**ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΗΓΜΕΝΟΙ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ**

**Εργαστηριακή Άσκηση 2**

Δούρου Βασιλική Ευαγγελία – Α.Μ: 1072633

Πουρής Βασίλειος – Α.Μ.: 1072485

**1.** Κατά τη συγγραφή των παρακάτω έγιναν οι ακόλουθες παραδοχές:

* Στο δεύτερο και στο τρίτο ερώτημα, αν η οικιακή συσκευή συναντήσει εμπόδιο με τον πλαϊνό αισθητήρα, στρίβει και σταματάει να κινείται. Έπειτα, αρχίζει πάλι να κινείται με την αρχική λειτουργία του και να δέχεται τιμές από τον πλαϊνό αισθητήρα.
* Στο τρίτο ερώτημα, όταν πατηθεί ο διακόπτης για την ανάποδη πορεία και η συσκευή επιστρέψει στην αρχική της θέση, το πρόγραμμα ολοκληρώνεται.

Ο κώδικας του πρώτου ερωτήματος της άσκησης για τη κίνηση της οικιακής συσκευής σε τετράγωνο δωμάτιο είναι ο ακόλουθος:

#include <avr/io.h>

#include <avr/interrupt.h>

#define left 4

int x=0;

int y=0;

int corners=4;

int main(){

PORTD.DIR |= PIN1\_bm; //PIN1 is output

PORTD.DIR |= PIN2\_bm; //PIN2 is output

while (corners>0){

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on

PORTD.OUT |= PIN2\_bm; //PIN2 is off

//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; //The bit

//Enable Debug Mode

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm;

//Window Comparator Mode

ADC0.WINLT |= 4; //Set threshold

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

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

sei();

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

while (y==0) {

}

PORTD.OUT |= PIN1\_bm; //PIN1 is off

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=left;

TCA0.SINGLE.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while (x==0) {

}

x=0;

cli();

}

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.CTRLA = 0; //Disable ADC

ADC0.INTFLAGS = intflags;

y=1;

}

ISR(TCA0\_CMP0\_vect){

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

corners--;

x=1;

y=0;

}

Ο κώδικας του δεύτερου ερωτήματος της άσκησης για τη κίνηση της οικιακής συσκευής σε τυχαίο δωμάτιο είναι ο παρακάτω:

#include <avr/io.h>

#include <avr/interrupt.h>

#define left 4

#define right 4

#define forwardsensor 13 //afou T=2sec apo ton tupo value=T\*ftimer=2sec\*19,531KHz=39062, alla gia thn prosomoiwsh xrhsimopoioume mikrotero xrono

#define sidesensor 12 //afou T=3sec apo ton tupo value=T\*ftimer=3sec\*19,531KHz=58593, alla gia thn prosomoiwsh xrhsimopoioume mikrotero xrono

int x=0; //gia to interrupt tou timer

int y=0; //gia to interrupt tou adc

int lturns=0;

int rturns=0;

int corners=0;

int main(){

PORTD.DIR |= PIN0\_bm; //PIN0 is output for right

PORTD.DIR |= PIN1\_bm; //PIN1 is output for forward

PORTD.DIR |= PIN2\_bm; //PIN2 is output for left

corners=lturns-rturns;

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //The bit

//Enable Debug Mode

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm;

//Window Comparator Mode

ADC0.WINLT |= 4; //Set threshold

ADC0.WINHT |= 4; //Set threshold

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

TCA0.SINGLE.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc;

while (corners<4){

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on

PORTD.OUT |= PIN2\_bm; //PIN2 is off

PORTD.OUT |= PIN0\_bm; //PIN0 is off

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=sidesensor;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

sei();

while(x==0){

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.CTRLE = 0b00000010; //Interrupt when RESULT > WINHT

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

while ((y==0)&&(x==0)) {

}

if(x==0){

rturns++;

PORTD.OUT |= PIN1\_bm; //PIN1 is off

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=right;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while (x==0) {

}

}

}

cli();

x=0;

if(y==0){

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=forwardsensor;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

ADC0.CTRLA = 0;

sei();

while(x==0){

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

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

ADC0.CTRLE = 0b00000001; //Interrupt when RESULT < WINLT

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

while ((y==0)&&(x==0)) {

}

if(x==0){

lturns++;

PORTD.OUT |= PIN1\_bm; //PIN1 is off

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=left;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while (x==0) {

}

}

}

cli();

x=0;

}

y=0;

corners=lturns-rturns;

}

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.CTRLA = 0; //Disable ADC

ADC0.INTFLAGS = intflags;

y=1;

