**ΕΡΓΑΣΤΗΡΙΑΚΗ ΆΣΚΗΣΗ 3**

**ΜΈΛΟΙ ΟMAΔΑΣ:**

**Ψυρρή Διονυσία: AM:1080424, έτος: 4ο**

**Σίδερης Στυλιανός:ΑΜ:1075217.έτος: 4ο**

Σε αυτή την άσκηση καλούμαστε να υλοποιήσουμε ένα σύστημα ασφαλείας. Κάναμε τα ζητούμενα αλλά προσθέσαμε και παραπάνω λειτουργείες λόγο ότι έχουνε ένα σύστημα ασφαλείας και πρέπει τα κενά ασφαλείας να είναι μηδαμινά .Επίσης προσθέσαμε κάποιες λειτουργίες να κάνουν τον χειρισμό του συστήματος ασφαλείας ευκολότερο.

Ας αναφέρουμε τι ακριβώς υλοποιήσαμε.

Σαν χρόνο εξόδου έχοντας το πληκτρολόγιο εγκατεστημένο σε σωστή θέση(κοντά στην κεντρική είσοδο) ορίζουμε τα 40 sec μια default τιμή όπως στα περισσότερα γνωστά Σ/Α(Caddx,Paradox κτλ).

Σαν χρόνο εισόδου ορίζουμε τα 10 sec σύμφωνα με τα περισσότερα Σ/Α.

Καθώς θέλουμε να φύγουμε από τον χώρο μας και να βάλουμε σε λειτουργεία το Σ/Α οπλίζουμε το σύστημα. Όμως ξαφνικά καταλαβαίνουμε ότι κάτι ξεχάσαμε πχ τα κλειδιά του αυτοκινήτου μας. Τι κάνουμε; Περιμένουνε να τελειώσει ο χρόνος εξόδου και μετά μπαίνουμε στον χώρο και αφοπλίζουμε το σύστημα με τον κωδικό μας; Όχι! Μπορούμε πριν τελειώσει ο χρόνος εξόδου να πατήσουμε τον κωδικό μας και να αφοπλίσουμε το σύστημα χωρίς να περιμένουμε .

Κατά την παραπάνω περίπτωση παρόλο που δεν έχει προλάβει το Σ/Α να μπει σε λειτουργία για να αφοπλίσουμε(πχ γιατί ξεχάσαμε τα κλειδιά του αυτοκινήτου και το καταλάβαμε πριν οπλίσει το Σ.Α ) δεν πρέπει να κανούμε λάθος τρείς φορές τον λάθος κωδικό γιατί θα χτυπήσει η σειρήνα.

Καθώς το Σ/Α είναι σε λειτουργεία και λόγο ότι σε πολλές εγκαταστάσεις δεν υπάρχει κάποιος ανιχνευτής να κοιτάει μέτωπο στο πληκτρολόγιο ίσως κάποιος καταφέρει να βρει κενό και χωρίς να έχει δώσει alarm κάποιος ανιχνευτής ,να αρχίζει να πατάει κωδικούς. Σε αυτή την περίπτωση σε τρείς λάθος κωδικούς θα έχουμε συναγερμό. Ανεξάρτητα με το αν έχει πιάσει κίνηση κάποιος ανιχνευτής.

**Υλοποίηση Άσκησης(καλύπτει και τα 3 ερωτήματα)**

Για την ενεργοποίηση του συστήματος βάζοντας σωστό κωδικό εκμεταλλευτήκαμε τις διακοπές του PORTF και κάποιες μεταβλητές όπως:

int digit\_code=1; : μας λέει σε ποιο ψηφίο είμαστε όταν πατάμε σωστά κουμπιά.

int tries = 3; : πόσες λανθασμένες προσπάθειες μας έχουν μείνει .

int suc\_code=0; :γίνεται 1 όταν έχουμε εισάγει σωστό κωδικό.

int falsecode=0; :γίνεται 1 όταν έχουμε εισάγει 3 φορές λάθος κωδικό. Έτσι παρακάμπτουμε μεγάλο μέρος κώδικα και πάμε κατευθείαν σε ενεργοποίηση σειρήνας.

Αν πατηθεί κάποιος από τους διακόπτες 5 και 6 τότε θα έχουμε interrupt και θα μπούμε στην ISR του PORTF.Εκεί με συνθήκες πού εξετάζουν ποιο από τα πλήκτρα 5 και 6 έχουν πατηθεί πάμε και στην ανάλογη ροή.

