**ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ**

**ΔΕΥΤΕΡΗ ΑΣΚΗΣΗ**

Στοιχεία μελών ομάδας:

1ο μέλος ομάδας

Ονοματεπώνυμο: Γκουργκούτας Βασίλειος

ΑΜ: 1084667

Έτος: 4ο

2ο μέλος ομάδας

Ονοματεπώνυμο: Κοντογιώργος Αναστάσιος

ΑΜ: 1090084

Έτος: 4ο

**Ερωτήματα Εργαστηριακής Άσκησης 2**

1. Ο κώδικας που υλοποιήθηκε για την κίνηση της οικιακής συσκευής, όταν το δωμάτιο είναι τετράγωνο με 900 γωνίες είναι ο εξής:

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdio.h>

int gonies = 0; // Initialize the number of turns that the smart cleaner will make

int flag = 1; // We define a variable for the while loop

int main(){

PORTD.DIR |= PIN1\_bm; //PIN1\_bm //PIN is output // Pin 1 indicates that the smart cleaner will move forward

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

PORTD.DIR |= PIN2\_bm; //PIN2\_bm //PIN is output // Pin 2 Indicates that the smart cleaner will turn left

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

// Initialize the ADC for single conversion 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

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

ADC0.WINLT |= 5; //Set threshold

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

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

sei(); // Enable Interrupts

while(gonies != 4) // We check if the smart cleaner has completed the square

{

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on // We enable the led for moving forward

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

while(flag==1){ //We stay inside the while loop until the interrupt from the ADC is enabled

sei();

}

flag = 1; // We set the flag again for the while loop

PORTD.OUT |= PIN1\_bm; //PIN1 is off // We disable the led for moving forward

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on // We enable the led for turning left

gonies = gonies + 1; // We increase the number of turns that the smart cleaner has made

}

cli(); //Disable Interrupts

}

ISR(ADC0\_WCOMP\_vect){

cli();

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

flag = 0; // The flag changes value so that we can move on from the while loop

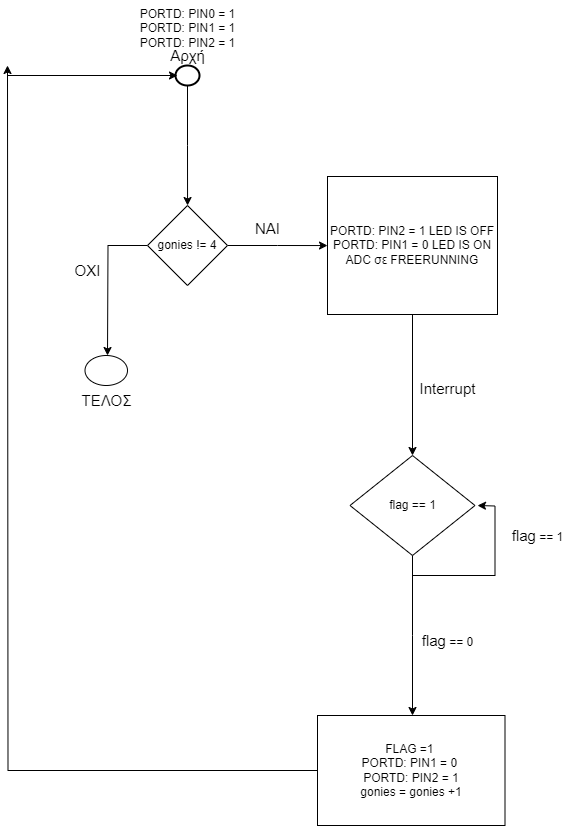
sei();