}

ISR(TCA0\_CMP0\_vect){

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

x=1;

}

Ο κώδικας του τρίτου ερωτήματος της άσκησης για την ανάποδη λειτουργία είναι ο παρακάτω:

#include <avr/io.h>

#include <avr/interrupt.h>

#define left 4

#define right 4

#define forwardsensor 13 //afou T=2sec apo ton tupo value=T\*ftimer=2sec\*19,531KHz=39062, alla gia thn prosomoiwsh xrhsimopoioume mikrotero xrono

#define sidesensor 12 //afou T=3sec apo ton tupo value=T\*ftimer=3sec\*19,531KHz=58593, alla gia thn prosomoiwsh xrhsimopoioume mikrotero xrono

#define all 6

int x=0; //gia to interrupt tou timer

int y=0; //gia to interrupt tou adc

int z=0; //gia to interrupt tou button

int lturns=0;

int rturns=0;

int corners=0;

int totalturns=0;

int main(){

PORTD.DIR |= PIN0\_bm; //PIN0 is output for right

PORTD.DIR |= PIN1\_bm; //PIN1 is output for forward

PORTD.DIR |= PIN2\_bm; //PIN2 is output for left

corners=lturns-rturns;

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

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //The bit

//Enable Debug Mode

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm;

//Window Comparator Mode

ADC0.WINLT |= 4; //Set threshold

ADC0.WINHT |= 4; //Set threshold

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

TCA0.SINGLE.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc;

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;

sei();

while ((corners<4)&&(z==0)){

if(z==0){

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on

PORTD.OUT |= PIN2\_bm; //PIN2 is off

PORTD.OUT |= PIN0\_bm; //PIN0 is off

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=sidesensor;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

}

while((x==0)&&(z==0)){

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.CTRLE = 0b00000010; //Interrupt when RESULT > WINHT

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

while ((y==0)&&(x==0)&&(z==0)) {

}

if((x==0)&&(y==1)){

rturns++;

totalturns++;

PORTD.OUT |= PIN1\_bm; //PIN1 is off

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=right;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while (x==0) {

}

}

}

x=0;

if((y==0)&&(z==0)){

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=forwardsensor;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

ADC0.CTRLA = 0;

while((x==0)&&(z==0)){

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

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

ADC0.CTRLE = 0b00000001; //Interrupt when RESULT < WINLT

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

while ((y==0)&&(x==0)&&(z==0)) {

}

if((x==0)&&(y==1)){

lturns++;

totalturns++;

PORTD.OUT |= PIN1\_bm; //PIN1 is off

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=left;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while (x==0) {

}

}

}

x=0;

}

y=0;

corners=lturns-rturns;

}

//edw ginetai anastrofh

if(z==1){

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=all;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while(x==0){

}

x=0;

while (totalturns>0){

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on

PORTD.OUT |= PIN2\_bm; //PIN2 is off

PORTD.OUT |= PIN0\_bm; //PIN0 is off

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=sidesensor;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while(x==0){

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.CTRLE = 0b00000010; //Interrupt when RESULT > WINHT

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

while ((y==0)&&(x==0)) {

}

if(x==0){

totalturns--;

PORTD.OUT |= PIN1\_bm; //PIN1 is off

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=left;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while (x==0) {

}

}

}

x=0;

if(y==0){

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=forwardsensor;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

ADC0.CTRLA = 0;

while(x==0){

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

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

ADC0.CTRLE = 0b00000001; //Interrupt when RESULT < WINLT

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

while ((y==0)&&(x==0)) {

}

if(x==0){

totalturns--;

PORTD.OUT |= PIN1\_bm; //PIN1 is off

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=right;

TCA0.SINGLE.CTRLA |=1;

TCA0.SINGLE.INTCTRL =TCA\_SINGLE\_CMP0\_bm; // INTERRUPT ENABLE

while (x==0) {

}

}

}

x=0;

}

y=0;

}

corners=0;

z=0;

}

cli();

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.CTRLA = 0; //Disable ADC

ADC0.INTFLAGS = intflags;

y=1;

}

ISR(TCA0\_CMP0\_vect){

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

x=1;

}

ISR(PORTF\_PORT\_vect){

//clear the interrupt flag

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

z=1;

}

**2.** Το διάγραμμα ροής του πρώτου ερωτήματος είναι το εξής:

Diagram

Description automatically generated

Αρχικά, ορίζουμε τις δύο μεταβλητές x και y, οι οποίες αρχικοποιούνται με μηδέν, και χρησιμοποιούνται ως flags για το πότε έχει περάσει ο χρόνος του timer και πότε έχει έρθει τιμή μικρότερη του threshold στον ADC. Επιπλέον, ορίζουμε και τη μεταβλητή corners, η οποία αρχικοποιείται με 4 και υπολογίζει πόσες στροφές έχει κάνει η οικιακή συσκευή. Επίσης, ορίζουμε το PIN1 και το PIN2 του PORTD ως έξοδο.

