**ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΗΓΜΕΝΟΙ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ**

**4η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ**

**ΜΕΛΗ:**

ΚΑΛΥΒΑ ΚΥΡΙΑΚΗ ΑΜ: 1089601

ΚΥΡΙΑΖΟΠΟΥΛΟΥ ΒΑΣΙΛΙΚΗ ΑΜ: 1090035

**Λεπτομέρειες από εκφώνηση της άσκησης για την υλοποίησή της.**

Για το έξυπνο θερμοκήπιο χρησιμοποιούμε:

* ADC
* Χρονιστή
* PWM
* 3 LEDs
* 3 Διακόπτες

Ο ADC δέχεται συνεχώς τιμές και τις ελέγχει σύμφωνα με τα δύο κατώφλια υγρασίας.

* Αν οι τιμές **μικρότερες** της κατώτερης τότε πρέπει να ποτιστούν τα φυτά άρα ανάβει το LED0.
* Αν οι τιμές **μεγαλύτερες** της ανώτερης τότε σημαίνει πως η υγρασία έχει υπερβεί τα όρια άρα ανάβει το LED1.

Χρησιμοποιούμε ADC για τον αισθητήρα υγρασίας.

Με την βοήθεια της ISR ελέγχουμε τις τιμές για να δούμε σε ποιο κατώφλι είμαστε έτσι ώστε να ενεργοποιήσουμε τον αερισμό ή το πότισμα ανάλογα την τιμή που έχουμε.

Στο δεύτερο ερώτημα για το masking χρησιμοποιήσαμε πύλη AND.

Θα δίνουμε μια τιμή στον καταχωρητή RES και ανάλογα με που βρίσκεται σχετικά με τα thresholds (10,20) θα λειτουργεί σε κανονικές συνθήκες ή θα ανοίγει ο εξαερισμός ή θα ανοίγει το πότισμα.

Για το switch 5 θα έχουμε (PORTF.INTFLAGS&&0b00100000)==0b01000000)

Για το switch 6 θα έχουμε (PORTF.INTFLAGS&&0b01000000)==0b01000000)

Για το τρίτο ερώτημα πρέπει να ελέγχουμε αν ο διακόπτης που πατήσαμε είναι ο σωστός.

**ΕΡΩΤΗΜΑ 1**

#include <avr/io.h>

#include <avr/interrupt.h>

int lower\_thres = 10;

int higher\_thres = 20;

int main(){

PORTD.DIR |= PIN0\_bm; //ygrasia hamili i endiksi lathous tautohrona me LED1, LED2

PORTD.DIR |= PIN1\_bm; //ygrasia ipsili i endiksi lathous tautohrona me LED0, LED2

PORTD.DIR |= PIN2\_bm; //systima aerismou i endiksi lathous tautohrona me LED0, LED1

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

//energopoiisi metatropea ADC (aisthitiras ygrasias)

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;//to psifio pou syndeetai me ton ADC

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode gia na mi stamataei pote na trehei o ADC

//Litourgia Sygkrisis (Window Comparator Mode)

ADC0.WINLT |= 10; //Set min threshold

ADC0.WINHT |= 20; //Set max threshold

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

ADC0.CTRLE |= 0b00000100; //Interrupt when RESULT < WINLT or RESULT >WINHT

sei();

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start Conversion, oi times tou convertion katagrafontai ston reg RES

while(1){

;

}

}

ISR(ADC0\_WCOMP\_vect){ //timi tou ADC panw apo max\_threashlod i katw apo min\_threashold

cli();

int intflags = ADC0.INTFLAGS; //clears INTFLAGS

ADC0.INTFLAGS = intflags; //clears INTFLAGS

if(ADC0.RES<lower\_thres){//an interrupt logo RES<WINLT

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on -> theloun potisma

}

if(ADC0.RES>higher\_thres){//an interrupt logo RES>WINHT

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on -> theloun aerismo

}

}

***Επεξήγηση κώδικα:***

Στην αρχή ορίζουμε τα LEDακια που θα χρησιμοποιήσουμε στην συνέχεια και αμέσως μετά τα απενεργοποιούμε αφού στην αρχή δε θέλουμε να μας ειδοποιούν για κάτι. Έπειτα ενεργοποιούμε τον ADC σε free-running mode αφού θέλουμε να παίρνουμε συνεχώς τιμές και στη συνέχεια βάζουμε ως τυχαίες τιμές που επιλέξαμε για το ανώτερο και το κατώτερο κατώφλι τις 20 και 10 αντίστοιχα. Με την εντολή ADC0.CTRLE |= 0b00000100 δηλώνουμε πως θέλουμε να μεταφερθούμε στην ISR του ADC0 είτε όταν έχει παραβιαστεί το RESULT < WINLT είτε το RESULT >WINHT. Κάνουμε enable τα interrupts και ξεκινάμε την λειτουργία του ADC0.