}

**Επεξήγηση κώδικα πρώτου ερωτήματος**

Αρχικά ορίζουμε τα Pin 1 και Pin 2 για το συγκεκριμένο ερώτημα σαν outputs και τα απενεργοποιούμε. Έπειτα αρχικοποιούμε τον ADC, ορίζοντας το RES να είναι σε εύρος 10 bit (ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc;). Με την εντολή ADC0.CTRLA |= ADC\_FREERUN\_bm; ενεργοποιούμε τον ADC σε λειτουργία free-running mode. Στη συνέχεια ενεργοποιούμε τον ADC με την εντολή ADC0.CTRLA |= ADC\_ENABLE\_bm;. Η εντολή ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; ορίζει το bit με το οποίο συνδέεται ο ADC. Με την εντολή ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; ενεργοποιούμε το debug mode για τον ADC. Έπειτα η εντολή ADC0.WINLT |= 5; ορίζει το κατώφλι στην τιμή που έχουμε τοποθετήσει, δηλαδή την τιμή 5. Η εντολή ADC0.INTCTRL |= ADC\_WCMP\_bm; ενεργοποιεί τις διακοπές για το Window Compare. Μετέπειτα η εντολή ADC0.CTRLE |= ADC\_WINCM0\_bm; ενεργοποιεί την διακοπή όταν το RES είναι μικρότερο από το κατώφλι που έχουμε ορίσει. Επιπρόσθετα ορίζεται μια δομή επανάληψης while η οποία εκτελείται για έως ότου η μεταβλητή gonies γίνει ίση με τέσσερα, δηλαδή όταν η έξυπνη συσκευή ολοκληρώσει το περίγραμμα του τετραγώνου. Μέσα στην while απενεργοποιούμε το Pin2 για την αριστερή στροφή και ενεργοποιούμε το Pin1 για δηλώσουμε ότι η έξυπνη συσκευή κινείται ευθεία. Έπειτα ξεκινάει η μετατροπή στον ADC με την εντολή ADC0.COMMAND |= ADC\_STCONV\_bm; Και εφόσον έχουμε τοποθετήσει τιμή μικρότερη από το κατώφλι, δηλαδή μικρότερη από το 5 στην συγκεκριμένη περίπτωση, τότε ενεργοποιείται η ISR του ADC όπου θα καθαριστούν τα flags και η μεταβλητή flag θα γίνει μηδέν ώστε να περάσουμε από την εσωτερική while. Στη συνέχεια, απενεργοποιούμε το Pin1 για την κίνηση ευθεία και ενεργοποιούμε το Pin2 για δηλώσουμε την κίνηση προς τα αριστερά. Υπάρχει μία μεταβλητή gonies όπου θα αυξάνεται κατά ένα κάθε φορά που στρίβει η έξυπνη συσκευή όταν συναντάει γωνία μπροστά της. Η εξωτερική while εκτελείται 4 φορές δηλαδή όσες φορές χρειάζεται ώστε η έξυπνη συσκευή να δημιουργήσει το περίβλημα του τετραγώνου και τέλος να επιστρέψει στην αρχική της θέση.

Διάγραμμα Ροής για το ερώτημα 1



1. O κώδικας που υλοποιήθηκε για την κίνηση της οικιακής συσκευής για ένα τυχαίο δωμάτιο που περιέχει και δύο αμβλείες (270 μοίρες) γωνίες είναι ο εξής:

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdio.h>

#define plagios\_sensor 15 // T1= 1msec, value=T1\*f(timer) = 2msec\*19,531KHz=19.531, for the simulation we used smaller values

#define mprostinos\_sensor 20 // T2=2msec, value=T2\*f(timer) = 2msec\*19,531KHz=39.062, for the simulation we used smaller values

int turned = 0; // We define a variable for the while loop

int gonies = 0; // Initialize the number of turns that the smart cleaner will make

int flag = 0; // We define a variable for the while loop

int gonies\_dexia = 0; // Initialize the number of right turns that the smart cleaner will make

int gonies\_aristera = 0; // Initialize the number of left turns that the smart cleaner will make

int main(){

PORTD.DIR |= PIN0\_bm; //PIN0\_bm //PIN is output // Pin 0 Indicates that the smart cleaner will turn right

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

PORTD.DIR |= PIN1\_bm; //PIN1\_bm //PIN is output // Pin 1 indicates that the smart cleaner will move forward

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

PORTD.DIR |= PIN2\_bm; //PIN2\_bm //PIN is output // Pin 2 Indicates that the smart cleaner will turn left

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

// Initialize the ADC for single conversion mode

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

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

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

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

ADC0.WINLT |= 5; //Set threshold

ADC0.WINHT |= 5; //Set threshold

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

sei(); // Enable Interrupts

while(gonies != 4) // We check if the smart cleaner has completed the square

{

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on // We enable the led for moving forward

// T1 = 1ms

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

TCA0.SINGLE.CTRLB = 0; //Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc)

TCA0.SINGLE.CMP0 = plagios\_sensor; //When CMP0 reaches this value -> interrupt

//CLOCK\_FREQUENCY/1024

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Interrupt Enable (=0x10)

while(turned==0)

{

// Initialize the ADC for Single Conversion mode

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

ADC0.CTRLA = ADC0.CTRLA & 0b11111101; // We set the mode back to Single Conversion

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

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc;

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

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

ADC0.CTRLE = 0b00000010; // Interrupt when RES> WINHT

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

while((flag==0) && (turned==0)) // We can't move on from the while loop, until either the ADC will trigger an Interrupt or the Timer will trigger an Interrupt.