Αν έχουν πατηθεί και οι 2 διακόπτες τότε θα μπούμε σε μια συνθήκη που συμπεριφέρεται σαν να έχουμε πατήσει λάθος κωδικό : if(!(intflags ^ 0b01100000)).

Αν έχουν άσσο και τα δύο bit 5 και 6 του καταχωρητή intflags με το xor θα γίνουν και τα 2 μηδέν και συνολικά θα μας δώσει η συνθήκη 0 και αφού έχουμε αντιστροφή θα μπούμε στην if.

Για καταλάβουμε αν έχει πατηθεί ο διακόπτης 5 : if (intflags & PIN5\_bm)

Για καταλάβουμε αν έχει πατηθεί ο διακόπτης 6 : if (intflags & PIN6\_bm)

Για χρόνο εισόδου και εξόδου λειτουργούμε τον μετρητή TCA0 μέσο συνάρτησης και δίνουμε όρισμα το ENTRY\_TIME ή το ΕΧΙΤ\_ΤΙΜΕ.

Για τον αισθητήρα λειτουργούμε τον ADC σε FREE RUNNING MODE διότι θέλουμε να είναι σε συνεχή εποπτεία. Όταν ο αισθητήρας δεν έχει κάτι στο οπτικό του πεδίο δίνει μέγιστη αναλογική τιμή στον ΑΤmega4808.Αν βρεθεί κάτι μπροστά δίνει χαμηλότερη τιμή. Βάζουμε σαν threshold μικρή τιμή (30) για να αποφύγουμε ψευδείς συναγερμούς(πεταλούδες μύγες κτλ).

Για την παραγωγή PWM την σειρήνα μας λειτουργούμε πάλι τον TCΑ0 αφού κατά την διάρκεια που χτυπάει η σειρήνα δεν μετράμε κάποιον άλλο χρόνο.Εξάγουμε την PWM στο PORTC PIN0 στον καταχωρητή ΙΝ και την ένδειξη ότι η σειρήνα είναι σε λειτουργεία από το PIN0 του PORTD.Στην ISR ΤΟΥ TCA0 για overflow κάνουμε TONGLE την έξοδο αυτή για να ανάβει και να σβήνει σε κάθε περίοδο του παλμού.

Για να καταφέρουμε να πάρουμε τους χρόνους 40 sec και 10 sec με ρολόι συστήματος 20MHz και prescaler 1024 που είναι και το μέγιστο δεν μπορούμε να μετρήσουμε 40 sec διότι 20000000/1024 = 19531 θα γράψει ο μετρητής μας σε 1 sec, οπότε σε 40 sec θα γράψει 40\*19531 = 781250 που είναι μια τιμή που δεν χωράει σε 16 bit μετρητή. Οπότε θα κατεβάσω το ρολόι του συστήματος στα 32KΗz και prescaler 64 που αυτό σημαίνει σε 1 sec ο μετρητής μας θα γράψει 32000/64=500.Αφου θέλουμε 40 sec για την έξοδο 40\*500 = 20000 που είναι μια τιμή που χωράει στον 16 μπιτ μετρητή μας. Για tην παραγωγή |PWM δεν θα έχω prescaler και θεωρώ ότι η σειρήνα θα έχει παλμό με περίοδο 2msec και duty cycle 50%.

Αναλυτικά:

* Χρόνος εισόδου 10 sec με MAIN CLOCK 32KHz και prescaler 64 θα εχω:(32000/64)\*10 =5000.Οπότε θα βάλω CMP0=5000.
* Χρόνος εξόδου 40 sec με MAIN CLOCK 32KHz και prescaler 64 θα εχω:(32000/64)\*40 =20000. Οπότε θα βάλω CMP0=20000.
* PWM με περίοδο 2msec(500Hz) με MAIN CLOCK 32KHz και prescaler 1 θα έχω: 32000/500=64 -1 =63 στον καταχωρητή PER=63 και στον CMP0=63/2.

Όταν Αρχικοποιώ τον μετρητή για χρόνο εισόδου-εξόδου βάζω prescaler 1024 και όταν για PWM βάζω χωρίς prescaler γιατί διαφέρουν οι χρόνοι σε μέγεθος.

Εντολές για να κατέβει το ρολόι στα 32KHz:

//epitrepsi prosvasis se eidikoys kataxorites

CPU\_CCP = CCP\_IOREG\_gc;

//syxnothtia sta 32 KHh

CLKCTRL.MCLKCTRLA=CLKCTRL\_CLKSEL\_OSCULP32K\_gc;

Ο ταλαντωτής δεν πέφτει από τα 20MHz στα 32KHz ακαριαία αλλά απαιτητέ κάποιος χρόνος και για αυτό θα πρέπει να σταθεροποιηθεί στα 32ΚHz για να συνεχίσουμε. Αυτό το πετυχαίνουμε μέσο του καταχωρητή MCLKSTATUS.

