παρακάτω κώδικα, της συνάρτησης printByte, και τις μετρήσεις των αισθητήρων απόστασης κάθε φορά ο πίνακας led θα δείχνει την διαδρομή μέχρι την πρώτη διαθέσιμη θέση στάθμευσης.

```
 if(distance1 > 6){
   printByte(N1);
   Num[0] = 0;
   }
 else if(distance2 > 6){
   printByte(N2);
   Num[1] = 0;
   }
 else if(distance3 > 6){
   printByte(N3);
   Num[2] = 0;
   }
 else if(distance4 > 6){
   printByte(N4);
   Num[3] = 0;
   }
 else if(distance5 > 6){
   printByte(N5);
   Num[4] = 0;
   }
   else if(distance6 > 6){
   printByte(N6);
   Num[5] = 0;
   }
  else{
       printByte(keno);

   }

void printByte(byte character [])
{
    int i = 0;
    for(i=0;i<8;i++)
    {
       lc.setRow(0,i,character[i]);
    }
}
```

2.4 Θεωρητική Ανάλυση Δικτύου

2.4.1 Ανάλυση κριτηρίων χωρητικότητας-Πόσα κανάλια χρειάζονται

Θεωρήθηκε ότι ο αριθμός των parking που συμμετέχουν στην εφαρμογή είναι 20, ο ρυθμός παραγωγής πακέτων κάθε parking-κόμβου είναι 10 πακέτα/sec. Η διάρκεια κάθε πακέτου είναι 3ms.

Χρησιμοποιώντας το πρωτόκολλο επικοινωνίας ALOHA ο απαραίτητος αριθμός καναλιών είναι:

$$N = \frac{20 \cdot \left(\frac{10\ \dot{\pi}\alpha\kappa\acute{\epsilon}\tau\alpha}{1\ sec}\right) \cdot 3ms}{0.186} = \frac{20 \cdot \left(\frac{10\ \dot{\pi}\alpha\kappa\acute{\epsilon}\tau\alpha}{1sec}\right) \cdot 0.003s}{0.186} \approx 4$$

2.4.2 Ανάλυση κριτηρίων κάλυψης

Σκοπός είναι να επιλεχτεί ο ελάχιστος αριθμός σταθμών βάσης, οι οποίοι πρέπει να τοποθετηθούν κατάλληλα για την δεδομένη γεωμετρία.

Χρήσιμα δεδομένα για τους υπολογισμούς

- ✓ Αριθμός Parking – Κόμβων: 20
- ✓ Ρυθμός παραγωγής πακέτων ανά κόμβο: 10πακ/sec
- ✓ Μήκος πακέτου: 45 bytes = 45*8 = 360 bits
- ✓ Διάρκεια πακέτου: T = 3ms
- ✓ Μέση καθυστέρηση: $\bar{X}$ = 1500ms
- ✓ Ρυθμός μετάδοσης δικτύου: B = $\frac{\mu\acute{\eta}\kappa o\varsigma\ \pi\alpha\kappa\acute{\epsilon}\tau o\upsilon}{\delta\iota\acute{\alpha}\rho\kappa\epsilon\iota\alpha\ \pi\alpha\kappa\acute{\epsilon}\tau o\upsilon} \approx$ 120 kbps
- ✓ Ισχύς σε απόσταση: 100m -42dBm
- ✓ Διασπορά σ κανονικής κατανομής (διαλείψεις): 10 dB
- ✓ Απόσβεση ισχύος συναρτήσει της 2^{ης} δύναμης της απόστασης
- ✓ Επιτυχής κάλυψη όλων των χρηστών του δικτύου για το 95% του χρόνου
- ✓ N_{max} = 30 κανάλια ανά σταθμό.

$$S = \lambda \cdot T = 20 \cdot 10 \cdot 0.003 = 0.6$$

Δηλαδή, αν υπήρχε τέλεια κατανομή της κίνησης στο πεδίου του χρόνου, χρειάζεται ένα συχνοτικό κανάλι.