{

sei();

}

if (flag==1)

{

flag = 0; // We set the flag variable to zero again

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUT |= PIN1\_bm; //PIN1 is off // We disable the led for moving forward

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on// We enable the led for the right side

while(turned==0) // We can't move on until the timer T1 is finished

{

sei();

}

gonies\_dexia = gonies\_dexia + 1; // The smart cleaner will make a left turn

}

}

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on // We enable the led for moving forward

flag = 0; // We set the flag again for the while loop

turned = 0; // We set the variable turned to 0 again for the while loop

// T2 = 2ms

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

TCA0.SINGLE.CTRLB = 0; //Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc ??? 207)

TCA0.SINGLE.CMP0 = mprostinos\_sensor; //When CMP0 reaches this value -> interrupt

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc ??? 224

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Interrupt Enable (=0x10)

while(turned==0)

{

// Initialize the ADC for Free-Running Conversion 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;

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

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

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

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

while((flag==0) && (turned==0))

{

sei();

}

if(flag==1)

{

flag = 0;

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN1\_bm; //PIN1 is off // We disable the led for moving forward

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on// We enable the led for the left side

while(turned==0)

{

sei();

}

gonies\_aristera = gonies\_aristera + 1; // The smart cleaner made a right turn

}

}

gonies = gonies\_aristera - gonies\_dexia; // We want the left turns minus the right turns to equal to 4.

turned = 0;

flag = 0;

}

cli(); //Disable Interrupts

}

ISR(ADC0\_WCOMP\_vect){

cli();

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

ADC0.CTRLA &= ~ADC\_ENABLE\_bm; //Disable ADC

flag = 1; // The flag changes value so that we can move on from the while loop

sei();

}

ISR(TCA0\_CMP0\_vect){

TCA0.SINGLE.CTRLA = 0; //Disable

int intflags = TCA0.SINGLE.INTFLAGS;

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

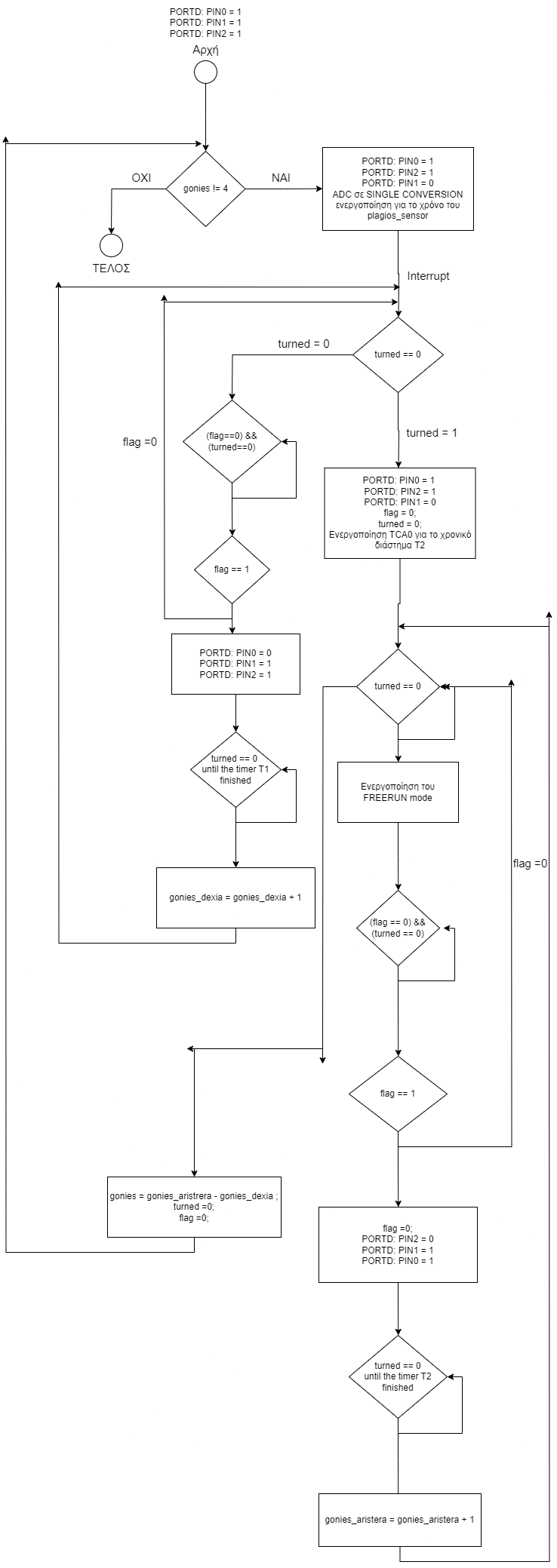
turned=1; //We change this variable to get out of the loop