//ELENXOS AN TO ROLOI EINAI STATHERO

while (!(CLKCTRL.MCLKSTATUS & 0x20))

{

;

}

Παρακάτω παρουσιάζετε ο κώδικας της άσκησης. Στις τιμές των χρόνων για τους ,μετρητές έχουμε βάλει τιμές για να μπορέσετε αν θέλετε να τρέξετε τον κώδικα. Οι τιμές που πρέπει να μπούνε κανονικά έχουν αναφερθεί παραπάνω.

**Κώδικας:**

#include <avr/io.h>

#include <avr/interrupt.h>

#define SIREN\_PERIOD 500

#define DUTY\_CYCLE\_SIREN SIREN\_PERIOD/2

#define ENTRY\_TIME 400

#define EXIT\_TIME 100

#define THRESS 30

volatile int ARMED=0;

int digit\_code=1;//posa psifia exo patisei sista

int tries = 3;

int suc\_code=0;

int test=1;

//metavlites poy voithane na meno i na feygo apo while

int timer=1;

int alarm=0;

int falsecode=0;

//sinarthsh gia timer gia xrono eidodoy exodu

void init\_timer\_entry\_exit(int time){

TCA0.SINGLE.CTRLB =0x00;

//xekiano metrisi apo 0

TCA0.SINGLE.CNT = 0x00;

//interrypt otan ftaso timi cmp

TCA0.SINGLE.CMP0= time;

//interrupt gia otan ftasv tini cmp

TCA0.SINGLE.INTCTRL=TCA\_SINGLE\_CMP0\_bm;

// kano dieresi sixnotitas/xekianao metrisi

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV64\_gc|TCA\_SINGLE\_ENABLE\_bm;

timer=1;

}

//sinarthsh gia arxikopoihsh PWM

void init\_timer\_PWM\_siren(void){

//ENERGOPIHSI GIA PWM

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1\_gc;

TCA0.SINGLE.PER = SIREN\_PERIOD; //select the resolution

TCA0.SINGLE.CMP0 = DUTY\_CYCLE\_SIREN; //select the duty cycle

TCA0.SINGLE.CTRLB |=TCA\_SINGLE\_CMP0EN\_bm |TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc; //enable interrupt Overflow

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_OVF\_bm;

//vgazo exodo PWM sto PORTC pin0

PORTMUX.TCAROUTEA |=PORTMUX\_TCA0\_PORTC\_gc;

//TCA0.SINGLE.INTCTRL |= TCA\_SINGLE\_CMP0\_bm;

//ksekinao apo 0

TCA0.SINGLE.CNT=0x00;

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

}

//sinartisi gia arxikopoihsh ADC

void ADC\_init(void){

//resolution 10 bit

ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc;

//analogiki eisodo apo pin 7

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

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm;

//energopioisi interrupt

ADC0.INTCTRL |= ADC\_WCMP\_bm;

//synexeia epopteyei

ADC0.CTRLA |= ADC\_FREERUN\_bm;

//katofli

ADC0.WINLT |= THRESS;

//otan res<thress

ADC0.CTRLE |= ADC\_WINCM0\_bm;

//ADC0.RES =0xff;

//xejinaei

ADC0.CTRLA |= ADC\_ENABLE\_bm;

ADC0.COMMAND |= ADC\_STCONV\_bm;

//alarm=0;

}

int main(){

//epitrepsi prosvasis se eidikoys kataxorites

CPU\_CCP = CCP\_IOREG\_gc;

//syxnothtia sta 32 KHh

CLKCTRL.MCLKCTRLA=CLKCTRL\_CLKSEL\_OSCULP32K\_gc;

//ELENXOS AN TO ROLOI EINAI STATHERO

while (!(CLKCTRL.MCLKSTATUS & 0x20))

{

;

}

//orizo ta led poy tha exo kai ola kleista gia arxi

//PORTA.DIR |=PIN0\_bm|PIN1\_bm|PIN2\_bm;

PORTC.DIR |=PIN0\_bm;

PORTD.DIR |=PIN0\_bm|PIN1\_bm|PIN2\_bm;

PORTD.OUT |=PIN0\_bm|PIN1\_bm|PIN2\_bm;

//energopoio ta dyo switch

PORTF.PIN5CTRL |=PORT\_PULLUPEN\_bm | PORT\_ISC\_RISING\_gc;

PORTF.PIN6CTRL |=PORT\_PULLUPEN\_bm | PORT\_ISC\_RISING\_gc;

sei();

while(1){

if (!ARMED)

{

while(!suc\_code){}

//pleon exoyme balei sosto kodiko kai pame na figoyme

//apo ton xoro.

