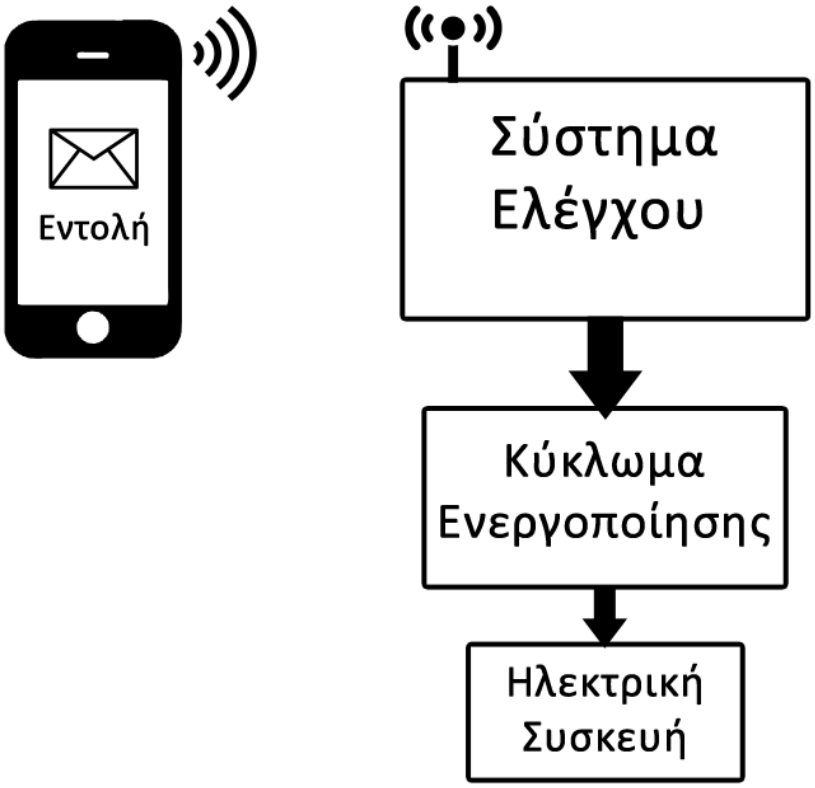
Περιεχόμενα

Εισαγωγή

Σκοπός της εργασίας αυτής είναι η κατασκευή ενός συστήματος ελέγχου ηλεκτρικών συσκευών από απόσταση μέσω εντολών ενεργοποίησης και απενεργοποίησης.

Οι εντολές θα στέλνονται από κινητά τηλέφωνα μέσω γραπτών μηνυμάτων SMS, στο σύστημα ελέγχου, το οποίο θα την αναλύει, θα ελέγχει την εγκυρότητά της και στην συνέχεια την εκτελεί μέσω των κυκλωμάτων ενεργοποίησης.

Η χρησιμότητα ενός τέτοιου συστήματος είναι πολύ μεγάλη, μπορεί να χρησιμοποιηθεί από απλές οικιακές λειτουργίες μέχρι και για αυτοματισμούς σε βιομηχανίες.



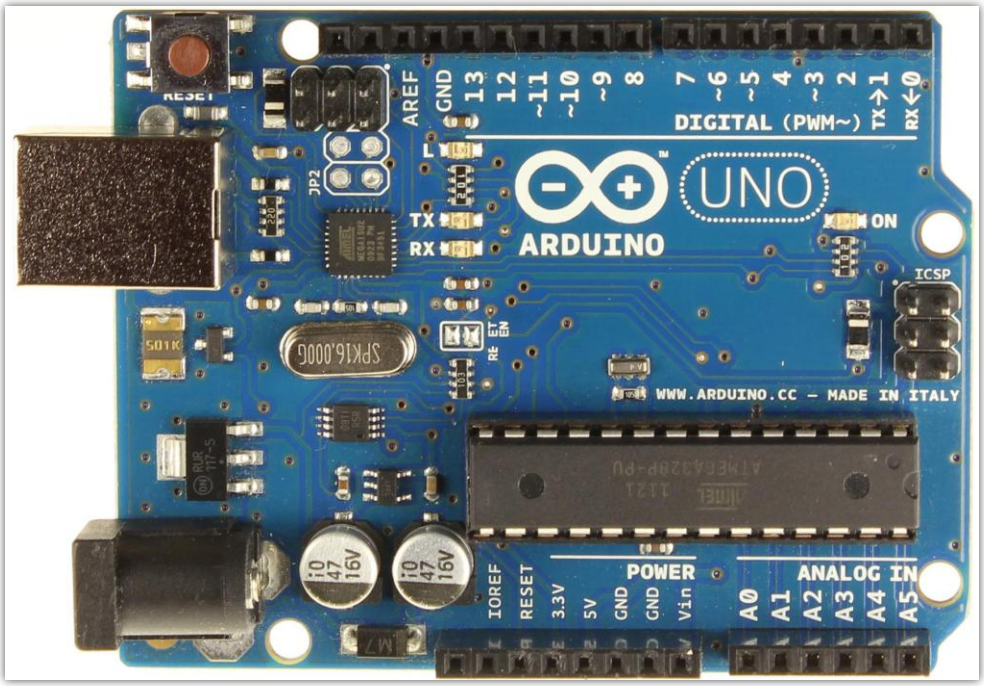
Για την υλοποίηση του συστήματος ελέγχου έγινε η επιλογή της προγραμματιζόμενης πλατφόρμας Arduino, είναι πολύ διαδεδομένη, την επιλέγουν κυρίως λόγο της απλότητας στην κατασκευή πρωτοτύπων και το πλήθος του διαθέσιμου υλικού εκμάθησης που υπάρχει. Για τον λόγω αυτό προτιμούν το Arduino διάφοροι χρήστες από πολύ αρχάριους ως και έμπειρους στον χώρο.

Σε συνδυασμό με την πλατφόρμα Arduino επιλέχθηκε το Arduino GSM Shield, για την υλοποίηση τμήματος της επικοινωνίας με τους χρήστες που θα στέλνουν τις εντολές. Το GSM Shield είναι και αυτό σχεδιασμένο από την ίδια εταιρία, κάτι που θα διευκολύνει πολύ το χειρισμό της. Περισσότερες λεπτομέρειες υπάρχουν στην ενότητα Υλικό.

Τέλος το κύκλωμα ενεργοποίησης θα υλοποιηθεί από ρελέ. Το οποίο θα αναλάβει την διακοπτητική λειτουργία της ηλεκτρικής συσκευής που είναι συνδεδεμένο. Με αυτόν τον τρόπο θα είναι δυνατός ο έλεγχος από την πλατφόρμα Arduino που λειτουργεί σε χαμηλή, οποιασδήποτε συσκευής αρκεί η κατανάλωση ισχύος της να είναι μέσα στα όρια των προδιαγραφών του επιλεγμένου ρελέ.

Υλικά

Arduino Uno Rev3



Το Arduino είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring.

Τεχνικά Χαρακτηριστικά

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6

DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Arduino GSM Shield

Το GSM Shield δίνει την δυνατότητα στη πλατφόρμα Arduino να λειτουργεί σαν ένα GSM κινητό τηλέφωνο δεύτερης γενιάς, κάνει και δέχεται τηλεφωνικές κλίσεις, γραπτά μηνύματα SMS και μπορεί να συνδεθεί στο Internet μέσω GPRS δικτύου στη μέγιστη ταχύτητα των 85.6 kbps.

Το Shield αυτό εφαρμόζει ακριβώς πάνω στο Arduino και είναι συμβατό με τα μοντέλα Uno, Mega και Leonardo.

Για τον προγραμματισμό των λειτουργιών υπάρχει η βιβλιοθήκη GSM Library, η οποία δίνει πρόσβαση σε δυνατότητες του Shield. Το Shield χρησιμοποιεί τις θύρες 2,3 για σειριακή είσοδο/έξοδο καθώς και την θύρα 7 για επανεκκίνηση Αυτές οι τρεις θύρες του Arduino Uno δεν θα είναι διαθέσιμες για το υπόλοιπο της κατασκευής. Η τοποθέτηση μίας κάρτας SIM είναι απαραίτητη για την λειτουργία του Shield.