}

**Επεξήγηση κώδικα δευτέρου ερωτήματος**

Αρχικά προσθέτουμε ένα ακόμη PIN0 το οποίο αφορά την ένδειξη ότι η έξυπνη συσκευή θα στρίψει δεξιά. Αρχικοποιούμε τον ADC και θέτουμε σαν κατώφλι τον αριθμό 5. Εάν το RES είναι μικρότερο από πέντε, τότε η έξυπνη συσκευή θα στρίψει αριστερά, ενώ εάν το RES είναι μεγαλύτερο από το 5, τότε θα στρίψει δεξιά. Ορίζουμε μία while η οποία θα εκτελείται για όσο ο συνολικός αριθμός των γωνιών είναι διάφορος από τέσσερα. Μέσα στην while έχουμε ορίσει έναν μετρητή ο οποίος θα μετράει 1ms και έπειτα θα μεταβαίνει από Single Conversion mode σε Free Running mode. Στην συνέχεια ορίζουμε μία εμφωλευμένη while στην οποία αρχικοποιούμε τον ADC σε Single Conversion mode. Έπειτα ορίζεται μία επόμενη while η οποία μας αποτρέπει να συνεχίσουμε την εκτέλεση του κώδικα μέχρι να δημιουργηθεί κάποιο interrupt είτε από τον ADC είτε από τον TCA0. Στην συνέχεια υπάρχει μία if η οποία εκτελείται εάν προκληθεί διακοπή από τον ADC, δηλαδή εάν το RES είναι μεγαλύτερο από το κατώφλι που ορίσαμε στο WINHT. Μέσα στην if ενεργοποιούμε το led του pin0 για να δηλώσουμε ότι η έξυπνη συσκευή θα στρίψει δεξιά και αυξάνουμε την μεταβλητή που κρατάει τις δεξιές στροφές κατά ένα (variable:gonies\_dexia). Εφόσον ο μετρητής TCA0 έχει ολοκληρώσει την μέτρηση του χρονικού διαστήματος T1=1ms τότε μπορούμε να συνεχίσουμε την εκτέλεση του κώδικα. Στην συνέχεια ξαναλλάζουμε τα pins ενεργοποιώντας το pin1 που είναι για την κίνηση μπροστά και απενεργοποιούμε τα pins 0 και 2 που αφορούν την δεξιά και αριστερή στροφή αντίστοιχα. Έπειτα ορίζουμε και πάλι τον μετρητή TCA0 για το χρονικό διάστημα T2=2ms όπου γίνεται η εναλλαγή από free running mode σε single conversion. Στην συνέχεια ορίζουμε και πάλι μία while στην οποία αρχικοποιούμε και πάλι τον ADC για free running mode. Όμοια λογική ακολουθείται μέσα στην συγκεκριμένη while όπου αυτή την φορά ελέγχεται εάν το RES είναι μικρότερο από το κατώφλι που έχουμε ορίσει στο WINLT. Στην συνέχεια εάν ισχύει η παραπάνω συνθήκη (RES<WINLT) τότε αυξάνεται ο μετρητής που έχουμε ορίσει για τις αριστερές γωνίες όπου στρίβει η έξυπνη συσκευή (variable:gonies\_aristera). Τέλος στην μεταβλητή gonies αφαιρούμε από τις αριστερές γωνίες τις δεξιές γωνίες και αν το αποτέλεσμα που θα προκύψει είναι ίσο με 4, τότε η έξυπνη συσκευή έχει ολοκληρώσει το περίγραμμα του τυχαίου δωματίου με αμβλείες γωνίες.

