ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση Συστήματος Απομακρυσμένης Επιτήρησης Κυψέλης Μελισσών

Καποδίστριας Νικόλαος (Α.Μ: 39452)
Σμυρνάκης Απόστολος (Α.Μ: 39182)

Εισηγητής: Δρ. Ιωάννης Έλληνας, Καθηγητής

Περιεχόμενα:

ΠΕΡΙΛΗΨΗ

Αντικείμενο της παρούσας εργασίας αποτελεί ο σχεδιασμός και η κατασκευή σταθμού, για τον απομακρυσμένο έλεγχο μελισσιών.

Τα δεδομένα από τα αισθητήρια συλλέγονται ανα τακτά χρονικά διαστήματα και αποστέλλονται μέσω του δικτύου WiFi στην πλατφόρμα ThingSpeak. Τα δεδομένα παρουσιάζονται και στην συνοδευτική ιστοσελίδα της πτυχιακής εργασίας.

Επιπρόσθετα, πέραν της απεικόνισης των δεδομένων προβλέπεται και η σχεδίαση κατάλληλης αρχιτεκτονικής για τον απομακρυσμένο έλεγχο της κατασκευής, και για την ενημέρωση του χρήστη μέσω προσωπικού μηνύματος SMS.

Η σύνδεση του σταθμού στο διαδίκτυο θα πραγματοποιείται με δύο διαφορετικούς τρόπους. Ο πρώτος είναι μέσω σύνδεσης σε δίκτυο WiFi, αν και εφόσον υπάρχει διαθέσιμο εντός του δικτύου εγκατάστασης του σταθμού. Ο δεύτερος, είναι μέσω κινητής τηλεφωνίας με τη χρήση δεδομένων στο δίκτυο κινητής τηλεφωνίας GPRS.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα θέλαμε να ευχαριστήσουμε ιδιαίτερα τους καθηγητές μας κ. Ιωάννη Έλληνα και κ. Σταμάτη Αλατσαθιανό, αφενός για την εμπιστοσύνη που μας έδειξαν καθ' όλη τη διάρκεια ανάπτυξης της εργασίας και αφετέρου για τον χρόνο που αφιέρωσαν κατά την διάρκεια της υλοποίησής της.

Στη συνέχεια, θα θέλαμε να ευχαριστήσουμε τις οικογένειές μας για τη στήριξη που μας προσέφεραν κατά τη διάρκεια των σπουδών μας και που στέκονται δίπλα μας όλα αυτά τα χρόνια.

1. Οδηγίες χρήσης

1.1. Σύνδεση σε δίκτυο WiFi

Οι ρυθμίσεις του δικτύου WiFi (όνομα δικτύου SSID και κωδικός Password) αποθηκεύονται σε ένα ξεχωριστό αρχείο τύπου header (.h) με όνομα *secrets.h* . Στο αρχείο αυτό δηλώνουμε τις σταθερές που αντιπροσωπεύουν το SSID και το Password και έπειτα το “εισάγουμε” στον κύριο κώδικα της κατασκευής.

```
#include "secrets.h"
```

Μέσα στο secrets.h έχουμε δηλώσει όλα τα “μυστικά” του κώδικα, κωδικούς και προσωπικά κλειδιά που δεν θέλουμε να είναι άμεσα διαθέσιμα προς όλους στο Github.

```
#define WIFI_MOBILE_SSID "Android_hostpot"
#define WIFI_MOBILE_PASS "0123456789"
#define THINGSP_WR_APIKEY “ZssfSsAqPCmZd+jsdISkSe9EeFSrsL”
```

Κατόπιν, μέσα στον κύριο κώδικά μας κάνουμε “ανάγνωση” αυτών των μυστικών και τα αναθέτουμε σε μεταβλητές.

```
char defaultSSID[ ] = WIFI_MOBILE_SSID;
char defaultPASS[ ] = WIFI_MOBILE_PASS;
char apiKey[ ] = THINGSP_WR_APIKEY;
```

Με αυτό τον τρόπο πετυχαίνουμε να έχουμε άμεσα διαθέσιμα στον κώδικα (κατά τη διαδικασία της μεταγλώττισης) όλα τα κλειδιά και όλους τους κωδικούς, χωρίς να χρειάζεται να τα μοιραστούμε μέσω Github.

Κατά την εκκίνηση της πλακέτας, το ESP8266 προσπαθεί να συνδεθεί στο δίκτυο που ορίσαμε (μεταβλητή *defaultSSID[ ]*) με χρήση του κωδικού *defaultPASS[ ]* . Αν η σύνδεση είναι επιτυχής, το δίκτυο WiFi θα χρησιμοποιηθεί για όλες τις επικοινωνίες. Σε περίπτωση αδυναμίας σύνδεσης, τότε η πλακέτα θα λειτουργήσει σε GPRS mode.
Θα χρησιμοποιήσει δηλαδή το δίκτυο κινητής τηλεφωνίας για ανταλλαγή δεδομένων, ενεργοποιώντας τη σύνδεση μόνο πριν από κάθε αποστολή.

1.2. Λήψη δεδομένων μέσω γραπτού μηνύματος SMS

Για να λάβουμε γραπτό μήνυμα κειμένου (SMS) με τις τιμές των αισθητήρων την τρέχουσα χρονική στιγμή, αρκεί να στείλουμε μήνυμα SMS με το κείμενο **"report"** στο τηλεφωνικό νούμερο της πλακέτας.
Με τη λήψη αυτού του μηνύματος, ο μικροελεγκτής κάνει ανάγνωση των αισθητήρων και μας απαντάει με γραπτό μήνυμα τις τιμές τους.

Σε περίπτωση που η ανάγνωση κάποιου αισθητήρα αποτύχει, η τιμή αυτού θα παραληφθεί από το εξερχόμενο μήνυμα κειμένου. Παραδείγματα στο Παράρτημα 3.1.

1.3. Ρύθμιση ρυθμού αποστολής δεδομένων στην πλατφόρμα ThingSpeak

Για μια μοναδική αποστολή των τιμών των αισθητήρων στην πλατφόρμα Thing Speak, αρκεί να στείλουμε στη συσκευή μήνυμα γραπτού κειμένου με το περιεχόμενο **"upload"**.
Ο μικροελεγκτής θα κάνει ανάγνωση των τιμών των αισθητήρων και θα αποστείλει τα δεδομένα στην πλατφόρμα Thing Speak, με χρήση είτε του δικτύου WiFi είτε του δικτύου κινητής τηλεφωνίας.

