ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΗΓΜΕΝΟΙ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ

Εργαστηριακή Άσκηση 03:

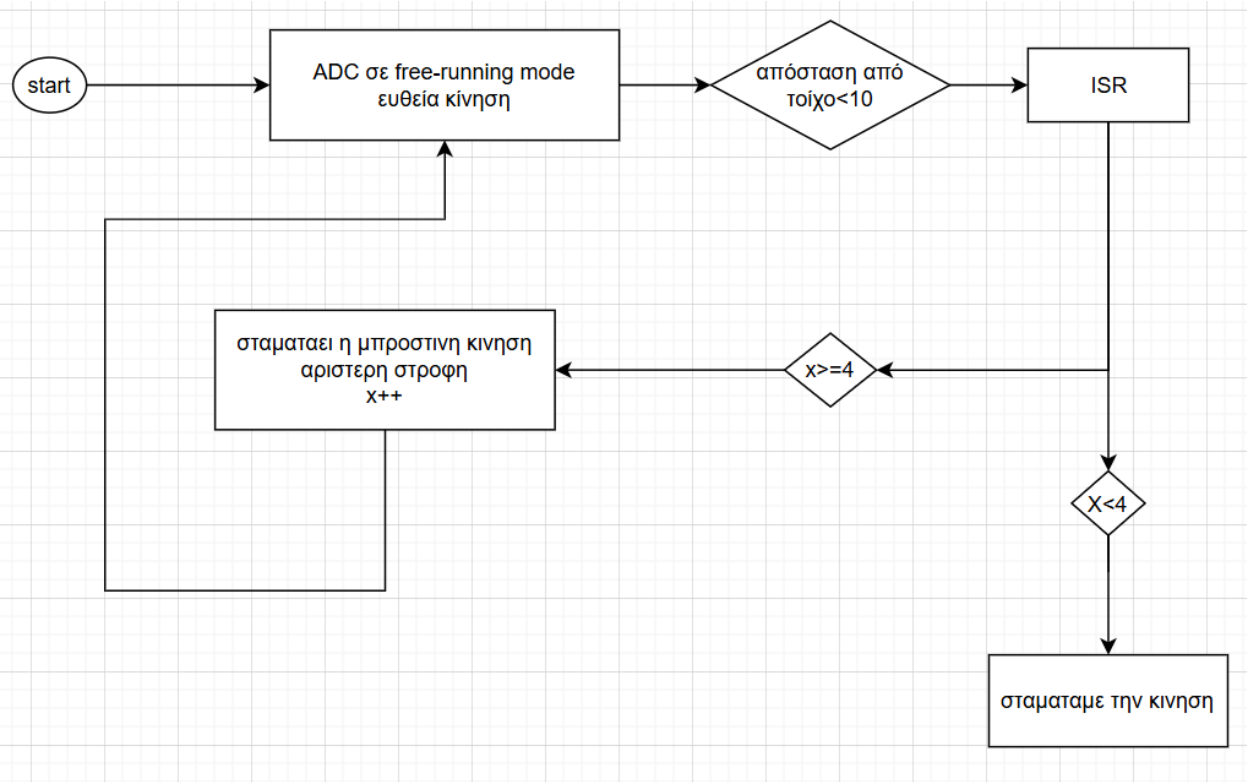
Έξυπνη Οικιακή Συσκευή, με Κίνηση στο Χώρο

Χαράλαμπος Κωνσταντακόπουλος, 1090059

Εβελίνα Σενή, 1080416

Ερώτημα 1:

Διάγραμμα ροής:



Κώδικας:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

int x = 0; // current strofi
int y = 0;

int main(){
	PORTD.DIR |= PIN1_bm;  //PIN1 is output (mprosta kinisi)
	PORTD.DIR |= PIN2_bm;  //PIN2 is output (aristeri kinisi)

	//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 position bit
        ADC0.DBGCTRL |= ADC_DBGRUN_bm;     //Enable Debug Mode

        //Window Comparator Mode
        ADC0.WINLT |= 10; //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
        PORTD.OUT |= PIN1_bm; //pame mprosta

        while(y==0){
                ;
        }
        cli();
}
ISR(ADC0_WCOMP_vect){
        cli();

        int intflags = ADC0.INTFLAGS;
        ADC0.INTFLAGS = intflags;

        if(x<4){
        PORTD.OUT |= PIN1_bm; //stamatame thn mprostini kinisi
        PORTD.OUTCLR= PIN2_bm; //LED2 is on
        PORTD.OUT |= PIN2_bm; //LED2 is off
        x++;