//arxikopio metriti gia xrono exodoy

init\_timer\_entry\_exit(EXIT\_TIME);

//edo menoyme oso xrono xreiazete to systhma na oplisei

//afoy exoyme valei kodiko :XRONOS EXODOY

while(timer){}

}

if(ARMED){

//xekianei o aisthitiras kinisis meso toy ADC

ADC\_init();

//edo menoyme oso to systhma einai oplismeno alla

//den xtypaei synagermos,o esthitiras den pianei kati.

while(!alarm){}

//se ayth thn if den tha mpoyme an eno eimaste aplismenoi alla den exei

//disei alarm o aisthitiras exoyme pathsei 3 fores lathos ton kodiko

//den tha xreiastei na metrisoyme xrono eisodoy, tha xtipisei h seirina akariaia.

if(falsecode==0){

//xekinaei o xronos eisodoy

init\_timer\_entry\_exit(ENTRY\_TIME);

//edo menoyme oso metrane ta sec gia eisodo efoson epoiase o aisthitiras.

//an omos exoyme balei 3 fores lathos kodiko tha xekolisoyme apo tin

//while kai akariaia tha xtipisei alarm

while(timer){}

if(!ARMED){

alarm=0;

}

}

if(ARMED){

//exoyme synagermo den apenergopihthike kata ton xrono eisodoy

//energopoioyme thn sirina

init\_timer\_PWM\_siren();

}

while(alarm){}//tha mino edo oso xtyaei o sinagermos

//stamatao na parago PWM opote kai h sirina

TCA0.SINGLE.CTRLA &=0x0;

falsecode=0;

suc\_code=0;

ARMED=0;

//PREPEI NA SVISO LED SIRINA

PORTD.OUT |=0B00000001;

}

}

cli();

}

// Interrupt handler for TCA0

ISR(PORTF\_PORT\_vect) {

int intflags = PORTF.INTFLAGS;

PORTF.INTFLAGS = intflags;

if(!(intflags ^ 0b01100000))

{

if(ARMED | suc\_code) {

//meiononte oi prospathies mono an eimaste armed

tries--;

if(tries==0){

alarm=1;

falsecode=1;

timer=0;

ARMED=1;

}

}

digit\_code = 1;

return;

}

if (intflags & PIN5\_bm) {//otan patithei to pin 5

if (digit\_code == 1 || digit\_code == 3)

{

digit\_code++;

}

else {

if(ARMED | suc\_code) {

//meiononte oi prospathies mono an eimaste armed

tries--;

if(tries==0){

alarm=1;

falsecode=1;

timer=0;

ARMED=1;

}

//afoy egine lathos pame pali apo to 1

}

//se kathe periptosi me lathos kodiko pame pali sto proto psifio

digit\_code = 1;

}

}

if ( intflags & PIN6\_bm) {//otasn patithei to 6

if (digit\_code == 2) {

digit\_code++;

}

else if(digit\_code == 4) {

//edo tha mpo otan exo valei sosto kodiko

//opote oi prospathies tha ginoyn pali 3

//kai tha pame sto proto psifio kodikoy gia thn epomenh fora

if(suc\_code==0)

{suc\_code=1;}

else{ suc\_code=0;}

if(ARMED)ARMED=0;

digit\_code = 1;

tries = 3;

timer=0;

alarm=0;

}

else{

//edo tha mpoyme an to systima einai oplismeno

if(ARMED | suc\_code){

tries--;

if(tries==0){

alarm=1;

falsecode=1;

timer=0;

ARMED=1;

}

}

digit\_code=1;

}

}

}

ISR(ADC0\_WCOMP\_vect) {

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

TCA0.SINGLE.CNT = 0;

alarm=1;

//TCA0.SINGLE.CTRLA |= 1;

PORTD.OUT |= PIN0\_bm;

ADC0.CTRLA &=0xfe;

}

ISR(TCA0\_OVF\_vect)

{

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

//led on off

PORTD.OUTTGL =PIN0\_bm;

}

ISR(TCA0\_CMP0\_vect) {

//Disable TCA0

TCA0.SINGLE.CTRLA &= 0xFE;

//Clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

timer=0;

if (!ARMED )

ARMED=1; //oplizo to systhma

}

//------------------------------------------