Διάγραμμα Ροής για το ερώτημα 2



1. Ο κώδικας που υλοποιήθηκε για την ανάποδη κίνηση της οικιακής συσκευής είναι ο εξής:

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdio.h>

#define all\_leds\_on 25

#define plagios\_sensor 15 // T1= 1msec, value=T1\*f(timer) = 2msec\*19,531KHz=19.531, for the simulation we used smaller values

#define mprostinos\_sensor 20 // T2=2msec, value=T2\*f(timer) = 2msec\*19,531KHz=39.062, for the simulation we used smaller values

int turned = 0; // We define a variable for the while loop

int gonies = 0; // Initialize the number of turns that the smart cleaner will make

int flag = 0; // We define a variable for the while loop

int gonies\_dexia = 0; // Initialize the number of right turns that the smart cleaner will make

int gonies\_aristera = 0; // Initialize the number of left turns that the smart cleaner will make

int button = 0; // This variable will be used for the opposite direction movement

int number\_of\_turns = 0; //This variable will hold all the number of turns the smart cleaner has made and it will be used for the opposite direction mode

int main(){

PORTD.DIR |= PIN0\_bm; //PIN0\_bm //PIN is output // Pin 0 Indicates that the smart cleaner will turn right

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

PORTD.DIR |= PIN1\_bm; //PIN1\_bm //PIN is output // Pin 1 indicates that the smart cleaner will move forward

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

PORTD.DIR |= PIN2\_bm; //PIN2\_bm //PIN is output // Pin 2 Indicates that the smart cleaner will turn left

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

// Initialize the ADC for single conversion mode

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

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

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

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

ADC0.WINLT |= 5; //Set threshold

ADC0.WINHT |= 5; //Set threshold

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

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; // PIN5 will be used for button

sei(); // Enable Interrupts

while( (gonies != 4) && (button ==0) ) // We check if the smart cleaner has completed the square and we ensure that it isn't running on opposite direction mode

{

if(button==0)

{

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on // We enable the led for moving forward

// T1 = 1ms

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

TCA0.SINGLE.CTRLB = 0; //Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc)

TCA0.SINGLE.CMP0 = plagios\_sensor; //When CMP0 reaches this value -> interrupt

//CLOCK\_FREQUENCY/1024

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Interrupt Enable (=0x10)

}

gonies = 0; // We initialize the number of turns to 0 to ensure that we don't have a problem with the opposite direction of the smart cleaner

while((turned==0) && (button==0) )

{

// Initialize the ADC for Single Conversion mode

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

ADC0.CTRLA = ADC0.CTRLA & 0b11111101; // We set the mode back to Single Conversion

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

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc;

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

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

ADC0.CTRLE = 0b00000010; // Interrupt when RES> WINHT

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

while((flag==0) && (turned==0) && (button==0)) // We can't move on from the while loop, until either the ADC will trigger an Interrupt, the Timer will trigger an Interrupt or the button for opposite direction is triggered

{

sei();

}

if (flag==1)

{

flag = 0; // We set the flag variable to zero again

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUT |= PIN1\_bm; //PIN1 is off // We disable the led for moving forward

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on// We enable the led for the right side

while(turned==0) // We can't move on until the timer T1 is finished

{

sei();

}

number\_of\_turns = number\_of\_turns +1; // We increase the number of turns in order to see how many turns the smart cleaner has made with the normal mode

gonies\_dexia = gonies\_dexia + 1; // The smart cleaner will make a right turn

}

}

if(button==0)

{

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on // We enable the led for moving forward

flag = 0; // We set the flag again for the while loop

turned = 0; // We set the variable turned to 0 again for the while loop

// T2 = 2ms

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

TCA0.SINGLE.CTRLB = 0; //Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc)

TCA0.SINGLE.CMP0 = mprostinos\_sensor; //When CMP0 reaches this value -> interrupt

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Interrupt Enable (=0x10)

while( (turned==0) && (button==0) )

{

// Initialize the ADC for Free-Running Conversion 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;

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

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

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

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

while((flag==0) && (turned==0) && (button==0))

{

sei();

}

if(flag==1)

{

flag = 0;

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN1\_bm; //PIN1 is off // We disable the led for moving forward

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on// We enable the led for the left side

while(turned==0)

{

sei();

}

number\_of\_turns = number\_of\_turns +1; // We increase the number of turns in order to see how many turns the smart cleaner has made with the normal mode

gonies\_aristera = gonies\_aristera + 1; // The smart cleaner made a right turn

}

}

}

gonies = gonies\_aristera - gonies\_dexia; // We want the left turns minus the right turns to equal to 4.

turned = 0;

flag = 0;

}

if(button==1)

{

//When we change to opposite direction mode, we turn all the leds on, to indicate the opposite mode

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

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

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

//Timer that counts for how many time all the leds are simultaneously activated for a period of time

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

TCA0.SINGLE.CTRLB = 0; //Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc)