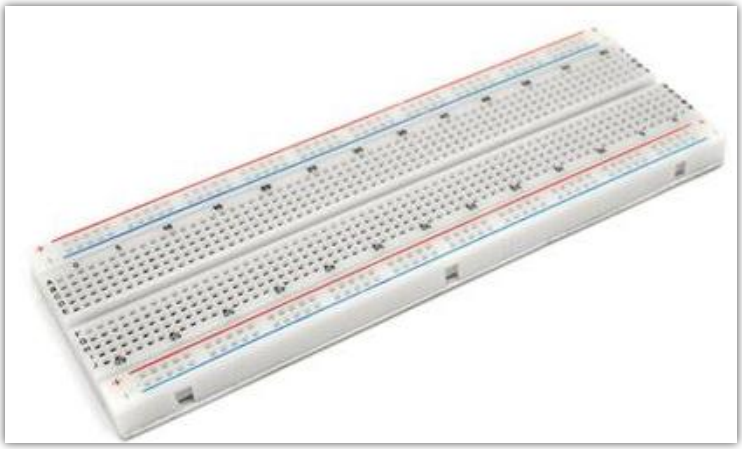
GSM Antenna

Κεραία για την λήψη/εκπομπή σημάτων του GSM Shield στα δίκτυα GSM/GPRS ή 2.4/5.8GHz.

Power Supply 12VDC

Τροφοδοτικό για το Arduino ονομαστικής τάσης 12 V και ρεύματος 2A ώστε να καλύπτει το μέγιστο ρεύμα που χρειάζεται το GSM Shield, όπως αναφέρεται στις προδιαγραφές του.

Breadboard

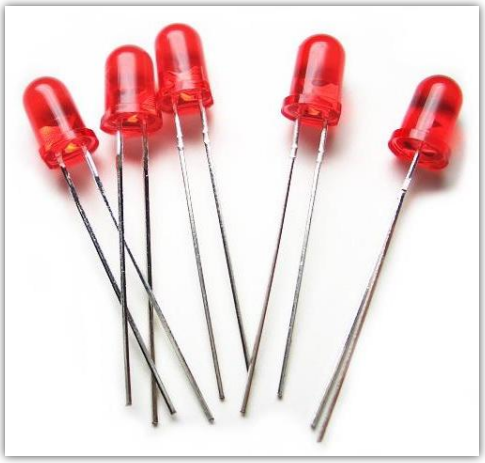


Diode 1N4007

Transistor 546 B

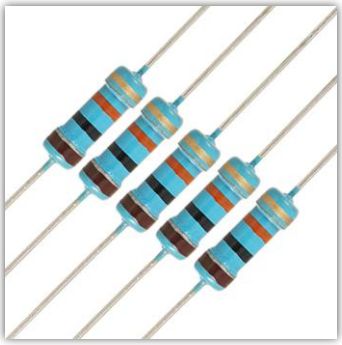
Red LED 5mm

1.8-2.2VDC forward drop
Max current: 20mA



Αντιστάσεις 330 Ω

Αντιστάσεις 10ΚΩ



Relay - NHG 5V JZC-6F(4098)

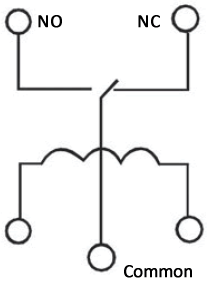
Το ρελέ (relay) είναι ένας ηλεκτρικός διακόπτης που ανοίγει και κλείνει ένα ηλεκτρικό κύκλωμα κάτω από τον έλεγχο ενός άλλου ηλεκτρικού κυκλώματος.

Όταν ηλεκτρικό ρεύμα διαρρέει το πηνίο του ηλεκτρονόμου, το παραγόμενο μαγνητικό πεδίο έλκει έναν οπλισμό που είναι μηχανικά συνδεδεμένος σε μια κινούμενη επαφή. Μόλις το ηλεκτρικό ρεύμα στο πηνίο διακοπεί, ο οπλισμός επιστέφει στη θέση ηρεμίας του.

Έτσι μπορεί και λειτουργεί σαν διακόπτης ελεγχόμενος από ηλεκτρικό ρεύμα.

Χαρακτηριστικά:

Coil voltage	5 - 48 V
Coil power	<= 0.2W
Contact capacity	2A/125VAC 2A/30VDC
Resistance	70Ω



Οι 5 ακροδέκτες του ρελέ

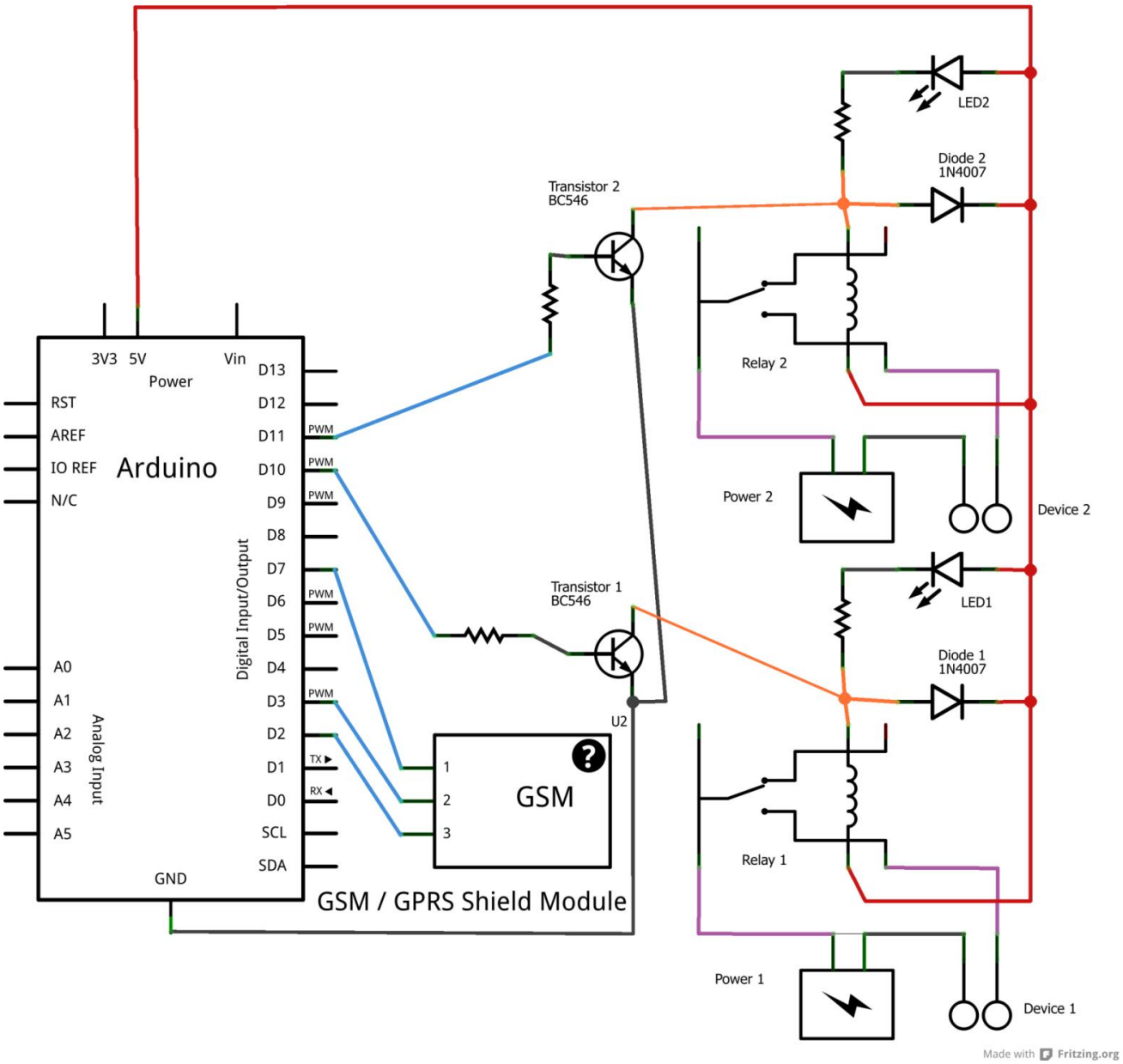
Συγκεντρωτικά η λίστα των υλικών και τα τεμάχια :

	Τεμάχια
Arduino Uno Rev3	1
Arduino GSM Shield	1
GSM Antenna	1
Power Supply 12VDC	1
Breadboard	1
Diode 1N4007	2
Transistor 546 B	2
Red LED 5mm	2
Αντιστάσεις 220Ω	2
Αντιστάσεις 10ΚΩ	2
NHG JZC-6F 5V Relay	2

Σχεδίαση

Κατασκευή

Ακολουθεί το σχεδιάγραμμα του κυκλώματος ελέγχου καθώς και του κυκλώματος ενεργοποίησης.