Για το συγκεκριμένο πρωτόκολλο, από τους πίνακες, η μέγιστη εξυπηρετούμενη κίνηση S ανά συχνοτικό κανάλι που εξασφαλίζει τη μέγιστη καθυστέρηση των 300 ms είναι 0.108. Άρα, χρειάζονται 4/0.108=37.03, δηλαδή τουλάχιστον 38 συχνοτικά κανάλια. Αφού κάθε σταθμός μπορεί να έχει μέχρι 30, θα τοποθετηθούν τουλάχιστον 2 σταθμοί.

$$P_{min} = -130dBm + 10\log_{10} B$$

$$P_{min} = -130dBm + 10 \cdot \log_{10} 120 \cdot 10^3$$

$$P_{min} = -130 + 50.7918 = -79.208\ dBm$$

Για το συγκεκριμένο ρυθμό μετάδοσης, βρέθηκε:

$$P_{sensitivity} = -79.208\ dBm$$

Ωστόσο, λόγω διαλείψεων, απαιτείται περισσότερη ισχύς στον δέκτη. Αντικαθίσταται q = 0.95→ $2q - 1 = 0.9 = \text{erf}\left(\frac{\gamma}{\sqrt{2}\sigma}\right)$

$$\frac{\gamma}{\sqrt{2}\sigma} = 1.15 \rightarrow \gamma = 1.15 \cdot \sqrt{2} \cdot \sigma = 16.26\ dB$$

Επομένως η ελάχιστη ισχύς στον δέκτη πρέπει να είναι

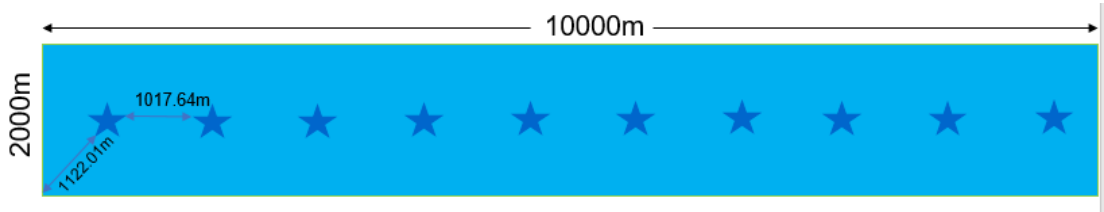
$$P_{\ min} = P_{sensitivity} + \gamma = -62.948\ dBm$$

Λύνοντας ως προς r_{max}

$$r_{max} = \sqrt{\frac{10^{-4.2}}{10^{-6.3}}\,100^2 m} = 1122.01\ m$$

Έστω ότι η πόλη έχει την γεωμετρία ορθογωνίου με εμβαδόν 20km^2 και πλευρές 10km, 2km αντίστοιχα. Υπολογίζεται η θέση του 1^{ου} σταθμού (από αριστερά), έτσι ώστε να καλύπτεται το αριστερό άκρο της πόλης.

$$d = \sqrt{1122.01^2 - 1000^2} = 508.82m$$



Σταθμοί Βάσης του Δικτύου

Τα κανάλια που πρέπει να πάρει κάθε σταθμός είναι:

$$N = \frac{\text{επιφάνεια κάλυψης σταθμού}}{\text{συνολική επιφάνεια}} \cdot \text{απαιτούμενο πλήθος καναλιών}$$

$$N = \frac{2035280}{20 \cdot 10^6} 38 = 3.86 \approx 4 \text{ κανάλια}$$

3 Ψευδοκώδικες

3.1 Tx1 and Tx2

Include libraries

Define nodes of network

Initialize object of RF22

Initialize variables of LCD, sensors, servos and LEDs

Define pins of sensors

Setup():

Set input and output pins of sensors and LEDs

Initialize servo

Open serial port

Initialize LCD

Setup the appropriate messages, for the free parking slots, to be printed on the LCD

Set parking RED LEDs to be ON at the start

Define RF22 characteristics

Add all the routes

Create the random seed

Loop:

if(Serial communication is available){

Read incoming value from Arduino bluetooth(phone application)

Print incoming value on serial communication

while(incoming value is 9){

Print message "Control mode"

Delay 4 seconds

Read incoming value from Arduino bluetooth(phone application)

if(incoming value is 0){

Serial communication will ask the user how many are the total parking slots

Decode and print max parking slots on serial communication(48 is character '0' in ASCII)

}

else if(incoming value is 1){ //car just entered parking 1

Increase number of reserved parking slots by 1

Print free parking slots on serial communication

Print the first line of parking LCD with message "Parking slots"

Calculate and store number of free slots in variable data

Print parking 1 free slots on serial communication

Print the second line of LCD with message "P1" + number of free parking slots for parking 1

}

else if(incoming value is 2){ //car just exited parking 1

Print the first line of parking LCD with message "Parking slots"

Decrease number of reserved parking slots by 1

Calculate and store number of free slots in variable data

Print free parking slots on serial communication

Print parking 1 free slots on serial communication

Print the second line of LCD with message "P1" + number of free parking slots for parking 1

}

else if(incoming value is 3){ //no cars in parking

Set number of reserved parking slots to 0

Print the first line of parking LCD with message "Parking slots"