TCA0.SINGLE.CMP0 = all\_leds\_on; //When CMP0 reaches this value -> interrupt

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Interrupt Enable (=0x10)

while(turned==0)

{

sei();

}

turned = 0;

while(number\_of\_turns > 0)

{

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on // We enable the led for moving forward

// T1 = 1ms

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

TCA0.SINGLE.CTRLB = 0; //Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc)

TCA0.SINGLE.CMP0 = plagios\_sensor; //When CMP0 reaches this value -> interrupt

//CLOCK\_FREQUENCY/1024

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Interrupt Enable (=0x10)

while(turned==0)

{

// Initialize the ADC for Single Conversion mode

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

ADC0.CTRLA = ADC0.CTRLA & 0b11111101; // We set the mode back to Single Conversion

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

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc;

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

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

ADC0.CTRLE = 0b00000010; // Interrupt when RES> WINHT

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

while((flag==0) && (turned==0)) // We can't move on from the while loop, until either the ADC will trigger an Interrupt or the Timer will trigger an Interrupt.

{

sei();

}

if (flag==1)

{

flag = 0; // We set the flag variable to zero again

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN1\_bm; //PIN1 is off // We disable the led for moving forward

PORTD.OUTCLR = PIN2\_bm; //PIN2 is on// We enable the led for the left side

while(turned==0) // We can't move on until the timer T1 is finished

{

sei();

}

number\_of\_turns = number\_of\_turns -1;// We decrease the number of turns because the smart cleaner will run in opposite direction mode

}

}

PORTD.OUT |= PIN0\_bm; //PIN0 is off // We disable the led for the right side

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on // We enable the led for moving forward

flag = 0; // We set the flag again for the while loop

turned = 0; // We set the variable turned to 0 again for the while loop

// T2 = 2ms

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

TCA0.SINGLE.CTRLB = 0; //Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc ??? 207)

TCA0.SINGLE.CMP0 = mprostinos\_sensor; //When CMP0 reaches this value -> interrupt

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc ??? 224

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Interrupt Enable (=0x10)

while(turned==0)

{

// Initialize the ADC for Free-Running Conversion 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;

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

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

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

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

while((flag==0) && (turned==0))

{

sei();

}

if(flag==1)

{

flag = 0;

PORTD.OUT |= PIN2\_bm; //PIN2 is off // We disable the led for the left side

PORTD.OUT |= PIN1\_bm; //PIN1 is off // We disable the led for moving forward

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on// We enable the led for the right side

while(turned==0)

{

sei();

}

number\_of\_turns = number\_of\_turns -1;

}

}

turned =0;

}

}

cli(); //Disable Interrupts

}

ISR(ADC0\_WCOMP\_vect){

cli();

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

ADC0.CTRLA &= ~ADC\_ENABLE\_bm; //Disable ADC

flag = 1; // The flag changes value so that we can move on from the while loop

sei();

}

ISR(TCA0\_CMP0\_vect){

TCA0.SINGLE.CTRLA = 0; //Disable

int intflags = TCA0.SINGLE.INTFLAGS;

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

turned=1; //We change this variable to get out of the loop

}

ISR(PORTF\_PORT\_vect){

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y; //clear the interrupt flag