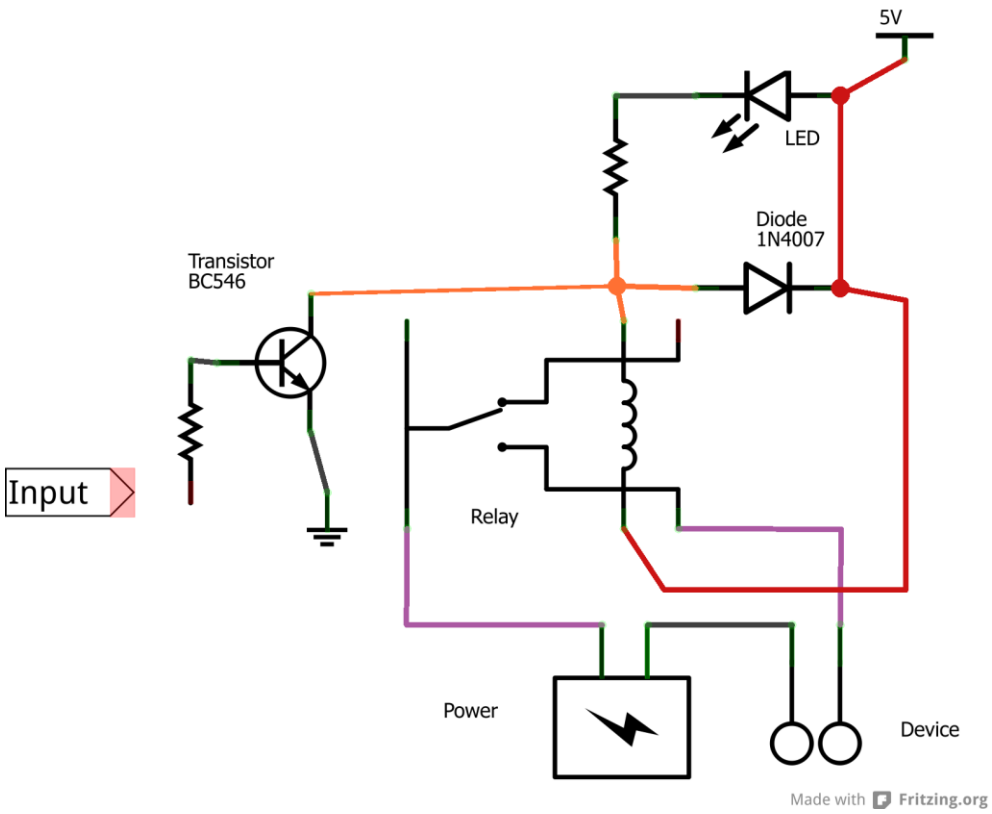
Σχεδιάγραμμα κυκλώματος έλεγχου και ενεργοποίησης.

Όπως φαίνεται το κύριο μέρος του κυκλώματος ελέγχου αποτελείτε από την πλατφόρμα Arduino και το GSM Shield που συνδέεται σε αυτήν. Η σύνδεση τους όπως θα φανεί παρακάτω, είναι πάρα πολύ απλά, το GSM Shield εφαρμόζει ακριβώς πάνω στις ακίδες του Arduino, αυτό είναι και το κύριο πλεονέκτημα των συμβατών shield που κυκλοφορούν.

Οι ακίδες του Arduino εξακολουθούν να είναι διαθέσιμες από τις ακίδες του Shield που θα βρίσκονται ακριβώς από πάνω, όλες εκτός από αυτές που θα χρησιμοποιεί το Shield και δεν πρέπει να γίνει χρήση τους.

Το κύκλωμα ενεργοποίησης, όπως φαίνεται και στο παρακάτω σχήμα, αποτελείται από μια ένα npn transistor με μια αντίσταση στη βάση του, ο λόγος που υπάρχει το transistor είναι η ενίσχυση του ρεύματος των ψηφιακών εξόδων του Arduino, ώστε να φτάσει σε ικανή τιμή για την ενεργοποίηση του ρελέ.

Το απαιτούμενο ρεύμα ενεργοποίησης του πηνίου του ρελέ είναι I = V / R = 5 / 70 ~= 71 mA, ενώ οι ψηφιακές έξοδοι του Arduino Uno μπορούνε να παρέχουν εως και 20mA μόνο.



Οι 5 ακροδέκτες του ρελέ

Παράλληλα στο πηνίο του ρελέ υπάρχει μια δίοδος η οποία υπάρχει για λόγους προστασίας από ανάστροφα ρεύματα το ρελέ, τα οποία μπορούν να βλάψουν σοβαρά το υπόλοιπο κύκλωμα.

Επίσης παράλληλα υπάρχει και ένα LED με μια αντίσταση για την ένδειξη λειτουργίας του ρελέ.

Στους ακροδέκτες του ρελέ συνδέετε στο ένα άκρο η μία φάση της τροφοδοσίας η οποία καταλήγει στην ηλεκτρική συσκευή. Η άλλη φάση της τροφοδοσίας συνδέετε απευθείας στη συσκευή. Έτσι όταν ενεργοποιείστε το ρελέ θα υπάρχει σύνδεση και ροή ρεύματος προς τη συσκευή ενώ όταν είναι κλειστό δεν θα υπάρχει ροή και η συσκευή θα παραμένει κλειστή.

Θα υπάρχουν δυο τέτοια κυκλώματα ενεργοποίησης, φυσικά και θα μπορούσαν να είναι περισσότερα.

Εντολές

Ο μικροελεγκτής δέχεται τις εντολές με μήνυμα κειμένου SMS, το οποίο αποστέλλετε στον τηλεφωνικό αριθμό της κάρτας SIM που είναι τοποθετημένη στο GSM Module. Τα οποία τα διαβάζει, τα αποκωδικοποιεί, ελέγχει την εγκυρότητά τους και στην συνέχεια εκτελεί την απαιτουμένη εντολή.

Οι υποστηριζόμενες εντολές είναι η επιλογή συσκευής και η επιλογή κατάστασης η οποία μπορεί να είναι άνοιγμα, κλείσιμο ή εναλλαγή.

Υπάρχουν τρεις μεταβλητές η p, η d και η f. Η μορφή των εντολών είναι pdf : όπου :

- p : Κωδικός *(Αρχικός κωδικός)*
    - Τιμή : ο χαρακτήρας-κωδικός της συσκευής πχ @

- d : Συσκευή *(Αριθμός συσκευής)*
    - Τιμές : { 0, 1 }

- f : Λειτουργία *(Κλείσιμο, Άνοιγμα, Εναλλαγή)* [προαιρετική]
    - Τιμές { 0 , 1 , t , - }
        - 0 : Κλείσιμο
        - 1 : Άνοιγμα
        - t : Εναλλαγή
        - - : προκαθορισμένη (αγνόηση)
    - Προκαθορισμένη τιμή : t

Παραδείγματα :

Εντολή : @*11* σημάνει επέλεξε την συσκευή 1 και ενεργοποίησε
Εντολή : @*0*- σημάνει επέλεξε την συσκευή 0 και αγνόησε την μεταβλητή 'λειτουργία' , οπότε εκτελείτε η προκαθορισμένη λειτουργία : εναλλαγή
Εντολή : @*00* σημάνει επέλεξε την συσκευή 0 και απενεργοποίησε

