



Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Πανεπιστήμιο Πατρών
Πολυτεχνική Σχολή

Τομέας Υλικού και Αρχιτεκτονικής των Υπολογιστών

Διδάσκουσα: Μαριλένα Δούναβη

Ακαδημαϊκό Έτος: 2023 – 2024

Ημ/νία Παράδοσης: 16/04/2024

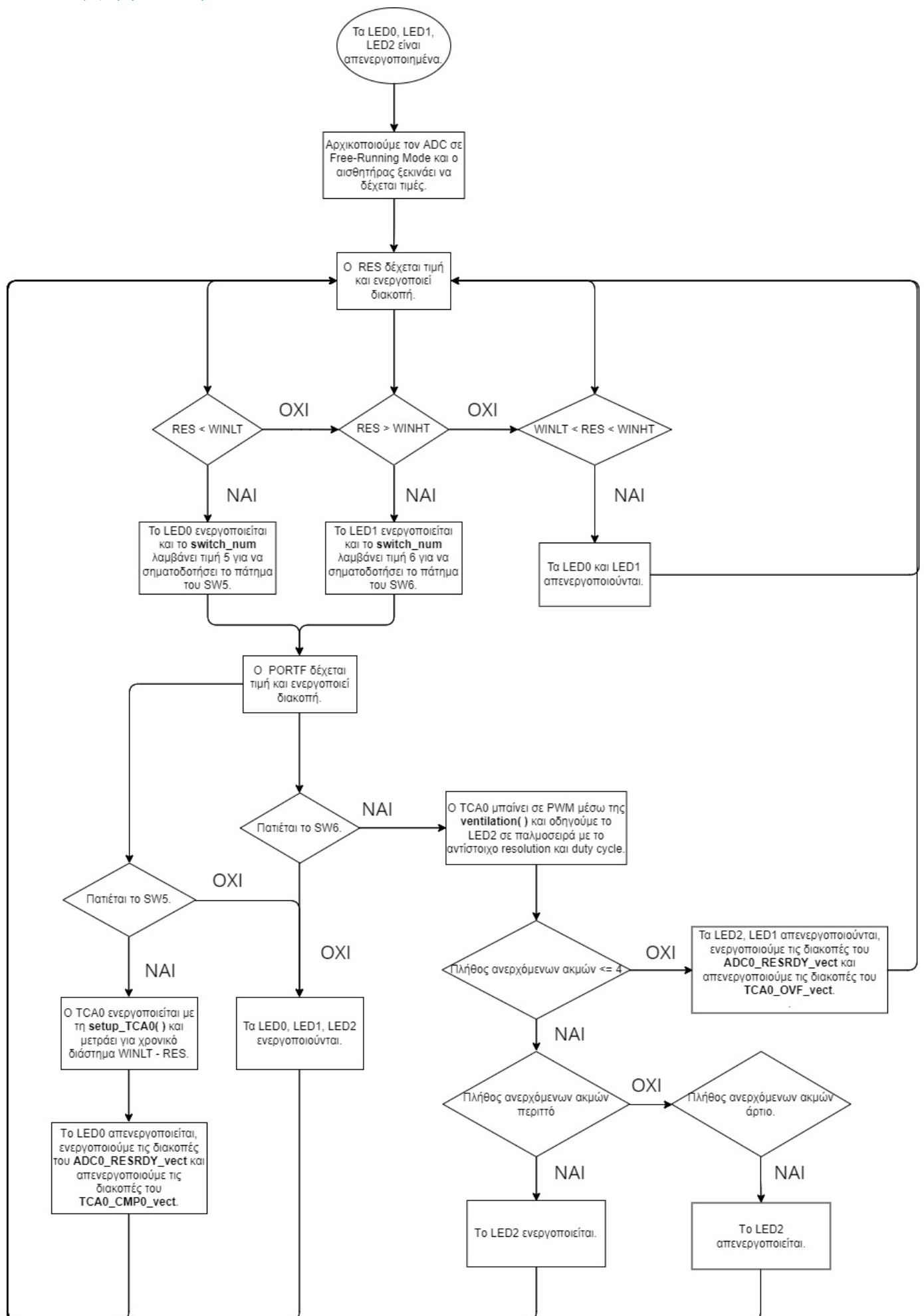
Εργαστήριο Προηγμένων Μικροϋπολογιστών 4^η Εργαστηριακή Άσκηση Λειτουργία Έξυπνου Θερμοκηπίου