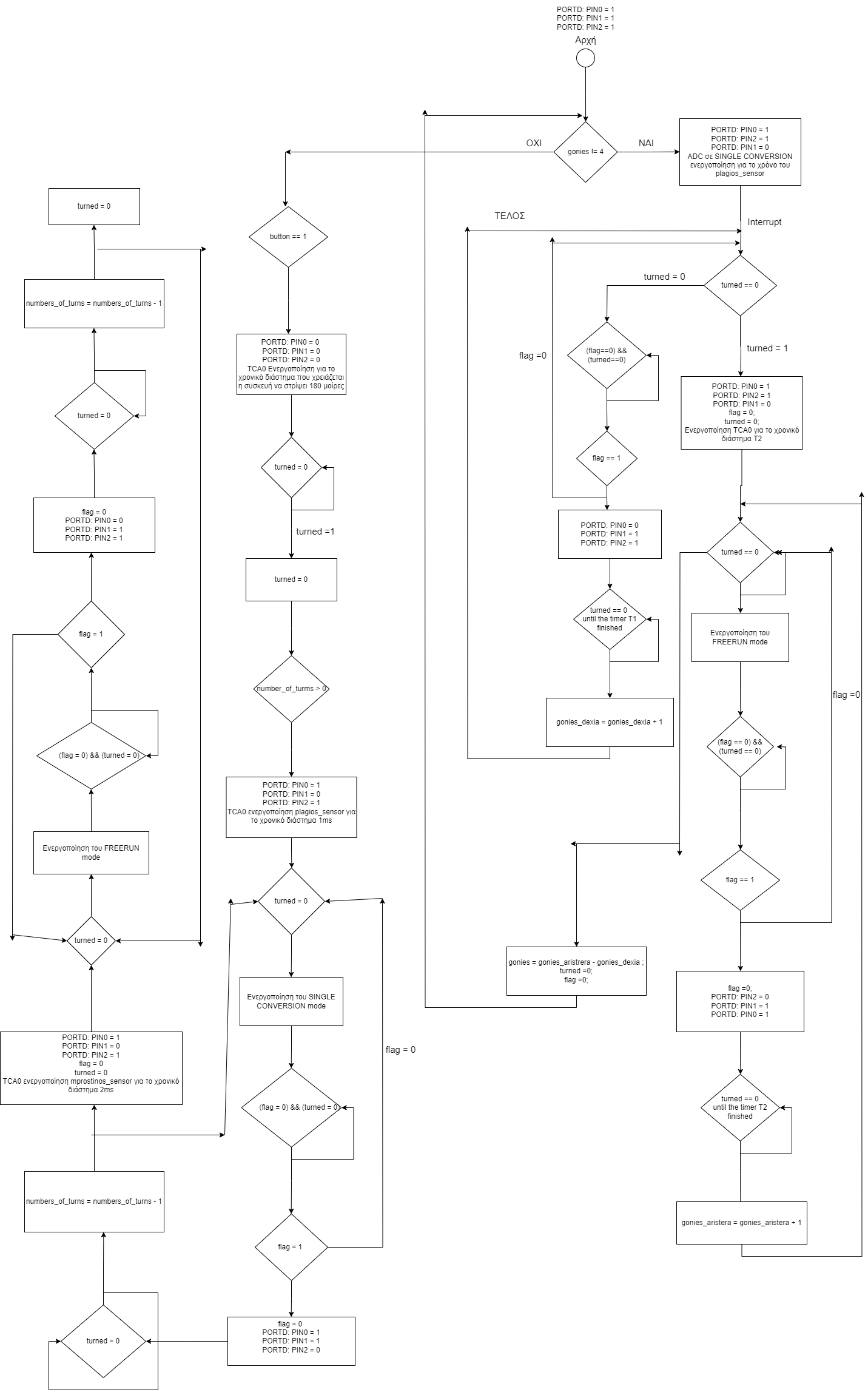
button=1; // The button turns on which means, that the smart cleaner will change to opposite movement