        PORTD.OUTCLR= PIN1_bm; //pame pali mprosta
        }else {
                PORTD.OUT |= PIN1_bm; //stamatame thn mprostini kinisi
        }

        sei();
}
```

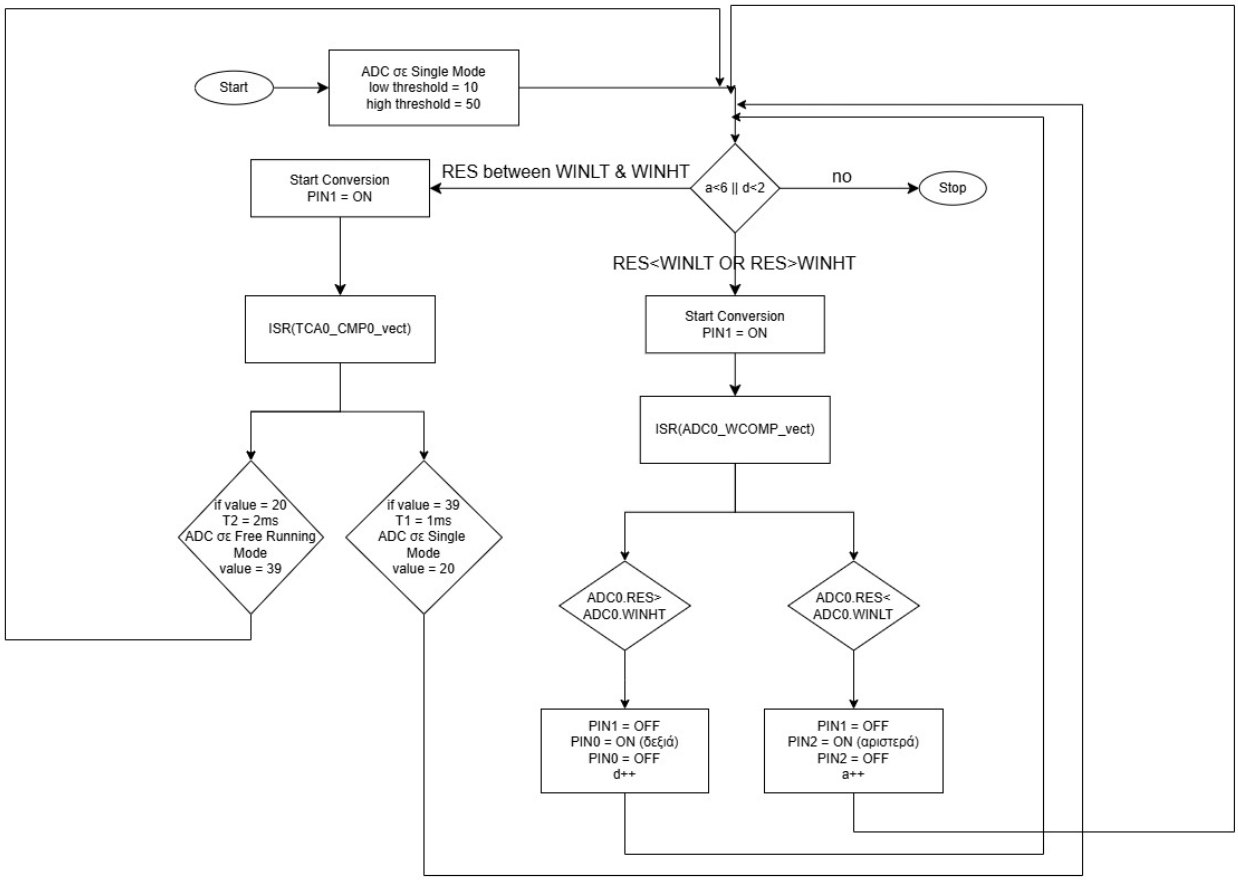
<u>Αναφορά</u>:

Ξέρουμε από την εκφώνηση ότι στο συγκεκριμένο ερώτημα το δωμάτιο είναι τετράγωνο και η συσκευή ξεκινά από μια γωνία του δωματίου, επομένως θα χρειαστούμε 3 στροφές για να φτάσει στο σημείο από το οποίο ξεκίνησε η συσκευή. Στην αρχικοποίηση ορίζουμε τον ADC σε free running mode ώστε να διαβάζει συνεχώς την απόσταση από τον τοίχο. Παράλληλα, ορίζουμε δύο εξόδους στο PORTD, την έξοδο PIN1 για την ευθεία κίνηση και την έξοδο PIN2 για τη στροφή αριστερά. Η λειτουργία ξεκινάει με την ενεργοποίηση της ευθείας κίνησης (ανάβει το LED του PIN1). Ο ADC, καθώς λειτουργεί σε συνεχή (free-running) λειτουργία, πραγματοποιεί συνεχώς μετατροπές και ελέγχει αν η απόσταση από τον τοίχο γίνει μικρότερη από το προκαθορισμένο όριο (threshold = 10). Όταν συμβεί αυτό, ενεργοποιείται μια διακοπή (interrupt), με την οποία εκτελείται η συνάρτηση ISR(ADC0_WCOMP_vect). Μέσα στη συνάρτηση της διακοπής (ISR), σταματά η ευθεία κίνηση (σβήνει το LED του PIN1) και ενεργοποιείται η στροφή αριστερά (ανάβει και σβήνει το LED του PIN2). Κάθε φορά που πραγματοποιείται στροφή, αυξάνεται κατά ένα η τιμή της μεταβλητής x, που αντιστοιχεί στον αριθμό των στροφών που έχουν γίνει. Στη συνέχεια, η συσκευή ελέγχει την τιμή της μεταβλητής x. Αν η τιμή του x είναι μικρότερη από 4, η συσκευή συνεχίζει τη διαδικασία, επαναφέροντας την κίνηση ευθεία (ανάβει ξανά το LED του

PIN1). Η διαδικασία αυτή επαναλαμβάνεται κυκλικά μέχρι η συσκευή να συμπληρώσει συνολικά 4 ευθύγραμμα τμήματα και 3 στροφές. Όταν ο μετρητής x φτάσει στην τιμή 4, η συσκευή σταματά οριστικά την κίνησή της, σβήνοντας το LED που δείχνει την ευθεία κίνηση (PIN1). Με αυτόν τον τρόπο, έχει διαγράψει την πλήρη τετράγωνη τροχιά γύρω από το δωμάτιο και έχει επιστρέψει στο σημείο από το οποίο ξεκίνησε.

Ερώτημα 2:

Διάγραμμα ροής:



Κώδικας:

