Προηγμένοι Μικροεπεξεργαστές

Εργαστηριακή Άσκηση 2

Λουδάρος Ιωάννης (1067400) - Χριστίνα Κρατημένου (1067495)

Μπορείτε να δείτε την τελευταία έκδοση του Project εδώ ή σκανάροντας τον κωδικό QR που βρίσκεται στην επικεφαλίδα.

Περιγραφή Αναφοράς

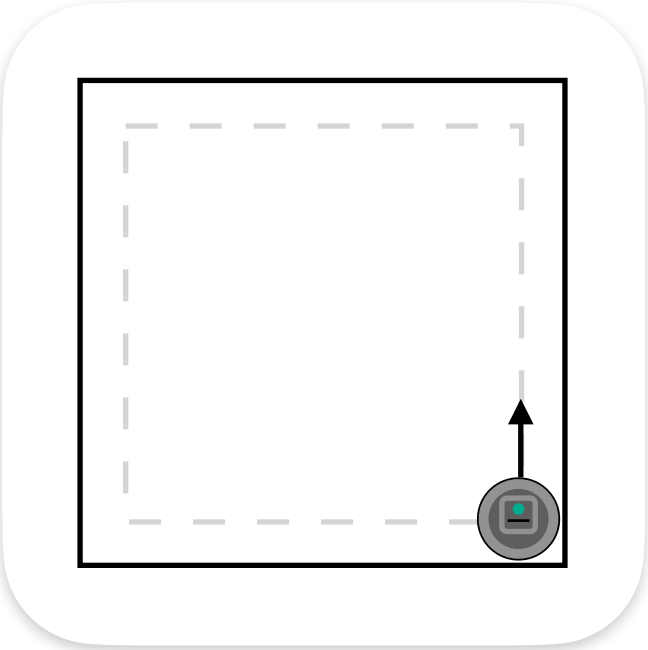
Παρακάτω παραθέτουμε τις απαντήσεις μας στην "Εργαστηριακή Άσκηση 2" του μαθήματος "Προηγμένοι Μικροεπεξεργαστές" καθώς και σχόλια τα οποία προέκυψαν κατά την εκπόνηση της.

Περιεχόμενα

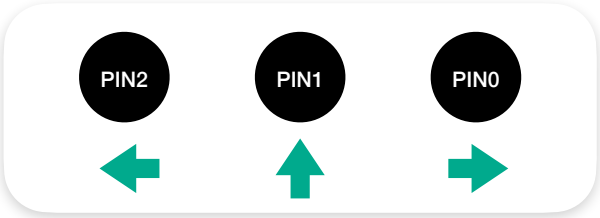
Απαντήσεις

Ερώτημα 1

Σχεδιασμός



Σχεδιάγραμμα του δωματίου



Προσομοίωση Κίνησης

Κώδικας

Τροποποιούμε το παράδειγμα 4 ώστε να ανάβουμε τα κατάλληλα leds. Για να επιτύχουμε τον στόχο μας αξιοποιούμε και τον Timer TCA0.

```
int turn_left=0;
int forward=1;
int turn_right=0;

int end=0;
```

Χρησιμοποιούμε τις παραπάνω μεταβλητές ώστε να παρακολουθούμε την κατάσταση της κίνησης της συσκευής μας κάθε στιγμή. Το end είναι μια ειδική μεταβλητή που μετράει τις φορές που η συσκευή μας έχει στρίψει αριστερά ώστε να καταλάβει πότε ακριβώς φτάνει στο σημείο από το οποίο ξεκίνησε.

```
while(1){
		if (end==4) break;

		while (turn_left)
		{
			//Set the timer
			TCA0.SINGLE.CNT = 0; //clear counter
			TCA0.SINGLE.CTRLB = 0; //Normal Mode
			TCA0.SINGLE.CMP0 = ped; //When reaches this value ->
						interrupt CLOCK Frequency/1024
			TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc; //= 0x7<<1
			TCA0.SINGLE.CTRLA |=1; //Enable
			TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm; //Interrupt Enable
								(=0x10)
			while (turn_left){}

		}

		while (turn_right)
		{

		}

	}
```

