

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

3 Η ΑΣΚΗΣΗ

07/05/2021

**Παπαβασιλείου Μάριος 1059649**

**Ντάκος Γεώργιος 1059569**

**Ανάλυση κώδικα:**

Στην συγκεκριμένη άσκηση μας ζητείται να φτιάξουμε μια έξυπνη οικιακή συσκευή η οποία θα κάνει τις εξής κινήσεις:

1. Κινείται ευθεία
2. Στριβεί δεξιά αν δεν έχει τοίχο στα δεξιά του
3. Στριβεί αριστερά αν βρει τοίχο μπροστά του

Τις παραπάνω κινήσεις της κάνει όταν βρίσκεται σε normal mode αν αλλάξουμε σε reverse mode τότε στριβεί 180 μοίρες και κάνεις τις παρακάτω κινήσεις:

1. Κινείται ευθεία
2. Στριβεί αριστερά αν δεν έχει τοίχο στα αριστερά
3. Στριβεί δεξιά αν βρει τοίχο μπροστά του

Το πρόγραμμα μας στην ουσία εξομοιώνει τις παραπάνω κινήσεις για το συγκεκριμένο δωμάτιο που μας δίνει η άσκηση. Έχουμε 2 μεταβλητές corners και reverse οπού η πρώτη είναι ένας counter που μετράει τις γωνίες που έχουμε περάσει οπότε αν corners = 8 το πρόγραμμα τερματίζει διότι η συσκευή έχει ολοκληρώσει όλη την περίμετρο του δωμάτιου. Τώρα αν έχουμε reverse = 1 μπαίνουμε σε reverse mode και κάνει τις ανάλογες κινήσεις μόνο που τώρα οι γωνίες μειώνονται και όταν φτάσει το corners = -1 τότε σημαίνει ότι η συσκευή έφτασε στην αρχική της θέση οπού και τερματίζει το πρόγραμμα. Έχουμε χρησιμοποιήσει ένα ADC 10-bit resolution για την άσκηση με low threshold = 10 δηλαδή αν RES < 10 τότε σημάνει ότι έχουμε τοίχο μπροστά και η συσκευή μας κάνει τις ανάλογες κινήσεις ανάλογα με την τιμή της μεταβλητής reverse(normal ή reverse mode) . Στην ουσία το RES είναι ένας αισθητήρας και στην συγκεκριμένη περίπτωση ένα ποτενσιόμετρο. Επειδή δεν λειτουργούμε πάνω στην πλακέτα αλλά σε simulator αυτής τότε κάθε φορά είμαστε εμείς υπεύθυνοι για την αλλαγή τιμής του RES(δηλαδή γίνεται χειροκίνητα). Οπότε όσο έχουμε RES > 10 τότε η συσκευή μας κινείται ευθεία οπού αυτό μας το δηλώνει το PIN1 του PORTD όπως φαίνεται και στην Εικόνα 1. Tώρα για τα για τις κινήσεις να στριβεί δεξιά σε normal mode και αριστερά σε reverse mode και να μπαίνει σε reverse mode χρησιμοποιούμε 2 switch interrupts. Το πρώτο για να στριβεί το περνούμε από το 5o bit των flags του PORTF ενώ το 2ο switch από το PORTC πάλι στο 5ο bit των flags του. Οπότε όταν ενεργοποιούμε το 5ο bit του PORTF στο πεδίο των INTFLAGS η συσκευή μας καταλαβαίνει στα δεξιά του δεν έχει τοίχο οπότε στριβεί δεξιά ενώ αν είναι σε reverse mode δεν έχει τοίχο στα αριστερά οπότε και στριβεί αριστερά. Το αριστερό στρίψιμο δηλώνεται από το PIN2 ενώ το στρίψιμο δεξιά από το PIN0. Αυτό μπορούμε να το δούμε και από τις Εικόνα 2 και Εικόνα 3. Τώρα με το switch του PORTC όταν το ενεργοποιήσουμε η συσκευή μας αρχίζει να στριβεί 180 μοίρες οπότε και μπαίνει και σε reverse mode. Η στροφή 180 μοίρες δηλώνεται όταν ανάβουν και τα 3 PIN(PIN0,PIN1,PIN02). Αυτό φαίνεται και στην Εικόνα 4. Όλες οι στροφές δεξιά, αριστερά ή 180 μοίρες πραγματοποιούνται σύμφωνα με έναν TIMER που έχουμε υλοποιήσει οπότε όσο τρέχει ο TIMER η συσκευή μας στριβεί και όταν σταματήσει ο TIMER τότε και η συσκευή μας συνεχίζει να προχωράει ευθεία. Όλες οι κινήσεις βρίσκονται μέσα σε συναρτήσεις για καλύτερη ευ ανάγνωση του κώδικα και για πιο βέλτιστα αποτελέσματα. Για να δούμε τις κινήσεις και να καταλάβουμε πως λειτουργεί ο κώδικας η χρήση των breakpoint σε κάθε συνάρτηση είναι απαραίτητη.