Μάθημα Κορμού – CEID_NY463
Εαρινό Εξάμηνο 2024

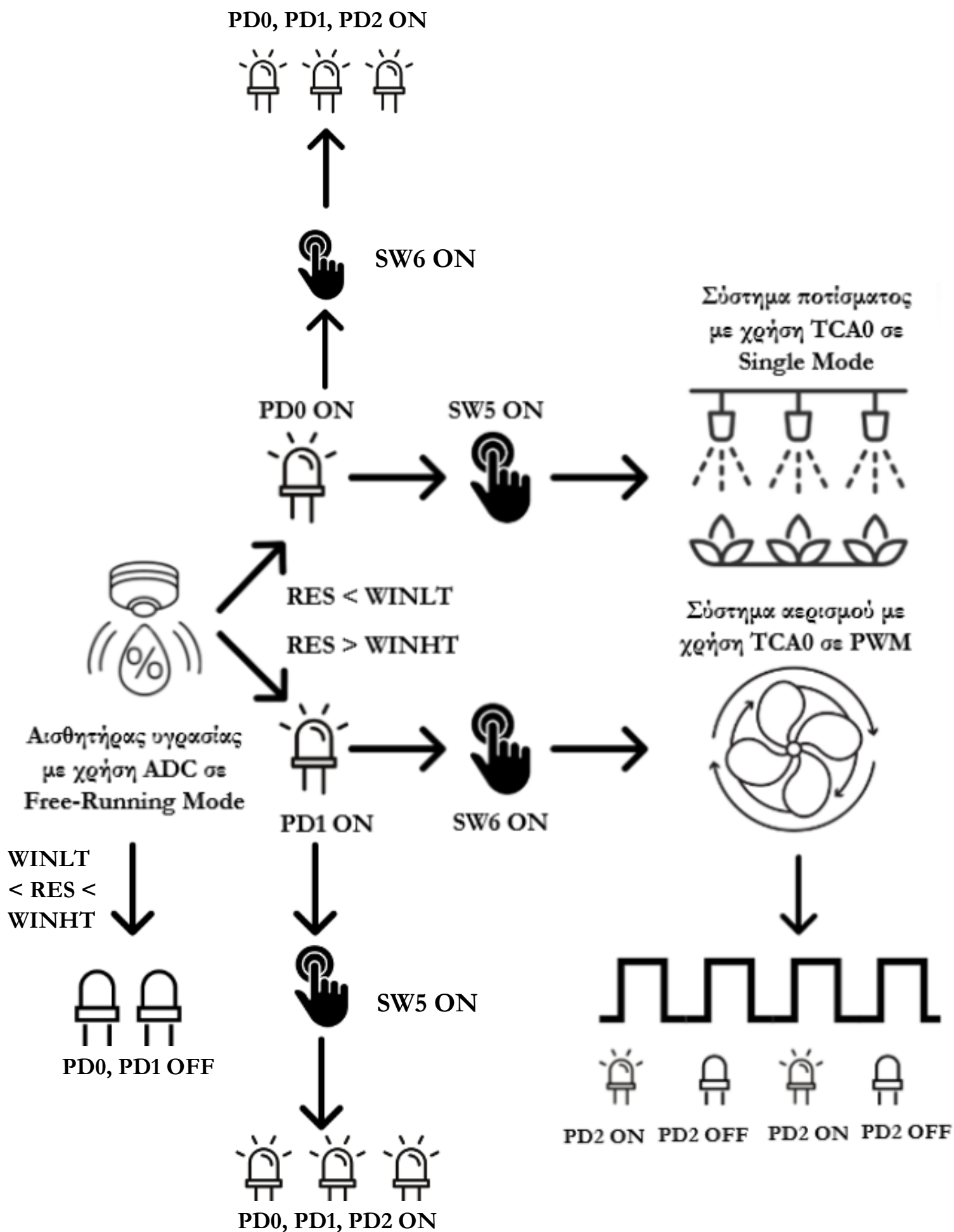
Στοιχεία Φοιτητών (ΤΜΗΜΑ Α2):