```
#include <avr/io.h>
#include <avr/interrupt.h>

int value = 20;
int a = 0; //counter gia aristeres strofes
int d = 0;    //counter gia dexies strofes

int main(){
	//initialize leds - OFF
	PORTD.DIR |= PIN0_bm | PIN1_bm | PIN2_bm;
	PORTD.OUT |= PIN0_bm | PIN1_bm | PIN2_bm; //OFF
```

```
	//initialize the ADC for Single mode
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
	ADC0.CTRLA |= 0; //Single mode enabled
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The position bit
	ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
	ADC0.DBGCTRL |= ADC_DBGRUN_bm; //Enable Debug Mode

	//Window Comparator Mode
	ADC0.WINLT |= 10; //Set low threshold (empodio mprosta, dld strofi aristera)
	ADC0.WINHT |= 50; //Set high threshold (anoigma sto plai, dld strofi dexia)
	ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
	ADC0.CTRLE |= 0x4; //Interrupt when RESULT < WINLT  or RESULT > WINHT

	InitializeTimer(value);
	ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion

	sei();
	while (a<6 || d<2) { //Stamata otan ftasei 6 aristeres + 2 dexies strofes
		ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion
		PORTD.OUTCLR= PIN1_bm; //LED1 is on
	}
	cli();
}

void InitializeTimer(value){
	//initialize timer
	TCA0.SINGLE.CNT = 0; //clear counter
	TCA0.SINGLE.CTRLB = 0;            //Normal Mode
	TCA0.SINGLE.CMP0 = value;   //When CMP0 reaches this value -> interrupt gia 1ms h
2ms
	TCA0.SINGLE.CTRLA = 0x7<<1; //CLOCK_FREQUENCY/1024
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm; //Interrupt Enable (=0x10)
	TCA0.SINGLE.CTRLA |= 1;           //Enable
}

