ΑΝΑΦΟΡΑ PROJECT

ΠΙΠΕΡΑΚΗΣ ΓΙΩΡΓΟΣ 2018030012
ΛΑΜΠΡΙΝΙΔΗΣ ΙΩΑΝΝΗΣ 2018030075

Σκοπος

Ο Σκοπός της εργασίας ήταν η δημιουργία ενός ολοκληρωμένου προγράμματος στον μικροελεγκτή avr και η εξοικείωση σε έναν καλό βαθμό με τη σχεδιαστική λογική ενός συστήματος.

Υλοποιηση Διεπαφης

Για την υλοποίηση της διεπαφής χρησιμοποιήθηκε μια σειριακή θύρα όπου δέχεται εντολές και δεδομένα εισόδου σε μορφή ascii για την δημιουργία ενός sudoku και επιστρέφει πάλι σε μορφή ascii τον πίνακα συμπληρωμένο. Πιο αναλυτικά δημιουργήθηκαν 9 εντολές:

	Εντολή από PC ή απάντηση από το PC(ASCII)	Ενέργεια Εντολής	Απάντηση προς PC (ASCII) ή εντολή απο AVR προς PC
1)	AT<CR><LF>	Απλή Απάντηση OK μετά το , καμμία άλλη ενέργεια	OK<CR><LF>
2)	C<CR><LF>	Καθαρισμός δεδομένων πίνακα (αρχικοποίηση κελιών στο 0) και οθόνης LED (όλα τα LED σβυσμένα) και απάντηση OK μετά το , ισοδύναμο με αρχικοποίηση παιχνιδιού	OK<CR><LF>
3)	N<X><Y><VALUE><CR><LF>	Τιμές κελιών, απάντηση από τον AVR OK μετά το X,Y,VALUE: [1-9] χαρακτήρας ASCII που όλοι αντιστοιχούν σε δεκαδικό ψηφίο	OK<CR><LF>
4)	P<CR><LF>	Play New Game (ξεκινάμε να λύνουμε τον πίνακα, δεν υπάρχουν άλλα δεδομένα).	OK<CR><LF>
5)	S<CR><LF>	Από AVR: Done (τελείωσε η επεξεργασία, ο AVR είναι έτοιμος να αρχίσει να αποστέλλει αποτελέσματα), η	D<CR><LF>

		απάντηση S από το PC σημαίνει «στείλε το πρώτο αποτέλεσμα κελιού»	
6)	T<CR><LF>	Τιμές κελιών, απάντηση T από το PC μετά το κάθε μηνύματος ώστε να ξεκινήσει ο AVR να στέλνει το επόμενο. X,Y,VALUE: [1-9] χαρακτήρας ASCII που όλοι αντιστοιχούν σε δεκαδικό ψηφίο	N<X><Y><VALUE><CR><LF>
7)	OK<CR><LF>	Done sending results από AVR, η απάντηση OK από PC σημαίνει ότι τα έλαβα όλα	D<CR><LF>
8)	B<CR><LF>	Break – Σταμάτησε τους υπολογισμούς ή την μετάδοση OK αποτελεσμάτων αλλά μην αρχικοποιήσεις τους πίνακες ακόμη – κάτι σαν warm start της εφαρμογής (π.χ. για παρατήρηση προόδου αν φαίνεται ότι «κρέμασε» η εφαρμογή)	OK<CR><LF>
9)	D<CR><LF>	Debug – Στείλε μου τα περιεχόμενα του κελιού (X,Y). Η απάντηση N περιλαμβάνει τις συντεταγμένες και την τιμή του κελιού (ΠΡΟΧΟΧΗ: η τιμή μπορεί να περιλαμβάνει και το 0 αν δεν έχει ακόμη καθοριστεί η τιμή του κελιού)	N<X><Y><VALUE><CR><LF>

Εντολές Εισόδου:

Οι εντολές 1 εως 4 είναι εντολές που στέλνει το pc για την δημιουργία του προβλήματος sudoku με την εντολή C για καθαρισμό την εντολή N για την συμπλήρωση ενός κελιού στον πίνακα και την εντολή P για να ενημερώσει οτι δεν υπάρχουν άλλα δεδομένα και αρχίσει να λύνει ο το πρόγραμμα το sudoku.

