

**Αναφορά:****1. Περιγραφή κώδικα**

Στην Τρίτη εργασία καλούμαστε να δημιουργήσουμε ένα έξυπνο θερμόμετρο κάνοντας χρήση ενός αισθητήρα απόστασης, ενός αισθητήρα θερμότητας και μιας οθόνης LCD.

Στο αρχείο main.c αρχικά προσδιορίζονται με τη χρήση του #define τα pins του nucleo που αντιστοιχίστηκαν στα pins των περιφερειακών (Trigger/Echo pins του αισθητήρα απόστασης, το pin του αισθητήρα θερμότητας, τα pins για τα leds) καθώς και οι τιμές των θερμοκρασιών που πάνω από τη μία τιμή θα ανάβει όπως ζητείται το ένα led, και κάτω από την άλλη θα ανάβει το άλλο led.

```
#define echopin PA_1
#define triggerpin PA_0
#define tempPin PC_0
#define ledHOT PA_6
#define ledCOLD PA_7
#define ledBOARD PA_5
#define HOTTEMP 26
#define COLDTEMP 22
```

Στη συνέχεια έχουμε τη δήλωση κάποιων global μεταβλητών και flags που θα βοηθήσουν στη συνέχεια του προγράμματος.

Για την απλοποίηση της main πολλές λειτουργίες υλοποιήθηκαν σε συναρτήσεις που καλούνται στη main και είναι οι εξής:

Πρώτα έχουμε τη συνάρτηση **void initTIM4()**, στην οποία γίνεται initialization/ρύθμιση του TIM4 timer ώστε να «μετράει» μέχρι το 50.000 (period = 50000) με prescaler = 1599. Το ρολόι που «ακούει» βρέθηκε ότι είναι τα 16Mhz, επομένως, με την τιμή που δώθηκε στον prescaler, ο timer πλέον δουλεύει με συχνότητα 10kHz και μετράει μέχρι το 50000 και όταν φτάσει αυτή την τιμή καλεί έναν interrupt handler που η λειτουργία του θα εξηγηθεί παρακάτω. Επομένως έχουμε: 10kHz, άρα μετράει μέχρι το 10.000 σε ένα δευτερόλεπτο, άρα φτάνει στο 50.000 σε 5 δευτερόλεπτα. (Θέλουμε ο αισθητήρας θερμότητας να παίρνει μια μέτρηση ανά 5 δευτερόλεπτα.)

Μετά έχουμε τη συνάρτηση **void initTIM3()**, στην οποία γίνεται initialization του TIM3 timer ώστε αυτός να μετράει μέχρι το 3000 με τον prescaler = 1599 (όπως και ο TIM4) άρα ο TIM3 αντίστοιχα καλεί τον interrupt handler του κάθε 300ms.

Ακολουθεί η συνάρτηση **void sendPulse()**, η οποία όταν καλείται παρέχει στο trigger pin του αισθητήρα απόστασης έναν παλμό των 10uS, για να τον θέσει σε λειτουργία, κάτι που βρέθηκε από το datasheet του αισθητήρα HC-SR04 που ζητήθηκε να χρησιμοποιήσουμε. (Να σημειωθεί ότι μεταξύ των κλήσεων της συνάρτησης αυτής χρειάζεται να παρέλθει χρόνος τουλάχιστον 60ms σύμφωνα με το datasheet.)

Στη συνέχεια έχουμε τις συναρτήσεις **void TempWrite()**, **uint8\_t tempRead()** και **float tempSensor()** που ουσιαστικά είναι οι drivers του συγκεκριμένου αισθητήρα θερμοκρασίας που χρησιμοποιήθηκε και η υλοποίησή τους έγινε με βάση την περιγραφή που έγινε από το datasheet αυτού του αισθητήρα. Επίσης, όσον αφορά στην εμφάνιση της μέτρησης που θα παίρνει ο αισθητήρας στις στιγμές που απαιτούνται, υλοποιήθηκε η συνάρτηση **void displayTemp(float temp)** στην οποία θα μπαίνει ως όρισμα η μέτρηση θερμοκρασίας και θα

εμφανίζεται στην οθόνη LCD ( ουσιαστικά κάνει clear την οθόνη και εκτυπώνει σε αυτή το μήνυμα TEMP: xx C, όπου xx η τιμή του ορίσματος που κλήθηκε η συνάρτηση, κάνοντας χρήση των συναρτήσεων του driver της LCD που δίνεται).