**BONUS 1:**

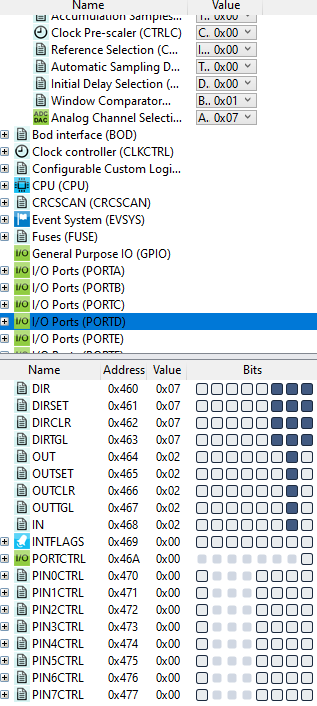
Ένας τρόπος που θα μπορούσαμε να κάνουμε την συσκευή μας να καταλάβει ότι έχει φτάσει στην αρχική της θέση χωρίς να ξέρουμε τη διαρρύθμιση του δωμάτιο/χώρου είναι η εξής λογική: Κάθε φορά που θα στριβεί δεξιά θα αυξάνουμε μια μεταβλητή κατά 1 ενώ κάθε φορά που θα στριβεί αριστερά θα μειώνουμε την ιδιά μεταβλητή κατά 1 οπότε έχοντας ολοκληρώσει ένα κύκλο δηλαδή 360 μοίρες που σημαίνει ότι έχει ολοκληρώσει όλο το δωμάτιο και έχει φτάσει στην αρχική του θέση. Αυτό σημαίνει ότι η μεταβλητή που έχουμε θα έχει πάρει την τιμή -4. Τα ιδιά ισχύουν και το reverse mode μόνο που αντί για -4 η μεταβλητή θα έχει την τιμή +4.Στο πρόγραμμα μας αυτή η μεταβλητή είναι το rotations και οπότε στριβεί αριστερά ή δεξιά κάνει τις ανάλογες πράξεις και ανανεώνουμε την νέα τιμή του rotations

**BONUS 2:**

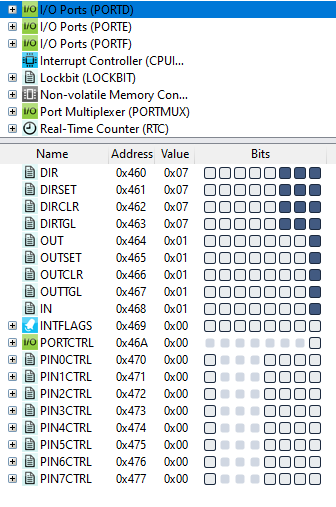
O παραπάνω τρόπος που σκεφτήκαμε είναι αρκετά κάλος και μπορεί να χρησιμοποιηθεί σε αρκετά δωμάτια. Από την άλλη βέβαια σε κάποια δωμάτια μπορεί να εμφανίσει προβλήματα, δηλαδή η μεταβλητή να έχει φτάσει -4 ή +4 αντίστοιχα και να μην έχει ολοκληρώσει το περίγραμμα όλου του δωμάτιου. Αυτό γίνεται αν μέσα στο δωμάτιο υπάρχει υποχωράς του δωμάτιου που για να πραγματοποιήσουμε το περίγραμμα του η συσκευή κάνει αναγκαστικά κύκλο οπότε η μεταβλητή φτάνει την τιμή -4 η +4 αντίστοιχα και στην ουσία δεν έχει τελειώσει το περίγραμμα όλου του δωμάτιου. Ένα τέτοιο παράδειγμα δωμάτιου φαίνεται στην Εικόνα 5.