ISR(ADC0_WCOMP_vect){
	cli();

	int intflags = ADC0.INTFLAGS;            //Procedure to
	ADC0.INTFLAGS =intflags;                 //clear interrupt flag

	if(ADC0.RES < ADC0.WINLT){
		PORTD.OUT |= PIN1_bm;  //stop
		PORTD.OUTCLR= PIN2_bm; //aristeri strofi
		PORTD.OUT |= PIN2_bm; //telos aristeris strofis
		a++;
		}else if(ADC0.RES > ADC0.WINHT){
		PORTD.OUT |= PIN1_bm;  //stop
		PORTD.OUTCLR= PIN0_bm; //dexia strofi
		PORTD.OUT |= PIN0_bm;       //telos dexias strofis
		d++;
	}
	sei();
}

ISR(TCA0_CMP0_vect){
	TCA0.SINGLE.CTRLA = 0;                                  //Disable
	int intflags = TCA0.SINGLE.INTFLAGS;     //Procedure to
```

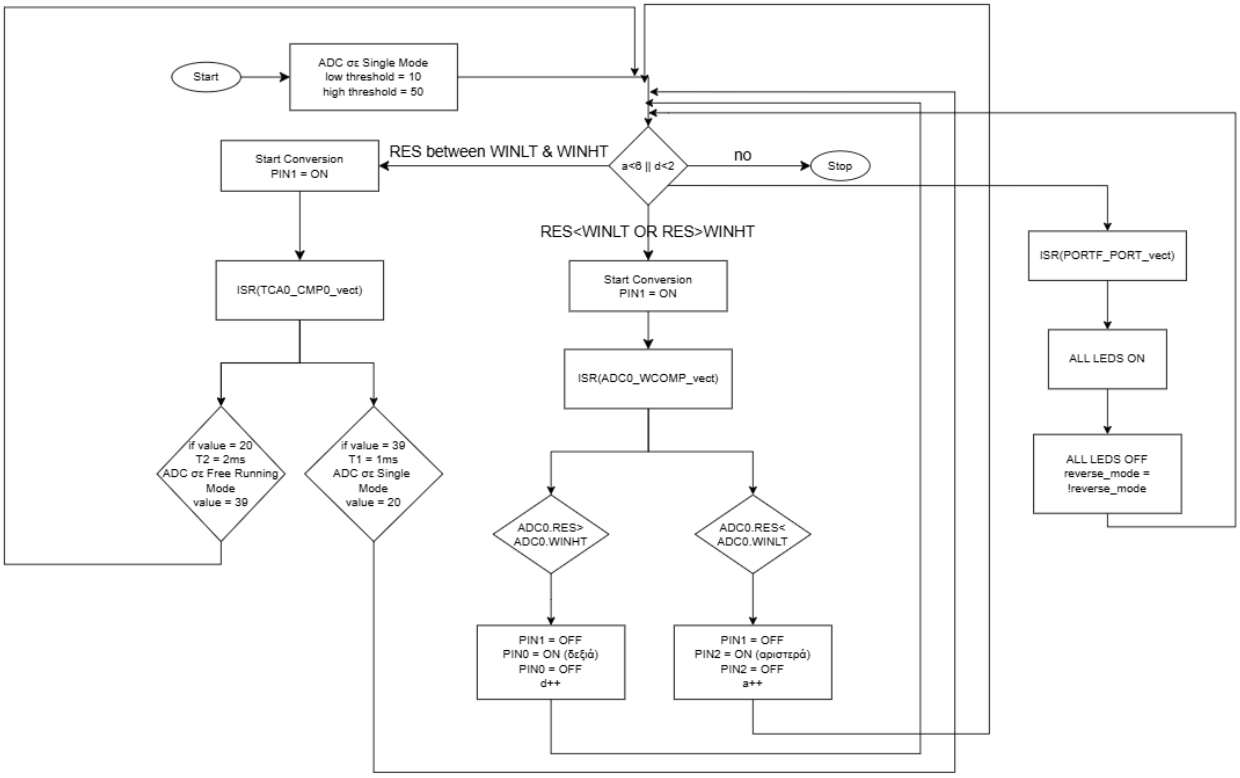
```
        TCA0.SINGLE.INTFLAGS=intflags;                  //clear interrupt flag

        if(value == 20){
                InitializeTimer(39); //2ms
                ADC0.CTRLA |= ADC_FREERUN_bm;       //Free-Running mode enabled
                value = 39;