Όσο, η οικιακή συσκευή δεν έχει κάνει ακόμη 4 στροφές, δηλαδή όσο η μεταβλητή corners έχει τιμή μεγαλύτερη του 0 (αφού σε κάθε στροφή ελαττώνεται κατά 1), επαναλαμβάνονται τα ακόλουθα. Θέτουμε το PIN1 του PORTD ίσο με 0 για να ανάψει το LED που δηλώνει ότι η συσκευή κινείται στην ευθεία, και το PIN2 του PORTD ίσο με 1 για να κλείσει το LED που υποδηλώνει αριστερή στροφή. Έπειτα, αρχικοποιούμε τον ADC σε free-running mode, θέτουμε το threshold, ενεργοποιούμε τις διακοπές όταν ισχύει RES<WINLT, αρχίζουμε τη μετατροπή και περιμένουμε διακοπή. Όταν, προκληθεί διακοπή, θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags, η τιμή του y θα αλλάξει προκειμένου να βγούμε από το while loop. Έπειτα, θέτουμε το PIN1 ίσο με 1 για να σβήσει το LED της μπροστινής κίνησης και το PIN2 ίσο με 0 για να ανάψει το LED της αριστερής στροφής. Στη συνέχεια, γίνεται clear ο counter του TCA, θέτουμε τον TCA σε normal mode, θέτουμε την τιμή που θέλουμε να γίνει το interrupt, δηλαδή ο χρόνος που θέλουμε για την αριστερή στροφή, ορίζουμε τη τιμή του prescaler και κάνουμε enable τον timer και τα interrupts. Μετά, όσο ο timer δεν έχει φτάσει ακόμη στη τιμή που έχουμε ορίσει, θα μείνει το πρόγραμμα σε ένα while loop. Όταν ο timer φτάσει στην τιμή που θέλουμε, θα γίνει διακοπή και θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags, θα ελαττωθούν οι εναπομείναντες στροφές και η τιμή του x θα αλλάξει προκειμένου να βγούμε από το while loop.

Το διάγραμμα ροής του δεύτερου ερωτήματος:

Diagram, schematic

Description automatically generated

Αρχικά, ορίζουμε τις δύο μεταβλητές x και y, οι οποίες αρχικοποιούνται με μηδέν, και χρησιμοποιούνται ως flags για το πότε έχει περάσει ο χρόνος του timer και πότε έχει έρθει τιμή μικρότερη ή μεγαλύτερη του threshold στον ADC αντίστοιχα. Επιπλέον, ορίζουμε και τις μεταβλητές lturns, rturns και corners, οι οποίες αρχικοποιούνται με 0 και υπολογίζουν πόσες αριστερές και δεξιές στροφές έχει κάνει η οικιακή συσκευή και τη διαφορά lturns-rturns αντίστοιχα. Επίσης, ορίζουμε το PIN0, το PIN1 και το PIN2 του PORTD ως έξοδο.

Παράλληλα, θέτουμε τα threshold, ενεργοποιούμε τα interrupts για τον ADC και ορίζουμε τον prescaler του timer.

Όσο, η οικιακή συσκευή δεν έχει κάνει τον κύκλο του τυχαίου δωματίου, δηλαδή όσο η μεταβλητή corners έχει τιμή μικρότερη του 4 (αφού σε ένα δωμάτιο οι αριστερές στροφές μείον τις δεξιές είναι ίσες με 4), επαναλαμβάνονται τα ακόλουθα. Αρχικά, θέτουμε το PIN1 του PORTD ίσο με 0 για να ανάψει το LED που δηλώνει ότι η συσκευή κινείται στην ευθεία, και τα PIN2 και PIN0 του PORTD ίσα με 1 για να κλείσουν το LED που υποδηλώνουν αριστερή και δεξιά στροφή αντίστοιχα. Έπειτα, γίνεται clear ο counter του TCA, θέτουμε τον TCA σε normal mode, θέτουμε την τιμή που θέλουμε να γίνει το interrupt, δηλαδή ο χρόνος που θέλουμε να ελέγξουμε