Η tempSensor() εκτός απο το να παίρνει τη μέτρηση της θερμοκρασίας από τον αισθητήρα και να την επιστρέφει (**return** temperature), αναλαμβάνει και να ελέγχει κάθε φορά αυτή την τιμή και στην περίπτωση που ξεπεραστεί η τιμή **HOTTEMP** (έγινε #define στην αρχή) να ανάβει (μόνο) το led που αντιστοιχεί στο **ledHOT** (κόκκινο led), αν ξεπεραστεί η HOTTEMP κατά 3 βαθμούς να ανάβει και το ledHOT και το **ledBOARD** (το ledBOARD ουσιαστικά αντιστοιχεί στο led του nucleo) και αν είναι μικρότερη της θερμοκρασίας **COLDTEMP**, να ανάβει (μόνο) το led **ledCOLD** (πράσινο led). (Το ledBOARD θα μπορούσε να αντικατασταθεί από έναν διακόπτη (ρελέ) ενός ανεμιστήρα, όπως ζητείται στην εκφώνηση της εργασίας.)

Μετά από τις συναρτήσεις που έχουν να κάνουν με τον αισθητήρα θερμότητας, ακολουθούν οι ορισμοί των handlers από τους timers που χρησιμοποιήθηκαν για την υλοποίηση του project. Αρχικά έχουμε τον **void TIM4\_IRQHandler()**, που θέτει, όταν καλείται, ένα flag (tempFlag) σε 1, ώστε να «δίνει δικαίωμα» στη main (η οποία ελέγχει το flag με polling) να πάρει μέτρηση απο τον αισθητήρα θερμότητας (ο TIM4 αναφέρθηκε προηγουμένως πως καλεί τον interrupt handler του ανά 5 δευτερόλεπτα, άρα βγάζοντας από την εξίσωση τον μικρό έξτρα χρόνο που προσθέτει η τεχνική του polling, καταφέρνουμε έτσι να παίρνουμε μια μέτρηση θερμοκρασίας ανά 5 δευτερόλεπτα). Ακολουθεί ο **void TIM3\_IRQHandler()**, που όπως και στην περίπτωση του TIM4, θέτει ένα flag (countDistanceFlag) σε 1, ώστε να επιτρέψει αντίστοιχα στη main να καλέσει τη συνάρτηση sendPulse() μια φορά κάθε 300ms (Το όριο κλήσης της όπως αναφέρθηκε είναι 60ms, αλλά με testing βρέθηκε ότι δουλεύει απροβλημάτιστα με delay intervals 300ms, και επιλέχθηκε η μεγαλύτερη τιμή ως πιο safe). Τέλος έχουμε την υλοποίηση του interrupt handler του SysTick timer (ο οποίος θα αρχικοποιηθεί στη συνέχεια να «χτυπάει» κάθε 1sec), και αναλαμβάνει να ενημερώσει το πρόγραμμα κατάλληλα όταν παρέλθουν 10sec, από κάθε φορά που θα ενεργοποιείται (αυξάνει μια global μεταβλητή κατά ένα κάθε φορά που καλείται ο handler και όταν θα πάρει την τιμή 10, σημαίνει οτι πέρασαν 10 δευτερόλεπτα). (Χρειαζόμαστε να μένει το μήνυμα της μέσης τιμής των θερμοκρασιών για 10sec στην οθόνη, σύμφωνα με την εκφώνηση, από τη στιγμή που θα υπολογιστεί αυτή, άρα ο handler θα θέτει το flag tenSecFlag σε 0, όταν παρέλθει αυτό το διάστημα των 10sec, το οποίο flag θα το βλέπει η main και θα πάψει να εμφανίζει τη μέση τιμή στην οθόνη)