Ονοματεπώνυμο:	Μηλιτιάδης Μαντές	Χρυσauγή Πατέλη
A.M.:	1084661	1084513
E – mail:	up1084661@ac.upatras.gr	up1084513@ac.upatras.gr
Εξάμηνο:	Η'	Η'

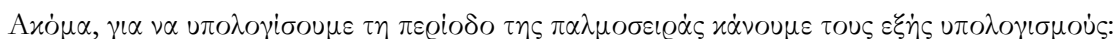
4.0 Διάγραμμα Ροής



4.1 Σχολιασμός κώδικα



Παρακάτω εξηγούμε πώς προκύπτει η κυματομορφή του παλμού που χρησιμοποιούμε στην εξομοίωση:



- Στην συνάρτηση **setup_TCA0(int t)** αρχικά θέτουμε τον χρονιστή σε κανονική λειτουργία και θέτουμε την τιμή του ίση με το μηδέν. Στην συνέχεια, ορίζουμε την τιμή στην οποία όταν φτάσει ο χρονιστής θα προκληθεί διακοπή, η τιμή αυτή υπολογίζεται ως το αποτέλεσμα της αφαίρεσης της τρέχουσας τιμής του **RES** από το κατώφλι που θα δίνεται ως όρισμα στη συνάρτηση για να μπορούμε να εισάγουμε το κατώφλι που επιθυμούμε κάθε φορά (**WILNT** ή **WINHT**). Επίσης, ενεργοποιούμε τον χρονιστή μέσω του καταχωρητή **CTRLA**. Τέλος, ενεργοποιούμε τις διακοπές μέσω του καταχωρητή **INTCTRL**.

Η συνάρτηση **ventilation()** χρησιμοποιείται για την αρχικοποίηση του χρονιστή **TCA0**. Αρχικά, το εσωτερικό ρολόι του μικροελεγκτή διαιρείται με παράγοντα 1024 πριν την είσοδο στον χρονιστή. Επιπλέον, ενεργοποιούμε την δημιουργία κυματομορφών. Ορίζουμε την μέγιστη τιμή **TOP** μέχρι την οποία θα μετρήσει ο παλμός και ορίζουμε και τον κύκλο λειτουργίας του παλμού. Τέλος, ενεργοποιούμε την διακοπή που προκαλεί ο **OVF** κατά την υπερχείλισης του χρονιστή και ξεκινάμε το ρολόι.

Στη συνάρτηση **main()** αρχικά ορίζουμε με την εντολή **PORT.DIR** τα bit που θα χρησιμοποιήσουμε ως έξοδο, δηλαδή τα **PIN1**, **PIN2** και **PIN0**. Στη συνέχεια, με την εντολή **PORT.OUT** απενεργοποιούμε τα LEDs στα **PIN1**, **PIN2** και **PIN0**, έτσι ώστε να έχουμε ως αρχική συνθήκη ότι όλα τα pins είναι απενεργοποιημένα και ενεργοποιούμε το pull-up resistor για το **PIN5** και **PIN6** του **PORTF** ώστε να ρυθμίζεται η συμπεριφορά του **PIN5**, **PIN6** για να προκαλεί διακοπή σε κάθε αλλαγή. Ακολουθώντας, αρχικοποιούμε τον **ADC** καλώντας την συνάρτηση **setup_ADC_FRM()**. Στην συνέχεια, το πρόγραμμα εισέρχεται στον βρόγχο **while**. Με την εντολή **sei()** ξεκινάμε να δεχόμαστε διακοπές.