Σε αυτό το σημείο δίνουμε στην καταχωρητή RES του ADC0 κάποια τιμή. Το πρόγραμμα παραμένει στον βρόγχο while για πάντα εφόσον η τιμή που δώσαμε στον RES είναι μεταξυ του 10 και του 20 δηλαδή του κατώτερου και ανώτερου κατωφλίου αντίστοιχα – οι τιμές υγρασίας του θερμοκηπίου είναι φυσιολογικές. Αν η ανισότητα 10<RES<20 δεν ισχύει τότε σημαίνει ότι κάτι δεν πάει καλά με της τιμές υγρασίας. Δημιουργείται interrupt και καλείται η ISR(ADC0\_WCOMP\_vect) και στο σώμα της ελέγχουμε τι ακριβώς δεν πάει καλά στις τιμές. Δηλαδή αν δημιουργήθηκε interrupt λόγω RES<10(lower) ή λόγω RES>20(higher). Στο τέλος της απενεργοποιούμε τον ADC0 ώστε να πάψει να δέχεται τιμές μέχρι να ολοκληρωθεί ο κύκλος των λειτουργιών που έχουμε να εκτελέσουμε. Ανάλογα με σε ποια περίπτωση βρισκόμαστε ενεργοποιείται και το αντίστοιχο LEDαδκι ώστε να ειδοποιηθεί ο χρήστης και να δράσεις αναλόγως.

**Διάγραμμα Ροής:**

Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

**ΕΡΩΤΗΜΑ 2**

#include <avr/io.h>

#include <avr/interrupt.h>

int lower\_thres = 10;

int higher\_thres = 20;

int rising\_edges = 0;

int x=0; //metavliti pou mas krataei sti loop tis main

int main(){

PORTD.DIR |= PIN0\_bm; //ygrasia hamili i endiksi lathous tautohrona me LED1, LED2

PORTD.DIR |= PIN1\_bm; //ygrasia ipsili i endiksi lathous tautohrona me LED0, LED2

PORTD.DIR |= PIN2\_bm; //systima aerismou i endiksi lathous tautohrona me LED0, LED1

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //SW5-> systima potismatos

PORTF.PIN6CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //SW6-> systima aerismou

//energopoiisi metatropea ADC (aisthitiras ygrasias)

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;//to psifio pou syndeetai me ton ADC

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode gia na mi stamataei pote na trehei o ADC

//Litourgia Sygkrisis (Window Comparator Mode)

ADC0.WINLT |= lower\_thres; //Set min threshold

ADC0.WINHT |= higher\_thres; //Set max threshold

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

ADC0.CTRLE |= 0b00000100; //Interrupt when RESULT < WINLT or RESULT >WINHT

sei();

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start Conversion, oi times tou convertion katagrafontai ston reg RES

while(x==0){

;

}

}

ISR(ADC0\_WCOMP\_vect){ //timi tou ADC panw apo max\_threashlod i katw apo min\_threashold

int intflags = ADC0.INTFLAGS; //clears INTFLAGS

ADC0.INTFLAGS = intflags; //clears INTFLAGS

if(ADC0.RES<lower\_thres){//an interrupt logo RES<WINLT

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on -> theloun potisma

}

if(ADC0.RES>higher\_thres){//an interrupt logo RES>WINHT

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on -> theloun aerismo

}

ADC0.CTRLA=0; //Disable ADC0

}

ISR(PORTF\_PORT\_vect){ //ISR twn SWITCHES

cli();

//eleghoume poio switch prokalese tin isr

if((PORTF.INTFLAGS&0b00100000)==0b00100000){ //an patithike to switch 5 energopoioume to systima potismatos

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

int value = lower\_thres - ADC0.RES;

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

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

TCA0.SINGLE.CMP0 = value; //When reaches this value -> interrupt CLOCK FREQUENCY/1024