                }else if(value==39){
                InitializeTimer(20); //1ms
                ADC0.CTRLA |= 0; //Single mode enabled
                value = 20;
        }
}
```

<u>Αναφορά</u>:

Αρχικά ορίζουμε τρεις εξόδους στο PORTD, την PIN0 για την δεξιά στροφή, την PIN1 για την ευθεία κίνηση και την PIN2 για τη αριστερή στροφή. Χρησιμοποιούμε τους μετρητές a και d για να μετρούν τις αριστερές και δεξιές στροφές αντίστοιχα. Αρχικά, όλες οι έξοδοι τίθενται στην λογική κατάσταση 1 (LEDS = OFF). Στη συνέχεια, ρυθμίζεται ο ADC σε Single mode, ενεργοποιούμε δηλαδή μόνο τον πλαϊνό αισθητήρα και ορίζουμε δύο threshold το WINLT = 10 (Low Threshold που υποδηλώνει αν υπάρχει εμπόδιο μπροστά) και WINHT = 50 (High Threshold που υποδηλώνει ότι δεν υπάρχει τοίχος στο πλάι). Ορίζουμε επίσης τον ADC0.CTRLE |= 0x4;, που συμβολίζει ότι το interrupt θα συμβαίνει όταν η τιμή του RES του ADC είναι είτε μικρότερη του 10 είτε μεγαλύτερη του 50. Για την εναλλαγή από Single mode σε Free-running mode και αντίστροφα, χρησιμοποιούμε τον timer TCA0, τον οποίο ορίζουμε στη συνάρτηση InitializeTimer(). Στην αρχή, ορίζουμε value = 20 (άρα 1 ms) και καλείται η συνάρτηση ώστε να ξεκινήσει ο timer. Σε κάθε εκτέλεση του βρόχου while, ενεργοποιείται ξανά η μετατροπή (ADC0.COMMAND |= ADC_STCONV_bm;) και το LED1 ανάβει(ευθεία). Όταν όμως από τη διακοπή του ADC προκύψει ότι η μέτρηση είναι κάτω από 10 (RES < WINLT), εκτελείται αριστερή στροφή (LED2) και ο μετρητής a αυξάνεται. Αντιθέτως, αν η μέτρηση είναι πάνω από 50 (RES > WINHT), εκτελείται δεξιά στροφή (LED0) και αυξάνεται ο μετρητής d. Παράλληλα, η συνάρτηση ISR(TCA0_CMP0_vect) σταματά σε κάθε CMP0 interrupt και αποφασίζει αν θα ρυθμίσει τον Timer στα 2 ms (θέτοντας value = 39 και ενεργοποιώντας το Free-Running mode του ADC) ή θα επιστρέψει στα 1 ms (με value = 20 και επαναφορά σε Single mode). Με τον τρόπο αυτό προσομοιώνονται οι δύο ζητούμενοι αισθητήρες: ο πλαϊνός σε single mode και ο μπροστά σε free-running, ανά τακτά εναλλάξ χρονικά διαστήματα. Το πρόγραμμα τερματίζει όταν συμπληρωθούν 6 αριστερές στροφές (a=6) και 2 δεξιές (d = 2) έχοντας επαναφέρει την συσκευή στην αρχική της θέση.

Ερώτημα 3:

Διάγραμμα ροής:



Κώδικας:

```
#include <avr/io.h>
#include <avr/interrupt.h>

int value = 20;
int a = 0; //counter gia aristeres strofes
int d = 0;    //counter gia dexies strofes
int reverse_mode = 0; // flag for reverse mode
int y=0;

int main(){
	//initialize leds - OFF
	PORTD.DIR |= PIN0_bm | PIN1_bm | PIN2_bm;
	PORTD.OUT |= PIN0_bm | PIN1_bm | PIN2_bm; //OFF

	//pullup enable and Interrupt enabled with sense on both edges
	PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;