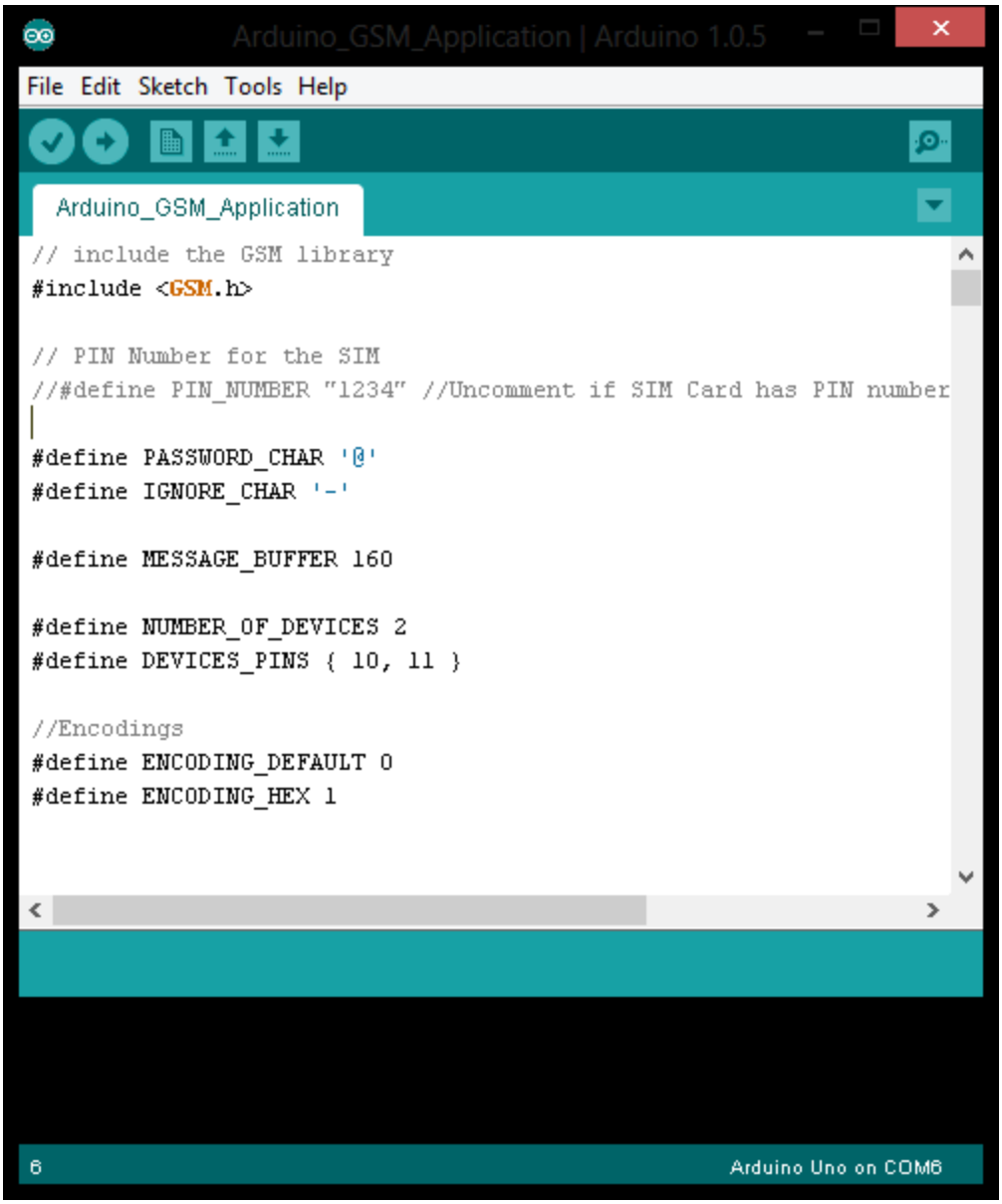
Η χρησιμότητα της μεταβλητή κωδικού p, είναι διπλή. Παρέχει μια τυπική προστασία απέναντι σε κακόβουλες εντολές ή λάθη, μπορεί να επιλεχθεί οποιοδήποτε αναγνώσιμος χαρακτήρας ώστε το κάθε σύστημα να έχει μοναδικό κωδικό.

Επίσης είναι απαραίτητος, όπως θα παρουσιαστεί παρακάτω, για την αντιμετώπιση του προβλήματος της εύρεση κωδικοποίησης του μηνύματος-εντολής.

Λογισμικό

Ο προγραμματισμός του μικροελεγκτή της πλατφόρμας Arduino γίνετε απευθείας από το Arduino IDE το οποίο είναι διαθέσιμο από την επίσημη ιστοσελίδα του.

Μέσω του Arduino IDE γίνετε εύκολος προγραμματισμός και η εγγραφή του κώδικα στον μικροελεγκτή, περιέχει διάφορες βιβλιοθήκες όπως η GSM library που θα χρειαστεί επίσης δίνει πρόσβαση στην σειριακή είσοδο και έξοδο της πλατφόρμας όταν αυτή είναι συνδεδεμένη σε ηλεκτρονικό υπολογιστή, διαβάζοντας την έξοδος και στέλνοντας εντολές απευθείας από τη σειριακή είσοδο διευκολύνοντας πολύ τον έλεγχο της πλατφόρμας και είναι ιδιαίτερα χρήσιμη κατά το στάδιο των δοκιμών για την αποσφαλμάτωση του κώδικα.



Arduino IDE

Το βασικό μέρος ενός προγράμματος για τον μικροελεγκτή αποτελείται από τις συναρτήσεις setup() και loop(), η setup() εκτελείτε μία φορά κατά την εκκίνηση του προγράμματος με σκοπό την αρχικοποίηση παραμέτρων και η loop() εκτελείτε επανειλημμένως κατά την λειτουργία του μικροελεγκτή.

Βιβλιοθήκη GSM library

Στην βιβλιοθήκη υπάρχουν διαθέσιμες οι εξής κλάσεις

- ***GSM***, αναλαμβάνει την επικοινωνία με το modem και είναι απαραίτητη για όλα τα GSM/GPRS προγράμματα
- ***GSMVoiceCall*** , για την διαχείριση τηλεφωνικών κλήσεων.
- ***GSM_SMS***, για Αποστόλη και λήψη γραπτών μηνυμάτων SMS.
- ***GPRSClass***, για την σύνδεση στο διαδίκτυο.
- ***GSMServer***, για την υλοποίηση εξυπηρετητή (server) και ***GSMClient*** για την υλοποίηση πελάτη.
- Και τέλος τις βοηθητικές ***GSMScanner*** και ***GSMModem.***

Για το σύστημα αυτό θα γίνει χρήση της GSM και της GSM_SMS.

Η βασική λειτουργία του συστήματος είναι η λήψη γραπτών μηνυμάτων για καλύτερη κατανόηση ακολουθεί ένα απλό παράδειγμα λήψης.

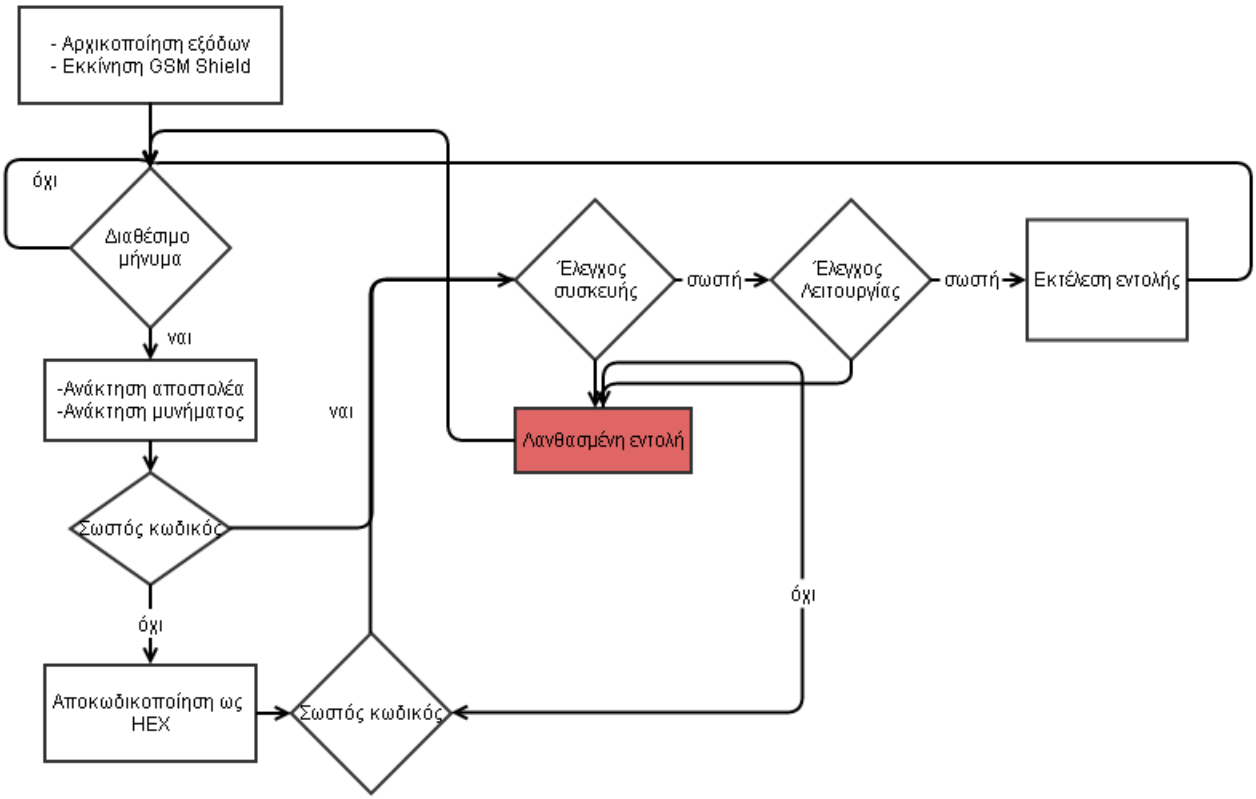
```
// include the GSM library
#include <GSM.h>

// initialize the library instances
GSM gsmAccess;
GSM_SMS sms;

// Array to hold the number a SMS is retreived from
char senderNumber[20];

void setup()
{
  // initialize serial communications and wait for port to open:
  Serial.begin(9600);

  Serial.println("SMS Messages Receiver");

  // connection state
  boolean notConnected = true;

  // Start GSM connection
  while(notConnected)
  {
    if(gsmAccess.begin()==GSM_READY)
      notConnected = false;
```

```
    else
    {
      Serial.println("Not connected");
      delay(1000);
    }
  }

  Serial.println("GSM initialized");
  Serial.println("Waiting for messages");
}

void loop()
{
  char c;

  // If there are any SMSs available()
  if (sms.available())
  {
    Serial.println("Message received from:");

    // Get remote number
    sms.remoteNumber(senderNumber, 20);
    Serial.println(senderNumber);

    // Read message bytes and print them
    while(c=sms.read())
      Serial.print(c);

    Serial.println("\nEND OF MESSAGE");

    // Delete message from modem memory
    sms.flush();
  }
  delay(1000);
}
```