Στην διακοπή **ISR(ADC0_RESRDY_vect)** μεταβαίνουμε κάθε φορά που ο καταχωρητής **RES** έχει τιμή και έχει ενεργοποιηθεί η διακοπή. Αρχικά, αν ο καταχωρητής **RES** έχει μικρότερη τιμή από το κατώφλι **WINLT** τότε ανάβουμε το **PIN0** για να σηματοδοτήσουμε την ανάγκη για πότισμα, θέτουμε στην μεταβλητή **switch_num** την τιμή 5 (πρέπει να πατηθεί ο διακόπτης 5) και απενεργοποιούμε τις διακοπές για τον **RESRDY** μέχρι να γίνουν οι απαραίτητες ενέργειες. Αν ο καταχωρητής **RES** έχει μεγαλύτερη τιμή από το κατώφλι **WINHT**, τότε ανάβουμε το **PIN1** για να σηματοδοτήσουμε την ανάγκη για αερισμό, θέτουμε στην μεταβλητή **switch_num** την τιμή 6 (πρέπει να πατηθεί ο διακόπτης 6) και απενεργοποιούμε τις διακοπές για τον **RESRDY** μέχρι να γίνουν οι απαραίτητες ενέργειες. Αν δεν ισχύει καμία από τις παραπάνω συνθήκες, τότε σβήνουμε τα **PIN0** και **PIN1**.

Στην διακοπή **ISR(PORTF_PORT_vect)** μεταβαίνουμε κάθε φορά που πατηθεί ο διακόπτης 5 ή 6. Στην περίπτωση που έχει πατηθεί ο διακόπτης 5 και η τιμή του **switch_num** είναι 5 τότε σημαίνει ότι θα ενεργοποιηθεί η λειτουργία ποτίσματος οπότε αρχικοποιούμε τον **TCA0** καλώντας την συνάρτηση **setup_TCA0(ADC0.WINLT)**. Στην περίπτωση που έχει πατηθεί ο διακόπτης 6 και η τιμή του **switch_num** είναι 6 τότε σημαίνει ότι θα ενεργοποιηθεί η λειτουργία αερισμού οπότε αρχικοποιούμε τον **TCA0** σε PWM καλώντας την συνάρτηση **ventilation()**. Ενώ στην περίπτωση που ο διακόπτης που έχει πατηθεί έχει διαφορετική τιμή από την τιμή του **switch_num** τότε σημαίνει ότι έχει ενεργοποιηθεί λανθασμένη λειτουργία οπότε ενεργοποιούμε όλα τα **PIN** (**PIN0**, **PIN1**, **PIN2**) και στη συνέχεια τα απενεργοποιούμε βηματικά. Επίσης, ενεργοποιούμε τις διακοπές για τον **RESRDY** και καθαρίζουμε τις σημαίες διακοπής του **PORTF**.

Στην διακοπή **ISR(TCA0_CMP0_vect)** μεταβαίνουμε κάθε φορά που πραγματοποιείται διακοπή υπερχείλισης από τον **TCA0**, δηλαδή αφότου έχει ενεργοποιηθεί η λειτουργία ποτίσματος. Οπότε απενεργοποιούμε το **PIN0**, ενεργοποιούμε τις διακοπές για τον **RESRDY** και απενεργοποιούμε την διακοπή για τον **TCA0**.

Η διακοπή **ISR(TCA0_OVF_vect)**, ενεργοποιείται κάθε φορά που έχουμε διακοπή από τον καταχωρητή **OVF**, δηλαδή κάθε φορά που έχουμε ανοδική ακμή στον παλμό του **TCA0**. Η συγκεκριμένη διακοπή ενεργοποιείται μετά την ενεργοποίηση της λειτουργίας αερισμού. Αρχικά, κάθε φορά που εισέρχεται το πρόγραμμα στην διακοπή αυξάνουμε το **edges_count** κατά 1 για να δηλώσουμε σε ποια ανοδική ακμή βρισκόμαστε. Αν η τιμή του **edges_count** είναι μικρότερη ή ίση με το 4 ελέγχουμε αν η τιμή του είναι περιττός αριθμός αν είναι ανάβουμε το **PIN2**. Αν είναι άρτιος τότε σβήνουμε το **PIN2**. Στην περίπτωση που η τιμή του **edges_count** είναι μεγαλύτερη του 4 απενεργοποιούμε το **PIN1** και **PIN2**, ενεργοποιούμε τις διακοπές για τον **RESRDY** και απενεργοποιούμε την διακοπή για τον **TCA0**. Τέλος, καθαρίζουμε τις σημαίες διακοπής του **TCA0**.