**BONUS 3:**

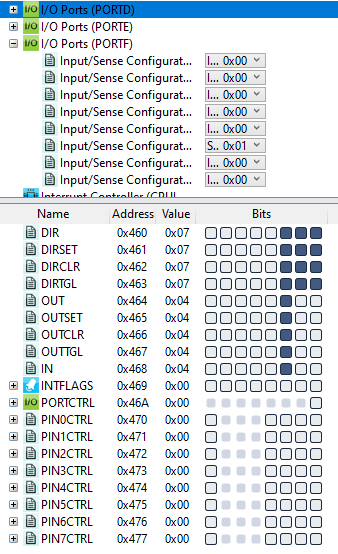
Ο κώδικας φαίνεται μέσα από το πρόγραμμα στην ουσία με του που φτάσουμε rotation = -4 τότε σημαίνει ότι έχουμε φτάσει στην αρχική θέση οπότε ικανοποιεί την συνθήκη if μέσα στην while οπότε και κάνει break και βγαίνει από την while και το πρόγραμμα μας τερματίζει. Η εξήγηση της σκέψης φαίνεται στην απάντηση του ερωτήματος BONUS 1.



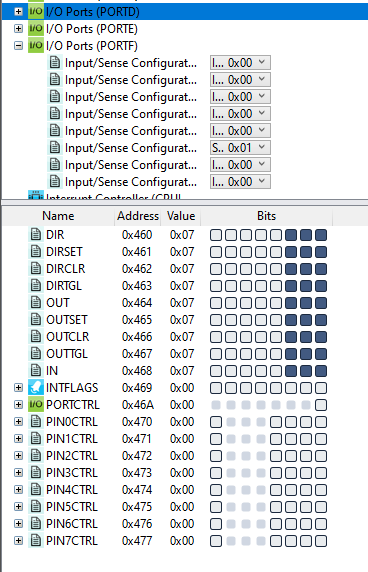
Εικόνα 1



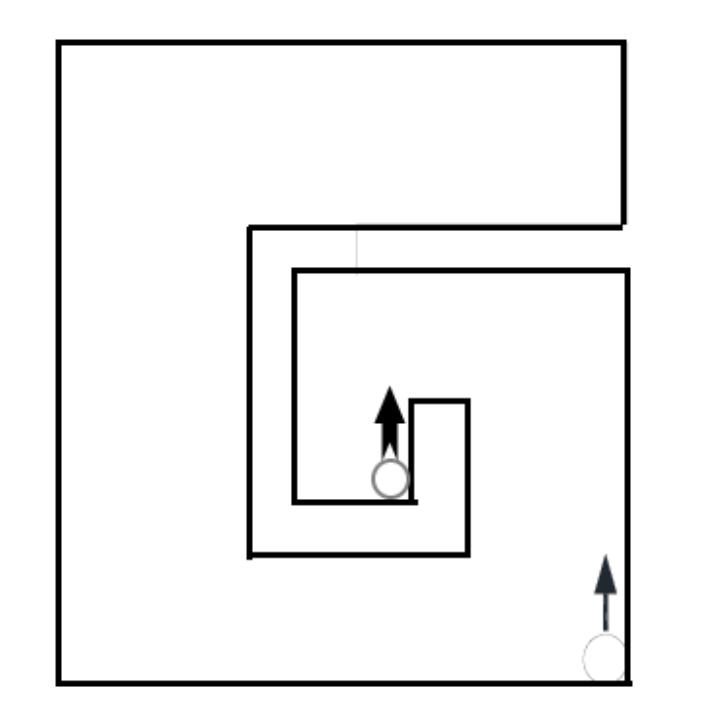
Εικόνα 2



Εικόνα 3



Εικόνα 4



Εικόνα

**Κώδικας Προγράμματος:**

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

int rotations=0; //-1 for left turn and +1 for right turn | When this reaches -4 it means it completed a 360 degree turn

int corners=0; //Number of corners passed

int reverse=0; //0: Normal mode 1: Reverse mode

/\*

RES Value < 10 ==> Wall up front !!!!! MONO TA 8 DEKSIA BIT PIANEI TO RES !!!!!

PORTF intflags pin 5 ==> No wall on right (normal mode) or no wall on left (reverse mode)

PORTC intflags pin 5 ==> Begin reverse mode

Ta breakpoints einai ypoxrewtika!!!

Kathe fora pou mpainei stin routina interrupt gia toixo mprosta (RES<10) prepei xeirokinita

na allazoume thn timh tou res pali se ipsili timi (Amesws me to pou mpei sto breakpoint!!!),

enw sta alla interrupts den xreiazetai, ta flags kanoun clear mona tous.