Ύστερα έχουμε τη συνάρτηση **void printMeanValue()**, η οποία έχει αντίστοιχη λειτουργία με τη συνάρτηση displayTemp, και όταν καλείται εκτυπώνει το μήνυμα "Mean is: xx ", εμφανίζει δηλαδή τη μέση τιμή των μετρήσεων θερμοκρασίας στην οθόνη, όποτε χρειάζεται (ανά 24 μετρήσεις θερμοκρασίας / 2 λεπτά)

Τέλος έχουμε τη συνάρτηση **void initStuff()**, η οποία, «αναλαμβάνει» να καλέσει τις κατάλληλες συναρτήσεις (που παρουσιάστηκαν προηγουμένως) ώστε να γίνουν τα απαραίτητα initializations (των timers, των pin που χρησιμοποιήθηκαν, της lcd, να καλέσει την \_\_enable\_irq() κτλ).

Μετά τις συναρτήσεις που εξηγήθηκαν ακολουθεί η **main**. Στη **main** αρχικά καλείται η **initStuff()** για να γίνουν τα απαραίτητα Initializations (αλλά και λίγες βοηθητικές μεταβλητές) και μετά ακολουθεί μια **while(1)**. Όπως αναφέρθηκε και προηγουμένως, οι interrupt handlers που έχουν χρησιμοποιηθεί αναλαμβάνουν μόνο να «δώσουν άδεια» στη **main** (η οποία διαβάζει τα flags που οι handlers ενημερώνουν κάθε όποτε πρέπει) να εκτελέσει κομμάτια κώδικα για τη λειτουργία του θερμοστάτη. Αρχικά για τον υπολογισμό της απόστασης ελέγχεται το flag **countDistanceFlag**, και αν έχει την τιμή 1, εκτελείται η εξής διαδικασία:

- Πρώτα καλείται η συνάρτηση **sendPulse()**
- Ύστερα ανοίγει μια **while** που ουσιαστικά περιμένει μέχρι το **echo pin** του αισθητήρα απόστασης να γίνει 1.
- Όταν το **echo pin** γίνει 1, βγαίνει το πρόγραμμα από αυτή τη **while** και μπαίνει σε μία καινούργια η οποία αυξάνει μια μεταβλητή **i** όσο το **echo pin** είναι high (Όσο πιο μακριά είναι ένα αντικείμενο από τον αισθητήρα απόστασης τόσο περισσότερη ώρα θα μείνει το **echo pin** high, οπότε τόσο μεγαλύτερη τιμή θα προλάβει να αποκτήσει η μεταβλητή **i**)
- Διαιρείται το **i** με έναν συντελεστή (31) που βρέθηκε μετά απο testing, ώστε η τιμή του **i** να είναι η απόσταση του αντικειμένου απο τον αισθητήρα σε εκατοστά.
- Ακολουθεί ένας έλεγχος της τιμής του **i**, όπου αν κάτι πλησιάσει τον αισθητήρα απόστασης στα 7 εκατοστά (ή πιο κοντά) (**if(i<7)**), θα εμφανίζεται στην οθόνη η τελευταία μέτρηση θερμοκρασίας καθώς και η τελευταία μέση τιμή θερμοκρασιών.
- Αντίστοιχα αν **δεν** υπάρχει αντικείμενο σε απόσταση 7 εκατοστών ή μικρότερη, η οθόνη είτε δεν εμφανίζει τίποτα (**lcd\_clear**), είτε εμφανίζει το μήνυμα "HOT" (ή "COLD"), αν η τελευταία μέτρηση θερμοκρασίας είναι μεγαλύτερη από την **HOTTEMP** (ή μικρότερη από την **COLDTEMP**). (Το πιο μήνυμα θα εμφανίζεται καθορίζεται στη συνάρτηση **tempSensor()**, μέσα στην οποία αποθηκεύεται σε έναν **global** πίνακα χαρακτήρων το μήνυμα που πρέπει να εμφανιστεί ανάλογα τη θερμοκρασία)

