ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΟΛΟΓΙΑΣ & ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΗΓΜΕΝΩΝ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

2Η ΑΣΚΗΣΗ

07/04/2022

ΧΑΡΑΛΑΜΠΟΣ ΠΑΠΠΑΣ - 1053359
ΓΕΩΡΓΙΟΣ ΚΟΝΤΟΓΙΑΝΝΗΣ - 1070908

Ασκηση 1

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

int turns = 4;

int main(void)
{

	// PIN0, turn left
	PORTD.DIR |= PIN0_bm;
	PORTD.OUT |= PIN0_bm;

	// PIN2, represents straight line
	PORTD.DIR |= PIN2_bm;
	// turn on straight's led
	PORTD.OUTCLR = PIN2_bm;

	// initialize ADC
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc;
	ADC0.CTRLA |= ADC_FREERUN_bm;
	ADC0.CTRLA |= ADC_ENABLE_bm;
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc;

	ADC0.DBGCTRL |= ADC_DBGRUN_bm;

	// limit is 5
	ADC0.WINLT |= 5;
	ADC0.INTCTRL |= ADC_WCMP_bm;
	ADC0.CTRLE |= ADC_WINCM0_bm;

	sei();

	// start conversion
	ADC0.COMMAND |= ADC_STCONV_bm;

	while(1){
		// walk straight
		// if perimeter completed, stop/ exit programm
		if (turns == 0)
			break;
	}
}

ISR(ADC0_WCOMP_vect){
	int intflags = ADC0.INTFLAGS;
	ADC0.INTFLAGS = intflags;
	// turn straight led off
	PORTD.OUT |= PIN2_bm;
	// turn on left led
	PORTD.OUTCLR = PIN0_bm;
	_delay_ms(2);
	// turn off left led
	PORTD.OUT |= PIN0_bm;
	// turn straight led on
	PORTD.OUTCLR = PIN2_bm;
	// reduce each side of the perimeter
	turns -= 1;}
```

Σχολια & Παραδοχες Κωδικα

- Για το 1^{ο} ερωτημα της ασκησης, προκειμενου να προσομοιωθει η κινηση του τετραγωνου, θετουμε το PIN2 ανεμενο για την ευθεια κινηση και τον PIN0 για την αριστερη στροφη. Ολες οι αλλες περιπτωσεις στο συγκεκριμεο παραδειγμα δεν μας αποσχολουν καθως η συσκευη στριβει μονο αριστερα.
- Με τη μεταβλητη “turns”, κραταμε τον αριθμο στροφων που εχει κανει, και μολις αυτο φτασει στο τεσσερα (4 στροφες κατα μεγιστο μπορει να κανει), θα ειναι ουσιαστικα στην θεση εκκινησης, οποτε και το προγραμμα σταματαει.
- Καθε φορα που πυροδοτειται το interrupt του ADC, το LED της ευθειας σβηνει, αναβει το LED της στροφης. Η στροφη αυτη προσομοιωνειται με μια μικρη καθυστερηση. Στη συνεχεια, αναβει ξανα το LED της ευθειας.Τελος, η μεταβλητη turns μειωνεται κατα 1.

Ασκηση 2-3

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

#define t 10; // 1 second delay

int cnt = 0;
int still = 1;
int t_interr = 0;
int result = 0;
int reverse = 0;

int main(void)
{

	// PIN0, turn left
	PORTD.DIR |= PIN0_bm;
	PORTD.OUT |= PIN0_bm;

	// PIN1, turn right
	PORTD.DIR |= PIN1_bm;
	PORTD.OUT |= PIN1_bm;

	// PIN2, represents straight line
	PORTD.DIR |= PIN2_bm;
	// turn on straight's led
	PORTD.OUTCLR = PIN2_bm;


	// initialize ADC
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc;
	ADC0.CTRLA |= ADC_FREERUN_bm;
	ADC0.CTRLA |= ADC_ENABLE_bm;
	ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc;

	ADC0.DBGCTRL |= ADC_DBGRUN_bm;

	// limit is 5
	ADC0.WINLT |= 5;
	ADC0.WINHT |= 20;
	ADC0.INTCTRL |= ADC_WCMP_bm;
	ADC0.CTRLE |= ADC_WINCM0_bm;

	// switch interrupt enable
	PORTF.PIN5CTRL |= (PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc);

	//Timer set up
	TCA0.SINGLE.CNT = 0; //Clear counter
	TCA0.SINGLE.CTRLB = 0; //Normal Mode
	TCA0.SINGLE.CMP0 = t; //Stop turning when this value is reached
	TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc;
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;
```

Ασκηση 2-3

```
while(1){
		// go straight
		sei();
		// start conversion
		ADC0.COMMAND |= ADC_STCONV_bm;

		PORTD.OUTCLR = PIN2_bm;
		_delay_ms(1);
		// check for right distance
		single();

		// perimeter is completed. Stop device
		if(!still)
		break;
	}
}


ISR(ADC0_WCOMP_vect){
	int intflags = ADC0.INTFLAGS;
	ADC0.INTFLAGS = intflags;

	if(!reverse){
		if (result <=5){
			cnt++;
			turnLeft();

			}else if(result >= 20){
			cnt++;
			turnRight();
		}
		// reverse mode enabled
		}else{
		// increase counter
		if (result <=5){
			cnt--;
			turnRight();

			}else if(result >= 20){
			cnt--;
			turnLeft();
		}
	}
	// CLEAR RES
	ADC0.RES = 0;
	// if perimeter completed
	if(cnt >= 7 || cnt <= 0)
	still = 0;

}
```