Αποστέλλοντας γραπτό μήνυμα με το κείμενο **"auto30"** ζητάμε από το μικροελεγκτή να κάνει αυτόματη αποστολή στην πλατφόρμα Thing Speak κάθε 30 λεπτά. Οι επιτρεπόμενες τιμές για αυτόματη αποστολή είναι τα 30, 45, 90 και 120 λεπτά. Αντίστοιχα, το μήνυμα θα πρέπει να είναι **"auto30"** ή **"auto45"** ή **"auto90"** ή **"auto120"** .

Για να ακυρώσουμε την αυτόματη αποστολή δεδομένων, αρκεί να στείλουμε γραπτό μήνυμα με περιεχόμενο **"autocancel"**.

ΣΗΜΕΙΩΣΗ: για τις ανάγκες της παρουσίασης και για καλύτερη αναπαράσταση/προσομοίωση πραγματικών συνθηκών, οι παραπάνω τιμές αντιστοιχούν σε δευτερόλεπτα και όχι σε λεπτά.

2. Τεχνολογίες

2.1. Analog to Digital Converter (ADC)

Πέραν των ψηφιακών σημάτων κάποιοι μικροελεγκτές παρέχουν και την εφαρμογή αναλογικών σημάτων. Η συγκεκριμένη ιδιότητα είναι ιδιαίτερα χρήσιμη καθώς πολλά αισθητήρια δεν χρησιμοποιούν κάποιο σειριακό πρωτόκολλο επικοινωνίας, αλλά εξάγουν ένα επίπεδο τάσης που αντιστοιχεί στη τιμή του μετρούμενου μεγέθους.

Το γεγονός ότι πολλοί μικροελεγκτές έχουν ενσωματωμένο πάνω τους κάποιο κύκλωμα μετατροπής αναλογικού σε ψηφιακό σήμα καθιστά την ανάπτυξη του τελικού συστήματος αρκετά πιο απλή, καθώς δεν υπάρχει η ανάγκη χρήσης εξωτερικών ADC μετατροπέων. Αυτό βέβαια έρχεται σε κάποιο κόστος αφού η χρήση των ενσωματωμένων μετατροπέων συνήθως είναι λιγότερο ακριβής στη μέτρηση αναλογικών τάσεων.

Η λογική του κυκλώματος ενός ADC ακολουθεί κάποια βήματα για την μετατροπή ενός αναλογικού σήματος σε ψηφιακό. Αρχικά, μέσω κατάλληλης δειγματοληψίας με ρυθμό διπλάσιο έστω της συχνότητας του σήματος σύμφωνα με το θεώρημα του Nyquist, και μέσω ενός κυκλώματος δειγματοληψιας αποθηκεύει το επίπεδο της τάσης του σήματος για ένα μικρό χρονικό διάστημα και το τροφοδοτεί στο κύκλωμα μετατροπής του σε ψηφιακό.

Ένας πολύ συχνός τρόπος μετατροπής αναλογικού σε ψηφιακό σήμα είναι μέσω του successive approximation (SAR) που πραγματοποιεί δυαδική αναζήτηση σε όλα τα επίπεδα τάσης μέχρις να βρει το πιο κοντινό που είναι αποθηκευμένο στον sample and holder. Με την εύρεση του επιπέδου τάσης σηματοδοτείται το σύστημα μέσω του σήματος EOC (end of conversion). Πρέπει να τονιστεί πως ο παραπάνω τρόπος μετατροπής αναλογικού σήματος σε ψηφιακό δεν είναι ο μοναδικός, αλλά είναι ο τρόπος με τον οποίο μεγάλο ποσοστό μικροελεγκτών υλοποιούν το κύκλωμα ADC που περιλαμβάνουν.

2.2. SIM900

Το SIM900 είναι ένα module το οποίο παρέχει πρόσβαση στο δίκτυο GSM. Κύριο μέλημα σε αυτό το σημείο είναι να δούμε τι ακριβώς είναι το GSM δίκτυο αλλά και το GPRS.

2.3. Δίκτυο Κινητής Τηλεφωνίας

Για την πρόσβαση μέσω του κινητού μας στο internet ή την πραγματοποίηση κλήσεων είναι απαραίτητο να υπάρχει διαθέσιμο ένα δίκτυο κινητής τηλεφωνίας. Όταν χρησιμοποιούμε το κινητό τηλέφωνο, στέλνει και λαμβάνει σήματα από και προς έναν σταθμό βάσης, ο οποίος στη συνέχεια επικοινωνεί με κάποια κέντρα αναδιανέμοντας την πληροφορία.

Ο σταθμός βάσης είναι το σύνολο των εγκαταστάσεων μιας εταιρείας κινητής τηλεφωνίας που τοποθετούνται για την υποστήριξη του ασυρμάτου δικτύου της.

Οι σταθμοί βάσης αποτελούνται από ειδικές κεραίες εκπομπής και λήψης σημάτων, καθώς και ηλεκτρονικό εξοπλισμό για την επεξεργασία των σημάτων. Συνήθως, είναι τοποθετημένοι σε μεταλλικούς πυλώνες ή σε οροφές ψηλών κτιρίων.

Κάθε σταθμός βάσης καλύπτει μία συγκεκριμένη γεωγραφική περιοχή, η οποία χωρίζεται σε μία ή περισσότερες κυψέλες. Το επιθυμητό μέγεθος της κυψέλης καθορίζεται βάσει των ιδιαίτερων γεωγραφικών χαρακτηριστικών της περιοχής, αλλά και του επιθυμητού αριθμού κινητών τηλεφώνων εξυπηρέτησης. Κάθε σταθμός βάσης μπορεί να εξυπηρετήσει περιορισμένο αριθμό κινητών τηλεφώνων, έτσι είναι σύνηθες εντός πόλης οι κυψέλες να είναι αρκετά μικρού μεγέθους ενώ εντός αγροτικών περιοχών να είναι μεγαλύτερες.

Στο σύστημα GSM η περιοχή συχνοτήτων που έχει εκχωρηθεί για την λειτουργία των δικτύων κινητής τηλεφωνίας υποδιαιρείται σε περισσότερες περιοχές συχνοτήτων, συγκεκριμένων καναλιών επικοινωνίας εύρους 200kHz.

Στη συνέχεια κάθε κανάλι μέσω της τεχνικής πολυπλεξίας, είναι σε θέση να επιτρέψει σε εναν αριθμό συνδρομητών να το χρησιμοποιήσουν ταυτόχρονα. Κάθε σταθμός βάσης επικοινωνεί με τα κινητά τηλέφωνα που βρίσκονται στην περιοχή. Μεταξύ γειτονικών κυψελών γίνεται προσπάθεια χρήσης όσο το δυνατόν διαφορετικών καναλιών για την αποφυγή παρεμβολών μεταξύ τους. Από πλευράς κινητών τηλεφώνων ή οποιασδήποτε συσκευής που κάνει χρήση του δικτύου, επιλέγεται ο σταθμός με το ισχυρότερο σήμα ισχύος ενώ σε περίπτωση μεταβολής της θέσης της συσκευής γίνεται αλλαγή σταθμού δίχως να το αντιλαμβάνεται ο χρήστης. Τέλος, είναι σημαντικό να γνωρίζουμε την μπάντα συχνοτήτων μέσα στην οποία θέλουμε το σύστημα μας να λειτουργεί.