Εντολές Απάντηση:

Οι εντολές 5 εως 7 είναι εντολές απάντησης του pc απο εντολή που στέλνει η σειριακή θύρα. Δηλαδή η εντολή S θα σταλθεί αφού δεχτεί ο υπολογιστής το μήνυμα του μικροεπεξεργαστή D<CR><LF> το οποίο υποδηλώνει οτι ο AVR έχει τελειώσει την επεξεργασία το sudoku και είναι έτοιμος να αρχίσει να στέλνει το λυμένο πίνακα. Ο AVR μόλις δεχτεί την εντολή S στέλνει το πρώτο στοιχείο του λυμένου πίνακα. Η εντολή T είναι η απάντηση του pc απο την εντολή N<X><Y><VALUE><CR><LF> και ειδοποιεί τον μικροεπεξεργαστή οτι δέχτηκε το στοιχείο που του έστειλε έτσι ώστε να αρχίσει να στέλνει το επόμενο στοιχείο. Τέλος η εντολή OK στέλνετε ως απάντηση απο την εντολή D όπου υποδηλώνει ότι ο AVR έχει στείλει ολόκληρο τον πίνακα και ο υπολογιστής τον έχει δεχτεί.

Εντολές Αποσφαλματωσης:

Οι εντολές 8 και 9 έχουν δημιουργηθεί για σκοπούς debugging και για εξέταση της ορθότητας των υπολογισμών του AVR. Η εντολή B <<παγώνει>> όλους τους υπολογισμούς και της αποστολές του AVR ενώ η εντολή D στέλνει στοιχεία του πίνακα στο pc.

Flags:

Για την υλοποίηση του προγράμματος χρησιμοποιήθηκαν 2 σημαντικά flags.

```
int choose_command = 0;      //choose which format of command we want
const char ok_command[4] PROGMEM = {0x4f,0x4b,0x0d,0x0a};    //1
const char done_command[3] PROGMEM = {0x44,0x0d,0x0a};       //2
char value_command[6] = {0x4e,0x10,0x10,0x10,0x0d,0x0a};     //3
```

Το flag choose_command έλυσε το πρόβλημα για την επιλογή ποιού μηνύματος πρέπει να στείλει ο AVR μετά από κάθε εντολή. Παραπάνω μπορείτε να δείτε τα μηνύματα του μικρο-επεξεργαστή και δίπλα την τιμή του flag για αυτήν την περίπτωση. Μόλις έρθει το interrupt data empty register της σειριακής θύρας (το interrupt που στέλνει byte byte το μήνυμα προς τον υπολογιστή) ελέγχει το flag choose_command και στέλνει το αντίστοιχο μήνυμα.

```
int control_flag = 0;    //0 initial
                         //1 solving
                         //2 break
                         //3 done
```

Ακόμα ένα σημαντικό flag που χρησιμοποιήθηκε είναι το control_flag. Το control_flag πρόκειται για το flag που <<κρατάει>> την κατάσταση του πίνακα sudoku. Είναι χρήσιμο για την υλοποίηση πολλών εντολών όπως της P της B και πολλών άλλων. Παραπάνω βλέπεται τον ορισμό αλλά και της τιμές που έχουν αποδοθεί για κάθε κατάσταση.

Μνήμη:

Στην Ram έχουμε αποθηκεύσει όλα τα δεδομένα του πίνακα sudoku την εντολή εισόδου απο το pc και άλλες global μεταβλητές για την διαχείριση και την επίλυση του πίνακα. Στην flash μνήμη αποθηκεύονται τα μηνύματα που στέλνει ο avr προς το pc εκτός του μηνύματος N<X><Y><VALUE><CR><LF> καθώς δεν είναι κάτι στατικό αλλά αλλάζει.

Υλοποίηση LED:

Για την υλοποίηση της ένδειξης προόδου της επίλυσης μέσω των LED χρησιμοποιήθηκε ο timer1 και ενεργοποιείται το overflow interrupt κάθε ένα δευτερόλεπτο. Για την ένδειξη του LED εμφανίζεται το ακέραιο πηλίκο του πλήθους των συμπληρωμένων κελιών του πίνακα με το 10. Η τιμή του χρόνου ανανέωσης επιλέχθηκε αυθαίρετα με την υπόθεση ότι δεν επιθυμούμε να είναι ένα χρονικό διάστημα αρκετά μικρό ώστε να διακόπτει συνέχεια το πρόγραμμα αλλά ούτε αρκετά μεγάλο ώστε να υπάρχουν μεγάλα κενά στην ένδειξη για την πρόοδο του προγράμματος.