	//initialize the ADC for Single mode
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
	ADC0.CTRLA |= 0; //Single mode enabled
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The position bit
	ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
	ADC0.DBGCTRL |= ADC_DBGRUN_bm; //Enable Debug Mode
```

```
	//Window Comparator Mode
	ADC0.WINLT |= 10; //Set low threshold (empodio mprosta, dld strofi aristera)
	ADC0.WINHT |= 50; //Set high threshold (anoigma sto plai, dld strofi dexia)
	ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
	ADC0.CTRLE |= 0x4; //Interrupt when RESULT < WINLT  or RESULT > WINHT

	InitializeTimer(value);
	ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion

	sei();

	while (y==0) {
		ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion
		PORTD.OUTCLR= PIN1_bm; //LED1 is on

		//stamata otan kaneis 6 aristeres & 2 dexies
		if (reverse_mode == 0 && a >= 6 && d >= 2) {
			PORTD.OUT |= PIN0_bm | PIN1_bm | PIN2_bm; //all LEDs OFF
			break;
		}

		//stamata otan ftaseis se 0 aristeres kai 0 dexies
		if (reverse_mode == 1 && a == 0 && d == 0) {
			PORTD.OUT |= PIN0_bm | PIN1_bm | PIN2_bm; //all LEDs OFF
			break;
		}
	}
	cli();
}

void InitializeTimer(value){
	//initialize timer
	TCA0.SINGLE.CNT = 0; //clear counter
	TCA0.SINGLE.CTRLB = 0;			//Normal Mode
	TCA0.SINGLE.CMP0 = value;   //When CMP0 reaches this value -> interrupt gia 1ms h
2ms
	TCA0.SINGLE.CTRLA = 0x7<<1; //CLOCK_FREQUENCY/1024
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm; //Interrupt Enable (=0x10)
	TCA0.SINGLE.CTRLA |= 1;			//Enable
}

ISR(PORTF_PORT_vect) {
	cli();

	int intflags = PORTF.INTFLAGS;			//Procedure to
	PORTF.INTFLAGS =intflags;			//clear interrupt flag

	PORTD.OUTCLR = PIN0_bm | PIN1_bm | PIN2_bm; //all leds on
	InitializeTimer(5);
	PORTD.OUT |= PIN0_bm | PIN1_bm | PIN2_bm; //all leds off

	reverse_mode = !reverse_mode; //enable or disable reverse mode

	sei();
}

ISR(ADC0_WCOMP_vect) {
```

```
	cli();

	int intflags = ADC0.INTFLAGS;			//Procedure to
	ADC0.INTFLAGS =intflags;			//clear interrupt flag

	if (ADC0.RES < ADC0.WINLT) {
		PORTD.OUT |= PIN1_bm;  //Stop

		if (reverse_mode == 0) {
			PORTD.OUTCLR = PIN2_bm;  //turn left
			PORTD.OUT |= PIN2_bm;	  //stop left
			a++;
			} else {
			PORTD.OUTCLR = PIN0_bm;  //turn right
			PORTD.OUT |= PIN0_bm;	  //stop right
			d--;
		}
	}
	else if (ADC0.RES > ADC0.WINHT) {
		PORTD.OUT |= PIN1_bm;  //Stop

		if (reverse_mode == 0) {
			PORTD.OUTCLR = PIN0_bm;  //turn right
			PORTD.OUT |= PIN0_bm;	  //stop right
			d++;
			} else {
			PORTD.OUTCLR = PIN2_bm;  //turn left
			PORTD.OUT |= PIN2_bm;	  //stop left
			a--;
		}
	}
	sei();
}

ISR(TCA0_CMP0_vect){
	TCA0.SINGLE.CTRLA = 0;					//Disable
	int intflags = TCA0.SINGLE.INTFLAGS;	//Procedure to
	TCA0.SINGLE.INTFLAGS=intflags;			//clear interrupt flag

	if(value == 20){
		InitializeTimer(39); //2ms
		ADC0.CTRLA |= ADC_FREERUN_bm;	 //Free-Running mode enabled
		value = 39;