Ασκηση 2-3

```
ISR(TCA0_CMP0_vect){
	//TCA0.SINGLE.CTRLA = 0; //Disable Timer
	TCA0.SINGLE.CTRLA &= 0xFE;
	//Clear timer interrupt flag
	int intflags = TCA0.SINGLE.INTFLAGS;
	TCA0.SINGLE.INTFLAGS = intflags;
	t_interr = 1;
}


ISR(PORTF_PORT_vect) {
	int intflags = PORTF.INTFLAGS;
	PORTF.INTFLAGS = intflags;

	reverse = 1;
}




void turnLeft(){

	// turn straight led off
	PORTD.OUT |= PIN2_bm;
	// turn on left led
	PORTD.OUTCLR = PIN0_bm;
	_delay_ms(2);
	// enable timer
	/*
	TCA0.SINGLE.CTRLA |= 1;
	sei();
	while(!t_interr){
		// wait timer
	}
	//TCA0.SINGLE.CTRLA &= 0xFE;
	cli();
	*/
	// turn off left led
	PORTD.OUT |= PIN0_bm;
	// turn straight led on
	PORTD.OUTCLR = PIN2_bm;
}

void turnRight(){

	// turn straight led off
	PORTD.OUT |= PIN2_bm;
	// turn on right led
	PORTD.OUTCLR = PIN1_bm;
	_delay_ms(2);
	// turn off left led
	PORTD.OUT |= PIN1_bm;
	// turn straight led on
	PORTD.OUTCLR = PIN2_bm;
}
```

Ασκηση 2-3

```
void single(){
	// clear ADC
	ADC0.CTRLA = 0;
	// initialize ADC
	ADC0.CTRLA |= ADC_RESSEL_10BIT_gc;
	ADC0.CTRLA |= ADC_ENABLE_bm;

	ADC0.CTRLE = 0x02;
	sei();
	// start conversion
	ADC0.COMMAND |= ADC_STCONV_bm;
	result = ADC0.RES;
	_delay_ms(2);

	// re-set ADC to free-running
	ADC0.CTRLA |= ADC_FREERUN_bm;
	ADC0.CTRLE = 0x01;
	ADC0.DBGCTRL |= ADC_DBGRUN_bm;
	// start conversion
	ADC0.COMMAND |= ADC_STCONV_bm;
}
```

Σχολια & Παραδοχες Κωδικα

- Η λογικη του προγραμματος ειναι ιδια με πριν, με τη διαφορα οτι τα LED0, LED1 υποδηλωνουν την αριστερα και δεξια στροφη αντιστοιχα και το LED2 την ευθεια πορεια. Επισης, εχουν δημιουργηθει δυο συναρτησεις “turnLeft” & “turnRight”, οι οποιες προσομοιωνουν τις στροφες, αναβοσβηνοντας τα αντιστοιχα LED καθε φορα με τη βοήθεια ενός timer, οπως στην ασκηση 1. Επίσης, η μεταβλητή turns τώρα ονομάζεται cnt.

- Έχει οριστεί ένα άνω και κάτω όριο (WINLT=5, WINHT=20) που υποδηλώνουν την ενεργοποίση ανίχνευσης μπροστινού και δεξιού εμποδίου, αντίστοιχα. Δηλαδή, εισάγωντας στον καταχωρητή RES τιμή μικρότερη ίση του 5, η συσκευή ανιχνεύει σε free-running mode μπροστινά εμπόδια, και για RES μεγαλύτερο ίσο του 20, ελέγχθει μια φορά δεξιά με τη βοήθεια της συνάρτησης single(). Η συνάρτηση single() αλλάζει τον ADC0 σε mode 0, δειγματοληπτεί την απόσταση μέσω του RES και εάν ξεπερνάει κάποιο όριο ενεργοποιείται το `ISR(ADC0_WCOMP_vect)` το οποίο αποφασίζει αν θα στρίψει δεξιά ή όχι. Αν ναί, καλείται η συνάρτηση turnRight(), γίνονται οι αντίστοιχες ενέργειες, μηδενίζει τον RES, ελέγχει άν έχει διανύσει και τις επτά (7) πλευρές (ελέγχει αν cnt >= 7, αν ναί κάνει το flag still=0) και ξανα επιστρέφει στο free-running mode. Επιστρέφει στη main και ελέγχει το flag still. Άν είναι «1», η παραπάνω διαδικασία συνεχίζεται. Στη περίπτωση όπου το RES <= WINLT πυροδοτείται το `ISR(ADC0_WCOMP_vect)` εκτελείται η αριστερά στροφή με τη βοήθεια της turnLeft(), ελέγχει το still και η διαδικασία επαναλαμβάνεται. Τελος, οταν το cnt φτασει τη τιμη ‘7’, θετουμε τη μεταβλητη still = 0 , διακοπτεται το προγραμμα και η συσκευη βρισκεται στην αρχικη της θεση.

- Η μεταβλητή reverse=0 μας ενημερώνει ότι η συσκεύη χαράζει κανονική πορεία και όλες οι παραπάνω λειτουργίες εκτελούνται όπως αναλύθηκαν υπό τη συνθήκη οτι if(!reverse)=true. Η λειτουργια της επαναφορας θεσης, εκτελειται με την πυροδότηση του `ISR(PORTF_PORT_vect)` μετά την ενεργοποιηση του switch στο PIN 5, θέτοντας το reverse=1, ενημερώνοντας έτσι το σύστημα if(!reverse)=false ότι θέλουμε αντίστροφη πορεία. Με το τρόπο αυτό, στη πυροδότηση του `ISR(ADC0_WCOMP_vect)` εκτελείται κάθε φορά ένα ξεχωριστό τμήμα κώδικα το οποίο κάνει τις καθρεπτικές λειτουργίες απο πρίν, βασιζόμενο στη μέχρι τότε τιμή του cnt, μέχρις ώτου cnt=0.