Στο παράδειγμα αυτό στη συνάρτηση `setup()` αφού γίνετε η ενεργοποίηση του GSM Shield.

Στη συνάρτηση loop(), η οποία εκτελείτε συνεχώς, γίνετε έλεγχος αν υπάρχουν διαθέσιμα νέα μηνύματα SMS τότε τυπώνετε ο αριθμός του αποστολέα και στην συνέχεια διαβάζετε το περιεχόμενο του μηνύματος χαρακτήρα προς χαρακτήρα μέχρι το τέλος και μετά γίνετε η διαγραφή του μηνύματος το οποίο έχει πλέον τυπωθεί στην σειριακή έξοδο.

Διάγραμμα ροής του συστήματος



Αρχικά στη συνάρτηση setup() θα γίνει η αρχικοποίηση των εξόδων, τα διαθέσιμα pin ορίζονται ως pin εξόδου.

Γίνετε εκκίνηση του GSM Shield, σε αυτό το σημείο γίνετε προσπάθεια σύνδεσης στο δίκτυο 2G των κινητών τηλεφώνων, άν η κάρτα SIM έχει αριθμό pin θα πρέπει να δοθεί.

Αφού γίνει η σύνδεση εκτελείται η συνάρτηση loop() του μικροελεγκτή που μαζί με την setup() είναι οι δυο βασικές για την λειτουργία του. Γίνετε συνεχώς έλεγχος για νέα διαθέσιμα γραπτά μηνύματα που έχουν ληφθεί.

Γίνετε ανάκτηση του αριθμού του αποστολέα και ανάγνωση του περιεχομένου του μηνύματος, ακολουθεί η διαδικασία ανάλυσης του μηνύματος, γίνετε έλεγχος του χαρακτήρα κωδικού αν αυτός δεν είναι σωστός τότε θα γίνει μια προσπάθεια αποκωδικοποίηση του μηνύματος από δεκαεξαδική μορφή, παρακάτω θα δοθούν περισσότερες λεπτομέρειες για αποκωδικοποίηση, αν και πάλι ο κωδικός είναι λανθασμένος τότε η εντολή είναι λάθος και διακόπτεται η διαδικασία.

Σε περίπτωση σωστού κωδικού η διαδικασία συνεχίζετε με τον έλεγχο του αριθμού της συσκευής και της εντολής λειτουργίας. Αν κάποια από τις δυο είναι λανθασμένη τότε αγνοείτε η εντολή, σε αντίθετη περίπτωση γίνετε η εκτέλεση της, το

σύστημα επιστρέφει στην επαναληπτική διαδικασία loop( ) όπου θα περιμένει για το επόμενο μήνυμα.

Αποκωδικοποίηση

Η διαδικασία της αποκωδικοποίησης είναι απαραίτητη για τη σωστή λειτουργία του συστήματος καθώς, το μήνυμα εντολή που θα σταλεί μπορεί να είναι κωδικοποιημένο σε αυτή τη μορφή, αυτό εξαρτάται από την τηλεφωνική συσκευή ή το λογισμικό της. Η κωδικοποίηση αυτή είναι η GSM 03.38 και χρησιμοποιείτε κυρίως στα μηνύματα SMS, κάθε αναγνώσιμος χαρακτήρας λαμβάνετε από το GSM Shield ως τέσσερα δεκαεξαδικά αλφαριθμητικά ψηφία.

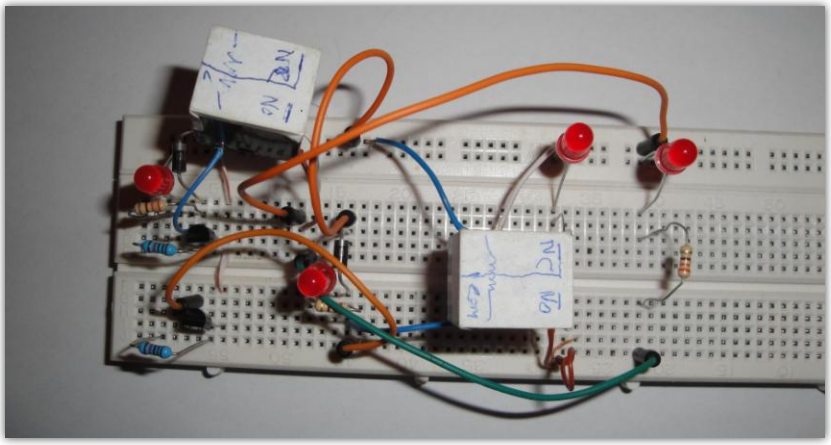
Η διαδικασία που ακολουθείτε είναι η ανάγνωση αυτών των τεσσάρων ψηφίων μετατροπή τους σε δεκαδικό αριθμό και από εκεί μετατροπή σε χαρακτήρα βάση της γνωστής κωδικοποίησης ASCII.

Επίσης θα χρειαστεί και η αντίθετη διαδικασία δηλαδή κωδικοποίηση ASCII χαρακτήρα σε αυτήν την μορφή, αυτή χρειάζεται για την μετατροπή του χαρακτήρα-κωδικού ώστε να μπορεί να γίνει ο έλεγχος

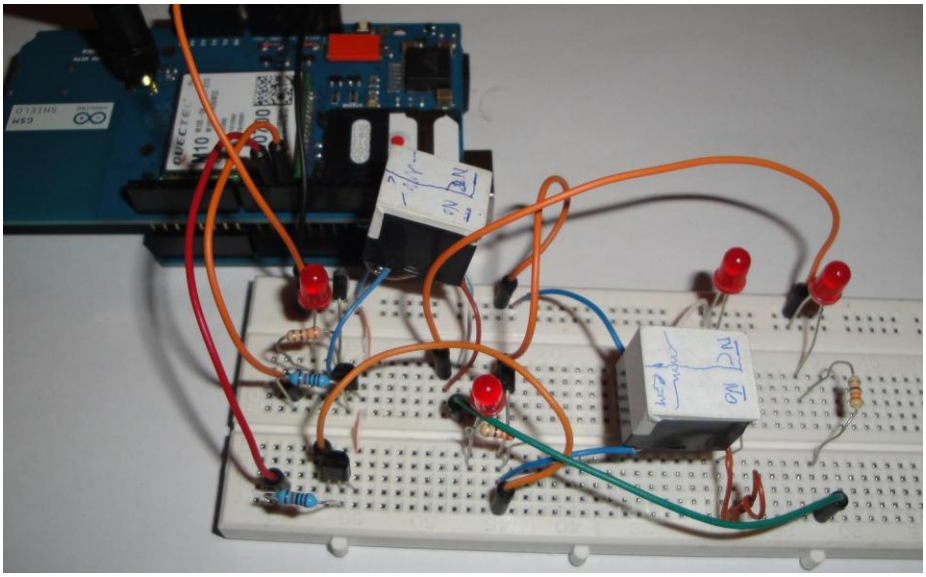
Συναρμολόγηση

Αρχικά έγινε η συναρμολόγησης του κυκλώματος ενεργοποίησης όπως σχεδιάστηκε καθώς και δοκιμές μίας και μπορεί να λειτουργήσει και ως ανεξάρτητο κύκλωμα χωρίς να χρειάζεται η πλατφόρμα Arduino.