Για τον υπολογισμό της θερμοκρασίας ακολουθεί η εξής διαδικασία στη **main**:

-Ελέγχεται το flag **tempFlag** αν έχει τεθεί σε 1 (**tempFlag=1** κάθε 5 δευτερόλεπτα μέσω του interrupt handler του TIM4)

-Αν **tempFlag=1**, ελέγχεται η τιμή της μεταβλητής **counts24**. Αν η τιμή της μεταβλητής αυτής είναι <24, παίρνουμε μια μέτρηση και την αποθηκεύουμε στη θέση **counts24** του πίνακα

```
while(!gpio_get(echopin)){
}
```

```
while(gpio_get(echopin)){
    i++;
}
```

```
if(temperature>=HOTTEMP)
{ //*****when temp is high (HOTTEMP defines high
    if(!tenSecFlag)lcd_clear();
    if(!tenSecFlag)lcd_set_cursor(0,0);
    gpio_set(ledHOT,1);
    if(temperature>(HOTTEMP+3))gpio_set(ledBOARD,1)
    if(!tenSecFlag){
        lcd_print("    HOT");
        sprintf(lastTempMessage,"    HOT");
    }
    //*****
}
if(temperature<=COLDTEMP)
{ //*****when temp is low (COLDTEMP defines low
    lcd_clear();
    lcd_set_cursor(0,0);
    gpio_set(ledCOLD,1); //if less than COLDTEMP tur
    if(!tenSecFlag){
        lcd_print("    COLD");
        sprintf(lastTempMessage,"    COLD");
    }
    //*****
}

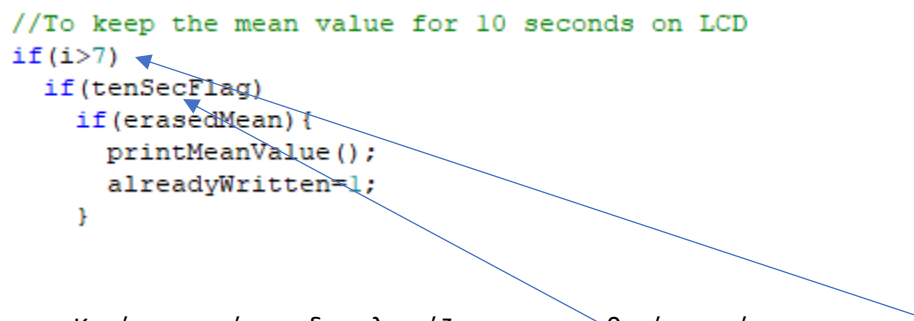
return temperature;
```

tempArray, ο οποίος διατηρεί τις μετρήσεις των θερμοκρασιών, ώστε μετά τις 24 μετρήσεις να βγεί ένας μέσος όρος αυτών. (Η μεταβλητή counts24, φτάνει μέχρι την τιμή 24, και αυξάνει κατά 1 σε κάθε μέτρηση θερμοκρασίας, ώστε να ενημερωθεί το πρόγραμμα ότι έλαβε 24 μετρήσεις και να «ξέρει» ότι τότε πρέπει να εξάγει έναν μέσο όρο θερμοκρασιών)

- Όταν η counts24==24, υπολογίζεται ο μέσος όρος των 24 μετρήσεων θερμοκρασιών και εμφανίζεται στην LCD (μέσω της συνάρτησης printMeanValue()). Παράλληλα τίθεται και το flag tenSecFlag σε 1, και γίνεται enable ο systick timer (**timer\_enable()**), ώστε μετά το πέρας των 10 δευτερολέπτων να τεθεί πάλι σε 0. (Θέλουμε ανά 24 μετρήσεις θερμοκρασίας, δηλαδή 2 λεπτά, αφού έχουμε μια μέτρηση ανά 5 δευτερόλεπτα, να εμφανίζεται στην οθόνη η μέση τιμή των 24 αυτών θερμοκρασιών για 10 δευτερόλεπτα)