MAIN:

Η main του προγράμματος είναι πολύ λιτή. Καλείται η συνάρτηση init για αρχικοποίηση όλων των δομών του προγράμματος και η συνάρτηση που θα λύνει το sudoku. Η ρουτίνα για την επίλυση του πίνακα sudoku είναι στην κύρια ροή του προγράμματος και όχι σε interrupt.

Αλγόριθμος:

Η επίλυση του sudoku πραγματοποιείται με έναν απλό backtracking αλγόριθμο. Ο λόγος που επιλέχθηκε αυτός ο αλγόριθμος είναι διότι το space complexity του δεν υπερβαίνει το μέγεθος της μνήμης. Άλλοι αλγόριθμοι που εξετάστηκαν ήταν ο αλγόριθμος X του Knuth, που είναι αρκετά γρήγορος αλλά το απαιτεί αρκετά περισσότερη μνήμη από αυτή που έχουμε.

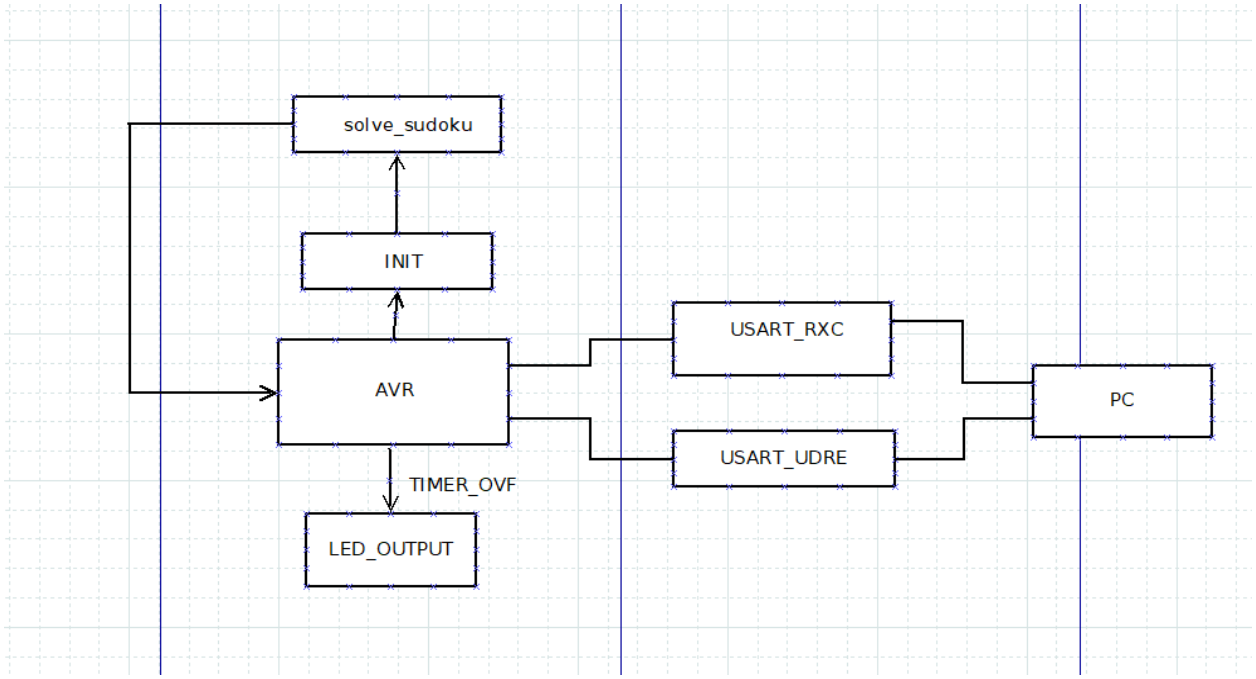
Ο αλγόριθμος backtracking είναι ένα είδος bruteforce search. Παρόλο που είναι αρκετά αργός σε σχέση με άλλους αλγόριθμους, είναι σίγουρο πως θα λύσει οποιοδήποτε επιλύσιμο sudoku του δοθεί.