Στους ακροδέκτες των δύο συσκευών τοποθετήθηκαν LED σε σειρά με αντιστάσεις για να γίνει εμφανής οπτικά η σωστή λειτουργία του κυκλώματος.



Στη συνέχεια αφού τοποθετήθηκε το GSM Shield πάνω στο Arduino, συνδέθηκαν οι ακροδέκτες 10 και 11 με το κύκλωμα ενεργοποίησης καθώς και οι απαραίτητοι ακροδέκτες τροφοδοσίας +5V και GND.



Παρουσίαση κώδικα

Παρακάτω είναι όλος ο κώδικας που θα περαστεί στον μικροελεγκτή της πλατφόρμας, αποτελεί την υλοποίηση του διαγράμματος ροής που παρουσιάστηκε στη σχεδίαση.

Η ανάγνωση του μηνύματος γίνετε από την συνάρτηση `readMessage` και η ανάλυση του από την εντολή `parseMessage.`

Υπάρχουν και δύο βοηθητικές συναρτήσεις η int_to_hexarray για την μετατροπή χαρακτήρα (αριθμού) σε συμβατή δεκαεξαδική κωδικοποίηση, η message_length που επιστρέφει το μήκος του μηνύματος σε χαρακτήρες και τέλος η decodeHEX_message που υλοποιεί την αποκωδικοποίηση από την δεκαεξαδική μορφή.

```
// include the GSM library
#include <GSM.h>

// PIN Number for the SIM
//#define PIN_NUMBER "1234" //Uncomment if SIM Card has PIN number

#define PASSWORD_CHAR '@'
#define IGNORE_CHAR '-'

#define MESSAGE_BUFFER 160

#define NUMBER_OF_DEVICES 2
#define DEVICES_PINS { 10, 11 }

//Encodings
#define ENCODING_DEFAULT 0
#define ENCODING_HEX 1



GSM gsmAccess;
GSM_SMS sms;


char hex[] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'
};

// Array to hold the number a SMS is retreived from
char senderNumber[20];
char message[MESSAGE_BUFFER];
```

```
byte devices = NUMBER_OF_DEVICES; //Number of devices
byte device_pins[ ] = DEVICES_PINS; // I/O Pin for each device

//Setup function
void setup()
{
  for( int i =0 ; i < devices ; i++){
    pinMode( device_pins[ i ], OUTPUT);
  }
  // initialize serial communications and wait for port to open:
  Serial.begin(9600);
  Serial.println("Messages Receiver Starting");
  // connection state
  boolean notConnected = true;

  // Start GSM connection
  while(notConnected)
  {
#ifndef PIN_NUMBER
    if(gsmAccess.begin()==GSM_READY)
#else
      if(gsmAccess.begin( PIN_NUMBER )==GSM_READY)
#endif
        notConnected = false;
      else
      {
        Serial.println("Not connected");
        delay(1000);
      }
  }
  Serial.println("GSM initialized");
  Serial.println("Waiting for messages");
}
//Read incoming message
void readMessage( ){
  char c;
  byte i=0;
  while( c=sms.read() ){
    Serial.print(c);
    if( i < MESSAGE_BUFFER ){
      message[ i ] = c;
    }
    i++;
  }
  message[ i ] = '\0';
```

```
  Serial.println();
}
//Parse SMS message
boolean parseMessage( ){
  byte index = 0;
  char c;

  byte length = message_length();

  if( !length ){
    Serial.println( "Invalid request" );
  }
  byte encoding = ENCODING_DEFAULT;

  //Check password
  if( message[ 0 ] == PASSWORD_CHAR ){
    Serial.println( "Password is correct" );
  }else{
    // Try to decode as HEX
    Serial.println( "Try to decode as HEX" );

    if( length < 4 ){
      Serial.println( "Invalid request length < 4" );
    }

    char password_hex[4];
    int_to_hexarray( PASSWORD_CHAR, password_hex );

    boolean password_match = true;
    Serial.println( "Password");
    for( byte i = 0; i < 4; i++ ){
      Serial.print( password_hex[i] );
      if( password_hex[i] != message[i] ){
        password_match = false;
      }
    }
    Serial.println( "" );
    if( !password_match ){
      Serial.println( "Password is incorrect" );
      return false;
    }
    Serial.println( "Password is correct" );
    Serial.println( "Encoding is HEX" );
    encoding = ENCODING_HEX;

    if( !decodeHEX_message( length ) ){
```

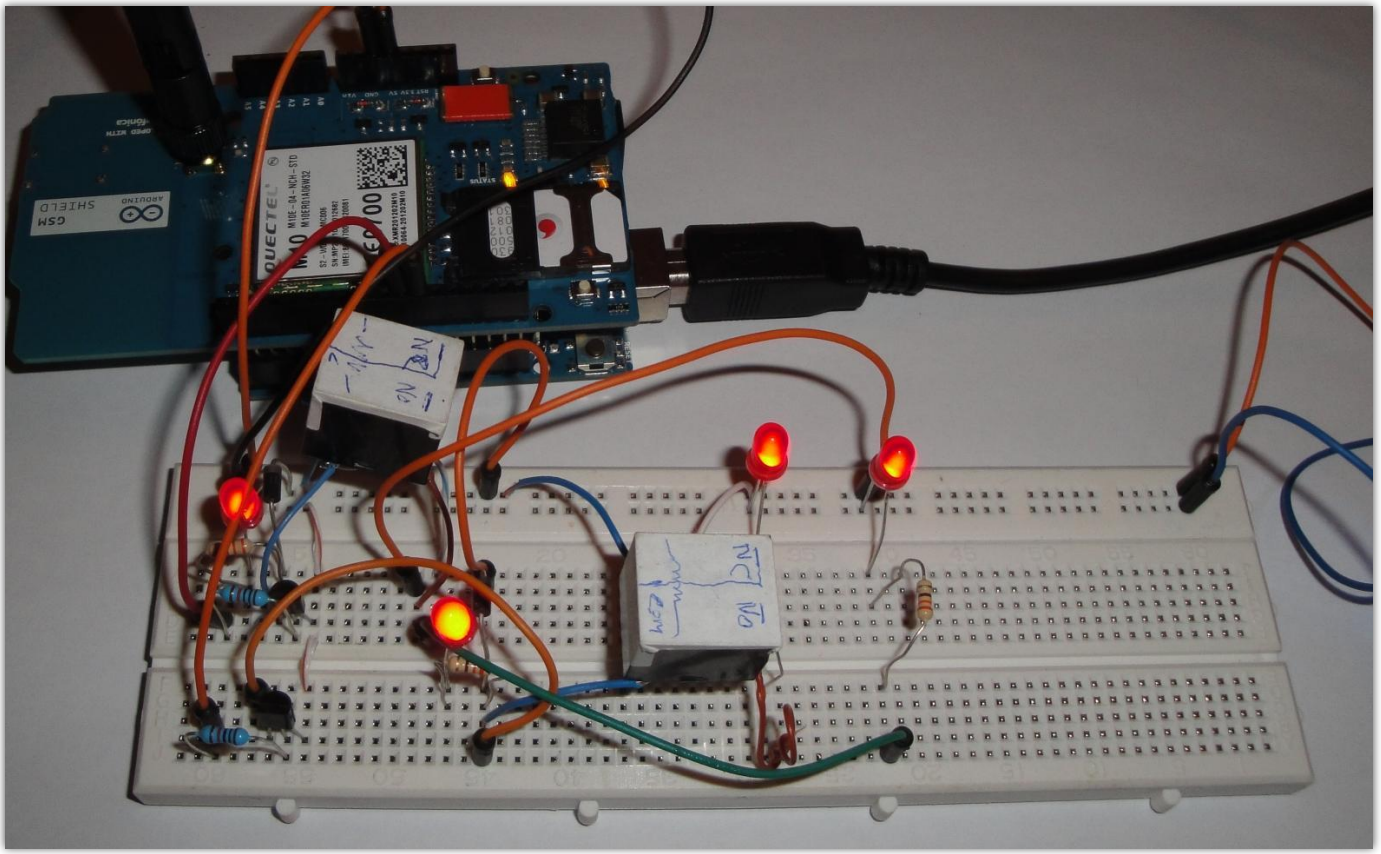
```
    Serial.println( "Error decoding message" );
    return false;
  }
  Serial.println( "ENCODED MESSAGE" );
  int length=0;
  while( message[ length ] != '\0' ){
    Serial.print( message[ length ] );
    length++;
  }
  Serial.println( "" );
  Serial.println( "END OF ENCODED MESSAGE" );
  length = message_length();

}
byte device = 0;
char operation = 't';

//Read device byte
c = message[ 1 ];
if( c >= '0' && c - '0' <= (devices-1) ){
  device = (byte)( c - '0' );
}
else{
  Serial.println( "Invalid device" );
  return false;
}

//Read operation
if( length > 2 ){
  c = message[ 2 ];
  if( c != IGNORE_CHAR && c != '\0' ){
    if( c == '0' || c == '1' || c == 't' ){
      operation = c;
    }
    else{
      Serial.println( "Invalid operation" );
      return false;
    }
  }
}

//Process request
Serial.println( "Valid request" );

Serial.print( "Device : ");
Serial.println( device );
```

```
  Serial.print( "Operation : ");
  Serial.println( operation );

  Serial.print( "Device pin : ");
  Serial.println( device_pins[ device ] );

  //do operation I/0
  int OutputValue = LOW;
  if( operation == '0' ){
    OutputValue = LOW;
  }
  else if( operation == '1' ){
    OutputValue = HIGH;
  }
  else if( operation == 't' ){
    if( digitalRead( device_pins[ device ] ) ){
      OutputValue = LOW;
    }else{
      OutputValue = HIGH;
    }
  }
  Serial.print( "Set output : " );
  Serial.print( OutputValue );
  digitalWrite( device_pins[ device ], OutputValue );

  //log

  return true;
}
//Loop function
void loop()
{

   // If there are any SMSs available()
  if (sms.available())
  {
    Serial.println("Message received from:");

    // Get remote number
    sms.remoteNumber(senderNumber, 20);
    Serial.println(senderNumber);
    Serial.println( "message :" );

    //Read message
    readMessage();
```