Ακολουθεί μέσα στην while(1) ο κώδικας:

```
//To keep the mean value for 10 seconds on LCD
if(i>7)
    if(tenSecFlag)
        if(eraseMean){
            printMeanValue();
            alreadyWritten=1;
        }
```



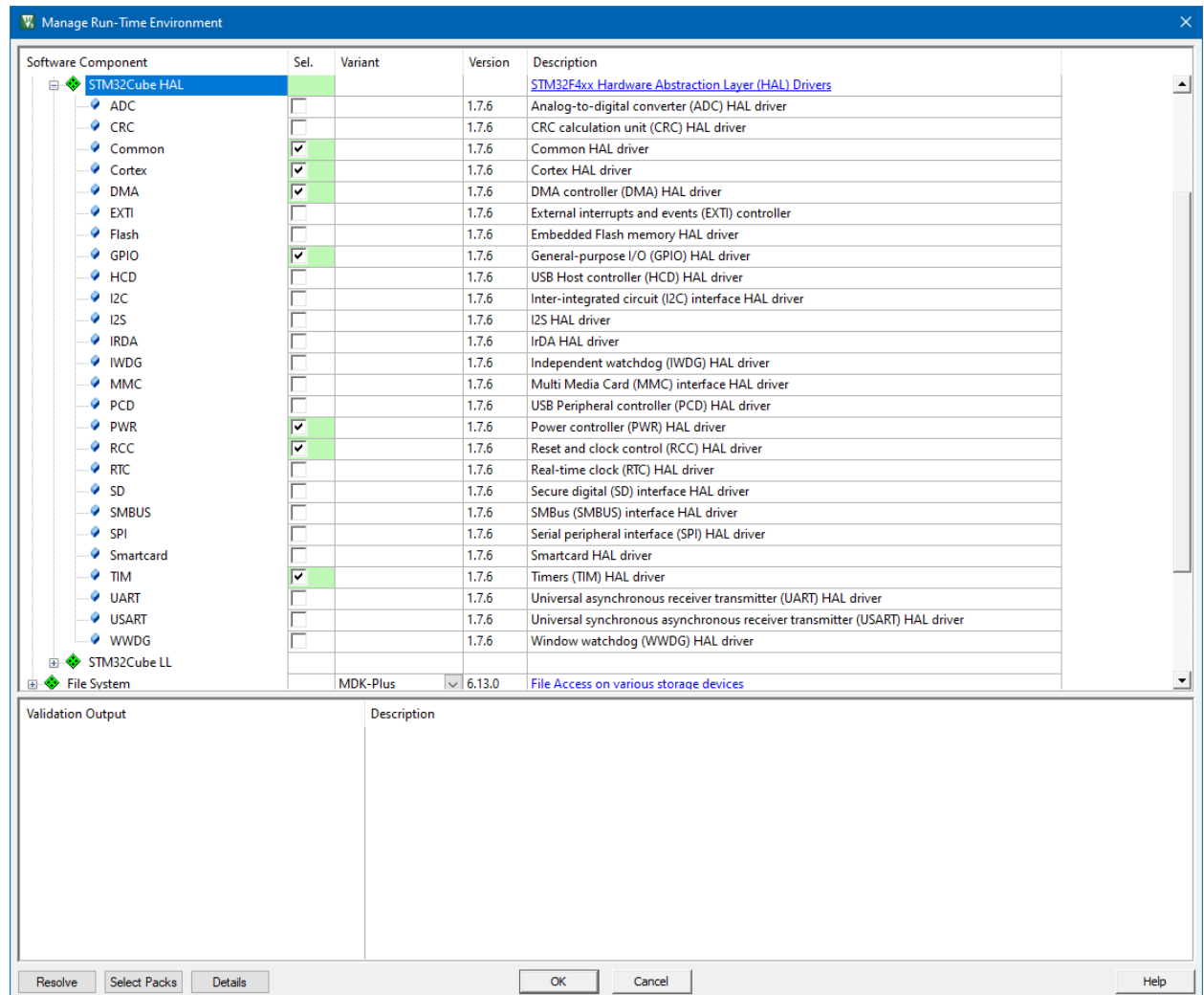
Κατά τον οποίο, αν δεν πλησιάζουμε τον αισθητήρα απόστασης σε απόσταση 7 εκατοστών ή μικρότερη **και** βρισκόμαστε στο παράθυρο 10 δευτερολέπτων που θέλουμε να εμφανίζεται στην οθόνη το μήνυμα της μέσης τιμής θερμοκρασιών, εκτυπώνεται στην οθόνη το μήνυμα της μέσης τιμής.