τον πλαϊνό αισθητήρα και κάνουμε enable τον timer και τα interrupts. Μετά, όσο ο timer δεν έχει φτάσει ακόμη στη τιμή που έχουμε ορίσει, θα μείνει το πρόγραμμα σε ένα while loop. Στη διάρκεια του, αρχικοποιούμε τον ADC σε free-running mode, κάνουμε enable τον ADC, ενεργοποιούμε τις διακοπές όταν ισχύει RES>WINHT, αρχίζουμε τη μετατροπή και περιμένουμε διακοπή. Όταν, προκληθεί διακοπή, θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags και η τιμή του y θα αλλάξει προκειμένου να βγούμε από το while loop. Έπειτα, θέτουμε το PIN1 ίσο με 1 για να σβήσει το LED της μπροστινής κίνησης και το PIN0 ίσο με 0 για να ανάψει το LED της δεξιάς στροφής και αυξάνουμε κατά 1 τον μετρητή των δεξιών στροφών. Στη συνέχεια, γίνεται clear ο counter του TCA, θέτουμε τον TCA σε normal mode, θέτουμε την τιμή που θέλουμε να γίνει το interrupt, δηλαδή ο χρόνος που θέλουμε για την δεξιά στροφή και κάνουμε enable τον timer και τα interrupts. Μετά, όσο ο timer δεν έχει φτάσει ακόμη στη τιμή που έχουμε ορίσει, θα μείνει το πρόγραμμα σε ένα while loop. Όταν ο timer φτάσει στην τιμή που θέλουμε, θα γίνει διακοπή και θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags και η τιμή του x θα αλλάξει προκειμένου να βγούμε από το while loop. Αν δεν γίνει διακοπή από τον ADC όσο ελέγχουμε τον πλαϊνό αισθητήρα, όταν ο timer φτάσει στην τιμή που θέλουμε, θα γίνει διακοπή και θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags και η τιμή του x θα αλλάξει προκειμένου να βγούμε από το while loop. Έπειτα, γίνεται clear ο counter του TCA, θέτουμε τον TCA σε normal mode, θέτουμε την τιμή που θέλουμε να γίνει το interrupt, δηλαδή ο χρόνος που θέλουμε να ελέγξουμε τον μπροστινό αισθητήρα, κάνουμε enable τον timer και τα interrupts και καθαρίζουμε το CTRLA του ADC. Μετά, όσο ο timer δεν έχει φτάσει ακόμη στη τιμή που έχουμε ορίσει, θα μείνει το πρόγραμμα σε ένα while loop. Στη διάρκεια του, αρχικοποιούμε τον ADC σε single conversion mode, κάνουμε enable τον ADC, ενεργοποιούμε τις διακοπές όταν ισχύει RES<WINLT, αρχίζουμε τη μετατροπή και περιμένουμε διακοπή. Όταν, προκληθεί διακοπή, θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags και η τιμή του y θα αλλάξει προκειμένου να βγούμε από το while loop. Έπειτα, θέτουμε το PIN1 ίσο με 1 για να σβήσει το LED της μπροστινής κίνησης και το PIN2 ίσο με 0 για να ανάψει το LED της αριστερής στροφής και αυξάνουμε κατά 1 τον μετρητή των αριστερών στροφών. Στη συνέχεια, γίνεται clear ο counter του TCA, θέτουμε τον TCA σε normal mode, θέτουμε την τιμή που θέλουμε να γίνει το interrupt, δηλαδή ο χρόνος που θέλουμε για την αριστερή στροφή και κάνουμε enable τον timer και τα interrupts. Μετά, όσο ο timer δεν έχει φτάσει ακόμη στη τιμή που έχουμε ορίσει, θα μείνει το πρόγραμμα σε ένα while loop. Όταν ο timer φτάσει στην τιμή που θέλουμε, θα γίνει διακοπή και θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags και η τιμή του x θα αλλάξει προκειμένου να βγούμε από το while loop. Τέλος, υπολογίζεται η διαφορά αριστερών και δεξιών στροφών για να εξακριβώσουμε αν έχει ολοκληρωθεί το περίγραμμα του δωματίου.