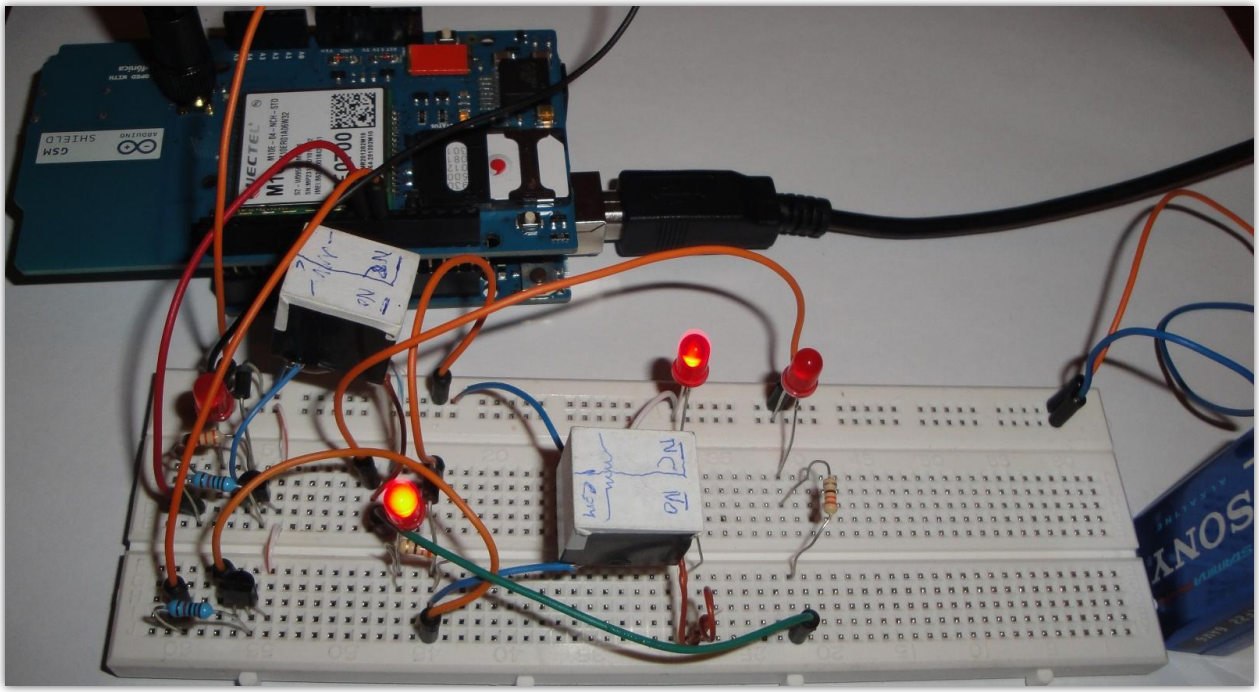
```
    Serial.println( "PARSE MESSAGE" );
    parseMessage();

    Serial.println("\nEND OF MESSAGE");

    // Delete message from modem memory
    sms.flush();
    Serial.println("MESSAGE DELETED");
  }

  delay(1000);
}

//Returns the HEX encoded value from unsigned int
void int_to_hexarray( unsigned int value, char hex_value[]  ){
  int length = 3;
  /* int to hex */
  do{
    unsigned int power = 1;

    for( int i = 0; i < length ; i++){
      power *= 16U;

    }
    hex_value[ 3- length ] = (char)hex[ ( unsigned int )( value / power ) ] ;
    value = value % power;
    length--;
  }
  while( length >= 0 );
}

//Get the length of the message
int message_length(){
  int length=0;
  while( message[ length ] != '\0' ){
    length++;
  }
  return length;
}

//Decode the message from the HEX encoding
boolean decodeHEX_message( int length ){
  char new_message[ (int)(length/4)  ];
  int value = 0;
  byte index = 0;
  byte new_index = 0;
```

```
  for( ;; ){
    //if end of message
    if( message[ index ] == '\0' ){
      new_message[ new_index ] = '\0';
      break;
    }
    unsigned int power = 1;
    for( int i = 0; i < 3-(index%4) ; i++){
      power*= 16;
    }

    int character_index = -1;
    for( int i = 0 ; i < 16 ; i++){
      if( hex[i] == message[ index ] ){
        character_index = i;
        break;
      }
    }
    //if character not found
    if( character_index == -1 ){
      Serial.println( "ERROR" );
      return false;
    }
    value += character_index*power;
    //End of 4 bytes block
    if( !((index+1)%4)  ){
      new_message[ new_index ] = (char)value;
      new_index++;
      value = 0;
    }
    index++;
  }
  //Make sure stings ends with null character
  if( new_message[ new_index ] != '\0' ){
    new_message[ new_index + 1 ] = '\0';
  }

  index = 0;
  //Copy
  do{
    message[ index ] = new_message[ index ];
    index++;
  }while( new_message[ index-1 ] != '\0' );
    return true;
}
```

Λειτουργία



Λειτουργία και με τις δυο συσκευές ενεργοποιημένες.



Λειτουργία και με ενεργοποιημένη την πρώτη συσκευη.

Παράδειγμα εξόδου

Ακολουθούν αποσπάσματα της εξόδου του συστήματος στην σειριακή έξοδο η οποία μπορεί να διαβαστεί και από ηλεκτρονικό υπολογιστή.

Message received from:
+3069xxxxxxxx
message :
0040003100310030
PARSE MESSAGE
Try to decode as HEX
Password is correct
Encoding is HEX
ENCODED MESSAGE
@11
END OF ENCODED MESSAGE
Valid request
Device : 1
Operation : 1
Device pin: 11
Set output : 1

Message received from:
+3069xxxxxxxx
message :
@10
PARSE MESSAGE
Password is correct
Valid request
Device : 1
Operation : 0
Device pin: 11
Set output : 0

Message received from:
+3069xxxxxxxx
message :
Hello World
Password is incorrect

Το πρώτο μήνυμα είναι μία εντολή κωδικοποιημένη με δεκαεξαδικούς χαρακτήρες και εκτελείτε η αποκωδικοποίηση του μηνύματος. Η εντολή του μηνύματος αυτού είναι ενεργοποίηση την συσκευή 1.

Στο δεύτερο η εντολή του μνήματος είναι απενεργοποίηση την συσκευή 1. Τέλος στο τρίτο έχει σταλεί κατά λάθος ένα τυχαίο μήνυμα το οποίο αγνοείτε καθώς δεν ξεκινάει με τον σωστό κωδικό.