\*/

int main(void)

{

//initialize the ADC for Free-Running mode

ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc; //10-bit resolution

ADC0.CTRLA |= ADC\_FREERUN\_bm; //Free-Running mode enabled

ADC0.CTRLA |= ADC\_ENABLE\_bm; //Enable ADC

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //Analog input

//Enable Debug Mode

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm;

//Window Comparator Mode

ADC0.WINLT |= 10; //Set low threshold at 10

ADC0.INTCTRL |= ADC\_WCMP\_bm; //Enable Interrupts for WCM

ADC0.CTRLE |= ADC\_WINCM0\_bm; //Interrupt when RESULT < WINLT (RES<10)

//LED Setup

PORTD.DIR |=PIN0\_bm; // Turning right led

PORTD.DIR |=PIN1\_bm; // Moving straight led

PORTD.DIR |=PIN2\_bm; // Turning left led

//Switches Setup

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; // No wall on right or left (interrupt)

PORTC.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; // Reverse mode switch (interrupt)

//Timer Setup

TCA0.SINGLE.CNT = 0; // Clear counter

TCA0.SINGLE.CTRLB = 0; // Normal Mode

TCA0.SINGLE.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc; // Clock Frequency/1024

sei(); //Enable interrupts

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start ADC Conversion

PORTD.OUTSET= PIN1\_bm; //Start walking straight

while(corners>-1) //When in reverse mode, if corners=-1 we reached the starting point.

{

if (rotations==-4) break; //When in normal mode, rotations=-4 means (in most cases) we reached the starting point.

}

}

ISR(ADC0\_WCOMP\_vect) //Interrupt for wall up front (RES Value<10)

{

if (reverse==0) //Turning left routine (Normal mode)

{

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

turnleft();

corners++;

}

else if (reverse==1 && corners==0) //No need to turn right or left when you are on the starting point

{

int intflags=ADC0.INTFLAGS;

ADC0.INTFLAGS=intflags;

corners--;

}

else if (reverse==1) //Turning right routine (Reverse mode)

{

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

turnright();

corners--;

}

}

ISR(PORTF\_PORT\_vect) //Interrupt for no wall on right or left depending on mode (PORTF pin 5 intflags)

{

if (reverse==0) //No wall on right so turn right (Normal mode)

{

int intflags=PORTF.INTFLAGS;

PORTF.INTFLAGS=intflags;

turnright();

corners++;

}

else if (reverse==1 && corners==0) //No need to turn right or left when you are on the starting point

{

int intflags=PORTF.INTFLAGS;

PORTF.INTFLAGS=intflags;

corners--;

}

else if (reverse==1) //No wall on left so turn left (Reverse mode)

{

int intflags=PORTF.INTFLAGS;

PORTF.INTFLAGS=intflags;

turnleft();

corners--;

}

}

ISR(PORTC\_PORT\_vect) //Interrupt to begin Reverse mode (PORTE pin 5 intflags)

{

int intflags=PORTC.INTFLAGS;

PORTC.INTFLAGS=intflags;

reverse=1;

//Start turning 180 degrees

PORTD.OUTSET= PIN0\_bm;

PORTD.OUTSET= PIN2\_bm;

\_delay\_ms(1);

begintimer();

//Stop turning 180 degrees and start walking straight again

PORTD.OUTCLR= PIN0\_bm;

PORTD.OUTCLR= PIN2\_bm;

\_delay\_ms(1);

}

begintimer() //Timer for LEDs (Turning right,left and 180 degrees)

{

//Oi times tou timer einai oti xeirotero ston simulator

TCA0.SINGLE.CTRLA |=1; //Start timer

while (TCA0.SINGLE.CNT<1){} //Stop program flow until the counter reaches the desired value

TCA0.SINGLE.CTRLA = 0; // Stop Timer

TCA0.SINGLE.CNT=0; // Zeroing counter

\_delay\_ms(2); //To stop the program from progressing too fast (Not needed normally)

}

turnleft() //Function to turn left

{

PORTD.OUTCLR= PIN1\_bm; //Stop walking straight

\_delay\_ms(1);

PORTD.OUTSET= PIN2\_bm; //Turn left

begintimer();

PORTD.OUTCLR |= PIN2\_bm; //Stop turning left

\_delay\_ms(1);

PORTD.OUTSET= PIN1\_bm; //Start walking straight again

\_delay\_ms(1);

rotations--;

}

turnright() //Function to turn right

{

PORTD.OUTCLR= PIN1\_bm; //Stop walking straight

\_delay\_ms(1);

PORTD.OUTSET= PIN0\_bm; //Turn right

begintimer();

PORTD.OUTCLR |= PIN0\_bm; //Stop turning right

\_delay\_ms(1);

PORTD.OUTSET= PIN1\_bm; //Start walking straight again

\_delay\_ms(1);

rotations++;

}