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

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

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

sei(); //begin accepting interrupt signals

}

if((PORTF.INTFLAGS&0b01000000)==0b01000000){ //an patithike to switch 6 enrgopoioume to systima eksaerismou

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1024\_gc; //prescaler=1024

TCA0.SINGLE.PER = 20; //Τ = 1 msec -> value = T\*ftimer = 0,001 (sec) \* 19.531,25 (Hz) = 19,53125 ≃20

TCA0.SINGLE.CMP0 = 10; //duty cycle 50%

TCA0.SINGLE.CTRLB=TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc; //select Single\_Slope\_PWM

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_OVF\_bm; //enable interrupt Overflow

TCA0.SINGLE.CTRLA |= TCA\_SINGLE\_ENABLE\_bm; //Enable TCA

sei();

}

int y = PORTF.INTFLAGS; //clears INTFLAGS

PORTF.INTFLAGS=y; //clears INTFLAGS

}

ISR(TCA0\_CMP0\_vect){ //ISR tou TIMER gia compare

TCA0.SINGLE.CTRLA = 0; //Disable TCA

PORTD.OUT |=PIN0\_bm; //pin0 off

int intflags = TCA0.SINGLE.INTFLAGS; //clears INTFLAGS

TCA0.SINGLE.INTFLAGS=intflags; //clears INTFLAGS

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

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

}

ISR(TCA0\_OVF\_vect){ //ISR tou TIMER gia overflow

rising\_edges++;

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

int intflags = TCA0.SINGLE.INTFLAGS; //clears INTFLAGS

TCA0.SINGLE.INTFLAGS = intflags; //clears INTFLAGS

if(rising\_edges==4){ //apenergopoiisi otan 4 rising edges

TCA0.SINGLE.CTRLA = 0; //Disable TCA

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

rising\_edges=0;

}

//energopoiisi metatropea ADC (aisthitiras ygrasias)

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

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

}

***Επεξήγηση κώδικα:***

Αρχικά αυτό που έχουμε προσθέσει συγκριτικά με το προηγούμενο ερώτημα είναι τον ορισμό των δύο διακοπτών που θα χρησιμοποιήσουμε, την ISR του PORTF, ενεργοποίηση TCA0 και την ISR αυτού και PWM καθώς και την ISR του για όταν εμφανίζεται rising edge.

Όπως και πριν κάνουμε τις απαραίτητες εντολές προκειμένου να αρχικοποιήσουμε και να ενεργοποιήσουμε τον ADC0. Στο ερώτημα 2 ήμαστε στη φάση που ο χρήστης βλέπει ποιο LED έχει ανάψει και επιλέγει ποιόν διακόπτη θα πατήσει βάσει αυτής της πληροφορίας. Όσο δεν έχει πατηθεί κάποιος διακόπτης το πρόγραμ παραμένει στον βρόγχο while. Με το πάτημα κάποιου διακόπτη καλείται η ISR(PORTF\_PORT\_vect). Στο σώμα της πρέπει να ελέγξουμε από ποιόν διακόπτη έχει προκληθεί το interrupt αφού εκτελούμε διαφορετικές λειτουργίες για τον καθένα.

Ο έλεγχος για το ποιος διακόπτης πατήθηκε επιτυγχάνεται χάρης τις:

if((PORTF.INTFLAGS&0b00100000)==0b00100000) & if((PORTF.INTFLAGS&0b01000000)==0b01000000)