4.2 Κώδικας

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

int x = 0;
int edges_count = 0; // Μετρητής που αποθηκεύει το πλήθος των ανερχόμενων ακμών
int switch_num = 0; // Μεταβλητή που καθορίζει ποιο switch πρέπει να πατηθεί
int PER = 20; // Προσομοιώνουμε T1 = 1 ms
int CMP = 10 ; // Προσομοιώνουμε D1 = 50% * T1

void setup_ADC_FRM(void){
    ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; // 10-bit resolution
    ADC0.CTRLA |= ADC_ENABLE_bm; // Ενεργοποίηση του ADC
    ADC0.CTRLA |= ADC_FREERUN_bm; // Ενεργοποιούμε το Free-Running Mode του ADC
    ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; // Ορίζεται το bit με το οποίο θα συνδεθεί ο ADC
    ADC0.DBGCTRL |= ADC_DBGRUN_bm; // Ενεργοποίηση Debug Mode
    ADC0.WINLT |= 5; // Θέτουμε το κατώτερο κατώφλι
    ADC0.WINHT |= 10; // Θέτουμε το ανώτερο κατώφλι
    ADC0.INTCTRL |= ADC_RESRDY_bm; // Ενεργοποιούμε τις διακοπές για τον RESRDY
    ADC0.COMMAND |= ADC_STCONV_bm; // Αρχίζει η μετατροπή
}

void setup_TCA0(int t){
    TCA0.SINGLE.CNT = 0; // Θέτουμε τον χρονιστή
    TCA0.SINGLE.CTRLB = 0;
    TCA0.SINGLE.CMP0 = abs(t - ADC0.RES); // Ρυθμίζουμε τον χρονιστή TCA0 με όριο το αποτέλεσμα
της αφαίρεσης της τρέχουσας τιμής του RES από το κατώφλι
    TCA0.SINGLE.CTRLA = 0x7<<1; // Χρήση παράγοντα διαίρεσης 1024
    TCA0.SINGLE.CTRLA |= 1; // Ενεργοποιείται ο χρονιστής
    TCA0.SINGLE.INTCTRL |= TCA_SINGLE_CMP0_bm; // Ενεργοποίηση διακοπών
}

void ventilation(void){
    TCA0.SINGLE.CTRLA=TCA_SINGLE_CLKSEL_DIV1024_gc; // Χρήση παράγοντα διαίρεσης 1024
    TCA0.SINGLE.CTRLB |= TCA_SINGLE_WGMODE_SINGLESLOPE_gc; // Ενεργοποίηση δημιουργίας
κυματομορφών
    TCA0.SINGLE.PER = PER; // Ορίζουμε την μέγιστη τιμή TOP μέχρι την οποία θα μετρήσει ο παλμός
    TCA0.SINGLE.CMP0 = CMP; //Ορίζουμε τον κύκλο λειτουργίας του παλμού
    TCA0.SINGLE.CTRLA = TCA_SINGLE_ENABLE_bm; // Ενεργοποιούμε τον TCA0 και ξεκινάμε το ρολόι
    TCA0.SINGLE.INTCTRL |= TCA_SINGLE_OVF_bm; //Ενεργοποιούμε τη διακοπή που προκαλεί το OVF κατά
την υπερχείλιση του χρονιστή
}

int main(void)
{
    PORTD.DIR |= PIN2_bm | PIN1_bm | PIN0_bm; // Ρυθμίζουμε τα PIN0, PIN1 και PIN2 ως output
    PORTD.OUT |= PIN2_bm | PIN1_bm | PIN0_bm; // Αρχικά όλα τα pins είναι απενεργοποιημένα
    PORTF.DIRCLR = PIN5_bm | PIN6_bm; // Ρυθμίζουμε τα pins 5, 6 ως input

    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc; // Ενεργοποιούμε τον pull-up
    register για το PIN5
    PORTF.PIN6CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc; // Ενεργοποιούμε τον pull-up
    register για το PIN6

    setup_ADC_FRM(); // Αρχικοποιούμε τον ADC σε Free-Running Mode
    sei(); // Αρχίζουμε να δεχόμαστε σήματα διακοπής
    while (x==0)
    {
        ;
    }
    cli(); // Σταματάμε να δεχόμαστε σήματα διακοπής
}