Είναι σημαντικό το module που θα επιλέξουμε να υποστηρίζει τη συχνότητα λειτουργίας του παρόχου μας.

3. Κατασκευή

3.1. Πορεία κατασκευής

Σχεδίαση

Το πρώτο βήμα κάθε εργασίας / project είναι η σχεδίαση και η νοητική απεικόνιση της ιδέας. Σε αυτό το βήμα θα γίνει αξιολόγηση και ανάλυση των απαιτήσεων και θα παρθεί η απόφαση για τις τελικές παραμέτρους της κατασκευής. Κατόπιν θα αναλυθεί το υλικό που είναι διαθέσιμο στην αγορά ώστε να επιλέξουμε το καταλληλότερο. Πάνω σε αυτό το υλικό θα αναπτυχθεί ο κώδικας.

Κώδικας

Η λειτουργία της συσκευής είναι διπλή: επαναλαμβανόμενη αποστολή δεδομένων στην πλατφόρμα ThingSpeak και/ή αποστολή δεδομένων μέσω SMS κατ'απαίτηση του χρήστη. Τα δεδομένα συλλέγονται από τους αισθητήρες μόνο όταν πραγματικά χρειάζονται (πριν την αποστολή) έτσι ώστε να εξοικονομούμε ενέργεια.

Έχουμε δύο τρόπους σύνδεσης στο διαδίκτυο: είτε μέσω WiFi είτε με χρήση των δεδομένων κινητής τηλεφωνίας. Κατά την εκκίνηση του συστήματος, γίνεται απόπειρα σύνδεσης σε δίκτυο WiFi. Αν αυτό αποτύχει, τότε θα γίνεται χρήση του δικτύου κινητής τηλεφωνίας.

Η αποστολή μηνύματος κειμένου SMS γίνεται όταν το αιτηθεί ο χρήστης και περιέχει τις τιμές των αισθητήρων.

Υλικό

Μετά την ακριβή περιγραφή της λειτουργίας της κατασκευής και αφού έχουμε αναλύσει τη ροή του κώδικα, ακολουθεί η σύνδεση των επιμέρους κυκλωμάτων. Τα περιφερειακά μας είναι:

- ο μικροελεγκτής
- το υποσύστημα διασύνδεσης μέσω WiFi (ESP8266)
- το μόντεμ κινητής τηλεφωνίας (SIM900)
- οι αισθητήρες βάρους μαζί με την πλακέτα ελέγχου (γέφυρα Whinstone - HX711)
- ο αισθητήρας θερμοκρασίας / υγρασίας (DHT11)

Οι αισθητήρες βάρους τοποθετούνται πάνω σε μια επίπεδη σκληρή επιφάνεια και στηρίζουν τη βάση του μελισσιού.

3.2. Προγραμματισμός συστήματος - Γλώσσα Προγραμματισμού

Για τον προγραμματισμό του μικροελεγκτή χρησιμοποιούμε το Arduino IDE. Παρέχει πρόσβαση σε όλες τις απαραίτητες βιβλιοθηκες, μας δίνει άμεση επισκόπηση των Warnings / Errors του κώδικα καθώς και εύκολη διασύνδεση με τον μικροελεγκτή.

Οι βιβλιοθήκες που χρησιμοποιήσαμε είναι οι απαραίτητες για τη λειτουργία των επιμέρους περιφερειακών. Επιπλέον, χρησιμοποιήσαμε τη βιβλιοθήκη *SoftwareSerial.h* η οποία μας δίνει την ευκαιρία να έχουμε ένα επιπλέον κανάλι σειριακής επικοινωνίας. Χρειαζόμαστε δύο σειριακά κανάλια, ένα για το GSM modem και ένα για την επικοινωνία του μικροελεγκτή με τον ηλεκτρονικό υπολογιστή.

3.3. Επεξήγηση Κώδικα

Σε αυτό το σημείο, θα αναλύσουμε βήμα βήμα τα επιμέρους κομμάτια κώδικα (code blocks) της κατασκευής και θα περιγράψουμε την λειτουργία του καθενός από αυτά.
Όλος ο κώδικάς της εφαρμογής είναι ελεύθερα διαθέσιμος στην πλατφόρμα GitHub στη διεύθυνση: https://github.com/smyrnakis/beeHiveSystem

Μέρος 1ο:

Για να μπορέσει ο compiler να αναγνωρίσει τις εντολές για κάθε ένα από τα επιμέρους modules της κατασκευής, πρέπει να “εισάγουμε” στον κώδικα τα κατάλληλα αρχεία που περιγράφουν τις ιδιότητες και τις μεθόδους του κάθε module. Αυτό γίνεται με την εντολή **#include** ακολουθούμενη από το όνομα του αρχείου που εισάγουμε.

Με τη χρήση της εντολής **#define** δίνουμε “ψευδώνυμο” στα pin του μικροελεγκτή. Δεν χρειάζεται λοιπόν να θυμόμαστε ότι το κόκκινο LED έχει συνδεθεί στο pin 5. Απλά χρησιμοποιούμε τη λέξη *REDLED* όπου θέλουμε ο μικροελεγκτής να καταλάβει “pin 5”.

Μέρος 2ο:

Σε αυτό το σημείο κάνουμε δήλωση των μεταβλητών και των σταθερών που θα χρησιμοποιήσουμε στον κώδικά μας. Πέρα από τις γενικές μεταβλητές/σταθερές, δηλώνουμε επίσης την διεύθυνση του εξυπηρετητή της πλατφόρμας ThingSpeak, το API key (ιδιωτικό κλειδί που μας επιτρέπει να ανεβάζουμε δεδομένα στο ThingSpeak κανάλι μας), καθώς επίσης και το SSID / Password του WiFi hotspot που επιθυμούμε να συνδεθεί η κατασκευή μας.

Μέρος 3ο:

Αυτές οι γραμμές κώδικα δηλώνουν στον μικροελεγκτή ότι πρέπει να αρχικοποιήσει τα τρία βασικά modules:

- Αισθητήρα DHT, ο οποίος συνδέεται στο pin *DHTPIN*, είναι τύπου *DHT11*.
- Γέφυρα αισθητήρα βάρους HX711, με σύνδεση δεδομένων στο pin *HX711_DAT* και σύνδεση ρολογιού (clock) στο pin *HX711_CLK*.
- Ενσωματωμένο WiFi module της πλακέτας ESP12-E.