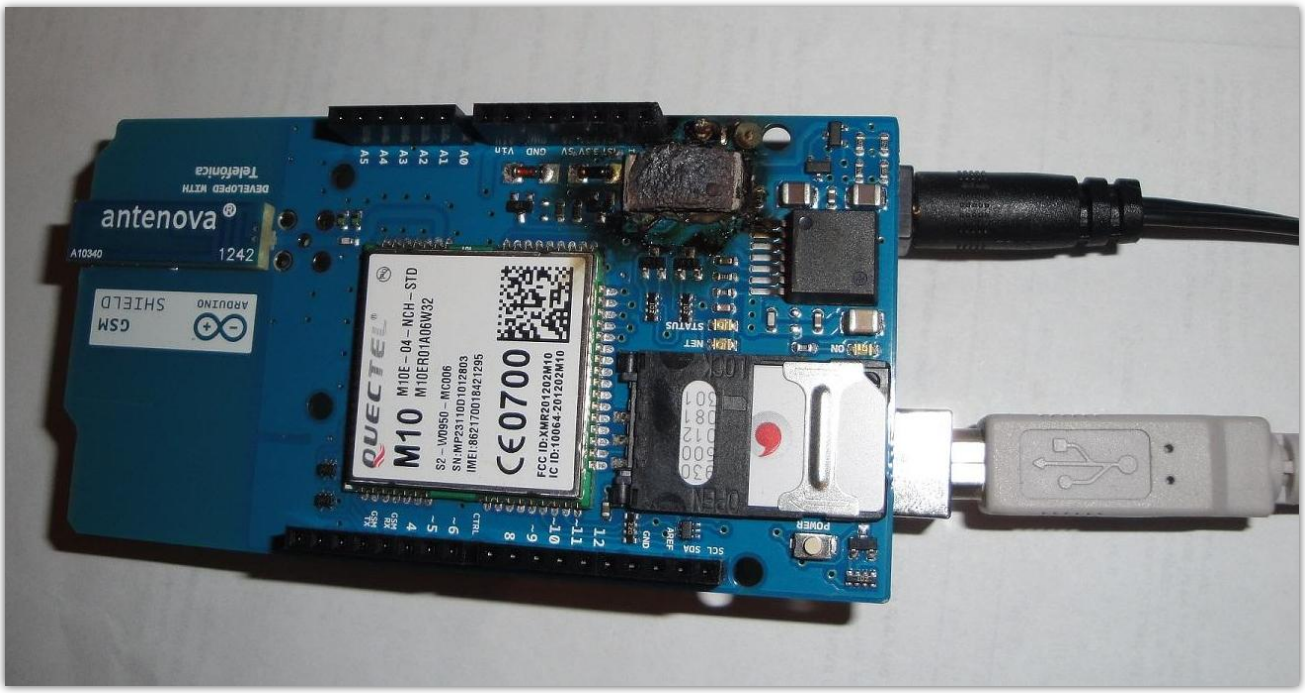
Σε κάθε περίπτωση λάθους μηνύματος υπάρχουν όλοι οι απαραίτητοι έλεγχοι εγκυρότητας της εντολής προτού γίνει η εκτέλεση της.

Προβλήματα κατά την υλοποίηση

Δεν αντιμετωπίστηκαν ιδιαίτερα προβλήματα κατά την υλοποίηση απο πλευράς λογισμικού, καθώς η ομοιότητα της Wiring, της γλώσσα που χρησιμοποιεί η πλατφόρμα Arduino, με την γλώσσα C είναι πολύ μεγάλη διευκολύνοντας έτσι την κατανόηση και την γραφή του απαιτουμένου κώδικα.

Από πλευράς υλικού υπήρξαν αρκετά προβλήματα, αρχικά είχε σχεδιαστεί να χρησιμοποιηθεί το EFcom/GPRS Shield, παρόλο τις προσπάθειές που έγιναν δεν μπόρεσε να λειτουργήσει σωστά.

Τελικά επιλέχθηκε το Arduino GSM Shield στη θέση. Κατά την διάρκεια των δοκιμών όμως, ο πυκνωτής τροφοδοσίας πήρε φωτιά και εξερράγη καταστρέφοντας το Shield και το Arduino Uno. Η ζημία δεν αναγνωρίστηκε ως ελάττωμα του Shield και δεν έγινε η αλλαγή του προϊόντος.



Συμπεράσματα

Ο έλεγχος ηλεκτρικών συσκευών από απόσταση μπορεί να φανεί πολύ χρήσιμος στις καθημερινές μας ασχολίες, η υλοποίηση που ακολουθήθηκε σε αυτήν την εργασία είναι ένα πολύ πρακτικός και ολοκληρωμένος τρόπος ελέγχου, συνδυάζει την ευκολία που προσφέρουν τα κινητά τηλέφωνα, που όλοι μας έχουμε πάντα μαζί μας και μέσω μια πολύ απλής διαδικασίας όπως η αποστολή μηνύματος SMS μπορούμε να ελέγξουμε οποιαδήποτε συσκευή όπου και αν βρίσκετε αυτή.

Το κόστος της υλοποίησης δεν είναι απαγορευτικό, καθώς η τεχνολογία προχωράει το κόστος των ηλεκτρονικών συνεχώς μειώνετε. Το κόστος των στοιχείων για το σύστημα ενεργοποίησής είναι πολύ χαμηλό, της πλακέτας Arduino είναι αρκετά καλό και μπορεί να μειωθεί και άλλο αγοράζοντας έναν κλώνο που κυκλοφορεί ή ακόμα και συναρμολογώντας εναν από την αρχή, αφού είναι open source, κάτι που είναι και από τα βασικά προτερήματα της πλατφόρμας arduino.

Τέλος το μεγαλύτερο κόστος το έχει το GSM Shield, καθώς είναι το πιο τεχνολογικά αναπτυγμένο τμήμα του συστήματος. Υπάρχουν φθηνότερες εναλλακτικές που μπορούν να χρησιμοποιηθούν Επίσης μπορεί να γίνει χρήση άλλων συστημάτων όχι απαραίτητα GSM, όπως WiFi, Ethernet για λειτουργία σε τοπικό δίκτυο ή και μέσω διαδικτύου, επίσης μια άλλη λύση για να γίνει μηδενικό το κόστος είναι η χρήση του τηλεφωνικού δικτύου απευθείας για λήψη των εντολών ενεργοποίησης.

Φυσικά μπορούν να χρησιμοποιηθούν διαφορετικές πλατφόρμες, όπως το Raspberry PI για παράδειγμα και πολλούς άλλους τρόπους. Η εργασία αυτή αποτελεί μια πάρα πολύ καλή βάση για περαιτέρω ανάπτυξη τέτοιων συστημάτων, στη παρακάτω ενότητα ακολουθεί πλήθος ιδεών που μπορεί να υλοποιηθεί με βάση όλων όσων παρουσιάστηκαν.

Ιδέες για μελλοντικές υλοποιήσεις

Υπάρχουν πάρα πολλοί τρόπο που θα μπορούσε να γίνει έλεγχος εξ αποστάσεως, παρακάτω παρουσιάζονται συνοπτικά μερικές ιδέες, πολλές από αυτές μπορούν να συνδυαστούν μεταξύ τους.

- Σύνδεση πολλών συσκευών
    - στις υπόλοιπες διαθέσιμες θύρες.
    - επιλογή Arduino Mega αντί του Uno, 54 ψηφιακές έξοδοι συνολικά.
    - σχεδίαση κυκλώματος με πολυπλέκτες και Flip-Flop για την διαχείριση πολλών κυκλωμάτων ενεργοποίησης.
- Προχωρημένες εντολές
    - Διάρκεια λειτουργίας.
    - Χρονοκαθυστέρηση εκτέλεσης.
    - Αναφορά λειτουργίας.
- Ασφάλεια χρηστών, αποδοχή εντολών από συγκεκριμένους τηλεφωνικούς αριθμούς ή και δημιουργία λογαριασμών χρηστών για πρόσβαση σε συγκεκριμένες συσκευές μόνο.
- Σχεδίαση του συστήματος με άλλους δέκτες όπως τηλεκοντρόλ υπερύθρων, Ethernet ή Wifi για εναλλακτικό τρόπο έλεγχου.
- Κατασκευή της πλατφόρμας Arduino από την αρχή (Minduino) ώστε να έχει το επιθυμητό μέγεθος / σχήμα
- Διαχείριση και μέσω Internet, δημιουργία WEB εφαρμογής με πρόσβαση στις λειτουργίες και στα στατιστικά χρήσης.
- Κεντρικό σύστημα ελέγχου σε έναν χώρο με επικοινωνία συνδεδεμένων μεταξύ τους κόμβων για τον έλεγχο διαφόρων συσκευών στον χώρο.

Βιβλιογραφία

- http://www.arduino.cc
- http://arduino.cc/en/Main/ArduinoBoardUno
- http://arduino.cc/en/Main/ArduinoGSMShield
- http://en.wikipedia.org/wiki/Relay
- http://en.wikipedia.org/wiki/GSM_03.38
- http://www.instructables.com/id/Connecting-a-12V-Relay-to-Arduino/