Το διάγραμμα ροής του τρίτου ερωτήματος:

Diagram, schematic

Description automatically generated

Αρχικά, ορίζουμε τις τρεις μεταβλητές x, y και z, οι οποίες αρχικοποιούνται με μηδέν, και χρησιμοποιούνται ως flags για το πότε έχει περάσει ο χρόνος του timer, πότε έχει έρθει τιμή μικρότερη ή μεγαλύτερη του threshold στον ADC και πότε έχει πατηθεί το κουμπί της αντίστροφης πορείας αντίστοιχα. Επιπλέον, ορίζουμε και τις μεταβλητές lturns, rturns, corners και totalturns, οι οποίες αρχικοποιούνται με 0 και υπολογίζουν πόσες αριστερές και δεξιές στροφές έχει κάνει η οικιακή συσκευή, τη διαφορά lturns-rturns και τις συνολικές στροφές αντίστοιχα. Επίσης, ορίζουμε το PIN0, το PIN1 και το PIN2 του PORTD ως έξοδο.

Παράλληλα, θέτουμε τα threshold, ενεργοποιούμε τα interrupts για τον ADC, ορίζουμε τον prescaler του timer και κάνουμε enable το pullup και το interrupt για το PIN5 του PORTF, ώστε αυτό να χρησιμοποιηθεί ως το κουμπί της αναστροφής. Όσο δεν έχει πατηθεί το κουμπί της αντίστροφης πορείας, ο κώδικας λειτουργεί όπως στο δεύτερο ερώτημα. Αν πατηθεί το κουμπί, γίνεται διακοπή και καλείται η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags και η τιμή του z θα αλλάξει ώστε να βγούμε από το while loop. Τότε, θέτουμε τα PIN0, PIN1 και PIN2 ίσα με 0 ώστε να ανάψουν ταυτόχρονα και τα τρία LED για την μπροστινή, αριστερή και δεξιά κίνηση ταυτόχρονα. Στη συνέχεια, γίνεται clear ο counter του TCA, θέτουμε τον TCA σε normal mode, θέτουμε την τιμή που θέλουμε να γίνει το interrupt, δηλαδή ο χρόνος που θέλουμε για να είναι ταυτόχρονα αναμμένα και τα τρία LED και κάνουμε enable τον timer και τα interrupts. Μετά, όσο ο timer δεν έχει φτάσει ακόμη στη τιμή που έχουμε ορίσει, θα μείνει το πρόγραμμα σε ένα while loop. Όταν ο timer φτάσει στην τιμή που θέλουμε, θα γίνει διακοπή και θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags και η τιμή του x θα αλλάξει προκειμένου να βγούμε από το while loop. Τέλος, όσο το totalturns είναι μεγαλύτερο του 0, δηλαδή όσο δεν έχουμε φτάσει ακόμη στην αρχική θέση, θα επαναλαμβάνεται ένα while loop. Αυτό είναι παρόμοιο με το while loop της κανονικής κίνησης στο δωμάτιο, με τη διαφορά ότι όταν έχουμε διακοπή στη διάρκεια του free-running mode του πλαϊνού αισθητήρα η οικιακή συσκευή θα στρίβει αριστερά και θα ελαττώνεται το totalturns κατά 1, ενώ όταν έχουμε διακοπή στη διάρκεια του single conversion mode του μπροστινού αισθητήρα η οικιακή συσκευή θα στρίβει δεξιά και θα ελαττώνεται το totalturns κατά 1. Όταν η οικιακή συσκευή φτάσει στην αρχική της θέση, θα τερματίσει το πρόγραμμα.

Οι τιμές που επιλέχθηκαν για την διάρκεια της αριστερής και της δεξιάς στροφής και της αναστροφής είναι οι 4, 4 και 6 αντίστοιχα. Οι τιμές αυτές προέκυψαν από τον τύπο , όπου Τ είναι ο χρόνος του χρονιστή και είναι η συχνότητα του ρολογιού, όπως προκύπτει από τον τύπο , με =20MHz και Ν=1024.

Ως χρόνος για τον έλεγχο του πλαϊνού αισθητήρα δόθηκε το Τ=3sec, οπότε η τιμή του καταχωρήτη CMP0 θα ήταν ίση με .

Ως χρόνος για τον έλεγχο του μπροστινού αισθητήρα δόθηκε το Τ=2sec, οπότε η τιμή του καταχωρήτη CMP0 θα ήταν ίση με .

Για την πιο γρήγορη προσομοίωση, όμως, χρησιμοποιήθηκαν διαφορετικές τιμές από τις δοσμένες. Συγκεκριμένα, για τον χρόνο ελέγχου του μπροστινού αισθητήρα χρησιμοποιήθηκε value=13, ενώ για τον χρόνο ελέγχου του πλαϊνού αισθητήρα χρησιμοποιήθηκε value=12.