Calculate and store number of free slots in variable data

```
                    Print free parking slots on serial communication
                    Print parking 1 free slots on serial communication
                    Print the second line of LCD with message "P1" + number of
free parking slots for parking 1
                }
                else if(incoming value is 4){ //all parking slots are reserved
                    Set number of reserved parking slots to maximum
                    Print the first line of parking LCD with message "Parking
slots"
                    Calculate and store number of free slots in variable data
                    Print free parking slots on serial communication
                    Print parking 1 free slots on serial communication
                    Print the second line of LCD with message "Sorry, no (free)
spots"
                }
                else if(incoming value is 5){
                    Raise both parking bars
                }
                else if(incoming value is 6){
                    Lower both parking bars
                }
            }

        }

        if(Fire sensor analog value is lower than 500){
            Turn ON parking's exit green LEDs
            Print warning message of fire on LCD
            Turn ON parking's entrance red LEDs
            Turn ON parking's exit green LEDs
            Turn OFF parking's exit red LEDs
            Raise both parking bars
            Turn ON fire alarm(buzzer)
            Turn OFF fire alarm(buzzer) after some time.
```

Print warning message of fire on serial communication

Print warning message of fire on LCD and empty all free slots

} Store 1999 in variable data

if(touch sensor is pressed){

Increase number of reserved parking slots by 1

if(reserved parking slots exceed maximum number of parking slots){

Set number of free parking slots to 0

Print free parking slots messages on both LCD and serial communication

Turn ON and OFF red and green LEDs of parking entrance respectively.

}

else{

Turn ON and OFF green and red LEDs of parking entrance respectively.

Raise parking bar of entrance

Wait 3 seconds

Print the first line of parking LCD with message "Parking slots"

Calculate and store number of free slots in variable data

Print free parking slots on serial communication

Print parking 1 free slots on serial communication

Print the second line of LCD with message "P1" + number of free parking slots for parking 1

Wait 3 seconds

while(FSR sensor is still being pressed){

Wait

}

if(FSR sensor is not being pressed){

Lower entrance bar

Turn ON and OFF red and green LEDs of parking entrance respectively.

}

if(number of reserved parking slots equals the number of maximum parking slots){

```
                                Wait 1 second
                                Set number of free parking slots to 0
                                Print parking 1 free slots on serial communication
                                Print no slots message on parking's LCD
                        }
                }
        }
        else if(parking's exit button is pressed and number of reserved slots is bigger than
0){
                Turn ON and OFF green and red LEDs of parking exit respectively.
                Raise exit bars
                Wait 3 seconds
                Calculate and store number of free slots in variable data
                Print number of free parking slots on both serial communication and parking
1 LCD
                Wait 3 seconds
                while(exit FSR sensor is still being pressed){
                        Wait
                }
                if(exit FSR sensor is not being pressed){
                        Lower exit bars
                        Turn ON and OFF red and green LEDs of parking exit respectively.
                }

        }
        if(number of reserved parking slots is changed or fire code 1999 is about to be
transmitted){
                if(number of free slots equals number of maximum slots){
                        Data = free slots
                }
                Print transmitted value on serial communication
                Load data in one packet
                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);
                }
        }
}
```

Ο Ψευδοκώδικας για τον Tx2 είναι όμοιος με του Tx1

3.2 Rx

Include libraries

Define nodes of network

Initialize an object RF22

Initialize LCD

Make some helper variables and store the values of free slots in each parking

Setup():

- Open the Serial Port
- Define RF22 characteristics
- Define the routes for this network
- Initialize LCD
- Print text of free parking slots for each parking on LCD

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 received_value
        }

        If(transmitter address == NODE_ADDRESS_1){
                Store received value from transmitter to helper variable
                If(received value from transmitter == Fire code){
                        Set parking free slots to 0
                        Print warning message to call the fire brigade in serial
communication
                }
                else{
            Print parking 1 free slots to serial communication
                }
        }

        Else If(transmitter address == NODE_ADDRESS_2){
                Store received value from transmitter to helper variable
         If(received value from transmitter == Fire code){
                        Set parking free slots to 0
                        Print warning message to call the fire brigade in serial
communication
                }
              else{
            Print parking 2 free slots to serial communication
                }