Η λογική του αλγόριθμου που υλοποιήθηκε είναι η εξής:
Άμα υπάρχει κενή θέση στο sudoku τότε θα βάλει έναν αριθμό, θα ελέγξει αν είναι έγκυρη εκχώρηση και και αν είναι τότε η συνάρτηση θα ξανακαλέσει τον εαυτό της για την επόμενη εκχώρηση. Αν είναι λανθασμένη τότε θα αδειάσει το κελί και θα δοκιμάσει την επόμενη τιμή.
Ο αλγόριθμος τερματίζει μόλις γεμίσουν όλα τα κελιά.

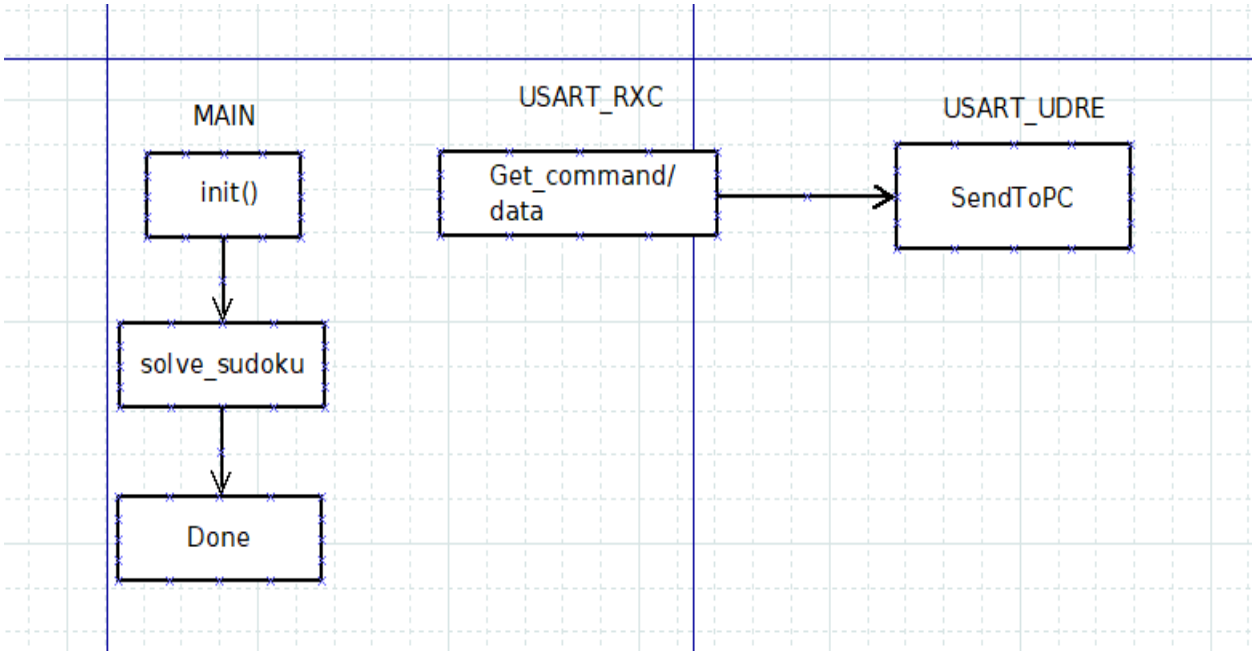
Προβλήματα:

Το σημαντικότερο πρόβλημα που αντιμετωπίσαμε ήταν ένα καμένο ATmega16 το οποίο μας πήρε ώρες debugging μέχρι που το αλλάξαμε. Επίσης για κάποιο λόγο δε μπορούσαμε να προγραμματίσουμε τον μικροελεγκτή με το πρόγραμμα που υλοποιήσαμε. Αυτό έχει ως αποτέλεσμα να έχει γίνει έλεγχος του προγράμματος μόνο με τον debugger του Microchip Studio.

Simulation board:



Σχήμα 1: Block Diagram



Σχήμα 2: Basic Flowchart