Μέρος 4ο:

Η συνάρτηση setup() χρησιμοποιείται για αρχικοποίηση και προετοιμασία του μικροελεγκτή. Εδώ δηλώνουμε τις εισόδους / εξόδους, το baud rate του σειριακού ελεγκτή κ.α. Ακολουθεί η επεξήγηση.

Η πρώτη εντολή **gprsMode = false;** είναι αρχικοποίηση της boolean μεταβλητής *gprsMode*. Όταν η τιμή της είναι *false* ο μικροελεγκτής προσπαθεί να συνδεθεί στο δίκτυο μέσω WiFi. Όταν η τιμή είναι *true*, η σύνδεση στο διαδίκτυο γίνεται μέσω GPRS.

Με τη χρήση της εντολής **pinmode** ορίζουμε αν τα pin του μικροελεγκτή είναι είσοδοι ή έξοδοι. Γράφοντας λοιπόν **pinmode(REDLED, OUTPUT)** στην ουσία λέμε στο μικροελεγκτή πως το pin 5 (=REDLED) θα είναι έξοδος. Άρα, αργότερα στον κώδικα θα μπορούμε να γράψουμε ψηφιακές τιμές (0 ή 1) σε αυτό το pin.

Η εντολή **digitalWrite** χρησιμοποιείται για να ορίζουμε την έξοδο του μικροελεγκτή σε ένα pin το οποίο έχει προηγουμένως οριστεί ως έξοδος. Οι τιμές που μπορούμε να γράψουμε είναι δύο: LOW (0 ή αλλιώς ground) και HIGH (1 ή συνήθως 5 volt).
Σε αυτό το σημείο σβήνουμε όλα τα LED γράφοντας την τιμή LOW σε κάθε pin.

Ακολουθούν τέσσερις πιο ειδικές εντολές:
- Serial.begin(115200);
- dht.begin();
- scale.setscale(-101800);
- scale.tare();

Η πρώτη εντολή ενεργοποιεί το σειριακό module του μικροελεγκτή και το αρχικοποιεί στο 115200 baud rate. Από αυτό το σημείο, ο μικροελεγκτής μπορεί να επικοινωνήσει μέσω της σειριακής διεπαφής με τον υπολογιστή που είναι συνδεδεμένος.
Αμέσως μετά, ενεργοποιούμε τον έλεγχο του αισθητήρα DHT. Ο αισθητήρας είναι έτοιμος και ο μικροελεγκτής μπορεί τώρα να διαβάσει τις τιμές που μετράει.
Οι δύο επόμενες εντολές ρυθμίζουν τη ζυγαριά. Πρώτα αρχικοποιούμε το module με το νούμερο -101800. Αυτό το νούμερο δε δηλώνει κάτι συγκεκριμένο. Είναι η απόκλιση που μπορεί να έχει το κάθε αισθητήριο λόγω διαφορών στην διαδικασία κατασκευής. Το νούμερο το βρήκαμε πειραματικά και ελέγχοντας το αποτέλεσμα ζύγισης γνωστών βάρων.
Τέλος, μηδενίζουμε τη ζυγαριά με τη χρήση της μεθόδου *tare*.

Σε διάφορα σημεία του κώδικα υπάρχει η εντολή **delay();** με όρισμα έναν ακέραιο αριθμό. Η εντολή αυτή εισάγει παύση σε milliseconds ίση με την τιμή του ορίσματος.

Μέρος 5ο:

Έχουμε φτάσει στη συνάρτηση **loop()**. Ο κώδικας που βρίσκεται εντός αυτής της συνάρτησης επαναλαμβάνεται συνεχώς για όσο χρόνο ο μικροελεγκτής βρίσκεται σε λειτουργία.

Η loop() κάνει κατά σειρά:

- έλεγχος για εισερχόμενα SMS
- δράση πάνω στο κείμενο του SMS

  Αν το μήνυμα ζητάει απάντηση μέσω SMS, η σχετική συνάρτηση καλείται επιτόπου

  Αν το κείμενο του μηνύματος είναι αποδεκτό, σε αυτό το σημείο ορίζουμε τις κατάλληλες μεταβλητές που θα ορίσουν τις μετέπειτα ενέργειες

- Ακολουθεί έλεγχος του ρυθμού αποστολής δεδομένων (εφόσων το γραπτό μήνυμα έχει ορίσει αυτή τη μεταβλητή) και κλήση των συναρτήσεων ελέγχου αισθητήρων και αποστολής δεδομένων

Μέρος 6ο:

Μετά το τέλος της loop() έχουμε τις συναρτήσεις που εκτελούν επιμέρους βήματα / εργασίες. Πιο αναλυτικά:

- Κλήση αισθητήρων και αποθήκευση των τιμών. Ταυτόχρονα, τα δεδομένα οργανώνονται στην πρόταση που αποστέλλεται στον παραγωγό με γραπτό μήνυμα (εφόσον αυτό ζητηθεί).
- Έλεγχος και ανάγνωση μηνύματος κειμένου. Η συνάρτηση επιστρέφει την τιμή επανάληψης (ρυθμός αποστολής δεδομένων στην πλατφόρμα Thing Speak). Τελικά, το εισερχόμενο διαγράφεται για να μην γεμίσουμε τη μνήμη της κάρτας SIM.
- Αποστολή γραπτού μηνύματος SMS: το μήνυμα αποστέλλεται στον τηλεφωνικό αριθμό που το αιτήθηκε.
- Ακολουθούν οι συναρτήσεις αποστολής δεδομένων προς την πλατφόρμα Thing Speak. Οι συναρτήσεις είναι δύο, μια για χρήση με WiFi και μια για χρήση του δικτύου κινητής τηλεφωνίας.
- Τέλος, μια μικρή αλλά ιδιαίτερα χρήσιμη συνάρτηση: όταν υπάρχουν δεδομένα στο Software Serial, τα ανακατευθύνει στο Hardware Serial ώστε να μπορούν να προβληθούν στον ηλεκτρονικό υπολογιστή.

Μέρος 7ο:

Ένα από τα σημαντικότερα κομμάτια της εργασίας είναι η σύνδεση στο διαδίκτυο μέσω WiFi. Για να αποφύγουμε να έχουμε τα στοιχεία του WiFi εγγεγραμμένα στον κώδικα (hard coded), καταφύγαμε στη χρήση της βιβλιοθήκης WiFiManager. Αν το όνομα δικτύου και ο κωδικός είναι μόνιμα εγγεγραμμένα στον κώδικα, με κάθε αλλαγή ρυθμίσεων Wifi (ή αλλαγή χώρου της συσκευής) θα χρειαζόταν να γίνει εκ νέου προγραμματισμός της κατασκευής με τα νέα στοιχεία σύνδεσης.