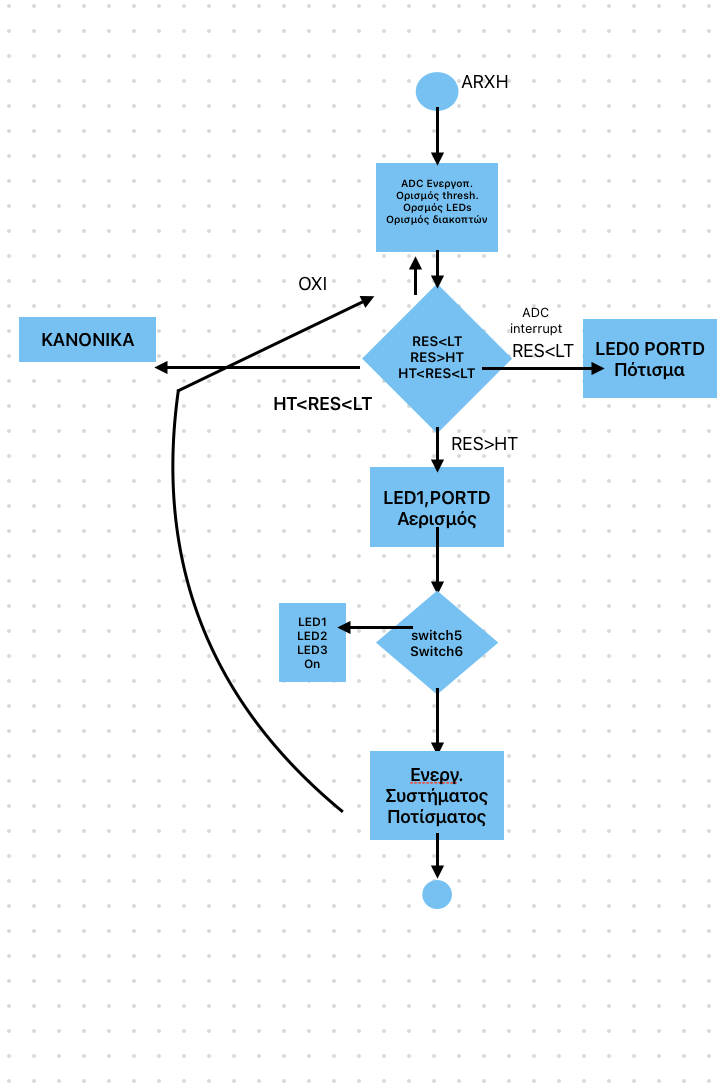
Ελέγχουμε το περιεχόμενο του καταχωρητή PORTF.INTFLAGS αν έχει 1 στη θέση 6->SW5 ή στην θέση 7->SW6. Ο έλεγχος αυτός γίνεται με bit masking ώστε να βεβαιωθούμε πως πατήθηκε όντως μόνο αυτό το ένα κουμπί και όχι και τα δύο ταυτόχρονα.

Αν πατήθηκε ο διακόπτης 5 μπαίνουμε στον πρώτο έλεγχο if. Ενεργοποιούμε το LED0, PORTD ώστε να δηλώσουμε ότι ενεργοποιείται το σύστημα ποτίσματος. Το σύστημα αυτό για να λειτουργήσει σωστά χρειάζεται έναν μετρητή που τρέχει για χρόνο ίσο με την τιμή int value = lower\_thres - ADC0.RES. Ενεργοποιούμε τον TCA0 και αρχίζουμε να δεχόμαστε διακοπές. Όταν περάσει το χρονικό διάστημα που ορίσαμε καλείται η ISR του TCA0(ISR(TCA0\_CMP0\_vect)). Στο σώμα της κάνουμε πάλι enable τον ADC0 ώστε στο επόμενο βήμα που το πρόγραμμα θα επιτρέψει στη while της main και θα αρχίσει πάλι να καταγράφει τιμές του θερμοκηπίου και παραμένει στον βρόγχο μέχρι να θέσουμε πάλι κάποια τιμή στον RES.

Αν πατήθηκε ο διακόπτης 6 μπαίνουμε στον δεύτερο έλεγχο if. Σε αυτό το σημείο έχουμε ενεργοποίηση του συστήματος αερισμού. Δίνουμε στον TCA0 τις κατάλληλες τιμές ώστε να λειτουργήσει ως PWM. Συγκεκριμένα Τ = 1 msec -> value = T\*ftimer = 0,001 (sec) \* 19.531,25 (Hz) = 19,53125 ≃20 και duty cycle 50%->10. Ενεργοποιούμε interrupt. Όταν φτάσουμε σε rising edge τότε έχουμε interrupt και καλείται η ISR(TCA0\_OVF\_vect). Στο σώμα της έχουμε αύξηση της μεταβλητής rising\_edges και γίνεται έλεγχος αν αυτή έχει περάσει την τιμή 4. Όταν φτάσει αυτή την τιμή κάνουμε disable τον TCA0 και ξανα ενεργοποιούμε τον ADC0.

***Παρατήρηση*:** Είναι σημαντικό να κάνουμε clear τον PORTF.INTFLAGS αφού θα έχουμε ελέγξει ποιός διακόπτης έχει πατηθεί. Αν το κάναμε πριν αυτών τότε το πρόγραμμα δε θα έμπαινε σε καμία if.