Με αυτό το κομμάτι ουσιαστικά πετυχαίνουμε το εξής:

Αν είμαστε στο 10 second window που πρέπει να εμφανίζεται η μέση τιμή θερμοκρασιών στην οθόνη, και πλησιάσουμε τον αισθητήρα απόστασης, αυτός θα μας δώσει στιγμιαία (όσο είμαστε κοντά στον αισθητήρα απόστασης) την ένδειξη της τελευταίας μέτρησης θερμοκρασίας καθώς και την τελευταία μέση τιμή θερμοκρασιών) και όταν θα απομακρυνθούμε θα επαναφέρει το μήνυμα της μέσης τιμής. Έτσι δίνεται προτεραιότητα στον χρήστη που ζητάει (πλησιάζοντας τον αισθητήρα) ουσιαστικά να δει την τελευταία μέτρηση θερμοκρασίας.

Ο λόγος που χρησιμοποιήθηκαν οι TIM3 και TIM4 (και δεν περιορίστηκα στη χρήση του SysTick που δίνεται στους drivers) είναι: πρώτον για ευκολότερη διαχείριση του κάθε χρόνου που χρειαζόμαστε ξεχωριστά (πχ στον TIM3 κατέληξα στα 300ms περίοδο μετά από δοκιμές που έγιναν πολύ πιο γρήγορα αλλάζοντας απλά το period του κατά το initialization) και δεύτερον για πιο απλούς interrupt handlers (Ο καθένας έχει μία δουλειά.. Αν χρησιμοποιούσα μόνο έναν timer, ο handler του θα είχε πιο σύνθετο κώδικα )

Επίσης χρειάστηκε να γίνουν οι ρυθμίσεις που φαίνονται στην ακόλουθη φωτογραφία για να ενεργοποιηθούν οι timers:



## 2. Testing

Για να διαπιστωθεί η σωστή λειτουργία του αισθητήρα απόστασης (της τεχνικής δηλαδή που χρησιμοποιήθηκε για τον υπολογισμό της απόστασης) έγιναν πολλές δοκιμές που επιβεβαιώθηκαν με τη χρήση χάρακα.

Ο σωστή λειτουργία του αισθητήρα θερμότητας επιβεβαιώθηκε συγκρινόμενος με ένα θερμόμετρο.

Ως προς την επιβεβαίωση της σωστής ρύθμισης των timer (χρονικά) , έγινε ένα απλό τεστ που κάθε φορά ο interrupt handler άναβε (αν ήταν σβηστό) ή έσβηνε (αν ήταν αναμένο) ένα led και χρονομετρούσα τις μεταβάσεις απο την μία κατάσταση στην άλλη.

Για την απόδειξη της λειτουργίας όλου του συστήματος δημιουργήθηκε ένα μικρό βιντεάκι που καλύπτει όλες τις δυνατές περιπτώσεις που ζητείται στην εκφώνηση να πληρούνται.