Η βιβλιοθήκη WiFiManager παίρνει σαν ορίσματα τις αρχικές τιμές που θέλουμε να χρησιμοποιήσει ο μικροελεγκτής κατά την εκκίνηση του, όταν θα δημιουργήσει ένα φορητό σημείο πρόσβασης. Έπειτα, συνδεόμαστε σε αυτό το φορητό σημείο πρόσβασης από οποιαδήποτε συσκευή έχει δυνατότητα σύνδεσης σε WiFi, χρησιμοποιώντας το SSID και password που δώσαμε για προεπιλογή.

Μας εμφανίζεται τότε ένα ιδιαίτερα λιτό και εύχρηστο μενού όπου μπορούμε να ζητήσουμε από τον μικροελεγκτή να συνδεθεί σε ένα δίκτυο WiFi που είναι διαθέσιμο κοντά μας. Ο μικρότερης μας παρουσιάζει τη λίστα με τα διαθέσιμα δίκτυα που είναι στο εύρος του και κατόπιν επιλέγουμε το επιθυμητό και πληκτρολογούμε τον κωδικό του.

Οι ρυθμίσεις αυτές παραμένουν στη μνήμη του μικρότερη, συνεπώς δεν χρειάζεται να ακολουθήσουμε την ίδια διαδικασία για σύνδεση σε γνωστό δίκτυο.

Αφού πληκτρολογήσουμε τον κωδικό του δικτύου WiFi, ο μικροελεγκτής σταματάει την λειτουργία hotspot και προσπαθεί να συνδεθεί στο δίκτυο επιλέξαμε.

Μέρος 8ο:

Αφού ολοκληρωθεί η σύνδεση στο διαδίκτυο, γίνεται η αρχικοποίηση της μεταβλητής **gprsMode**. Αν η συσκευή έχει συνδεθεί σε δίκτυο WiFi, τότε η τιμή είναι “ψευδής” και ο κώδικας συνεχίζει χρησιμοποιώντας το δίκτυο WiFi (εντολή **WiFi.begin** με ορίσματα το όνομα του δικτύου “ssid” και τον κωδικό του). Αν η επικοινωνία πρόκειται να γίνει μέσω GPRS, τότε δεν χρειάζεται να κάνουμε κάτι σε αυτό το βήμα.

Σημείωση:

Όλο το **μέρος #7** (διαχείριση σύνδεσης δικτύου WiFi) διεκπεραιώνεται από έναν δεύτερο (βοηθητικό) μικροελεγκτή. Η χρήση ενός δευτερευόντως μικροελεγκτή ελαφρύνει κατά πολύ τις διεργασίες που έχει να εκτελέσει ο κύριος μικροελεγκτής και συνεπώς δεν δεσμεύει πόρους χρήσιμους για την ανάλυση των τιμών των αισθητήρων.

Η επικοινωνία των δύο μικροελεγκτών γίνεται σειριακά: με χρήση μόνο 3 καλωδίων ενώνουμε τους ακροδέκτες Rx και Tx μεταξύ τους (χιαστί) και γεφυρώνουμε τις 2 γειώσεις.

Ο βοηθητικός μικροελεγκτής είναι βασισμένος στο ολοκληρωμένο κύκλωμα ESP8266, έναν ελεγκτή με μνήμη 1MB και επιπλέον, ενσωματωμένο ελεγκτή WiFi και κεραίας. Το ESP8266 διαβάζει την εισερχόμενη σειριακή κίνηση και αναλύει τα δεδομένα. Αν η εισερχόμενη "λέξη" περιέχει τρεις αριθμούς χωρισμένους με τον χαρακτήρα **"&"**, τότε ξέρει ότι πρόκειται για τις τιμές *θερμοκρασίας*, *υγρασίας* και *βάρους* αντίστοιχα. Καλώντας την κατάλληλη συνάρτηση, ανεβάζει τις τιμές στην πλατφόρμα IoT ThingSpeak και απαντάει για acknowledgement στον κύριο μικροελεγκτή.

Σε περίπτωση που διακοπεί η σύνδεση WiFi, ο μικροελεγκτής κάνει reset έτσι ώστε να προσπαθήσει εκ νέου να συνδεθεί σε κάποιο από τα αποθηκευμένα δίκτυα ή (εάν αυτό αποτύχει) να μπει σε λειτουργία σημείου πρόσβασης (Access Point) περιμένοντας από τον χρήστη νέες παραμέτρους.

3.4. Μελλοντικές αναβαθμίσεις

Επεκτασιμότητα: η κατασκευή λειτουργεί αυτόνομα ως μονάδα. Πιθανή αναβάθμιση στο λογισμικό και στο υλικό της κατασκευής, θα επέτρεπε την παράλληλη και συνεργατική λειτουργία περισσότερων από μία μονάδες, σε περίπτωση που ο παραγωγός το επιθυμούσε και αυτό με ιδιαίτερα χαμηλό κόστος.

Ασφάλεια: οποιοσδήποτε γνωρίζει το νούμερο της κάρτας SIM της συσκευής, έχει δυνατότητα να στείλει μηνύματα ελέγχου και να ενημερωθεί για την τρέχουσα κατάσταση του μελισσιού μέσω SMS. Με αναβάθμιση του κώδικα, υπάρχει η δυνατότητα να ορίσουμε είτε κωδικό ελέγχου (master password) είτε λίστα επιτρεπόμενων αριθμών τηλεφώνου.

Ο κωδικός ελέγχου θα είναι απαραίτητος για να απαντήσει η συσκευή στην εντολή ελέγχου. Για παράδειγμα, η εντολή <<report>> θα λειτουργήσει μόνο αν ο αποστολέας την συνοδεύσει με τον κύριο κωδικό: <<report 123456789>>.

Η λίστα επιτρεπόμενων αριθμών θα λειτουργεί ως φίλτρο και η συσκευή θα απαντά μόνο σε αριθμούς που περιέχονται στη λίστα.

4. Παράρτημα

4.1. Παραδείγματα απάντησης γραπτού κειμένου (SMS)

Ορθό μήνυμα:

Temp: 23 °C
Hum: 38 %
Weight: 78 kg

Μήνυμα όπου απέτυχε η ανάγνωση του βάρους:

Temp: 23 °C
Hum: 38 %

Μήνυμα όπου απέτυχε η ανάγνωση όλων των αισθητήρων:

Error reading sensors!