**Διάγραμμα Ροής:**



**ΕΡΩΤΗΜΑ 3**

#include <avr/io.h>

#include <avr/interrupt.h>

int lower\_thres = 10;

int higher\_thres = 20;

int rising\_edges = 0;

int x=0; //metavliti pou mas krataei sti loop tis main

int main(){

PORTD.DIR |= PIN0\_bm; //ygrasia hamili i endiksi lathous tautohrona me LED1, LED2

PORTD.DIR |= PIN1\_bm; //ygrasia ipsili i endiksi lathous tautohrona me LED0, LED2

PORTD.DIR |= PIN2\_bm; //systima aerismou i endiksi lathous tautohrona me LED0, LED1

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //SW5-> systima potismatos

PORTF.PIN6CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //SW6-> systima aerismou

//energopoiisi metatropea ADC (aisthitiras ygrasias)

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;//to psifio pou syndeetai me ton ADC

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode gia na mi stamataei pote na trehei o ADC

//Litourgia Sygkrisis (Window Comparator Mode)

ADC0.WINLT |= lower\_thres; //Set min threshold

ADC0.WINHT |= higher\_thres; //Set max threshold

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

ADC0.CTRLE |= 0b00000100; //Interrupt when RESULT < WINLT or RESULT >WINHT

sei();

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start Conversion, oi times tou convertion katagrafontai ston reg RES

while(x==0){

;

}

}

ISR(ADC0\_WCOMP\_vect){ //timi tou ADC panw apo max\_threashlod i katw apo min\_threashold

int intflags = ADC0.INTFLAGS; //clears INTFLAGS

ADC0.INTFLAGS = intflags; //clears INTFLAGS

if(ADC0.RES<lower\_thres){//an interrupt logo RES<WINLT

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on -> theloun potisma

}

if(ADC0.RES>higher\_thres){//an interrupt logo RES>WINHT

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on -> theloun aerismo

}

else{ //patithike latho koumpi

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

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

PORTD.OUTCLR = PIN3\_bm; //PIN3 is on

}

ADC0.CTRLA=0; //Disable ADC0

}

ISR(PORTF\_PORT\_vect){ //ISR twn SWITCHES

cli();

//eleghoume poio switch prokalese tin isr

if(((PORTF.INTFLAGS&0b00100000)==0b00100000) && (ADC0.RES<lower\_thres)){ //an patithike to switch 5 energopoioume to systima potismatos

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

int value = lower\_thres - ADC0.RES;

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

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

TCA0.SINGLE.CMP0 = value; //When reaches this value -> interrupt CLOCK FREQUENCY/1024

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

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

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

sei(); //begin accepting interrupt signals

}

if(((PORTF.INTFLAGS&0b01000000)==0b01000000) && (ADC0.RES>higher\_thres)){ //an patithike to switch 6 enrgopoioume to systima eksaerismou

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1024\_gc; //prescaler=1024

TCA0.SINGLE.PER = 20; //Τ = 1 msec -> value = T\*ftimer = 0,001 (sec) \* 19.531,25 (Hz) = 19,53125 ≃20

TCA0.SINGLE.CMP0 = 10; //duty cycle 50%

TCA0.SINGLE.CTRLB=TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc; //select Single\_Slope\_PWM

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_OVF\_bm; //enable interrupt Overflow

TCA0.SINGLE.CTRLA |= TCA\_SINGLE\_ENABLE\_bm; //Enable TCA

sei();

}

int y = PORTF.INTFLAGS; //clears INTFLAGS

PORTF.INTFLAGS=y; //clears INTFLAGS

}

ISR(TCA0\_CMP0\_vect){ //ISR tou TIMER gia compare

TCA0.SINGLE.CTRLA = 0; //Disable TCA

PORTD.OUT |=PIN0\_bm; //pin0 off

int intflags = TCA0.SINGLE.INTFLAGS; //clears INTFLAGS

TCA0.SINGLE.INTFLAGS=intflags; //clears INTFLAGS

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

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

}

ISR(TCA0\_OVF\_vect){ //ISR tou TIMER gia overflow

rising\_edges++;