        }
        Print text of free parking slots for each parking on LCD
}
```

3.3 Τοπικό Arduino

Include necessary libraries

Set the pins for all ultrasonic ranging sensors and led matrix

Initialize Bytes of LED array and create an object of this type

Set printByte function to turn ON specific LEDs of the array

setup():

Set pins of all sensors to output/input mode

Begin serial communication

loop():

Send pulse with ultrasonic sensor 1 (parking slot 1)

Calculate the distance between object and ultrasonic sensor 1

Print the distance of ultrasonic sensor 1 on serial communication

Send pulse with ultrasonic sensor 2 (parking slot 2)

Calculate the distance between object and ultrasonic sensor 2

Print the distance of ultrasonic sensor 2 on serial communication

Send pulse with ultrasonic sensor 3 (parking slot 3)

Calculate the distance between object and ultrasonic sensor 3

Print the distance of ultrasonic sensor 3 on serial communication

Send pulse with ultrasonic sensor 4 (parking slot 6)

Calculate the distance between object and ultrasonic sensor 4

Print the distance of ultrasonic sensor 4 on serial communication

Read and print on serial communication the distance of object from laser sensor at parking slot 4

Read and print on serial communication the distance of object from laser sensor at parking slot 5

if(distance1 > 6){//meaning if distance is bigger than a certain value then no car is occupying the parking slot

Set LED array to point to the first parking slot number 1 as it is the closest available

Set current parking slot value to 0//meaning it's not occupied by a car

}

else if(distance2 > 6){//meaning if distance is bigger than a certain value then no car is occupying the parking slot

Set LED array to point to the first parking slot number 2 as it is the closest available

Set current parking slot value to 0//meaning it's not occupied by a car

}

else if(distance3 > 6){//meaning if distance is bigger than a certain value then no car is occupying the parking slot

Set LED array to point to the first parking slot number 3 as it is the closest available

Set current parking slot value to 0//meaning it's not occupied by a car

}

else if(distance4 > 6){//meaning if distance is bigger than a certain value then no car is occupying the parking slot

Set LED array to point to the first parking slot number 4 as it is the closest available

Set current parking slot value to 0//meaning it's not occupied by a car

}

else if(distance5 > 6){//meaning if distance is bigger than a certain value then no car is occupying the parking slot

Set LED array to point to the first parking slot number 5 as it is the closest available

Set current parking slot value to 0//meaning it's not occupied by a car

}

else if(distance6 > 6){//meaning if distance is bigger than a certain value then no car is occupying the parking slot

Set LED array to point to the first parking slot number 6 as it is the closest available

Set current parking slot value to 0//meaning it's not occupied by a car

}

else{//no available slots

Set LED array to blank

}

if(distance1 < 6){//meaning if distance is lower than a certain value then a car is occupying the parking slot

Set current parking slot value to 1//meaning it is occupied by a car

}

else if(distance2 < 6){//meaning if distance is lower than a certain value then a car is occupying the parking slot

Set current parking slot value to 1//meaning it is occupied by a car