4.2. Κώδικας

Ο κώδικας της πτυχιακής μας εργασίας είναι διαθέσιμος στο GitHub, στην παρακάτω διεύθυνση: https://github.com/smyrnakis/beeHiveSystem/blob/uno/beeHive.ino

```
#include <SoftwareSerial.h>

#include <GPRS_Shield_Arduino.h>
#include <sim900.h>
#include <HX711.h>
#include <Wire.h>
#include <DHT.h>

#include "secrets.h"


// ~~~ PIN declaration
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#define PCBLED 13
#define ESPLED 13

// #define ANLG_IN A0
#define DHTPIN 5

#define HX711_CLK 2
#define HX711_DAT 3

#define PIN_TX_GSM 7        // 5        // yellow cable
#define PIN_RX_GSM 8        // 6        // green cable
#define PIN_TX_ESP 11
#define PIN_RX_ESP 12


// ~~~ Variables - constants
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#define BAUDRATE 9600

// SIM connection info
// 0,1      --> connected
// 0,2      --> not connected, searching
// 0,4      --> unknown connection state
// 0,5      --> connected, roaming
#define SIM900checkNetReg "AT+CREG?"

// Signal quality in dB: [0-31] (higher: better)
#define SIM900checkSignal "AT+CSQ"

// SIM card number
#define SIM900simInfo "AT+CCID"

// Check if modem is ready
#define SIM900isReady "AT+CPIN?"

// Board info
#define SIM900boardInfo "ATI"

// Check if internet is connected
#define SIM900internet "AT+COPS?"

// Operators available in the network
#define SIM900operators "AT+COPS=?"

// Check battery level (2nd num: bat % , 3rd num: voltage in
mV)
#define SIM900battery "AT+CBC"

// List UNREAD SMS
#define SIM900unreadSMS "AT+CMGL=\"REC UNREAD\""

// List READ SMS
#define SIM900readSMS "AT+CMGL=\"REC READ\""

// Delete ALL READ SMSs
#define SIM900delRead "AT+CMGD=1,1"

// Delete ALL SMSs
#define SIM900delAll "AT+CMGD=1,4"


const char* thingSpeakServer  = "api.thingspeak.com";
            // 184.106.153.149
char apiKey[] = THINGSP_WR_APIKEY;
                                                // API key w/ write
access

int temperature = 0;
int humidity = 0;
float weight = 0.0;

unsigned int uploadInterval   = 0;
unsigned long currentMillis   = 0;
unsigned long startMillisTEMP = 0;
unsigned long startMillisDeb  = 0;
unsigned long startMillisInte = 0;
const unsigned int smsInterv  = 10000;
const unsigned int seconds30  = 30000;
const unsigned int seconds45  = 45000;
const unsigned int seconds90  = 90000;
const unsigned int seconds120 = 120000;

int SMS_command = 0;
            // After incoming SMS message

char SMS_phone[16];
char SMS_datetime[24];
#define MESSAGE_LENGTH 160
            // SMS charachter limit                    // int
SMS_messageLength = 160;
char SMS_message[MESSAGE_LENGTH];          //
Incoming SMS
```

```
int messageIndex = 0;
        // Defined in the readSMS() func

bool allowSMS           = false;        //
Debounce for SMS send
bool gprsMode           = false;        // True if
no WiFi connection
bool printInSerial      = true;         // Printing
in HW serial

const char* smsReport           = "report";
                        // --> reply back with SMS
const char* smsUpload           = "upload";
                        // --> upload
instantly - once
const char* smsUpload30     = "auto30";
                        // --> upload every 30 seconds
const char* smsUpload45     = "auto45";
                        // --> upload every 45 seconds
const char* smsUpload90     = "auto90";
                        // --> upload every 90 seconds
const char* smsUpload120    = "auto120";
                // --> upload every 120 seconds
const char* smsUploadCancel = "autocancel";
                // --> cancel auto upload

char beeHiveMessage;
                                        //
Contents of outgoing SMS message
String dataToESP;
                                        // Content
of data sent to ESP
String dataToSMS;
                                        // String
with outgoing SMS text

String cmd;
String inboundSerialESP;
                                    // Used for
communication across devices
String inboundSerialGSM;
String outboundSerialESP;
String outboundSerialGSM;


// ~~~ Initialising
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
DHT dht(DHTPIN, DHT11);
HX711 scale;

SoftwareSerial mySerialGSM(PIN_TX_GSM,PIN_RX_GSM);
SoftwareSerial mySerialESP(PIN_TX_ESP,PIN_RX_ESP);

GPRS gprs(PIN_TX_GSM,PIN_RX_GSM,BAUDRATE);


// ~~~ Initializing
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void setup() {
        pinMode(DHTPIN, INPUT);
        pinMode(PCBLED, OUTPUT);        // setting
I/O
        pinMode(ESPLED, OUTPUT);

        digitalWrite(PCBLED, HIGH);     // turning
LEDs OFF
        digitalWrite(ESPLED, HIGH);

        Serial.begin(BAUDRATE);
        // starting serial
        Serial.println("Serial enabled.\n\r");
        delay(100);

        short gprsInitTimeout = 30;
        Serial.print("Initialising GPRS...");
        gprs.init();

        while((!gprs.checkPowerUp()) && (gprsInitTimeout
> 0)) {
                delay(1000);
                gprsInitTimeout--;
                Serial.print(".");
                gprs.init();
        }
        if (gprs.checkPowerUp()) {
                Serial.println(" done\n\r");
        }
        else {
                Serial.println(" failed\n\r");
        }
        delay(100);

        // Serial.print("Initialising Software Serials ...");
        // mySerialGSM.begin(BAUDRATE);
        // mySerialESP.begin(BAUDRATE);
        // Serial.println(" done\n\r");
        // delay(100);

        Serial.print("Initialising DHT ...");
        dht.begin();
        Serial.println(" done\n\r");
        delay(100);

        Serial.print("Initialising scale ...");
        scale.begin(HX711_DAT, HX711_CLK);
        scale.set_scale(-101800);
        scale.tare();
        Serial.println(" done\n\r");
        delay(100);

        Serial.print("Configuring SIM900 ...");
        mySerialGSM.begin(BAUDRATE);
        delay(5);
        // Inform for new SMS w/ index number (default)
        // mySerialGSM.println("AT+CNMI=2,1,0,0,0");
        // Forward new SMS to Serial monitor
        mySerialGSM.println("AT+CNMI=2,2,0,0,0");
        mySerialGSM.end();
        Serial.println(" done\n\r");
        delay(100);
```

```
	mySerialGSM.flush();
	Serial.flush();
}


// ~~~ Main loop
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void loop() {
	currentMillis = millis();

	// Send to serial whatever SIM900 says
	mySerialGSM.begin(BAUDRATE);
	delay(10);
	mySerialGSM.listen();
	if (mySerialGSM.available()) {
		// String readString;
		while (mySerialGSM.available()) {
			inboundSerialGSM =
mySerialGSM.readString();
			// readString =
mySerialGSM.readString();
			// inboundSerialGSM =
mySerialGSM.readStringUntil('\r\n');
		}
		Serial.println("GSM >>>> ");
		Serial.println(inboundSerialGSM);
		// Serial.println(readString);
		// inboundSerialGSM = readString;