// Διακοπή για τον έλεγχο του καταχωρητή RES σε σχέση με το κατώφλι
ISR(ADC0_RESRDY_vect){
    cli(); //Απενεργοποιούμε την δυνατότητα να ενεργοποιηθεί άλλη διακοπή
```

```

// Καθαρισμός σημαίας διακοπής
int intflags = ADC0.INTFLAGS;
ADC0.INTFLAGS = intflags;

if(ADC0.RES < ADC0.WINLT){
    PORTD.OUTCLR = PIN0_bm; // Ανάβουμε το PIN0 για να σηματοδοτήσουμε την ανάγκη για
πότισμα
    switch_num=5; // Θέτουμε στην μεταβλητή switch_num την τιμή 5
    ADC0.INTCTRL = 0b00000000; // Απενεργοποιούμε τις διακοπές για τον RESRDY
}
else if(ADC0.RES > ADC0.WINHT){
    PORTD.OUTCLR = PIN1_bm; // Ανάβουμε το PIN1 για να σηματοδοτήσουμε την ανάγκη για
αερισμό
    switch_num=6; // Θέτουμε στην μεταβλητή switch_num την τιμή 6
    ADC0.INTCTRL = 0b00000000; // Απενεργοποιούμε τις διακοπές για τον RESRDY
}
else{
    PORTD.OUT |= PIN0_bm | PIN1_bm; //Σβήνουμε το PIN0 και PIN1
}
}

// Διακοπή για το πάτημα των κουμπιών
ISR(PORTF_PORT_vect){
    cli(); // Απενεργοποιούμε την δυνατότητα να ενεργοποιηθεί άλλη διακοπή

    if(switch_num==5 && (PORTF.INTFLAGS & 0b00100000)== 0b00100000){ // Ελέγχουμε αν η τιμή του
switch_num ισούται με 5 και αν έχει πατηθεί το PIN5 του PORTF
        setup_TCA0(ADC0.WINLT); // Αρχικοποιούμε τον TCA0
    }
    else if(switch_num==6 && (PORTF.INTFLAGS & 0b01000000)== 0b01000000){ //Ελέγχουμε αν η τιμή
του switch_num ισούται με 6 και αν έχει πατηθεί το PIN6 του PORTF
        ventilation(); // Αρχικοποιούμε τον TCA0 σε PWM
    }
    else{
        PORTD.OUTCLR = PIN0_bm | PIN1_bm |PIN2_bm; // Ενεργοποιούμε τα PIN0, PIN1 και PIN2
        PORTD.OUT |= PIN0_bm | PIN1_bm |PIN2_bm; // Ενεργοποιούμε τα PIN0, PIN1 και PIN2
        ADC0.INTCTRL |= ADC_RESRDY_bm; // Ενεργοποιούμε τις διακοπές για τον RESRDY
    }
    int y = PORTF.INTFLAGS; // masking
    PORTF.INTFLAGS=y;
}

// Διακοπή υπερχείλισης για τον χρονιστή TCA0
ISR(TCA0_CMP0_vect){
    cli(); // Απενεργοποιούμε την δυνατότητα να ενεργοποιηθεί άλλη διακοπή

    TCA0.SINGLE.CTRLA = 0; // Καθαρισμός TCA0
    // Καθαρισμός σημαίας διακοπής
    int intflags = TCA0.SINGLE.INTFLAGS;
    TCA0.SINGLE.INTFLAGS=intflags;

    PORTD.OUT |= PIN0_bm; // Απενεργοποιούμε το PIN0
    ADC0.INTCTRL |= ADC_RESRDY_bm; // Ενεργοποιούμε τις διακοπές για τον RESRDY
    TCA0.SINGLE.CTRLA =0; // Απενεργοποιείται ο χρονιστής
    TCA0.SINGLE.INTCTRL &= ~TCA_SINGLE_CMP0_bm; // Απενεργοποιούμε τις διακοπές του CMP0
}