		}else if(value==39){
		InitializeTimer(20); //1ms
		ADC0.CTRLA |= 0; //Single mode enabled
		value = 20;
	}
}
```

<u>Αναφορά</u>:

Αρχικά ορίζουμε τρεις εξόδους στο PORTD, την PIN0 για την δεξιά στροφή, την PIN1 για την ευθεία κίνηση και την PIN2 για τη αριστερή στροφή. Χρησιμοποιούμε τους μετρητές a και d για να μετρούν τις αριστερές και δεξιές στροφές αντίστοιχα και την μεταβλητή reverse_mode που χρησιμοποιείται για την

ανάποδη πορεία της συσκευής. Αρχικά, όλες οι έξοδοι τίθενται στην λογική κατάσταση 1 (LEDs = OFF) και η ενεργοποίηση της ανάποδης λειτουργίας γίνεται με το πάτημα του κουμπιού Switch 5, το οποίο αντιστοιχεί στο PIN5 του PORTF. Στη συνέχεια, ρυθμίζεται ο ADC σε Single mode, ενεργοποιούμε δηλαδή μόνο τον πλαϊνό αισθητήρα και ορίζουμε δύο threshold το WINLT = 10 (Low Threshold που υποδηλώνει αν υπάρχει εμπόδιο μπροστά) και WINHT = 50 (High Threshold που υποδηλώνει ότι δεν υπάρχει τοίχος στο πλάι). Ορίζουμε επίσης τον ADC0.CTRLE |= 0x4;, που συμβολίζει ότι το interrupt θα συμβαίνει όταν η τιμή του RES του ADC είναι είτε μικρότερη του 10 είτε μεγαλύτερη του 50. Για την εναλλαγή από Single mode σε Free-running mode και αντίστροφα, χρησιμοποιούμε τον timer TCA0, τον οποίο ορίζουμε στη συνάρτηση InitializeTimer(). Στην αρχή, ορίζουμε value = 20 (άρα 1 ms) και καλείται η συνάρτηση ώστε να ξεκινήσει ο timer. Σε κάθε εκτέλεση του βρόχου while, ενεργοποιείται ξανά η μετατροπή (ADC0.COMMAND |= ADC_STCONV_bm;) και το LED1 ανάβει(ευθεία). Έπειτα, ελέγχει αν βρισκόμαστε σε reverse mode, όπου αν είμαστε και ισχύει ότι a & d = 0, δηλαδή ότι έχουμε βρεθεί στην αρχική θέση τότε σβήνει όλα τα LEDs και διακόπτεται ο βρόχος. Αν δεν βρισκόμαστε σε reverse mode και ισχύει ότι έχει κάνει 6 αριστερές στροφές και 2 δεξιές τότε σβήνει όλα τα LEDs και διακόπτεται ο βρόχος. Στην περίπτωση που δεν εκτελείται reverse_mode, τότε αν από τη διακοπή του ADC προκύψει ότι η μέτρηση είναι κάτω από 10 (RES < WINLT), εκτελείται αριστερή στροφή (LED2) και ο μετρητής a αυξάνεται. Αντιθέτως, αν η μέτρηση είναι πάνω από 50 (RES > WINHT), εκτελείται δεξιά στροφή (LED0) και αυξάνεται ο μετρητής d. Η συνάρτηση ISR(PORTF_PORT_vect) ενεργοποιείται μόλις πατηθεί το κουμπί PIN5 στο PORTF και μέσα σε αυτήν αναβοσβήνουν τα LEDs και η μεταβλητή reverse_mode γίνεται 1 ή 0 ανάλογα με την τιμή που είχε πριν (ενεργοποιείται ή απενεργοποιείται η ανάποδη λειτουργία). Με την είσοδο στο reverse mode, οι μεταβλητές a και d που μετρούν τις στροφές χρησιμοποιούνται αντίστροφα, δηλαδή για RES < WINLT σημαίνει ότι δεν υπάρχει εμπόδιο πίσω (μπορεί να γίνει στροφή δεξιά) και για RES > WINHT σημαίνει ότι δεν υπάρχει τοίχος στο πλάι (μπορεί να γίνει στροφή αριστερά). Παράλληλα, η συνάρτηση ISR(TCA0_CMP0_vect) σταματά σε κάθε CMP0 interrupt και αποφασίζει αν θα ρυθμίσει τον Timer στα 2 ms (θέτοντας value = 39 και ενεργοποιώντας το Free-Running mode του ADC) ή θα επιστρέψει στα 1 ms (με value = 20 και επαναφορά σε Single mode).