		// <?<>?<?><?><?><?><?><?><?>
	}
	else {
		inboundSerialGSM = "\r\n\0";
	}
	mySerialGSM.end();

	// Send to serial whatever ESP8266 says
	mySerialESP.begin(BAUDRATE);
	delay(10);
	mySerialESP.listen();
	if (mySerialESP.available()) {
		// String readString;
		while (mySerialESP.available()) {
			inboundSerialESP =
mySerialESP.readString();
			// readString =
mySerialESP.readString();
			// inboundSerialESP =
mySerialESP.readStringUntil('\r\n');
		}
		Serial.println("ESP >>>> ");
		Serial.println(inboundSerialESP);
		// Serial.println(readString);
		// if
((String(inboundSerialESP)).indexOf('report') > 0) {
		//	Serial.println("Report
requested!\r\n");
		// }
	}
	else {
		inboundSerialESP = "\r\n\0";
	}
	mySerialESP.end();

	// Send to devices whatever we send in serial
	// delay(10);
	if (Serial.available()) {
		delay(10);
		// String cmd = "";
		cmd = "";
		while (Serial.available()) {
			cmd += (char)Serial.read();
		}
		Serial.println();
	Serial.print(">>>> ");
	Serial.println(cmd);

		if (cmd.indexOf('AT') > 0) {

	mySerialGSM.begin(BAUDRATE);
			mySerialGSM.print(cmd);
			mySerialGSM.end();
		}
		else {

	mySerialESP.begin(BAUDRATE);
			mySerialESP.print(cmd);
			mySerialESP.end();
		}
	}


	if ((inboundSerialGSM.indexOf("+3069XXXXXXX")
>= 0) ||

(inboundSerialGSM.indexOf("+30697XXXXXXX") >= 0)) {
		Serial.println("User requested a report
by SMS!\r\n");
		SMS_command = readSMS();
	}
	else {
		SMS_command = -1;
	}

	if (cmd.indexOf(smsUpload) >= 0) {
		getMeasurements();

		dataToESP = String(temperature);
		dataToESP += "&";
		dataToESP += String(humidity);
		dataToESP += "&";
		dataToESP += String(weight);
		dataToESP += "\r\n";

		mySerialESP.print(dataToESP);

		cmd = "";
	}

	if (cmd.indexOf(smsReport) >= 0) {

		SMS_command = 1000;
```

```
		cmd = "";
	}

	if (cmd.indexOf(smsUploadCancel) >= 0) {

		SMS_command = 0;

		cmd = "";
	}

	if (cmd.indexOf(smsUpload30) >= 0) {

		SMS_command = 30;

		cmd = "";
	}

	if (cmd.indexOf(smsUpload45) >= 0) {

		SMS_command = 45;

		cmd = "";
	}

	if (cmd.indexOf(smsUpload90) >= 0) {

		SMS_command = 90;

		cmd = "";
	}

	if (cmd.indexOf(smsUpload120) >= 0) {

		SMS_command = 120;

		cmd = "";
	}


	switch (SMS_command) {
		case -2:
			// Invalid SMS content
			;
		break;
		case -1:
			// No unread SMS
			;
		break;
		case 0:
				// Auto upload
canceled
			uploadInterval = 0;
		break;
		case 1:
				// Upload data once
			uploadInterval = 1;
		break;
		case 30:
			// Upload every 30 seconds
			uploadInterval = seconds30;
		break;
		case 45:
			// Upload every 45 seconds
			uploadInterval = seconds45;
		break;
		case 90:
			// Upload every 90 seconds
			uploadInterval = seconds90;
		break;
		case 120:
			// Upload every 120 seconds
			uploadInterval = seconds120;
		break;
		case 1000:
				// Reply with SMS
			if (allowSMS) {
				allowSMS = false;
				getMeasurements();
				sendSMS();
			}
		break;
		default:
			// Invalid return code
			Serial.println("WARNING:
unexpected readSMS() reply!");
		break;
	}


	if (uploadInterval == 1) {
		// Reset uploadInterval if request was to
upload once
		uploadInterval = 0;

		Serial.println("Uploading once
(WiFi)...\r\n");

		getMeasurements();
		if (!gprsMode)		// Sending data
using WiFi
		{
			dataToESP =
String(temperature);
			dataToESP += "&";
			dataToESP +=
String(humidity);
			dataToESP += "&";
			dataToESP += String(weight);
			dataToESP += "\r\n";


	mySerialESP.print(dataToESP);
			// Serial.println(dataToESP);
		} else
		{
			// Send2ThingSpeakGPRS();
			Serial.println("Uploading once
(GPRS) ...\r\n");
		}

	}
	else if (
```

```
                (uploadInterval != 0) &&
                (currentMillis - startMillisInte >=
uploadInterval)
        ) {

                Serial.println("Recurring upload (WiFi)
...\r\n");

                getMeasurements();
                if (!gprsMode)          // Sending data
using WiFi
                {
                        dataToESP =
String(temperature);
                        dataToESP += "&";
                        dataToESP +=
String(humidity);
                        dataToESP += "&";
                        dataToESP += String(weight);
                        dataToESP += "\r\n";


        mySerialESP.print(dataToESP);
                        // Serial.println(dataToESP);
                } else
                {
                        // Send2ThingSpeakGPRS();
                        Serial.println("Recurring
upload (GPRS) ...\r\n");
                }

                startMillisInte = currentMillis;
        }


        // Debounce every 10 sec
        if (currentMillis - startMillisDeb >= 10000) {
                allowSMS = true;
                // Serial.println("Debounce reset");
                startMillisDeb = currentMillis;
        }

        // delay(1);
}


// ~~~ Getting sensor data
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void getMeasurements() {
        digitalWrite(PCBLED, HIGH);

        // read values
        temperature = dht.readTemperature();
        delay(50);
        humidity = dht.readHumidity();
        delay(50);
        weight = fabs(scale.get_units(10));
        // weight = abs(scale.get_units(10));
        delay(50);

        // check values - build SMS text
        if (isnan(temperature)) {
                Serial.println("Failed to read
temperature sensor!");
                temperature = -100;
        }
        else {
                Serial.print("Temperature: ");
                Serial.print(temperature);
                Serial.println(" °C");
        }

        if (isnan(humidity)) {
                Serial.println("Failed to read humidity
sensor!");
                humidity = -100;
        }
        else {
                Serial.print("Humidity: ");
                Serial.print(humidity);
                Serial.println(" %");
        }

        if (isnan(weight)) {
                Serial.println("Failed to read weight
sensor!");
                weight = -100;
        }
        else {
                Serial.print("Weight: ");
                Serial.print(weight);
                Serial.println(" kg");
        }

        // When no sensor data
        if ((temperature == -100) && (humidity == -100)
&& (weight == -100)) {
                Serial.println("Error reading sensors!");
        }

        digitalWrite(PCBLED, LOW);
}


// ~~~ Checking / Reading SMS
~~~~~~~~~~~~~~~~~~~~~~~~~~~
int readSMS() {
        digitalWrite(ESPLED, HIGH);
        int returnValue = -2;

        if (inboundSerialGSM.indexOf(smsReport) >= 0) {
                Serial.println("Requested SMS report
...");
                returnValue = 1000;
        }
        else if (inboundSerialGSM.indexOf(smsUpload) >=
0) {
                Serial.println("Requested to upload once
...");
                returnValue = 1;
        }
```