// Διακοπή για το χειρισμό ανερχόμενης ακμής από τη κυματομορφή του TCA0
ISR(TCA0_OVF_vect){
    cli(); //Απενεργοποιούμε την δυνατότητα να ενεργοποιηθεί άλλη διακοπή
    edges_count++; // Αυξάνουμε το πλήθος των ανερχόμενων ακμών κατά 1









    if(edges_count<=4){ // Ελέγχουμε αν το πλήθος των ανερχόμενων ακμών είναι μικρότερο ή ίσο του
4
        if(edges_count%2 == 1){ // Ελέγχουμε αν έχουμε περιττό πλήθος ανερχόμενων ακμών
            PORTD.OUTCLR = PIN2_bm; // Ενεργοποιούμε τον PIN2
        }
        else if(edges_count%2 == 0){ // Ελέγχουμε αν έχουμε άρτιο πλήθος ανερχόμενων ακμών
            PORTD.OUT |= PIN2_bm; // Απενεργοποιούμε το PIN2
        }
    }
}

```







```
    }  
}  
else{  
    PORTD.OUT |= PIN2_bm | PIN1_bm; // Απενεργοποιούμε το PIN2 και PIN1  
    ADC0.INTCTRL |= ADC_RESRDY_bm; // Ενεργοποιούμε τις διακοπές για τον RESRDY  
    TCA0.SINGLE.CTRLB &= ~TCA_SINGLE_WGMODE_gm; //Απενεργοποιείται ο χρονιστής  
    TCA0.SINGLE.INTCTRL &= ~TCA_SINGLE_OVF_bm; // Απενεργοποιούμε τις διακοπές του OVF  
}  
}
```


4.3 Παράδειγμα Χρήσης



1. Η τιμή του **RES** είναι **0x0001**, δηλαδή μικρότερη του **WINLT (0x0005)**, οπότε το LED0 ενεργοποιείται:

 RES	0x610	0x0001		ADC
 WINLT	0x612	0x0005		ADC
 WINHT	0x614	0x000A		ADC
 OUT	0x464	0x06		PORTD













2. Πατάμε το **SW5** του PORTF και ενεργοποιείται το σύστημα ποτίσματος για χρονικό διάστημα **WINLT – RES** (δηλαδή **0x0005 – 0x0001 = 0x0004**). Όταν τελειώσει η χρονομέτρηση το LED0 απενεργοποιείται:

 INTFLAGS	0x4A9	0x20		PORTF
 CMP0	0xA28	0x0004		TCA0
 OUT	0x464	0x07		PORTD





3. Η τιμή του **RES** είναι **0x0081**, δηλαδή μεγαλύτερη του **WINHT (0x000A)**, οπότε το LED1 ενεργοποιείται:

 RES	0x610	0x0081		ADC
 OUT	0x464	0x05		PORTD

4. Πατάμε το **SW6** του PORTF και ενεργοποιείται το σύστημα αερισμού με το PWM. Μετράμε 4 διαδοχικές ανερχόμενες ακμές του παλμού και όταν τελειώσει η μέτρηση το LED1 απενεργοποιείται:

 INTFLAGS	0x4A9	0x40		PORTF
 OUT	0x464	0x01		PORTD
 OUT	0x464	0x05		PORTD
 OUT	0x464	0x01		PORTD
 OUT	0x464	0x05		PORTD
 OUT	0x464	0x07		PORTD

5. Η τιμή του **RES** είναι ξανά **0x0001**, δηλαδή μικρότερη του **WINLT**, οπότε το LED0 ενεργοποιείται. Πατάμε λανθασμένα τώρα το **SW6** του PORTF και όλα τα LEDs ενεργοποιούνται:

 RES	0x610	0x0001		ADC
 WINLT	0x612	0x0005		ADC
 WINHT	0x614	0x000A		ADC
 OUT	0x464	0x06		PORTD
 INTFLAGS	0x4A9	0x40		PORTF
 OUT	0x464	0x00		PORTD

6. Απενεργοποιούμε χειροκίνητα όλα τα LEDs και η διαδικασία επαναλαμβάνεται.