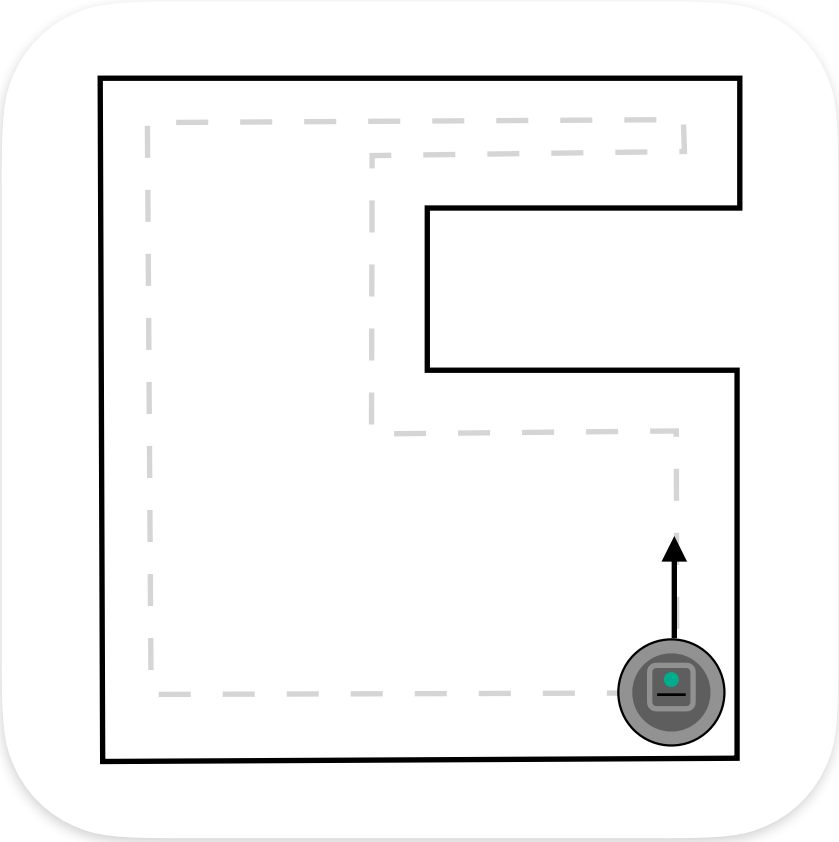
Ο παραπάνω βρόγχος ελέγχει την κίνηση της συσκευής κάθε στιγμή. Υπάρχει ένα while για κάθε state που ορίζεται από τις μεταβλητές που αναφέραμε πριν. Στο συγκεκριμένο ερώτημα, ενδιαφέρον έχει μόνο ο βρόγχος που ελέγχει την αριστερή στροφή, ο οποίος απλά σετάρει το timer και το περιμένει να τελειώσει. Παρατηρούμε ότι σε κάθε πέρασμα του κεντρικού βρόγχου ελέγχεται η συνθήκη τερματισμού του συστήματος.

```
ISR(ADC0_WCOMP_vect){
//clear the flags of the ADC
	int intflags = ADC0.INTFLAGS;
	ADC0.INTFLAGS = intflags;

	PORTD.OUT |= (PIN0_bm|PIN1_bm|PIN2_bm); //no movement
	PORTD.OUTCLR |= PIN2_bm; //turn_left
	forward=0;
	turn_left=1;

}

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

	turn_left=0;
	forward=1;

	PORTD.OUT |= (PIN0_bm|PIN1_bm|PIN2_bm); //no movement
	PORTD.OUTCLR |= PIN1_bm; //forward


	end++;
}
```

Παραπάνω βλέπουμε τις συναρτήσεις που διαχειρίζονται τα interrupts του ADC και του TCA αντίστοιχα. Ουσιαστικά, το κάθε interrupt ρυθμίζει τα led κατάλληλα και ενημερώνει το σύστημα για την κατάσταση στην οποία βρίσκεται με τις μεταβλητές που έχουμε αναφέρει.

Ερώτημα 2

Σχεδιασμός



Σχεδιάγραμμα του δωματίου

Κώδικας

```
void check_right();
void turning_timer();
void initialize_ADC0();
void frunning_ADC0();
void start_ADC0();

int turn_left=0;
int forward=1;
int turn_right=0;

int end=0;
```

Ο κώδικας μας άρχισε να έχει αρκετά επαναχρησιμοποιούμενα κομμάτια, τα οποία πακετάραμε σε συναρτήσεις. Συγκεκριμένα:

```
void check_right();
```

Σταματάει στιγμιαία το free_running mode και ελέγχει χειροκίνητα αν η τιμή του RES είναι μεγαλύτερη αυτής του limit (που έχουμε θέσει στο 10).

```
void turning_timer();
```

Σετάρει και ξεκινάει το timer.

```
void initialize_ADC0();
void frunning_ADC0();
void start_ADC0();
```