}

**Επεξήγηση κώδικα τρίτου ερωτήματος**

Για το συγκεκριμένο ερώτημα επεκτείναμε τον κώδικα του δεύτερου ερωτήματος. Πιο συγκεκριμένα προσθέσαμε μία μεταβλητή button η οποία αφορά το pin5 του PORTF, το οποίο όταν ενεργοποιείται σηματοδοτεί την έναρξη της ανάποδης λειτουργίας της έξυπνης συσκευής. Επιπλέον δηλώνουμε μια μεταβλητή number\_of\_turns στην οποία αποθηκεύουμε τον συνολικό αριθμό των γωνιών που έχει κάνει η έξυπνη συσκευή. Επιπρόσθετα τοποθετήθηκε στις συνθήκες ελέγχου που έχουμε χρησιμοποιήσει στον κώδικα μας (while, if), μία ακόμη συνθήκη για το κουμπί που θα πατηθεί από το pin5 (μεταβλητή button) για ανάποδη λειτουργία ενώ βρισκόμαστε στην εκτέλεση του κώδικα. Με αυτό τον τρόπο μπορούμε να φύγουμε από την εκτέλεση της while( ( gonies != 4 ) && (button==0) ) και να μεταβούμε στην if που υλοποιήθηκε και αφορά την ανάποδη λειτουργία της έξυπνης συσκευής. Επιπλέον στην κανονική λειτουργία προστέθηκε στο κομμάτι του κώδικα όπου η έξυπνη συσκευή στρίβει, η αύξηση των συνολικών γωνιών που έχει διανύσει (number\_of\_turns) η συσκευή ώστε να αξιοποιήσουμε αυτό το δεδομένο στην ανάποδη λειτουργία. Στην συνέχεια προστέθηκε κάτω από την συνθήκη while( ( gonies != 4 ) && (button==0) ), μία συνθήκη if η οποία εκτελείται μόνο όταν πατηθεί το pin5 και μεταβούμε στην ISR λόγω διακοπής. Μέσα στην ISR το button θα γίνει ίσο με ένα, επομένως μπορούμε να μεταβούμε μέσα στην συνθήκη if. Κατά την εκτέλεση της συνθήκης if αρχικά ανάβουμε όλα τα leds των pins 0,1,2 και στην συνέχεια ορίζουμε ένα timer TCA0 ο οποίος θα εκτελεστεί για ένα μικρό χρονικό διάστημα μέχρι η συσκευή μας να στρίψει πλήρως και να εκτελέσει ανάποδη πορεία. Εφόσον μεταβούμε στην ISR του timer, τότε μπορούμε να περάσουμε από την while που έχει οριστεί και αποτρέπει την εκτέλεση του κώδικα μέχρι να ολοκληρωθεί η μέτρηση από τον timer. Στη συνέχεια ορίζεται μία while η οποία θα εκτελείται για όσο ο συνολικός αριθμός των γωνιών που είχε διαπεράσει η έξυπνη συσκευή στην κανονική της λειτουργία είναι μεγαλύτερος του μηδενός. Αυτό συμβαίνει καθώς με την ανάποδη λειτουργία θέλουμε η οικιακή συσκευή να επιστρέψει στην αρχική της θέση. Μέσα στην while(number\_of\_turns > 0) αρχικά ενεργοποιούμε το pin1 για να δηλώσουμε ότι η συσκευή κινείται ευθεία. Επιπλέον ορίζεται και ένας timer ο οποίος είναι υπεύθυνος για να προκαλέσει αλλαγή του mode από single conversion σε free running mode έπειτα από T1=1ms. Επιπρόσθετα ορίζουμε κατάλληλα τον ADC σε single conversion mode και αυτή την φορά επειδή εκτελούμε ανάποδη λειτουργία, όταν RES > WINHT θα έχουμε ότι η έξυπνη συσκευή μας θα στρίψει αριστερά. Επιπλέον θα μειώσουμε τον συνολικό αριθμό των γωνιών κατά ένα καθώς η συσκευή επιστρέφει πίσω από όπου ήρθε για να πάει στην αρχική της θέση (number\_of\_turns = number\_of\_turns -1). Το led του pin2 ενεργοποιείται για να δηλώσουμε ότι η συσκευή πρόκειται να στρίψει αριστερά. Έπειτα από την εκτέλεση της στροφής προς τα αριστερά, η συσκευή κινείται και πάλι ευθεία οπότε ενεργοποιούμε το pin1 και το led ανάβει. Έπειτα ορίζουμε και πάλι ένα timer ο οποίος θα διαρκέσει για 2ms πριν από την εναλλαγή από free running mode σε single conversion. Στην συνέχεια ορίζεται και πάλι κατάλληλα ο ADC για free running mode. Όταν το RES γίνει μικρότερο από το WINLT, τότε εκτελείται η συνθήκη if και ενεργοποιείται το led του pin1 ώστε να στρίψουμε δεξιά. Και πάλι ο συνολικός αριθμός των γονιών μειώνεται κατά ένα. Τέλος η συνθήκη while(numbers\_ of\_turns > 0) θα εκτελεστεί έως ότου η συσκευή επιστρέψει στο σημείο από το οποίο ξεκίνησε.

Διάγραμμα Ροής για το ερώτημα 3



**Χρόνοι που χρησιμοποιήθηκαν για τους timers**

Οι τιμές που επιλέχθηκαν για τους χρονιστές είναι οι εξής:

Σένσορες για πλάγια: 15

Σένσορες για μπροστινή πορεία: 20

Χρόνος για αναστροφή 1800 : 25

Κανονικά αξιοποιείται ο τύπος value=T\*f(timer).

Οι τιμές που υπολογίστηκαν από τον τύπο αυτό είναι οι εξής:

Σένσορες για πλάγια: value=T1\*f(timer)= 2msec\*19.531KHz=19.531

Σένσορες μπροστινή πορεία: value=T2\*f(timer)= 1msec\*19.531KHz=39.062

Με f(timer):

, με =20MHz και Ν=1024.



Για την πιο γρήγορη προσομοίωση της άσκησης χρησιμοποιήθηκαν μικρότεροι χρόνοι και συγκεκριμένα οι χρόνοι 15,20, 25 που δόθηκαν παραπάνω.