}

else if(distance3 < 6){//meaning if distance is lower than a certain value then a car is occupying the parking slot

Set current parking slot value to 1//meaning it is occupied by a car

}

else if(distance4 < 6){//meaning if distance is lower than a certain value then a car is occupying the parking slot

Set current parking slot value to 1//meaning it is occupied by a car

}

else if(distance5 < 6){//meaning if distance is lower than a certain value then a car is occupying the parking slot

Set current parking slot value to 1//meaning it is occupied by a car

}

else if(distance6 < 6){//meaning if distance is lower than a certain value then a car is occupying the parking slot

Set current parking slot value to 1//meaning it is occupied by a car

}

print all parking slot values on serial communication

Put a delay of 2 seconds

4 Μελλοντικές Προεκτάσεις και εναλλακτικές σχεδίασης

Σκοπός της εφαρμογής είναι να παρέχει χρήσιμες πληροφορίες στους οδηγούς των αυτοκινήτων μιας πόλης, προκειμένου εύκολα και γρήγορα να βρίσκουν θέση στάθμευσης.

Η πληροφόρηση σε επίπεδο demo επιταχύνθηκε μέσω οθονών lcd. Σε επόμενο στάδιο, θα μπορούσε να αναπτυχθεί και ένα application για smartphones, ώστε οι οδηγοί μέσω του κινητού τηλεφώνου τους να πληροφορούνται για την διαθεσιμότητα των parking. Η εφαρμογή αυτή θα συλλέγει δεδομένα από μια βάση δεδομένων στην οποία θα συγκεντρώνονται οι πληροφορίες που στέλνονται από κάθε parking.

Ακόμα, με την χρήση GPS/χαρτών οι οδηγοί μέσω της εφαρμογής θα μπορούν να γνωρίζουν την ελάχιστη διαδρομή για τον πιο κοντινό σε αυτούς χώρο στάθμευσης με ελεύθερη θέση, καθώς και το χρόνο που χρειάζονται για να φτάσουν σε αυτόν (το τελευταίο θα μπορούσε να εμφανίζεται και σαν πληροφορία στις lcd στους δρόμους της πόλης).

Επιπλέον, στο μέλλον σε διάφορα σημεία της πόλης θα μπορούσαν να τοποθετηθούν κατάλληλοι αισθητήρες, ώστε μέσω των lcd ή της εφαρμογής οι οδηγοί να πληροφορούνται παράλληλα και για τις καιρικές συνθήκες που επικρατούν σε διάφορες περιοχές της πόλης. Οι αισθητήρες που θα μπορούσαν να χρησιμοποιηθούν για αυτόν τον σκοπό είναι:

- Αισθητήρας Θερμοκρασίας και Υγρασίας DHT11

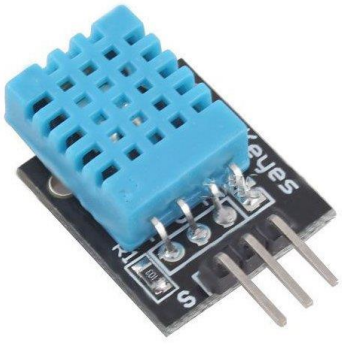


Figure 19 DHT11

- Αισθητήρας βροχής

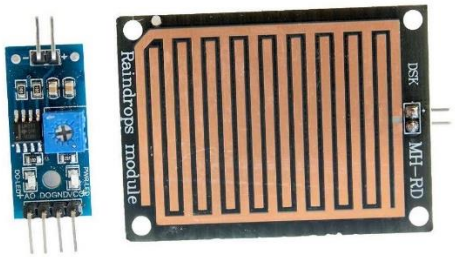


Figure 20 Rain Sensor Module

Όσον αφορά τον μικροελεγκτή, σε επίπεδο demo χρησιμοποιείται ο μικροελεγκτής Arduino uno της AVR. Θα μπορούσε, ωστόσο, να χρησιμοποιηθεί κάποιος άλλος μικροελεγκτής που να διαθέτει μεγαλύτερο αριθμό pins ή και να διαθέτει κάποια επιπλέον χαρακτηριστικά π.χ nucleo stm32fxx, raspberry pi, Arduino mega κλπ. Επιπλέον, για πιο μικρή κατανάλωση ισχύος, κατά την ανάπτυξη του κώδικα θα μπορούσε να χρησιμοποιηθούν interrupts αντί της μεθόδου «polling» που χρησιμοποιήθηκε.

Ο FSR που χρησιμοποιήθηκε δεν συνίσταται πέρα από το επίπεδο demo. Στη θέση του μπορεί να χρησιμοποιηθεί κάποιος αισθητήρας απόστασης ultrasonic ή κάποιος κατάλληλος αισθητήρας δύναμης/πίεσης.

Τέλος, αντί για το πρωτόκολλο επικοινωνίας ALOHA, που χρησιμοποιήθηκε, θα μπορούσε να εφαρμοστεί το slotted- ALOHA που αξιοποιεί το 36.2% του διαθέσιμου καναλιού, έναντι του ALOHA που χρησιμοποιεί το 18.6%, ή μπορεί να χρησιμοποιηθεί και CSMA-NP/1-persistent/p-persistent.

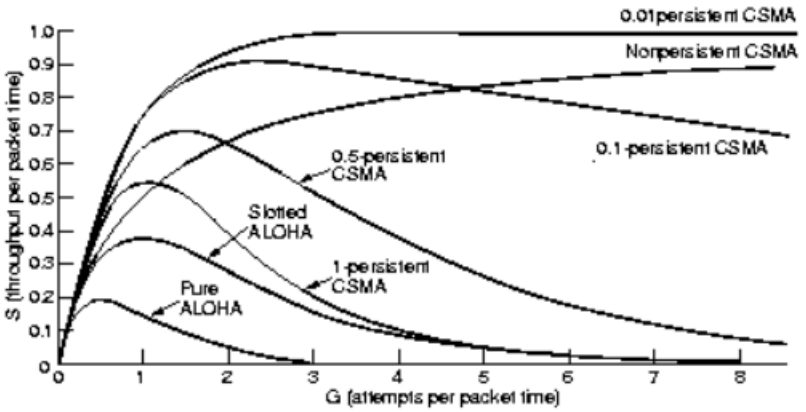


Figure 21 Σύγκριση Πρωτόκολλων Επικοινωνίας

Ακόμη, μπορεί να χρησιμοποιηθεί και το διαδίκτυο (Wi-Fi ή erthrnet) για την αποστολή των δεδομένων, μέσω κατάλληλης ασύρματης κάρτας Wi-Fi.