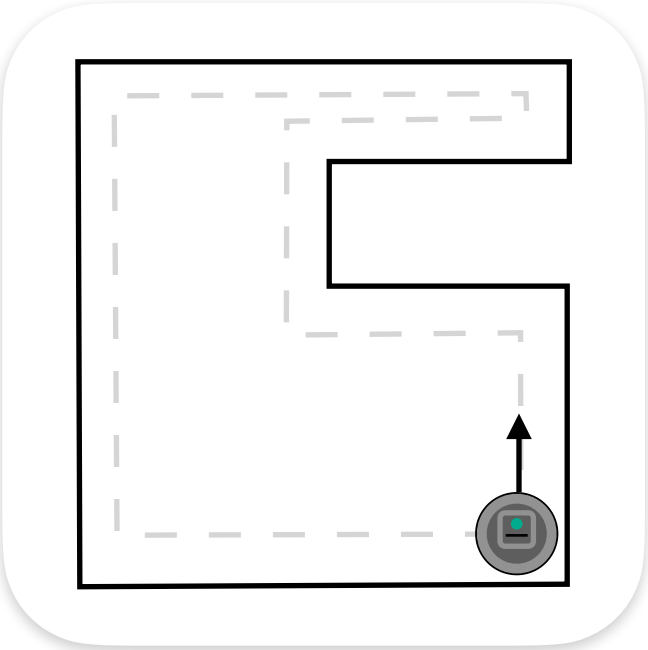
Οι παραπάνω συναρτήσεις φροντίζουν το initialization του ADC, την ρύθμιση του free running mode, και την εκκίνηση των συγκρίσεων αντίστοιχα.

```
int main(){
	PORTD.DIRSET = PIN0_bm|PIN1_bm|PIN2_bm; // 2:left 1:forward 0:right
	PORTD.OUT |= (PIN0_bm|PIN2_bm); //start going forward

	initialize_ADC0();
	frunning_ADC0();
	start_ADC0();
	sei();

	while(1){

		check_right();

		if (end==4) break;


		while (turn_left)
		{
			end++;
			turning_timer();
			while (turn_left){}
		}

		while (turn_right)
		{
			end=end-1;
			turning_timer();
			while (turn_right){}

		}

	}
```

Βλέπουμε παραπάνω πως μετασχηματίστηκε η main και ο κύριος βρόγχος.

Ερώτημα 3

Σχεδιασμός



Σχεδιάγραμμα του δωματίου

Ανάποδη Λειτουργία

Κώδικας

Χρησιμοποιώντας την δουλειά μας μέχρι τώρα, χρειάζεται να προσθέσουμε στο σύστημα μας το πως θα συμπεριφέρεται σε περίπτωση που πατηθεί το κουμπί.

```
int main(){
    PORTD.DIRSET = PIN0_bm|PIN1_bm|PIN2_bm; // 2:left 1:forward 0:right
    PORTD.OUT |= (PIN0_bm|PIN2_bm); //start going forward

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

Σετάρουμε το κουμπί στην αρχή της main.

```
while(1){

        check_right();


        if (end==4) break;


        while (turn_left)
        {

            turning_timer();
            while (turn_left){}
        }

        while (turn_right)
        {
            turning_timer();
            while (turn_right){}

        }

        while (inverse)
        {
            PORTD.OUTCLR |= (PIN0_bm|PIN1_bm|PIN2_bm); //U turn, all leds open
            turning_timer();
            while(inverse){}
        }

    }
```

Βλέπουμε ότι στον κύριο βρόγχο συμπεριλάβαμε την περίπτωση να έχει πατηθεί το κουμπί για την αντίστροφη λειτουργία.

Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι συναρτήσεις που προηγουμένως μπορούσαν να εκτελέσουν μόνο μια στροφή. Παρουσιάζουμε ενδεικτικά την check_right( ), η οποία αλλάζει την συμπεριφορά της σύμφωνα με τη φορά λειτουργίας.

```
void check_right(){

    ADC0.CTRLA |= 0x05; // disable free-running
    ADC0.CTRLE |= 0x00; // disable comparison

    if (ADC0.RES>limit) {

        if (inverse)
        {
            PORTD.OUT |= (PIN0_bm|PIN1_bm|PIN2_bm); //no movement
            PORTD.OUTCLR |= PIN2_bm; //turn_left
            forward=0;
            turn_left=1;
            end=end-1;
        }

        else
        {
            PORTD.OUT |= (PIN0_bm|PIN1_bm|PIN2_bm); //no movement
            PORTD.OUTCLR |= PIN0_bm; //turn_right
            forward=0;
            turn_right=1;
            end=end-1;
        }
```

Υπολογισμός χρόνου

$$f_{timer} = \frac{20MHz}{1024} = 19{,}531KHz$$

$$T = \frac{10}{f_{timer}} \approx 0.5s$$