```
        else if (inboundSerialGSM.indexOf(smsUpload30)
>= 0) {
                Serial.println("Requested to upload
every 30 seconds ...");
                returnValue = 30;
        }
        else if (inboundSerialGSM.indexOf(smsUpload45)
>= 0) {
                Serial.println("Requested to upload
every 45 seconds ...");
                returnValue = 45;
        }
        else if (inboundSerialGSM.indexOf(smsUpload90)
>= 0) {
                Serial.println("Requested to upload
every 90 seconds ...");
                returnValue = 90;
        }
        else if
(inboundSerialGSM.indexOf(smsUpload120) >= 0) {
                Serial.println("Requested to upload
every 120 seconds ...");
                returnValue = 120;
        }
        else if
(inboundSerialGSM.indexOf(smsUploadCancel) >= 0) {
                Serial.println("Requested to cancel auto
upload.");
                returnValue = 0;
        }
        else {
                Serial.println("Invalid SMS text.");
                returnValue = -2;
        }

        // delay(1000);
        mySerialGSM.println(SIM900delAll);

        digitalWrite(ESPLED, LOW);
        return returnValue;
}


// ~~~ Sending SMS
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void sendSMS() {
        Serial.println("Sending SMS message ...");

        digitalWrite(ESPLED, HIGH);

        dataToSMS = "Temp: ";
        dataToSMS += String(temperature);
        dataToSMS += "C\r\n";
        dataToSMS += "Hum: ";
        dataToSMS += String(humidity);
        dataToSMS += "%\r\n";
        dataToSMS += "Wei: ";
        dataToSMS += String(weight);
        dataToSMS += "kg\r\n";
        // dataToSMS += "\0";

        // Configuring TEXT mode
        mySerialGSM.print("AT+CMGF=1\r");
        delay(100);
        // while(mySerialGSM.available()) {
        //      Serial.write(mySerialGSM.read());
        // }

        // mySerialGSM.println("AT + CMGS =
\"+30695XXXXXXX\"");
        // mySerialGSM.println("AT + CMGS =
\"+30697XXXXXXX\"");
        mySerialGSM.println("AT + CMGS =
\"+306944553090\"");

        delay(100);
        // mySerialGSM.print("AT+CMGS=\"");
        // // mySerialGSM.print(String(SMS_phone));
        // // mySerialGSM.print("+30697XXXXXXX");
        // mySerialGSM.print("+30695XXXXXXX");
        // mySerialGSM.println("\"");
        //
mySerialGSM.println("AT+CMGS=\"+30697XXXXXXX\"");
        // while(mySerialGSM.available()) {
        //      Serial.write(mySerialGSM.read());
        // }

        // SMS content
        // mySerialGSM.print(beeHiveMessage);
        mySerialGSM.print(dataToSMS);
        delay(100);
        // while(mySerialGSM.available()) {
        //      Serial.write(mySerialGSM.read());
        // }

        // Ctrl+Z character
        // mySerialGSM.println((char)26);
        mySerialGSM.write("26");
        // mySerialGSM.print("\x1A");
        // mySerialGSM.println( 0x1a );
        delay(100);
        mySerialGSM.println();
        delay(5000);

        // delay(1000);
        mySerialGSM.println(SIM900delAll);

        Serial.println("SMS sent.");

        digitalWrite(ESPLED, LOW);
}


// ~~~ Thingspeak GPRS
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void Send2ThingSpeakGPRS() {
        digitalWrite(ESPLED, LOW);

        mySerialGSM.println("AT+CBAND=\"EGSM_DCS
_MODE\"");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(200);
```

```
        //mySerialGSM.println("AT+IPR=9600");
        //if (printInSerial) { ShowSerialDataGSM(); }
        //delay(100);

        mySerialGSM.println("AT");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(200);

        mySerialGSM.println("AT+CREG?");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(200);

        //Set the connection type to GPRS
        mySerialGSM.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(500);

        //APN Vodafone: 'internet.vodafone.gr'. APN Cosmote: 'internet', APN Q: myq
        // https://wiki.apnchanger.org/Greece
        mySerialGSM.println("AT+SAPBR=3,1,\"APN\",\"myq\"");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(500);

        //Enable the GPRS
        mySerialGSM.println("AT+SAPBR=1,1");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(3000);

        //Query if the connection is setup properly, if we get back a IP address then we can proceed
        mySerialGSM.println("AT+SAPBR=2,1");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(500);

        //We were allocated a IP address and now we can proceed by enabling the HTTP mode
        mySerialGSM.println("AT+HTTPINIT");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(500);

        //Start by setting up the HTTP bearer profile identifier
        mySerialGSM.println("AT+HTTPPARA=\"CID\",1");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(500);

        //Setting up the url to the 'thingspeak.com' address
        String tempCall;
        tempCall = "AT+HTTPPARA=\"URL\",\"http://api.thingspeak.com/update?api_key=";
        tempCall += String(apiKey);
        tempCall += "&field1=";
        tempCall += String(temperature);
        tempCall += "&field2=";
        tempCall += String(humidity);
        tempCall += "&field3=";
        tempCall += String(weight);
        tempCall += "\"";
        mySerialGSM.println(tempCall);
        //mySerialGSM.println("AT+HTTPPARA=\"URL\",\"http://api.thingspeak.com/update?api_key=THINGSP_WR_APIKEY&field1=22&field2=15&field3=10\"");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(500);

        //Start the HTTP GET session
        mySerialGSM.println("AT+HTTPACTION=0");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(500);

        //end of data sending
        mySerialGSM.println("AT+HTTPREAD");
        // if (printInSerial) { ShowSerialDataGSM(); }
        delay(100);
        digitalWrite(ESPLED, HIGH);
}
```

4.3. Φωτογραφίες