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

int intflags = TCA0.SINGLE.INTFLAGS; //clears INTFLAGS

TCA0.SINGLE.INTFLAGS = intflags; //clears INTFLAGS

if(rising\_edges==4){ //apenergopoiisi otan 4 rising edges

TCA0.SINGLE.CTRLA = 0; //Disable TCA

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

rising\_edges=0;

}

//energopoiisi metatropea ADC (aisthitiras ygrasias)

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

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

}

***Επεξήγηση κώδικα:***

Αρχικά αυτό που έχουμε προσθέσει σε σχέση με τον προηγούμενο κώδικα είναι επιπλέον ελέγχους στο σώμα της ISR(PORTF\_PORT\_vect) ώστε να διαπιστώσουμε αν έχει πατηθεί το σωστό κουμπί με βάσει τις ανάγκες τους θερμοκηπίου.

Συγκεκριμένα ενώ πριν ελέγχαμε απλά πιο κουμπί πατήθηκε τώρα έχουμε:

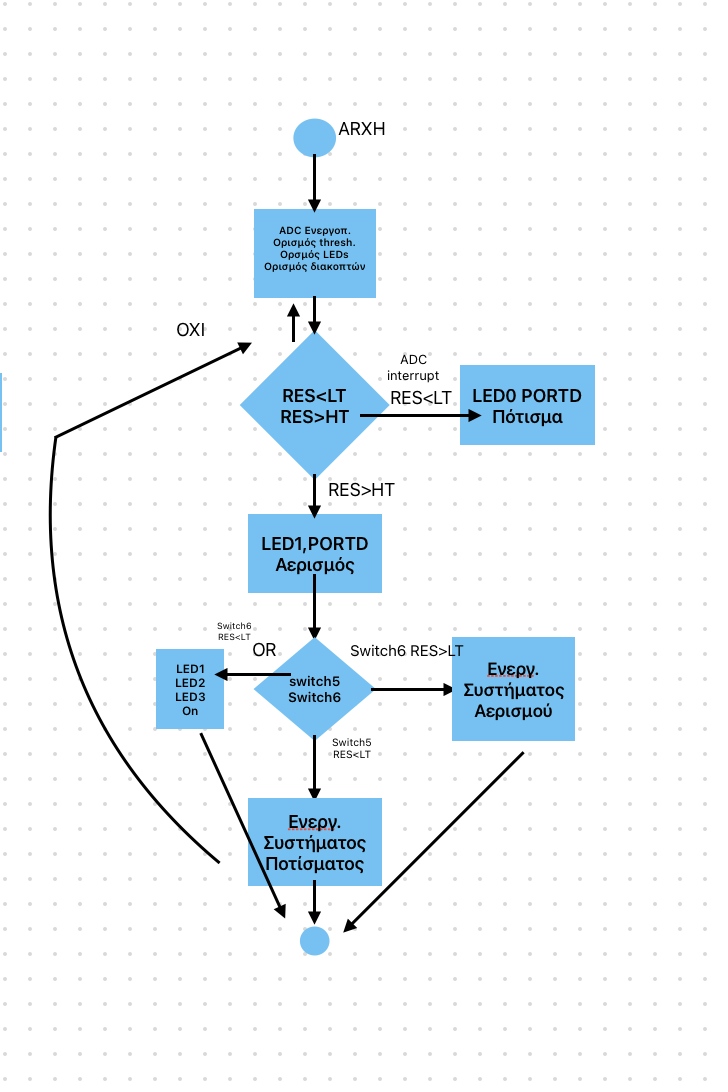
if(((PORTF.INTFLAGS&0b00100000)==0b01000000) && (ADC0.RES<lower\_thres)) και

if(((PORTF.INTFLAGS&0b01000000)==0b01000000) && (ADC0.RES>higher\_thres))

Το δεύρετο μέρος τους ελέγχει πως όταν πατάει το κουμπί 5 έχουμε όντως χαμηλή υγρασία και αντίστοιχα για το κουμπί 6 ότι πατιέται όταν έχουμε υψηλή υγρασία.

Επιπλέον υπάρχει και μια else που εκτελείται όταν ο χρήστης πατάει λάθος κουμπί. Σε αυτή την περίπτωση απλά ενεργοποιούμε και τα 3 LEDακια ώστε να τον ενημερώσουμε για το λάθος του.

**Διάγραμμα Ροής:**



**ΤΕΛΟΣ ΕΡΓΑΣΙΑΣ**